TAW10

ABAP Workbench Fundamentals - Part 1

PARTICIPANT HANDBOOK
INSTRUCTOR-LED TRAINING

Course Version: 10
Course Duration: 5 Day(s)
Material Number: 50117658

Prohibida su reproduccion

SAP Copyrights and Trademarks

© 2014 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

- Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.
- IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.
- Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.
- Oracle is a registered trademark of Oracle Corporation
- UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
- Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.
- HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
- Java is a registered trademark of Sun Microsystems, Inc.
- JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
- SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.
- Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.
- Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase, Inc. Sybase is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

Prohibida su reproduccion

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation	
Demonstration	
Procedure	
Warning or Caution	
Hint	
Related or Additional Information	
Facilitated Discussion	
User interface control	*Example text*
Window title	*Example text*

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Contents

xi	Course Overview	
1	Unit 1:	SAP Portfolio Overview
2		Lesson: Identifying the Parts of the SAP Portfolio
13		Lesson: Listing the Key Capabilities of SAP NetWeaver
21		Lesson: Considering the Release Strategy of SAP
27		Exercise 1: Access the SAP Release Strategy Document
35	Unit 2:	Navigation
36		Lesson: Navigating Through SAP Systems
49		Exercise 2: Log on to the System and Navigate on the Screen
52		Lesson: Performing Advanced Navigation in the SAP GUI
57		Exercise 3: Call Functions
65	Unit 3:	The System Core
66		Lesson: Exploiting the SAP NetWeaver Application Server Architecture
72		Lesson: Distinguishing the Processes of SAP NetWeaver AS Java and ABAP
85		Exercise 4: Determine the Structure of an Instance
88		Lesson: Processing AS ABAP Requests
95		Exercise 5: List the Work Processes Configured on an Instance of AS ABAP
98		Lesson: Explaining AS ABAP Processes
107		Exercise 6: Set and Monitor Lock Entries
113		Exercise 7: Print a Screenshot in AS ABAP
117		Exercise 8: Schedule a Simple Report in the Background
129	Unit 4:	Communication and Integration Technologies
130		Lesson: Integrating Existing Applications with SAP Applications
137		Exercise 9: Use a Business Application Programming Interface
140		Lesson: Linking Cross-System Business Processes
143		Lesson: Working with Web Services on SAP NetWeaver Application Server
147		Lesson: Explaining Service-Oriented Architecture
157	Unit 5:	Flow of an ABAP Program
158		Lesson: Describing the Processing of ABAP Programs

© Copyright . All rights reserved.

Prohibida su reproduccion

173	**Unit 6:**	**ABAP Workbench Introduction**
174		Lesson: Introducing the ABAP Development Environment
184		Lesson: Organizing ABAP Developments
191		Exercise 10: Create a Package
194		Lesson: Developing Programs
205		Exercise 11: Develop a Simple ABAP Program
209		Lesson: Finalizing Development
213		Exercise 12: Create a Transaction
225	**Unit 7:**	**Basic ABAP Language Elements**
226		Lesson: Defining Elementary Data Objects
238		Lesson: Using Basic ABAP Statements
247		Exercise 13: Use Basic ABAP Statements
254		Lesson: Working with the ABAP Debugger
261		Exercise 14: Debug Statements on Elementary Data Objects
271	**Unit 8:**	**Modularization**
273		Lesson: Introducing Modularization
278		Lesson: Modularizing Using Subroutines
289		Exercise 15: Implement Subroutines
296		Lesson: Modularizing Using Function Modules
305		Exercise 16: Use a Function Module
312		Lesson: Implementing Function Modules
315		Exercise 17: Create a Function Group
321		Exercise 18: Create a Function Module
328		Lesson: Modularizing Using BAPIs
333		Lesson: Modularizing Using Global Classes
345		Exercise 19: Use a Global Static Method
351		Lesson: Implementing Simple Global Classes and Static Methods
355		Exercise 20: Create a Simple Global Class
357		Exercise 21: Create a Simple Static Method
362		Lesson: Modularizing Using Local Classes
371	**Unit 9:**	**Complex Data Objects**
372		Lesson: Working with Structures
379		Exercise 22: Work with Structures
385		Lesson: Working with Internal Tables
403		Exercise 23: Work with Internal Tables

© Copyright . All rights reserved.

Prohibida su reproduccion

413	**Unit 10:**	**Data Modeling and Data Retrieval**
415		Lesson: Modeling Data
425		Exercise 24: Analyze Transparent Tables in the Data Dictionary
431		Lesson: Reading Single Database Records
439		Exercise 25: Implement Single Record Access
445		Lesson: Reading Multiple Database Records
447		Exercise 26: Implement a SELECT Loop
455		Exercise 27: Implement an Array Fetch
461		Lesson: Handling Other Aspects of Database Access
471		Lesson: Working with Authorization Checks
477		Exercise 28: Implement an Authorization Check
489	**Unit 11:**	**Classic ABAP Report**
490		Lesson: Implementing ABAP Lists
494		Lesson: Implementing Selection Screens
503		Exercise 29: Create a Selection Screen and a Classic ABAP List
510		Lesson: Implementing Events of ABAP Reports
517		Exercise 30: Implement Events of ABAP Reports
527	**Unit 12:**	**Program Analysis Tools**
528		Lesson: Using the Code Inspector
537	**Unit 13:**	**Program Calls and Memory Management**
538		Lesson: Calling Programs Synchronously
547		Exercise 31: Implement Program Calls
554		Lesson: Understanding the ABAP Runtime and Memory Management
575		Exercise 32: Use ABAP Memory and SAP Memory
595	**Unit 14:**	**ABAP Open SQL in Detail**
596		Lesson: Implementing Complex WHERE Conditions and Special INTO Clauses
605		Lesson: Processing and Aggregating Datasets in the Database
609		Exercise 33: Retain Ordered and Condensed Datasets from the Database
623		Exercise 34: Perform Calculations on the Database
629		Lesson: Reading from Multiple Database Tables
635		Exercise 35: Implement a Join of Three Database Tables
645		Exercise 36: Implement Full Buffering and Buffering on Demand
655		Exercise 37: Read Additional Data with the FOR ALL ENTRIES addition

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Course Overview

TARGET AUDIENCE

This course is intended for the following audiences:

- Application Consultant
- Data Consultant
- Development Consultant
- Industry / Business Analyst Consultant
- Support Consultant
- Technology Consultant
- Developer
- System Administrator
- IT Support

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 1 SAP Portfolio Overview

Lesson 1

Identifying the Parts of the SAP Portfolio 2

Lesson 2

Listing the Key Capabilities of SAP NetWeaver 13

Lesson 3

Considering the Release Strategy of SAP 21

Exercise 1: Access the SAP Release Strategy Document 27

UNIT OBJECTIVES

- Determine the components of the SAP portfolio that best meet your needs
- List the key capabilities of SAP NetWeaver
- Find upgrade and maintenance information for SAP solutions

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 1
Lesson 1

Identifying the Parts of the SAP Portfolio

LESSON OVERVIEW
This lesson describes the SAP portfolio and the structure of SAP software solutions.

Business Example

Your company plans to introduce SAP software. You are a member of a project team and need an overview of the entire SAP portfolio. For this reason, you will need the following knowledge:

- An understanding of the key components of the SAP portfolio

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Determine the components of the SAP portfolio that best meet your needs

Key Areas of SAP

Figure 1: Key Areas of SAP

In order to support customers in the ever-changing business world, SAP is expanding its solution portfolio with innovations to provide instant value to customers' IT needs. While expanding the solution portfolio, SAP maintains consistency of the data, processes, and applications in the portfolio.

SAP focuses on the following key areas:

© Copyright . All rights reserved.

Prohibida su reproduccion

- Scaling the core businesses of customers (on premise)
- Innovating through extensions (on demand)
- Extending the reach of customers (on device)

SAP: On Premise, On Demand, and On Device

SAP: On Premise

SAP will continue to evolve and strengthen its core business of providing solutions for all industries and market segments.

SAP solutions scale the core businesses of customers and deliver instant value to them. To make these solutions easy to use, SAP ensures that they are preintegrated and implemented in a nondisruptive way. SAP solutions are developed using software principles such as decoupling, layering, abstraction, and design.

With SAP's approach to service-oriented architecture (SOA), third-party, non-SAP solutions are integrated consistently with the core business processes of customers.

The SAP Business Suite 7 software, one major deliverable within this area, is already available. More deliverables are planned in subsequent years.

SAP: On Demand

SAP provides rapid innovations that deliver extended, on-demand functionality for a specific line of business.

Customers can instantly use SAP's innovations without needing onsite IT to manage the infrastructure.

Because the innovations are integrated smoothly with the core business processes of customers, customers benefit from end-to-end business process integration.

SAP: On Device

SAP also aims to extend the reach of customers in the following ways:

- By enabling end users to use SAP data and processes from various devices everywhere by using SAP applications for mobile business
- By providing end users with information from both inside and outside the enterprise so that they can make decisions based on a broad array of data according to their use cases
- By helping end users cooperate and optimize performance across a dynamic business network of people
- By enabling partners, customers, and end users to extend the functionality of SAP to build user experiences

Orchestration

To ensure that all the solutions and software components fit together perfectly in the customer landscape, an orchestration layer that works across all applications is required.

SAP will continue to invest in life cycle management, master data management, and process orchestration.

© Copyright . All rights reserved.

Prohibida su reproduccion

The goal of this investment is to continuously reduce the cost of ownership and ensure data consistency and process management across multiple layers of applications.

Products for Companies of All Sizes

Figure 2: Different Products for Companies in Various Sizes

SAP offers a vast range of scalable products that can be adjusted to any size and adapted to the continuously changing processes of companies of any size.

The following SAP solutions are available to small and midsize business customers, as shown in the figure:

- SAP Business One
- SAP Business ByDesign
- SAP Business All-in-One

SAP Business One

SAP Business One is a comprehensive, integrated enterprise resource planning (ERP) application with an interface similar to Windows. The application has been developed specifically for small and midsize companies.

SAP Business One has the following features:

- SAP Business One provides simple navigation and expansion options, innovative Drag&Relate functions and integration with Microsoft Word and Excel.
- SAP Business One enables you to uniformly manage important business areas, such as sales distribution and finance, with a single integrated application. The SAP Business One application provides an up-to-date overview of the important data of a given company. This data helps companies better meet customer requirements, which is an important prerequisite for profitable growth.
- SAP Business One can be hosted locally within a company or by a partner and can be implemented within two months.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Business One is the ideal solution for small companies with around 100 employees who are looking for an affordable system to ensure the successful and profitable operation of their core processes (such as finance, sales, customer service, and operations). SAP Business One is sold exclusively through a global partner network of certified and experienced partners. These partners offer 24-hour support, which covers the enhancements from independent software providers. SAP Business One is currently being used by more than 34,000 customers as of 2012 in over 40 countries.

For more information about SAP Business One, see http://www.sap.com/solutions/sme/businessone.

SAP Business ByDesign

SAP Business ByDesign is a solution for small (around 100 employees) and midsize (100 to 500 employees) companies that want to use an on-demand solution to improve their core processes and lower the entry and operating costs. The solution is aimed at midmarket companies that are not using integrated business software.

SAP Business ByDesign has the following features:

- It combines the advantages of an integrated, end-to-end application with the benefits of an on-demand solution. Having a low risk potential and low operating costs, SAP Business ByDesign provides integrated service and support, adaptability, and user-friendly features. The solution also guarantees transparency and adheres to legal requirements.
- It is the most complete, fully integrated, on-demand solution to help midsize companies manage their finances, human resources, projects, production, and supplier and customer relationship management. It also supports the regional and local legal requirements of companies. The design of SAP Business ByDesign addresses country-specific compliance (statutory) issues.

SAP Business ByDesign was not developed merely for standard business applications or customer relationship management. Instead, the application was developed with user interfaces that can be customized by user role to support all business processes within a company. The application offers end-to-end processes for the work center of the future.

For more information about SAP Business ByDesign, see http://www.sap.com/solutions/sme/businessbydesign.

SAP Business All-in-One

SAP Business All-in-One is powered by SAP NetWeaver and includes the SAP ERP application, the SAP Customer Relationship Management (SAP CRM) application, and applications from the SAP BusinessObjects portfolio. SAP All-in-One is the brand name for vertical solutions (solutions that are industry-specific or country-specific) that are based on the SAP NetWeaver technology and developed with SAP partners.

SAP Business All-in-One has the following features:

- It allows businesses to leverage the best practices of industry leaders at an affordable fixed price. And, it is all on a platform that businesses will not outgrow.
- It can be used as local software or hosted by a partner of a company.

© Copyright . All rights reserved.

Prohibida su reproduccion

- It is based on various SAP products with a focus on SAP ERP, SAP CRM, and the SAP BusinessObjects portfolio.
- It provides a portfolio of industry-specific functions, based on SAP Best Practices.
- It can be planned easily.

More than 13,000 small and midsize customers in more than 50 countries run SAP Business All-in-One. More than 1,000 SAP partners around the world deliver local implementation and customization support for SAP Business All-in-One, giving you access to expert assistance everywhere. Preconfigured business scenarios and implementation accelerators can be put into operation in this application – within 8 to 16 weeks.

For more information about SAP Business All-in-One, see http://www.sap.com/solutions/sme/businessallinone.

SAP Solutions

The following table shows a comparison of the SAP solutions for small and midsize companies:

Parameters	SAP Business One	SAP Business ByDesign	SAP Business All-in-One
Overview	A single, integrated application to manage an entire business	Single, integrated business management solution delivered as software as a service (SaaS)	A comprehensive, integrated industry solution to power businesses end to end
Best fit for companies	For companies that have outgrown packaged accounting-only software and need to replace multiple, non integrated applications	For companies that need to replace point solutions, manual processes, and spreadsheets	**For the following companies:** • Companies that have outgrown point solutions, legacy or custom applications, or current ERP software • Companies that have current solutions that may either not meet industry-specific requirements or that lack cross-

© Copyright . All rights reserved.

Prohibida su reproduccion

Parameters	SAP Business One	SAP Business ByDesign	SAP Business All-in-One
			border functionality • Companies that are experiencing increasing business process complexity and need systems that can scale to meet growth plans
Company size	Up to 100	100–500	Up to 2,500
IT staff and capabilities	Have limited or outsourced IT resources	Have limited or outsourced IT resources	Have in-house IT staff
Deployment model	On premise or hosted by a partner	On demand	On premise or hosted by a partner

Overview of SAP Business Suite

The SAP software portfolio is constantly extended, optimized, and tailored to the needs of the market. SAP has developed many industry-specific applications over the years and added more products for small-to-midsize companies to the software portfolio. SAP enables integration between various applications (modules) from different vendors and across company boundaries.

SAP Business Suite applications are an extended family of business applications that enable companies to manage their entire value chains. These applications provide users with consistent results throughout the network of a company and the flexibility needed to address dynamic market situations. The applications also support cross-company processes.

SAP Business Suite provides the following evolved features:

- A complete spectrum of business solutions
- A technological infrastructure that combines openness and flexibility with maturity and stability
- Interfaces for integrating non-SAP products
- Components that can be adapted to meet multiple business requirements
- Numerous industry-specific functions

SAP Business Suite plays a central role in businesses. Many companies are already profiting from the comprehensive and flexible business applications in SAP Business Suite because of their highly evolved functions. These functions include complete integration, industry-typical functions, unlimited scalability, and smooth collaboration over the Internet.

© Copyright . All rights reserved.

Prohibida su reproduccion

For more information about SAP Business Suite, see http://www.sap.com/solutions/business-suite.

Architecture of SAP Business Suite

Figure 3: SAP Business Suite – Architecture

Refer to the figure for the components in the architecture of SAP Business Suite.

As shown in the figure, the core applications of SAP Business Suite are SAP ERP, SAP CRM, SAP Product Lifecycle Management (SAP PLM), SAP Supplier Relationship Management (SAP SRM), and SAP Supply Chain Management (SAP SCM). These applications support all the essential business processes of a company.

SAP ERP

The SAP ERP application is a market-leading application for optimizing business and IT by reducing IT complexity, increasing adaptability, and delivering more IT value at a lower cost than the traditional ERP solutions. It supports mission-critical, end-to-end business processes for finance, human capital management, asset management, sales, procurement, and other essential corporate functions.

SAP ERP also supports industry-specific processes by providing industry-specific business functions that can be activated selectively by using the switch framework. This framework keeps the application core stable and helps to ensure maximum performance.

SAP Customer Relationship Management

The SAP CRM application provides a comprehensive platform for marketing, sales, and service professionals to obtain complete customer intelligence that they can leverage to manage customers and customer-related processes effectively.

SAP CRM enables multichannel customer interactions, including mobile smart phones, the Internet, and social media. It also offers a dedicated communications infrastructure that helps to connect all users anytime, anywhere.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Product Lifecycle Management, SAP Supplier Relationship Management, and SAP Supply Chain Management

SAP Product Lifecycle Management

The SAP PLM application helps companies manage, track, and control all product-related information over the complete product and asset lifecycle and throughout the extended supply chain.

SAP PLM facilitates creativity and frees the process of product innovation from organizational constraints.

SAP Supplier Relationship Management

The SAP SRM application provides a procurement platform that helps organizations in all industries improve their centralized sourcing and contract management and enables them to interact with suppliers through multiple channels.

SAP SRM accelerates and optimizes the entire end-to-end process of procure-to-pay by supporting integrated processes and enforcing contract compliance. Integrated processes and better contract compliance typically result in dramatic savings.

SAP Supply Chain Management

The SAP SCM application allows companies to adapt their supply chain processes to a changing, competitive environment.

SAP SCM transforms traditional supply chains from linear, sequential processes into open, configurable, and responsive supply networks. In such networks, customer-centric, demand-driven companies can sense and respond more intelligently and quickly to dynamics of supply and demand across a globally distributed environment.

Industry Applications

SAP addresses the requirements of specific business processes for many industries by complementing the basic business processes that are common to all large enterprises.

Support for these industry-specific processes is delivered as part of SAP ERP or as a separate industry application, such as the SAP Dealer Business Management application, which integrates with the other applications of SAP Business Suite. The architecture and business functionality of the industry applications result from SAP's in-depth knowledge and understanding of the industry-specific business requirements and business processes. SAP industry portfolios are continuously enhanced by adding new applications that address the highly specialized business needs of customers in target markets.

The following table lists some of the industry applications that SAP offers:

© Copyright . All rights reserved.

Prohibida su reproduccion

Industry Sector	Industry Portfolio
Discrete	Industries SAP for Aerospace and Defense SAP for Automotive SAP for Engineering, Construction, and Operations SAP for High Tech SAP for Industrial Machinery and Components
Process Industries	SAP for Life Sciences SAP for Mill Products SAP for Mining SAP for Oil and Gas SAP for Chemicals
Consumer Industries	SAP for Consumer Products SAP for Retail SAP for Wholesale Distribution
Service Industries	SAP for Media SAP for Professional Services SAP for Telecommunications SAP for Transportation and Logistics SAP for Utilities
Public Services	SAP for Defense and Security SAP for Healthcare SAP for Higher Education & Research SAP for Public Sector
Financial Services	SAP for Banking SAP for Insurance

Supplementary Applications and Enhancement Packages

Supplementary applications deliver value, appeal to specialized business users, and offer a high degree of flexibility in a process.

Supplementary applications drive specialized business processes common to a large number of industries, such as manufacturing applications, SAP solutions for auto-ID and item serialization, and applications for mobile business.

SAP NetWeaver

© Copyright . All rights reserved.

Prohibida su reproduccion

The SAP NetWeaver technology platform provides a reliable, secure, and scalable foundation to run business applications like SAP Business Suite and SAP BusinessObjects applications.

SAP NetWeaver ensures that large enterprises can perform mission-critical business processes.

As the technical foundation for Service-Oriented Architecture (SOA), SAP NetWeaver delivers a comprehensive set of middleware functions in a modular software environment with the aim of reducing IT complexity and increasing business flexibility across heterogeneous IT landscapes. SAP NetWeaver provides IT organizations with the lowest cost of operation and best business availability for SAP applications across heterogeneous IT landscapes through unified lifecycle management, identity management, secure communications, and end-to-end monitoring.

Enhancement Packages

SAP also has a proven method for continually delivering updates to SAP Business Suite and SAP NetWeaver without disrupting a company's operations. SAP provides enhancement packages with collections of new or improved business functions that companies can deploy in a modular fashion and on their own schedule.

Enhancement packages enable companies to take advantage of ongoing innovations while keeping their core software stable. The packages were introduced with the SAP ERP 6.0 application. Several enhancement packages have been released since 2006. All core applications of SAP Business Suite are now enabled for continuous innovation through enhancement packages.

In the future, SAP will continue to deliver enhancement packages for the core applications, which will minimize the need for potentially disruptive upgrade projects in companies.

SAP Applications and Components

Figure 4: SAP Applications and Components

SAP Business Suite provides numerous applications to address business challenges. However, the submodules of many applications have similar or identical requirements for business functions. Different applications, therefore, contain similar software components in parts. A

© Copyright . All rights reserved.

Prohibida su reproduccion

component is the smallest, separately producible, deliverable, installable, and maintainable software unit. Examples of components are SAP ERP Central Component (ECC), SAP SCM, and SAP NetWeaver Portal.

Refer to the figure for an overview of SAP applications and components.

LESSON SUMMARY

You should now be able to:

- Determine the components of the SAP portfolio that best meet your needs

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 1
Lesson 2

Listing the Key Capabilities of SAP NetWeaver

LESSON OVERVIEW
This lesson explains the central role of the SAP NetWeaver Application Server (SAP NetWeaver AS) in SAP NetWeaver and the key capabilities of SAP NetWeaver.

Business Example

As an administrator, you want to find out more about SAP NetWeaver and the role that the SAP NetWeaver AS plays in SAP NetWeaver. For this reason, you will need the following knowledge:

- An understanding of the key capabilities of SAP NetWeaver
- An understanding of how to access the technology map for SAP NetWeaver
- An understanding of the use cases for SAP NetWeaver

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- List the key capabilities of SAP NetWeaver

SAP NetWeaver Technology Map – Capability View

The SAP NetWeaver technology platform is a reliable, secure, and scalable foundation for running business applications, such as SAP Business Suite and SAP BusinessObjects applications. It also enables large enterprises to perform mission-critical business processes.

SAP NetWeaver provides the following benefits:

- As the technical foundation for Service-Oriented Architecture (SOA), SAP NetWeaver delivers a comprehensive set of middleware functions in a modular software environment. These functions help reduce IT complexity and increase business flexibility across heterogeneous IT landscapes.
- SAP NetWeaver provides IT organizations with the lowest cost of operation and best business availability for SAP applications across heterogeneous IT landscapes. Low-cost operations and business availability are made possible through unified lifecycle management, identity management, secure communications, and end-to-end monitoring in SAP NetWeaver.
- SAP NetWeaver provides the flexibility to integrate and extend the business processes that run across SAP, SAP-certified partners, and custom-built applications. This flexibility comes from the prebuilt integration content and enterprise services and the model-driven tools that support rapid deployment in SAP NetWeaver.

© Copyright . All rights reserved.

Prohibida su reproduccion

- SAP NetWeaver supports business process management, which helps monitor the efficiency, integrity, and security of mission-critical business processes. Business users can also use SAP NetWeaver to define business rules to ensure consistency across a business network.
- SAP NetWeaver integrates and connects people, information, and business processes across technologies and companies. It enables companies to adjust to changes quickly. SAP NetWeaver ensures that a company's crucial business processes are reliable, safe, and scalable. In addition, SAP NetWeaver enables companies to maximize the benefits from their current software and systems.
- SAP NetWeaver consolidates nonuniform technologies and provides predefined business content, which reduces the manual effort involved in maintaining this content. SAP NetWeaver is based on a technology that uses industry standards and can be enhanced using popular development tools.

Capability-Based Technology Map of SAP NetWeaver

Figure 5: Capability-Based Technology Map of SAP NetWeaver (Edition 2010)

The technology map of SAP NetWeaver has a simplified structure, as shown in the figure. The structure makes it easy to reference SAP NetWeaver capabilities by using application scenarios while supporting a more intuitive drilldown procedure for the components of SAP NetWeaver. The technology map is not linked to a specific SAP NetWeaver product or release but contains all the capabilities provided by the SAP NetWeaver platform.

SAP NetWeaver has various capabilities, such as user productivity, business intelligence (BI), and business process composition, as shown in the figure.

SAP NetWeaver Capabilities - User Productivity, BI, Business Process Composition, and EIM

User Productivity

SAP NetWeaver can help users and groups improve their productivity by providing enhanced collaboration, optimized knowledge management, and intuitive search in business objects and unstructured content.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP NetWeaver can provide personalized access to mission-critical applications and data by using portals, desktop clients, and mobile interfaces. The flexible UI technology of SAP NetWeaver enables IT organizations and partners to build their own state-of-the-art applications.

Business Intelligence

The Business Intelligence (BI) solutions of SAP provide comprehensive BI functionality that can empower users to make effective, informed decisions based on data and data analysis.

The BI solutions provide access to information to all users, from the high-end analyst to the casual business user.These solutions require minimal dependence on IT resources and developers and enable users to access, format, analyze, navigate, and share information across an organization.

Business Process Composition

Business process composition enables users to design, model, implement, run, monitor, operate, and improve business processes.

Customers who use SAP NetWeaver can focus on the model-driven development of composite applications instead of the traditional code-based approach to developing these applications.

Enterprise Information Management

Enterprise information management (EIM) is the business activity of creating, cleansing, integrating, managing, governing, and archiving the structured and unstructured data used by an organization.

The EIM activities are supported by the SAP product families:

- Enterprise data warehouse (EDW) management
- Master data management
- Data integration and data quality management
- Metadata management
- Information lifecycle management
- Enterprise content management

EIM can help organizations integrate data and reconcile business information in heterogeneous IT landscapes and business networks that are cross-company.

Data integration and reconciliation, in turn, help create a trusted data foundation for operational, analytical, and governance initiatives. SAP delivers solutions from different product families, including SAP NetWeaver, SAP BusinessObjects EIM, and SAP solution extensions.

By managing information assets effectively, organizations can minimize data integration efforts, streamline end-to-end business process execution, and gain well-founded business insight.

SAP NetWeaver Capabilities – SOA Middleware, Custom Development, Security and Identity Management, and Application Lifecycle Management

Service-Oriented Architecture (SOA) Middleware

SAP NetWeaver enables IT organizations to use standards-based web services for forming new, innovative business solutions that meet changing business needs. In particular, SAP NetWeaver

© Copyright . All rights reserved.

Prohibida su reproduccion

provides SOA middleware that facilitates communication between disparate applications. From a technical view, SOA middleware consists of an enterprise services repository and registry, an Enterprise Services Bus, and SOA management tools.

With SOA middleware, IT professionals can rely on an open and standards-based platform to accelerate business integration.

Custom Development

Using SAP NetWeaver, IT professionals can extend or enhance existing applications.

The professionals can create custom applications using ABAP or Java programming languages.

Security and Identity Management

Organizations face a difficult challenge in the security-conscious world today. They have to support transparent enterprise boundaries, location-independent users, and the growing demands for regulatory compliance.

With SAP NetWeaver, IT organizations can introduce safeguards that protect their business, while helping users and allowing business processes to proceed unhindered by security operations.

Application Lifecycle Management

Application lifecycle management provides processes, tools, services, and an organizational model to manage SAP and non-SAP solutions throughout the complete application life cycle.

Instead of just focusing on individual phases, SAP provides a holistic approach. Results of a specific phase can be leveraged by other phases due to the integration provided by SAP Solution Manager. This type of integration helps implement and operate IT solutions more quickly and at lower costs.

Mapping Table – SAP NetWeaver Technology Maps

The following table provides a comparison between the SAP NetWeaver Technology Map editions 2007 and 2010:

SAP NetWeaver Technology Map – Edition 2007	SAP NetWeaver Technology Map – Edition 2010
User productivity enablement	User productivity
Data unification	Information management
Business information management	Information management
Business event management	Business process management
End-to-end process integration	Business process management, SOA middleware
Custom development	Composition and application development
Unified lifecycle management	Lifecycle management
Application governance and security management	Security and identity management

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP NetWeaver Technology Map – Edition 2007	SAP NetWeaver Technology Map – Edition 2010
Consolidation	Lifecycle management, information management
Enterprise SOA design and deployment	SOA middleware

SAP NetWeaver Application Server

Figure 6: SAP NetWeaver AS – The Basis for SAP Systems

Many SAP systems (including SAP ERP Central Component, SAP NetWeaver Portal, SAP CRM, and SAP SRM) are based on SAP NetWeaver AS and use it as a runtime environment.

Together with a database, SAP NetWeaver AS is the application platform of SAP NetWeaver.

© Copyright . All rights reserved.

Prohibida su reproduccion

Properties of SAP NetWeaver Application Server

Figure 7: SAP NetWeaver Application Server

SAP NetWeaver AS is the result of the logical development of the technology of the SAP Application Server (previously known as SAP Basis), where special attention is paid to web-based applications.

SAP NetWeaver AS offers the following features:

- A reliable and extensively tested runtime environment, which has been developed over a period of 10 years
- A framework for executing complex business processes that meet high security standards
- A reliable and user-friendly development environment
- Support for open standards, such as HTTP, HTTPS, SMTP, WebDAV, SOAP, SSL, SSO, X. 509, Unicode, HTML, XML and WML, WS Policy, SOAP, WS Trust, Java SE, JSR, WSRP, and SAML
- High scalability
- Support for different operating system and database platforms

The applications delivered by SAP do not always require both the ABAP and Java runtime environments. There are different installation options for SAP NetWeaver AS.

Installation Options for SAP NetWeaver Application Server

The following installation options are available for SAP NetWeaver AS:

© Copyright . All rights reserved.

Prohibida su reproduccion

- SAP NetWeaver AS ABAP

  Complete infrastructure in which ABAP-based applications can be developed and used

- SAP NetWeaver AS Java

  Complete infrastructure in which Java 2 Enterprise Edition (J2EE)-compliant applications can be developed and used

- SAP NetWeaver AS ABAP + Java (dual stack)

  Complete infrastructure in which ABAP-based and J2EE-based applications can be developed and used

SAP NetWeaver – Use Cases and Capabilities

The following table provides the main use cases for SAP NetWeaver 7.3x and the capabilities of the application that map to the use cases:

Use Case	Corresponding Capability of SAP NetWeaver
Building composite applications	SAP NetWeaver Composition Environment
Data warehousing	SAP NetWeaver Business Warehouse
Integrating and self-servicing content with the SAP NetWeaver portal	SAP NetWeaver Process Integration
Building integration scenarios	SAP NetWeaver Portal
Mobilizing business processes	SAP NetWeaver Mobile
Custom application development (ABAP)	All areas that allow ABAP development on AS ABAP

The use cases illustrate how you can use your SAP NetWeaver implementation.

> Note:
> SAP NetWeaver Mobile is available only as an add-on to the EHP for SAP NetWeaver 7.3. For more information, see SAP Note 1645275.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP NetWeaver Functionality

Figure 8: SAP NetWeaver - Functionality

SAP NetWeaver provides the following functionality, as shown in the figure:

- Composition environment
- Process integration
- Application lifecycle management
- Security and identity management
- Application platform
  - ABAP Server
  - Java Server

> Note:
> For more information, see the Master Guide SAP NetWeaver 7.3 EHP1 on SAP Service Marketplace at http://service.sap.com/instguides. On SAP Service Marketplace, choose *SAP NetWeaver → SAP NetWeaver 7.3 → Installation → Master Guide for SAP Enhancement Package 1 for SAP NetWeaver 7.3.* Alternatively, see SAP Developer Network at http://www.sdn.sap.com/irj/sdn/nw-73.

LESSON SUMMARY

You should now be able to:

- List the key capabilities of SAP NetWeaver

© Copyright . All rights reserved.

Prohibida su reproduccion

Considering the Release Strategy of SAP

LESSON OVERVIEW
This lesson explains the different phases that a typical SAP solution will go through until its next upgrade to a higher release. It also explains the maintenance duration that SAP guarantees for its products.

Business Example

You want up-to-date information about the availability of the releases of SAP solutions to make a good decision about the purchase or upgrade of an SAP solution. For this reason, you will need the following knowledge:

- An understanding of availability, maintenance, and update information
- An understanding of how to access the SAP release strategy document

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Find upgrade and maintenance information for SAP solutions

Availability and Maintenance Strategy of SAP Solutions

Figure 9: Phases of Software Delivery by SAP

© Copyright . All rights reserved.

Prohibida su reproduccion

The SAP release and maintenance strategy determines the availability of new SAP software releases (including releases or enhancement packages of SAP Business Suite, SAP NetWeaver, and the SAP Business Objects portfolio). The strategy also determines the length and conditions of the maintenance of releases and the dependencies between the releases.

Shipments of SAP releases and the corresponding enhancement packages are introduced into the market in two or three shipment phases, such as beta, restricted, and unrestricted.

SAP enhancement packages contain new developments and application enhancements for products like SAP Business Suite, SAP NetWeaver, and SAP Business Objects portfolio.

Beta and Restricted Shipment Phases

Beta Shipment Phase

In some cases, SAP offers a beta shipment before the restricted shipment phase (which is before the release to customer date). Beta shipments are used for testing purposes only.

They are not available for productive use.

Restricted Shipment Phase

New software releases are available to only a restricted number of customers. The release to customer date marks the start of the restricted shipment phase.

During this phase, the new software release passes through a standardized process called the ramp-up. Ramp-up is a name given to the process of introducing new products or new releases of SAP software in the market.

The SAP software can be used for a productive implementation.

During the ramp-up, the software is available to only a limited circle of customers, who applied to SAP to participate.

The ramp-up serves to take customer wishes into consideration before the respective SAP software is made available.

Participants in the ramp-up are given direct contact to the SAP development environment and are provided with wide-ranging support in implementing the SAP software.

A ramp-up takes about 6 months.

> Note:
> For more information about the ramp-up, see the SAP Service Marketplace at the quick link */ramp-up*.

Unrestricted Shipment Phase

With the successful conclusion of the SAP ramp-up program, the unrestricted shipment phase of an SAP release begins on its default release date. After the default release date, all customers can request the new software without restrictions.

The latest available version of SAP software is called a default release. When SAP customers order software from SAP, they are given the current default release.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Maintenance Strategy

Figure 10: SAP Maintenance Strategy

SAP supports your maintenance contract through the SAP Service Marketplace. The contract includes support packages, which solve component-dependent problems that arise when using SAP software. SAP provides these support packages at certain intervals.

> Note:
> The duration of mainstream maintenance might be different for some software products from SAP. For example, some core applications of SAP Business Suite have a significantly increased duration of mainstream maintenance.

A 5-1-2 maintenance strategy was introduced in 2004 to support aligned maintenance phases between SAP releases. To continue providing a strong, reliable support structure for the leading solutions of the industry, SAP extended its 5-1-2 strategy to 7-2, which is seven years of mainstream maintenance, with an additional option for two years of extended maintenance.

Some applications, such as SAP Business One and composite applications, do not follow the maintenance strategy rules. They have a minimum mainstream maintenance duration of two years and do not offer extended maintenance.

As a part of SAP Enterprise Support, customers receive extended maintenance for selected releases automatically and without an additional fee. Special rules apply to content.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Business Suite 7 Maintenance Extension

Figure 11: SAP Business Suite 7 Maintenance Extension

For more information, see SAP Note 1648480.

Types of Maintenance

The following types of maintenance are available from SAP:

- Mainstream maintenance

  Mainstream maintenance starts from the release to customer date and ends after seven years. This maintenance covers the restricted and unrestricted shipment phases.

- Extended maintenance

  Customers can extend the support period in which corrections or legal changes are taken into account for an increased maintenance fee.

  The scope of support for the entire extended maintenance period is similar to the scope of support provided during mainstream maintenance. SAP publishes information on the availability and scope of extended maintenance in advance.

- Customer-specific maintenance

  You do not adjust the release to meet external requirements after a certain point in time during the lifecycle of a release. Such requirements include the implementation of legal changes or the support of new technologies. The end of extended maintenance or the end of mainstream maintenance marks the time in question, as long as no extended maintenance is offered. Customer-specific maintenance starts at this time.

A release automatically enters into customer-specific maintenance. There is no need to apply for an additional contract. During this phase, customers receive support services similar to those offered in the mainstream maintenance phase, but with some additional restrictions. For example, SAP does not deliver new support packages or updates to cover legal changes, and technology updates are limited. Resolution of problems is customer-specific, which means that customers are charged for solutions to problems that are not yet known to SAP.

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
See SAP Service Marketplace, quick link /maintenance, for additional information on the maintenance strategy. See SAP Note 52505 for details about customer-specific maintenance.

Upgrade

SAP provides upgrade paths for SAP releases and the corresponding content releases. SAP also provides access to upgrade procedures for the current release. Usually, customers can upgrade directly from one release to another release as long as the releases occur during the mainstream maintenance phase. Depending on the technological constraints, an upgrade to a release that is several releases away from a customer's current release may have to be performed in more than one step.

To facilitate smooth upgrades, SAP provides a sufficient overlap of two successive releases to make sure that the time in which customers may upgrade from one release to the next release is at least one year. For content releases, the overlap is at least six months.

Note:
For more information about upgrades, see SAP Service Marketplace, quick link / *upgrade*.

History of Selected Software Components

Remark	"Basis" Functionality AS = Application Server	Business Functionality ECC = ERP Central Component	Remark
Part of SAP NetWeaver 7.3 Enhancement Package 1	SAP NetWeaver AS 7.31		
Part of SAP NetWeaver 7.3	SAP NetWeaver AS 7.30		
Used by • SAP NetWeaver CE • SAP NetWeaver Enterprise Search	SAP NetWeaver AS 7.20		
Used by • SAP NetWeaver PI • SAP NetWeaver CE • SAP NetWeaver Mobile	SAP NetWeaver AS 7.10, 7,11		
Part of SAP NetWeaver 7.0 Enhancement Package 3	SAP NetWeaver AS 7.03	SAP ECC 6.06	Part of SAP ERP 6.0 Enhancement Package 6
Part of SAP NetWeaver 7.0 Enhancement Package 2	SAP NetWeaver AS 7.02	SAP ECC 6.05	Part of SAP ERP 6.0 Enhancement Package 5
Part of SAP NetWeaver 7.0 Enhancement Package 1	SAP NetWeaver AS 7.01	SAP ECC 6.04	Part of SAP ERP 6.0 Enhancement Package 4
Part of SAP NetWeaver 7.0 (=SAP NetWeaver 2004s)	SAP NetWeaver AS 7.00	SAP ECC 6.01, 6.02, 6.03	Part of SAP ERP 6.0 Enhancement Package 1, 2, 3
		SAP ECC 6.00	Part of SAP ERP 6.0 (=SAP ERP 2005)
Part of SAP NetWeaver 2004	SAP Web AS 6.40	SAP ECC 5.00	Part of mySAP ERP 2004
"prior to SAP NetWeaver"	SAP Web AS 6.20	SAP R/3 Enterprise 4.7	
	SAP Web AS 6.10		
	SAP Basis 4.6D		
	SAP Basis 4.6C	SAP R/3 4.6C	

Figure 12: Evolution of the Basis and Business Functionality

© Copyright . All rights reserved.

Prohibida su reproduccion

The figure shows the historical development of the following components:

- The basis technology (*SAP Basis → SAP Web AS → SAP NetWeaver AS*)
- The core business functionality (*SAP R/3 → SAP R/3 Enterprise → SAP ERP Central Component*)

Technical Basis (Application Server) in SAP

When SAP offered two products (SAP R/2 and SAP R/3), the development of the technical basis was closely linked to application development. The release names of SAP Basis corresponded to the SAP R/3 version. For example, SAP Basis 4.0B was the technical basis for SAP R/3 4.0B.

The SAP portfolio grew significantly as time went on. New products were created, and they required more frequent changes and enhancements of SAP Basis than SAP R/3. This marked the transition from the classic SAP Basis (last version SAP Basis 4.6D) to SAP Web Application Server (SAP Web AS). New Internet technologies (Internet Communication Manager from SAP Web AS 6.10 onwards) and the supplementing of the classical ABAP environment with Java or JEE (from SAP Web AS 6.20 onwards) were important milestones.

SAP Web AS 6.40 formed the technical basis (also known as “application platform”) of SAP NetWeaver 2004. SAP NetWeaver offers extensive capabilities, such as the SAP NetWeaver Business Warehouse (SAP NetWeaver BW) component, which are all based on the application platform. From SAP NetWeaver 7.0 (previously SAP NetWeaver 2004s), the product names and releases were adapted further. SAP NetWeaver 7.0 is now based on SAP NetWeaver AS.

While the releases 7.1x and 7.20 of SAP NetWeaver AS are used only with certain components (such as SAP NetWeaver Process Integration (PI) and SAP NetWeaver Composition Environment (CE)), SAP NetWeaver 7.3 is a full shipment of the SAP NetWeaver technology stack.

Central ERP Functions

With the SAP R/3 releases, the technical basis and application development were interlinked, up to and including SAP R/3 4.6C.

With SAP R/3 Enterprise (4.7), which is based on SAP Web AS 6.20, the concept of SAP R/3 Enterprise Extensions was introduced.

The central application (previously known as a “solution”) of SAP Business Suite is SAP ERP. The central software component of SAP ERP is SAP ERP Central Component (ECC). SAP ECC 5.00 can be considered the technical successor of SAP R/3 Enterprise and is based on SAP Web AS 6.40. At the time this document was created, the current version was SAP ERP 6.0 (previously: SAP ERP 2005), which includes SAP ECC 6.00. SAP ECC 6.00 operates on SAP NetWeaver AS 7.00 and other components. Enhancement packages make functional enhancements available for different software components.

SAP NetWeaver AS 7.1x or 7.20 is not used as the technical basis for an SAP ERP Central Component system. While other SAP NetWeaver components, such as SAP NetWeaver Process Integration (PI) and SAP NetWeaver Composition Environment (CE), require this SAP NetWeaver AS release level.

© Copyright . All rights reserved.

Prohibida su reproduccion

Access the SAP Release Strategy Document

Business Example

You want to determine the end of extended maintenance for SAP enhancement package 6 for SAP ERP 6.0.

Use the SAP Service Marketplace to find the end of extended maintenance of SAP enhancement package 6 for SAP ERP 6.0.

> Hint:
> This exercise requires a registered user on the SAP Service Marketplace. If you do not have access to your S-user, skip this exercise.

1. When does the extended maintenance for SAP enhancement package 6 for SAP ERP 6.0 end?

© Copyright . All rights reserved.

Prohibida su reproduccion

Access the SAP Release Strategy Document

Business Example

You want to determine the end of extended maintenance for SAP enhancement package 6 for SAP ERP 6.0.

Use the SAP Service Marketplace to find the end of extended maintenance of SAP enhancement package 6 for SAP ERP 6.0.

> Hint:
> This exercise requires a registered user on the SAP Service Marketplace. If you do not have access to your S-user, skip this exercise.

1. When does the extended maintenance for SAP enhancement package 6 for SAP ERP 6.0 end?
   a) Navigate to the SAP Service Marketplace, quick link or release strategy. See https://service.sap.com/releasestrategy.
   b) Choose *SAP's Release Strategy for Large Enterprises*. Open the document named *SAP's Release Strategy for Large Enterprises for all Major Releases from 2005 onward*.
   c) Within the document, search for *SAP ERP*. One of the hits will lead you to the table giving the SAP ERP details. Extended maintenance for SAP EHP 6 for SAP ERP 6.0 will end December 2020.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Find upgrade and maintenance information for SAP solutions

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Non-SAP solutions can be integrated consistently with the customers' core business processes through _____.

   Choose the correct answer.

   - ☐ A A service-oriented architecture(SOA)
   - ☐ B SAP Customer Relationship Management (CRM)
   - ☐ C SAP ERP
   - ☐ D SAP NetWeaver

2. SAP Business All-in-One is the ideal solution for small and midsize companies (having up to 2,500 employees) with stable processes and that want to use a preconfigured system from SAP.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

3. SAP ERP does not support industry-specific processes.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

4. Identify the key capabilities of SAP NetWeaver.

   Choose the correct answers.

   - ☐ A Service-oriented architecture (SOA) middleware
   - ☐ B Security and Identity management
   - ☐ C Business intelligence

© Copyright . All rights reserved.

Prohibida su reproduccion

5. Identify the different installation options for SAP NetWeaver Application Server (AS).

   Choose the correct answers.

   - ☐ A SAP NetWeaver AS ABAP
   - ☐ B SAP NetWeaver AS Java
   - ☐ C SAP NetWeaver AS ABAP + Java
   - ☐ D SAP NetWeaver AS C++

6. In the customer-specific maintenance phase, Support Packages are delivered to customers to solve system and customer issues.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

7. _____ are used to solve (component-dependent) problems that arise when using SAP software.

   Choose the correct answer.

   - ☐ A Support packages
   - ☐ B Extended maintenance components
   - ☐ C Central ERP functions

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Non-SAP solutions can be integrated consistently with the customers' core business processes through _____.

   Choose the correct answer.

   - [x] A A service-oriented architecture(SOA)
   - [ ] B SAP Customer Relationship Management (CRM)
   - [ ] C SAP ERP
   - [ ] D SAP NetWeaver

2. SAP Business All-in-One is the ideal solution for small and midsize companies (having up to 2,500 employees) with stable processes and that want to use a preconfigured system from SAP.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

3. SAP ERP does not support industry-specific processes.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Identify the key capabilities of SAP NetWeaver.

   Choose the correct answers.

   [X] A Service-oriented architecture (SOA) middleware

   [X] B Security and Identity management

   [X] C Business intelligence

5. Identify the different installation options for SAP NetWeaver Application Server (AS).

   Choose the correct answers.

   [X] A SAP NetWeaver AS ABAP

   [X] B SAP NetWeaver AS Java

   [X] C SAP NetWeaver AS ABAP + Java

   [ ] D SAP NetWeaver AS C++

6. In the customer-specific maintenance phase, Support Packages are delivered to customers to solve system and customer issues.

   Determine whether this statement is true or false.

   [ ] True

   [X] False

7. _____ are used to solve (component-dependent) problems that arise when using SAP software.

   Choose the correct answer.

   [X] A Support packages

   [ ] B Extended maintenance components

   [ ] C Central ERP functions

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 2 Navigation

Lesson 1

Navigating Through SAP Systems 36
Exercise 2: Log on to the System and Navigate on the Screen 49

Lesson 2

Performing Advanced Navigation in the SAP GUI 52
Exercise 3: Call Functions 57

UNIT OBJECTIVES

- Navigate through SAP systems by using the SAP GUI
- Call functions in ABAP-based systems

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 2
Lesson 1

Navigating Through SAP Systems

LESSON OVERVIEW
This lesson provides an overview of the navigation options in SAP systems. You will learn basic navigation commands by using the SAP Graphical User Interface (GUI) for Windows.

Business Example

You want to learn about the available options for your users to access SAP systems through the SAP GUI. For this reason, you require the following knowledge:

- An understanding of user interfaces in the SAP environment
- How to log on to the system
- How to work with the screen structure
- An understanding of SAP NetWeaver Business Client

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Navigate through SAP systems by using the SAP GUI

User Interfaces in the SAP Environment

Most SAP applications are based on SAP NetWeaver AS ABAP. Examples of such applications are SAP Customer Relationship Management (CRM) and SAP Supply Chain Management (SCM).

Some SAP applications and components, such as SAP NetWeaver Portal, SAP NetWeaver Process Integration (SAP NetWeaver PI), and SAP NetWeaver Composition Environment (SAP NetWeaver CE), are based on SAP NetWeaver AS Java.

Some applications, such as SAP Business One and the SAP StreamWork application, and solutions like SAP Crystal solutions are not based on SAP NetWeaver Application Server (SAP NetWeaver AS) at all.

> Note:
> Some SAP solutions, such as SAP Solution Manager, are based on AS ABAP + Java.
>
> Depending on their usage, SAP solutions use separate user interfaces.

Different Access Technologies

Certain access technologies are frequently used by end users of the SAP system, but are rarely used by technical users.

© Copyright . All rights reserved.

Prohibida su reproduccion

The following table lists examples of access technologies:

SAP Product	Typical Access Technology
SAP ERP	SAP GUI for Windows or SAP GUI for HTML
SAP NetWeaver Business Intelligence (BI)	Browser-based interface (various browsers possible)
SAP Business ByDesign	Browser-based interface (various browsers possible), software as a service (SaaS)
SAP Business All-in-One solutions	SAP NetWeaver Business Client

Note:
In this course, we focus on SAP NetWeaver and its technology.

This list does not cover all SAP products. It does not also cover all possible access technologies. For example, a system using SAP ERP can be accessed through your browser or SAP GUI for Java. SAP NetWeaver BI is also accessible by SAP GUI for Windows.

The presentation interface, SAP GUI, enables a user to interact with SAP systems based on AS ABAP. SAP GUI uses the functions provided by the relevant presentation environment. The SAP GUI is based on the Windows Style Guide and is available to several platforms. It provides the same functions for each platform. There are various types of SAP GUI, which can all operate in the same way, regardless of the underlying platform.

© Copyright . All rights reserved.

Prohibida su reproduccion

Variants of SAP GUI

Figure 13: Variants of SAP GUI

The figure shows the types of SAP GUI and their flow of communication with the ABAP dispatcher or the Internet Communication Manager (ICM).

Note:
The official terms of the SAP GUI are SAP GUI for the Windows environment, SAP GUI for the Java environment, and SAP GUI for Java. These terms are available only in English. Colloquially, the terms SAP GUI for Windows, SAP GUI for Java or Java GUI, and Web GUI are used synonymously with the official terms. The colloquial terms are used frequently.

Caution:
The variants of the SAP GUI are suitable only for working with SAP systems based on AS ABAP. SAP GUI for Java does not offer access to SAP systems based on AS Java. The name SAP GUI for Java refers to its implementation technology. This GUI has been implemented in Java and can be used on front ends that do not run a Windows operating system (for example, front ends that run Mac OS X or Linux).

SAP applications based on AS Java are accessed using a browser.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP GUI

SAP GUI for the Windows environment (SAP GUI for Windows) is the SAP GUI implementation in a Microsoft Windows environment. SAP GUI for Windows is written in C or C++, runs on Windows-based platforms, and supports all transactions in an SAP system.

The data flow between the presentation and application layers does not consist of prepared screens, but rather of logical and compact information about control elements and user input (DIAG protocol). The volume of data transferred for each screen change is a few KB, so you can also easily connect presentation servers over WANs.

SAP GUI for the Java Environment (SAP GUI for Java) is written in Java and is the platform-independent implementation of SAP GUI. Like SAP GUI for the Windows environment, this GUI also uses the DIAG protocol. You can install SAP GUI on either the front end or a terminal server. If you install the SAP GUI on a terminal server, you can access it by using terminal clients.

SAP GUI for HTML requires SAP Internet Transaction Server (ITS) that is part of AS ABAP on the server side, and a Web browser on the client side. SAP ITS provides the services needed for creating responses within the SAP system in HTML format. The responses are transferred to the Web server or Web client outside the SAP system, using the ICM of AS ABAP.

The primary advantage of the SAP GUI for HTML is that it is simple to install on the front end. It requires a browser installation.

Note:
You can find further information on the types of SAP GUI in the following SAP Notes:

- SAP Note 66971: Supported SAP GUI platforms
- SAP Note 314568: SAP GUI for HTML functionality/Limitations/Sp. Behavior
- SAP Note 146505: SAP GUI for the Java environment (Platform Independent GUI)
- SAP Note 138869: SAP GUI on Windows Terminal Server (WTS)
- SAP Note 147519: Maintenance strategy/deadlines for SAP GUI

System Logon

Figure 14: The SAP Logon Program

© Copyright . All rights reserved.

Prohibida su reproduccion

Most of the solutions shipped by SAP can be accessed using the SAP GUI. SAP Logon is a Windows program, which mediates between the SAP system and the SAP GUI.

The steps to access an SAP system by using SAP Logon are as follows:

1. SAP Logon shows a list of SAP systems. This list is taken from a file on the front end, saplogon.ini, which is centrally preconfigured and provided for end users. Select the system to which you want to log on. During logon, SAP Logon enables logon load distribution by using the resources available to the selected system.

2. When logging on to an SAP system, you are prompted to enter a user name and a password, among other things. If you have implemented a Single Sign-On (SSO) environment, you may not need to enter this information. When you log on, you also select a certain client of the SAP system.

   Hint:
   Clients correspond to independent business entities and can be used to represent a company in an SAP system. This means that if an SAP system has several clients, then several companies can be represented and made simultaneously active in that system. The client has a corresponding key field in the tables of the database used by that SAP system. When working in a client, you only have access to business data from the same client.

   You can also select a logon language supported by that system. Standard SAP installation includes German and English, but you can installed other languages. The number of installed languages determines the languages that your system supports.

3. On the logon screen, you use the New password pushbutton to change your password. You can change your password once per day, at most.

In the course of one logon session to a system, you can work in several SAP GUI sessions simultaneously. Your system administrator can use a system parameter rdisp/max_alt_modes to define how many SAP GUI sessions are permissible for each logon to the SAP system.

For details of logon restrictions, see SAP Note 12466. SAP Notes contain detailed information on certain functions or corrections for known errors in SAP products. You can access SAP Notes from, for example, the SAP Service Marketplace on the Internet, by using a valid user (S-user) and password. For more information about SAP Notes, see https://service.sap.com/notes.

© Copyright . All rights reserved.

Prohibida su reproduccion

The Logon Screen for an SAP System

Figure 15: The Logon Screen for an SAP System

The user data in SAP systems is called the user master record. An SAP client stores the user master record in the system. This means that you can log on to only those clients where your user (master record) exists. If your user exists only for client 100, you can successfully log on to client 100. If you can log on to more than one client, then you have more than one user master record, which can allow various activities depending on the client where you log on. Your system administrator can add text to your logon screen.

To see the steps to add text to your logon screen, see SAP Note 205487: Own text on the *SAP GUI Logon* screen.

Attempts to log on to an SAP system multiple times are logged for both security and licensing reasons. If the same user logs on more than once, then for each subsequent logon, the system displays a warning message.

This message offers the following options to the user:

- Continue with the current logon and end all other existing logons.
- Continue with the current logon without ending any existing logons (this is recorded in the log. SAP reserves the right to view the log).
- Terminate this logon.

The second option can be deactivated centrally and system-wide.

© Copyright . All rights reserved.

Prohibida su reproduccion

To Change the Initial Password

Use the initial password assigned to you when you log on to the system for the first time. During the first logon attempt, the system prompts you to change and save your password. For security reasons, it makes sense to occasionally change your password.

Use the following steps to change your password:

1. Enter your new password in the *New Password* and *Repeat Password* fields. You can move between these fields by using a mouse click or the Tab key. Note that numerous rules regarding the structure of a password can be set up for your system.

Figure 16: Dialog Box for Changing the Password

2. Choose ✔ (*Continue*) to save your new password.

To Change the Changed Password Again

User can change their passwords at most once a day on the initial logon screen. In contrast, user administrators with authorization for transaction `SU01` can change user passwords as often as they want. Actually, the administrator can only assign new initial passwords. An initial password requires a change during the next logon of the corresponding user.

The following are the steps to change your password again:

1. On the logon screen of the SAP system, enter your data and current password.
2. Choose the *New Password* pushbutton.
3. Change your password and save your data.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Easy Access Screen Structure

Figure 17: SAP Easy Access Screen

Once you have logged on successfully, the initial screen of the SAP system (SAP Easy Access screen) appears.

The SAP Easy Access screen is the default initial screen in SAP systems. The left panel of the screen contains a tree hierarchy of the menus available to you in the SAP system.

You can use the right side of the screen to display your company logo. This figure is made available centrally by your system administrator and cannot be customized by individual users.

You can find a detailed description of the steps to configure the figure on the *SAP Easy Access* screen under *Extras* → *Administration information* if you have the correct authorization. Note that the figure is stored in the system and transported to the front end every time a user logs on. Because the transportation takes place in compressed form, the figure must not exceed 20 KB.

You can also prevent the figure from being displayed by either of the following options:

- Choose the setting *Low Speed Connection* in SAP Logon.

  See SAP Note 161053: Using SAP GUI in WAN.

- Choose *Extras* → *Settings* on the *SAP Easy Access* screen.

© Copyright . All rights reserved.

Prohibida su reproduccion

Screen Elements in the SAP GUI

Figure 18: Screen Elements in the SAP GUI

A typical screen (with some common elements) appears if you choose the *Defaults* tab page on *System → User Profile → Own Data*. Then, you see the screen shown in the figure with the data for your user.

An SAP user interface can, for example, contain the following simple screen elements:

- Menu bar
- System toolbar
- Command field
- Title bar
- Application toolbar
- Tabs
- Input field checkboxes
- Radio buttons
- Status bar

You can start applications directly by entering their transaction codes in the command field. You can hide and unhide the command field by using the small triangle next to the field. You can find the transaction code for an application in the *SAP* menu on the *SAP Easy Access* screen, in the status bar, or in the application itself under *System → Status*.

© Copyright . All rights reserved.

Prohibida su reproduccion

The menu bar is the top line of any dialog in the SAP system. The menus shown here depend on the application being used. The menus may also contain cascading menus.

The pushbuttons in the standard toolbar are shown on every *SAP Easy Access* screen. If the pushbuttons are not available in an application, they are deactivated. If you place the cursor over a pushbutton for a short time, the system displays a flag with the name or function of the pushbutton. If applicable, the system also displays the corresponding function key setting.

Descriptions of the screen elements are as follows:

- The title bar gives the name of the function that you are currently using.
- The application toolbar shows the pushbuttons available in the application that you are currently using.
- Tabs organize several sub-screens to improve clarity.
- Checkboxes within a field group allow you to select several options at the same time.
- Radio buttons allow you to select only select one option within one field group.
- Input fields and pushbuttons are additional screen elements.
- The status bar displays information about the current system status, such as warnings and errors. You can also change the display variant to show, for example, the transaction code of the current transaction.

To Log Off the System

The steps to log off the system are as follows:

1. On the *SAP Easy Access* screen, choose *System* → *Log off*.

> Hint:
> Alternatively, choose the *Exit* pushbutton on the *SAP Easy Access* screen or enter the **/NEND** command in the command field. If several SAP GUI sessions (window sessions) are open, choose the *Exit* pushbutton to close only the active window. The dialog field for logging off appears in the last window only.

> Caution:
> The */NEX* command immediately terminates your logon without querying the dialog windows first. All SAP GUI sessions of your terminal session are closed and the entries that have not been saved are lost.

2. The *Log Off* dialog box appears and tells you that you will lose data that has not been saved when you log off.

© Copyright . All rights reserved.

Prohibida su reproduccion

Figure 19: Log off Dialog Box

3. To log off, choose the *Yes* pushbutton.

> Hint:
> From the perspective of the Windows operating system, an SAP session is a normal application. You can also close the SAP GUI sessions by using the key combination Alt + F4 on the operating system. When you close the last window of your session, the *Log Off* dialog box is also displayed.

SAP NetWeaver Business Client

Figure 20: Starting SAP NetWeaver Business Client

SAP NetWeaver Business Client provides a unified look and feel when working on SAP systems based on SAP NetWeaver. For example, you can work on systems using SAP NetWeaver Portal, SAP Solution Manager, or SAP ERP by using a similar interface.

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
The following SAP Notes might prove helpful when you are starting to work with SAP NetWeaver Business Client:

SAP Note 1029940: NetWeaver Business Client requirements and restrictions

SAP Note 1400383: SAP BP: SAP NWBC 3.0 as of EHP4 for SAP ERP 6.0, ABAP Config

SAP Note 900000: NetWeaver Business Client - FAQ

Connection Settings

In our training environment, you can access the SAP ECC system DCC using transaction NWBC or the SAP NetWeaver Business Client with the following connection settings:

Connection Settings	Description
Name	The SID of your training system
URL	The URL to connect SAP NetWeaver Business Client to an SAP system. For example, http://server.<full qualified name>:<Port of ICM or HTTP(S)>. Note that the string /sap/bc/nwbc depends on the type of system with which you want to connect.
Type	For example, ABAP or as needed by the system to which you want to connect
Client	The client to which you want to connect
Language	Language that SAP NetWeaver Business Client has to use for connecting to the SAP system (for example, EN or DE)
SAP GUI Logon description	Usually, you do not need to use this field. Please see the context help for this field for further information.
Comment	Description on the *SAP NWBC Logon* screen

Caution:
Using SAP NetWeaver Business Client successfully in combination with SAP Solution Manager requires that your user in the SAP system has at least one work center role assigned.

© Copyright . All rights reserved.

Prohibida su reproduccion

Using SAP NetWeaver Business Client

Figure 21: Using SAP NetWeaver Business Client

After providing the correct credentials (logon screen not shown), you are able to work in the SAP system by using the special roles assigned to your user.

© Copyright . All rights reserved.

Prohibida su reproduccion

Log on to the System and Navigate on the Screen

Business Example

You want to work in the SAP Solution Manager system.

Log on to the training system PSM and carry out some basic exercises in the system (the menu path refers to the SAP standard menu).

> Hint:
> Use the logon data provided by your instructor.

1. Start the SAP Logon program and create a new entry according to the information given by your instructor. Choose the *Logon* pushbutton.

   Enter the client, user name, initial password, and logon language supplied by the instructor. When you log on for the first time, enter a new password of your choice two times in the window that appears.

2. What is the maximum number of parallel SAP GUI sessions that you can open using *System* → *Create Session*?

3. What is the name of the function you reach if you choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *User Overview*? What transaction code do you use to call this transaction instead of the specified menu path?

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 2
Solution 2

Log on to the System and Navigate on the Screen

Business Example

You want to work in the SAP Solution Manager system.

Log on to the training system PSM and carry out some basic exercises in the system (the menu path refers to the SAP standard menu).

> Hint:
> Use the logon data provided by your instructor.

1. Start the SAP Logon program and create a new entry according to the information given by your instructor. Choose the *Logon* pushbutton.

   Enter the client, user name, initial password, and logon language supplied by the instructor. When you log on for the first time, enter a new password of your choice two times in the window that appears.

   a) Follow the instruction in the exercise.

2. What is the maximum number of parallel SAP GUI sessions that you can open using *System → Create Session*?

   a) This training system should allow you to open up to 6 SAP GUI sessions.

3. What is the name of the function you reach if you choose *Tools → Administration → Monitor → System Monitoring → User Overview*? What transaction code do you use to call this transaction instead of the specified menu path?

   a) The function is called *User List*. You can find the relevant transaction code by choosing *System → Status*. In this case, the transaction code is `SM04`.

   b) You can also choose to display the transaction code for an active transaction in the status bar.

   c) Alternatively, you can permanently display the transaction codes on the *SAP Easy Access* screen. Choose *Extras → Settings* and activate the field *Display Technical Names*.

   d) A fourth option is to display the details by choosing the corresponding menu entry and then *Extras → Technical details*.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Navigate through SAP systems by using the SAP GUI

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 2
Lesson 2

Performing Advanced Navigation in the SAP GUI

LESSON OVERVIEW
This lesson describes various ways to call a function in SAP systems and provides an understanding of the management of the user and SAP menus and favorites.

Business Example

As an end user, you want to know how to start functions in the SAP system and the help that the system provides. For this reason, you require the following knowledge:

- An understanding of working on the user and SAP menus
- An understanding of favorites management
- The steps to create a list of transaction codes
- An understanding of calling functions

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Call functions in ABAP-based systems

© Copyright . All rights reserved.

Prohibida su reproduccion

User and SAP Menus

Figure 22: Favorites List and User Menu

The *SAP Easy Access* screen is automatically displayed after logging on. The screen represents the standard access point to an SAP system in SAP GUI for Windows. The navigation bar in the left part of the screen contains a tree structure, consisting of menu and favorites entries. You can adjust the *Favorites* area as per your needs. Only system administrators can change the entries of the SAP and user menus.

The end user can switch from a role-based user menu to the SAP standard menu (if allowed by the system settings).

Hint:
System administrators can use the *USERS_SSM* table to determine whether or not users are allowed to switch between the SAP menu and their user menus. The availability of the user or SAP menus does not affect the authorization of the user. This means that a user can always call authorized functions by using a transaction code, independent of the menu structure.

Role-Based User Menu

A role-based user menu is created, based on the role assigned to a user, and transferred to the front end. This means that it makes sense to keep the number of roles as small as possible because large roles could take a long time to transfer to the front end in certain circumstances.

If a user has several roles assigned, then some functions can be repeated in different user menus.

© Copyright . All rights reserved.

Prohibida su reproduccion

If you do not want these functions to appear twice, see SAP Note 357693 – Redundancy Avoidance in Easy Access (and the other notes listed) for information on identifying and deleting duplicates and related questions.

> Hint:
> As long as you have the appropriate authorization, you can also display user menus other than the ones assigned to you in your user master record. To perform this action, choose the *Other menu* pushbutton. Use the *Create role* pushbutton to start the role maintenance transaction, `PFCG`.

The user menu and SAP menus can be structured in a pretty complex way. It is then difficult to remember the exact path for the transaction you are looking for. Transactions `SEARCH_SAP_MENU` and `SEARCH_USER_MENU` can be used to search the corresponding menus for a predefined text pattern. The search result is prepared in a list view from which you can obtain the navigation path. The transactions found cannot start directly by double-clicking them.

Favorites Management

In addition to the user menu or the SAP menu, the user also gets functions in the Favorites area. Frequently used transactions, web links, or files can also be stored in the *Favorites* area.

The *Favorites* list contains references to SAP system functions, links to internet content, or to files on the front end computers of end users. Each end user can edit the *Favorites* list in accordance with his or her preferences. The list is initially empty. You can only view your own *Favorites* list. Because an SAP system stores the data on the favorites list of each user, a user might have different collections of favorites in different systems.

> Hint:
> The complete *Favorites* menu, including the folder structures, can easily be copied by uploading or downloading it from one SAP system to another SAP system. The downloaded *Favorites* menus are stored temporarily on the front-end computer.

Favorites Menu

To improve the structuring of the *Favorites* list, sort the favorites into folders. You can edit favorites on the *SAP Easy Access* screen by choosing *Favorites*.

You can perform the following tasks by using the Favorites menu:

- You can add a function from the user (or SAP) menu to your personal Favorites list by selecting the function and choosing *Favorites* → *Add*. Here, the menu path is also written with the text of the favorites entry to some extent.
- If you have highlighted a menu entry by using the mouse, you can copy the selected node in the *Favorites* area by right-clicking the node.
- Alternatively, you can also drag the elements from the menu area by using the mouse and store them in the *Favorites* area.

© Copyright . All rights reserved.

Prohibida su reproduccion

- You can add URLs or links to files to your list of favorites by choosing *Favorites* → *Add* other objects. Click the triangle symbol to the left of the file symbol to expand or collapse the list of favorites.

> Hint:
> To change the name of a transaction in favorites, highlight the corresponding favorite entry and select the menu path *Favorites* → *Change*. You can change and save the text in the dialog box that opens. You can delete the *Favorites* entries that you no longer need. Deleting the entries deletes the link to the corresponding functions from your *Favorites* list. If you want to delete the entire *Favorites* menu, highlight the entry *Favorites* and right-click to display the available context menu. Choose *Delete all favorites*.

Options to Call a Function

Figure 23: Options to Call a Function in an SAP System

The options for calling functions in an SAP system are as follows:

- Entering transaction codes in the command field
- Choosing items from the SAP menu
- Choosing items from a user menu
- Choosing items from the *Favorites* list
- Choosing items from menus in the menu bar

You can access the menus in the menu bar by using the key combination *Alt* + <#> (# corresponds to the underlined letter of the selected menu item) or the *Alt* and *cursor* navigation by using the arrow keys on the keyboard.

© Copyright . All rights reserved.

Prohibida su reproduccion

You can use the keyboard shortcut keys to access the *SAP Easy Access* screen or the command field and call system functions from there.

Hint:

Use the following keys to navigate using the keyboard:

- *TAB* allows you to move from one field element to the next element within a field group.
- *Ctrl* + *TAB* allows you to move from one field group to the first element of the next field group.
- Ctrl + / allows you to move directly to the *OK Code* field.

You can find further shortcuts by using *Customizing of local layout (Alt + F12)* → *SAP GUI Help*.

You can use the F1 key on the command field to display the following possible entries:

- `/N` to cancel the current transaction
- `/NXXXX` to call transaction `XXXX` directly from another transaction. Without the prefix, you can only call `XXXX` from the *SAP Easy Access* screen.
- `/O` to display the overview of the SAP GUI sessions on your computer
- `/OXXXX` to call transaction `XXXX` in a new window directly from another transaction
- `/NEND` to end the log on session with a confirmation dialog box
- `/NEX` to end the log on session without a confirmation dialog box
- `/I` to delete the window you are currently using

Note:
Pressing the F4 key in the command field does not display the input help. Instead, the key lists the last 15 entries in the field on your front end. This list is stored in the registry on the front end and is valid for all sessions on the front end, regardless of the system that is used.

© Copyright . All rights reserved.

Prohibida su reproduccion

Call Functions

Business Example

As an end user, familiarize yourself with various ways of calling functions.

Identify who is currently logged on to the training system PSM and create a *Favorites* menu for your user.

Task 1

Find out various ways of viewing the user overview.

1. Run the transaction for displaying a list of users who are logged on. On the *SAP Easy Access* screen, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *User Overview*. Note that this menu path might not be available in your training system by default. In this case, enter **`/nS000`** in the command field.
2. Exit the user overview by using either the F3 key or the *Back* pushbutton in the standard toolbar.

   Run transaction code `SM04`, which takes you directly to the user overview.

Task 2

Create a *Favorites* menu. Save the frequently used transactions in a *Favorites* menu.

1. To view the user overview, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *User Overview* on the *SAP Easy Access* screen.

   Choose the transaction entry. On the *SAP Easy Access* screen, choose *Favorites* → *Add*. The transaction is saved in the *Favorites* area.
2. Add transactions `RSPFPAR`, `SEARCH_SAP_MENU`, and `SEARCH_USER_MENU` to the *Favorites* menu without expanding the SAP menu.
3. Can you distinguish between the favorites entries by the method used to insert them?
4. Create a folder *Search Options* in which you can store both search transactions.
5. Find out how often and where transaction `SM04` for the user overview is stored in the SAP menu.

© Copyright . All rights reserved.

Prohibida su reproduccion

Call Functions

Business Example

As an end user, familiarize yourself with various ways of calling functions.

Identify who is currently logged on to the training system PSM and create a *Favorites* menu for your user.

Task 1

Find out various ways of viewing the user overview.

1. Run the transaction for displaying a list of users who are logged on. On the *SAP Easy Access* screen, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *User Overview*. Note that this menu path might not be available in your training system by default. In this case, enter **`/nS000`** in the command field.
   a) See the exercise for the solution.
2. Exit the user overview by using either the F3 key or the *Back* pushbutton in the standard toolbar.

   Run transaction code `SM04`, which takes you directly to the user overview.
   a) See the exercise for the solution.

Task 2

Create a *Favorites* menu. Save the frequently used transactions in a *Favorites* menu.

1. To view the user overview, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *User Overview* on the *SAP Easy Access* screen.

   Choose the transaction entry. On the *SAP Easy Access* screen, choose *Favorites* → *Add*. The transaction is saved in the *Favorites* area.
   a) See the exercise for the solution.

   > Hint:
   > Alternatively, you can use a drag-and-drop action or the context menu to copy transactions into the Favorites area. The context menu is opened by a right-click action.

2. Add transactions `RSPFPAR`, `SEARCH_SAP_MENU`, and `SEARCH_USER_MENU` to the *Favorites* menu without expanding the SAP menu.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) On the *SAP Easy Access* screen, choose *Favorites* → *Insert transaction*.

b) In the *Transaction Code* field, enter the desired transaction.

c) Choose the *Continue* pushbutton.

d) Execute these steps for every transaction.

3. Can you distinguish between the favorites entries by the method used to insert them?

   a) Yes, when inserting with the previous selection in the menu, a part of the path information of the menu access is stored as a favorites text.

4. Create a folder *Search Options* in which you can store both search transactions.

   a) On the *SAP Easy Access* screen, choose *Favorites* → *Insert folder*.

   b) In the *Folder Name* field, enter search options.

   c) Drag both the search transactions `SEARCH_SAP_MENU` and `SEARCH_USER_MENU` to the file.

5. Find out how often and where transaction `SM04` for the user overview is stored in the SAP menu.

   a) Run the search transaction `SEARCH_SAP_MENU`.

   b) In the *Search text entry* field, enter `SM04`.

   c) Transaction `SM04` is stored three times in different paths in the SAP menu, as shown in the following table.

Transaction	Path on the SAP Easy Access screen
`SM04`	*Tools → Administration → Monitor → System Monitoring → User Overview*
`SM04`	*Tools → Administration → Monitor → Performance → Exceptions/User → Active User → Local User*
`SM04`	*Tools → CCMS → Control/Monitoring → Performance → Exceptions/User → Active User → Local User*

   In the same way, you can search in the user menu.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Call functions in ABAP-based systems

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. The user data in SAP systems is called the _____.

   Choose the correct answer.

   - [ ] A User record
   - [ ] B User master record
   - [ ] C Stored record

2. You can work in only one window (session) at a time in an SAP system.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

3. The following strings are valid entries in the command field in the SAP Easy Access screen.

   Choose the correct answers.

   - [ ] A /nend
   - [ ] B /nex
   - [ ] C ?SM04
   - [ ] D /nsm04
   - [ ] E From SAP Easy Access: SM04
   - [ ] F From SAP Easy Access: search_sap_menu

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Identify the statements that accurately describe a client in an SAP system.

   Choose the correct answers.

   - ☐ A A client represents a completely independent business entity.
   - ☐ B A client has its own database.
   - ☐ C A client corresponds to a customer.
   - ☐ D A client may represent an entire company.

5. Only _____ can change the entries of the SAP and user menus.

   Choose the correct answer.

   - ☐ A Current users
   - ☐ B System administrators

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. The user data in SAP systems is called the _____.

   Choose the correct answer.

   - [ ] A User record
   - [x] B User master record
   - [ ] C Stored record

   The user data in SAP systems is called the user master record. User master records are stored in the system by a client. This means that you can only log on to the clients where your user (master record) exists.

2. You can work in only one window (session) at a time in an SAP system.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

   Within one log on, it is possible to work in several windows (SAP GUI sessions) in parallel.

© Copyright . All rights reserved.

Prohibida su reproduccion

3. The following strings are valid entries in the command field in the SAP Easy Access screen.

   Choose the correct answers.

   - [x] A /nend
   - [x] B /nex
   - [ ] C ?SM04
   - [x] D /nsm04
   - [x] E From SAP Easy Access: SM04
   - [x] F From SAP Easy Access: search_sap_menu

   You can enter valid transaction codes directly on the SAP Easy Access screen. If you are using another function, enter /n in front of the transaction code. /nend and /nex are two different logoff options. ?SM04 is not a valid entry. The valid entry is search_sap_menu because it creates a search screen for entries in the SAP menu.

4. Identify the statements that accurately describe a client in an SAP system.

   Choose the correct answers.

   - [x] A A client represents a completely independent business entity.
   - [ ] B A client has its own database.
   - [ ] C A client corresponds to a customer.
   - [x] D A client may represent an entire company.

   The data for all clients in an SAP system is stored within a single, common database. This data is nevertheless divided by clients so that different companies can be administered and controlled in different clients. A client is not a customer within an SAP system.

5. Only _____ can change the entries of the SAP and user menus.

   Choose the correct answer.

   - [ ] A Current users
   - [x] B System administrators

   Only system administrators can change the entries of the SAP and user menus.

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 3 The System Core

Lesson 1

Exploiting the SAP NetWeaver Application Server Architecture 66

Lesson 2

Distinguishing the Processes of SAP NetWeaver AS Java and ABAP 72
Exercise 4: Determine the Structure of an Instance 85

Lesson 3

Processing AS ABAP Requests 88
Exercise 5: List the Work Processes Configured on an Instance of AS ABAP 95

Lesson 4

Explaining AS ABAP Processes 98
Exercise 6: Set and Monitor Lock Entries 107
Exercise 7: Print a Screenshot in AS ABAP 113
Exercise 8: Schedule a Simple Report in the Background 117

UNIT OBJECTIVES

- Understand the architecture of SAP NetWeaver AS
- Identify the processes that run on SAP NetWeaver AS
- Plan the system architecture of different types of SAP NetWeaver AS
- Describe the processing of requests in AS ABAP
- Describe the processing of transactions in AS ABAP
- Explain ABAP processes and their purposes

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 3
Lesson 1

Exploiting the SAP NetWeaver Application Server Architecture

LESSON OVERVIEW
This lesson introduces the principal architecture of SAP systems and the processes of SAP NetWeaver Application Server (AS), based on simple client/server configurations. The lesson also explains the term instance.

Business Example

In the course of implementing an SAP system, you need to understand the architecture of SAP systems and determine how you are going to distribute the required SAP system processes among the available hardware. For this reason, you require the following knowledge:

- An understanding of the client/server configurations
- An understanding of the term instance

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understand the architecture of SAP NetWeaver AS

Basic Terms

SAP systems are used for running business processes or business applications. These applications need to be implemented independent of the hardware environment used (operating system and database) to the greatest extent possible.

SAP NetWeaver provides the following runtime environments:

- ABAP runtime environment (AS ABAP)
- Java runtime environment (AS Java)

ABAP Runtime Environment

ABAP is a programming language developed by SAP. Many business applications in the SAP system are written in ABAP. ABAP has been optimized for developing highly scalable business applications. Customers can use the ABAP Workbench for developing completely new applications or enhancing and modifying standard SAP applications. Applications are developed or enhanced using the infrastructure of AS ABAP. AS ABAP provides the runtime environment for programs written in ABAP.

SAP provides a runtime environment for ABAP programs and Java programs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Java Runtime Environment

AS Java is an application server, based on the Java 2 Enterprise Edition (J2EE) standard. Java is an object-oriented and a platform-independent programming language that has spread across many areas. The Java concept enables the development of a large range of applications from classical applications to applets used in websites to client/ server applications.

As of SAP Web Application Server 6.20, SAP has provided a complete Java EE-compatible runtime infrastructure. Up to SAP NetWeaver AS Java 7.0, the infrastructure was compliant with J2EE 1.3. AS Java 7.1x are EE 5-compliant, and as of AS Java 7.3x, they are EE6-compliant. The architecture of an application server in accordance with the Java EE specification is similar to the ABAP runtime environment used for SAP solutions. Both runtime environments are platform-independent, executed in a multi-level architecture, and have a separate presentation logic and business logic.

Using Java EE in SAP technology has the following advantages for SAP Netweaver Application Server:

- The open integration architecture SAP NetWeaver integrates perfectly into the openness of Java EE.
- Java EE is further proof of the strategy of platform-independence pursued by SAP.
- The Java EE Connector architecture allows standardized integration of external applications.
- Web Services technologies are supported directly by Java.
- The quickly-growing Java community provides simple access to experienced developers.

Client and Server – Terminology Definition

Figure 24: Hardware-Oriented and Software-Oriented Views

The concepts of client and server can be defined in the following ways:

© Copyright . All rights reserved.

Prohibida su reproduccion

- In the hardware-oriented view, the term server means the central server in a network that provides data, memory, and resources for workstations (clients).
- In the software-oriented view, client and server are both defined at the process level (service). The process requesting a service is the client, and the server is the process providing the service.

In the context of SAP systems, the terms "client" and "server" are used as defined in the software-oriented view.

Client/Server Configuration for SAP Systems

Figure 25: Simple Client/Server Configurations

The following processes are used for operating business application software:

- Presentation processes (for example, for displaying screens)
- Application processes (for example, for executing applications)
- Database processes (for example, for managing and organizing databases)

When you are installing and configuring an SAP system, you need to decide how you are going to distribute the required processes among the available hardware. SAP ECC (SAP Enterprise Core Component) is an example of business application software. The figure shows different types of configurations in relation to the number of hardware layers used.

Single-Tier, Two-Tier, and Three-Tier Configurations

Single-Tier Configuration

In single-tier configuration, only one computer performs all the processing tasks (database, application, and presentation processes). This is classic mainframe processing.

Single-tier configurations are used for tests and demonstrations (for example, an SAP system on a laptop). If many users want to work on a system in single-tier configuration, then the extra

© Copyright . All rights reserved.

Prohibida su reproduccion

hardware cost for each additional user becomes greater than the costs of implementing additional hardware levels (for example, moving the presentation processes to other hosts).

Two-Tier Configuration

Special presentation servers that are used to format the graphical interface usually implement two-tier configurations. A two-tier configuration with distributed presentation processes can maintain good performance for a high number of users. There is no substantial increase in the hardware costs compared to a single-tier configuration. The load resulting from the presentation processes is distributed among various front-end computers so that the performance of the database host is not affected.

An alternative two-tier configuration is to install powerful desktop systems and use them for presentation and application processes (two-tier client/server). This configuration is especially suitable for applications that have high processor demands (for example, for simulations or software developers). Such a configuration is implemented in the SAP environment only for testing purposes because it requires additional administration.

Three-Tier Configuration

In a three-tier configuration, each layer runs on its host. Multiple application servers can simultaneously work with the data of a database server.

If the number of users exceeds a certain upper limit, the central host, on which we run our application and database processes, will become a bottleneck. To prevent such a bottleneck, you can improve the performance of the SAP system by distributing the application-layer processes among several hosts.

The addition of a hardware layer specifically for application processes facilitates scalability. If the number of SAP users in a system increases over time, it negatively affects system performance. To improve system performance, another host can be added for application processes in most cases.

In the SAP Business Suite environment, more complex client/server configurations consisting of more than three tiers are both theoretically possible and used in practice. An additional level could be a Web server, for example.

© Copyright . All rights reserved.

Prohibida su reproduccion

Instance

Figure 26: The Instance of an SAP System (Example)

An instance is an administrative unit that combines the SAP system components together to provide one or more services. The services provided by an instance are started or stopped together. A common instance profile is used to set the parameters of the components in an instance. Each instance has its own buffer areas. An instance runs on one physical computer, but it is possible for multiple instances to be there on one computer. An instance is identified by the system ID (SID) and the instance number.

> Hint:
> The terms SAP instance and Application Server are often used synonymously.

When you install an SAP system, you have the option of separating the processes at the application level away from the database level. This means that the installation and operation of the database for an SAP system can be done on a separate physical computer, separated from the instances of the SAP system. There is exactly one database for each SAP system. The database has the same system ID (DB ID) as the SAP system.

Central Instance

The central instance of an SAP system offers services that no other instance of the system offers. For AS ABAP, the central instance provides the message server and the enqueue work process.

The central services instance provides the message service and the enqueue service for AS Java. For AS ABAP, these services can also be moved to the ABAP central services instance for high-

© Copyright . All rights reserved.

Prohibida su reproduccion

availability reasons. If the services are moved, the AS ABAP systems would no longer have a central instance.

All other instances of a system are typically called dialog instances.

If the central instance and the database (and for AS Java, the central services instance also) are installed on the same computer, the computer is referred to as a central system.

SAP System

SAP system is an installed software system that provides a defined set of functions that are part of an SAP solution. These functions are implemented in a set of software components. An SAP system is installed and configured as a unit. It consists of a database, one or more application server instances (ABAP and/or Java), central services (such as a message server and enqueue server), and optional components (such as TREX and liveCache). The system is identified by its SAP system ID, which consists of three letters or digits (for example DCC).

LESSON SUMMARY

You should now be able to:

- Understand the architecture of SAP NetWeaver AS

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 3
Lesson 2

Distinguishing the Processes of SAP NetWeaver AS Java and ABAP

LESSON OVERVIEW

This lesson explains the architecture and process of the different types of SAP NetWeaver Application Server (SAP NetWeaver AS).

Business Example

In the course of implementing an SAP system, you need to establish the architecture of SAP systems and how you are going to distribute the required SAP system processes among the available hardware. For this reason, you require the following knowledge:

- An understanding of AS ABAP Processes
- An understanding of how to identify the processes that run on SAP NetWeaver AS
- An understanding of the types of SAP NetWeaver AS
- An understanding of AS Java Architecture
- An understanding of AS ABAP Architecture

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Identify the processes that run on SAP NetWeaver AS
- Plan the system architecture of different types of SAP NetWeaver AS

© Copyright . All rights reserved.

Prohibida su reproduccion

AS ABAP Processes

Figure 27: AS ABAP Processes

In AS ABAP, these ABAP processes include the dispatcher and a number of work processes on the application server, depending on the hardware resources, as shown in the figure.

The dispatcher distributes the requests to the work processes.

The available work processes are as follows:

- Dialog work processes fulfill all requests for the execution of the dialog steps triggered by an active user. Every dispatcher requires at least two dialog work processes.
- Spool work processes pass a sequential flow of data to printers. At least one spool work process is required for each SAP system. It is possible to configure more than one spool work process for each dispatcher.

> Note:
> If a system is not used for printing, a spool work process is still required for storing the lists generated in the context of the background process in TemSe (temporary sequential object).

- Update work processes execute update requests. Similar to spool work processes, you need at least one update work process per SAP system. You can configure more than one update work process per dispatcher.

© Copyright . All rights reserved.

Prohibida su reproduccion

- Background work processes execute programs that run without interacting with a user. For the normal operation of an SAP system, a single background process would be sufficient. But, for an upgrade or the import of ABAP transport requests, you require a second background work processes. You can configure more than one background work process for each dispatcher.
- The enqueue work process administers the lock table in the shared memory. This table contains the logical database locks of the ABAP runtime environment. Only one enqueue work process is needed for each SAP system.
- Configuration of additional enqueue work processes has to be done on the same instance as the first enqueue work process because these work processes have to access the same lock table.

To summarize, the dispatcher of an ABAP instance manages different types of work processes in its instance, such as the dialog, update, background, enqueue, and spool work processes. These work processes work on different tasks when executing the business procedures in an SAP system.

The administrator of an SAP system can use profile parameters to configure the number of work processes of various types.

Hint:
The types and number of the work processes to be started for each instance are configured using the SAP system profile parameter.

Note:
To learn more about the number of ABAP work processes to be configured for an instance, see SAP Note 39412.

Additional Services of the ABAP Runtime System

In addition to the work processes, the ABAP runtime system provides the following services (not work processes) for internal and external communication:

- The message server (MS) handles communication between the distributed dispatchers within the AS ABAP, which enables the scalability of several parallel application servers. The message server is configured only once per SAP system.
- The gateway process (GW) enables communication among SAP systems or between SAP systems and external application systems. There is one gateway reader per dispatcher.
- The Internet Communication Manager (ICM) enables communication with the SAP system by using Web protocols, such as HTTP. The ICM receives requests from a client and forwards them to an SAP system for processing. In an ABAP + Java system, the ICM recognizes whether the request is a call for AS ABAP or AS Java and forwards the request accordingly. The ICM can also direct HTTP requests from an SAP system to a Web server and send the response from the Web server back to the SAP system. You can configure a maximum of one ICM process per application server (software-based view).

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP NetWeaver AS Java Processes 7.0x and Lower

Figure 28: SAP NetWeaver AS Java Processes 7.0x and Lower

The following processes exist in AS Java:

- The dispatcher distributes incoming requests to the server processes.
- The server process executes the Java applications. Every server process is multithreaded and can process a number of requests in parallel. This is in contrast to ABAP work processes. For each dispatcher, there is at least one server process and up to 16 server processes.

> Note:
> Generally, fewer Java server processes are configured per Java dispatcher than the work processes configured for each ABAP dispatcher. A maximum of 16 Java server processes can be configured for each instance.

- The Java message service manages a list of Java dispatchers and server processes. It is responsible for communication within the Java runtime environment. The message service is configured only once per SAP system on the Java Central Services instance.
- The Java enqueue service manages logical locks that are set by the executed Java application program in a server process. The enqueue service is configured only once per SAP system on the Java Central Services instance.
- For each SAP Java system, there is one SDM process and it resides on the Java central instance.
- The Software Deployment Manager (SDM) is the standard tool used to install Java software components on SAP NetWeaver AS Java.

© Copyright . All rights reserved.

Prohibida su reproduccion

For 7.0x and lower, a Java instance that is not of type Java CS (central services) has exactly one Java dispatcher and requires a minimum of one server process. There can be multiple instances installed on one host.

Processes in SAP NetWeaver AS Java 7.1x and Higher

Figure 29: Processes in SAP NetWeaver AS Java 7.1x and Higher

An AS Java cluster consists of several instances. Each instance has an instance number and runs on a Java Virtual Machine (JVM). You can start, stop, and monitor the instances separately.

The processes of AS Java 7.1x and higher are as follows:

- The central services instance consists of a message service, an enqueue service, and a gateway service. These services are responsible respectively for message exchange and load balancing with the Java cluster, lock administration, and external communication.
- ICM handles requests from clients and dispatches them to the available server processes. Data is transferred from the ICM to the server processes and the other way round by using the Fast Channel Architecture (FCA), which allows fast and reliable communication between the ICM and server processes. ICM replaces the Java Dispatcher.
- The server process executes the Java applications. The process is responsible for processing incoming requests, which are assigned by the ICM. Each server process is multi-threaded and can simultaneously process a large number of requests.

© Copyright . All rights reserved.

Prohibida su reproduccion

Types of SAP NetWeaver AS – Installation Options (7.0x and Lower)

Figure 30: Possible Types of SAP NetWeaver AS (7.0x and lower Versions)

Depending on the application or product used, the following types of application servers are installed:

- AS ABAP system

  This type of AS provides the complete infrastructure where ABAP-based applications can be developed and used.

- AS Java system

  This type of AS provides the complete infrastructure for developing and using J2EE-based applications.

- AS ABAP + Java system

  This type of AS provides the complete infrastructure in which ABAP-based and J2EE-based applications can be developed and used. Such a system needs to be installed only if it is explicitly required by a solution, such as SAP NetWeaver PI 7.0 or SAP Solution Manager 7.1.

One of the main characteristics of SAP NetWeaver AS is that the ABAP schema of the database stores ABAP tables, programs, and application data while the Java schema stores the Java data. Here, the ABAP runtime environment can access the ABAP schema of the database. The Java runtime environment can access the Java schema. In an ABAP + Java system, various runtime environments communicate directly through the SAP Java Connector (JCo).

© Copyright . All rights reserved.

Prohibida su reproduccion

Possible Types of SAP NetWeaver AS (7.10 and Higher)

Figure 31: Possible Types of SAP NetWeaver AS (7.10 and Higher)

Architecture of AS ABAP

Figure 32: Architecture of AS ABAP (7.0x and Lower)

In AS ABAP 7.0x and lower, the central instance is distinguished by the fact that the message server and the enqueue work process run there. All other instances of the system are called dialog instances. Alternatively, you can also name the instances after the services provided.

© Copyright . All rights reserved.

Prohibida su reproduccion

For SAP NetWeaver AS ABAP, no changes were made from an architecturual point of view between 7.0x and higher releases. In 7.1 and above, only the terms “central instance” was renamed to “Primary Application Server (PAS)” and “dialog instance” was renamed to “Additional Application Sever (AAS)”.

The types of work processes in an application server determine the services provided by the application server. An application server can take on several roles. For example, an application server can function as a dialog server and an update server if it provides several dialog work processes and at least one update work process.

> Note:
> An overview of the AS ABAP instances is available in `SM51`. On the *SAP Easy Access* screen, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *Servers*.

Use transaction `SM50` to display an overview of the work processes on the instance to which you are logged on. You can also display this overview by choosing *Tools* → *Administration* → *Monitor* → *System Monitoring* → *Process Overview* on the *SAP Easy Access* screen.

Message Server

The functions of a message server are as follows:

- Provide AS ABAP with a central message service for internal communication (for example, for starting updates, requesting and removing locks, and triggering background requests).
- Provide information about the instances of the system that are currently available.
- Enable interprocess communication among the ABAP dispatchers of the application servers. The message server is installed exactly once per SAP system.
- Perform a load distribution of users to the available instances when users log on to AS ABAP by using logon groups on the SAP GUI for Windows or the SAP GUI for Java.

The load distribution, which takes place during the logon procedure, is also known as logon load balancing. After the message server performs the load distribution, the SAP GUI communicates directly with the dispatcher. The user remains logged on to this instance until he logs off again.

> Note:
> An overview of users who are logged on to the instance to which you are also logged on, is available by using transaction `SM04` (alternatively, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *User Overview*).

You can see the instance to which you are logged on under *System* → *Status*.

If you are accessing AS ABAP by using Web protocols, such as HTTP, in a browser, the ICM receives the request. The ICM forwards the request to the dispatcher of the instance that is allotted after load balancing.

Communication from other SAP systems through the remote function call (RFC) is accepted by the gateway reader.

© Copyright . All rights reserved.

Prohibida su reproduccion

Architecture of AS Java (7.0x and Lower)

Figure 33: Architecture of AS Java 7.0 and Lower

In AS Java 7.0x, you distinguish the central instance by the fact that the SDM runs there. The central services message server and enqueue server run in the central services (CS) instance. All other instances of the system are called dialog instances.

In AS Java 7.3 EHP1, the Java instance consists of the ICM and one or more server processes. The SDM process was discontinued.

> Note:
> The entirety of the Java environment (all processes and the database scheme) is also referred to as a Java cluster. The individual processes (dispatcher and server) are referred to the Java cluster as nodes.

You can get an overview of the Java processes that were started by using the system information of the Java runtime environment (http://<Hostname>:<Java-Dispatcher-HTTP-Port of the Instance> /sap/ monitoring/ SystemInfo – for example, http://twdf1234.wdf.sap.corp:50000, choose *System Information*).

© Copyright . All rights reserved.

Prohibida su reproduccion

Architecture of AS Java 7.1x and Higher

Figure 34: Architecture of AS Java 7.1x and Higher

Analogous to AS ABAP, the message service of AS Java provides a central message service for internal communication. The Java message service also provides information about the instances and nodes of AS Java that are available. Each node of the Java cluster can communicate directly with the message service.

In AS Java, the enqueue service holds logical locks. Each node of the Java cluster can communicate directly with the enqueue service. When AS Java is accessed using a browser, the Java dispatcher receives requests, which are then processed by the server processes.

© Copyright . All rights reserved.

Prohibida su reproduccion

Architecture of AS ABAP + Java

Figure 35: Architecture of AS ABAP + Java 7.0x and Lower

In AS ABAP + Java (meaning ABAP and Java processes in the same SAP system, under the same system ID), the same architectural principles apply as the principles for separate AS ABAP and AS Java systems. However, there are some peculiarities because both runtime environments are integrated with one another.

> Note:
> AS ABAP + Java is called an add-in installation because it is possible to install AS ABAP first and then supplement it with AS Java later.

The central instance of an AS ABAP + Java system can be recognized by the processes ABAP-MS, an enqueue work process, and SDM. The central services of the Java runtime environment (Java-MS and Java-ES) are also provided in the Java central services instance here. All other instances are called dialog instances.

Both the AS ABAP and Java runtime environments can answer requests using Web protocols, so the ICM has to determine the environment to which the request is addressed. The ICM decides the address of the request by its URL. In case of a request to the ABAP runtime environment, for example, the call of an ABAP Web Dynpro, the ICM forwards the request to the ABAP dispatcher and the work processes respond to the request.

If the system makes a request for the Java runtime environment, for example, the call of a Java Server Page (JSP), the ICM forwards the request to the Java dispatcher and one of the server processes responds to the request.

In an AS ABAP + Java system, data is also kept in separate database schemas (but in the same database installation). That is, work processes can access only ABAP data and server processes

© Copyright . All rights reserved.

Prohibida su reproduccion

can access only Java data. In the data exchange, both runtime environments then communicate using the SAP Java Connector (JCo). This communication is necessary, for example, if billing data stored in the ABAP data schema needs to be displayed in a Java user interface.

The SAP JCo is integrated into AS Java and used when an AS Java system has to communicate with a remote AS ABAP system.

> Note:
> The SAP JCo provides the SAP RFC protocol for the Java environment. The JCo is also available in a separate installation, for example, to connect non-SAP products in Java with SAP ABAP systems. Another special feature of AS ABAP + Java is the joint storage. In this case, the User Management Engine (AS Java) uses the users from AS ABAP.

Architecture of AS ABAP + Java 7.1x and Higher

Figure 36: Architecture of AS ABAP + Java 7.1x and Higher

The figure shows the architecture of an SAP system, based on SAP NetWeaver AS 7.10 or higher versions.

Changes Introduced with SAP NetWeaver AS 7.1x

As of SAP NetWeaver AS 7.10, the following changes were introduced in the architecture:

- The former central instance was renamed to primary application server (PAS).
- The former dialog instance was renamed to additional application server (AAS).
- The Java dispatcher was replaced by the ICM process.

© Copyright . All rights reserved.

Prohibida su reproduccion

- The SDM process was discontinued.
- The (Java) central services instance comprises a gateway process.

The instance that is installed first is referred to as the PAS Instance. Further instances for a system are referred to as AAS.

© Copyright . All rights reserved.

Prohibida su reproduccion

Determine the Structure of an Instance

Business Example

As the system administrator, you need to have an overview of the processes of your SAP system.

Determine the number of instances that an SAP system can have.

1. With which instance are you currently working in system PSM?
2. Which (AS ABAP) instances does training system PSM offer?
   Which services are provided?
   What is the system number of the central instance of the system?
3. Which AS Java instances exist in system PSM?

© Copyright . All rights reserved.

Prohibida su reproduccion

Determine the Structure of an Instance

Business Example

As the system administrator, you need to have an overview of the processes of your SAP system.

Determine the number of instances that an SAP system can have.

1. With which instance are you currently working in system PSM?
   a) Log on to the *SAP Easy Access* screen by using the SAP GUI for Windows.
   b) On the *SAP Easy Access* screen, choose *System* → *Status*.
   c) The name of the instance to which you are logged on is available in the *Server Name* field and has the structure: <computer name>_<SID>_<instance number>.
2. Which (AS ABAP) instances does training system PSM offer?
   Which services are provided?
   What is the system number of the central instance of the system?
   a) On the *SAP Easy Access* screen, run transaction code `SM51` (alternatively, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *Servers*).
   b) You can see a list of all AS ABAP instances that were started and the services provided.
   c) The instance in which the enqueue work process is configured is the central instance.
   d) In transaction `SM51`, you can recognize this instance by the *message type* enqueue.
3. Which AS Java instances exist in system PSM?
   a) Start a Web browser.
   b) Run the start page of the AS Java of in SAP system PSM.
   c) You reach the start page at http://<host name>:<port-number>, for example http://twdf1928.wdf.sap.corp:58000/
   d) On the AS Java homepage, choose the *System Information* link.
   e) Log on with the user and password provided by the instructor.
   f) You see information about the AS Java instances and central services (SCS).

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Identify the processes that run on SAP NetWeaver AS
- Plan the system architecture of different types of SAP NetWeaver AS

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 3
Lesson 3

Processing AS ABAP Requests

LESSON OVERVIEW
This lesson discusses how SAP NetWeaver Application Server ABAP (AS ABAP) works. It also explains the working of AS ABAP processes.

Business Example

Your company has decided to implement SAP ERP. The business core functions, such as accounting in SAP ERP, are processed by the ABAP runtime environment. Employees in different technical departments (development, system administration, work preparation, security, and so on) need to understand how AS ABAP works in more detail. For this reason, you require the following knowledge:

- An understanding of how AS ABAP works
- An understanding of the database interface of AS ABAP
- An understanding of the processing of requests to AS ABAP

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe the processing of requests in AS ABAP

© Copyright . All rights reserved.

Prohibida su reproduccion

Processing of Requests by the SAP GUI

Figure 37: Processing a User Request in ABAP

Users can log on to an SAP system by using either the SAP GUI or a Web client (browser). When users log on by using the SAP GUI, the ABAP runtime environment processes the user requests. When users log on by using a browser, processing can be done by the ABAP or Java runtime environment, depending on the user request.

Users log on by using the (ABAP) message server (load balancing) or directly on the ABAP dispatcher. The work processes execute the user requests.

The processing of a user request in AS ABAP, as outlined in the figure, involves different processes on all three layers (presentation, application, and database).

The screen entries of a user are accepted by the SAP presentation program SAP GUI, converted to an internal format, and then forwarded to AS ABAP.

ABAP Dispatcher

The (ABAP) dispatcher is the central process of AS ABAP, which performs the following functions:

- Manages the resources for the applications written in ABAP in coordination with the respective operating system
- Distributes the requests among the work processes
- Integrates the presentation layer
- Organizes the communication activities

© Copyright . All rights reserved.

Prohibida su reproduccion

- Saves the processing requests in request queues and then process them according to the first in, first out principle

Data is processed in a work process, and the user who creates a request using the SAP GUI is not always assigned the same work process. There is no fixed assignment of work processes to users. To process user requests, it is often necessary to read data from the ABAP schema of the database or to write to it. Every work process connects directly to the ABAP schema of the database to process user requests.

Note:
The database schema (all tables of AS ABAP) belongs to a user, who logs on to the database using the work processes. Up to release 4.6C, this user was called SAPR3. After that, the user name was changed to SAP<SID> so that several SAP systems could be operated in one database. You can look up this user in the SAP system under *System → Status*. For AS Java, the user (the database schema) is called SAP<SID>DB.

Once the work process is complete, its result is sent through the ABAP dispatcher back to the SAP GUI. The SAP GUI interprets the result and generates the output screen for the user.

The buffers help to speed up the processing of user requests. Data that is often read but seldom changed (for example, programs or customizing data such as clients, currencies, or company codes) can be kept as a copy of the database content in the shared memory of the application server. This means that the data does not need to be read from the database every time it is needed. But the data can be called quickly from the buffer. Each instance has its own buffers.

Note:
During initialization of the SAP system, the dispatcher executes the following actions (among others):

- Read system profile parameters.
- Start work processes.
- Log on to the message server.

© Copyright . All rights reserved.

Prohibida su reproduccion

Process Flow for Requests

Figure 38: Process Flow for Requests

Work processes execute the process logic of application programs. In addition to internal memory, a work process has a task handler that coordinates the actions within a work process, software processors, and a database interface. The dynpro processor executes the screen flow logic of the application program, calls the processing logic modules, and transfers field content to the processing logic. The ABAP interpreter executes the actual processing logic of the ABAP application programs. The screen processor informs the ABAP processor about the subprogram that needs to be executed, depending on the processing status of the screen flow logic.

The dialog work process selected by the dispatcher first performs a roll-in of the user context. The data that contains the current processing status of a running program as well as the data that characterizes the user is made known to the work process. The work process then processes the user request, which may involve, for example, requesting data from the database or the buffers in the shared memory. After the dialog work process has processed the dialog step, the work process returns the result, rolls the user context back to the shared memory, and, becomes available again for a new user request from the request queue. The result is transferred to the SAP GUI and the user sees the new screen.

Database Interface of AS ABAP

Relational Database Management Systems (RDBMS) are used to manage large sets of data. An RDBMS saves data and relationships between data in the form of two-dimensional tables. These are known for their logical simplicity. Data, tables, and, table relationships are defined at database level in the database catalog (the data dictionary) of the RDBMS.

© Copyright . All rights reserved.

Prohibida su reproduccion

Within the SAP programming language ABAP, you can use ABAP Open Structured Query Language (SQL), a database query language, to access the application data in the database, regardless of the RDBMS used. The database interface, which is part of every work process of AS ABAP, translates Open SQL statements from ABAP into the corresponding SQL statements for the specific database used (Native SQL). This allows ABAP programs to be database independent.

> Note:
> ABAP Open SQL is a database query language, based on the (ISO) SQL standard. The language contains enhancements that are not included in the SQL standard.

When interpreting Open SQL statements, the SAP database interface checks the syntax of these statements and ensures the optimal utilization of the local SAP buffers in the shared memory of the application server. Data that is frequently required is stored in these buffers so that the system does not need to access the database server to read this data. In particular, all technical data, such as ABAP programs, screens, ABAP Dictionary information, and a number of business administration parameters, remains unchanged in an operational system. The technical data is ideally suited to buffering.

Flow of a Database Query

Figure 39: Flow of a Database Query

You can use native SQL commands directly in ABAP without using the local buffers and having the database interface interpret the commands. You can do this by including the commands in an EXEC SQL - END EXEC bracket in the ABAP program. The ABAP interpreter does not check the syntax of any commands within this bracket. If you use native SQL to develop programs, you can no longer ensure the platform independence of the programs

© Copyright . All rights reserved.

Prohibida su reproduccion

> Note:
> If there is more than one instance, then there is also more than one buffer. Therefore, you may need to synchronize buffers occasionally.

Processing of Dialog Requests

Figure 40: Multiplexing of Dialog Work Processes

As illustrated in the figure, a program dialog step is assigned to one specific dialog work process during execution. The individual dialog steps for a program consisting of several screens can be executed by various dialog work processes during program runtime. This is called multiplexing of work processes. A dialog work process sequentially processes dialog steps for various users and programs.

SAP applications differentiate between user interaction and processing logic. The user actions are technically realized using screens – also called dynpros (from dynamic programs) – which consist of a screen image and the underlying flow logic. The dynpro processor of the work process executes the screen flow logic of the application, calls processing logic modules, and transfers field content to the processing logic.

The screen flow logic itself is further divided into the following modules:

- Process Before Output (PBO)

  It is processed before the screen image is sent.

  This part logically belongs to the subsequent screen image.

- Process After Input (PAI)

  It is processed after a user interaction on the screen.

© Copyright . All rights reserved.

Prohibida su reproduccion

This part of a dialog step logically belongs to the preceding screen image.

The actual processing logic of ABAP programs is executed by the ABAP interpreter. The screen processor informs the ABAP processor about the subprogram that needs to be executed, which depends on the processing status of the screen flow logic.

If data needs to be exchanged with the database or the buffers during a dialog step, this exchange takes place through the database interface. This enables access to database tables, ABAP programs, or the ABAP Dictionary, among other things.

© Copyright . All rights reserved.

Prohibida su reproduccion

List the Work Processes Configured on an Instance of AS ABAP

Business Example

You want to gain an overview of the configured work processes and their current statuses.

Use transaction `SM50` to answer the following questions (alternatively, on *SAP Easy Access* screen, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *Process Overview*).

1. How many dialog work processes are there on the instance to which you are logged on?
2. Click *Refresh* several times in the process overview.
   Are your queries always processed by the same work process?

© Copyright . All rights reserved.

Prohibida su reproduccion

List the Work Processes Configured on an Instance of AS ABAP

Business Example

You want to gain an overview of the configured work processes and their current statuses.

Use transaction `SM50` to answer the following questions (alternatively, on *SAP Easy Access* screen, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *Process Overview*).

1. How many dialog work processes are there on the instance to which you are logged on?
   a) On the *SAP Easy Access* screen, run transaction code `SM50`. This action displays the numbers of dialog work processes.
   b) Count the work processes of the type *DIA*.
      Transaction `SM50` displays the work processes on the instance to which you are currently logged on.
      If you want to display the work processes for the entire SAP system, run transaction `SM66`. (In this case, you need to make the appropriate settings by using the *Select process* and the *Settings* pushbuttons.)
2. Click *Refresh* several times in the process overview.
   Are your queries always processed by the same work process?
   a) In transaction `SM50`, choose the *Refresh* pushbutton.
      Find your work process by finding your user name in the *User Names* column.
      You may find that your queries are processed by various work processes.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Describe the processing of requests in AS ABAP

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 3
Lesson 4

Explaining AS ABAP Processes

LESSON OVERVIEW
This lesson describes the processing of transactions in AS ABAP and explains the ABAP processes and their purposes.

Business Example

You want to find out what the term transaction means and what database and ABAP transactions are.

You also want to manage the lock entries. For this reason, you require the following knowledge:

- An understanding of the term transaction
- An understanding of database and ABAP transactions
- An understanding of lock management
- An understanding of the AS ABAP processes

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe the processing of transactions in AS ABAP
- Explain ABAP processes and their purposes

Transactions

Transactions are processing units that functionally belong together.

They have the following principal characteristics, which together form the acronym ACID:

- Atomic

  This means that a transaction is either fully successful or does not have any effects at all. If a transaction-oriented system goes down, inconsistent and partial results are not stored.

- Consistent

  This means that the system status changes from one status to another status where both are accurate and consistent in business terms.

- Isolated

  This means that the changes made within a transaction can be seen only by other transactions, even those transactions that run simultaneously, after the final confirmation (Commit).

© Copyright . All rights reserved.

Prohibida su reproduccion

- Durable

  The results of a transaction are durable because after the final confirmation, they are stored permanently in the database.

Database and ABAP Transactions

Every work process connects to a specific communication partner at database level for the duration of an SAP instance at runtime. Work processes cannot exchange communication partners at runtime. This is why a work process can only make changes to the database within one database transaction.

Database Transaction

In accordance with the ACID principle, a database transaction is a nondivisible sequence of database operations, at the beginning and end of which the dataset on the database must be consistent. The beginning and end of a database transaction are defined by a commit command (database commit) to the database system. During a database transaction (between two commit commands), the database system itself ensures that the dataset is consistent. The database system itself takes on the task of restoring the dataset to its previous state after a transaction has terminated with an error (rollback).

Business Transaction

Business transactions are processing units that are grouped to provide a specific function. These processing units execute changes to the database that are consistent and make sense in business terms. Typical examples of transactions are credit and debit updates, which only make sense together, or creating an order and reserving the relevant materials. Correspondingly, an AS ABAP transaction is defined as a nondivisible business process whose execution takes place completely or not at all. AS ABAP transactions are implemented as sequences of logically-related dialog steps that are consistent in business terms. One screen image represents one user dialog step.

Relationship Between Database and SAP Transactions

Figure 41: Relationship Between Database and SAP Transactions

© Copyright . All rights reserved.

Prohibida su reproduccion

The relationship between database and SAP transactions is shown in the figure.

It is not necessary that the execution of an SAP transaction takes place within one single dialog work process. Within a transaction that changes data in the database, the user requests database changes by using the displayed screens. Once the transaction is complete, the changes must result in a consistent database status. You can process a dialog box by various work processes (multiplexing of work processes). Each work process sequentially handles dialog steps for unrelated applications. Applications, for which dialog steps are executed by the same work process one after the other, cannot run within the same database transaction if they are unrelated. So, a work process must start a new database transaction for each dialog step.

Lock Management

Figure 42: Lock Management in AS ABAP

To ensure data consistency within an SAP system, you must ensure that more than one user cannot access and change data records at any one time. The SAP system has its own lock management concept.

From a database perspective, every dialog step forms physical and logical units, such as the database transaction. The database lock administration can coordinate only this type of database transaction.

From the SAP point of view, however, this is not sufficient. This is because the SAP transactions that are formed from a sequence of logically-related work steps, which are consistent in business terms, are made up of several dialog steps. SAP systems need to have their own lock management. SAP implements this by using the enqueue work process. This also ensures that the platform-independence of the lock management is maintained.

© Copyright . All rights reserved.

Prohibida su reproduccion

The SAP lock concept works on the principle that SAP programs make lock entries for the data records to be processed in a lock table. Lock entries can only be made if they do not exist in the lock table.

The enqueue work process manages the logical lock of the SAP transactions in the lock table. You can locate the lock table in the main memory of the instance where the enqueue work process is configured.

> Note:
> The instance for which the main memory contains the lock table is also known as the enqueue server.

Lock Request

If a user wants to change any data, the executing dialog work process will request a lock (the application developer must program this request explicitly).

If a dialog request is processed on the enqueue server, the dialog work process can access the lock table directly. The dialog work process checks whether a new lock can be generated, that is, whether there is a collision with the locks that have already been set. If a lock can be set, the dialog work process creates it and the user (lock owner) is given the lock key. The lock key is kept in the user context in the shared memory.

If the dialog work process that processes the user request and the enqueue work process are not running on the same instance, these two work processes communicate through the message server. In this case, the lock request is forwarded from the dialog work process to the enqueue work process through the dispatchers and the message server.

The enqueue work process now checks whether a lock can be set. If this is possible, the lock is set by the enqueue work process. The lock key is transferred to the requesting dialog work process through the dispatcher and the message server.

When the lock is requested, the system checks whether the requested lock conflicts with the existing entries in the lock table. If the lock table already contains the corresponding entries, the lock request is refused. The application program can then inform the user that the requested operation cannot currently be executed.

> Note:
> For the system to execute lock requests, the lock object must be defined in the ABAP Dictionary. The lock object contains one or more tables whose entries are to be locked. You can define a lock mode for a lock object.

Lock Modes

The application developer can choose the following lock modes:

- Write locks (lock mode exclusive)

  Only one user can edit the lock data. The requests for another write lock and another read lock are rejected. A write lock protects the locked objects against all types of other transactions. Only the same lock owner can set the lock again (cumulate).

© Copyright . All rights reserved.

Prohibida su reproduccion

- Read locks (lock mode shared)

  Several users can have read access to the locked data at the same time. The requests for additional read locks are accepted even if they are from other users. A write lock is rejected.

- Enhanced write locks (lock mode exclusive noncumulative)

  While the same transaction can successively request and release the write locks, you can request an enhanced write lock only once, even by the same transaction. All other requests for locks are rejected.

- Optimistic locks (lock mode optimistic)

  Optimistic locks respond like read locks at first and can be changed to write locks. An optimistic lock is set if the user displays the data in change mode. Optimistic locks on the same object do not collide. If the user wants to save the (changed) data, the optimistic lock must be changed to a write lock (mode E). This fails if someone sets a nonoptimistic lock on the object earlier. Other optimistic locks on the object are deleted in the process.

The locks set by an application program are released either by the application program itself or by the update program once the database has been changed. Locks that have been passed to an update work process are also written to a file at operating system level and can be restored if the enqueue server goes down.

Transaction `SM12` (*Tools → Administration → Monitor → Lock Entries*) displays the locks that are currently set. If a lock has been inherited by the update process, the backup flag is also set. Such a lock is also included in the lock table again after restarting the enqueue server.

Removal of a Lock

There are two ways of removing the locks held by users, as follows:

- Ending the user session in the user overview transaction `SM04` (*Tools → Administration → Monitor → System Monitoring → User Overview*)
- Manually deleting the lock entries in `SM12`

The first method (ending the user session) also results in the original lock owner leaving the transaction and releasing all locks held. The second method (manually deleting using `SM12`) merely deletes the lock entry from the lock table. This method theoretically enables several users to change the same data records simultaneously.

> Caution:
> Before deleting locks, the system administrator must first check (using transaction `SM04`) whether the user who owns the lock is still logged on to the system. You must only delete lock entries with transaction `SM12` if the lock owner is no longer logged on to the system but still owns the lock (for example, if the connection between the SAP GUI and the SAP system has been broken because the user has switched off his or her front-end computer without logging off from the system).

Stand-alone Enqueue Server/ABAP Central Services

To ensure high availability, the enqueue work process together with the ABAP message server can also be detached from the central instance and installed as an ABAP central services

© Copyright . All rights reserved.

Prohibida su reproduccion

instance (ASCS). For more details, see the SAP NetWeaver Library and choose *The SAP Lock Concept → Standalone Enqueue Server*.

Processing of Updates

Figure 43: The Principle of Asynchronous Updates

Very often a SAP transaction, which maps a business transaction (for example, the creation of an order) contains several screen changes. Data changes effected by this process are supposed to be executed completely or not at all in the database. If an error occurs during the transaction and the operation terminates, the transaction is not supposed to make any database changes at all. The SAP update system takes care of this.

Update System and Process

The update system offers increased security and performance. The system also offers the ability to restore data during the execution of database changes.

The updating system uses a technology that allows SAP transactions to offload time-intensive database changes. These are then carried out asynchronously in special update work processes. It also circumvents the rollback problems caused by the difference in the conception of the logical unit of work (LUW) in an SAP transaction and the database.

If data that is temporarily stored for processing is passed to an update work process for further processing during a dialog work process, the dialog work process does not wait for the update request to be completed. The update is asynchronous (not simultaneous).

The dialog part completes with the ABAP command COMMIT WORK, and the update part of the transaction starts.

The update server transfers the update request to an update work process. Here, each dialog step corresponds to a database transaction (which is executed either completely or not at all in the database and is completed with a COMMIT command). The system executes the update part

© Copyright . All rights reserved.

Prohibida su reproduccion

of the SAP transaction in one database transaction. It is only at this point that the system copies the data to the application table.

Asynchronous Updates

Figure 44: Asynchronous Updates

If users want to change a data record in an SAP transaction, they call the corresponding transaction in the dialog mode, make the appropriate entries on the screens, and initiate the update process by saving the data.

Steps in Asynchronous Updates

The asynchronous update process triggers the following steps:

1. The program locks the data record against other users by addressing the enqueue work process (using the message server, if appropriate). The enqueue work process makes the relevant entry in the lock table. If another user has already locked the data, the enqueue work process informs the user that the data record cannot currently be changed.

2. If the enqueue work process succeeds in writing the lock entry to the lock table, it passes the lock key that it created to the user. The program reads the record to be changed from the database. The user can change the record on the screen image of the SAP transaction.

3. In the active dialog work process, the program calls a function module by using CALL FUNCTION ..... IN UPDATE TASK and writes the update request to the database update tables. These are also called VB* tables because their names begin with VB. They act as temporary memory and store the data to be changed until it can be collected and written to the application tables in the database (in a single database transaction).

© Copyright . All rights reserved.

Prohibida su reproduccion

4. At the end of the dialog phase of the transaction (for example, when the user saves the data – possibly after completing other dialog steps), the program initiates the close of the transaction with the COMMIT WORK statement. The work process that is handling the active dialog step completes the update header and triggers an update work process.
5. Based on the information (key of the update order and lock key) transferred from the dialog work process, the update work process reads the log records that belong to this SAP transaction from the VB* tables.
6. The update work process transfers the changes marked and collected in the VB* tables to the database as an update request and evaluates the database response. If the changes were successfully written to the target tables, the update work process triggers a database commit after the last change to the database and deletes the entries from the VB* tables. If an error occurs, the update work process triggers a database rollback, leaves the log records in the VB* tables, and marks them as defective.
7. The lock entries in the lock table are reset.

> Hint:
> Behind the VB* tables, there are VBHDR (update headers), VBMOD (update modules), VBDATA (update data), and VBERROR (possible error information), the table structures can be displayed using the ABAP Dictionary (transaction `SE11`).

> Note:
> The application developer decides whether and how to use asynchronous updates while programming the transaction. Besides the asynchronous update, there are some other update techniques (for example, synchronous or local).

Types of Updates

To improve performance further, application developers can configure different types of updates as follows:

- Time-critical, primary V1 updates

  These updates are relevant to objects that have controlling functions in the SAP system, such as a change to the material stock or an order creation.

- Non-time-critical, secondary V2 updates

  These updates depend on the V1 updates. These updates are, for example, purely statistical updates such as the calculation of results.

- Non-time-critical updates

  These updates are collected and processed at a later point in time (collective run).

The V1 modules for an SAP transaction are processed sequentially in a single update work process. If an SAP system has a work process for V2 updates (type UP2), then V2 modules will only be updated on the SAP system. Once it has successfully completed processing, the V1

© Copyright . All rights reserved.

Prohibida su reproduccion

update work process releases the relevant locks again. This means that the normal update work processes are available again more quickly for time-critical V1 updates and the relevant lock entries are deleted sooner. If you have not configured any V2 update work processes, then the V1 work process handles all updates.

Modules are not updated automatically in the collective run. The modules are updated only when a special report (report RSM13005) triggers the update. All calls of the function modules are then collected, aggregated, and updated at once. The function modules are handled like V2 update modules.

If an error occurs during an update, then the processing of the active update component terminates. Users get a notification automatically by express mail when an update terminates.

If a dialog work process terminates while writing data to the VB* tables, the tables contain data that may not be updated. This data is deleted automatically the next time you start the system. The data can also be deleted manually. The application tables remain unchanged after the data is deleted.

An asynchronous update may terminate for various reasons. If, for example, several attempts are made to enter the same data record (using insert) in a table, this triggers the exception condition Duplicate Key in the coding because an entry already exists in the table under this key. Therefore, the corresponding data record cannot be written to the database table more than once.

When an update terminates, the system sends an express mail to the user who triggered the update. Any additional steps must be carried out by the system administrator. Transaction `SM13` (update requests) provides system administrators with analysis tools to handle the terminated updates. Once the error that caused the termination has been corrected (for example, hardware damage repaired), the end user needs to restart the processing.

© Copyright . All rights reserved.

Prohibida su reproduccion

Set and Monitor Lock Entries

Business Example

You want to know how enqueue (lock) handling works in AS ABAP.

Learn how lock administration works in SAP systems.

1. Change your user data.
2. Open a new window and check the list of lock entries to determine the tables on which a lock has been set.
3. In another window, try changing your own user data again.
4. Close the window displaying your user data that is ready for change.
   What happens to the lock entries?

© Copyright . All rights reserved.

Prohibida su reproduccion

Set and Monitor Lock Entries

Business Example

You want to know how enqueue (lock) handling works in AS ABAP.

Learn how lock administration works in SAP systems.

1. Change your user data.
   a) On the *SAP Easy Access* screen, run transaction code `SU3` for maintaining some of your user data (alternatively, choose *System* → *User Profile* → *Own Data*).

   > Note:
   > Your user data is now exclusively locked at the SAP system level.

2. Open a new window and check the list of lock entries to determine the tables on which a lock has been set.
   a) On the *SAP Easy Access* screen, choose *System* → *Create Session* to open a new window.
   b) Run transaction code `SM12` (alternatively, choose *Tools* → *Administration* → *Monitor* → *Lock Entries*).
   c) Press ENTER without changing the default values.
   You see a lock entry (held by your user) for your data record in table *USR04*.

3. In another window, try changing your own user data again.
   a) On the *SAP Easy Access* screen, (choose *System* → *Create Session*) run transaction `SU3` again to open the second window.

   > Note:
   > The system returns an information message that the requested data is already in change access by another user (in this case, yourself). Data locked can still be viewed. Because transaction `SU3` is always in change mode, you are required to use transaction `SU01` to display the user data that is currently being changed.

4. Close the window displaying your user data that is ready for change.
   What happens to the lock entries?

© Copyright . All rights reserved.

Prohibida su reproduccion

a) To end your user change window, run transaction code `SM04` (alternatively, choose *Tools → Administration → System Administration → Monitor → System Monitoring → User Overview*).

b) Select your user and choose the *Sessions* pushbutton.

c) Position the cursor on the *Maintain Users Own Data* row and choose the *End Session* pushbutton.

d) If you now select your current lock entries by using transaction `SM12`, the lock on table *USR04* will not appear.

> Note:
> If an action terminates in an SAP system (whether deliberately or not), the locks held by the action are deleted.

© Copyright . All rights reserved.

Prohibida su reproduccion

Printing

Figure 45: Printing in AS ABAP

SAP systems provide a wide variety of options for representing business and other data. This data, which is created and formatted in a dialog step, can then be sent to printers and other output interfaces (fax, e-mail, and so on). You must first set up the printer in the system before addressing.

Select a printer that has already been set up by choosing the *Print* command (*Ctrl* + *P*) and then using the F4 help. A standard printer is usually set as default in your user profile.

Once a printer has been set up, the SAP system has all the information it needs to create a spool request.

A spool request contains information about the data to be output, its formatting, and the printer model used. The spool request generated is stored in the TemSe (temporary sequential file).

> Hint:
> Spool requests can be created by dialog work processes or background work processes. Spool work processes do not create spool requests.

A spool work process formats the data specified in the spool request and creates an output request. The output request contains all data in an appropriate format for the printer. This data can either be passed on to an appropriate operating system spool process locally (on the same computer) or remotely (over a network connection).

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
In an SAP system, the connection between a spool work process and the operating system spool process is known as the access method. There are access methods other than those we have discussed. These are the two most commonly used access methods for connecting printers to SAP systems. In this context, remote or local do not refer to the physical location of the output but to the place where the spool work process is connected to the operation system spool process.

For print processing, the best performance is achieved by sending the data to be printed to the operating system as soon as possible. To send the data to be printed, use the local access method. The operating system then performs all remaining tasks, such as queuing and transferring the data to the selected printer.

Hint:
Each selectable printer must print at the operating system level.

You can display your spool and output requests by choosing *System → Own Spool Requests* (transaction `SP02`).

Choose *System → User Profile → Own Data* (transaction code `SU3`) to specify personal settings for printing on the *Defaults* tab page in the *Spool Control* section.

Note:
Course ADM100 provides additional information on printing and setting up printers in AS ABAP.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Print a Screenshot in AS ABAP

Business Example

System administration needs a list of the work processes set up on each instance. For this reason, you must be able to print a screenshot in AS ABAP. You must also be able to explain the difference between spool requests and output requests.

Print a simple list and generate an output request from a spool request.

Print the list of work processes.

1. Call transaction `SM50`. Print the list.
2. Display your own spool requests and generate an output request from the spool request.

© Copyright . All rights reserved.

Prohibida su reproduccion

Print a Screenshot in AS ABAP

Business Example

System administration needs a list of the work processes set up on each instance. For this reason, you must be able to print a screenshot in AS ABAP. You must also be able to explain the difference between spool requests and output requests.

Print a simple list and generate an output request from a spool request.

Print the list of work processes.

1. Call transaction `SM50`. Print the list.
   a) Call transaction `SM50` (alternatively, choose *Tools* → *Administration* → *Monitor* → *System Monitoring* → *Process Overview* and choose *Print* (Ctrl + P).
   b) Use the *F4* help to select an *Output Device*, such as LP01, and choose *Continue*.
   c) Choose *Continue* to confirm your setting in the displayed dialog box.
2. Display your own spool requests and generate an output request from the spool request.
   a) On the *SAP Easy Access* screen, choose *System* → *Own Spool Requests*.
      Check the status and the number of pages of your spool request.
   b) Select your spool request and choose *Print* directly pushbutton to create an output request for the spool request you selected.
   c) Select your spool request and choose the *Output* Requests pushbutton to display the output requests for your spool request.
      The status text Waiting for output formatter is displayed. Because there is no printer connected to the training system, you will not be able to print to paper.

© Copyright . All rights reserved.

Prohibida su reproduccion

Background Processing

Figure 46: Scheduling Background Tasks (Jobs)

SAP background processing is a method for automating routine tasks and optimizing the SAP computing resources of your organization.

The characteristics of background processing are as follows:

- You can use background processing to automatically start scheduled programs for you.
- You can use background processing to execute long-running or resource-intensive programs at off-peak times.
- You can use it to assign the task to execute reports and programs to the system. There is no strain on your dialog resources and reports that are run in the background are not subject to the runtime restrictions of dialog processing (termination of the program after a runtime of 10 minutes).

The segregation of background processing to special work processes gives an additional dimension for separating background processing and interactive work. Background processing and interactive work in the system take place at different times. For example, the system is used interactively during the day and background processing takes place at night. You can use background work processes to separate background processing and interactive work by servers. This is because background jobs are only executed on servers that offer background processing.

Execution of Programs in the Background

Execution of a program in background processing takes place as follows:

- An end user can schedule the program to be started in the background as a job from the application transaction.

© Copyright . All rights reserved.

Prohibida su reproduccion

- The job then waits for the planned execution time in the job scheduling table.
- If the planned execution time of the job arrives and free background work processes are available, the job is distributed to a background work process by the background scheduler and then executed.
- Users can display the result in the application transaction or, in case of list generating programs, look at the spool request belonging to the job (see the printing section). To display your own jobs, choose *System → Own Jobs* on the *SAP Easy Access* screen (transaction code `SMX`).

The system administration and the work preparation have access to a tool for scheduling various background tasks with transaction `SM36` (*Tools → CCMS → Background Processing → Define Jobs*). The system wide monitoring of jobs takes place with transaction `SM37` (*Tools → CCMS → Background Processing → Jobs – Overview and Administration*).

For cross-system scheduling and monitoring of background tasks, use SAP Central Process Scheduling (SAP CPS) and other licensed partner products.

© Copyright . All rights reserved.

Prohibida su reproduccion

Schedule a Simple Report in the Background

Business Example

As a system administrator or an end user, you need to schedule the execution of a report in the background.

Schedule and monitor jobs. Differentiate between executing a report in dialog mode and background mode. Schedule the execution of a simple job.

1. On the *SAP Easy Access* screen, run transaction code `SA38` (*System* → *Services* → *Reporting*) and select report `RSUSR000` in the dialog mode.
2. Call transaction `SA38` to execute report `RSUSR000` in the background.

   Display the result of the job.

© Copyright . All rights reserved.

Prohibida su reproduccion

Schedule a Simple Report in the Background

Business Example

As a system administrator or an end user, you need to schedule the execution of a report in the background.

Schedule and monitor jobs. Differentiate between executing a report in dialog mode and background mode. Schedule the execution of a simple job.

1. On the *SAP Easy Access* screen, run transaction code `SA38` (*System* → *Services* → *Reporting*) and select report `RSUSR000` in the dialog mode.
   a) On the *SAP Easy Access* screen, run transaction `SA38`.
   b) On the *ABAP Editor: Initial Screen* screen, enter the report name `RSUSR000` in the *Program* field and choose the *Execute* pushbutton.

      Users who are logged on are listed.
2. Call transaction `SA38` to execute report `RSUSR000` in the background.

   Display the result of the job.
   a) On the *SAP Easy Access* screen, run transaction code `SA38`.
   b) On the *ABAP: Program Execution* screen, enter the report name `RSUSR000` in the *Program* field.
   c) Choose the *Background* pushbutton.
   d) Choose the *Execute immed* pushbutton. You can also choose the *Schedule* pushbutton.

      Make sure that the *Job Name* and *Execution Time* have been entered.

      Ensure that the execution time is not too far in the future (about 2 minutes, for example). Otherwise, you will not be able to see the result of the job soon.
   e) Then choose the *Schedule once* pushbutton.
   f) On the *Schedule Report* screen, select *System* → *Own Jobs*.

      You will see the job you just scheduled in the displayed list.

      If the job is marked as “released”, you have to wait until the execution time you selected has passed. If the job is marked as “finished”, it is finished, and you can display the result in the form of a spool list.

      To display the result, select the job and then choose the *Spool* pushbutton. Select the spool request and choose the *Display contents* pushbutton. You can see a list of users who were logged on at the time the job was executed.

© Copyright . All rights reserved.

Prohibida su reproduccion

Communication Through the Gateway

Figure 47: Gateway Communication

Each instance of an AS ABAP system contains a gateway. The gateway is used for communication between the work processes of various instances or SAP systems. The gateway is also used for communication between work processes and external programs. The gateway reader (usually called a gateway) is the main process of the gateway system. The dispatcher starts the gateway and checks it periodically.

During the communication between the instances of a system or systems by using Remote Function Call (RFC) or Common Programming Interface for Communications (CPIC), the gateway is always involved. If a dialog work process has to establish an RFC connection to a remote system in the context of a request, for example, to fetch customer data, it uses the gateway. The gateway then takes care of the communication with the remote system. The gateway forwards the request to the gateway of the remote system. The remote gateway transfers the request to the dispatcher, which in turn forwards the request to one of its work processes. The work processes then communicate directly with the gateway of the dispatcher.

Inbound RFC connections are always received by the gateway. Outbound connections are initiated by the work process.

© Copyright . All rights reserved.

Prohibida su reproduccion

Processing Web Requests

Figure 48: Processing Web Request for 7.0x or Lower

The Internet Communication Manager enables SAP systems to communicate using the HTTP, HTTPS, and SMTP protocols. The ICM can process requests from the Internet that include its server/port combination in their URLs. HTTP(S) requests can be processed either in the ABAP work process (for example, ABAP Web dynpro applications) or forwarded to AS Java (for example, Java Web dynpro applications). The ICM can use the URL to decide to where it forwards the request if it is a dual-stack. If database data is required for AS ABAP, a commection to a work process is created using memory pipes.

If the request is directed at AS Java, it is sent to the Java dispatcher (2b) for SAP NetWeaver 7.0 or lower, which then forwards it to a Java server process (3b). Java Dispatcher is replaced by ICM in SAP NetWeaver 7.1x or higher. The Java server process then refers to the database's Java schema, if applicable, and sends the response back to the ICM. Finally, the ICM sends the response from the SAP system to the user who sent the request (5).

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Describe the processing of transactions in AS ABAP
- Explain ABAP processes and their purposes

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Java is an object-oriented and a platform-independent programming language.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

2. In a two-tier client/server configuration, each layer runs on its own hosts.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

3. What are the advantages of a three-tier client/server configuration as compared with a single-tier or two-tier configuration?

   Choose the correct answers.

   - [ ] A Simpler scalability
   - [ ] B Simpler administration
   - [ ] C Load balancing
   - [ ] D None of the above

4. The Software Deployment Manager (SDM) is the standard tool used to install ABAP software components.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

5. The gateway reader enables communication between SAP systems or between SAP systems and external application systems.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

6. Data does not need to be read from the database every time it is required in an instance. But the data can be called quickly from the buffer.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

7. _____ is used to manage large sets of data.

   Choose the correct answer.

   - [ ] A A Graphical User Interface (GUI)
   - [ ] B A Relational Database Management System
   - [ ] C A Dynpro processor
   - [ ] D An ABAP dispatcher

8. Identify the types of work processes in an AS ABAP system.

   Choose the correct answers.

   - [ ] A Dialog work process
   - [ ] B Message Server process
   - [ ] C Update work process
   - [ ] D Internet Communication Manager (ICM) process
   - [ ] E Background work process
   - [ ] F Enqueue work process
   - [ ] G Spool work process

© Copyright . All rights reserved.

Prohibida su reproduccion

9. In _____, several users can have read access to locked data at the same time.

   Choose the correct answer.

   - [ ] A Write locks
   - [ ] B Read locks
   - [ ] C Enhanced write locks
   - [ ] D Optimistic locks

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Java is an object-oriented and a platform-independent programming language.

   Determine whether this statement is true or false.

   [X] True

   [ ] False

   The Java programming language was first introduced by Sun Microsystems Inc.™ in 1995. Java is an object-oriented and platform-independent programming language that has spread across many areas.

2. In a two-tier client/server configuration, each layer runs on its own hosts.

   Determine whether this statement is true or false.

   [ ] True

   [X] False

   In a three-tier configuration, each layer runs on its own hosts. Many application servers can simultaneously work with the data of a database server.

3. What are the advantages of a three-tier client/server configuration as compared with a single-tier or two-tier configuration?

   Choose the correct answers.

   [X] A Simpler scalability

   [ ] B Simpler administration

   [X] C Load balancing

   [ ] D None of the above

   In a three-tier configuration, implementing an additional hardware layer for application processes makes it easier to adapt an SAP system if the number of users changes (scalability). The additional layer also makes it easy to assign user groups to specific application servers (software-oriented view) (load balancing). The additional hardware layer does not, however, reduce the administrative workload.

© Copyright . All rights reserved.

Prohibida su reproduccion

4. The Software Deployment Manager (SDM) is the standard tool used to install ABAP software components.

   Determine whether this statement is true or false.

   ☐ True

   ☒ False

   The SDM is the standard tool used to install Java software components on SAP NetWeaver AS Java.

5. The gateway reader enables communication between SAP systems or between SAP systems and external application systems.

   Determine whether this statement is true or false.

   ☒ True

   ☐ False

   The gateway reader enables communication between SAP systems or between SAP systems and external application systems. There is one gateway reader per dispatcher.

6. Data does not need to be read from the database every time it is required in an instance. But the data can be called quickly from the buffer.

   Determine whether this statement is true or false.

   ☒ True

   ☐ False

   Data that is read but seldom changed (for example, programs or customizing data such as clients, currencies, or company codes) can be kept as a copy of the database content in the shared memory of the application server. This means that the data does not need to be read from the database every time it is required. But it can be called quickly from the buffer. Each instance has its own buffers.

© Copyright . All rights reserved.

Prohibida su reproduccion

7. _____ is used to manage large sets of data.

   Choose the correct answer.

   - [ ] A A Graphical User Interface (GUI)
   - [x] B A Relational Database Management System
   - [ ] C A Dynpro processor
   - [ ] D An ABAP dispatcher

   Relational Database Management Systems are used to manage large sets of data.

8. Identify the types of work processes in an AS ABAP system.

   Choose the correct answers.

   - [x] A Dialog work process
   - [ ] B Message Server process
   - [x] C Update work process
   - [ ] D Internet Communication Manager (ICM) process
   - [x] E Background work process
   - [x] F Enqueue work process
   - [x] G Spool work process

   In an AS ABAP system, the dialog, update, background, enqueue, and spool work processes are present.

9. In _____, several users can have read access to locked data at the same time.

   Choose the correct answer.

   - [ ] A Write locks
   - [x] B Read locks
   - [ ] C Enhanced write locks
   - [ ] D Optimistic locks

   In read locks, several users can have read access to the locked data at the same time.

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 4 Communication and Integration Technologies

Lesson 1

Integrating Existing Applications with SAP Applications 130

Exercise 9: Use a Business Application Programming Interface 137

Lesson 2

Linking Cross-System Business Processes 140

Lesson 3

Working with Web Services on SAP NetWeaver Application Server 143

Lesson 4

Explaining Service-Oriented Architecture 147

UNIT OBJECTIVES

- Explain RFCs, the Business Object Repository (BOR), and BAPIs
- Describe ALE
- Work with Web services on SAP NetWeaver AS
- Describe SOA in the context of SAP
- Differentiate between Web and enterprise services for SOA

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 4
Lesson 1

Integrating Existing Applications with SAP Applications

LESSON OVERVIEW

This lesson provides an overview of the interface technologies available in an SAP system, while focusing on the significance of a remote function call (RFC) and Business Application Programming Interfaces (BAPIs).

Business Example

You need to integrate existing applications with SAP applications. Standard interfaces are of particular interest in the integration. For this reason, you require the following knowledge:

- An understanding of various interface technologies used by SAP systems
- An understanding of how to create an RFC
- An understanding of the use of business objects and their BAPIs

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain RFCs, the Business Object Repository (BOR), and BAPIs

© Copyright . All rights reserved.

Prohibida su reproduccion

Overview of Interfaces

Figure 49: Interface Technologies Used in SAP Systems

SAP systems have interfaces at various communication levels, ranging from highly technical connection options, for example, using the TCP/IP protocol or CPI-C, to highly specialized interfaces designed for business objects, such as BAPIs or the IDoc Interface used in the Application Link Enabling (ALE) environment. Higher interfaces that access business objects or processes use the same technology –RFCs.

SAP systems use the following interface technologies as listed in the figure:

- ALE
- BAPIs
- Common Program Interface Communication (CPI-C)
- Electronic Data Interchange (EDI)
- HyperText Transfer Protocol (HTTP(S))
- Logical Unit 6.2 (LU 6.2)
- RFC
- Object Linking and Embedding (OLE)
- Simple Mail Transfer Protocol (SMTP)
- Simple Object Access Protocol (SOAP)
- Transmission Control Protocol/Internet Protocol (TCP/IP)
- Extensible Markup Language (XML)

© Copyright . All rights reserved.

Prohibida su reproduccion

Remote Function Call

Figure 50: Options for Connections Using Remote Function Call

The RFC interface is an SAP interface protocol based on CPI-C and TCP/IP and simplifies the programming of communication processes between various systems. RFCs enable you to call and execute predefined functions in a remote system or within the same system. RFCs manage the communication process, transfer of parameters within a system or between systems, and error handling.

RFC describes an interface, not the programming language in which the function runs. When RFC communication is used between two SAP systems, the calling system uses an RFC definition in the other system to access a specific function. You can also use RFCs to call functions in non-SAP systems.

If you want to remotely start external programs, you need an RFC interface outside the SAP system. This could be, for example, a dynamic link library (DLL). Every RFC interface is bidirectional, which enables external programs to use RFC to access functions in SAP systems.

> Note:
> All function modules (including those that are remote-enabled) are created, together with their import and export parameters by using the Function Builder. To call the Function Builder, choose *Tools → ABAP Workbench → Development → Function Builder* or use transaction code `SE37`.

To call an RFC module from an SAP system, you must know the import and export parameters (defined in the Function Builder). Also, a technical connection must exist between the two systems. This connection is called an RFC connection or an RFC destination.

© Copyright . All rights reserved.

Prohibida su reproduccion

You can manage RFC connections by choosing *Communication and Integration Technologies → RFC Destinations (Display/Maintain)* or using transaction `SM59`.

Remote Function Call in Detail

Figure 51: Remote Function Call in Detail

The figure shows the calling system, in which an RFC destination named DEST has been created. Do not confuse an RFC destination in transaction `SM59` with an SAP system because an RFC connection can only point to one client in an SAP system. These connections are also referred to as connections between logical systems.

You can have at least as many RFC connections between two systems as there are clients in the target system. Because you can specify a logon user for the destination in each RFC connection, you can also access clients in the target system several times, for example, with a different logon user each time. If you need a bidirectional RFC connection between two systems – that is, you need the system called to also execute RFC modules in the calling system – then you must set up an equivalent RFC connection in the system that is called.

When defining RFC destinations in SAP systems, based on AS ABAP, RFC connections always address one client when targeting an SAP system, based on AS ABAP. The connections are usable from all clients within the system (provided authorizations allow).

In ABAP, you use RFCs to call a function module in another system. Calling the function module is done as follows:

```
CALL FUNCTION <Name>
DESTINATION <Target>
```

EXPORTING ...

IMPORTING ...

© Copyright . All rights reserved.

Prohibida su reproduccion

The function to be executed in the target system is named. The name of the target must refer to one of the RFC connections available. When you are creating an RFC connection, you must specify logon data for the target system or enter the logon parameters when you start the RFC. Exporting and Importing are used to pass parameters to the target function and to receive the parameters returned by the target function. The function called in the target system is executed using the user ID entered for the connection.

> Note:
> You can also create RFC connections for which the user making the call is used in the target system. That means various users can use the same connection in the target system. This procedure is also called Trusted RFC. It is a prerequisite that identical users are created in the source and target systems.

The RFC has become the most important interface in the SAP environment. Some special RFC modules, which follow certain conventions, are also known as BAPIs.

Business Object Repository and Business Application Programming Interfaces

Figure 52: Business Object Repository and Business Application Programming Interfaces

A BAPI is a standardized programming interface that facilitates internal and external access to business processes and data in SAP systems. A BAPI is used with SAP Business Workflow. BAPIs are defined in the BOR as methods of SAP business objects and enable an object-oriented view of business data in an SAP system. Functions that can be called using BAPIs are implemented and stored in the Function Builder (in the ABAP Workbench) as RFC-enabled function modules. You can display an overview of the available BAPIs in the BOR, for example, by activating the Business Object Repository pushbutton in the Business Object Builder. To access the Business Object

© Copyright . All rights reserved.

Prohibida su reproduccion

Builder, choose *Tools → ABAP Workbench → Development → Business Object Builder* or use transaction `SWO1`. You can access the BOR directly by using transaction code `BAPI`.

BAPIs are used in various contexts for:

- Linking business processes across system boundaries (for example, when using ALE)
- Integrating various solutions in the framework of SAP Business Suite
- Connecting an SAP system to the Internet
- Connecting to external programs

Note:
BAPIs are created and tested in the same way as other function modules, using the Function Builder or transaction `SE37`. The BAPIs are then defined as BAPIs in the BOR.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Use a Business Application Programming Interface

Business Example

You require user data from another SAP system.

Use a BAPI to display the address data for your user in the system.

1. Start the overview transaction for the BOR.
2. Find the *USER.Change* method. Display the documentation for this method.
3. Call the *Function Builder* for the *USER.Display* method. You can now see the source code for the BAPI in the *Function Builder*.
4. Execute this BAPI for your user with the RFC destination NONE.
5. Once you confirm the dialog box, the result screen for the query is displayed.
   You can view the return code and the response time for your query.

© Copyright . All rights reserved.

Prohibida su reproduccion

Use a Business Application Programming Interface

Business Example

You require user data from another SAP system.

Use a BAPI to display the address data for your user in the system.

1. Start the overview transaction for the BOR.
   a) On the *SAP Easy Access* screen, choose *Tools → ABAP Workbench → Development → Business Object Builder* or enter transaction `SWO1`.

      If you are following the Business Object Builder menu path, select *BAPI* in the dialog box.
2. Find the *USER.Change* method. Display the documentation for this method.
   a) Choose the *Alphabetical* tab page. Then, on the *BAPI Explorer* screen, look for the *User* business object.

      From the BAPIs available for this business object, choose the *USER.Change* BAPI.

      Use the appropriate tab page to display the documentation for this BAPI on the right side of the *BAPI Explorer* screen.
3. Call the *Function Builder* for the *USER.Display* method. You can now see the source code for the BAPI in the *Function Builder*.
   a) Switch to the *USER.Display* BAPI.
   b) On the *Tools* tab page, choose the *Function Builder*.
   c) Choose the *Display* pushbutton.
4. Execute this BAPI for your user with the RFC destination NONE.
   a) On the *Function Builder* screen, choose the *Test/Execute* pushbutton.
   b) On the *Test Function Module: Initial Screen*, enter **NONE** in the *RFC target system entry* field (which refers to your system).
   c) Enter your user name on the query screen (for example, SAPTEC-04).
   d) Choose the *Execute* pushbutton.

      Your user data, which is requested through RFC by the BAPI, is displayed.
5. Once you confirm the dialog box, the result screen for the query is displayed.

   You can view the return code and the response time for your query.
   a) The return code for your query is displayed as the value for the export parameter *Return* on the *Test Function Module: Result* Screen.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Explain RFCs, the Business Object Repository (BOR), and BAPIs

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 4
Lesson 2

Linking Cross-System Business Processes

LESSON OVERVIEW
This lesson explains the fundamentals of the Application Link Enabling (ALE) concept.

Business Example

Your company wants to implement the e-commerce functions of SAP Customer Relationship Management (CRM) for selling goods to customers on the Internet. For this reason, you require the following knowledge:

- An understanding of the cross-system business processes
- An understanding of the ALE concept

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe ALE

Significance of Cross-System Business Processes

Let us start by defining cross-system business processes, using common situations as examples.

For example, within a company, the human resources system may be separate from the rest of the business software system but the systems cannot be completely separate because the accounting system needs the wage data of the employees. In this situation, you need cross-system business processes to exchange the relevant data.

An example of when cross-system business processes are used is when two companies collaborate closely and send joint orders to a vendor. The business IT system of the companies needs to communicate with each other to consolidate the quantities to be ordered. In this case, the business process not only crosses system boundaries, but also company boundaries.

An additional example of when cross-system business processes are used is in the transfer of a limited quantity of specific data, for example, the electronic transfer of account statement data from a bank to a company. Developments in the past suggest that the significance of cross-system business processes continues to increase rapidly.

© Copyright . All rights reserved.

Prohibida su reproduccion

Application Link Enabling (ALE)

Figure 53: Distribution of Business Processes by Using Application Link Enabling

ALE is a means of creating and operating distributed applications. The basic concept of ALE is to ensure the operation of a distributed, yet integrated, system landscape. This involves business-controlled message exchange by using consistent data across loosely linked application systems. The applications are integrated through synchronous and asynchronous communication, not through a central database.

Systems that use ALE to exchange data can belong to the same company, or they may belong to separate companies. One of the characteristics of ALE is that different systems are linked in business terms through secure and consistent data transfer.

You can also describe ALE in terms of the elements: who exchanges data, which data is exchanged when, with whom the data is exchanged, and by what means.

Implementing Application Link Enabling

To implement ALE, clarify the following points:

- Identify the business process and the objects involved.
- Identify the information to be transmitted.
- Specify the format for the data to be transferred.
- Decide the technology to be used for transfer.
- Decide the type of transfer.
- Specify the destination of the data transfer.

© Copyright . All rights reserved.

Prohibida su reproduccion

The following table contains an example of data synchronization in a business process:

Process	e-Commerce for Consumers Using SAP CRM
Identify the information to be transmitted	Sequence of data from the SAP CRM system, which is to be passed to an ERP back end
Specify the format of the data to be transferred	IDoc format
Decide the technology for transfer	By remote function call (RFC)
Decide the type of transfer	Asynchronously, every 60 seconds
Decide the objective of transfer	Provision of goods and/or services for sale on the Internet

We identify data within an SAP system by using a business object and its Business Application Programming Interfaces (BAPIs). A BAPI is a method of a business object, for example, the material master record. A permissible method could be creating or changing the material master data. BAPIs enable you to edit all data belonging to an object.

The IDoc format describes the structure of intermediate documents. There are various kinds of IDoc formats for different types of data to be exchanged. ALE can also be used to transfer data in an agreed XML format.

You can select your preferred data transfer technology within the constraints imposed by a system. For example, you can transfer data by RFC or using HTTP or HTTPS.

Types of Transfers

The following are the basic types of transfer:

- Synchronous
- Asynchronous

Synchronous transfer means that the data is transferred when it is created or changed. You can start asynchronous transfers at intervals of your choice.

There are few restrictions when trying to link systems. The systems being linked must have the technical capability to receive the communications (RFC-enabled and HTTP-enabled) and interpret the format transferred (IDoc or XML). SAP systems of different releases can be linked using ALE.

> Note:
> BIT300 and BIT350 are more advanced training courses on ALE.

LESSON SUMMARY

You should now be able to:

- Describe ALE

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 4
Lesson 3

Working with Web Services on SAP NetWeaver Application Server

LESSON OVERVIEW
This lesson introduces the subject of Web services and their administration.

Business Example

Your company wants to implement online services using Web services. For this reason, you require the following knowledge:

- An understanding of Web services
- An understanding of the SAP NetWeaver Application Server (SAP NetWeaver AS)

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Work with Web services on SAP NetWeaver AS

Web Services Fundamentals

SAP NetWeaver AS is a development platform for Web services, which is an amenity that provides an application programming interface (API). In practice, the Web service is made available through an Internet protocol, and the application using the service is operated using a Web browser. Note that this is not the actual definition of a Web service.

Web services are the technical basis for making the functions of an application directly available. The existing function of an application can be addressed using standardized access protocols and content can be exchanged in a structured form. Cross-application functions can also be developed flexibly. The combination of several granular services, in the sense of self-contained business scenarios, is called an enterprise service.

The following basic standards for Web services are implemented in SAP NetWeaver AS:

- Extensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)

  SOAP describes a protocol that you can use to call Web services in distributed system landscapes. SOAP uses HTTP as a transport protocol. An SOAP message has a header with specific information and a body with the actual message.
- Web Services Description Language (WSDL)

  WSDL is a meta language used to describe the function of a Web service. Functions, parameters, and return codes in particular are described in a machine-readable form. WSDL is

© Copyright . All rights reserved.

Prohibida su reproduccion

standardized by the World Wide Web Consortium (W3C). For more information on the standardization of WSDL, see the following URL: http://www.w3.org/TR/wsdl.html.

- Universal Description, Discovery, and Integration (UDDI)

  UDDI is a directory service for dynamic Web services. A directory of Web services is provided through an SOAP interface. The information here is highlighted in white, yellow, and green pages. For more information on UDDI, see http://www.uddi.xml.org. SAP itself operates UDDI server. You will find information about UDDI at SAP here: http://scn.sap.com/docs/DOC-18439 . (Navigation panel: Service Oriented Architecture ES Workplace or Navigation panel: Service Oriented Architecture Discovery System). Here you fine information on which Web services can be registered and found. SAP NetWeaver AS comes with the UDDI-based Enterprise Services Repository and Services Registry, which can be used for company-wide Web service governance.

Web Services and SAP NetWeaver Application Server

Figure 54: Outline of a Web Service Scenario

With SAP NetWeaver, the development and configuration tools for Web services have been considerably enhanced for SAP NetWeaver AS ABAP as well as for SAP NetWeaver AS Java.

Using SAP NetWeaver, we can accomplish the scenario shown in the figure as follows:

- A Web service provided or implemented on SAP NetWeaver AS can be published to the services registry, where it can be found by a potential consumer.
- The use of Web services is supported by various applications of SAP NetWeaver. The implementation of Web services is fully supported in the ABAP and Java development tools.
- Web service consumers can call the Web service in the provider system.

  The contract for the parameters to be passed and the result to be returned is the WSDL of the Web service. WSDL is published by the provider and is the basis for both implementing the Web service itself and calling the service.

The following approaches are used to implement the Web services:

© Copyright . All rights reserved.

Prohibida su reproduccion

- Inside-out approach

  An existing functionality is exposed as a Web service. This functionality can be an existing RFC-capable function module on the ABAP side or an existing Enterprise Bean or Java class on the Java side. WSDL is derived from the interface provided by the methods of the bean/class responding to the function module.

- Outside-in approach

  The Web service is created using an existing WSDL as a basis. The Web service interfaces that WSDL represents can be designed for instances in the Enterprise Services Repository by following the guidelines of Web service governance and using predefined data types. From the standpoint of governance, this approach is the recommended way of implementing Web services.

For Web service consumers, WSDL is always needed to generate the proxy classes according to the Web service interface.

Tools for Configuration of Web Services

Figure 55: Logical Port (Consumer) – Service Endpoint (Provider)

Before you can call a Web service, both the provider and consumer sides must be configured. To make a service callable, the endpoints need to be configured. An endpoint determines the runtime configuration settings (for example, for security and reliable messaging) that are enforced on the consumer-provider communication. A service can have several endpoints.

The configuration of end points is also applicable on the consumer side. On the consumer side, you need to assign runtime configuration settings. A consumer can have several logical ports. A Web service call can be performed successfully only if logical ports and endpoints match with respect to their settings. For instance, it is not possible to call a Web service with user and password authentication if only Single Sign-On tickets are allowed by the endpoint configuration.

The configuration can be performed using the following tools:

- AS ABAP (transaction code `SOAMANAGER`)
- AS Java (*SAP NetWeaver Administrator* → *SOA*)

On both the consumer and provider sides, the providers and consumers are configured by using the same tool.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Work with Web services on SAP NetWeaver AS

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 4
Lesson 4

Explaining Service-Oriented Architecture

LESSON OVERVIEW
This lesson discusses an overview of the principles of service-oriented architecture (SOA) and explains the infrastructure of SOA in the context of SAP.

Business Example

To explain the principles and infrastructure of SOA, you want to implement a complex system landscape that serves one solution using SOA with SAP. For this reason, you require the following knowledge:

- An understanding of SOA in the context of SAP
- An understanding of the difference between Web and enterprise services

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe SOA in the context of SAP
- Differentiate between Web and enterprise services for SOA

Overview

SOA is a blueprint for an adaptable, flexible, and open IT architecture for developing services-based, modular business solutions. In the past, IT solutions and application systems were homomorphic. As long as only one application system serves as an IT solution for your business needs, there is no demand for SOA.

A demand for SOA comes with complex system landscapes serving one solution. SOA relies on the ability of coupling application systems. The aim of coupling is to support processes that interact across the systems of an application landscape that may be open.

Coupling must be installable by a mechanism that is independent from the technology of the application systems on which the landscape is built. Otherwise, future investments would become more dependent on the technologies on which the existing solutions are based.

Coupling mechanisms and standards for coupling arise with the evolution of the Internet. In addition, technology innovations and new standards, such as Web services, information process modeling, business reengineering, and business transformation are the key drivers for SOA.

The shift in architecture to enterprise SOA is comparable to past architecture shifts, such as the transition from mainframe architecture to client-server architecture.

© Copyright . All rights reserved.

Prohibida su reproduccion

A Brief History of Service-Oriented Architecture

	Mainframe (1970 – today)	Client/Server (1985 – today)	Component (1995 – today)	SOA (2005 - ?)
Business needs	Core Processes Collaboration Automation Agility	Automated core processes Collaboration Automation Collaboration	Automated departmental processes Collaboration Automatic Agility	Virtual company Collaboration Automatic Agility
Applications	Custom applications Core financials	Integrated standard software (ERP)	Composite applications (CRM, SCM, extended ERP)	Orchestrated services Modeling
Key middleware	OS	RDBMS, MOM	EAI, application server, first-generation Web services	ESB, BPM engines, Web services
Topology	Centralized 1-2-tier	Distributed 2-3-tier	Distributed 3-N-tier	Distributed N-tier
Networking	Dumb terminal SNA	Ethernet	Internet/http	Internet SOAP

Figure 56: A Brief History of Service-Oriented Architecture

SOA applications allow an organization to leverage its existing IT investments. Functionality can be exposed from existing applications in a standard way, and can be reused in SOA applications. SOA is an architecture that is not tied to a specific technology, and may be implemented using a wide range of technologies, such as the Distributed Component Object Model (DCOM), the Common Object Request Broker Architecture (CORBA), and Web services.

SOA is an architecture that uses loosely coupled services to support business process requirements. It is an environment where resources are made available as independent services that can be accessed without knowing the underlying platform implementation.

SOA is based on the following guiding principles:

- Reuse of granularity, modularity, componentization, and interoperability
- Compliance with standards (both common and industry-specific standards)

SOA provides a uniform way to handle functions because widely known and accepted Web standards are used.

The Web standards are as follows:

- Web Service Definition Language (WSDL)
- Simple Object Access Protocol (SOAP)
- Universal Description, Discovery, and Integration (UDDI)

© Copyright . All rights reserved.

Prohibida su reproduccion

Service-Oriented Architecture in the Context of SAP

Figure 57: Service-Oriented Architecture in the Context of SAP

SAP enables its customers to standardize and innovate on one platform as follows:

- SAP Business Suite provides ready-to-run business processes for 24 industries. New features and support for new business processes are continually added, for example, with enhancement packages that deliver innovation without disruption. SAP Business Suite is service enabled by shipping more than 2,800 enterprise services. Using SAP NetWeaver Process Integration (SAP NetWeaver PI), customers can leverage enterprise services.
- The SAP NetWeaver Composition Environment (SAP NetWeaver CE) is one type of use of SAP NetWeaver Application Server (AS) Java and helps create flexible company-specific processes that SAP does not yet address with standard software. This is where SAP NetWeaver CE plays an important role. SAP NetWeaver CE is a set of tools that gives companies the ability to compose new processes and extend core business processes. This means that these composite processes are built on top of SAP Business Suite and are extensions of SAP Business Suite. These processes are typically agile and tend to change often.

Types of Composite Business Processes

Composite business processes are as follows:

- Human centric

  Major activities in composite business processes are driven by human activities. This is where new offerings of SAP, such as SAP NetWeaver Business Process Management (SAP NetWeaver BPM) and SAP NetWeaver Business Rules Management (SAP NetWeaver BRM), fit in. Both offerings are an integral part of SAP NetWeaver CE.

- System centric

© Copyright . All rights reserved.

Prohibida su reproduccion

System-centric processes focus on integrating core application processes with third-party systems. Typical system-centric processes also occur in business-to-business (B2B) cases, for example, for integration with suppliers and customers. This is covered with SAP NetWeaver PI.

Web and Enterprise Services

SAP enables businesses to adopt SOA as part of their implementation of SAP solutions. At the heart of the open-standards approach of SAP to SOA is the concept of enterprise services. Enterprise services are highly integrated Web services, which are architected with business logic and harmonized semantics that can be accessed and used repeatedly to enable end-to-end business processes.

Also at the heart of the open-standards approach of SAP to SOA is the concept of enterprise services interfaces. These interfaces provide business processes, or business process steps, that can be used to compose business scenarios while ensuring business integrity and ease of reuse. SAP designs and implements enterprise services in such a way that safeguards semantic harmonization and business relevance. Using enterprise services, you can leverage SAP solutions with partner solutions and homegrown solution landscapes to build new, flexible, and innovative solutions, based on a consistent integration concept.

Enterprise services allow you to leverage SAP solutions to include a wide range of composite applications provided by customers and partners of SAP. SAP publishes all enterprise services that support SAP Business Suite functions at the Enterprise Services Workplace (ES Workplace) site in the SAP Community Network (SCN).

Difference Between Web and Enterprise Services

The following characteristics differentiate enterprise services from regular Web services:

- Business semantics

  Enterprise services are structured according to a harmonized enterprise model based on business objects, process components, and global data types (GDTs). They are defined using an outside-in approach (common business rules), rather than an SAP-specific implementation.

- Quality and stability

  Enterprise services safeguard a stable interface for future versions (backward compatibility). Their behavior, prerequisites, dependencies of use, and possibilities of configuration are well documented.

- Standards

  Enterprise services are based on open standards. The interfaces are described according to WSDL and are created using GDTs, which are based on UN/CEFACT Core Component Technical Specification (CCTS). B2B enterprise services are defined in compliance with e-business standards, wherever applicable.

Various applications exist for enterprise services, for example, application-to- application (A2A) and B2B integration or user interfaces.

Enterprise services are identified at a business level that supports business processes. With Enterprise Services Community (ES Community), SAP actively engages its customers and partners in the process of identifying and defining enterprise services. In this respect, SAP continually analyzes all business processes from the perspective of the main business drivers for

© Copyright . All rights reserved.

Prohibida su reproduccion

SOA, such as business process automation, business process innovation, user efficiency, and flexibility of deployment.

SAP defines and develops products for a comprehensive set of enterprise services to help customers, partners, and developers build flexible applications on SAP Business Suite. These services are published in ES Workplace.

For more information about SOA and ES Workplace, see

http://scn.sap.com/comunity/soa and

http://scn.sap.com/docs/DOC-25499.

LESSON SUMMARY

You should now be able to:

- Describe SOA in the context of SAP
- Differentiate between Web and enterprise services for SOA

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. _____ facilitates internal and external access to business processes and data in SAP systems.

   Choose the correct answer.

   - [ ] A The Business Object Repository (BOR)
   - [ ] B A Business Application Programming Interface (BAPI)
   - [ ] C Application Link Enabling (ALE)

2. Identify the interfaces and communication options that are supported by SAP systems.

   Choose the correct answers.

   - [ ] A HyperText Transfer Protocol (HTTP)
   - [ ] B Simple Mail Transfer Protocol (SMTP)
   - [ ] C Remote Function Call (RFC)
   - [ ] D Business Application Programming Interfaces (BAPIs)
   - [ ] E Extended Data Transfer Protocol
   - [ ] F SAP Transfer Protocol

3. For which of the following functions can Business Application Programming Interfaces (BAPIs) be used?

   Choose the correct answers.

   - [ ] A Request data from an SAP system
   - [ ] B Pass data to an SAP system
   - [ ] C Transfer SAP screen images to third-party applications (such as Microsoft Word)
   - [ ] D Access business processes in SAP systems

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Business _____ Programming Interfaces (BAPIs) are specialized _____ modules. They are accessed using the _____ interface. They are created and managed using the _____ Builder.

   Choose the correct answer.

   - [ ] A Application, function, Remote Function Call, Function
   - [ ] B Application, Remote Function Call, Function, function
   - [ ] C Application, Function, Remote Function Call, function
   - [ ] D Function, Remote Function Call, function, Application

5. What action can you perform with Application Link Enabling (ALE)?

   Choose the correct answers.

   - [ ] A Exchange data exclusively between SAP systems of the same release
   - [ ] B Exchange data across system boundaries, but only between SAP applications
   - [ ] C Exchange data between collaborating enterprises, using certain formats and technologies
   - [ ] D Communicate between the systems of your system landscape
   - [ ] E Update your order data using the appropriate Business Application Programming Interface (BAPI) (only once every 24 hours)

6. Composite processes are extensions of SAP Business Suite.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. _____ facilitates internal and external access to business processes and data in SAP systems.

   Choose the correct answer.

   - [ ] A The Business Object Repository (BOR)
   - [x] B A Business Application Programming Interface (BAPI)
   - [ ] C Application Link Enabling (ALE)

2. Identify the interfaces and communication options that are supported by SAP systems.

   Choose the correct answers.

   - [x] A HyperText Transfer Protocol (HTTP)
   - [x] B Simple Mail Transfer Protocol (SMTP)
   - [x] C Remote Function Call (RFC)
   - [x] D Business Application Programming Interfaces (BAPIs)
   - [ ] E Extended Data Transfer Protocol
   - [ ] F SAP Transfer Protocol

3. For which of the following functions can Business Application Programming Interfaces (BAPIs) be used?

   Choose the correct answers.

   - [x] A Request data from an SAP system
   - [x] B Pass data to an SAP system
   - [ ] C Transfer SAP screen images to third-party applications (such as Microsoft Word)
   - [x] D Access business processes in SAP systems

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Business _____ Programming Interfaces (BAPIs) are specialized _____ modules. They are accessed using the _____ interface. They are created and managed using the _____ Builder.

   Choose the correct answer.

   - [x] A Application, function, Remote Function Call, Function
   - [ ] B Application, Remote Function Call, Function, function
   - [ ] C Application, Function, Remote Function Call, function
   - [ ] D Function, Remote Function Call, function, Application

5. What action can you perform with Application Link Enabling (ALE)?

   Choose the correct answers.

   - [ ] A Exchange data exclusively between SAP systems of the same release
   - [ ] B Exchange data across system boundaries, but only between SAP applications
   - [x] C Exchange data between collaborating enterprises, using certain formats and technologies
   - [x] D Communicate between the systems of your system landscape
   - [ ] E Update your order data using the appropriate Business Application Programming Interface (BAPI) (only once every 24 hours)

6. Composite processes are extensions of SAP Business Suite.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 5 Flow of an ABAP Program

Lesson 1

Describing the Processing of ABAP Programs 158

UNIT OBJECTIVES

- Describe the SAP NetWeaver Application Server architecture
- Describe the execution of a simple dialog program by the ABAP runtime system

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 5
Lesson 1

Describing the Processing of ABAP Programs

LESSON OVERVIEW
This lesson explains how a simple dialog program is executed by SAP NetWeaver Application Server (AS).

Business Example

Your task is to explain the SAP NetWeaver AS architecture and the execution of ABAP programs. For this reason, you require the following knowledge:

- An understanding of the SAP NetWeaver AS architecture
- An understanding of how a simple dialog program is executed by the ABAP runtime system

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe the SAP NetWeaver Application Server architecture
- Describe the execution of a simple dialog program by the ABAP runtime system

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP System Architecture

Figure 58: Client/Server Architecture of SAP NetWeaver AS

SAP NetWeaver AS has a modular architecture that follows the software-oriented client/server principle.

In SAP NetWeaver AS, presentations, application logic, and data storage can be assigned to different systems. This serves as the basis for the scalability of the system.

Layers in the Architecture of SAP NetWeaver Application Server

The various layers in the Architecture of SAP NetWeaver AS are as follows:

- Database Layer

  Database layer is the lowest layer in the Architecture of SAP NetWeaver AS. At this layer, data is managed with the help of Relational Database Management System (RDBMS). In addition to application data, it includes programs and the metadata that the SAP system requires for operation.

- Application Server Layer

  Application Server layer is the intermediate layer in the Architecture of SAP NetWeaver AS. At this layer, ABAP programs, such as applications that SAP provides and any custom-developed applications, run on the application server. ABAP programs read data from the database, process it, and, if necessary, store new data in the database.

- Presentation Server Layer

© Copyright . All rights reserved.

Prohibida su reproduccion

Presentation Server layer is the third layer in the Architecture of SAP NetWeaver AS. This layer contains the user interface where each user can access the program, enter new data, and receive the results of a work process.

The technical distribution of software is independent of its physical location on the hardware. Vertically, all layers can be installed on top of each other on the same computer or each layer on a separate computer. Horizontally, you can divide the presentation and application server layer among any number of computers. The horizontal distribution of database components depends on the type of database installed.

Excerpt for an ABAP Program

Figure 59: Excerpt for an ABAP Program

> Note:
> You can simplify the figure for most of the topics that will be discussed during this course. This course focuses on the interaction between one user and one ABAP program.

Understanding how to write an ABAP program is more important than understanding the exact processes involved in the application server. Therefore, you will be working with a simplified figure that does not explicitly show the dispatcher and the work process. Certain slides will, however, be enhanced to include these details whenever they are relevant to ABAP programming.

ABAP programs are processed on the application server. The design of user dialogs and database accesses is of particular importance when writing application programs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Overview of Program Flow

Figure 60: View for the User

> Note:
> This section explains the basic process that starts when an ABAP program is executed. To do this, you use a program with which the user enters an airline ID (for example, LH) and finds details of the airline displayed as a result.

The SAP system is like a black box to an average user. Users do not need to know the precise process flow of an ABAP program. An average user is only interested in the business process and how to enter or display data. The technical aspects of the program are of minor interest to an average user.

In contrast, developers need to understand the interplay between both the server levels and the process flow, when programs are executed, to develop their own programs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Interplay Between Server Levels and Program Flow

Figure 61: Interplay Between Server Levels and Program Flow

After the user performs an action such as choosing *Enter*, a function key, a menu function, or a pushbutton, control passes from the presentation server to the application server.

If another user dialog is triggered from the ABAP program, the system transmits the screen, and control is once again passed back to the presentation server.

A program is not made up of a single block, but of several units. This is known as modularization. You can use most of these modularization units in more than one program. This is why they are often termed as reuse units. A good program should have database accesses encapsulated in such reuse units. The reuse unit creates a division between the design of the user dialog and database accesses, making it possible to use the same database accesses for different user dialogs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Program Flow in Detail - Selection Screen and List

Figure 62: Program Start

Whenever a user logs on to the system, a screen is displayed. From this screen, the user can start an ABAP program using the menu path.

Figure 63: System Loads Program Context

In this case, the system first loads the program context onto the application server. The program context contains memory areas for variables and complex data objects, information on the screens for user dialogs, and ABAP processing blocks. The runtime system gets all these

© Copyright . All rights reserved.

Prohibida su reproduccion

program information from the Repository, which is a special part of the database. The sample program has a selection screen as the user dialog, a variable and a structure as data objects, and one ABAP processing block. The list used to display the data is created dynamically at runtime. The ABAP runtime system controls the subsequent program flow.

Figure 64: Runtime System Sends Selection Screen

Since the program contains a selection screen, the ABAP runtime system sends it to the presentation server. The presentation server controls the program flow for as long as the user has not finished entering data in the input fields.

Selection screens allow users to enter selection criteria required by the program for it to continue.

© Copyright . All rights reserved.

Prohibida su reproduccion

Figure 65: Input Values Are Inserted into Data Objects

As soon as the user has finished entering data on the selection screen, he or she can trigger further processing of the program by choosing Execute.

The entered data is automatically placed in its corresponding data objects in the program and the ABAP runtime system resumes control.

In our simple program example there is only one ABAP processing block. The ABAP runtime system triggers sequential processing of this ABAP processing block.

If the entries made by the user do not have the correct type, an error message is automatically triggered. The user must correct his/her entries.

Figure 66: Reuse Unit is Loaded

© Copyright . All rights reserved.

Prohibida su reproduccion

A reusable unit is called in the processing block that encapsulates the database access. The reuse unit is a special processing block in an individual program. The actual example shows a method in a global class. When the reuse unit is called, the program in which it is contained is also read from the Repository and loaded to the application server.

Figure 67: Reuse Unit is Called

In the call, the required data is transferred to the called program, after which thereuse unit is executed. The execution is synchronous, which means that the calling program waits until the reuse unit has been processed completely

Figure 68: Program Requests Data Record from the Database

© Copyright . All rights reserved.

Prohibida su reproduccion

In the example program, read access to the database is programmed in the reuse unit. Correspondingly, information about the database table to be accessed and the row in the table to be read is passed on to the database.

Figure 69: Database Supplies the Data Record

The database returns the requested data record to the program and the runtime system ensures that this data is placed in the appropriate data objects.

If a single record is accessed, this data object is usually a structure that contains relevant components for all the required database fields.

Figure 70: Reuse Unit Returns the Data

© Copyright . All rights reserved.

Prohibida su reproduccion

This concludes the processing of the reuse unit and control is returned to the calling program, which resumes immediately after the call. When the system exits the reuse unit, the data that was read from the database is written to a corresponding data object for the calling program, from where it can be processed further.

Figure 71: Runtime System Sends the List

After the reuse unit has been called, the ABAP processing block receives statements for structuring the list with which the result is to be displayed. After the processing block finishes, the runtime system sends the list as a screen to the presentation server.

Course Structure

The table of contents for the course is as follows:

- Unit 1: Flow of an ABAP Program
- Unit 2: ABAP Workbench Introduction
- Unit 3: Basic ABAP Language Elements
- Unit 4: Modularization
- Unit 5: Complex Data Objects
- Unit 6: Data Modeling and Data Retrieval
- Unit 7: Classic ABAP Report
- Unit 8: Screen
- Unit 9: SAP List Viewer
- Unit 10: Web Dynpro ABAP

© Copyright . All rights reserved.

Prohibida su reproduccion

- Unit 11: Program AnalysisTool
- Unit 12: SAP Standard Software Adjustments

The program discussed already demonstrates numerous concepts that make up the content of this course. The table of contents, which shows all topics dealt with and their corresponding units, serves as an orientation aid for the remaining sections of this course.

LESSON SUMMARY

You should now be able to:

- Describe the SAP NetWeaver Application Server architecture
- Describe the execution of a simple dialog program by the ABAP runtime system

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Which of the following levels contains the user interface where each user can access a program, enter new data, and receive the results of a work process?

   Choose the correct answer.

   - ☐ A Application server level
   - ☐ B Presentation server level
   - ☐ C Application logic level
   - ☐ D SAP NetWeaver Application

2. When a program is not made up of a single block but of several units, the concept is referred to as _____.

   Choose the correct answer.

   - ☐ A Isolation
   - ☐ B Modularity
   - ☐ C Universality
   - ☐ D Multiplicity

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Which of the following levels contains the user interface where each user can access a program, enter new data, and receive the results of a work process?

   Choose the correct answer.

   - [ ] A Application server level
   - [x] B Presentation server level
   - [ ] C Application logic level
   - [ ] D SAP NetWeaver Application

2. When a program is not made up of a single block but of several units, the concept is referred to as _____.

   Choose the correct answer.

   - [ ] A Isolation
   - [x] B Modularity
   - [ ] C Universality
   - [ ] D Multiplicity

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 6 ABAP Workbench Introduction

Lesson 1

Introducing the ABAP Development Environment 174

Lesson 2

Organizing ABAP Developments 184
Exercise 10: Create a Package 191

Lesson 3

Developing Programs 194
Exercise 11: Develop a Simple ABAP Program 205

Lesson 4

Finalizing Development 209
Exercise 12: Create a Transaction 213

UNIT OBJECTIVES

- Describe the structure of the ABAP Repository
- Use the search tools of the ABAP Repository
- Use the Object Navigator for displaying Repository objects
- Describe the ABAP development infrastructure
- Create packages
- Create ABAP programs
- Use the ABAP Editor
- Activate ABAP programs
- Create transactions
- Release change requests

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 6
Lesson 1

Introducing the ABAP Development Environment

LESSON OVERVIEW
This lesson contains an overview of the ABAP Repository and the components of the ABAP Workbench. It presents the Object Navigator as a central development tool.

Business Example

You have to describe the structure of the Repository, use suitable tools to search for Repository objects, and analyze their structure.

For this reason, you require the following knowledge:

- An understanding of the structure of the Repository
- An understanding of how to name and use the search tools of the Repository
- An understanding of how to use the Object Navigator for displaying Repository objects

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe the structure of the ABAP Repository
- Use the search tools of the ABAP Repository
- Use the Object Navigator for displaying Repository objects

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Repository

Figure 72: The Cross-Client Capability of the Repository

The Repository consists of all system development objects, such as programs, function modules, definitions of database tables, and so on. The Repository contains objects delivered by SAP as well as those defined by the customer. The Repository is in the database and is independent of the client. This means that a Repository object can be accessed from any client and looks the same in each client in the system.

The database also contains application and Customizing data, which is normally client-dependent. This means that every data record is assigned to a particular client and can only be read and changed by users who have logged on to that particular client.

© Copyright . All rights reserved.

Prohibida su reproduccion

Structure of the Repository

Figure 73: Structure of Repository

The system subdivides the Repository according to application components.

An application component, for example, Materials Management (MM), consists of several packages containing relevant objects for a more detailed logical subdivision.

Whenever a Repository object is created, it must be assigned to a package.

Some Repository objects can have subobjects that are themselves Repository objects. Furthermore, Repository objects can reference other Repository objects.

© Copyright . All rights reserved.

Prohibida su reproduccion

Search Tools in the ABAP Repository

Figure 74: Repository Information System

The Repository Information System is suitable for random (that is, not application-specific) searches of Repository objects, for example, all programs by a certain developer or all function modules that were changed after a certain date.

To access the Repository Information System, perform the following steps:

1. In the *SAP Easy Access* menu, choose *Tools → ABAP Workbench → Overview → Information System*.
2. Double-click an object type. A selection screen appears allowing you to limit your search.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Application Hierarchy

Figure 75: SAP Application Hierarchy

Use the application hierarchy for the application-specific search for Repository objects. The hierarchy of the application components is depicted in the form of a tree structure.

To access the SAP application hierarchy, perform the following steps:

1. In the *SAP Easy Access* menu, choose *Tools → ABAP Workbench → Overview → Application Hierarchy → SAP*.
2. Expand the component node to display all the packages assigned to the corresponding component.
3. Choose the *Select* pushbutton to select any subtree and navigate to the Repository Information System using the *Information System* pushbutton.

The system remembers all relevant packages of the selected subtrees and enters them in the selection screen of the information system. This allows you to carry out a search in the previously selected applications.

© Copyright . All rights reserved.

Prohibida su reproduccion

Object Navigator

Figure 76: ABAP Workbench Tools

The ABAP Workbench includes all tools required for creating and editing Repository objects. These tools cover the entire software development cycle.

Some of the ABAP Workbench tools are as follows:

- The ABAP Editor for editing the source code
- The ABAP Dictionary for editing database table definitions, data types, and so on
- The Screen Painter for configuring screens together with functions for user dialogs
- The Menu Painter for designing user interfaces: menu bar, standard toolbar, application toolbar, and function key settings
- The Function Builder for maintaining function modules
- The Class Builder for maintaining global classes and interfaces

You can call each of these tools explicitly and then load a Repository object for processing, but it is more convenient to access them using the Object Navigator. You can have the requested Repository objects listed in this central development tool. To edit one of the Repository objects, you double-click it. The corresponding tool is called automatically and includes the selected Repository for displaying or editing.

© Copyright . All rights reserved.

Prohibida su reproduccion

Screen Layout in the Object Navigator

Figure 77: Screen Layout in the Object Navigator

The Object Navigator screen is split into the following areas:

- The navigation area for displaying a hierarchical object list
- The tool area for displaying and editing a development object using the appropriate tool

You can display or hide the navigation area (*Fullscreen on/off*).

In both areas, you can choose the functions using a context menu, which you access using the right mouse button. The context menu only offers functions that have been designed for the respective object.

© Copyright . All rights reserved.

Prohibida su reproduccion

Work with the Navigation Area

Figure 78: Working with the Navigation Area

The system displays object lists in the navigation area. For example, if you choose to display a package, all Repository objects belonging to the specified package are listed.

Double-clicking an object allows you to display or edit it.

You can navigate between object lists that have been previously displayed in the current Object Navigator session (blue arrows).

You can add frequently used object lists to your favorites.

© Copyright . All rights reserved.

Prohibida su reproduccion

Functions in the Tool Area

Figure 79: Functions in the Tool Area

In the tool area, a Repository object is displayed in the corresponding tool.

You can navigate between objects that have been previously displayed in the current Object Navigator session (blue arrows).

You can also display a subwindow with the previous navigation history. When you double-click an object in the navigation history, the object is displayed in the tool area.

© Copyright . All rights reserved.

Prohibida su reproduccion

Synchronization of the Navigation and Tool Areas

Figure 80: Synchronizing the Navigation and Tool Areas

Navigation in the navigation area is independent of navigation in the tool area. This allows both areas to be used in a flexible manner.

If necessary, you can synchronize both areas as follows:

- You can display an object in the tool area by double-clicking in the navigation area or by using the corresponding context menu function of the object.
- You can display the object list of an object that you are currently editing in the tool area by choosing the *Display Object List* pushbutton in the navigation area.

To create a new object, use the context menu for an object type in the corresponding object list. Alternatively, use the *Edit Object* or *Other Object* pushbutton to create an object.

LESSON SUMMARY

You should now be able to:

- Describe the structure of the ABAP Repository
- Use the search tools of the ABAP Repository
- Use the Object Navigator for displaying Repository objects

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 6
Lesson 2

Organizing ABAP Developments

LESSON OVERVIEW

This lesson explains how to create new Repository objects, packages, programs, and transaction codes. This lesson also explains how change requests in the ABAP Workbench are used in the SAP environment to organize development projects and ensure consistent transport of changes to the productive systems.

Business Example

As part of a development project, you have to create a new package and an ABAP program and make this program available to users by way of a transaction code. For this reason, you require the following knowledge:

- An understanding of how to name and use utilities for orderly software development
- An understanding of how to create packages, programs, and transactions

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Describe the ABAP development infrastructure
- Create packages

© Copyright . All rights reserved.

Prohibida su reproduccion

Transport of Development Objects

Figure 81: Transporting Development Objects

Development projects are carried out in a development system. The development objects edited or created in a project are transported to subsequent systems (test and/or production system) on project completion. At the start of a development project, the project manager creates a change request in which the project manager names the employees of this project in the Transport Organizer or directly in the ABAP Workbench. The Transport Organizer then creates a task for each project employee in the change request.

When a development object is edited or created, the relevant employee assigns this to the change request. The object is entered into the task of the employee. All Repository objects that an employee works on during a development project are collected within the employee's task.

© Copyright . All rights reserved.

Prohibida su reproduccion

Organization of Development Projects

Figure 82: Organization of a Development Project in a Change Request

Organizing a development project using a change request offers the following advantages:

- Employees can monitor their project-specific activities.
- Employees involved in the project can process all repository objects assigned to the project. For developers who do not belong to the project team, Repository objects remain locked until the project is completed (the change request is released).
- Assigning the objects to the change request ensures that all development objects created or changed during the project are transported automatically at the time of project completion (i.e. when the change request is released). The transport route of the packages involved (in which development took place) specifies to which subsequent system they are transported.

Unlike packages, which distinguish between Repository objects in a logical and functional way, change requests are project-related and, therefore, delimit the objects over a period of time. Although a program always belongs to only one package, it can, at different times, belong to different projects.

© Copyright . All rights reserved.

Prohibida su reproduccion

Creation of Packages

Figure 83: Setting Package Attributes (Example)

The attributes of a package have the following meaning (detailed information can be obtained by using the field help, F1):

- Application Component

  Determine the location of the package within the application hierarchy by specifying the corresponding application component.

- Software Component

  For customer developments, enter **HOME** for the software component.

- Transport Layer

  The transport layer determines if the objects of this package are to be transported to a subsequent system and, if so, to which system. For customer developments you have to specify the transport layer which your system administrator set up for this purpose.

- Package Type

  Choose between the three package types: standard package (can contain Repository objects and other packages), main package (can only contain other packages), and structure package (can only contain main packages).

© Copyright . All rights reserved.

Prohibida su reproduccion

Specifying a Transport Route

Figure 84: Specifying a Transport Route

Even a simple system landscape you need at least two different transport layers to distinguish between customer developments and modifications to SAP objects. In a more complex system landscape further transport layers may exist with different target systems.

© Copyright . All rights reserved.

Prohibida su reproduccion

Assignment to a Change Request

Figure 85: Assignment to a Change Request

You can display all change requests in which you have a task using the *My Tasks* pushbutton. To select the relevant request, double-click the transport request.

> Note:
> The system takes care of the assignment for your task.

> Hint:
> You must assign all Repository objects created or changed to the change request of the respective project.

To Create a Package

1. Navigate to the *Object Navigator*. You have the following options:
   a) In the navigation area, choose the *Package* object type and enter the name of the package in the input field below, ensuring that you comply with the customer namespace conventions. Press ENTER. If the specified package does not already exist, the system will branch off to a dialog to create a package.

© Copyright . All rights reserved.

Prohibida su reproduccion

   b) Choose the *Edit Object* pushbutton on the initial screen of the *Object Navigator*. In the dialog box, search for the option of specifying a package and enter the name of the package. Press F5 to create the object.

2. Set the attributes of the package to be created.
3. Assign the package to a change request.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Package

Business Example

You are a developer for an airline group and have been given the task of developing evaluation programs for several airline companies.

To group Repository objects logically and transport them, you have to create a package.

Create a package.

Task 1

Log on to the training system.

1. Log on to the training system. Use the client, user name, and initial password given to you by your instructor.
2. Enter a new personal password.

Task 2

Create a package named **`ZBC400_##`**, where ## represents the group number given to you by your instructor.

1. Open the *Object Navigator* and create the package there.
2. In the dialog box that appears, enter a short description of your package. Assign the package to the application component **`CA`** and the software component **`HOME`**. Take the default value for all the other input fields in the dialog box.
3. In the following dialog box, assign the package to a change request. Use the change request in which the instructor has already created a task for your user.

> Hint:
> To determine this request, choose the *Own Requests* pushbutton.

You have successfully created your own package. Assign all Repository objects that you create in the following exercises to this package. Use the same change request in each case.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Package

Business Example

You are a developer for an airline group and have been given the task of developing evaluation programs for several airline companies.

To group Repository objects logically and transport them, you have to create a package.

Create a package.

Task 1

Log on to the training system.

1. Log on to the training system. Use the client, user name, and initial password given to you by your instructor.
2. Enter a new personal password.

Task 2

Create a package named **ZBC400_##**, where ## represents the group number given to you by your instructor.

1. Open the *Object Navigator* and create the package there.
   a) Open the *Object Navigator* using transaction **SE80**.
   b) In the drop-down menu, select *Package* and enter **ZBC400_##** in the next field and return.
2. In the dialog box that appears, enter a short description of your package. Assign the package to the application component **CA** and the software component **HOME**. Take the default value for all the other input fields in the dialog box.
   a) Enter a short description for your package and choose *Continue*.
3. In the following dialog box, assign the package to a change request. Use the change request in which the instructor has already created a task for your user.

> Hint:
> To determine this request, choose the *Own Requests* pushbutton.

You have successfully created your own package. Assign all Repository objects that you create in the following exercises to this package. Use the same change request in each case.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Describe the ABAP development infrastructure
- Create packages

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 6
Lesson 3

Developing Programs

LESSON OVERVIEW
This lesson explains how to create ABAP programs. It also introduces you to the ABAP programming language and syntax.

Business Example

You have to create a new package and a new ABAP program as a part of a development project. You have to make this program available to users with a transaction code. For this reason, you require the following knowledge:

- An understanding of how to create ABAP programs
- An understanding of the ABAP Editor
- An understanding of how to activate ABAP programs

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Create ABAP programs
- Use the ABAP Editor
- Activate ABAP programs

© Copyright . All rights reserved.

Prohibida su reproduccion

Creation of ABAP Programs

Figure 86: Creating an ABAP Program

The figure shows the creation of an ABAP program without a top include and the attributes of the created program.

You will learn how to create programs and transactions in the ABAP Workbench.

To Create an ABAP Program

1. Navigate to the Object Navigator using any of the following options:
   a) In the navigation area, choose the *Program* object type and enter the name of the program in the input field, ensuring that you comply with the customer namespace conventions. Press ENTER. If the program does not exist, the system goes to the dialog that allows you to create a program.
   b) Display the package in which you want to create the program. You can branch to the dialog for creating a program using the context menu for the package or the Programs node.
   c) Choose the *Edit Object* pushbutton on the initial screen of the *Object Navigator*. In the dialog box, choose the *Program* tab page and enter the name of the program. To create the object, press F5.
2. Remove the flag for *With TOP Include*, to prevent your source code from being distributed to several programs.
3. Change the title to a self-explanatory short text and, for the purposes of this course, choose *Executable Program* as the program type. All other program attributes are optional. To access help or get more details, press F1.

© Copyright . All rights reserved.

Prohibida su reproduccion

Basic ABAP Principles

The features of the ABAP programming language are as follows:

- It is typed.
- It enables multi-language application.
- It enables SQL access.
- It is enhanced as an object-oriented language.
- It is platform-independent.
- It is upward-compatible.

The ABAP programming language is designed for dialog-based business applications.

To support the type-specific processing of data, type conversions and type casting are supported.

Using translatable text elements, you can develop multi-language applications.

The Open SQL standard embedded in ABAP allows direct database access.

ABAP Objects is the object-oriented enhancement of the ABAP programming language. ABAP syntax is platform-independent. This means that ABAP syntax always has the same meaning or function, irrespective of the relational database system and operating system for the application and presentation server. Applications implemented in ABAP will also be able to run in future releases because of the upward compatibility of the language.

© Copyright . All rights reserved.

Prohibida su reproduccion

General ABAP Syntax I

Figure 87: General ABAP Syntax I

Characteristics of ABAP syntax:

- ABAP programs consist of individual statements.
- The first word in a statement is called an ABAP keyword.
- Each statement ends with a period.
- A space must always separate two words.
- Statements can be indented.
- The ABAP runtime system does not differentiate between upper and lowercase in keywords, additions, and operands.

> Hint:
> Although the ABAP runtime system does not differentiate between upper and lowercase, it has become customary to write keywords and their additions in uppercase letters and operands in lowercase. This course will use this form of representation.

For indentations, and for converting uppercase or lowercase letters, use Pretty Printer in the ABAP Editor. Use the following menu path in the Object Navigator to make a user-specific setting for the Pretty Printer: *Utilities → Settings → ABAP Editor → Pretty Printer*.

© Copyright . All rights reserved.

Prohibida su reproduccion

General ABAP Syntax II

Figure 88: General ABAP Syntax II

Additional features of ABAP syntax:

- Statements can extend beyond one line.
- Several statements can lie in a single line (although it is not recommended).
- Lines that begin with an asterisk * in the first column are recognized as comment lines by the ABAP runtime system and are ignored.
- A double quotation mark (") indicates that the rest of a line is a comment.

You can combine consecutive statements with an identical beginning into a chained statement as follows:

- Write the identical beginning part of the statements, followed by a colon.
- List the end parts of the statements, separated by commas.
- You may use blank spaces and line breaks before and after separators, that is, colons, commas, and periods.

Hint:
This short form represents a simplified form of syntax and does not offer an improvement in performance. The ABAP runtime system processes each of the individual statements.

© Copyright . All rights reserved.

Prohibida su reproduccion

Keyword Documentation in the ABAP Editor

Figure 89: Keyword Documentation in the ABAP Editor

You can navigate to the documentation for an ABAP statement in the following ways:

- Use F1 to navigate directly to the documentation for the statement where the cursor is positioned.
- Choose the "*i*" pushbutton with the description *Help on ...* to navigate to a dialog box, where you can enter the required ABAP keyword.

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Editor

Figure 90: Previous ABAP Editor

You can use several editors to develop ABAP programs, depending on the release and support package level. The choice of the editor is user-specific and can be changed (in the Object Navigator, choose *Utilities → Settings → ABAP Editor*).

The latest ABAP Editor (new front-end editor) offers plenty of number of new options and easy-to-use additional functions compared to the classic ABAP Editor (old front-end editor). It was developed for SAP NetWeaver 7.0. It can also be used in previous releases after a certain support package level: SAP NetWeaver AS 6.40 (SP18) and SAP Web Application Server 6.20 (SP59).

© Copyright . All rights reserved.

Prohibida su reproduccion

New ABAP Editor

Figure 91: New ABAP Editor

Some of the important options provided by the new ABAP Editor are as follows:

- You can choose different display colors for different objects in the source code.
- You can set fonts and font sizes for each individual user.
- You can compress different blocks of source code, such as loops and conditional branches, to provide a better overview.
- You can use bookmarks to find relevant points in the source code faster.
- You can display line numbers and the current nesting to improve orientation.
- You can have complete words suggested by the ABAP Editor when you type the first few characters of ABAP keywords and data objects.
- You can have a small popup list generated by the new ABAP Editor with suitable suggestions for the current cursor position (since NetWeaver 7.0 EhP2) when you press CTRL + SPACE.

© Copyright . All rights reserved.

Prohibida su reproduccion

Activation of Programs

Figure 92: Inactive and Active Development Objects

Whenever you create or change a development object and then save it, the system stores only one inactive version in the Repository.

You have an active version and an inactive version of the object. When you complete object development, you have to activate the inactive version (editing version) of the object. This version becomes the new active version of the object.

> Note:
> The request release and the transport of the developed objects are only possible if all objects in the request have been activated.

If your program is available in both active and inactive versions, you can switch between the displays of these two versions by using the corresponding button in the ABAP Editor.

Whenever you activate a program, the system displays a list of all inactive objects that you have processed. This is your work list. Choose those objects that you wish to activate with your current activation transaction.

The activation of an object includes the following functions:

- Saving the object as an inactive version
- Checking the syntax or consistency of the inactive version
- Overwriting the previously active version with the inactive version (only after a successful check)

© Copyright . All rights reserved.

Prohibida su reproduccion

- Generating the relevant runtime object for later execution, if the object is a program

Generation of Runtime Objects

Figure 93: Generating Runtime Objects

When you generate a development object, the system creates a separate runtime object (LOAD compilation) and stores it in the Repository. This generated version is the version that is executed or interpreted at runtime.

If your program has an inactive version as well as an active version, you can get both versions to run in the following ways:

- If you start your program using the context menu of the navigation area, or by executing a transaction, the active version is used. This means that the LOAD generated for the last activation is executed.
- If you start the inactive version loaded to the ABAP Editor using the F8 key, a temporary runtime object is generated from it and executed.

In this way, it is possible to continue the development of a Repository object without changing the current system status. All changes to the development object become visible throughout the system only after the activation of the object.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Develop a Simple ABAP Program

Business Example

You want to create an ABAP program that takes simple user input, modifies the data, and outputs the information to a screen.

In your package, create an executable program named **`ZBC400_##_HELLO`**. Enable the user to enter a name on the selection screen, which will be output later in the list, together with the 'Hello' text.

1. Create program **`ZBC400_##_HELLO`** without a *TOP include*.
2. In the next dialog box, make sure that the program type is *Executable Program*. Set the status of the program to a meaningful value.
3. Define input field **`pa_name`** on the selection screen with type **`string`**.
4. Implement the output of the **`Hello World!`** text. To do this, use the ABAP **`WRITE`** statement.
5. Make sure that the following output appears on a new line. To do so, use the **`NEW-LINE`** statement.
6. Using a chained statement, implement the output text **`'Hello'`** together with the input name from the selection screen.
7. Check your program for syntax errors. Test and activate it.

© Copyright . All rights reserved.

Prohibida su reproduccion

Develop a Simple ABAP Program

Business Example

You want to create an ABAP program that takes simple user input, modifies the data, and outputs the information to a screen.

In your package, create an executable program named **ZBC400_##_HELLO**. Enable the user to enter a name on the selection screen, which will be output later in the list, together with the 'Hello' text.

1. Create program **ZBC400_##_HELLO** without a *TOP include*.
   a) Open the Object Navigator using transaction **SE80**.
   b) In the drop-down menu, select *Package* and enter **ZBC400_##** in the next field.
   c) Under *Object Name*, right-click *ZBC400_##* and choose *Create* → *Program*.
   d) Enter **ZBC400_##_HELLO** in the *Program* field. Uncheck *With TOP INCL*.
   e) Choose *Continue*.
2. In the next dialog box, make sure that the program type is *Executable Program*. Set the status of the program to a meaningful value.
   a) Choose *Executable Program* from the *Type* drop-down list.
   b) Choose *Test Program* from the *Status* drop-down list and choose *Save*.
   c) In the next dialog box, enter the name of your package and choose the *Save (Enter)* pushbutton.
3. Define input field **pa_name** on the selection screen with type **string**.
   a) Enter the following code in the ABAP Editor:

      ```
      PARAMETERS: pa_name TYPE string.
      ```
4. Implement the output of the **Hello World!** text. To do this, use the ABAP **WRITE** statement.
   a) Enter the following code in the ABAP Editor:

      ```
      WRITE 'Hello World!'.
      ```
5. Make sure that the following output appears on a new line. To do so, use the **NEW-LINE** statement.
   a) Enter the following code in the ABAP Editor:

      ```
      NEW-LINE.
      ```


© Copyright . All rights reserved.

Prohibida su reproduccion

6. Using a chained statement, implement the output text **'Hello'** together with the input name from the selection screen.
   a) Enter the following code in the ABAP Editor:

      WRITE: 'Hello',

      pa_name.

7. Check your program for syntax errors. Test and activate it.
   a) Choose the *Check* (*Ctrl+F2*), *Direct* (*F8*), and *Activate* (*Ctrl+F3*) pushbuttons.

      Complete source code

```
*&- - - - - - - - -- - - - - - - - - - - - - - - - - - - - -*
*& Report  BC400_GSS_HELLO
*&- - - - - - - - -- - - - - - - - - - - - - - - - - - - - -*

REPORT  bc400_gss_hello.

PARAMETERS: pa_name TYPE string.

WRITE 'Hello World!'.

NEW-LINE.

WRITE: 'Hello',
       pa_name.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Create ABAP programs
- Use the ABAP Editor
- Activate ABAP programs

© Copyright . All rights reserved.

Prohibida su reproduccion

Finalizing Development

LESSON OVERVIEW

This lesson explains how to create transactions and release change requests.

Business Example

You need to create transactions and release change requests. For this reason, you require the following knowledge:

- An understanding of how to create transactions
- An understanding of how to release change requests

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Create transactions
- Release change requests

Creation of Transactions

Figure 94: Creating a Transaction

© Copyright . All rights reserved.

Prohibida su reproduccion

You can only include transactions in a role menu and in a user's favorites. If you want to place a program in either a role menu or a user's favorites, you must create a transaction that represents the program and integrate the transaction into the menu. Alternatively, you can start the program by entering the transaction code in the command field.

To Create a Transaction

1. In the *Object Navigator*, display the object list for the program.
2. In the navigation area, select your program and choose *Create* → *Transaction* from the context menu.
3. Enter the required transaction code. Make sure you comply with the customer namespace conventions.

   Assign a short text and select the *Program and Selection Screen* (*Report Transaction*) radio button.
4. On the next screen, enter the name of the program and choose *Professional User Transaction*.

   Under *GUI enabled*, set the *SAP GUI for Windows* indicator.
5. Save the transaction.
6. Assign the transaction to a package and to a change request on the following screens.

> Note:
> Each transaction is a Repository object.

Inserting a Transaction into Personal Favorites

Figure 95: Inserting a Transaction into Personal Favorites

© Copyright . All rights reserved.

Prohibida su reproduccion

To Add a Transaction to Personal Favorites

1. Navigate to the initial screen, the *SAP Easy Access* screen.
2. In the *Favorites* context menu, choose *Insert Transaction*.
3. In the dialog box, enter the required transaction code.

   The short text of the transaction now appears under the *Favorites* context menu. You can start the corresponding program by double-clicking it.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Transaction

Business Example

Users in your company must be able to start their ABAP programs as transactions or define them as personal favorites.

Create a report transaction. Include the new transaction in your personal favorites in the *SAP Easy Access* menu.

Task 1

Create a report transaction (suggested name: **`ZBC400_##_HELLO`**) that you can use to start your executable program ZBC400_##_HELLO.

1. Create the new transaction in your package. Enter a short description and select the appropriate transaction type (start object).
2. Refer to the program that is to be executed. Choose the correct transaction classification and make sure that it is GUI-enabled.
3. Save and test the transaction.
4. Confirm that the program can now be started by entering the transaction code in the command field.

Task 2

Include the new transaction in your personal favorites in the *SAP Easy Access* menu.

1. Open the *SAP Easy Access* menu, choose the function *Insert Transaction* under *Favorites*, and enter the transaction code.
2. Double-click the new favorite entry and execute the program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Transaction

Business Example

Users in your company must be able to start their ABAP programs as transactions or define them as personal favorites.

Create a report transaction. Include the new transaction in your personal favorites in the *SAP Easy Access* menu.

Task 1

Create a report transaction (suggested name: **`ZBC400_##_HELLO`**) that you can use to start your executable program ZBC400_##_HELLO.

1. Create the new transaction in your package. Enter a short description and select the appropriate transaction type (start object).
   a) Make sure that you select the *Program and Selection Screen (Report Transaction)* radio button as the start object.
2. Refer to the program that is to be executed. Choose the correct transaction classification and make sure that it is GUI-enabled.
   a) Perform this step as described in the Participant Handbook. Make sure that the transaction is classified as a *Professional User Transaction* and set it as GUI-enabled for *SAP GUI for Windows*.
3. Save and test the transaction.
   a) Choose *Save* and *Test* for this purpose
4. Confirm that the program can now be started by entering the transaction code in the command field.

Task 2

Include the new transaction in your personal favorites in the *SAP Easy Access* menu.

1. Open the *SAP Easy Access* menu, choose the function *Insert Transaction* under *Favorites*, and enter the transaction code.
   a) Carry out this step as described in the Participant Handbook.
2. Double-click the new favorite entry and execute the program.
   a) Carry out this step in the usual way.

© Copyright . All rights reserved.

Prohibida su reproduccion

Release of Change Requests

Figure 96: The Developer Releases the Task

After a project employee completes the required development task, the employee carries out a quality check and releases the task within the change request.

The system transfers the corresponding object entries and the change locks of the object for third parties that were automatically set at the start of the project, from the task to the request.

All project employees can still edit these objects. After all the tasks in a change request have been released, the project manager carries out the final check for the objects and releases the change request. This concludes the project.

© Copyright . All rights reserved.

Prohibida su reproduccion

Closing of the Project by Releasing the Request (Project Manager)

Figure 97: Closing the Project by Releasing the Request (Project Manager)

When the request is released, all the object locks that belong to the request are removed.

Copies of the developed objects are exported to the system's own transport directory where they stay until the system administrator imports them to their intended target system.

Another copy of the exported development objects is stored in the system's own version database.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Create transactions
- Release change requests

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Which of the following system development objects are included in the Repository?
   Choose the correct answers.

   - [ ] A Programs
   - [ ] B Function modules
   - [ ] C Object Navigator
   - [ ] D Definitions of database tables

2. Which of the following ABAP Workbench tools is used for editing the source code?
   Choose the correct answer.

   - [ ] A ABAP Editor
   - [ ] B ABAP Dictionary
   - [ ] C Repository Information System
   - [ ] D Menu Painter

3. Which of the following are package types?
   Choose the correct answers.

   - [ ] A Application component
   - [ ] B Standard package
   - [ ] C Main package
   - [ ] D Software component

© Copyright . All rights reserved.

Prohibida su reproduccion

4. When is the transport of development objects for a development request triggered?

   Choose the correct answer.

   - [ ] A When an object is saved
   - [ ] B When an object is activated
   - [ ] C When a task is released
   - [ ] D When a request is released

5. Whenever you create or change a development object and save it, the system stores two inactive versions in the Repository.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

6. Which of the following statements is correct about the ABAP programming language?

   Choose the correct answers.

   - [ ] A It enables multi-language applications.
   - [ ] B It enables SQL access.
   - [ ] C It is platform-dependent.
   - [ ] D It is typed.

7. ABAP Objects is the object-oriented enhancement of the ABAP programming language.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

8. How do you include comment lines?

   Choose the correct answer.

   - [ ] A Begin the line with a pound # in the first column.
   - [ ] B Begin the line with an asterisk * in the first column.
   - [ ] C End the line with an asterisk * in the last column.
   - [ ] D End the line with a pound # in the last column.

9. After a project employee completes the required development task, he or she should do the following:

   Choose the correct answer.

   - [ ] A Transfer the task to the Production System
   - [ ] B Release the task within the Change request
   - [ ] C Unlock and export the task to the Transport Directory
   - [ ] D Export the Project Request

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Which of the following system development objects are included in the Repository?

   Choose the correct answers.

   - [x] A Programs
   - [x] B Function modules
   - [ ] C Object Navigator
   - [x] D Definitions of database tables

2. Which of the following ABAP Workbench tools is used for editing the source code?

   Choose the correct answer.

   - [x] A ABAP Editor
   - [ ] B ABAP Dictionary
   - [ ] C Repository Information System
   - [ ] D Menu Painter

3. Which of the following are package types?

   Choose the correct answers.

   - [ ] A Application component
   - [x] B Standard package
   - [x] C Main package
   - [ ] D Software component

© Copyright . All rights reserved.

Prohibida su reproduccion

4. When is the transport of development objects for a development request triggered?

   Choose the correct answer.

   - [ ] A When an object is saved
   - [ ] B When an object is activated
   - [ ] C When a task is released
   - [x] D When a request is released

5. Whenever you create or change a development object and save it, the system stores two inactive versions in the Repository.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

6. Which of the following statements is correct about the ABAP programming language?

   Choose the correct answers.

   - [x] A It enables multi-language applications.
   - [x] B It enables SQL access.
   - [ ] C It is platform-dependent.
   - [x] D It is typed.

7. ABAP Objects is the object-oriented enhancement of the ABAP programming language.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

8. How do you include comment lines?

   Choose the correct answer.

   - [ ] A Begin the line with a pound # in the first column.
   - [x] B Begin the line with an asterisk * in the first column.
   - [ ] C End the line with an asterisk * in the last column.
   - [ ] D End the line with a pound # in the last column.

9. After a project employee completes the required development task, he or she should do the following:

   Choose the correct answer.

   - [ ] A Transfer the task to the Production System
   - [x] B Release the task within the Change request
   - [ ] C Unlock and export the task to the Transport Directory
   - [ ] D Export the Project Request

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 7 Basic ABAP Language Elements

Lesson 1

Defining Elementary Data Objects 226

Lesson 2

Using Basic ABAP Statements 238

Exercise 13: Use Basic ABAP Statements 247

Lesson 3

Working with the ABAP Debugger 254

Exercise 14: Debug Statements on Elementary Data Objects 261

UNIT OBJECTIVES

- Know the differences between types and variables and how to use them
- Use the ABAP syntax of standard basic programming concepts
- Use the ABAP Debugger to analyse values of elementary data objects during program execution.

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 7
Lesson 1

Defining Elementary Data Objects

LESSON OVERVIEW
This lesson explains the difference between data types and data objects, and how to define and use them in a program. You will also learn some basic ABAP statements. After this Lesson, you will know how to use the ABAP Debugger for analyzing a program flow.

Business Example

You want to use simple variables in your programs and edit them with simple statements. Furthermore, you want to use the ABAP Debugger to find the semantic errors in your programs. For this reason, you need to know:

- The definition of elementary data objects
- How to use basic ABAP statements with elementary data objects
- How to execute and analyze programs in debugging mode

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Know the differences between types and variables and how to use them

Data Types and Data Objects

Figure 98: Using Data Types

© Copyright . All rights reserved.

Prohibida su reproduccion

A formal variable description is called a data type. A variable or constant that is defined concretely by data type is called a data object.

The figure shows how data types can be used.

You can use data types in the following ways:

- To define data objects

  The type of a data object defines its technical properties.

- To define interface parameters

  The type of an interface parameter determines the type of the actual parameter that is transferred when the modularization unit is called.

- To define input/output fields

  The type of an input/output field can provide additional information to the technical characteristics, such as the field and value input help.

This Lesson will focus on how to use data types for defining internal program variables.

ABAP Standard Data Types

ABAP standard data types (built-in data types) are divided into two groups:

- Complete
- Incomplete

Complete ABAP Standard Data Types

The built-in data ABAP standard data types that already contain a type-specific, fixed-length specification are considered a complete data type.

Table 1: Complete ABAP Standard Data Types

Standard Types	Description
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)
DECFLOAT16	Type for saving (DECimal FLOATing point) numbers with mantissa and exponent, length 8 bytes with 16 decimal places (fixed) (since 7.0 EhP2)

© Copyright . All rights reserved.

Prohibida su reproduccion

Standard Types	Description
DECFLOAT34	Type for saving (**DEC**imal **FLOAT**ing point) numbers with mantissa and exponent, length 16 bytes with 34 decimal places (fixed) (since 7.0 EhP2)

Incomplete ABAP Standard Data Types

The standard types that do not contain a fixed length are considered an incomplete data type. When they are used to define data objects, you need to specify the length of the variable.

Table 2: Incomplete ABAP Standard Data Types

Standard Types	Description
C	Type for character string (**C**haracter) for which the length is to be specified
N	Type for numerical character string (**N**umerical character) for which the length is to be specified
X	Type for byte sequence (He**X**adecimal string) for which the length is to be specified
P	Type for packed number (**P**acked number) for which the length is to be specified (In the definition of a packed number, the number of decimal points may also be specified.)

For more information about predefined ABAP types, refer to the keyword documentation for the TYPES or DATA statement.

Note:
The data types DECFLOAT16 and DECFLOAT34 was introduced in NetWeaver 7.0 EhP2. They combine the advantages of the classic data types P (exact calculation in decimal mode) and F (large value range), while avoiding the drawbacks of both classic data type.

© Copyright . All rights reserved.

Prohibida su reproduccion

Local Data Types

ABAP Program

```
REPORT ...

Declaration of local data types

TYPES gty_c_type TYPE c LENGTH 8.

TYPES gty_n_type TYPE n LENGTH 5.

TYPES gty_p_type TYPE p LENGTH 3 DECIMALS 2.
```

Figure 99: Declaring Local Types

Using standard data types, you can declare local data types in the program that can be more complete or complex than the underlying standard data types. Local data types only exist in the program in question and, hence, can only be used there. The declaration is made using the TYPES statement.

Hint:
There is an alternative syntax to specifying the length with the LENGTH addition that you will find in older programs.

The length is specified in parentheses directly after the name of the type using the following example:

TYPES gty_c_type(3) TYPE c.

TYPES gty_p_type(3) TYPE p DECIMALS 2.

To improve the readability of your program, you should no longer use this obsolete syntax.

© Copyright . All rights reserved.

Prohibida su reproduccion

Global Data Types

Figure 100: Global Data Types in the Dictionary

A data type defined in the ABAP Dictionary is called global, as it can be used throughout the entire SAP system concerned.

In this lesson, you will only learn how data elements are used as data types for elementary data objects.

© Copyright . All rights reserved.

Prohibida su reproduccion

Definition of Variable Data Objects

Figure 101: Defining Data Objects

Categories of data types:

- Built-In
- Local
- Global

These types will be used to define variables (data objects).

Data objects are always defined with the DATA keyword. You can use an ABAP standard type, a local type, or a global type to define a data object.

You can refer to a presently defined data object when defining additional variables (LIKE addition).

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
To increase the readability of your code, SAP recommends that you use naming conventions for names of self-defined types (declared with the TYPES statement) or variables (declared with the DATA statement). Throughout this course we use the following naming convention:

Purpose	Prefix
Program global* type	gty_
Program global* variable	gv_
Local** type	lty_
Local** variable	lv_

* Note: program global in this case means a type or variable that is globally visible within your program. However, it is locally defined in your program. You also can define types as system global (via entries in the Data Dictionary). These type definitions are visible (and usable) within all ABAP programs of your SAP system.

** Note: local in this case means **local to a subroutine**.

Examples of the Definition of Elementary Data Objects

Figure 102: Examples of the Definition of Elementary Data Objects

© Copyright . All rights reserved.

Prohibida su reproduccion

You can use the VALUE addition to pre-assign the value of an elementary data object.

> Hint:
> In the DATA statement, you also have the option of specifying the length in parentheses after the name of the variable using the following example:
>
> DATA gv_myvar(15) TYPE c.
>
> DATA gv_myvar_p(4) TYPE p DECIMALS 2.
>
> However in order to increase the readability of your program coding, we recommend to use the LENGTH addition.

If the length specification is missing from a variable definition, a default length for the (incomplete) standard data type is used (length 1 for types C, N, and X, and length 8 for type P). If the data type is missing as well the standard type C is used. For example, the DATA gv_myvar. statement having no type and length specification, defines a type C variable with length 1.

> Hint:
> To improve readability, you should always specify the data type and length.

For more details, refer to the keyword documentation for the TYPES or DATA statement.

Literals, Constants, and Text Symbols

Figure 103: Literals and Constants (Fixed Data Objects)

Fixed data objects have a fixed value that is defined when the source text is written and cannot be changed at runtime. Literals and constants belong to the fixed data objects.

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
For names of fixed data objects (constants) we add the following definitions to our naming convention:

Purpose	Prefix
Program global* constant	gc_

* Note: program global in this case means a type or variable that is globally visible within your program. However, it is locally defined in your program. You also can define types as system global (via entries in the ABAP Data Dictionary). These type definitions are visible (and usable) within all ABAP programs of your SAP system.

You can use literals to specify fixed values in your programs. There are numeric literals (specified without inverted commas) and text literals (specified with inverted commas). In figure "Text Symbols" you find an example for a text literal (3rd line of the WRITE statement).

You define constants using the CONSTANTS statement. The TYPE addition is used similarly to the use in the DATA. The VALUE addition is mandatory for constants. With it you define the value of constants.

Hint:
If possible, avoid literals completely when using statements. Use constants and text symbols instead. This makes it considerably easier to maintain your program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Text Symbols

Figure 104: Text Symbols

A very important principle in ABAP development is multilingual capability. This means that the logon language of the current user is taken into account when texts are displayed on the user interface. The use of text literals in this context is critical because the literal exists in only one version in the source code, and is language-independent.

Only create language-dependent texts as text literals for testing purposes. For productive programs that are executable with various logon languages, the ABAP programming language provides the text symbols.

Each text symbol is used in the particular program to which it belongs. These symbols are stored outside the source code in their own Repository object, the text pool for the program. Text symbols can be translated into various languages and each of them is stored with a language indicator in the text pool, as shown in the figure. If the program accesses a text symbol when it is executed, the system takes account of the logon language of the user and supplies the text in this language.

A text symbol is identified by means of a three-character alphanumeric ID, '-xxx'.

To use a text symbol in your program, you need to address it as TEXT-xxx, where xxx stands for the three-character text symbol ID.

To make specifying a text symbol more intuitive, you can also use the following syntax instead of TEXT-xxx: '...'(xxx). Here, '...' should be the text of the text symbol in the original language of the program.

Options for defining text symbols for your program:

© Copyright . All rights reserved.

Prohibida su reproduccion

- In the *ABAP Editor*, choose *Goto* → *Text Elements* → *Text Symbols*.
- Address the text symbol in your source code using the syntax TEXT-xxx: '...'(xxx) and double-click its ID (Forward navigation).

To translate the text symbols of your program, choose *Goto* → *Translation* from the menu in the *ABAP Editor*.

> Hint:
> Remember that text elements also have to be activated.

Comparison – Local and Global Data Types

Figure 105: Local Versus Global Data Types

Local data types can only be used in the program in which they are defined. Global data types, in contrast, can be used throughout the entire system.

Global data types have the following advantages:

- Global data types can be used throughout the system, which increases the system's consistency. The fact that they can be reused reduces the amount of maintenance efforts required.
- Global data types can have a where-used list generated in the ABAP Dictionary. The where-used list includes repository objects that use the data type in question.

© Copyright . All rights reserved.

Prohibida su reproduccion

- Global data types can contain semantic information that corresponds to the business descriptions of the objects to be defined in addition to technical information. They can then also be used for designing screen displays (for example, the short description on the left of the input field).

Local data types should be defined if they are only needed in one program, and only if the semantic information does not matter for the definition of the corresponding data objects.

LESSON SUMMARY

You should now be able to:

- Know the differences between types and variables and how to use them

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 7
Lesson 2

Using Basic ABAP Statements

LESSON OVERVIEW
This lesson explains how to fill elementary data objects with values, and how to perform calculations in ABAP. You will also be given an introduction to the constructions you can use to control the program flow dependent on the content of data objects.

Business Example

You are to create a simple ABAP program for basic calculation types. You must be able to enter values and the arithmetic operator. For this reason, you need to know:

- How to fill elementary data objects with values
- How to perform calculations in ABAP

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use the ABAP syntax of standard basic programming concepts

© Copyright . All rights reserved.

Prohibida su reproduccion

Value Assignments

Figure 106: Value Assignments

When you start a program, the program context is loaded into a memory area of the application server and memory is made available for the data objects defined in the program. Every elementary data object is pre-assigned the type-specific initial value, except if a different value was set using the VALUE addition.

You can use the MOVE statement to transfer the contents of a data object to another data object.

The following syntax variants have the same effect:

- `MOVE gv_var1 TO gv_var2.`
- `gv_var2 = gv_var1.`

If the two data objects gv_var1 and gv_var2 are of different types, there is a type conflict. In this case, a type conversion is carried out automatically if a conversion rule exists. For detailed information about copying and the conversion rules, refer to the keyword documentation for the MOVE statement.

The CLEAR statement resets the content of a data object to the type-related initial value. For detailed information about the initial values for a particular type, refer to the keyword documentation for the CLEAR statement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Calculations and Arithmetic Expressions

Figure 107: Calculations

In ABAP, you can program arithmetic expressions nested to any depth.

Valid operators:

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division)
- ** (Exponentiation)
- DIV (Integral division without remainder)
- MOD (Remainder after integral division)

Several functions for different data types are predefined in the ABAP runtime environment. For example, the following statement provides the current length of the content of a character variable.

```
gv_length = STRLEN (gd_cityfrom).
```

In the case of functions, the opening parenthesis is part of the function name. The rest must be separated by at least one space.

Standard algebraic rules apply to the processing sequence. Expressions in the parenthesis come first, followed by functions, powers, multiplication and division, and addition and subtraction.

For more information about the available operations and functions, refer to the keyword documentation for the COMPUTE statement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Conditional Branches and Logical Expressions

Figure 108: Conditional Branches

ABAP offers the following ways to execute different sequences of code depending on conditions:

- In the IF construct, you can define any logical expressions as check conditions. If the condition is met, the system executes the relevant statement block. Otherwise, the condition specified in the next ELSEIF branch (several branches are possible) is checked. If none of the specified conditions are fulfilled, the ELSE branch executes, provided it exists. The ELSEIF and ELSE branches are optional. For more information about formulating a logical expression, refer to the keyword documentation on the IF statement.
- In the CASE construct, you can clearly distinguish cases. The content of the field specified in the CASE part is checked against the data objects listed in the WHEN branch to see whether they match. If the field contents match, the respective statement block is processed. If no comparison is successful, the system executes the OTHERS branch if it is available. Except for the first WHEN branch, all further branches are optional.

In both scenarios, the condition, or match check, happens sequentially from the top down. As soon as the statement block of a branch executes, the system immediately jumps to ENDIF or ENDCASE.

Hint:
If you want to implement similarity checks between a field and different values, you should use the CASE construct in preference to the IF statement, as it is more transparent and performs better.

© Copyright . All rights reserved.

Prohibida su reproduccion

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.
```

Figure 109: Examples - IF Statement

The figure shows several simple examples (negation, AND and OR links with parentheses, and negation before logical conditions) of using the IF statement.

You can formulate negations by placing the NOT operator before the logical expression. When negating the IS INITIAL query, you can use the special IS NOT INITIAL query.

> Hint:
> A statement "IF NOT var IS INITIAL" is a valid statement as well (correct syntax). However, to increase the readability of your programming code SAP recommends to not use this statement/syntax.

You can nest the IF and CASE structures in any way, but make sure that the logic of every structure is correct here. Every structure must be closed, for example:

```
IF <log. condition1> . ...
    IF <log. condition2> . .. .
    ENDIF.
ENDIF.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Loops

Figure 110: Loops

There are four loop constructs in ABAP. The system field will make sense only if queried within a loop. In nested loops, sy-index always contains the loop pass number of the loop in which it is located. In the DO and WHILE loops, the sy-index system field contains the number of the current loop pass. Hence, you should only query this system field within a loop.

Loop constructs:

- Unconditional and index-controlled loops

  The statement block between DO and ENDDO executes continuously until the loop is left using termination statements such as EXIT. Specify the maximum number of loop passes; otherwise, you may get an endless loop.

- Header-controlled loops

  The statement block between WHILE and ENDWHILE continuously executes until the specified condition is no longer met. The condition is always checked before executing the statement block.

- Read loops

  The SELECT loop reads several entries of a database table in succession. In an internal table (table variable in the program), the same read function is implemented with the LOOP.

© Copyright . All rights reserved.

Prohibida su reproduccion

System Fields (Excerpt)

Table 3: Some Interesting System Fields

System Field	Meaning
sy-mandt sy-uname sy-langu	Logon Client Logon Name of the User Logon Language of the User
sy-datum sy-uzeit	Local Date of the ABAP System Local Time of the ABAP System
sy-tcode sy-repid	Current Transaction Code Name of the Current ABAP Program
sy-index	Loop Counter at DO and WHILE Loops

In the ABAP source code, you can use several data objects without explicitly declaring them previously (for example, sy-datum and sy-index). The runtime system uses these system fields to provide the application program with information about the actual system status. The table shows several system fields.

You will find a complete list of system fields in the keyword documentation under the term System Fields.

To access system fields in your programs, use read-only access. Write access can result in the loss of important information for program parts that require this information. In addition the runtime system may change the field content again. Therefore SAP recommends to only read these fields.

© Copyright . All rights reserved.

Prohibida su reproduccion

Return Code of an ABAP Statement

```
REPORT ...

PARAMETERS pa_carr TYPE scarr-carrid.

DATA gs_scarr TYPE scarr.

SELECT SINGLE * FROM scarr
                INTO gs_scarr
                WHERE carrid = pa_carr.

IF sy-subrc = 0.

  NEW-LINE.

  WRITE: gs_scarr-carrid,
         gs_scarr-carrname,
         gs_scarr-url.
ELSE.

  WRITE 'Sorry, no data found!'.

ENDIF.
```

Figure 111: Return Code of an ABAP Statement

One of the most important system fields is the sy-subrc field. With many statements, it is supplied by the ABAP runtime system with the corresponding return code to indicate whether the statement could be executed successfully. The value zero means that the statement was executed successfully. Read the keyword documentation for the respective statements to find out if and how this return value is set in individual cases.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Use Basic ABAP Statements

Business Example

You are to create a simple ABAP program for the four basic calculation types. You must be able to enter the values and the arithmetic operator on a selection screen. The result should be displayed in a list.

Template

None

Solution

BC400_DOS_COMPUTE

Task 1

Create a program.

1. Create the executable program **`ZBC400_##_COMPUTE`** without a *TOP include* and assign it to your package.

Task 2

Define a selection screen with three input parameters.

1. Define two input parameters for integers as operands (suggested names: **`pa_int1`**, **`pa_int2`**) and one character type arithmetic operator of one character in length (suggested name: **`pa_op`**). Use the ABAP keyword PARAMETERS to do this.

Task 3

Implement the basic calculation types and output the result in a list.

1. Create an elementary data object (suggested name: **`gv_result`**) using the DATA statement for the calculation result. Define it as a packed number type of 16 digits in length and two decimal places.
2. Calculate the result dependent on the specified arithmetic operator. Use the CASE statement for a case distinction.
3. Output the result in a list using the WRITE statement. If you want to output texts in the process, make sure that these texts can be translated and are displayed as language-dependent.

© Copyright . All rights reserved.

Prohibida su reproduccion

Hint:
Create text symbols and use these instead of literals.

Task 4

Catch any errors that may occur.

1. Display an error message on the list if the user has specified an invalid arithmetic operator. Use the IF statement for checking. Use a translatable text.
2. Display an error message on the list if the user tries to divide by zero.

Task 5

Activate and test your program.

1. Activate your program.
2. Test your program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Use Basic ABAP Statements

Business Example

You are to create a simple ABAP program for the four basic calculation types. You must be able to enter the values and the arithmetic operator on a selection screen. The result should be displayed in a list.

Template

None

Solution

BC400_DOS_COMPUTE

Task 1

Create a program.

1. Create the executable program **`ZBC400_##_COMPUTE`** without a *TOP include* and assign it to your package.
   a) Perform this step in the same way as in the previous exercises.

Task 2

Define a selection screen with three input parameters.

1. Define two input parameters for integers as operands (suggested names: **`pa_int1`**, **`pa_int2`**) and one character type arithmetic operator of one character in length (suggested name: **`pa_op`**). Use the ABAP keyword PARAMETERS to do this.
   a) See the source code excerpt from the model solution below.

Task 3

Implement the basic calculation types and output the result in a list.

1. Create an elementary data object (suggested name: **`gv_result`**) using the DATA statement for the calculation result. Define it as a packed number type of 16 digits in length and two decimal places.
   a) See the source code excerpt from the model solution below.
2. Calculate the result dependent on the specified arithmetic operator. Use the CASE statement for a case distinction.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) See the source code excerpt from the model solution below.

3. Output the result in a list using the WRITE statement. If you want to output texts in the process, make sure that these texts can be translated and are displayed as language-dependent.

> Hint:
> Create text symbols and use these instead of literals.

a) See the source code excerpt from the model solution below.

Task 4

Catch any errors that may occur.

1. Display an error message on the list if the user has specified an invalid arithmetic operator. Use the IF statement for checking. Use a translatable text.
   a) See the source code excerpt from the model solution below.
2. Display an error message on the list if the user tries to divide by zero.
   a) See the source code excerpt from the model solution below.

Task 5

Activate and test your program.

1. Activate your program.
   a) In the *ABAP Editor* toolbar, choose (*Activate*).
2. Test your program.
   a) In the *ABAP Editor* toolbar, choose (*Direct Processing*).
   b) Supply the selection screen parameters with values and choose (*Execute*).
   c) Check the result. Test the message output in the event of an error too.

Model solution:

```
*&---------------------------------------*
*& Report BC400_DOS_COMPUTE
*&---------------------------------------*
*& Simple calculator
*&---------------------------------------*
REPORT  bc400_dos_compute.


PARAMETERS:
  pa_int1  TYPE i,
  pa_op    TYPE c LENGTH 1,
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
  pa_int2  TYPE i.


DATA gv_result TYPE p LENGTH 16 DECIMALS 2.

IF ( pa_op = '+' OR
     pa_op = '-' OR
     pa_op = '*' OR
     pa_op = '/' AND pa_int2 <> 0 ).


  CASE pa_op.
    WHEN '+'.
      gv_result = pa_int1 + pa_int2.
    WHEN '-'.
      gv_result = pa_int1 - pa_int2.
    WHEN '*'.
      gv_result = pa_int1 * pa_int2.
    WHEN '/'.
      gv_result = pa_int1 / pa_int2.
  ENDCASE.

  WRITE: 'Result'(res), gv_result.

ELSEIF pa_op = '/' AND pa_int2 = 0.

  WRITE 'No division by zero!'(dbz).

ELSE.

  WRITE 'Invalid operator!'(iop).

ENDIF.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

Dialog Messages

```
MESSAGE tnnn(message_class) [ WITH v1 [ v2 ] [ v3 ] [ v4 ] ] .
```

Type	Meaning	Dialog behavior	Message appears in
i	Info Message	Program continues after breakpoint	Modal dialog box
s	Set Message	Program continues without breakpoint	Status bar *) of next screen
w	Warning	Context-dependent	Status bar *)
e	Error	Context-dependent	Status bar *)
a	Termination	Program terminated	Modal dialog box
x	Short Dump	Runtime error MESSAGE_TYPE_X is triggered	Short dump

*) Display in modal dialog box also possible through GUI settings

Figure 112: Dialog Messages

You use the MESSAGE statement to send dialog messages to the users of your program. Specify the three-digit message number and the message class when using the MESSAGE statement.

Message number and message class clearly identify the message to be displayed. Use the message type to specify where you want the message to be displayed. You can test the display behavior for various message types by means of the DEMO_MESSAGES demonstration program that is delivered with the SAP standard system.

If the specified message contains placeholders, supply them with values from your program by using the WITH addition. Instead of the placeholders, the transferred values appear in the displayed message text.

For further information about syntactical alternatives to the MESSAGE statement, refer to the keyword documentation.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Use the ABAP syntax of standard basic programming concepts

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 7
Lesson 3

Working with the ABAP Debugger

LESSON OVERVIEW

This lesson explains the working of the ABAP Debugger for elementary data objects.

Business Example

As an ABAP programmer, you are developing programs on ABAP. You are to debug the programs that you develop. For this reason, you require the following knowledge:

- An understanding of the working of the ABAP Debugger for elementary data objects

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use the ABAP Debugger to analyse values of elementary data objects during program execution.

ABAP Debugger

Figure 113: Executing a Program in Debugging Mode

© Copyright . All rights reserved.

Prohibida su reproduccion

The two ways of starting a program in debugging mode from the Object Navigator are as follows:

- In the navigation area, open the context menu for the selected program an choose *ExecuteDebugging*.
- In the ABAP Editor area, select the requested program line from which you wish to debug. Choose the *Set* or *Delete breakpoint* pushbutton. Start the program by pressing F8 or, *by opening the context menu in the navigation area and choosing Execute → Direct.*

  The above-described setting of a breakpoint in the ABAP Editor is only possible for active source texts.

Switching to Debugging Mode at Runtime

Figure 114: Switching to Debugging Mode at Runtime

If you want to debug a certain function of a program, start the program without the ABAP Debugger and then switch to debug mode immediately before executing the function (for example, pushbutton).

Ways of switching to debugging at run time:

- Choose *System → Utilities → Debugging ABAP (or screen).*
- Enter `/h` in the command field in the standard toolbar and press ENTER.

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Debugger – Single Step and Field Contents

Figure 115: ABAP Debugger - Single Step and Field Contents

In the ABAP Debugger, you can choose single step processing to execute the program statement-by-statement. Moreover, you can display data objects and their current content in the variable display. Simply enter the name of the data object in the variable display. Alternatively you can double-click on the data object name in the source code to transfer the variable name into the debugger field list.

> Note:
> In systems on a release below SAP NetWeaver AS 6.40, you only have access to the "classic" ABAP Debugger. With higher release levels, you can use both the new and the classic ABAP Debugger. You can switch easily from the new to the classic ABAP Debugger by choosing *Debugger → Switch to Classic ABAP Debugger* from the menu.
>
> In the classic ABAP Debugger, you can display the contents of up to eight data objects. To do this, proceed as you would with the new ABAP Debugger.
>
> In the *Object Navigator*, you can define which ABAP Debugger you want to use as standard by choosing *Utilities → Settings → ABAP Editor → Debugging.*

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Debugger – Breakpoints

Figure 116: ABAP Debugger - Breakpoints

In the new ABAP Debugger, you can set a breakpoint with a single-click before the line in the source code (in the classic ABAP Debugger, you do this with double-click).

You can also set a breakpoint for specific ABAP statements by choosing *Breakpoints → Breakpoint at → Statement*. If you choose *Continue*, the program executes up to the next breakpoint.

The set breakpoints are only valid for the current ABAP Debugger session. However, if you choose *Save*, the breakpoints will stay in place for the duration of your current SAP session.

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Debugger – Tracing Data Changes

Figure 117: ABAP Debugger - Tracing Data Changes

Watchpoints are breakpoints that depend on the field content.

If you set a watchpoint without specifying a relational operator and a comparative value on a field and choose *Continue*, the program executes until the content of the field changes.

If you specify the relational operator and the comparative value, the program executes until the specified condition is met.

In the classic ABAP Debugger, you can set a maximum of 10 watchpoints. You can link the watchpoints using a logical operator (AND or OR) in the same way as in the new ABAP Debugger.

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Debugger – Changing Field Contents

Figure 118: ABAP Debugger - Changing Field Contents

If you want to change the content of a field during debugging, double-click the *pencil* icon in the variable display. The value is then ready for input. Now change the field value and confirm it by pressing ENTER. The value changes while the ABAP Debugger is running.

In the classic ABAP Debugger, you can change the content directly in the field view. You then click the *pencil* icon to accept the changed value.

Functions of the New ABAP Debugger

The new ABAP Debugger allows the following additional functions:

- ABAP Debugger in second session

  Inputs for the application program visible in parallel
- Parallel display options through freely configurable display areas
- Integration of the new ABAP Editor

  (As of SAP NetWeaver 7.0)
- Quick Info in the source code area for data object value display

  (As of SAP NetWeaver 7.0)
- Watchpoints for internal tables and object references

  (As of SAP NetWeaver 7.0)

© Copyright . All rights reserved.

Prohibida su reproduccion

- Value comparison of strings, structures, and internal tables

  (As of SAP NetWeaver 7.0)

In SAP NetWeaver 7.0 EhP2 a set of new features and improvements have been added to the new ABAP Debugger.

Features of SAP NetWeaver 7.0 EhP2

The new ABAP Debugger of SAP NetWeaver 7.0 EhP2 has the following features:

- Request-based debugging of HTTP and RFC requests
- Software-layer aware debugging (SLAD)
- Automated debugging-debugger scripting
- Memory consumption analysis
- Enhanced Web Dynpro tool
- Enhanced table tool
- Statement debugging

© Copyright . All rights reserved.

Prohibida su reproduccion

Debug Statements on Elementary Data Objects

Business Example

You are given a program and have to use the ABAP Debugger to work out its semantic behavior. You will need to execute a program in the ABAP Debugger, set breakpoints in the ABAP Debugger, set content-dependent breakpoints (watchpoints), and change field contents during debugging.

Task 1

Execute program BC400_DED_DEBUG_EXERCISE in debugging mode and trace the content of GV_COUNT.

1. Start program BC400_DED_DEBUG_EXERCISE in debugging mode by way of the object list in the Object Navigator.
2. Check the content of the data objects in the program at the start. To do this, include all data objects defined in the program and the SY-INDEX system field in the variable view. Take a look at the typing of the fields, too.

Task 2

Analyze the WHILE loop by setting breakpoints and tracking the content of GV_COUNT and SY-INDEX.

1. Follow the program flow in single steps until you have run through the WHILE loop twice. In each case, track how the field content changes.
2. Set a breakpoint at the statement that assigns a new value to the GV_COUNT data object. Choose (*Continue*) to continue the program and observe changes to the field content. Repeat the final step until the program has run its course and the list is displayed.
3. Execute the program again in debugging mode. Analyze the change to the GV_COUNT data object by creating a watchpoint for the field. Choose *Continue* to continue the program and observe the change to the content of the data object and the SY-INDEX system field.

Task 3

Analyze the nested DO loop by setting a breakpoint at WRITE and EXIT statements.

1. Restart the program in debugging mode. Set breakpoints at all WRITE and EXIT statements in the program (choose *Breakpoints → Breakpoint at → Breakpoint at Statement*). Analyze how

© Copyright . All rights reserved.

Prohibida su reproduccion

the contents of loop counters GV_IDX_OUTER and GV_IDX_INNER change, the information that is output, and when it is output.

2. What effect does the EXIT statement have in a loop?

_____

_____

_____

Task 4

Use the ABAP Debugger to change the program flow in such a way that the last statement in the program is also executed.

1. Make the program execute the last statement. Use the option of changing the content of data objects in debugging mode.

© Copyright . All rights reserved.

Prohibida su reproduccion

Debug Statements on Elementary Data Objects

Business Example

You are given a program and have to use the ABAP Debugger to work out its semantic behavior. You will need to execute a program in the ABAP Debugger, set breakpoints in the ABAP Debugger, set content-dependent breakpoints (watchpoints), and change field contents during debugging.

Task 1

Execute program BC400_DED_DEBUG_EXERCISE in debugging mode and trace the content of GV_COUNT.

1. Start program BC400_DED_DEBUG_EXERCISE in debugging mode by way of the object list in the Object Navigator.
   a) Display the program in the *Object Navigator*, right-click the program and choose *Execute → Debugging.*
   b) Alternatively, in the ABAP Editor, set a breakpoint at the first executable statement.
2. Check the content of the data objects in the program at the start. To do this, include all data objects defined in the program and the SY-INDEX system field in the variable view. Take a look at the typing of the fields, too.
   a) In the source code in the ABAP Debugger, double-click the GV_COUNT, GV_IDX_INNER, GV_IDX_OUTER, and SY-INDEX fields. Scroll to the right of the variable display to display field types.

Task 2

Analyze the WHILE loop by setting breakpoints and tracking the content of GV_COUNT and SY-INDEX.

1. Follow the program flow in single steps until you have run through the WHILE loop twice. In each case, track how the field content changes.
   a) Carry out the single step analysis by choosing *Debugger → Single Step*, (*Single Step*), or F5.
2. Set a breakpoint at the statement that assigns a new value to the GV_COUNT data object. Choose (*Continue*) to continue the program and observe changes to the field content. Repeat the final step until the program has run its course and the list is displayed.
   a) Click once in the first column in front of the value assignment to set a breakpoint there.

© Copyright . All rights reserved.

Prohibida su reproduccion

b) Choose *Debugger* → *Continue* from the menu, (*Continue*), or F8.

3. Execute the program again in debugging mode. Analyze the change to the GV_COUNT data object by creating a watchpoint for the field. Choose *Continue* to continue the program and observe the change to the content of the data object and the SY-INDEX system field.
   a) To access the ABAP Debugger, repeat the first step in this exercise (see step 1 of Task 1).
   b) Position the cursor on the GV_COUNT data object in the source code displayed in the ABAP Debugger. Choose the *Watchpoint* pushbutton and confirm the dialog box that appears with .
   c) To restart the program, choose (*Continue*).

Task 3

Analyze the nested DO loop by setting a breakpoint at WRITE and EXIT statements.

1. Restart the program in debugging mode. Set breakpoints at all WRITE and EXIT statements in the program (choose *Breakpoints* → *Breakpoint at* → *Breakpoint at Statement*). Analyze how the contents of loop counters GV_IDX_OUTER and GV_IDX_INNER change, the information that is output, and when it is output.
   a) To access the ABAP Debugger, proceed as before.
   b) In the ABAP Debugger, choose *Breakpoints* → *Breakpoint at* → *Breakpoint at Statement* and enter the WRITE statement in the dialog box that appears. Confirm the dialog box that appears with .
   c) Choose (*Continue*) and analyze the program flow.
2. What effect does the EXIT statement have in a loop?

   The EXIT statement causes the system to cease processing the current DO loop.

Task 4

Use the ABAP Debugger to change the program flow in such a way that the last statement in the program is also executed.

1. Make the program execute the last statement. Use the option of changing the content of data objects in debugging mode.
   a) Set a breakpoint at the last IF statement and choose *Continue* to run the program to this point.
   b) Include the *GV_IDX_INNER* field in the variable view. Double-click the pencil icon in the variable view. Change the field value from 6 to 5 and confirm the change by choosing *Enter*.
   c) Continue the program by choosing *Single Step* or *Continue*.

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
Alternatively, you can also execute the last statement using the *Goto Statement* function. In debugging mode, place the cursor on this line at any event and choose *Debugger → Goto Statement.*

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Use the ABAP Debugger to analyse values of elementary data objects during program execution.

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Which of the following are complete ABAP standard types?

   Choose the correct answers.

   - [ ] A T (Time)
   - [ ] B C (Character)
   - [ ] C N (Numerical character)
   - [ ] D D (Date)

2. In IF statements, negations are usually formulated by placing the _____ operator before the logical expression.

   Choose the correct answer.

   - [ ] A AND
   - [ ] B OR
   - [ ] C END
   - [ ] D NOT

3. Which of the following are required in the syntax of the Message statement?

   Choose the correct answers.

   - [ ] A Message number
   - [ ] B Message type
   - [ ] C Message class
   - [ ] D Message role

© Copyright . All rights reserved.

Prohibida su reproduccion

4. In nested loops, which of the following contains the loop pass number of the loop in which it is located?

   Choose the correct answer.

   - [ ] A sy-index
   - [ ] B sy-repid
   - [ ] C sy-uname
   - [ ] D sy-mandt

5. To switch to debugging mode at run time, we can enter _____ in the command field .

   Choose the correct answer.

   - [ ] A /d
   - [ ] B /h
   - [ ] C /i
   - [ ] D /a

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Which of the following are complete ABAP standard types?

   Choose the correct answers.

   - [x] A T (Time)
   - [ ] B C (Character)
   - [ ] C N (Numerical character)
   - [x] D D (Date)

2. In IF statements, negations are usually formulated by placing the _____ operator before the logical expression.

   Choose the correct answer.

   - [ ] A AND
   - [ ] B OR
   - [ ] C END
   - [x] D NOT

3. Which of the following are required in the syntax of the Message statement?

   Choose the correct answers.

   - [x] A Message number
   - [x] B Message type
   - [x] C Message class
   - [ ] D Message role

© Copyright . All rights reserved.

Prohibida su reproduccion

4. In nested loops, which of the following contains the loop pass number of the loop in which it is located?

   Choose the correct answer.

   - [x] A sy-index
   - [ ] B sy-repid
   - [ ] C sy-uname
   - [ ] D sy-mandt

5. To switch to debugging mode at run time, we can enter _____ in the command field .

   Choose the correct answer.

   - [ ] A /d
   - [x] B /h
   - [ ] C /i
   - [ ] D /a

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 8 Modularization

Lesson 1

Introducing Modularization 273

Lesson 2

Modularizing Using Subroutines 278
Exercise 15: Implement Subroutines 289

Lesson 3

Modularizing Using Function Modules 296
Exercise 16: Use a Function Module 305

Lesson 4

Implementing Function Modules 312
Exercise 17: Create a Function Group 315
Exercise 18: Create a Function Module 321

Lesson 5

Modularizing Using BAPIs 328

Lesson 6

Modularizing Using Global Classes 333
Exercise 19: Use a Global Static Method 345

Lesson 7

Implementing Simple Global Classes and Static Methods 351
Exercise 20: Create a Simple Global Class 355
Exercise 21: Create a Simple Static Method 357

Lesson 8

Modularizing Using Local Classes 362

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT OBJECTIVES

- Describe the basics and an overview of modularization
- Modularize using subroutines
- Use function modules of function groups
- Create function groups
- Create function modules
- Modularize using BAPIs
- Describe object-oriented programming
- Use methods of global classes
- Handle instances
- Create simple global classes
- Handle local classes

© Copyright . All rights reserved.

Prohibida su reproduccion

Introducing Modularization

LESSON OVERVIEW
This lesson explains reasons why it could be reasonable to store parts of programs in modularization units. You can also gain an overview of the various modularization options in ABAP programs.

Business Example

An employee in quality assurance has discovered that many program parts are frequently repeated. Your task is to find out which modularization techniques can be implemented. For this reason, you require the following knowledge:

- An understanding of the basic modularization techniques

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe the basics and an overview of modularization

Modularization Techniques

Figure 119: Options for Using Modularization Units

© Copyright . All rights reserved.

Prohibida su reproduccion

A modularization unit is a part of a program that encapsulates a particular function. You store part of the source code in a module to improve the transparency of the program, as well as to use the corresponding function in the program several times without having to re-implement the entire source code on each occasion (see the figure).

The improvement in transparency is a result of the program becoming more function-oriented. It divides the overall task into sub-functions, which are the responsibility of corresponding modularization units.

Modularization makes it easier to maintain programs, since you only need to make changes to the function or corrections in the respective modularization units, and not at various points in the main program. Furthermore, you can process a call as a unit in the ABAP Debugger while executing your program and then see the result. This makes it easier to find the source of an error.

Local Program Modularization

Figure 120: Local Program Modularization

The techniques for local program modularization in the ABAP programming language are as follows:

- Subroutines, also known as form routines
- Methods in local classes

With both local modularization techniques, modularization units are only available in the program in which they were implemented. To call the local module, no other program must be loaded to the user context at runtime. Local classes, methods, and subroutines can have the same name in different programs without producing conflicts. This is because the source code for the programs is handled separately in the main memory of the application server.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> It is technically possible to call a subroutine from another program. Do not use this option, however, because this technique contradicts the principle of encapsulation of data and functions.

Global Modularization

Figure 121: Global Modularization

The techniques for global modularization in the ABAP programming language are as follows:

- Function modules organized in function groups
- Methods in global classes

Globally defined modularization units can be used by any number of programs at the same time. These modularization units are stored centrally in the Repository and loaded when called into the context of the calling program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Data Encapsulation

Figure 122: Separating Data

Ideally, the modularization units that are called do not use the data objects of the calling program directly. Conversely, the calling program does not directly change the data in the modularization units. This principle is known as data encapsulation.

Data encapsulation is an important aid in developing transparent, maintainable source codes. Data encapsulation makes it far easier to determine where in the program the contents of data objects were changed. In addition, data encapsulation makes it easier to ensure that data within the modularization units changes consistently when, for example, the contents of several data objects within a modularization unit are mutually dependent.

As an example, consider a modularization unit in which you process a series of invoice documents. Entering different invoice numbers for invoice items that belong to one another can have serious consequences. It can then be difficult for those responsible for the modularization unit to determine the cause of the data inconsistency.

© Copyright . All rights reserved.

Prohibida su reproduccion

Data Transports, Parameters, and Interface

Figure 123: Data Transports Between the Program and the Modularization Unit

Parameters are used to exchange data between the program and the module. The total number of parameters in a modularization unit is referred to as the interface, or signature. The developer who defines the modularization unit also determines the parameters.

Parameters are differentiated on the basis of whether they are used to do the following:

- Pass data to the modularization unit (importing parameters).
- Return data from the modularization unit to the caller (exporting parameters).
- Pass data to the modularization unit and return the data after it has been changed (changing parameters).

> Hint:
> With subroutines (form routines), only changing parameters and the very specialized using parameters are available, which severely restricts your options for controlling data transport. For this reason, use local classes for local program modularization, where possible.

LESSON SUMMARY

You should now be able to:

- Describe the basics and an overview of modularization

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 2

Modularizing Using Subroutines

LESSON OVERVIEW

This lesson explains how to employ subroutines in ABAP programs. It also shows how the interface of a subroutine is used to pass parameters, and how various transfer types are used.

Business Example

You need to structure your program and encapsulate the source code that is executed several times in a subroutine. For this reason, you require the following knowledge:

- An understanding of how to define and call subroutines
- An understanding of the execution of subroutines in debugging mode

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Modularize using subroutines

Program-internal Modularization with Subroutines

Figure 124: Simple Example of a Subroutine

© Copyright . All rights reserved.

Prohibida su reproduccion

A subroutine is a modularization unit within a program. For the ABAP interpreter, a subroutine is always part of the main program. No parameters are used in the example shown in the figure, which makes the syntax of the subroutine very simple. But a subroutine normally uses values from data objects and also returns values. The next figure illustrates how these variables can be used in the subroutine.

Parameter Definition for Subroutines

Figure 125: Passing Parameters – Visibility of Global Variables

Variables defined in the main program are globally visible within the program and can be changed at any point within the program. This means that also subroutines defined within the program can change the variables the variables.

© Copyright . All rights reserved.

Prohibida su reproduccion

Passing Parameters – Defining an Interface

Figure 126: Passing Parameters – Defining an Interface

From a subroutine, you can address all global variables defined in the main program. However, to call a subroutine with different data objects for each situation, you must use placeholders instead of global variables. These placeholders are replaced with the required global variables when the subroutine is called. These placeholders are called formal parameters and together they form the subroutine interface. You need to declare the interface when you define the subroutine.

When the subroutine is called, formal parameters must be specialized by means of corresponding global variables (actual parameters). This assignment of actual parameters to formal parameters when calling a subroutine is called parameter passing.

© Copyright . All rights reserved.

Prohibida su reproduccion

Ways of Passing Interface Parameters

Figure 127: Ways of Passing Interface Parameters

The way these main program variables are passed to the formal parameters of the subroutine is called the pass type and is specified for each parameter in the subroutine interface.

Pass Types for Subroutines

Pass types for subroutines:

- Call by value

  The system makes a copy of the actual parameter and assigns this copy to the formal parameter. Any value assignments to the corresponding formal parameter in the subroutine, therefore, refer only to the copy of the actual parameter, not to the original.

  Use this pass type to make the value of a global variable available to the subroutine, in the form of a variable copy, without allowing the subroutine to change the respective global variable. Creating a copy of the variable value protects the original. However, creating copies, especially for large internal tables, can be time consuming.

- Call by value and result

  The call by value and result pass type is similar to the call by value pass type. However, at the regular end of the subroutine, the value that was changed in the copy is written back to the original. When the program is prematurely terminated by a STOP statement or a type E user message, the writing back of the value is suppressed.

© Copyright . All rights reserved.

Prohibida su reproduccion

You use this pass type to transfer the value of a global variable to the subroutine and to write the fully processed final value of the copy back to the original. However, it can be time consuming to create copies and write back values, especially for large internal tables.

- Call by reference

  The system assigns the actual parameter directly to the formal parameter. This means that value assignments to the formal parameter are carried out directly on the actual parameter.

  You use this pass type to run subroutine processing directly on the specified actual parameter. It is a useful way of avoiding the time-consuming creation of copies for large internal tables.

Definition and Call of Subroutines

Figure 128: Defining and Calling Subroutines

Define a subroutine in the following way:

1. Introduce a subroutine with FORM.
2. Specify the name and the interface of the subroutine after FORM. The statements of the subroutine then follow.
3. Conclude the subroutine with the ENDFORM statement.

In the interface definition, you can list the formal parameters of the subroutine (f1, f2, and f3) and type them.

Specify the required pass type for each parameter as follows:

- Call by value

© Copyright . All rights reserved.

Prohibida su reproduccion

List each of the formal parameters that must have the pass type “call by value” (f1) with the VALUE prefix under USING. Refer to the figure for the syntax.

- Call by value and result

  List each of the formal parameters that must have the pass type “call by value and result” (f2) with the VALUE prefix under CHANGING. Refer to the figure for the syntax.

- Call by reference

  List each of the formal parameters that must have the pass type “call by reference” (f3) without the VALUE prefix under CHANGING. Refer to the figure for the syntax.

> Hint:
> A parameter placed under USING without the VALUE prefix also has the pass type “call by reference”. However, this declaration syntax only makes sense for formal parameters to which larger internal tables are passed. These formal parameters are not changed in the subroutine (see the documentation for USING), but are passed using “call by reference” to avoid making time-consuming copies.

When the subroutine is called, the actual parameters to be transferred without the VALUE prefix are specified under USING or CHANGING. The order of specification determines their assignment to the formal parameters. In the example shown in the figure, a is passed to f1, b to f2, and c to f3.

Typing of Interface Parameters

Figure 129: Typing the Interface Parameters

© Copyright . All rights reserved.

Prohibida su reproduccion

A formal parameter is typed generically when it is typed using TYPE ANY, or not typed at all. Actual parameters of any type can be transferred to such a parameter.

At runtime, the type of the actual parameter is determined and assigned to the formal parameter (type inheritance) when the subroutine is called. However, when the statements in the subroutine are not suited to the inherited type, a runtime error can occur (type conflict). Only use generic typing when the type of the actual parameter is not known when the program is created, or when the type of the actual parameter can vary at runtime (dynamic programming).

You implement the concrete typing of a formal parameter by specifying a global or built-in type in the TYPE addition. In doing so, you determine that only actual parameters of the specified type can be passed to the subroutine. A violation of the type consistency between formal and actual parameters is detected in the syntax check. The syntax check increases the stability of your program by preventing type conflicts in statements within the subroutine.

When you type with the standard types P, N, C, or X, the missing characteristic 'field length' is passed from the actual parameter to the formal parameter at runtime. You achieve complete typing with these types (that is, including the field length) by defining and specifying locally defined types.

Hint:
Besides elementary data objects, ABAP also knows structures and internal tables. Formal parameters for such data objects must always be typed in order for their components to be accessed within the subroutine.

Note:
“<scope><kind>_<name>” -> “<scope>” p = using/passing, “<scope>” c = changing

© Copyright . All rights reserved.

Prohibida su reproduccion

Local and Global Data Objects

```
TYPES gty_perc TYPE p DECIMALS 2.

DATA: gv_int1   TYPE i,
      gv_int2   TYPE i,
      gv_result TYPE gty_perc.
...
PERFORM calc_perc
           USING gv_int1
                 gv_int2
        CHANGING gv_result.
...
```

Global variables

```
FORM calc_perc
  USING
    pv_act TYPE i
    pv_max TYPE i
  CHANGING
    cv_pc  TYPE gty_perc.

   DATA lv_pc TYPE p LENGTH 16 DECIMALS 1.

...

ENDFORM.
```

Formal parameters (only visible locally)

Local variables (only visible locally)

Figure 130: Visibility of Global and Local Data Objects

Variables defined in the main program are global data objects. They are visible and can be addressed in the entire main program, as well as in every subroutine called.

Variables defined within a subroutine are called local because they exist in the relevant subroutine in the same way as formal parameters. Memory space for formal parameters and local data objects is allocated when the subroutine is called and is released again after execution.

> Note:
> “<scope><kind>_<name>” -> “<scope>” l = local

The formal parameters and local data objects of a subroutine cannot have the same names. When there is a global data object with the same name as a formal parameter or a local data object, the formal parameter or local data object is addressed within the subroutine, and the global data object is addressed outside the subroutine. This is the so-called shadow rule: within a subroutine, the local data object shadows the global one with the same name.

To identify your internal program objects uniquely, use the following prefixes for your subroutine objects. **`g...`** for global data objects, **`p...`** for using parameters, **`c...`** for changing parameters, and **`l...`** for local data objects.

© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Local Auxiliary Variable for Rounding

```
TYPES gty_perc TYPE p DECIMALS 2.

DATA: gv_int1   TYPE i,
      gv_int2   TYPE i,
      gv_result TYPE gty_perc.
...
PERFORM calc_perc
           USING gv_int1
                 gv_int2
        CHANGING gv_result.
...
```

```
FORM calc_perc
  USING
    value(pv_act) TYPE i
    value(pv_max) TYPE i
  CHANGING
    value(cv_pc)  TYPE gty_perc.

   DATA lv_pc TYPE p LENGTH 16 DECIMALS 1.

   lv_pc = pv_act * 100 / pv_max.
   cv_pc = lv_pc.

ENDFORM.
```

Figure 131: Syntax Example – Local Auxiliary Variables for Rounding

In the syntax example illustrated in the figure, the result of the percentage calculation must be rounded internally to have one decimal place, but, nonetheless, return with two decimal places.

To do this, you create a local variable with only one decimal place and store the result of the calculation there first. The runtime system rounds the result to the nearest whole number in accordance with the available number of decimal places. When you copy the result to the return parameter, a zero is added for the second decimal place.

© Copyright . All rights reserved.

Prohibida su reproduccion

Call of Subroutines

Figure 132: Implementing a Subroutine Call

When calling a subroutine, you can make the ABAP workbench generate the PERFORM statement into your source code. Define the subroutine and then save your main program. The newly defined subroutine appears in the navigation area. Move it to the required call point in your program by using the drag-and-drop function. Then replace the formal parameters in the generated source code with corresponding actual parameters.

Alternatively, you can implement call generation by using the *Pattern* pushbutton in the *ABAP Editor*.

The advantage of generating the call is that it is impossible to forget or mix parameters.

© Copyright . All rights reserved.

Prohibida su reproduccion

Modularization Units in the ABAP Debugger

Figure 133: Subroutine in Debugging Mode

If the current statement is a subroutine call, you can execute the entire subroutine without stopping by choosing the *Execute* pushbutton. Processing stops after the subroutine has been completed.

In contrast, you can use single steps to stop at the first statement of the subroutine and trace its operations in more detail.

If the current statement is located in a subroutine, you can execute the rest of the subroutine without stopping by choosing the *Return* pushbutton. Processing stops after the subroutine has been completed.

Control functions of the ABAP Debugger, including single step, execute, return, and continue, are also available in the classic ABAP.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Subroutines

Business Example

Modify your calculator program for the basic calculation types so that you can perform another form of calculation (percentage calculation) in a subroutine.

Template:

BC400_DOS_COMPUTE

Solution:

BC400_MOS_SUBROUTINE

Create subroutines and use the subroutine interface to pass data.

Task 1

Copy your program ZBC400_##_COMPUTE or the template and give it the name **`ZBC400_##_SUBROUTINE`**.

1. Copy your program or the template.

Task 2

Create a subroutine (suggested name: **`CALC_PERCENTAGE`**) that carries out a percentage calculation using two input parameters and returns the result. The subroutine should calculate the percentage value that corresponds to the first parameter, if the value of the second parameter represents 100%.

1. At the end of your program, create a subroutine for percentage calculation.
2. Define two USING parameters for transferring the operands (suggested names: **`PV_ACT`** and **`PV_MAX`**) as well as a CHANGING parameter for returning the result (suggested name: **`CV_RESULT`**). Type the parameters appropriately to the corresponding global data objects in the main program.

> Hint:
> You cannot use implicit typing for return parameters (**`TYPE p LENGTH 16 DECIMALS 2`**). Instead, declare a local program type at the start of your program with the TYPES statement. You can then use this local program type for the actual parameter and the formal parameter.

© Copyright . All rights reserved.

Prohibida su reproduccion

3. Implement the percentage calculation in the subroutine. Make sure that you avoid the error of division by zero. In this case, output a warning text on the list.

Task 3

Extend the main program so that you call your new subroutine when the user enters the arithmetic operator “%”.

1. Extend the IF or CASE structure with a branch that is processed when the parameter contains the value “%”.
2. In the branch created in step 1, call the subroutine and supply the parameters with suitable actual parameters.

> Hint:
> Generate the subroutine call by using the drag-and-drop function in the navigation area or by choosing the *Pattern* pushbutton.

Task 4

Test and debug your program.

1. Execute your program and check the result.
2. Follow the program flow with the ABAP Debugger. Use the appropriate function keys or pushbuttons to exit or skip the subroutine.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Subroutines

Business Example

Modify your calculator program for the basic calculation types so that you can perform another form of calculation (percentage calculation) in a subroutine.

Template:

BC400_DOS_COMPUTE

Solution:

BC400_MOS_SUBROUTINE

Create subroutines and use the subroutine interface to pass data.

Task 1

Copy your program ZBC400_##_COMPUTE or the template and give it the name **`ZBC400_##_SUBROUTINE`**.

1. Copy your program or the template.
   a) Display your template in the navigation area of the *Object Navigator*.
   b) Right-click the selected program to open the context menu and choose *Copy*.
   c) In the dialog box that follows, enter the name of the target program and confirm your entry with ✔ *(Copy)*.
   d) Select all parts of the program using (*Select All*) and confirm this with ✔ *(Copy)*.
   e) Assign this program to your package and your request in the usual way.

Task 2

Create a subroutine (suggested name: **`CALC_PERCENTAGE`**) that carries out a percentage calculation using two input parameters and returns the result. The subroutine should calculate the percentage value that corresponds to the first parameter, if the value of the second parameter represents 100%.

1. At the end of your program, create a subroutine for percentage calculation.
   a) See the source code excerpt from the model solution.
2. Define two USING parameters for transferring the operands (suggested names: **`PV_ACT`** and **`PV_MAX`**) as well as a CHANGING parameter for returning the result (suggested name:

© Copyright . All rights reserved.

Prohibida su reproduccion

CV_RESULT). Type the parameters appropriately to the corresponding global data objects in the main program.

> Hint:
> You cannot use implicit typing for return parameters (**TYPE p LENGTH 16 DECIMALS 2**). Instead, declare a local program type at the start of your program with the TYPES statement. You can then use this local program type for the actual parameter and the formal parameter.

a) See the source code excerpt from the model solution.

3. Implement the percentage calculation in the subroutine. Make sure that you avoid the error of division by zero. In this case, output a warning text on the list.
   a) See the source code excerpt from the model solution.

Task 3

Extend the main program so that you call your new subroutine when the user enters the arithmetic operator “%”.

1. Extend the IF or CASE structure with a branch that is processed when the parameter contains the value “%”.
   a) See the source code excerpt from the model solution.
2. In the branch created in step 1, call the subroutine and supply the parameters with suitable actual parameters.

> Hint:
> Generate the subroutine call by using the drag-and-drop function in the navigation area or by choosing the *Pattern* pushbutton.

a) See the source code excerpt from the model solution.

Task 4

Test and debug your program.

1. Execute your program and check the result.
   a) Carry out this step as usual.
2. Follow the program flow with the ABAP Debugger. Use the appropriate function keys or pushbuttons to exit or skip the subroutine.
   a) Perform this step as described in the course materials.

© Copyright . All rights reserved.

Prohibida su reproduccion

Solution: BC400_MOS_SUBROUTINE

```
*&---------------------------------------------*
*& Report BC400_MOS_SUBROUTINE
*&---------------------------------------------*
REPORT bc400_mos_subroutine.

TYPES gty_result TYPE  p LENGTH 16 DECIMALS 2.

PARAMETERS: pa_int1 TYPE i,
            pa_op TYPE c LENGTH  1,
            pa_int2 TYPE i.

DATA gv_result TYPE gty_result.

IF  ( pa_op  = '+' OR pa_op  = '-' OR
      pa_op  = '*' OR pa_op  = '/' AND
      pa_int2  <>  0  OR  pa_op = '%'  ).

  CASE pa_op.
    WHEN '+'.
      gv_result  = pa_int1  + pa_int2.
    WHEN '-'.
      gv_result  = pa_int1  - pa_int2.
    WHEN '*'.
      gv_result  = pa_int1  * pa_int2.
    WHEN '/'.
      gv_result  = pa_int1  / pa_int2.
    WHEN '%'.
      PERFORM calc_percentage USING pa_int1 pa_int2
                          CHANGING gv_result.
  ENDCASE.

  WRITE: 'Result:'(res), gv_result.
ELSEIF pa_op  = '/' AND pa_int2  =  0.
  WRITE: 'No division by zero!'(dbz).
ELSE.
  WRITE: 'Invalid operator!'(iop).
ENDIF.

*&---------------------------------------------*
*&      Form  calc_percentage
*&---------------------------------------------*
*    Calculate percentage value
*&---------------------------------------------*
*    -> PV_ACT      actual value
*    -> PV_MAX      maximum value
*    <- CV_RESULT   result
*&---------------------------------------------*

FORM calc_percentage  USING  pv_act TYPE i
                             pv_max TYPE i
                      CHANGING cv_result TYPE gty_result.

* Simple Error Handling
  IF pv_max = 0.
    cv_result = 0.
    WRITE 'Error in percentage calculation'.
  ELSE.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
* Calculate result
    cv_result = pv_act / pv_max * 100.
  ENDIF.

ENDFORM.  " calc_percentage
```


© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Modularize using subroutines

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 3

Modularizing Using Function Modules

LESSON OVERVIEW

This lesson explains how to search for function modules, analyze their interfaces and documentation, and test their functions. You will learn how to use existing function modules in your programs and catch and handle errors. Finally, you will create function modules and encapsulate functions that you can then reuse in other programs.

Business Example

You need a function for your program that is already available in the form of a function module. In addition, you want to use this function module in your program. For this reason, you require the following knowledge:

- An understanding of how to search for function modules
- An understanding of how to acquire information about the functionality and use of function modules
- An understanding of how to call function modules in your program
- An understanding of how to create a function group and a function module
- An understanding of BAPIs and their special properties

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use function modules of function groups

© Copyright . All rights reserved.

Prohibida su reproduccion

Structure of Function Groups

Figure 134: Function Group

A function module is a procedure with a corresponding function that is stored centrally in the Function Library of the SAP system. Each function module has an interface with parameters for importing or exporting data. The purpose function modules is their reusability.

Function modules are assigned to function groups. Each function group is a collection of function modules that have related functions and/or process the same data.

A function group can contain the same components as an executable program.

Components of a function group:

- Data objects

  Data objects are global in relation to the function group; that is, they are visible to, and can be changed by, all function modules within the group.

- Subroutines

  Subroutines can be called from all function modules in the group.

- Screens

  Screens can be called from all function modules in the group.

© Copyright . All rights reserved.

Prohibida su reproduccion

Function Modules

Figure 135: Function Modules

The properties of a function module include a short description of the module and the name of the function group to which it belongs.

Similar to subroutines, a function module can contain its own local type and data object definitions. Local types and data objects can only be seen within the function module.

The interface of a function module may contain the following elements:

- Import parameter

  When the function module is called, the calling program transfers values or variables to the function module. If a parameter is optional, there is no need to supply it with a value or a variable.

- Export parameter

  The calling program accepts the output of the function module by way of the assignment of a “receiving variable” to an export parameter. Export parameters are always optional.

- Changing parameter

  When the function module uses changing parameters variables are transferred from the calling program to the function module and returned at the end of the call.

- Exceptions

  Exceptions can be raised by the function module in certain error situations and provide information about the respective processing error in the function module. Exceptions should be handled by the calling program.

In general, interface parameters are defined using types from the ABAP Dictionary.

© Copyright . All rights reserved.

Prohibida su reproduccion

Data Objects Within a Function Group

Figure 136: Data Objects Within a Function Group

When a program calls a function module, the corresponding function group is loaded and the function module is executed. The function group remains loaded in the working memory until the calling program is closed. When you call another function module of this group, it is processed without being reloaded, and with the same global data of the function group.

If a function module that writes values to the global data of the function group is called, other function modules in the same function group can access the data when they are called in the same program run.

Apart from the global data of its function group, a function module can also access its own locally defined data objects and interface parameters. Interface parameters receive data or return it to the calling program.

Search for Function Modules

You can search for function modules as follows:

- Application-related search

  Search within a particular application component (application hierarchy).

- Application-independent search

  Perform a free search, independent of application components (Repository Information System).

- Program-related search

© Copyright . All rights reserved.

Prohibida su reproduccion

Search for the CALL FUNCTION statement within the program source code.

Before triggering a program function (for example, choosing a pushbutton) using “/h”, activate the ABAP Debugger and set a breakpoint at the CALL FUNCTION statement.

When the searched function module passes a screen, use F1 and “Technical Information” to determine the screen number, navigate to it by double-clicking on it, and execute the “Where-Used List in Programs”.

Search scenarios:

- An application-related search through the application hierarchy must be used to search for function modules within one or more application components.
- You use the free search through the Repository Information System to search for function modules independent of application components.
- You use the program-related search when the function module for which you are searching is called within an existing program.

After you have found a function module, determine whether it has been released (attributes of the function module). Only with released function modules do you have the right to support and upward compatibility. It is recommended that you only use released function modules.

You can use the documentation of the function module to determine its functionality and obtain further information.

Examination of a Function Module

Figure 137: Function Module Interface

The interface of the function module consists of the import, export, changing parameters, and exceptions.

As with subroutines, you can distinguish between pass by value and pass by reference.

Non-optional parameters must be supplied when the function module is called. To find out which additional parameters are to be passed and when, refer to the function module documentation and the documentation for interface parameters.

© Copyright . All rights reserved.

Prohibida su reproduccion

Documentation and Test Environment

Figure 138: Documentation and Test Environment

Features of documentation:

- Short text
- Functional description
- Example of use
- Notes
- Parameter description
- Exception description

Function modules that are intended and released for use in other programs should be documented. A distinction is made between function module documentation, which describes the function of the module, and parameter documentation, which tells you how individual parameters or exceptions are used.

You can access the function module documentation by choosing the *FModule Documentation* pushbutton. Click on the displayed links to go to the parameter documentation (only in display mode). You can also access parameter documentation by choosing the pushbutton for the respective parameter in the *Long Text* column. The “green light” indicates that documentation has been created for the parameter.

If you are unsure whether the function module does precisely what you want it to do, you can test it in the test environment prior to installing it in your program. An input template allows you to specify values for the import and changing parameters. To check values during runtime, you can also switch to debugging mode. The values in the export and changing parameters are listed after execution.

If the function module triggers an exception due to an error, the exception is displayed, where applicable, with its message text.

© Copyright . All rights reserved.

Prohibida su reproduccion

The test framework for function modules also displays the runtime. These values are subject to the same restrictions as the runtime analysis. Therefore, repeat the tests several times using the same data.

Store test data in a test data directory and create test sequences.

Call of Function Modules

Figure 139: Calling a Function Module

Function modules are called using the CALL FUNCTION statement. The name of the function module follows in capital letters enclosed in single quotation marks.

In the EXPORTING block, the system passes values to the import parameters of the function module.

In the IMPORTING block, actual parameters are assigned to the export parameters. You can use them to access the results of the call.

From the perspective of the calling program, values are exported to the import parameters of the function module and values are imported for the receiving variables assigned to the export parameters of the function module.

In the call syntax, you need to specify the name of the interface parameter (formal parameter) on the left side of the parameter assignment and the data object, or the value for the calling program (actual parameter) on the right side of the parameter assignment.

© Copyright . All rights reserved.

Prohibida su reproduccion

Source Code Creation for a Function Module Call

Figure 140: Creating Source Code for a Function Module Call

To avoid spelling errors, use the *Pattern* pushbutton to generate the call when you implement a call in the source code. Alternatively, you can display the function group in the navigation area and drag-and-drop the function module into the ABAP Editor area to generate the call in the source code.

This generates the entire call syntax, which lists all interface parameters, as well as exceptions. Optional information is commented out in the call.

Afterwards, you need to fill in the actual parameters and, when necessary, implement exception handling.

© Copyright . All rights reserved.

Prohibida su reproduccion

Classical Exception Handling

Figure 141: Handling Exceptions

If application errors occur when a function module is processed (for example, a value is not suitable for the calculation), the function module raises a corresponding exception. The exception cancels the processing of the function module and returns the system to the calling program. If the exception is listed (caught) in the EXCEPTIONS block of the call, the specified return code is entered in the sy-subrc system field of the calling program. By checking this field after the call, you can determine if and, where applicable, which exception was raised so you can react accordingly. If no exception is raised by the function module, the sy-subrc of the calling program is set to zero.

You also have the option of setting a return code explicitly for some of the possible exceptions and setting a different return code for all unmentioned exceptions. To do so, list the formal exception OTHERS with the desired return code.

Catch all exceptions in your call and react to them in the program. If a raised exception is not caught, a runtime error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Use a Function Module

Business Example

You need to use a globally defined function module in your program to calculate powers.

Template

BC400_MOS_SUBROUTINE

Solution

BC400_MOS_FUNCTION_MODULE_1

Search for a function module in the Repository and analyze its interface and function.

Task 1

Search for a function module to calculate powers and analyze its interface and function

1. Search in the Repository for a function module that calculates a result from a base number and an exponent.

   Hint:
   If the options described do not yield a suitable function module (*Pattern* pushbutton or Repository Information System), use function module BC400_MOS_POWER.

2. Analyze the function module interface and read the documentation.
3. Answer the following:

   Which parameters do you have to pass with the call?

   _____
   _____
   _____

   How are they typed?

   _____
   _____
   _____

© Copyright . All rights reserved.

Prohibida su reproduccion

What restrictions apply to the value range?

_____

_____

_____

4. Answer the following:

   Which parameter contains the calculation result and how is it typed?

   _____

   _____

   _____

5. Which exceptions can the function module raise? Which errors do these exceptions refer to?
6. Test the function module.

Task 2

Copy your program ZBC400_##_SUBROUTINE or copy the template to the new name ZBC400_##_FUNCTION_MODULE and implement a function module call in the event that the user enters the operator "P".

1. Copy your program or the template.
2. Extend the corresponding IF or CASE structure with a branch that is processed when the corresponding parameter contains the value "P".
3. Call the function module in this processing branch. Transfer the two values entered by the user and receive the result.

   Hint:
   Generate the function module call by using the drag-and-drop function in the navigation area or by choosing the *Pattern* pushbutton.

4. Handle any exceptions that occur and output an explanatory text in the result list.

© Copyright . All rights reserved.

Prohibida su reproduccion

Use a Function Module

Business Example

You need to use a globally defined function module in your program to calculate powers.

Template

BC400_MOS_SUBROUTINE

Solution

BC400_MOS_FUNCTION_MODULE_1

Search for a function module in the Repository and analyze its interface and function.

Task 1

Search for a function module to calculate powers and analyze its interface and function

1. Search in the Repository for a function module that calculates a result from a base number and an exponent.
   a) Perform this step as described in the Participant Handbook. Search in the Repository Information System for function modules that have the term "power" in the short text, for example.

> Hint:
> If the options described do not yield a suitable function module (*Pattern* pushbutton or Repository Information System), use function module BC400_MOS_POWER.

2. Analyze the function module interface and read the documentation.
   a) Perform this step as described in the Participant Handbook.
3. Answer the following:

   Which parameters do you have to pass with the call?

   Since parameter iv_power is optional, it does not have to be specified (the number is then squared).

© Copyright . All rights reserved.

Prohibida su reproduccion

How are they typed?

Parameter iv_base and iv_power are whole numbers.

What restrictions apply to the value range?

The parameter iv_power can have no value greater than 4.

4. Answer the following:

   Which parameter contains the calculation result and how is it typed?

   Parameter ev_result is a packed number with 31 figures, two of which are decimal places.

5. Which exceptions can the function module raise? Which errors do these exceptions refer to?
   a)

Exception	Error
POWER_VALUE_TOO_HIGH	Exponent too high (>4)
RESULT_VALUE_TOO_HIGH	Result too high

6. Test the function module.
   a) Perform this step as described in the Participant Handbook.

Task 2

Copy your program ZBC400_##_SUBROUTINE or copy the template to the new name ZBC400_##_FUNCTION_MODULE and implement a function module call in the event that the user enters the operator "P".

1. Copy your program or the template.
   a) Carry out this step in the usual way.
2. Extend the corresponding IF or CASE structure with a branch that is processed when the corresponding parameter contains the value "P".
   a) See the source code excerpt from the model solution.
3. Call the function module in this processing branch. Transfer the two values entered by the user and receive the result.

> Hint:
> Generate the function module call by using the drag-and-drop function in the navigation area or by choosing the *Pattern* pushbutton.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) See the source code excerpt from the model solution.

4. Handle any exceptions that occur and output an explanatory text in the result list.

   a) See the source code excerpt from the model solution.
   Source Code Excerpt:

```
*&---------------------------------------*
*& Report BC400_MOS_FUNCTION_MODULE_1
*&---------------------------------------*
REPORT bc400_mos_function_module_1.

TYPES gty_result TYPE p LENGTH  16 DECIMALS  2.
PARAMETERS: pa_int1 TYPE i, pa_op TYPE c LENGTH 1,
            pa_int2 TYPE i.
DATA gv_result TYPE gty_result.

IF  ( pa_op  = '+' OR pa_op  = '-' OR
      pa_op  = '*' OR pa_op  = '/' AND
      pa_int2  <>  0 OR pa_op  = '%' OR
      pa_op = 'P').

  CASE pa_op.
    WHEN '+'.
      gv_result  = pa_int1  + pa_int2.
    WHEN '-'.
      gv_result  = pa_int1  - pa_int2.
    WHEN '*'.
      gv_result  = pa_int1  * pa_int2.
    WHEN '/'.
      gv_result  = pa_int1  / pa_int2.
    WHEN '%'.
      PERFORM calc_percentage USING pa_int1 pa_int2
                              CHANGING gv_result.
    WHEN 'P'.
      CALL FUNCTION 'BC400_MOS_POWER'
        EXPORTING
          iv_base    =
          pa_int1
          iv_power              = pa_int2
        IMPORTING
          ev_result    =
          gv_result
        EXCEPTIONS
          power_value_too_high  = 1
          result_value_too_high = 2
          OTHERS                = 3.
      CASE sy-subrc.
        WHEN 0.
*    no action needed
        WHEN 1.
          WRITE 'Max value of power is 4' (mvp).
        WHEN 2.
          WRITE 'Result value too high' (rvh).
        WHEN 3.
          WRITE 'Unknown error' (uer).
      ENDCASE.
  ENDCASE.
  WRITE: 'Result:'(res), gv_result.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
ELSEIF pa_op  = '/' AND pa_int2  =  0.
  WRITE: 'No division by zero!'(dbz).
ELSE.
  WRITE: 'Invalid operator!'(iop).
ENDIF.

*&---------------------------------------------------*
*& Form calc_percentage
*&---------------------------------------------------*
*  calculate percentage value
*----------------------------------------------------*
*  -->PV_ACT actual value
*  -->PV_MAX maximum value
*  <--CV_RESULT result
*----------------------------------------------------*
FORM calc_percentage USING pv_act TYPE i pv_max TYPE i
                     CHANGING cv_result TYPE gty_result.

* Simple Error Handling
  IF pv_max  =  0.
    cv_result  =  0.
    WRITE 'WARNING!!! Error in percentage calculation'(epc).
  ELSE.
* Calculate result
    cv_result  = pv_act  / pv_max  *  100.
  ENDIF.
ENDFORM. " calc_percentage
```


© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Use function modules of function groups

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 4

Implementing Function Modules

LESSON OVERVIEW

This lesson explains how to create function groups and function modules.

Business Example

You are in charge of implementing calculation functions and want to create a global function module to calculate percentage values.

For this reason, you require the following knowledge:

- An understanding of how to create function groups and function modules

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Create function groups
- Create function modules

Creation of Function Groups

Figure 142: Creating a Function Group

© Copyright . All rights reserved.

Prohibida su reproduccion

Our goal is to create function modules. Because a function module must be contained in a function group, you first can learn how to create a new function group.

How to Create a Function Group

To create the function group, proceed as follows:

1. Choose the object list for a package in the *Object Navigator*, and choose Create -> Function Group from the context menu.
2. Enter the name of the new function group and choose the *Enter* or *Display* pushbutton. Be sure to use the customer namespace.

   A dialog box appears, asking you whether you want to create the new function group.
3. Choose *Yes* to confirm.

   A dialog box with the attributes of the function group appears.
4. Enter a short text and save it by choosing *Save*.
5. Assign the function group to a package and a workbench request in the usual manner in the dialog boxes that follow.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Function Group

Business Example

Your company wants to develop an application in which several functions are used in different programs.

You are in charge of implementing calculation functions. You decide to program the application as a function group with function modules. Implement this project successively in this exercise and the subsequent exercises in this unit.

Template

None

Solution

BC400_MOS (function group)

Create a function group.

1. Create the function group **`ZBC400_##_COMP`**.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Function Group

Business Example

Your company wants to develop an application in which several functions are used in different programs.

You are in charge of implementing calculation functions. You decide to program the application as a function group with function modules. Implement this project successively in this exercise and the subsequent exercises in this unit.

Template

None

Solution

BC400_MOS (function group)

Create a function group.

1. Create the function group **ZBC400_##_COMP**.
   a) Perform this step as described in the Participant Handbook.

© Copyright . All rights reserved.

Prohibida su reproduccion

Creation of Function Modules

Figure 143: Creating a Function Module

To create a function module, proceed as follows:

1. Choose a function group in which to create the new function module.
2. Display the object list for the function group in the navigation area of the *Object Navigator*.
3. In the tree structure, open the context menu for the function group and choose *Create → Function Module*. A dialog box with the attributes of the function module appears.
4. Enter a name and a short text for the function module. Be sure to use the customer namespace for function modules (customer function modules start with "Z_" or "Y_").
5. Choose *Save* to save your entries.
6. When necessary, assign the function module to a package and a workbech request in the dialog boxes that follow.

© Copyright . All rights reserved.

Prohibida su reproduccion

Editing the Source Code

Figure 144: Editing the Source Code

After defining the corresponding IMPORTING and EXPORTING parameters, you can switch to the *Source code* tab page to implement the functions of the function module.

> Hint:
> The comment block directly under the keyword FUNCTION is created by the Function Builder and is changed when changes are made to the parameters.

To Create a Function Module

To create the function module, proceed as follows:

1. Choose a function group in which to create the new function module.
2. Display the object list for the function group in the navigation area of the *Object Navigator*.
3. In the tree structure, open the context menu for the function group and choose *Create → Function Module*.

   A dialog box with the attributes of the function module appears.
4. Enter a name and a short text for the function module. Be sure to use the customer namespace for function modules (customer function modules start with "Z_" or "Y_").
5. Choose *Save* to save your entries.

© Copyright . All rights reserved.

Prohibida su reproduccion

6. When necessary, assign the function module to a package and a workbech request in the dialog boxes that follow.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Function Module

Business Example

You want to create a global function module to calculate percentage values. This is meant to replace the subroutine from a previous exercise.

Template

BC400_MOS_FUNCTION_MODULE_1 (program)

Solution

BC400_MOS_FUNCTION_MODULE_2 (program)

BC400_MOS_PERCENTAGE (function module)

Create and use a function module.

Task 1

In the function group you defined in the last exercise (ZBC400_##_COMP), create a function module named **`Z_BC400_##_COMP_PERCENTAGE`** that can be used to calculate percentage values.

1. Create the function module.
2. Maintain the interface for the function module.
   To do this, create the following parameters and type them with the specified data elements:

Parameter Name	Parameter Type	Data element
iv_max	Import	**`BC400_MAX`**
iv_act	Import	**`BC400_ACT`**
ev_percentage	Export	**`BC400_PERC`**

3. Create an exception that is raised when the values that are transferred result in a division by zero (suggested name: **`DIVISION_BY_ZERO`**).
4. Implement the function module. Use the subroutine in the template for orientation. If an error occurs, raise the exception instead of outputting an error text.

> Hint:
> To raise the exception, use the **`RAISE <exception name>`** statement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 2

Copy your program ZBC400_##_FUNCTION_MODULE or the template to the new name **`ZBC400_##_FUNCTION_MODULE_2`** and replace the subroutine call with a call for the new function module.

1. Copy the program.
2. Remove the subroutine call and implement a call for the new function module in its place.

> Hint:
> Generate the function module call by using the drag-and-drop function in the navigation area or the *Pattern* pushbutton.

3. Handle the exception and output a corresponding text if an error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Function Module

Business Example

You want to create a global function module to calculate percentage values. This is meant to replace the subroutine from a previous exercise.

Template

BC400_MOS_FUNCTION_MODULE_1 (program)

Solution

BC400_MOS_FUNCTION_MODULE_2 (program)

BC400_MOS_PERCENTAGE (function module)

Create and use a function module.

Task 1

In the function group you defined in the last exercise (ZBC400_##_COMP), create a function module named **`Z_BC400_##_COMP_PERCENTAGE`** that can be used to calculate percentage values.

1. Create the function module.
   a) Perform this step as described in the Participant Handbook.

2. Maintain the interface for the function module.
   To do this, create the following parameters and type them with the specified data elements:

Parameter Name	Parameter Type	Data element
iv_max	Import	**`BC400_MAX`**
iv_act	Import	**`BC400_ACT`**
ev_percentage	Export	**`BC400_PERC`**

   a) Perform this step as described in the Participant Handbook.

3. Create an exception that is raised when the values that are transferred result in a division by zero (suggested name: **`DIVISION_BY_ZERO`**).
   a) Perform this step as described in the Participant Handbook.

4. Implement the function module. Use the subroutine in the template for orientation. If an error occurs, raise the exception instead of outputting an error text.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> To raise the exception, use the **`RAISE <exception name>`** statement.

a) See the source code excerpt from the model solution.

Task 2

Copy your program ZBC400_##_FUNCTION_MODULE or the template to the new name **`ZBC400_##_FUNCTION_MODULE_2`** and replace the subroutine call with a call for the new function module.

1. Copy the program.
   a) Carry out this step in the usual manner.
2. Remove the subroutine call and implement a call for the new function module in its place.

> Hint:
> Generate the function module call by using the drag-and-drop function in the navigation area or the *Pattern* pushbutton.

   a) See the source code excerpt from the model solution.
3. Handle the exception and output a corresponding text if an error occurs.
   a) See the source code excerpt from the model solution.

   Source Code Excerpt

```
*&---------------------------------------*
*& Report    BC400_MOS_FUNCTION_MODULE_2
*&---------------------------------------*
REPORT    bc400_mos_function_module_2.

PARAMETERS: pa_int1    TYPE i,
            pa_op    TYPE c LENGTH  1,
            pa_int2    TYPE i.

DATA gv_result TYPE p LENGTH  16 DECIMALS  2.

IF  ( pa_op  = '+' OR pa_op  = '-' OR
      pa_op  = '*' OR pa_op  = '/' AND
      pa_int2  <>  0 OR pa_op  = '%' OR
      pa_op  = 'P' ).

  CASE pa_op.
    WHEN '+'.
      gv_result  = pa_int1  + pa_int2.
    WHEN '-'.
      gv_result  = pa_int1  - pa_int2.
    WHEN '*'.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
      gv_result  = pa_int1  * pa_int2.
    WHEN '/'.
      gv_result  = pa_int1  / pa_int2.
    WHEN 'P'.
      CALL FUNCTION 'BC400_MOS_POWER'
        EXPORTING
          iv_base    = pa_int1
          iv_power      = pa_int2
        IMPORTING
          ev_result    =  gv_result
        EXCEPTIONS
          power_value_too_high  = 1
          result_value_too_high = 2
          OTHERS                = 3.
      CASE sy-subrc.
        WHEN  0.
*    no action needed
        WHEN  1.
          WRITE 'Max value of power is  4'(mvp).
        WHEN  2.
          WRITE 'Result value too high'(rvh).
        WHEN  3.
          WRITE 'Unknown error'(uer).
      ENDCASE.
    WHEN '%'.
      CALL FUNCTION 'BC400_MOS_PERCENTAGE'
        EXPORTING
          iv_act           = pa_int1
          iv_max           = pa_int2
        IMPORTING
          ev_percentage    = gv_result
        EXCEPTIONS
          division_by_zero = 1
          OTHERS           = 2.
      IF sy-subrc <> 0.
        WRITE 'Error in Function Module' (efm).
      ENDIF.
  ENDCASE.
  WRITE: 'Result:'(res), gv_result.
ELSEIF    pa_op  = '/'    AND    pa_int2  =  0.
  WRITE: 'No division by zero!'(dbz).
ELSE.
  WRITE: 'Invalid operator!'(iop).
ENDIF.


* Source code extract from the function module:

FUNCTION BC400_MOS_PERCENTAGE.
*&----------------------------------------------------*
*  IMPORTING
*        REFERENCE(IV_ACT) TYPE  BC400_ACT
*        REFERENCE(IV_MAX) TYPE  BC400_MAX
*  EXPORTING
*        REFERENCE(EV_PERCENTAGE) TYPE  BC400_PERC
*  EXCEPTIONS
*         DIVISION_BY_ZERO
*&----------------------------------------------------*
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
* Error handling
  IF iv_max = 0.
  ev_percentage = 0.
  RAISE division_by_zero.
ELSE.

* Calculate result
  ev_percentage = iv_act / iv_max * 100.
ENDIF.
ENDFUNCTION.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Create function groups
- Create function modules

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 5

Modularizing Using BAPIs

LESSON OVERVIEW

This lesson introduces the Business Application Programming Interface (BAPI).

Business Example

You are working in the SAP system and want to access data. This can be done with the help of a BAPI. For this reason, you require the following knowledge:

- An understanding of BAPIs

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Modularize using BAPIs

Business Application Programming Interface (BAPI)

Figure 145: Business Objects and BAPIs

The Business Object Repository (BOR) in the SAP system contains business objects. A business object is a class and corresponds to an SAP table or a table hierarchy. A business object has

© Copyright . All rights reserved.

Prohibida su reproduccion

BAPIs as methods. You can call these BAPIs to access the corresponding table(s). A BAPI is a means of accessing data in the SAP system.

BAPIs usually exist for the following functions of a business object:

- Creating an object
- Retrieving object attributes
- Changing object attributes
- Listing objects

The functions of a BAPI are encapsulated in a function module that can be called remotely. BAPIs can, therefore, be called by ABAP programs of the same SAP system, as well as by external programs.

Use of BAPIs

Figure 146: Using BAPIs

The figure shows how BAPIs can be used.

There are standard methods in the form of BAPIs with standardized names.

Standard BAPIs:

- GetList

  All GetList BAPIs returns a list of available objects that meet the specified selection criteria.
- GetDetail

© Copyright . All rights reserved.

Prohibida su reproduccion

All GetDetail BAPIs returns detailed information (attributes) for an object (the complete key must be specified).

- Create, Change, Delete, Cancel

  All Create, Change, Delete, Cancel BAPIs allows you to create, change, and delete objects.

- AddItem, RemoveItem

  All AddItem, RemoveItem BAPIs adds and removes subobjects (for example, an item for an order).

BAPI Explorer

Figure 147: BAPI Explorer

You can use the BAPI Explorer to list business objects as well as the corresponding BAPIs with reference to the application. To call the BAPI Explorer, use the following path in the *SAP Easy Access* menu: *Tools → Business Framework → BAPI Explorer* or transaction BAPI.

After you have located the required business object or BAPI, you can display the relevant details on the right side of the screen by selecting BAPI. You can navigate to its display using the Function Builder by double-clicking the displayed function module.

© Copyright . All rights reserved.

Prohibida su reproduccion

Properties of a BAPI Function Module

Figure 148: Properties of a BAPI Function Module

Function modules for BAPIs must fulfill the following technical requirements:

- Naming convention: BAPI <business object name> <method name>
- Remote-enabled
- No user dialogs or messages
- Name prefix for BAPI structures: BAPI

  Interface parameters are typed with components of structures from the ABAP Dictionary created for this BAPI.
- No changing parameters until Release 4.6
- Raise no exceptions

  Errors are reported to the user through the special export parameter RETURN.

© Copyright . All rights reserved.

Prohibida su reproduccion

BAPI Call in an ABAP Program

Figure 149: BAPI Call in an ABAP Program

To use a BAPI within the same SAP system, you need to call the relevant function module directly. Note the restrictions mentioned in the Properties of a BAPI Function Module section.

Hint:
BAPI interfaces are created according to external call requirements; that is, a call from an external system. Amounts are, therefore, expected in an external format with 4 or 9 decimal places. During the call, the amounts must be transferred to the interface in an appropriately converted format, even if the corresponding currency has no decimal places.

For this conversion or re-conversion, you can use function modules from the BACV function group (package SBF_BAPI).

LESSON SUMMARY
You should now be able to:

- Modularize using BAPIs

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 6

Modularizing Using Global Classes

LESSON OVERVIEW

This lesson introduces you to another technique for cross-program modularization: call of methods of global classes. As with function groups and function modules, we will first look for, analyze, test, and use existing classes and methods. In addition, we will create a global class with a simple method.

This lesson is not intended to provide a comprehensive introduction to object-oriented programming with ABAP. Familiarize yourself with the basic terms of object-oriented programming so that you can use the functions of existing global classes in your own programs.

Business Example

In your program, you need a function that may already exist in the form of a method of a global class. You want to use this method in your program.

If the function does not already exist, or does not meet your requirements, you need to create your own global class and method. For this reason, you require the following knowledge:

- An understanding of the basic terms of object-oriented programming
- An understanding of how to acquire information about the function and use of global classes and their methods
- An understanding of the call of global classes in your programs
- An understanding of how to create global classes
- An understanding of how to create and implement simple methods in global classes

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Describe object-oriented programming
- Use methods of global classes
- Handle instances

© Copyright . All rights reserved.

Prohibida su reproduccion

Principles of Object-Oriented Programming

Figure 150: Representation of a Global Class

Before using global classes and methods, you need to know some basic terms used in object-oriented programming.

You have read about function groups that make reusable units available in the form of function modules.

Object-oriented enhancements to ABAP have introduced the use of global classes, which make functions available in the form of methods. Similar to function modules, methods have an interface known as a signature that comprises import, export, changing parameters, and exceptions.

Classes have other components, in addition to methods. For example, classes contain global data objects known as attributes. In the same way that all function modules can access the global data objects in a function group, all the methods of a class can access the attributes of that class.

© Copyright . All rights reserved.

Prohibida su reproduccion

Example of Access Options for a Global Class

Figure 151: Example of Access Options for a Global Class

Global data objects within the function group are not visible outside the function group. This is referred to as encapsulation of data in the function group.

Attributes are normally encapsulated in the class, and can therefore only be read or changed using methods of the same class.

In contrast to function modules, classes allow you to make specific attributes visible to users of the class. A distinction is, therefore made between public and private attributes.

You can apply the public and private distinction not only to attributes, but also to methods. Whereas all function modules can be called from outside the function group, only public methods are available outside the class. Private methods can only be called by methods of the same class, and are similar in this respect to subroutines within a function group.

© Copyright . All rights reserved.

Prohibida su reproduccion

Multiple Instantiation of Classes

Figure 152: Classes and Objects

The major difference between global classes and function groups is that a function group with global data objects can only be loaded to the program context once for each main program, whereas a global class can be loaded as many times as you like. This is known as multiple instantiation of the class. Values in the global data objects of a function group are the same for all function module calls.

By contrast, a class can have several instances (also known as objects), each of which is stored separately in the program context of the main program. Each instance can, therefore, have different attribute values. Consequently, a method sees different values in the attributes, depending on the instance for which it was called.

It is necessary to generate a class instance explicitly in the ABAP source code, using the CREATE OBJECT statement.

Instance Components and Static Components

Attributes that can have a different value for each instance are known as instance attributes, to distinguish them from static attributes or class attributes. Static attributes exist only once for each program context, regardless of how many class instances are generated when a program runs. When instance methods access a static attribute, all instances read the same value. An instance method can change the value. All other instances then read the new value.

Regarding methods, also a distinction is made between static methods and instance methods. The main difference is that an instance method can only be called when a class instance is created beforehand. By comparison, static methods can be called without any previous instantiation of the class.

© Copyright . All rights reserved.

Prohibida su reproduccion

Hint:
This course focuses primarily on static methods.

Attributes of Global Classes

Figure 153: Definition of Attributes

As with other Repository objects, the display and processing of global classes is incorporated in the Object Navigator. You have several options for opening a global class. In the navigation area, for example, choose *Class/Interface* and enter the name of the class in the input field directly under the name. You can also display the object list for a package first and then double-click the name of the required class in the subnode *Class Library → Classes*.

Choose the *Attributes* tab page to open the list of attribute definitions in the class.

If you only want to use the class, only those attributes that are public are of interest. You can address these attributes directly outside the class.

© Copyright . All rights reserved.

Prohibida su reproduccion

Methods and Their Signature

Figure 154: Definition of Methods

Choose the *Methods* tab page to open the list of all method definitions in the class. With methods, only entries that are identified as public are of interest to the user. Only these public methods can be called externally; all other methods are used for modularization within the global class.

A method identified as a static method can be called directly, without the need to first generate an instance of the class. With instance methods, you need to create an instance and then call the method specifically for that instance. Under normal circumstances, the instance method then accesses instance attributes that contain different values for the various instances.

Method Parameters

Figure 155: Method Parameters

© Copyright . All rights reserved.

Prohibida su reproduccion

Parameters for a method are displayed in a separate window. You access the window from the method list by placing the cursor on the required method and clicking the *Parameter* pushbutton. In contrast to function modules, there is no separate list for export, import, and changing parameters. Instead, the parameter type is entered in the *Type* column.

Method Exceptions

Figure 156: Method Exceptions

You need to navigate to a separate display window to display exceptions for a method. You access it from the method list by placing the cursor on the required method and clicking the *Exceptions* pushbutton.

Basis Release 6.10 introduced class-based exceptions in ABAP. Handling these exceptions is very different from handling the classic exceptions of function modules. A method can either raise only new class-based exceptions or only classic exceptions. It is not possible to mix both concepts. A checkbox shows you which exception concept is used by the respective method.

While the classic exception concept allows you to choose any exception identifier, with the new exception concept, you have to specify the names of special classes, the exception classes.

Hint:
The class-based exception concept is also available for function modules.

© Copyright . All rights reserved.

Prohibida su reproduccion

Global Class Documentation and Testing

Figure 157: The Class Builder Testing Environment

With global class documentation, a distinction is made between the documentation of the class as a whole and the documentation of individual components.

You can access the class documentation by choosing the *Class Documentation* pushbutton. To consult the documentation for an individual method or attribute, navigate to the corresponding list, select the required component with the cursor, and choose the *Documentation* pushbutton.

You can test active global classes.

Temporary storage is allocated for the components of the class. For static components, this is allocated immediately, whereas for instance components the system only allocates storage after you have chosen the *Instance* pushbutton.

The system only lists the public components. You can test methods using the *Execute Method* pushbutton.

You do not need to generate an instance to test a static method. You can execute the static method immediately.

Any importing parameters that exist appear on the screen after you choose the *Execute* pushbutton.

After supplying the parameters with values, you can test the method. The system then displays the result of the exporting parameter.

© Copyright . All rights reserved.

Prohibida su reproduccion

Call of Static Methods

Figure 158: Method Calls Using Drag-and-Drop

You use the CALL METHOD statement to call a method, and you then specify the method. You need to distinguish between an instance method and a static method. With static methods, this specification comprises the name of the class and the method, separated by the static component selector “=>” (double-headed arrow).

The parameters are then passed to the function module call in both an EXPORTING block and an IMPORTING block.

As with function modules, you have several options for generating the method call. SAP recommends using these options to avoid typing errors.

In the navigation area, select a method name and drag it to the editing area while holding down the left mouse button. When the statement is generated this way, you only need to add the actual parameters.

Alternatively, you can choose the *Pattern* pushbutton. Locate the *Call Method* option under *ABAP Objects Pattern*. Enter the class under *Class/Interface* and enter the name of the method under *Method.* There is no need to fill the *Instance* field with static methods.

Exception Handling

Methods can raise either classic exceptions or class-based exceptions. Classic exceptions are handled in the same manner that you encountered with function modules.

Classic Exception Handling

© Copyright . All rights reserved.

Prohibida su reproduccion

```
CALL METHOD cl_bc400_compute=>get_power
       EXPORTING
           iv_base   = pa_int1
           iv_power  = pa_int2
       IMPORTING
           ev_result = gv_result
      EXCEPTIONS
         POWER_VALUE_TOO_HIGH  = 1
         RESULT_VALUE_TOO_HIGH = 2.

  CASE sy-subrc.
    WHEN 0.
      WRITE gv_result.
    WHEN 1.
      WRITE 'Max Value for Power is 4'.
    WHEN 2.
      WRITE 'Result value was too high'.
  ENDCASE.
```

In the EXCEPTIONS block of the method call, the you assign a return code to the exception. When the method terminates with this classic exception, this return code is placed in the sy-subrc system field. By querying sy-subrc, the calling program can react to the exception.

In comparison, handling class-based exceptions is much more complex.

Class-Based Exception Handling

```
TRY.
    CALL METHOD cl_bc400_compute=>get_power
         EXPORTING
             iv_base   = pa_int1
             iv_power  = pa_int2
         IMPORTING
             ev_result = gv_result.
    WRITE gv_result.
  CATCH cx_bc400_power_too_high .
    WRITE 'Max Value for Power is 4'.
  CATCH cx_bc400_result_too_high .
    WRITE 'Result value was too high'.
ENDTRY.
```

The call must be bound between the TRY. and ENDTRY. statements. The exception is then handled in a processing block that begins with the CATCH <exception class>. statement, where <exception class> represents the exception class that is to be handled. If the corresponding exception is raised within the TRY-ENDTRY block, processing is terminated and the program flow branches directly to the relevant CATCH block.

© Copyright . All rights reserved.

Prohibida su reproduccion

Exception Handling with Exception Classes

Figure 159: Exception Handling with Exception Classes

If you generate method calls by using drag–and-drop function or by using the Pattern pushbutton, the system generates the statements for handling the class-based exceptions as well. You then need to remove the comments and implement the CATCH-blocks.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Use a Global Static Method

Business Example

You want to use a static method of a global class to calculate powers. This is meant to replace the function module call from a previous exercise.

Template

BC400_MOS_FUNCTION_MODULE_2 (program)

Solution

BC400_MOS_GLOBAL_CLASS_1 (program)

Call a static method of a global class.

Task 1

Open the CL_BC400_COMPUTE global class and familiarize yourself with the interface and the scope of functions of the method(s) it contains.

1. Analyze the interface of the get_power method and read the documentation about the class.
2. Test the get_power static method.

Task 2

Copy your program ZBC400_##_FUNCTION_MODULE_2 or the copy template to the new name **`ZBC400_##_GLOBAL_CLASS_1`** and replace the function module call for the power calculation with a call for the get_power method.

1. Copy your program or the template.
2. Remove the call of the BC400_MOS_POWER function module and implement a call of the cl_bc400_compute=>get_power method in the same place.

> Hint:
> Generate the method call by using the drag-and-drop function in the navigation area or the *Pattern* pushbutton.

3. Handle exceptions and output a corresponding text if an error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Use a Global Static Method

Business Example

You want to use a static method of a global class to calculate powers. This is meant to replace the function module call from a previous exercise.

Template

BC400_MOS_FUNCTION_MODULE_2 (program)

Solution

BC400_MOS_GLOBAL_CLASS_1 (program)

Call a static method of a global class.

Task 1

Open the CL_BC400_COMPUTE global class and familiarize yourself with the interface and the scope of functions of the method(s) it contains.

1. Analyze the interface of the get_power method and read the documentation about the class.
   a) Perform this step as described in the course materials.
2. Test the get_power static method.
   a) Perform this step as described in the course materials.

Task 2

Copy your program ZBC400_##_FUNCTION_MODULE_2 or the copy template to the new name **ZBC400_##_GLOBAL_CLASS_1** and replace the function module call for the power calculation with a call for the get_power method.

1. Copy your program or the template.
   a) Carry out this step in the usual way.
2. Remove the call of the BC400_MOS_POWER function module and implement a call of the cl_bc400_compute=>get_power method in the same place.

> Hint:
> Generate the method call by using the drag-and-drop function in the navigation area or the *Pattern* pushbutton.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) See the source code extract from the model solution.

3. Handle exceptions and output a corresponding text if an error occurs.

a) See the source code extract from the model solution.

Source Code Extract

```
REPORT BC400_MOS_GLOBAL_CLASS_2.

PARAMETERS: pa_int1 TYPE i,
            pa_op TYPE c LENGTH  1,
            pa_int2 TYPE i.

DATA gv_result TYPE p LENGTH  16 DECIMALS  2.

IF  ( pa_op  = '+' OR pa_op  = '-' OR
      pa_op  = '*' OR pa_op  = '/' AND
      pa_int2  <>  0 OR pa_op  = '%' OR
      pa_op  = 'P' ).
  CASE pa_op.
    WHEN '+'.
      gv_result  = pa_int1  + pa_int2.
    WHEN '-'.
      gv_result  = pa_int1  - pa_int2.
    WHEN '*'.
      gv_result  = pa_int1  * pa_int2.
    WHEN '/'.
      gv_result = pa_int1  / pa_int2.
    WHEN 'P'.
      TRY.
          CALL METHOD cl_bc400_compute=>get_power
            EXPORTING
              iv_base   = pa_int1
              iv_power  = pa_int2
            IMPORTING
              ev_result = gv_result.
        CATCH cx_bc400_power_too_high  .
          WRITE 'Max value of Power is  4'(mvp).
        CATCH cx_bc400_result_too_high  .
          WRITE 'Result value was too high'(rvh).
      ENDTRY.
    WHEN '%'.
      CALL METHOD cl_bc400_mos_compute=>get_percentage
        EXPORTING
          iv_act        = pa_int1
          iv_max        = pa_int2
        IMPORTING
          ev_percentage = gv_result.
  ENDCASE.
  WRITE: 'Result:'(res), gv_result.
ELSEIF pa_op  = '/' AND pa_int2  =  0.
  WRITE: 'No division by zero!'(dbz).
ELSE.
  WRITE: 'Invalid operator!'(iop).
ENDIF.
```

Source code extract for the method:

```
METHOD GET_PERCENTAGE.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
*Simple error handling
IF iv_max = 0.
  ev_percentage = 0.
ELSE.
* Calculate result
  ev_percentage = iv_act / iv_max * 100.
ENDIF.

ENDMETHOD.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Instance Creation and Call of Instance Methods

Figure 160: Creating Objects and Calling Methods

Because instances do not have names, you need to define reference variables to create and address instances of classes. Reference variables are pointers that you can direct to corresponding instances. Each of the reference variables has a name that can be used to address the corresponding instance.

Reference variables are defined using DATA reference_name TYPE REF TO class_name.

When the program starts, a reference variable still has its initial value (it does not point to an instance). After it has been used to create an instance, the reference variable no longer has the initial value and points to that instance.

You can use the CREATE OBJECT reference_name statement to create an instance of the class that was specified in the definition of reference variables. Afterwards, the reference variable points to the newly created instance.

When you use CREATE OBJECT, you may need to supply the import parameters of the special method CONSTRUCTOR with data. This special method is automatically executed immediately after the creation of the instance. With its import parameters, this special method maintains the corresponding attributes of the new instance.

You call methods of an instance using the CALL METHOD reference_name->method_name statement.

In contrast to calling a function module, the method name alone is not sufficient in this case. You need to specify the relevant instance as well, because the program can have several instances of that class.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Describe object-oriented programming
- Use methods of global classes
- Handle instances

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 7

Implementing Simple Global Classes and Static Methods

LESSON OVERVIEW

This lesson explains how to create a global class with a method to encapsulate a function for reuse. This lesson limits itself to creating a static method.

Business Example

Your company wants to develop an application in which several functions are used in various programs. You are in charge of implementing calculation functions. You decide to program it as a global class with static methods. For this reason, you require the following knowledge:

- An understanding of how to create global classes and static methods

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Create simple global classes

© Copyright . All rights reserved.

Prohibida su reproduccion

Creation of Global Classes

Figure 161: Creating Global Classes in the Object Navigator

To create a global class, open the context menu for your package in the navigation area and choose *Create → Class Library → Class*. In the dialog box that appears, enter the name of the class, a short description of the class, and several other properties. After saving your entries, assign the class to a workbench request in the usual way.

Alternatively, you can choose *Class/Interface* from the dropdown list in the navigation area, enter the name of the new class in the field below it, and choose the *Display* pushbutton. Confirm that you want to create this new class.

© Copyright . All rights reserved.

Prohibida su reproduccion

Creation of Static Methods

Figure 162: Calling the Source Code Editor

To create a static method, simply enter the method name in the method list. You can maintain the visibility and type of method (static or instance) using the corresponding input help.

To create a parameter for a method, branch to the parameter list for the corresponding method. Switch to change mode and enter the name of the parameter in the list. Use the input help to maintain the parameter type (import, export, and so on) and specify an associated type.

To implement the source code for a method, select the method in the method list and choose the *Source Code* pushbutton. Implement the source code between the METHOD <method_name>. and ENDMETHOD. statements.

© Copyright . All rights reserved.

Prohibida su reproduccion

Display of the Signature in the Source Code Editor

Figure 163: Showing the Signature in the Source Code Editor

While you are editing source code, you can display the method signature method. To do this, choose the *Signature* pushbutton, as shown in the figure.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Simple Global Class

Business Example

Your company wants to develop an application in which several functions are used in various programs.

You are in charge of implementing calculation functions. You decide to program it as a global class with static methods. Implement this project successively in this exercise and subsequent exercises.

Template

None

Solution

CL_BC400_MOS_COMPUTE (global class)

Create a global class.

1. Create the **`ZCL_##_COMPUTE`** global class and activate it.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Simple Global Class

Business Example

Your company wants to develop an application in which several functions are used in various programs.

You are in charge of implementing calculation functions. You decide to program it as a global class with static methods. Implement this project successively in this exercise and subsequent exercises.

Template

None

Solution

CL_BC400_MOS_COMPUTE (global class)

Create a global class.

1. Create the **`ZCL_##_COMPUTE`** global class and activate it.
   a) Run transaction `SE80` to open the *Object Navigator*.
   b) In the navigation area, choose *Class/Interface* from the dropdown list and enter the name of the new class in the field below it.
   c) Choose *(Display)*.
   d) In the modal dialog box that appears, confirm that you want to create a new class.
   e) Enter a short text to explain the function of this class in the *Description* field.
   f) Make no other entries and confirm your description by choosing *(Save)*.
   g) In the dialog box that appears, assign your class to your package and the ABAP Workbench request allocated to you by your instructor.
   h) Activate the global class using *(Activate)*.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Simple Static Method

Business Example

You want to use global static methods to calculate percentages. These are meant to replace the function module calls from a previous exercise. A new percentage calculation method is to be created for this.

Template

BC400_MOS_GLOBAL_CLASS_1 (program)

Solution

BC400_MOS_GLOBAL_CLASS_2 (program)

CL_BC400_MOS_COMPUTE (global class)

Task 1

Create a static method for calculating percentage in the global class that you defined previously. For orientation, use the function module that is called in the template program.

1. Create the **`GET_PERCENTAGE`** static method in your class *ZCL_##_COMPUTE*.
2. Create parameters for the method. Two parameters are to be passed to the method for the calculation and one parameter returned as the result. For orientation, use the function module that is to be replaced.
3. Implement the source code for the *GET_PERCENTAGE* static method. For orientation, use the function module that is to be replaced.

Task 2

Copy your program ZBC400_##_GLOBAL_CLASS_1 or the copy template to the new name **`ZBC400_##_GLOBAL_CLASS_2`** and replace the function module call for the percentage calculation with a call for your new method.

1. Copy the program.
2. Remove the call of the function module and implement a call of the new method in the same location.

> Hint:
> Generate the method call using the drag-and-drop function in the navigation area or the *Pattern* pushbutton.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Simple Static Method

Business Example

You want to use global static methods to calculate percentages. These are meant to replace the function module calls from a previous exercise. A new percentage calculation method is to be created for this.

Template

BC400_MOS_GLOBAL_CLASS_1 (program)

Solution

BC400_MOS_GLOBAL_CLASS_2 (program)

CL_BC400_MOS_COMPUTE (global class)

Task 1

Create a static method for calculating percentage in the global class that you defined previously. For orientation, use the function module that is called in the template program.

1. Create the **`GET_PERCENTAGE`** static method in your class *ZCL_##_COMPUTE*.
   a) Display the global class in change mode.
   b) Switch to the *Methods* tab page and enter the name of the new method in the *Methods* column.
   c) In the *Level* column, select *Static Method* from the input help.
   d) In the *Visibility* column, select *Public* from the input help.
   e) Enter an explanatory text in the *Description* column.
2. Create parameters for the method. Two parameters are to be passed to the method for the calculation and one parameter returned as the result. For orientation, use the function module that is to be replaced.
   a) Select the *GET_PERCENTAGE* method with a single-click of the mouse.
   b) Choose *Parameters*.
   c) Define the following parameters:

Parameter Name	Parameter Type	Associated Type
`iv_act`	*Import*	**`BC400_ACT`**
`iv_max`	*Import*	**`BC400_MAX`**

© Copyright . All rights reserved.

Prohibida su reproduccion

Parameter Name	Parameter Type	Associated Type
ev_percentage	*Export*	**BC400_PERC**

3. Implement the source code for the *GET_PERCENTAGE* static method. For orientation, use the function module that is to be replaced.
   a) Switch to the method list and select the method by clicking on it.
   b) Choose *Source Code*.
   c) Implement the function as specified in the model solution for the GET_PERCENTAGE method.

Task 2

Copy your program ZBC400_##_GLOBAL_CLASS_1 or the copy template to the new name **ZBC400_##_GLOBAL_CLASS_2** and replace the function module call for the percentage calculation with a call for your new method.

1. Copy the program.
   a) Carry out this step in the usual manner.
2. Remove the call of the function module and implement a call of the new method in the same location.

> Hint:
> Generate the method call using the drag-and-drop function in the navigation area or the *Pattern* pushbutton.

   a) See the source code excerpt from the model solution.

Source Code Excerpt:

```
REPORT  BC400_MOS_GLOBAL_CLASS_2.

PARAMETERS:
  pa_int1  TYPE i,
  pa_op    TYPE c LENGTH 1,
  pa_int2  TYPE i.

DATA gv_result TYPE p LENGTH 16 DECIMALS 2.

IF ( pa_op = '+' OR
     pa_op = '-' OR
     pa_op = '*' OR
     pa_op = '/' AND pa_int2 <> 0 OR
     pa_op = '%' OR
     pa_op = 'P' ).

  CASE pa_op.
    WHEN '+'.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
      gv_result = pa_int1 + pa_int2.

    WHEN '-'.
      gv_result = pa_int1 - pa_int2.

    WHEN '*'.
      gv_result = pa_int1 * pa_int2.

    WHEN '/'.
      gv_result = pa_int1 / pa_int2.

    WHEN 'P'.
      TRY.
          CALL METHOD cl_bc400_compute=>get_power
            EXPORTING
              iv_base   = pa_int1
              iv_power  = pa_int2
            IMPORTING
              ev_result = gv_result.
        CATCH cx_bc400_power_too_high .
          WRITE 'Max value of Power is 4'(mvp).
        CATCH cx_bc400_result_too_high .
          WRITE 'Result value was too high'(rvh).
      ENDTRY.

    WHEN '%'.
      CALL METHOD cl_bc400_mos_compute=>get_percentage
        EXPORTING
          iv_act        = pa_int1
          iv_max        = pa_int2
        IMPORTING
          ev_percentage = gv_result.
  ENDCASE.

  WRITE: 'Result:'(res), gv_result.

ELSEIF  pa_op = '/'  AND  pa_int2 = 0.
  WRITE: 'No division by zero!'(dbz).

ELSE.
  WRITE: 'Invalid operator!'(iop).

ENDIF.
```

Source Code Extract for the method:

```
METHOD GET_PERCENTAGE.

* Simple error handling
  IF iv_max = 0.
    ev_percentage = 0.
ELSE.
* Calculate result
    ev_percentage = iv_act / iv_max * 100.
ENDIF.

ENDMETHOD.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Create simple global classes

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 8
Lesson 8

Modularizing Using Local Classes

LESSON OVERVIEW

In this lesson, you are given a brief preview of how local classes enable the use of object-oriented programming techniques for internal program modularization.

Business Example

You want to encapsulate a function in a static method of a local class. For this reason, you require the following knowledge:

- An understanding of the definition, implementation, and use of local classes

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Handle local classes

Working with Local Classes

Figure 164: Defining a Local Class

You have already learned how to define and use global classes. This lesson explains the definition and usage of local classes. Local classes are named as such because they can only be used locally within the program in which they were defined.

© Copyright . All rights reserved.

Prohibida su reproduccion

The main difference between global classes and local classes is the way in which they are defined. While global classes are maintained with a special tool called the Class Builder, local classes are created directly in the source code of the respective main program.

The figure shows how to create a local class in the source code of the program. A distinction is made between the definition part and the implementation part of the class. The description of the attributes and signatures of the methods are located in the definition part, whereas the implementation part contains the source code of the methods.

The CLASS ... ENDCLASS statement is a local definition in the program. Just as the TYPES statement defines local data types, the CLASS ... ENDCLASS statement defines local object types.

The definition part is divided into several sections in which the public and private components are defined (PUBLIC SECTION. and PRIVATE SECTION.).

Syntax for Static Methods

Figure 165: Syntax for Static Methods

The figure shows a schematic representation of how a public static method is defined and implemented. The method is public because the definition is located in the PUBLIC SECTION. section of the class. To define an instance method, as opposed to the static method, the METHODS statement is used instead of CLASS-METHODS.

Methods have a signature (interface parameters and exceptions) that enables them to receive values when they are called and pass values back to the calling program. Methods can have any number of IMPORTING, EXPORTING, and CHANGING parameters. All parameters can be passed as values or as references.

You can define all input parameters (IMPORTING and CHANGING parameters) as optional parameters in the declaration using the OPTIONAL addition. The calling program does not have

© Copyright . All rights reserved.

Prohibida su reproduccion

to pass these parameters when the system calls the object. The DEFAULT addition enables you to specify a start value.

Implementation and Use of a Static Method

Figure 166: Implementing and Using a Static Method

The syntax example shows how to define a local class with a static method (upper-right section) and how to implement the method (lower-right section).

In the section on the left, you can see the static method call from the main program.

Notice that the call is identical to a static method call in a global class.

You can define the local class directly in the source code of the main program or in an INCLUDE program.

> Hint:
> You need to define the local class before the call.
>
> This is different from subroutines, which you normally define below the call statement.

LESSON SUMMARY

You should now be able to:

- Handle local classes

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. What are the uses of modularization?

   Choose the correct answers.

   - [ ] A To improve performance
   - [ ] B To provide a better overview of program layout
   - [ ] C To encapsulate a function that is required many times within a program for multiple use
   - [ ] D To implement the central maintainability of a function within a program
   - [ ] E To make a function available across the system

2. The assignment of actual parameters to formal parameters when calling a subroutine is called _____.

   Choose the correct answer.

   - [ ] A interface assignment
   - [ ] B parameter passing
   - [ ] C interface call
   - [ ] D subroutine call

3. Variables defined in the main program are _____.

   Choose the correct answer.

   - [ ] A global data objects
   - [ ] B local data types
   - [ ] C local and global data types

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following elements does the interface of a function module contain?

   Choose the correct answers.

   - [ ] A Export parameter
   - [ ] B Subroutines
   - [ ] C Changing parameter
   - [ ] D Screen

5. After defining the corresponding IMPORTING and EXPORTING parameters, you can switch to the _____ tab page to implement the function module.

   Choose the correct answer.

   - [ ] A standard
   - [ ] B exceptions
   - [ ] C source code
   - [ ] D attributes

6. Data in the SAP system can be accessed by means of a Business Application Programming Interface (BAPI).

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

7. Which of the following can be used as a visibility option for an attribute ?

   Choose the correct answers.

   - [ ] A Public
   - [ ] B Private
   - [ ] C Static
   - [ ] D Instance

© Copyright . All rights reserved.

Prohibida su reproduccion

8. Methods are called using the CALL CLASS METHOD statement.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

9. When you identify a method as a static method, it can be called directly without the need to generate an instance of the class first.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

10. To create a static method, all you need to do is enter its name in the Constructor list.

    Determine whether this statement is true or false.

    - [ ] True
    - [ ] False

11. Which of the following special tools maintains global classes?

    Choose the correct answer.

    - [ ] A Object Builder
    - [ ] B Class Builder
    - [ ] C Method Builder
    - [ ] D Attribute Builder

12. To define an instance method as opposed to a static method in a local class, the METHODS statement is used instead of CLASS-METHODS.

    Determine whether this statement is true or false.

    - [ ] True
    - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. What are the uses of modularization?

   Choose the correct answers.

   - [ ] A To improve performance
   - [x] B To provide a better overview of program layout
   - [x] C To encapsulate a function that is required many times within a program for multiple use
   - [x] D To implement the central maintainability of a function within a program
   - [x] E To make a function available across the system

2. The assignment of actual parameters to formal parameters when calling a subroutine is called _____.

   Choose the correct answer.

   - [ ] A interface assignment
   - [x] B parameter passing
   - [ ] C interface call
   - [ ] D subroutine call

3. Variables defined in the main program are _____.

   Choose the correct answer.

   - [x] A global data objects
   - [ ] B local data types
   - [ ] C local and global data types

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following elements does the interface of a function module contain?

   Choose the correct answers.

   - [x] A Export parameter
   - [ ] B Subroutines
   - [x] C Changing parameter
   - [ ] D Screen

5. After defining the corresponding IMPORTING and EXPORTING parameters, you can switch to the _____ tab page to implement the function module.

   Choose the correct answer.

   - [ ] A standard
   - [ ] B exceptions
   - [x] C source code
   - [ ] D attributes

6. Data in the SAP system can be accessed by means of a Business Application Programming Interface (BAPI).

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

7. Which of the following can be used as a visibility option for an attribute ?

   Choose the correct answers.

   - [x] A Public
   - [x] B Private
   - [ ] C Static
   - [ ] D Instance

   Correct. You can use Public and Private as a visibility option for an attribute.

© Copyright . All rights reserved.

Prohibida su reproduccion

8. Methods are called using the CALL CLASS METHOD statement.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

9. When you identify a method as a static method, it can be called directly without the need to generate an instance of the class first.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

10. To create a static method, all you need to do is enter its name in the Constructor list.

    Determine whether this statement is true or false.

    - [ ] True
    - [x] False

11. Which of the following special tools maintains global classes?

    Choose the correct answer.

    - [ ] A Object Builder
    - [x] B Class Builder
    - [ ] C Method Builder
    - [ ] D Attribute Builder

12. To define an instance method as opposed to a static method in a local class, the METHODS statement is used instead of CLASS-METHODS.

    Determine whether this statement is true or false.

    - [x] True
    - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 9 Complex Data Objects

Lesson 1

Working with Structures 372
Exercise 22: Work with Structures 379

Lesson 2

Working with Internal Tables 385
Exercise 23: Work with Internal Tables 403

UNIT OBJECTIVES

- Define structured data objects
- Implement basic ABAP statements for structured data objects
- Analyze structured data objects in debugging mode
- Define internal tables
- Implement basic ABAP statements with internal tables
- Analyze internal tables in debugging mode

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 9
Lesson 1

Working with Structures

LESSON OVERVIEW
This lesson covers the definition and structured data objects (structured variables) and their analysis in the ABAP Debugger. You will also learn how to use basic ABAP statements for structured data objects.

Business Example

You are to process your own data structures and search your programs for semantic errors using the ABAP Debugger. For this reason, you need to know:

- How to define structured data objects (structured variables)
- How to use basic ABAP statements for structured data objects
- How to analyze structured data objects in debugging mode

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Define structured data objects
- Implement basic ABAP statements for structured data objects
- Analyze structured data objects in debugging mode

© Copyright . All rights reserved.

Prohibida su reproduccion

Definition of Structured Data Objects

Figure 167: Definition of Structures with Global Types

By defining structured data objects (known as structured variables or simply structures) in ABAP, you can combine values that belong together logically to one data object. Structures can be nested. This means that components can be made up of structures or even internal tables.

You define structure variables in the program with the DATA statement like you do with elementary data objects.

When you set the types, you can refer to the following:

- A structure in the ABAP Dictionary (global structure type)
- A structure type that you declare locally in the program

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
For names of structure types and structure varaiables we add the following definitions to our naming convention:

Purpose	Prefix
Program global* structured type	gty_s_
Program global* structured variables	gs_
Local** structured type	lty_s_
Local** structured variable	ls_

* Note: program global in this case means a type or variable that is globally visible within your program. However, it is locally defined in your program. You also can define types as system global (via entries in the ABAP Data Dictionary). These type definitions are visible (and usuable) within all ABAP programs of your SAP system.

** Note: local in this case means **local to a subroutine**.

Definition of Structures with Local Types

Figure 168: Defining Structures with Local Types

The figure shows the definition of a structure variable using a locally declare structured type.

Use the TYPES statement to define local structure types. Here, the components are enclosed as shown.

```
TYPES: BEGIN OF structure_name, ... , END OF structure_name.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Assign a type to each component by using the TYPE addition. For more details, refer to the keyword documentation for the TYPES statement.

Define the data object in the usual way.

If necessary, you can also define a structured data object directly. To do so, all you have to do is replace the leading key word TYPES with DATA.

```
DATA: BEGIN OF structure_name,  ...  , END OF structure_name.
```

Access to Structure Components

Figure 169: Access to Structure Components

Components of a structure are always addressed using a hyphen: structure_name-component_name.

> Hint:
> In principle, the ABAP syntax allows the names of data objects to contain hyphens, such as DATA h-var TYPE c LENGTH 5. However, to avoid confusion with the addressing structure components, you should avoid using hyphens as part of the name.

© Copyright . All rights reserved.

Prohibida su reproduccion

Usage of Structured Data Objects

Figure 170: Copying Structure Components with the Same Name

The statement, MOVE-CORRESPONDING, copies the content of the source structure to the target structure, one component at a time. Here, only those components that are available under the same name in both the source structure and the target structure are considered. All other components of the structures remain unchanged.

The individual assignment are executed like in the MOVE statement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Structured Data Objects in Debugging Mode

Figure 171: Structures in Debugging Mode

Trace the field content of a structure in the ABAP Debugger by entering the structure name in the Variable 1 area, or, copying it from the source code by double-clicking on it. Display the component of the structure by double-clicking on the structure name in the Variable 1 area. In addition, you can configure a display area for structure display on one of the desktops.

To display a structure variable in the classic ABAP Debugger, use the double-click method. By double-clicking on the structure in the source code, you copy the structure to the field view. Double-clicking on the field view takes you to the structure display, where you can read the name, content, and type of the individual components.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Work with Structures

Business Example

You want to display airline and flight data based on a particular date input by your user. Using structures during data retrieval, merge the data into one output structure for your report.

Use the ABAP statement MOVE-CORRESPONDING to assign values between structures and use the ABAP Debugger to trace the data flow and understand the connections between processing blocks.

Template

None

Solution

BC400_STS_STRUCTURE

Task 1

Create a program and define a selection screen for entering an airline, a flight number, and the flight date.

1. Create the executable program **`ZBC400_##_STRUCTURE`** without TOP include. Assign the program to your package and your change request.
2. Define the input options for an airline (suggested name: **`pa_car`**), flight number (suggested name: **`pa_con`**), and flight date (suggested name: **`pa_date`**). When typing the input fields, refer to suitable fields of the *Dictionary* structure type BC400_S_FLIGHT.

Task 2

Create two structures based on the *Dictionary* structure types for data retrieval at a later stage and one structure intended for output defined by a local type.

1. Define two structures in the program with reference to the *Dictionary* structure types BC400_S_CARRIER (suggested name: **`gs_carrier`**) and BC400_S_FLIGHT (suggested name: **`gs_flight`**).
2. Define a third structure that combines all the components of the two previous structures. The reference type should be defined locally in the program (suggested name: **`gty_s_carrierflight`**). Use the TYPES statement for the definition. Use this local structure type to define a structure variable (suggested name: **`gs_carrierflight`**).

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 3

Retrieve the flight data for the flight date along with information about the selected airline. Output this data in a simple list.

1. To retrieve the data, use the GET_CARRIER and GET_FLIGHT methods of class CL_BC400_FLIGHTMODEL. Include the methods in your program by drag-and-drop and supply the signature of the methods with the correct type of data. The input parameters of the selection screen serve as import parameters and the first two structures serve as export parameters here.
2. Merge the received data from the first two structures into your output structure. To do this, use the MOVE-CORRESPONDING statement.
3. Use the WRITE statement to output the content of the output structure in a simple list.
4. (OPTIONAL) Suppress the list output if no flight was found. Instead, output a short message in the error handling of the methods using the WRITE statement.

> Hint:
> You can check with IS INITIAL to determine whether the output structure is filled with data.

Task 4

Execute your program and check the result.

1. Check the filling of all the structures in the *ABAP Debugger*. Test the case when no flight data is found.

© Copyright . All rights reserved.

Prohibida su reproduccion

Work with Structures

Business Example

You want to display airline and flight data based on a particular date input by your user. Using structures during data retrieval, merge the data into one output structure for your report.

Use the ABAP statement MOVE-CORRESPONDING to assign values between structures and use the ABAP Debugger to trace the data flow and understand the connections between processing blocks.

Template

None

Solution

BC400_STS_STRUCTURE

Task 1

Create a program and define a selection screen for entering an airline, a flight number, and the flight date.

1. Create the executable program **`ZBC400_##_STRUCTURE`** without TOP include. Assign the program to your package and your change request.
   a) Carry out this step in the usual way.
2. Define the input options for an airline (suggested name: **`pa_car`**), flight number (suggested name: **`pa_con`**), and flight date (suggested name: **`pa_date`**). When typing the input fields, refer to suitable fields of the *Dictionary* structure type BC400_S_FLIGHT.
   a) See the source code extract from the model solution.

Task 2

Create two structures based on the *Dictionary* structure types for data retrieval at a later stage and one structure intended for output defined by a local type.

1. Define two structures in the program with reference to the *Dictionary* structure types BC400_S_CARRIER (suggested name: **`gs_carrier`**) and BC400_S_FLIGHT (suggested name: **`gs_flight`**).
   a) See the source code extract from the model solution.
2. Define a third structure that combines all the components of the two previous structures. The reference type should be defined locally in the program (suggested name:

© Copyright . All rights reserved.

Prohibida su reproduccion

gty_s_carrierflight). Use the TYPES statement for the definition. Use this local structure type to define a structure variable (suggested name: **gs_carrierflight**).

a) See the source code extract from the model solution.

Task 3

Retrieve the flight data for the flight date along with information about the selected airline. Output this data in a simple list.

1. To retrieve the data, use the GET_CARRIER and GET_FLIGHT methods of class CL_BC400_FLIGHTMODEL. Include the methods in your program by drag-and-drop and supply the signature of the methods with the correct type of data. The input parameters of the selection screen serve as import parameters and the first two structures serve as export parameters here.
   a) See the source code extract from the model solution.
2. Merge the received data from the first two structures into your output structure. To do this, use the MOVE-CORRESPONDING statement.
   a) See the source code extract from the model solution.
3. Use the WRITE statement to output the content of the output structure in a simple list.
   a) See the source code extract from the model solution.
4. (OPTIONAL) Suppress the list output if no flight was found. Instead, output a short message in the error handling of the methods using the WRITE statement.

> Hint:
> You can check with IS INITIAL to determine whether the output structure is filled with data.

   a) See the source code excerpt from the model solution.

Task 4

Execute your program and check the result.

1. Check the filling of all the structures in the *ABAP Debugger*. Test the case when no flight data is found.
   a) Set a session breakpoint in the *ABAP Editor* and execute the program directly using (*Direct Processing*).
   Source Code Excerpt:

```
REPORT  bc400_sts_structure.

PARAMETERS: pa_car  TYPE bc400_s_flight-carrid,
            pa_con  TYPE bc400_s_flight-connid,
            pa_date TYPE bc400_s_flight-fldate.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
DATA: gs_carrier TYPE bc400_s_carrier,
      gs_flight  TYPE bc400_s_flight.

TYPES: BEGIN OF gty_s_carrierflight,
          carrid     TYPE bc400_s_flight-carrid,
          connid     TYPE bc400_s_flight-connid,
          fldate     TYPE bc400_s_flight-fldate,
          seatsmax   TYPE bc400_s_flight-seatsmax,
          seatsocc   TYPE bc400_s_flight-seatsocc,
          percentage TYPE bc400_s_flight-percentage,
          carrname   TYPE bc400_s_carrier-carrname,
          currcode   TYPE bc400_s_carrier-currcode,
          url        TYPE bc400_s_carrier-url,
       END OF gty_s_carrierflight.

DATA: gs_carrierflight TYPE gty_s_carrierflight.

* Get data
TRY.
    CALL METHOD cl_bc400_flightmodel=>get_flight
      EXPORTING
        iv_carrid = pa_car
        iv_connid = pa_con
        iv_fldate = pa_date
      IMPORTING
        es_flight = gs_flight.

    CALL METHOD cl_bc400_flightmodel=>get_carrier
      EXPORTING
        iv_carrid  = pa_car
      IMPORTING
        es_carrier = gs_carrier.

  CATCH cx_bc400_no_data .
    WRITE: 'No data found!'(ndf).
  CATCH cx_bc400_no_auth .
    WRITE: 'No authority for this carrier!'(nau).
ENDTRY.

* Fill gs_carrierflight
MOVE-CORRESPONDING gs_carrier TO gs_carrierflight.
MOVE-CORRESPONDING gs_flight  TO gs_carrierflight.

IF gs_carrierflight IS NOT INITIAL.
  WRITE: / gs_carrierflight-carrid,
           gs_carrierflight-connid,
           gs_carrierflight-fldate,
           gs_carrierflight-carrname,
           gs_carrierflight-currcode,
           gs_carrierflight-seatsmax,
           gs_carrierflight-seatsocc,
           gs_carrierflight-percentage, '%',
           gs_carrierflight-url.
ENDIF.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Define structured data objects
- Implement basic ABAP statements for structured data objects
- Analyze structured data objects in debugging mode

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 9
Lesson 2

Working with Internal Tables

LESSON OVERVIEW

This lesson explains how to define internal tables and use them in ABAP programs. You will also analyze the internal tables at runtime in the ABAP Debugger.

Business Example

You are to use table variables as data stores in your programs and then search for semantic errors in those programs using the ABAP Debugger. For this reason, you need to know:

- How to define internal tables
- How to use basic ABAP statements with internal tables
- How to analyze internal tables in debugging mode

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Define internal tables
- Implement basic ABAP statements with internal tables
- Analyze internal tables in debugging mode

© Copyright . All rights reserved.

Prohibida su reproduccion

Table Types

Figure 172: Internal Tables – Usage Options

An internal table is a data object in which you can keep several identically structured data records at runtime (table variable). The number of data records is restricted only by the capacity of specific system installations.

The ABAP runtime system dynamically manages the size of the internal table. This means that memory management is not a concern for the developer.

The individual data records in an internal table are known as table rows or table entries. The individual components in a row are referred to as fields or columns of the internal table.

Internal tables are a simple way of processing large data sets in a structured manner.

Typical uses of an internal table:

- Storing of data from database tables or sequential files for future processing.
- Preparation of data for screen or printer output (for example, sort).
- Preparation of data for using other services (for example, for method, function module, or subroutine calls).

© Copyright . All rights reserved.

Prohibida su reproduccion

Properties of Internal Tables

Figure 173: Attributes of Internal Tables

Properties of an internal table:

- Line type

  The line type describes the structure of table rows. The developer usually specifies a structure type for that, but any data types are possible.

- Primary key

  The primary key of an internal table consists of key fields in a particular order. The system uses the sequence of the key fields, among other things, for sorting according to keys. Depending on the access type, the primary key can be defined as unique or non-unique. Uniqueness means that a particular combination of key field values can only appear once within the table.

- Table kind

  Table kinds are of three types: standard, sorted, and hashed. Depending on the access type used, you should use the appropriate table kind for the definition to enable high performance access.

© Copyright . All rights reserved.

Prohibida su reproduccion

Attributes and Uses of Table Kinds

	Index Tables		Hashed Table
Table Kind	STANDARD TABLE	SORTED TABLE	HASHED TABLE
Index Access	✓	✓	
Key Access	✓	✓	✓
Key Uniqueness	NON-UNIQUE	UNIQUE/NON-UNIQUE	UNIQUE
Use in	Mainly Index Access	Mainly Key Access	Only Key Access

Figure 174: Attributes and Uses of Table Kinds

The figure illustrates the selection of the appropriate table type.

You can access a table entry in the following ways:

- Index access

  Identify the table entry by its row number.

- Key access

  Identify the table entry by its values in fields.

Depending on the access type, choose the most suitable kinds of table to enable high performance access.

Table kinds:

- Standard tables

  The runtime environment maintains an internal row numbering (index). Both index and key accesses are possible.

  Choose this table kind if you mainly use index access for this internal table.

- Sorted tables

  The runtime environment keeps the data records sorted in by the key fields in ascending order. It also maintains the internal index. Both index and key accesses are possible.

  Choose this table kind if you normally access the internal table with the key or would like the table to be automatically sorted by key.

- Hashed tables

© Copyright . All rights reserved.

Prohibida su reproduccion

The runtime environment manages the data records for fast key access using the hashing procedure. A unique key is required here. With hashed tables, only key accesses are possible.

Choose this table kind if the internal table is very large and you want to access it by key only.

> Hint:
> Only standard tables are used for this course. However, with the exception of a few special cases, the syntax is identical for all three table kinds.

Definition of Internal Tables

Figure 175: Defining Internal Tables with Global Types

The type of an internal table is called a table type. Table types can be defined globally, in the ABAP Dictionary, or locally, within a program.

The figure illustrates a table type declared in the ABAP Dictionary, as well as the program-internal definition of a table variable with reference to the table type.

For detailed information about the declaration of global table types in the ABAP Dictionary, refer to the online documentation. You can access this using (*Application Help*) when displaying the table type.

© Copyright . All rights reserved.

Prohibida su reproduccion

Definition of Internal Tables with Local Types

Figure 176: Defining Internal Tables with Local Types

The figure illustrates a table type declared locally in the program as well as the internal definition of a table variable in the program with reference to the locally declared table type.

When you list key fields in the table type, note that the sequence matters for certain types of processing (such as “sort by key”). For detailed information about declaring local table types, refer to the keyword documentation for the TYPES statement.

Hint:
Instead of defining a table type first you can also define an internal table directly. All you have to do here is use DATA instead of TYPES.

Caution:
A common beginner’s error consists of the following syntax:

```
DATA gt_itab TYPE TABLE OF <Table type>.
```

This would result in the definition of an internal table with rows that are internal tables of the specified table type.

© Copyright . All rights reserved.

Prohibida su reproduccion

Note:
For names of table types and internal tables (data objects) that are defined within your program we add the following definitions to our naming convention:

Purpose	Prefix
Program global* table type	gty_t_
Program global* internal table	gt_
Local** table type	lty_t_
Local** internal table	lt_

* Note: program global in this case means a type or variable that is globally visible within your program. However, it is locally defined in your program. You also can define types as system global (via the ABAP Data Dictionary). These type definitions are visible (and usable) within all ABAP programs of your SAP system.

** Note: local in this case means **local to a subroutine**.

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.
```

Local Structure Type

Internal Table

```
                    STANDARD
DATA  gt_itab  TYPE SORTED   TABLE OF gty_s_type
                    HASHED

                    NON-UNIQUE
               WITH UNIQUE       KEY ...
```

Figure 177: Independent Definition of Internal Tables

In the previous definitions of internal tables, ABAP Dictionary objects, such as a table type (BC400_T_FLIGHTS) or a structure type (BC400_S_FLIGHT), were used. The figure illustrates an independent table definition.

The independent table enables you to implement internal tables with any kind of structure without referring to existing Dictionary types.

© Copyright . All rights reserved.

Prohibida su reproduccion

Possible Definitions of Internal Tables

```
1  DATA gt_itab TYPE  <Table Type> .

                      STANDARD
2  DATA gt_itab TYPE  SORTED    TABLE OF  <Structure Type>
                      HASHED
                 WITH NON-UNIQUE  KEY ...
                      UNIQUE

3  DATA gt_itab TYPE TABLE OF  <Structure Type> .
   (Short form for definition of a standard table with
   non-unique default key)
```

Figure 178: Possible Definitions of Internal Tables

The figure illustrates an overview of possible definitions of internal tables.

The short form of a table definition illustrated in the figure implicitly uses the following default values:

- Table kind – standard (default)
- Uniqueness of key – non-unique (only option for a standard table)
- Table key – default key (all non-numeric table fields are key fields)

Hint:
As the default key usually cannot be used in a meaningful way, you should only use it to define an internal table, if you do not need the key for processing your table.

© Copyright . All rights reserved.

Prohibida su reproduccion

Usage of Internal Tables

Figure 179: Accessing Single Records (Overview)

For processing single records of an internal table, a structure variable that has the same type as the line type of the internal table is required. This structure variable is known as the work area.

The figure illustrates the processing of an internal table using the corresponding work area.

The following statements exist for accessing single records of internal tables:

- APPEND

  This appends the content of a structure to an internal table, and this operation can only be used with standard tables.

- INSERT

  This inserts the content of a structure into an internal table.

- READ TABLE

  This copies the content of a table row to a structure.

- MODIFY TABLE

  This overwrites an internal table row with the content of a structure.

- DELETE

  This deletes a row of an internal table.

- COLLECT

© Copyright . All rights reserved.

Prohibida su reproduccion

This accumulates the content of a structure in row of an internal table that has the same key and in doing so, only non-key fields are added. Hence, this statement can only be used for tables whose non-key fields are all numeric.

For detailed information about the ABAP statements described, refer to the relevant keyword documentation.

Processing Sets of Records (Overview)

Figure 180: Processing Sets of Records (Overview)

The following statements exist for accessing sets of records:

- LOOP AT . . . ENDLOOP

  This places the rows of an internal table into the structure specified in the INTO clause one-by-one. Within the LOOP, the current content of the structure can be output, or you can change and write the current content back to the table.

- DELETE

  This deletes all rows of the internal table that satisfy the logical condition <condition>.

- INSERT LINES OF

  This copies the content of several rows of an internal table to another internal table.

- APPEND LINES OF

  This appends the content of several rows of an internal table to another standard table.

For detailed information about the ABAP statements described here, refer to the relevant keyword documentation.

You will now be introduced to some actual examples of syntax for the most common statements.

© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Inserting a Row

Figure 181: Syntax Example – Inserting a Row

Insert a row into an internal table by writing the data for the required record into the prepared work area and then inserting it into the internal table with an INSERT statement.

With standard tables, this content is appended. With sorted tables, the row is inserted in the right place to keep the table sorted by its key fields and, in hashed tables, it is inserted according to a hash algorithm.

© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Outputting an Internal Table Row-by-Row

Figure 182: Syntax example – Outputting an Internal Table Row-by-Row

Read and edit the content of an internal table row-by-row using a LOOP. Within each cycle, the sy-tabix system field contains the row number of the current table entry.

In the example, the system processes all rows of the internal table consecutively and produces the output using the WRITE statement.

If you want to change the content of the current table row within a loop pass, change the copy of the row in the work area and then write it back to the current table row using the MODIFY statement. The syntax for this is:

```
MODIFY itab FROM wa.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Reading by Index

```
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.
```

sy-tabix

```
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,
         '%'.
```

Figure 183: Syntax Example – Reading by Index

In the LOOP statement, restrict access to specific rows using the FROM-TO addition. In the example, the system only processes the first five rows of the internal table consecutively.

Use the READ TABLE statement to read a single record. Use the INDEX addition to specify the row number of the required record.

Accessing an internal table by index is only possible with index tables, such as standard and sorted.

© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Reading by Key

```
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.
```

sy-tabix

```
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.
```

sy-tabix

Figure 184: Syntax Example – Reading by Key

While in the LOOP, restrict access to specific rows using the WHERE addition. In the example, the system only processes those internal table rows in which the CARRID field has the value LH.

> Hint:
> With regard to the runtime requirement, a sorted table with the CARRID field as the first key field is the most suitable type for this kind of processing.

You can use the READ TABLE syntax in the figure to read a specific row of the internal table. In this case, you have to specify the complete key value of the entry to be read after the WITH TABLE KEY addition. The return code sy-subrc is only set to zero if a corresponding row was found in the internal table.

> Hint:
> With regard to the runtime requirement, a hash table is most suitable for this kind of access when you have a large dataset.

With the WITH TABLE KEY addition, you have to supply all key fields with value. If you want to supply an arbitrary set of fields with values, you must use the WITH KEY addition.

© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Sorting and Deleting Content

```
SORT gt_flightinfo.

SORT gt_flightinfo BY carrid.

SORT gt_flightinfo BY percentage DESCENDING
                      carrid     ASCENDING
```

```
REFRESH gt_flightinfo.

FREE gt_flightinfo.
```

Figure 185: Syntax Example – Sorting and Deleting Content

Standard and hash tables can be sorted in ascending or descending order by the table key or arbitrary columns by using the SORT statement. If neither ASCENDING nor DESCENDING is specified as the sorting order for a sort field, the field is sorted in ascending order by default.

Language-specific sort rules can even be taken into account, if necessary.

Use the optional AS TEXT addition to implement lexicographical sorting. In a German-speaking environment, for example, this means that “ä” comes before “b”, and not behind “z”, as would be the case with non-lexicographical sorting.

When using the optional STABLE addition, the relative order of data records that have identical sort keys will remain intact during sorting.

For more details, refer to the keyword documentation for the SORT statement.

Statements for Deleting the Table Content

You can use the following statements for deleting the table content:

- REFRESH

  This statement deletes the entire content of the internal table. A part of the previously used memory remains available for future insertions.

- CLEAR

  This statement has the same effect as REFRESH for most internal tables (all those that have been previously defined in the course).

  For internal tables with header lines (see later), this statement only initializes the header line but does not delete the table content.

© Copyright . All rights reserved.

Prohibida su reproduccion

- FREE

  This statement deletes the entire content of the internal table and releases the previously used memory. Use the FREE statement for internal tables that have already been evaluated and are no longer required in the further course of the program. By this you make the no longer required memory available again.

Internal Tables in Debugging Mode

Figure 186: Anaylze table in debugging mode

In the ABAP Debugger, trace the content of an internal table by entering the table name in the Variable 1 area and then branching to the table display by double-clicking the table name in the *Variable* column.

Alternatively, choose the *Table* tab page and specify the table name in the *Internal Table* field. Press ENTER for displaying the table content. In addition, you can configure an area for displaying an internal table on one of the desktops.

To display an internal table in the classic ABAP Debugger, the tool provides a key with the label “Table”. As with the new ABAP Debugger, however, you can also navigate to this display by double-clicking the table name.

© Copyright . All rights reserved.

Prohibida su reproduccion

Internal Tables with Header Lines

Figure 187: Comparison – Internal Tables with and Without Header Lines

Using the WITH HEADER LINE addition within the definition of an internal table, you have the option of creating a table with a header line. If an internal table has a header line, it is not necessary to define a work area for this table. The automatically created header line can serve as work area for all accesses to this internal table. This also simplifies the syntax of the statements, as the system always uses the automatically created header line, which no longer has to be specified explicitly.

Reasons for Not Using Tables with Header Lines

You should no longer use tables with header lines because of the following reasons:

- The automatically generated work area will have the same name as the internal table, which hinders program readability.
- The table with a header line is not allowed in nested data objects (structures and internal tables that have internal tables as components) and ABAP Objects (object-oriented extension of ABAP).

Internal tables with header lines are mentioned here because some older programs still use them and you have to work with such programs from time to time. It is for this purpose that further particularities of the header line are listed.

If an internal table with header line is called "itab" you can use itab-comp to address component "comp" of the header line (work area).

You can address the body of the table with itab[].

© Copyright . All rights reserved.

Prohibida su reproduccion

The following example illustrates this situation:

```
DATA itab1 TYPE TABLE OF scarr WITH HEADER LINE.
```

```
DATA itab2 LIKE itab1.
```

```
itab1 = itab2.      "Operation only with header lines
```

```
itab1[] = itab2[].  "Operation with the table bodies
```

The following syntax also defines an internal table with the header line, even though that is not stated specifically:

```
DATA: BEGIN OF itab OCCURS n,
      field1 TYPE ... ,
      field2 TYPE ... ,
... ,
END OF itab.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Work with Internal Tables

Business Example

You want to transfer flight connection data from a database table to an internal table (as temporary storage) and output it in a list.

Template

None

Solution

BC400_ITS_ITAB_LOOP

Search for suitable table types in the ABAP Dictionary, define internal tables based on a global table type, and process the content of internal tables using a loop.

Task 1

Create a program and define an internal table for flight connections within it.

1. Create the executable program **`ZBC400_##_LOOP`** without a TOP include. Assign the program to your package and your change request.
2. To buffer flight connection data in an internal table, it is recommended that you define your internal table in such a way that your line type is compatible with the structure type BC400_S_CONNECTION.

   In the ABAP Dictionary, search for all table types that match this condition.

   > Hint:
   > Display structure BC400_S_CONNECTION in the ABAP Dictionary (the cursor should be placed on the structure name). Use the appropriate pushbutton to list (*where-used list*) for BC400_S_CONNECTION. (Pay attention to the correct selection when triggering the where-used list).

3. Define an internal table (suggested name: **`gt_connections`**) based on one of the global table types that were found.
4. Define a suitable work area for the internal table (suggested name: **`gs_connection`**).

Task 2

Fill your internal table and output the content in a list.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. Get the flight connection data by calling the GET_CONNECTIONS method of class CL_BC400_FLIGHTMODEL. Choose the internal table you defined as the passed parameter.
2. If no flight connection was found, output a short message to the list. Use the error-handling of the method call to do this.
3. Use the LOOP statement to output the buffered data in the internal table in a list.
4. (OPTIONAL) Use the SORT statement to sort the internal table in ascending order according to the departure time before the data is output in the list.

Task 3

Execute your program and check the result.

1. Use the ABAP Debugger to check the output of the internal table in the list.

© Copyright . All rights reserved.

Prohibida su reproduccion

Work with Internal Tables

Business Example

You want to transfer flight connection data from a database table to an internal table (as temporary storage) and output it in a list.

Template

None

Solution

BC400_ITS_ITAB_LOOP

Search for suitable table types in the ABAP Dictionary, define internal tables based on a global table type, and process the content of internal tables using a loop.

Task 1

Create a program and define an internal table for flight connections within it.

1. Create the executable program **`ZBC400_##_LOOP`** without a TOP include. Assign the program to your package and your change request.
   a) Carry out this step in the usual way.
2. To buffer flight connection data in an internal table, it is recommended that you define your internal table in such a way that your line type is compatible with the structure type BC400_S_CONNECTION.

   In the ABAP Dictionary, search for all table types that match this condition.

   > Hint:
   > Display structure BC400_S_CONNECTION in the ABAP Dictionary (the cursor should be placed on the structure name). Use the appropriate pushbutton to list (*where-used list*) for BC400_S_CONNECTION. (Pay attention to the correct selection when triggering the where-used list).

   a) Carry out this step as described.
3. Define an internal table (suggested name: **`gt_connections`**) based on one of the global table types that were found.
   a) See the source code excerpt from the model solution.
4. Define a suitable work area for the internal table (suggested name: **`gs_connection`**).
   a) See the source code excerpt from the model solution.

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 2

Fill your internal table and output the content in a list.

1. Get the flight connection data by calling the GET_CONNECTIONS method of class CL_BC400_FLIGHTMODEL. Choose the internal table you defined as the passed parameter.
   a) See the source code excerpt from the model solution.
2. If no flight connection was found, output a short message to the list. Use the error-handling of the method call to do this.
   a) See the source code excerpt from the model solution.
3. Use the LOOP statement to output the buffered data in the internal table in a list.
   a) See the source code excerpt from the model solution.
4. (OPTIONAL) Use the SORT statement to sort the internal table in ascending order according to the departure time before the data is output in the list.
   a) See the source code excerpt from the model solution.

Task 3

Execute your program and check the result.

1. Use the ABAP Debugger to check the output of the internal table in the list.
   a) Set a session breakpoint in the *ABAP Editor* and execute the program using (*Direct Processing*).
   Source Code Excerpt:

```
REPORT  bc400_its_itab_loop.

DATA: gt_connections TYPE bc400_t_connections,
      gs_connection  TYPE bc400_s_connection.

* Get data
TRY.
    CALL METHOD cl_bc400_flightmodel=>get_connections
      IMPORTING
        et_connections = gt_connections.
  CATCH cx_bc400_no_data .
    WRITE: / 'No data found!'(ndf).
ENDTRY.

SORT gt_connections ASCENDING BY deptime.

* Output
LOOP AT gt_connections INTO gs_connection.
  WRITE: / gs_connection-carrid,
           gs_connection-connid,
           gs_connection-cityfrom,
           gs_connection-airpfrom,
           gs_connection-cityto,
           gs_connection-airpto,
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
            gs_connection-fltime,
            gs_connection-deptime,
            gs_connection-arrtime.

ENDLOOP.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Define internal tables
- Implement basic ABAP statements with internal tables
- Analyze internal tables in debugging mode

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Which of the following statements copies the content of the source structure to the target structure, one component at a time?

   Choose the correct answer.

   - [ ] A COPY-CORRESPONDING
   - [ ] B REPLACE-CORRESPONDING
   - [ ] C MOVE-CORRESPONDING
   - [ ] D REMOVE-CORRESPONDING

2. Which of the following statements is used for defining local structure types?

   Choose the correct answer.

   - [ ] A TYPES
   - [ ] B BEGIN
   - [ ] C END
   - [ ] D START

3. Which of the following specifications are required in the definition of an internal table?

   Choose the correct answers.

   - [ ] A Line type
   - [ ] B Primary key
   - [ ] C Secondary key
   - [ ] D Table kind

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following is used for adding a row into an internal table ?

   Choose the correct answer.

   - [ ] A ADD LINE
   - [ ] B INSERT ROW
   - [ ] C APPEND
   - [ ] D UPDATE TABLE

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Which of the following statements copies the content of the source structure to the target structure, one component at a time?

   Choose the correct answer.

   - [ ] A COPY-CORRESPONDING
   - [ ] B REPLACE-CORRESPONDING
   - [x] C MOVE-CORRESPONDING
   - [ ] D REMOVE-CORRESPONDING

2. Which of the following statements is used for defining local structure types?

   Choose the correct answer.

   - [x] A TYPES
   - [ ] B BEGIN
   - [ ] C END
   - [ ] D START

3. Which of the following specifications are required in the definition of an internal table?

   Choose the correct answers.

   - [x] A Line type
   - [x] B Primary key
   - [ ] C Secondary key
   - [x] D Table kind

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following is used for adding a row into an internal table ?

   Choose the correct answer.

   - [ ] A ADD LINE
   - [ ] B INSERT ROW
   - [x] C APPEND
   - [ ] D UPDATE TABLE

   Correct. You use APPEND to add a row into an internal table.

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 10 Data Modeling and Data Retrieval

Lesson 1

Modeling Data 415
Exercise 24: Analyze Transparent Tables in the Data Dictionary 425

Lesson 2

Reading Single Database Records 431
Exercise 25: Implement Single Record Access 439

Lesson 3

Reading Multiple Database Records 445
Exercise 26: Implement a SELECT Loop 447
Exercise 27: Implement an Array Fetch 455

Lesson 4

Handling Other Aspects of Database Access 461

Lesson 5

Working with Authorization Checks 471
Exercise 28: Implement an Authorization Check 477

UNIT OBJECTIVES

- Explain the purpose and benefits of data models
- Describe the SAP flight data model
- Explain transparent tables
- Read single database records
- Implement a SELECT loop
- Implement an array fetch

© Copyright . All rights reserved.

Prohibida su reproduccion

- Read client-specific data
- Use database indexes
- Explain the SAP table buffer
- Read data from several database tables
- Identify ways to change data in a database table
- Explain the SAP authorization concept
- Implement authorization checks

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 10
Lesson 1

Modeling Data

LESSON OVERVIEW

This lesson gives an overview of data modeling. You need to perform data modeling before application development. You will also learn to understand the description of database tables in the ABAP Dictionary.

Business Example

You want to develop a program that accesses database tables. To do this, you need to know the underlying data model and the structure of database tables. For this reason, you require the following knowledge:

- An understanding of the purpose and benefits of using a data model in application development
- An understanding of the SAP flight data model
- An understanding of the meaning and structure of a transparent table

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the purpose and benefits of data models
- Describe the SAP flight data model
- Explain transparent tables

© Copyright . All rights reserved.

Prohibida su reproduccion

Data Model Overview

Figure 188: Data Modeling

In the development of business applications, relevant aspects of the real world must be represented in data form. A business unit represents an entity. These entities exist in relationships with each other, which are fixed in the underlying data model, also referred to as the Structured Entity Relationship Model (SERM).

The developer uses this data model as the basis for implementing appropriate table definitions (transparent tables), including their relationships with each other in the ABAP Dictionary.

By activating table definitions, corresponding database tables are automatically created in the database. The actual application data is entered into those tables later on.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Flight Data Model

Figure 189: SAP Flight Data Model for ABAP Training Courses

Typical inquiries at the travel agency:

- Relevant airports
- Relevant flight connections
- Relevant flight times
- Relevant information about flights, for example, pricing, utilization of capacity or availability, and so on

The ABAP training courses, online documentation, and ABAP keyword documentation use the same flight data model as an example. Repository objects for the flight data model are in the package SAPBC_DATAMODEL.

Customers of a travel agency who want to travel from one place to another by air want to obtain the following specific information from the travel agency.

Required information:

- Which connection offers the best and most direct flight?
- What are the acceptable flight times on the proposed days of travel?
- Which connections offer the cheapest flights, the fastest connections, and connections with target arrival times?

On the other hand, the travel agency has different requirements. In the data model designed for managing the required data, the data is stored according to the technical criteria in tables in a central database. The amount of the stored data exceeds that needed to respond to the requests of the customer.

© Copyright . All rights reserved.

Prohibida su reproduccion

You, therefore, need to be able to compile the data needed to satisfy the requests of the customer using application programs.

Relational Data Model

Figure 190: Relational Data Model

The flight data model contains the following entities for all business information that is logically connected:

- Cities
- Airports
- Airlines
- Flight routes
- Flights

These entities relate to each other in the following ways:

- Each flight schedule contains one airline, one departure airport, and one destination airport.
- Each bookable flight belongs to one existing flight schedule.
- Each assignment can be set between the relevant cities and the nearby airports.

You can manage all of the necessary data without redundancy by using these relationships. At the same time, the travel agency is able to obtain all of the data requested by the customer.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implementation Using Transparent Tables

Figure 191: Implementation Using Transparent Tables

For each entity fixed in the data model, the developer creates a transparent table in the ABAP Dictionary. This is a platform-independent description of the database table and not the actual database table. However, when the transparent table is activated, the system creates a table of the same name in the database.

A transparent table contains various fields (columns) to store and manage data records in a structured way. Table fields are declared as key fields when the content is to be used for the unique identification of data records within the database table. The key of a table (table key) consists of key fields. This is also called a primary key. Data records in the same table must be unique with regard to primary key values. The key value of a data record thus is its unique ID within the table.

© Copyright . All rights reserved.

Prohibida su reproduccion

Transparent Tables

Figure 192: Technical Structure of Transparent Tables

A transparent table in the ABAP Dictionary is the description of the corresponding database table which contains the actual data used by the application. The fields of the transparent table form columns in the corresponding database table with identical names. Data elements, which you already know as globally defined elementary data types, are normally used for describing individual fields. Data elements refer to domains for their technical properties.

© Copyright . All rights reserved.

Prohibida su reproduccion

Transparent Tables in the ABAP Dictionary

Figure 193: Transparent Tables in the ABAP Dictionary

In addition to the list of fields, transparent tables contain other information that the system requires to create a table of the same name in the database and to describe the complete properties of the table.

Transparent tables contain the following additional information:

- Determination of the key for the database table (key fields)
- Technical properties required by the database to create the database table (expected size and expected frequency of access)
- Settings for technologies that can speed up access to the database table (secondary indexes and buffering)

© Copyright . All rights reserved.

Prohibida su reproduccion

Structures in the ABAP Dictionary

Figure 194: Structures in the ABAP Dictionary

The definition of a transparent table appears to be very similar to the definition of a global structure type. Transparent tables can be used in the same way as structure types in programming; for example, transparent tables can be used for defining a structured data object (structure variable), for typing an interface parameter, or as the line type of a global or a local table type. Only the list of fields is important. Other properties of the transparent table, such as the key definition or the technical properties, are irrelevant when it is being used as a data type.

© Copyright . All rights reserved.

Prohibida su reproduccion

Transparent Table as an ABAP Data Type

Figure 195: Transparent Table as an ABAP Data Type

> Hint:
> In older programs, transparent tables were very freely used as data types. Current recommendations are to use a transparent table directly in connection with access to the database. Above all, transparent tables should not be used in the process of defining user interfaces because this creates unwanted dependency between the definition of database objects or interfaces and the user interface.

In addition to the properties related to the database, there is another difference between transparent tables and structure types.

A transparent table is a list of elementary fields, whereas the components of a structure type can themselves be structured again (nested structures). The components of a structure can even be typed with a table type (deep structures).

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Analyze Transparent Tables in the Data Dictionary

Business Example

You want to familiarize yourself with the flight data model. To do this, you analyze some of the relevant transparent tables in the ABAP Dictionary and display the table content using the *Data Browser*.

Analyze the properties of the transparent tables in the ABAP Dictionary and use the *Data Browser* to acquire an overview of the table content.

Task 1

Examine the properties of the database table SPFLI and analyze the name and data types of the table fields.

1. Display the relevant transparent table.
2. Answer the following:

   The table SPFLI has 16 fields. How many of these make up the key to this table?

   Choose the correct answer.

   - ☐ A 16
   - ☐ B 3
   - ☐ C 1
   - ☐ D No key is defined
3. Answer the following:

   How long are the fields for the points of departure (CITYFROM) and destination (CITYTO)? Why do they have the same technical properties?

   _____

   _____

   _____

Task 2

Search for other database tables to use with the flight data model. Take advantage of the fact that the relevant transparent tables are in the same package.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. In the navigation area of the *Object Navigator*, list all of the transparent tables that belong to the same package as the database table SPFLI.

> Hint:
> Access the object list more quickly and without the name of the package by choosing the *Display Object List* pushbutton from the database table SPFLI display.

2. In which database table is the flight customer data located? What is the name of the field for the name of a flight customer?

_____

_____

_____

Task 3

Find the flight customer number of customer SAP AG. Determine the number of flights for which customer SAP AG has a booking. Find the most immediate flight, determine the point of departure and destination of this flight, and determine whether the flight is fully booked.

1. Use the *Data Browser* to display the content of the SCUSTOM database table for customer SAP AG.
2. What is the flight customer number for SAP AG?

_____

_____

_____

3. Display the content of the SBOOK database table to determine the number of bookings for customer SAP AG and find the most immediate flight.

> Hint:
> The data is regenerated before each training course; so no general statement can be made regarding the most immediate flight at a particular point in time.

4. Which fields can be used to uniquely identify the corresponding flight?

_____

_____

_____

5. Use the content of the database table SPFLI to determine the point of departure and destination of the flight that was found.
6. Is the flight fully booked?

© Copyright . All rights reserved.

Prohibida su reproduccion

Analyze Transparent Tables in the Data Dictionary

Business Example

You want to familiarize yourself with the flight data model. To do this, you analyze some of the relevant transparent tables in the ABAP Dictionary and display the table content using the *Data Browser*.

Analyze the properties of the transparent tables in the ABAP Dictionary and use the *Data Browser* to acquire an overview of the table content.

Task 1

Examine the properties of the database table SPFLI and analyze the name and data types of the table fields.

1. Display the relevant transparent table.
   a) Choose the *Other Object* pushbutton.
   b) Choose the *Dictionary* tab page.
   c) Under *Database Table*, enter the table names and choose (*Display*).
2. Answer the following:

   The table SPFLI has 16 fields. How many of these make up the key to this table?

   Choose the correct answer.

   - [ ] A 16
   - [x] B 3
   - [ ] C 1
   - [ ] D No key is defined

3. Answer the following:

© Copyright . All rights reserved.

Prohibida su reproduccion

How long are the fields for the points of departure (CITYFROM) and destination (CITYTO)? Why do they have the same technical properties?

The fields are 20 characters in length. They have the same technical properties because they refer to the same domain, S_CITY.

Task 2

Search for other database tables to use with the flight data model. Take advantage of the fact that the relevant transparent tables are in the same package.

1. In the navigation area of the *Object Navigator*, list all of the transparent tables that belong to the same package as the database table SPFLI.
   a) You will find the name of the package on the *Properties* tab page for the transparent table (SAPBC_DATAMODEL).
   b) In the navigation area, open the object list for this package in the usual way. You will find the transparent table under the node *Dictionary Objects* → *Database Tables*.

> Hint:
> Access the object list more quickly and without the name of the package by choosing the *Display Object List* pushbutton from the database table SPFLI display.

2. In which database table is the flight customer data located? What is the name of the field for the name of a flight customer?

   The flight customer data is located in the SCUSTOM table; the name of the customer is in the NAME table field.

Task 3

Find the flight customer number of customer SAP AG. Determine the number of flights for which customer SAP AG has a booking. Find the most immediate flight, determine the point of departure and destination of this flight, and determine whether the flight is fully booked.

1. Use the *Data Browser* to display the content of the SCUSTOM database table for customer SAP AG.
   a) Open the SCUSTOM transparent table in the *Object Navigator* as you did before.
   b) Choose the *Content* pushbutton to branch to the *Data Browser*.
   c) Enter the customer name as a selection and choose the *Execute* pushbutton.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. What is the flight customer number for SAP AG?

   The flight customer number for SAP AG is 00000001.

3. Display the content of the SBOOK database table to determine the number of bookings for customer SAP AG and find the most immediate flight.
   a) Proceed in a similar manner to the previous step to display the content of the SBOOK database table. Specify the flight customer number to limit the content.
   b) The number of bookings found (hits) is specified in the title bar of the display.
   c) Sort the display according to the flight date to determine the most immediate booking.

   Hint:
   The data is regenerated before each training course; so no general statement can be made regarding the most immediate flight at a particular point in time.

4. Which fields can be used to uniquely identify the corresponding flight?

   The necessary key fields are Client (MANDT), Airline Code (CARRID), Connection Number (CONNID), and Flight Date (FLDATE).

5. Use the content of the database table SPFLI to determine the point of departure and destination of the flight that was found.
   a) When displaying the data, restrict the airline ID and flight number to the flight that was found previously.
   b) You will find the point of departure and destination in the CITYFROM and CITYTO fields and the departure and arrival airports in the AIRPFROM and AIRPTO fields.

6. Is the flight fully booked?
   a) Display the content of the database table SFLIGHT.
   b) Limit the display to the airline code, flight number, and date of the previously determined flight.
   c) You will find the number of occupied seats and the maximum number of available seats in the SEATSOCC and SEATSMAX fields.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Explain the purpose and benefits of data models
- Describe the SAP flight data model
- Explain transparent tables

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 10
Lesson 2

Reading Single Database Records

LESSON OVERVIEW

This lesson concludes with a note about types of database accesses that initiate changes.

Business Example

You need to evaluate data from database tables. As there are no suitable reuse components for accessing the database tables that you want to read, you implement new ones. For this reason, you require the following knowledge:

- An understanding of the various methods for searching relevant database tables
- An understanding of how to program read access to specific columns and rows within a particular database table

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Read single database records

Data Retrieval with Structured Query Language

Figure 196: Database Access (Architecture)

© Copyright . All rights reserved.

Prohibida su reproduccion

Structured Query Language (SQL) is a language that enables programs to define, change, and have read access to database tables.

Every relational database system has a native SQL that is specific to the database. Therefore, an ABAP program with native SQL statements cannot be used without restrictions in all other SAP systems due to differences in database systems of various SAP systems.

In contrast, Open SQL is an SAP-defined, database-independent SQL standard used in the ABAP language. The system dynamically converts Open SQL statements to native SQL statements of the target database system and, as such, Open SQL statements remain independent of the database. Open SQL allows the ABAP programmer uniform access to data, regardless of the installed database system.

Search for Database Tables

Options for searching required database tables:

- Search by application

  Search within a particular application component, in the application hierarchy.

- Search by program

  Search within a program that accesses the table being searched for:

  - Source code search: search for SELECT statements in the *ABAP* Editor.
  - Function debugging: switch to debugging mode (/h) prior to executing a subfunction and set a breakpoint at the SELECT statement.
  - Screen field information: display a corresponding structure field using F1 + *Technical Information*, navigate to the relevant data element and query Where-Used List in Table Fields.

You can also execute a free search using the Repository Information System.

Before you program direct accesses to database tables, look for reuse components that take care of the read process. The figure gives an overview of those read routines that SAP supplies, which you can use in your program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Reuse Components for Data Retrieval

Figure 197: Reuse Components for Data Retrieval

Types of reuse components that encapsulate database accesses are as follows:

- Logical databases

  Logical databases are data retrieval programs that read data from tables that belong together hierarchically.

- Function modules

  Function modules are procedures stored in the function library of the SAP system with encapsulated functions, such as reading from hierarchically related tables.

- Business Application Programming Interfaces (BAPIs)

  Examples of BAPIs include methods of business objects with read functions.

- Methods of global classes

If there are no useable reuse components available for your data selection, implement the read access using the following methods. Encapsulate the access into a reuse component; in other words, create function modules or methods of global classes.

© Copyright . All rights reserved.

Prohibida su reproduccion

Database Read Access (Overview)

Figure 198: Database Read Access (Overview)

Use the Open SQL statement SELECT to program database read access.

The SELECT statement contains a series of clauses, each of which has a different task as follows:

- The SELECT clause names the fields of the table that are to be read.
- The FROM clause names the source (database table or view) from which data is to be selected.
- The INTO clause determines the target variable into which the selected data is to be placed.
- The WHERE clause specifies the columns of the table that are to be selected.

For information about other clauses, refer to the keyword documentation for the SELECT statement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Read One Row of Data

Figure 199: Reading Single Records

The SELECT SINGLE statement allows you to read a single record from the database table. To ensure a unique access, fill in all key fields of the WHERE clause. The client field is an exception. If nothing is specified, the current client is assumed. A client can only be specified in the SELECT statement in combination with the CLIENT SPECIFIED addition.

Use an asterisk (*) to specify that all selected fields of the table row are to be read. If you only want a specific selection of columns, replace the * with a list of required fields, as shown in the next figure.

Use the INTO clause to specify the target variable to which the data record is to be copied. Target structure has to be left-justified in the same way as the field list.

If the system finds a suitable record, SY-SUBRC returns the value of 0.

© Copyright . All rights reserved.

Prohibida su reproduccion

Corresponding Target Structure for the Field List

Figure 200: Corresponding Target Structure for the Field List

If you only want a certain selection of fields from the table row to be read, specify these as shown in the field list within the SELECT statement (shown in the figure). Then, name a target structure variable in the INTO clause that has the same structure as the field list (at least in the beginning); in other words, it contains the names of the fields that are in the field list and in the same order. Only the corresponding field types have to match. The names of the target structure fields are not taken into account.

Alternatively, list the corresponding target fields in the INTO clause as follows: INTO (field_1, ... , field_n).

© Copyright . All rights reserved.

Prohibida su reproduccion

Fields in the Target Structure with the Same Names as Those in the Field List

Figure 201: Fields in the Target Structure with the Same Names as Those in the Field List

If you want to use a structure variable for receiving the read record, which has fields with the same names as the fields in the target list but has a different structure (additional fields, different order of fields), use the CORRESPONDING FIELDS OF addition. This has the effect that the system only fills the fields of the target area that have the same name as the fields of the database table. Make sure that the field types are also the same for corresponding fields; otherwise (as in the MOVE statement), a complicated conversion starts, and it is possible that only incomplete data (caused by cutoffs) will be transported to the target fields.

The advantages of using INTO CORRESPONDING FIELDS OF are as follows:

- The target structure does not have to be left-justified in the same way as the field list.
- The construction is easy to maintain, because extending the field list or target structure does not require other changes to be made to the program, as long as there is a field in the structure that has the same name (and if possible, also the same type).

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Single Record Access

Business Example

You want to develop a program in which data for a single flight connection is read from the database. You encapsulate the required database access by a single record access in a function module so that you can reuse it.

Template

None

Solution

BC400_DDS (function group)

BC400_DDS_CONNECTION_GET (function module)

Create a function group and a function module and read from the database using a single record access.

Task 1

Create a function group.

1. Create a new function group in your package (suggested name: **`ZBC400_##`**).

Task 2

Create a new function module in the function group. Maintain the interface for the function module in such a way that it imports an airline ID and a flight connection number and returns flight connection data. Raise an exception for the case in which no data is found.

1. Create a new function module in your function group (suggested name: **`Z_BC400_##_CONNECTION_GET`**).
2. Create an export parameter for the detailed data for a flight connection (suggested name: **`ES_CONNECTION`**). For the data type of the parameter, use the Dictionary structure type BC400_S_CONNECTION.
3. Create a separate import parameter for the airline ID and the connection number (suggested names: **`IV_CARRID`** and **`IV_CONNID`**). For the data type of the parameter, choose suitable components from the Dictionary structure type BC400_S_CONNECTION.
4. Create a (classic) exception (suggested name: **`NO_DATA`**).

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 3

In the source code of the function module, implement a single record access to the database table SPFLI. Use the interface parameters of the function module to do this. Raise an exception for the case in which no data can be found for the selection.

1. Implement a single record access (SELECT SINGLE) to the database table SPFLI that fills the export parameter of the function module with values. List all of the table fields that appear as components in the export parameter. Make sure that the order of the fields matches the order of the structure components.
2. Apply a selection condition (WHERE clause) to the SELECT statement. Use both import parameters of the function module to do this.
3. Raise the exception for the function module if the database access returns no data (sy-subrc <> 0).

Task 4

Activate and test your function module.

1. Activate the function module.
2. Test your function module.

> Hint:
> Check also the raising of the exception, if an error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Single Record Access

Business Example

You want to develop a program in which data for a single flight connection is read from the database. You encapsulate the required database access by a single record access in a function module so that you can reuse it.

Template

None

Solution

BC400_DDS (function group)

BC400_DDS_CONNECTION_GET (function module)

Create a function group and a function module and read from the database using a single record access.

Task 1

Create a function group.

1. Create a new function group in your package (suggested name: **`ZBC400_##`**).
   a) In the navigation area, open the context menu for the package and choose *Create → Function Group.*
   b) Enter the name of the function group and a short text and choose *Save*.
   c) Assign the function group to your package and your request in the usual way.

Task 2

Create a new function module in the function group. Maintain the interface for the function module in such a way that it imports an airline ID and a flight connection number and returns flight connection data. Raise an exception for the case in which no data is found.

1. Create a new function module in your function group (suggested name: **`Z_BC400_##_CONNECTION_GET`**).
   a) In the navigation area, open the context menu for your function group and choose *Create → Function Module.*
   b) Enter the name of the function module and a short text and choose *Save*.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. Create an export parameter for the detailed data for a flight connection (suggested name: **ES_CONNECTION**). For the data type of the parameter, use the Dictionary structure type BC400_S_CONNECTION.
   a) Open the *Export* tab page. Enter the name of the parameter and the data type.
3. Create a separate import parameter for the airline ID and the connection number (suggested names: **IV_CARRID** and **IV_CONNID**). For the data type of the parameter, choose suitable components from the Dictionary structure type BC400_S_CONNECTION.
   a) Open the *Import* tab page. Enter the names of the parameters and the data types.
4. Create a (classic) exception (suggested name: **NO_DATA**).
   a) Open the *Exceptions* tab page. Enter the name of the exception and a short text. You must not select the *Exception Classes* field.

Task 3

In the source code of the function module, implement a single record access to the database table SPFLI. Use the interface parameters of the function module to do this. Raise an exception for the case in which no data can be found for the selection.

1. Implement a single record access (SELECT SINGLE) to the database table SPFLI that fills the export parameter of the function module with values. List all of the table fields that appear as components in the export parameter. Make sure that the order of the fields matches the order of the structure components.
   a) See the source code extract from the model solution.
2. Apply a selection condition (WHERE clause) to the SELECT statement. Use both import parameters of the function module to do this.
   a) See the source code extract from the model solution.
3. Raise the exception for the function module if the database access returns no data (sy-subrc <> 0).
   a) See the source code extract from the model solution.

Task 4

Activate and test your function module.

1. Activate the function module.
   a) In the *Function Builder* toolbar, choose the *Activate* pushbutton.
2. Test your function module.

> Hint:
> Check also the raising of the exception, if an error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) In the *Function Builder* toolbar, choose the *Test/Execute* pushbutton.

b) Supply the import parameters with values and choose the *Execute* pushbutton.

c) Check the result.

Source Code Excerpt - Function Module:

```
FUNCTION BC400_DDS_CONNECTION_GET .
*"----------------------------------------------------------
*"  IMPORTING
*"     REFERENCE(IV_CARRID) TYPE  BC400_S_CONNECTION-CARRID
*"     REFERENCE(IV_CONNID) TYPE  BC400_S_CONNECTION-CONNID
*"  EXPORTING
*"     REFERENCE(ES_CONNECTION) TYPE  BC400_S_CONNECTION
*"  EXCEPTIONS
*"      NO_DATA
*"----------------------------------------------------------
   SELECT SINGLE carrid connid cityfrom airpfrom
                 cityto airpto fltime deptime arrtime
         FROM spfli
         INTO es_connection
         WHERE carrid = iv_carrid
           AND connid = iv_connid.

  IF sy-subrc <> 0.
    RAISE no_data.
  ENDIF.
ENDFUNCTION.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Read single database records

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 10
Lesson 3

Reading Multiple Database Records

LESSON OVERVIEW

This lesson explains how to read multiple data records from the same database table.

Business Example

You need to evaluate data from database tables. But instead of single records you want to read and process a whole set of data records. For this reason, you require the following knowledge:

- An understanding of how to implement a SELECT loop
- An understanding of how to implement an array fetch

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Implement a SELECT loop
- Implement an array fetch

Read Several Rows of Data Using a Loop

Figure 202: Reading Several Rows Using a Loop

© Copyright . All rights reserved.

Prohibida su reproduccion

You can use the SELECT loop to read several rows of a database table into the program in succession.

The WHERE clause determines which lines are read into the target structure and processed using the statement block specified in the loop body. You can logically connect multiple logical conditions within the WHERE clause using AND or OR.

The database delivers the data to the database interface of the application server in packages. The database interface copies the records to the target area row-by-row for processing in the processing block between SELECT and ENDSELECT.

The loop is left automatically after all of the required rows have been read and evaluated.

You should query the return value after the SELECT loop, that is, after the ENDSELECT statement. The sy-subrc returns a value of 0 if the system read at least one row. In this case, sy-dbcnt contains the number of records read.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement a SELECT Loop

Business Example

You want to develop a program in which all of the flight times for a single flight connection are read from the database. You encapsulate the required database access by a SELECT loop in a function module so that you can reuse it.

Read from the database using a SELECT loop and fill an internal table record by record.

Template

None

Solution

BC400_DDS (function group)

BC400_DDS_FLIGHTLIST_GET (function module)

Task 1

Create a function group, if one does not already exist.

1. Create a new function group in your package (suggested name: **`ZBC400_##`**).

Task 2

Create a new function module in the function group. Maintain the interface for the function module in such a way that it imports an airline ID and a flight connection number and returns a list of flight times. Raise an exception for the case in which no data can be found for the selection.

1. Create a new function module in your function group (suggested name: **`Z_BC400_##_FLIGHTLIST_GET`**).
2. Create an export parameter for the list of flight dates (suggested name: **`ET_FLIGHTS`**). For the data type of the parameter, use the Dictionary table type *BC400_T_FLIGHTS*.
3. Create a separate import parameter for airline ID and the connection number (suggested names: **`IV_CARRID`** and **`IV_CONNID`**). For typing, choose suitable components of structure type *BC400_S_FLIGHT* (line type of table type *BC400_T_FLIGHTS*).
4. Define a classic exception (suggested name: **`NO_DATA`**).

Task 3

© Copyright . All rights reserved.

Prohibida su reproduccion

In the source code of the function module, implement a SELECT loop to the database table SFLIGHT. Make sure that the export parameter of the function module is filled with a new row in each loop pass. Raise an exception for the case in which no data can be found for the selection.

1. Declare a local, structured data object in the function module (suggested name: **`ls_flight`**) that can be used as a work area for the export parameter (internal table).
2. Implement a SELECT loop in database table *SFLIGHT* that fills the local data object with values on every loop pass. List all of the table fields that appear as components in the line type of the export parameter. Make sure that the order of the fields matches the order of the structure components.
3. Apply a selection condition (WHERE clause) to the SELECT statement. Use both import parameters of the function module to do this.
4. Use the local data object to add a new row to the export parameter for each loop pass (APPEND statement). Calculate the utilization for each data record as a percentage beforehand within the loop (structure component PERCENTAGE). To do this, use the maximum number of seats and the number of occupied seats (structure components SEATSMAX and SEATSOCC).
5. Make sure that the export parameter does not contain any data before the first loop pass.

> Hint:
> If you set the call by value for the export parameter, the parameter is initial before each function module call. For parameters of table type, call by reference is recommended for performance reasons. In this case, the export parameter can already contain data if the function module is called with a non-initial actual parameter.

6. Raise the exception for the function module if the database access returns no data (sy-subrc <> 0).
7. (OPTIONAL) After the select statement has successfully been called, sort the flight list in descending order by percentage utilization before returning the data.

Task 4

Activate and test your function module.

1. Activate the function module.
2. Test your function module.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement a SELECT Loop

Business Example

You want to develop a program in which all of the flight times for a single flight connection are read from the database. You encapsulate the required database access by a SELECT loop in a function module so that you can reuse it.

Read from the database using a SELECT loop and fill an internal table record by record.

Template

None

Solution

BC400_DDS (function group)

BC400_DDS_FLIGHTLIST_GET (function module)

Task 1

Create a function group, if one does not already exist.

1. Create a new function group in your package (suggested name: **`ZBC400_##`**).
   a) In the navigation area, open the context menu for the package and choose *Create → Function Group.*
   b) Enter the name of the function group and a short text and choose *Save*.
   c) Assign the function group to your package and submit your request in the usual way.

Task 2

Create a new function module in the function group. Maintain the interface for the function module in such a way that it imports an airline ID and a flight connection number and returns a list of flight times. Raise an exception for the case in which no data can be found for the selection.

1. Create a new function module in your function group (suggested name: **`Z_BC400_##_FLIGHTLIST_GET`**).
   a) In the navigation area, open the context menu for your function group and choose *Create → Function Module.*
   b) Enter the name of the function module and a short text and choose *Save*.
2. Create an export parameter for the list of flight dates (suggested name: **`ET_FLIGHTS`**). For the data type of the parameter, use the Dictionary table type *BC400_T_FLIGHTS*.

© Copyright . All rights reserved.

Prohibida su reproduccion

   a) Open the *Export* tab page. Enter the name of the parameter and the data type.

3. Create a separate import parameter for airline ID and the connection number (suggested names: **`IV_CARRID`** and **`IV_CONNID`**). For typing, choose suitable components of structure type *BC400_S_FLIGHT* (line type of table type *BC400_T_FLIGHTS*).
   a) Open the *Import* tab page. Enter the names of the parameters and the data types.

4. Define a classic exception (suggested name: **`NO_DATA`**).
   a) Open the *Exceptions* tab page. Enter the name of the exception and a short text. Do not choose the *Exception Classes* field.

Task 3

In the source code of the function module, implement a SELECT loop to the database table SFLIGHT. Make sure that the export parameter of the function module is filled with a new row in each loop pass. Raise an exception for the case in which no data can be found for the selection.

1. Declare a local, structured data object in the function module (suggested name: **`ls_flight`**) that can be used as a work area for the export parameter (internal table).
   a) Declare the data object either with direct reference to the line type (Dictionary structure type *BC400_S_FLIGHT*) or relative to the export parameter (LIKE LINE OF).
   b) See the source code extract from the model solution.

2. Implement a SELECT loop in database table *SFLIGHT* that fills the local data object with values on every loop pass. List all of the table fields that appear as components in the line type of the export parameter. Make sure that the order of the fields matches the order of the structure components.
   a) See the source code extract from the model solution.

3. Apply a selection condition (WHERE clause) to the SELECT statement. Use both import parameters of the function module to do this.
   a) See the source code extract from the model solution.

4. Use the local data object to add a new row to the export parameter for each loop pass (APPEND statement). Calculate the utilization for each data record as a percentage beforehand within the loop (structure component PERCENTAGE). To do this, use the maximum number of seats and the number of occupied seats (structure components SEATSMAX and SEATSOCC).
   a) See the source code extract from the model solution.

5. Make sure that the export parameter does not contain any data before the first loop pass.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> If you set the call by value for the export parameter, the parameter is initial before each function module call. For parameters of table type, call by reference is recommended for performance reasons. In this case, the export parameter can already contain data if the function module is called with a non-initial actual parameter.

a) REFRESH statement. See the source code extract from the model solution.

6. Raise the exception for the function module if the database access returns no data (sy-subrc <> 0).
   a) See the source code extract from the model solution.
7. (OPTIONAL) After the select statement has successfully been called, sort the flight list in descending order by percentage utilization before returning the data.
   a) SORT statement. See the source code extract from the model solution.

Task 4

Activate and test your function module.

1. Activate the function module.
   a) In the *Function Builder* toolbar, choose the *Active* pushbutton.
2. Test your function module.
   a) In the *Function Builder* toolbar, choose the *Test/Execute* pushbutton.
   b) Supply the import parameters with values and choose the *Execute* pushbutton.
   c) Check the result.

> Note:
> Check also the raising of the exception if an error occurs.

Source Code Extract - Function Module:

```
FUNCTION bc400_dds_flightlist_get.
*"----------------------------------------------------------
*"  IMPORTING
*"     REFERENCE(IV_CARRID) TYPE  BC400_S_FLIGHT-CARRID
*"     REFERENCE(IV_CONNID) TYPE  BC400_S_FLIGHT-CONNID
*"  EXPORTING
*"     REFERENCE(ET_FLIGHTS) TYPE  BC400_T_FLIGHTS
*"  EXCEPTIONS
*"      NO_DATA
*"----------------------------------------------------------

 DATA ls_flight TYPE bc400_s_flight.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
 REFRESH et_flights.

 SELECT carrid connid fldate seatsmax seatsocc
  FROM sflight INTO ls_flight
  WHERE carrid = iv_carrid
  AND connid = iv_connid.
   ls_flight-percentage =
      ls_flight-seatsocc / ls_flight-seatsmax * 100.
  APPEND ls_flight TO et_flights.
 ENDSELECT.

 IF sy-subrc <> 0.
  RAISE no_data.
 ELSE.
  SORT et_flights BY percentage DESCENDING.
 ENDIF.
ENDFUNCTION.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Read Several Rows of Data Using an Array Fetch

Figure 203: Reading Several Rows of Data Using an Array Fetch

You can use the INTO TABLE addition to copy the selected part of the database directly into an internal table instead of doing so row-by-row. This technique is called an array fetch. It is a high performance technique for filling an internal table with entries from a database table, as the data transport is carried out in blocks and not in rows.

An array fetch is not a type of loop processing; so no ENDSELECT statement is required or allowed.

In the same way as in the other SELECT variants, the internal table that is specified as the target in the array fetch must be structured with left-justification just like the field list. If the internal table does not meet this prerequisite, you have to use the INTO CORRESPONDING FIELDS OF TABLE addition instead of the INTO TABLE addition. There, the database columns specified in the field list are copied to the columns of the internal table that have the same names. Make sure that the field types of the corresponding columns match in order to avoid complex conversions and possibly incomplete data transfer to the target table.

With array fetch, any content that might be present in the internal table is overwritten. If you want to append rows instead of overwriting, use the APPENDING TABLE addition.

If the system copies at least one record to the internal table, sy-subrc returns the value of 0. The number of rows that were copied is returned in sy-dbcnt.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement an Array Fetch

Business Example

You want to develop a program in which all of the flight times for a single flight connection are read from the database. You encapsulate the required database access using an array fetch in a function module so that you can reuse it.

Template

BC400_DDS (function group)

BC400_DDS_FLIGHTLIST_GET (function module)

Solution

BC400_DDS (function group)

BC400_DDS_FLIGHTLIST_GET_OPT (function module)

Read from the database using an array fetch and completely fill an internal table.

Task 1

Create a function group, if one does not already exist.

1. Create a new function group in your package (suggested name: **`ZBC400_##`**).

Task 2

Copy function module BC400_DDS_FLIGHTLIST_GET from the function group BC400_DDS or your own function module Z_BC400_##_FLIGHTLIST_GET under a new name to your own function module.

1. Copy the function module (new name: **`Z_BC400_##_FLIGHTLIST_GET_OPT`**).

Task 3

Turn the SELECT loop into a comment and replace it with a database access in which the export parameter of the function module (internal table) is filled directly with data (array fetch). Implement a loop through the internal table to calculate the utilization percentage for each flight time.

1. Turn the entire SELECT loop from SELECT to ENDSELECT into a comment.
2. Replace the SELECT loop with an array fetch, which fills the export parameter of the function module directly. Use the same field list and the WHERE condition as for the SELECT loop.

© Copyright . All rights reserved.

Prohibida su reproduccion

3. Implement a loop through the internal table (LOOP ... ENDLOOP). The loop should only be executed if the database access has delivered data. Use the local structured data object as the work area and calculate the utilization percentage within the loop as before.
4. Use the MODIFY statement to write the changed line back to the internal table within the loop after the calculation.

   Use the TRANSPORTING percentage addition to update only the changed field.

   Use the INDEX sy-tabix addition to change the current line.

Task 4

Activate and test your function module.

1. Activate the function module.
2. Test your function module.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement an Array Fetch

Business Example

You want to develop a program in which all of the flight times for a single flight connection are read from the database. You encapsulate the required database access using an array fetch in a function module so that you can reuse it.

Template

BC400_DDS (function group)

BC400_DDS_FLIGHTLIST_GET (function module)

Solution

BC400_DDS (function group)

BC400_DDS_FLIGHTLIST_GET_OPT (function module)

Read from the database using an array fetch and completely fill an internal table.

Task 1

Create a function group, if one does not already exist.

1. Create a new function group in your package (suggested name: **`ZBC400_##`**).
   a) In the navigation area, open the context menu for the package and choose *Create → Function Group*.
   b) Enter the name of the function group and a short text and choose *Save*.
   c) Assign the function group to your package and your request in the usual way.

Task 2

Copy function module BC400_DDS_FLIGHTLIST_GET from the function group BC400_DDS or your own function module Z_BC400_##_FLIGHTLIST_GET under a new name to your own function module.

1. Copy the function module (new name: **`Z_BC400_##_FLIGHTLIST_GET_OPT`**).
   a) In the navigation area, display the function group that contains the function module to be copied.
   b) Open the context menu for the function module and choose *Copy*.
   c) Enter the name of the new function module as well as the function group in which it should be created, and choose *Copy*.

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 3

Turn the SELECT loop into a comment and replace it with a database access in which the export parameter of the function module (internal table) is filled directly with data (array fetch). Implement a loop through the internal table to calculate the utilization percentage for each flight time.

1. Turn the entire SELECT loop from SELECT to ENDSELECT into a comment.
   a) Select the corresponding area in the ABAP Editor.
   b) Open the context menu in this area and, depending on which editor is set, choose either *Comment* or *Format → Comment Lines.* Alternatively, you can use *Ctrl* + < in both editors.
2. Replace the SELECT loop with an array fetch, which fills the export parameter of the function module directly. Use the same field list and the WHERE condition as for the SELECT loop.
   a) See the source code excerpt from the model solution.
3. Implement a loop through the internal table (LOOP ... ENDLOOP). The loop should only be executed if the database access has delivered data. Use the local structured data object as the work area and calculate the utilization percentage within the loop as before.
   a) See the source code excerpt from the model solution.
4. Use the MODIFY statement to write the changed line back to the internal table within the loop after the calculation.

   Use the TRANSPORTING percentage addition to update only the changed field.

   Use the INDEX sy-tabix addition to change the current line.
   a) See the source code excerpt from the model solution.

Task 4

Activate and test your function module.

1. Activate the function module.
   a) In the *Function Builder* toolbar, choose the *Activate* pushbutton.
2. Test your function module.
   a) In the *Function Builder* toolbar, choose the *Test/Execute* pushbutton.
   b) Supply the import parameters with values and choose the *Execute* pushbutton.
   c) Check the result.

> Hint:
> Check the raising of the exception, if an error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Source Code Excerpt - Function Module:

```
FUNCTION bc400_dds_flightlist_get_opt .
*"----------------------------------------------------------
*"  IMPORTING
*"     REFERENCE(IV_CARRID) TYPE  BC400_S_FLIGHT-CARRID
*"     REFERENCE(IV_CONNID) TYPE  BC400_S_FLIGHT-CONNID
*"  EXPORTING
*"     REFERENCE(ET_FLIGHTS) TYPE  BC400_T_FLIGHTS
*"  EXCEPTIONS
*"      NO_DATA
*"----------------------------------------------------------

  DATA ls_flight TYPE bc400_s_flight.

*  SELECT carrid connid fldate seatsmax seatsocc
*         FROM sflight
*         INTO ls_flight
*         WHERE carrid = iv_carrid
*           AND connid = iv_connid.
*
*    ls_flight-percentage =
*          ls_flight-seatsocc / ls_flight-seatsmax *100.
*    APPEND ls_flight TO et_flights.
*  ENDSELECT.

  SELECT carrid connid fldate seatsmax seatsocc
        FROM sflight
        INTO TABLE et_flights
        WHERE carrid = iv_carrid
          AND connid = iv_connid.

  IF sy-subrc <> 0.
    RAISE no_data.
  ELSE.
    LOOP AT et_flights INTO ls_flight.
      ls_flight-percentage =
        ls_flight-seatsocc / ls_flight-seatsmax * 100.
      MODIFY  et_flights FROM  ls_flight
        INDEX sy-tabix
        TRANSPORTING percentage.
    ENDLOOP.
    SORT et_flights BY percentage DESCENDING.
  ENDIF.
ENDFUNCTION.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Implement a SELECT loop
- Implement an array fetch

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 10
Lesson 4

Handling Other Aspects of Database Access

LESSON OVERVIEW

This lessson contains a brief outlook on some further aspects of database access.

Business Example

You want to know more about database accesses. You want to make your database accesses secure and efficient.For this reason, you require the following knowledge:

- An understanding of client-specific data
- An understanding of database indexes
- An understanding of the SAP table buffer
- An understanding of how to read data from several database tables
- An understanding of how to identify ways to change data in a database table

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Read client-specific data
- Use database indexes
- Explain the SAP table buffer
- Read data from several database tables
- Identify ways to change data in a database table

© Copyright . All rights reserved.

Prohibida su reproduccion

Read Data from Client-Specific Tables

Figure 204: Reading Data from Client-Specific Tables

A database table is classified as a client-specific table if it has a client field (data type CLNT) as the first key column and contains client-specific entries.

If you select data from a client-specific table without specifying the client, only data records from the current client are read (a restriction to the current client is automatically added to the WHERE clause of the SELECT statement).

If you want to read data from an explicitly specified client, specify the client name in the WHERE clause. However, the CLIENT SPECIFIED addition must be made after the FROM clause.

As cross-client reading is rarely requested in practice and is not relevant in the context of this course, the client field is usually omitted in presentations.

General Database Performance Issues

In most cases, database accesses make considerable demands on the runtime requirement of an ABAP application. In order to avoid unnecessarily overloading the system and to keep the wait time for the other users to a minimum, pay close attention to runtime requirements for database accesses.

Open SQL has a range of technologies available to enable you to optimize the runtime requirement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Database Indexes

Figure 205: Access Through Key Fields

Every database manages the data records within the database table based on key fields. If access to the table is restricted to all, or at least the first few key fields, the database can quickly and effectively access the required data records.

© Copyright . All rights reserved.

Prohibida su reproduccion

Access Through Non-Key Fields

Figure 206: Access Through Non-Key Fields

If database access is directed at fields that do not belong to the table key (non-key fields), the internal ordering principle cannot be used for rapid access. In the worst-case scenario, the entire table, or at least a very large part of the table, has to be searched for the required entries. This is referred to as a sequential search, and the wait time for the database access can be very long.

> Note:
> In the figure, the data records within the database table are sorted according to the key field and stored on this basis. Some database systems actually work in this way. With others, the data records are not sorted. Instead, an index is created using the key fields. This index is referred to as the primary index. This is of no significance here.

© Copyright . All rights reserved.

Prohibida su reproduccion

Access Through a Secondary Index

Figure 207: Access Through a Secondary Index

If you access a database table frequently using a certain selection of search fields, defining a secondary index containing the fields used in the selection will speed up corresponding accesses.

The secondary index is a separate, small table that contains the index fields and a reference to the relevant data records for each data record in the actual database table. If the fields in the selection match the fields in the secondary index (at least left-aligned and gap-free), the database searches within the secondary index and then reads the corresponding data records through the reference.

Create a secondary index in display mode for the relevant transparent table in the ABAP Dictionary using the *Indexes* pushbutton. When you activate the index, the secondary index is created in the database.

Whether and how the system uses an established secondary index in the database access is a function of the database system known as the Database Optimizer. In Open SQL, it is neither possible nor necessary to trigger the use of a secondary index by specifying one explicitly in the SELECT statement.

Hint:
For selections from client-specific tables, the client always transmits to the database (that is, it is always part of the selection). Therefore, including the client field when defining an index for such tables makes sense. At runtime, the database can use the client field to restrict the database search to the relevant client block in the secondary index.

© Copyright . All rights reserved.

Prohibida su reproduccion

SAP Table Buffer

Figure 208: Access Through the SAP Table Buffer

For data retrieval, the system uses a major proportion of runtime to transfer the data from the database server to the application server. If the data is frequently read but seldom changed, you can reduce runtime by buffering the data on the application server.

Whether and how data can be buffered is a decision that needs to be made separately for each database table. Whether to buffer the data depends on how frequently the data is read or changed. The buffer settings are defined in the transparent tables in the ABAP Dictionary (*Technical Settings* pushbutton).

> Caution:
> The decision about buffering a database table is not simple and needs be made by experienced ABAP developers in consultation with the system administrator. A separate SAP table buffer exists for each application server. If a system is comprised of several applications, a special synchronization mechanism is needed to ensure that the corresponding buffer contents are invalidated after changes have been made in the database. However, time lags in the synchronization process mean that for a short period of time, old data could be read from the buffer. This issue needs to be taken into account when making decisions about buffering.

If an ABAP program reads a buffered table, the database interface first tries to get the required data from the SAP table buffer. Reading data from the SAP buffer table speeds up access by a factor of between 10 and 100 compared to reading data from the database. The precise increase in speed depends on the structure of the table and on the exact system configurations.

© Copyright . All rights reserved.

Prohibida su reproduccion

If the required data is not yet in the SAP table buffer, the database is accessed. Afterwards the database interface also stores the read data in the SAP table buffer.

> Hint:
> There are variants and additions for the SELECT statement that call for data to be read directly from the database, regardless of buffer settings. Such access is said to bypass the buffer. This kind of access can cause performance problems when accessing buffered tables and should be avoided.

Join Mechanism

Figure 209: Example of a Table Join

There is often a requirement to read data from various tables and display it on one table.

In general, the technique with the best performance for this task is a table join.

You will read and output records from the database table SPFLI. For each record, the full names of each respective airline, which are stored in SCARR, are to be output as well. The figure shows the logical creation of the corresponding table join from which you can select all of the required data using the SELECT statement.

© Copyright . All rights reserved.

Prohibida su reproduccion

Entries for Defining a Table Join

Figure 210: Entries for Defining a Table Join

Specify the following details when defining a table join:

- Join tables

  Which database tables should be accessed and joined?

- Join conditions

  What are the conditions for summarizing the corresponding records in the tables to be merged in a table join?

- Join columns

  Which columns from the selected tables should be made available in the table join?

© Copyright . All rights reserved.

Prohibida su reproduccion

Implementation Options for Table Joins

Figure 211: Implementation Options for Table Joins

The different ways for implementing a table join are as follows:

- In the ABAP Dictionary, create a database view that corresponds to a table join and select from it in your program.

  For detailed information, refer to the online documentation for the ABAP Workbench in the ABAP Dictionary section.

- In your program, you define the join directly in your SELECT statement (ABAP join). At runtime, the system dynamically generates an appropriate database query in the database interface.

  For more information, see the keyword documentation for the FROM clause of the SELECT statement.

> Hint:
> Table joins and a database views are only specific views on the content of database tables. They do not store the same data redundantly. When a selection is made the data is read from the underlying database tables.

You can join more than two tables in a table join.

© Copyright . All rights reserved.

Prohibida su reproduccion

Database Change Accesses

Figure 212: Database Change Accesses

In addition to the SELECT statement, Open SQL contains the UPDATE, INSERT, DELETE, and MODIFY statements. However, it is important to understand the SAP transaction concept associated with these database change accesses to avoid causing data inconsistencies.

LESSON SUMMARY

You should now be able to:

- Read client-specific data
- Use database indexes
- Explain the SAP table buffer
- Read data from several database tables
- Identify ways to change data in a database table

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 10
Lesson 5

Working with Authorization Checks

LESSON OVERVIEW

This lesson explains why an authorization check is useful and how to include it in your programs.

Business Example

Authorization checks in your programs are necessary to protect the data from unauthorized access. For this reason, you require the following knowledge:

- An understanding of the SAP authorization concept
- An understanding of how to implement authorization checks

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the SAP authorization concept
- Implement authorization checks

SAP Authorization Concept

Figure 213: SAP Authorization Concept

© Copyright . All rights reserved.

Prohibida su reproduccion

You must protect critical data and parts of the functional scope of the SAP system from unauthorized access. You implement authorization checks in your program so that each user may only access areas for which he or she is authorized.

Authorization Objects and Authorizations (Example)

Figure 214: Authorization Objects and Authorizations (Example)

Authorization objects are defined within object classes. When defining an authorization object, the developer specifies the appropriate fields (without values). An actual authorization is derived from an authorization object by assigning values to these fields. This authorization can be integrated into the required user master record through an authorization profile.

Several different authorizations (for integration into different user master records) can be created for an authorization object.

© Copyright . All rights reserved.

Prohibida su reproduccion

Authorization Check (Principle)

Figure 215: Authorization Check (Principle)

At runtime, use the AUTHORITY-CHECK statement to check whether the user has the authorization required to execute the function being called in the user master record.

Depending on the check result (SY_SUBRC), continue the program accordingly, as follows:

- SY_SUBRC = 0

  User has the required authorization to perform the function (for example, SELECT), therefore continue with the program.

- SY_SUBRC <> 0

  User does not have sufficient authorization, therefore, issue an error/warning message.

> Hint:
> In addition to the technique described, it is also possible to control access to the entire program and transactions using authorizations. However, consider such checks as additions to, and not substitutions for the explicit authorization check by the developer.

Under normal circumstances, the definition of authorization objects is part of data modeling and the creation of database tables. In this course, you will access an existing data model so that you can use authorization object S_CARRID in this data model.

Implementing the authorization concept is one of the tasks of the developer who programs access to database tables.

The subsequent steps, such as defining the authorizations and profiles and designing the user master records, are tasks of the administrator.

© Copyright . All rights reserved.

Prohibida su reproduccion

Display Authorization Objects

Figure 216: Displaying Authorization Objects

Before you can implement the required authorization check in your program, determine the structure (the fields) of the respective authorization concept. An object usually consists of the *ACTVT* (*Activity*) field and one other field, which specifies the data type to be protected (that is, material number, airline, and so on). The values of these authorization fields specify what each user is authorized to do.

You can display an authorization object directly in the *ABAP Workbench* (for example, by using the *Other Object* pushbutton or the *Other* tab page). You can also use transaction `SU21`. Here, you will get an overview of all of the authorization objects created in the system.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implementation of Authorization Checks

Figure 217: Authorization Check (Syntax Example)

The figure shows how to implement an authorization check.

For the authorization check in the program, you specify the authorization that is to be checked in the master record of the current user. You specify the authorization by specifying the authorization object, its fields, and the appropriate field values. Refer to the syntax shown in the figure.

In this example, the user authorization for access to the S_CARRID object is checked; field *CARRID (airline)* contains the airline entered by the user, and field *ACTVT (activity)* contains the value '03' (display).

After the AUTHORITY-CHECK statement, check return code SY-SUBRC and implement the further processing of your program accordingly.

Hint:
To exclude a check for a field, either do not enter the field in the AUTHORITY-CHECK statement or enter DUMMY as the field value. DUMMY is a predefined description entered without quotation marks.

An example of a suppressed field check: When a change transaction is called, the system should immediately check whether the user has authorization to make a change for any airline. If the check fails, an appropriate message is to be output to the user immediately. Such a check can be implemented with the following syntax:

```
AUTHORITY-CHECK OBJECT 'S_CARRID'
ID 'CARRID' DUMMY
ID 'ACTVT' FIELD '02'.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Authorization Checks in Programs

Figure 218: Implementing Authorization Checks in Programs

To avoid spelling errors in object and field names, generate the AUTHORITY-CHECK statement in your source code by using the *Pattern* pushbutton. Then, maintain the field values and implement the evaluation of SY-SUBRC.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement an Authorization Check

Business Example

You want to develop a program in which data for the flight model is read from the database. Access to the data for this model is controlled using authorizations in authorization object S_CARRID. You, therefore, need to perform an authorization check. To make the authorization check reusable, you encapsulate it in a function module.

Template

None

Solution

BC400_DDS_AUTH_CHECK (function module)

Implement authorization checks.

Task 1

Create a new function module in the function group. Maintain the interface for the function module so that it imports an airline ID and an activity for the authorization check. Create two exceptions for the confirmation: one in case no authorization exists and another in case the caller has specified an invalid activity.

1. Create a new function module in your function group (suggested name: **`Z_BC400_##_AUTH_CHECK`**).
2. Create a separate import parameter for the airline ID and the activity (suggested names: **`IV_CARRID and IV_ACTIVITY`**). To type the data element, choose *S_CARR_ID* and *ACTIV_AUTH*.

   > Hint:
   > The correct data elements are defined in the authorization object. To do this, open authorization object S_CARRID in display mode and choose the input help for one of the authorization fields. You will find the relevant data element (second column) for each authorization field (first column) here.

3. Create two (classic) exceptions (suggested names: **`NO_AUTH and WRONG_ACTIVITY`**).

Task 2

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement an authorization check in the source code for authorization object *S_CARRID*. Raise an exception for the case in which the authorization does not exist or the caller has not specified any of the relevant activities.

1. Implement an authorization check for authorization object *S_CARRID*. Supply the authorization fields with the import parameters for the function module. Evaluate the return value (sy-subrc) and raise an exception for the case in which authorization is missing.

> Hint:
> Use the *Pattern* function to make the authorization check, because it is very intolerant of typos and uppercase/lowercase errors.

2. Before the authorization check, make sure that the caller has specified only one of the relevant activities. If necessary, raise an exception.

> Hint:
> The relevant activities are defined in the authorization object. To do this, open the authorization object in display mode and choose the *Permitted Activities* pushbutton at the bottom of the screen.

Task 3

Activate and test your function module.

1. Activate the function module.
2. Test your function module.

> Hint:
> Check also the raising of the exception, if an error occurs.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement an Authorization Check

Business Example

You want to develop a program in which data for the flight model is read from the database. Access to the data for this model is controlled using authorizations in authorization object S_CARRID. You, therefore, need to perform an authorization check. To make the authorization check reusable, you encapsulate it in a function module.

Template

None

Solution

BC400_DDS_AUTH_CHECK (function module)

Implement authorization checks.

Task 1

Create a new function module in the function group. Maintain the interface for the function module so that it imports an airline ID and an activity for the authorization check. Create two exceptions for the confirmation: one in case no authorization exists and another in case the caller has specified an invalid activity.

1. Create a new function module in your function group (suggested name: **`Z_BC400_##_AUTH_CHECK`**).
   a) In the navigation area, open the context menu for your function group and choose *Create → Function Module*.
   b) Enter the name of the function module and a short text and choose *Save*.
2. Create a separate import parameter for the airline ID and the activity (suggested names: **`IV_CARRID and IV_ACTIVITY`**). To type the data element, choose *S_CARR_ID* and *ACTIV_AUTH*.

> Hint:
> The correct data elements are defined in the authorization object. To do this, open authorization object S_CARRID in display mode and choose the input help for one of the authorization fields. You will find the relevant data element (second column) for each authorization field (first column) here.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) Open the *Import* tab page. Enter the names of the parameters and the data types.

3. Create two (classic) exceptions (suggested names: **`NO_AUTH and WRONG_ACTIVITY`**).
   a) Open the *Exceptions* tab page. Enter the names of the exceptions and a short text. Do not set the *Exception Classes* indicator.

Task 2

Implement an authorization check in the source code for authorization object *S_CARRID*. Raise an exception for the case in which the authorization does not exist or the caller has not specified any of the relevant activities.

1. Implement an authorization check for authorization object *S_CARRID*. Supply the authorization fields with the import parameters for the function module. Evaluate the return value (sy-subrc) and raise an exception for the case in which authorization is missing.

> Hint:
> Use the *Pattern* function to make the authorization check, because it is very intolerant of typos and uppercase/lowercase errors.

   a) See the source code excerpt from the model solution.

2. Before the authorization check, make sure that the caller has specified only one of the relevant activities. If necessary, raise an exception.

> Hint:
> The relevant activities are defined in the authorization object. To do this, open the authorization object in display mode and choose the *Permitted Activities* pushbutton at the bottom of the screen.

   a) See the source code excerpt from the model solution.

Task 3

Activate and test your function module.

1. Activate the function module.
   a) In the *Function Builder* toolbar, choose the *Activate* pushbutton.

2. Test your function module.

> Hint:
> Check also the raising of the exception, if an error occurs.

   a) In the *Function Builder* toolbar, choose the *Test/Execute* pushbutton.

© Copyright . All rights reserved.

Prohibida su reproduccion

b) Supply the import parameters with values and choose the *Execute* pushbutton.

c) Check the result.
Source Code Excerpt - Function Module:

```
FUNCTION bc400_dds_auth_check.
*"----------------------------------------------------
*"  IMPORTING
*"     REFERENCE(IV_CARRID) TYPE  S_CARR_ID
*"     REFERENCE(IV_ACTIVITY) TYPE  ACTIV_AUTH
*"  EXCEPTIONS
*"      NO_AUTH
*"      WRONG_ACTIVITY
*"----------------------------------------------------
  CASE iv_activity.
    WHEN '01' OR '02' OR '03'.

      AUTHORITY-CHECK OBJECT 'S_CARRID'
               ID 'CARRID' FIELD iv_carrid
               ID 'ACTVT'  FIELD iv_activity.
      IF sy-subrc <> 0.
        RAISE no_auth.
      ENDIF.

    WHEN OTHERS.
      RAISE wrong_activity.
  ENDCASE.
ENDFUNCTION.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Explain the SAP authorization concept
- Implement authorization checks

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. For each entity fixed in the data model, the developer creates a transparent table in the _____.

   Choose the correct answers.

   - [ ] A transparent field
   - [ ] B ABAP Dictionary
   - [ ] C data field
   - [ ] D database

2. When using the transparent table as a data type, other properties, such as the key definition or the technical properties, are relevant.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

3. Which of the following are the types of reuse components that encapsulate database access?

   Choose the correct answers.

   - [ ] A Conceptual database
   - [ ] B Function modules
   - [ ] C Business Application Programming Interfaces (BAPIs)
   - [ ] D Methods of local classes

© Copyright . All rights reserved.

Prohibida su reproduccion

4. The SELECT clause determines which lines are read into the target structure and processed using the statement block which you specify in the loop body.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

5. If you want to append rows in a Select statement, you can use the _____ addition.

   Choose the correct answer.

   - [ ] A INTO TABLE
   - [ ] B APPENDING TABLE
   - [ ] C INTO CORRESPONDING FIELD OF TABLE
   - [ ] D END SELECT

6. If you select data from client-specific tables without specifying the client, data records from the current and all other clients are read.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

7. In the worst-case scenario, the system has to search the entire table, or at least a very large part thereof, for the required entries. This concept is referred to as _____.

   Choose the correct answer.

   - [ ] A direct search
   - [ ] B binary search
   - [ ] C sequential search
   - [ ] D indirect search

© Copyright . All rights reserved.

Prohibida su reproduccion

8. Which of the following should you specify when defining a table join?

   Choose the correct answers.

   - [ ] A Join tables
   - [ ] B Join conditions
   - [ ] C Join rows
   - [ ] D Join columns

9. If an ABAP program reads a buffered table, the database interface tries to get the required data from the SAP table buffer.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

10. Which of the following statements are contained within Open SQL?

    Choose the correct answers.

    - [ ] A CREATE
    - [ ] B UPDATE
    - [ ] C INSERT
    - [ ] D SELECT

11. At runtime, which of the following statements can you use to check whether the actual user has the authorization required for executing the function in the user master record?

    Choose the correct answer.

    - [ ] A AUTHORITY
    - [ ] B SELECT
    - [ ] C AUTHORITY-CHECK
    - [ ] D VALID AUTHORITY

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. For each entity fixed in the data model, the developer creates a transparent table in the _____.

   Choose the correct answers.

   - [ ] A transparent field
   - [x] B ABAP Dictionary
   - [ ] C data field
   - [ ] D database

2. When using the transparent table as a data type, other properties, such as the key definition or the technical properties, are relevant.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

3. Which of the following are the types of reuse components that encapsulate database access?

   Choose the correct answers.

   - [ ] A Conceptual database
   - [x] B Function modules
   - [x] C Business Application Programming Interfaces (BAPIs)
   - [x] D Methods of local classes

© Copyright . All rights reserved.

Prohibida su reproduccion

4. The SELECT clause determines which lines are read into the target structure and processed using the statement block which you specify in the loop body.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

5. If you want to append rows in a Select statement, you can use the _____ addition.

   Choose the correct answer.

   - [ ] A INTO TABLE
   - [x] B APPENDING TABLE
   - [ ] C INTO CORRESPONDING FIELD OF TABLE
   - [ ] D END SELECT

6. If you select data from client-specific tables without specifying the client, data records from the current and all other clients are read.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

7. In the worst-case scenario, the system has to search the entire table, or at least a very large part thereof, for the required entries. This concept is referred to as _____.

   Choose the correct answer.

   - [ ] A direct search
   - [ ] B binary search
   - [x] C sequential search
   - [ ] D indirect search

© Copyright . All rights reserved.

Prohibida su reproduccion

8. Which of the following should you specify when defining a table join?

   Choose the correct answers.

   - [x] A Join tables
   - [x] B Join conditions
   - [ ] C Join rows
   - [x] D Join columns

9. If an ABAP program reads a buffered table, the database interface tries to get the required data from the SAP table buffer.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

10. Which of the following statements are contained within Open SQL?

    Choose the correct answers.

    - [ ] A CREATE
    - [x] B UPDATE
    - [x] C INSERT
    - [x] D SELECT

11. At runtime, which of the following statements can you use to check whether the actual user has the authorization required for executing the function in the user master record?

    Choose the correct answer.

    - [ ] A AUTHORITY
    - [ ] B SELECT
    - [x] C AUTHORITY-CHECK
    - [ ] D VALID AUTHORITY

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 11 Classic ABAP Report

Lesson 1

Implementing ABAP Lists 490

Lesson 2

Implementing Selection Screens 494

Exercise 29: Create a Selection Screen and a Classic ABAP List 503

Lesson 3

Implementing Events of ABAP Reports 510

Exercise 30: Implement Events of ABAP Reports 517

UNIT OBJECTIVES

- Implement ABAP lists
- Describe the attributes and benefits of selection screens
- Implement options for restricting selections of selection screens
- Implement the events of ABAP reports

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 11
Lesson 1

Implementing ABAP Lists

LESSON OVERVIEW
This lesson discusses classic ABAP reports. ABAP programs of this type are closely linked with the classic event control technique as well as the special selection screen and the ABAP list user dialogs.

Business Example

You want to develop a program that uses selections on a selection screen to read and list data from a database. For this reason, you require the following knowledge:

- An understanding of how to describe the attributes and benefits of ABAP lists

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Implement ABAP lists

Properties of an ABAP List

Figure 219: Features of Lists

© Copyright . All rights reserved.

Prohibida su reproduccion

You use a list to minimize programming effort to output data.

Lists take into account the special requirements of business data in the following ways:

- Lists can be designed for a number of languages. For example, texts and headers appear in the logon language whenever a corresponding translation is available.
- Lists can display monetary values in the appropriate currency.

The following options are available when outputting a list:

- Screen

  You can add colors and icons to the screen.
- Printer

  You can print lists.
- Internet or intranet

  You can have the system convert the list to HTML format.
- Save

  You can save lists within the SAP system as well as outside, for example, for processing using spreadsheet programs.

Standard List Functions

Figure 220: Standard List Functions

The standard user interface for a list offers a range of functions.

© Copyright . All rights reserved.

Prohibida su reproduccion

You can use the Menu Painter to adapt the default list interface to your own needs. More information is available in the documentation for this tool and in the corresponding training course.

List Titles and Column Headers

Figure 221: List and Column Headers

You know how to implement translatable text symbols to make the interface of your program dependent on the logon language of the user. Text symbols are available in all types of programs, for example, for module pools, function groups, and global classes.

For executable programs (reports) in particular, you have the option of maintaining a single line list header and up to four rows of column headers in addition to the text symbols for the classic ABAP list.

Together with text symbols and selection texts, list and column headers belong to the text elements of a program.

For initial maintenance of the headers, you have to activate your program, then create the list by executing the program. You can maintain the headers directly above your list using *System → List → List Header*. The next time you start the program, they will appear in the list automatically.

To change the maintained header (follow-up maintenance), you do not have to start the program again and generate the list. Instead, starting from the editor in which you load the program, you can access the maintenance environment for changing the headers by choosing *Goto → Text Elements → List Headers*.

To translate list and column headers from within the *ABAP Editor*, choose *Goto → Translation*.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Implement ABAP lists

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 11
Lesson 2

Implementing Selection Screens

LESSON OVERVIEW
This lesson discusses classic ABAP reports. This type of ABAP program is closely linked with the classic event control technique as well as the special selection screen.

Business Example

You want to develop a program that uses selections on a selection screen to be read. For this reason, you require the following knowledge:

- An understanding of the properties of the ABAP list and benefits of selection screens
- An understanding of how to implement the options for restricting selections on the selection screen

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe the attributes and benefits of selection screens
- Implement options for restricting selections of selection screens

Selection Screen Properties

Figure 222: Architecture and Purpose of Selection Screens

© Copyright . All rights reserved.

Prohibida su reproduccion

You use selection screens for entering selection criteria for data selection. For example, if the program creates a list of data from a very large database table, it makes sense for users to select the data records they actually require and for the system to read only this data from the database. Apart from reducing the memory requirement, this also reduces the network load.

From a technical perspective, selection screens are dynpros. However, the developer does not design them directly with the Screen Painter; the system generates them in accordance with declarative statements in the source code.

Selection Screen Attributes

Figure 223: Selection Screen Attributes

The selection screen has the following standard functions:

- You can maintain texts on the selection screen in several languages. At runtime, the system displays the texts in the user's logon language.
- The system automatically performs a type check: If the user enter something that does not correspond to the type of the input field, the SAP GUI will ignore it.
- You can implement complex selections (SELECT-OPTIONS) on the selection screen in addition to single value entries (PARAMETERS). The user can then enter intervals, comparative conditions, or even patterns as restrictions.
- The user can display the field documentation for the input field by pressing F1 if the developer defines the input field using a Dictionary element (for example, data element).
- The user can call the search help attached to the Dictionary type to display possible inputs by pressing F4.

© Copyright . All rights reserved.

Prohibida su reproduccion

- The user can save completed selection screens as variants for reuse or use in background operations.

Use of Selection Options

Figure 224: Using Selection Options

The figure illustrates the use of selection options, which enable complex entries. For more information about the operation in question, choose the *Help on Screen* pushbutton on the additional screen for multiple selections.

© Copyright . All rights reserved.

Prohibida su reproduccion

Semantic Information of Global Types on the Selection Screen

Figure 225: Semantic Information of Global Types on the Selection Screen

If an input field is typed with a data element, the following additional semantic information is available on the selection screen:

- The long field label of the data element can be adopted to describe the input field on the selection screen (selection text).
- The documentation of the data element is automatically available as input help (F1).
- The search help is available as input help (F4) if it is linked to the data element.

When typing an input field with a structure field instead of a data element, semantic information is also available on the selection screen. The semantic information is then taken from the data element used in the definition of the structure field

For more information, refer to the online documentation for the ABAP Dictionary.

© Copyright . All rights reserved.

Prohibida su reproduccion

Selection Texts

Figure 226: Selection Texts

On the selection screen, the names of the input fields appear as their description by default. However, you can replace these with corresponding selection texts and then translate these texts into any other languages you require. At runtime, the system displays selection texts in the logon language of the user (automatic language).

Just like list headers and text symbols, selection texts belong to the text elements of the program. From the *ABAP Editor*, choose *Goto → Text Elements → Selection Texts* to maintain them. You can implement your translation using *Goto → Translation*.

If you type the input field directly or indirectly with a data element, you can take over the long field name from one of the texts stored in the ABAP Dictionary ("Dictionary reference"). This provides you with an easy option for standardizing texts for all selection screens.

© Copyright . All rights reserved.

Prohibida su reproduccion

Single-Value Input

Figure 227: Single-Value Input (PARAMETERS Statement)

The figure illustrates the usage and the runtime behavior of an input variable defined using a PARAMETERS statement. The definition of such an input variable creates a variable in the system and generates a selection screen with a corresponding input option.

The only difference between defining an input variable and an ordinary variable is that the PARAMETERS keyword is used instead of DATA.

Take the following aspects into consideration:

- The name of the input variable may be up to 8 characters long.
- The input variable must not be typed with the standard types F, STRING, and XSTRING.
- The assignment of a default value is implemented with the DEFAULT addition, not the VALUE addition.

The system displays a default value by means of the DEFAULT as a default value that can be overwritten.

For example, if the user enters a value and chooses the *Execute* pushbutton, the system transfers the input value to the internal variable. The system can then be used to restrict the database selection.

© Copyright . All rights reserved.

Prohibida su reproduccion

Complex Selections

Figure 228: Entering Complex Selections (SELECT-OPTIONS Statement)

You can use the SELECT-OPTIONS <name> FOR <data object> statement to define a selection option for entering complex selections, where <name> is the name of the select option and <data object> is a predefined variable. Such a definition creates an internal table of the specified name within the program (so_car) and generates a selection screen with an input option for limiting the specified variable (gs_spfli-carrid).

The system transports user entries to the automatically generated internal table when the user choose the *Execute* pushbutton. The internal table has four columns: sign, option, low, and high.

The figure illustrates the following entries created for user input:

- If LH is entered, a row is generated with the values I (inclusive), EQ (equal), LH, and Space.
- If the interval AA to DL is entered, a row is generated with the values I (inclusive), BT (between), AA, and DL.
- If BA is entered as the single value to be excluded, a row is generated with the values E (exclusive), EQ (equal), BA, and Space.

Once the internal table has been filled with the selection criteria, it can be used to limit the database selection, as shown in the figure.

The table content is interpreted as follows:

- If I_1, ... , I_n and E_1, ... , E_m are the inclusive or exclusive conditions of the internal table, the composite condition that is used to limit the data selection is (I_1 OR ... OR I_n) AND (NOT E_1) AND ... AND (NOT E_m).
- If the table is empty, the WHERE condition for the relevant field is met because there are no selections.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> The IN operator can also be used in logical expressions, for example, IF GS_SPFLI-CARRID IN SO_CAR.

The same three special features apply for the definition of select-options and the PARAMETERS statement.

If the program fills the internal table of the selection option using the DEFAULT addition or the APPEND statement before the display of the selection screen (INITIALIZATION), the system displays content on the selection screen as proposed conditions that can be overwritten.

For further details, refer to the keyword documentation for SELECT-OPTIONS.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Selection Screen and a Classic ABAP List

Business Example

Instead of a single flight number, you want to be able to enter complex selection criteria on a selection screen.

Furthermore, you want to make the design of the selection screen and the ABAP list more appealing by using translatable texts and, possibly, colors and icons.

Template

BC400_RPT_REP_A

Solution

BC400_RPS_REP_A

Use the ABAP SELECT-OPTIONS statement to implement a complex selection option on the selection screen, transfer complex selections to a data retrieval method, use translatable text elements on the selection screen and in the ABAP list, and use icons and colors in a classic list (optional).

Task 1

1. Copy the template to the name **`ZBC400_##_REP_A`**.

Task 2

Replace the parameter for the flight number on the selection screen with a complex selection (select-options). To adjust the data retrieval in such a way that the select-options is used, call a method of class CL_BC400_FLIGHTMODEL to which you can transfer the select-options table.

1. Replace the declaration of parameter PA_CON with the declaration of a select-options (suggested name: **`so_con`**).

> Hint:
> Reference the same structure type in the Dictionary when typing the select-options and parameter PA_CAR so that the corresponding search help on the selection screen is context-dependent.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. Replace the call of method CL_BC400_FLIGHTMODEL=>GET_FLIGHTS with a call of method CL_BC400_FLIGHTMODEL=>GET_FLIGHTS_RANGE. Transfer the select option table to the method as an actual parameter.

> Hint:
> Data object SO_CON is an internal table with a header. The parameters can only be transferred if you explicitly transfer the body of the table (also SO_CON[]).

3. Activate and test your program.

Task 3

Make sure language-dependent texts are displayed on the selection screen instead of the names of the data objects.

1. Maintain the text elements of the program and define selection texts. Use texts that have already been defined and translated in the ABAP Dictionary.
2. Activate and test your program.

Task 4

If no data is available, output a translatable text in the list and create translatable headers.

1. Maintain a text symbol and output it with the WRITE statement if the method returns no data.
2. Maintain the headers for the list.

Task (Optional)

Define the key fields for the flight times as light blue. Output an icon at the start of each row to show utilization by means of a traffic light. Use red for fully booked flights, yellow for well-booked flights, and green for flights with few or no bookings.

1. Provide the output of each of the key fields (CARRID, CONNID, FLDATE) with a COLOR addition and the color constant for key fields from the COL type group.
2. Output a traffic light icon at the start of each row. Use one of the constants ICON_RED_LIGHT, ICON_YELLOW_LIGHT, and ICON_GREEN_LIGHT from the ICON type group, depending on the utilization percentage of the flight.

© Copyright . All rights reserved.

Prohibida su reproduccion

Create a Selection Screen and a Classic ABAP List

Business Example

Instead of a single flight number, you want to be able to enter complex selection criteria on a selection screen.

Furthermore, you want to make the design of the selection screen and the ABAP list more appealing by using translatable texts and, possibly, colors and icons.

Template

BC400_RPT_REP_A

Solution

BC400_RPS_REP_A

Use the ABAP SELECT-OPTIONS statement to implement a complex selection option on the selection screen, transfer complex selections to a data retrieval method, use translatable text elements on the selection screen and in the ABAP list, and use icons and colors in a classic list (optional).

Task 1

1. Copy the template to the name **`ZBC400_##_REP_A`**.
   a) Carry out this step in the usual manner.

Task 2

Replace the parameter for the flight number on the selection screen with a complex selection (select-options). To adjust the data retrieval in such a way that the select-options is used, call a method of class CL_BC400_FLIGHTMODEL to which you can transfer the select-options table.

1. Replace the declaration of parameter PA_CON with the declaration of a select-options (suggested name: **`so_con`**).

> Hint:
> Reference the same structure type in the Dictionary when typing the select-options and parameter PA_CAR so that the corresponding search help on the selection screen is context-dependent.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) See the source code excerpt from the model solution.

2. Replace the call of method CL_BC400_FLIGHTMODEL=>GET_FLIGHTS with a call of method CL_BC400_FLIGHTMODEL=>GET_FLIGHTS_RANGE. Transfer the select option table to the method as an actual parameter.

> Hint:
> Data object SO_CON is an internal table with a header. The parameters can only be transferred if you explicitly transfer the body of the table (also SO_CON[]).

a) See the source code excerpt from the model solution.

3. Activate and test your program.
   a) Carry out this step in the usual manner.

Task 3

Make sure language-dependent texts are displayed on the selection screen instead of the names of the data objects.

1. Maintain the text elements of the program and define selection texts. Use texts that have already been defined and translated in the ABAP Dictionary.
   a) Open the maintenance of the selection screen by choosing *Goto → Text Elements → Selection Texts*.
   b) Select the checkbox in the *Dictionary Reference* column for all the listed elements of the selection screen.

> Hint:
> Only those parameters and select options are offered her which are part of active version of the source code.

   c) Save and activate the texts.
2. Activate and test your program.
   a) Carry out this step in the usual manner.

> Hint:
> You have to activate the texts (REPT type Repository object) before they are displayed when the program is executed.

Task 4

If no data is available, output a translatable text in the list and create translatable headers.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. Maintain a text symbol and output it with the WRITE statement if the method returns no data.
   a) Open the maintenance of the text symbols by choosing *Goto → Text Elements → Text Symbols*.
   b) Assign a three-character abbreviation and enter a text.
   c) Save and activate the texts.
   d) Output the text symbol on the list as shown in the source code excerpt from the model solution.
2. Maintain the headers for the list.
   a) Execute the program.
   b) Branch from the list display to header maintenance (*System → List → List Header*).
   c) Maintain the headers.
   d) Restart the program for testing.

> Hint:
> The list will be buffered on the presentation server. The headers will only be displayed when you restart the program.

Task (Optional)

Define the key fields for the flight times as light blue. Output an icon at the start of each row to show utilization by means of a traffic light. Use red for fully booked flights, yellow for well-booked flights, and green for flights with few or no bookings.

1. Provide the output of each of the key fields (CARRID, CONNID, FLDATE) with a COLOR addition and the color constant for key fields from the COL type group.
   a) See the source code excerpt from the model solution.
2. Output a traffic light icon at the start of each row. Use one of the constants ICON_RED_LIGHT, ICON_YELLOW_LIGHT, and ICON_GREEN_LIGHT from the ICON type group, depending on the utilization percentage of the flight.
   a) See the source code excerpt from the model solution.
   Source Code Excerpt:

```
*&---------------------------------------------*
*& Report  BC400_RPS_REP_A
*&---------------------------------------------*
REPORT  bc400_rps_rep_a.

TYPE-POOLS:
     icon,
     col.

CONSTANTS:
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
     gc_limit_red    TYPE s_flghtocc VALUE 98,
     gc_limit_yellow TYPE s_flghtocc VALUE 75.

DATA:
     gt_flights TYPE bc400_t_flights,
     gs_flight  TYPE bc400_s_flight.

PARAMETERS:
     pa_car TYPE bc400_s_flight-carrid.

SELECT-OPTIONS:
     so_con FOR gs_flight-connid.

TRY.
    CALL METHOD cl_bc400_flightmodel=>get_flights_range
      EXPORTING
        iv_carrid  = pa_car
        it_connid  = so_con[]  " brackets needed for so_con
      IMPORTING
        et_flights = gt_flights.
  CATCH cx_bc400_no_data.
    WRITE / 'No flights for selected flight connection'(non).
*   WRITE / text-non.   "alternative usage of text symbol
ENDTRY.

LOOP AT gt_flights INTO gs_flight.
  NEW-LINE.
  IF gs_flight-percentage >= gc_limit_red.
    WRITE  icon_red_light AS ICON.
  ELSEIF gs_flight-percentage >= gc_limit_yellow.
    WRITE  icon_yellow_light AS ICON.
  ELSE.
    WRITE  icon_green_light AS ICON.
  ENDIF.

  WRITE:  gs_flight-carrid color COL_KEY,
           gs_flight-connid color COL_KEY,
           gs_flight-fldate color COL_KEY,
           gs_flight-seatsmax,
           gs_flight-seatsocc,
           gs_flight-percentage.

ENDLOOP.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Describe the attributes and benefits of selection screens
- Implement options for restricting selections of selection screens

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 11
Lesson 3

Implementing Events of ABAP Reports

LESSON OVERVIEW

This lesson explains the implementation of ABAP reports.

Business Example

You want to develop a program that demonstrates implementation of ABAP reports. For this reason, you require the following knowledge:

- An understanding of how to implement the events of ABAP reports

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Implement the events of ABAP reports

ABAP Events

Figure 229: Structures and Procedures of an Executable ABAP Program

When you start an ABAP program, the system creates all global data objects of the programs in the working memory. The runtime system then triggers various events in succession. If a

© Copyright . All rights reserved.

Prohibida su reproduccion

processing block exists for a triggered event in the program, the system executes the statements in this block in sequence.

The figure illustrates the basic events that are triggered, the sequence in which they are triggered, and the program implements the corresponding processing blocks. An executable ABAP program is thus a collection of processing blocks that are processed for the respective events.

Outputs created by means of WRITE statements are stored in list buffers, and only displayed as a list after the system has processed the START-OF-SELECTION block.

Use of ABAP Events

Figure 230: Using ABAP Events

If you assign values to the PARAMETERS variables in the INITIALIZATION block, the system displays these as changeable default values in the input fields when the selection screen is subsequently displayed. You have the option of specifying a default value for the respective input field in the PARAMETERS definition by means of the DEFAULT addition. However, you can use the value assignment described in the INITIALIZATION block to assign another default value (dynamic pre-population of the selection screen) dynamically (that is, with reference to the situation).

If the user clicks a pushbutton on the selection screen (thus triggering Process After Input [PAI]), the system transports the entries into the corresponding internal PARAMETERS variables of the program and triggers the AT SELECTION-SCREEN event. The corresponding processing block is, therefore, suitable for an input or authorization check. If, for example, a type E message is sent in this event block because a user does not have the required authorization, the selection screen will be displayed again with the error message, allowing users to correct their entries.

Only if no error message is sent in the AT SELECTION-SCREEN block is START-OF-SELECTION then triggered.

© Copyright . All rights reserved.

Prohibida su reproduccion

The main processing of the program should then take place in the corresponding processing block.

Event Block Characteristics

- **Introduced with an event keyword**
- **Ends by beginning the next processing block**
- **Cannot be nested**
- **Existence is not absolutely necessary**
- **Sequence of event blocks is unimportant**
- **Implicit standard event block in executable program:**

  START-OF-SELECTION

Figure 231: Event Block Characteristics

Processing blocks cannot be nested, because nesting would contradict the concept of ABAP events.

The system does not execute any statement and triggers the next event if the processing block is missing.

The ABAP runtime system controls the triggering of the events and the execution of the processing blocks. The sequence in which you place the event blocks in the program is not important.

If no blocks are implemented in the program, the system assigns all statements to the standard processing block START-OF-SELECTION.

© Copyright . All rights reserved.

Prohibida su reproduccion

The INITIALIZATION Event

Figure 232: Example Program

The example illustrated here contains a selection screen with an input field for a date. Under normal circumstances, the current date should appear as the default value (DEFAULT sy-datum). However, under certain conditions (IF), the date of the same weekday of the following week (pa_date = pa_date + 7.) is to be displayed as the default value.

The figure also illustrates how the runtime system reacts when the processing block is missing. There are no statements executed for the corresponding event, and the next event is triggered.

© Copyright . All rights reserved.

Prohibida su reproduccion

AT SELECTION-SCREEN Event

Figure 233: AT SELECTION-SCREEN Event

From the selection screen, the AT SELECTION-SCREEN event is triggered both by choosing *Enter* as well as F8 *(Execute)*. After the corresponding processing block has been processed, if the users chooses the *Execute* pushbutton, the following START-OF-SELECTION event is triggered and the processing starts.

However, if the user chooses *Enter*, the system again displays the selection screen.

© Copyright . All rights reserved.

Prohibida su reproduccion

Error Messages in AT SELECTION-SCREEN

Figure 234: Error Messages in AT SELECTION-SCREEN

Immediately before processing the AT SELECTION-SCREEN block, the system transfers user entries to the corresponding variables in the program. Hence, it makes sense to implement required input and authorization checks in this processing block. If the check result is negative, you can simply send out an error message to the user and have the selection screen displayed again. The user then has the chance to make new entries.

In contrast, the system generates an error message in the START-OF-SELECTION block on program termination.

© Copyright . All rights reserved.

Prohibida su reproduccion

Syntax Example – Authorization Check with Error Dialog

```
 ...
PARAMETERS pa_car TYPE s_carr_id.

CONSTANTS  gc_actvt_display TYPE activ_auth VALUE '03'.

* Event processed after leaving the selection screen

AT SELECTION-SCREEN.

TRY.
      CALL METHOD cl_bc400_flightmodel=>check_authority
        EXPORTING
          iv_carrid   = pa_car
          iv_activity = gc_actvt_display.
    CATCH cx_bc400_no_auth .
*     Show selection screen again with error message
      MESSAGE e046(bc400) WITH pa_car.
ENDTRY.

START-OF-SELECTION.
  ...
```

Figure 235: Syntax Example – Authorization Check with Error Dialog

The figure illustrates a simple example program with an authorization check and an error dialog on the selection screen.

Additional information can be found in the keyword documentation for AT SELECTION-SCREEN.

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 11
Exercise 30

Implement Events of ABAP Reports

Business Example

You want to make a dynamic pre-assignment for the airline ID on the selection screen. If users enter an airline for which they do not have display authorization, an error message should appear and the selection screen should display again.

Template

BC400_RPS_REP_A

Solution

BC400_RPS_REP_B

Implement a dynamic pre-assignment on the selection screen and error dialogs for the standard selection screen.

Task 1

Copy your program ZBC400_##_REP_A or the copy template.

1. Copy the template to the name **`ZBC400_##_REP_B`**.

Task 2

Create an ABAP event block that is run before the selection screen is displayed and implement a dynamic pre-assignment of the parameter for the airline ID.

1. Create event block INITIALIZATION and fill parameter PA_CAR dynamically with the value LH.

> Hint:
> You have to assign the previous executable part of your program explicitly to the START-OF-SELECTION event block before you define the INITIALIZATION block.

Task 3

Make sure that after a user action on the selection screen, the system checks whether the user is authorized to display data for the airline that was entered. Implement the check in such a way that you can display the selection screen again with an error message if the authorization does not exist.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. Create the AT SELECTION-SCREEN event block and implement an authorization check. To do this, use the method check _authority of class CL_BC400_FLIGHTMODEL.
2. If the authorization is missing, send an appropriate message from message class BC400.
3. Which message type do you have to use so that the selection screen is displayed again and the user cannot access the START-OF-SELECTION processing block without authorization?

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Events of ABAP Reports

Business Example

You want to make a dynamic pre-assignment for the airline ID on the selection screen. If users enter an airline for which they do not have display authorization, an error message should appear and the selection screen should display again.

Template

BC400_RPS_REP_A

Solution

BC400_RPS_REP_B

Implement a dynamic pre-assignment on the selection screen and error dialogs for the standard selection screen.

Task 1

Copy your program ZBC400_##_REP_A or the copy template.

1. Copy the template to the name **`ZBC400_##_REP_B.`**
   a) Carry out this step in the usual manner.

Task 2

Create an ABAP event block that is run before the selection screen is displayed and implement a dynamic pre-assignment of the parameter for the airline ID.

1. Create event block INITIALIZATION and fill parameter PA_CAR dynamically with the value LH.

> Hint:
> You have to assign the previous executable part of your program explicitly to the START-OF-SELECTION event block before you define the INITIALIZATION block.

   a) See the source code excerpt from the model solution.

Task 3

Make sure that after a user action on the selection screen, the system checks whether the user is authorized to display data for the airline that was entered. Implement the check in such a way

© Copyright . All rights reserved.

Prohibida su reproduccion

that you can display the selection screen again with an error message if the authorization does not exist.

1. Create the AT SELECTION-SCREEN event block and implement an authorization check. To do this, use the method check _authority of class CL_BC400_FLIGHTMODEL.
   a) See the source code excerpt from the model solution.
2. If the authorization is missing, send an appropriate message from message class BC400.
   a) See the source code excerpt from the model solution.
3. Which message type do you have to use so that the selection screen is displayed again and the user cannot access the START-OF-SELECTION processing block without authorization?
   a) Error message, message type E.

   Source Code Excerpt:

```
*&---------------------------------------------
*& Report  BC400_RPS_REP_B
*&---------------------------------------------
REPORT  bc400_rps_rep_b.

TYPE-POOLS:
     icon,
     col.

CONSTANTS:
     gc_limit_red     TYPE s_flghtocc VALUE 98,
     gc_limit_yellow  TYPE s_flghtocc VALUE 75,
     gc_actvt_display TYPE activ_auth VALUE '03'.

DATA:
     gt_flights TYPE bc400_t_flights,
     gs_flight  TYPE bc400_s_flight.

PARAMETERS:
     pa_car TYPE bc400_s_flight-carrid.

SELECT-OPTIONS:
     so_con FOR gs_flight-connid.

INITIALIZATION.
  pa_car = 'LH'.

AT SELECTION-SCREEN.
  TRY.
      CALL METHOD cl_bc400_flightmodel=>check_authority
        EXPORTING
          iv_carrid   = pa_car
          iv_activity = gc_actvt_display.
    CATCH cx_bc400_no_auth .
      MESSAGE e046(bc400) WITH pa_car.
  ENDTRY.

START-OF-SELECTION.

  TRY.
      CALL METHOD cl_bc400_flightmodel=>get_flights_range
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
      EXPORTING
        iv_carrid  = pa_car
        it_connid  = so_con[] "brackets needed for so_con
      IMPORTING
        et_flights = gt_flights.
  CATCH cx_bc400_no_data.
    WRITE / 'No flights for the selected flight connection'(non).
ENDTRY.

LOOP AT gt_flights INTO gs_flight.

  NEW-LINE.

  IF gs_flight-percentage >= gc_limit_red.
    WRITE  icon_red_light AS ICON.
  ELSEIF gs_flight-percentage >= gc_limit_yellow.
    WRITE  icon_yellow_light AS ICON.
  ELSE.
    WRITE  icon_green_light AS ICON.
  ENDIF.

  WRITE:  gs_flight-carrid    COLOR COL_KEY,
          gs_flight-connid   COLOR COL_KEY,
          gs_flight-fldate   COLOR COL_KEY,
          gs_flight-seatsmax,
          gs_flight-seatsocc,
          gs_flight-percentage.
ENDLOOP.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Implement the events of ABAP reports

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. You can use the _____ to adapt the default list interface to your own needs.

   Choose the correct answer.

   - [ ] A WRITE statement
   - [ ] B Menu Painter
   - [ ] C Pattern function

2. Which of the following are standard functionality on a selection screen?

   Choose the correct answers.

   - [ ] A Singular capability
   - [ ] B Type check
   - [ ] C Variants
   - [ ] D Value entry

3. Which of the following statements is used for defining an input variable?

   Choose the correct answer.

   - [ ] A DATA
   - [ ] B VALUE
   - [ ] C PARAMETERS
   - [ ] D DEFAULT

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following are the characteristics of an event block?

   Choose the correct answers.

   - [ ] A Ends by beginning the next processing block
   - [ ] B Can be nested
   - [ ] C Existence not absolutely necessary
   - [ ] D Sequence of event blocks important

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. You can use the _____ to adapt the default list interface to your own needs.

   Choose the correct answer.

   - [ ] A WRITE statement
   - [x] B Menu Painter
   - [ ] C Pattern function

2. Which of the following are standard functionality on a selection screen?

   Choose the correct answers.

   - [ ] A Singular capability
   - [x] B Type check
   - [x] C Variants
   - [x] D Value entry

3. Which of the following statements is used for defining an input variable?

   Choose the correct answer.

   - [ ] A DATA
   - [ ] B VALUE
   - [x] C PARAMETERS
   - [ ] D DEFAULT

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following are the characteristics of an event block?

   Choose the correct answers.

   - [x] A Ends by beginning the next processing block
   - [ ] B Can be nested
   - [x] C Existence not absolutely necessary
   - [ ] D Sequence of event blocks important

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 12 Program Analysis Tools

Lesson 1

Using the Code Inspector 528

UNIT OBJECTIVES

- Explain the purpose and use of the Code Inspector
- Use the Code Inspector for simple analysis of programs

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 12
Lesson 1

Using the Code Inspector

LESSON OVERVIEW
This lesson explains the basic function and purpose of the Code Inspector. This is only a brief introduction to the tool. You will find more information in the relevant documentation or by attending the advanced courses.

Business Example

You need to check your programs for performance, typical semantic programming errors, and security gaps. For this reason, you require the following knowledge:

- An understanding of the purpose and use of the Code Inspector
- An understanding of the most important propertie of the Code Inspector
- An understanding of how to use the Code Inspector for simple analysis of your programs

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain the purpose and use of the Code Inspector
- Use the Code Inspector for simple analysis of programs

The Code Inspector

The Code Inspector offers you the option of analyzing your programs with regard to performance, security, and typical semantic errors. Here are a few examples of what you can check:

Typical Performance Issues:
- Are indexes used for database access?
- Are SELECT statements embedded in loops?

Typical Security Issues:
- Is data read from a client other than the login client?
- Is the database table or the WHERE clause dynamically specified in the SELECT statement?

Typical Semantic Errors:
- Is the sy-subrc field checked after each AUTHORITY-CHECK statement?
- Is a client actually specified for CLIENT SPECIFIED?
- Are several messages of type E (E messages) sent in direct succession?

© Copyright . All rights reserved.

Prohibida su reproduccion

Calling of the Code Inspector

Figure 236: Calling the Code Inspector

There are several different ways to call the Code Inspector.

The figure shows one possibility. Call the Code Inspector from the context menu of the object list by choosing *Check → Code Inspector* and carry out a standard inspection. You will get the same call if you choose *Function Module → Check → Code Inspector*.

© Copyright . All rights reserved.

Prohibida su reproduccion

Usage of the Code Inspector

Figure 237: Inspection result

As the result of an inspection, you receive a list of error and warning messages. The *information* pushbutton that belongs to the message displays a detailed error description as well as suggestions for improvement. However, if you double-click the error text, you branch to the corresponding program statement.

In certain instances, it is practical, or even essential, to write ABAP source code that the Code Inspector would query, such as a transaction call. You can provide the comment "#EC * for such commands, which is the same action to be taken with an extended syntax check.

When you initiate a check from inside the ABAP Editor, the Code Inspector uses a default check variant with a predefined set of checks to be performed. You can overwrite this default check variant by creating a new check variant with the name DEFAULT.

However, this new check variant will override the standard variant for your user in all clients. If you delete your default check variant, the standard variant is automatically used for future checks. For this reason, it is recommended that you define your own check variants instead of overwriting the default variant.

© Copyright . All rights reserved.

Prohibida su reproduccion

Definition of Check Variants, Object Sets, and Inspections

Figure 238: Advanced Code Inspector Check

To define individual checks, start the Code Inspector with transaction code `SCI`, or from the *SAP Easy Access* menu by choosing *Tools → ABAP Workbench → Test → Code Inspector*.

The initial screen contains the following areas:

- Check Variant

  The check variant determines which checks to perform in the system, such as the programming conventions or a performance check.

- Object Set

  The object set determines which Repository objects to check.

- Inspection

  Inspection refers to the name of the actual check. To save the inspection and its results or execute it in the background, you need to give it a name. If you do not enter a name for the inspection, the system does not save the result.

Inspection uses a previously defined check variant and object set. However, it is also possible to create temporary object sets and check variants that are only valid for one inspection and cannot be re-used in other inspections.

> Hint:
> You can create check variants, object sets, and inspections either as private or public. To switch between these two categories, use the pushbutton that is placed in the input field. Note that only you can use your private objects whereas public objects are available to all users of the system.

© Copyright . All rights reserved.

Prohibida su reproduccion

A check variant consists of one or more check categories, which in turn consist of one or more single checks (inspections). You can parameterize these single checks by using a keyword or an indicator for a specific partial aspect of the check. In most cases, single checks investigate a specific object type, such as the "Check of Table Attributes", which only examines DDIC tables.

Check Variant

Figure 239: Check variant

The system assigns single checks to various check categories.

The most important check categories are as follows:

- *General Checks*

  General checks contain data formatting, such as the list of table names from SELECT statements.

- *Performance Checks*

  Performance checks contain checks for performance and resource usage, such as an analysis of the WHERE condition for SELECT or UPDATE and DELETE, SELECT statements that bypass the table buffer, and low-performance access to internal tables.

- *Security Checks*

  Security checks contain checks of critical statements, cross-client queries, and insufficient authority checks.

- *Syntax Check* and *Generation*

  Syntax check and generation contain the ABAP syntax check, an extended program check, and generation.

- *Programming Conventions*

  Programming conventions contain checks of naming conventions.

- *Search Functions*

© Copyright . All rights reserved.

Prohibida su reproduccion

Search functions contain searches for tokens (words) and statements in ABAP source code.

LESSON SUMMARY

You should now be able to:

- Explain the purpose and use of the Code Inspector
- Use the Code Inspector for simple analysis of programs

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. Which of the following statements about the Code Inspector are true?

   Choose the correct answers.

   - [ ] A You can only use the DEFAULT check variant.
   - [ ] B You can create a check variant to define the details of what to check.
   - [ ] C Standard inspection is carried out when check is performed through context menu from the Object Navigator.
   - [ ] D You can define check variants, object sets, and inspections using transaction SCI.

2. Which aspects does the Code Inspector take into consideration when examining a program?

   Choose the correct answers.

   - [ ] A Syntax check
   - [ ] B Typical semantic errors (for example, AUTHORITY-CHECK statement without subsequent SY-SUBRC check)
   - [ ] C Performance (for example, nested SELECT statements)
   - [ ] D Security (for example, cross-client data accesses)
   - [ ] E Formatting of the source code (for example, indenting of the program lines within loops)

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. Which of the following statements about the Code Inspector are true?

   Choose the correct answers.

   - [ ] A You can only use the DEFAULT check variant.
   - [x] B You can create a check variant to define the details of what to check.
   - [x] C Standard inspection is carried out when check is performed through context menu from the Object Navigator.
   - [x] D You can define check variants, object sets, and inspections using transaction SCI.

2. Which aspects does the Code Inspector take into consideration when examining a program?

   Choose the correct answers.

   - [ ] A Syntax check
   - [x] B Typical semantic errors (for example, AUTHORITY-CHECK statement without subsequent SY-SUBRC check)
   - [x] C Performance (for example, nested SELECT statements)
   - [x] D Security (for example, cross-client data accesses)
   - [ ] E Formatting of the source code (for example, indenting of the program lines within loops)

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 13 Program Calls and Memory Management

Lesson 1

Calling Programs Synchronously 538
Exercise 31: Implement Program Calls 547

Lesson 2

Understanding the ABAP Runtime and Memory Management 554
Exercise 32: Use ABAP Memory and SAP Memory 575

UNIT OBJECTIVES

- Call programs synchronously
- Generate and activate programs
- Explain the user session memory management
- Transfer data between programs
- Explain the memory management of deep data objects
- Use boxed components

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 13
Lesson 1

Calling Programs Synchronously

LESSON OVERVIEW
This lesson discusses various techniques for calling complete programs synchronously. It also discusses how to insert and restart programs, and how to pass data on to the called program during calls.

Business Example

You want to call existing programs from within a new application. To do so, you must familiarize yourself with the techniques for calling programs. For this reason, you require the following knowledge:

- How programs are called from within other programs
- An understanding of the various options for passing data on to the called program during program calls

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Call programs synchronously

Synchronous Program Calls

Figure 240: Synchronous Program Calls

You have two options when calling another program synchronously:

© Copyright . All rights reserved.

Prohibida su reproduccion

- Insert

  The process flow of the calling program pauses while the called program executes, and resumes when the called program is complete.

- New start

  The calling program terminates and the called program starts.

Complete programs can only run in a sequence within a single user session. SAP refers to this processing scheme as sequential calls. To run activities in parallel, you must use function modules.

> Note:
> More information on asynchronous execution is available in other training courses and in the ABAP keyword documentation for the CALL FUNCTION ... STARTING NEW TASK ... statement.

Synchronous Calls of Executable Programs

Figure 241: Calling an Executable Program

To start an executable (type 1) program, use the SUBMIT statement. If you use the VIA SELECTION-SCREEN addition, the system displays the standard selection screen.

If you use the AND RETURN addition, the system resumes processing with the first statement of the calling program after the SUBMIT statement, once the execution of the called program finishes.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Note:
> For more information, see the ABAP keyword documentation for the SUBMIT statement.

Passing Data Using the Program Interface

Figure 242: Passing Data Using the Program Interface

When you call ABAP programs that have a standard selection screen, you can pass data for the input fields in the call.

Data for input fields can be passed in the following ways:

- By specifying a variant for the selection screen when you call the program
- By specifying specific values for the input fields when you call the program

© Copyright . All rights reserved.

Prohibida su reproduccion

Pre-assignment of Input Fields

Figure 243: Pre-assignment of Input Fields

The WITH addition in the SUBMIT statement allows you to assign values to the fields on a standard selection screen. The relational operators such as EQ, NE, and so on, and the inclusion operators I and E are similar in meaning to SELECT OPTIONS.

If you want to set several selections for a selection option, you can use the RANGE type instead of individual WITH additions. Using the RANGE type creates a selection table that you can fill, as though it were a selection option. You then pass the table to the executable program.

To display the standard selection screen when you call the program, use the VIA SELECTION-SCREEN addition. Use the statement pattern in the ABAP Editor to insert a program call via SUBMIT.

The pattern automatically supplies the names of the parameters and selection options that are available on the standard selection screen.

© Copyright . All rights reserved.

Prohibida su reproduccion

Synchronous Transaction Calls

Figure 244: Calling a Transaction

If you use the LEAVE TO TRANSACTION 'T_CODE' statement, the system terminates the current program and starts the transaction with transaction code T_CODE. This statement is the equivalent of entering /NT_CODE in the command field.

With the CALL TRANSACTION ‘T_CODE’ statement, you can pause the execution of the calling program until the called program finishes its execution.

You can use the LEAVE PROGRAM statement to force the termination of a program. For cases in which this statement executes in a program called by CALL TRANSACTION 'T_CODE' or SUBMIT prog_name AND RETURN, you continue with the next statement after this call. In all other cases, the system returns to the application menu from which the user started the program.

If you use the ... AND SKIP FIRST SCREEN addition, the system does not display the contents of the first screen in the transaction.

> Caution:
> The system does process the flow logic.

When you start a transaction using CALL TRANSACTION 'T_CODE' and that transaction uses update techniques, you can use the UPDATE ... addition to specify the update technique- asynchronous (default), synchronous, or local, that the program can use.

© Copyright . All rights reserved.

Prohibida su reproduccion

Pass Data Using an Internal Table

Figure 245: Passing Data Using an Internal Table

When you call a transaction using the CALL TRANSACTION 'T_CODE' USING bi_itab statement, you can run transaction T_CODE using the values from internal table bi_itab in the screen fields.

This internal table must have the global structure type BDCDATA, and you should fill it accordingly.

The MODE addition allows you to specify whether the screen displays all its contents (A – default setting) only when an error occurs (E), or not at all (N).

You can use the MESSAGES INTO addition to specify an internal table into which any system messages can be written. The corresponding internal table must have the type of global structure BDCMSGCOLL.

You can use the return code (`SY-SUBRC`), to verify that the system executes the transaction successfully.

This technique of passing data using an internal table is useful in the following cases:

- Processing in dialog, but not filling the input fields using GET parameters
- Processing the transaction in the background, in which case where you must pass the function codes in the table as well

Fields in the Global Type BDCDATA

The fields in the global type BDCDATA are as follows:

Field Name	Length	Description	Note When Filling
program	40	Program name	Only in first record for screen

© Copyright . All rights reserved.

Prohibida su reproduccion

Field Name	Length	Description	Note When Filling
dynpro	4	Screen number	Only in first record for screen
dynbegin	1	First record	‘x’ for first record for screen, otherwise ‘ ‘
fnam	132	Field name	
fval	132	Field value	Case sensitive

The information you require to fill the internal table in batch input format is as follows:

- Identify each screen that you want to fill and process automatically in the transaction by a line in which only the PROGRAM, DYNPRO, and DYNBEGIN fields are filled.
- After the record identifies the screen, use a new BDCDATA record for each field you want to fill. These records use the FNAM and FVAL fields. You can provide values for the following types of fields: input or output fields (with data), the command field BDC_OKCODE (with a function code), or the field for cursor positioning BDC_CURSOR (with a field name).

The technique of passing data using internal table is also one of the possible ways of transferring data from non-SAP systems. When doing so, the internal table has to contain the complete input that is needed for an execution of the transaction in background.

Data Transfer Using an Internal Table, Application Example

Figure 246: Data Transfer Using an Internal Table, Application Example

The figure shows an example of the transaction `BC402_CALD_CTA_U`. When the user creates a new customer entry, the program calls transaction `BC402_CALD_CRE_CUST`. This transaction does not implement an import from ABAP memory, and the transaction input fields are not set as

© Copyright . All rights reserved.

Prohibida su reproduccion

GET parameters. After the CALL TRANSACTION statement, the system passes the customer data using an internal table and processes them in background.

If the operation is successful, you can enter new customer record in the waiting list. The figure shows the internal table in that you fill, in BDCDATA format.

At runtime, CURRENT_NAME stands for the customer name from the input field and CURRENT_CITY stands for the city. Address the command field using BDC_OKCODE, which captures the function code that the user triggers by choosing a function key, button or menu option during the dialog flow. The user can also enter a code directly in the command field.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 13
Exercise 31

Implement Program Calls

Business Example

An existing program displays a list of data. You want to enhance the program by enabling users to access an existing report or dialog transaction by double-clicking on a line.

You need to implement calls of executable programs, implement transaction calls, and pass data on to executable programs and transactions.

Task 1

Copy program *BC402_PCT_CONN_LIST* and all its components to the name **`ZBC402_##_CONN_LIST`**, where **`##`** is your group number. Familiarize yourself with the program and how it works.

1. Copy the template.
2. Activate and test the program.
3. Analyze the source code. How is the display of the flight connections implemented?

   _____

   _____

   _____

4. The program is already set to handle double-click events. Which processing block do you have to enhance to do so?

   _____

   _____

   _____

Task 2

Implement the handler method that runs when a user double-clicks a flight connection. Make sure a list of flights is returned for the chosen connection if the user double-clicks in the *CARRID* column. To do so, use executable program *BC402_INS_FLIGHT_LIST* or your own program, **`ZBC402_##_FLIGHT_LIST`**.

1. Implement the on_double_click method of the local class lcl_handler. The COLUMN import parameter contains the information as to which column the user clicked. Use the SUBMIT statement to call the executable program.
2. Make sure users can return to the display of flight connections after displaying the dates.

© Copyright . All rights reserved.

Prohibida su reproduccion

3. Supply the interface of the called program with the key values of the chosen flight connection. To do so, determine the names of the input fields on the selection screen.

Task 3

Implement the handler method that runs when a user double-clicks a flight connection. Make sure details for the airline are output for the chosen connection if the user double-clicks in the *CARRID* column. To do so, call dialog transaction `BC402MCAR`.

1. Call the transaction to insert it synchronously.
2. Pass the airline ID of the chosen connection on to the transaction. To do so, use the USING addition. Determine the program name, screen number, and field label of the input field on the initial screen of transaction `BC402MCAR`.
3. Pass the function code that is linked with the pushbutton on the initial screen on to the transaction.
4. Make sure the transaction is processed without screen display. To do so, use the MODE addition with a suitable value.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Program Calls

Business Example

An existing program displays a list of data. You want to enhance the program by enabling users to access an existing report or dialog transaction by double-clicking on a line.

You need to implement calls of executable programs, implement transaction calls, and pass data on to executable programs and transactions.

Task 1

Copy program *BC402_PCT_CONN_LIST* and all its components to the name **`ZBC402_##_CONN_LIST`**, where **`##`** is your group number. Familiarize yourself with the program and how it works.

1. Copy the template.
   a) Carry out this step in the usual manner.
2. Activate and test the program.
   a) Carry out this step in the usual manner.
3. Analyze the source code. How is the display of the flight connections implemented?

   The program uses an instance of class cl_salv_table. The static factory method is called to generate an instance of the class and pass it to the table with the data. The display method shows the data in an ALV grid control.
4. The program is already set to handle double-click events. Which processing block do you have to enhance to do so?

   The on_double_click method of the local class lcl_handler.

Task 2

Implement the handler method that runs when a user double-clicks a flight connection. Make sure a list of flights is returned for the chosen connection if the user double-clicks in the *CARRID* column. To do so, use executable program *BC402_INS_FLIGHT_LIST* or your own program, **`ZBC402_##_FLIGHT_LIST`**.

1. Implement the on_double_click method of the local class lcl_handler. The COLUMN import parameter contains the information as to which column the user clicked. Use the SUBMIT statement to call the executable program.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) See the source code excerpt from the model solution.

2. Make sure users can return to the display of flight connections after displaying the dates.
   a) AND RETURN addition; see source code excerpt from the model solution.
3. Supply the interface of the called program with the key values of the chosen flight connection. To do so, determine the names of the input fields on the selection screen.
   a) See the source code excerpt from the model solution.

Task 3

Implement the handler method that runs when a user double-clicks a flight connection. Make sure details for the airline are output for the chosen connection if the user double-clicks in the *CARRID* column. To do so, call dialog transaction `BC402MCAR`.

1. Call the transaction to insert it synchronously.
   a) CALL TRANSACTION. See source code excerpt from the model solution.
2. Pass the airline ID of the chosen connection on to the transaction. To do so, use the USING addition. Determine the program name, screen number, and field label of the input field on the initial screen of transaction `BC402MCAR`.
   a) See the source code excerpt from the model solution.
3. Pass the function code that is linked with the pushbutton on the initial screen on to the transaction.
   a) See the source code excerpt from the model solution.
4. Make sure the transaction is processed without screen display. To do so, use the MODE addition with a suitable value.
   a) A source code excerpt from the model solution appears as follows:

```
REPORT  bc402_pcs_conn_list MESSAGE-ID bc402.

TYPES: BEGIN OF gty_s_conn,
        carrid   TYPE spfli-carrid,
        connid   TYPE spfli-connid,
        cityfrom TYPE spfli-cityfrom,
        cityto   TYPE spfli-cityto,
        deptime  TYPE spfli-deptime,
        arrtime  TYPE spfli-arrtime,
        period   TYPE spfli-period,
      END OF gty_s_conn.

TYPES gty_t_conn TYPE STANDARD TABLE OF gty_s_conn
                 WITH NON-UNIQUE DEFAULT KEY.

DATA gt_conn TYPE gty_t_conn.

DATA gv_msg TYPE string.

DATA: go_alv TYPE REF TO cl_salv_table,
      go_evt TYPE REF TO cl_salv_events_table,
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
      go_fct TYPE REF TO cl_salv_functions_list,
      gx_msg TYPE REF TO cx_salv_msg.


*---------------------------------------------------*
*       CLASS lcl_handler DEFINITION
*---------------------------------------------------*
*
*---------------------------------------------------*
CLASS lcl_handler DEFINITION.

  PUBLIC SECTION.

    CLASS-METHODS:
          on_double_click FOR EVENT double_click
                                  OF if_salv_events_actions_table
                          IMPORTING row column.

ENDCLASS.                    "lcl_handler DEFINITION


*-----------------------------------------------------*
*       CLASS lcl_handler IMPLEMENTATION
*-----------------------------------------------------*
*
*-----------------------------------------------------*
CLASS lcl_handler IMPLEMENTATION.

  METHOD on_double_click.

   DATA: lt_bdc TYPE TABLE OF bdcdata,
         ls_bdc TYPE          bdcdata.

    DATA ls_conn LIKE LINE OF gt_conn.

    READ TABLE gt_conn INTO ls_conn INDEX row.

    CASE column.
      WHEN 'CARRID'.

        ls_bdc-program  = 'SAPMBC402_IND_CARRIER'.
        ls_bdc-dynpro   = '0100'.
        ls_bdc-dynbegin = 'X'.
        APPEND ls_bdc TO lt_bdc.

        CLEAR ls_bdc.
        ls_bdc-fnam = 'BC402_S_CARRIER-CARRID'.
        ls_bdc-fval = ls_conn-carrid.
        APPEND ls_bdc TO lt_bdc.

        CLEAR ls_bdc.
        ls_bdc-fnam = 'BDC_OKCODE'.
        ls_bdc-fval = 'GO'.
        APPEND ls_bdc TO lt_bdc.

        CALL TRANSACTION 'BC402MCAR'
             USING lt_bdc
             MODE  'E'.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
      WHEN OTHERS.

        SUBMIT bc402_ins_flight_list
               AND RETURN
               WITH pa_car EQ ls_conn-carrid
               WITH so_con EQ ls_conn-connid.

    ENDCASE.

  ENDMETHOD.                    "on_double_click

ENDCLASS.                    "lcl_handler IMPLEMENTATION


*---------------------------------------------------*
START-OF-SELECTION.

  SELECT carrid connid cityfrom cityto
         deptime arrtime period
    FROM spfli
    INTO TABLE gt_conn.

  TRY.
      CALL METHOD cl_salv_table=>factory
        IMPORTING
          r_salv_table = go_alv
        CHANGING
          t_table      = gt_conn.
    CATCH cx_salv_msg INTO gx_msg.
      gv_msg = gx_msg->get_text( ).
      MESSAGE gv_msg TYPE 'E'.
  ENDTRY.

  go_evt = go_alv->get_event( ).

  SET HANDLER lcl_handler=>on_double_click
          FOR go_evt.

  go_alv->display( ).
```


© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Call programs synchronously

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 13
Lesson 2

Understanding the ABAP Runtime and Memory Management

LESSON OVERVIEW

This lesson explains in detail how the ABAP runtime environment executes a program. The lesson also explains ABAP memory and SAP memory and their usage techniques for transient data exchange across program boundaries.

Additionally, this lesson explains how data objects with variable length, known as dynamic data objects, are managed in memory.

Business Example

You want to develop complex applications that include program calls, modularization, and dynamic data objects. To ensure your applications have a robust, high-performance design, you want to find out how the runtime environment and memory management work in detail. You also want to learn about the options presented by the ABAP memory and SAP memory to exchange data across program boundaries during program calls. For this reason, you require the following knowledge:

- An understanding of how the ABAP runtime environment executes programs
- An understanding of the importance of the runtime object
- An understanding of the roll area and program execution area (PXA)
- An understanding of memory management in the ABAP runtime environment
- An understanding of how dynamic data objects are managed
- An understanding of how Boxed Components are managed and when they should be used

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Generate and activate programs
- Explain the user session memory management
- Transfer data between programs
- Explain the memory management of deep data objects
- Use boxed components

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP Compiler and Runtime Objects

Figure 247: Developed Version

Development objects are stored in the Repository in the form in which they are developed. These are the only versions of the development objects that can be displayed in the *Workbench*.

To run a program in the ABAP runtime environment, you must first generate the program, using the *ABAP Compiler*.

© Copyright . All rights reserved.

Prohibida su reproduccion

Generated Version

Figure 248: Generated Version

When you generate a development object, the system generates a runtime object, or LOAD. The LOAD consists largely of byte code statements that are linked to C functions. To avoid generating programs every time they run, save program LOADs in the database and buffer them on the application server.

The system generates the LOAD automatically in the following situations:

- When the program is called and no LOAD exists yet.
- When the LOAD in the database or program buffer is obsolete

You can also generate programs explicitly, for example, in the *ABAP Editor* using menu path *Program → Generate*.

A saved or buffered LOAD is considered obsolete under either of the following conditions:

- The developed version of the program is changed.
- The object that the program uses is changed in *Data Dictionary*.

© Copyright . All rights reserved.

Prohibida su reproduccion

Hint:
Transports between SAP systems always involve developed versions. After being imported into a target system, the system discards all affected runtime objects.

The system does not generate new LOADs automatically. They are generated at the first instance, when a user calls the corresponding program (directly or indirectly). After an import, users may temporarily experience wait times and low performance until all the required LOADS are generated again. To avoid this, you can start the SAP load generator (transaction code SGEN) after the import.

The Concept of Inactive Sources

Figure 249: Active and Inactive Versions

When you change a development object and save it, the system creates an active version and an inactive version.

Hint:
Saving an inactive version has no effect on the program LOAD. Accordingly, the syntax of the inactive version does not necessarily have to be correct. You can also save an interim, non-executable version of the development object inactively. However, the active version remains available as an executable version.

You can toggle the view between the active and inactive versions, but you can only edit the inactive version. The system performs a syntax check on the version in development (the inactive version), but not on the generated version. When running source code for executable programs from the ABAP Editor, the system refers back to the inactive version. In this case, the system

© Copyright . All rights reserved.

Prohibida su reproduccion

creates a temporary generated version that is not buffered or stored in the database. In all other cases, the system always accesses the active version when you generate the program.

Program Activation

Figure 250: Activating a Program

All inactive objects that the system assigns to your user ID are listed in your work list.

The following steps occur when you activate an object:

1. The object is saved, that is, a new inactive version is created. This is subject to a syntax check.
2. The system overwrites the current active version. The previously inactive version is now the active version, which means there is no longer an inactive version.
3. The system creates a new runtime version and updates the LOAD.

© Copyright . All rights reserved.

Prohibida su reproduccion

PXA and Roll Area

Figure 251: Loading a Runtime Object to Working Memory

When the runtime system generates a new runtime object (the LOAD) of a program or loads it from the database, it places the program in the working memory of the application server. In the process, the runtime system treats the modifiable parts of the program differently than the non-modifiable parts.

The non-modifiable parts of a program are as follows:

- Byte code for statements
- Values of constants and literals
- Program texts
- Screen definitions

The modifiable parts of a program are the Data objects (variable).

The system stores the non-modifiable parts of the program in the PXA and buffers to the maximum. This memory area is shared by all users on the application server and exists only once on each server. After the system executes a program, the program's byte code remains buffered in the PXA to the maximum extent. Buffering prevents the system from loading the code from the database again during the next execution.

The modifiable parts of the program are created in the memory during every execution, in a memory space called the roll area. When the program executes several times concurrently (by the same user or different users), each execution has its own roll area. This ensures that the system executes the programs independently of one another.

© Copyright . All rights reserved.

Prohibida su reproduccion

Memory Management from the Program Perspective

Figure 252: Logical Memory Model

The figure shows the way in which main memory is organized from the program's perspective is represented in the figure.

The distinction between internal and external sessions is as follows:

- External session (main session)

  An external session corresponds to a SAPGUI window. You can create a new session by choosing *System → New Session* or by calling /o<t_code>. Up to 16 external sessions are possible in SAP NetWeaver 7.0 and later. The maximum number of sessions available in a given system is defined by system parameter rdisp/max_alt_modes, which has a default value of 6.

- Internal session

  External sessions are subdivided into internal sessions (placed in a stack). Each call of an ABAP program (SUBMIT, CALL TRANSACTION, and so on) generates a new internal session. A maximum of nine internal sessions can exist in each external session. Internal session must belong to the same call stack.

Hint:
The system reserves one roll area for each internal session.

© Copyright . All rights reserved.

Prohibida su reproduccion

Program Groups

Figure 253: Program Groups

When a program calls external modularization units, the system loads the corresponding programs into the same internal session. The programs within an internal session form program groups. When the system opens an internal session with a program call, the system always creates the main program group. There can be one or more additional program groups. Each program group contains one main program, and other programs as well.

The program data objects are only visible within its program group, which means programs in different program groups do not share any data objects. All programs and objects of an internal session can use instances of classes. An object continues to exist as long as there are users, that is, references for it.

© Copyright . All rights reserved.

Prohibida su reproduccion

Memory Management for Program Calls

Figure 254: Before Inserting a Program

The figure illustrates how a stack of internal sessions inside an external session changes with various program calls.

When inserting a program, the starting point for the first case is the execution of the SUBMIT ... AND RETURN or CALLTRANSACTION statements within a program.

Inserted Program

Figure 255: The Inserted Program Runs

© Copyright . All rights reserved.

Prohibida su reproduccion

When you insert a program, the system creates a new internal session which contains the new program context. The system places the new session in the stack. The program context of the calling program remains in the stack.

After the Inserted Program Ends

Figure 256: After the Inserted Program Ends

When the called program is completed, the system deletes the called program's internal session (the top one in the stack). The system then begins processing the next-highest internal session in the stack.

Before Starting a New Executable Program

Figure 257: Before Starting a New Executable Program

© Copyright . All rights reserved.

Prohibida su reproduccion

The system makes a distinction between calling an executable program and calling a transaction, with regard to memory areas.

The starting point for the second case is the execution of a SUBMIT statement within a program.

The New Executable Program Runs

Figure 258: The New Executable Program Runs

When you start an executable program with the SUBMIT statement, the following actions occur:

- The system ends the calling program.
- The system destroys the internal session of the program that you are terminating (the top program in the stack).
- The system creates a new internal session, which contains the program context of the called program. The system then places the new session in the stack and retains any previously created program contexts. The topmost internal session in the stack is thereby replaced.

© Copyright . All rights reserved.

Prohibida su reproduccion

Before Starting a New Transaction

Figure 259: Before Starting a New Transaction

The starting point for the third case is the execution of a LEAVE TO TRANSACTION statement within a program.

The New Transaction Runs

Figure 260: The New Transaction Runs

When you start a program using its transaction code, the system destroys all internal sessions in the stack. The system then creates a new internal session, which contains the program context of the called program.

© Copyright . All rights reserved.

Prohibida su reproduccion

The system initializes the complete stack of internal sessions, which also means it initializes the ABAP memory after the call.

Memory Management for Calls of Function Modules and Methods

Figure 261: Main or Additional Program Groups

The figure explains the impact of external procedure calls on Function Pools (type F) and Class Pools (type K).

Cross-program modularization, which calls function modules and uses global classes, is not implemented using internal sessions. Instead, the program groups play a central role.

When you open an internal session, the system always creates the main program group.

You can also create any number of additional program groups in the same internal session. Each program group has only one main program.

An additional program group is created in the following cases:

- When the program calls a function module from a function group that has not yet been loaded.
- When the program calls a method of a global class that has not yet been loaded.

The system loads each function group and each global class only once into a given internal session.

© Copyright . All rights reserved.

Prohibida su reproduccion

Same Function Group in Different Internal Sessions

Figure 262: Same Function Group in Different Internal Sessions

When two programs running in different internal sessions call up function modules from the same function group, the system loads the function group into each of the internal sessions separately. The system creates the global data of the function group separately in each internal session. Therefore, there is no possibility or risk of the two main programs exchanging data through the function group.

© Copyright . All rights reserved.

Prohibida su reproduccion

Same Class in Different Internal Sessions

Figure 263: Same Class in Different Internal Sessions

When programs that run in different internal sessions use the same global class, the global class is created separately in each internal session. The static attributes of the class can have different values in different internal sessions.

Same Function Group or Class Within an Internal Session

Figure 264: Same Function Group or Class Within an Internal Session

© Copyright . All rights reserved.

Prohibida su reproduccion

When two programs run in the same internal session, for example, an executable program and the AB function group it addresses, and both programs address the same XY function group, then the system loads the function group XY in the internal session once. Therefore, the main program and function group AB can exchange data using function group XY. The same applies for global classes.

Caution:
In practice, this situation can result in frequent errors because the main program and function group AB (or class CL_A) affect each other inadvertently. The main program may buffer data in function group XY, for example, but function group AB or class CL_A can unintentionally overwrite the data or initialize it. This can result in program termination and even data inconsistency.

The causes of such errors are difficult to find, particularly when the program does not use the function group AB or class CL_A directly and uses other function groups or classes instead.

Hint:
When the shared program is a global class (class CL_B in the figure), there is only a risk of unintended interaction with regard to the static attributes. The two programs used see only the same instances (and therefore the instance attribute values) of a class when they specifically exchange references to these instances.

An elegant way of avoiding unintended interaction for global classes is to use instance attributes instead of static attributes. Also, make sure that every user of the class generates a separate instance.

Several Programs in the Same Program Group

Figure 265: Several Programs in the Same Program Group

The figure explains the impact of external subroutine calls on Executable programs (type 1), Modul pools (type M) and subroutine pools (type S).

© Copyright . All rights reserved.

Prohibida su reproduccion

When the user calls a subroutine externally, the system does not generate any new program group in the internal session. Instead, it loads the main program of this subroutine into the calling program's existing program group.

> Hint:
> Subroutines of a function group are an exception. When a subroutine of a function group is called externally, the system loads the corresponding function group into an additional program group.

When two programs are in the same program group, the developer must observe the following constraints:

- Within a program group, all programs share the interface work areas that are declared with TABLES, NODES, and COMMON PART.
- Within a program group, CALL SCREEN can only be used to call screens of the main program.

> Caution:
> When you use external subroutine calls, you always run the risk that data objects from one program can be shared with other programs. While you may do this purposely in some exceptional cases, you should avoid it wherever possible considering the aspects of encapsulation and ease of maintenance. Therefore, avoid using external subroutine calls altogether.

Overview of Data Transfer Options

Figure 266: Data Transfer Between Programs – Overview

© Copyright . All rights reserved.

Prohibida su reproduccion

When programs run in different program groups, internal sessions, or external sessions, they do not share any data objects. However, various options are available for exchanging data between these programs.

Some techniques for passing data are as follows:

- Interface of the called program
- ABAP memory
- SAP memory
- Database tables
- Local files on the presentation server

> Note:
> For more information about passing data using database tables or the shared buffer, see the ABAP keyword documentation for the EXPORT and IMPORT statements.
>
> For more information about transferring data between an ABAP program and your presentation server, see the documentation of function modules WS_UPLOAD and WS_DOWNLOAD. You can also use the methods of class CL_GUI_FRONTEND_SERVICES for this purpose.

ABAP Memory and SAP Memory

Figure 267: Range of ABAP Memory and SAP Memory

When you need to transfer data between two different programs but cannot specify the data as an addition in the program call, you can use the SAP memory and the ABAP memory to transfer data between programs.

Consider the following while using SAP memory and ABAP memory to transfer data between programs:

© Copyright . All rights reserved.

Prohibida su reproduccion

- The SAP memory is a user-specific memory area for storing field values. It is therefore not useful for passing data between internal sessions.

  The system retains values in the SAP memory for the duration of the user's terminal session. You can also use the SAP memory between sessions in the same terminal session. All external sessions of a user session can access the SAP memory. You can use the contents of the SAP memory as default values for screen fields.

- The ABAP memory is also user specific. There is a local ABAP memory for each external session. You can use a local ABAP memory to exchange any ABAP data objects (such as fields, structures, internal tables, and complex objects) between the internal sessions in any one external session.

  When the user exits an external session (in the command field), the system automatically initializes or releases the corresponding ABAP memory.

Access to ABAP Memory

Figure 268: Passing Data Using ABAP Memory

The EXPORT ... TO MEMORY statement copies any number of ABAP data objects with their current values (data cluster) to the ABAP memory. The ID addition, which can be a maximum of 60 characters long, enables you to identify different clusters.

When you use a new EXPORT TO MEMORY statement for an existing data cluster, the new statement overwrites the old one. The IMPORT ... FROM MEMORY ... statement allows you to copy data from the ABAP memory to the corresponding fields of your ABAP program.

You can also restrict the selection to a part of the data cluster in the IMPORT statement. The variables into which you want to read data from the cluster in ABAP memory must have the same types, in both the exporting and the importing programs.

You use the FREE MEMORY ID ... statement to release a data cluster explicitly.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Caution:
> When you call programs using transaction code, you can only use the ABAP memory to pass data to the transaction in the insert case (CALL TRANSACTION).

Access to SAP Memory

Figure 269: Passing Parameters Using SAP Memory

You can define memory areas (called parameters) in the SAP memory in different ways, such as the following:

- You can create input or output fields with reference to *ABAP Dictionary*.

  These fields take the parameter names of the data elements to which they refer. Alternatively, you can enter a name in the attributes of the input or output fields. Then, you can also choose whether you want to transfer the entries from the field to the parameter (SET) or fill out the input field with the value from the parameter (GET). To find out about the names of the parameters assigned to input fields, display the field help using the F1 key and choose the *Technical info* pushbutton.

- You can also fill out the memory areas directly using the SET PARAMETER ID 'PAR_ID' FIELD var. statement and read from them using the GET PARAMETER ID 'PAR_ID' FIELD var. statement.

- You can define parameters in *Object Navigator* and let the user enter the values.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Use ABAP Memory and SAP Memory

Business Example

An existing program selects data based on user entries and outputs it in a list. You call this program from within another program. Instead of displaying the data directly with the called report, you want to load the data you select into the SAP memory, and process it further in the calling program. You also want to use parameters in the ABAP memory to pre assign fields when calling a transaction.

Template

BC402_INS_FLIGHT_LIST (called program)

BC402_PCS_CONN_LIST (calling program)

Solution

BC402_MMS_FLIGHT_LIST (called program)

BC402_MMS_CONN_LIST (calling program)

Use the ABAP memory to exchange data between programs and the SAP memory to preassign fields in a transaction call.

Task 1

Copy the called program, *BC402_INS_FLIGHT_LIST* (or your own program, **`ZBC402_##_FLIGHT_LIST`**), to the name **`ZBC402_##_FLIGHT_LIST_MEM`**. Enhance the program so that it can selectively output the flights in a list or save the internal table with its data in the ABAP memory. Create an invisible parameter on the selection screen that enables storage in the ABAP memory when the program is called with SUBMIT.

1. Copy the program and all its subcomponents.
2. Define an additional parameter for the selection screen (suggested name: **`PA_LIST`**). Type the parameter with type ABAP_BOOL from the ABAP type group, for example.
3. Check the documentation to find a suitable addition for the PARAMETERS statement that suppresses the parameter when displaying the selection screen.
4. Supplement the program so that it suppresses the list output when the parameter is not set to its default value. In this case, you want to export the read data into the ABAP memory. Choose a suitable ID for your data cluster (suggested name: **`BC402_##_FLIGHTS`**).

Task 2

© Copyright . All rights reserved.

Prohibida su reproduccion

Copy the calling programBC402_PCS_CONN_LIST (or your own program, ZBC402_##_CONN_LIST) to the name **ZBC402_##_CONN_LIST_MEM** and modify the name of the called program in the SUBMIT statement. Call the program so that it saves the data in the ABAP memory instead of directly direct output.

1. Copy the calling program.
2. Modify the SUBMIT statement to call program ZBC402_##_FLIGHT_LIST_MEM. Supply the new parameter with a value to suppress the list display and write the data to the ABAP memory.

Task 3

After the program call, read the data from the ABAP memory and use function module BC402_DISPLAY_TABLE to display the data in an ALV grid. Delete the data in the ABAP memory after it is read.

1. After the program call, import the data from the ABAP memory by creating a suitable internal table. When an error occurs, respond with error message 361 from message class BC402.
2. Implement a call of function module BC402_DISPLAY_TABLE. Pass on the data from the ABAP memory on to this function module for display.
3. Optional: Deal with the (class-based) exception of the function module.
4. Release the memory for your cluster to delete the data from the ABAP memory.

Task 4

Analyze transaction BC402MCAR. Is it possible to preassign the input field on the first screen using a parameter in the SAP memory? If so, use this option. Remove the USING and MODE additions from the CALL TRANSACTION statement and suppress the display of the first screen.

1. Find out whether the input field from screen 0100 in program SAPMBC402_IND_CARRIER is linked with a parameter ID. If so, what is the parameter ID name?
2. Check whether automatic pre-assignment from the SAP memory is active for the field.
3. Use a suitable statement to set the value of the parameter to the desired value in the SAP memory before the transaction call.
4. Remove the USING and MODE additions from the CALL TRANSACTION statement. Instead, use a suitable addition to suppress the first screen.

© Copyright . All rights reserved.

Prohibida su reproduccion

Use ABAP Memory and SAP Memory

Business Example

An existing program selects data based on user entries and outputs it in a list. You call this program from within another program. Instead of displaying the data directly with the called report, you want to load the data you select into the SAP memory, and process it further in the calling program. You also want to use parameters in the ABAP memory to pre assign fields when calling a transaction.

Template

BC402_INS_FLIGHT_LIST (called program)

BC402_PCS_CONN_LIST (calling program)

Solution

BC402_MMS_FLIGHT_LIST (called program)

BC402_MMS_CONN_LIST (calling program)

Use the ABAP memory to exchange data between programs and the SAP memory to preassign fields in a transaction call.

Task 1

Copy the called program, *BC402_INS_FLIGHT_LIST* (or your own program, **`ZBC402_##_FLIGHT_LIST`**), to the name **`ZBC402_##_FLIGHT_LIST_MEM`**. Enhance the program so that it can selectively output the flights in a list or save the internal table with its data in the ABAP memory. Create an invisible parameter on the selection screen that enables storage in the ABAP memory when the program is called with SUBMIT.

1. Copy the program and all its subcomponents.
   a) Carry out this step in the custom manner.
2. Define an additional parameter for the selection screen (suggested name: **`PA_LIST`**). Type the parameter with type ABAP_BOOL from the ABAP type group, for example.
   a) See the source code excerpt from the model solution.
3. Check the documentation to find a suitable addition for the PARAMETERS statement that suppresses the parameter when displaying the selection screen.
   a) NO-DISPLAY addition. See source code excerpt from the model solution.
4. Supplement the program so that it suppresses the list output when the parameter is not set to its default value. In this case, you want to export the read data into the ABAP memory. Choose a suitable ID for your data cluster (suggested name: **`BC402_##_FLIGHTS`**).
   a) See the source code excerpt from the model solution.

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 2

Copy the calling programBC402_PCS_CONN_LIST (or your own program, ZBC402_##_CONN_LIST) to the name **ZBC402_##_CONN_LIST_MEM** and modify the name of the called program in the SUBMIT statement. Call the program so that it saves the data in the ABAP memory instead of directly direct output.

1. Copy the calling program.
   a) Carry out this step in the usual manner.
2. Modify the SUBMIT statement to call program ZBC402_##_FLIGHT_LIST_MEM. Supply the new parameter with a value to suppress the list display and write the data to the ABAP memory.

Task 3

After the program call, read the data from the ABAP memory and use function module BC402_DISPLAY_TABLE to display the data in an ALV grid. Delete the data in the ABAP memory after it is read.

1. After the program call, import the data from the ABAP memory by creating a suitable internal table. When an error occurs, respond with error message 361 from message class BC402.
   a) See the source code excerpt from the model solution.
2. Implement a call of function module BC402_DISPLAY_TABLE. Pass on the data from the ABAP memory on to this function module for display.
   a) See the source code excerpt from the model solution.
3. Optional: Deal with the (class-based) exception of the function module.
4. Release the memory for your cluster to delete the data from the ABAP memory.
   a) See the source code excerpt from the model solution.

Task 4

Analyze transaction BC402MCAR. Is it possible to preassign the input field on the first screen using a parameter in the SAP memory? If so, use this option. Remove the USING and MODE additions from the CALL TRANSACTION statement and suppress the display of the first screen.

1. Find out whether the input field from screen 0100 in program SAPMBC402_IND_CARRIER is linked with a parameter ID. If so, what is the parameter ID name?
   a) Start the transaction.
   b) Call up the help (F1 key) for the input field and then the technical information.
   c) The Parameter ID has the value CAR.
2. Check whether automatic pre-assignment from the SAP memory is active for the field.

© Copyright . All rights reserved.

Prohibida su reproduccion

a) Navigate to screen 0100.

b) Analyze the attributes for the input screen (either in the list of elements or using the attributes window in the graphical layout editor).

c) The *Get Parameters* checkbox is set.

3. Use a suitable statement to set the value of the parameter to the desired value in the SAP memory before the transaction call.

   a) See the source code excerpt from the model solution.

4. Remove the USING and MODE additions from the CALL TRANSACTION statement. Instead, use a suitable addition to suppress the first screen.

   a) AND SKIP FIRST SCREEN addition. See source code excerpt from the model solution.

```
A source code excerpt from the model solution is as follows:

BC402_MMS_FLIGHT_LIST (called program)

REPORT  bc402_mms_flight_list MESSAGE-ID bc402.

TYPE-POOLS: abap.

DATA: gt_flight TYPE TABLE OF sflight,
      gs_flight TYPE          sflight.

PARAMETERS pa_car TYPE sflight-carrid.
SELECT-OPTIONS so_con FOR gs_flight-connid.

PARAMETERS pa_list TYPE abap_bool DEFAULT abap_true NO-DISPLAY.


START-OF-SELECTION.

  SELECT * FROM sflight INTO TABLE gt_flight
           WHERE carrid =  pa_car AND
                 connid IN so_con.

  IF sy-subrc <> 0.
    MESSAGE e038.
  ENDIF.

  IF pa_list = abap_true.

    LOOP AT gt_flight INTO gs_flight.

      WRITE: /
             gs_flight-carrid,
             gs_flight-connid,
             gs_flight-fldate,
             gs_flight-price    CURRENCY gs_flight-currency,
             gs_flight-currency,
             gs_flight-planetype,
             gs_flight-seatsmax,
             gs_flight-seatsocc.

    ENDLOOP.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
  ELSE.

    EXPORT flights FROM gt_flight
        TO MEMORY ID 'BC402_FLIGHTS'.

  ENDIF.


BC402_MMS_CONN_LIST (calling program), Method: on_double_click

  METHOD on_double_click.

    DATA lt_sflight TYPE TABLE OF sflight.

    DATA ls_conn LIKE LINE OF gt_conn.


    READ TABLE gt_conn INTO ls_conn INDEX row.

    CASE column.
      WHEN 'CARRID'.

        SET PARAMETER ID 'CAR' FIELD ls_conn-carrid.

        CALL TRANSACTION 'BC402MCAR'
             AND SKIP FIRST SCREEN.

      WHEN OTHERS.

        SUBMIT bc402_mms_flight_list
               AND RETURN
               WITH pa_car EQ ls_conn-carrid
               WITH so_con EQ ls_conn-connid

               WITH pa_list EQ space.

        IMPORT flights TO lt_sflight
               FROM MEMORY ID 'BC402_FLIGHTS'.

        IF sy-subrc <> 0.
          MESSAGE e361.
        ELSE.
          CALL FUNCTION 'BC402_DISPLAY_TABLE'
            CHANGING
              ct_table = lt_sflight.
        ENDIF.

        FREE MEMORY ID 'BC402_FLIGHTS'.

    ENDCASE.

  ENDMETHOD.                    "on_double_click
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Dynamic Data Objects

Strings and internal tables are dynamic data objects (also called deep data objects) with memory requirements that can change during the program runtime. The system manages the working data of these data objects internally, using references.

> Note:
> Other deep data objects include references and structures that contain at least one deep component, regardless of nesting.

Memory management treats deep data objects and flat data objects differently.

Memory Management for Flat Data Objects

Figure 270: Memory Management for Flat Data Objects

Flat data objects are elementary data types with fixed lengths and structures that only have flat components. The content of flat data objects represents actual working data. The system reserves memory for flat data objects as soon as it loads the program into the internal session. The system does not release the memory reserved for flat objects until it deletes the internal session.

© Copyright . All rights reserved.

Prohibida su reproduccion

Memory Management for Strings

Figure 271: Memory Management for Strings

When the system loads a program, it creates only a reference for the statically declared deep data objects at first, such as strings and internal tables. The static memory requirement for this reference is 8 bytes.

When you use the deep data object, additional working memory is requested at program runtime. For dynamic data objects (strings and internal tables), this memory is for the administration information (header) and the data object itself.

The memory requirement for a header depends on the hardware platform, but is 100 bytes.

When the requested (allocated) memory is not sufficient for later access, additional memory can be reserved.

The system allocates more memory than required for the current content of the data object, to avoid frequent requests as the content increases. The system also limits the size of strings to the largest contiguous block of memory that is requested. This limit is a maximum of 2 GB, but is usually restricted further with profile parameter *ztta/max_memreq*_MB.

> Caution:
> The requested memory must be available in the current internal mode otherwise, a runtime error occurs.

When you initialize a string with CLEAR or FREE, the system deletes the data, but the reference variables and the header are retained and reused during the next memory request. Accordingly, when a string has been used and then deleted, its memory requirement consists of the reference and the requirement of the header.

© Copyright . All rights reserved.

Prohibida su reproduccion

Memory Management for Internal Tables

Figure 272: Memory Management for Internal Tables

Memory management for internal tables is similar to management for strings.

Memory management for internal tables is different from the management of strings as follows:

- The headers of internal tables require more memory than other headers, because internal table header data can be distributed among several blocks. This means that internal table header can manage references to these blocks.
- The CLEAR, REFRESH, and FREE statements have different effects. CLEAR and REFRESH only delete the data, while FREE can also delete the table header when it takes up too much memory.
- The amount of memory that is allocated during initial access can be changed when you define the internal table (optional addition INITIAL SIZE).

You can specify the number of rows, n, as numeric literals or numeric constants in the INITIAL SIZE addition. The system then selects the size of the first block in memory to fit the exact number of lines.

> Hint:
> For global table types in the *ABAP Dictionary*, the *Initial Number of Lines* input field on the *Initialization and Access* tab page performs this function.

If you do not specify the addition, the system automatically specifies the number 0. When the value of *n* exceeds a maximum value, the system automatically assigns a suitable memory area during the first access. The system allocates additional memory based on the number of lines

© Copyright . All rights reserved.

Prohibida su reproduccion

that exceeds the specified value *n* during or after initial access. Therefore, n is not a fixed upper limit for the memory capacity of the internal table.

> Caution:
> Exceeding the specified initial memory requirement when the internal table is filled can cause the system to allocate more extra memory than actually needed. Therefore, SAP recommends that you only specify the initial memory requirement when you know the number of entries in the table and want to specify the size of the initial main memory requirement.

> Hint:
> Specifying an initial memory requirement is appropriate when the internal tables are components of table types (nested tables) and the inner tables only contain a few lines.

Dynamic Data Objects Sharing

Figure 273: Sharing Between Strings

To minimize runtime and conserve memory, assigned values are shared between strings and between internal tables.

Sharing means the system initially does not copy the actual data values during an assignment. Instead, it copies only the necessary management entries. The target object therefore refers to the same data as the source object. When strings are involved, the second internal reference is filled to point to the existing string header. The system cancels sharing when you access either the source or the target in the change mode. This approach is called copy-on-write semantics. In this case, the system actually copies data values and changes the references and header accordingly.

© Copyright . All rights reserved.

Prohibida su reproduccion

Share Data in Internal Tables

Figure 274: Sharing Between Internal Tables

With internal tables only the actual data is shared. The system assigns each table a separate table header, which refers to the existing table body. Tables whose line types contain other table types, are not shared.

> Hint:
> Sharing is also used to pass values to procedures. Therefore, there is no major difference between using call by value and call by reference with regard to runtime requirements for passing strings and internal tables to forms. A runtime-intensive copy operation takes place if you use call by value and then access the formal parameters in change mode within the procedure. In contrast, when you use call by reference, the actual and formal parameters continue to use the same memory.

Definition of Boxed Components

Large static data objects can use large amount of memory even when they are INITIAL. When you have many identical unused static data objects in your program, it can lead to memory wastage.

Consider the following example:

- An instance attribute with a static structure which is INITIAL in most of the instances of a class.
- An internal table where a large number of columns are not used for many table lines.

As of SAP NetWeaver 7.0 EhP2, you can use Boxed Components to reduce the memory consumption of static data objects.

Boxed components can be as follows:

© Copyright . All rights reserved.

Prohibida su reproduccion

- Substructures of nested structures
- Structured attributes of classes and interfaces

These components are also called static boxes. A static box is declared with the BOXED addition when defining substructures using TYPES, or when declaring structured attributes of classes or interfaces using DATA. For global structure types, classes, and interfaces, a new value for the property *Typing Method* is available.

> Hint:
> It is not possible to use the BOXED addition in DATA statements that lie outside class definitions. To define boxed components outside class definitions, you have to use the TYPES statement or define a suitable global structure type.

Boxed Components and Initial Value Sharing

Figure 275: Structure with Boxed Component

The figure illustrates an example of a nested structure with a boxed static substructure. The contents of the boxed subcomponent SUB are not stored inside the structure GS_BOXED directly. Instead, the subcomponent is internally represented by a reference. As long as the subcomponent is still INITIAL, this reference points to an appropriately typed initial structure stored centrally in the PXA. This is called Initial Value Sharing for boxed components.

The system revokes the initial value sharing, if, for example, a write access to the substructure takes place, after which the substructure is no longer initial. As shown in the figure, the system creates the substructure dynamically within the internal session of the program, and changes the reference so that it points to the new substructure.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> Once you revoke the initial value sharing, it is retained. The CLEAR or FREE statement only initializes the components of the substructure but does not release the memory allocated for the substructure.

Operations that revoke the initial value sharing of a boxed component are as follows:

- Write access to the static box or one of its components
- Static box or one of its components as an actual parameter for procedure calls
- Address the static boxed component or one of its subcomponents with a data reference or a field symbol

> Hint:
> A structure that contains a boxed component is always deep even when all its components, including the boxed subcomponent, are of static length.

Examples for Boxed Components

```
TYPES: BEGIN OF ty_s_struct,
        comp1 TYPE i,
        comp2 TYPE c LENGTH 4,
       END OF ty_s_struct.

CLASS lcl_boxed DEFINITION.
  PUBLIC SECTION.
    DATA:
       attr TYPE ty_s_struct BOXED.
ENDCLASS.

DATA go_boxed TYPE REF TO lcl_boxed.
```

```
CREATE OBJECT go_boxed.


 go_boxed->attr-comp2 = 'abcd'.
```

Figure 276: Instance Attribute Defined as Boxed Component

The figure illustrates an example for an instance attribute of a class defined as a boxed component.

If the boxed attribute remains initial in many instances of the class, all instances share the same initial value. This can significantly reduce the memory consumption of the program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Example – Internal Tables and Boxed Components

```
TYPES: BEGIN OF ty_s_sub,
        subc1 TYPE i,
        subc2 TYPE c LENGTH 4,
       END OF ty_s_sub.

TYPES: BEGIN OF ty_s_boxed,
        comp1 TYPE c LENGTH 4,
        sub   TYPE ty_s_sub BOXED,
        comp2 TYPE i,
      END OF ty_s_boxed.

DATA:
   gs_boxed TYPE ty_s_boxed.
   gt_boxed TYPE TABLE OF ty_s_boxed.
```

```
DO 3 TIMES.
  gs_boxed-comp1 = 'abcd'.
  gs_boxed-comp2 = sy-index.
  APPEND gs_boxed TO gt_boxed.
ENDDO.
```

Figure 277: Example – Internal Tables and Boxed Components

You can reduce the overall size of an internal table if you use a structure type having a boxed component as line type. This is illustrated in the figure.

Use Cases for Boxed Components

Boxed components should fulfil requirements such as follows:

- Boxed component often remains initial

  The advantage of boxed components lies in the reduced memory usage through initial value sharing. This implies that a boxed component has no advantage if the component never remains INITIAL most of the times.

- Boxed component is not too small

  If the substructure or structured attribute is small, the memory savings cannot counterbalance the 8-byte overhead for the reference.

- Boxed component is used often

  The memory savings through boxed components scale with the number of times the initial structure is actually shared.

> Hint:
> It is not necessary that the class or nested structure must have several instances within the same execution of the program. As the PXA exists only once per server instance, you can also benefit from boxed components if the system executes the program many times on the same server instance.

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Generate and activate programs
- Explain the user session memory management
- Transfer data between programs
- Explain the memory management of deep data objects
- Use boxed components

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. If you use the _____ statement, the system terminates the current program and starts the transaction with transaction code T_CODE.

   Choose the correct answer.

   - [ ] A SUBMIT
   - [ ] B AND RETURN
   - [ ] C LEAVE TO TRANSACTION 'T_CODE'
   - [ ] D CALL TRANSACTION 'T_CODE'

2. A saved or buffered LOAD is considered obsolete if the developed version of the program was changed.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

3. The modifiable parts of a program are created in memory in roll areas during every execution. Even the program is executed several times by the same user, each execution has its own roll area. Which of the following is an example of modifiable parts of a program?

   Choose the correct answer.

   - [ ] A Byte code for statements
   - [ ] B Values of constants and literals
   - [ ] C Program texts
   - [ ] D Screen definitions
   - [ ] E Data objects (variables)

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following are true when you activate a new object, for example a program?

   Choose the correct answers.

   - [ ] A Object is saved
   - [ ] B Syntax of the object is checked
   - [ ] C The newly created version becomes the current active version
   - [ ] D An executable binary file is generated

5. What statement can also delete the table header if it takes up too much memory?

   Choose the correct answer.

   - [ ] A CLEAR
   - [ ] B REFRESH
   - [ ] C FREE

6. To define boxed components outside of class definitions, you have to use the DATA statement.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. If you use the _____ statement, the system terminates the current program and starts the transaction with transaction code T_CODE.

   Choose the correct answer.

   - [ ] A SUBMIT
   - [ ] B AND RETURN
   - [x] C LEAVE TO TRANSACTION 'T_CODE'
   - [ ] D CALL TRANSACTION 'T_CODE'

2. A saved or buffered LOAD is considered obsolete if the developed version of the program was changed.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

3. The modifiable parts of a program are created in memory in roll areas during every execution. Even the program is executed several times by the same user, each execution has its own roll area. Which of the following is an example of modifiable parts of a program?

   Choose the correct answer.

   - [ ] A Byte code for statements
   - [ ] B Values of constants and literals
   - [ ] C Program texts
   - [ ] D Screen definitions
   - [x] E Data objects (variables)

© Copyright . All rights reserved.

Prohibida su reproduccion

4. Which of the following are true when you activate a new object, for example a program?

   Choose the correct answers.

   - [ ] A Object is saved
   - [ ] B Syntax of the object is checked
   - [ ] C The newly created version becomes the current active version
   - [ ] D An executable binary file is generated

5. What statement can also delete the table header if it takes up too much memory?

   Choose the correct answer.

   - [ ] A CLEAR
   - [ ] B REFRESH
   - [x] C FREE

6. To define boxed components outside of class definitions, you have to use the DATA statement.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

© Copyright . All rights reserved.

Prohibida su reproduccion

UNIT 14 ABAP Open SQL in Detail

Lesson 1

Implementing Complex WHERE Conditions and Special INTO Clauses 596

Lesson 2

Processing and Aggregating Datasets in the Database 605
Exercise 33: Retain Ordered and Condensed Datasets from the Database 609
Exercise 34: Perform Calculations on the Database 623

Lesson 3

Reading from Multiple Database Tables 629
Exercise 35: Implement a Join of Three Database Tables 635
Exercise 36: Implement Full Buffering and Buffering on Demand 645
Exercise 37: Read Additional Data with the FOR ALL ENTRIES addition 655

UNIT OBJECTIVES

- Delimit datasets selected in the database using WHERE-clause
- Use different types of data objects as the target for SELECT statements
- Read large volumes of data from the database
- Retain ordered or condensed datasets from the database
- Perform calculations on the database
- Identify disadvantages of nested SELECT
- Use ABAP joins and database views
- Explain special techniques for reading from multiple database tables
- Use explicit buffering techniques

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 14
Lesson 1

Implementing Complex WHERE Conditions and Special INTO Clauses

LESSON OVERVIEW
This lesson explains the logical expressions that are used in WHERE conditions of Open SQL statements. It also discusses a variety of other operators apart from simple value comparisons with “=” and “EQ”.

Business Example

You want to implement database accesses with Open SQL in your applications. These database accesses must be sophisticated and efficient. Therefore, you need to find out about the special options of using logical expressions in the WHERE condition and specifying data objects in INTO clauses. For this reason, you require the following knowledge:

- An understanding of the importance of the possible operators in WHERE conditions
- An understanding of how to use the different operators correctly
- An understanding of the options available for specifying data objects after INTO
- An understanding of how to implement sequential processing of large data volumes

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Delimit datasets selected in the database using WHERE-clause
- Use different types of data objects as the target for SELECT statements
- Read large volumes of data from the database

Operators in WHERE Conditions

When formulating WHERE conditions, you mainly compared single values and evaluated selection tables (select-options). The comparison values here were data objects (variables and literals). However, Open SQL also supports a variety of operators and enables you to compare database table fields with one another.

Possible operators in WHERE conditions are listed in the following table:

© Copyright . All rights reserved.

Prohibida su reproduccion

Operator	Meaning and Use
=, <, >, <=, <=, <> EQ, LT, GT, LE, GE, NE	Comparison with a single value In character-type fields, the result of size comparisons may depend on the database code page.
IN (dobj1, dobj2, ...)	Comparison with a list of single values
BETWEEN dobj1 AND dobj2	Comparison with an interval In character-type fields, the result may depend on the database code page.
LIKE dobj	Comparison with character strings The _ and % placeholders let you define a comparison pattern in DOBJ.
IN seltab	Evaluation of a selection table (select-options)
IS [NOT] NULL	Checks whether the database field has a null value
AND, OR	Link of logical expressions
NOT	Negation of a logical expression

Note:
Some of the operators can have a negative impact on performance if the operators use potential index fields for delimitation because the database optimizer does not find an efficient access strategy.

© Copyright . All rights reserved.

Prohibida su reproduccion

Operators in WHERE Conditions (2)

Example: SELECT with complex WHERE condition

```
DATA lt_customers TYPE TABLE OF scustom.

SELECT * FROM scustom INTO TABLE lt_customers

     WHERE country  IN ('DE','US')

       AND discount BETWEEN '005' AND '010'

       AND postcode LIKE '___5_'

       AND name     LIKE '%ra%'.
```

Country selection using list of single values

Range for discount

Postal code has five places and the number "5" in position 4

Name contains substring "ra" in any position

Wildcards for LIKE:

_ = Placeholder for a single character

% = Placeholder for substring of any length

Figure 278: Operators in WHERE Conditions

The figure illustrates an example of a complex search for airline customers using name and address data.

Hint:
The “_” and “%” placeholders correspond to the SQL standard in delimitation with LIKE. In other ABAP statements, the “+” and “*” characters are used for similar comparisons, for example, within comparison tables.

To search for the “_” or “%” characters, use the ESCAPE addition to define an escape symbol. A condition that the column contents must contain a “_” in any location is then formulated as LIKE '%! _%' ESCAPE '!'.

When the developer links conditions, the AND operators are evaluated before the OR operators. Set parentheses to change this order as needed.

Note:
The selection of the operator can have a major impact on the usability of a condition for an index search string. The formulation of the WHERE condition influences the search strategy of the optimizer and the performance of the database access.

© Copyright . All rights reserved.

Prohibida su reproduccion

Table Field Comparison

All domestic connections (country of departure = country of arrival)

```
DATA lt_spfli TYPE TABLE OF spfli.

SELECT * FROM spfli
         INTO TABLE lt_spfli
         WHERE countryfr = spfli~countryto.
```

Table name must be specified

Both fields must have the same (technical type), otherwise the result is dependent on the database!

All flights with full business class,
but available seats in economy class

```
DATA lt_flight TYPE TABLE OF sflight.

SELECT * FROM sflight AS a
         INTO TABLE lt_flight
         WHERE  seatsocc_b = a~seatsmax_b
         AND    seatsocc   < a~seatsmax.
```

Alias for the table → shorter WHERE condition

Figure 279: Comparing Table Fields

In addition to variables and constants, you can also enter fields from the same database tables after the comparison operators (for example, =, EQ, <>, NE, <, LT) in the WHERE conditions of open SQL statements. Entering fields from the same database tables after the comparison operators makes it possible to compare two database fields with each other in the WHERE condition and only read the records whose contents fulfill the condition.

To avoid the confusion of the second table field with a data object, the name of the database table must be specified with the field label connected with the "~" character. Alternatively, use an alias for the table name, which you define with the AS addition after the table name in the FROM clause.

Note:
Aliases for table names play a subordinate role here. They are used to abbreviate the WHERE condition. However, you see aliases again in the formulation of joins, where they are crucial if the same database table appears several times in the FROM clause.

Caution:
The fields that are compared must have the same basic type and length (ideally, they are based on the same domain). Otherwise, the result depends on how the respective database system stores the different types and how it handles spaces at the end of values.

Dimension of Target Areas

© Copyright . All rights reserved.

Prohibida su reproduccion

When you specify data objects as targets of SELECT statements, you need to differentiate whether the target area that is specified after INFO is single line or multiline.

Do not confuse this with single line and multiline result sets, as listed in the following table:

Statement	Result-Set	Target Area
SELECT SINGLE...	single-line	single-line
SELECT... INTO TABLE...	multiline	multiline
SELECT... ENDSELECT.	multiline	single-line

Single Line Target Area

Filling any list of elementary fields

```
DATA: lv_max TYPE sflight-seatsmax,
      lv_occ TYPE sflight-seatsocc.

SELECT SINGLE seatsmax seatsocc
              FROM sflight INTO (lv_max, lv_occ)
              WHERE ... .
```

Fields comma-separated, no spaces behind open parenthesis

Targeted filling of individual components of a structure

```
DATA ls_struct TYPE sflight.

SELECT SINGLE seatsmax seatsocc
   FROM sflight
   INTO (ls_struct-seatsmax, ls_struct-seatsocc)
   WHERE ... .

SELECT SINGLE seatsmax seatsocc
   FROM sflight
   INTO CORRESPONDING FIELDS OF ls_struct
   WHERE ... .
```

Explicit components as an alternative to CORRESPONDING FIELDS OF ls_struct

Figure 280: List of Data Objects Instead of Single Structure

Single line target areas are used for single record access using SELECT SINGLE and in single loops using SELECT ... ENDSELECT. The target may be a structure (structured data object), and is always specified with the INTO addition (without TABLE) or with INTO CORRESPONDING FIELDS OF. You can also specify a list of elementary data objects or elementary structure components. In this case, the individual fields are separated by commas, while the entire list is enclosed in parentheses.

Hint:
The syntax of this value is correct only if there are no spaces between the open parentheses and the first data object. Any number of spaces can appear before the closing parentheses, before and after commas, but are not required.

© Copyright . All rights reserved.

Prohibida su reproduccion

Specifying individual structure components is a more robust alternative to specifying the entire structure after the CORRESPONDING FIELDS OF addition. In theory, you can also specify a list of individual data objects and structure components after INTO CORRESPONDING FIELDS OF.

Multiline Target Area

```
DATA lt_spfli TYPE TABLE OF spfli.
```

Filling an internal table (overwrite)

```
SELECT * FROM spfli
       INTO TABLE lt_spfli
       WHERE ... .

SELECT cityfrom cityto FROM spfli
       INTO CORRESPONDING FIELDS OF
       TABLE lt_spfli
       WHERE ... .
```

Appending lines to an internal table

```
SELECT * FROM spfli
       APPENDING TABLE lt_spfli
       WHERE ... .

SELECT cityfrom cityto FROM spfli
       APPENDING CORRESPONDING FIELDS OF
       TABLE lt_spfli
       WHERE ... .
```

Figure 281: APPENDING Instead of INTO

When SELECT statements have multiline target areas, you must specify an internal table as the data object. Specify the table, for example, after the INTO TABLE addition or after INTO CORRESPONDING FIELDS OF TABLE.

Alternatively, replace INTO TABLE and INTO CORRESPONDING FIELDS OF TABLE with APPENDING TABLE and APPENDING CORRESPONDING FIELDS OF TABLE, respectively. If you use INTO, the contents of the internal table are replaced completely with the selected data. If you use APPEND, the data is added to any existing lines in the table.

Hint:
If you use APPENDING for sorted and hashed tables, no append is performed. Specifically, the addition is not treated as an index access to the internal table, which means it can also be used for hashed tables. In sorted tables, the new lines are inserted in the existing lines in accordance with the sort sequence.

© Copyright . All rights reserved.

Prohibida su reproduccion

Explicit Package Size

Figure 282: PACKAGE SIZE Addition

When you process multiline result sets, you read them into an internal table (array fetch). If huge result sets are involved, the program can terminate because the available working memory is insufficient for an internal table of that size.

One solution to this is sequential processing in a SELECT loop. This approach has low memory requirements because only one record is transported to the working memory at any one time.

> Note:
> The database interface reads entire packages from the database but then copies the records to the work area individually. When a package is processed completely, the next one is requested from the database. As a result, the application server never transports the entire data volume at once.

A disadvantage of sequential processing in the SELECT loop is the fact that only one record is available to the application log at a time. The availability of a record is particularly problematic when you want to change the records in the database because the changes can only be carried out by single record access. However, a large number of single record write accesses have a negative impact on the database and program performance.

For such cases, Open SQL provides a variant of the SELECT statement in which the data is processed in a loop (SELECT ... ENDSELECT) but the target area is multiline (INTO TABLE). The number of records that are placed in the internal table at the same time and processed in the current loop pass is defined with the PACKAGE SIZE addition. You can specify this number either statically or dynamically, but it is defined once at the start of the loop. The same number of lines are processed during each loop pass.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> The value after PACKAGE SIZE determines the number of lines that the database interface provides to the application program in each loop pass. It has no major impact on the size of the packages that are transferred from the database to the application server.

Explicit Cursor

Figure 283: OPEN CURSOR ... FETCH ... CLOSE CURSOR

Packages of different sizes can be processed sequentially with the syntax illustrated in the figure.

In this variant of database read access, the access itself is detached from the transfer of the result set to the application program.

The OPEN CURSOR statement defines and executes the access, but does not pass any data on to the application program.

The FETCH NEXT CURSOR statement retrieves the desired number of lines (PACKAGE SIZE addition) from the result set and copies them to an internal table.

The CLOSE CURSOR statement completes the database access.

The importance of cursor variable, which is a variable with a special CURSOR type, is highlighted as follows:

- The cursor variable identifies database access in the subsequent statements.

© Copyright . All rights reserved.

Prohibida su reproduccion

- The cursor variable stores the position in the result set of database access up to which processing is already complete.

After OPEN CURSOR, the cursor variable points to a position before the first line.

The sequential FETCH statements not only allow you to read different numbers of records, but also allow you to specify a different data object as the target (structures or internal tables) after each INTO.

Hint:
You can also keep a limited number of accesses open at the same time, for example, to process several database tables in parallel.

Caution:
The cursor is closed implicitly by a database commit or database rollback. Therefore, no statements that cause an implicit DB commit or DB rollback can appear between OPEN CURSOR and CLOSE CURSOR. When a FETCH statement is performed for a cursor that has already been closed, a catchable exception is raised (exception class CX_SY_OPEN_SQL_DB).

LESSON SUMMARY
You should now be able to:

- Delimit datasets selected in the database using WHERE-clause
- Use different types of data objects as the target for SELECT statements
- Read large volumes of data from the database

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 14
Lesson 2

Processing and Aggregating Datasets in the Database

LESSON OVERVIEW

This lesson discusses several ways to use the SELECT statement to sort and process retrieved data before it is transferred, aggregated, or used to perform calculations.

Business Example

You want to implement database access with Open SQL in your applications. You want the data access of your application to be efficient and effective.

Therefore, you need to learn about the options and functions for sorting and aggregating data.

For this reason, you require the following knowledge:

- An understanding of how to request sorted or aggregated data from the database
- An understanding of how to use aggregate functions correctly

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Retain ordered or condensed datasets from the database
- Perform calculations on the database

© Copyright . All rights reserved.

Prohibida su reproduccion

Ordered Datasets

Sorting by primary key (ascending):

```
SELECT *
   FROM sflight INTO ... WHERE ...
   ORDER BY PRIMARY KEY.

SELECT mandt carrid connid fldate ...
   FROM sflight INTO ... WHERE ...
   ORDER BY PRIMARY KEY.
```

ORDER BY must appear after the WHERE condition

All primary key fields (including MANDT) must appear in the field list

Sorting by any key (ascending):

```
SELECT *
   FROM sflight INTO ... WHERE ...
   ORDER BY connid seatsocc.

SELECT ... connid ... seatsocc ....
   FROM sflight INTO ... WHERE ...
   ORDER BY connid seatsocc.
```

Sequence of fields determines priority

All sort fields must be in the field list

Sorting by any key (ascending or descending):

```
SELECT *
   FROM sflight INTO ... WHERE ...
   ORDER BY connid DESCENDING seatsocc ASCENDING.
```

Figure 284: ORDER BY Addition Variants

In Open SQL, you can have the database sort the dataset by specific criteria directly as part of the SELECT operation. To do so, use the ORDER BY clause for the SELECT statement.

In the simplest form of the ORDER BY PRIMARY KEY clause, the table is sorted by its complete primary key in ascending order.

To use ORDER BY PRIMARY KEY, the prerequisites are as follows:

- The FROM clause contains a single database table (no views or joins).
- The field list after SELECT contains all the table's key fields (including the client, which is often forgotten).

The syntax ORDER BY COL1, COL2 ... sorts the data by any existing columns. The sequence of the specified columns determines their sort priority.

The restrictions on ORDER BY are as follows:

- You cannot sort pooled tables and cluster tables by any field.
- You can only sort by columns that appear after the SELECT statement, which means the sort columns must be a part of the result set.
- You cannot sort by fields with type LCHAR, LRAW, STRING, or RAWSTRING.

© Copyright . All rights reserved.

Prohibida su reproduccion

ORDER BY Addition

```
TYPES: BEGIN OF lty_s_flightocc,
         connid   TYPE sflight-connid,
         fldate   TYPE sflight-fldate,
         seatsocc TYPE sflight-seatsocc,
       END OF lty_s_flightocc.
DATA lt_flightocc TYPE TABLE OF lty_s_flightocc.

SELECT connid fldate seatsocc
       FROM sflight
       INTO TABLE lt_flightocc
       WHERE seatsocc > 200
       ORDER BY connid DESCENDING seatsocc ASCENDING.
```

Result set sorted first by CONNID, then by SEATSOCC within the CONNID groups

ORDER BY must appear after the WHERE condition

MANDT	CARRID	CONNID	FLDATE	SEATSOCC
001	AA	0014	20081217	250
001	AA	0014	20090121	243
001	AA	0017	20081230	243
001	AA	0017	20090102	250
001	AA	0017	20090106	300
~~001~~	~~AA~~	~~0017~~	~~20090117~~	~~143~~
001	AA	0017	20090119	300
~~001~~	~~AA~~	~~0018~~	~~20081203~~	~~143~~
~~001~~	~~AA~~	~~0018~~	~~20081207~~	~~155~~
~~001~~	~~AA~~	~~0018~~	~~20090106~~	~~165~~
001	AA	0019	20081205	244
001	AA	0019	20090114	312

lt_flightocc

CONNID	FLDATE	SEATSOCC
0019	20081205	244
0019	20090114	312
0017	20081230	243
0017	20090102	250
0017	20090106	300
0017	20090119	300
0014	20090121	243
0014	20081217	250

Figure 285: ORDER BY Addition

The figure shows an example of the ORDER BY addition with free column selection.

If you do not use an addition, the system sorts retrieved datasets in ascending order. You can use the optional DESCENDING and ASCENDING clauses after a given field to define the sort direction.

> Note:
> Sorting can be a computationally expensive process if the result set is large and the sort fields are not also index fields.

© Copyright . All rights reserved.

Prohibida su reproduccion

Condensed Datasets

Figure 286: SELECT DISTINCT Statement

The SELECT DISTINCT statement in Open SQL instructs the database to aggregate the result set of a SELECT statement such that it does not contain any duplicate entries. All fields of the result set are used for the comparison. You can either load the result of SELECT DISTINCT into an internal table (array fetch) or process it sequentially (SELECT loop).

For SELECT DISTINCT, the applicable restrictions are as follows:

- The field list cannot contain any columns with type LCHAR, LRAW, STRING, or RAWSTRING.
- You must use SELECT DISTINCT * to access pooled tables and cluster tables. You cannot select individual columns.

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 14
Exercise 33

Retain Ordered and Condensed Datasets from the Database

Business Example

You want to enhance a program that outputs a customer's flight bookings.

Template

BC402_DBT_SQL_CONDENSE

Solution

BC402_DBS_SQL_CONDENSE

Use the DISTINCT clause for SELECT statements and request sorted data from the database.

Task 1

Copy the template BC402_DBT_SQL_CONDENSE to **ZBC402_##_CONDENSE**, where **##** is your group number. Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
2. Which input options are available on the selection screen?

   _____

   _____

   _____

3. Which data is output in the list? Which missing data do you still need?

   _____

   _____

   _____

Task 2

Implement subroutine GET_TRAVELAGS to determine the travel agencies that made the bookings. Make sure that each travel agency appears only once and that travel agencies are sorted alphabetically by name. Implement a single database access that returns the data completely in the desired form. Read from database view BC402_SCUS_BOOK.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. Analyze the interface of the subroutine, particularly the typing of the interface parameters. Which component contains the line type of CT_TRAVELAGS?

   _____

   _____

   _____

2. Implement a SELECT statement to fill CT_TRAVELAGS with the data for the travel agencies. Access database view BC402_SCUS_BOOK and limit the access with the customer number.
3. Supplement the SELECT statement with a clause that removes repeated entries from the result set.
4. Supplement the SELECT statement with a clause that returns the result set sorted by travel agency name.

© Copyright . All rights reserved.

Prohibida su reproduccion

Retain Ordered and Condensed Datasets from the Database

Business Example

You want to enhance a program that outputs a customer's flight bookings.

Template

BC402_DBT_SQL_CONDENSE

Solution

BC402_DBS_SQL_CONDENSE

Use the DISTINCT clause for SELECT statements and request sorted data from the database.

Task 1

Copy the template BC402_DBT_SQL_CONDENSE to **ZBC402_##_CONDENSE**, where **##** is your group number. Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
   a) Perform this step in the standard manner.
2. Which input options are available on the selection screen?

   An airline customer number.
3. Which data is output in the list? Which missing data do you still need?

   Data output is the customer's name, form of address, and a list of all the customer's bookings (without cancellations). Still missing are a list of total bookings (separated by currency) and a list of travel agencies who made the bookings.

Task 2

Implement subroutine GET_TRAVELAGS to determine the travel agencies that made the bookings. Make sure that each travel agency appears only once and that travel agencies are sorted alphabetically by name. Implement a single database access that returns the data completely in the desired form. Read from database view BC402_SCUS_BOOK.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. Analyze the interface of the subroutine, particularly the typing of the interface parameters. Which component contains the line type of CT_TRAVELAGS?

   AGENCYNUM: Travel agency number, NAME: Travel agency name, CITY: City.

2. Implement a SELECT statement to fill CT_TRAVELAGS with the data for the travel agencies. Access database view BC402_SCUS_BOOK and limit the access with the customer number.
   a) See the source code excerpt from the model solution.
3. Supplement the SELECT statement with a clause that removes repeated entries from the result set.
   a) DISTINCT clause. See the source code excerpt from the model solution.
4. Supplement the SELECT statement with a clause that returns the result set sorted by travel agency name.
   a) ORDER BY clause. See the source code excerpt from the model solution.
   Source code excerpt from the model solution:

```
REPORT  bc402_dbs_sql_condense.

TYPES: BEGIN OF gty_s_sums,
         forcuram  TYPE sbook-forcuram,
         forcurkey TYPE sbook-forcurkey,
       END OF gty_s_sums.

TYPES: BEGIN OF gty_s_travelags,
         agencynum  TYPE bc402_scus_book-agencynum,
         name       TYPE bc402_scus_book-name,
         city       TYPE bc402_scus_book-city,
       END OF gty_s_travelags.


TYPES:
   gty_t_sums      TYPE SORTED TABLE OF gty_s_sums
                        WITH UNIQUE KEY forcurkey,

   gty_t_bookings  TYPE STANDARD TABLE OF scus_book
                        WITH NON-UNIQUE KEY
                        carrid connid fldate bookid,

   gty_t_travelags TYPE STANDARD TABLE OF gty_s_travelags
                        WITH NON-UNIQUE KEY agencynum.

DATA:
  gs_customer  TYPE scustom,
  gt_bookings  TYPE gty_t_bookings,
  gt_sums      TYPE gty_t_sums,
  gt_travelags TYPE gty_t_travelags.

PARAMETERS:
  pa_cust TYPE sbook-customid DEFAULT '00000001'.


START-OF-SELECTION.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
  SELECT SINGLE * FROM scustom INTO gs_customer
         WHERE id = pa_cust.

  SELECT * FROM scus_book
           INTO TABLE gt_bookings
           WHERE customid =  pa_cust
           AND  cancelled <> 'X'.

  PERFORM get_sums USING    pa_cust
                   CHANGING gt_sums.

  PERFORM get_travelags USING    pa_cust
                        CHANGING gt_travelags.

  PERFORM output_list USING gs_customer
                            gt_bookings
                            gt_sums
                            gt_travelags.


*&---------------------------------------------*
*&      Form  GET_SUMS
*&---------------------------------------------*
FORM get_sums  USING    pv_customid TYPE sbook-customid
               CHANGING ct_sums     TYPE gty_t_sums.
  SELECT SUM( forcuram ) forcurkey
         FROM sbook
         INTO TABLE ct_sums
         WHERE customid = pv_customid
         GROUP BY forcurkey.
ENDFORM.                    " GET_SUMS

*&---------------------------------------------*
*&      Form  GET_TRAVELAGS
*&---------------------------------------------*
FORM get_travelags  USING     pv_customid TYPE sbook-customid
                    CHANGING ct_travelags TYPE gty_t_travelags.
  SELECT DISTINCT agencynum name city
         FROM bc402_scus_book
         INTO TABLE ct_travelags
         WHERE customid = pv_customid
         ORDER BY name.
ENDFORM.                    " GET_TRAVELAGS

*&---------------------------------------------*
*&      Form  OUTPUT_LIST
*&---------------------------------------------*
FORM output_list  USING ps_customer  TYPE scustom
                        pt_bookings  TYPE gty_t_bookings
                        pt_sums      TYPE gty_t_sums
                        pt_travelags TYPE gty_t_travelags.

  DATA:
    ls_bookings  LIKE LINE OF pt_bookings,
    ls_sums      LIKE LINE OF pt_sums,
    ls_travelags LIKE LINE OF pt_travelags.
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
  DATA lv_text TYPE string.

  CONCATENATE text-wcm
              ps_customer-form
              ps_customer-name
              INTO lv_text
              SEPARATED BY space.
  CONDENSE lv_text.

  WRITE: / lv_text.
  SKIP.

  WRITE / text-lob.
  ULINE.
  SKIP.

  LOOP AT pt_bookings INTO ls_bookings.
    WRITE: /
      ls_bookings-bookid,
      ls_bookings-carrid RIGHT-JUSTIFIED,
      ls_bookings-connid,
      ls_bookings-fldate,
      ls_bookings-cityfrom RIGHT-JUSTIFIED,
      '->',
      ls_bookings-cityto.
  ENDLOOP.
  SKIP 2.

  WRITE: / text-sum.
  ULINE.
  SKIP.

  LOOP AT pt_sums INTO ls_sums.
    WRITE: /
      ls_sums-forcuram CURRENCY ls_sums-forcurkey,
      ls_sums-forcurkey.
  ENDLOOP.
  SKIP 2.

  WRITE / text-tag.
  ULINE.
  SKIP.

  LOOP AT pt_travelags INTO ls_travelags.
    WRITE: /
      ls_travelags-name,
      ls_travelags-city.
  ENDLOOP.

ENDFORM.                    " OUTPUT_LIST
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Aggregate Expressions

An aggregate expression uses an aggregate function to perform calculations on a specified column in the SELECT statement. Aggregate expressions determine values from multiple rows in a column of a database table. The calculation is performed in the database system.

The aggregate functions supported by the ABAP Open SQL are as follows:

Function	Meaning of Result and Conditions	Data Type of Result
MIN(col)	Minimum value in the col column within the result set	Like the col column
MAX(col)	Maximum value in the col column within the result set	Like the col column
SUM(col)	Sum of the contents of the col column in the result set-col column must be numeric	Like the col column
AVG(col)	Average (arithmetic mean) value of the contents of the col column in the result set-col column must be numeric	Float (F)
COUNT(*)	Number of lines in the result set	Integer (I)

When an INTO clause is used with aggregate expressions, the data object in the INTO clause must provide a structure component or table column with the appropriate type for each aggregate expression in the SELECT statement.

When aggregate functions are used, there is a major difference between cases in which the field list consists exclusively of aggregate expressions, and cases in which the field list also contains field labels that are not arguments to an aggregate function.

© Copyright . All rights reserved.

Prohibida su reproduccion

Field Lists Only with Aggregate Expressions

```
TYPES: BEGIN OF lty_s_flightocc,
         cntall TYPE i,
         minocc TYPE sflight-seatsocc,
         maxocc TYPE sflight-seatsocc,
         sumocc TYPE sflight-seatsocc,
       END OF lty_s_flightocc.
DATA ls_flightocc TYPE lty_s_flightocc.

SELECT COUNT(*) MIN( seatsocc ) MAX( seatsocc ) SUM( seatsocc )
       FROM sflight
       INTO ls_flightocc.
```

Field list only contains aggregate functions → One-line result

MANDT	CARRID	CONNID	FLDATE	SEATSOCC
001	AA	0014	20081217	250
001	AA	0014	20090121	243
001	AA	0017	20081230	243
001	AA	0017	20090102	250
001	AA	0017	20090106	300
001	AA	0017	20090117	143
001	AA	0017	20090119	300
001	AA	0018	20081203	143
001	AA	0018	20081207	155
001	AA	0018	20090106	165
001	AA	0019	20081205	244
001	AA	0019	20090114	312

ls_flightocc

CNTALL	MINOCC	MAXOCC	SUMOCC
12	143	312	2748

Figure 287: SELECT Statement with Aggregate Expressions

If the field list in a SELECT statement contains only aggregate expressions, the result contains a single line. A structure may be specified as the target if an INTO clause is used (similar to the SELECT SINGLE statement). The figure illustrates an example.

Hint:
If the field list only contains the MIN, MAX, SUM, and AVG functions, but not COUNT, an internal table can also be specified as the target. As the result is a single line, only the first line of the internal table is filled.

© Copyright . All rights reserved.

Prohibida su reproduccion

Aggregate Functions with the DISTINCT Addition

```
TYPES: BEGIN OF lty_s_flightocc,
         cntall TYPE i,
         cntcon TYPE i,
         sumocc TYPE sflight-seatsocc,
       END OF lty_s_flightocc.
DATA ls_flightocc TYPE lty_s_flightocc.

SELECT COUNT(*) COUNT( DISTINCT connid ) SUM( DISTINCT seatsocc )
       FROM sflight
       INTO ls_flightocc.
```

Identical values are only taken into account once during totaling

Number of distinct CONNID values

MANDT	CARRID		CONNID	FLDATE	SEATSOCC
001	AA	1	0014	20081217	250
001	AA		0014	20090121	243
001	AA	2	0017	20081230	~~243~~
001	AA		0017	20090102	~~250~~
001	AA		0017	20090106	300
001	AA		0017	20090117	143
001	AA		0017	20090119	~~300~~
001	AA	3	0018	20081203	~~143~~
001	AA		0018	20081207	155
001	AA		0018	20090106	165
001	AA	4	0019	20081205	244
001	AA		0019	20090114	312

ls_flightocc

CNTALL	CNTCON	SUMOCC
12	4	1812

Figure 288: Aggregate Functions with the DISTINCT Addition

In addition to COUNT(*), the COUNT function is used in combination with a DISTINCT clause, in the form COUNTDISTINCT(col).

This aggregate expression returns the number of unique, distinct values in the specified column.

You can use the DISTINCT addition for all other aggregate functions as well, for example, AVG (DISTINCT col). Values that occur multiple times in the column in the selection result are only included once in the aggregate calculation. The table is retrieved before the aggregate is calculated.

> Note:
> In the example, the aggregate SUM (DISTINCT seatsocc) is only intended as an illustration, not as a typical example of the technique. In general, the DISTINCT clause is most useful when applied together with the COUNT function, and is not commonly combined with the MIN, MAX, SUM, and AVG functions.

© Copyright . All rights reserved.

Prohibida su reproduccion

Special Situations

Figure 289: Special Situations

When writing SELECT statements that consist only of aggregate expressions, keep in mind the following considerations:

- Special case 1

  The dataset is empty before aggregate calculation. Unlike SELECT statements, these database accesses return a result even if no matching records are found in the database. In this case, the result of COUNT is zero and the other aggregate functions return initial values. SY-SUBRC is set to 0 and SY-DBCNT to 1.

> Caution:
> If the MIN (col) aggregate function returns an initial value, for example, you cannot conclude that the table actually contains at least one line with the initial value in the col column. It might also mean that the table does not contain any entries that satisfy the WHERE condition.

- Special case 2

  The SELECT statement only contains the COUNT (*) function. If the field list only contains the COUNT (*) aggregate function, you can omit the INTO clause completely. If no data is found in the database, SY-SUBRC is set to 4 and SY-DBCNT to 0.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Caution:
> For performance reasons, do not use the COUNT function to perform existence checks. Use SELECT ... UP TO 1 ROWS WHERE ... instead. While COUNT (*) looks for all records that match the WHERE condition, scanning through the entire table, ... UP TO 1 ROWS stops searching after the first record is found.

Field Lists with Aggregate Expressions and Field Names

Figure 290: GROUP BY Addition

If the field list of a SELECT statement contains field labels in addition to aggregate expressions, the result is always multiline. As a result, it must be loaded into an internal table (array fetch) or processed sequentially (SELECT loop). The SELECT statement must contain the GROUP BY clause and all fields that are not arguments of an aggregate function in the field list must be listed after the GROUP BY clause.

When the GROUP BY clause is specified, the database does not apply the aggregate functions to all the retrieved records together, but instead sorts them into groups first. A group contains all the records that have the same contents in the columns specified after GROUP BY. The aggregate functions are evaluated separately for each of these groups. Each group then corresponds to one line in the selection result.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> All the fields in the field list that are not arguments of an aggregate function must be listed after GROUP BY. GROUP BY can also contain field labels that are not part of the field list - at least theoretically. However, this makes it difficult to trace which line in the result belongs to a specific group. Therefore, every field after GROUP BY should also appear as a column in the selection result.

For the GROUP BY clause, the restrictions are as follows:

- Individual columns are listed after SELECT (SELECT * is not allowed)
- No pooled tables or cluster tables
- The fields after GROUP BY cannot have type STRING or RAWSTRING

HAVING Addition

Figure 291: HAVING Addition

When you use GROUP BY, you can specify a logical expression after the HAVING expression to restrict the result set further.

In contrast to the WHERE condition, the logical expression after HAVING can also contain aggregate functions (in SAP Web AS 6.10 and later). Apart from the aggregate functions, the logical expression after HAVING can only contain fields that are specified after GROUP BY.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Caution:
> If you use field labels after HAVING that are not listed after GROUP BY, this does not cause a syntax error but instead results in a catchable runtime error (CX_SY_OPEN_SQL_DB).

The aggregate functions after HAVING are different from the aggregate functions after SELECT. In the example, you could specify a condition COUNT (*) > 1 after HAVING to only evaluate groups that contain more than one flight. You do not have to use the aggregate COUNT (*) function after SELECT in this case.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Perform Calculations on the Database

Business Example

You want to output the total bookings by currency and an alphabetical list of the involved travel agencies based on the list of bookings.

Template

BC402_DBT_SQL_CONDENSE

Solution

BC402_DBS_SQL_CONDENSE

Use aggregate functions correctly.

Implement subroutine GET_SUMS to calculate the booking totals separated by currency. Program a single database access that returns the data completely in the desired form. Read directly from database table SBOOK.

1. Analyze the interface of the subroutine, particularly the typing of the interface parameters. Which component does the line type of CT_SUMS contain?

   _____

   _____

   _____

2. Implement a SELECT statement to fill CT_SUMS with the booking totals. Access database table SBOOK and limit the access with the customer number.

   Use aggregate functions to have the database calculate the totals.

3. Which addition do you use to calculate the totals separated by currency?

© Copyright . All rights reserved.

Prohibida su reproduccion

Perform Calculations on the Database

Business Example

You want to output the total bookings by currency and an alphabetical list of the involved travel agencies based on the list of bookings.

Template

BC402_DBT_SQL_CONDENSE

Solution

BC402_DBS_SQL_CONDENSE

Use aggregate functions correctly.

Implement subroutine GET_SUMS to calculate the booking totals separated by currency. Program a single database access that returns the data completely in the desired form. Read directly from database table SBOOK.

1. Analyze the interface of the subroutine, particularly the typing of the interface parameters. Which component does the line type of CT_SUMS contain?

   FORCURAM - Price of booking in foreign currency (dependent on booking location), FORCURKEY - Payment currency.

2. Implement a SELECT statement to fill CT_SUMS with the booking totals. Access database table SBOOK and limit the access with the customer number.

   Use aggregate functions to have the database calculate the totals.

   a) See the source code excerpt from the model solution.

3. Which addition do you use to calculate the totals separated by currency?

   a) GROUP BY forcurkey

```
REPORT  bc402_dbs_sql_condense.

TYPES: BEGIN OF gty_s_sums,
         forcuram  TYPE sbook-forcuram,
         forcurkey TYPE sbook-forcurkey,
       END OF gty_s_sums.

TYPES: BEGIN OF gty_s_travelags,
         agencynum  TYPE bc402_scus_book-agencynum,
         name       TYPE bc402_scus_book-name,
         city       TYPE bc402_scus_book-city,
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
       END OF gty_s_travelags.


TYPES:
   gty_t_sums      TYPE SORTED TABLE OF gty_s_sums
                        WITH UNIQUE KEY forcurkey,

   gty_t_bookings  TYPE STANDARD TABLE OF scus_book
                        WITH NON-UNIQUE KEY
                        carrid connid fldate bookid,

   gty_t_travelags TYPE STANDARD TABLE OF gty_s_travelags
                        WITH NON-UNIQUE KEY agencynum.

DATA:
  gs_customer  TYPE scustom,
  gt_bookings  TYPE gty_t_bookings,
  gt_sums      TYPE gty_t_sums,
  gt_travelags TYPE gty_t_travelags.

PARAMETERS:
  pa_cust TYPE sbook-customid DEFAULT '00000001'.


START-OF-SELECTION.

  SELECT SINGLE * FROM scustom INTO gs_customer
         WHERE id = pa_cust.

  SELECT * FROM scus_book
           INTO TABLE gt_bookings
           WHERE customid =  pa_cust
           AND  cancelled <> 'X'.

  PERFORM get_sums USING    pa_cust
                   CHANGING gt_sums.

  PERFORM get_travelags USING    pa_cust
                        CHANGING gt_travelags.

  PERFORM output_list USING gs_customer
                            gt_bookings
                            gt_sums
                            gt_travelags.


*&---------------------------------------------------*
*&      Form  GET_SUMS
*&---------------------------------------------------*
FORM get_sums  USING    pv_customid TYPE sbook-customid
               CHANGING ct_sums     TYPE gty_t_sums.
  SELECT SUM( forcuram ) forcurkey
         FROM sbook
         INTO TABLE ct_sums
         WHERE customid = pv_customid
         GROUP BY forcurkey.
ENDFORM.                     " GET_SUMS
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
*&---------------------------------------------------*
*&      Form  GET_TRAVELAGS
*&---------------------------------------------------*
FORM get_travelags  USING     pv_customid TYPE sbook-customid
                    CHANGING ct_travelags TYPE gty_t_travelags.
  SELECT DISTINCT agencynum name city
         FROM bc402_scus_book
         INTO TABLE ct_travelags
         WHERE customid = pv_customid
         ORDER BY name.
ENDFORM.                    " GET_TRAVELAGS

*&---------------------------------------------*
*&      Form  OUTPUT_LIST
*&---------------------------------------------*
FORM output_list  USING ps_customer  TYPE scustom
                        pt_bookings  TYPE gty_t_bookings
                        pt_sums      TYPE gty_t_sums
                        pt_travelags TYPE gty_t_travelags.

  DATA:
    ls_bookings  LIKE LINE OF pt_bookings,
    ls_sums      LIKE LINE OF pt_sums,
    ls_travelags LIKE LINE OF pt_travelags.

  DATA lv_text TYPE string.

  CONCATENATE text-wcm
              ps_customer-form
              ps_customer-name
              INTO lv_text
              SEPARATED BY space.
  CONDENSE lv_text.

  WRITE: / lv_text.
  SKIP.

  WRITE / text-lob.
  ULINE.
  SKIP.

  LOOP AT pt_bookings INTO ls_bookings.
    WRITE: /
      ls_bookings-bookid,
      ls_bookings-carrid RIGHT-JUSTIFIED,
      ls_bookings-connid ,
      ls_bookings-fldate,
      ls_bookings-cityfrom RIGHT-JUSTIFIED,
      '->',
      ls_bookings-cityto.
  ENDLOOP.
  SKIP 2.

  WRITE: / text-sum.
  ULINE.
  SKIP.

  LOOP AT pt_sums INTO ls_sums.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
    WRITE: /
      ls_sums-forcuram CURRENCY ls_sums-forcurkey,
      ls_sums-forcurkey.
  ENDLOOP.
  SKIP 2.

  WRITE / text-tag.
  ULINE.
  SKIP.

  LOOP AT pt_travelags INTO ls_travelags.
    WRITE: /
      ls_travelags-name,
      ls_travelags-city.
  ENDLOOP.

ENDFORM.                    " OUTPUT_LIST
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Retain ordered or condensed datasets from the database
- Perform calculations on the database

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 14
Lesson 3

Reading from Multiple Database Tables

LESSON OVERVIEW

This lesson discusses the various techniques that can be used to read data from multiple tables.

Business Example

You want to implement database access with Open SQL in your applications. You want the data access of your application to be efficient and effective. Cases in which data has to be read from several databases in combination are especially performance-critical. Therefore, you need to learn about the techniques you can use to improve performance when accessing multiple database tables.

For this reason, you require the following knowledge:

- An understanding of how to use views and joins correctly
- An understanding of subqueries and their usage
- An understanding of how to use the FOR ALL ENTRIES clause correctly

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Identify disadvantages of nested SELECT
- Use ABAP joins and database views
- Explain special techniques for reading from multiple database tables
- Use explicit buffering techniques

© Copyright . All rights reserved.

Prohibida su reproduccion

The Problem with Nested Selects

Figure 292: Nested SELECTs

In a relational database system such as Open SQL, attempts to access data stored in a primary table require reads from the associated secondary tables as well. The obvious programmatic method to perform those accesses is to loop over the records in the primary table with nested SELECTs (see figure 191 for an example). However, secondary tables may themselves have associated tables that need to be read in turn. There is no theoretical or practical limit to the nesting depth; the nested SELECT model can indefinitely chain tables as long as the tables are linked by common data columns.

For a number of reasons, nested SELECTS are the most resource-intensive method possible for reading multiple linked database tables, and good program design will avoid this method for any but the smallest and simplest of data access procedures. Nested SELECTS place a high load on the database server and produce large amounts of network traffic, because many partially-filled data packages are transferred and the same data may be read many times in a row (identical accesses).

In this lesson, you will learn several techniques that enable you to largely avoid nested SELECTs, and to reduce their negative impact on system performance when their use is unavoidable.

© Copyright . All rights reserved.

Prohibida su reproduccion

ABAP-Joins

Figure 293: Example – Inner or Outer Join

To read data that is distributed across multiple tables, create a link between the functionally dependent tables. The link between the tables is known technically as a join, and the corresponding database operator is join.

To implement a join, you can use either database views in the ABAP dictionary or ABAP joins.

> Hint:
> Database views and ABAP joins can only be used with transparent tables. If you are working with pooled tables or cluster tables, you have to use other techniques.

You can derive the logic of the inner join and outer join from the intended result set.

An inner join produces the result set that only considers the records from the outer table for which suitable data records exist in the inner table (as in the example).

A left outer join produces the result set that contains all the records from the outer table, regardless of whether or not suitable records exist in the inner table. If no suitable records exist in the inner table, the fields of the inner (the other table that is used in the left outer join condition) table are set to zero values in the result set. The tables involved in a join are called base tables. A projection (column selection) or a selection (line selection) can be applied on the result of a join.

Not all of the databases supported by SAP support the standard syntax for ON conditions. Therefore, use restricted syntax to ensure that only joins that return the same result set on all database systems are allowed.

© Copyright . All rights reserved.

Prohibida su reproduccion

Restrictions for an outer join are as follows:

- You can only have a table or a view to the right of the join operator. You cannot have another join expression.
- Only AND can be used as a logical operator in an ON condition.
- Every comparison in the ON condition must contain a field from the table on the right.
- None of the fields in the table on the right can appear in the WHERE conditions of the left outer join.

For more information about implementing join functions, see the ABAP documentation.

Example – ABAP Inner Join

Example: ABAP INNER Join

```
SELECT <fieldlist> INTO <target>
  FROM <dbtab1> [AS <alias1>]
  INNER JOIN <dbtab2> [AS <alias2>]
  ON <alias1>~<dbtab1-field1> = <alias2>~<dbtab2-field1>
  AND <alias1>~<dbtab1-field2> = <alias2>~<dbtab2-field2>
  AND ...
  WHERE ...
  ...
ENDSELECT.
```

```
REPORT zselect_view.

SELECT f~carrid b~connid ... INTO ( ... )
  FROM scarr AS f INNER JOIN spfli AS b
  ON f~carrid = b~carrid
  WHERE ...
 ...
ENDSELECT.
```

Figure 294: Example – ABAP Inner Join

A disadvantage of using ABAP joins is that the statement is more complex than a dictionary view, where the syntax of the SELECT statement (specifically, the FROM clause) corresponds to a regular table access.

Hint:
Note that ABAP joins always bypass the table buffer, which is only a disadvantage for joins involving buffered tables.

Dictionary views can be buffered in SAP R/3 Release 4.0 and later (depending on which tables are used in the view).

Properties and Advantages of Database Views

Attributes and benefits of database views are as follows:

© Copyright . All rights reserved.

Prohibida su reproduccion

- You can use views in other programs as well.
- You can use views for lists and search functions; for example, SE84 and SE81 find existing views quickly.
- You can buffer views (technical settings) like database tables.
- Fields common to both tables (join fields) are only transferred from the database to the application server once.
- The view is implemented in the ABAP Dictionary as an inner join. This means no data is transferred if the inner table does not contain any entries that correspond to the outer table.
- If you do not want to use an inner join to read from a text table, use an ABAP left outer join. For example, if you have a situation where the results of an inner join do not contain any records because no entry is available in a certain language, you can use an ABAP left outer join instead (see the figure “Example – Inner or Outer Join”).

Hint:
A disadvantage of both ABAP joins and database views is that redundant data from the outer table appears in the result set if there is a 1:n relationship between the outer and inner tables. This can considerably increase the amount of data transferred from the database. For this reason, use a field list with a join to retrieve only those fields that are needed. The runtime of a join formation is highly dependent on the query optimizer, especially when the join involves more than two tables. However, it is usually faster than using nested SELECT statements.

© Copyright . All rights reserved.

Prohibida su reproduccion

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 14
Exercise 35

Implement a Join of Three Database Tables

Business Example

Your company uses a custom program that displays a list of bookings for a selection of travel agencies, customers, and flight data.

Users frequently complain about long runtimes.

An analysis shows that the poor performance is due to numerous single record accesses of tables SCARR and STRAVELAG within a loop.

You have been asked to redesign the program so it can read the necessary data in one consolidated database access.

Template

BC402_DBT_3JOIN

Solution

BC402_DBS_3JOIN

Formulate joins over multiple tables, explain the difference between inner and outer joins, and understand why using joins for buffered tables negatively impacts performance.

Task 1

Copy template BC402_DBT_3JOIN to **`ZBC402_##_3JOIN`**, where **`##`** is your group number.

Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
2. From which database tables does the program read the data? Which tables are read in a loop?

   _____

   _____

   _____

3. The database administrators detected the repeated single record accesses for tables SCARR and STRAVELAG, but not for SCUSTOM. Do you have an explanation for this?

   _____

   _____

   _____

© Copyright . All rights reserved.

Prohibida su reproduccion

4. In this situation, does it make sense to combine the access to SCUSTOM with an access to the other tables in a JOIN?

_____

_____

_____

Task 2

Implement a database access that reads from several tables together in a JOIN. Select the three tables for which this makes sense. Note where an outer join is required to ensure the same number of bookings is read. Only read the columns that are required. Enhance structure type GTY_S_BOOKINGS with the appropriate components.

1. Enhance the database access to table SBOOK. Change the FROM clause to read the necessary data from tables SCARR and STRAVELAG in a triple join. Note that SBOOK also contains postings that were not made by travel agencies, which means the *agencynum* field is INITIAL (suggesting that it is empty).
2. Which changes do you have to make to the WHERE condition?

   The *agencynum* field is no longer unique. Therefore, you have to prefix it with the table name, SBOOK, or an appropriate alias.

   See the source code excerpt from the model solution.
3. In the SELECT clause, add the fields you want to read from database tables SCARR and STRAVELAG. Enhance the line type of the internal table (structure type GTY_S_BOOKINGS) with appropriately typed components.

   Which other changes in the SELECT clause do you have to make?

Task 3

Remove or comment out unnecessary statements and declarations. Activate and test your program.

1. Remove or comment out the SELECT statements for SCARR and STRAVELAG.
2. In the WRITE statement, output the additional fields from internal table GT_BOOKINGS.
3. Remove or comment out the declaration of the unnecessary data objects.
4. Activate and test your program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement a Join of Three Database Tables

Business Example

Your company uses a custom program that displays a list of bookings for a selection of travel agencies, customers, and flight data.

Users frequently complain about long runtimes.

An analysis shows that the poor performance is due to numerous single record accesses of tables SCARR and STRAVELAG within a loop.

You have been asked to redesign the program so it can read the necessary data in one consolidated database access.

Template

BC402_DBT_3JOIN

Solution

BC402_DBS_3JOIN

Formulate joins over multiple tables, explain the difference between inner and outer joins, and understand why using joins for buffered tables negatively impacts performance.

Task 1

Copy template BC402_DBT_3JOIN to **`ZBC402_##_3JOIN`**, where **`##`** is your group number.

Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
   a) Perform this step in the standard manner.
2. From which database tables does the program read the data? Which tables are read in a loop?

   The database tables from which the program reads the data are SBOOK - Flight bookings, SCUSTOM - Flight customers, SCARR - Airlines, and STRAVELAG - Travel agencies. The database tables from which the program reads the data in the loop are SCUSTOM - Flight customers, SCARR - Airlines, and STRAVELAG - Travel agencies.

© Copyright . All rights reserved.

Prohibida su reproduccion

3. The database administrators detected the repeated single record accesses for tables SCARR and STRAVELAG, but not for SCUSTOM. Do you have an explanation for this?

   Table SCUSTOM uses single-record buffering, which means the data for a specific customer is only read from the database once on a given server.

4. In this situation, does it make sense to combine the access to SCUSTOM with an access to the other tables in a JOIN?

   No, because the buffering of SCUSTOM is ignored in a JOIN.

Task 2

Implement a database access that reads from several tables together in a JOIN. Select the three tables for which this makes sense. Note where an outer join is required to ensure the same number of bookings is read. Only read the columns that are required. Enhance structure type GTY_S_BOOKINGS with the appropriate components.

1. Enhance the database access to table SBOOK. Change the FROM clause to read the necessary data from tables SCARR and STRAVELAG in a triple join. Note that SBOOK also contains postings that were not made by travel agencies, which means the *agencynum* field is INITIAL (suggesting that it is empty).
   a) See the source code excerpt from the model solution.

      The triple join consists of an inner join between SBOOK and SCARR (the *carrid* field is filled for all bookings) and a left outer join between this result and table STRAVELAG.

2. Which changes do you have to make to the WHERE condition?

   The *agencynum* field is no longer unique. Therefore, you have to prefix it with the table name, SBOOK, or an appropriate alias.

   See the source code excerpt from the model solution.

3. In the SELECT clause, add the fields you want to read from database tables SCARR and STRAVELAG. Enhance the line type of the internal table (structure type GTY_S_BOOKINGS) with appropriately typed components.

   Which other changes in the SELECT clause do you have to make?

   a) The *carrid* and *agencynum* fields are no longer unique. Therefore, you have to prefix them with the table name, SBOOK, or an appropriate alias.

      See the source code excerpt from the model solution.

Task 3

Remove or comment out unnecessary statements and declarations. Activate and test your program.

1. Remove or comment out the SELECT statements for SCARR and STRAVELAG.
   a) See the source code excerpt from the model solution.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. In the WRITE statement, output the additional fields from internal table GT_BOOKINGS.
   a) See the source code excerpt from the model solution.
3. Remove or comment out the declaration of the unnecessary data objects.
   a) See the source code excerpt from the model solution.
4. Activate and test your program.
   a) Perform this step in the standard manner.
      Source code excerpt from the model solution:

```
REPORT  bc402_dbs_3join.

TYPES: BEGIN OF gty_s_booking,
         carrid     TYPE sbook-carrid,
         connid     TYPE sbook-connid,
         fldate     TYPE sbook-fldate,
         bookid     TYPE sbook-bookid,
         customid   TYPE sbook-customid,
         agencynum  TYPE sbook-agencynum,
         carrname   TYPE scarr-carrname,
         agencyname TYPE stravelag-name,
         agencycity TYPE stravelag-city,
      END OF gty_s_booking.

TYPES:
   gty_t_bookings TYPE STANDARD TABLE OF gty_s_booking
                      WITH NON-UNIQUE KEY
                      carrid connid fldate bookid.

DATA:
  gt_bookings TYPE gty_t_bookings,
  gs_booking  TYPE gty_s_booking.

DATA:
  gv_custname   TYPE scustom-name.
* gv_carrname   TYPE scarr-carrname,
* gv_agencyname TYPE stravelag-name,
* gv_agencycity TYPE stravelag-city.
FIELD-SYMBOLS:
  <fs_booking> LIKE LINE OF gt_bookings.

SELECT-OPTIONS :
      so_agy FOR gs_booking-agencynum DEFAULT '100',
      so_cus FOR gs_booking-customid,
      so_fld FOR gs_booking-fldate.


START-OF-SELECTION.
  SELECT b~carrid b~connid b~fldate b~bookid
         b~customid b~agencynum
         c~carrname a~name a~city
         FROM sbook AS b INNER JOIN scarr AS c
           ON b~carrid   = c~carrid
           LEFT OUTER JOIN stravelag AS a
             ON b~agencynum = a~agencynum
         INTO TABLE gt_bookings
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
         WHERE b~agencynum IN so_agy AND
               b~customid  IN so_cus AND
               b~fldate    IN so_fld AND
               cancelled   <> 'X'.
  LOOP AT gt_bookings ASSIGNING <fs_booking>.

* scustom is buffered - no need for optimizations
    SELECT SINGLE name FROM scustom INTO gv_custname
           WHERE id = <fs_booking>-customid.
*   SELECT SINGLE carrname FROM scarr
*          INTO gv_carrname
*          WHERE carrid = <fs_booking> -carrid.
*
*   SELECT SINGLE name city FROM stravelag
*          INTO (gv_agencyname, gv_agencycity)
*          WHERE agencynum = <fs_booking> -agencynum.
    WRITE: /
        <fs_booking>-carrid,
*       gv_carrname,
        <fs_booking>-carrname,
        <fs_booking>-connid,
        <fs_booking>-fldate,
        <fs_booking>-bookid,
        gv_custname,
*       gv_agencyname,
*       gv_agencycity.
        <fs_booking>-agencyname,
        <fs_booking>-agencycity.

  ENDLOOP.
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Subselects and Subqueries

Subquery returns single value

```
SELECT * FROM sflight
         INTO TABLE lt_flights
         WHERE seatsocc =
                     ( SELECT MAX( seatsocc ) FROM sflight ).
```

Select all flights in SFLIGHT with a maximum of passengers

Subquery returns single-column, multiline result

```
SELECT * FROM scarr
         INTO TABLE lt_carriers
         WHERE carrid IN
                      ( SELECT DISTINCT carrid FROM spfli
                               WHERE cityfrom = 'FRANKFURT' ).
```

Select airlines with departure city "Frankfurt"

Subquery returns any result

```
SELECT * FROM scarr
         INTO TABLE lt_carriers
         WHERE EXISTS ( SELECT carrid FROM spfli
                                WHERE carrid = scarr~carrid
                                AND cityfrom = 'FRANKFURT' ).
```

Select airlines with departure city "Frankfurt"

Figure 295: Subselects and Subqueries

A subquery is a query within a SELECT, UPDATE, or DELETE statement. It is formulated in the WHERE or HAVING clause to check whether the data in various database tables or views possess certain attributes.

A SELECT statement with a subquery has a more restricted syntax than a SELECT statement without a subquery.

If the subquery returns exactly one value, use the usual comparison operators apart from LIKE and BETWEEN. If you use a subquery with a comparison operator instead of with EXISTS, then the SELECT clause of the subquery can only contain a single column, which can be a field in the database table or an aggregate expression.

In this example, the subquery is supposed to return several lines, each with one value. If you want to compare all the returned values, use IN. Subqueries whose WHERE condition contains fields from the main query are called correlated subqueries. If subqueries are nested, each subquery can use all the fields from the higher-level subqueries in the hierarchy. Query designers should formulate positive subqueries whenever possible; negative formulations may result in performance-degrading database reads if no adequate index is available.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> In many cases, you can also use a join to obtain the same result as that produced with a subquery. The SQL code for a join is easier for other developers to read and understand, but no blanket statements can be made about the relative performance of subqueries and joins, because the table design, query logic, and amount of data to be retrieved all can have major impacts on the performance of both methods. For performance-critical systems, it is advisable to prototype the query using each method and running performance profiles with real-world data to ascertain which approach will produce faster output.

Parallel Cursors

Figure 296: Initial Situation – Nested SELECTs

You know how to use the Open SQL statements OPEN CURSOR, FETCH NEXT CURSOR, and CLOSE CURSOR to program explicit cursor handling. You also know how cursors are opened in several database tables at the same time. This knowledge enables you process the contents of multiple database tables in parallel without using nested SELECTs.

You want the program to process all flights from database table SFLIGHT, and you want to read the non-canceled bookings in the economy class for each flight from database table SBOOK and process the bookings together with the flight. The obvious solution is a nested SELECT, but that solution is likely to perform slowly.

Due to the large data volume involved, do not buffer the bookings completely in an internal table.

© Copyright . All rights reserved.

Prohibida su reproduccion

Parallel Cursors Instead of Nested SELECTs

Figure 297: Parallel Cursors Instead of Nested SELECTs

Rather than solving the problem with nested SELECTs, you can achieve the same result by addressing the two database tables with two concurrently open database cursors (see Figure 195: SELECT DISTINCT Statement).

The concurrent cursor approach works as follows:

- Each fetch for the first cursor (cursor variable LV_CURSOR_SFLIGHT) places the next respective record from database table SFLIGHT into structure LS_FLIGHT (similar to a SELECT loop).
- Each fetch for the second cursor (cursor variable LV_CURSOR_SBOOK) reads the corresponding non-canceled economy bookings into internal table LT_BOOKINGS. However, these bookings are not actually selected based on their key. Instead, the next n records are simply read. The value of n corresponds to the value of the seatsocc field in the record for the flight. Because the selection results are sorted (ORDER BY clause), the next n bookings are the bookings for the current flight.

For this technique to work correctly, the table and record design need to fulfill the following prerequisites:

- Both tables must be sorted by the same criteria (ORDER BY clause).
- Every record in the outer table must specify exactly (and correctly) how many corresponding records exist in the inner table.

If these conditions for the data are not met, then the concurrent cursor program design will return incorrect results, or fail to work at all. It is therefore a less robust, though much better-performing, approach to the problem.

© Copyright . All rights reserved.

Prohibida su reproduccion

> Hint:
> To circumvent the second criterion and improve the robustness of the solution, you can open two cursors for the same database table and have the outer cursor use the *count (*)* aggregate function to determine the number of records in each group.

Motivation and Basic Idea

Avoiding database accesses in the first place is the simplest way to reduce database load. You can do this by taking table contents that have already been read and saving them in an internal table (with type SORTED or HASHED, if possible).

In many cases, the application logic does not permit the use of joins. If you need to read data from multiple tables in such cases, you can use a separate internal table to buffer the read data.

Read on Demand and Buffering

```
LOOP AT gt_bookings INTO gs_booking.
  PERFORM read_travelag USING    gs_booking-agencynum
                        CHANGING gs_travelag.
  ...
ENDLOOP.
```

```
FORM read_travelag USING    pv_agencynum
                   CHANGING cs_travelag.

  STATICS st_travelags TYPE HASHED TABLE OF stravelag
                       WITH UNIQUE KEY agencynum.

  IF pv_agencynum IS NOT INITIAL.
    READ TABLE st_travelags INTO cs_travelag
         WITH TABLE KEY agencynum = pv_agencynum.

    IF sy-subrc <> 0.
      SELECT SINGLE * FROM stravelag
                      INTO cs_travelag
                      WHERE agencynum = pv_agencynum.
      INSERT cs_travelag INTO TABLE st_travelags.
    ENDIF.
  ENDIF.
ENDFORM.
```

Static table: Retains the data Hashed table: High-performance read access

First check whether record is already buffered

Only read from the database if necessary

Figure 298: Read on Demand and Buffering

In the example, you want to read the matching data from STRAVELAG (travel agency master data) within a loop for the bookings from table SBOOK. Accomplish this with a SELECT SINGLE statement or a read routine. If you use SELECT SINGLE in this approach, execute identical SQL statements. Therefore encapsulate the reading of STRAVELAG records in a read routine. In the example, the table contents read from STRAVELAG are buffered in a static internal table. Before each database access, the system checks whether the corresponding table entry has already been read.

The read routine can also be a method of a local or global class instead of a subroutine. In this case, a private attribute of the corresponding class is used as buffer instead of the static internal table ST_TRAVELAGS.

© Copyright . All rights reserved.

Prohibida su reproduccion

Unit 14
Exercise 36

Implement Full Buffering and Buffering on Demand

Business Example

Your company uses a standard SAP program that creates a list of bookings for a customer. This program has been enhanced as follows to meet your company's needs:

1. An append was used to add several fields to structure type BC402_S_BOOKING.
2. A BAdI implementation was created to fill the additional fields.

After a release upgrade, you discover that SAP has enhanced the BAdI with a method that is called before the loop over the bookings.

You realize that you can improve your enhancement by moving the filling of the buffer (instance attributes) to an implementation of this new method.

> Hint:
> To avoid making this exercise unnecessarily complicated, you do not use an actual BAdI. Instead, simulate the BAdI implementation using a local class. Only change the definition and implementation of this local class during the exercise. Consider that the source code outside of this class represents the SAP program and is unchangeable.

Template

BC402_DBT_BUFFER

Solution

BC402_DBS_BUFFER

To buffer data completely when it is used in a loop and read data in a loop on demand and then buffer it, perform the following tasks:

Task 1

Copy template BC402_DBT_BUFFER to **`ZBC402_##_BUFFER`**, where **`##`** is your group number. Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. Which methods does class LCL_IM_BADI have? Which parameters exist in these methods? How are they typed?

3. Display dictionary structure type BC402_S_BOOKING. Which fields were added in an append?

4. Where is method ADD_DATA of class LCL_IM_BADI called?

5. Which database tables are accessed within the local class LCL_IM_BADI?

Task 2

Edit the definition of class LCL_IM_BADI. Create two table-type private instance attributes to buffer the data that is read from the database in method ADD_DATA. Create private structure types and table types for this, also in the class definition.

1. Create a structure type in the private section of the class definition whose components include the key and required fields from table (suggested name: **`TY_S_SCARR`**).
2. Create a table type in the private section of the class definition that has the structure type as line type (suggested name: **`TY_T_SCARR`**).
3. Which table type is particularly suited to this type of buffering?

4. Create an instance attribute in the private section of the class definition (suggested name: **`mt_carriers`**) and type it with this table type.
5. Create an appropriate structure type, table type, and instance attribute to buffer the data from database table STRAVELAG (suggested names: **`ty_s_stravelag, ty_t_stravelag, mt_travelags`**).

Task 3

© Copyright . All rights reserved.

Prohibida su reproduccion

Edit the implementation of method ADD_DATA.

Make sure the data is buffered in the instance attributes of the class after it is read from the database.

Full buffering is possible for one of the tables. Which one? Make sure that all entries in this database table are read and buffered the first time the method is called.

Full buffering does not make sense for the other database table. Why not? Make sure the data from this table is read and buffered individually on demand.

1. Which database table is a candidate for full buffering? Why?

   _____

   _____

   _____

2. Program a database access in which the database table that you want to buffer is loaded completely into the corresponding attribute. Ensure that the access is only made during the first call of the ADD_DATA method.
3. Replace the single record access to the database table with an access to the internal table (the *Instance* attribute).
4. Make sure that each time the database table that you do not want to buffer is accessed (on a single record basis), the individual data is buffered in the *Instance* attribute.
5. Implement an access to the internal table (the *Instance* attribute) before the database access. Make sure the database access is skipped if the corresponding record is already found in the internal table.
6. Activate and test your program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Implement Full Buffering and Buffering on Demand

Business Example

Your company uses a standard SAP program that creates a list of bookings for a customer. This program has been enhanced as follows to meet your company's needs:

1. An append was used to add several fields to structure type BC402_S_BOOKING.
2. A BAdI implementation was created to fill the additional fields.

After a release upgrade, you discover that SAP has enhanced the BAdI with a method that is called before the loop over the bookings.

You realize that you can improve your enhancement by moving the filling of the buffer (instance attributes) to an implementation of this new method.

> Hint:
> To avoid making this exercise unnecessarily complicated, you do not use an actual BAdI. Instead, simulate the BAdI implementation using a local class. Only change the definition and implementation of this local class during the exercise. Consider that the source code outside of this class represents the SAP program and is unchangeable.

Template

BC402_DBT_BUFFER

Solution

BC402_DBS_BUFFER

To buffer data completely when it is used in a loop and read data in a loop on demand and then buffer it, perform the following tasks:

Task 1

Copy template BC402_DBT_BUFFER to **`ZBC402_##_BUFFER`**, where **`##`** is your group number. Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
   a) Perform this step in the usual manner.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. Which methods does class LCL_IM_BADI have? Which parameters exist in these methods? How are they typed?

   The class only has one method, ADD_DATA. The only parameter is changing parameter CS_BOOKING, which is typed with dictionary structure type BC402_S_BOOKING.

3. Display dictionary structure type BC402_S_BOOKING. Which fields were added in an append?

   CARRNAME, AGENCYNAME, and AGENCYCITY.

4. Where is method ADD_DATA of class LCL_IM_BADI called?

   Within a loop of internal table GT_BOOKINGS.

5. Which database tables are accessed within the local class LCL_IM_BADI?

   The database tables SCARR - Airlines and STRAVELAG - Travel agencies are accessed within the local class LCL_IM_BADI.

Task 2

Edit the definition of class LCL_IM_BADI. Create two table-type private instance attributes to buffer the data that is read from the database in method ADD_DATA. Create private structure types and table types for this, also in the class definition.

1. Create a structure type in the private section of the class definition whose components include the key and required fields from table (suggested name: **`TY_S_SCARR`**).
   a) See the source code excerpt from the model solution.
2. Create a table type in the private section of the class definition that has the structure type as line type (suggested name: **`TY_T_SCARR`**).
   a) See the source code excerpt from the model solution.
3. Which table type is particularly suited to this type of buffering?

   A hashed table, because the same unique key is always used to access the content later in single record access.

4. Create an instance attribute in the private section of the class definition (suggested name: **`mt_carriers`**) and type it with this table type.
   a) See the source code excerpt from the model solution.
5. Create an appropriate structure type, table type, and instance attribute to buffer the data from database table STRAVELAG (suggested names: **`ty_s_stravelag, ty_t_stravelag, mt_travelags`**).
   a) See the source code excerpt from the model solution.

© Copyright . All rights reserved.

Prohibida su reproduccion

Task 3

Edit the implementation of method ADD_DATA.

Make sure the data is buffered in the instance attributes of the class after it is read from the database.

Full buffering is possible for one of the tables. Which one? Make sure that all entries in this database table are read and buffered the first time the method is called.

Full buffering does not make sense for the other database table. Why not? Make sure the data from this table is read and buffered individually on demand.

1. Which database table is a candidate for full buffering? Why?

   Database table SCARR. It only has a few entries, so the probability that many of these entries will actually be needed is high. In contrast, database table STRAVELAG is fairly large, and you can assume that a given flight customer has only booked flights at a few travel agencies.

2. Program a database access in which the database table that you want to buffer is loaded completely into the corresponding attribute. Ensure that the access is only made during the first call of the ADD_DATA method.
   a) See the source code excerpt from the model solution.
3. Replace the single record access to the database table with an access to the internal table (the *Instance* attribute).
   a) See the source code excerpt from the model solution.
4. Make sure that each time the database table that you do not want to buffer is accessed (on a single record basis), the individual data is buffered in the *Instance* attribute.
   a) See the source code excerpt from the model solution.
5. Implement an access to the internal table (the *Instance* attribute) before the database access. Make sure the database access is skipped if the corresponding record is already found in the internal table.
   a) See the source code excerpt from the model solution.
6. Activate and test your program.
   a) Perform this step in the standard manner.

   Interface lif_badi

```
INTERFACE lif_badi.
  METHODS:
      add_data
           CHANGING cs_booking TYPE bc402_s_booking.
ENDINTERFACE.                    "lif_badi
```


© Copyright . All rights reserved.

Prohibida su reproduccion

Class lcl_im_badi (Definition)

```
CLASS lcl_im_badi DEFINITION.
  PUBLIC SECTION.
    INTERFACES lif_badi.
  PRIVATE SECTION.

    TYPES: BEGIN OF ty_s_scarr,
             carrid TYPE scarr-carrid,
             carrname TYPE scarr-carrname,
           END OF ty_s_scarr.

    TYPES: ty_t_scarr TYPE HASHED TABLE OF ty_s_scarr
                      WITH UNIQUE KEY carrid.

    TYPES: BEGIN OF ty_s_stravelag,
             agencynum  TYPE stravelag-agencynum,
             name       TYPE stravelag-name,
             city       TYPE stravelag-city,
           END OF ty_s_stravelag.

    TYPES: ty_t_stravelag TYPE HASHED TABLE OF ty_s_stravelag
                          WITH UNIQUE KEY agencynum.

    DATA: mt_carriers  TYPE ty_t_scarr,
          mt_travelags TYPE ty_t_stravelag.
ENDCLASS.                    "lcl_im_badi DEFINITION
```

Class lcl_im_badi (Implementation)

```
CLASS lcl_im_badi IMPLEMENTATION.
  METHOD lif_badi~add_data.
    DATA:
        ls_carrier  LIKE LINE OF mt_carriers,
        ls_travelag LIKE LINE OF mt_travelags.

* carrname - Buffer full
*   SELECT SINGLE carrname FROM scarr
*          INTO cs_booking-carrname
*          WHERE carrid = cs_booking-carrid.

    IF mt_carriers IS INITIAL.
      SELECT carrid carrname FROM scarr
             INTO TABLE mt_carriers.
    ENDIF.

    READ TABLE mt_carriers INTO ls_carrier
         WITH TABLE KEY carrid = cs_booking-carrid.

    cs_booking-carrname = ls_carrier-carrname.

* agencyname and agencycity - buffer on demand

*   SELECT SINGLE name city FROM stravelag
*          INTO (cs_booking-agencyname,
*                cs_booking-agencycity)
```

© Copyright . All rights reserved.

Prohibida su reproduccion

```
*          WHERE agencynum = cs_booking-agencynum.

    IF cs_booking-agencynum IS NOT INITIAL.

      READ TABLE mt_travelags INTO ls_travelag
           WITH TABLE KEY agencynum = cs_booking-agencynum.
      IF sy-subrc <> 0.
        SELECT SINGLE agencynum name city
               FROM stravelag
               INTO ls_travelag
               WHERE agencynum = cs_booking-agencynum.

        INSERT ls_travelag INTO TABLE mt_travelags.
      ENDIF.

      cs_booking-agencyname = ls_travelag-name.
      cs_booking-agencycity = ls_travelag-city.

    ENDIF.
  ENDMETHOD.                    "lif_badi~add_data

ENDCLASS.                    "lcl_im_badi IMPLEMENTATION
```


© Copyright . All rights reserved.

Prohibida su reproduccion

FOR ALL ENTRIES Addition

Figure 299: SELECT with FOR ALL ENTRIES Addition

SELECT ... FOR ALL ENTRIES was created in Open SQL at a time when it was not possible to perform database joins (this was not supported for all SAP-approved DBMS). The connection between the inner and outer database tables was created in ABAP.

Nowadays, this technique is often used when some data is already available in an internal table, but additional data is to be read from the database. In such cases, SELECT ... FOR ALL ENTRIES replaces a SELECT SINGLE statement inside a LOOP over the internal table, and normally shows better performance than such a LOOP.

> Note:
> As a possible scenario, use a program exit in which data can be added to an internal table provided by the standard SAP program.

The figure illustrates an example of how the statement works.

The content of the internal table (driving table) is used as a restriction for the database access, and may depend on the database system. In general, the database interface takes a certain number of entries in the internal table (in the example it takes 5) and sends one native SQL statement to the database for each group. In the end, the results of the individual native SQL statements are combined to form the result of the Open SQL statement. Duplicate entries are automatically removed. Note that the size of the packages is controlled by a profile parameter.

© Copyright . All rights reserved.

Prohibida su reproduccion

Caution:
If you use FOR ALL ENTRIES, make sure that the driving table is not blank. If the table is blank, it provides no boundary condition for the SELECT statement and often the full database table is read. This can cause very long execution times for the statement.

Hint:
Because duplicates are automatically removed from the result, the driving table does not have to be free of duplicate entries. However, you can improve the performance of the statement if you condense the driving table before the FOR ALL ENTRIES statement (maybe by using a copy of the table). If you do not remove duplicate entries from the driving table, the same data will be read multiple times from the database and only removed in the final combination of the result.

If you want to read large data volumes, use FOR ALL ENTRIES only in exceptional cases.

© Copyright . All rights reserved.

Prohibida su reproduccion

Read Additional Data with the FOR ALL ENTRIES addition

Business Example

Your company uses a standard SAP program that creates a list of bookings for a customer. This program has been enhanced as follows to meet your company's needs:

1. An append was used to add several fields to structure type BC402_S_BOOKING.
2. A BAdI implementation was created to fill the additional fields.

After a release upgrade, you discover that SAP has enhanced the BAdI with a method that is called before the loop over the bookings.

You realize that you can improve your enhancement by moving the filling of the buffer (instance attributes) to an implementation of this new method.

> Hint:
> To avoid making this exercise unnecessarily complicated, you do not use an actual BAdI. Instead, simulate the BAdI implementation using a local class. Only change the definition and implementation of this local class during the exercise. You should consider the source code outside of this class, which represents the SAP program, to be unchangeable.

Template

BC402_DBT_FOR_ALL_ENTRIES

Solution

BC402_DBS_FOR_ALL_ENTRIES

To use the FOR ALL ENTRIES addition correctly in the WHERE condition of a SELECT statement, perform the following tasks:

Task 1

Copy template BC402_DBT_FOR_ALL_ENTRIES to **`ZBC402_##_FOR_ALL_ENTRIES`**, where **`##`** is your group number. Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
   Perform this step in the standard manner.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. Which methods does class LCL_IM_BADI have? Which parameters exist in these methods? How are they typed?

3. Where is the prepare method of class LCL_IM_BADI called?

Task 2

Edit the implementation of method PREPARE.

Program a database access to the database tables SCARR and STRAVELAG. Fill instance attribute MT_CARRIERS with the data for all airlines and fill instance attribute MT_TRAVELAGS with the data for the travel agencies. Make sure you do not buffer the data for all the travel agencies, but instead only for those you require during the loop pass.

1. Copy the filling of MT_CARRIERS from method ADD_DATA to method PREPARE.
2. Do you still need the query that determines whether this call is the first call?

3. Program a SELECT statement (array fetch) in which you fill instance attribute MT_TRAVELAGS. Use the contents of the IT_BOOKINGS import parameter to ensure that only the travel agencies that appear in the bookings in IT_BOOKINGS are read.
4. Why is it essential to check whether IT_BOOKINGS is filled? What would happen if IT_BOOKINGS were blank?

Task 3

Edit the implementation of method ADD_DATA.

Remove all the database accesses and replace them with accesses to the instance attributes with the buffered data.

1. Remove the SELECT statement for database table SCARR.
2. Remove the SELECT statement for database buffer STRAVELAG as well as the filling of the buffer.

© Copyright . All rights reserved.

Prohibida su reproduccion

3. Activate and test your program.

© Copyright . All rights reserved.

Prohibida su reproduccion

Read Additional Data with the FOR ALL ENTRIES addition

Business Example

Your company uses a standard SAP program that creates a list of bookings for a customer. This program has been enhanced as follows to meet your company's needs:

1. An append was used to add several fields to structure type BC402_S_BOOKING.
2. A BAdI implementation was created to fill the additional fields.

After a release upgrade, you discover that SAP has enhanced the BAdI with a method that is called before the loop over the bookings.

You realize that you can improve your enhancement by moving the filling of the buffer (instance attributes) to an implementation of this new method.

> Hint:
> To avoid making this exercise unnecessarily complicated, you do not use an actual BAdI. Instead, simulate the BAdI implementation using a local class. Only change the definition and implementation of this local class during the exercise. You should consider the source code outside of this class, which represents the SAP program, to be unchangeable.

Template

BC402_DBT_FOR_ALL_ENTRIES

Solution

BC402_DBS_FOR_ALL_ENTRIES

To use the FOR ALL ENTRIES addition correctly in the WHERE condition of a SELECT statement, perform the following tasks:

Task 1

Copy template BC402_DBT_FOR_ALL_ENTRIES to **`ZBC402_##_FOR_ALL_ENTRIES`**, where **`##`** is your group number. Familiarize yourself with the program and how it works.

1. Copy the program and all its subcomponents.
   Perform this step in the standard manner.

© Copyright . All rights reserved.

Prohibida su reproduccion

2. Which methods does class LCL_IM_BADI have? Which parameters exist in these methods? How are they typed?

   Class LCL_IM_BADI has a method called ADD_DATA and another one called PREPARE. The only parameter in the ADD_DATA method is the changing parameter, CS_BOOKING, which is typed with dictionary structure type BC402_S_BOOKING. The only parameter in the prepare method is the import parameter, IT_BOOKINGS, which is typed with dictionary table type BC402_T_BOOKINGS.

3. Where is the prepare method of class LCL_IM_BADI called?

   The prepare method of class LCL_IM_BADI is called directly before the loop over internal table GT_BOOKINGS.

Task 2

Edit the implementation of method PREPARE.

Program a database access to the database tables SCARR and STRAVELAG. Fill instance attribute MT_CARRIERS with the data for all airlines and fill instance attribute MT_TRAVELAGS with the data for the travel agencies. Make sure you do not buffer the data for all the travel agencies, but instead only for those you require during the loop pass.

1. Copy the filling of MT_CARRIERS from method ADD_DATA to method PREPARE.
   a) See the source code excerpt from the model solution.
2. Do you still need the query that determines whether this call is the first call?

   No, because the PREPARE method is only called before the loop over the bookings.

3. Program a SELECT statement (array fetch) in which you fill instance attribute MT_TRAVELAGS. Use the contents of the IT_BOOKINGS import parameter to ensure that only the travel agencies that appear in the bookings in IT_BOOKINGS are read.
   a) See the source code excerpt from the model solution.
4. Why is it essential to check whether IT_BOOKINGS is filled? What would happen if IT_BOOKINGS were blank?

   The FOR ALL ENTRIES addition is only a restriction if the specified internal table is filled. Therefore, if IT_BOOKINGS is blank, all records from STRAVELAG are buffered, even though no data is required later, because ADD_DATA is not executed.

Task 3

Edit the implementation of method ADD_DATA.

Remove all the database accesses and replace them with accesses to the instance attributes with the buffered data.

© Copyright . All rights reserved.

Prohibida su reproduccion

1. Remove the SELECT statement for database table SCARR.
   a) See the source code excerpt from the model solution.
2. Remove the SELECT statement for database buffer STRAVELAG as well as the filling of the buffer.
   a) See the source code excerpt from the model solution.
3. Activate and test your program.
   a) Perform this step in the standard manner.

Interface lif_badi

```
INTERFACE lif_badi.
  METHODS:
      prepare
          IMPORTING it_bookings TYPE bc402_t_bookings,
      add_data
           CHANGING cs_booking TYPE bc402_s_booking.
ENDINTERFACE.                    "lif_badi
```

Class lcl_im_badi (Definition)

```
CLASS lcl_im_badi DEFINITION.
  PUBLIC SECTION.
    INTERFACES lif_badi.
  PRIVATE SECTION.
    TYPES: BEGIN OF ty_s_scarr,
             carrid   TYPE scarr-carrid,
             carrname TYPE scarr-carrname,
           END OF ty_s_scarr.

    TYPES: ty_t_scarr TYPE HASHED TABLE OF ty_s_scarr
                      WITH UNIQUE KEY carrid.

    TYPES: BEGIN OF ty_s_stravelag,
             agencynum  TYPE stravelag-agencynum,
             name       TYPE stravelag-name,
             city       TYPE stravelag-city,
           END OF ty_s_stravelag.

    TYPES ty_t_stravelag TYPE HASHED TABLE OF ty_s_stravelag
                         WITH UNIQUE KEY agencynum.

    DATA: mt_carriers  TYPE ty_t_scarr,
          mt_travelags TYPE ty_t_stravelag.

ENDCLASS.                    "lcl_im_badi DEFINITION
```

Class lcl_im_badi (Implementation)

```
CLASS lcl_im_badi IMPLEMENTATION.
  METHOD lif_badi~prepare.
* carrname - buffer full
```


© Copyright . All rights reserved.

Prohibida su reproduccion

```
    SELECT carrid carrname FROM scarr
           INTO TABLE mt_carriers.

* agencyname agencycity - buffer for all entries

    IF it_bookings IS NOT INITIAL.
      SELECT agencynum name city
             FROM stravelag
             INTO TABLE mt_travelags
             FOR ALL ENTRIES IN it_bookings
             WHERE agencynum = it_bookings-agencynum.
    ENDIF.
  ENDMETHOD.                    "lif_badi~prepare


  METHOD lif_badi~add_data.

    DATA:
        ls_carrier  LIKE LINE OF mt_carriers,
        ls_travelag LIKE LINE OF mt_travelags.

* carrname - read from buffer
*   IF mt_carriers IS INITIAL.
*     SELECT carrid carrname FROM scarr
*            INTO TABLE mt_carriers.
*   ENDIF.
    READ TABLE mt_carriers INTO ls_carrier
         WITH TABLE KEY carrid = cs_booking-carrid.

    cs_booking-carrname = ls_carrier-carrname.

* agencyname and agencycity - read from buffer
    IF cs_booking-agencynum IS NOT INITIAL.

      READ TABLE mt_travelags INTO ls_travelag
           WITH TABLE KEY agencynum = cs_booking-agencynum.
*     IF sy-subrc <> 0.
*       SELECT SINGLE agencynum name city
*              FROM stravelag
*              INTO ls_travelag
*              WHERE agencynum = cs_booking-agencynum.
*
*       INSERT ls_travelag INTO TABLE mt_travelags.
*     ENDIF.
      cs_booking-agencyname = ls_travelag-name.
      cs_booking-agencycity = ls_travelag-city.

    ENDIF.


  ENDMETHOD.                    "lif_badi~add_data

ENDCLASS.                    "lcl_im_badi IMPLEMENTATION
```

© Copyright . All rights reserved.

Prohibida su reproduccion

LESSON SUMMARY

You should now be able to:

- Identify disadvantages of nested SELECT
- Use ABAP joins and database views
- Explain special techniques for reading from multiple database tables
- Use explicit buffering techniques

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment

1. The _____ statement defines and executes the access, but does not pass any data on to the application program.

   Choose the correct answer.

   - ☐ A FETCH
   - ☐ B OPEN CURSOR
   - ☐ C FETCH NEXT CURSOR
   - ☐ D CLOSE CURSOR

2. The _____ operator is used in the where clause of a Select statement when you want to compare the data with a single list of values..

   Choose the correct answer.

   - ☐ A LIKE
   - ☐ B IN
   - ☐ C BETWEEN
   - ☐ D IS

3. When you process multiline result sets, you should read them into an internal table (array fetch).

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

© Copyright . All rights reserved.

Prohibida su reproduccion

4. An aggregate expression uses an aggregate function to specify a column in the SELECT DISTINCT statement.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

5. Two prerequisites of using the ORDER BY PRIMARY KEY addition to a select statement are that the FROM clause must contain a single database table (no views or joins) and that the field list after SELECT must contain all key fields of the table.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

6. _____ corresponds to the result set that only considers the records from the outer table for which suitable data records exist in the inner table.

   Choose the correct answer.

   - ☐ A LEFT OUTER JOIN
   - ☐ B INNER JOIN
   - ☐ C OUTER JOIN

7. A SELECT statement with a subquery has a more restricted syntax than a SELECT statement without a subquery.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

© Copyright . All rights reserved.

Prohibida su reproduccion

8. Which of the following technique in a Select statement should be used when you want to do a join between an internal table and a database table.

   Choose the correct answer.

   - ☐ A Select with for all entries
   - ☐ B Select with Inner Join
   - ☐ C Select with Outer Join
   - ☐ D Select with Read table

9. To read large data volumes, you should use _____ only in exceptional cases.

   Choose the correct answer.

   - ☐ A SELECT
   - ☐ B SELECT SINGLE
   - ☐ C FOR ALL ENTRIES

© Copyright . All rights reserved.

Prohibida su reproduccion

Learning Assessment - Answers

1. The _____ statement defines and executes the access, but does not pass any data on to the application program.

   Choose the correct answer.

   - [ ] A FETCH
   - [x] B OPEN CURSOR
   - [ ] C FETCH NEXT CURSOR
   - [ ] D CLOSE CURSOR

2. The _____ operator is used in the where clause of a Select statement when you want to compare the data with a single list of values..

   Choose the correct answer.

   - [ ] A LIKE
   - [x] B IN
   - [ ] C BETWEEN
   - [ ] D IS

3. When you process multiline result sets, you should read them into an internal table (array fetch).

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

4. An aggregate expression uses an aggregate function to specify a column in the SELECT DISTINCT statement.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

5. Two prerequisites of using the ORDER BY PRIMARY KEY addition to a select statement are that the FROM clause must contain a single database table (no views or joins) and that the field list after SELECT must contain all key fields of the table.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

6. _____ corresponds to the result set that only considers the records from the outer table for which suitable data records exist in the inner table.

   Choose the correct answer.

   - [ ] A LEFT OUTER JOIN
   - [x] B INNER JOIN
   - [ ] C OUTER JOIN

7. A SELECT statement with a subquery has a more restricted syntax than a SELECT statement without a subquery.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

© Copyright . All rights reserved.

Prohibida su reproduccion

8. Which of the following technique in a Select statement should be used when you want to do a join between an internal table and a database table.

   Choose the correct answer.

   - [x] A Select with for all entries
   - [ ] B Select with Inner Join
   - [ ] C Select with Outer Join
   - [ ] D Select with Read table

9. To read large data volumes, you should use _____ only in exceptional cases.

   Choose the correct answer.

   - [ ] A SELECT
   - [ ] B SELECT SINGLE
   - [x] C FOR ALL ENTRIES

© Copyright . All rights reserved.

Prohibida su reproduccion