UNIT II

SAP: History – SAP R/2 – SAP R/3 – Characteristics of SAP R/3 – Architecture of SAP R/3 - SAP Modules, Net Weaver, Customer Relationship Management, Business Warehouse, Advanced Planner and Optimiser. **ABAP/4: Workbench** - Workbench Tools - ABAP/4 Data Dictionary - ABAP/4 Repository Information – Structure of ABAP/4 program - ABAP/4 syntax – Data types – Constants and Variables – Statements : DATA, PARAMETERS, TABLE, MOVE, MOVE-CORRESPONDING, CLEAR, WRITE, CHECK, FORMAT. LOOP STRUCTURES. Sample programs.

TWO MARK QUESTIONS:

1. **Define SAP.**

SAP (Systems, Applications & Products in Data Processing) is a European multinational software corporation that makes enterprise software to manage business operations and customer relations

2. List out the Various SAP Modules:

- Financial Accounting (FI)
- Financial Supply Chain Management (FSCM)
- Controlling (CO)
- Materials Management (MM)
- Sales and Distribution (SD)
- Logistics Execution (LE)
- Production Planning (PP)
- Quality Management (QM)
- Plant Maintenance (PM)
- Project System (PS)
- Human Resources (HR)

3. What is CRM?

Customer relationship management (CRM) is an approach to managing a company's interaction with current and future customers. It often involves using technology to organize, automate, and synchronize sales, marketing, customer service, and technical support.

4. Define Business Warehouse

Business Warehouse (BW) is a combination of databases and database management tools that are used to support management decision making.

5. Define SAP Netweaver

NetWeaver is a combination of the underlying SAP Kernel (also known as the SAP OS layer, basically the WEB AS) and any SAP software tool for business enablement.

6. Define Planning & Optimiser

APO (Advanced Planning and Optimization) application is at the heart of SCM. It offers planning and optimization functionalities in different business processes of Demand Planning, Supply Planning, Supply and Demand Matching, Production Planning Detailed Scheduling, and Global Available to Promise and Transportation Management.

7. What is ABAP?

ABAP – the programming language used in SAP for developing business application support and development. ABAP is a programming language that runs in the SAP ABAP runtime environment, created and used by SAP for the development of application programs

8. **Define ABAP Data Dictionary?**

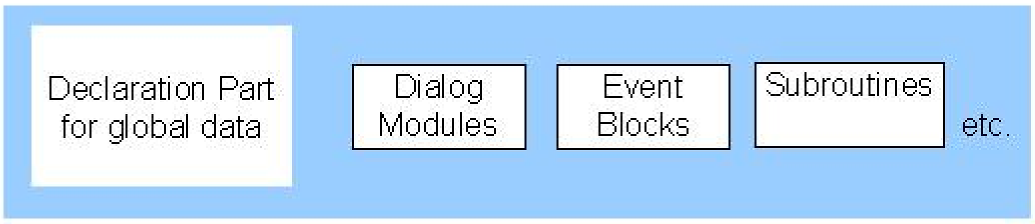
ABAP Dictionary is one of the important tools of ABAP Workbench. It is used to create and manage data definitions without redundancies. ABAP Dictionary always provides the updated information of an object to all the system components. The presence and role of ABAP Dictionary ensures that data stored in an SAP system is consistent and secure.

9. Define Repository Information System

ABAP/4 Repository Information System (in short: Repository Information System) given here is restricted to aspects of relevance for work with the Data Modeler. Using the ABAP/4 Repository Information System, you can search for all modeling objects of the Data Modeler that you wish to display or edit.

10. **Define the Structure of ABAP/4 Program**

The following diagram shows the structure of an ABAP program:



ELEVEN MARK QUESTIONS:

1. Explain in detail about History of SAP.

SAP-History

SAP (Systems, Applications & Products in Data Processing) is a European multinational software corporation that makes enterprise software to manage business operations and customer relations. SAP is headquartered in Walldorf, Baden-Württemberg, Germany, with regional offices in 130 countries. The company has over 293,500 customers in 190 countries. The company is a component of the Euro Stoxx 50 stock market index. SAP is the world leader in enterprise applications in terms of software and software-related service revenue. Based on market capitalization, it is the world's third largest independent software manufacturer supporting all sizes of industries helping them to operate profitability, grow sustainably and stay ahead of the competition in the market.

Company Perspectives: Whether opening up new markets or servicing existing ones, SAP is committed to its tested strategy: concentrating on its core business developing and selling standard enterprise applications software and partnering with hardware vendors, software suppliers, technology providers, value-added resellers, and consulting firms.

Company History:

SAP AG is the fifth largest independent software producer worldwide and the largest producer of standard enterprise-wide business applications for the roughly $9 billion global client-server software market. The company's principal business activities are the development and marketing of an integrated line of prepackaged computer software for over 1,000 predefined business processes, from financial accounting, human resources, and plant maintenance to quality assurance, materials management, sales and distribution, and business workflow. Its two major products, the R/2 and R/3 suite of business software applications, are used by over 4,000 companies in the oil and gas, banking, insurance, utilities, telecommunications, pharmaceuticals, consumer products, automotive, retail, health care, chemicals, and high tech and electronics industries. Through its 28 international subsidiaries, led by its U.S. subsidiary SAP America, SAP AG markets its software and consulting, training, and support services in more than 50 countries.

1970s Founding

SAP AG was founded in 1972 by five German engineers with IBM in Mannheim, Germany; interestingly four of the founders HassoPlattner, DietmarHopp, Klaus Tschira, and Hans Werner Hector were still with SAP in early 1996. When an

IBM client asked IBM to provide enterprise-wide software to run on its mainframe, the five began writing the program only to be told the assignment was being transferred to another unit. Rather than abandon the project altogether, they left IBM and founded SAP in Walldorf, near Heidelberg. SAP eventually came to stand for Systeme, Anwendungen, und Produkte in Datenverarbeitung (systems, applications, and products in data processing).

Without the benefit of loans from banks, venture capitalists, or the German government, SAP began fashioning its software business gradually through the cash flow generated by an ever-growing stable of customers. Working at night on borrowed computers to land their first contracts, Plattner and colleagues built SAP's client list with German firms in its region, beginning with a German subsidiary of the global chemical company ICI and later adding such major German multinationals as Siemens and BMW.

R/2 in the Late 1970s

In 1978 SAP began developing, and the following year released, R/2 (R for "real-time"), a mainframe-based, standard business software suite in which integratable modules for accounting, sales and distribution, and production enabled customers to consolidate their financial and operational data into a single database and eliminate costly paperwork and data entry. Because the modules were self-standing, businesses could select only those they needed, which could then be further customized to their unique requirements. The promise of real-time integration of mission-critical corporate data, viewable through the spreadsheet-like windows of specialized software, offered the potential for uniform data flow, streamlined business operations, and centralized decision-making.

Relying on word of mouth filtering through the overseas branches of its German customers, SAP soon began selling its software outside Europe. With corporate giants like Dow Chemical and Bayer already running R/2, SAP could rely on the fear of obsolescence of its customers' rivals to sell its software to the major competitors in each industry. Among the large corporations who began to adopt R/2 were Dupont, General Mills, Goodyear Tire and Rubber, Heinz, and Shell Oil, as well as 80 of the 100 largest companies in Germany, such as Hoechst, Daimler Benz, and BASF. By 1991 R/2 had gone through four releases or versions and had established itself as the standard for integrated corporate business software in Europe. By 1994, SAP could claim more than 1,400 R/2 installations worldwide.

R/3 Development in the 1980s

As R/2's potential began to peak in the mid-1980s, Plattner and company's former employer, IBM, announced a new "system applications architecture" (SAA) technology in which all IBM operating systems and platforms would be fully harmonized such that code written for one product would work with any other. Seeing the ramifications of such integratability for its own products, in 1987 SAP began developing R/3 for use in the decentralized, non-mainframe computing environment known as client-server. In client-server arrangements, data is processed not by a single costly mainframe but by many cheaper networked "server" computers, which display their data on flexibly arrangeable PCs called "clients." While R/2 focused on providing data processing solutions for static, individual functions of business operations, such as inventory tracking or shipping, R/3 was designed to allow a business to view its entire business operation as a single integrated process in which data entered into any single application in the system would simultaneously be registered in every other. In theory, a company's entire data network would now be a cohesive, interpretable whole that would enable management to more efficiently allocate resources, develop products, manage inventory, forecast trends, streamline manufacturing processes, and automate routine operations.

R/3 itself consisted of IBM's OS/2 operating system as its "front end" or user interface, IBM's DB2 program as its database component, and SAP's own proprietary application component, which was based on AT&T's Unix operating system because it offered the greatest functionality with other vendors' systems. Thus was created the three-tiered architecture--interface or desktop + database + application--on which all later versions of R/3 would be based.

Introduction of R/3 in the 1990s

After five years in development, R/3 had been launched with the expectation that it would complement R/2's multinational-oriented niche by extending SAP's reach into the mid-sized, less mainframe-dominated business software market. Unexpectedly, however, R/3's release coincided with a growing trend toward corporate downsizing, and even SAP's largest customers began eyeing R/3 as a less labor-intensive replacement for R/2. As a result, in the space of one year (1992-93), the percentage of SAP America's total revenue generated by R/3 catapulted from five to 80 percent, and R/2's status as SAP's flagship product dwindled from 95 percent of revenues to only 20 percent. R/3 was suddenly hot, and virtually overnight SAP had translated its reputation as Germany's wunderfirma to the global stage.

On the strength of R/3's rocketing sales, by the mid-1990s SAP had traveled from the relative anonymity of 1992 to the business applications vendor of choice for nine of the ten largest U.S. corporations, one-third of the Fortune 500, seven of the ten largest Business Week Global 1000, and 80 percent of the Fortune 100 companies in software, computers, peripherals, and semiconductors. Total sales revenues had nearly tripled between 1991 and 1995 to DM 2.7 billion and had increased 66 percent between 1993 and 1994 alone. Such major corporations as Apple, Chevron, Colgate-Palmolive, Digital Equipment, and Polaroid were jumping on the R/3 bandwagon, and by September 1995 SAP could claim over 1,100 installations of R/3 for companies with $1 billion or more in sales (in addition to 700 R/2 installations) and more than 1,300 installations in smaller companies (with 800 R/2 installations). SAP's share price had in the meantime grown 1,000 percent since its introduction on the German stock exchange in 1988, and by 1996 it ranked as the highest valued company in Germany.

Competition and Other Challenges

SAP's two major competitors, Oracle Systems (United States) and Baan (the Netherlands), were meanwhile making inroads into SAP's market share. Although Oracle's database product was the program most often used as R/3's database component--making SAP the largest value-added reseller of Oracle products--in 1995 Oracle announced it would overtake SAP as the world's leading provider of industry-specific software within three years. Baan, though dwarfed by SAP in sales and customers, scored major coups in the mid-1990s when both Boeing and German giant Siemens Nixdorf rejected R/3 in favor of Baan's quick-installing business software package. SAP board member Henning Kagermann dismissed the setbacks, telling the Deutsche Presse Agentur, "If SAP wins a large order, it's accepted as natural. When we lose a potential customer, immediately it's a big headline." SAP management also dispelled the severity of the threat posed by Oracle, pointing out that it in head-to-head competition SAP still won the contract 80 percent of the time.

Two public relations disasters in the mid-1990s suggested not only the extent of the controversy that had begun to surround R/3 but also SAP's saavy in handling criticism. In March 1995 the German business magazine Wirtschaftswoche published an article accusing SAP of accepting commissions from hardware vendors for computers sold to SAP customers and quoting several users' disparaging remarks about the expense and installation time required by R/3. As share prices nose-dived, SAP lashed back. Its hardware partners unanimously denied any kickback arrangement with SAP, and SAP itself took out a court order on the magazine for inaccuracy and deliberate misquotation and ran four-page ads in major German print outlets in which the article's sources claimed they were misquoted and expressed satisfaction with R/3. Then, in early 1996, U.S. computer industry analyst Forrester Research published a study in which it argued that SAP's R/3 was based on an obsolete architecture that could not keep pace with the open, nonproprietary architecture increasingly favored by the software industry. SAP, Forrester claimed, knew that R/3 would be obsolete by 1997 and secretly planned to foist a brand new "object-based" system called R/10 on its customers in 1999, masking its deployment through a series of add-ons to R/3. All SAP customers, Forrester advised, should minimize their dependency on R/3 and prepare "exit strategies" to avoid being trapped into an expensive installation of a new SAP product.

SAP reacted by prematurely releasing quarterly financial figures showing that R/3 sales had not in fact peaked and by vehemently denying that it was planning to abandon R/3. It further vowed to spend DM3 billion on R&D over the next five years and announced plans for new versions of R/3 that reflected its willingness to make the product, which was based on its own proprietary programming language, more open to integration with other vendors' products. SAP, moreover, signaled it was embracing the Internet-driven trend toward "object-oriented" software in which applications could be embedded with other vendors' mini-programs (called "objects" or "applets"). The strategy worked, and Forrester Research was soon announcing that SAP was "leading in the new Internet game."

The Future

In the mid-1990s industry observers agreed that SAP's continued dominance of the client-server business software market rested on its ability to stay ahead of the breathtaking pace of change in the global software market. In the mid-1990s, for example, SAP was directly affected by the rise of the "intranet," a microcosmic version of the Internet created by companies as in-house data networks, mirroring the structure and appearance of the World Wide Web but protected from the cyber surfing public by so-called firewalls. By seeming to offer the potential to perform many of the same business applications and data processing features of R/3, such intranets represented a plausible threat to SAP's market leadership. SAP responded by announcing new features that would turn R/3 into an Internet-capable tool. Using a browser connected to the Web, for example, two companies with R/3 installed in their systems could process orders in real time over the Internet, while consumers could order products electronically from a company's online catalog and be confident the order was registered immediately in the company's R/3 system.

SAP's ability to sustain its success also depended on its willingness to continue working, Ã la Microsoft, with its hundreds of strategic partner firms throughout the computer and services industries. SAP's Platform Partners program, for example, had enabled it to cooperate with computer manufacturers such as Compaq and IBM in tailoring SAP products to new hardware developments. And its partnership program with such Big Six accounting firms as Arthur Andersen and Price Waterhouse had spawned a lucrative new subindustry of R/3 consultants whose institutional independence from SAP enabled it to focus more of its resources on improving its product. Finally, SAP's participation with other software vendors in industry-wide initiatives (such as the Open Application Group) to determine standards for new technologies demonstrated its willingness to cooperate with potential competitors to ensure the continued functionality and influence of its products.

Significantly, in 1994 SAP formed an alliance with America's software giant Microsoft to make SAP software integratable with such Microsoft products as Windows NT, an operating system for networked computers, and SQL Server, a database product. In 1995, Microsoft returned the favor by selecting R/3 for its global finance and accounting data system. In early 1996, Microsoft founder and chairman Bill Gates paid a symbolic visit to SAP AG's German headquarters to talk up the two megacompanies' budding relationship. "We love SAP," he said. "SAP has had more impact on our general product direction than any other software company we have worked with.... [Microsoft and SAP] are the two best companies to be in."

By learning how to quash media and public relations flare-ups and better market its products, by continuing to modify R/3 to capitalize on new technologies like the Internet, and by encouraging third-party vendors to develop specialized add-on applications to extend the number of business areas in which R/3 could be used, SAP appeared to have positioned itself to remain a formidable presence in the global business software market.

SAP R/2: is 2-tier architecture. In which all 3 layers [Presentation **+** Application **+**Database] are installed in two separate systems/server. (Server One – Presentation, Server Two – Application +Database)

SAP R/3: is 3-tier architecture. In which all 3 layers [**Presentation** + **Application** +**Database**] are installed in three separate systems/server. (Server One – Presentation, Server Two – Application, Server Three-Database)

SAP R/3 : Three-Tier Architecture

With SAP R/3, SAP users in a new generation of enterprise software — from mainframe computing (client-server architecture) to the three-tier architecture of database, application, and user interface.

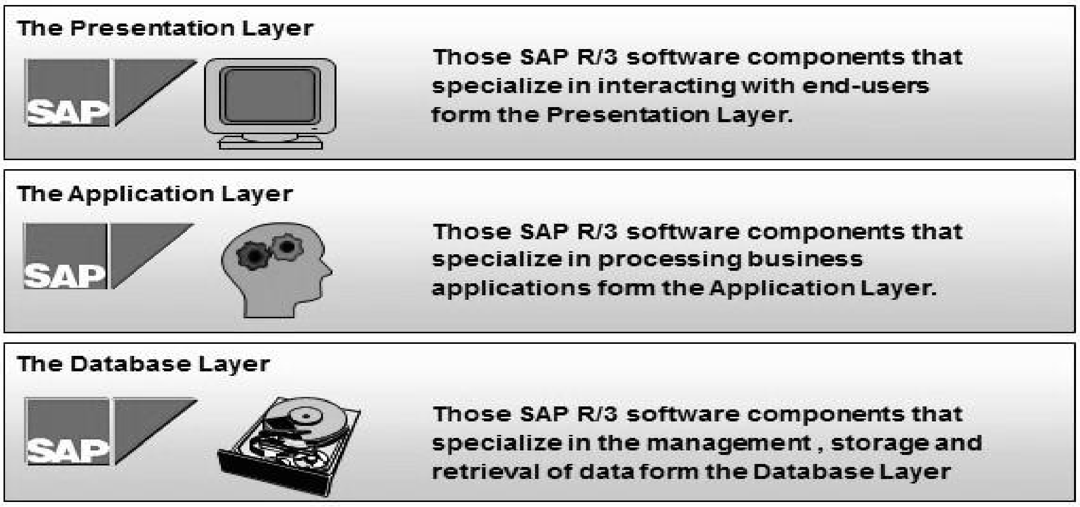
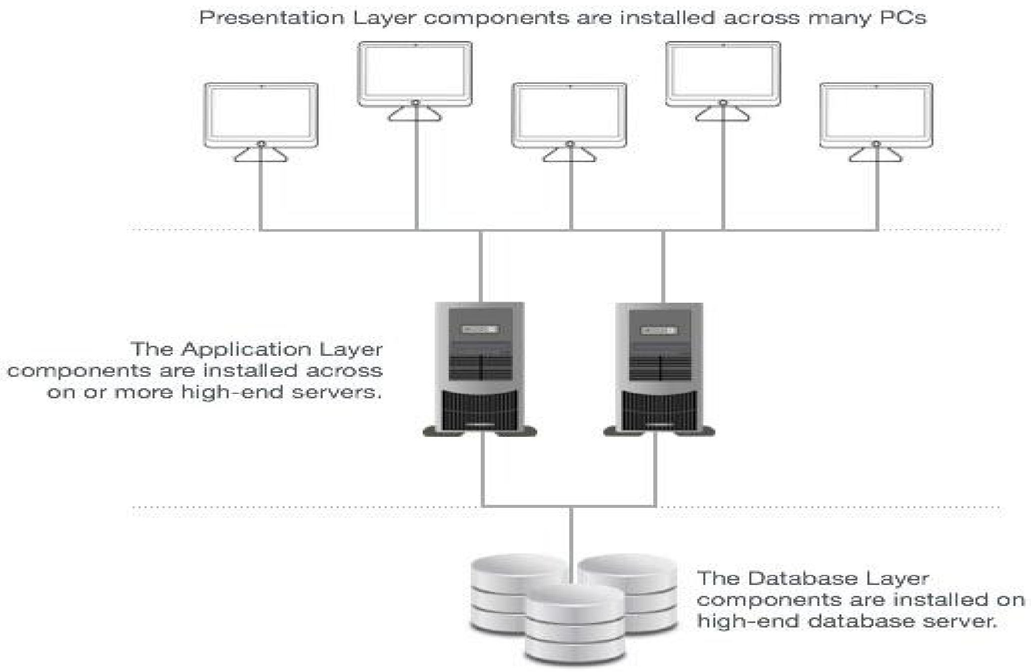

The Presentation Layer
SAP
Those SAP R/3 software components that specialize in interacting with end-users form the Presentation Layer.
The Application Layer
SAP
Those SAP R/3 software components that specialize in processing business applications form the Application Layer.
The Database Layer
SAP
Those SAP R/3 software components that specialize in the management , storage and retrieval of data form the Database Layer

Fig: Three-Tier Architecture of SAP R/3

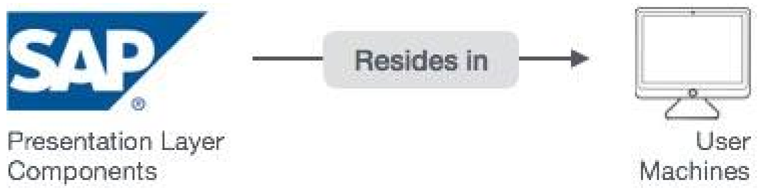

Presentation Layer components are installed across many PCs
The Application Layer components are installed across on or more high-end servers.
The Database Layer components are installed on high-end database server.

Three-Tier Architecture

Presentation Layer Servers

Presentation servers contain systems capable of providing a graphical interface.

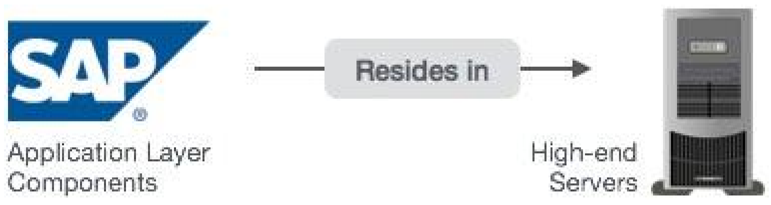
- Presentation Layer is also known as client Layer
- Presentation Layer is a user interaction
- In SAP-User interaction purpose we use GUI
- GUI stands for Graphical user interface
- Example − Desktop, Mobile Devices, laptops



Application Layer Servers

Application servers include specialized systems with multiple CPUs and a vast amount of RAM.

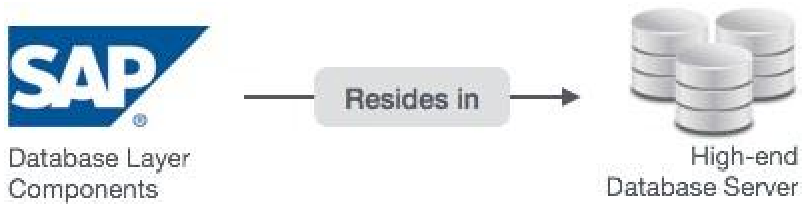
- Application Layer is also known as Kernel Layer and Basic Layer.
- SAP application programs are executed in Application Layer.
- Application Layer serves as a purpose of a communicator between Presentation and Database Layer.
- Application server is where the dispatcher distributes the work load to the different work processes makes the job done.



Database Layer Servers

Database servers contain specialized systems with fast and large hard-drives.

- Database layer stores the data
- Data store can be Business data, SAP system data, SAP tables, Programs.
- Examples − Oracle, Microsoft SQL Server, IBM DB/2, Siebel, Sybase, etc.



2. Explain in detail about SAP R/3 – CHARACTERISTICS

.SAP R/3 – CHARACTERISTICS

- SAP's R/3, introduced in 1992, is the most used ERP system in the world.
- The R/3 software package is designed to allow businesses to effectively and efficiently operate a variety of business processes within a single integrated information system.

- The software is customizable using SAP's proprietary programming language, ABAP/4. R/3 is scalable and highly suited for many types and sizes of organizations and runs on six different platforms.
- SAP's R/3 has been designed to be the best ERP system in the four areas of human resources, financial, supply chain management, and marketing. R/3 is also an international product, and meets the local fiscal, language, and tax requirements of most countries.
- SAP's R/3 is very versatile, as it will operate on six different platforms, including the recently added Microsoft NT.
- The R/3 package includes several very attractive features like it has a three-tier client/server system. Providing three tiers offers scalability and easier adaptation to the specific needs of large companies and fast-growing companies.
- SAP's R/3 is available in 14 different languages (German, English, Spanish, etc.) and also incorporates multiple currency features that provide essential information processing capabilities for multinational corporations.
- R/3's modules are organized by the functional areas of financial, human resources, supply chain management, and marketing. While information is entered separately for each specific module, the modules are fully-integrated and provide real-time applications. This means that data entered into one module is immediately and automatically updated and reflected in all of the functional areas.
- R/3 is composed of a single, virtual file structure with no subsystems.
- In addition, SAP has released "MySAP.com" which is software that provides for data interaction and processing connections with the Web.
- Financial and managerial accounting tools in SAP R/3 are contained in the financial accounting (FI) and the controlling (CO) modules. The General Ledger function in the FI module provides a comprehensive record of all information needed for external financial reporting. The accounting data is complete and accurate because the SAP system fully integrates all business transactions that were entered from all the operational areas of a company. In addition to the FI and CO modules, the SAP system includes the Investment Management (IM), Sales and Distribution (SD), Materials Management (MM), and Human Resources (HR) modules.

- Management accounting tools in SAP R/3 are cost center accounting, internal orders, product costing, and activity based costing, profitability analysis and profit center accounting.
- SAP R/3's accounting features are modeled on German approaches to accounting, and thus they are well-organized and very efficient in processing accounting information and providing accounting statements and financial reports.
- As stated previously, R/3 offers multiple currency features and a three-tier system that is capable of meeting very high demands from the accounting system for either transaction processing or financial reporting.
- SAP was the first to implement integrated treasury capabilities. This attractive feature allows a corporate treasury department to function as an in-house bank by automating the control of cash flow, investment trades, and portfolio management.
- R/3 provides check writing capability in its Accounts Receivable component which very few other programs offer.
- Additionally, there is equal access to all data in the system. This means that personnel can access financial data directly from a computer screen rather than physically meet with the treasurer, controller, or some other similar person. In other words, R/3 offers real-time, immediately updated reporting.
- R/3 also provides for a "single data entry point" where the data entered from any location is instantly sent to all other appropriate modules in the ERP system.
- The accounts payable component of SAP R/3 contains four types of transaction blocks namely:
    - ✓ The audit block
    - ✓ The receiving block
    - ✓ The vendor block
    - ✓ A manual block
- These blocks make it much less likely that improper payments will occur.
- SAP R/3 is organized with the concept that a business operates as a series of processes, which means that the company implementing R/3 may have to change and reorganize itself to properly fit with R/3 and use it effectively. Thus SAP R/3 on the whole as stated above gives:
- Significant cost and time savings.

- Minimum operating costs: no retention of redundant data in the back office.
- High level of stability and performance: response times are consistently under one second.
- Good user interface available which makes system user friendly and requires no training for the end user.

3. Explain in detail about SAP R/3 – Architecture

SAP R/3 ARCHITECTURE

Presentation layer/tier/server:

It is the interface to a user. This is the only layer from where users connect to the SAP system. DIAG (Dynamic Information Action Gateway) is the protocol which is used to communicate b/w user and SAP system. Using this we can have, our own font settings, and our own languages settings. It is user friendly. With the help of message server which identifies favorite server and logs onto it. It is intelligent server. It is operating system &db independent.

Presentation layer is nothing but SAP GUI: SAP GUI is to facilitate users to log into R/3 system. This logon can be used to all the components of SAP (CRM, APO, BW, XI etc.)

Types of SAP GUI:

- SAP GUI for Windows.
- SAP GUI for HTML.
- SAP GUI for JAVA.

SAP GUI for Windows: It is for the windows environment. Support platforms Includes windows 98, windows NT4, Windows 2000 and Windows XP.

SAP GUI for HTML: Front end requires only a web browser, and it is necessary to convert the presentation into HTML.

SAP GUI for JAVA: It is used only where java is supporting. It supports Windows 98, windows NT4, Windows 2000 and Windows XP, MacOS 9, MacOS x Linux, HP UX, Solaris, AIXOS/2.

Application layer/tier/server:

It is used to Provides business areas and Configure work process and Reduce traffic on DB. It is used Configure memory areas and Business logic & presentation logic handled.It consists of dispatcher, work processes, memory areas, buffer areas and interpreters.

Dispatcher: There will be only one dispatcher per instance. This is used to handle the user requests. Dispatcher receives the users request and keeps them in the queue (dispatcher queue) based on the available free resources, user request will be assigned with work process on FIFO basis. Dispatcher runs by an executable disp+work.exe. This can be monitor by using a command line tool DPMON (It listens on the port 32<sysnr>). Dispatcher assigns the user request to a dialog work process, so it will distribute request to respected work process.

Dialog process: It is used for handling generation of reports, updating the temporary tables, updating the spool tables, updating the background tables so that update, spool background processes reads those tables for execution. If the request is long running job then it will assigns to its relevant work process. Dialog work process run time is restricted to 600 sec to 1800 sec based on the parameter (rdisp/max_wprun_time).

Update work process: This process is used to update the database initially update requests are handled by dialog work process as they couldn't execute within the specified time, it is called asynchronous update process. If the task has been moved to update work process then first dialog process updates the temporary tables (VBHDR, VBDATA, VBERR, VBMOD) update process reads the temporary tables and update the database.

Enqueue process: Enqueue process is used to lock and unlock SAP objects. It will update the database and takes the users request. In order to handle this mechanism SAP has defined enqueue and dequeue (unlock) modules. Enqueue process will issue locks to message server to all the dialog instances. That is dialog communicates with message server & message server in term talks to enqueue to get the lock. Dialog process communicates with the message server and message server communicates to enqueue. Dialog processes on central instance can communicate with enqueue directly to obtain locks.

Background Process: The long running, time consuming and expensive reports or updates will be used to schedule in the non-dialog mode using the background process. Dialog work process receives the background request & updates background request & updates background job tables. Background work process reads the job tables for every 60 sec & executes them

Message Server: Message server is used to communicate with all the available dispatchers under the port number 3600+sys no. If logon load balance is configured, message server identifies the least loaded server in the logon group. It is run by an executable msg_server.exe. This is also used to communicate with enqueue to issue locks to the work process coming from dialog instance.

Gateway: There will be one gateway work process for each instance. Gateway is used to communicate with external system.

Spool Process: Spool process is used to output the documents to the printer, fax, email, pager and sms. Dialog process receives the spool request and updates spool tables or stores spool data at OS level. Spool process reads the spool tables or spool data and output to specific device.

Note: All the work process runs with executable disp+work.exe.

Memory Areas: In order to define a work process we should have enough resources at the rate of 75mb to 150mb for each work process. When the user request is assigned to a work process, work process requires certain amount of memory to execute the user request.

Eg: Roll memory, extended memory and heap memory Buffer.

Interpreters:

1. ABAP Interpreter: This is used to interpret the ABAP code embedded in the user request.
2. Screen Interpreter: This is used to interpret the screens.
3. SQL Interpreter: This is used to interpret SQL Statements in the ABAP program.

Dispatcher: It receives user request and assigns work process or keep user request in dispatcher queue.

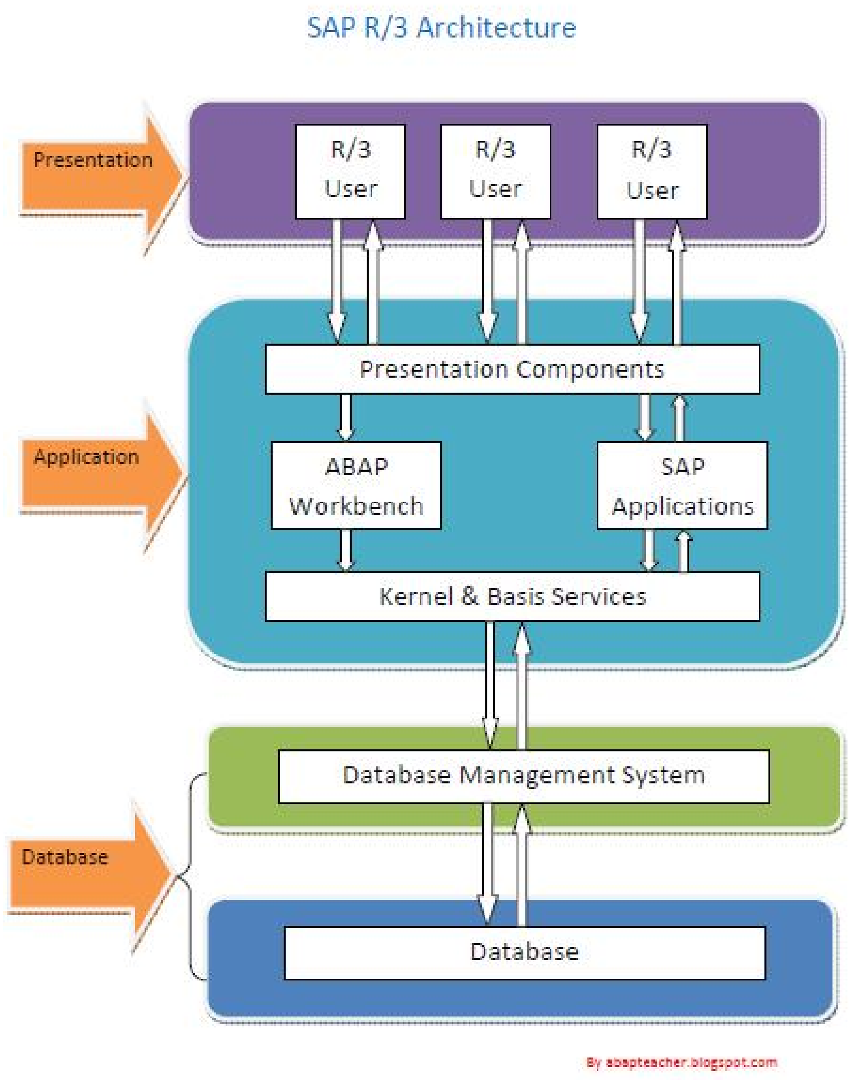
Task Handler: It is the agent which processes the user request by segregating into screen, ABAP, SQL interpreters.

User Context: The user context is the buffer area where it stores user logon attributes, authorization parameters.

Dispatcher Queue: It is the queue where user exists when work processor is busy. It follows FIFO.

Database Layer/tier:

It is the layer where database is hosted. It has its own memory areas, buffer areas, work processes etc. A central RDBMS realizes the database layer of SAP R/3 systems. Initially SAP database will use open SQL but database client will convert open SQL into native SQL. That is the reason SAP supports different databases.

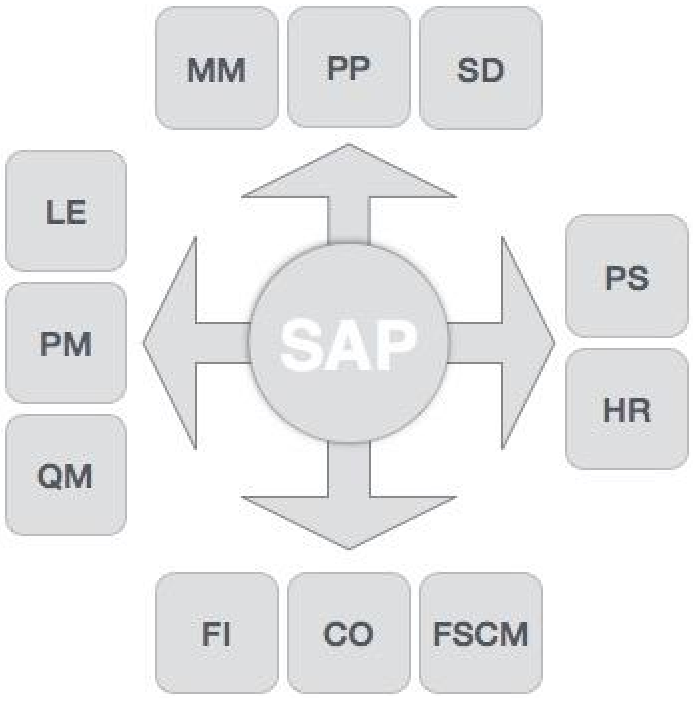


4. Explain in detail about SAP Modules.

SAP MODULES

SAP solutions include a number of functional modules, which support transactions to execute key business processes, such as

- Financial Accounting (FI)
- Financial Supply Chain Management (FSCM)
- Controlling (CO)
- Materials Management (MM)
- Sales and Distribution (SD)
- Logistics Execution (LE)
- Production Planning (PP)
- Quality Management (QM)
- Plant Maintenance (PM)
- Project System (PS)
- Human Resources (HR)



Finance and Controlling (FICO)

SAP FICO is a combination of two ERP modules, i.e., Finance Accounting (FI) and Controlling (CO). Under Finance in SAP and at an enterprise level, the following modules take part −

- FI − Finance
- CO − Controlling
- IM − Investment Management
- TR − Treasury
- EC − Enterprise Controlling

SAP FI (Financial Accounting) is accountable for tracking the flow of financial data across the organization in a controlled manner and integrating all the information for effective strategic decision-making.

Activities Involved in SAP FI

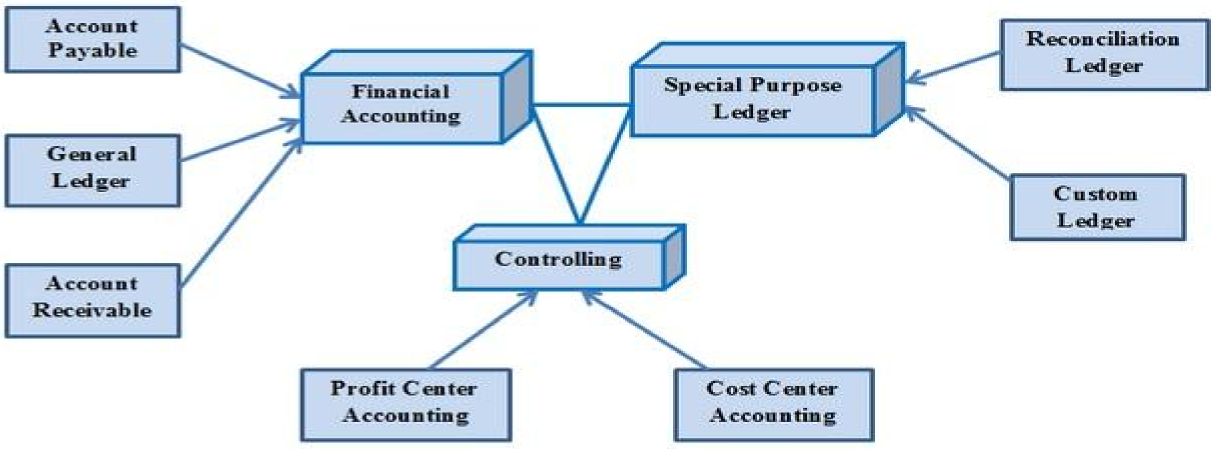
- Creation of Organizational Structure (Defining Company, Company Codes, business Areas, Functional Areas, Credit Control, Assignment of Company Codes to Credit Controls)
- Financial Accounting Global Settings (Maintenance of Fiscal Year, Posting Periods, defining Document types, posting keys, Number ranges for documents)
- General Ledger Accounting (Creation of Chart of Accounts, Account groups, defining data transfer rules, creation of General Ledger Account)
- Tax Configuration & Creation and Maintenance of House of Banks

- Account Payables (Creation of Vendor Master data and vendor-related finance attributes like account groups and payment terms)
- Account Receivables (Creation of Customer Master data and customer-related finance attributes like account groups and payment terms
- Asset Accounting
- Integration with SD and MM

SAP CO (Controlling) module facilitates coordinating, monitoring, and optimizing all the processes in an organization. It controls the business flow in an organization. This module helps in analyzing the actual figures with the planned data and in planning business strategies.

Activities Involved in SAP CO

- Cost Element Accounting (Overview of the costs and revenues that occur in an organization)
- Cost Center Accounting
- Activity-Based-Accounting (Analyzes cross-departmental business processes)
- Internal Orders
- Product Cost Controlling (Calculates the costs that occur during the manufacture of a product or provision of a service)
- Profitability Analysis (Analyzes the profit or loss of an organization by individual market segments)
- Profit Center Accounting (Evaluates the profit or loss of individual, independent areas within an organization)



Sales & Distribution Management (SD):

SAP SD is one of the most important modules in SAP. It has a high level of integration complexity. SAP SD is used by organizations to support sales and distribution activities of products and services, starting from enquiry to order and then ending with delivery. SAP SD can monitor a plethora of activities that take place in an organization such as products enquires, quotation (pre-sales activities), placing order, pricing, scheduling deliveries (sales activity), picking, packing, goods issue, shipment of products to customers, delivery of products and billings.

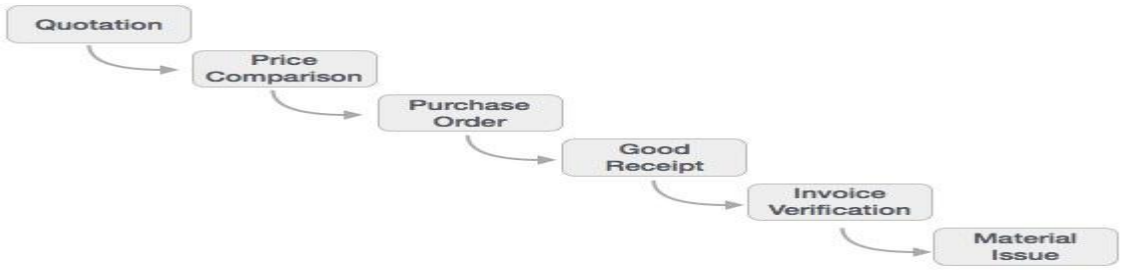
In all these processes, multiple modules are involved such as FI (Finance Accounting), CO (Controlling), MM (Material Management), PP (Production Planning), LE (Logistics Execution), etc., which shows the complexity of the integration involved.

Activities Involved in SAP SD

- Setting up Organization Structure (creation of new company, company codes, sales organization, distribution channels, divisions, business area, plants, sales area, maintaining sales offices, storage location)
- Assigning Organizational Units (Assignment of individual components created in the above activities with each other according to design like company code to company, sales organization to company code, distribution channel to sales organization, etc.)
- Defining Pricing Components (Defining condition tables, condition types, condition sequences)
- Setting up sales document types, billing types, and tax-related components
- Setting up Customer master data records and configuration

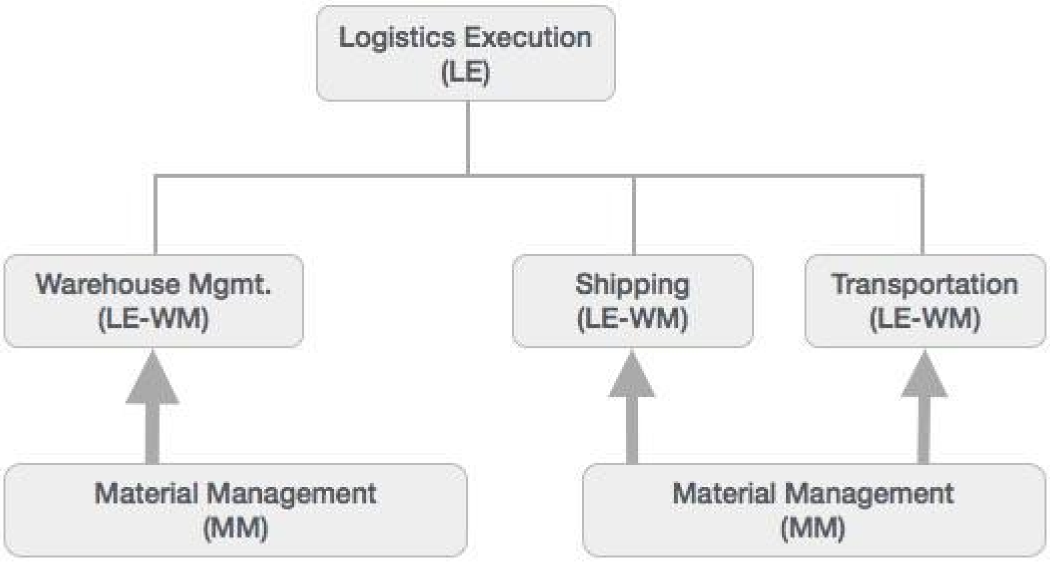
Material Management (MM)

Material Management deals with movement of materials via other modules like logistics, supply chain management, sales and delivery, warehouse management, production and planning.



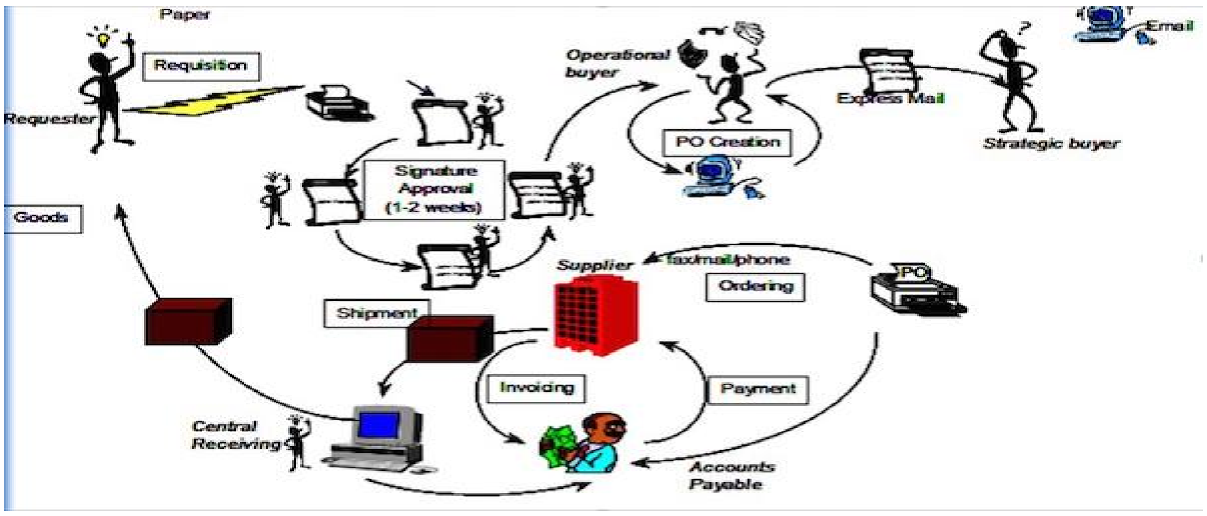
Logistic Execution (LE)

Logistic Execution can be divided into two sub-modules, i.e., shipment of goods (purchase to procurement process) and warehouse management (storage of goods). These two modules are integrated with sale and distribution, material management, and production and planning.



Supplier Relationship Management (SRM)

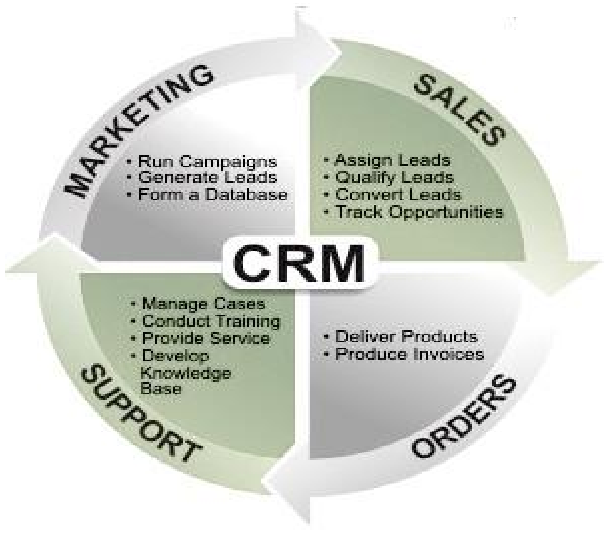
As the name SRM suggests, this module deals with the effective and efficient transition of products and services between an organization and its suppliers. The main process covered in this section is procurement of products like direct materials, indirect materials, and services. This module can effectively integrate with planning, accounting, and inventory system.



Customer Relationship Management (CRM)

CRM deals with end-to-end customer related processes. CRM is designed to centralize the data related to all the customers associated with an organization. It helps an organization –

- Maintain its sales, services, and build marketing strategies according the market demand and customer data analysis.
- Remain focused on its customers and via information analysis, help the business to know more about its customers.
- Improve sales and services and building better relationships with customers.

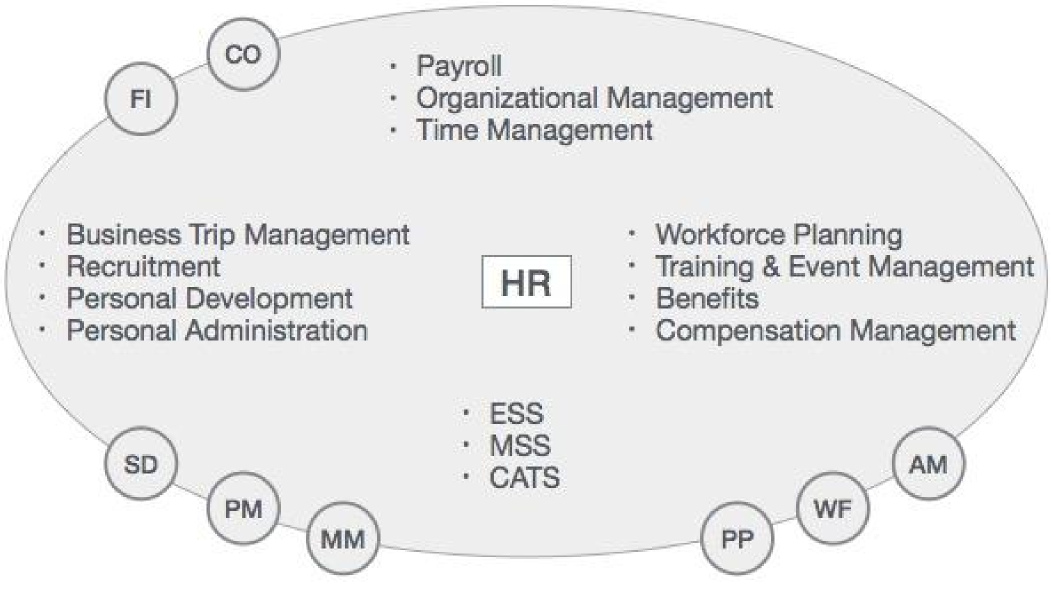


Human Resource (HR)

The most important objective of master data administration in Human Resources is to enter employee-related data for administrative, time-recording, and payroll purposes. A new employee can be hired without using Recruitment. Instead you can hire someone by running a personnel action in Personnel Administration, thereby creating the necessary data for the employee to be hired. Employee data must be kept current. After an employee is hired, circumstances can always arise which necessitate either the entry of new data or the correction of current data. For instance −

- An employee moves to his or her new address must be stored in the system.
- An employee gets a pay hike at the start of the year. The new salary must be stored for the relevant date.
- An employee changes jobs within the organization. His or her organizational assignment, working time, and salary also change.
- Data can be stored for the past, present, or future.

Note − Entering payroll-relevant data in the past triggers retroactive accounting.


CO
FI
• Payroll
• Organizational Management
• Time Management
• Business Trip Management
• Recruitment
• Personal Development
• Personal Administration
HR
• Workforce Planning
• Training & Event Management
• Benefits
• Compensation Management
• ESS
• MSS
• CATS
SD
PM
MM
PP
WF
AM

The HR module is comprised of major areas of functionality known as sub-modules. The HR module is a true demonstration of the strength of the SAP product in Enterprise Resource Planning.

The HR system has very strong integration points (where data is passed back and forth without human intervention) with just about all of the other SAP modules. In addition, there is very tight integration amongst the HR sub-modules.

5. Explain in detail about SAP NetWeaver.

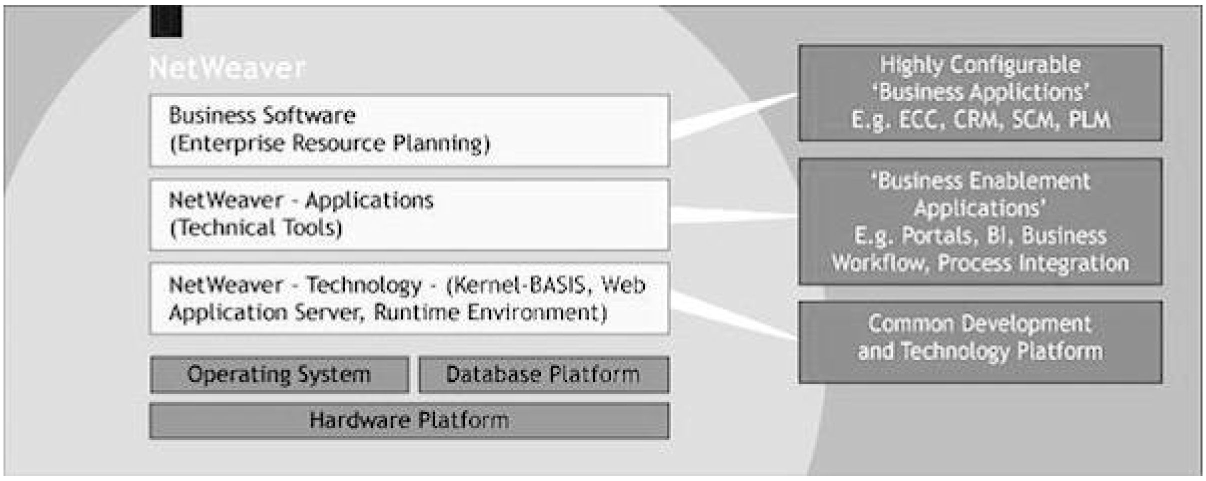
SAP - Net Weaver

NetWeaver is a combination of the underlying SAP Kernel (also known as the SAP OS layer, basically the WEB AS) and any SAP software tool for business enablement.

NetWeaver at a Glance

SAP NetWeaver describes all the software and services used for 'Business Enablement'. The SAP Business suite, such as ECC or SRM, contains the software components for that specific business solution. SAP NetWeaver is an open technology platform that offers a comprehensive set of technologies for running mission-critical business applications and integrating people, processes, and information.

SAP NetWeaver is a web-based, open integration, application platform that serves as the foundation for enterprise service-oriented architecture (enterprise SOA) and allows the integration and alignment of people, information, and business processes across business and technology boundaries. It utilizes open standards to enable integration with information and applications from almost any source or technology. SAP NetWeaver is the foundation of SAP Business Suite and SAP Business by Design. It also powers partner solutions and customer custom-built applications.


NetWeaver
Business Software (Enterprise Resource Planning)
NetWeaver - Applications (Technical Tools)
NetWeaver - Technology - (Kernel-BASIS, Web Application Server, Runtime Environment)
Operating System
Database Platform
Hardware Platform
Highly Configurable 'Business Applictions' E.g. ECC, CRM, SCM, PLM
'Business Enablement Applications' E.g. Portals, BI, Business Workflow, Process Integration
Common Development and Technology Platform

SAP NetWeaver Components

SAP NetWeaver includes a comprehensive set of components, applications, and tools.

SAP NetWeaver Application Server

It supports platform-independent web services, business applications, and standards-based development, enabling you to leverage existing technology assets for Web-services-oriented solutions.

SAP NetWeaver Business Warehouse

It enables you to integrate data from across the enterprise and transform it into practical, timely business information to drive sound decision making.

SAP NetWeaver Gateway

It enables developers to create applications that link business users to SAP software from any environment and through any device.

SAP NetWeaver Master Data Management

It ensures cross-system data consistency and helps integrate business processes across the extended value chain.

SAP NetWeaver Process Orchestration

It helps improve processes, from simple workflows to integrated processes that span applications and organizational boundaries. It includes capabilities for business process management, business rules management, and process integration.

SAP NetWeaver Portal

It unifies critical information and applications to give users role-based views that span the enterprise, enabling you to take full advantage of your information resources.

SAP Auto-ID Infrastructure

It gives you all the capabilities you need to integrate all automated sensing devices including RFID readers and printers, Bluetooth devices, embedded systems, and barcode devices.

SAP NetWeaver Identity Management

It addresses access and provisioning issues facing a typical enterprise. It creates a new opportunity for integrating business processes, and helps you to integrate systems in a heterogeneous IT environment.

SAP NetWeaver Information Lifecycle Management

It allows you to archive data in a readily accessible format according to regulatory retention rules that you define.

SAP NetWeaver Tools: SAP NetWeaver includes the following tools

Adaptive Computing Controller:

It provides a central point of control for assigning computing resources and optimizing their use.

SAP NetWeaver Composition Environment

It provides a robust environment for design, deployment, and running of composite applications that comply with a service-oriented architecture.

SAP NetWeaver Developer Studio

It offers a convenient user interface and rich functionality for developing J2EE applications.

SAP NetWeaver Visual Composer

It simplifies the creation of portal content and analytics applications, enabling business analysts to build or customize applications using a visual user interface rather than manual coding.

SAP Solution Manager

It facilitates technical support for distributed systems with functionality that covers all key aspects of solution deployment, operation, and continuous improvement.

SAP NetWeaver Applications: SAP NetWeaver includes the following applications

SAP NetWeaver Enterprise Search

It provides a simple and secure gateway to enterprise objects and transactions.

SAP NetWeaver Single Sign-On

It offers a comprehensive single sign-on solution, enabling reuse of a person's initial authentication for subsequent log-ins to all applications.

---

5. Explain in detail about Customer Relationship Management.

CUSTOMER RELATIONSHIP MANAGEMENT (CRM)

- Customer relationship management (CRM) is a term that refers to practices, strategies and technologies that companies use to manage and analyze customer interactions and data throughout the customer lifecycle, with the goal of improving business relationships with customers, assisting in customer retention and driving sales growth.
- CRM systems are designed to compile information on customers across different channels -- or points of contact between the customer and the company -- which could include the company's website, telephone, live chat, direct mail, marketing materials and social media.
- CRM systems can also give customer-facing staff detailed information on customers' personal information, purchase history, buying preferences and concerns.
- Customer relationship management (CRM) entails all aspects of interaction that a company has with its customers, whether it is sales or service-related.

- While the phrase customer relationship management is most commonly used to describe a business-customer relationship (B2C), CRM systems are also used to manage business to business to business (B2B) relationships. Information tracked in a CRM system includes contacts, clients, contract wins and sales leads and more.

CRM software

CRM software consolidates customer information and documents into a single CRM database so business users can more easily access and manage it. The other main functions of this software include recording various customer interactions (over email, phone calls, social media or other channels, depending on system capabilities), automating various workflow processes such as tasks, calendars and alerts, and giving managers the ability to track performance and productivity based on information logged within the system.

Common features of CRM software include:

- **Marketing automation**: CRM tools with marketing automation capabilities can automate repetitive tasks to enhance marketing efforts to customers at different points in the lifecycle. For example, as sales prospects come into the system, the system might automatically send them marketing materials, typically via email or social media, with the goal of turning a sales lead into a full-fledged customer.
- **Sales force automation**: Also known as sales force management, sales force automation is meant to prevent duplicate efforts between a salesperson and a customer. A CRM system can help achieve this by automatically tracking all contact and follow-ups between both sides.
- **Contact center automation**: Designed to reduce tedious aspects of a contact center agent's job, contact center automation might include pre-recorded audio that assists in customer problem-solving and information dissemination. Various software tools that integrate with the agent's desktop tools can handle customer requests in order to cut down the time of calls and simplify customer service processes.
- **Geolocation technology or location-based services**: Some CRM systems include technology that can create geographic marketing campaigns based on customers' physical locations, sometimes integrating with popular location-based GPS apps. Geolocation technology can also be used as a networking or contact management tool in order to find sales prospects based on location.

CRM Strategy

Customer relationship management is often thought of as a business strategy that enables businesses to improve in a number of areas. The CRM strategy allows you to to following:

- Understand the customer
- Retain customers through better customer experience
- Attract new customers
- Win new clients and contracts
- Increase profitably
- Decrease customer management costs

The Benefits of CRM

- The biggest benefit most businesses realize when moving to a CRM system comes directly from having all your business data stored and accessed from a single location.
- Before CRM systems, customer data was spread out over office productivity suite documents, email systems, mobile phone data and even paper note cards and Rolodex entries.
- Storing all the data from all departments (e.g., sales, marketing, customer service and HR) in a central location gives management and employees immediate access to the most recent data when they need it.
- Departments can collaborate with ease, and CRM systems help organization to develop efficient automated processes to improve business processes.
- Other benefits include a 360-degree view of all customer information, knowledge of what customers and the general market want, and integration with your existing applications to consolidate all business information.

---

6. Explain in detail about SAP Business Information Warehouse

Business Information Warehouse (Business Warehouse or SAP BW)

Business Warehouse (BW) is a combination of databases and database management tools that are used to support management decision making.

The reporting, analysis, and interpretation of business data is of central importance to a company in guaranteeing its competitive edge, optimizing processes, and enabling it to react quickly and in line with the market.

As a core component of SAP NetWeaver, the **SAP Business Warehouse** provides data warehousing functionality, a business intelligence platform, and a suite of business intelligence tools that enable businesses to attain these goals. Relevant business information from productive SAP applications and all external data sources can be integrated, transformed, and consolidated in BW with the toolset provided.

Business Warehouse provides flexible reporting and analysis tools to support you in evaluating and interpreting data, as well as facilitating its distribution. Managers are able to make well-founded decisions and determine target-orientated activities on the basis of this analysis.

Advantages:

On one hand SAP Business Information Warehouse's comprehensive Business Content enables a quick and cost-effective implementation.

On the other hand, it provides a model that can be used as a guideline during implementation using experience gained from other implementations.

- Business Information Warehouse (sometimes shortened to "Business Warehouse" or BW) is a packaged, comprehensive business intelligence product centered around adata warehouse that is optimized for (but not limited to) the R/3 environment from SAP
- SAP Business Warehouse SAP Business Warehouse (also known as SAP NetWeaver Business Warehouse or SAP BW) is the cornerstone of SAP's strategic Enterprise Data Warehouse solutions and runs on industry standard RDBMS and SAP's HANA in-memory DBMS.
- Like most data warehouses, BW is a combination of databases and database management tools that are used to support management decision making.
- BW supplies the infrastructure typical of data warehouses, but also includes preconfigured data extractors, analysis and report tools, and business process models.
- Among the other features of BW are: Business Application Programming Interfaces (BAPIs) that enable connections to non-R/3 applications; preconfigured business content; an integrated OLAP processor; automated data extraction and loading routines; a metadatarepository; administrative tools; multiple language support; and Business Explorer, a Web-based user interface.

- SAP Business Warehouse is an integral component of the company's mySAP Business Intelligence group of products.

---

7. Explain in detail about SAP Advanced Planning and Optimization

SAP Advanced Planning and Optimization (SAP APO)

- ✓ APO (Advanced Planning and Optimization) application is at the heart of SCM.
- ✓ It offers planning and optimization functionalities in different business processes of Demand Planning, Supply Planning, Supply and Demand Matching, Production Planning Detailed Scheduling, Global Available to Promise and Transportation Management.
- ✓ SAP APO functionalities enable Plan to Inventory End to End Business Processes.
- ✓ Collaboration with Suppliers and Customers is also possible through newer application Supply Network Collaboration (SNC) till recently known as Inventory Collation Hub (ICH)
- ✓ With SCM 5.0 a new set of functionalities under Services Parts Planning were added specifically catering to Spare Parts Management.
- ✓ APO as a application is tightly integrated to execution (OLTP) system like ERP using a standard interface called Core Interface Function (Interfaces - Core Interface (CIF) and BAPIs). It also has full BI (erstwhile BW) component for Data Mart as well as Reporting functionalities.

SAP Advanced Planning and Optimization (SAP APO) provides a fully integrated range of functions that you require for planning and executing your logistic processes. SAP APO supports the following:

- ✓ Intercompany interaction on a strategic, tactical, and operative planning level
- ✓ Collaboration with logistic partners from order receipt through stock monitoring to product shipping
- ✓ Maintenance of relationships with customers and business partners
- ✓ Continuous optimization and measurement of the performance of the logistic network

Purpose

- ✓ SAP Advanced Planning and Optimization (SAP APO) offers a fully integrated pallet of functions that you use to plan and execute your supply chain processes. SAP APO supports the following:
- ✓ Business collaboration on a strategic, tactical, and operational planning level
- ✓ Co-operation between partners at all stages of the supply chain process; from order receipt, stock monitoring, through final shipping of the product
- ✓ Cultivation of customer and business partner relationships
- ✓ Constant optimization and evaluation of the supply chain network's efficiency

Features

SAP Advanced Planning and Optimization (SAP APO) can be deployed as part of an SAP SCM Server installation or as an add-on to SAP ERP (as of SAP enhancement package 6 for SAP ERP 6.0).

An SAP SCM Server installation provides the complete scope of SAP APO components and functions. In an add-on deployment to SAP ERP, the following SAP APO components and functions are available:

- ✓ Demand Planning (SCM-APO-FCS)
- ✓ Supply Network Planning (SCM-APO-SNP)
- ✓ Production Planning and Detailed Scheduling (SCM-APO-PPS)
- ✓ Global Available-to-Promise (SCM-APO-ATP)

---

8. Explain in detail about ABAP & its Workbench Tools

ABAP (ADVANCED BUSINESS APPLICATION PROGRAMMING)

ABAP – the programming language used in SAP for developing business application support and development. ABAP is a programming language that runs in the SAP ABAP runtime environment, created and used by SAP for the development of application programs including:

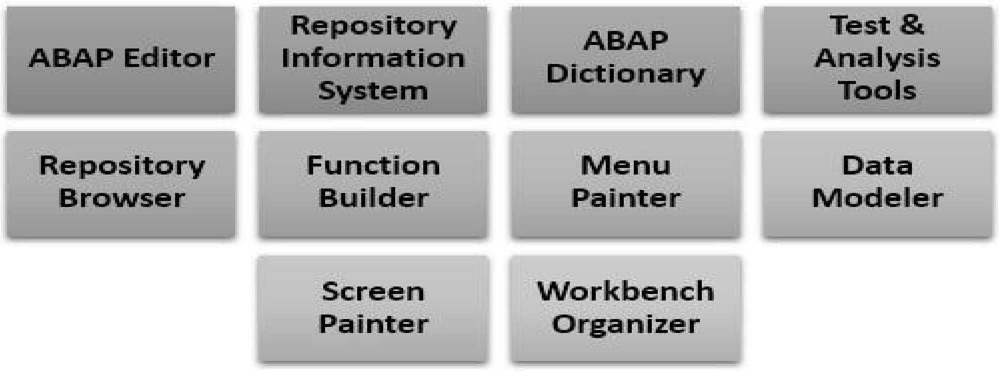
- ✓ Reports
- ✓ Module Pool Programming
- ✓ Interfaces
- ✓ Forms
- ✓ Data conversions
- ✓ User Exits & BADI

All of R/3's applications and even parts of its basis system were developed in ABAP. ABAP is an event-driven programming language. User actions and system events control the execution of an application. ABAP is also called ABAP/4. The "4" in ABAP/4 stands for "Fourth Generation Language" or 4GL.

ABAP Workbench

The ABAP Workbench is used by SAP for the development of standard and custom application software. The ABAP Workbench is also used to create dictionary objects. It consists of the following components −

- ✓ **ABAP Editor** is used to maintain programs.
- ✓ **ABAP Dictionary** is used to maintain Dictionary objects.
- ✓ **Repository Browser** is used to display a hierarchical structure of the components in a package.
- ✓ **Menu Painter** is used to develop graphical user interfaces including menu bars and toolbars.
- ✓ **Screen Painter** is used to maintain screen components for online programs.
- ✓ **Repository Information System** contains information about development and runtime objects, such as data models, dictionary types and table structures, programs, and functions.
- ✓ **Test and Analysis Tools**, such as the Syntax Check and the Debugger.
- ✓ **Function Builder**, which allows to create and maintain function groups and function modules.
- ✓ **Data Modeler**, a tool which supports graphical modeling.
- ✓ **Workbench Organizer**, which maintains multiple development projects and manages their distribution.



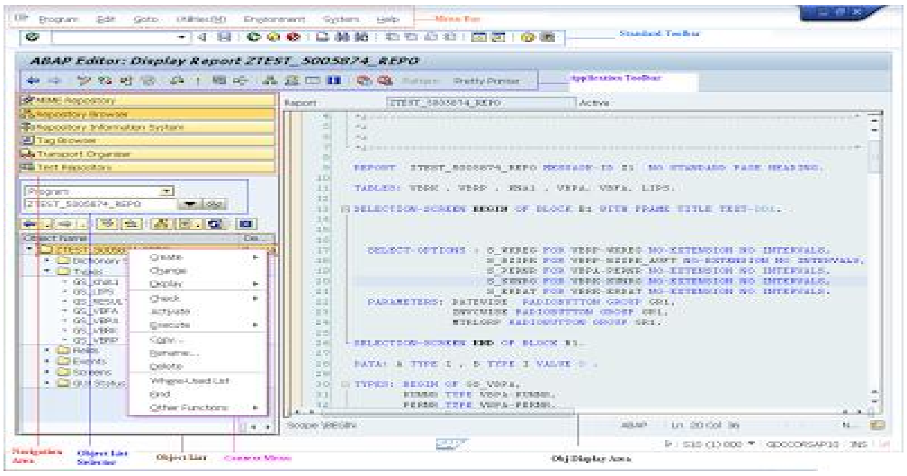
ABAP WORKBENCH TOOLS

The ABAP Workbench is a collection of tools used to develop, test and run ABAP programs. It is a Programming environment GUI in SAP to develop different business applications using ABAP language.

ABAP Workbench Tools - Quick Guide

Object Navigator

Object Navigator is the central point of entry into ABAP workbench as you can access any object of SAP system through it. In SAP all the Development objects are properly arranged together in an object list under some category such as packages, global class, programs etc. The transaction code to Open Object navigator is **SE80.**



ABAP Editor: Display Report ZTEST_5005874_REPO

Report ZTEST_5005874_REPO Active

```
*-----------------------------------------------------------------
*
*
*-----------------------------------------------------------------

REPORT  ZTEST_5005874_REPO MESSAGE-ID Z1  NO STANDARD PAGE HEADING.

TABLES: VBRK , VBRP , KNA1 , VBPA, VBFA, LIPS.

SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME TITLE TEXT-001.



  SELECT-OPTIONS : S_WERKS FOR VBRP-WERKS NO-EXTENSION NO INTERVALS,
                   S_BZIRK FOR VBRP-BZIRK_AUFT NO-EXTENSION NO INTERVALS,
                   S_PERNR FOR VBPA-PERNR NO-EXTENSION NO INTERVALS,
                   S_KUNRG FOR VBRK-KUNRG NO-EXTENSION NO INTERVALS,
                   S_ERDAT FOR VBRK-ERDAT NO-EXTENSION NO INTERVALS.
  PARAMETERS: DATEWISE  RADIOBUTTON GROUP GR1,
              DWTCWISE RADIOBUTTON GROUP GR1,
              WTHLORP RADIOBUTTON GROUP GR1.

SELECTION-SCREEN END OF BLOCK B1.

DATA: A TYPE I , B TYPE I VALUE 0 .

TYPES: BEGIN OF GS_VBPA,
        KUNNR TYPE VBPA-KUNNR,
        PERNR TYPE VBPA-PERNR.
```

Components of Object Navigator Interaface

We can choose browsers from Object Navigation area list. The various browsers are:-

1. **MIME Repository**- Multipurpose Internet Mail Exchange Files Repository- It displays all the directories with the MIME files which were imported into current system.
2. **Repository Browser** - It displays repository objects (Please note that all ABAP programs are Objects to SAP) in the form of object list which are organized by selection categories like programs, packages, classes etc. This is the default displayed browser by object navigator.
3. **Repository Information System**- Unlike Repository browser it displays all the available objects from information system without any search category.
4. **Tag Browser** displays all the TAGs for web-apps.
5. **Transport Organizer-** It displays the Transport request sent to it by user based on the request or task number.
6. **Test Repository-** Displays results of the test cases after testing repository objects.

Object Navigator facilitates users to perform the following tasks:

- ✓ Select a browser and navigate in the object list.
- ✓ Use the tools for development objects .
- ✓ Navigate from one window to another window.
- ✓ Perform syntax checks in the integrated window.
- ✓ Open an object in a new session using an Additional dialog box.

ABAP Editor

SAP has provided Transaction SE38 for ABAP editor. All the reports and includes and other programs are created/edited using this transaction. ABAP pers spend most of their time in the ABAP editor.

- ✓ **Function Builder**

  Transaction SE37 is used for accessing function modules. Through this we can access all the SAP standard function modules and we can also create our own 'Z' function modules.

- ✓ **Class Builder**

  Similarly to the function Builder is the class builder which is used for ABAP objects programming. We can access all the SAP standard classes and can as well create our classes using class builder. Transction code SE24 is used for this purpose.

- ✓ **Screen Painter**

  Screen Painter is mainly used for either creating forms or Dialog Programs. Screen Painter provides us with various tools like Text I/P O/P, box, table wizards etc which could be used to create our own GUI screens for programs. Screen painter is accessible via transaction code SE51.

- ✓ **Menu Painter**

  Menu Painter is used to design the user interfaces for the programs. Using Menu Painter we can customize the user menu. By default the SAP provides the user with all the available options for all the 'Z' programs we create. Now to change this or limit the user options we create our pf_status using transaction SE41 (Menu Painter) ans set the same PF_STATUS in our program.

- ✓ **Message Maintenance**

  Messages helps SAP system to communicate with the user. Messages of types Information-I, Error-E, Warning-W, Success-S could be displayed using message classes. To create a message class could be done at transaction- SE91. Then we could use the message class in our reports/Programs.

ABAP Dictionary

ABAP Dictionary is one of the important tools of ABAP Workbench. It is used to create and manage data definitions without redundancies. ABAP Dictionary always provides the updated information of an object to all the system components. The presence and role of ABAP Dictionary ensures that data stored in an SAP system is consistent and secure. ABAP dictionary could be accessed via transaction SE11.

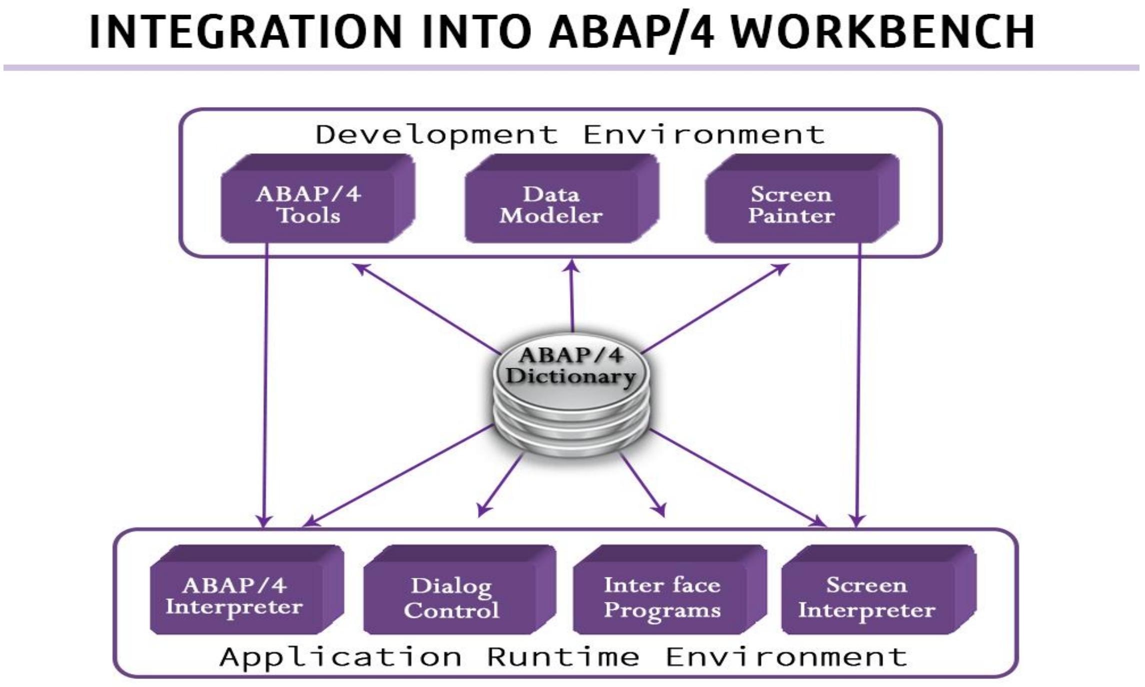
9. Explain in detail about ABAP Data Dictionary

ABAP Data Dictionary

A data dictionary is a central source of information for the data in a information management system. Its main function is to support the creation and management of data definitions (or "metadata").

Data dictionary is used for

- ✓ Management of data definitions
- ✓ Provision of information for evaluations
- ✓ Support for software development
- ✓ Support for documentation
- ✓ Ensuring that data definitions are flexible and up-to-date



Objects in the ABAP Dictionary resided on three levels that support their re-usability. These levels are:

- ✓ Tables and structures
- ✓ Data elements

- ✓ Domains

Domains

- ✓ Describes the technical characteristics of a table field
- ✓ Specifies a value range which describes allowed data values for the fields
- ✓ Fields referring to the same domain (via the data elements assigned to them) are changed when a change is made to the domain
- ✓ Ensures consistency
- ✓ Ex. Purchasing document number (EBELN)

Data Elements

- ✓ Describes the role played by a field in a technical context
- ✓ Fields of same semantic meaning can refer to the same data element
- ✓ Contains the field information

Tables

- ✓ Represent the Database Tables where data actually resides.
- ✓ Tables can be defined independently of the database in the ABAP Dictionary.
- ✓ The fields of the table are defined with their (database-independent) SAP ABAP data types and lengths.

Structures

- ✓ Are record declarations that do NOT correspond to a Database Table.
- ✓ Just like user-defined data type.
- ✓ Defined like a table and can then be addressed from ABAP programs.
- ✓ Structures contain data only during the runtime of a program.

Aggregated Objects of ABAP Dictionary

Aggregated means consisting of several components. In the ABAP Dictionary, aggregated objects are objects which come from several different transparent tables.

- ✓ Views
- ✓ Search Help
- ✓ Lock Objects

Views

- ✓ Views in SAP _ ABAP are used to summarize data which is distributed among several tables
- ✓ The data of a view is not actually physically stored. The data of a view is instead derived from one or more other tables
- ✓ It is tailored to the needs of a specific application

Search Help

- ✓ A Search help is a tool to help you search for data records in the system

- ✓ An efficient and user-friendly search assists users where the key of a record is unknown

Lock Objects

- ✓ Simultaneous accessing of the same data record by two users in the SAP system is synchronized by a lock mechanism.
- ✓ Locks are set and released by calling certain function modules. These function modules are generated automatically from the definition of so-called lock objects in the ABAP/4 Dictionary
- ✓ **Function modules** :Enqueue_<obj name> - to lock the table dequeue_<obj name> - to release the lock

10. Explain in detail about ABAP Repository Information System

ABAP/4 REPOSITORY INFORMATION SYSTEM

The description of the ABAP/4 Repository Information System (in short: Repository Information System) given here is restricted to aspects of relevance for work with the Data Modeler. Using the ABAP/4 Repository Information System, you can search for all modeling objects of the Data Modeler that you wish to display or edit.

Repository Information System: overview

- ✓ The ABAP/4 Repository Information System provides you with the two basic functions Find and Where-used list.
- ✓ The Find function allows you to find objects of a specific object class that correspond to certain selection criteria. You can, for example, search for a list of all entity types belonging to a particular development class.
- ✓ The Where-used list function allows you to determine the other objects in which a particular object is used. For example, you could search for all data models in which a specific entity type occurs.

Repository Information System: access

- ✓ From the initial screen of the Data Modeler, there are two ways of calling the ABAP/4 Repository Information System.

Call the Repository Information System from the Data Modeler

1. from the menu

- ✓ You can call the ABAP/4 Repository Information System from the initial screen of the Data Modeler by selecting EnvironmentRepository Info Sys. The screen listing the areas of the ABAP/4 Repository Information System is displayed. Most of the areas are preceded by a symbol. This indicates that subareas exist. To display these subareas, select the line you are interested in followed by EditExpand sub tree. The subareas hidden beneath the nodes are displayed.

- ✓ To allow searching for modeling objects, position the cursor on Modeling and select Edit Expand sub tree. Under the heading Data modeling, you will find all the points relating to the Data Modeler, namely Data models, Entity types, and Entity type attributes. When you double-click on one of these points, the corresponding selection screen appears.

2. from a modeling object

- ✓ Leaving the entry field Modeling object empty, select the object class you require under Selection followed by the menu option Find.
- ✓ The corresponding selection screen (for entity types or data models) appears.
- ✓ You can call the ABAP/4 Repository Information System from other points in the Data Modeler. You will be informed of this at the appropriate points of this documentation.

REPOSITORY INFORMATION SYSTEM: SEARCHING FOR OBJECTS

- ✓ You can search for objects in the Data Modeler with the Repository Information System.

Search for objects with the Repository Information System

- ✓ Call the relevant selection screen in the ABAP/4 Repository Information System.
- ✓ The standard selections for the object class in question are displayed.
- ✓ The maximum number of hits to be selected is also shown. These values are preset in your user settings.
- ✓ If you wish to display all the objects from a particular object class, enter * in the first search field on the first line. Then select the menu options Program Execute.

- ✓ A list corresponding to your selection will be output.
- ✓ If you wish to search via fields that are not included in the standard selections, you can display all available selections for a particular object class.
- ✓ To do so, choose all selections. The additional selection options are now displayed underneath the standard selections.
- ✓ To search for objects with particular attributes, you have to make the relevant entries in the search fields provided. The way in which these entries are analyzed is determined via the arrow pushbutton next to the input field.
- ✓ You can search for single values or for ranges of values. If you want to search for an object with a specific attribute, it is sufficient to enter this attribute.
    1. If you wish to search for all data models with names starting with U, it is sufficient to enter **U*** in the field Data model. The option Pattern is then selected automatically.
    2. If you wish to search for all data models whose names start with a letter before U in the alphabet and after X, on the line Data model you must enter **U** in the first field and press the arrow pushbutton. A dialog box appears. In the second field, enter **X**. Position the cursor on each field and click on Options. Select the relevant search criteria and start the search.

Repository Information System: settings

- ✓ The selection options to be defaulted when you are searching for objects and the maximum number of objects matching the search criteria to be displayed are laid down in your user settings

Define the user settings in the Repository Information System

Defining the user settings

- ✓ To define your user settings, you have to call the initial screen of the ABAP/4 Repository Information System.
- ✓ To do so, call the ABAP/4 Development Workbench from the R/3 initial screen with Tools ABAP/4 Workbench.
- ✓ In the ABAP/4 Development Workbench select Overview Repository Info Sys.

Setting a start variant

- ✓ The selection options available to you are determined by the choice of variant. A standard variant is normally set. You can change the variant by selecting Settings User parameters in the initial screen of the ABAP/4 Repository Information System.

- ✓ A dialog box containing a number of different variants appears. Select the variant you require and click on the Save pushbutton. The selected variant is now adopted as your start variant.

Specifying the maximum number of hits

- ✓ The maximum number of hits determines the maximum number of objects matching the search criteria that is to be selected. For example, if 100 is entered as the maximum number of hits and you are searching for all tables with names beginning with U, only the first 100 tables matching this selection criterion will be selected. You can change the standard entry for the maximum number of hits.

- ✓ To do so, select Settings User parameters in the initial screen of the ABAP/4 Repository Information System. A dialog box appears in which you can enter the value you require. Click on Save. Your selection for the maximum number of hits is now adopted as your standard setting.

Repository Information System: selection options

- ✓ By specifying selection options for a field, you can determine how the entry you have made is analyzed during the search process.

Set the selection options in the Repository Information System

- ✓ Press the arrow pushbutton beside the relevant field. A dialog box appears. Click on the Options pushbutton to maintain the selection options:
- ✓ Single value
- ✓ All entries matching the entry value are selected.
- ✓ Greater than or equal
- ✓ All entries greater than or equal to the entry value are selected.
- ✓ Less than or equal
- ✓ All entries with values less than or equal to the entry value are selected.
- ✓ Pattern
- ✓ All entries matching the pattern are selected. This option is displayed only when you have made a generic entry in the field, such as U* in the field Data model.
- ✓ Exclude pattern
- ✓ All entries that do not match the pattern are selected. This option is displayed only if you have made a generic entry in the field, such as U* in the field Data model.

- ✓ Not equal
- ✓ All entries not equal to the entry are selected.
- ✓ Less than
- ✓ All entries with values that are less than the entry value are selected.
- ✓ Greater than
- ✓ All entries with values greater than the entry value are selected.

Repository Information System: examples for selection options

- ✓ When selecting data models, **U** is entered in the field Data model. Then the arrow pushbutton is used and the Options pushbutton chosen.
- ✓ After selecting ProgramExecute, the output list that is displayed varies according to the selection options set for the field Data model.

- Single value
  - ✓ No table is selected, since there is no table of the name U.
  - ✓ Greater than or equal: All tables with names starting with U or with a letter occurring after U in the alphabet are selected.
  - ✓ Less than or equal: All tables with names starting with U or with a letter occurring before U in the alphabet are selected.
  - ✓ Not equal: All tables with names that do not begin with U are selected.
  - ✓ Less than: All tables with names starting with a letter that occurs before U in the alphabet are selected.
  - ✓ Greater than: All tables with names starting with a letter occurring after U in the alphabet are selected.
  - ✓ When selecting data models, **U*** is entered in the field Data model. The selection options Pattern and Exclude pattern result in the following selections being made:
  - ✓ Pattern
  - ✓ All tables with names starting with U are selected.
  - ✓ Exclude pattern
  - ✓ All tables with names that do not begin with U are selected.

Repository Information System: where-used list

This function allows you to determine the other objects in which a particular object is used.

Use the Repository Information System to find out where objects are used

- ✓ Access the ABAP/4 Repository Information System.

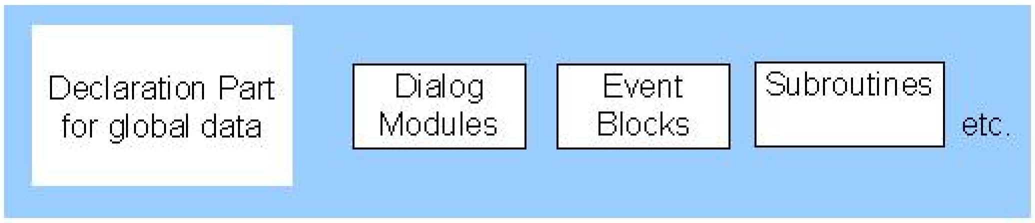
- ✓ To do so, select ToolsABAP/4 Workbench in the initial screen of the R/3 System. In the ABAP/4 Development Workbench, select Overview Repository Info Sys.
- ✓ To find out where modeling objects are used, position the cursor on Modeling and select the menu options Edit Expand sub tree.
- ✓ Under the heading Data modeling, you will find all the points relating to the Data Modeler, namely Data models, Entity types, and Entity type attributes.
- ✓ Click on Data models or Entity types and select Repository Infosys Where-used list.
- ✓ A dialog box appears in which you have to enter the name of the object for which you wish to see a where-used list. In the case of entity types, you also have to specify which type of where-used list you require (i.e. for use in data models or in tables).
- ✓ Once you have done so, click on Cont. A hit list is displayed.
- ✓ You can execute the following functions from within this list.
- ✓ You can access the display screen for an object by selecting the object you are interested in choosing Display.
- ✓ You can access the maintenance screen for an object from within the list by selecting the object in the first column of the display and selecting Change.
- ✓ You can display the occurrences of an object from the list in other objects by selecting the object and choosing Utilities Where-used list.

11. Explain in detail about Structure of ABAP Programs

STRUCTURE OF ABAP PROGRAMS:

ABAP processing logic is responsible for processing data in R/3 application programs. ABAP was designed specifically for dialog-oriented database applications. The following sections deal with how an ABAP program is structured and executed. ABAP programs are responsible for data processing within the individual **dialog steps** of an application program. This means that the program cannot be constructed as a single sequential unit, but must be divided into sections that can be assigned to the individual dialog steps. To meet this requirement, ABAP programs have a modular structure. Each module is called a **processing block**. A processing block consists of a set of ABAP statements. When you run a program, you effectively call a series of processing blocks. They cannot be nested.

The following diagram shows the structure of an ABAP program:



Each ABAP program consists of the following two parts:

Declaration Part for Global Data, Classes and Selection Screens

- ✓ The first part of an ABAP program is the declaration part for global data, classes, and selection screens. This consists of:
- ✓ All declaration statements for global data. Global data is visible in all internal processing blocks. You define it using declarative statements that appear before the first processing block, in dialog modules, or in event blocks. You cannot declare local data in dialog modules or event blocks.
- ✓ All selection screen definitions.
- ✓ All local class definitions (CLASS DEFINITION statement). Local classes are part of ABAP Objects, the object-oriented extension of ABAP.
- ✓ Declaration statements which occur in procedures (methods, subroutines, function modules) form the declaration part for local data in those processing blocks. This data is only visible within the procedure in which it is declared.

Container for Processing Blocks

- ✓ The second part of an ABAP program contains all of the processing blocks for the program. The following types of processing blocks are allowed:
- ✓ Dialog modules (no local data area)
- ✓ Event blocks (no local data area)
- ✓ Procedures (methods, subroutines and function modules with their own local data area).
- ✓ Whereas dialog modules and procedures are enclosed in the ABAP keywords which define them, event blocks are introduced with event keywords and concluded implicitly by the beginning of the next processing block.
- ✓ All ABAP statements (except declarative statements in the declaration part of the program) are part of a processing block. Non-declarative ABAP statements,

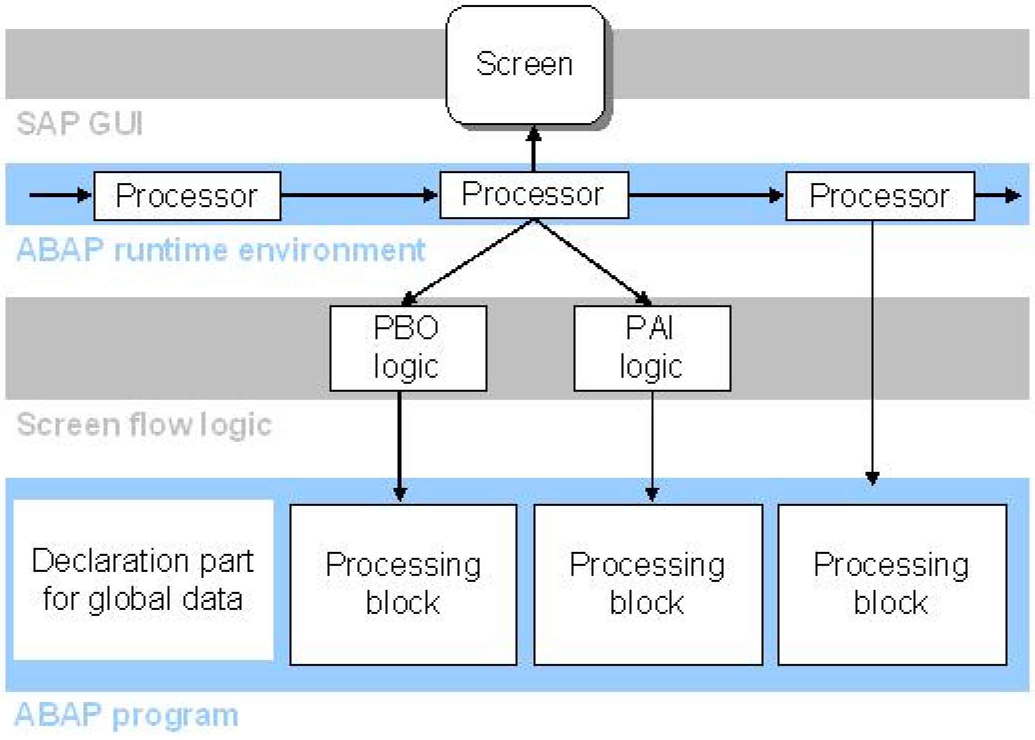
which occur between the declaration of global data and a processing block are automatically assigned to the START-OF-SELECTION processing block.

Calling Processing Blocks

- ✓ You can call processing blocks either from outside the ABAP program or using ABAP commands which are themselves part of a processing block. Dialog modules and event blocks are called from outside the ABAP program. Procedures are called using ABAP statements in ABAP programs.
- ✓ Calling event blocks is different from calling other processing blocks for the following reasons:
- ✓ An event block call is triggered by an event. User actions on selection screens and lists, and the runtime environment trigger events that can be processed in ABAP programs. You only have to define event blocks for the events to which you want the program to react (whereas a subroutine call, for example, must have a corresponding subroutine).

Program Types and Execution

When you run an ABAP program, you call its processing blocks. ABAP programs are controlled from outside the program itself by the processors in the current work process. For the purposes of program flow, we can summarize the screen processor and ABAP processor into the ABAP runtime environment. The runtime environment controls screens and ABAP processing blocks. It contains a range of special control patterns that call screens and processing blocks in certain orders. These sections are also called processors. When you run an ABAP program, the control passes between various processors.



In the R/3 System, there are various types of ABAP program. The program type determines the basic technical attributes of the program, and you must set it when you create it. The main difference between the different program types is the way in which the runtime environment calls its processing blocks.

When you run an application program, you must call at least the first processing block from outside the program, that is, from the runtime environment. This processing block can then either call further processing blocks or return control to the runtime environment. When you start an ABAP program, the runtime environment starts a processor (dependent on the program type), which calls the first ABAP processing block. An ABAP program can be started either by the user or by the system (for example, in background processing), or through an external interface (for example, Remote Function Call).

There are two ways of allowing users to execute programs - either by entering the program name or by entering a transaction code. You can assign a transaction code to any program. Users can then start that program by entering the code in the command field. Transaction codes are also usually linked to a menu path within the R/3 System.

The following program types are relevant to application programming:

Type 1: Type 1 programs have the important characteristic that they do not have to be controlled using user-defined screens. Instead, they are controlled by the runtime environment, which calls a series of processing blocks (and selection screens and lists where necessary) in a fixed sequence. User actions on screens can then trigger further processing blocks.

You can start a type 1 program and the corresponding processor in the runtime environment using the SUBMIT statement in another ABAP program. There are also various ways of starting a type1 program by entering its program name. This is why we refer to type 1 programs as executable programs.

When you run a type 1 program, a series of processors run in a particular order in the runtime environment. The process flow allows the user to enter selection parameters on a selection screen. The data is them selected from the database and processed. Finally, an output list is displayed. At no stage does the programmer have to define his or her own screens. The runtime environment also allows you to work with a logical database. A logical database is a special ABAP program which combines the contents of certain database tables. The flow of a type 1 program is oriented towards reporting, whose main tasks are to read data from the database, process it, and display the results. This is why executable programs (type 1) in the R/3 System are often referred to as **reports**, and why running an executable program is often called **reporting**.

Since it is not compulsory to define event blocks, you can yourself determine the events to which your ABAP program should react. Furthermore, you can call your own screens or processing blocks at any time, leaving the prescribed program flow. You can use this, for example, to present data in a table on a dialog screen instead of in a list. The simplest executable program (report) contains only one processing block (START-OF-SELECTION).

Executable programs do not require any user dialog. You can fill the selection screen using a variant and output data directly to the spool system instead of to a list. This makes executable programs (reports) the means of background processing in the R/3 System.

You can also assign a transaction code to an executable program. Users can then start it using the transaction code and not the program name. The reporting-oriented runtime environment is also called when you run a report using a transaction code. This kind of transaction is called a **report transaction**.

It is appropriate to use executable programs (reports) when the flow of your program corresponds either wholly or in part to the pre-defined flow of the runtime environment. Until Release 4.5A, the only way to use a logical database was to use an executable program. However, from Release 4.5A, it is also possible to call logical databases on their own.

Type M: The most important technical attribute of a type M program is that it can only be controlled using screen flow logic. You must start them using a transaction code,whcih is linked to the program and one of its screens (initial screen). Another feature of these programs is that you must define your own screens in the Screen Painter (although the intial screen can be a selection screen).

When you start a program using a transaction code, the runtime environment starts a processor that calls the initial screen. This then calls a dialog module in the corresponding ABAP program. The remainder of the program flow can take any form. For example, the dialog module can:

- ✓ return control to the screen, after which, the processing passes to a subsequent screen. Each screen has a following screen, set either statically or dynamically.
- ✓ call other sequences of screens, selection screens or lists, from which further processing blocks in the ABAP program are started.
- ✓ call other processing blocks itself, either internally or externally.
- ✓ call other application programs using CALL TRANSACTION (type M program) or SUBMIT (type 1 program).

ABAP programs with type M contain the dialog modules belonging to the various screens. They are therefore known as **module pools**. It is appropriate to use module pools when you write dialog-oriented programs using a large number of screens whose flow logic largely determines the program flow.

Type F: Type F programs are containers for **function modules**, and cannot be started using a transaction code or by entering their name directly. Function modules are special procedures that you can call from other ABAP programs.

Type F programs are known as **function groups**. Function modules may only be programmed in function groups. The **Function Builder** is a tool in the ABAP Workbench that you can use to create function groups and function modules. Apart from function modules, function groups can contain global data declarations and subroutines. These are visible to all function modules in the group. They can also contain event blocks for screens in function modules.

Type K: You cannot start type K programs using a transaction code or by entering the program name. They are containers for **global classes** in ABAP Objects . Type K programs are known as **class definitions**. The **Class Builder** is a tool in the ABAP Workbench that you can use to create class definitions.

Type J: You cannot start type J programs using a transaction code or by entering the program name. They are containers for **global interface** in ABAP Objects . Type J programs are known as **interface definitions**. Like class definitions, you create interface definitions in the **Class Builder**.

Type S: You cannot start a type S program using a transaction code or by entering the program name. Instead, they are containers for subroutines, which you can call externally from other ABAP programs. Type S programs are known as **subroutine pools**. They cannot contain screens.

Type I: Type I programs - called **includes** - are a means of dividing up program code into smaller, more manageable units. You can insert the coding of an include program at any point in another ABAP program using the INCLUDE statement. There is no technical relationship between include programs and processing blocks. Includes are more suitable for logical programming units, such as data declarations, or sets of similar processing blocks. The ABAP Workbench has a mechanism for automatically dividing up module pools and function groups into include programs.

---

12. Explain in detail about ABAP4 Syntax:

ABAP SYNTAX

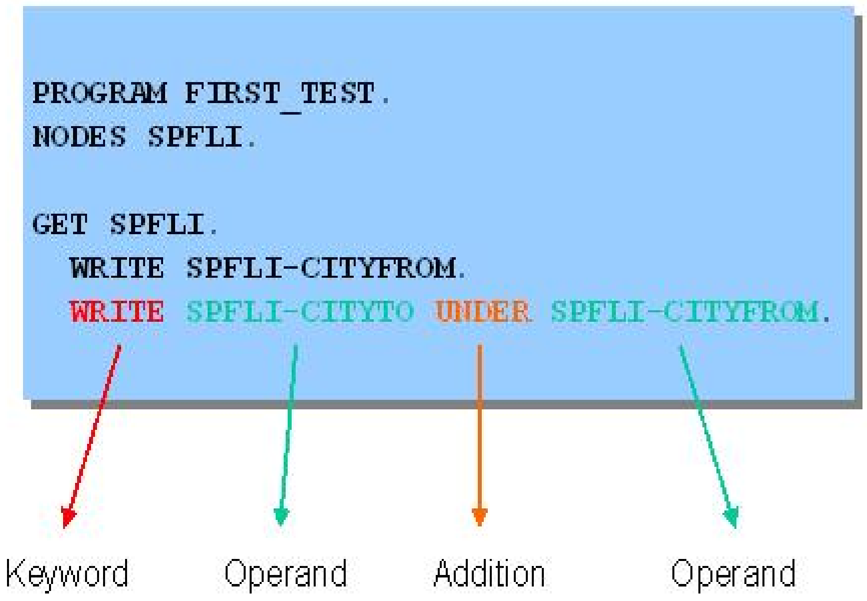
The syntax of the ABAP programming language consists of the following elements:

Statements

An ABAP program consists of individual ABAP statements. Each statement begins with a keyword and ends with a period.

```
PROGRAM FIRST_PROGRAM.
WRITE 'My First Program'.
```

This example contains two statements, one on each line. The keywords are PROGRAM and WRITE. The program displays a list on the screen. In this case, the list consists of the line "My First Program". The keyword determines the category of the statement. For an overview of the different categories, refer toABAP Statements.



```
PROGRAM FIRST_TEST.
NODES SPFLI.

GET SPFLI.
  WRITE SPFLI-CITYFROM.
  WRITE SPFLI-CITYTO UNDER SPFLI-CITYFROM.
```

This diagram shows the structure of an ABAP statement.

13. Explain in detail about ABAP4 Data Types

ABAP Data Types

Data Type describes the technical characteristics of a Variable of that type. Data type is just the blue print of a variable.

Predefined ABAP Types

)ATA TYPE	**)ESCRIPTION**	**)EFAULT .ENGTH**	**)EFAULT 'ALUE**
:	:haracter		‘
I	Iumeric		)
)	)ate	.	)0000000
\`	'ime	)	)00000
:	Iexa Decimal		:'0'
	nteger	.	)
'	'acked	.	)

`	`loat		

User defined data types

Use **TYPES** keyword to define the data types.

```
TYPES: name(10) TYPE c,
length   TYPE p DECIMALS 2,
counter  TYPE i,
id(5)    TYPE n.
```

Structured data types

Structured data type is grouping of several simple data types under one name. Use the keywords **BEGIN OF** and **END OF** to create a structured data type.

```
TYPES: BEGIN OF student,
id(5)     TYPE n,
name(10)  TYPE c,
dob       TYPE d,
place(10) TYPE c,
       END OF student.
```

Constants

Constants are used to store a value under a name. We must specify the value when we declare a constant and the value cannot be changed later in the program. Use **CONSTANTS** keyword to declare a constant.

```
CONSTANTS: pi  TYPE p DECIMALS 2 VALUE '3.14',
yes TYPE c VALUE 'X'.
```

Existing Data Types

- ✓ **ACCP**: Posting period. The length is set to 6 places for this data type. The format is YYYYMM. In input and output, the system inserts a point between the year and month, so the template of this data type has the format '____.__'.
- ✓ **CHAR**: Character string. Fields of type CHAR can have a maximum length of 1333 in tables. If you want to use longer character fields in tables, you must choose data type LCHR. There are no restrictions on the length of such fields in structures.
- ✓ **CLNT**: Client. Client fields always have three places.

- ✓ **CUKY**: Currency key. Fields of this type are referenced by fields of type CURR. The length is set to 5 places for this data type.
- ✓ **CURR**: Currency field. Equivalent to an amount field DEC. A field of this type must refer to a field of type CUKY (reference field). The maximum length for this data type is 31 places.
- ✓ **DATS**: Date. The length is set to 8 places for this data type. The output template can be defined with the user profile.
- ✓ **DEC**: Counter or amount field with decimal point, sign, and commas separating thousands. A DEC field has a maximum length of 31 places.
- ✓ **FLTP**: Floating point number. The length (including decimal places) is set to 16 places for this data type.
- ✓ **DF34_RAW**: Normalized decimal floating point number. Representation on the database based on type RAW. The values can be sorted and compared according to their numerical value, and they can be used in indexes. Database arithmetic is not available. Decimal floating point numbers of this type have 34 digits in the mantissa, and conform to the IEEE 754r standard. Valid values are numbers between 1E-6143 and 9.999999999999999999999999999999999E+6144, plus the corresponding negative numbers and zero.
- ✓ **DF34_SCL**: Scaled decimal floating point number. The difference between this type and DF34_RAW is that DF34_SCL has an additional column of the type INT2 for the scale. This column is visible, but its value is written and read automatically. The values having this data type can be sorted and compared according to their numerical value, and they can be used in indexes. Database arithmetic is not available. The system supports up to 34 decimal digits in the coefficient. Decimal floating point numbers of this type are represented internally with 34 decimal places according to the IEEE-754 standard. Valid values are numbers between 1E-6143 and 9.999999999999999999999999999999999E+6144, plus the corresponding negative numbers and zero.
- ✓ **DF34_DEC**: Decimal floating point number. Representation on the database with type DEC, length and number of decimal places must be specified by the programmer. The values have at most 31 digits on the database, with at most 14 decimal places. The advantage of this type is that database arithmetic is available. The disadvantage is that values are silently rounded to the specified number of decimal places when they are written into the database. An

overflow can also occur when writing values into the database. In this case the system throws an ABAP-OO exception.

- ✓ **DF16_RAW**: Normalized decimal floating point number. Representation based on type RAW. The values can be sorted and compared according to their numerical value, and they can be used in indexes. Database arithmetic is not available. The system supports up to 16 decimal digits in the coefficient. Decimal floating point numbers of this type are represented internally with 16 decimal places according to the IEEE-754r standard. Valid values are numbers between 1E-383 and 9.999999999999999E+384, plus the corresponding negative numbers plus zero.
- ✓ **DF16_SCL**: Scaled decimal floating point number. The difference between this type and DF16_RAW is that DF16_SCL has an additional column of type INT2 for the scale. This column is visible, but the value is written and read automatically. The values having this data type can be sorted and compared according to their numerical value, and they can be used in indexes. Database arithmetic is not available. The system supports up to 16 decimal digits in the coefficient. Decimal floating point numbers of this type are represented internally with 16 decimal places according to the IEEE-754r standard. Valid values are numbers between 1E-383 and 9.999999999999999E+384, plus the corresponding negative numbers plus zero.
- ✓ **DF16_DEC**: Decimal floating point number. Representation on the database with type DEC, length and number of decimal places must be specified by the programmer. The values have at most 15 digits on the database, with at most 14 decimal places. The advantage of this type is that database arithmetic is available. The disadvantage is that values are silently rounded to the specified number of decimal places when they are written into the database. An overflow can also occur when writing values into the database. In this case, the system throws an ABAP-OO exception.
- ✓ **INT1**: 1-byte integer between 0 and 255. The length is set to 3 places for this data type.
- ✓ **INT2**: 2-byte integer between -32767 and 32767. Fields of this type must be used only for length fields. The system positions these length fields immediately in front of a long field (type LCHR, LRAW). With INSERT or UPDATE on the long field, the database interface enters the length which was actually used in the length field. The length is set to 5 places for this data type.

- ✓ **INT4**: 4-byte integer between -2147483648 and 2147483647. The length for this data type is limited to 10 places.
- ✓ **LANG**: Language key. It has its own field format for special functions. This data type always has length 1. The language key is displayed at the user interface with 2 places, but is stored with 1 place in the database. The conversion exit ISOLA converts the display at the user interface for the database and vice versa. This conversion exit is automatically allocated to a domain with data type LANG at activation.
- ✓ **LCHR**: Character string of any length, but has to be declared with a minimum of 256 characters. You must locate fields of this type at the end of transparent tables (in each table there can be only one such field) and must be preceded by a length field of type INT2. If there is an INSERT or UPDATE in ABAP programs, this length field must be filled with the length actually required. If the length field is not filled correctly, this leads to a data loss in the LCHR field. Fields of this type cannot be used in the WHERE condition of a SELECT statement.
- ✓ **LRAW**: Uninterpreted byte string of any length, but has to be declared with a minimum length of 256. You must locate fields of this type at the end of transparent tables (in each table there can be only one such field) and must be preceded by a length field of type INT2. If there is an INSERT or UPDATE in ABAP programs, this length field must be filled with the length actually required. If the length field is not filled correctly, this leads to a data loss in the LRAW field. A field of this type cannot be used in the WHERE condition of a SELECT statement.
- ✓ **NUMC**: Long character field in which only numbers can be entered. The length of this field is limited to a maximum of 255 places.
- ✓ **PREC**: Obsolete data type. The length is set to 2 places for this data type but internally it is treated like INT2. Dynpro fields of type PREC are restricted to 2 places and must not contain a sign.
- ✓ **QUAN**: Quantity. Equivalent to an amount field DEC. A field of this type must always refer to a units field with UNIT format (reference field). The maximum length for this data type is 31 places.
- ✓ **RAW**: Uninterpreted byte string. Fields of type RAW may have only a maximum length of 255 in tables. If longer raw fields are required in tables, you should select data type LRAW.

- ✓ **RAWSTRING**: Uninterpreted byte string of variable length. In the Dictionary a length can be specified for this type (at least 256 characters). This data type can be used in types (data elements, structures, table types) and domains. You can store binary data of type RAWSTRING in the database. There are restrictions; for a description of them, refer to the documentation of the ABAP statement 'STRING'. In ABAP, this type is implemented as a reference to a storage area of variable size. The system proposes 132 characters as the default for the output length. You cannot attach search helps to components of this type.
- ✓ **STRING**: Character string with variable length This data type can be used only in types (data elements, structures, table types) and domains. In the dictionary a length can be specified for this type (at least 256 characters). It can be used in database tables only with restrictions. For a description of them, refer to the documentation of the ABAP statement 'STRING'. In ABAP, this type is implemented as a reference to a storage area of variable size. The system proposes 132 characters as default for the output length. You cannot attach search helps to components of this type.
- ✓ **SSTRING**: Short character string with variable length. In the Dictionary the number of characters can be specified for this type (from 1 to 1333). This data type can be used only in types (data elements, structures, table types) and domains. It can be used in database tables. To do so, refer to the documentation of the ABAP statement 'STRING'. In ABAP, this type is implemented as a reference to a storage area of variable size. String fields of this type can be used in indexes and in the WHERE condition of a SELECT statement. You cannot use them in table keys.
- ✓ **TIMS**: Time. The length is set to 6 places for this data type. The format is HHMMSS. The template for input and output has the form '__.__.__'.
- ✓ **UNIT**: Unit. Fields of this type are referenced by fields of type QUAN. The length of this data type is set to 2 or 3 places.
- ✓ **VARC**: Character field of variable length. Creation of new fields of this data type is no longer supported.

Reference Types

Reference types describe single fields that can contain references to **global** classes and interfaces from the ABAP class library.

In an ABAP program, you can use the **TYPE** addition to refer directly to a data element. The predefined Dictionary data types of the domain are then converted into the corresponding ABAP types.

---

14. Explain in detail about ABAP4 Tables:

Table Basics

Internal table is a data object in ABAP that exists only at run time of a program. It means when the program execution is complete then the internal table will be lost. We use internal table to store database table data after fetching it by a select query. The ABAP runtime system dynamically manages the internal table's memory. It means we developer do not need to work on memory management of internal table.

Internal table has three parts – **rows, columns & work area**.

- ✓ Rows are the line type of internal table. It is a structure which contains several fields. Those fields are of data elements. We need to declare the structure locally or globally to declare the internal table.
- ✓ Columns are the fields of internal table. Those fields are of different data elements declared by locally or globally.
- ✓ The most important part of an internal table is its work area. Work area is basically the line type of an internal table. It means it has the same structure of the rows of internal table. Work area contains the same fields of same type of the rows. It is of two types – implicit & explicit work area.
- ✓ When we declare an internal table with header line then a work area is automatically created with the same name of the table. This work area is called implicit work area which is actually the header line. There is no need to declare work area separately. This work area / header line contains the same table as of the internal table.

---

15. Explain in detail about Various ABAP4 Statements

1.MOVE

To assign the value of a data object **source** to a variable **destination**, use the following statement:

MOVE source TO destination.

or the equivalent statement

destination = source.

The content of **source** remains unchanged, **source** does not therefore have to be a variable - it can also be a literal, a text symbol, or a constant. You must always specify decimal points with a period (.), regardless of the user's personal settings.

Multiple assignments

f4 = f3 = f2 = f1.

are also possible. ABAP processes them from right to left as follows:

MOVE f1 TO f2.

MOVE f2 TO f3.

MOVE f3 TO f4.

In the **MOVE** statement (or when you assign one value to another with the equal sign), it is not possible to specify the field names dynamically as the contents of other fields. If you need to do this, you must use field symbols .

The source and target fields can be of different data types. The result of the value assignment depends on whether these data types are compatible and whether a type conversion can be performed. If there is no conversion rule between the data types in question, no assignment can be made.

```
DATA:t(10)  TYPE c,
 number TYPE p DECIMALS 2,
 count TYPE i.
t = 1111.
MOVE '5.75' TO number.
count = number.
```

Following these assignments, the fields **t**, **number** and **count** have the values '1111 ', 5.75, and 6 respectively. When you assign the number literal 1111 to T, it is converted into a character field with length 10. When you assign **number** to **count** , the decimal number is rounded to an integer (as long as the program attribute Fixed pt. arithmetic has been set).

Assigning Values Between Components of Structures

The rules for value assignments between data objects also apply to structures. With the command

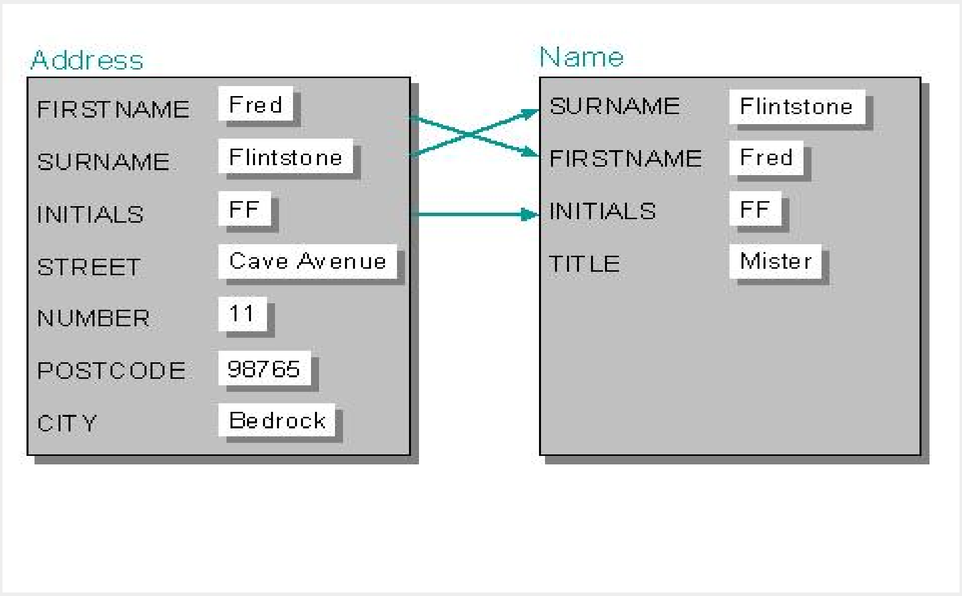
```
DATA: struct1 TYPE structure,
struct2 TYPE structure.
struct1 = struct2.
```

two structures of the same type can be assigned to one another without difficulty. Here, the entire source structure is seen as a unit and copied to the source structure. It is then possible to access the components individually again. If the structures in question are not compatible, see the conversion rules for structures.
In practice, however, you will often only need to assign certain components of a structure to be certain components of another structure. ABAP has a special statement for this purpose:

2.MOVE-CORRESPONDING sourcestruct TO destinationstruct.
This statement assigns the contents of the components of structure **sourcestruct** to the components of the**destinationstruct** structure that have **identical names**.
When it is executed, it is broken down into a set of **MOVE**statements, one for each pair of fields with identical names, as follows:
MOVE sourcestruct-comp1 TO destinationstruct-comp1.
MOVE sourcestruct-comp2 TO destinationstruct-comp2.
...
Any necessary type conversions are performed individually.

```
DATA: BEGIN OF address,
 firstname(20) TYPE c VALUE 'Fred',
 surname(20) TYPE c VALUE 'Flintstone',
 initials(4) TYPE c VALUE 'FF',
 street(20) TYPE c VALUE 'Cave Avenue',
 number TYPE i VALUE '11',
 postcode(5) TYPE n VALUE '98765',
 city(20) TYPE c VALUE 'Bedrock',
 END OF address.
DATA: BEGIN OF name,
 surname(20) TYPE c,
 firstname(20) TYPE c,
 initials(4) TYPE c,
 title(10) TYPE c VALUE 'Mister',
 END OF name.
MOVE-CORRESPONDING address TO name.
```

In this example, the values of **name-surname**, **name-firstname** and **name-initials** are set to 'Flintstone', 'Fred', and 'FF'. **name-title** always has the value 'Mister'.



3.Loop structures

In a loop, a statement block is executed several times in succession. There are four kinds of loops in ABAP:

- ✓ Unconditional loops using the **DO** statement.
- ✓ Conditional loops using the **WHILE** statement.
- ✓ Loops through internal tables and extract datasets using the **LOOP** statement.
- ✓ Loops through datasets from database tables using the **SELECT** statement.
- ✓ This section deals with **DO** and **WHILE** loops. **SELECT** is an Open SQL statement, and is described in the Open SQLsection. The **LOOP** statement is described in the sections on internal tables and extract datasets.

Unconditional Loops

To process a statement block several times unconditionally, use the following control structure:

DO [n TIMES] ...

[statement_block]

ENDDO.

Use the **TIMES** addition to restrict the number of loop passes to **n**.

If you do not specify any additions, the statement block is repeated until it reaches a termination statement such as **EXIT** or **STOP** (see below). The system field **sy-index** contains the number of loop passes, including the current loop pass.

You can nest **DO** loops and combine them with other loop forms.

Simple example of a **DO** loop:

```
DO.
WRITE sy-index.
 IF sy-index = 3.
EXIT.
ENDIF.
ENDDO.
```

The list output is:

```
  1 2 3
```

The loop is processed three times. Here, the processing passes through the loop three times and then leaves it after the **EXIT** statement.

Example of two nested loops with the **TIMES** addition:

```
DO 2 TIMES.
 WRITE sy-index.
SKIP.
 DO 3 TIMES.
 WRITE sy-index.
ENDDO.
SKIP.
ENDDO.
```

The list output is:

```
  1
  1 2 3
  2
  1 2 3
```

The outer loop is processed twice. Each time the outer loop is processed, the inner loop is processed three times. Note that the system field **sy-index** contains the number of loop passes for each loop individually.

Conditional Loops

To repeat a statement block for as long as a certain condition is true, use the following control structure:

```
WHILE log_exp
[statemaent_block]
ENDWHILE.
```

log_exp can be any logical expression. The statement block between **WHILE** and **ENDWHILE** is repeated as long as the condition is true or until a termination statement such as **EXIT** or **STOP** occurs. The system field **sy-index**contains the number of loop passes, including the current loop pass.

You can nest **WHILE** loops to any depth, and combine them with other loop forms.

```
REPORT demo_flow_control_while.
DATA: length  TYPE i VALUE 0,
 strl  TYPE i VALUE 0,
 string(30) TYPE c VALUE 'Test String'.
strl = strlen( string ).
WHILE string NE space.
 WRITE string(1).
 length = sy-index.
 SHIFT string.
ENDWHILE.
WRITE: / 'STRLEN: ',strl.
WRITE: / 'Length of string:', length.
```

The output appears as follows:

```
T e s t  S t r i n g
STRLEN:                                                                    11
Length of String: 11
```

Here, a **WHILE** loop is used to determine the length of a character string. This is done by shifting the string one position to the left each time the loop is processed until it contains only blanks. This example has been chosen to demonstrate the **WHILE** statement. Of course, you can determine the length of the string far more easily and efficiently using the **strlen** function.

Terminating Loops

4. ABAP contains termination statements that allow you to terminate a loop prematurely. There are two categories of termination statement - those that only apply to the loop, and those that apply to the entire processing block in which the loop occurs. The **STOP**and **REJECT** statements belong to the latter group (see Exiting Eventblocks).

The termination statements that apply only to the loop in which they occur are **CONTINUE**, **CHECK**and **EXIT**. You can only use the **CONTINUE** statement in a loop. **CHECK** and **EXIT**, on the other hand, are context-sensitive.Within a loop, they only apply to the execution of the loop itself. Outside of a loop, they terminate the entire processing block in which they occur (subroutine, dialog module, event block, and so on).

CONTINUE, **CHECK** and **EXIT**can be used in all four loop types in ABAP (**DO**, **WHILE**, **LOOP** and **SELECT**).

Terminating a Loop Pass Unconditionally

To terminate a single loop pass immediately and unconditionally, use the **CONTINUE** statement in the statement block of the loop.

CONTINUE.

After the statement, the system ignores any remaining statements in the current statement block, and starts the next loop pass.

```
DO 4 TIMES.
 IF sy-index = 2.
                                                      CONTINUE.
                                                        ENDIF.
 WRITE sy-index.
ENDDO.
```

The list output is:

```
  1 3 4
```

The second loop pass is terminated without the **WRITE** statement being processed.

Terminating a Loop Pass Conditionally

To terminate a single loop pass conditionally, use the **CHECK condition** statement in the statement block of the loop.

5.CHECK condition.

If the condition is not true, any remaining statements in the current statement block after the **CHECK** statement are ignored, and the next loop pass starts. **condition** can be any logical expression.

```
DO 4 TIMES.
 CHECK sy-index BETWEEN 2 and 3.
 WRITE sy-index.
ENDDO.
```

The list output is:

```
  2 3
```

The first and fourth loop passes are terminated without the **WRITE** statement being processed, because **sy-index** is not between 2 and 3.

Exiting a Loop

To terminate an entire loop immediately and unconditionally, use the **EXIT** statement in the statement block of the loop.

6.EXIT.

After this statement, the loop is terminated, and processing resumes after the closing statement of the loop structure (**ENDDO**, **ENDWHILE**, **ENDLOOP**, **ENDSELECT**). In nested loops, only the current loop is terminated.

```
DO 4 TIMES.
 IF sy-index = 3.
                                                   EXIT.
                                                  ENDIF.
 WRITE sy-index.
ENDDO.
```

The list output is:

```
  1 2
```

In the third loop pass, the loop is terminated before the **WRITE** statement is processed.

MOVE-CORRESPONDING

MOVE-CORRESPONDING is a keyword used in SAP ABAP programming

Basic form

MOVE-CORRESPONDING rec1 TO rec2.

Effect Interprets rec1 and rec2 as field strings. If, for example, rec1 and rec2 are tables, executes the statement for their header lines.

Searches for the sub-fields which occur both in rec1 and rec2 and then generates, for all relevant field pairs which correspond to the sub-fields in ,statements of the form

MOVE rec1-ni TO rec2-ni.

The other fields remain unchanged.With complex structures, the full names of the corresponding field pairs must be identical.

Example

DATA: BEGIN OF INT_TABLE OCCURS 10,

WORD(10),

NUMBER TYPE I,

INDEX LIKE SY-INDEX,

END OF INT_TABLE,

BEGIN OF RECORD,

NAME(10) VALUE ‘not WORD’,

NUMBER TYPE I,

INDEX(20),

END OF RECORD.

MOVE-CORRESPONDING INT_TABLE TO RECORD.

This MOVE-CORRESPONDING statement is equivalent to both the following statements:

MOVE INT_TABLE-NUMBER TO RECORD-NUMBER.

MOVE INT_TABLE-INDEX TO RECORD-INDEX.

CHECK (ABAP Keyword)

Basic form

CHECK logexp.

Effect

CHECK evaluates

the subsequent logical expression . If it is true, the processing continues with the next statement. In loop structures like

DO ...

ENDDO

WHILE ... ENDWHILE

LOOP ... ENDLOOP

SELECT ...

ENDSELECT

CHECK with a negative outcome terminates the current loop pass and goes back to the beginning of the loop to start the next pass, if there is one.

In structures like

FORM ... ENDFORM

FUNCTION ...ENDFUNCTION

MODULE ... ENDMODULE

AT

CHECK with a negative

Outcome terminates the routine or modularization unit.
If CHECK is not in a loop or a routine or a modularization unit, a negative logical expression
terminates the current event. In contrast, the statement REJECT terminates the current event, even from loops or subroutines.

CHECK – special for reports with logical databases

Variants
1. CHECK sel.

2. CHECK

SELECT-OPTIONS.
Variant 1
CHECK sel.
Effect
Checks the
selection criterion requested by the statement SELECT-OPTIONS sel ... .
This statement is equivalent to f IN sel , if sel was defined by SELECT-OPTIONS sel FOR f and can be used anywhere in logical expressions
If the result of this check is negative, the processing in this event is terminated and the GET events for any subordinate database tables
are not processed either.
This variant of the CHECK statement should be used only if the logical database for the corresponding table does not support
dynamic selections (see CHECK SELECT-OPTIONS), or SELECT-OPTIONS with the addition NO DATABASE SELECTION . Otherwise, the relevant record is not read from the database and made available to the program.
Variant 2
CHECK
SELECT-OPTIONS.
Effect
Called only after a GET event.
This statement checks all the selections for SELECT-OPTIONS where the reference field after FOR belongs to the current table dbtab (specified after GET . However,

this applies only if the logical database for dbtab does not support dynamic selections . Otherwise, the selections are passed directly to the logical database (with the exception: addition ” NO DATABASE SELECTION ” to SELECT-OPTIONS).

This variant of the CHECK statement only makes sense if the logical database does not support dynamic selections for the corresponding table or SELECT-OPTIONS are defined with the addition ” NO DATABASE SELECTION“.

You can determine from the ABAP/4 Development Workbench whether dynamic selections are defined and, if so, for which logical database tables by selecting Development -> Programming environ. -> Logical databases followed by Extras -> Dynamic selections .

Example

The logical database F1S of the demo flight reservation system contains the tables SPFLI

with, and the table SFLIGHT without, dynamic selections.

TABLES:

SPFLI,

SFLIGHT.

SELECT-OPTIONS:

SF_PRICE FOR SFLIGHT-PRICE, SP_CARR FOR SPFLI-CARRID, SP_FROM FOR SPFLI-CITYFROM NODATABASE SELECTION,SP_DEPT FOR SPFLI-DEPTIME.

Since dynamic selections are defined with the table SPFLI, but not with the table SFLIGHT, the following procedure applies:

…

GET SFLIGHT.

CHECK SELECT-OPTIONS.

This

CHECK statement is equivalent to the following statement:

CHECK

SF_PRICE.

With

GET SPFLI.

CHECK

SELECT-OPTIONS.

The CHECK statement is equivalent to the following statement:

CHECK SP_FROM.

Note

With CHECK

SELECT-OPTIONS, fields from superior tables in the database hierarchy are not (!) checked.

Note

Runtime errors

CHECK_SELOPT_ILLEGAL_OPTION:

Wrong ” OPTION ” in SELECT-OPTIONS or RANGES table

CHECK_SELOPT_ILLEGAL_SIGN

: Wrong ” SIGN ” in SELECT-OPTIONS or RANGES table