

ABAP Part I

Lesson 01: R/3 Architecture

Lesson Objectives

- After completing this lesson, participants will be able to -
 - Know the meaning of ERP and SAP
 - Understand the R/3 system
 - Understand the Basics of SAP
 - Log on to SAP and do the Basic Navigations



Introduction to ERP

- What is ERP?
 - E – Enterprise R – Resource P – Planning
- Definition:
 - An integrated information system that serves all departments within an enterprise.
 - ERP is a way to integrate the data and processes of an Organization into one single system.
 - Software solution that addresses the enterprise needs taking the process
 - View of an organizational goals tightly integrating all functions of an enterprise



Copyright © Capgemini 2015. All Rights Reserved 3

Introduction to ERP

SAP	Oracle	PeopleSoft	JDEdwards
SD	Marketing, Sales	Supply chain	Order management
MM	Procurement	Supplier relationship	Inventory, procurement
PP	Manufacturing		Manufacturing mgmt
QM		Enterprise perform	Technical foundation
PM	Service	Enterprise service	
HR	Human Resources	Human capital mgmt	Workforce management
FI	Financials	Financial mgmt sol.	Financial management
CO			Time & Expense mgmt
AM	Asset Management		Enterprise asset mgmt
PS	Projects		Project management
WF	Order Management		
	Contracts		Subcontract, real estate



Copyright © Capgemini 2015. All Rights Reserved. 4

Introduction to ERP

What benefit	How
Reliable information access	Common DBMS, consistent and accurate data, improved reports.
Avoid data and operations redundancy	Modules access same data from the central database, avoids multiple data input and update operations.
Delivery and cycle time reduction	Minimizes retrieving and reporting delays.
Cost reduction	Time savings, improved control by enterprise-wide analysis of organizational decisions.
Easy adaptability	Changes in business processes easy to adapt and restructure.
Improved scalability	Structured and modular design with "add-ons."
Improved maintenance	Vendor-supported long-term contract as part of the system procurement.
Global outreach	Extended modules such as CRM and SCM.
E-Commerce, e-business	Internet commerce, collaborative culture.



Copyright © Capgemini 2015. All Rights Reserved 5

Introduction to SAP

- What is SAP?
 - S – Systems A – Applications P – Products in Data Processing
- SAP, was started in 1972 by five former IBM employees in Mannheim, Germany, states that it is the world's largest inter-enterprise software company and the world's fourth-largest independent software supplier, overall.
- SAP have a very high level of integration among its individual applications which guarantee consistency of data throughout the system and the company itself



Copyright © Capgemini 2015. All Rights Reserved 6

SAP Technical and Functional Modules

▪ Functional Modules

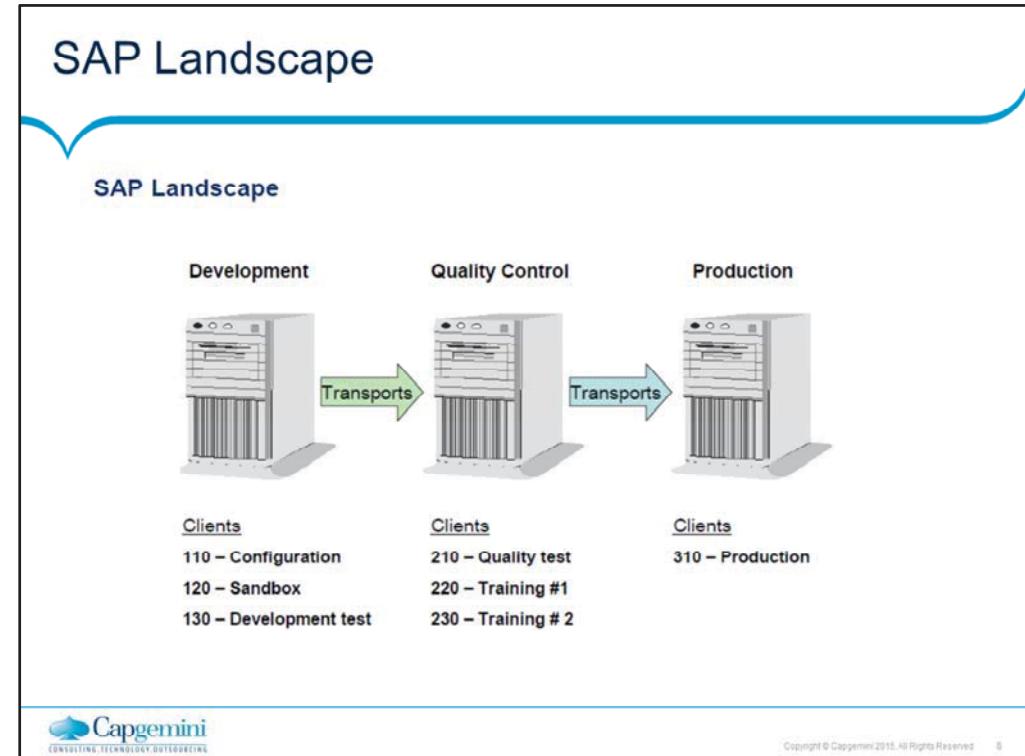
- FICO – Finance & Control
- PP – Production Planning
- MM – Material Management
- SD – Sales & Distribution
- HR – Human Resources

▪ Technical Modules

- ABAP – Advanced business applications programming
- XI – Exchange Infrastructure
- Basis –
- BIW – Business Information Warehousing



Copyright © Capgemini 2015. All Rights Reserved. 7



Introduction to NetWeaver

- NetWeaver is SAP's platform for composition and integration of loosely coupled applications following Service Oriented Architecture. The Application Platform is the runtime environment for SAP NetWeaver.
- Supports two languages (ABAP and Java) at the same time
- ABAP – Advanced Business Application Programming



Copyright © Capgemini 2015. All Rights Reserved 9

Why R/3

- The main purpose of an R/3 system is to provide a suite of tightly integrated, large-scale business applications. The standard set of applications delivered with each R/3 system are the following:
 - PP (Production Planning)
 - MM (Materials Management)
 - SD (Sales and Distribution)
 - FI (Financial Accounting)
 - CO (Controlling)
 - AM (Fixed Assets Management)
 - PS (Project System)
 - WF (Workflow)
 - IS (Industry Solutions)
 - HR (Human Resources)
 - PM (Plant Maintenance)
 - QM (Quality Management)



Copyright © Capgemini 2015. All Rights Reserved 10

The above applications are called the functional areas, or application areas, or at times the functional modules of R/3. All of these terms are synonymous with each other.

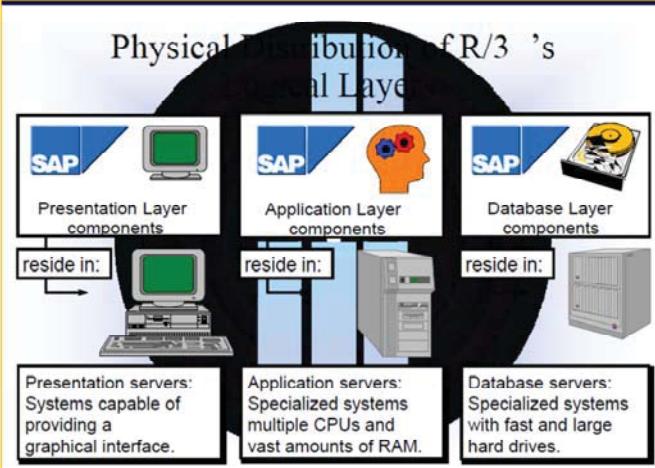
Defining R/3

- R/3 means Real-time 3-tier Architecture
- R/3 software supports all of a company's business transactions and links them together using real-time integration
- Real-time integration means that each change or update in one application causes the automatic change or update of the data in the other applications involved.
- R/3 also represents 3-tiered Client-Server Architecture.
- *The three Logical Layers of this R/3 Architecture are...*
 - The Presentation Layer: Collects user input and creates process request.
 - The Application Layer: Uses the Application logic of Program to collect and process the process request.
 - The Database Layer: Stores and Retrieves all Data.

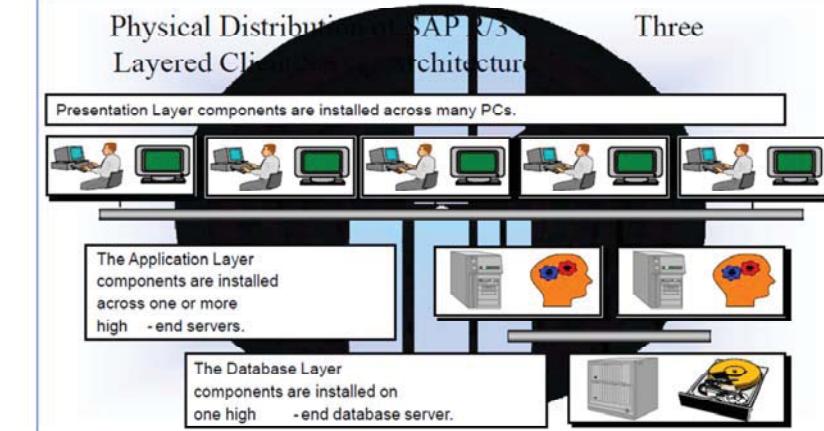


Copyright © Capgemini 2015. All Rights Reserved 11

SAP R/3 Architecture



SAP R/3 Architecture

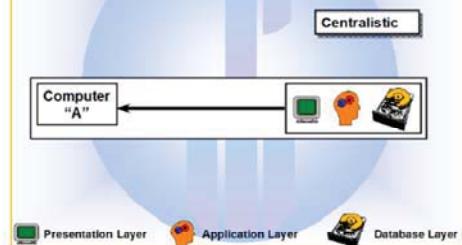


SAP R/3 Architecture

- Centralistic:

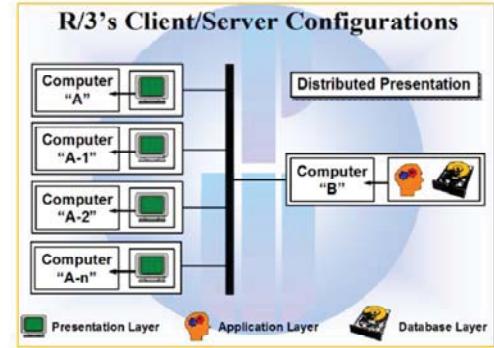
- All SAP R/3 layers reside on the same physical computer. One important distinguishing characteristic between the various R/3 client/server configurations is "Scalability".
- If a configuration is scalable, adding additional computers to the system will increase the overall performance of the system.
- Centralistic client/server configurations are not scalable at all.
- Consequently, this configuration is never used in a production environment.
- SAP has actually installed an entire SAP R/3 system on a notebook computer for use by SAP's sales representatives

R/3's Client/Server Configurations



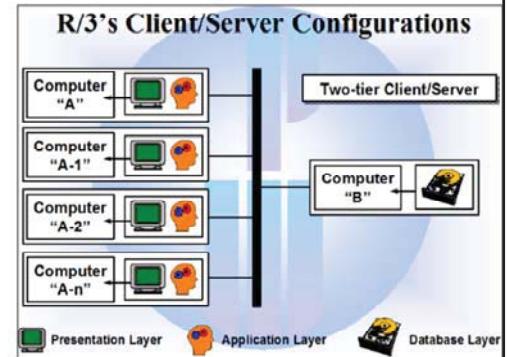
SAP R/3 Architecture

- Distributed presentation:
 - The presentation layer is “rolled out” to desktop PCs.
 - The application layer and the database layer are installed on the same computer.
 - In terms of increased performance, the Distributed presentation configuration is no more scalable than the centralistic configuration. This configuration is very “mainframe”-“ish”



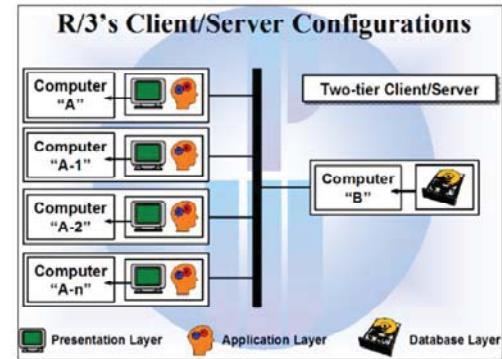
SAP R/3 Architecture

- Two-tier client/server :
 - The presentation and application layers are installed on the same computer. The database layer is installed on a separate computer.
 - Two-tier client/server configurations offer increased scalability.
 - However, two-tier client/server configurations create other problems.
 - The communication across the network between the front-end (presentation and application layers) and the backend (database layer) becomes a bottle-neck very quickly



SAP R/3 Architecture

- Three-tier client/server:
 - Presentation, application, and database layers run on separate computers.
 - Currently, three-tier client/server offers the best solution for most businesses.
 - It is highly scalable, and offers better distribution of process requests received from the users.
 - The computers in the application layer are often capable of satisfying the users process requests without accessing the database, which in turn boosts performance.



Application Server Architecture

- The components of an application server are shown in the figure below. It consists of a dispatcher and multiple work processes.

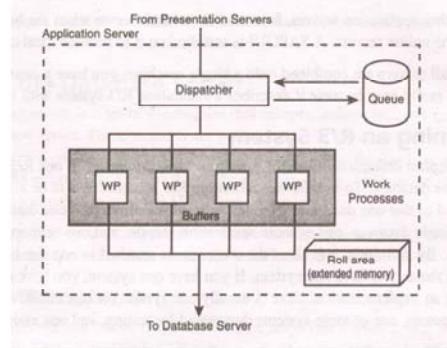


Fig: Application server architecture

Application Server Architecture

- All requests that come in from presentation servers are directed first to dispatcher.
- The dispatcher writes them first to the dispatcher queue.
- The dispatcher pulls the requests from the queue on a first-in, first-out basis.
- Each request is then allocated to the first available work process.
- A work process handles one request at a time.
- To perform any processing for a user's request, a work process needs to address two special memory areas: the *user context* and the *program roll area*.
- The *user context* is a memory area that contains information about the user, and the *roll area* is a memory area that contains information about the programs execution



Copyright © Capgemini 2015. All Rights Reserved 19

User Context

- A *user context* is memory that is allocated to contain the characteristics of a user that is logged on the R/3 system.
- It holds information needed by R/3 about the user, such as:
 - The user's current settings
 - The user's authorizations
 - The names of the programs the user is currently running
- When a user logs on, a user context is allocated for that logon.
- When they log off, it is freed.



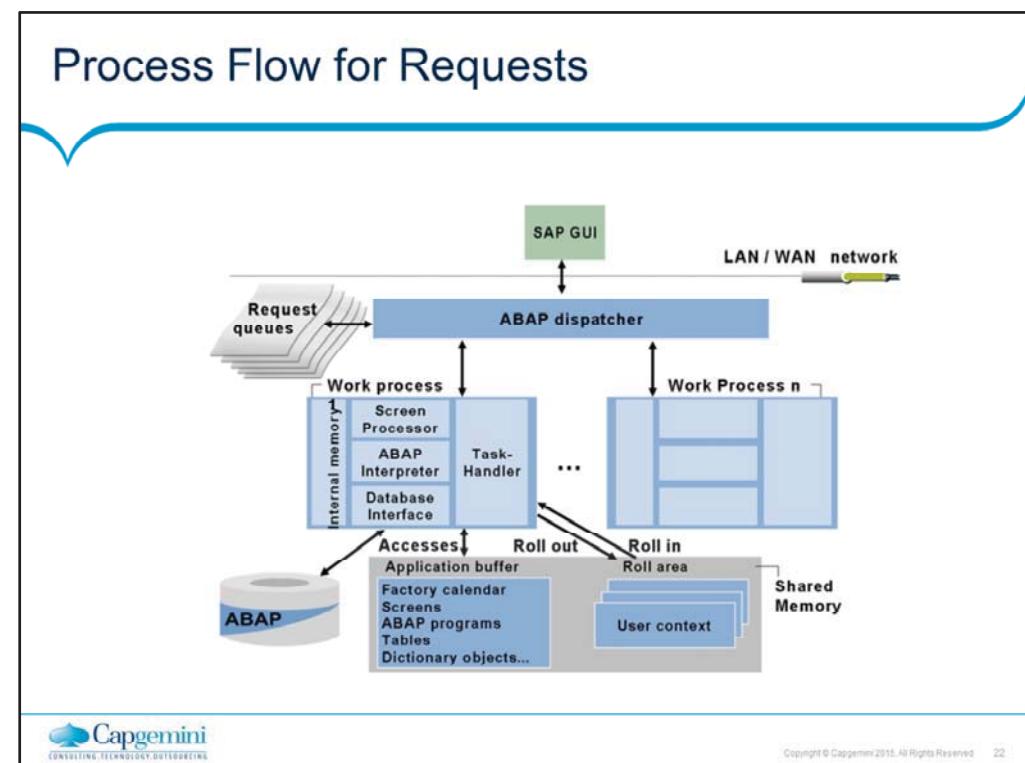
Copyright © Capgemini 2015. All Rights Reserved 20

Roll Area

- A roll area is memory that is allocated by a work process for an instance of a program.
- It holds information needed by R/3 about the program's execution, such as:
 - The values of the variables
 - The dynamic memory allocations
 - The current program pointer
 - Each time a user starts a program, a roll area is created for that instance of the program.
 - If two users run the same program at the same time, two roll areas will exist-one for each user.
 - The roll area is freed when the program ends.
 - The roll area and the user context play an important part in dialog step processing

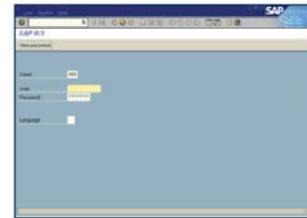


Copyright © Capgemini 2015. All Rights Reserved 21



Logon Client

- The term *logon client* has nothing to do with the Client/Server-it is completely different.
- The *logon client* refers to the number that the user types in the *Client* field on the logon screen.
- The number entered here by the user corresponds to a set of rows within each client-dependent table within the database



Client-Dependent and Client-Independent Tables

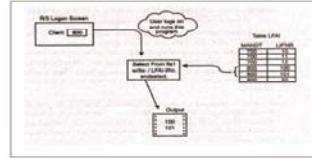
- There are two types of tables in the R/3 database: *client-dependent* and *client-independent*.
- A table is client-dependent if the first field is of type CLNT.
- The length will always be 3; and by convention, this field is named *mandt*.
- If the first field is not of type CLNT, the table is client independent



Copyright © Capgemini 2015. All Rights Reserved 24

Client-Dependent and Client-Independent Tables

- In the figure, the user logs on to client 800 and runs the program shown. This program selects rows from table Ifa1 and writes out Ifa1-lifnr. When this program is run, only two rows are selected: only those whose mandt equals 800. This happens automatically because the first field in the table is of the type CLNT.
- There are five rows in the table, but the program writes out only those rows where mandt equals 800.
- If the user were to log on to client 700 and run the same program, three rows of data would be found and written out. If the user were to log on to client 900, only one row of data would be found



Client-Dependent and Client-Independent Tables

- The logon client mechanism divides the rows within a client-dependent table into distinct groups.
- To access a different set of data, the user logs on and specifies a different client number.
- The user master records (containing R/3 user IDs) are client-dependent.
- Therefore, to gain access to a client, the system administrator must create a new user ID for you within that client
- Developers and testers use the logon client mechanism to create and access multiple, independent set of data within a single table



Copyright © Capgemini 2015. All Rights Reserved 26

Client-Dependent and Client-Independent Tables

- The average R/3 installation has three systems: *development*, *test*, and *production*.
- By default, each system comes with three clients installed: 000, 001, and 066.
- It is common to have from three to six clients in the development and test systems, but rarely will you see more than one client in production



Copyright © Capgemini 2015. All Rights Reserved 27

Using SAP's Open SQL

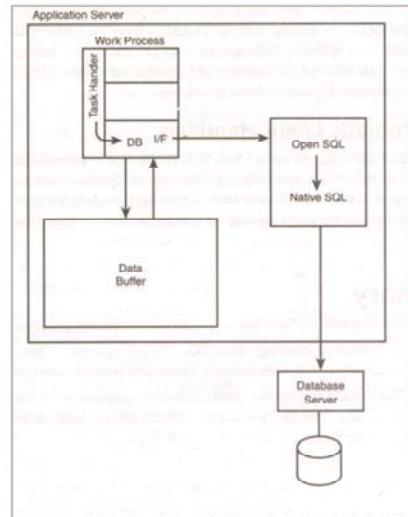
- ABAP/4 code is portable between databases.
- To access the database in an ABAP/4 program you will code SAP's *Open SQL*.
- Open SQL is a subset and variation of ANSI SQL.
- The ABAP/4 interpreter passes all Open SQL statements to the database interface part of the work process
- There, they are converted to SQL that is native to the installed RDBMS.
- For example, if you were running an Oracle database, your ABAP/4 Open SQL would be converted by the database interface to Oracle SQL statements.



Copyright © Capgemini 2015. All Rights Reserved 28

Using SAP's Open SQL

- If you use Open SQL, your SQL statements will be passed to the database interface.
- Using Open SQL has three main advantages.
- All of these advantages are implemented via the database interface



Portability

- The first advantage is the fact that your SQL statements will be portable between databases.
- For example, if for some reason your company wanted to switch from an Oracle to an Informix database, it could change the database, and your ABAP/4 code would continue to run without modification



Copyright © Capgemini 2015. All Rights Reserved 30

Buffering Data on the Application Server

- Secondly, the database interface buffers information from the database on the application server.
- When data is read from the database, it can be stored in the buffers on the application server.
- If a request were then made to access the same records, they would already be on the application server, and the request is satisfied from the buffer without having to go to the database.
- This buffering techniques reduces the load on the database sever and on the network link between the database and the application servers, and can speed up database access time by a factor of 10 to 100 times.



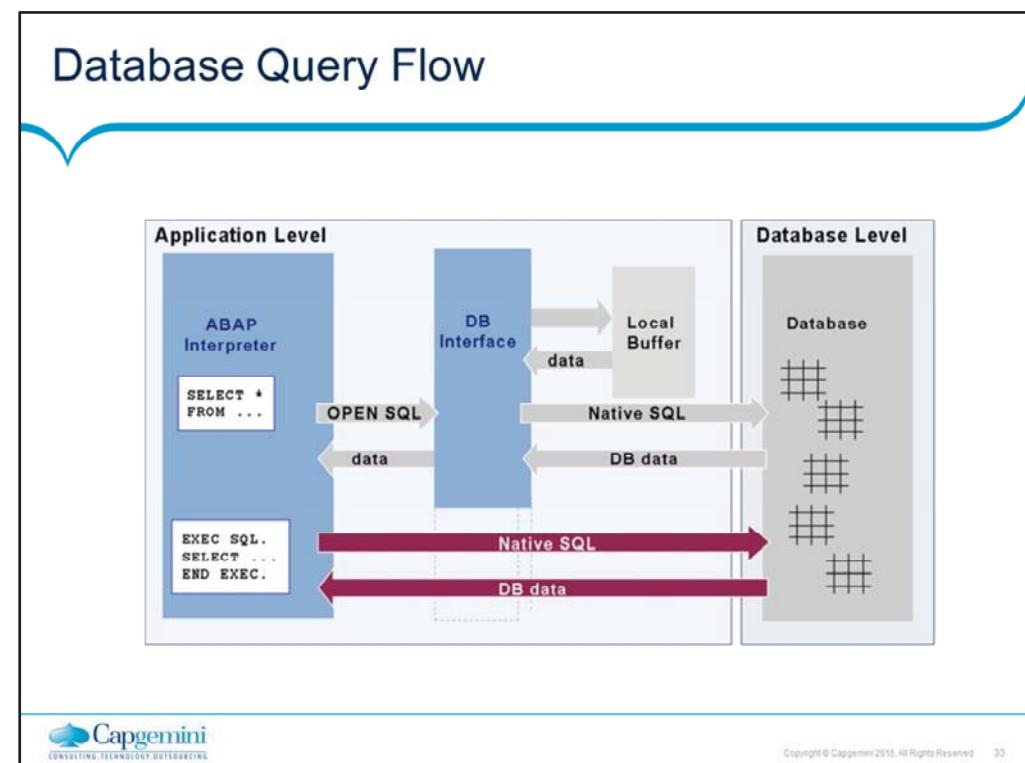
Copyright © Capgemini 2015. All Rights Reserved 31

Automatic Client Checking

- The third advantage of using Open SQL is *automatic client handling*.
- With Open SQL, the client field is automatically populated by the database interface.
- This gives your development and testing teams many advantages, such as ability to perform multiple simultaneous testing and training on a single database without interference from each other



Copyright © Capgemini 2015. All Rights Reserved 32



SAP R/3 – An Introduction

SYSTEMS APPLICATIONS and PRODUCTS
in Data Processing

R: Real Time

3: 3 Tier Client/Server Architecture

The diagram consists of two rectangular boxes with rounded corners. The left box contains the text 'R: Real Time' and has a blue arrow pointing down to it from above. The right box contains the text '3: 3 Tier Client/Server Architecture' and has a blue arrow pointing down to it from above. Both boxes are positioned below the main title 'SYSTEMS APPLICATIONS and PRODUCTS in Data Processing'.

Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 34

SAP – An Introduction

- ERP (Enterprise Resource Planning) Product
- Name of the Company and its Product
- German Based
- ERP Market Leader
- Industry Specific Best Practices



Copyright © Capgemini 2015. All Rights Reserved 35

SAP Logon

- A utility to logon to SAP
 - Choose an available SAP system
 - Program connects to the message server of that system and obtains the address of a suitable Application Server
 - Starts a SAP GUI (Graphical User Interface)
 - SAP GUI Starts the logon Screen
 - The user can open multiple sessions
 - Applications are run within a session
- The SAP GUI is based on Windows Style and is available for several Platforms, providing the same functions for each



Copyright © Capgemini 2015. All Rights Reserved 36

Basic Navigation

- Logging on to SAP
 - Client
 - Username
 - Password
 - Logon Language



The screenshot shows a SAP logon interface with the following fields:

- Client: 100
- User: [Redacted]
- Password: [Redacted]
- Logon Language: EN

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 37

Client: A unit in the R3 system, with separate set of Master data and its own set of Tables.

Client is used to group data within Database

The users have access to data for the specific Client

Once logged on, the user can work on several sessions

Demo

- SAP Logon



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 38

Basic Navigation

- Transaction Code
 - Acronym to access menu path
 - Sequence of Screens with Input and Output fields for Processing
- Possible Command Field Entries
 - /nxxxx – to call Transaction xxxx
 - /n – cancel Transaction
 - /oxxxx – to Call Transaction xxxx in a new Session
 - /o – display an overview of Sessions



Copyright © Capgemini 2015. All Rights Reserved 39

Basic Navigation

- Several options are available to log off from system
 - Menu Bar System Logoff
 - Choose Yellow Arrow in the SAP EASY ACCESS Menu. If several sessions are open, it only closes the session
 - Enter /nend in the command field
 - Enter /nex in the command field.



Copyright © Capgemini 2015. All Rights Reserved 40

Basic Navigation

- The Commonly Used Transaction Codes are
 - SE80 : Object Navigator (ABAP Development Workbench)
 - SE38 : ABAP Editor
 - SE37 : Function Builder
 - SE11 : ABAP Dictionary
 - SE21 : Package Builder
 - SE91 : Message Maintenance



Copyright © Capgemini 2015. All Rights Reserved 41

ABAP Workbench tools

- The ABAP Workbench is a collection of tools you use to develop, test and run ABAP programs
- Frequently Used Tools
 - ABAP Editor
 - ABAP Dictionary
 - Screen Painter
 - Menu Painter
 - Function Builder



Copyright © Capgemini 2015. All Rights Reserved 42

Review Question

- Question 1: _____ is like an operating system for R/3.
- Question 2: An _____ interprets the ABAP/4 programs and manage the input and output for them.
- Question 3: A _____ is memory that is allocated to contain the characteristics of a user that is logged on the R/3 system.
- The _____ is used to cancel the transaction.
- _____ is a collection of tools you use to develop, test and run ABAP programs



Summary

- In this lesson, you have learnt:
 - The R/3 System Architecture
 - The Application Server Architecture
 - The meaning of Logon Client
 - Advantages of using SAP's Open SQL
 - The Basics of SAP
 - How to Log on to SAP and do the Basic Navigations



ABAP Part I

Lesson 02: Introduction to ABAP
Programming

Lesson Objectives

- After completing this lesson, participants will be able to –
 - Understand The Need for ABAP
 - Know the types of ABAP/4 Programs
 - Create Reports
 - Write the Program Code
 - Test the Program
 - Know ABAP/4 Language Elements
 - Combine similar statements to one statement
 - Illustrate Defining Data Types and Data Objects
 - Recognize the System Variables



What is ABAP/4?

- ABAP/4 is a 4 generation programming language that you can use in three different ways:
 - You can select and edit data you want to process in traditional ways via the screens that guide you.
 - You can write reports using the interactive reporting facility.
 - ABAP/4 communicates with you via a dialog functions.
- ABAP/4 (Advanced Business Application Programming-4) language was developed by SAP to provide optimal working conditions for application programmers.
- It is the sole tool used by SAP to develop its own applications.
- SAP customers use ABAP/4 to adapt R/3 standard solutions to specific problems



Copyright © Capgemini 2015. All Rights Reserved 3

Features of ABAP/4

- Multi-Language Support
- Supports business data types and operations
- Open SQL
- Use of sub routines



Copyright © Capgemini 2015. All Rights Reserved. 4

Basic features of ABAP/4 are:

1) ABAP/4 contains

- Declarative elements for declaring data with various types and structures
 - Operational elements for data manipulation
 - Control elements for controlling the program flow
 - Event elements for reacting to external events
- 2) Multi-Language Support : Text elements (e.g. titles, headers, and other text) used in a program are stored separately from the program code. One can change, translate, and maintain these at any time without changing the program code.
- 3) Open SQL : ABAP/4 contains a subset of SQL called Open SQL. With Open SQL, one can read and access database tables regardless of the database system being used.
- 4) Use of Subroutines: ABAP/4 allows one to define and call subroutines. The subroutines may be in the same program or in other programs. Parameters can be passed to and from subroutines in various ways.

ABAP/4 also contains a special kind of subroutine known as a *Function Module*. The Function Modules are created and maintained in a central library. Function Modules have a clearly defined data interface between the calling program and the subroutine. They can be tested in a stand-alone mode independent of the calling program.

Report Program

- The purpose of a report is to read data from the database and write it out.
- It consists of only two screens.
 - The first screen is called the selection screen.
 - It contains input fields allowing the user to enter criteria for the report.
 - The second screen is the output screen.
 - It contains the list.
- The list is the output from the report, and usually does not have any input fields.
- The selection screen is optional. Not all reports have one. However, all reports generate a list.



Copyright © Capgemini 2015. All Rights Reserved 5

Report Program: An ABAP/4 report is a program that retrieves data from the database, groups/filters according to different criteria and presents it on screen or as a printed list. Reports are called TYPE 1 program. A type 1 program can be started by entering its program name.

They are also known as executable programs

Dialog programs

- In a typical dialog program, the system displays a screen where the user can input data.
- As a reaction to the user input, processing continues.
- The ABAP/4 code written to control the transaction is maintained in a collection of programs that together form the Module Pool
- Dialog Programs are called TYPE M programs



Copyright © Capgemini 2015. All Rights Reserved. 6

Type M programs can only be started using a transaction code.

A transaction code starts a screen, which consists of the screen itself and its flow logic.

Screen flow logic can call special processing blocks (dialog modules) in the corresponding ABAP/4 program.

Since type M programs contain mostly dialog modules, they are also known as module pools.

Development object

- A development object (also called Repository Object) is anything created by a developer.
 - Examples of development objects are programs, screens, tables, views, structures, data models, messages and includes
- All development objects are portable, meaning that you can copy from one R/3 system to another.
- This is usually done to move your development objects from the development system to production system.
- If the source and target systems are on different operating systems or use different database systems, your development objects will run as-is and without any modification.
- This is true for all platforms supported by R/3.



Copyright © Capgemini 2015. All Rights Reserved. 7

Creating Reports

- ABAP/4 report program can be created from the ABAP/4 editor (Transaction Code: SE38).
- Creating a report program involves the following steps
 - Creating the program
 - Specifying the program attributes
 - Writing the program code
 - Testing the program



Copyright © Capgemini 2015. All Rights Reserved. 8

ABAP/4 reports consist of five components (sub objects):

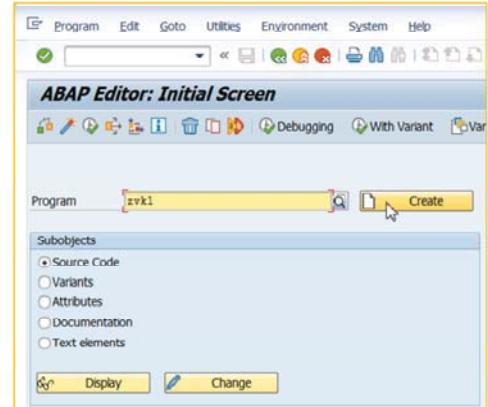
Source code
Attributes
Text elements
Documentation
Variants

Only the source code and the attribute components are required. The rest of the components are optional.

Note that all development objects and their subobjects are stored in the R/3 system database. For example, the source code for a report is stored in database table dd010s.

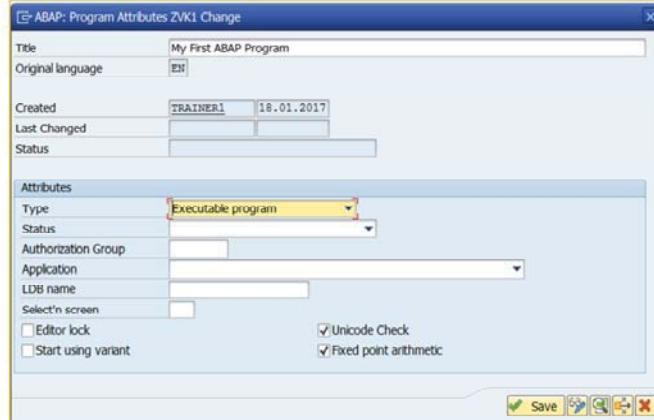
Creating program

- Run transaction SE38 to go to the ABAP/4 Editor.
- This enables to create report program.



Specifying program Attribute

- Program attributes determine to which application a program belongs .



Copyright © Capgemini 2015. All Rights Reserved 10

Program attributes determine to which application a program belongs and the logical database to which it is linked. Data to be entered include –
Title -The Title should ideally be descriptive of the function to be carried out by the program.

Type - The type defines the execution mode of the program.

Value 1 declares the program as a standalone, online program that can be run independently.

Value M, declares the program as a Module Pool. That means that the program cannot run standalone but serves as a main program for the program modules of dialog programming.

Value I, is an Include program. Program code in an Include program can be used by many programs and helps to modularize logically related code.

Status: The program status specifies whether the program is a test program, a system program or a production program. Depending on the status, some of the utilities cannot be used. This entry is optional. For E.g.: you cannot use ABAP/4 debugging with system programs.

Application: The Application field contains the abbreviation of the application, e.g. F for Financial Accounting. This enables the system to assign the program to the appropriate business area. If the program is not application specific, the '*' is used.

Authorization Group: Authorization group to which the program is assigned. The system

checks that the user belongs to an authorization group when

- Starting or editing a program
- Maintaining attributes
- Using other program development utilities

•Development class/Package: The development class is important for transporting between systems. When transporting, one can combine Workbench objects assigned to one development class together. Once the attributes are saved a dialog box is displayed to input the Development class. Test program should be assigned class \$TMP (or select the 'Local Object' button) Local Object cannot be transported from the development region to the testing region.

•Logical Databases (LDBs) from Application: This is only applicable for online programs(type 1) and determines which LDB the program uses to retrieve data from the database.

•Selection Screen Version: Is applicable to Online report using LDBs. LBDs may provide one or more selection screens on the basis of which data is retrieved. This field contains the selection screen number to be used.

•Uppercase/lowercase: If you want the ABAP/4 Editor to leave your program code as you have entered it when displaying and storing, leave this field blank. If you mark this field, all the program code (except text within quotation marks, and comments) is converted to uppercase.

•Editor lock: If this attribute is set, other users cannot modify, rename, or delete the program.

•Fixed Point Arithmetic :If this attribute is not set, the system does not recognize the decimal point for while using pack variables. Thus 3.14 would be interpreted as 314.

•Start via variant (report programs only): If this attribute is set, the user can only start your report program using a variant. Before starting the program at least one report variant must be created.

ABAP Syntax

▪ Rules for ABAP Syntax

- The first word in the statement is the ABAP keyword
- Each statement ends with a period (.)
- Comment line is marked with a ‘*’
- Comments from the middle of a line begins with “



Copyright © Capgemini 2015. All Rights Reserved 12

Writing Program

- ABAP/4 code is written from the ABAP/4 editor.



Copyright © Capgemini 2015. All Rights Reserved 13

To go to the ABAP/4 editor one can select the 'Source Code' button from the Attributes screen or return to the ABAP/4 editor main screen, select 'Source Code' from the Object Components and select the 'Change' button.

The system automatically enters the first ABAP/4 statement - the REPORT statement. I.e.:

REPORT <Report Name as input in the Initial Screen>.

Once the code is written, one can check the syntax by selecting the 'Check' button. Error messages, if any, are displayed and the cursor is placed on the line in which the error has occurred. After the code is error free it can be saved using the 'Save' button.

Test the Program

- To test the program, select the 'Execute' button.
- At runtime, the source code of the ABAP/4 program is compiled.
- This compilation process is known as generation.
- The generated form of the program is stored in the ABAP/4 repository.
- As the program is automatically generated at run time while execution, one does not have to generate it separately.
- The program will be regenerated at each run if some modifications have been made to the code.
- The ABAP/4 also provides for various debugging mechanisms



Copyright © Capgemini 2015. All Rights Reserved 14

ABAP/4 programs are interpreted; they are not compiled. The first time you execute a program, the system automatically generates a runtime object. The runtime object is pre-processed form of the source code. However, it is not an executable that you can run at the operating system level. Instead, it requires the R/3 system to interpret it. The runtime object is also known as the generated form of the program.

If you change the source code, the runtime object is automatically generated the next time you execute the program.

When you generate a development object, the system creates(compiles) a separate runtime object(LOAD) and stores it in the R/3 repository. This generated version is then the version that is executed(interpreted). Note that all changes to the development object become visible system-wide only when the program is activated. With inactive versions, you have a local, separate view of the R/3 repository, which provides the basis for a local runtime system. During Activation, the system creates a Run Time object. Upon execution, the system executes this Run Time Object

Demo

- Create first ABAP Program and execute it



Creating Reports

- ABAP/4 Language elements
 - A report consists of individual statements that start with a reserved word and end with a period.
 - E.g.

```
START-OF-SELECTION.  
WRITE XYZ.  
MOVE SALES TO TOTAL_SALES.
```
 - The first word of statement (the reserved word) determines the meaning of the whole statement.



Copyright © Capgemini 2015. All Rights Reserved 16

ABAP Language Elements

- Declarative Language Elements

- Declare the data items that can be addressed in the report:
 - TYPES, DATA, TABLES, PARAMETERS, SELECT-OPTIONS.
 - The declarative statement define data types and data objects

- Time Language Elements

- Specify the point in time (the event) when to execute a process.
 - START-OF-SELECTION, END-OF-SELECTION, AT SELECTION SCREEN.



Copyright © Capgemini 2015. All Rights Reserved 17

ABAP Language Elements

- Control Language Element
 - Control the processing flow:
 - IF...ENDIF, WHILE....ENDWHILE, CASE...ENDCASE.
- Operational Language Element
 - Process the data at certain times under certain conditions.
 - WRITE, MOVE.
 - OR
 - WRITE CITY UNDER STREET.



Copyright © Capgemini 2015. All Rights Reserved 18

Chained Statements

- Used to Combine statements
- The chain operator used is ‘:’
- Example
- Statement sequence:
 WRITE var1.
 WRITE var2.
 WRITE var3.
- Chain Statement:
- WRITE : var1, var2, var3.



Copyright © Capgemini 2015. All Rights Reserved 19

Processing a Report

- The ABAP/4 programming language is an event-oriented language.
- It does not necessarily process statements in sequential order.
- With timing language elements (START-OF-SELECTION etc), you can combine statements into “processing blocks”.
- Within a processing block ABAP/4 process the statements either sequentially or according to the control language elements.
- The sequence in which processing blocks are executed depends on when the associated event occurs.



Copyright © Capgemini 2015. All Rights Reserved 20

e.g.

report demo.

<declaration section>

at selection-screen. “ processing block
start-of-selection. “ processing block.

if ().

endif.

end-of-selection.

 write: ‘the sum is ‘, total.

A timing language element always introduces a new processing block. Note that there is no particular language element to mark the end of processing block. A new event or the definition of a subroutine (FORM...ENDFORM) ends the preceding processing block. For reasons of clarity you should list processing blocks in their order of execution.

ABAP Statements

- Declarative Statements
 - Define data types
 - Declare Data Objects
 - Examples
 - TYPES
 - DATA
 - TABLES



Copyright © Capgemini 2015. All Rights Reserved 21

Data Types

- Elementary Types
 - Fixed Length
 - 8 Predefined Types in ABAP
 - C → Character
 - N → Numeric Character
 - D → Date
 - T → Time
 - X → Hexa Decimal Byte Field
 - I → Integer
 - P → Floating Point Number
 - F → Packed Number
 - Variable Length
 - STRING
 - XSTRING



Copyright © Capgemini 2015. All Rights Reserved 22

Data objects

- Defined with the DATA keyword
- Physical units with which ABAP Statements work
- Has a set of technical attributes
 - Length
 - Number of Decimal Places
 - Data Type
- Can refer to an existing data object



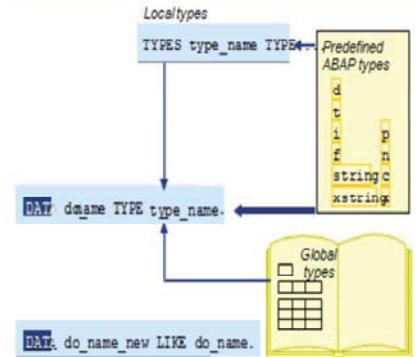
Copyright © Capgemini 2015. All Rights Reserved 23

Data objects

- Data objects are usually defined with the DATA statement
- After the name of the data object, a fully-specified type is assigned to it using the TYPE addition

Defining Data Objects

Defining Data Objects



Data objects

- Defining Data Objects

ABAP PROGRAM

```
DATA gd_myvar1 TYPE type_name  
DATA gd_myvar2 LIKE gd_myvar1.
```

Global Types ABAP Dictionary

Predefined Data Types

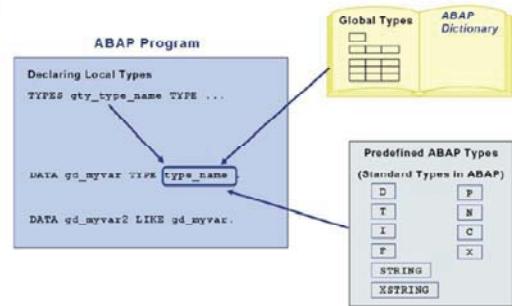
C	I
N	P
D	F
T	X

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 25

Data objects

- Data Objects are defined using the keyword DATA
- They are memory locations that you use to hold data while the program is running and interpret them according to the data type.
- Memory is allocated at runtime, depending on the size of the data object
- They are local to the program reused.



Data objects

- Data Objects whose contents can be changed using ABAP statements
 - Declared Using DATA, CLASS-DATA, STATICS, PARAMETERS, SELECT-OPTIONS statements
 - Example:
 - DATA name(20) TYPE c.
- Constants
 - Data Objects whose contents cannot be changed
 - Declared Using CONSTANTS Statement.
 - Example:
 - CONSTANTS pi TYPE p DECIMALS 3 VALUE '3.141'.



Copyright © Capgemini 2015. All Rights Reserved 27

When the program starts, the memory allocation for each data object occurs in the roll area of the program. While the program is running, you can read the contents of a non-modifiable data object or put data into a modifiable data object and then retrieve it. When the program ends, the system frees the memory for all data objects and their contents are lost.

Data objects - Visibility

- Data objects have three levels of visibility:
 - Local
 - Accessible only from inside the subroutine in which they are defined.
 - Global
 - Can be accessed from anywhere within the program
 - External
 - Are accessible from outside of the program by another program.
- The visibility of a data object indicates from where in the program the data object is accessible.



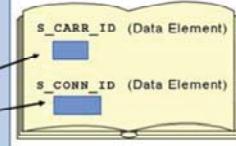
Copyright © Capgemini 2015. All Rights Reserved 28

Data objects

- Examples of the Definition of Elementary Data Objects

```
...
TYPES gty_percentage TYPE p LENGTH 3 DECIMALS 2.
DATA: percentage1 TYPE gty_percentage,
      percentage2 TYPE gty_percentage,
      number1      TYPE i VALUE 17,
      number2      LIKE number1,
      city         TYPE c LENGTH 15,
      carrid       TYPE s_carr_id,
      connid       TYPE s_conn_id.
```

percentage1	0 0 0 0 +
percentage2	0 0 0 0 +
number1	17
number2	0
city	
carrid	
connid	0 0 0 0



Data objects

- A literal is a non-modifiable data object.
- Literals can appear anywhere in a program, and they are defined merely by typing them where needed.
- There are two types:
 - Character string
 - Numeric



Copyright © Capgemini 2015. All Rights Reserved 30

Character literals are sequences of alphanumeric characters in the source code of an ABAP program enclosed in single quotation marks or backquotes. Character literals enclosed in quotation marks have the predefined ABAP type C and are described as **text field literals**. Literals enclosed in backquotes have the ABAP type STRING and are described as **string literals**.

Examples of text field literals:

'Antony Smith'
'69190 Walldorf'

Examples of string literals:

`Anton Schmitt`
`69190 Walldorf`

Number literals

Numeric literals are sequences of digits which may contain a plus or minus sign

Data objects

Literals and Constants (Fixed Data Objects)

Fixed Data Objects Without Label

Literals

Numeric Literals

Positive Integer : 123
Negative Integer : -123

Text Literals

String : 'Hello'
Decimal Number : '123.45'
Floating Point Number : '123.45E01'

Fixed Data Objects with Label

Constants

```
CONSTANTS gc_myconst TYPE type_name VALUE { literal | IS INITIAL }.
```



Copyright © Capgemini 2015. All Rights Reserved 31

Demo

- Program on using Data Objects



Pre-Defined Data Object

- SPACE
 - Constant
 - Type C
 - Length - 1 byte
 - Contains Space Character



Copyright © Capgemini 2015. All Rights Reserved 33

Data objects - Operations

- Assigning Values
 - MOVE
 - MOVE source TO destination
- Examples
 - MOVE '5.7' TO number.
 - DATA : num1 TYPE i, num2 TYPE i.
 - num1 = 10.
 - MOVE num1 TO num2.



Copyright © Capgemini 2015. All Rights Reserved 34

Data Objects – Operations (Contd.).

- Assigning Values

- MOVE-CORRESPONDING

- MOVE-CORRESPONDING sourcestruct TO destinationstruct

- Examples

```
DATA : BEGIN OF address,
```

```
    fname(15) TYPE c VALUE 'Robert',
```

```
    lname(15) TYPE c VALUE 'David',
```

```
    compname(30) TYPE c VALUE 'WIPRO TECHNOLOGIES',
```

```
    number TYPE i VALUE '72',
```

```
    street(30) TYPE c VALUE 'Keonics Electronic City',
```

```
    city(10) TYPE c VALUE 'BANGALORE',
```

```
END OF address.
```

```
DATA : BEGIN OF name,
```

```
    lname(15) TYPE c ,
```

```
    fname(15) TYPE c ,
```

```
END OF name.
```

```
MOVE-CORRESPONDING address TO name.
```



Copyright © Capgemini 2015. All Rights Reserved 35

Data Objects – Operations (Contd.).

- Assigning Values

- WRITE TO

- WRITE <f1> TO <f2> [<option>].

- This statement converts <f1> to TYPE C and places the string in <f2>

- Example

- DATA : number TYPE f VALUE '4.38',
text(10).

```
WRITE number TO text .  
WRITE text.
```

OUTPUT: 4.380E+00



Copyright © Capgemini 2015. All Rights Reserved 36

Resetting Variables to Initial Values

- **CLEAR var.**

- Resets var to appropriate initial value for its type
- Cannot use CLEAR to reset a CONSTANT
- Has different effect for different Data Types
 - Elementary ABAP Types
 - Sets the value of the variable to initial value
 - References
 - Resets reference variable to initial value, so that it doesn't point to any object
 - Structures
 - Resets individual components of a structure to their respective initial values
 - Internal Tables
 - Deletes the entire contents of Internal Table



Copyright © Capgemini 2015. All Rights Reserved 37

CLEAR var.

- Resetting Variables to Initial Values(Contd.).

- Example

```
DATA number TYPE i VALUE '10'.
WRITE number.
CLEAR number.
WRITE number.
```

- Output : 10 0



Copyright © Capgemini 2015. All Rights Reserved 38

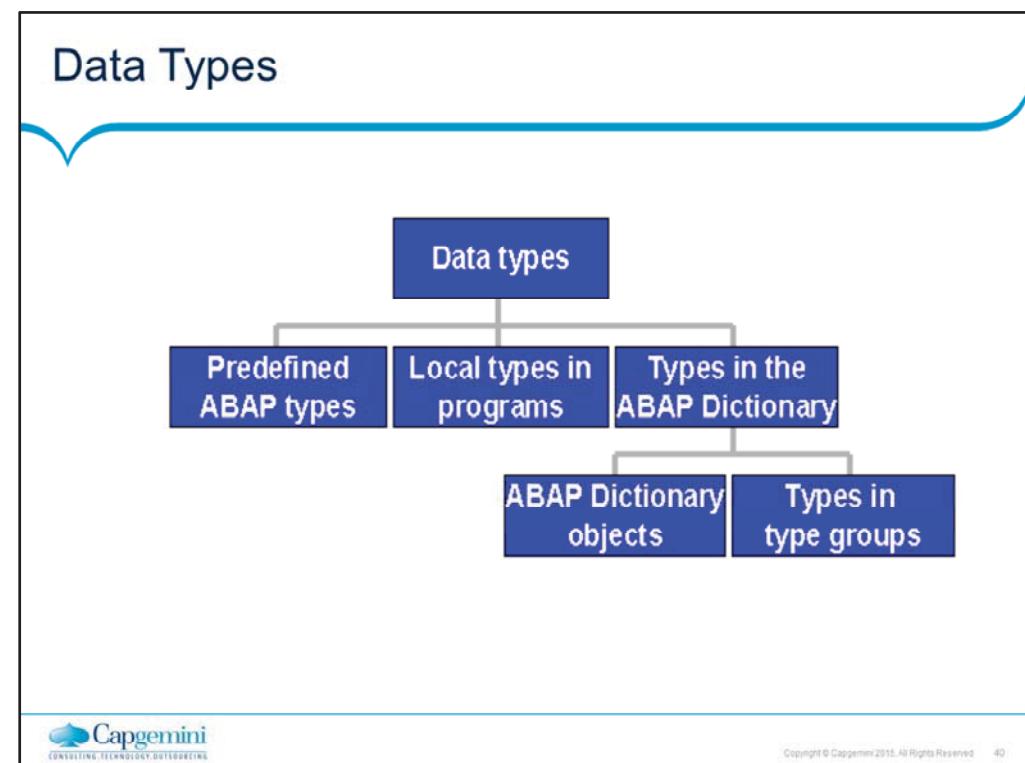
Arithmetic Operations

- The following arithmetic operators are used in mathematical expressions:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
DIV	Integer division
MOD	Reminder of Integer Division
**	Powers



Copyright © Capgemini 2015. All Rights Reserved 39



Data Types

- ABAP types d, t, i, f, string, and xstring are used directly to type data objects and are therefore called fully specified types.
- ABAP Types c, n, and x must include additional information before using them to define data objects:

Predefine ABAP Types			
	Description	Length in bytes	Attributes
Fixed length	Data Numeric i c p	Integer Float point number Packed number	4 8 1...16
			Differ in: Rules for storage Value range Arithmetic used
	Character string type n c d t	Number sequence Character sequence Date Time	1...65535 1...65535 8 6
Variable length	Character string type / hexadecimal string xstring	Character sequence Hexadecimal code	Character string operations (allowed for all types) + date calculations + time calculations
		1...65535	Bit operations
			Runtime system adjusts length dynamically

Data Types

- Complete ABAP Standard Types
 - D Type for date(D), format: YYYYMMDD, length 8 (fixed)
 - T Type for time (T), format: HHMMSS, length 6 (fixed)
 - I Type for integer (I), length 4 (fixed)
 - F Type for floating point number (F), length 8 (fixed)
 - STRING Type for dynamic length character string
 - XSTRING Type for dynamic length byte sequence (Hexadecimal string)



Copyright © Capgemini 2015. All Rights Reserved 42

Data Types

- Incomplete ABAP Standard Types
 - C Type for character string (Character) for which the fixed length is to be specified
 - N Type for numerical character string (Numerical character) for which the fixed length is to be specified
 - X Type for byte sequence (HeXadecimal string) for which the fixed length is to be specified
 - P Type for packed number (Packed number) for which the fixed length is to be specified. (In the definition of a packed number, the number of decimal points may also be specified.)



Copyright © Capgemini 2015. All Rights Reserved 43

Recommendations for Using Numeric Data Types

▪ Recommendations for Using Numeric Data Types

Required:	Recommended predefined ABAP data type:
Integers only	Type i, since calculations using integer arithmetic are fastest
Decimal numbers for business calculations	Type p
Decimal numbers for rough calculations performed on very small or very large numbers	Type F



Copyright © Capgemini 2015. All Rights Reserved 44

Predefined ABAP Types for Character Strings

- The initial value of each character string with fixed length is a space character.
- The initial value of each character in a numeric string is a zero.
- The initial value of a date is '0000000'.
- The initial value of a time is '0000000'.

Description	Type t	Type d	Type n	Type a	Type string
Time	Date	Sequence of digits	Fixed length char. string	Char. string of variable length	
Length	6Chars	8 digits	1 .. 65535 characters	1 .. 65535 characters	Variable
Value range	By clock	By Gregorian calendar	Digits		Depends on code page
Calculations	Time arithmetic	Date arithmetic	Conversion	Conversion	Conversion
Formatting options	HH:MM:SS	YYYYMMDD			

Data Types – User Defined

- Local Elementary Data Types are defined with reference to predefined elementary types
- Local to the program in which they are defined, cannot be reused
 - Syntax:
 - TYPES <t>[(<length>)] [TYPE <type>|LIKE <obj>] [DECIMALS <dec>]
 - Where type is a predefined data type.
 - If TYPE or LIKE is not used , system takes the default type C
- Example:
 - TYPES number TYPE I.
 - TYPES length TYPE p decimals 2.
 - TYPES code(3) TYPE c.



Copyright © Capgemini 2015. All Rights Reserved 46

Data Types (Contd.).

- **Reference Types**

- Describes Data Objects that contain references to other objects
- No predefined references

- **Complex Types**

- Allows to manage and process related data under a single name
- No predefined complex Type in ABAP
- Further divided into
 - Structures
 - Internal Tables



Copyright © Capgemini 2015. All Rights Reserved 47

Data Types - Complex Data Types

- Structures

- Sequence of any elementary types, reference types or complex data types
- Used in ABAP Programs to group work areas that logically belong together
- Example:

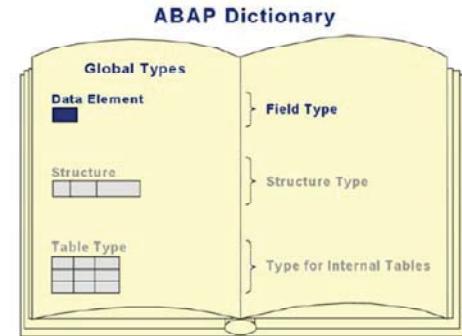
```
TYPES : BEGIN OF address,  
        name(20) TYPE c,  
        street(30) TYPE c,  
        city(20) TYPE c,  
    END OF address.
```



Copyright © Capgemini 2015. All Rights Reserved 48

Global Types

- Global Data Types in the Dictionary
 - Global Types are defined in SE11
 - No Memory is consumed at runtime
 - Can be reused by any program created in ABAP Workbench



Demo

- Program on creating and using Data Types



System Fields

- Few System Fields from Structure SY
 - SY-SUBRC
 - Return code for ABAP statements
 - 0 - if a statement is executed successfully
 - SY-UNAME
 - Logon name of the user
 - SY-REPID:
 - Current ABAP program
 - SY-TCODE
 - Current Transaction
 - SY-INDEX
 - Number of the current loop pass



Copyright © Capgemini 2015. All Rights Reserved 51

System Fields – Contd..

- System Fields from Structure SY – Contd..
 - SY-DATUM:
 - Current System Date
 - SY-UZEIT
 - Current System Time



Copyright © Capgemini 2015. All Rights Reserved 52

Demo

- Program on using System Fields



Reporting Concepts

- Following are the options for ReportName
 - 1.REPORT ZREPNAME [NO STANDARD PAGE HEADING] [LINE-SIZE COL] [LINE-COUNT n(m)] [MESSAGE-ID mid].
 - 2.NO STANDARD PAGE HEADING - Suppresses output of the standard page header
 - 3.LINE-SIZE COL - Creates a report with COL columns per line.
 - 4.LINE-COUNT n(m) - Creates a report list with n lines per page, of which m lines are reserved for the END-OF-PAGE processing.
- SUBMIT <rep> [AND RETURN] [<options>]. starts the report whose name is stored in field <rep>



Copyright © Capgemini 2015. All Rights Reserved 54

Reporting Concepts - Text Symbols

- Text symbols are simple text literal.
- They can be translated to any other Language.
- You can address text symbols in a program one of two ways:
- TEXT-<xxx> (xxx is a three digit character sequence)
- '<Text>'(<xxx>) (xxx is a three digit character sequence)



Copyright © Capgemini 2015. All Rights Reserved 56

Reporting Concepts - Write Statement

- Positioning Write on Output List
- Syntax
 - WRITE AT [/][<POs>][(LEN>)] <f>.
 - where '/' denotes a new line,
 - <pos> is a number or variable up to three digits long denoting the position on the screen,
 - <Len> is a number or variable up to three digits long denoting the output length.
 - In the default setting, you cannot create empty lines with the WRITE statement.
 - WRITE: 'One', / '', / 'Two'.
 - This produces the following output:
 - One
 - Two



Copyright © Capgemini 2015. All Rights Reserved 56

Write- Formatting Options

■ Syntax

- WRITE <f> <option>.
- Formatting options for all data types

Option	Function
LEFT-JUSTIFIED	Output is left-justified.
CENTERED	Output is centered.
RIGHT-JUSTIFIED	Output is right-justified.
UNDER <g>	Output starts directly under field <g>.
NO-GAP	The blank after field <f> is omitted.
USING EDIT MASK <m>	Specifies format template <m>.
USING NO EDIT MASK	Deactivates a format template specified in the ABAP Dictionary.
NO-ZERO	If a field contains only zeros, these are replaced by blanks. For type C and N fields, leading zeros are replaced automatically.



Copyright © Capgemini 2015. All Rights Reserved 57

Write- Formatting Options

- Formatting options for Numeric Fields

Option	Function
NO-SIGN	The leading sign is not displayed on the screen.
DECIMALS <d>	<d> defines the number of digits after the decimal point.
EXPONENT <e>	In type F fields, the exponent is defined in <e>.
ROUND <r>	Type P fields are multiplied by $10^{**(-r)}$ and then rounded.
CURRENCY <c>	Format according to currency <c> in table TCURX.
UNIT <u>	The number of decimal places is fixed according to unit <u> specified in table T006 for type P fields.



Copyright © Capgemini 2015. All Rights Reserved 58

Write- Formatting Options

- Format Color n.
- Format Color n Intensified On.
- FORMAT COLOR OFF INTENSIFIED OFF INVERSE OFF HOTSPOT OFF INPUT Off

No.	Color	INTENSIFIED	INTENSIFIED OFF	INVERSE
0	COL_BACKGROUND	0123456789	0123456789	0123456789
1	COL_HEADING	0123456789	0123456789	0123456789
2	COL_NORMAL	0123456789	0123456789	0123456789
3	COL_TOTAL	0123456789	0123456789	0123456789
4	COL_KEY	0123456789	0123456789	0123456789
5	COL_POSITIVE	0123456789	0123456789	0123456789
6	COL_NEGATIVE	0123456789	0123456789	0123456789
7	COL_GROUP	0123456789	0123456789	0123456789



Copyright © Capgemini 2015. All Rights Reserved 59

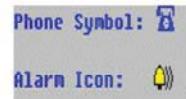
Write- Formatting Options

■ Displaying Symbols and Icons on the List

- You can output symbols or R/3 icons on a list by using the following syntax:

▪ Syntax:

- WRITE <symbol-name> AS SYMBOL.
- WRITE <icon-name> AS ICON.
- To make symbols and icons available to your program, you must import the appropriate include or the more comprehensive include <LIST> in your program.
- INCLUDE <symbol>.
- INCLUDE <icon>.
- WRITE: / 'Phone Symbol:', SYM_PHONE AS SYMBOL.
- SKIP.
- WRITE: / 'Alarm Icon: ', ICON_ALARM AS ICON.
- This produces the following output:



Write- Formatting Options

- Horizontal lines

- You can generate horizontal lines on the output screen by using the following
 - Syntax: ULINE [AT [/][<pos>]][(<len>)].

- Vertical lines

- You generate vertical lines on the output screen by using the following
 - Syntax: WRITE [AT [/][<pos>]] SY-VLINE.

- Blank lines

- You can generate blank lines on the screen by using the following :
 - Syntax: SKIP [<n>].



Copyright © Capgemini 2015. All Rights Reserved 61

Demo

- Program on options of write statement



Reporting - Displaying Field Content as Checkbox

- You can output the first character of a field as a checkbox on the output screen by using the following
 - Syntax: WRITE <f> AS CHECKBOX.
 - If the first character of field <f> is an "X", the checkbox is displayed filled.
 - If the first character is SPACE, the checkbox is displayed blank
- DATA: flag1(1) TYPE c VALUE '',
flag2(1) TYPE c VALUE 'X',
flag3(5) TYPE c VALUE 'Xenon'.
- WRITE: / 'Flag 1 ', flag1 AS CHECKBOX,
/ 'Flag 2 ', flag2 AS CHECKBOX,
/ 'Flag 3 ', flag3 AS CHECKBOX.



Copyright © Capgemini 2015. All Rights Reserved 63

Reporting – Determine list Width

- To determine the width of the output list, use the LINE-SIZE option of the REPORT statement.
- While creating the list, the system field SY-LINSZ contains the current line width.
- Syntax:** REPORT <rep> LINE-SIZE <width>.

```
REPORT demo_list_line_size LINE-SIZE 40.  
WRITE: 'SY-LINSZ', SY-LINSZ.  
ULINE.  
DO 20 TIMES.  
  WRITE sy-index.  
ENDDO.
```

Defining the width		
SY-LINSZ: 40		1
1 2 3		
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18
19	20	

```
REPORT demo_list_line_size LINE-SIZE 60.  
WRITE: 'SY-LINSZ', SY-LINSZ.  
ULINE.  
DO 20 TIMES.  
  WRITE sy-index.  
ENDDO.
```

Defining the width					
SY-LINSZ: 60					1
1 2 3 4 5					
6	7	8	9	10	
11	12	13	14	15	
16	17	18	19	20	



Reporting – Determine page Length

- To determine the page length of an output list, use the LINE-COUNT option of the REPORT statement.
- Syntax:** REPORT <rep> LINE-COUNT <length>[(<n>)].
- If you specify the optional number <n>, the system reserves <n> lines of the page length for the page footer.

```
REPORT DEMO_LIST_END_OF_PAGE LINE-SIZE 40
LINE-COUNT 6(2) NO STANDARD PAGE HEADING.
TOP-OF-PAGE.
WRITE: 'Page with Header and Footer'.
ULINE AT /(27).
END-OF-PAGE.
ULINE.
WRITE: /30 'Page', sy-pagno.
START-OF-SELECTION.
DO 6 TIMES.
  WRITE / sy-index.
ENDDO.
```

Page with Header and Footer		
1		Page 1
2		
Page with Header and Footer		
3		Page 2
4		
Page with Header and Footer		
5		
6		



Reporting- Programming Page Breaks - Unconditional Page Break

- To trigger a page break during list processing, use the basic form of the NEW-PAGE statement:
 - Syntax:** NEW-PAGE.
 - This statement ends the current page. All other output appears on a new page

```
REPORT DEMO_LIST_NEW_PAGE LINE-SIZE 40.  
TOP-OF-PAGE.  
WRITE: 'TOP-OF-PAGE', SY-PAGNO.  
ULINE AT (/17).  
START-OF-SELECTION.  
DO 2 TIMES.  
  WRITE / 'Loop:'.  
  DO 3 TIMES.  
    WRITE / sy-index.  
  ENDDO.  
  NEW-PAGE.  
ENDDO.
```

Standard Page Header	1
TOP-OF-PAGE	1
Loop:	1 2 3
Standard Page Header	2
TOP-OF-PAGE	2
Loop:	1 2 3



Reporting- Programming Page Breaks - Unconditional Page Break

- To execute a page break on the condition that less than a certain number of lines is left on a page, use the RESERVE statement:
 - Syntax: RESERVE <n> LINES.

```
REPORT DEMO_LIST_RESERVE LINE-SIZE 40
LINE-COUNT 8(2).

END-OF-PAGE.
ULINE.
START-OF-SELECTION.
DO 4 TIMES.
  WRITE / SY-INDEX.
ENDDO.
DO 2 TIMES.
  WRITE / SY-INDEX.
ENDDO.
RESERVE 3 LINES.
WRITE: / 'LINE 1',
       / 'LINE 2',
       / 'LINE 3'.
```

Standard Page Header	1
1	2
3	4
Standard Page Header	2
1	2
Standard Page Header	3
LINE 1	
LINE 2	
LINE 3	



Reporting - Standard Page Headers of Individual Pages

- The standard page header consists of list and column headers.
- To influence the representation of these individual components of the standard page header, use the following options.
 - Syntax: NEW-PAGE [NO-TITLE|WITH-TITLE] [NO-HEADING|WITH HEADING].

```
REPORT DEMO_LIST_NEW_PAGE_OPTIONS  
LINE-SIZE 40.  
  
WRITE: 'PAGE', SY-PAGNO.  
NEW-PAGE NO-TITLE.  
WRITE: 'PAGE', SY-PAGNO  
NEW-PAGE NO-HEADING.  
WRITE: 'PAGE', SY-PAGNO.  
NEW-PAGE WITH-TITLE.  
WRITE: 'PAGE', SY-PAGNO.  
NEW-PAGE WITH-HEADING.  
WRITE: 'PAGE', SY-PAGNO.
```

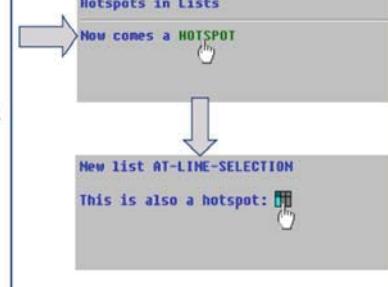
Standard Page Header	
Column	1
Page	1
Column	2
Page	2
Page	3
Standard Page Header	4
Page	4
Standard Page Header	5
Column	5
Page	5



Reporting - Output Fields as Hotspots

- If the user clicks once onto a hotspot field, an event is triggered (for example,
- AT LINE-SELECTION). To output areas as hotspots, use the following option of the FORMAT statement:
 - Syntax: FORMAT HOTSPOT [ON|OFF].

```
REPORT DEMO_LIST_FORMAT_HOTSPOT.  
INCLUDE <LIST>.   
START-OF-SELECTION.  
WRITE 'NOW COMES A'.  
FORMAT HOTSPOT ON COLOR 5 INVERSE ON.  
WRITE 'HOTSPOT'.  
FORMAT HOTSPOT OFF COLOR OFF.  
AT LINE-SELECTION.  
WRITE /'NEW LIST AT-LINE-SELECTION'.  
SKIP.  
WRITE 'THIS IS ALSO A HOTSPOT'.  
WRITE ICON_LIST AS ICON HOTSPOT.
```



Summary

- In this lesson, you have learnt:
 - The Need for ABAP
 - The types of ABAP/4 Programs
 - How to create, test and execute reports
 - ABAP/4 Language Elements
 - Chain Statement
 - Data Types and Data Objects
 - System Variables



Review Question

- Question 1: The ____ system variable displays the current ABAP program.
 - Option 1: SY-PRGNAME
 - Option 2: SY-REPID
 - Option 3: SY-PROG
 - Option 4: SY-PROG
- Question 2 : ____ triggers a page break during list processing.



ABAP Part I

Lesson 03: Selection Screen

Lesson Objectives

- After completing this lesson, participants will be able to understand -
 - Learn to create a selection screen



Defining Selection Screens

- There are three ABAP statements for defining selection screens:
 - PARAMETERS for single fields
 - SELECT-OPTIONS for complex selections
 - SELECTION-SCREEN for formatting the selection screen and defining user-specific selection screens
- The selection screen that is defined using PARAMETERS or SELECT-OPTIONS statements on their own, has a standard layout in which all parameters appear line by line.
- This layout is not always sufficient.
- For example, when you define a group of radio buttons, you should set off these buttons against other input fields so that the user can identify them as a group.



Copyright © Capgemini 2015. All Rights Reserved 3

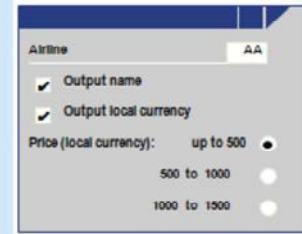
When defining input screens for the use of reporting on data, various selection elements can be combined to maintain data integrity and still be easy to use. These elements include standard input fields such as selection options and parameters, as well as others such as check boxes, radio buttons, and pushbuttons all arranged on a selection screen

Selection Screen: Declaring Fields with Parameters

```
PARAMETERS :      <E>[TYPE <type>][DECIMALS <n>][LIKE <f1>][MEMORY #Eds*]
[OBLIGATORY][DEFAULT <value>][LOWER CASE][VALUE CHECK]
[AS CHECKBOX]
[RADIOBUTTON GROUP <grp>]
```

```
REPORT sapbc405_  ascd_checkbox_ radiobutton .
...
PARAMETERS: pa_carr  LIKE aflight -carrid ,
             pa_name AS CHECKBOX DEFAULT 'X',
             pa_curr AS CHECKBOX DEFAULT 'X',
             pa_lim_1  RADIOBUTTON GROUP lim,
             pa_lim_2  RADIOBUTTON GROUP lim,
             pa_lim_3  RADIOBUTTON GROUP lim.
CONSTANTS mark VALUE 'X'.
* Check, if any checkbox has been selected
IF pa_name EQ mark. .... ENDOIF.
IF pa_curr EQ mark. .... ENDOIF.

* Check, which radiobutton has been selected
CASE mark.
  WHEN pa_lim_1. .... .
  WHEN pa_lim_2. .... .
  WHEN pa_lim_3. .... .
ENDCASE.
```



Copyright © Capgemini 2015. All Rights Reserved 4

PARAMETERS Statement

- A parameter is a special type of variable that is defined using the parameters statement.
- Parameters is a lot like the data statement, but when you run the program, the system will display the parameters as input fields on a selection screen before the program actually begins to execute.
- The rules for parameters names are the same as for variables names, except for the following:
 - The maximum length is 8 characters instead of 30.
 - In contrast to report-internal fields, you assign the initial value of a parameter using the parameter default.



Copyright © Capgemini 2015. All Rights Reserved 5

Syntax for PARAMETERS Statement

- The following code is the syntax for defining a variable using the parameters statement.

```
parameters p1[ (l) ]      [type t]      [decimals d]
```

Or

```
parameters p1 like v1  
[default 'xxx'] [obligatory] [lower case] [as check box]  
[radiobutton group g]
```



Copyright © Capgemini 2015. All Rights Reserved. 6

Where:

p1 is the parameter name

v1 is the name of a previously defined variable or parameter, or is the name of a field that belongs to a table or structure in the Data Dictionary

(l) is the internal length specification

t is the data type

d is the number of decimal places (used only with type p)

'xxx' is a literal or previously defined variable that supplies a default value

Example

```
parameters p1(2)          type c.  
parameters p2              like p1.  
parameters max_value       type i default 100.  
parameters cur_date        type d default '20030827'  
                           obligatory.  
parameters cur_date        Like sy-datum default sy-datum  
                           obligatory.
```



Example

- You declare parameters in the PARAMETERS statement analogous to report internal fields:

```
PARAMETERS: NAME(30) OBLIGATORY DEFAULT 'Renu ',  
            AGE(2)      TYPE P.
```

- In contrast to report-internal fields, you assign the initial value of a parameter using the parameter DEFAULT.
- The parameters statement can not be used for type f .



Copyright © Capgemini 2015. All Rights Reserved 8

Additions to Parameter statement

Addition	Use
Type	Same as the data
decimals	Same as the data
Like	Same as the data
default	Same as the value addition on the data statement
obligatory	The user must enter a value into the field before the program will execute
lower case	Prevents values from being translated to uppercase
as checkbox	Displays the input field as check box
Radio button group g	Displays the input field as a radio button belonging to group g



Copyright © Capgemini 2015. All Rights Reserved 9

Additions to Parameter statement

■ Using the Addition : Lower Case

- All values entered into a parameter are translated into uppercase by default. To turn off this translation, use the addition *lower case*.
- This translation applies only to character fields.

■ Using the Addition : Check Box

- A checkbox has two states: ticked and clear.
- You use them when you want to present the user with an on/off or true or false type of choice.
- You can use more than one checkbox on a screen.

E.g. parameters: cb1 as checkbox default 'X',
cb2 as checkbox,



Copyright © Capgemini 2015. All Rights Reserved 10

Additions to Parameter statement

- Using the Addition : radiobutton group
 - Like check boxes, a radio button also has two states: selected and not selected.
 - Unlike check boxes, radio buttons never operate alone; they operate in groups.
 - You can have any number of radio buttons in a group (greater than 1), but only one can be selected at a time.
 - They are used when you need to present the user with a list of alternatives in which only one option can be chosen
- To display a parameter as a radio button, use the addition radiobutton group g.
 - E.g. parameters: rb1 radiobutton group g1 default 'X',
rb2 radiobutton group g1,
rb3 radiobutton group g1.



Copyright © Capgemini 2015. All Rights Reserved 11

To display a parameter as a radio button, use the addition radiobutton group g. You cannot specify a data type or length; it will default to type c and length 1. g is an arbitrary group name one to four characters long. You can have more than one group in a program. The parameter will contain a capital X, if the radio button is selected; it will contain a blank if not selected. To be initially selected, the radio button should contain a default value of capital X. No other values are valid for a radio button. E

Parameter Fields

■ Parameter Input Field Labels

- On the selection screen to the left of each parameter's input field is a label.
- By default, the label is the same as the name of the parameter.
- You can set these labels manually.
- For parameters defined like Data Dictionary fields, you can retrieve the label automatically from the data element.

■ Changing Parameter Labels

- You can change the labels for the parameters appearing on the selection screen by using the text symbols.
- Follow this path to change the text symbols.
 - (Menu bar –GOTO->TEXT ELEMENTS ->SELECTION TEXTS)



Copyright © Capgemini 2015. All Rights Reserved 12

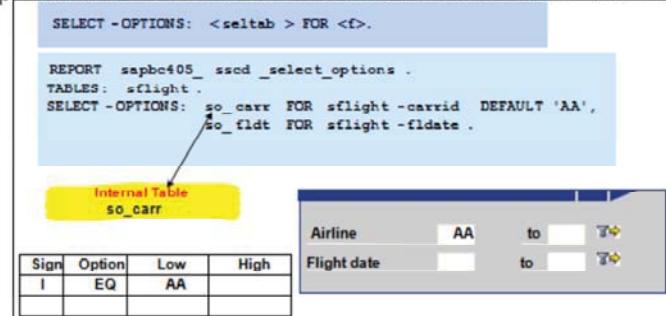
Demo

- Program on using Parameters and its options



Selection Screen: Declaring Fields with Select-Options

- MEMORY ID <PID>, the system retrieves the current value from SAP system memory and supplies it on the screen automatically
- OBLIGATORY generates a mandatory field.
- NO-EXTENSION suppresses multiple single or multiple range selections.
- NO INTERVALS suppresses the SELTAB-HIGH (upper interval limit) entry on the selection screen.



Demo

- Program on using select-options



Copyright © Capgemini 2015. All Rights Reserved 15

Defining Selection Screens

- The SELECTION-SCREEN statement has its own formatting options that you can use to define the layout for selection screens.
- You can define the layout of parameters and selection criteria and display comments and underlines on the selection screen.
- In addition, you can place pushbuttons in the application toolbar and on the screen itself



Copyright © Capgemini 2015. All Rights Reserved 16

User-defined selection screens

- Defined Using :

SELECTION-SCREEN BEGIN OF SCREEN numb [TITLE tit] [AS WINDOW].

...

SELECTION-SCREEN END OF SCREEN numb

- Define a user-defined selection screen with screen number numb

- Screen number numb is a four-digit number other than 1000

- AS WINDOW

- User-defined selection screen is called as a modal dialog box



Copyright © Capgemini 2015. All Rights Reserved 17

Formatting Selection Screen

- A standard layout is defined for the selection screen when defined using the PARAMETERS or SELECT-OPTIONS statements
- Changing Standard layout
 - use SELECTION-SCREEN statement
 - Has its own formatting options



Copyright © Capgemini 2015. All Rights Reserved 18

Blank Lines, Underlines, and Comments

- **Blank Lines**

- To place blank lines on the Selection screen, use
SELECTION-SCREEN SKIP [n]

- **Underlines**

- To place underlines on the Selection screen, use
SELECTION-SCREEN ULINE [/]pos(len) [MODIF ID key]

- **Comments**

- To place comments on the Selection screen, use
SELECTION-SCREEN COMMENT [/]pos(len) comm [FOR FIELD f]
[MODIF ID key]



Copyright © Capgemini 2015. All Rights Reserved 19

Example

```
PARAMETERS: r1 RADIobutton GROUP rad1,  
           r2 RADIobutton GROUP rad1.  
SELECTION-SCREEN ULINE /1(50).  
SELECTION-SCREEN COMMENT /10(30) comm1.  
SELECTION-SCREEN SKIP 2.  
SELECTION-SCREEN ULINE.  
INITIALIZATION.  
  comm1 = 'Comment in Selection'
```



selection-screen skip n - This statement creates a blank line for as many lines for n lines on the selection screen

selection-screen uline -This will place an underline on the screen at a specified location for a specified length.

selection-screen comment -This will place a comment on the selection screen.

Several Elements in a Single Line

SELECTION-SCREEN BEGIN OF LINE.

...

SELECTION-SCREEN END OF LINE.

- Example

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 1(10) text-001.

PARAMETERS: p1(3) TYPE c, p2(5) TYPE c, p3(1) TYPE c.

SELECTION-SCREEN END OF LINE.



selection-screen begin of line and **selection - screen end of line**-All input fields defined between these two statements are placed next to each other on the same line.

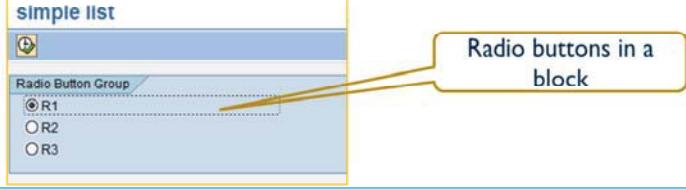
Blocks of Elements

SELECTION-SCREEN BEGIN OF BLOCK block
[WITH FRAME [TITLE title]]
[NO INTERVALS].

...
SELECTION-SCREEN END OF BLOCK block.

– Example

```
SELECTION-SCREEN BEGIN OF BLOCK rad1 WITH FRAME TITLE text-002.  
PARAMETERS r1 RADIOBUTTON GROUP gr1.  
PARAMETERS r2 RADIOBUTTON GROUP gr1.  
PARAMETERS r3 RADIOBUTTON GROUP gr1.  
SELECTION-SCREEN END OF BLOCK rad1.
```



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 22

Selection screen elements can be combined into cohesive units called blocks. These logical blocks are, in essence, a cosmetic screen feature that encapsulates a combination of screen input elements and can be created with a descriptive frame title. Logical blocks help to make the selection options easier to understand and use.

Calling User-Defined Selection Screens

CALL SELECTION-SCREEN numb [STARTING AT x1 y1] [ENDING AT x2 y2].

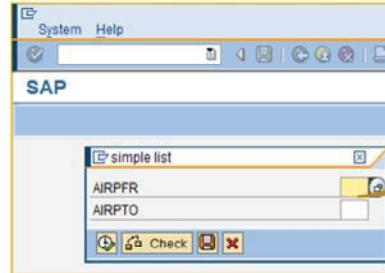
- Example:

```
SELECTION-SCREEN BEGIN OF SCREEN 500 AS WINDOW.
```

```
PARAMETERS: airpfr TYPE spfli-airpfrom,  
airpto TYPE spfli-airpto.
```

```
SELECTION-SCREEN END OF SCREEN 500.
```

```
CALL SELECTION-SCREEN 500 STARTING AT 10 1.
```



Selection Screen Processing

- Selection screens are special screens, defined with the help of ABAP statements
- The ABAP runtime environment completely controls the processing flow of the selection screens
- The ABAP runtime environment generates a number of special selection screen events before the selection screen is displayed and also after the user has executed actions on the selection screen
- Programmers can define event blocks in the program in order to react to these events



Copyright © Capgemini 2015. All Rights Reserved 24

Processing Selection Screen (Contd.).

- AT SELECTION-SCREEN event is
 - the basics of Selection Screen events
 - Occurs after all the input data is passed to the underlying ABAP program from selection screen



Copyright © Capgemini 2015. All Rights Reserved 25

Overview of Selection Screen Events

- Selection Screen Processing
 - Started after the INITIALIZATION event
 - Other events may be triggered for fields or for F4 help, depending upon user action on the selection screen
- The event AT SELECTION SCREEN has to be exited properly for further events to be processed



Copyright © Capgemini 2015. All Rights Reserved 26

Selection Screen - Basic Form

- On passing the input data from Selection Screen to ABAP Program by the runtime environment, AT SELECTION-SCREEN event is triggered
- To modify the Selection Screen elements before display, AT SELECTION-SCREEN OUTPUT event is used



Copyright © Capgemini 2015. All Rights Reserved 27

Demo

- Create a selection screen



Summary

- In this lesson, you have learnt:
 - How to create a selection screen



Review Question

- Question 1: Selection screen elements can be combined into cohesive units called ____.
- Question 2: The default selection screen has the number ____.



ABAP Part I

Lesson 04: DDIC

Lesson Objectives

- After completing this lesson, participants will be able to -
 - Use Data Dictionary to maintain Database Objects
 - Work with
 - Domain
 - Data Elements
 - Tables
 - Structures
 - Views
 - Table Types
 - Search Helps
 - Lock Objects



Copyright © Capgemini 2015. All Rights Reserved 2

Data Dictionary

- Data Definitions are created and Managed in ABAP Dictionary
- Describes the logical structure of objects used in application development
- Describes the mapping of data to the underlying Relational Database in tables and views
- System Independent interface to the Database
- Virtual Database
- Provides data for manipulation and processing
- Transaction Code : SE11



Copyright © Capgemini 2015. All Rights Reserved 3

ABAP Dictionary

- Object Types in ABAP Dictionary are
 - Tables
 - Defined in Dictionary
 - Independent of Database
 - Views
 - Logical Views
 - Types
 - Data elements
 - Structures
 - Table Types
 - Domain
 - Defines a Value Range
 - Search Helps
 - Input Help or F4 Help
 - Lock Objects
 - Lock Mechanism to set and release the locks



Copyright © Capgemini 2015. All Rights Reserved. 4

ABAP Dictionary

■ TABLES

- Has one or More fields
- Contains data in the form of Rows and Columns

■ DATA ELEMENTS

- Field of a table refers to Data Element
- Specifies Non-Technical Attributes

■ DOMAIN

- Specifies Technical Attributes
- Attached to Data Element



Copyright © Capgemini 2015. All Rights Reserved 5

Standard Tables

Table	Description
DD02L	List of All Tables
TSTC	List of All Tcodes
TADIR	R/3 Repository Objects
T000	Clients



Copyright © Capgemini 2015. All Rights Reserved

6

Types of Table

- Table Types
 - Transparent Tables
 - Pooled Tables
 - Cluster Tables



Copyright © Capgemini 2015. All Rights Reserved

7

Tables

■ Transparent Tables

- One-to-one relationship with tables in database
- Most commonly Used
- Holds Application data
- Master data or Transaction data Used by an application
 - Master Data : Vendor Table
 - Transaction data : Orders Placed By Customers



Copyright © Capgemini 2015. All Rights Reserved. 8

Tables

■ TABLE FIELDS

- Field Name
 - Should begin with an Alphabet
- Key Flag
 - Determines the Primary Key
- Field Type
 - Data Type
- Field Length
- Decimal Places
 - Number of Decimal Places
- Short Text – Description of the field



Copyright © Capgemini 2015. All Rights Reserved 9

Tables - Transparent

■ Creation of Tables

■ Top-Down Approach

- Table is first created
- Data element and Domain are created after creation of Table
- Easier to Use

■ Bottom-Up Approach

- Domain and Data element are created
- More intuitive for first timers
- Cumbersome



Copyright © Capgemini 2015. All Rights Reserved 10

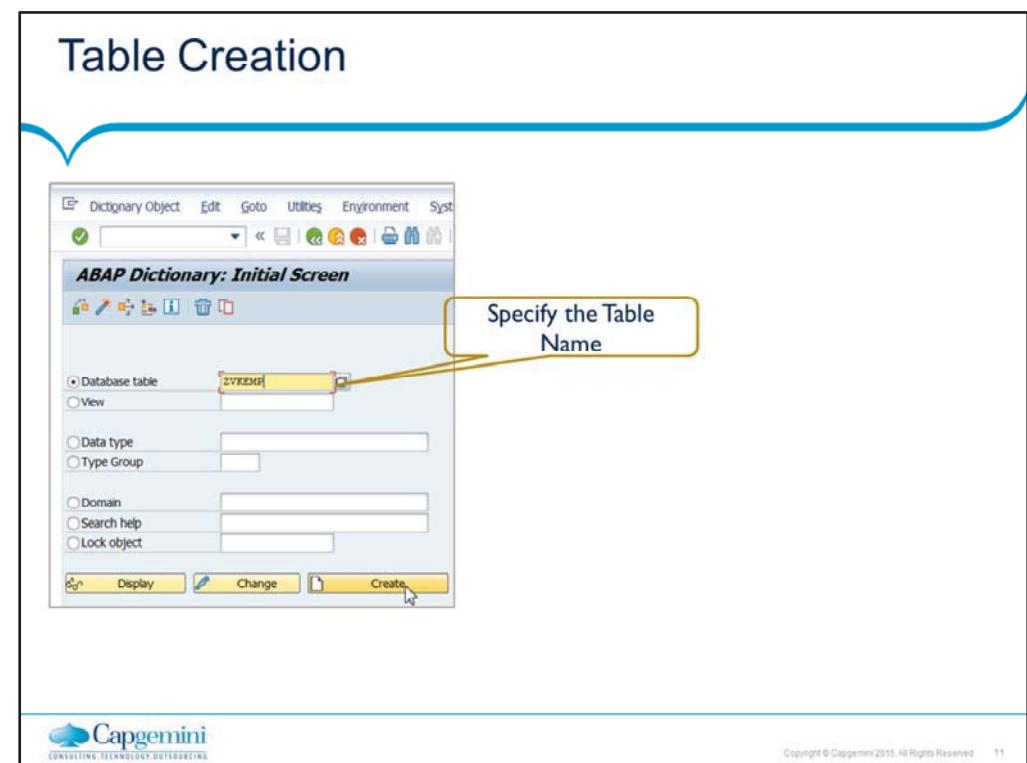
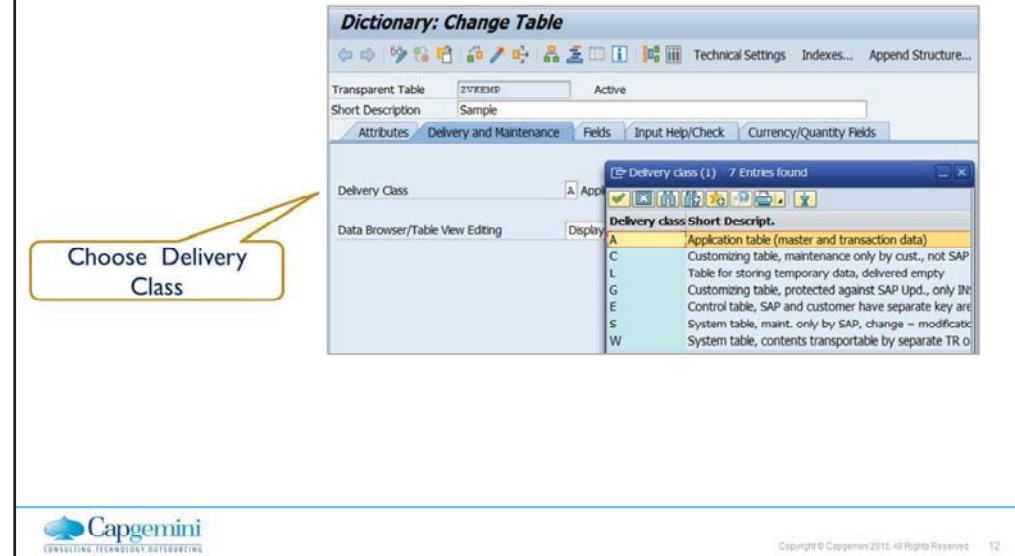


Table Creation (Contd.).



Delivery Classes

- The delivery class controls the transport of table data for installation, upgrade, client copy and when transporting between customer systems.
- There are the following development classes:
 - A: Application table (master and transaction data)
 - C: Customer table, data is only maintained by the customer.
 - L: Table for storing temporary data
 - G: Customer table, SAP may insert new data records but may not overwrite or delete existing ones
 - E: System table with its own namespace for customer entries. The customer namespace must be defined in table TRESC
 - S: System table, data changes have the status of program changes.
 - W: System table (e.g. table of the development environment) whose data is transported with its own transport objects (e.g. R3TR PROG, R3TR TABL, etc.).



Copyright © Capgemini 2015. All Rights Reserved 13

Table Creation - Fields

Primary Key Field

Specify the field names

Data Element

Field	Key Inf...	Data element	Data Type	Length	Deci..	Short Description
MANDT	<input checked="" type="checkbox"/>	MANZT	CLNT	3	0	Client
EMPNO	<input checked="" type="checkbox"/>	ZVKDEEMPNO	NUMC	4	0	This is the short descr f
ENAME	<input type="checkbox"/>	ZVKDEENAME	CHAR	10	0	ZVKDEENAME
JOB	<input type="checkbox"/>	ZVKDEJOB	CHAR	9	0	ZVKDEJOB
MGR	<input type="checkbox"/>	ZVKDEMGR	NUMC	4	0	ZVKDEMGR
SAL	<input type="checkbox"/>	ZVKDESAL	DEC	7	2	ZVKDESAL
COMM	<input type="checkbox"/>	ZVKDECOMM	DEC	5	2	ZVKDECOMM
DEPTNO	<input type="checkbox"/>	ZDEPTNO	NUMC	2	0	DEPTNO

Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 14

Tables Creation (Contd.).

▪ CONSTRAINTS

- Key Fields must be stored at the beginning of the field list
- Non-Key fields may not occur between two key fields
- Maximum of 16 key fields per table is allowed
- Table may not have more than 249 fields



Copyright © Capgemini 2015. All Rights Reserved 15

Tables

- Client Dependent Table
 - First Field has Data Type CLNT
 - Part of PRIMARY KEY Field

- Client Independent Table
 - A table whose First field is not of Data Type CLNT



Copyright © Capgemini 2015. All Rights Reserved 16

Tables (Contd.).

▪ Reference Fields

- Reference Fields required for the following Data Type
 - QUAN
 - CURRE
- Reference Fields should of Type
 - UNIT
 - CUKY
- Reference Fields can be in the same table or another table.



Copyright © Capgemini 2015. All Rights Reserved 17

Demo

- SE11 interface and create a simple table based on pre defined datatypes.



Tables – Technical Settings

Dictionary: Define Technical Settings

Name: EVKEMP | Transparent Table

Short Descrip.: Sample

Last Changed: TRAINER1 | 16.01.2017

Status: Revised | Saved

General Properties | DB-Specific Properties

Logical Storage Parameters

Data Class: APPL0 | Master Data, Transparent Tables

Size Category: 0 | Expected Data Records 0 to 8,500

Buffering

Buffering Not Allowed

Buffering allowed but switched off

Buffering Activated

Buffering Type

Single Records Buffered

Generic Area Buffered

Fully Buffered

Number of Key Fields: 1

Log Data Changes

Writes only with JAVA

Capgemini CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 19

Data Class

- The data class logically defines the physical area of the database where your base database table resided.
- Hence, you should choose the data class correctly, the table will automatically created in the appropriate area on the database when it is activated in the dictionary.
- The most important data classes are master data, transaction data, organizational data and system data
- The data class determines the table space that the table is assigned to.
- A tablespace is a physical file on the disk which is used to hold tables
- Every table is assigned to one tablespace

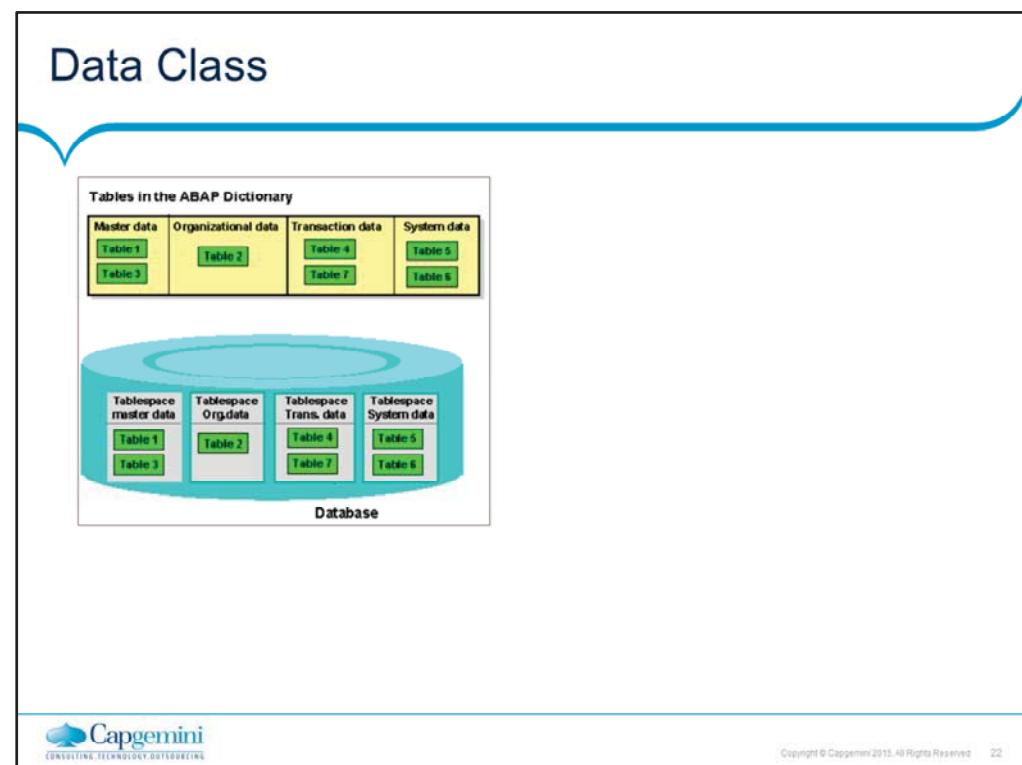


Data Class

- If you choose the data class correctly, your table is automatically assigned to the correct area (tablespace or DBspace) of the database when it is created.
- Each data class corresponds to a physical area in which all the tables assigned to this data class are stored.
- There are the following data classes:
 - APPL0 (Master Data): Data which is seldom changed. An example of master data is the data contained in an address file, such as the name, address and telephone number.
 - APPL1 (transaction data): Data that is frequently changed. An example of transaction data is the goods in a warehouse, which change after each purchase order.
 - APPL2 (organizational data): Customizing data that is defined when the system is installed and seldom changed. An example is the table with country codes.

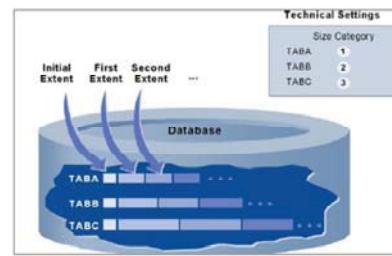


Copyright © Capgemini 2015. All Rights Reserved 21



Size Category

- The size category describes the expected storage requirements for the table on the database.
- An initial extent is reserved when a table is created on the database.
- The size of the initial extent is identical for all size categories.
- If the table needs more space for data at a later time, extents are added.
- These additional extents have a fixed size that is determined by the size category specified in the ABAP Dictionary.
- You can choose a size category from 0 to 4. A fixed extent size, which depends on the database system used, is assigned to each category.



Copyright © Capgemini 2015. All Rights Reserved 23

The size category describes the expected storage requirements for the table on the database.

You can choose a size category from 0 to 4.

You are setting the initial extents and the next extents as well as the maximum number of extents for this table

An extent is the amount of space allocated for a table

An initial extent is the amount of space allocated when the table is created

When you create a table, the system reserves initial space (an initial extent) in the database. If more space is required at a later time due to data entries, additional memory is added depending on the selected size category.

Correctly assigning a size category therefore ensures that you do not create a large number of small extents. It also prevents storage space from being wasted when creating extents that are too large.

Logging

- You can use logging to record and store modifications to the entries of a table.
- To activate logging, the corresponding field must be selected in the technical settings.
- Logging, however, only takes place if the R/3 system was started with the profile containing parameter rec/client.
- Only selecting the flag in the ABAP/4 dictionary is not sufficient to trigger logging.



Copyright © Capgemini 2015. All Rights Reserved 24

The parameter rec/client can have the following settings:

rec/client = ALL All clients should be logged

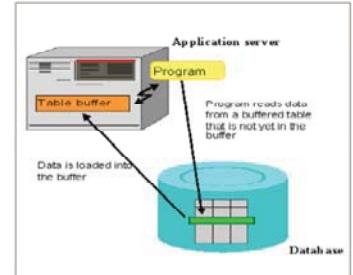
rec/client = 000 ... Only the specified clients should be logged

rec/client = OFF Logging is not enabled on this system

The data modifications are logged independently of the update. You can display the logs with the Transaction Table History (SCU3).

Table - Buffering

- Buffering allows you to access data quicker by letting you access it from the application server instead of the database.



Copyright © Capgemini 2015. All Rights Reserved 25

The R/3 system manages and synchronizes the buffers on the individual application servers.

If an application program accesses data of a table, the database interface determines whether this data lies in the application server.

If this is the case, the data is read directly from the buffer.

If the data is not in the buffer of the application server, it is read directly from the database and loaded into the buffer

The buffer can therefore satisfy the next access to this data.

Table - Buffering

- Buffers reside in each application server
- Improves Performance
- How are Buffers Filled?
 - Program accesses data of a buffered table
 - Database Interfaces checks if the data is available in buffer of Application Server
 - If Available, read from buffer
 - If Unavailable, read from database and load into buffer



Copyright © Capgemini 2015. All Rights Reserved 26

Tables

▪ Buffer Synchronization

- If Program changes data in a table on Application Server, it is noted in log table by Database Interface
- A synchronization Mechanism runs at a fixed time interval
- Log table is read and buffer contents are invalidated
- In next access, data is read from database table and updated in buffer.



Copyright © Capgemini 2015. All Rights Reserved 27

Tables - Buffering

- Buffering Tables

- Table that is frequently read and rarely changed
- The key fields of the buffered table should be of the Character data types (C, N, D, T)
- By pass the buffer if table data should be read from Database



Copyright © Capgemini 2015. All Rights Reserved 28

Table buffering

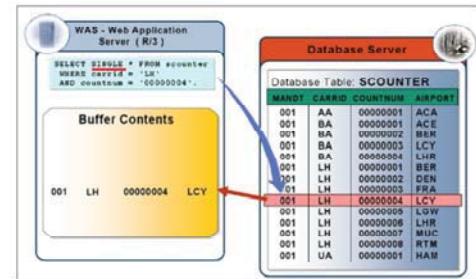
- The buffering type determines which records of the table are loaded into the buffer of the application server when a record of the table is accessed.
- There are the following buffering types:
 - **Full buffering:** When a record of the table is accessed, all the records of the table are loaded into the buffer.
 - **Generic buffering:** When a record of the table is accessed, all the records whose left-justified part of the key is the same are loaded into the buffer.
 - **Single-record buffering:** Only the record that was accessed is loaded into the buffer.



Copyright © Capgemini 2015. All Rights Reserved 29

Single Record Buffering

- Single-record buffering is recommended particularly for large tables in which only a few records are accessed repeatedly with SELECT SINGLE.
- If you access a record that was not yet buffered using SELECT SINGLE, there is a database access to load the record. If the table does not contain a record with the specified key, this record is recorded in the buffer as non-existent. This prevents a further database access if you make another access with the same key
- All the accesses to the table that do not use SELECT SINGLE bypass the buffer and directly access the database.



Full Buffering

- When full buffering, the table is either completely or not at all in the buffer. When a record of the table is accessed, all the records of the table are loaded into the buffer.
- When you decide whether a table should be fully buffered, you must take the table size, the number of read accesses and the number of write accesses into consideration.
- The smaller the table is, the more frequently it is read and the less frequently it is written, the better it is to fully buffer the table.

	WAS - Web Application Server (R/3)	Database Server
Buffer Contents	<pre> 091 AA 00000001 AGA 091 BA 00000001 ACR 091 BA 00000002 BER 091 BA 00000003 CYT 091 BA 00000004 LHR 091 BA 00000005 FRA 091 LH 00000001 BER 091 LH 00000002 DEN 091 LH 00000003 FRA 091 LH 00000004 LCY 091 LH 00000005 LGW 091 LH 00000006 LHR 091 LH 00000007 MUC 091 LH 00000008 RTM 091 UA 00000001 HAM </pre>	<pre> 061 AA 00000001 ACA 061 BA 00000001 ACE 061 BA 00000002 ACR 061 BA 00000003 LCY 061 BA 00000004 LHR 061 LH 00000001 BER 061 LH 00000002 DEN 061 LH 00000003 FRA 061 LH 00000004 LCY 061 LH 00000005 LGW 061 LH 00000006 LHR 061 LH 00000007 MUC 061 LH 00000008 RTM 061 UA 00000001 HAM </pre>
Select * FROM scounter WHERE carrier = 'LH' AND countnum = '00000004'.		

Copyright © Capgemini 2015. All Rights Reserved. 31

Fully buffering is also advisable for tables having frequent accesses to records that do not exist.

Since all the records of the table reside in the buffer, it is already clear in the buffer whether or not a record exists.

The data records stored in the buffer sorted by table key.

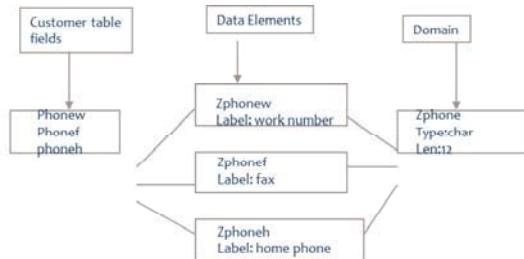
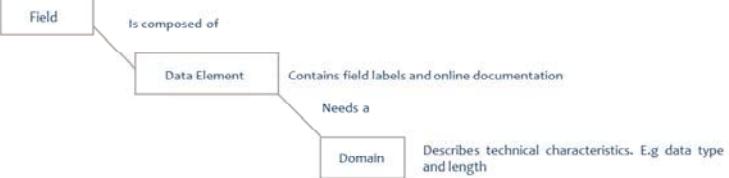
Generic Buffering

- With generic buffering, all records whose generic key fields agree with this record are loaded into the buffer when one record of the table is accessed.
- The generic key is left-justified part of the primary key of the table that must be defined when the buffering type is selected.



Copyright © Capgemini 2015. All Rights Reserved 32

Data elements , Domain and fields

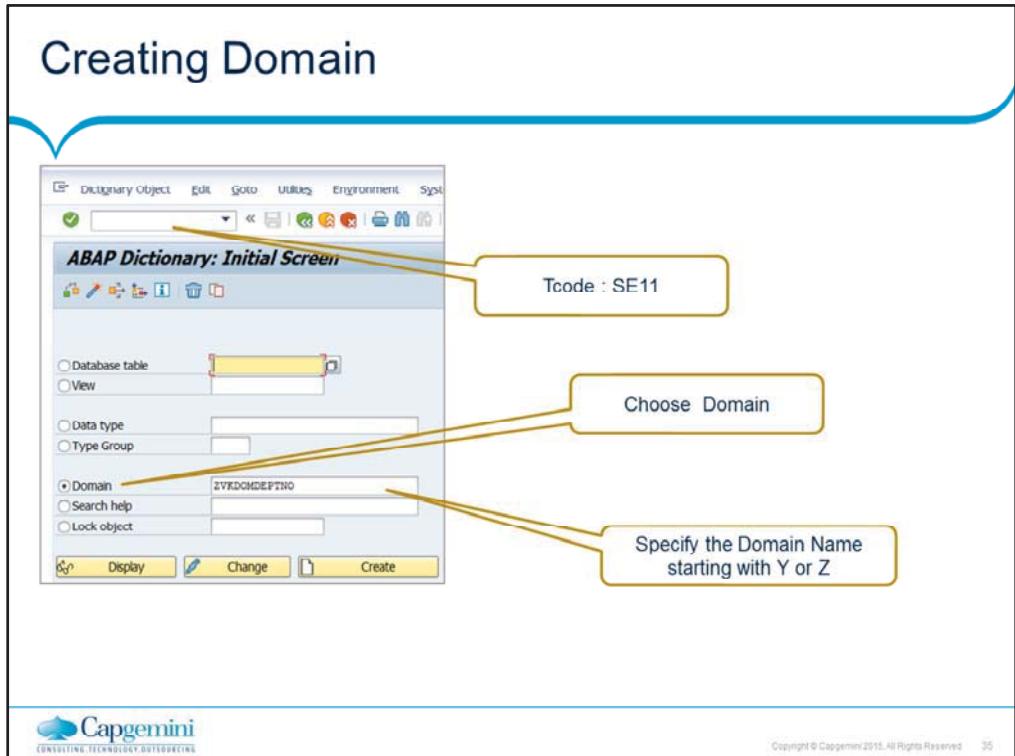


Domain

- Specifies the Technical Characteristics of a Field
 - Data Type
 - Length
- Defines a Value Range
- Can be restricted by defining Fixed Values
- Define Value Table to check against a Table
- Assigned to a Data Element Defines Value Range



Copyright © Capgemini 2015. All Rights Reserved 34



Creating Domain

The screenshot shows the SAP Dictionary: Display Domain screen. The domain name is ZVKDOMDEPTNO, and its short description is "Short Desc for ZVKDOMDEPTNO". The Data Type is set to "NUMERIC TEXT". The "No. Characters" field contains the value "2", and the "Decimal Places" field contains "0". A callout box points to the "Data Type" field with the text "Specify the Datatype". Another callout box points to the "No. Characters" field with the text "Length for the Datatype".

Dictionary: Display Domain

Domain: ZVKDOMDEPTNO Active

Short Description: Short Desc for ZVKDOMDEPTNO

Properties Definition Value Range

Format:

Data Type: NUMERIC TEXT

No. Characters: 2

Decimal Places: 0

Output Characteristics:

Output Length: 2

Routine:

Sign

Case-sensitive

Capgemini CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 36

Creating Domain (Contd.).

- Save the Domain
- Specify the Package
- Activate
- Domain is ready and can be attached to a Data Element

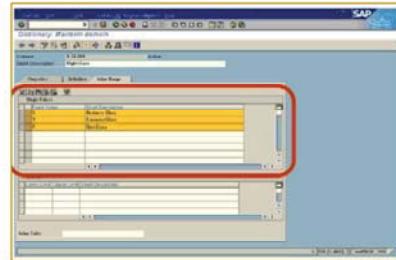


Copyright © Capgemini 2015. All Rights Reserved 37

Domain

■ Value Range and Fixed Values

- Used to restrict the values in the Domain
- Used in input check in screen templates
- If no other help is defined in field, Value Range or Fixed Values are offered in F4 help.
- Value Range or Intervals can be defined by specifying the upper and lower limits
- Although S_CLASS is a domain of the type C, it would accept no other character besides C/Y/F.



Copyright © Capgemini 2015. All Rights Reserved 36

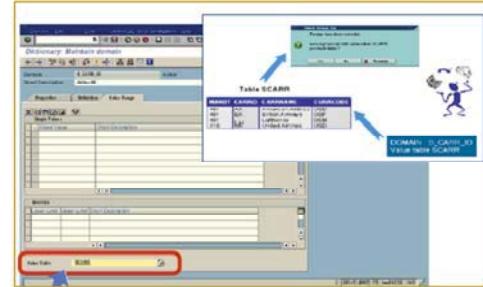
The domain describes the value range of a field by specifying its data type and field length. If only a limited set of values is allowed, they can be defined as fixed values. Specifying fixed values causes the value range of the domain to be restricted by these values. Fixed values are immediately used as check values in screen entries. There is also an F4 help.

Fixed values can either be listed individually or defined as an interval.

Fixed values are only checked in screens. No check is made when data records are inserted in a table by an ABAP program.

Value Table

- Maintained at Domain Level
- A check is not implemented by simply entering a value table.
- The check against the value table only takes effect when a foreign key* has been defined.



Copyright © Capgemini 2015. All Rights Reserved 39

The value range of a field can also be defined by specifying a value table in the domain.

In contrast to fixed values, however, simply specifying a value table does not cause the input to be checked. There is no F4 help either.

If you enter a value table, the system can make a proposal for the foreign key definition.

A value table only becomes a check table when a foreign key is defined.

If you refer to a domain with a value table in a field, but no foreign key was defined at field level, there is no check

Domain

- Input Check valid for few data types
 - Value Range
 - CHAR
 - NUMC
 - Fixed Values
 - CHAR
 - NUMC
 - DEC
 - INT1
 - INT2
 - INT4



Copyright © Capgemini 2015. All Rights Reserved 40

Data Elements

- Specifies the Semantic Characteristics of a Field
- Describes an Elementary type or a Reference Type
 - Elementary type
 - Defined by built-in data type and length
 - OR
 - Defined directly or specified through a Domain
 - Reference Types
 - Defines the type of Reference Variable to a Class or an Interface



Copyright © Capgemini 2015. All Rights Reserved 41

Data Elements

- Field Label

- Field Labels are used to display a screen field

- F1 Documentation

- The text appearing in the Field Help (F1 Help) comes from the documentation
 - If there is no Documentation, the short text appears



Copyright © Capgemini 2015. All Rights Reserved 42

Creating Data Element

■ Go to Tcode : SE11

The screenshot shows the ABAP Dictionary: Initial Screen. A callout box labeled "Choose Data Type" points to the "Data type" radio button, which is selected. Another callout box labeled "Specify the Data Element Name" points to the text input field containing "ZVKDEDEPTNO1". A third callout box labeled "Click on CREATE" points to the "Create" button at the bottom of the screen.

Choose Data Type

Specify the Data Element Name

Click on CREATE

ABAP Dictionary: Initial Screen

ZVKDEDEPTNO1

Data type

Create

Copyright © Capgemini 2015. All Rights Reserved 43

Creating Data Element

Dictionary: Change Data Element

Data element **ZVKDEDEPTNO** Active
Short Description **Short Descr for ZVKDEDEPTNO**

Attributes Data Type Further Characteristics Field Label

Elementary Type
 Domain **ZVKDOMDEPTNO** Short Desc for ZVKDOMDEPTNO
Data Type **NUMC** Numerical Text
Length **2**

Predefined Type Data Type Length **0**

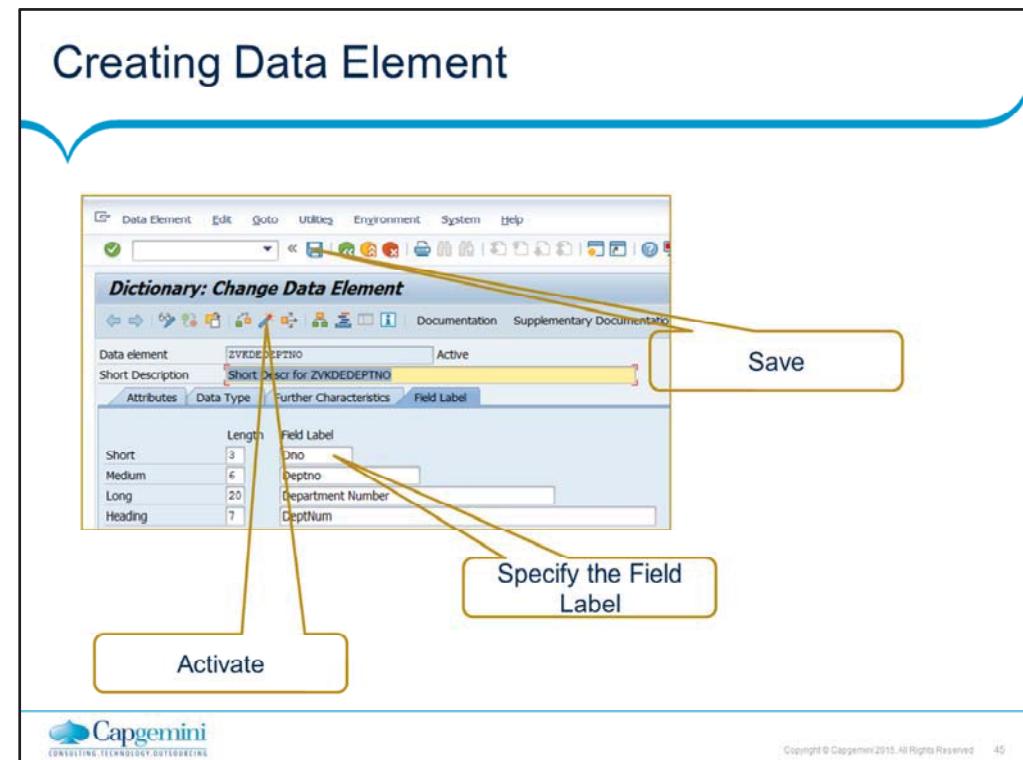
Reference Type Referenced Type

Reference to Predefined Type Data Type Length **0**

Specify the Domain Name

 Capgemini CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 44



Demo

- Create a simple table based data elements and domain



Copyright © Capgemini 2015. All Rights Reserved 46

Tables (Contd.).

▪ Pooled Tables

- Many-to-One Relationship with the table in Database
- SAP Proprietary Construct
- Stored in a Table Pool
 - Table Pools Hold large number of small Tables
- When activated, a single table is created in database
- Define Pooled tables within R/3 and assign them to the table pool



Copyright © Capgemini 2015. All Rights Reserved 47

Tables (Contd.).

- Holds CUSTOMIZING data
 - Codes, Field Validations, number ranges, parameters
 - Country Code table, Exchange Rate Table, etc...
- Data in Customizing Table is set by Functional Consultant during the initial Implementation



Copyright © Capgemini 2015. All Rights Reserved 48

Pooled Tables

- Definition of table pool contains 2 Key fields
 - Tabname
 - Varkey (Contains Entry from all key fields)

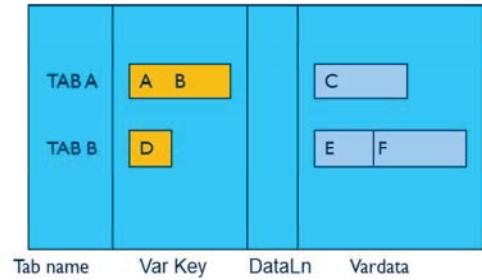
Pooled Table TAB A

Key	Data
A	B
C	

Pooled Table TAB B

Key	Data
D	E
F	

Table Pool in Database



Cluster Table

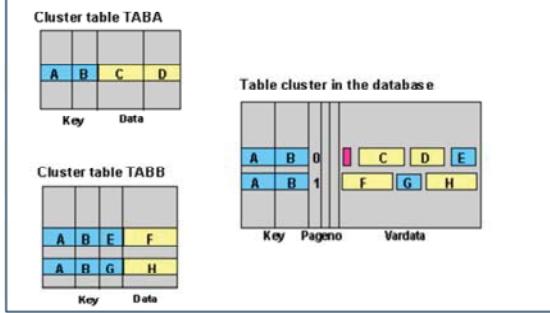
- Many-to-One relationship with table in database
- Cluster tables are stored in Table Cluster in Database
- SAP Proprietary
- Used when the tables have a part of Primary Key in common
- Data accessed Simultaneously
- Contain Fewer tables than Pool tables
- In a single I/O, all the related rows in a cluster table are retrieved
- Reduces the no. of Database reads and improves Performance



Copyright © Capgemini 2015. All Rights Reserved 50

Cluster Table

- The records of all cluster tables with the same key are stored under one key in the assigned table cluster. The values of the key fields are stored in the corresponding key fields of the table cluster.



Demo

- Create Cluster table and Pool table



Pooled and Clustered table

- Restrictions

- Secondary Indexes cannot be created
- Cannot Use ABAP/4 Constructs
 - Select DISTINCT
 - GROUP BY
- Cannot Use Native SQL
- Cannot Specify Field names in ORDER BY except for Primary Key



Copyright © Capgemini 2015. All Rights Reserved 53

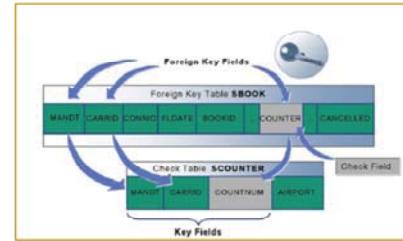
Tables

- Foreign Keys

- Defines Relationship between Tables
 - CHECK Table
 - FOREIGN Key Table

- Create Value Checks for input fields

- Link several tables in a view or a Lock Object



Copyright © Capgemini 2015. All Rights Reserved 54

A foreign key creates a link between two tables T1 and T2. Every primary key field from T2 (check table) is assigned a field from table T1 (foreign key field). The fields from T1 assigned to primary key fields are marked as foreign key fields.

The most important function of the foreign key is the support of data integrity. The foreign key fields can only accept values which appear in the primary key of the check table. During input the values of the foreign key fields can thus be checked against the entries of the assigned key fields of the check table.

Foreign keys are used to ensure that the data is consistent. Data that has been entered is checked against existing data to ensure that it is consistent

Tables – Foreign Keys

FOREIGN Key Table T1			
Field 1	Field 2	Field 3	Field 4
Primary Key			

CHECK Table T2		
Field 5	Field 6	Field 7

Primary
Key

Tables – Foreign Keys (Contd.).

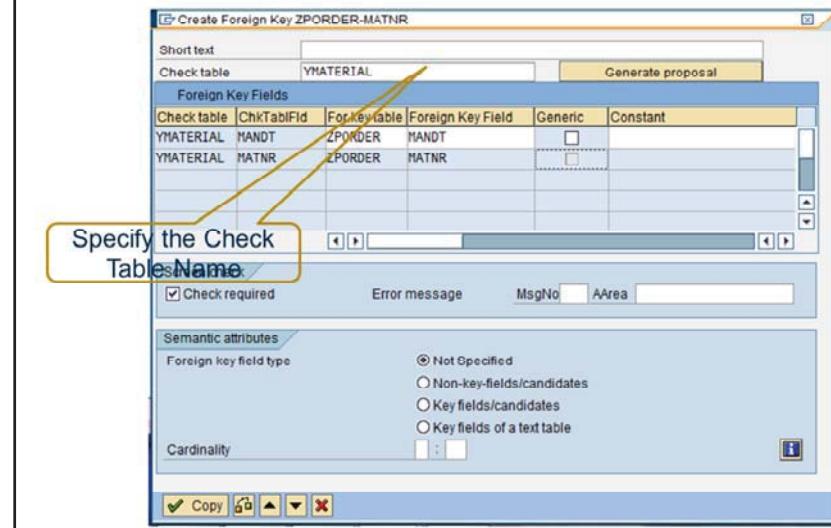
Creating Foreign Keys

The screenshot shows the SAP Dictionary: Maintain Table interface for table ZPORDER. The table is described as 'purchase order details'. The 'Foreign Key' tab is selected. A yellow box highlights the 'Key' column, which contains checkboxes for fields MANDT, EBELN, and MATNR. A yellow arrow points from the 'Key' column to the 'Foreign Keys' column, which lists the target fields MANDT, EBELN, and MATNR respectively. The MATNR row is highlighted with a yellow background.

Field	Key	Init	To element	Foreign Keys	length	Decim	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	MANDT	3	0	Client
EBELN	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EBELN	CHAR	10	0	Purchasing Document Number
MATNR	<input type="checkbox"/>	<input type="checkbox"/>	MATNR	CHAR	18	0	Material Number
QUANTITY	<input type="checkbox"/>	<input type="checkbox"/>	XTMNG	QUAN	13	3	Target Quantity
UOM	<input type="checkbox"/>	<input type="checkbox"/>	MEINS	UNIT	3	0	Base Unit of Measure

Copyright © Capgemini 2015. All Rights Reserved 56

Tables – Foreign Key (Contd.).



Demo

- Create primary key , foreign key relationship



Standard Tables

- Few Frequently Used Tables

Table Name	Description
MARA	Material Master
KNA1	Customer Master
LFA1	Vendor Master
VBAK	Sales Document : Header Data
VBAP	Sales Document : Item Data
EKKO	Purchase Document : Header
EKPO	Purchase Document : Item



Copyright © Capgemini 2015. All Rights Reserved 59

Indexes

- Copy of Database Table reduced to certain fields
- Always in sorted form
- Provides faster access to data records
- Contains a pointer to corresponding record of actual table
- Primary Index contains the key fields of the table
- Primary index created automatically when table is activated
- Possible to create secondary indexes



Copyright © Capgemini 2015. All Rights Reserved 60

Primary index: The primary index contains the key fields of the table and a pointer to the non-key fields of the table. The primary index is created automatically when the table is created in the database.

Secondary index: Additional indexes could be created considering the most frequently accessed dimensions of the table.

Tables - Index

- Secondary Index

- Created if the table is frequently accessed using fields which is not a part of primary key
- Index distinguished with a three place identifier
- For certain database systems, the index improves performance
- Unique Index
 - Index field has key function

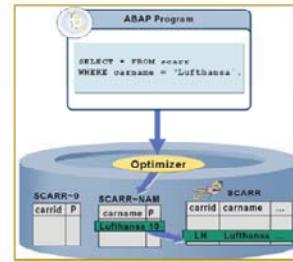


Copyright © Capgemini 2015. All Rights Reserved 61

Tables - Index

▪ Secondary Indexes

- Only Few indexes should be there in tables where the entries are frequently changed
- The database system does not use suitable indexes for selection even if there is one
- The index used depends on the optimizer used for the database system
- Creating an additional index might have side effects on performance



Tables - Index

- Creating Secondary Index

The screenshot shows the SAP Dictionary: Change Table interface. At the top, there's a toolbar with various icons and a menu bar. Below the toolbar, the table details are shown: Transparent Table ZVKEMP, Short Description Sample, and Status Inactive. There are tabs for Attributes, Delivery and Maintenance, Fields, Input Help/Check, and Currency/Quantity Fields. A sub-dialog titled 'Create Index' is open, showing 'Table Name' ZVKEMP and 'Index Name' ZER. A yellow callout box with the text 'Specify the Index Name' points to the 'Index Name' input field.

Dictionary: Change Table

Transparent Table ZVKEMP Inactive

Short Description Sample

Attributes Delivery and Maintenance Fields Input Help/Check Currency/Quantity Fields

Indices for Table ZVKEMP

Ind Ext_ Short text Status Unique Last

Create Index

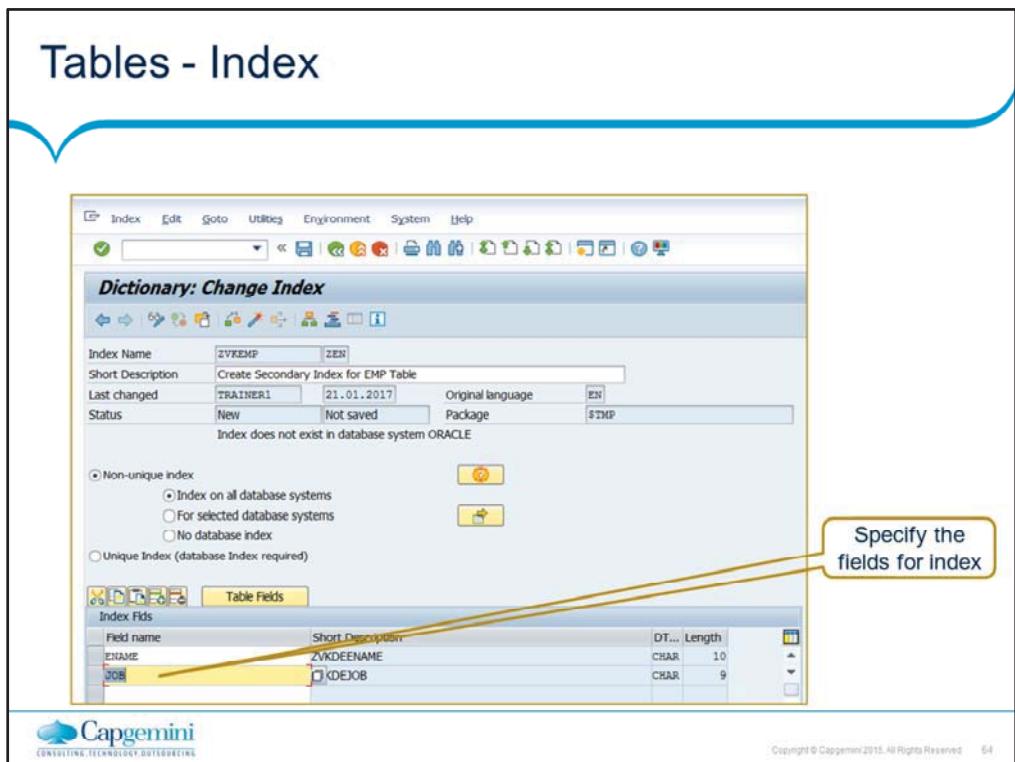
Table Name ZVKEMP

Index Name ZER

Specify the Index Name

Capgemini CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 63



Demo

- Create Secondary Index



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 65

Tables - Index

- The database optimizer decides which index on the table should be used by the database to access data records.
- You must distinguish between the primary index and secondary indexes of a table.
- The primary index contains the key fields of the table.
- The primary index is automatically created in the database when the table is activated.
- If a large table is frequently accessed such that it is not possible to apply primary index sorting, you should create secondary indexes for the table.



Copyright © Capgemini 2015. All Rights Reserved 66

Structures

- Structures
 - Contains Fields
 - User Defined Data Type
 - Fields can refer to
 - An elementary data type
 - Another structure
 - Table type
 - Types :
 - Flat Structures
 - Nested Structures
 - Deep Structures



Copyright © Capgemini 2015. All Rights Reserved 67

Structures (Contd.).

- Flat Structure

- In database Tables only Flat Structures can be included

Field A	Field B
---------	---------

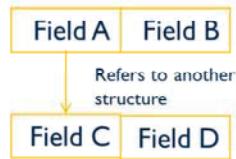
The image cannot currently be displayed.



Copyright © Capgemini 2015. All Rights Reserved 68

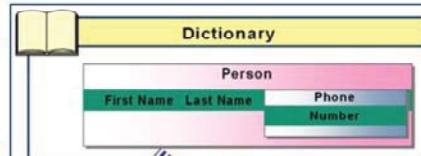
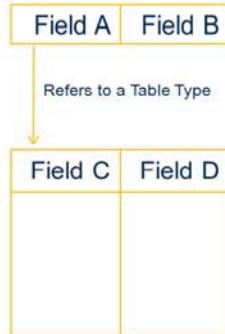
Structures (Contd.).

▪ Nested Structure



Structures (Contd.).

■ Deep Structure



ABAP Program

```
DATA wa_pers TYPE person.  
DATA wa_tel LIKE LINE OF wa_pers-telephone.  
  
wa_pers-name-firstname = 'Walter'.  
wa_pers-name-lastname = 'Schweizer'.  
wa_tel-number = '+49-6227-727272'.  
INSERT wa_tel INTO TABLE wa_pers-telephone.  
wa_tel-number = '+49-6227-742888'.  
INSERT wa_tel INTO TABLE wa_pers-telephone.  
  
READ TABLE wa_pers-telephone INDEX 1 INTO wa_tel.
```



Structures

- Create a Structure

The screenshot shows the ABAP Dictionary: Initial Screen. A yellow box highlights the 'Data type' field, which contains the value 'ZPLT_STK'. A callout bubble points to this field with the text 'Specify the Structure Name'. Another callout bubble points to the 'Create' button at the bottom right of the screen with the text 'Click on CREATE'.

ABAP Dictionary: Initial Screen

Specify the Structure Name

Click on CREATE

Capgemini

Copyright © Capgemini 2015. All Rights Reserved 71

Structures

Choose Structure

Specify the fields of a Structure

The screenshot shows the SAP ABAP Dictionary interface. At the top, a modal window titled "Create Type ZFLT_STR1" has a radio button selected for "Structure". Below it, the main dictionary screen displays a structure named "ZFLT_STR1" which is active. The structure has a short description of "Flat Structure". The "Components" tab is selected, showing three components: MATNR, MBRSH, and MTART. Each component is defined with a data type of CHAR, a length of 18, 1, or 4 respectively, and a short description of "Material number", "Industry sector", and "Material type".

Component	Type	Component-type	Data-type	Length	Dec.	Short Description
MATNR	Types	MATNR	CHAR	18	0	Material number
MBRSH	Types	MBRSH	CHAR	1	0	Industry sector
MTART	Types	MTART	CHAR	4	0	Material type

Copyright © Capgemini 2015. All Rights Reserved 72

Demo

- Create a Flat, Nested and Deep Structure



Copyright © Capgemini 2015. All Rights Reserved 73

Structures (Contd.).

- Maintain Foreign Keys if required
- Save
- Activate the Structure

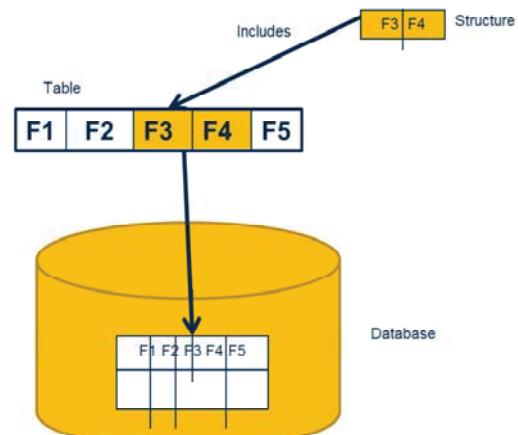


Copyright © Capgemini 2015. All Rights Reserved 74

Structures (Contd.).

- Includes

- To add fields of another structure in Tables or Structures



Structures (Contd.).

- Only Flat Structures can be included
- A Structure can be included more than once
- The field name of the structure should not be longer than 16 places



Copyright © Capgemini 2015. All Rights Reserved 76

Table Types

- Defines the structure of an Internal Table in ABAP
- Commonly used in ABAP Programs



Modifying Standard Tables

- Add fields to Standard Tables Using
 - Append Structures
 - Structure included in Single Table
 - Customizing includes
 - Structure can be included in multiple Tables
 - Is already integrated into SAP tables by SAP
 - The customer fills it with the desired additional fields



Copyright © Capgemini 2015. All Rights Reserved 78

Append Structures

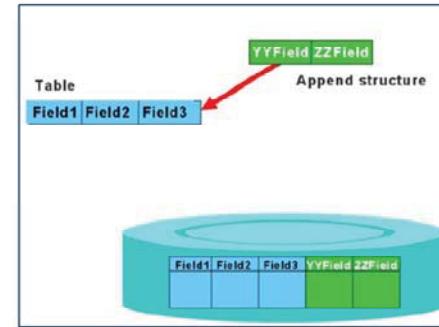
- Append Structures
 - Used for Table Enhancement
 - To Insert New Fields into Tables
 - Structure that is assigned to exactly one table
 - Created in customer namespace
 - Field Names begin with YY or ZZ
 - Created in Customer Namespace
 - Customers can create an append structure for an SAP table (without SAP preparation)
 - Multiple append structures can be used with a single SAP table



Copyright © Capgemini 2015. All Rights Reserved 79

Append Structures

- Append structures allow you to attach fields to a table without the need to modify the table itself.
- If you copy a table that has an append structure attached to it, the fields in the append structure become normal fields in the target table



the following points must be considered when using append structures:

You cannot create append structures for pool and cluster tables.

If a long field (data type **LCHR** or **LRAW**) occurs in a table, it cannot be extended with append structures. This is because long fields must always be in the last position of the field list, that is, they must be the last field of the table. No fields from an append structure may be added after them.

If you use an append structure to expand an SAP table, the field names in your append structure should be in the customer namespace, that is, they must begin with either **YY** or **ZZ**. This prevents name collisions with new fields inserted in the standard table by SAP.

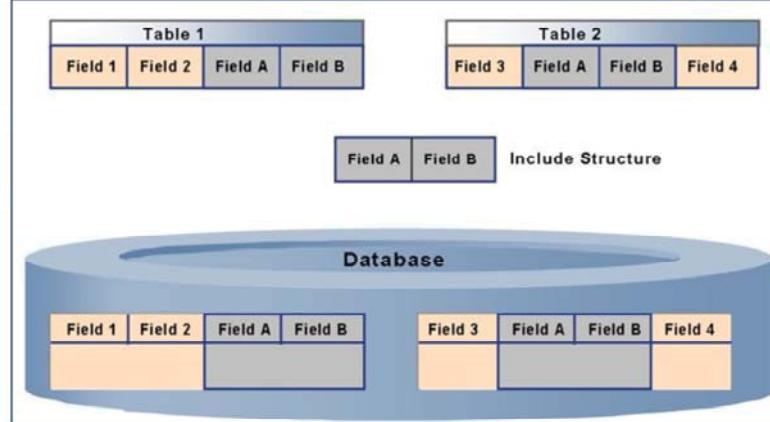
Demo

- Create an Append structure on a table created



Customizing Includes

- Structure Satisfying a Special Naming Convention CI_
- Can be included in Several Tables



Customizing Includes

- Some of the tables and structures delivered with the R/3 standard contain special include statements: These are known as **Customizing includes**
- Customizing includes are created by SAP, but the customer supply the fields for the include.
- Customizing includes begin with CI_ and is part of the customer namespace
- One Customizing include can be inserted into more than one table.



Copyright © Capgemini 2015. All Rights Reserved 53

Customizing Includes

- Consider the table RKPF which uses the Customizing include. A field can be added to CI_COBL. The field becomes a part of table RKPF after the include is activated.

The screenshot shows the SAP Dictionary: Display Table for the transparent table RKPF. The table lists various fields with their data types, lengths, and short descriptions. A specific field, 'CI_COBL', is highlighted in the 'Data element' column. The table has columns for Field, Key, Init..., Data element, Data Type, Length, Deci..., and Short Description. Fields listed include VZNR, JV_PART, PIPOS, C1_COBL, RECID, JV_RECINDI, PFER, DABRZ, FISTL, GEBER, BP_GEBER, PRZNR, CO_PRZNR, LSTAR, GM_GRANT_NBR, and FM_BUDGET_PERIOD.

Field	Key	Init...	Data element	Data Type	Length	Deci...	Short Description
VZNR			JV_PART	CHAR	10	0	Partner account number
PIPOS			PIPOS	CHAR	14	0	Commitment Item
.INCLUDE			C1_COBL	STRU	0	0	
RECID			JV_RECINDI	CHAR	2	0	Internal Recovery Indicator
PFER			PFER	CHAR	16	0	Functional Area
DABRZ			DABRZ	DAT'S	8	0	Reference date for settlement
FISTL			FISTL	CHAR	16	0	Funds Center
GEBER			BP_GEBER	CHAR	10	0	Fund
PRZNR			CO_PRZNR	CHAR	12	0	Business Process
LSTAR			LSTAR	CHAR	6	0	Activity Type
GRANT_NBR			GM_GRANT_NBR	CHAR	20	0	Grant
BUDGET_PD			FM_BUDGET_PERIOD	CHAR	10	0	FM: Budget Period



Copyright © Capgemini 2015. All Rights Reserved 54

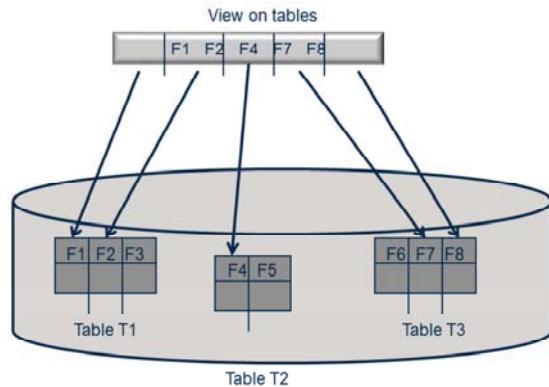
Demo

- Create an Include structure on a table created



Views

- View is data derived from one or more tables
- Used in ABAP Program for data selection
- Data is not stored physically



Views

- Structure of a View - Starting Situation

SCARR (T1)	
CARRID	CARNAME
LH	Lufthansa
DL	Delta Airline

SFLIGHT (T2)		
CARRID	CONNID	PLANETYPE
LH	401	A319
LH	402	DC-10-10
DL	106	A310-300
DL	1699	A319

Cross-product
of tables SCARR
and SFLIGHT

Resulting sets (using Join without ON condition)

T1~CARRID	T1~CARNAME	T2~CARRID	T2~CONNID	T2~PLANETYPE
LH	Lufthansa	LH	401	A319
LH	Lufthansa	LH	402	DC-10-10
LH	Lufthansa	DL	106	A310-300
LH	Lufthansa	DL	1699	A319
DL	Delta Airline	LH	401	A319
DL	Delta Airline	LH	402	DC-10-10
DL	Delta Airline	DL	106	A310-300
DL	Delta Airline	DL	1699	A319

Structure of a View - Join Condition

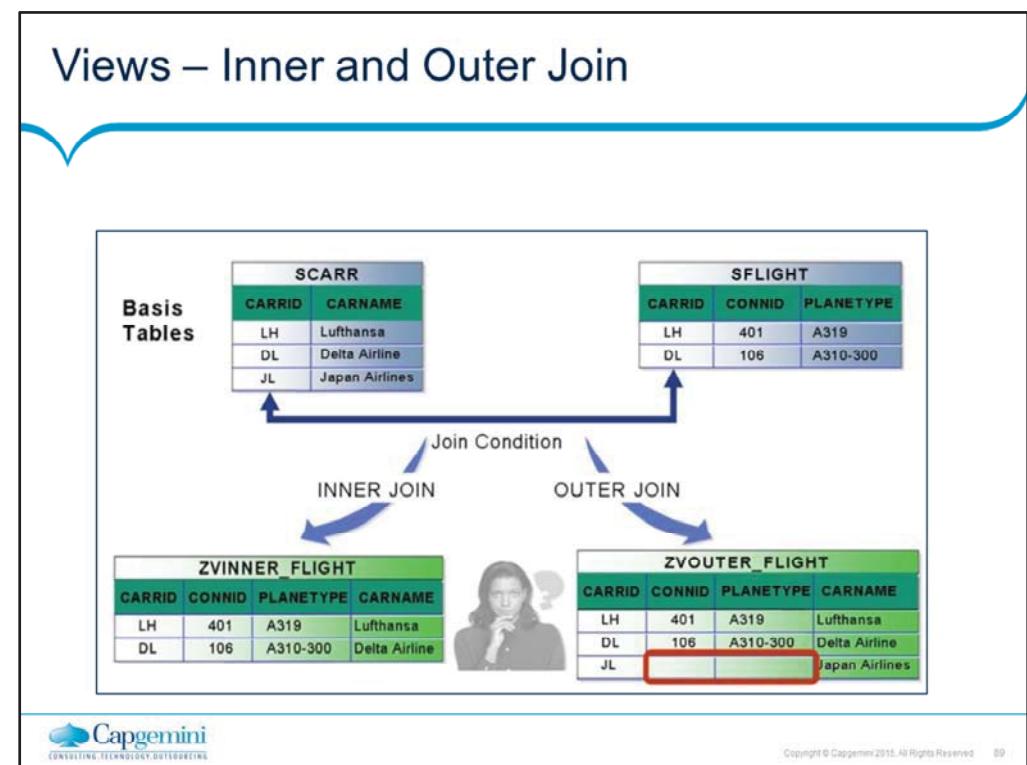
Resulting sets (using Join with ON condition)

T1~CARRID	T1~CARNAME	T2~CARRID	T2~CONNID	T2~PLANETYPE
LH	Lufthansa	LH	401	A319
LH	Lufthansa	LH	402	DC-10-10
LH	Lufthansa	DL	106	A310-300
LH	Lufthansa	DL	1699	A319
DL	Delta Airline	LH	401	A310
DL	Delta Airline	LH	402	DC-10-10
DL	Delta Airline	DL	106	A310-300
DL	Delta Airline	DL	1699	A319

Reduce
the cross-
product



Copyright © Capgemini 2015. All Rights Reserved 58



Types of Views

- Database View
- Projection View
- Maintenance View
- Help View



Copyright © Capgemini 2015. All Rights Reserved 90

Database View

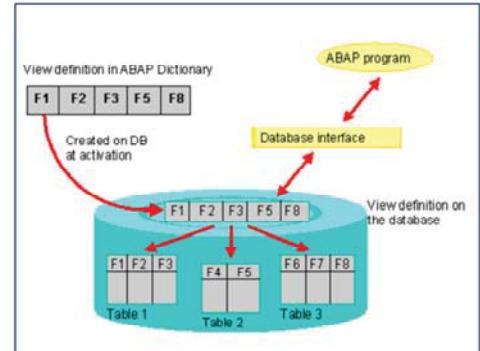
- Provides application specific view on data distributed in tables
- Created in the database
- accessed using Open SQL and Native SQL
- If there is only one table in the view , change access is possible
- Contains only transparent table
- Implements Inner Join



Copyright © Capgemini 2015. All Rights Reserved 91

Database View

- Data about an application object is often distributed on several database tables.
- A database view provides an application-specific view on such distributed data.
- Database views are defined in the ABAP Dictionary.
- A database view is automatically created in the underlying database when it is activated.



Copyright © Capgemini 2015. All Rights Reserved 92

Application programs can access the data of a database view using the database interface. You can access the data in ABAP programs with both OPEN SQL and NATIVE SQL. However, the data is actually selected in the database. Database views implement an **inner join**. If the database view only contains a single table, the maintenance status can be used to determine if data records can also be inserted with the view. If the database view contains more than one table, you can only read the data. A database view may only contain transparent tables

Demo

- Create a Database View

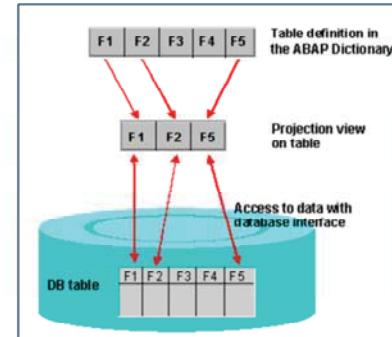


 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 93

Projection View

- Used to hide fields of a table
- Contains exactly one table
- Selection conditions cannot be defined



There is no corresponding object in the database for a projection view. It is also possible to access pooled tables or cluster tables with a projection view.
The maintenance status of the view controls how the data of the table can be accessed with the projection view

Projection View

- Structure of a View - Field Selection (Projection)

The diagram illustrates the projection of fields from a base table to a view. It shows two tables: **SFLIGHT** and **ZVP_SFLIGHT (View)**. The **SFLIGHT** table has columns CARRID, CONNID, PRICE, CURRENCY, and PLANETYPE. The **ZVP_SFLIGHT (View)** table has columns CARRID, CONNID, PRICE, and CURRENCY. Arrows point from the first four columns of the **SFLIGHT** table to the corresponding columns in the **ZVP_SFLIGHT** view. A red circle labeled "STOP" is positioned next to the last column of the **SFLIGHT** table, indicating that no further fields are selected. To the right, the text "Projection (Restriction of existing table fields)" is written.

SFLIGHT				
CARRID	CONNID	PRICE	CURRENCY	PLANETYPE
LH	401	666,00	EUR	A319
LH	401	666,00	EUR	DC-10-10
DL	106	611,01	USD	A319
DL	1699	422,94	USD	DC-10-10

ZVP_SFLIGHT (View)			
CARRID	CONNID	PRICE	CURRENCY
LH	401	666,00	EUR
LH	401	666,00	EUR
DL	106	611,01	USD
DL	1699	422,94	USD

Projection (Restriction of existing table fields)

Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 95

Demo

- Create a projection view

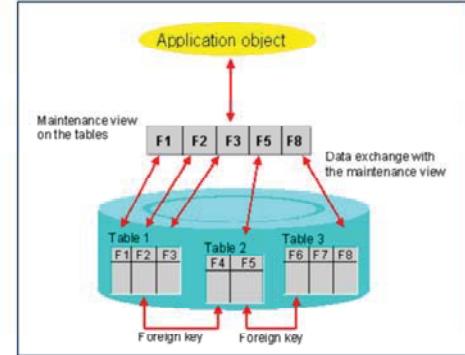


 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 96

Maintenance View

- Maintenance views offer easy ways to maintain complex application objects.
- Data distributed on several tables often forms a logical unit, for example an application object, for the user.
- You want to be able to display, modify and create the data of such an application object together



Copyright © Capgemini 2015. All Rights Reserved 97

A maintenance view permits you to maintain the data of an application object together. The data is automatically distributed in the underlying database tables. The maintenance status determines which accesses to the data of the underlying tables are possible with the maintenance view. All the tables in a maintenance view must be linked with foreign keys.

Demo

- Create a Maintenance view



Copyright © Capgemini 2015. All Rights Reserved 98

Help view

- You have to create a help view if a view with outer join is needed as selection method of a search help.
- The selection method of a search help is either a table or a view.
- If you have to select data from several tables for the search help, you should generally use a database view as selection method.
- However, a database view always implements an inner join.
- If you need a view with outer join for the data selection, you have to use a help view as selection method.
- A help view implements an outer join, i.e. all the contents of the primary table of the help view are always displayed.



Copyright © Capgemini 2015. All Rights Reserved 99

Demo

- Create a Help view



Copyright © Capgemini 2015. All Rights Reserved 100

About Search Help

- Used to Display list of all possible input values for a screen field on the press of F4
- Useful when the field requires the input of a formal key
- Has to be assigned to the screen field
- Types of Search Helps
 - Elementary
 - Describes a search path
 - Defines where the data of the hitlist should be read from
 - Collective
 - Combines several elementary search helps
 - Offers Alternative search paths



Copyright © Capgemini 2015. All Rights Reserved 101

Selection Method

- Possible input values are determined at runtime by database selection
- If the values are from a single table, the corresponding table is selected as selection method
- If the values are from multiple tables, they must be linked with a view (Database or Help View) which is selected in the Selection Method



Copyright © Capgemini 2015. All Rights Reserved 102

Search Help Parameters

- Defines the fields of selection method that should be used in input help
- Data element should be assigned to Search Help Parameter
- Import and Export Parameters
 - Import Parameter – Information from the screen is copied to help process
 - Export Parameter – Values from hitlist is returned to input template



Copyright © Capgemini 2015. All Rights Reserved 103

Attaching Search Help to Screen Fields

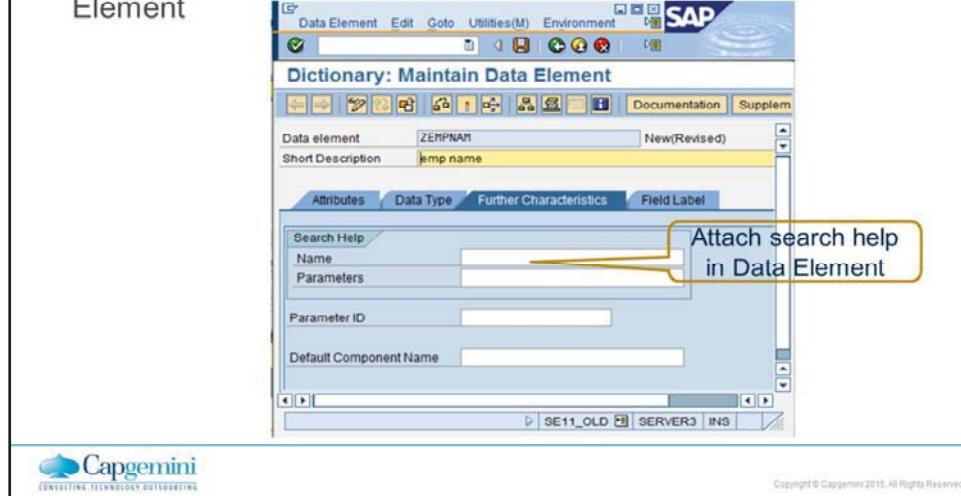
- Search helps can be attached to fields by the following ways
 - Attaching to Data Element
 - Attaching to Check Tables
 - Attaching to a table field or Structure Field
 - Attaching to Screen Fields



Copyright © Capgemini 2015. All Rights Reserved 104

Attaching to Data Element

- The search help is used by all screens that refer to this data element
- Export parameter of the search help must be assigned to Data Element



Copyright © Capgemini 2015. All Rights Reserved 105

Demo

- Create an Elementary search help and Collective search help



Attaching to Screen Field

- A search help can be directly assigned to Screen field by
 - Specifying name of search help in Screen Painter Attributes
 - Specifying the name of Search help in ABAP reports in PARAMETERS or SELECT-OPTIONS statement using AS SEARCH PATTERN.



Copyright © Capgemini 2015. All Rights Reserved 107

Creating Search Helps (Contd.).

- Specify
 - Import
 - Export
 - LPos
 - Spos
- Save and Activate the search help



Copyright © Capgemini 2015. All Rights Reserved 106

What are Lock Objects?

- Lock Mechanism is used by R/3 to synchronize simultaneous access to same data by several users
- Locks are set and released by calling Function Modules, which are automatically generated from the definition of lock objects
- Lock Arguments
 - Consists of Key Fields of the Tables
 - Used as input parameter in the function module for setting and releasing locks



Copyright © Capgemini 2015. All Rights Reserved 109

Lock Mode

- A lock mode can be assigned to each table in the lock object
- Lock mode
 - Defines how the users can access the locked records of the table
 - Write Lock
 - Locked Data can be displayed or edited by a single user.
 - Can be requested several times from the same transaction and are processed successively
 - A request for another Exclusive lock or Shared Lock is rejected
 - Read Lock
 - More than one user can access the locked data at the same time in display mode
 - A request for another shared lock is accepted
 - Exclusive But not Cumulative
 - Can be called only once from the transaction
 - All other lock requests are rejected



Copyright © Capgemini 2015. All Rights Reserved 110

Function Modules for Lock Requests

- Activating a lock object automatically creates function modules
 - ENQUEUE_<lock Object Name> - for setting the lock
 - DEQUEUE_<lock Object Name> - for releasing the lock
- The above function modules can be called directly from SE37 or an ABAP program created in SE38
- The TCode SM12 can be used to check whether a lock has been applied
- This can be done after executing the Enqueue function



Copyright © Capgemini 2015. All Rights Reserved 111

Summary

■ In this lesson, you have learnt:

- To Use Data Dictionary to maintain Database Objects
- To Work with
 - Domain
 - Data Elements
 - Tables
 - Structures
 - Views
 - Table Types
 - Search Helps
 - Lock Objects



Review Question

- Question 1: A _____ in the dictionary has a one to one relationship with a table in the database.
- Question 2: The _____ determines the table space that the table is assigned to.
- Question 3: An _____ can be used to speed up the selection of data records from a table.



ABAP Part I

Lesson 05: Common Control statements

Lesson Objectives

- In this lesson, you will learn about:
 - If Statement and Case Statement
 - The exit Statement
 - The do and the While statement
 - The continue and Check statement



Copyright © Capgemini 2015. All Rights Reserved. 2

IF statement

- The if statement in ABAP/4 has relational operators for equality and inequality and special relational operators for string comparisons and bit masks.
- Syntax:

```
if [not] exp [ and [not] exp ] [ or [not] exp ].  
---  
[elseif exp.  
---]  
[else.  
---]  
endif.
```
- where:
 - exp is a logical expression that evaluates to a true or false condition
 - --- represents any number of lines of code.
 - Even zeros lines are allowed



Copyright © Capgemini 2015. All Rights Reserved 3

The following points apply:

Every if must have a matching endif
else and elseif are optional

Parenthesis can be used. Each parenthesis must be separated by a space.
For example, if (f1 = f2) or (f1 = f3) is correct, and if (f1 = f2) or
(f1 = f3) is incorrect

Variables can be compared with blanks or zeros using the addition is initial.

For example, If f1 is initial will be true if f1 is type c and is blank.

If f1 is any data type, the statement is true if f1 contains zeros

To accomplish negation, not must precede the logical expression.

For example, if not f1 is initial is correct.

if f1 is not initial is incorrect.

Variables can be compared with nulls using the addition is null. For
example, if f1 is null.

IF statement

```
Negation
IF gv_carrid IS NOT INITIAL.
  Statements
ELSE.
  Statements
ENDIF.

AND and OR Links with Parentheses
IF ( gv_carrid = 'AA' OR gv_carrid = 'LH' )
  AND gv_fldate = sy-datum.
  Statements
ELSEIF ( gv_carrid = 'UA' OR gv_carrid = 'DL' )
  AND gv_fldate > sy-datum.
  Statements
ENDIF.

Negation Before Logical Conditions
IF NOT ( gv_carrid = 'AA' OR gv_carrid = 'UA' )
  AND gv_fldate > sy-datum.
  Statements
ENDIF.
```



Copyright © Capgemini 2015. All Rights Reserved 4

Demo

- Program on using If Statement



Copyright © Capgemini 2015. All Rights Reserved 5

Logical operators for operands of any type

Comparison	Alternate Forms	True When
v1 = v2	eq	v1 equals v2
v1 <> v2	ne, ><	v1 does not equal to v2
v1 > v2	gt	v1 is greater than v2
v1 < v2	lt	v1 is less than v2
v1 >= v2	ge, =>	v1 is greater than or equal to
v1 <= v2	le, =<	v1 is less than or equal to
v1 between v2 and v3		v1 lies between v2 and v3 (inclusive)
not v1 between v2 and v3		v1 lies outside of the range v2 and v3 (inclusive)

- In the above table v1 and v2 can be variables, or literals, or field strings.
- In the case of variables or literals, automatic conversion is performed if the data type or length does not match.
- Field strings are treated as type c variables.



Copyright © Capgemini 2015. All Rights Reserved 6

The operators ><, => and =< are obsolete.

Case Statement

- The case statement performs a series of comparisons.
Syntax:
`case v1.
when v2 [or vn ...].

when v3 [or vn ...].

[when others.
---]
endcase.`
- where:
 - v1 or v2 can be a variable, literal, constant, or field string
 - --- represents any number of line of code.
 - Even zero lines are allowed



Copyright © Capgemini 2015. All Rights Reserved. 7

The following points apply:

- Only statements following the first matching when are executed
- when others matches if none of the preceding whens match
- If when others is not coded and none of the whens match, processing continues with the first statement following endcase
- Expressions are not allowed
- Field strings are treated as type c variables

Case Statement

- case is very similar to *if/else*.
- The only difference is that on each *if/elseif*, you can specify complex expression.
- With case, you can specify only a single value to be compared, and values are always compared for equality



Copyright © Capgemini 2015. All Rights Reserved 8

Demo

- Program on using case Statement



IF and Case statement

▪ Conditional Branches

```
IF gv_var > 0 .  
  Statements  
ELSEIF gv_var = 0 .  
  Statements  
ELSE  
  Statements  
ENDIF
```

```
CASE gv_carrid.  
  WHEN 'AA'.  
    Statements  
  WHEN 'LH'.  
    Statements  
  WHEN OTHERS .  
    Statements  
ENDCASE
```



Copyright © Capgemini 2015. All Rights Reserved 10

Exit statement

- The exit statement prevents further processing from occurring.

Syntax:

exit.

- The following example shows a sample program using exit.
report zdemo506.

write: / 'Hi'.

exit.

write: / 'There'.

- The above code produces this output:

Hi



Copyright © Capgemini 2015. All Rights Reserved 11

Exit can be used in many situations. It can have varying effects depending on where it appears in the code. However, it always prevents further processing. Within a loop structure, it leaves loop processing introduced by statements such as loop, select, do, and while. Within subroutines, it leaves subroutines introduced by FORM.

Demo

- Program on using Exit Statement



Loops

▪ Loops

- Unconditional loops using the DO.....ENDDO.
- Conditional Loops Using the WHILEENDWHILE.
- Loops through Internal Tables using the LOOPENDLOOP.
- Loops through datasets from database Tables using the SELECTENDSELECT

```
DO.                                         Loop Counter  
    Statements                                sy-index  
    IF <abort_condition>. EXIT. ENDIF.  
ENDDO.  
  
DO n TIMES.                               Loop Counter  
    Statements                                sy-index  
ENDDO.  
  
WHILE <condition>.  
    Statements  
ENDWHILE.                                Loop Counter  
                                            sy-index  
  
SELECT ... FROM <dbtab> ...  
Statements  
ENDSELECT.  
  
LOOP AT <internal table> ...  
    Statements  
ENDDO.
```



Do statement

■ **Syntax:**

do [v1 times]

[exit.]

enddo.

■ **where:**

- v1 is a variable, literal, or constant
- ---- represents any number of lines of code



Copyright © Capgemini 2015. All Rights Reserved 14

The following points apply:

do loops can be nested an unlimited number of times

exit prevents further loop processing and exits immediately out of the current loop.

It does not terminate the program when inside of a do loop. Processing continues at the next executable statement after the enddo

You can create an infinite loop by coding without any additions. In that situations, use exit within the loop to terminate loop processing

Modifying the value of v1 within a loop does not affect loop processing

Within the loop, sy-index contains the current iteration number.

For example, the first time through the loop, sy-index will be 1.

The second time through, sy-index will be 2, and so on.

After enddo, sy-index contains the value it had before entering the loop.

With nested do loops, sy-index contains the iteration number of the loop in which it is used

Using the while Statement

- The *while* statement is a looping mechanism similar to *do*.

Syntax:

```
while  
----  
[exit.]  
----  
endwhile
```



Copyright © Capgemini 2015. All Rights Reserved 15

The following points apply:

while loops can be nested an unlimited number of times and also be nested within other type of loop

exit prevents further loop processing and exits immediately out of the current loop.
Processing continues at the next executable statement after endwhile

Within the loop, sy-index contains the current iteration number. After endwhile, sy-index contains the value it had before entering the loop. With nested while loops, sy-index contains the iteration number of the loop in which it is used

endwhile always copies the value of f1 back into the sending component

Demo

- Program on Loops – Do and While



Loops

- Terminating Loops
 - Terminating Loop Pass Unconditionally
 - CONTINUE
 - EXIT
 - Terminate Loop Pass Conditionally
 - CHECK



Copyright © Capgemini 2015. All Rights Reserved 17

Continue Statement

- The *continue* statement is coded within a loop.
- It acts like a *goto* passing control, immediately to the terminating statement of the loop and beginning a new loop pass.
- In effect, it causes the statement below it within the loop to be ignored and a new loop pass to begin.



Copyright © Capgemini 2015. All Rights Reserved 18

Continue Statement

Syntax:

It can be used within a *do*, *while*, *select*, or *loop*.

[*do/while/select/loop*]

continue.

[*enddo/endwhile/endselect/endloop*]

where:

--- represents any number of lines of code



Copyright © Capgemini 2015. All Rights Reserved 19

The following points apply:

continue can only be coded within a loop

continue has no additions

Example: Continue statement

The effect of the *continue* statement is shown in following code snippet.

```
do 10 times.  
  if sy-index between 3 and 8.  
    continue.  
  endif.  
  write sy-index.  
enddo.
```

The above code produces this output:

```
1      2      9      10
```

The *continue* statement jumps to the end of the loop, ignoring all statements after it for the current loop pass.



Demo

- Program on continue statement



Check Statement

- The *check* statement is coded within a loop.
- It can act very much like *continue*, passing control immediately to the terminating statement of the loop and bypassing the statements between.
- Unlike *continue*, it accepts a logical expression.
- If the expression is true, it does nothing.
- If it is false, it jumps to the end of the loop



Check Statement

- Syntax:

It can be used within a *do*, *while*, *select*, or *loop*.
[*do/while/select/loop*]

check exp.

[*enddo/endwhile/endselect/endloop*]

- where:

- *exp* is a logical expression

- --- represents any number of lines of code



Copyright © Capgemini 2015. All Rights Reserved 23

Check Statement

- The *check logic_expr* statement has the following effect:
 - Outside a loop, you can terminate a processing block prematurely.
 - The block statements after the *check* statement are skipped if the logical condition is not fulfilled (false).
 - The system then continues with the first statement in the next processing block
 - Within a loop, it has the effect that the next loop is processed.



Copyright © Capgemini 2015. All Rights Reserved 24

Check Statement

- The effect of the check statement is shown in following snippet of code.

do 10 times.

 check not sy-index between 3 and 8.

 write sy-index.

 enddo.

The above code produces this output:

1 2 9 10

- The check statement is a conditional continue statement.
- It jumps to the end of the loop, if the logical expression is false.
- If the expression is true, it does nothing.
- If it is false, it jumps to the end of the loop



Copyright © Capgemini 2015. All Rights Reserved 25

Demo

- Program on check statement



Comparing the exit, continue, and check Statements

Statement	Effect
exit	Leaves the current loop
continue	Unconditional jump to the end of the loop
check exp	Jumps to the end of the loop if exp is false



Copyright © Capgemini 2015. All Rights Reserved 27

Summary

- In this lesson, you have learnt:
 - If Statement and Case Statement
 - The exit Statement
 - The do and the While statement
 - The continue and Check statement



Review Question

- Question 1: In a case statement a complex expression can be compared
 - True/False
- Question 2: The _____ statement leaves the current loop.



ABAP Part I

Lesson 06: String Operations

Lesson Objectives

- In this lesson, you will learn about:
 - The various commands for handling strings



Using the Shift command

- This command is used to shift the contents of a string.
- It shifts a string by a given number of places
- Syntax:
 - *shift s_field [by n places] <mode>*.
- Following points apply:
 - If the by n places clause is omitted, the string is shifted by 1 place.
 - The <mode> defines the direction of movement the options being
 - right
 - left (default)
 - circular: shifts n positions to the left so that leftmost characters appear on the right.
- One can also specify shift...circular right



Copyright © Capgemini 2015. All Rights Reserved 3

Shift a string up to a given string

- Syntax:
 - *shift str_1 up to str_2 <mode>*
- Searches str_1 for str_2 and the string up to the field margin.
- If str_2 is not found sy-subrc is set to 4.
- <mode> specification is as given previously
- Eg:
 - data : alpha(10) value 'abcdef'.*
 - Shift alpha up to 'cd'.*
 - Write alpha.*
 - Shift alpha up to 'cd' right.*
 - Write / alpha.*
- The output is

Cdef

Cd



Copyright © Capgemini 2015. All Rights Reserved 4

Shifting the string up to the first or last character

- Shift str_1 left deleting leading str2.
Eg: *Shift '000100100500' left deleting leading '0'.*
Will output
100100500
- Shift str_1 right deleting trailing str2.
Eg: *shift '100100500' right deleting trailing '0'.*
Will output
1001005



Copyright © Capgemini 2015. All Rights Reserved 5

Demo

- Program on shift statement



Copyright © Capgemini 2015. All Rights Reserved 6

Translate

- The TRANSLATE statement converts characters into upper or lower case, or uses substitution rules to convert all occurrences of one character to another character.
- Substituting Characters
 - TRANSLATE text USING pattern.
 - This statement replaces all characters in the text field according to the substitution rule stored in pattern.
 - Pattern contains letter pairs where the first letter of each pair is replaced by the second.
 - Pattern can be a variable.
- Syntax:
 - *replace str_1 with str_2 into str_3 [length n].*



Copyright © Capgemini 2015. All Rights Reserved. 7

Translate

```
DATA: t(10) TYPE c VALUE 'AbCdEfGhIj',
      string LIKE t,
      rule(20) TYPE c VALUE 'AxbXCydYEzfZ'.
      string = t.
      WRITE string.
      TRANSLATE string TO UPPER CASE.
      WRITE / string.
      string = t.
      TRANSLATE string TO LOWER CASE.
      WRITE / string.
      string = t.
      TRANSLATE string USING rule.
      WRITE / string.
      Output:
      AbCdEfGhIj
      ABCDEFGHIJ
      abcdefghij
      xXyYzZGhI
```



Copyright © Capgemini 2015. All Rights Reserved 8

Translate

- translate ... to upper case.
 - Converts the contents of the field to uppercase.
 - Eg:
data : alpha(10) value 'abcd12'.
Translate alpha to upper case.
Write alpha.
Will output – ABCD12

- translate ... lower case
 - Converts the contents of the field to lower case.



Copyright © Capgemini 2015. All Rights Reserved 9

Translate

- translate X using Y.

- Modifies the contents of X using the values given in Y.
- Replaces each occurrence of the odd numbered characters of Y in X with the even numbered character in Y.

- Eg:

data : alpha(10) value 'abcd12'.

* a->A : B->c : 1->2

Translate alpha using 'aABc12'.

Write alpha.

Will output Abcd22.



Demo

- Program on Translate statement



Find

- FIND statement is used for finding occurrences of a pattern within a string.
- It can also be used for finding the string pattern within an internal table.
 - *Find <str> for <pattern> <option>.*
 - <str> is the string that is being searched.
 - <pattern> is the sequence of characters we're looking for.
 - While specifying the search patterns the following forms are available .
 - Find <first occurrence> <all occurrences> of pattern in <str> <respecting|ignoring case> <match count mcnt> <match length mlen>
 - <match offset moff> <results result_tab|result_wa>



Copyright © Capgemini 2015. All Rights Reserved 12

Demo

- Program on find statement



Copyright © Capgemini 2015. All Rights Reserved 13

Strlen

Used to find the length of the string.

Eg: data : len type I,
 string(12) value 'avbsok'.
 len = strlen(string).
 write len.
The output will be 6.



Copyright © Capgemini 2015. All Rights Reserved 14

Demo

- Program on strlen



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 15

Condense

- This command is used to remove superfluous spaces from a string.
- After using the *condense* statement, all the words in the string are separated by 1 blank space only.

Eg: *data : alpha(20) value 'lets get on-line whatsay??'.*

condense alpha.

write alpha.

Will display

lets get on-line whatsay??



Copyright © Capgemini 2015. All Rights Reserved 16

Condense no-gaps

- With the no-gaps option all the blank spaces from the string are removed and the string is left-justified.

Alpha = ' lets get on-line whatsay??'.

Condense alpha no-gaps.

Write alpha.

This will display

letsgeton-linewhatsay??



Copyright © Capgemini 2015. All Rights Reserved 17

Demo

- Program on condense and condense no-gaps



Copyright © Capgemini 2015. All Rights Reserved 18

Concatenate

- Is used to combine separate strings into a single string.

Eg: *data : f1(20) value 'A',
 F2(20) value 'B',
 F3(20).*

concatenate f1 f2 into f3.

write f3.

Gives the following output

AB



Concatenate

The separated by option is used as follows
concatenate f1 f2 into f3 separated by ''.*

write f3.

Gives the following output

A*B.



Copyright © Capgemini 2015. All Rights Reserved 20

Demo

- Program on concatenate statement



Copyright © Capgemini 2015. All Rights Reserved 21

Split command

- The *split* command is used to split a string at a given delimiter. The format is

- split <string> at <delimiter> into <f1> <f2> <f3>....***

Eg: data : string(20) value 'hello what was that??',
 F1(10),
 F2(10).

split string at ' ' into f1 f2.

F1 and f2 will contain 'hello' and 'what' respectively.



Operators for Character Strings

- These operators can be used in any comparison expression. The CS, NS, CP, and NP operators ignore trailing blanks and are not case sensitive

Operator	Means	True When	Case Sensitive?	Trailing Blanks Ignored?
v1 CO v2	Contains Only	v1 is composed solely of characters in v2	Yes	No
v1 CN v2	not v1 CO v2	v1 contains characters that are not in v2	Yes	No
v1 CA v2	Contains Any	v1 contains at least one character in v2	Yes	No
v1 NA v2	not v1 CA v2	v1 does not contain any character in v2	Yes	No
v1 CS v2	Contains String	v1 contains the character string v2	No	Yes
v1 NS v2	not v1 CS v2	v1 does not contain the character string v2	No	Yes
v1 CP v2	Contains Pattern	v1 contains the pattern in v2	No	Yes
v1 NP v2	not v1 CP v2	v1 does not contain the pattern in v2	No	Yes



Copyright © Capgemini 2015. All Rights Reserved 23

Pattern Matching Characters

- The *CP* (contains pattern) and *NP* (no pattern) operators perform a string search that allows pattern-matching characters.
- The expression *v1 CP v2* is true when *v1* contains a string that matches the pattern in *v2*.
- The expression *v1 NP v2* is true when *v1* does not contain a string that matches the pattern in *v2*.
- The pattern matching characters allowed in *v2* are given in table below.

Character	Used to
*	Match any sequence of characters
+	Match any single character
#	Interpret the next character literally



Copyright © Capgemini 2015. All Rights Reserved 24

Example of Wild card characters used for Pattern Matching

Statement	True When
v1 CP 'A+C'	v1 contains "a" in the first position and "c" in the third. Either character can be in upper- or lowercase. Any character can appear in the second position
v1 CP "Ab"	The string can appear anywhere within v1. Either character can be in upper- or lowercase.
v1 CP "#A#b"	v1 contains a capital A followed by lowercase b
v1 CP "##"	v1 contains a #

- # is the escape character.
- A single character following it is interpreted exactly.
- Special meaning, if it exists, is lost.
- # can also be used to make a search case sensitive or to search for the *, +, or # character.
- The escape character is needed when you want to perform a case-sensitive search using CS, NS, CP, or NP.
- You also need it if you want to perform a pattern search (CP or NP) for a string containing *, +, or #.



Copyright © Capgemini 2015. All Rights Reserved 25

Values of sy-fdpos with string comparisons

Comparison	if True sy-fdpos =	if False sy-fdpos =
v1 CO v2	length (v1)	1 st char (v1) not in v2
v1 CN v2	1 st char (v1) in v2	length (v1)
v1 CA v2	1 st char (v1) in v2	length (v1)
v1 NA v2	length (v1)	1 st char (v1) in v2
v1 CS v2	1 st char (v2) in v1	length (v1)
v1 NS v2	length (v1)	1 st char (v1) in v2
v1 CP v2	1 st char (v2) in v1	length (v1)

- Use of these operators always sets the system variables *sy-fdpos*.
- If the result of the comparison is true, *sy-fdpos* contains the zero-based offset of the first matching or non-matching character.
- Otherwise, *sy-fdpos* contains the length of *v1*.
- The value assigned to *sy-fdpos* by each operator is described in the above table



Copyright © Capgemini 2015. All Rights Reserved 26

Demo

- Program operators for Character Strings



Summary

- In this Lesson you learnt:
 - Shift, replace, translate, find, strlen, condense, concatenate, and split command
 - Operators for Handling Strings



Review Question

- Question 1: _____ is used to find the length of the string.
- Question 2: Condense and Concatenate command perform the same function.
- True/False



ABAP Part I

Lesson 07: Internal Tables

Lesson Objectives

- After completing this lesson, participants will be able to know:
 - To Define an Internal Table and understand its attributes
 - Types of Internal Tables
 - To Add, Read, Update and Delete Data from an internal Table
 - To Sort the Contents of an Internal Table
 - Control break statements on Internal Table



Internal Tables

- Provides a means of taking data from a fixed structure and storing it in working memory in ABAP
- Internal Tables fulfill the function of Arrays
- A very important use of internal tables is for storing and formatting data from a database table within a program.
- The data type an internal table is fully specified by its
 - line type
 - key
 - table type



Copyright © Capgemini 2015. All Rights Reserved 3

Internal Tables

■ Line Type

- Can be any data type
- The data type of an internal table is usually a Structure
- Each component of a structure is a column in internal table

■ Key

- Identifies Table rows
- Two kinds of keys
 - Standard Key
 - User-Defined Key

1	Line type	CARRID	CONNID	DISTANCE
①	index	AA	0017	2.572
②		LH	0400	6.162
③		LH	0402	7.273
④		QF	0005	10.000
⑤		SQ	0866	1.625
⑥		UA	0007	2.572

2 Key
■ Components
■ Uniqueness
■ Sequence

3 Table type
■ Standard
■ Sorted
■ Hashed

Internal Tables - Key

▪ Standard Key

- Tables with Structured and non -structured row type : all Character Type columns
- Tables with Elementary line-type : entire line is the default key
- If line type is an internal table : no default key
- Tables with non-structured row type :

▪ User-Defined Key

- Any column of an internal table which is not an internal table by itself
- Key can be UNIQUE or NON-UNIQUE



Copyright © Capgemini 2015. All Rights Reserved 5

Internal Tables – Table Type

- Determines how ABAP accesses the individual entries
- Three Types
 - Standard Tables
 - Sorted Tables
 - Hashed Tables



Copyright © Capgemini 2015. All Rights Reserved. 6

Operations on Individual Lines

- Key and Index Access

	Standard Table	Sorted Table	Hashed Table
Index Access	✓	✓	X
Key Access	✓	✓	✓
Key Values	Not Unique	Unique/Not Unique	Unique
Preferred Access	Mainly Index	Mainly Key	Key Only



Copyright © Capgemini 2015. All Rights Reserved

7

Internal Tables – Standard Table

- Have an internal linear index
- Records are accessed by index or keys
- The response time for key access is proportional to number of entries
- The key is always non-unique



Copyright © Capgemini 2015. All Rights Reserved. 8

Internal Tables – Sorted Table

- Always saved sorted by the key
- Have an internal index
- Records can be accessed by table index or key
- Uses Binary Search for access
- Key can be Unique or non-Unique



Copyright © Capgemini 2015. All Rights Reserved 9

Internal Tables – Hashed Table

- Has no linear index
- Only accessed using its key
- Key must be Unique
- Uses Hash Algorithm for accessing records
- Response time is constant



Copyright © Capgemini 2015. All Rights Reserved 10

Creating Internal Tables

- Syntax:

```
TYPES type TYPE|LIKE tabkind OF linetype [WITH key] [INITIAL SIZE n].  
DATA itab TYPE type|LIKE obj.
```

- Example:

```
DATA : BEGIN OF str_mat,  
        matno(18),  
        matname(30),  
        mattype(4), " Material Type  
        uom(10), "Unit of Measure  
    END OF str_mat.  
DATA IT_MAT like STANDARD TABLE OF str_mat."Internal Table  
with line type str_mat
```



Copyright © Capgemini 2015. All Rights Reserved 11

Demo

- Program on Internal table without header line



Internal Tables Objects

- Internal tables are dynamic variable data objects.
- Like all variables, declare them using the DATA statement.



Copyright © Capgemini 2015. All Rights Reserved 13

Reference to Declared Internal Table Types

- Declare internal table objects using the LIKE or TYPE.

DATA <itab> TYPE <type>|LIKE <obj> [WITH HEADER LINE].

- Here, the LIKE addition refers to an existing table object in the same program.
- The TYPE addition can refer to an internal type in the program declared using the TYPES statement, or a table type in the ABAP Dictionary.
- The optional addition WITH HEADER LINE declares an extra data object with the same name and line type as the internal table.



Copyright © Capgemini 2015. All Rights Reserved 14

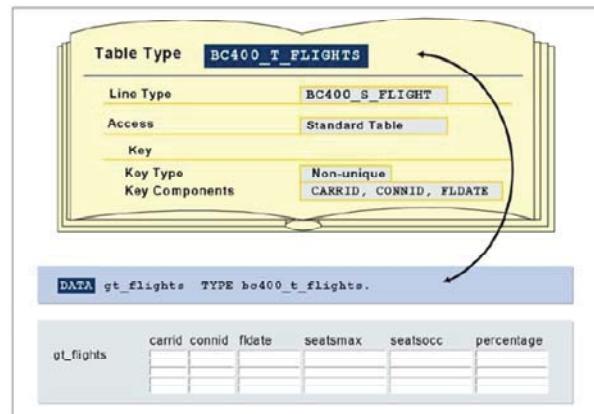
Reference to Declared Internal Table Types

- Use it as a work area when working with the internal table.
- When using internal tables with header lines, the header line and the body of the table have the same name.
- If there is an internal table with header line, to address the body of the table, place brackets after the table name (<itab>[]).
- If not, ABAP interprets the name as the name of the header line and not of the body of the table.
- It is possible to avoid this potential confusion by using internal tables without header lines.



Copyright © Capgemini 2015. All Rights Reserved 15

Defining Internal Tables with Global Types



Defining Internal Tables with Local Types

```
TYPES gty_t_flights
      TYPE STANDARD TABLE OF bc400_s_flight
      WITH NON-UNIQUE KEY carrid connid fldate.
```

```
DATA gt_flights  TYPE gty_t_flights.
```

Internal Table



Independent Definition of Internal Tables

```
TYPES: BEGIN OF gty_s_type,
         carrid TYPE s_carr_id,
         connid TYPE s_conn_id,
         ...
      END OF gty_s_type.  
  
          STANDARD
DATA gt_itab  TYPE SORTED TABLE OF gty_s_type
      HASHED
      WITH ... KEY ...
```

Local Structure
Type

Internal Table



Copyright © Capgemini 2015. All Rights Reserved 18

Possible Definition of Internal Tables

```
1 DATA gt_itab TYPE <Table Type> .  
  
2 DATA gt_itab TYPE STANDARD  
      SORTED TABLE OF <Structure Type>  
      HASHED  
      WITH ... KEY ...  
  
3 DATA gt_itab TYPE TABLE OF <Structure Type> .  
(Short form for definition of a standard table with  
non-unique default key)
```



Operations on Entire Internal Table

- The entire body of the table is addressed as a single Data Object
- The following operations are done on the entire Internal Table
 - Assigning Internal Tables
 - Initializing Internal Tables
 - Compare Internal Tables
 - Sort Internal Tables
 - Internal Tables as Interface Parameters
 - Determining the Attributes of Internal Tables



Copyright © Capgemini 2015. All Rights Reserved 20

Assigning Internal Tables

■ Syntax:

- MOVE itab1 TO itab2. or
- itab2 = itab1.

• Example:

```
DATA : it_mat LIKE TABLE OF str_mat,  
      it_mat1 LIKE TABLE OF str_mat.
```

```
str_mat-matno = 'm01'.  
str_mat-matname = 'Notepads'.  
str_mat-mattype = 'Stationery'.  
str_mat-uom = 'pcs'.
```

```
APPEND str_mat TO it_mat.  
MOVE it_mat to it_mat1. "or it_mat1 = it_mat.
```



Copyright © Capgemini 2015. All Rights Reserved 21

Initializing Internal Tables

- **CLEAR itab.**

- Releases the memory space of the internal table, but for the initial memory requirement.

- **REFRESH itab**

- Applies to the body of the internal table.
 - The initial memory requirement for the table remains reserved.

- **FREE itab**

- The internal table is initialized and the entire memory space is released.



Copyright © Capgemini 2015. All Rights Reserved 22

Demo

- Program on Free, Refresh and Clear



Internal Tables - Processing

- Sorting Internal Tables
 - Syntax

SORT itab [ASCENDING|DESCENDING] [AS text] [STABLE].

- To get the Attributes of Internal Tables
 - Syntax

DESCRIBE TABLE itab [LINES lin] [OCCURS n] [KIND knd].



Copyright © Capgemini 2015. All Rights Reserved 28

Sort Internal Table

```
SORT gt_flightinfo.  
  
SORT gt_flightinfo BY carrid.  
  
SORT gt_flightinfo BY percentage DESCENDING  
                      carrid          ASCENDING .
```



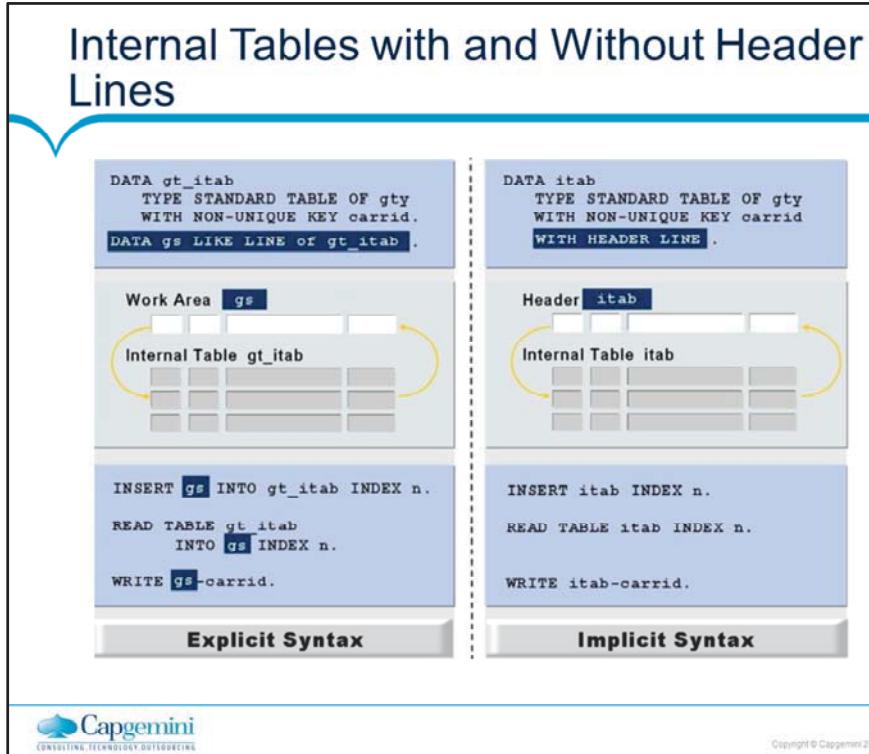
Copyright © Capgemini 2015. All Rights Reserved 25

Access Methods to Individual Table Entries

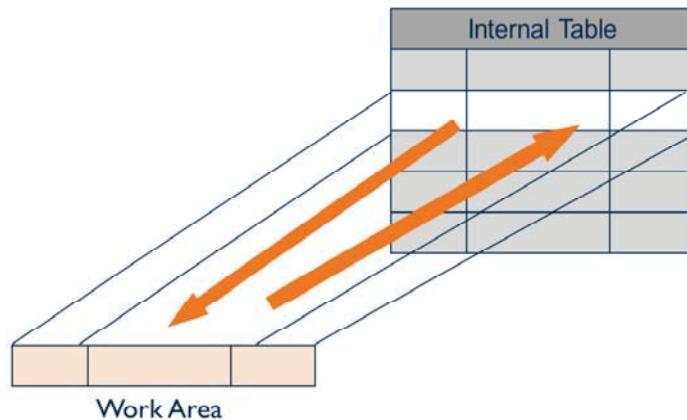
- 2 ways to access the individual rows of an Internal Table
 - Accessing the Internal Table Rows Using a Work Area
 - The data in the table is not directly accessed but through another Data Object referred as a Work Area
 - Work Area must be compatible with the line type of internal table or must be convertible into line type of the Internal Table
 - When a data is read from the table, the data overwrites the current contents of the Work Area
 - When data is written to the Internal Table, it must be placed in the Work Area and then transferred to the Internal Table
 - If the internal table has a Header Line, the Header Line can act as a Work Area.



Copyright © Capgemini 2015. All Rights Reserved 26



Access Using a Work Area



Accessing Single Records (Overview)

The diagram illustrates five ABAP statements for accessing single records:

- Append**: APPEND gs TO gt_itab.
- Insert**: INSERT gs INTO TABLE gt_itab <condition>.
- Read**: READ TABLE gt_itab INTO gs <condition>.
- Change**: MODIFY TABLE gt_itab FROM gs [<condition>]
- Delete**: DELETE gt_itab <condition>.

Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 29

Processing Sets of Records (Overview)

Processing record by record over the entire (or a part) of the internal table



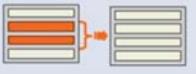
```
LOOP AT gt_it INTO gs <condition>.  
...  
ENDLOOP
```

Deleting several records



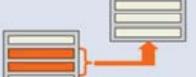
```
DELETE gt_it <condition>.
```

Inserting several rows from another internal table



```
INSERT LINES OF gt_it1 <condition1>  
INTO gt_it2 <condition2>.
```

Appending several rows from another internal table



```
APPEND LINES OF gt_it1 <condition>  
TO gt_it2.
```

Filling Internal Tables Line By Line

- To Insert Individual Row at a Specific Position

INSERT wa INTO TABLE itab.

- To Append Rows to the Internal Table

APPEND wa TO itab.

- To Create Unique or Aggregate Internal Tables

COLLECT wa INTO itab.



Copyright © Capgemini 2015. All Rights Reserved 31

Filling Internal Tables

- Standard Tables
 - Lines are always Appended to the end of the Internal Table.
- Sorted Tables
 - The lines are inserted into the table according to the table key
 - If the key is non-Unique, the duplicate entry is inserted above the existing entry with the same key.
- Hashed Tables
 - The lines are inserted according to the table key.



Copyright © Capgemini 2015. All Rights Reserved 32

Inserting Lines

■ Inserting Several Lines

- To insert several lines into the internal table use

INSERT LINES OF itab1 [FROM n1] [TO n2] INTO TABLE itab2.

■ Inserting Lines Using The Index

- Inserting a Single Line

INSERT WA INTO itab [INDEX idx].

- Inserting Several Lines

INSERT LINES OF itab1 [FROM n1] [TO n2] INTO itab2 [INDEX idx].



Copyright © Capgemini 2015. All Rights Reserved 33

Inserting a Row

```
* define internal table and workarea
DATA: gt_flightinfo TYPE bc400_t_flights,
      gs_flightinfo LIKE LINE OF gt_flightinfo.

gt_flightinfo
gs_flightinfo
```

Diagram illustrating the structure of the internal table `gt_flightinfo`. It shows a grid of 10 columns and 10 rows, representing a 10x10 matrix.

```
* fill structure with values
gs_flightinfo-carrid      = ...
gs_flightinfo-connid       = ...
gs_flightinfo-fldate        = ...
gs_flightinfo-seatsmax     = ...
gs_flightinfo-seatsocc     = ...
gs_flightinfo-percentage   = ...

* insert structure into internal table
INSERT gs_flightinfo INTO TABLE gt_flightinfo.
```



Copyright © Capgemini 2015. All Rights Reserved 34

Demo

- Program on Insert a record and Insert Multiple records



Appending Lines

- Appending a Single Line
 - APPEND line TO itab.
- Appending Multiple Lines
 - APPEND LINES of itab1 TO itab2.



Copyright © Capgemini 2015. All Rights Reserved 36

Reading Lines of Table

- To Read a Single Line

READ TABLE itab key result.

- The system field SY-SUBRC is set to 0 if there is a matching entry in the internal table and if not, it sets the value to 4.

- To Read a Single line based on the Condition

READ TABLE itab FROM wa result.

READ TABLE itab WITH TABLE KEY k1 = f1 ... kn = fn result.

- To Read a Single Line Using Index (Specifying the row number)

READ TABLE itab INDEX idx result.



Copyright © Capgemini 2015. All Rights Reserved 37

Demo

- Program on reading from Internal Tables



Processing Table Entries in Loops

▪ Syntax:

LOOP AT itab result condition.

Statements...

ENDLOOP.

- Depending on the table types the lines are processed

▪ Standard and Sorted Table

- Processed based on Index
- The system field SY-TABIX stores the index of the current line

▪ Hashed Table

- If the table is not sorted, the lines are accessed in the inserted order
- SY-TABIX = 0 , within the processing block.



Copyright © Capgemini 2015. All Rights Reserved 39

Outputting an Internal Table Row-by-Row

```
* define internal table and workarea
DATA: gt_flightinfo TYPE bc400_t_flights,
      gs_flightinfo LIKE LINE OF gt_flightinfo.
gt_flightinfo
gs_flightinfo
```

LOOP AT gt_flightinfo INTO gs_flightinfo.

```
WRITE: / gs_flightinfo-carrid,
        gs_flightinfo-connid,
        gs_flightinfo-fldate,
        gs_flightinfo-seatsmax,
        gs_flightinfo-seatsocc,
        gs_flightinfo-percentage,
        '%'.
ENDLOOP.
```

sy-tabix



Copyright © Capgemini 2015. All Rights Reserved 40

Reading Internal table

```
LOOP AT gt_flightinfo INTO gs_flightinfo
  FROM 1 TO 5 .
  WRITE: / gs_flightinfo-carrid,
           gs_flightinfo-connid,
           gs_flightinfo-fldate,
           gs_flightinfo-seatsmax,
           gs_flightinfo-seatsocc,
           gs_flightinfo-percentage,
           '%'.
  ENDLOOP.

READ TABLE gt_flightinfo INTO gs_flightinfo
  INDEX 3
  WRITE: / gs_flightinfo-carrid,
           gs_flightinfo-connid,
           gs_flightinfo-fldate,
           gs_flightinfo-seatsmax,
           gs_flightinfo-seatsocc,
           gs_flightinfo-percentage,
           '%'.  
sy-tabix
```



Copyright © Capgemini 2015. All Rights Reserved 41

Reading Internal table

```
LOOP AT gt_flightinfo INTO gs_flightinfo
      WHERE carrid = 'LH' .
      WRITE: /  gs_flightinfo-carrid,
              gs_flightinfo-connid,
              gs_flightinfo-fldate,
              gs_flightinfo-seatsmax,
              gs_flightinfo-seatsocc,
              gs_flightinfo-percentage,
              '%'.
ENDLOOP.

READ TABLE gt_flightinfo INTO gs_flightinfo
      WITH TABLE KEY  carrid = 'LH'
                      connid = '0400'
                      fldate = sy-datum .
      IF sy-subrc = 0.
      WRITE: /  gs_flightinfo-seatsmax,
              gs_flightinfo-seatsocc,
              gs_flightinfo-percentage,
              '%'.
ENDIF.
```



Changing Lines

- Changing Lines of an Internal Table

MODIFY itab FROM wa [TRANSPORTING f1 f2 ...].

- Changing Internal Table Contents based on Condition

MODIFY itab FROM wa TRANSPORTING f1 f2 ... WHERE cond.

- Changing the contents of an Internal Table Using an Index

MODIFY itab FROM wa [INDEX idx] [TRANSPORTING f1 f2...].



Copyright © Capgemini 2015. All Rights Reserved 43

Deleting Lines

`DELETE TABLE itab FROM wa.`

- Deletion Using Table Key

`DELETE TABLE itab WITH TABLE KEY k1 = f1 ... kn = fn.`

- Delete the Internal Table Contents Based on a Condition

`DELETE itab WHERE cond.`

- Deleting Duplicate Records

`DELETE ADJACENT DUPLICATE ENTRIES FROM itab [COMPARING f1 f2 ... |ALL FIELDS].`



Copyright © Capgemini 2015. All Rights Reserved 44

Deleting Lines – Contd..

- Deleting a line specifying the Index

DELETE itab [INDEX idx].

- Deleting Several Lines

DELETE [FROM n1] [TO n2] [WHERE condition].



Copyright © Capgemini 2015. All Rights Reserved 45

Demo

- Program on deleting from Internal tables



Determining the Number of Rows in an Internal Table

- To determine the number of rows in an internal table, use sy-tfill variable.
- It is set by the describe table statement.
- Syntax

The following is the syntax for the *describe table* statement.

describe table it [lines i] [occurs j].

- where:

- it is the name of an internal table
- i and j are numeric variables



Copyright © Capgemini 2015. All Rights Reserved 47

Determining the Number of Rows in an Internal Table

- The describe statement fills the three system variables shown in table below

Variable	Value
<i>Sy-tfill</i>	Number of rows
<i>Sy-tleng</i>	Length of a row in bytes
<i>Sy-toccu</i>	Current value of the occurs clause

- The following points apply:

- If the lines i addition is specified, the number of rows is placed in both *sy-tfill* and *i*
- If the occurs j addition is specified, the size of the occurs clause is placed in both *sy-toccu* and *j*



Copyright © Capgemini 2015. All Rights Reserved 48

There is only one instance where *sy-toccu* will differ from the occurs clause on the table definition.

When *sy-tleng* * *sy-toccu* > 8192, and after one row has been added to the internal table, *sy-toccu* will be zero.

This indicates that memory is being allocated in 8K chunks for this internal table.

Creating Top 10 Lists Using the append sorted by

- Imagine that you are asked to create a report of the top 10 sales representatives in your company.
- Assuming you have a way of obtaining the total sales for each rep, you could do one of the following:
 - Append all reps and their total sales into an internal table
 - Sort them descending by sales
 - Write out the first 10
- This seems like a logical way to proceed.
- However, using the append sorted by statement often can produce the same result and be twice as efficient



Copyright © Capgemini 2015. All Rights Reserved 49

Creating Top 10 Lists Using the append sorted by

- Syntax for the append sorted by Statement:

- append [wa to] it sorted by c.

- where:

- it is the name of an internal table
 - wa is a work area having the same structure as row of the internal table
 - c is a component of it



Copyright © Capgemini 2015. All Rights Reserved 50

The following points apply:

If wa to is not specified, the row to be appended is taken from the header line

If wa to is specified, the row to be appended is taken from the work area wa
Only one component c can be specified

Creating Top 10 Lists Using the append sorted by

- The append sorted by statement takes a row from the work area and inserts it into the internal table at the point where it belongs in the sort order.
- It has two unusual properties:
 - The number of rows that can be appended is limited by the value on the occurs clause.
 - For example, if the occurs is 10, a maximum of 10 rows can be appended to the internal table.
 - This is the only situation where occurs limits the number of rows that can be added to an internal table
 - It only sorts in descending order
- The net effect is a “top n list,” where n is the number on the occurs clause.



Copyright © Capgemini 2015. All Rights Reserved 51

Creating Top 10 Lists Using the append sorted by

- With each append, the system searches the existing table contents to determine where the new record fits.
- The sort order is by c descending.
- If there are fewer rows in the internal table than specified by n on the occurs clause, the row is as per the sort order.
- If there are n rows in the internal table, the row is inserted as per the sort order and the last row is discarded.
- If the value in c already exists in the internal table, the new row is always appended after existing rows that have the same value.
- Therefore, if occurs is 3 and row 3 contains 'X' in c, a new row having 'X' in c will not be appended



Copyright © Capgemini 2015. All Rights Reserved 52

Demo

- Program Append Sorted By



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 53

Filling an Internal Table Using collect

- Using the collect statement, you can create totals with an internal table as you fill it.

- Syntax for the collect Statement**

The following is the syntax for the collect statement.

collect [wa into] it.

- where:**

- it is an internal table
- wa is a work area having the same structure as it



Copyright © Capgemini 2015. All Rights Reserved 54

The following points apply:

If wa into is specified, the row to be collected is taken from the explicit work area wa. In this case, the header line of the internal table is ignored, if it has one.

If wa into is not specified, the internal table must have a header line. The row to be collected is taken from the header line for it.

When collect is executed, the system forms a key from the default key fields in the work area. The default key fields are the character fields (types c, n, d, t, and x).

Therefore the key is composed of the values from all fields of type c, n, d, t, and x. It doesn't matter if they are beside one another or separated from each other by other fields.

The system then searches the body of the internal table for a row that has the same key as the key in the work area. If it doesn't find one, the row is appended to the end of the table. If it does find one, the numeric fields (types i, p, and f) in the work area are added to the corresponding fields in the found row.

Remarks: If you use collect to add rows to an internal table, all rows should be added using collect. You should not combine collect with append or any other statement that adds data to an internal table. The results will be unpredictable. The only other statement that you can use to modify the contents of an internal table filled via collect is modify...transporting f1 f2..., where f1 and f2 are numeric fields only (non default key fields).

Demo

■ Program Collect



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 55

Control Break Processing

- After you fill an internal table with data, you often need to write the data out.
- This output will frequently contain summary information (such as totals) at the top or bottom of the report.
- To do this, you can read the data into the internal table and then, within loop at, use the following statements:
 - at first / endat
 - at last / endat
 - at new / endat
 - at end of / endat
 - sum
 - on change of / endon



Copyright © Capgemini 2015. All Rights Reserved 56

The first statement of each of these statement pairs-except for sum-controls when the code that lies between them is executed. This type of control is called a control break. Their purpose is to execute the code between them whenever a specific condition in the data is detected during the processing of the loop

Using the at first and at last Statements

- Use the at first and at last statements to perform processing during the first or last loop pass of an internal table.
- Syntax for the at first and at last Statements

The following is the syntax for the *at first* and *at last* statements.
loop at it.

```
...  
    ...  
        at first.  
        ...  
        endat.  
...  
    ...  
        at last.  
        ...  
        endat.  
    endloop
```

- where:

- it is an internal table
- ... represents any number of lines of code (even zero)



Copyright © Capgemini 2015. All Rights Reserved 57

The following points apply:

These statements can only be used within a loop at; they cannot be used within select at first does not have to come before at last.

The order of these statements can be interchanged

These statements can appear multiple times within the same loop.

For example, you could have two at first and three at last statements within one loop and they can appear in any order

These statements should not be nested inside of another (that is, at last should not be placed inside of at first and endat

The first time through the loop, the lines of code between at first and endat are executed. The last time through the loop, the lines of code between at last and endat are executed. If there are multiple occurrences of at first, they are all executed. at last behaves in a similar fashion.

Using the at first and at last Statements

- Use at first for:

- Loop initialization processing
- Writing totals at the top of a report
- Write headings

- Use at last for:

- Loop termination processing
- Writing totals at the bottom of a report
- Write footings



Copyright © Capgemini 2015. All Rights Reserved 58

Between the at first and endat, or between the at last and endat, the component values of the work area will not contain any data. The default key fields are filled with * (asterisks) and the numeric fields are set to zeros. The endat restores the contents to the values they had prior to entering the at. Changes to the work area within at and endat are lost.

Using the at new and at end of Statements

- Use the at new and at end of statements to detect a change in a column from one loop pass to the next.
- Syntax for the at new and at end of Statements

The following is the syntax for the at new and at end of statements.

```
sort by c.  
loop at it.  
...  
    at new c.  
    ...  
    endat.  
...  
    at end of c.  
    ...  
    endat.  
endloop
```

- where:

- it is an internal table
- c is a component of it
- ... represents any number of lines of code (even zero)



Copyright © Capgemini 2015. All Rights Reserved 59

The following points apply:

These statements can only be used within a loop at; they cannot be used within select at new does not have to come before at end of. These statements can appear in any order

These statements can appear multiple times within the same loop. For example, you could have two at new and three at end of statements within one loop and they can appear in any order

These statements should not be nested inside of another (that is, at end of should not be placed inside of at new/endat

There are no additions to these statements

Using at New

- Each time the value of c changes, the lines of code between at new and endat are executed.
- This block is also executed during the first loop pass or if any fields to the left of c change.
- Between at and endat, the numeric fields to the right of c are set to zero.
- The non-numeric fields are filled with asterisks (*).
- If there are multiple occurrences of at new, they are all executed.
- at end behaves in a similar fashion



Copyright © Capgemini 2015. All Rights Reserved 60

Using at New

- A control level is the component named on a control break statement; it regulates the control break.
- For example, in the following code snippet, f2 is a control level because it appears on the at new statement.
- loop at it.

```
at new f2.  
  "(some code here)  
  endat.  
endloop.
```



Copyright © Capgemini 2015. All Rights Reserved 61

It is said that a control break is triggered if the control level changes. This means that when the contents of the control level change, the code between the at and endat is executed.

A control break is also triggered if any of the fields prior to the control level in the structure change. Therefore, you should define the internal table structure to begin with the fields that form your control levels. You must also sort by all fields prior to and including c.

Between at and endat, numeric fields to the right of the control level will be zero and non-numeric fields will be filled with asterisks.

Using at end of

- The lines of code between at end of and endat are executed:
 - If the control level changes
 - If any field prior to the control level changes
 - If this is the last row of the table



Copyright © Capgemini 2015. All Rights Reserved 62

Do not use control break statements within a loop at it where... construct; the effects are unpredictable.

Using the sum Statement

- Use the sum statement to calculate totals for the rows of a control level.
- Syntax for the sum Statement

The following is the syntax for the sum statement.

at first / last / new / end of.

...

sum.

...

endat.

- where:

- ... represents any number of lines of code



Copyright © Capgemini 2015. All Rights Reserved

63

sum calculates a total for the current value of the control level field and all fields to the left of it.

This is clearer if you imagine that it does the following:

It finds all rows that have the same values within the control level field and all fields to the left of it

It sums each numeric column to the right of the control level

It places the totals in the corresponding fields of the work area

Summing can result in overflow because the total is placed into a field of the same length. Overflow causes a short dump with the error SUM_OVERFLOW. When using sum, avoid overflow by increasing the length of numeric fields before populating the internal table.

Using the on change of Statement

- Another statement you can use to perform control break processing is on change of. It behaves in a manner similar to at new.
- Syntax for the on change of Statement

The following is the syntax for the on change of statement.

on change of v1 [or v2 ...].

[else.

---]

endon.

- where:

- v1 and v2 are variable or field string names
- ... indicates that any number of or conditions might follow
- --- represents any number of lines of code



Copyright © Capgemini 2015. All Rights Reserved 64

Obsolete Statements

- Using HEADER LINE as Work Area.
- Example:

```
TYPES: BEGIN OF line,  
       num TYPE i,  
       sqr TYPE i,  
END OF line.  
DATA it_tab TYPE TABLE OF line WITH UNIQUE KEY col1 WITH HEADER LINE.  
DO 5 TIMES.  
    It_tab-num = sy-index.  
    It_tab-sqr = sy-index ** 2.  
    INSERT TABLE it_tab.  
  ENDDO.  
It_tab-sqr = 100.  
  MODIFY TABLE it_tab.  
It_tab-num = 4.  
  DELETE TABLE it_tab.
```



Copyright © Capgemini 2015. All Rights Reserved 65

Demo

- Program on control break processing



Copyright © Capgemini 2015. All Rights Reserved 66

Summary

- In this lesson, you have learnt:
 - To Define an Internal Table and understand its attributes
 - Types of Internal Tables
 - To Add, Read, Update and Delete Data from an internal Table
 - To Sort the Contents of an Internal Table
 - Control break statements on Internal Table



Copyright © Capgemini 2015. All Rights Reserved 67

Review Question

- Question 1: _____ is a field string with the same structure as a row of the body, but it can hold a single row.
- Question 2: _____ adds a single row to an internal
- Question 3: _____ variable is used to determine the number of rows in an internal table.
- Question 4: _____ statement accumulates the contents of a structure into an internal table.





ABAP/4

Module2 Lab Book

Table of Contents

Table of Contents.....	2
Lab 1-1 Introduction to ABAP Editor	4
Lab 2-1 Data Dictionary	5
Lab 3-1 String Operations	8
Lab 4-1 Internal tables	12
Lab 5-1 Advanced Internal tables.....	13

Getting Started

1.1 Overview

This lab book is a guided tour for learning SAP ABAP. It comprises of assignments to be done. Refer the demos and work out the assignments given by referring the case studies which will expose you to work with Java applications.

1.2 Setup Checklist for SAP ABAP

Here is what is expected on your machine in order to work with lab assignment.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 7 or higher.
- Memory: (1GB or more recommended)

Please ensure that the following is done:

- SAP GUI is installed
- Connection to the SAP Server is present

Lab 1-1 Introduction to ABAP Progarmming

Goals	<ul style="list-style-type: none">• Create simple programs in ABAP editor
Time	60 Minutes
Lab Setup	<ul style="list-style-type: none">• Connectivity to SAP server• Login details for connecting to SAP server

1.1. Write a program to display the Empid, EmpName, Emp_addr in chain statement using write.

Hint: All the variable should be declared using DATA statement.

1.2 .Write a program to display the value of System Date, System Time, Current User ID,Current report name by using single write statement.

1.3. Take the values of two numbers from user. If contents of both variables are negative then convert them into positive values then divide them by 2. If it is not divisible by two then truncate it and then swap them by using move statement.

1.4. Write a program to display the first 10 customers from customer master KNA1 table and consider any 10 fields to display the output.

1.5. Write a program to display the vendor master data from LFA1 table, vendors should be filtered by country code and order by city by accepting the selection screen range and consider any 10 fields to display the output.

Lab 2-1 Data Dictionary

Goals	<ul style="list-style-type: none"> Create Tables, Views and Search Helps
Time	60 Minutes
Lab Setup	<ul style="list-style-type: none"> Connectivity to SAP server Login details for connecting to SAP server

1. Create the tables with the following structure. Name of the table as z_empcode_emp.

z_nnnn_emp

Field Name	Data Element	Domain	Domain Data Type and length
EMPNO (PK)	Z_XX_DEEMPNO	Z_XX_DEMPNO	NUMC 4
ENAME	Z_XX_DEENAME	Z_XX_DENAME	CHAR 10
JOB	Z_XX_DEJOB	Z_XX_DJOB	CHAR 9
MGR	Z_XX_DEMGR	Z_XX_DMGR	NUMC 4
HIREDATE	Z_XX_DEHIREDATE	Z_XX_DHIREDATE	DATS
SAL	Z_XX_DESAL	Z_XX_DSAL	DECIMAL 7 2
COMM	Z_XX_DECOMM	Z_XX_DCOMM	DECIMAL 7 2
DEPTNO (FK)	Z_XX_DEDEPTNO	Z_XX_DDEPTNO	NUMC 2

z_nnnn_dept

Field Name	Data Element	Domain	Domain Data Type and length
DEPTNO(PK)	Z_XX_DEDEPTNO	Z_XX_DEDEPTNO	NUMC 2
DNAME	Z_XXDEDNAME	Z_XX_DEDNAME	CHAR 14
LOC	Z_XX_DELOC	Z_XX_DELOC	CHAR 13

Note: Data Element and Domains for deptno field is the same in both tables.

z_nnnn_salgrade

Field Name	Data Element	Domain	Domain Data Type and length
GRADE	Z_XX_DEGRADE	Z_XX_DGRADE	NUMC 2
LOSAL	Z_XX_DELOSSAL	Z_XX_DLOSSAL	DECIMAL 7 2
HISAL	Z_XX_DEHISAL	Z_XX_DHISAL	DECIMAL 7 2

Where:

- 1) NNNN is your empcode
- 2) XX: are the names of your initials.

This is to ensure uniqueness of your table name, domain and data element name.

2. Create an ABAP program 'Z_XX_ASSG_4A' where XX denotes Group Id of your assignment group.

1. ABAP Inputs:

The ABAP program will have parameters for employee number , employee name and salary with the definitions as per the database table fields.

2. Screen Validations

Mandatory input for all parameters. Employee number can contain only numbers. Employee name should not be blank and Salary cannot be zero.

3. Main Processing Logic :

Read the employee table created above using the Employee number, which has been input.

If matching entry exists, then check the input name and salary. If at least one of them is different than db record, then record should be modified with changed values.

If matching entry does not exist, new entry should be created in table with input values.

4. Output Format:

If record has been modified, display 'Record Updated'.

If record has been added, display 'Record added'.

5. Test Conditions

Report should be run with values for which entries exists in the db and also new values for which entries do not exist, so that both addition and modification of records can be tested..

Error checking: If employee code contains characters other then numbers, then suitable error message should be displayed. (Define Text elements for message text).

Hints

For this ABAP make use of following:

SELECT, INSERT, UPDATE.

Create a copy of the ABAP 'Z_XX_PASSG_4B' and as a variation use only MODIFY instead of INSERT/UPDATE.



CONSULTING.TECHNOLOGY.OUTSOURCING

3. Create tables and Data base views.

Step # 1: Create two ztables (Eg: ZEKKO_TAB, ZEKPO_TAB) by using predefined table fields of EKKO AND EKPO select at least 5 fields from each table and load some data into that two ztables.

Note: To enter the data/entries to ZTables Ref: EKKO and EKPO tables content/entries.

Step # 2: Maintain foreign key between the two ztables.

Step # 3: Create the data base view for the above two ztables.

Step # 4: Write an abap code and get the data from the database view (from the step # 3) and display the output.

4 Create help view and search help.

Step # 1: Create two ztables (Eg: ZVBAK_TAB, ZVBAP_TAB) by using predefined table fields of VBAK AND VBAP select at least 5 fields from each table and load some data into that two ztables.

Step # 2: Maintain foreign key between the two ztables.

Step # 3: Create the help view for the above two ztables.

Step # 4: Create the elementary search help and provide the help view name in the selection method of search help.

Step # 5: Attach the search help to the primary keys of the both the ztables.

Lab 3-1 String Operations

Goals	<ul style="list-style-type: none"> • Create an ABAP Program for String Operations
Time	90 Minutes
Lab Setup	<ul style="list-style-type: none"> • Connectivity to SAP server • Login details for connecting SAP server

1 Create an ABAP 'Z_XX_ASSG_1' where XX denotes Group Id of your assignment group.

ABAP will have following parameters.

P_STRING1 - input string. The input should be taken in LOWERCASE.

FLAGS should be displayed as radiobutton. Depending upon the value of the selected FLAG, corresponding function will be performed on the input string.

T_FLAG – to convert the string to UPPERCASE

L_FLAG – to return the length of the string

S_FLAG – to remove leading zeroes in the string

O_FLAG – to return subset of string starting from offset P_OFFSET and having length P_LENGTH.

P_OFFSET and P_LENGTH are parameters to take in the offset and length of the substring.

1. Screen Validations

Offset and Length should be numeric.

In case O_Flag not set, both offset and length must be 0.

O_Flag set, at least length must be non-zero positive value.

O_Flag set, then offset + length should be <= length of string.

S_Flag set, then string must contain leading zeroes.

2. Main Processing Logic:

Depending upon the FLAG, corresponding operation should be performed on the string.

3. Output Format:

If there is an error during validations, suitable error message should be displayed.

Or If the validations are successful, then the resultant string should be displayed after the operation is performed.

4. Test Conditions

Testing should be performed for all values of flag.

Error checking: Check out all scenarios where error will occur and check whether appropriate error message is displayed or not (Define text elements for message text).

Hints:

For this ABAP make use of following:
RADIOBUTTON GROUP, TRANSLATE, STRLEN, SHIFT.

- 2.** Create an ABAP 'Z_XX_ASSG_2' where XX denotes Group Id of your assignment group.

ABAP will have following parameters.

Define P_STRING – input string default value .
'00000000075001234, Material No: 00000000014634566, 53'.

Screen-validations:

Check if atleast 1 comma exists in the input string.

Main Processing Logic:

Split the input string at comma into 3 different variables.

Search 2nd of these variables for string 'Material No:'.

Move the Material no. into another variable.

Output Format:

Write the extracted material no. to the screen.

Hints:

Use SPLIT, SEARCH, STRLEN, commands

3. Create an ABAP 'Z_XX_ASSG_3' program where XX denotes Group Id of your assignment group.

ABAP will have following parameters.

Provide String_1 and String_2 value as a default in the program and user can change the values at runtime with own input values.

Screen-validations:

Check atleast one space should be exists in the input value of string_1 and it should be an obligatory and check String_2 value also an obligatory.

Main Processing Logic:

Accept two input strings (String_1 and String_2) from the user at runtime by using parameters and display these two string values at the list output as a heading.

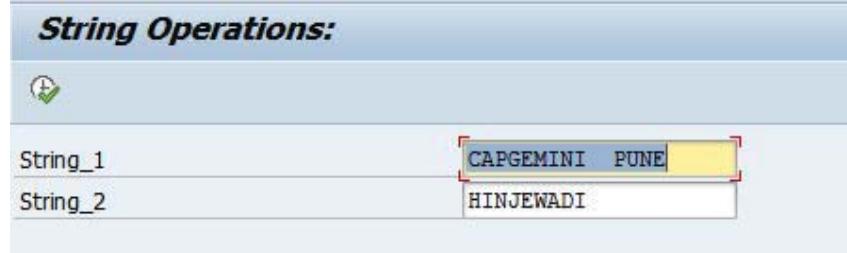
Hints:

1. Use CONCATENATE, TRANSLATE, CONDENSE, NO-GAPS, STRING LENGTH, SHIFT AND SHIFT BY PLACES commands to display the main logic.

2. Use split command by using Internal tables .

Input Format: Format of the input screen like below with proper text and title by using the text elements and list headings.

Input Screen:



String Operations:	
String_1	CAPGEMINI PUNE
String_2	HINJEWADI

Output Format: the output format should be like as shown below.

Output Screen:

String Operations:

String Operations:

Parameter String1 is: CAPGEMINI PUNE Parameter String2 is: HINJEWADI

Concatenation: CAPGEMINI PUNEHINJEWADI

Concatenation with Space: CAPGEMINI PUNE HINJEWADI

Condense with Gaps: CAPGEMINI HINJEWADI PUNE

Condense with no Gaps: CAPGEMINI HINJEWADI PUNE

The Length of the String1 title is: 15

Shift String BY 3 places left: GEMINI PUNE

Shift String BY 3 places right: CAPGEMINI PUNE

Shift String BY 3 places circular: GEMINI PUNE CAP

SPLIT By Using Internal Table:

CAPGEMINI

PUNE

Lab 4-1 Internal tables

Goals	<ul style="list-style-type: none"> Insert data into an Internal table and process it
Time	60 Minutes
Lab Setup	<ul style="list-style-type: none"> Connectivity to SAP server Login details for connecting to SAP server

- 1) Create an ABAP 'Z_XX_ASSG_2' where XX denotes Group Id of your assignment group.

ABAP will have following parameters.

P_ITEM – input string of 20 characters (select options)
 P_QTY – input number (select options)
 P_RATE – input number with 2 decimal points (select options)

1. Screen Validations

- All the inputs are mandatory.
- Qty and Rate cannot be zero.

2. Main Processing Logic:

- Accept 5 items each with qty and rate.
- Store in internal table.
- Calculate their price.
- Sort on price in ascending order.
- Display output as per section 6.

6. Output Format:

ITEM	QTY	RATE	PRICE
TOTAL			

7. Test Conditions

Testing should be performed for all options.

Error checking: Check out all scenarios where error will occur and check whether appropriate error message is displayed or not (Define Text elements for message text).

Hints

For this ABAP make use of following,
 APPEND, SUM, MODIFY

Lab 5-1 Advanced Internal tables

Goals	<ul style="list-style-type: none"> • Use control break logic to display data from Internal table and other commands. <p>To work with an Internal table commands INSERT, INSERT Multiple, UPDATE, MODIFY and DELETE.</p>
Time	60 Minutes
Lab Setup	<ul style="list-style-type: none"> • Connectivity to SAP server • Login details for connecting to SAP server

List Processing with control-break statements

- 1) Create an ABAP 'Z_XX_ASSG_11' where XX denotes Group Id of your assignment group.

1. ABAP Inputs :

ABAP will have following parameters.

- 1) Select option S_DEPTNO which is non-mandatory
- 2) parameter P_SALARY Default this with some salary amount.

2. Main Processing Logic:

The report will fetch data from employee table which you created in previous assignment.

If select-option S_DEPTNO has some values then data for only those dept.s should be fetched. Otherwise all records from table should be fetched. If P_SALARY is entered, records should be filtered on that basis. If both S_DEPTNO and P_SALARY are entered, records should be filtered based on both selection-criteria.

You need to output data for each dept. No.

The report output should be like.

DEPT No : Dept_1

Employee No. Salary

Emp1	20000
EMP2	40000

Total salary for Dept :

DEPT No. : Dept_2

Emp3 50000
Emp4 30000

Total salary for Dept :

3. Test Conditions

Testing should be carried out for different values of S_DEPTNO .

Test if proper records are selected when records should be fetched only where salary > 30000.
(You should enter p_salary > 30000 at selection screen.)

Hints

Use control-break statements.

- 2) Write an ABAP Program to create an internal table with the Following Specifications.

Field Name	Data Type	Length
Mat No	Char	18
Mat Description	Char	40
Mat Type	Char	4
Quantity	Quan	10
Price	Quan	10
Base UOM	Unit	3

Perform the below an operations on an Internal table and display the output in a list format:

- A. Insert records into an internal table.
- B. Display all the records from the table.
- C. Display the records whose price is > 20,000.
- D. Change the base unit of measure (UOM) from 'PC' to 'EA'.
- E. Delete the records where the quantity is < 10.
- F. Display the total amount.
- G. Delete the all the records.

- 3) Write an ABAP program to calculate the Grade of the students, based on the input marks.

- A. If the input Marks are > 90 - Grade A.
- B. If the input Marks are between 75 and 90 – grade B.
- C. If the input Marks are between 60 and 75 – grade C.
- D. If the input Marks are < 60 grade D.

- 4) Write an ABAP program for INSERT Records, INSERT Multiple Records, Modify and Delete the records from the Internal Table.

4.1 Create a field string with the following fields and data types for the Insert records of an internal table.

Field Name	Data Type	Length
Name	C	30
Age	I	NA
Weight	P	Decimals 3
Land	C	20

Hints:

1. Assign some hardcoded values to the internal table fields as shown the below screen.
2. Insert 3 records into an internal table.
3. Use standard or sorted tables to identify the Insert order of the records.

Output Screen:

Internal Table [Insert] By Using Sorted Table				
NAME	AGE	WEIGHT	COUNTRY	
DAVID	40	95.00	DE	
JOHAN	35	80.00	IND	
PETER	35	45.00	USA	

4.2 Create a field string (with DATA) with the following fields and data types, to Insert the multiple records of an internal table.

Field Name	Data Type
COL_1	I
COL_2	I

Hints:

1. Insert multiple lines with **Do Times** statements by using the **Sy-Index**.
2. Take the index values into COL_1 = (sy-index) and COL_2 = (2 * sy-index).
3. Loop the internal table values into Field string and multiply the COL_1 (3 * sy-tabix) and COL_2 (5 * sy-tabix) with the sy-tabix.

Output Screen:

Internal Table [Insert Multiple Lines]

TABIX	COL_1	COL_2
1	3	5
2	1	2
3	9	15
4	2	4

4.3 Create a field string (with DATA) with the following fields and data types to modify the records of an Internal table.

Field Name	Data Type
COL_1	I
COL_2	I

Hints:

1. Insert multiple lines with **Do 3 Times** statements by using the **Sy-Index**.
2. Take the index values into COL_1 (**Sy-Index**) and COL_2 (**sy-index ** 2.**).
3. Loop the internal table values into Field string and multiply the COL_1 (**sy-tabix * 10**) and COL_2 (**(sy-tabix * 10) ** 2**).
4. Modify the itab if **sy-tab index eq 2**.

Output Screen:

Internal Table [Modify]		
TABIX	COL_1	COL_2
1	1	1
2	20	400
3	3	9

4.4 Create a field string (with DATA) with the following fields and data types to delete the records of an internal table.

Field Name	Data Type
COL_1	I
COL_2	I

Hints:

1. Insert multiple lines with **Do 4 Times** statements by using the **Sy-Index**.
2. Take the index values into COL_1 = (Sy-Index) and COL_2 = (sy-index ** 2.) Insert from field string to itab.
3. Delete the internal table records, where the col_1 = 3rd row.

Output Screen:

Internal Table [Delete]	
COL_1	COL_2
1	1
2	4
4	16