ABAP Programming Overview

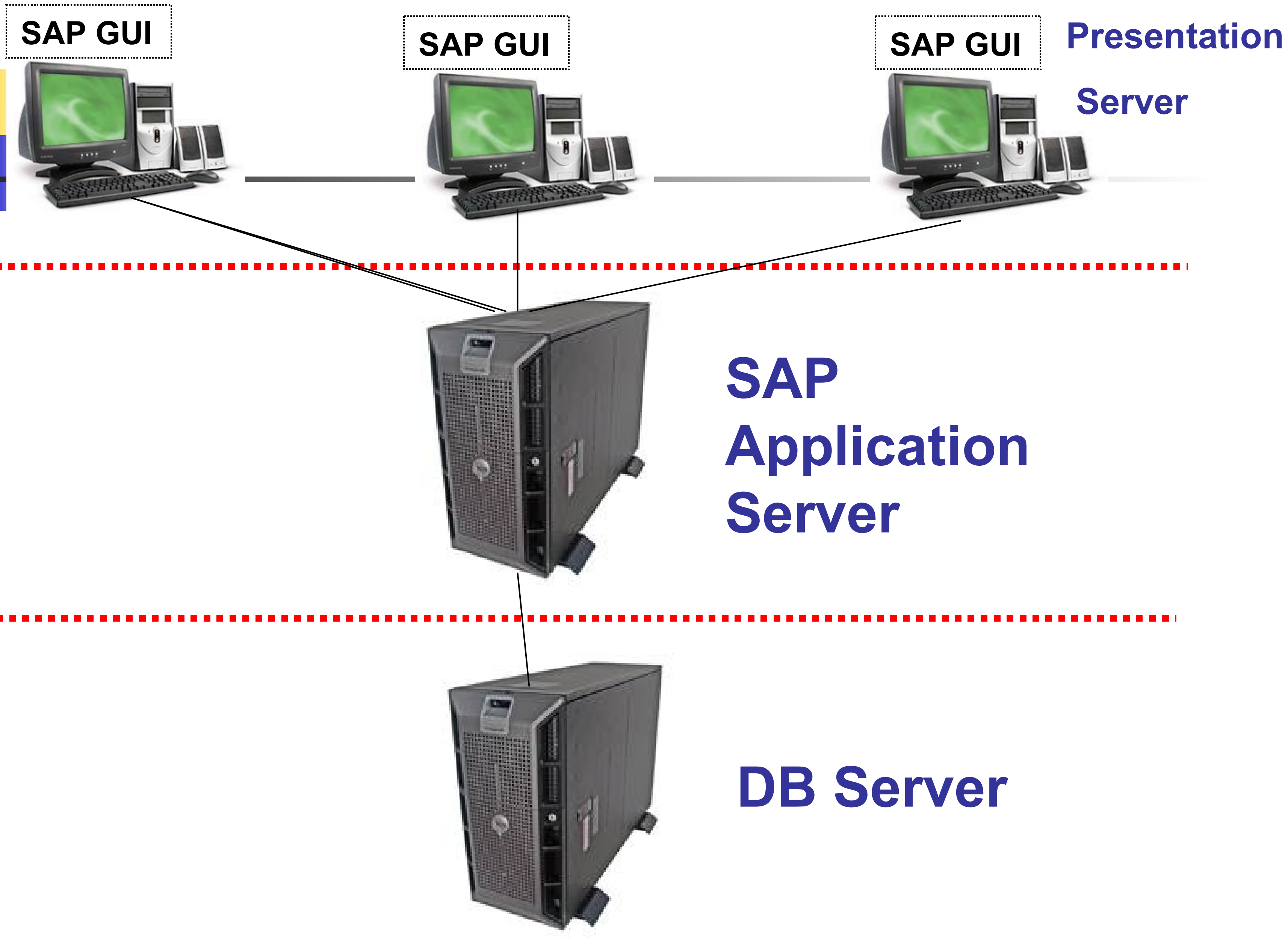
ABAP Course Outline

- Chapter 1 : Introduction to ABAP
- Chapter 2 : List Processing in ABAP
- Chapter 3 : Open SQL & Internal Table
- Chapter 4 : Event-driven Programming & Selection Screen
- Chapter 5 : Modularization & Catch Statement
- Chapter 6 : Message, Debugging, File Transfer and Type Group

ABAP Chapter 1

- Introduction to SAP Architecture
- ABAP Overview
- Data Object in ABAP

SAP System : 3 Tier Client/Server



SAP SYSTEM (3 Tier Architecture)

Presentation Layer

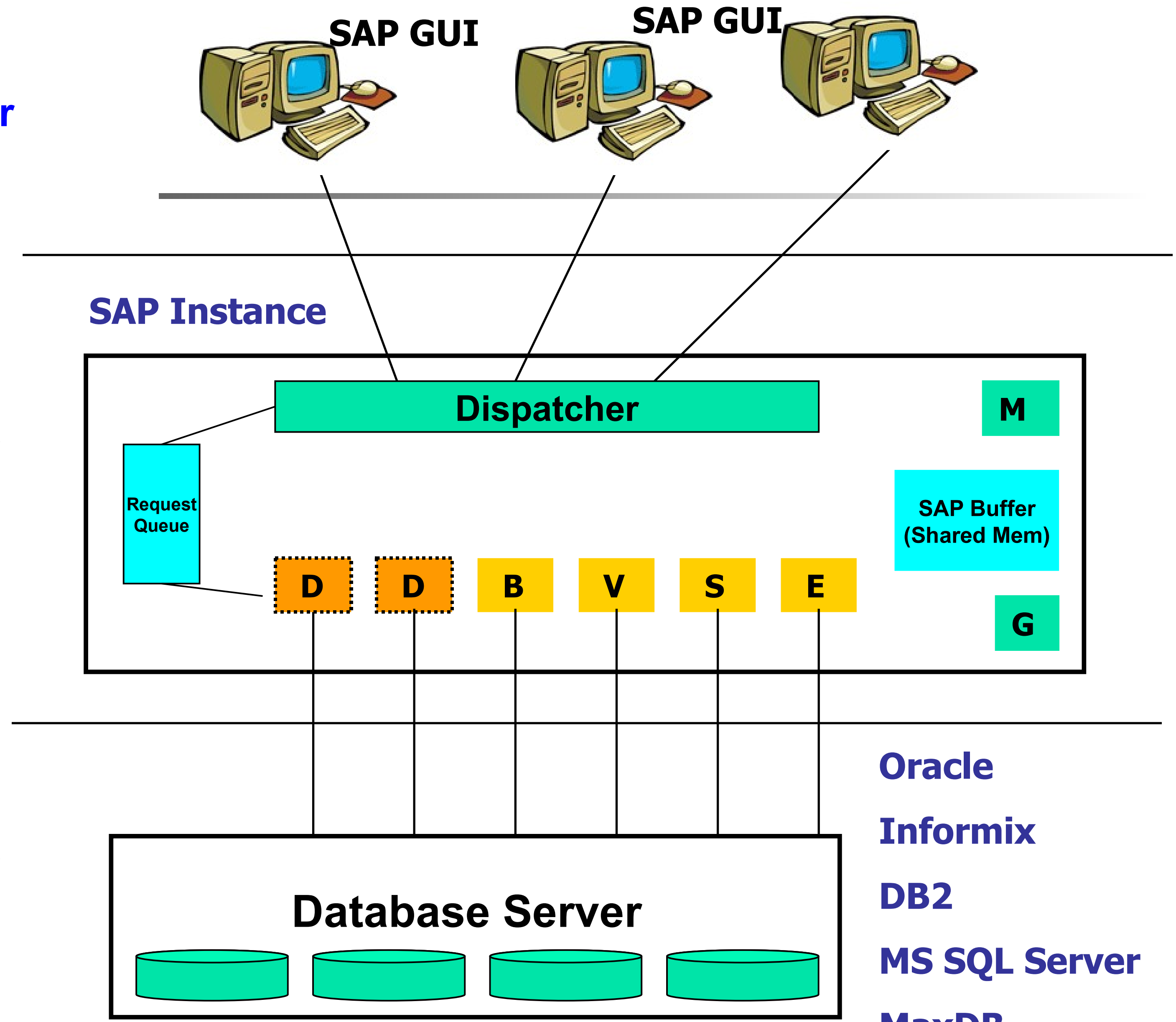
(Windows based)

Application Layer

(Windows Server/UNIX)

Database Layer

(Windows Server/UNIX)



SAP GUI

SAP GUI

SAP Instance

Dispatcher

Request Queue

D D B V S E

M

SAP Buffer (Shared Mem)

G

Database Server

Oracle

Informix

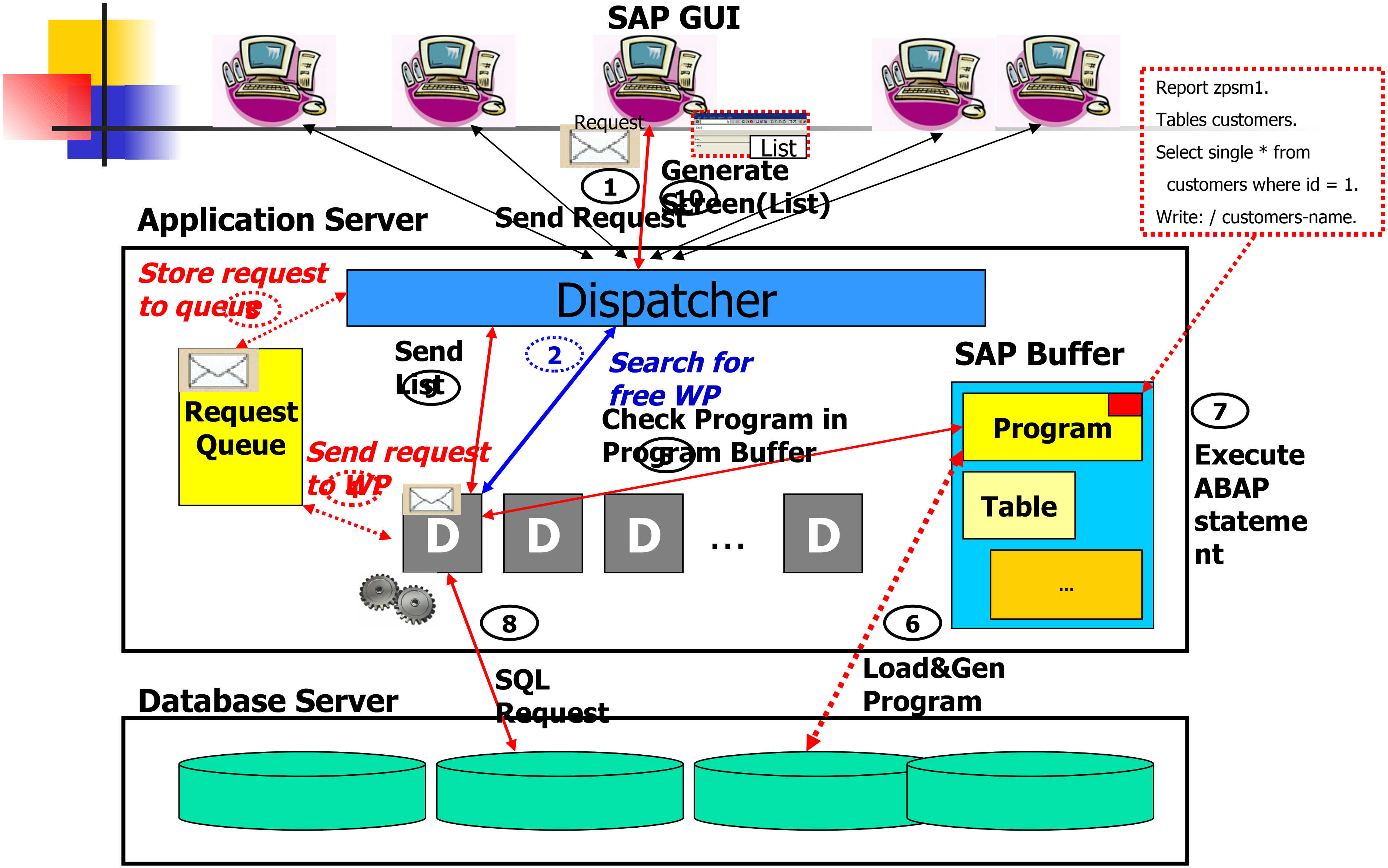
DB2

MS SQL Server

MaxDB

Dialog Processing

SAP System : Dialog Processing



SAP GUI

Request

List

1

Send Request

10

Generate Screen(List)

```
Report zpsm1.
Tables customers.
Select single * from
  customers where id = 1.
Write: / customers-name.
```

Application Server

Store request to queue 3

Dispatcher

Request Queue

Send List 9

2

Search for free WP

Check Program in Program Buffer 5

SAP Buffer

Program

Table

...

7

Execute ABAP statement

Send request to WP 4

D D D … D

8

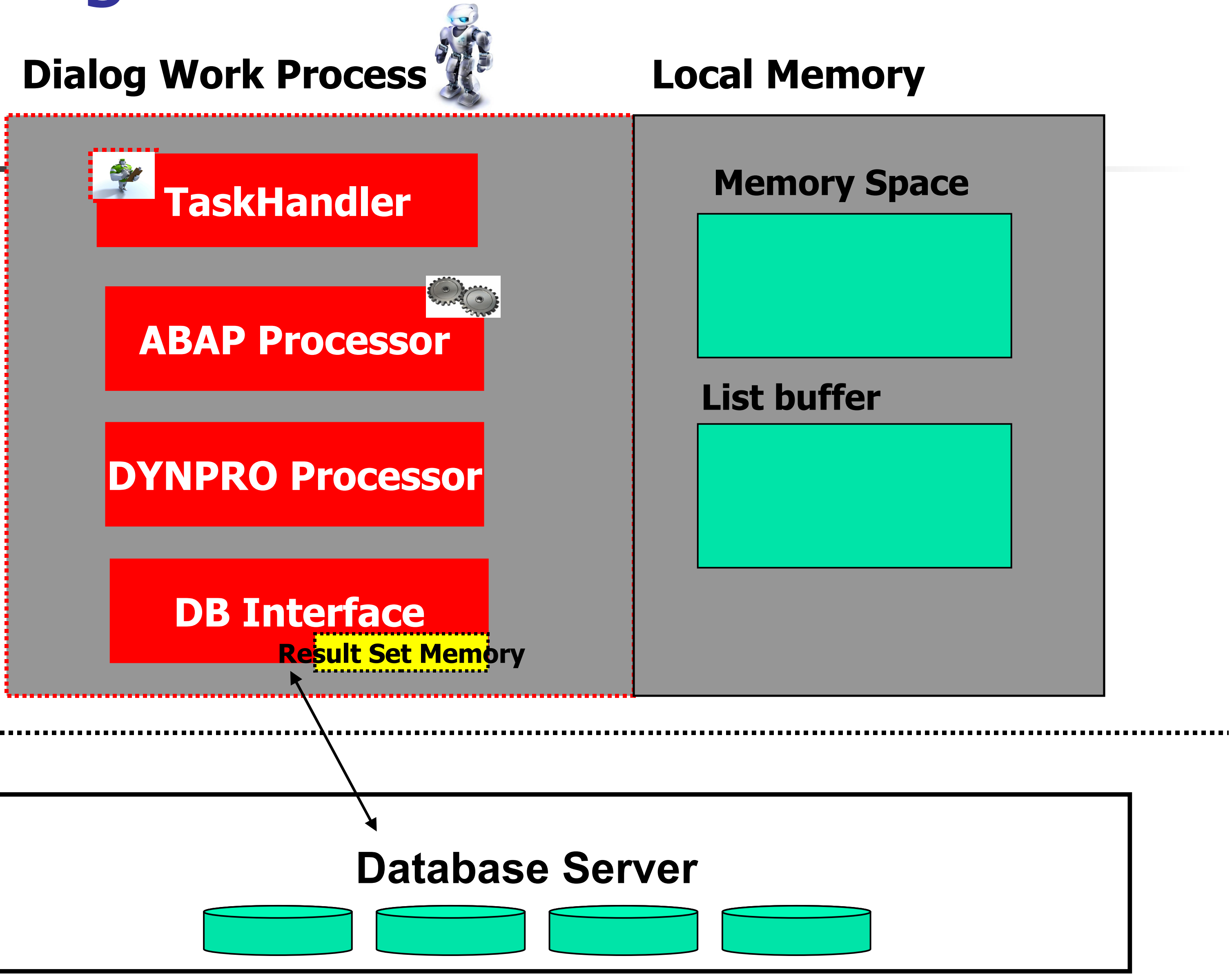
6

Database Server

SQL Request

Load&Gen Program

Dialog Work Process Architecture



ABAP Programming Overview

ABAP Overview

MOVE ...

IF ...

DATA ...

WHILE...

WRITE ...

SEARCH ...

SELECT ...

*Comment...

LOOP AT ...

DO ...

ABAP

Advanced

Business

Application

Programming

ABAP Feature

- Declaring data with various types and structure
- Operational elements for data manipulation
- Control elements for controlling the program flow
- Event elements for reacting to external events

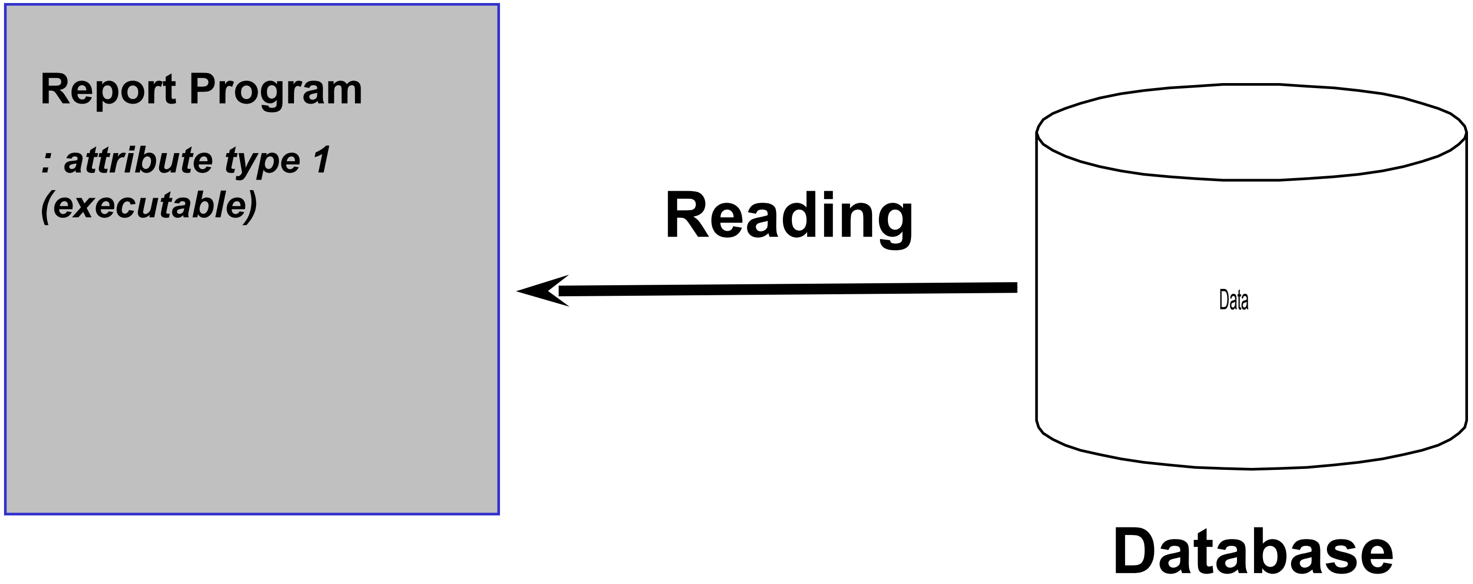
ABAP

- Operating/Database system-independent programming
- ABAP contains a subset of SQL called ***Open SQL*** for comfortable database access for ***various database***

ABAP Programming

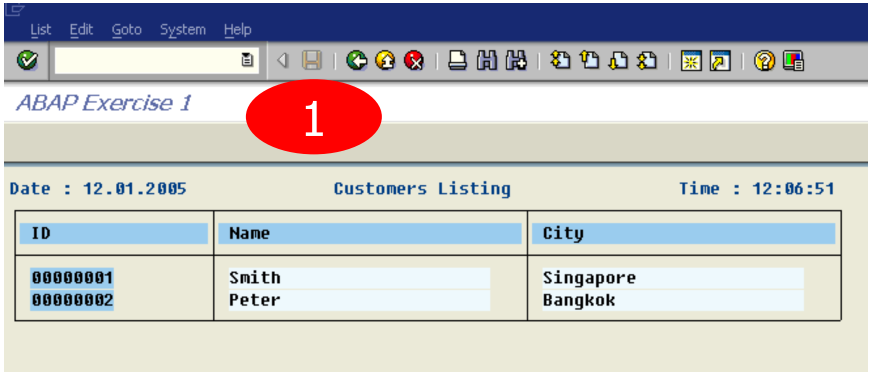
- ABAP Report
- Dialog Programming(Transaction)

ABAP Program : Report



- *Reading data*

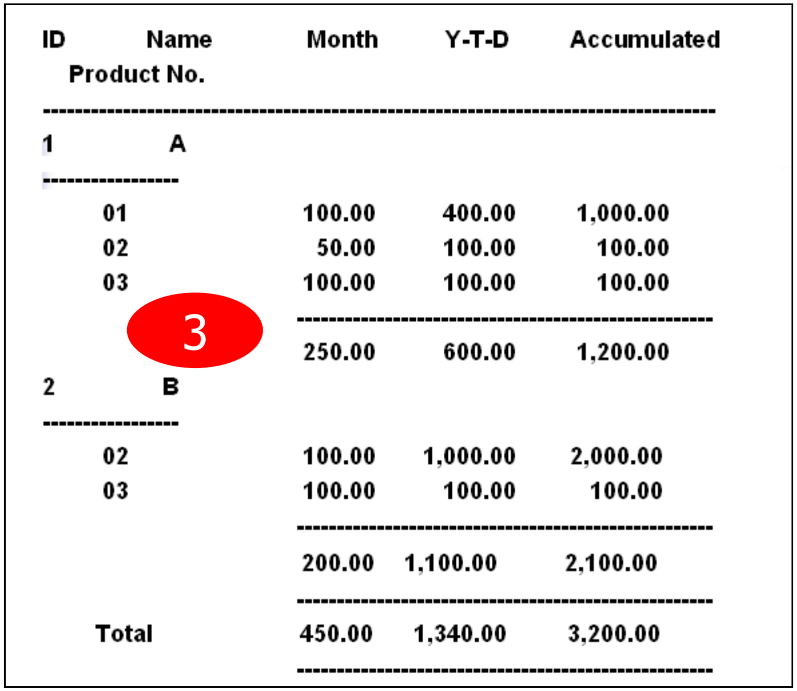
Types of ABAP Report



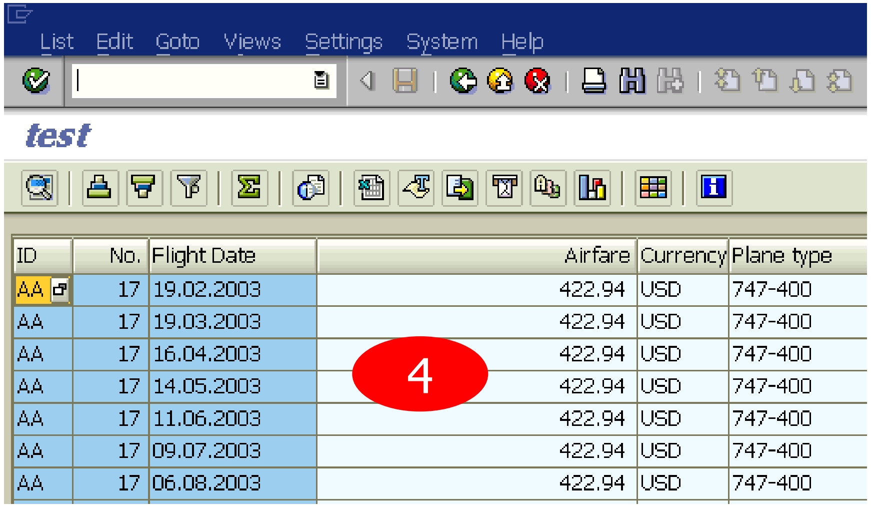
ABAP Exercise 1

```
Date : 12.01.2005          Customers Listing          Time : 12:06:51
```

ID	Name	City
00000001	Smith	Singapore
00000002	Peter	Bangkok



```
ID        Name          Month       Y-T-D       Accumulated
  Product No.
-----------------------------------------------------------
1           A
-----------
     01                 100.00      400.00      1,000.00
     02                  50.00      100.00        100.00
     03                 100.00      100.00        100.00
                        -----------------------------------
                        250.00      600.00      1,200.00
2           B
-----------
     02                 100.00    1,000.00      2,000.00
     03                 100.00      100.00        100.00
                        -----------------------------------
                        200.00    1,100.00      2,100.00
                        -----------------------------------
     Total              450.00    1,340.00      3,200.00
                        -----------------------------------
```

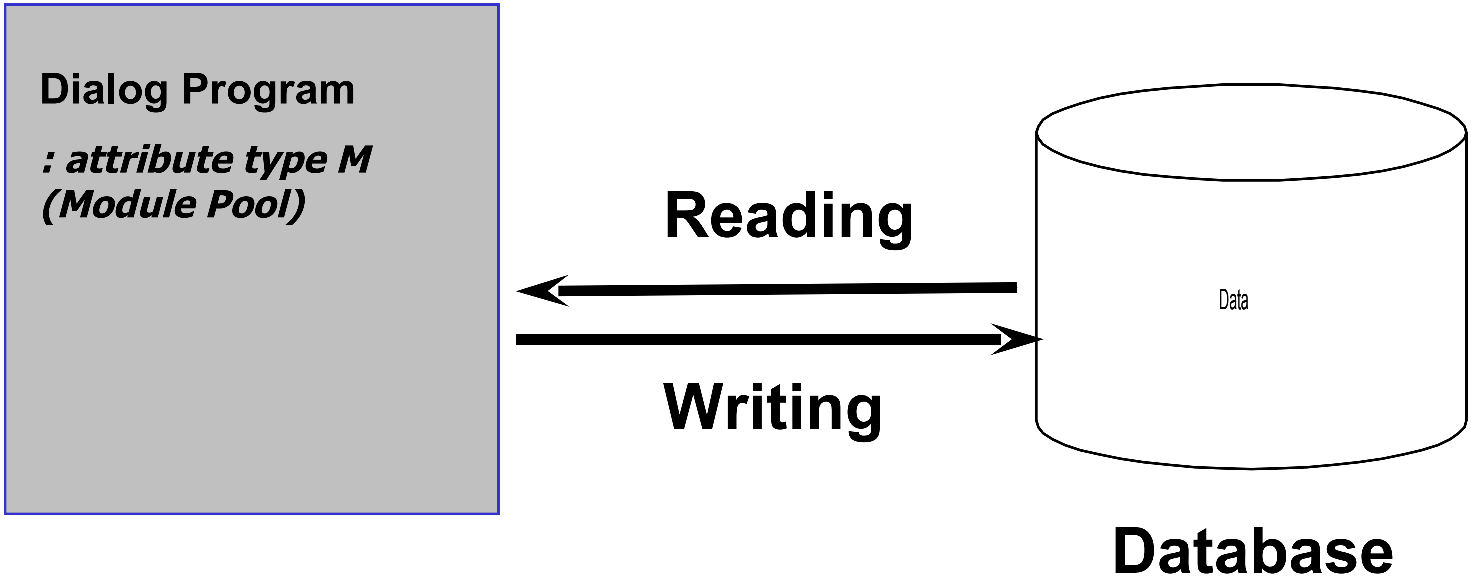


test

ID	No.	Flight Date	Airfare	Currency	Plane type
AA	17	19.02.2003	422.94	USD	747-400
AA	17	19.03.2003	422.94	USD	747-400
AA	17	16.04.2003	422.94	USD	747-400
AA	17	14.05.2003	422.94	USD	747-400
AA	17	11.06.2003	422.94	USD	747-400
AA	17	09.07.2003	422.94	USD	747-400
AA	17	06.08.2003	422.94	USD	747-400

1. **Report Listing**
2. **Drill-down Report**
3. **Control-break Report**
4. **ALV Report**

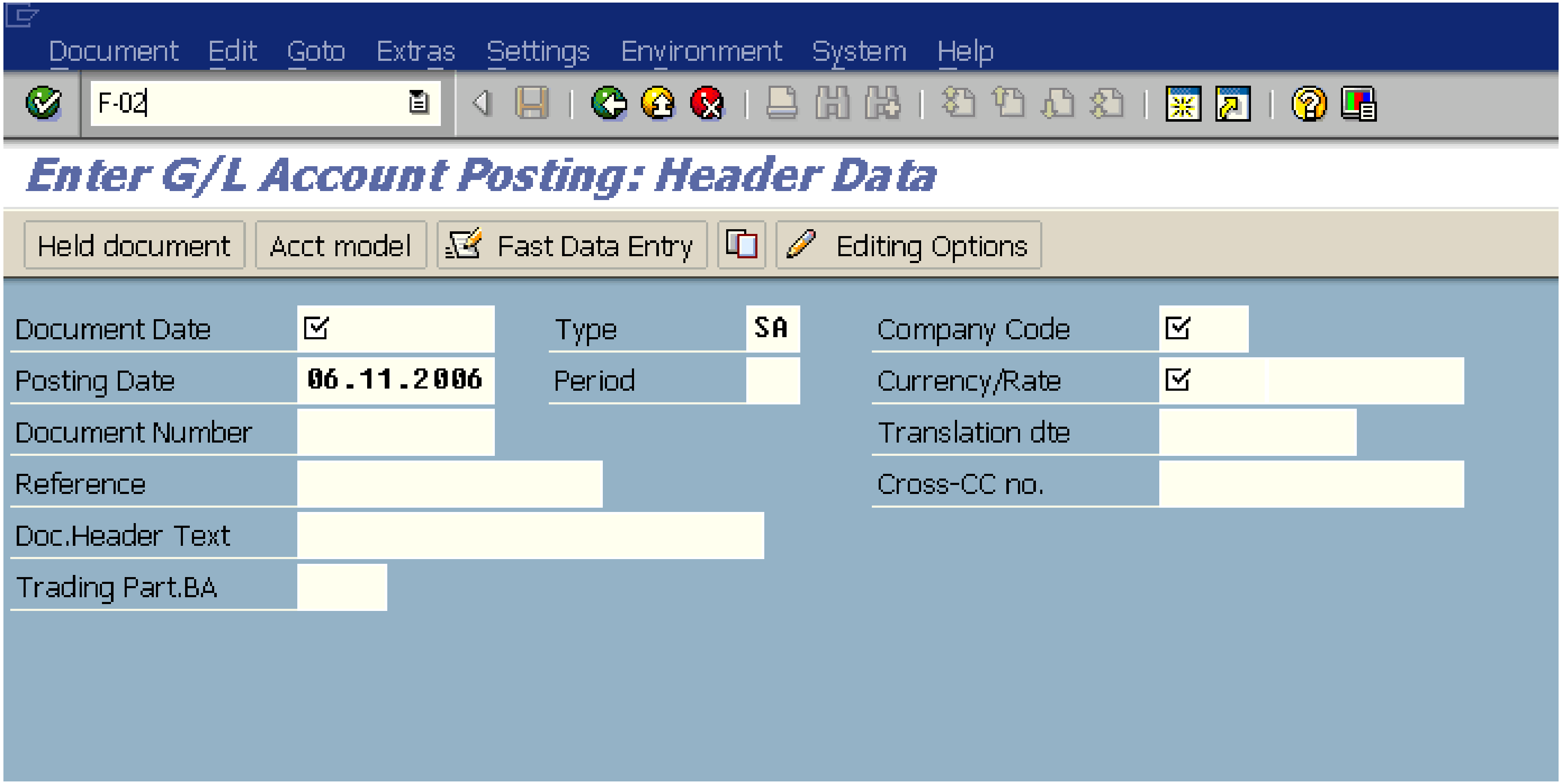
ABAP Program : Dialog Program


Dialog Program

: attribute type M (Module Pool)

- ❑ ***Reading and changing data***

Dialog Program : Transaction



F-02

Enter G/L Account Posting: Header Data

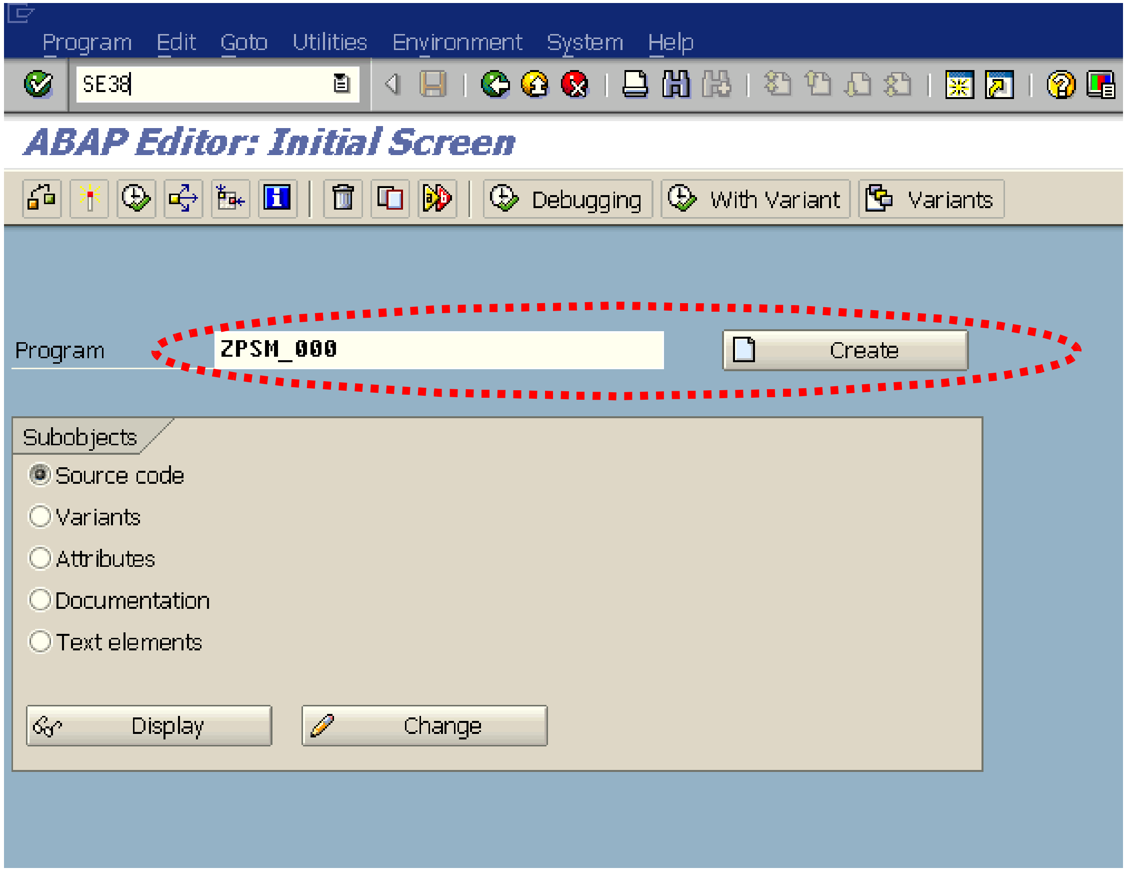
Document Date	☑	Type	SA	Company Code	☑
Posting Date	06.11.2006	Period		Currency/Rate	☑
Document Number				Translation dte	
Reference				Cross-CC no.	
Doc.Header Text					
Trading Part.BA					

ABAP Programming

How to create ABAP program

Transaction Code : SE38

Transaction : SE38



SE38

ABAP Editor: Initial Screen

Program ZPSM_000

Subobjects

● Source code

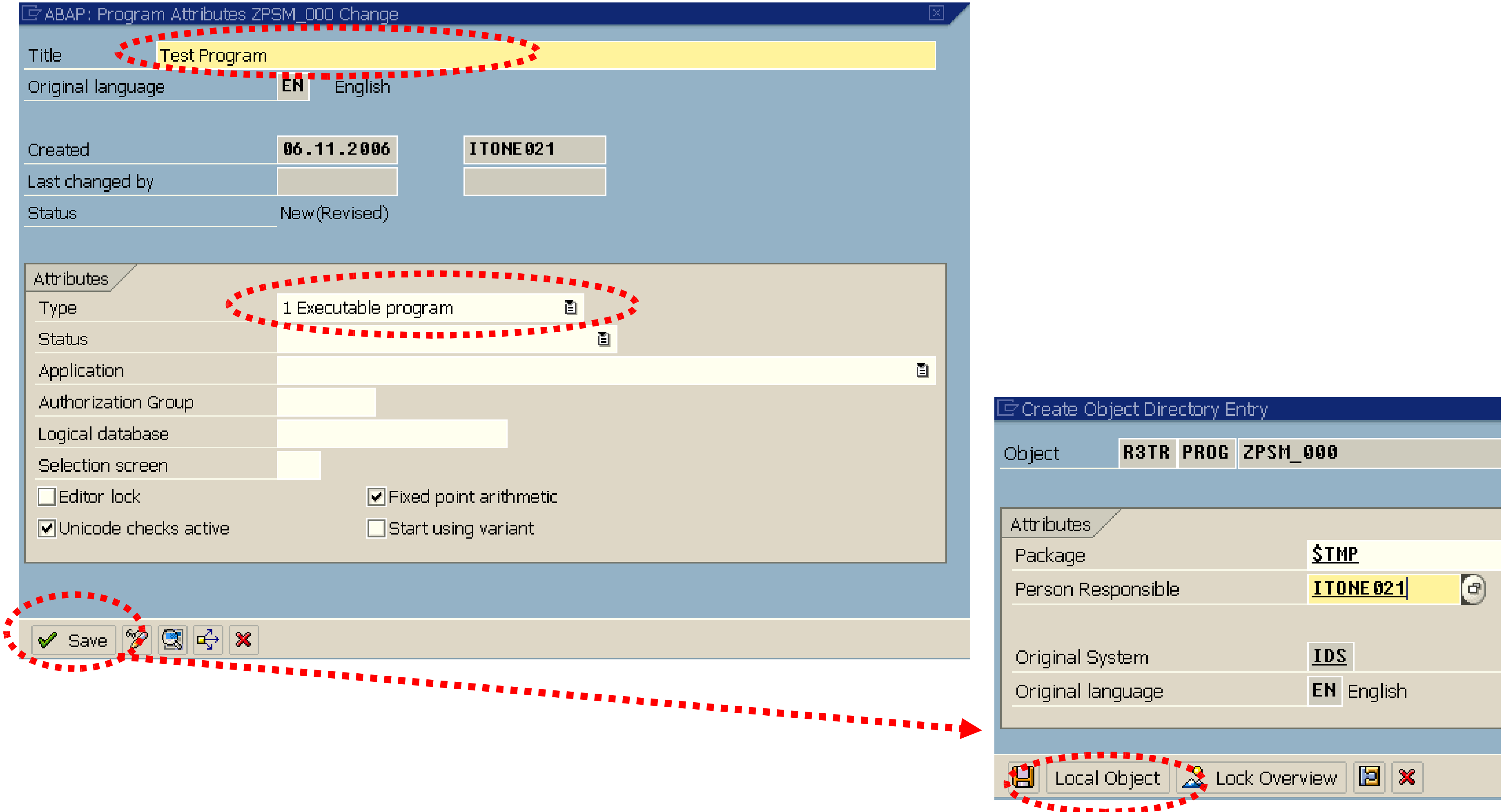
○ Variants

○ Attributes

○ Documentation

○ Text elements

Program Attribute



ABAP: Program Attributes ZPSM_000 Change

Title: Test Program

Original language: EN English

Created: 06.11.2006 ITONE021

Last changed by:

Status: New(Revised)

Attributes

Type: 1 Executable program

Status:

Application:

Authorization Group:

Logical database:

Selection screen:

□ Editor lock

☑ Fixed point arithmetic

☑ Unicode checks active

□ Start using variant

Create Object Directory Entry

Object: R3TR PROG ZPSM_000

Attributes

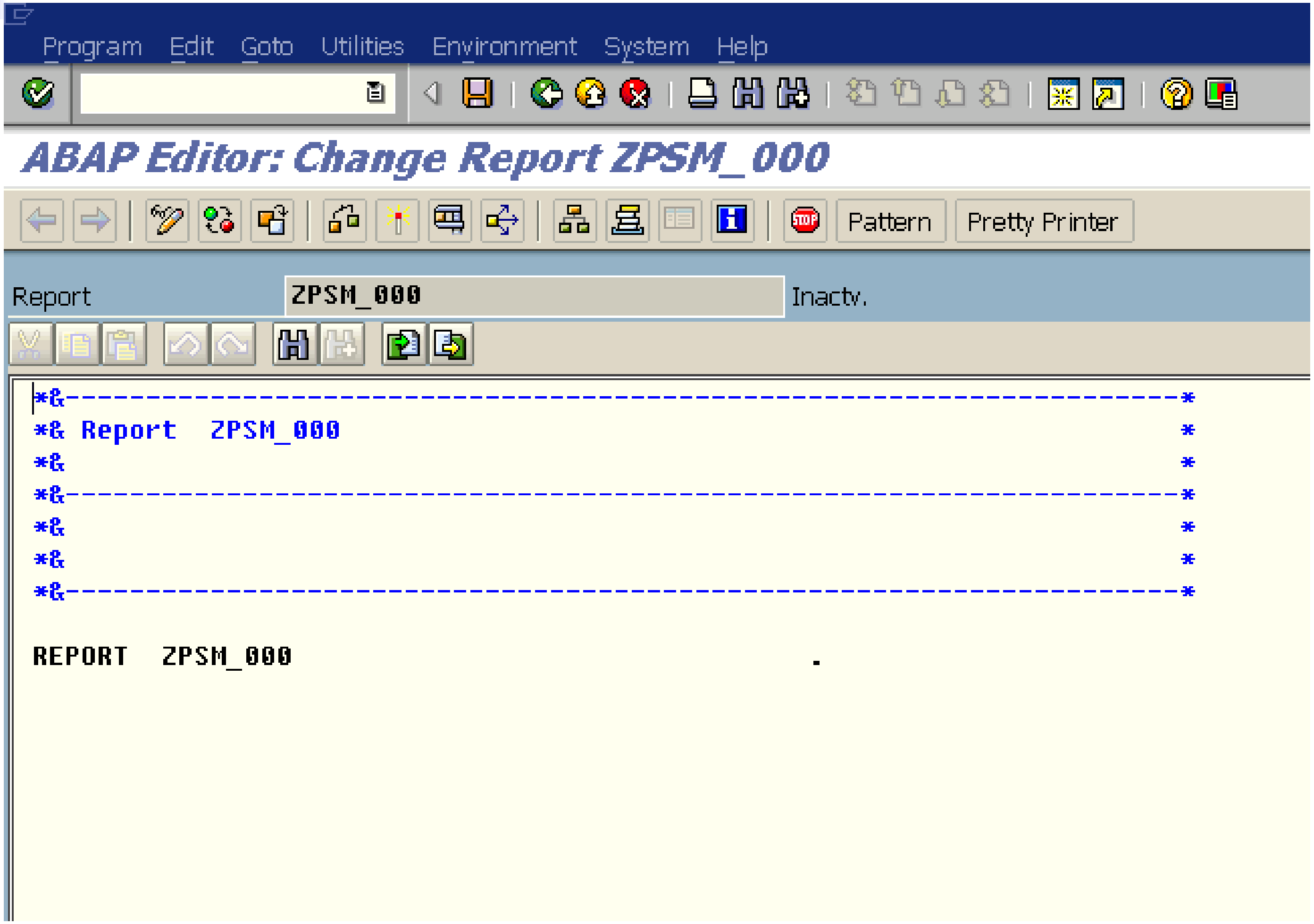
Package: $TMP

Person Responsible: ITONE021

Original System: IDS

Original language: EN English

ABAP Editor



ABAP Editor: Change Report ZPSM_000

Report ZPSM_000 Inactv.

```
*&---------------------------------------------------------------------*
*& Report  ZPSM_000                                                    *
*&                                                                     *
*&---------------------------------------------------------------------*
*&                                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*

REPORT  ZPSM_000                                       .
```

The Structure of the Language

- Each statement must end with a period

```
DATA  tmp  TYPE  I.
WRITE ‘Hello World’.  WRITE  ‘OK’.
```

Literal

```
DATA  tmp  TYPE  I.
WRITE ‘Hello World’.
WRITE ’10’.
MOVE  9 TO tmp.
```

Text Literal

Text Literal

Numeric Literal

Chained Statements

- Successive statements that have the same string segment can be combined to form a single chained statement
- To do so, you specify the identical starting segment once and conclude it with a colon (:), the remaining segments are then listed, separated by commas (,) and concluded with a period (.)
- At runtime, a chained statement is treated like an equivalent sequence of individual ABAP statements

Chained Statements

```
WRITE ‘Hello World’.
WRITE  ‘OK’.
                    =
WRITE: ‘Hello World’, ‘OK’.
```

```
DATA  tmp1  TYPE  I.
DATA  tmp2  TYPE  C.
                    =
DATA:  tmp1  TYPE  I,
             tmp2  TYPE  C.
```

Chained Statement

```
MOVE  sy-subrc  TO  tmp1.
MOVE  sy-subrc  TO  tmp2.
MOVE  sy-subrc  TO  tmp3.
               =
MOVE  sy-subrc  TO:  tmp1,
                     tmp2,
                     tmp3.
```

Chained Statement

```
PERFORM  cal_1  USING  a1  a2.
PERFORM  cal_1  USING  a3  a4.
                  =
PERFORM  cal_1  USING:  a1  a2,
                             a3  a4.
```

Comments

```
* This is full line comment
WRITE ‘Hello World’.    “ Write data (partial line comment)
WRITE ‘Test’.
```

ABAP Command : Case Sensitivity

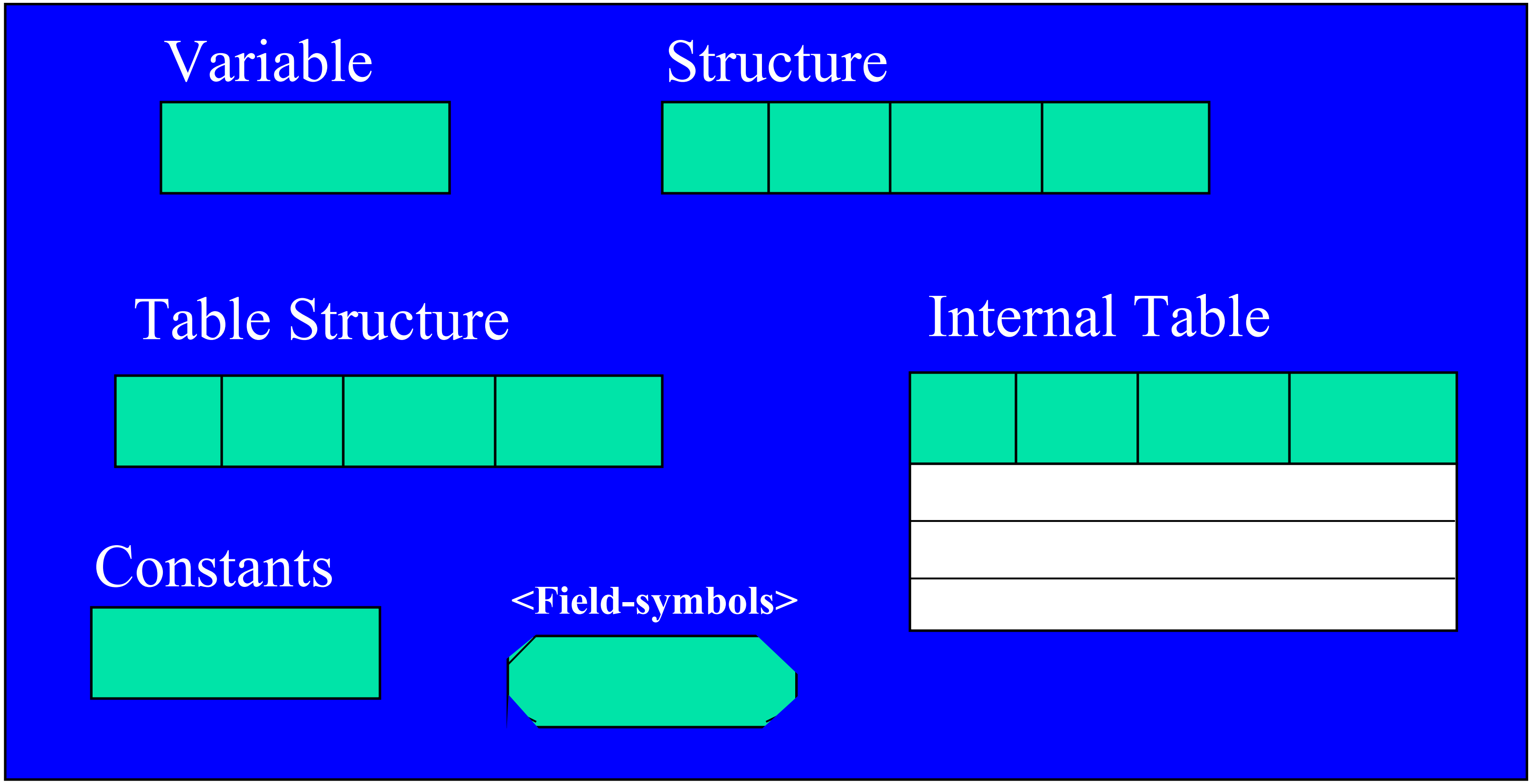
- **ABAP command is not case sensitive**

```
WRITE ‘Hello World’.
WriTe  ‘Hello World’.
wRiTE  ‘Hello World’.
```

Data Objects in ABAP

Data Objects in ABAP

Memory Space



Variable

Structure

Table Structure

Internal Table

Constants

<Field-symbols>

Variable

Variable

- Variables can be declared at any point in a program
- Variables can be up to 30 characters in length

```
REPORT ZTEST.
DATA firstname TYPE STRING.
firstname = ‘John’.
```

Predefined ABAP Data Types

Type	Description	Initial Value	Length
C	Character	Space	1 – 65535
D	Date	‘00000000’	8 characters
F	Floating Point	0.0	8 bytes
I	Integer	0	4 bytes
N	Numeric Text	‘0’	1 – 65535
P	Packed Decimal	0	1 – 16 bytes
T	Time	‘000000’	6 characters
X	Hexadecimal	’00’	1 – 65535
String	Variable-length	Space	Variable
xstring	Variable-length Hexadecimal	Blank string	Variable

Defining Variable with DATA Statement

* Syntax

DATA var[(*length*)] [Type *type*] [Decimals *number*].

DATA var LIKE Table-Field [VALUE *initial value*].

Defining Variable with DATA Statement

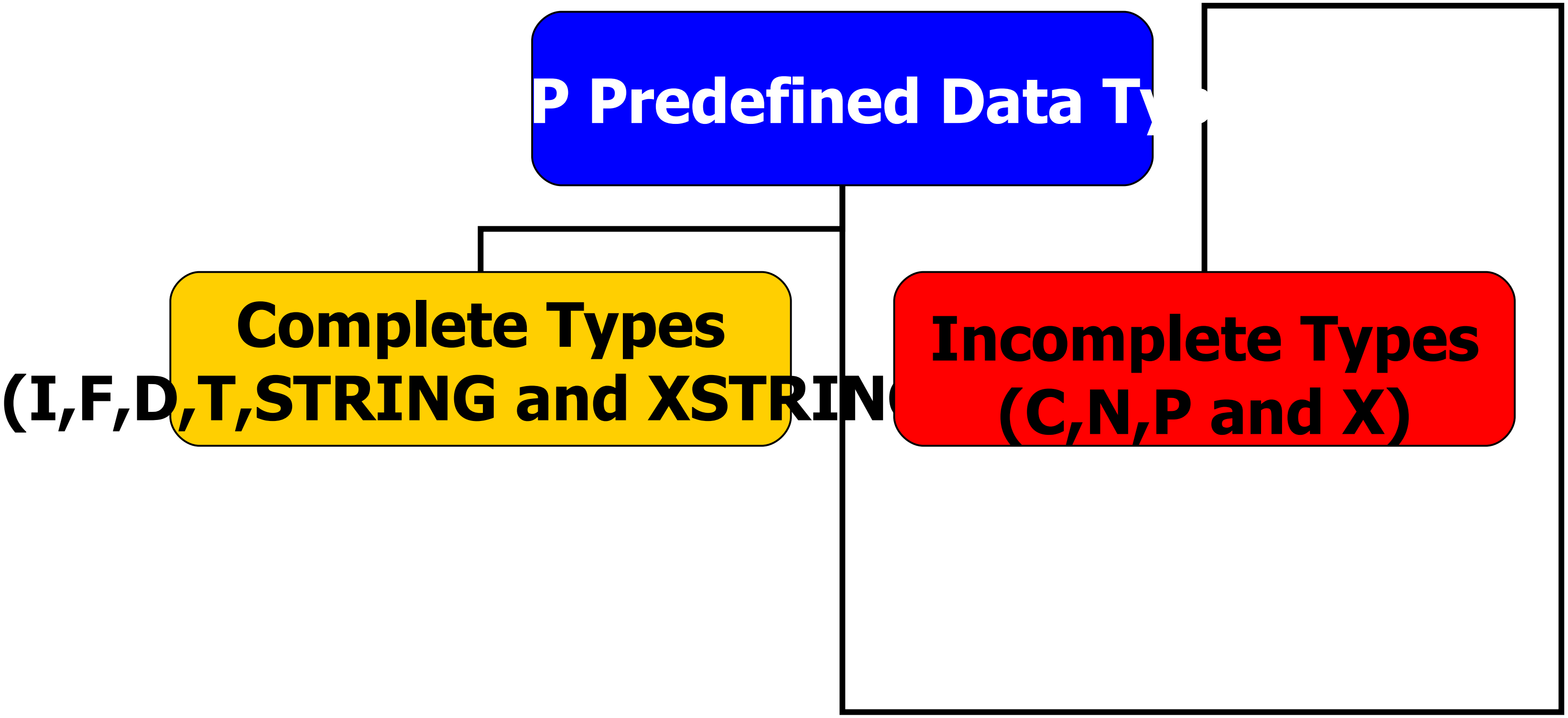
```
* Data Declaration
DATA: tmp(10)  TYPE  C,
         tmp1       TYPE  I,
         tmp2(8)   TYPE P  DECIMALS  2  VALUE ‘1.50’.
DATA: tmp3(5)   TYPE  N,
         tmp4.
```

Defining Variable with DATA Statement

```
* Data Declaration
DATA  customerno  LIKE  customers-id.
DATA  matnr   LIKE  mara-matnr.
```

```
DATA customerno TYPE customers-id.
DATA matnr TYPE mara-matnr.
```

ABAP Predefined Data Types



P Predefined Data Ty

Complete Types
(I,F,D,T,STRING and XSTRIN

Incomplete Types
(C,N,P and X)

Variable

- Data Type C,N and X length between 1 – 65535 (Default 1)

```
DATA tmp(10) TYPE C.
```

- Data Type P length between 1 – 16 (Default 8) and decimals length between 0 – 31

```
DATA tmp(5) TYPE P DECIMALS 2.
```

- Data Type I value between – 2^{31} to 2^{31} – 1

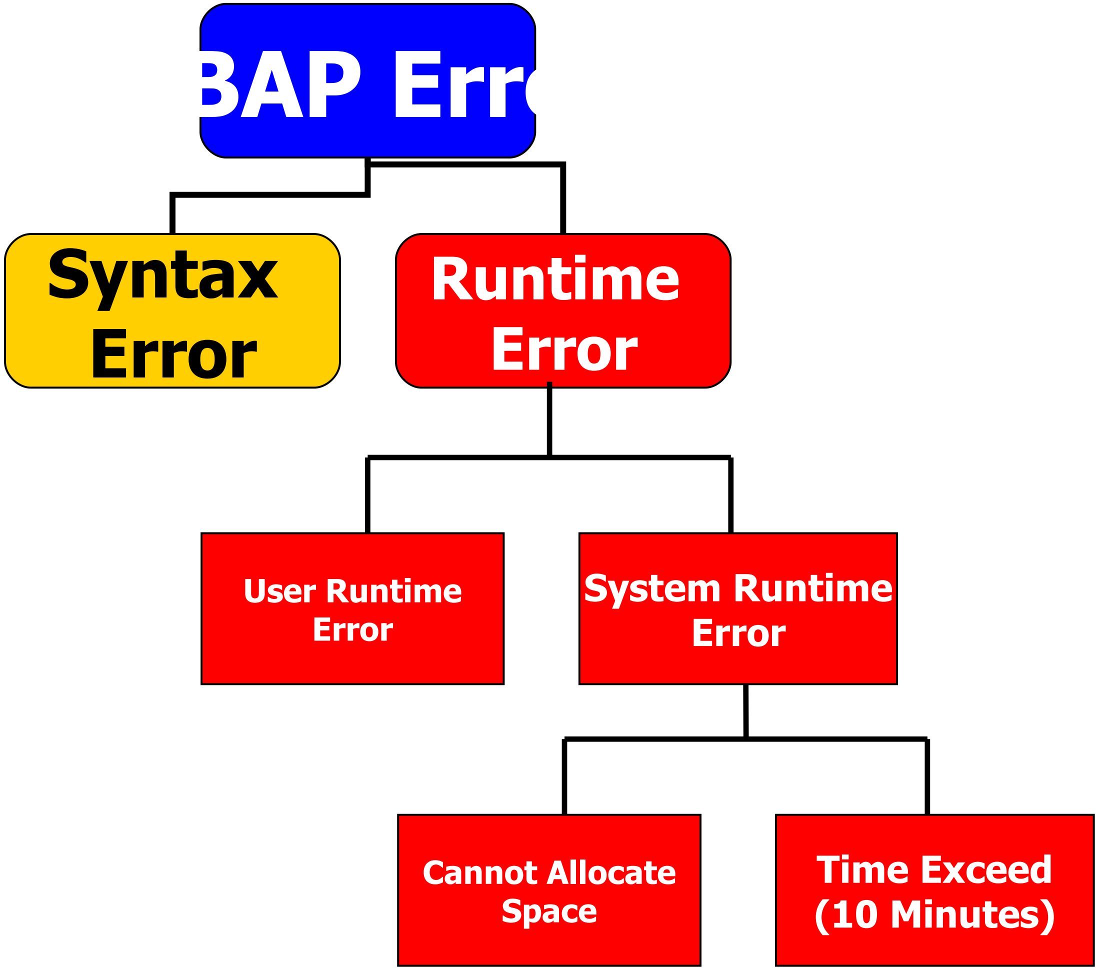
  or –2,147,483,648 to 2,147,483,647

```
DATA tmp TYPE I.
tmp = 1000000.
```

Data type N

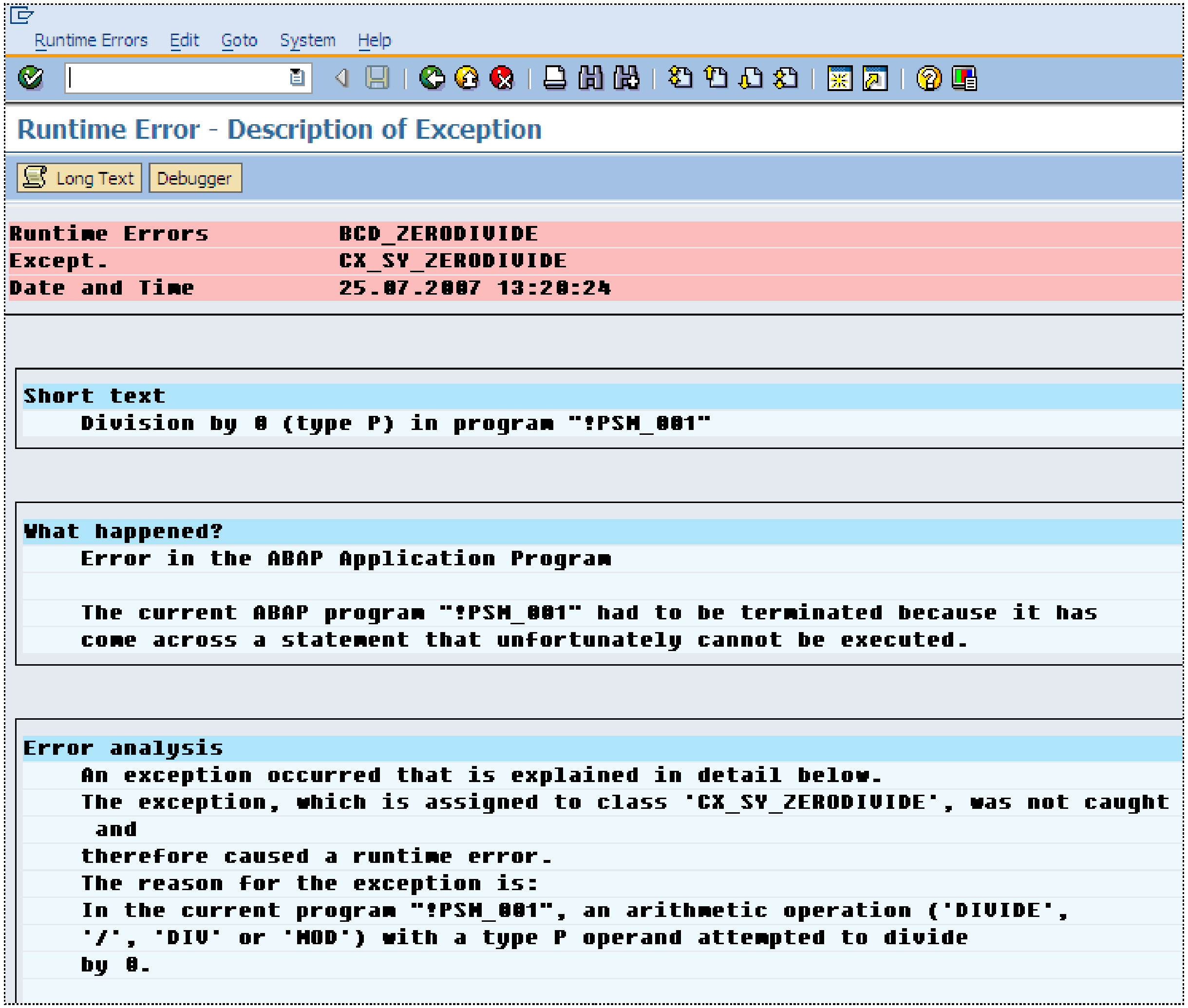
```
data tmp(5) type N.
tmp = ‘Xca9yy23K6’.
```

ABAP Error


BAP Err
Syntax Error
Runtime Error
User Runtime Error
System Runtime Error
Cannot Allocate Space
Time Exceed (10 Minutes)

User Runtime Error

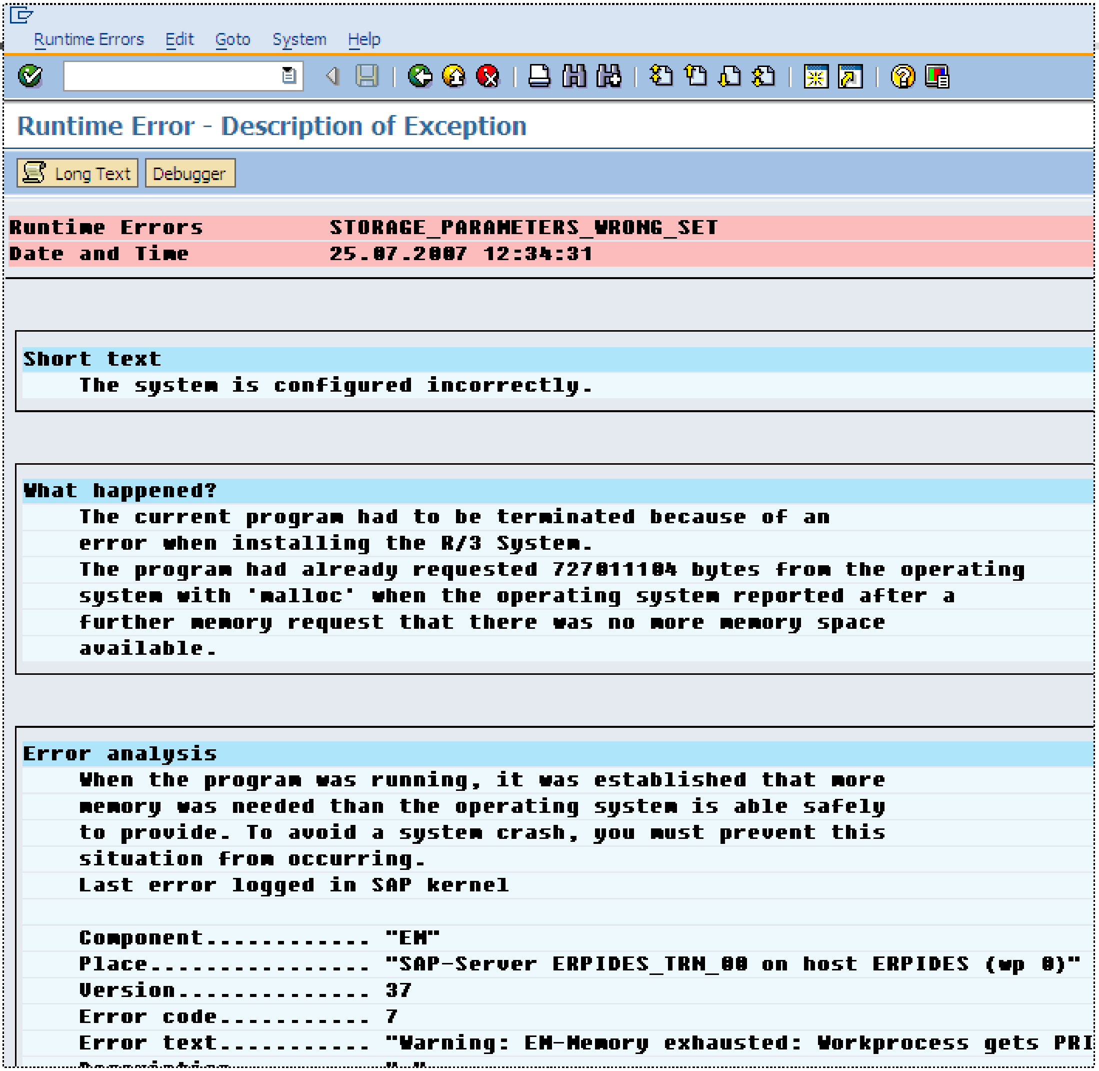
```
DATA result TYPE i.
result = 10 / 0.
```



Runtime Error - Description of Exception

```
Runtime Errors          BCD_ZERODIVIDE
Except.                 CX_SY_ZERODIVIDE
Date and Time           25.07.2007 13:20:24

Short text
    Division by 0 (type P) in program "!PSM_001"

What happened?
    Error in the ABAP Application Program

    The current ABAP program "!PSM_001" had to be terminated because it has
    come across a statement that unfortunately cannot be executed.

Error analysis
    An exception occurred that is explained in detail below.
    The exception, which is assigned to class 'CX_SY_ZERODIVIDE', was not caught
     and
    therefore caused a runtime error.
    The reason for the exception is:
    In the current program "!PSM_001", an arithmetic operation ('DIVIDE',
    '/', 'DIV' or 'MOD') with a type P operand attempted to divide
    by 0.
```

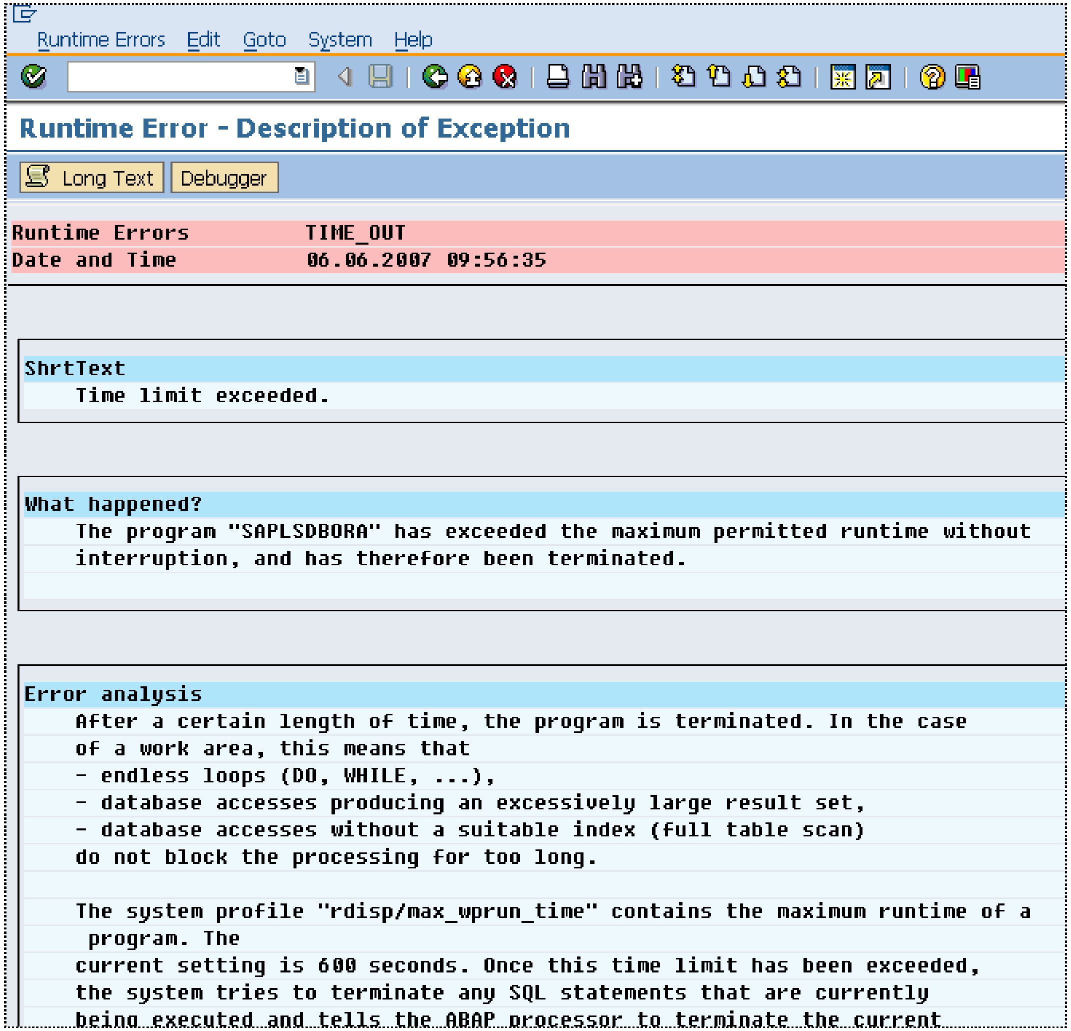
System Runtime Error : Space Allocation



Runtime Error - Description of Exception

```
Runtime Errors          STORAGE_PARAMETERS_WRONG_SET
Date and Time           25.07.2007 12:34:31


Short text
    The system is configured incorrectly.


What happened?
    The current program had to be terminated because of an
    error when installing the R/3 System.
    The program had already requested 727011104 bytes from the operating
    system with 'malloc' when the operating system reported after a
    further memory request that there was no more memory space
    available.


Error analysis
    When the program was running, it was established that more
    memory was needed than the operating system is able safely
    to provide. To avoid a system crash, you must prevent this
    situation from occurring.
    Last error logged in SAP kernel

    Component............ "EM"
    Place................ "SAP-Server ERPIDES_TRN_00 on host ERPIDES (wp 0)"
    Version.............. 37
    Error code........... 7
    Error text........... "Warning: EM-Memory exhausted: Workprocess gets PRI
```

System Runtime Error : Time Exceed



Runtime Error - Description of Exception

```
Runtime Errors          TIME_OUT
Date and Time           06.06.2007 09:56:35

ShrtText
    Time limit exceeded.

What happened?
    The program "SAPLSDBORA" has exceeded the maximum permitted runtime without
    interruption, and has therefore been terminated.

Error analysis
    After a certain length of time, the program is terminated. In the case
    of a work area, this means that
    - endless loops (DO, WHILE, ...),
    - database accesses producing an excessively large result set,
    - database accesses without a suitable index (full table scan)
    do not block the processing for too long.

    The system profile "rdisp/max_wprun_time" contains the maximum runtime of a
     program. The
    current setting is 600 seconds. Once this time limit has been exceeded,
    the system tries to terminate any SQL statements that are currently
    being executed and tells the ABAP processor to terminate the current
```

Non-elementary Type

```
* Data Declaration
TYPES  tname(30)   TYPE  c.
DATA:  customer_name  TYPE  tname,
        firstname TYPE tname.
```

Value Assignment

```
* Value  assignment
DATA:  name1(30),
          first_num  TYPE  I,
          next_num      TYPE  I.
MOVE  'XXXX'  TO  name1.
MOVE   5  TO  first_num.
COMPUTE  next_num  =  first_num  +  5.
name1  =  'SAP'.
ADD  1  TO  next_num.
```

Value Assignment

```
* Value  assignment
DATA:  tmp1 TYPE  i,
          tmp2 TYPE  i.
tmp1 = tmp2 = 10.
```

งการให้สร้างตัวแปรชื่อ firstname และ lastname โดยให้ค่าชื่อของ
แปร firstname และนามสกุลของคุณให้กับตัวแปร lastname พร้อม
ข้อมูล firstname กับ lastname ออกมาที่หน้าจอ

ABAP Practice

Structure

Structure

```
* Syntax
DATA BEGIN OF <structure name>.
DATA field1.
DATA field2.
    ...
    ...
DATA END OF <structure name>.
```

Structure

wa

id	name	city
00000000		

```
* Syntax
DATA  BEGIN  OF  wa.
DATA  id  LIKE  customers-id.
DATA  name LIKE customers-name.
DATA  city LIKE customers-city.
DATA  END  OF   wa.
MOVE  9  TO  wa-id.
WRITE wa-id.
```

Defining Structure (Include Structure)

```
* Include Structure
DATA  BEGIN  OF  wa.
    INCLUDE STRUCTURE  customers.
DATA  tel(7).
DATA  END  OF  wa.
```

Defining Structure

```
* LIKE option
DATA  wa LIKE customers.
wa-id = 1.
wa-name = ‘John’.
WRITE:  wa-id, wa-name.
```

รให้สร้าง Structure ชื่อ myname โดยมีฟิลด์ firstname และ lastn
ค่าชื่อของคุณกับฟิลด์ firstname และนามสกุลของคุณให้กับฟิลด์ la
ทั้งแสดงค่าข้อมูลของ Structure ที่ชื่อ myname ทั้งฟิลด์ firstname
me ออกมาที่หน้าจอ

ABAP Practice

Constants

Constants

```
* Constant variable
CONSTANTS max_no TYPE I VALUE 999.
DATA  counter TYPE I VALUE max_no.
WRITE:  max_no, counter.
```

Constants Using Example

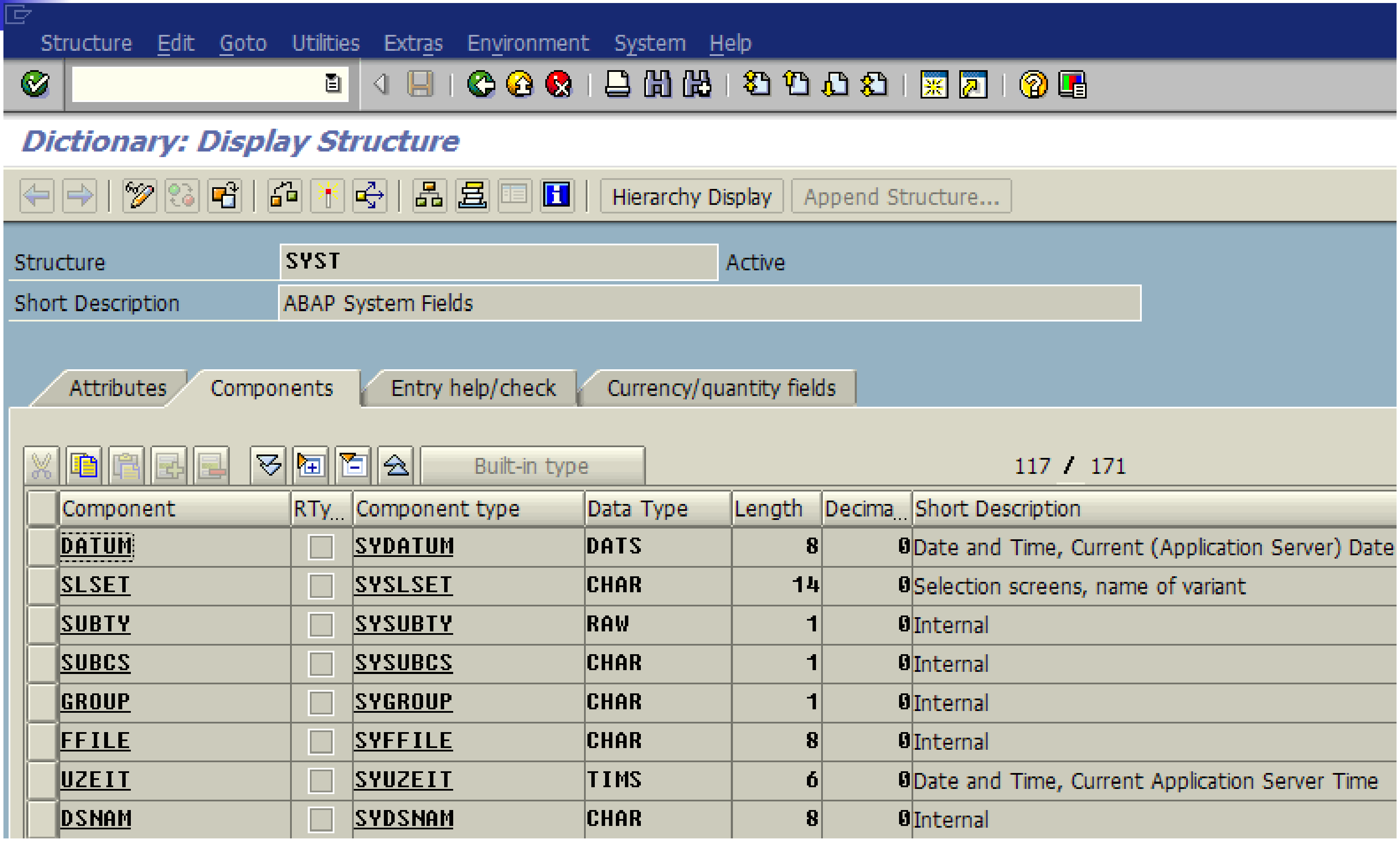
```
* Constant variable
CONSTANTS  ctext(11)  TYPE  C  VALUE  ‘Hello World’.
WRITE  ctext.
WRITE  ctext.
WRITE  ctext.
WRITE  ctext.
WRITE  ctext.
```

System Fields

- The system fields (structure syst) are filled by the runtime environment. You can use them to query the system status in an ABAP program
- You should access them only for reading
  - sy-datum = Current date of application server
  - sy-uzeit = Current time of application server
  - sy-datlo = Current date of SAP GUI
  - sy-timlo = Current time of SAP GUI
  - sy-mandt = Current client logon
  - **sy-subrc** = Return value of ABAP statement

syst-

ABAP System Fields : Structure SYST (SE11)



Dictionary: Display Structure

Structure	SYST	Active
Short Description	ABAP System Fields	

117 / 171

Component	RTy...	Component type	Data Type	Length	Decima...	Short Description
DATUM	□	SYDATUM	DATS	8	0	Date and Time, Current (Application Server) Date
SLSET	□	SYSLSET	CHAR	14	0	Selection screens, name of variant
SUBTY	□	SYSUBTY	RAW	1	0	Internal
SUBCS	□	SYSUBCS	CHAR	1	0	Internal
GROUP	□	SYGROUP	CHAR	1	0	Internal
FFILE	□	SYFFILE	CHAR	8	0	Internal
UZEIT	□	SYUZEIT	TIMS	6	0	Date and Time, Current Application Server Time
DSNAM	□	SYDSNAM	CHAR	8	0	Internal

DATE

```
* Fixed Length  8
* Include Representation  ‘YYYYMMDD’
DATA  today  TYPE  D.
today  =  sy-datum.
WRITE  today.
today  =  ‘19991231’.
WRITE  today.
```

TIME

```
* Fixed Length  6
* Format  ‘HHMMSS’
DATA  times  TYPE  T.
times  =  sy-uzeit.
WRITE  times.
```

```
HHMMSS
```

MOVE Statement

```
DATA  wa LIKE customers.
DATA  vender  LIKE  customers.
wa-id  =  ‘1234’.
wa-name  =  ‘Test#1’.
MOVE  wa  TO  vender.“vender = wa.
WRITE:  wa-id, vender-name.
```

MOVE-CORRESPONDING Statement

```
DATA: begin of wa1,
          f1,f2,f4,
          end of wa1.
DATA: begin of wa2,
          f2,f1,f3,
          end of wa2.
…
MOVE-CORRESPONDING  wa1  TO  wa2.
WRITE:  wa1-f1,wa2-f1 .
```

Field-symbols

Field-symbols

```
Data:  name(4)  Value  ‘Test’,
         num    Type  I  Value  10,
         today  Type  D  Value ‘19980429’.
Field-symbols  <temp>.
Assign  name  To  <temp>.
Write  <temp>.
Assign  num  To  <temp>.
Write  <temp>.
Assign  today  To  <temp>.
Write  <temp>.
```

Field-symbols : UNASSIGN

```
data:  name(4)  Value  ‘Test’,
field-symbols  <temp>.
assign  name  To  <temp>.
write  <temp>.
unassign  <temp>.
```

CLEAR Statement

atement sets a field to an initial value appropriate fo

CLEAR <*data object*>.

Example:

```
DATA tmp type i value 9.
tmp = 10.
CLEAR tmp.
```

CLEAR Structure

```
DATA wa like customers.
…
CLEAR wa.
```

ABAP Report : Program Structure

```
Report ztest.
*Data objects declaration
data ...
data begin of ...
*Program Logic(Data objects processing)
…
write ….
```

ABAP Practice