Programming Using T24 APIs

TEMENOS EDUCATION CENTRE

TEMENOS
The Banking Software Company

Objectives

After completing this learning unit/course, you will be able to:

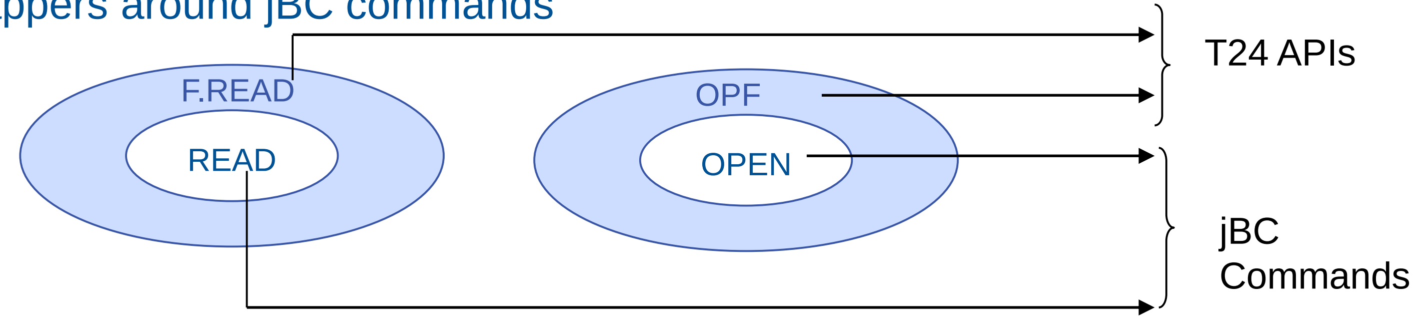
- Create subroutines using T24 APIs such as
  - OPF
  - F.READ
  - F.READU
  - F.WRITE
  - JOURNAL.UPDATE
  - F.RELEASE
  - EB.READLIST
  - F.DELETE

Task

- Display the currency and category of account 11967

T24 APIs and subroutines

- T24 APIs
  - API stands for Application Programming Interface
  - Wrappers around jBC commands



- Subroutines
  - Executed from within T24 (Not from the jsh prompt)
  - Can make use of T24 APIs.

```
* Comments
SUBROUTINE Subroutinename
           Actual statements
           Actual statements
RETURN
END
```

Algorithm again!!!

Algorithm to display the CATEGORY and CURRENCY of ACCOUNT 10693.

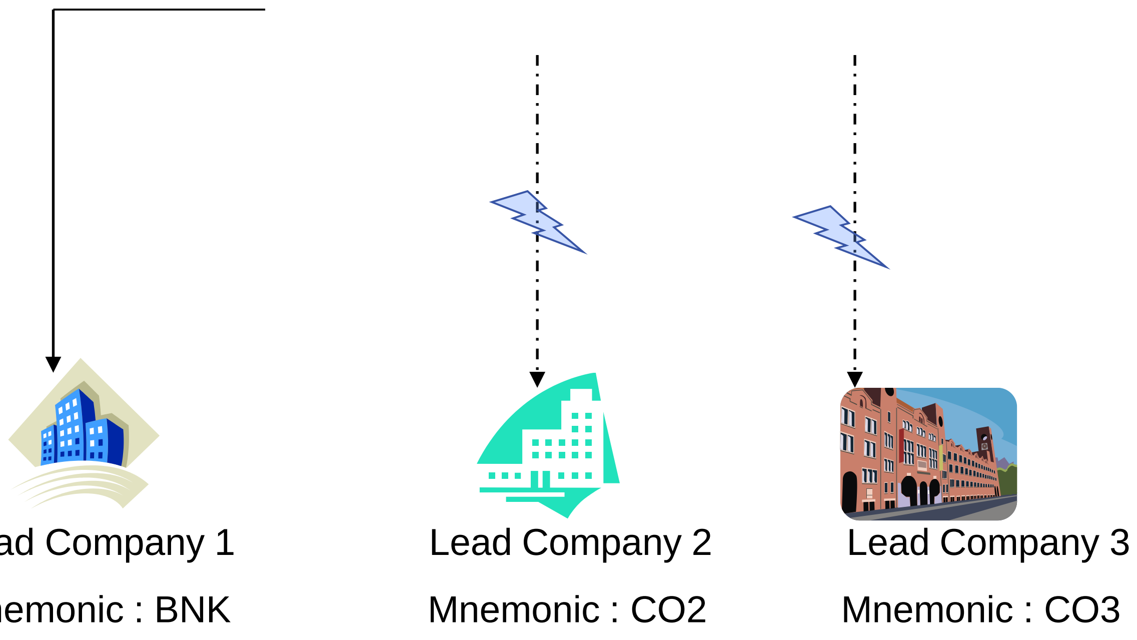
Subroutine to be created to achieve the task.

Action to be performed	jBASE command to be used
Open the ACCOUNT file	~~OPEN~~ OPF
Read the ACCOUNT record	~~READ~~ F.READ
Extract category and currency	?? You will learn as you proceed
Display category and currency	Use CRT to display

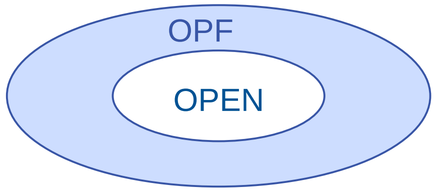
Disadvantages of using the OPEN command

- Disadvantages of using the OPEN command
  - OPEN ‘FBNK.ACCOUNT’ TO F.ACCOUNT THEN … ELSE…

    File names get hard coded
  - Code does not become portable in a multi company environment



T24 API - OPF



- Stands for Open File
- Syntax

```
CALL OPF(Parameter1,Parameter2)
```

- Example

```
FN.ACC = 'F.ACCOUNT'   * File Name
F.ACC = ''  * File Path

CALL OPF(FN.ACC,F.ACC) *Open the file
```

Program Variables and Common Variables

- Program variables
    - The scope of a variable used within a program is limited to the program. Meaning, the variable will loose its value when the program terminates
    - Any variable that is used within a program

- Common variables
    - Need to be defined as common
    - Values of these variables are lost only when the session is terminated

Hint to help you answer the question – I_COMMON

- What is I_COMMON?
  - It is a file under T24.BP
  - Contains the definition for most of the common variables used in
  - When do these variables get populated with values?

    Some get values when a user signs on

    Example :

    ID.COMPANY (ID of the user's currently signed on company)

    R.USER (Currently signed on user's record)

    R.COMPANY (Dimension array which holds the current company record)

    Applications populate data on to some variables

    Example :

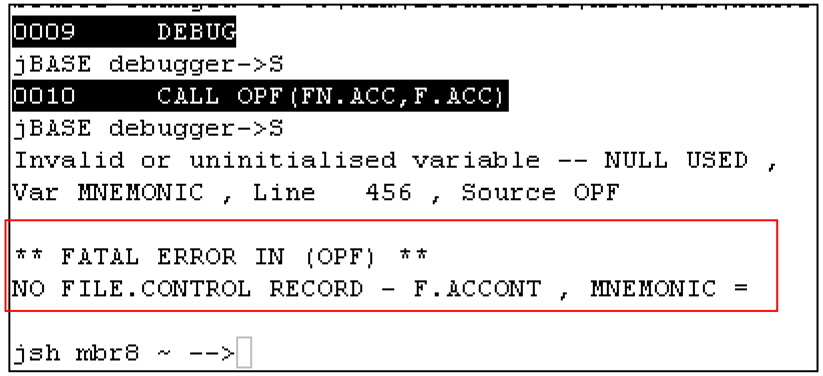
    ID.NEW (ID of the currently opened record)

    R.NEW (Contents of the currently opened record)

Error Handling - OPF

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT  I_EQUATE
FN.ACC='F.ACOUNT'
F.ACC=''
DEBUG
CALL OPF(FN.ACC,F.ACC)
CRT ETEXT
RETURN
END
```

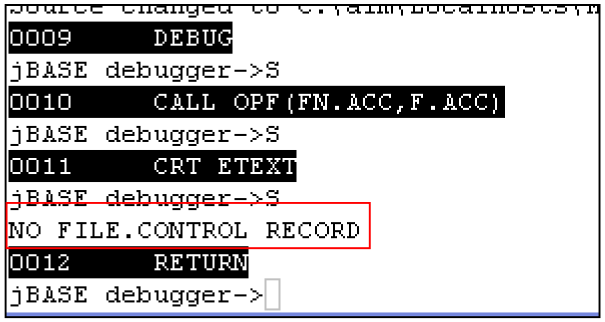
Note the Error



```
0009       DEBUG
jBASE debugger->S
0010       CALL OPF(FN.ACC,F.ACC)
jBASE debugger->S
Invalid or uninitialised variable -- NULL USED ,
Var MNEMONIC , Line   456 , Source OPF

** FATAL ERROR IN (OPF) **
NO FILE.CONTROL RECORD - F.ACCONT , MNEMONIC =

jsh mbr8 ~ -->
```

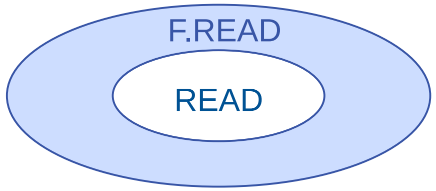
```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT  I_EQUATE
FN.ACC='F.ACOUNT'
FN.ACC<2>="NO.FATAL.ERROR"
F.ACC=''
DEBUG
CALL OPF(FN.ACC,F.ACC)
CRT ETEXT
RETURN
END
```

Note the Error



```
0009       DEBUG
jBASE debugger->S
0010       CALL OPF(FN.ACC,F.ACC)
jBASE debugger->S
0011       CRT ETEXT
jBASE debugger->S
NO FILE.CONTROL RECORD
0012       RETURN
jBASE debugger->
```

T24 API – F.READ



- F.READ – Read a record from a hashed file
- Will read a record only if a FILE.CONTROL record is present for the file being read
- Syntax

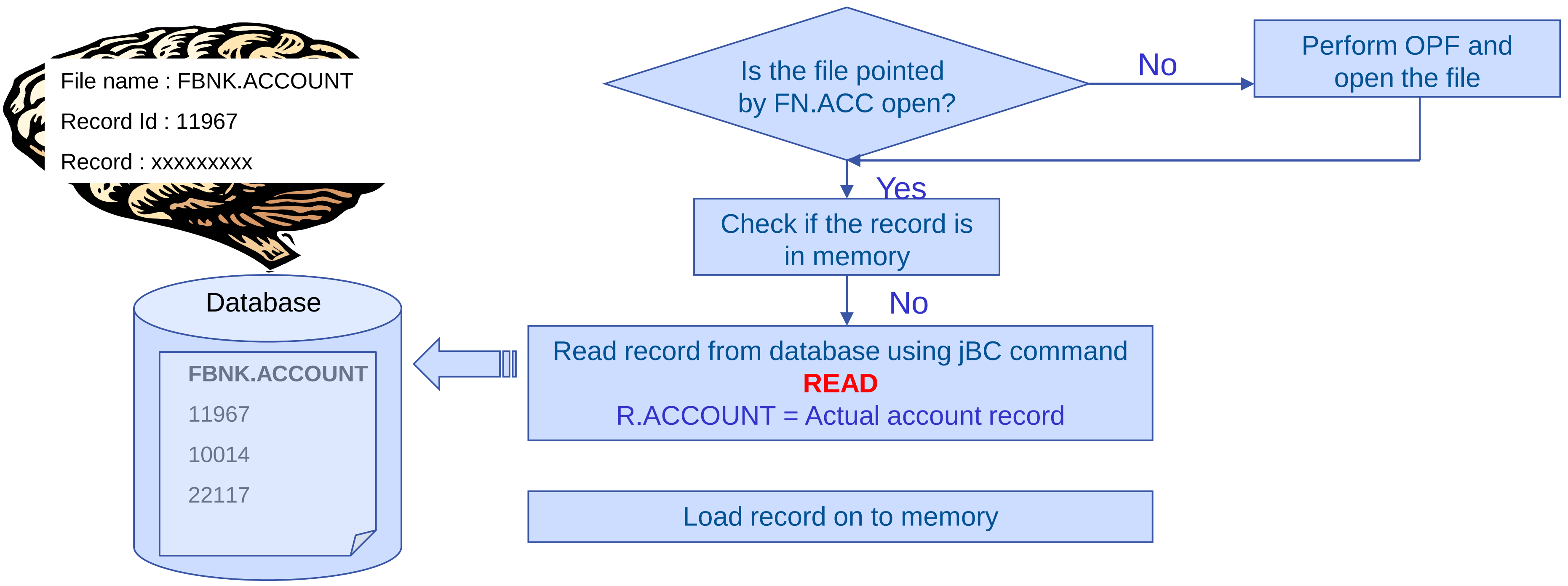
```
CALL F.READ(Filename,Key,Record,File path,Error variable)
```

- Example

```
CALL F.READ(FN.ACC,"11967",R.ACCOUNT,F.ACC,Y.ACC.ERR)
```

Internal working of F.READ

```
CALL F.READ(FN.ACC,"11967",R.ACCOUNT,F.ACC,Y.ACC.ERR)
```


File name : FBNK.ACCOUNT
Record Id : 11967
Record : xxxxxxxxx
Database
FBNK.ACCOUNT
11967
10014
22117
Is the file pointed by FN.ACC open?
No
Perform OPF and open the file
Yes
Check if the record is in memory
No
Read record from database using jBC command READ
R.ACCOUNT = Actual account record
Load record on to memory

Take a look at this sample routine

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT  I_EQUATE
FN.ACC='F.ACCOUNT'
F.ACC=''
Y.ACC.ID=11967
R.ACC=''
Y.ACC.ERR=''
CALL OPF(FN.ACC,F.ACC)
CALL F.READ(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR)
RETURN
END
```

Golden rules about F.READ

- Do you remember READU? READU helps lock and read a record where as READ only reads a record
- Since F.READ uses READ internally, it does not hold a lock on the record that is read
- When to use F.READ
  - When you wish to query data in a record, use F.READ
  - Example : You wish to check the category of an account
- When not to use F.READ
  - Never read a record using F.READ if you wish to update data that has been read
  - Example : You wish to update the balance in an account. In this case do not read the account record using F.READ

Working of F.READ with an example

TEMENOS

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT I_EQUATE
FN.ACC='F.ACCOUNT'
F.ACC=''
Y.ACC.ID=13935
R.ACC=''
Y.ACC.ERR=''
CALL OPF(FN.ACC,F.ACC)
CALL F.READ(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR)
CRT "RECORD DETAILS"
CRT R.ACC
RETURN
END
```

F.READ always returns the record in a dynamic array
Extract parts of a dynamic array using the following convention

Y.CURRENCY = R.ACC<8>
(or)
Y.CURRENCY = R.ACC<Name of the field CURRENCY in the ACCOUNT file>

Insert files

- There are some insert files that are common to entire T24 like
  - I_COMMON
  - I_EQUATE
- There are application specific insert files – one for each application
  - I_F.ACCOUNT
  - I_F.CUSTOMER
- All T24 core insert files will be available under GLOBUS.BP

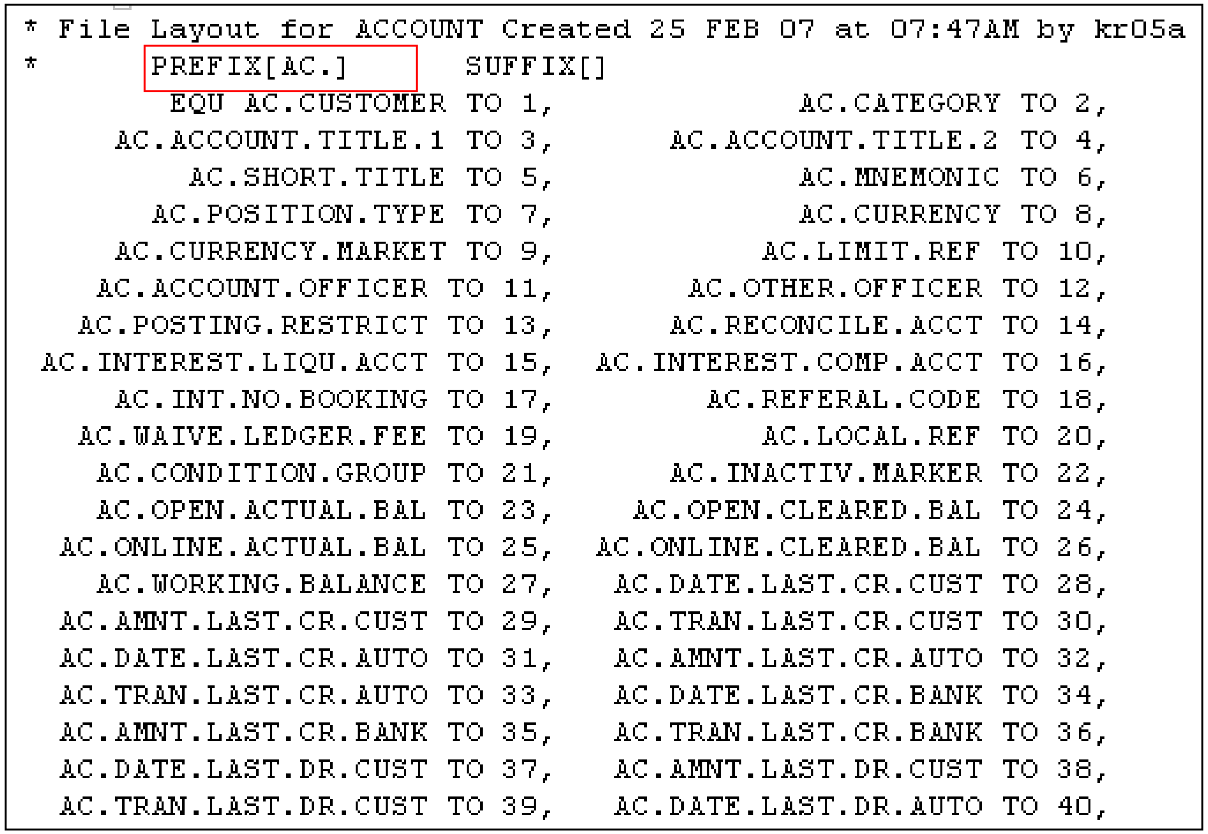
- Non application specific insert file naming convention

  I_<Insert file name>

- Application specific insert file naming convention

  I_F.<ApplicationName>

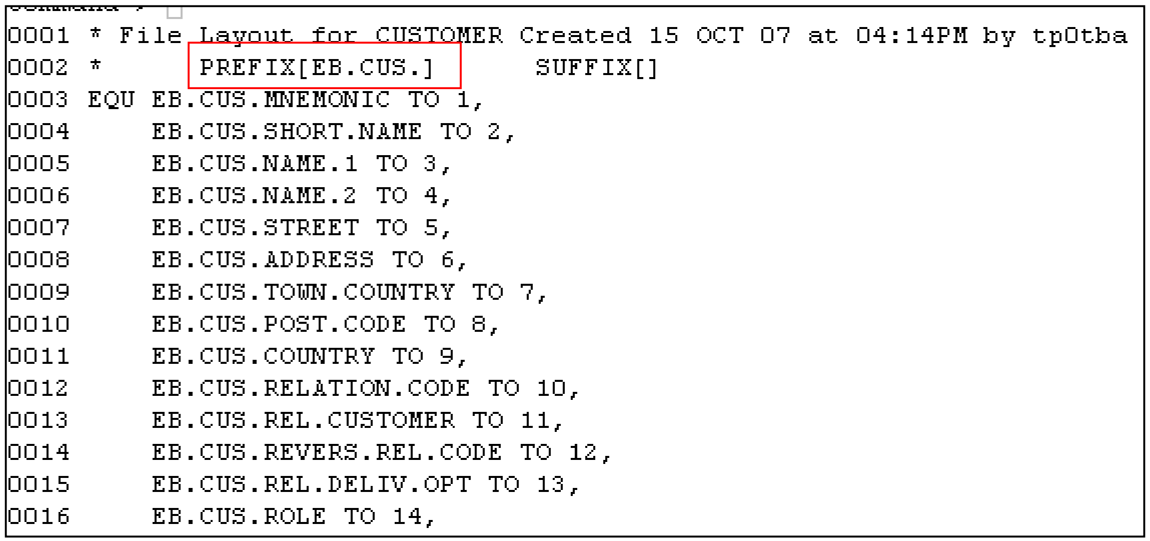
I_F.ACCOUNT



```
* File Layout for ACCOUNT Created 25 FEB 07 at 07:47AM by kr05a
*       PREFIX[AC.]         SUFFIX[]
          EQU AC.CUSTOMER TO 1,            AC.CATEGORY TO 2,
      AC.ACCOUNT.TITLE.1 TO 3,      AC.ACCOUNT.TITLE.2 TO 4,
          AC.SHORT.TITLE TO 5,             AC.MNEMONIC TO 6,
        AC.POSITION.TYPE TO 7,             AC.CURRENCY TO 8,
      AC.CURRENCY.MARKET TO 9,           AC.LIMIT.REF TO 10,
     AC.ACCOUNT.OFFICER TO 11,       AC.OTHER.OFFICER TO 12,
    AC.POSTING.RESTRICT TO 13,     AC.RECONCILE.ACCT TO 14,
  AC.INTEREST.LIQU.ACCT TO 15, AC.INTEREST.COMP.ACCT TO 16,
      AC.INT.NO.BOOKING TO 17,      AC.REFERAL.CODE TO 18,
    AC.WAIVE.LEDGER.FEE TO 19,         AC.LOCAL.REF TO 20,
     AC.CONDITION.GROUP TO 21,     AC.INACTIV.MARKER TO 22,
     AC.OPEN.ACTUAL.BAL TO 23,   AC.OPEN.CLEARED.BAL TO 24,
   AC.ONLINE.ACTUAL.BAL TO 25, AC.ONLINE.CLEARED.BAL TO 26,
     AC.WORKING.BALANCE TO 27,  AC.DATE.LAST.CR.CUST TO 28,
   AC.AMNT.LAST.CR.CUST TO 29,  AC.TRAN.LAST.CR.CUST TO 30,
   AC.DATE.LAST.CR.AUTO TO 31,  AC.AMNT.LAST.CR.AUTO TO 32,
   AC.TRAN.LAST.CR.AUTO TO 33,  AC.DATE.LAST.CR.BANK TO 34,
   AC.AMNT.LAST.CR.BANK TO 35,  AC.TRAN.LAST.CR.BANK TO 36,
   AC.DATE.LAST.DR.CUST TO 37,  AC.AMNT.LAST.DR.CUST TO 38,
   AC.TRAN.LAST.DR.CUST TO 39,  AC.DATE.LAST.DR.AUTO TO 40,
```

Part of the I_F.ACCOUNT file that links field names to field positions

For your information only



```
0001 * File Layout for CUSTOMER Created 15 OCT 07 at 04:14PM by tp0tba
0002 *       PREFIX[EB.CUS.]        SUFFIX[]
0003 EQU EB.CUS.MNEMONIC TO 1,
0004     EB.CUS.SHORT.NAME TO 2,
0005     EB.CUS.NAME.1 TO 3,
0006     EB.CUS.NAME.2 TO 4,
0007     EB.CUS.STREET TO 5,
0008     EB.CUS.ADDRESS TO 6,
0009     EB.CUS.TOWN.COUNTRY TO 7,
0010     EB.CUS.POST.CODE TO 8,
0011     EB.CUS.COUNTRY TO 9,
0012     EB.CUS.RELATION.CODE TO 10,
0013     EB.CUS.REL.CUSTOMER TO 11,
0014     EB.CUS.REVERS.REL.CODE TO 12,
0015     EB.CUS.REL.DELIV.OPT TO 13,
0016     EB.CUS.ROLE TO 14,
```

Part of the I_F.CUSTOMER file that links field names to field positions

How will you extract only CURRENCY and CATEGORY from R.ACC?

```
R.ACC<AC.CURRENCY>
R.ACC<AC.CATEGORY>
```

Take a look at the entire subroutine

```
    SUBROUTINE TRG.TEST2
    $INSERT I_COMMON
    $INSERT I_EQUATE
    $INSERT I_F.ACCOUNT
    GOSUB INIT
    GOSUB OPENFILES
    GOSUB PROCESS
    RETURN
INIT:
    FN.ACC = 'F.ACCOUNT'
    F.ACC = ''
    Y.ACC.ID = 11967
    R.ACC = ''
    Y.ACC.ERR = ''
    RETURN
OPENFILES:
    CALL OPF(FN.ACC,F.ACC)     ;* Open File
    RETURN
PROCESS:
    CALL F.READ(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR)   ;* Read Record
    CRT "Currency : ": R.ACC<AC.CURRENCY>
    CRT "Category : ": R.ACC<AC.CATEGORY>
    IF R.ACC<AC.CURRENCY> NE LCCY THEN
        *******
    END

    RETURN
END
```

Algorithm

Task	T24 API to be used
Open ACCOUNT file	OPF
Read record with key <accid>	F.READ
Insert a value “From Training” in the field ACCOUNT.TITLE.2	dynamicarray<position> = value
Write the record with key <accid>	F.WRITE
	CALL JOURNAL.UPDATE(“”)

Is the algorithm correct?

Corrected algorithm

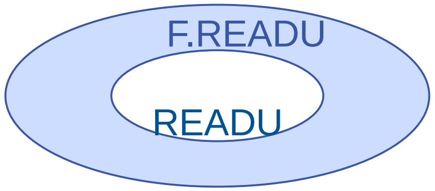
Task

- Open ACCOUNT file
- Read and lock record with key 11967
- Place "Valued customer" in the field TEXT
- Write the record with key 11967 to the database

T24 API to be used

- OPF
- F.READU NEW!
- dynamicarray<position> = value NEW!
- F.WRITE

T24 API – F.READU



- F.READU – Read and lock a record from a hashed file
- Syntax

```
CALL F.READU(Filename,Key,Record,File path,Errorvariable,Option)
```

- Example

```
CALL F.READU(FN.ACC,"11967",R.ACC,F.ACC,Y.ACC.ERR,'')
```

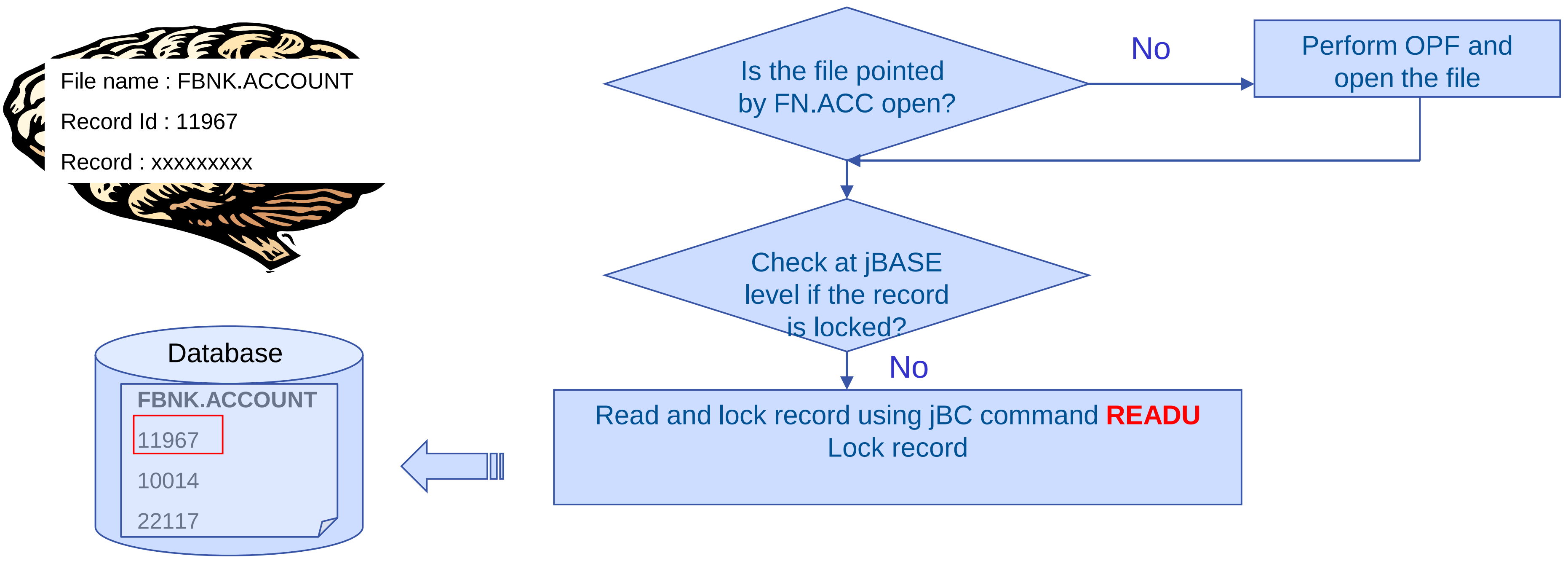
If the record is locked, wait until the lock is released and then lock

How will the subroutine look with F.READU incorporated?

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT  I_EQUATE
GOSUB INIT
GOSUB OPENFILES
GOSUB PROCESS
RETURN
INIT:
FN.ACC='F.ACCOUNT'
F.ACC=''
Y.ACC.ID=13935
R.ACC=''
Y.ACC.ERR=''
RETURN
OPENFILES:
CALL OPF(FN.ACC,F.ACC)
RETURN
PROCESS:
CALL F.READU(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR,'')
CRT "RECORD DETAILS"
CRT R.ACC
RETURN
END
```

Internal working of F.READU

```
CALL F.READU(FN.ACC,"11967",R.ACC,F.ACC,Y.ACC.ERR,'')
```



File name : FBNK.ACCOUNT
Record Id : 11967
Record : xxxxxxxxx
Is the file pointed by FN.ACC open?
No
Perform OPF and open the file
Check at jBASE level if the record is locked?
No
Read and lock record using jBC command READU
Lock record
Database
FBNK.ACCOUNT
11967
10014
22117

F.READ and F.READU - Think

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT  I_EQUATE
GOSUB INIT
GOSUB OPENFILES
GOSUB PROCESS
RETURN
INIT:
FN.ACC='F.ACCOUNT‘;F.ACC='‘;Y.ACC.ID=13935;R.ACC=''
Y.ACC.ERR=''
RETURN
OPENFILES:
CALL OPF(FN.ACC,F.ACC)
RETURN
PROCESS:
CALL TRG.TEST2
CALL
   F.READU(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR,'')
CRT "RECORD DETAILS"
CRT R.ACC
RETURN
END
```

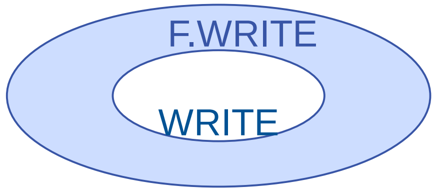
```
SUBROUTINE TRG.TEST2
------------------------
------------------------
Y.ACC.ID=13936
CALL
F.READ(FN.ACC,Y.ACC.ID,R
.ACC,F.ACC,Y.ACC.ERR)
RETURN
END
```

Will F.READU read the record 11967 from cache or will it read from the disk?

When does the lock get released?

You will learn this after you learn about F.WRITE

T24 API – F.WRITE



- F.WRITE – Write a record to the database
- Syntax

```
CALL F.WRITE(Filename,Key,Record)
```

- Example

```
CALL F.WRITE(FN.ACC,"11967",R.ACCOUNT)
```

How will the subroutine look with F.WRITE incorporated?

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT  I_EQUATE
$INSERT I_F.ACCOUNT
GOSUB INIT
GOSUB OPENFILES
GOSUB PROCESS
RETURN
INIT:
FN.ACC='F.ACCOUNT‘;F.ACC='‘;Y.ACC.ID=13935
R.ACC='‘;Y.ACC.ERR=''
RETURN
OPENFILES:
CALL OPF(FN.ACC,F.ACC)
RETURN
PROCESS:
CALL F.READU(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR,'')
CRT "RECORD DETAILS BEFORE WRITE"
CRT R.ACC
R.ACC<AC.ACCOUNT.TITLE.2>="FROM TRAINING";
CALL F.WRITE(FN.ACC,Y.ACC.ID,R.ACC)
CRT "RECORD DETAILS AFTER WRITE"
CRT R.ACC
RETURN
END
```

Name of the field picked up from I_F.ACCOUNT

Why doesn’t F.WRITE write directly to the disk? Why does it cache?

TEMENOS

- You know
  - Cache is maintained for every transaction
  - A request to T24 can be called a transaction
  - In this case TRG.TEST2 is a transaction
- Assume a scenario like the one below

```
*Routine to emphasise the working of F.READU and F.WRITE
SUBROUTINE TRG.TEST2
---------------
--------------
         CALL F.READU(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR,'') ; *Read and lock record
     R.ACC<AC.ACCOUNT.TITLE.2> = "From Training" ; *Set value
          CALL F.WRITE(FN.ACC,Y.ACC.ID,R.ACC) ; *Write record to file
     CALL F.READU(……………………………..)
     CALL F.WRITE(…………………….)
RETURN
END
```

  - Assume the first F.WRITE goes through without any errors and data is updated in the database
  - Assume the second F.WRITE fails (May be the file is corrupted or there are insufficient permissions on the file)

    Would it be fine if one of the F.WRITEs fail and the other one successfully updates the database?

Why doesn’t F.WRITE write directly to the disk? Why does it cache?

- To ensure that all data in a transaction is written to disk or none is written to disk, F.WRITEs cache data
- Who will then write data to the database
  - Answer : JOURNAL.UPDATE
- Transaction management when bulk messages (BROWSER)
  - Requests from browser are considered as BULK
  - Data is flushed to the disk when bulk transaction is complete

How will the subroutine look with JOURNAL.UPDATE incorporated?

TEMENOS

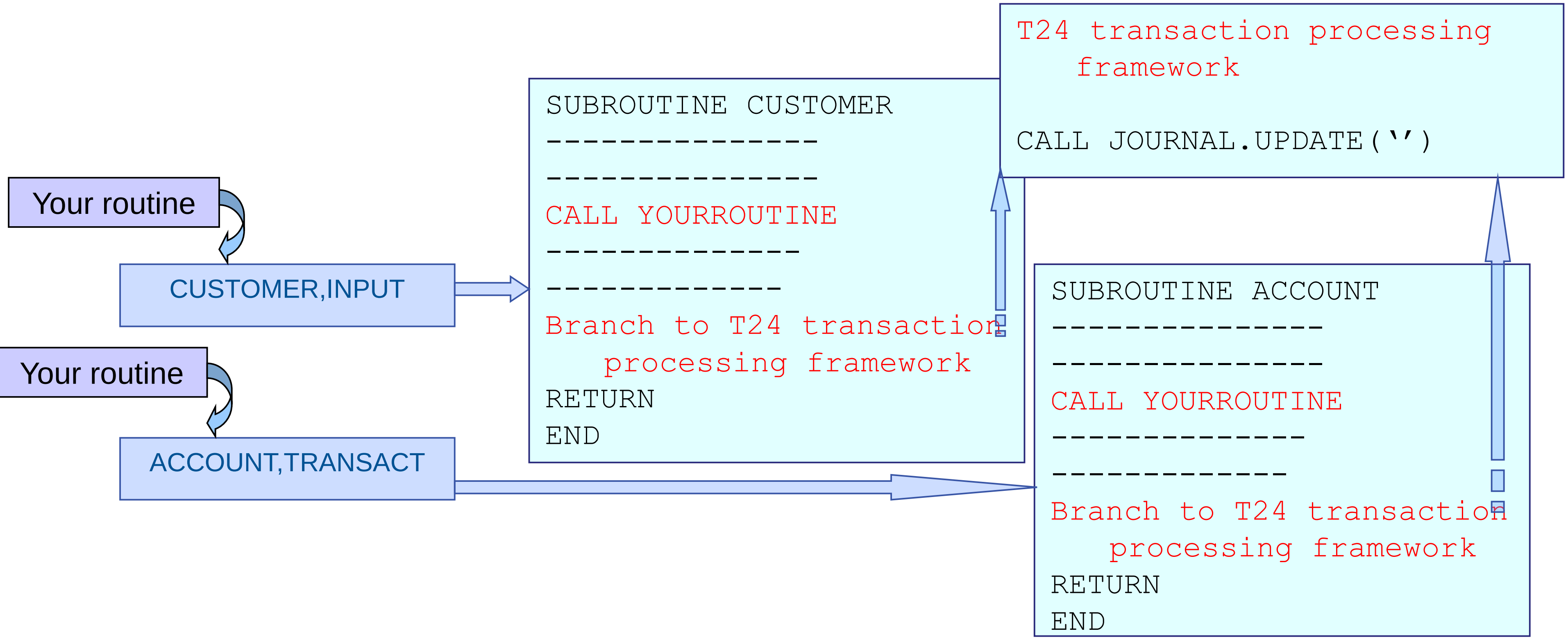
```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT I_EQUATE
$INSERT I_F.ACCOUNT
GOSUB INIT
GOSUB OPENFILES
GOSUB PROCESS
RETURN
INIT:
FN.ACC='F.ACCOUNT‘;F.ACC='‘;Y.ACC.ID=13935
R.ACC='‘;Y.ACC.ERR=''
RETURN
OPENFILES:
CALL OPF(FN.ACC,F.ACC)
RETURN
PROCESS:
CALL F.READU(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR,'')
CRT "RECORD DETAILS BEFORE WRITE"
CRT R.ACC
R.ACC<AC.ACCOUNT.TITLE.2>="FROM TRAINING";
CALL F.WRITE(FN.ACC,Y.ACC.ID,R.ACC)
CALL JOURNAL.UPDATE(Y.ACC.ID)
CRT "RECORD DETAILS AFTER WRITE"
CRT R.ACC
RETURN
END
```

More on JOURNAL.UPDATE

- Do I have to call JOURNAL.UPDATE for every routine that I write?

NO

- The routines that you are writing now are mainline routines
  - Standalone routines
  - Executed from the T24 command line
- Normally routines are written and attached to various applications in T24. T24 applications will call JOURNAL.UPDATE



```
SUBROUTINE CUSTOMER
---------------
---------------
CALL YOURROUTINE
--------------
-------------
Branch to T24 transaction
   processing framework
RETURN
END
```

```
T24 transaction processing
   framework

CALL JOURNAL.UPDATE('')
```

```
SUBROUTINE ACCOUNT
---------------
---------------
CALL YOURROUTINE
--------------
-------------
Branch to T24 transaction
   processing framework
RETURN
END
```

Back to the question – When do locks get released?

- A lock on a record is released when
  - An F.WRITE is executed on the record that has been locked
  - If within a transaction, then, F.WRITE will not release the lock. Once the transaction is complete, the lock gets released.

    When TRANSEND is called by JOURNAL.UPDATE, all locks get released
  - Internally calls the jBC command RELEASE
  - RELEASE <filename> releases all locks on the given file held by current session

T24 API – F.RELEASE

F.RELEASE
RELEASE

Use this with caution

- Used to release locks. Locks are released at the end of the transaction.
- Use it only if you have locked a record using F.READU but haven’t written the record back using F.WRITE

```
*Routine to emphasise the working of F.RELEASE
SUBROUTINE TRG.TEST2
----------------
         CALL F.READU(FN.ACC,11967,R.ACC,F.ACC,Y.ACC.ERR,'') ; *Read and lock record
          CALL F.READU(FN.ACC,11956,R.ACC,F.ACC,Y.ACC.ERR,'') ; *Read and lock record
    CALL F.WRITE(FN.ACC,11956,R.ACC) ; *Write record to file
    CALL F.RELEASE(FN.ACC,11967,F.ACC)
---------------
RETURN
END
```

- If no record key is specified – All locks on the file name specified are released
- If no file name is specified, all locks are released (Online only)

Workshop 2

TEMENOS

- Write a routine to update the field TEXT in the CUSTOMER application with a value “This is from training”. Use any customer number of your choice.
- Note : TEXT is a multi value field. Write code in such a way that data is always appended to the field and not overwritten

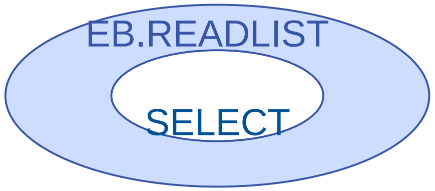
Task

- Write a routine to display the ID, category and currency of all accounts

Algorithm

Task	T24 API to be used
Select all account IDs	EB.READLIST NEW!
Start loop	jBC command - LOOP
Read each account record	F.READ
Extract category and currency value	Variable=dynamicarray<position>
Display ID, category and currency	CRT variable
Loop back	jBC command - REPEAT

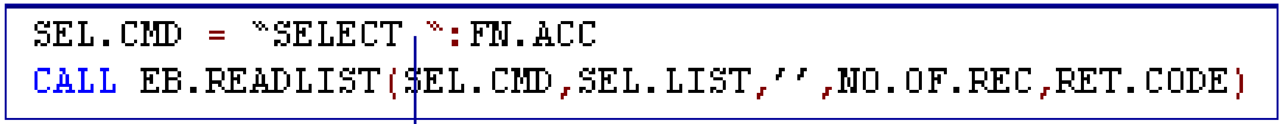
T24 API – EB.READLIST



- EB.READLIST – Execute a SELECT statement
- Syntax

```
CALL
EB.READLIST(Selectcommand,Selectedlist,'',NoOfRecordsSelected,
ReturnCode)
```

- Example



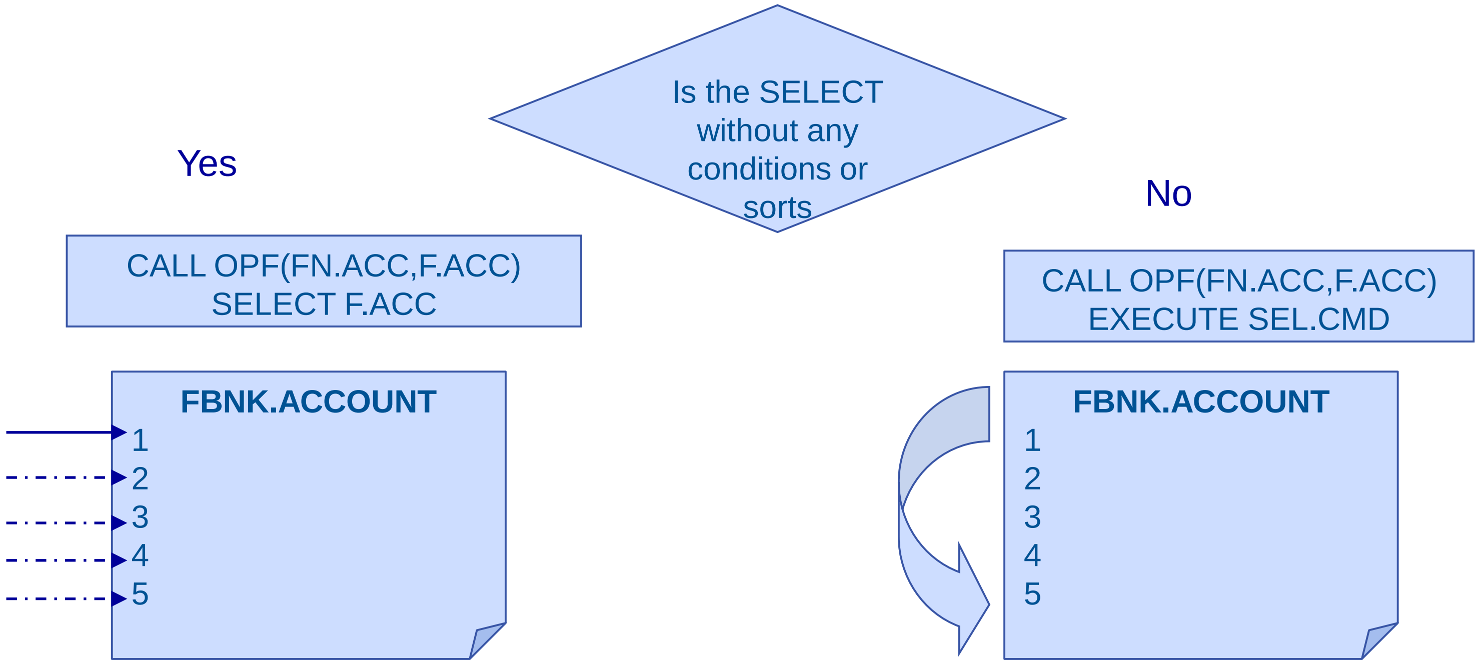
```
SEL.CMD = "SELECT ":FN.ACC
CALL EB.READLIST(SEL.CMD,SEL.LIST,'',NO.OF.REC,RET.CODE)
```

Note the space

You may use SSELECT instead of SELECT if you want data in sorted order

Internal working of EB.READLIST

```
SEL.CMD = "SELECT ":FN.ACC
CALL EB.READLIST(SEL.CMD,SEL.LIST,'',NO.OF.REC,RET.CODE)
```


Is the SELECT without any conditions or sorts
Yes
No
CALL OPF(FN.ACC,F.ACC) SELECT F.ACC
CALL OPF(FN.ACC,F.ACC) EXECUTE SEL.CMD
FBNK.ACCOUNT
1
2
3
4
5
FBNK.ACCOUNT
1
2
3
4
5

Subroutine with EB.READLIST incorporated

```
SUBROUTINE TRG.TEST1
$INSERT I_COMMON
$INSERT I_EQUATE
$INSERT I_F.ACCOUNT
GOSUB INIT
GOSUB OPENFILES
GOSUB PROCESS
RETURN
INIT:
        FN.ACC='F.ACCOUNT‘;F.ACC='‘;Y.ACC.ID=13935;R.ACC='‘;Y.ACC.ERR='‘

        RETURN
OPENFILES:
        CALL OPF(FN.ACC,F.ACC)
        RETURN
PROCESS:
        SEL.CMD="SELECT ":FN.ACC
        CALL EB.READLIST(SEL.CMD,SEL.LIST,'',NO.OF.REC,RET.CODE)
        LOOP
                REMOVE Y.ACC.ID FROM SEL.LIST SETTING POS
        WHILE Y.ACC.ID:POS
                CALL F.READ(FN.ACC,Y.ACC.ID,R.ACC,F.ACC,Y.ACC.ERR,'')
        CRT "ID-":Y.ACC.ID :"CATEGORY-
":R.ACC<AC.CATEGORY>:"CURRENCY":R.ACC<AC.CURRENCY>
        REPEAT
```

Some more APIs

- EB.READ.PARAMETER
  - To read parameter record of any module
  - Returns both dynamic and dimension array
  - Can indicate whether to read the record with a lock or not

- F.DELETE
  - Deletes the record from cache
  - During cob if write cache is not enabled delete the record immediately

Discussion

- Form 5 groups
- Topics for each group
  - Group 1 : OPF
  - Group 2 : F.READ and CACHE.READ
  - Group 3 : F.READU
  - Group 4 : F.WRITE
  - Group 5 : JOURNAL.UPDATE and F.RELEASE

- Discuss and understand the working of the T24 API given to you (10 minutes)
- Form questions to ask the other groups – 5 questions (10 minutes)
- Groups ask questions to each other
- Note the group that is the highest scorer

Workshop

- READ all of record from ACCOUNT table , print ACCOUNT.ID , CUSTOMER value
- READ all of record from ACCOUNT table , print ACCOUNT.ID , CUSTOMER value , Read SECTOR description by this CUSTOMER value from CUSTOMER table , print
- Update ACCOUNT table ACCOUNT.TITLE.2 field from the SECTOR SHORT.NAME

SECTOR 1 : CUSTOMER:111444 , ACCOUNT:46949

SECTOR 2 : CUSTOMER:111476 , ACCOUNT:74802

SECTOR 3 : CUSTOMER:100503 , ACCOUNT:36668

1.SUBROUTINE

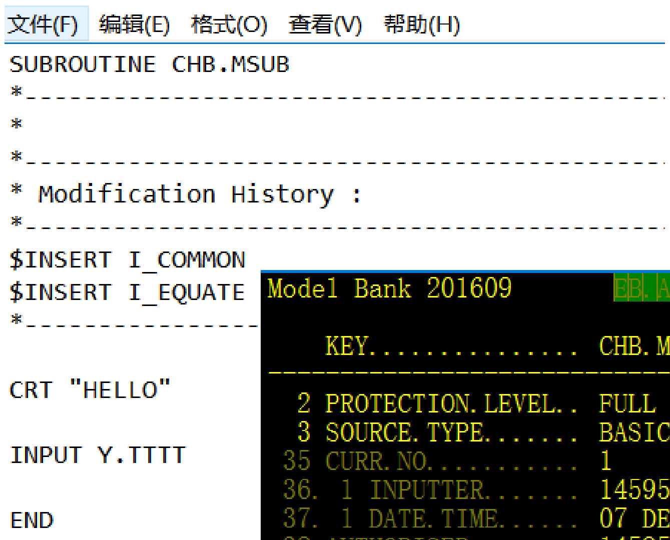
2.PGM.FILE M

3.EXECUTE under EX

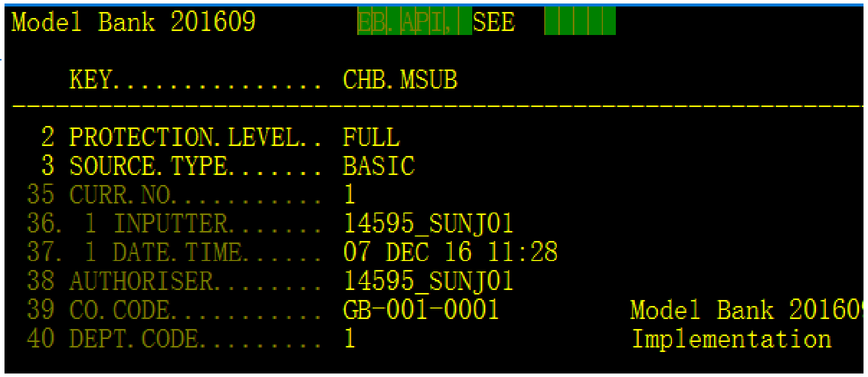
4.OPF/F.READ/F.READU/F.WRITE/EB.READLIST/JOURNAL.UPDATE

T24 Main Routine

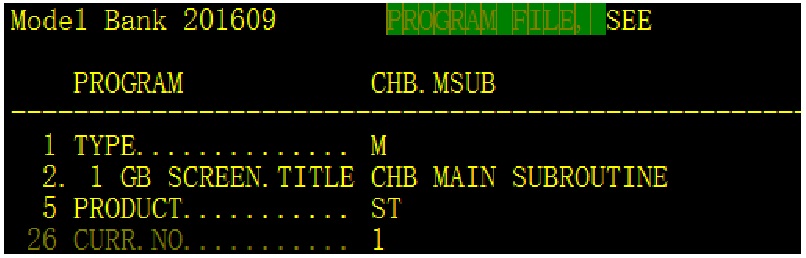
- Write routine
- TCOMPILE
- TRUN EX
- EB.API
- PGM.FILE M
- EXECUTE



```
SUBROUTINE CHB.MSUB
*-----------------------------------------
*
*-----------------------------------------
* Modification History :
*-----------------------------------------
$INSERT I_COMMON
$INSERT I_EQUATE
*----------------

CRT "HELLO"

INPUT Y.TTTT

END
```



```
Model Bank 201609         EB.API, SEE

   KEY............... CHB.MSUB
------------------------------------------------------------
  2 PROTECTION.LEVEL.. FULL
  3 SOURCE.TYPE....... BASIC
 35 CURR.NO........... 1
 36. 1 INPUTTER....... 14595_SUNJ01
 37. 1 DATE.TIME...... 07 DEC 16 11:28
 38 AUTHORISER........ 14595_SUNJ01
 39 CO.CODE........... GB-001-0001          Model Bank 20160
 40 DEPT.CODE......... 1                    Implementation
```



```
Model Bank 201609         PROGRAM FILE, SEE

   PROGRAM              CHB.MSUB
------------------------------------------------------------
  1 TYPE.............. M
  2. 1 GB SCREEN.TITLE CHB MAIN SUBROUTINE
  5 PRODUCT........... ST
 26 CURR.NO........... 1
```