

Tutorial 5

Datenbankzugriff mit CICS

Ziel dieses Tutorials ist es, mittels einer CICS-Transaktion auf die in Tutorial 4 erstellte DB2-Datenbank zuzugreifen.

Unser Anwendungsprogramm soll wieder aus zwei Teilen bestehen, einem C-Programm für die Business Logic und einem BMS-Programm für die Presentation Logic.

Unser Business Logic-Programm soll dabei SQL-Aufrufe enthalten.

Diese müssen durch einen SQL-Precompiler in das native API des DB2-Datenbanksystems übersetzt werden, ehe der C-Compiler das Business Logic-Programm übersetzen kann.

Aufgabe: Arbeiten Sie nachfolgendes Tutorial durch.

Wir loggen uns als TSO-Benutzer ein, und arbeiten uns zum DLIST-Panel vor.

```

Menu  Options  View  Utilities  Compilers  Help
-----
DSLIST - Data Sets Matching PRAKT20                               Row 1 of 13

Command - Enter "/" to select action                               Message                               Volume
-----
PRAKT20                                                           *ALIAS
PRAKT20.CICS.TEST04                                             SMS001
PRAKT20.CICSDB2.TEST01                                         SMS001
PRAKT20.DBRMLIB.DATA                                           SMS001
PRAKT20.ISPF.ISPPROF                                           SMS001
PRAKT20.LIB                                                     SMS001
PRAKT20.SPFL0G1.LIST                                           SMS001
PRAKT20.SPUFI.IN                                               SMS001
PRAKT20.SPUFI.OUT                                              SMS001
PRAKT20.TEST.C                                                 SMS001
PRAKT20.TEST.CNTL                                             SMS001
PRAKT20.TEST.LOAD                                              SMS001
***** End of Data Set list *****

Command ==>
F1=Help   F3=Exit   F5=Rfind  F12=Cancel

Scroll ==> PAGE

```

Abbildung 1: der DLIST-Panel

Wir hatten im vorangegangenen Tutorial alle hierfür erforderlichen Partitioned Datasets angelegt. Unser DLIST-Panel zeigt 11 Partitioned Datasets (s. Abbildung 1). "SPUFI.IN" wurde in unserer letzten Übung (Tutorial 4) benutzt, um unsere Datenbank anzulegen. "SPUFI.OUT" wurde vom SPUFI-Subsystem angelegt und enthält die Übersetzung unserer Eingaben.

Die nun benötigten Datasets "PRAKT20.DBRMLIB.DATA", "PRAKT20.LIB" und "PRAKT20.CICSDB2.TEST01" wurden ebenfalls bereits im letzten Tutorial angelegt. "PRAKT20.DBRMLIB.DATA" ist noch leer. Es wird während der Ausführung des SQL-

Ein leeres Edit Entry Panel erscheint (s. Abbildung 3). Unsere CICS-Anwendung soll wiederum aus einem BMS-Programm (Mapset) für die "Presentation Logic" und einem C-Programm für die Business Logic bestehen. Wir beginnen mit dem Mapset (PRAKSET).

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAKT20.CICSDB2.TEST01(PRAKSET) - 01.00          Columns 00001 00072
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to NUMBER OFF (from NUMBER ON STD).
==MSG>          Data does not have valid standard numbers.
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>          data does not contain any lower case characters.
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 //PREPARE JOB ( ),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID
000002 //ASSEM EXEC DFHMAPS,MAPNAME='S01SET',RMODE=24
000003 //SYSUT1 DD *
000004 S01SET DFHMSD TYPE=MAP,MODE=INOUT,LANG=C,STORAGE=AUTO,TIOAPFX=YES
000005 *          MENU MAP.
000006 LISTE DFHMDF SIZE=(24,80),CTRL=(PRINT,FREEKB)
000007 DFHMDF POS=(9,13),ATTRB=(ASKIP,NORM),LENGTH=20, X
000008          INITIAL='VORNAME'
000009 DFHMDF POS=(9,34),ATTRB=(ASKIP,NORM),LENGTH=20, X
000010          INITIAL='NACHNAME'
000011 VNAME1 DFHMDF POS=(11,13),ATTRB=(ASKIP,NORM),LENGTH=20
000012 NNAME1 DFHMDF POS=(11,34),ATTRB=(ASKIP,NORM),LENGTH=20
Command ==>          Scroll ==> PAGE
F1=Help          F3=Exit          F5=Rfind          F6=Rchange          F12=Cancel

```

Abbildung 4: Das BMS-Programm

Dies ist das vollständige BMS-Programm nach Fertigstellung. Es umfaßt 2 Panels. Mit den F8- bzw. F7-Tasten scrollen wir zwischen den beiden Panels hin und her.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAKT20.CICSDB2.TEST01(PRAKSET) - 01.01          Columns 00001 00072
000013 VNAME2 DFHMDF POS=(12,13),ATTRB=(ASKIP,NORM),LENGTH=20
000014 NNAME2 DFHMDF POS=(12,34),ATTRB=(ASKIP,NORM),LENGTH=20
000015 VNAME3 DFHMDF POS=(13,13),ATTRB=(ASKIP,NORM),LENGTH=20
000016 NNAME3 DFHMDF POS=(13,34),ATTRB=(ASKIP,NORM),LENGTH=20
000017 VNAME4 DFHMDF POS=(14,13),ATTRB=(ASKIP,NORM),LENGTH=20
000018 NNAME4 DFHMDF POS=(14,34),ATTRB=(ASKIP,NORM),LENGTH=20
000019 DFHMSD TYPE=FINAL
000020          END
000021 /*
000022 //
***** ***** Bottom of Data *****
Command ==> SUB          Scroll ==> PAGE
F1=Help          F3=Exit          F5=Rfind          F6=Rchange          F12=Cancel

```

Abbildung 5: Zweiter Teil des BMS-Programms

Die Zeilen 7 bis 10 definieren eine Überschrift, die aus 2 Feldern besteht. Die beiden Felder werden mit den Werten VORNAME und NACHNAME initialisiert.

Die Zeilen 11 bis 18 definieren 8 Felder, welche die Vornamen und Nachnamen von 4 Personen aufnehmen sollen, welche wir aus unserer DB2-Datenbank auslesen.

Wir geben "SUB" auf der Kommandozeile ein. Zusätzlich zu dem übersetzten Programm wird in dem Member "PRAKT20.LIB(S01SET)" ein Template für unser Business Logic-Programm (in C) abgespeichert.

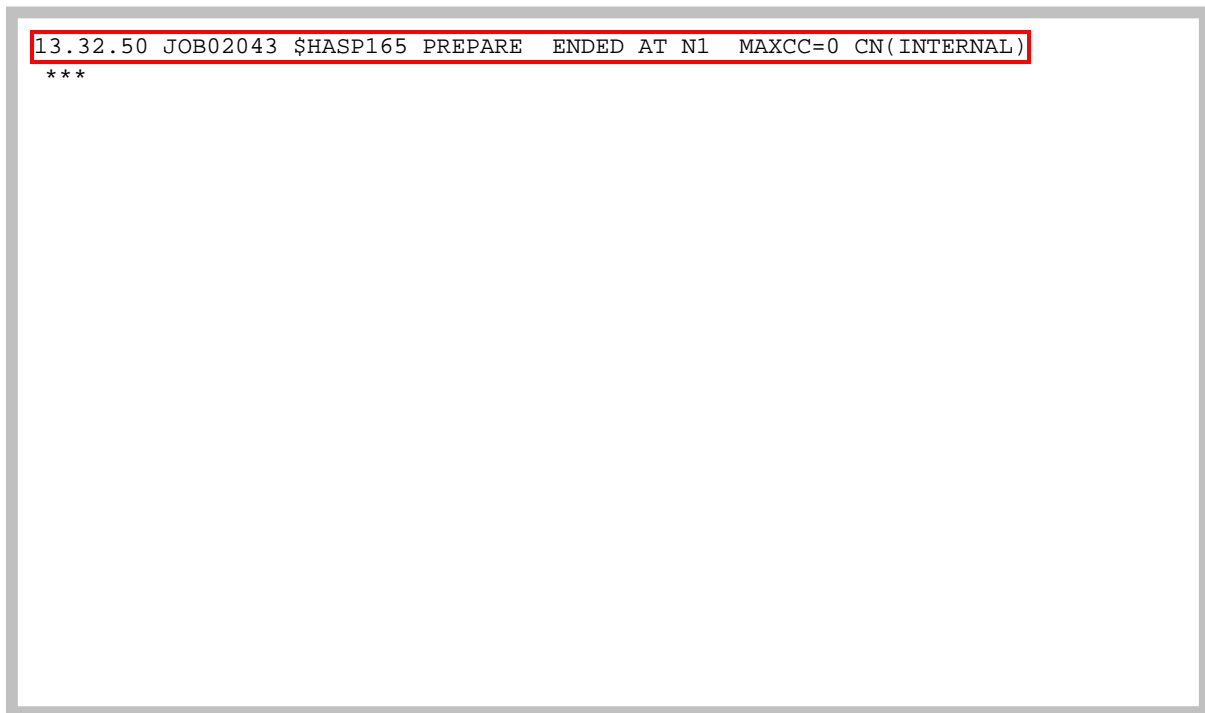


Abbildung 6: Bestätigung der Jobverarbeitung

Wir warten, bis "JES" unser BMS-Programm übersetzt hat (30-60 Sekunden). Durch das Betätigen der Eingabetaste erscheint das hier gezeigte Panel (s.Abbildung 6). "MAXCC=0" bestätigt, dass die Übersetzung erfolgreich war.

Die Eingabetaste bringt uns zurück zum vorhergehenden Screen.

Aufgabe: Legen Sie einen Member an, schreiben Sie das BMS-Programm und führen Sie es aus.

Wir betätigen zweimal die F3-Taste, um diesen Bildschirm zu verlassen.

Als nächstes sehen wir uns die Members von "PRAKT20.LIB" an.

Wir wechseln zu dem Partitioned Dataset "PRAKT20.LIB". Dort existiert jetzt das während der Übersetzung erstellte Member "PRAKT20.LIB(S01SET)". Wir schauen uns "PRAKT20.LIB(S01SET1)" an.

```

union
{
  struct {
    char      dfhms1Ý12";
    short int vnam1l;
    char      vnam1f;
    char      vnam1iÝ20";
    short int nnam1l;
    char      nnam1f;
    char      nnam1iÝ20";
    short int vnam2l;
    char      vnam2f;
    char      vnam2iÝ20";
    short int nnam2l;
    char      nnam2f;
    char      nnam2iÝ20";
    short int vnam3l;
    char      vnam3f;
    char      vnam3iÝ20";
    short int nnam3l;
    char      nnam3f;
    char      nnam3iÝ20";
    short int vnam4l;
    char      vnam4f;
    char      vnam4iÝ20";
    short int nnam4l;
    char      nnam4f;
    char      nnam4iÝ20";
  } listei;

  struct {
    char      dfhms2Ý12";
    short int dfhms3;
    char      vnam1a;
    char      vnam1oÝ20";
    short int dfhms4;
    char      nnam1a;
    char      nnam1oÝ20";
    short int dfhms5;
    char      vnam2a;
    char      vnam2oÝ20";
    short int dfhms6;
    char      nnam2a;
    char      nnam2oÝ20";
    short int dfhms7;
    char      vnam3a;
    char      vnam3oÝ20";
    short int dfhms8;
    char      nnam3a;
    char      nnam3oÝ20";
    short int dfhms9;
    char      vnam4a;
    char      vnam4oÝ20";
    short int dfhms10;
    char      nnam4a;
    char      nnam4oÝ20";
  } listeo;
} liste;

```



```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAKT20.CICSDB2.TEST01(TESTPR1) - 01.01          Columns 00001 00072
000015      EXEC SQL OPEN C1;
000016      EXEC SQL FETCH C1 INTO :vname, :nname;
000017      memcpy(liste.listei.vnam1i,vname,20);
000018      memcpy(liste.listei.nnam1i,nname,20);
000019      EXEC SQL FETCH C1 INTO :vname, :nname;
000020      memcpy(liste.listei.vnam2i,vname,20);
000021      memcpy(liste.listei.nnam2i,nname,20);
000022      EXEC SQL FETCH C1 INTO :vname, :nname;
000023      memcpy(liste.listei.vnam3i,vname,20);
000024      memcpy(liste.listei.nnam3i,nname,20);
000025      EXEC SQL FETCH C1 INTO :vname, :nname;
000026      memcpy(liste.listei.vnam4i,vname,20);
000027      memcpy(liste.listei.nnam4i,nname,20);
000028      EXEC SQL CLOSE C1;
000029
000030      EXEC CICS SEND MAP("liste") MAPSET("s01set") ERASE;
000031
000032
000033 }
Command ==>
F1=Help      F3=Exit      F5=Rfind     F6=Rchange   F12=Cancel   Scroll ==> PAGE

```

Abbildung 10: Der zweite Teil des Business Logic-Programms

In Zeile 000007 und Zeile 000008 von "PRAKT20.CICSDB2.TEST01(TESTPR1)" fällt das Sonderzeichen "Ý" auf. Dies hat wieder etwas mit dem "eckige-Klammern"- Problem bei der Erstellung von C-Programmen zu tun.

Wir erinnern uns: TSO speichert alle Daten im EBCDIC-Format. Im EBCDIC-Format gibt es 2 Darstellungen für die eckigen Klammern, nämlich:

Character	ASCII hex	Proper EBCDIC hex	Improper EBCDIC hex
Left Square ([)	x'5B'	x'AD'	x'BA'
Right Square (])	x'5D'	x'BD'	x'BB'

Der 3270-Emulator wird häufig hex BA und hex BB als eckige Klammern "[" und "]" darstellen, während hex AD und hex BD als "Ý" und "" dargestellt werden.

Wichtig ist, das Problem zu verstehen. Es gibt mehrere Möglichkeiten es anzugehen. Das einfachste Verfahren ist folgendes:

Bei der Programmeingabe normal die Symbole "[" und "]" verwenden. Diese werden dann fälschlicherweise als x'BA und x'BB abgespeichert.

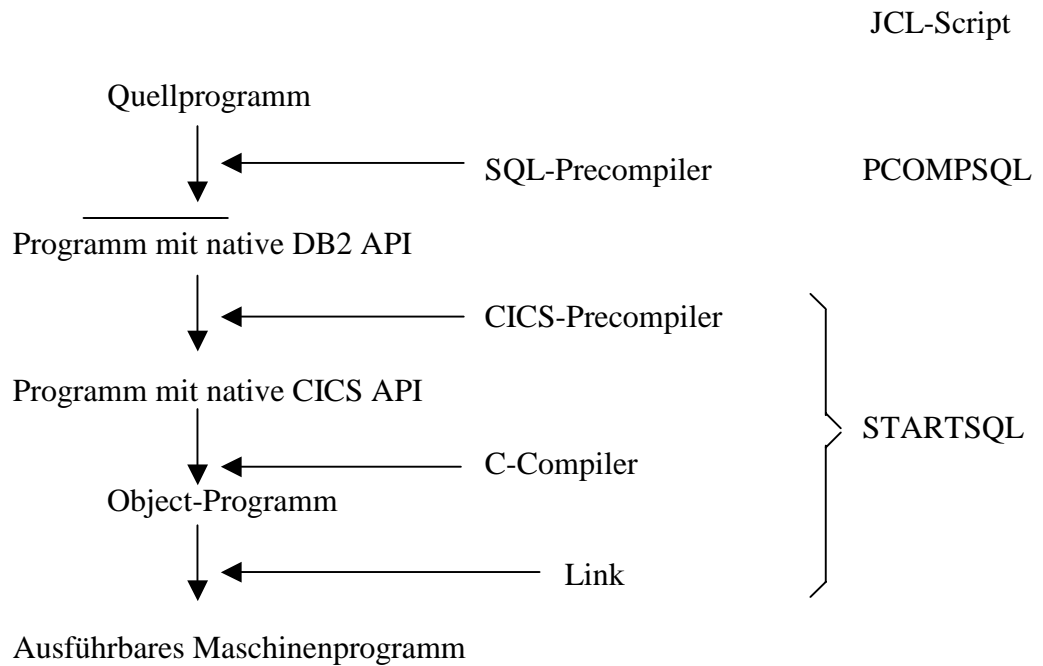
Nach Fertigstellung der Programmeingabe werden zwei globale ISPF "Change"-Kommandos eingegeben. Dies erfolgt durch Eingabe in der Kommandozeile:

```

C [ x'ad' all
C ] x'bd' all

```

Statt "c" kann auch "change" verwendet werden.



Wir drücken anschließend die Eingabetaste.

```

File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-----
EDIT      PRAKT20.CICSDB2.TEST01(PCOMPSQL) - 01.02      Columns 00001 00072
*****  ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 //DB2PCOMP JOB ( ),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000002 //          TIME=1440
000003 //PCOMP     EXEC PROC=DSNZEY
000004 //DBRMLIB  DD DSN=PRAKT20.DBRMLIB.DATA(TESTPR1),DISP=OLD
000005 //SYSCIN   DD DSN=PRAKT20.CICSDB2.TEST01(OUT),DISP=SHR
000006 //SYSLIB   DD DSN=PRAKT20.CICSDB2.TEST01,DISP=SHR
000007 //SYSIN   DD DISP=SHR,DSN=PRAKT20.CICSDB2.TEST01(TESTPR1)
*****  ***** Bottom of Data *****

Command ==> SUB
F1=Help      F3=Exit      F5=Rfind     F6=Rchange   F12=Cancel
Scroll ==> PAGE
  
```

Abbildung 12: Das JCL-Script

Wir erstellen das in Abbildung 12 dargestellte JCL-Script zum Aufruf des EXEC SQL-Precompilers.

"PRAKT20.DBRMLIB.DATA" enthält bis hierher kein Member. Nach der Ausführung des JCL-Scriptes hat der Precompiler einen Member "PRAKT20.DBRMLIB.DATA(TESTPR1)" angelegt.

Auf der Kommandozeile geben wir wieder den SUBMIT-Befehl "SUB" ein. Wir warten die Ausführung des JES-Jobs ab (s. Abbildung 13) und bestätigen diese dann mit der Eingabetaste.

```
17.12.04 JOB02051 $HASP165 DB2PCOMP ENDED AT N1 MAXCC=0 CN(INTERNAL)
***
```

Abbildung 13: Bestätigung der Jobverarbeitung

"MAXCC=0" zeigt an, dass der Befehl erfolgreich ausgeführt wurde. Wir bestätigen mit der Eingabetaste.

Aufgabe: Erstellen Sie einen neuen Member und schreiben Sie den Precompiler hinein. Führen Sie ihn anschließend aus.

"STARTSQL" erstreckt sich über 3 Panels.

Panel #1: Name unseres C-Programms (Zeile 10).

Mit der F8-Taste scrollen wir weiter.

```

File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-----
EDIT          PRAKT20.CICSDB2.TEST01(STARTSQL) - 01.04          Columns 00001 00072
000017 //DBRMLIB DD DISP=OLD,DSN=PRAKT20.DBRMLIB.DATA(TESTPR1)
000018 //SYSPRINT DD SYSOUT=*
000019 //SYSTSPRT DD SYSOUT=*
000020 //SYSUDUMP DD SYSOUT=*
000021 //SYSTSIN DD *
000022 DSN S(DBA1)
000023 BIND PLAN(PR1) MEMBER(TESTPR1) ACTION(REP) RETAIN ISOLATION(CS)
000024 END
000025 //*****
000026 //* GRANT
000027 //*****
000028 //GRANT EXEC PGM=IKJEFT01
000029 //STEPLIB DD DISP=SHR,DSN=DSN510.SDSNLOAD
000030 //SYSPRINT DD SYSOUT=*
000031 //SYSTSPRT DD SYSOUT=*
000032 //SYSUDUMP DD SYSOUT=*
000033 //SYSTSIN DD *
000034 DSN SYSTEM(DBA1)
000035 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA51) -
Command ==>
F1=Help          F3=Exit          F5=Rfind         F6=Rchange      F12=Cancel
Scroll ==> PAGE

```

Abbildung 16: Das JCL-Script (Panel#2)

Panel #2: Der SQL-Precompiler-Lauf hat im Dataset "PRAKT20.DBRMLIB.DATA" ein Member "TESTPR1" angelegt (Zeile 17).

Die Ausführung unseres Programms "TESTPR1" unter CICS benötigt einen Zeiger auf die anzusprechende Datenbank-Tabelle (als PLAN bezeichnet). Wir geben diesem Zeiger den Namen "PR1" (Zeile 23).

Mit der F8-Taste können wir uns das restliche Script ansehen.

```

File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-----
EDIT          PRAKT20.CICSDB2.TEST01(STARTSQL) - 01.04          Columns 00001 00072
000036          LIBRARY('DSN510.RUNLIB.LOAD')
000037  END
000038 //SYSIN      DD *
000039 GRANT EXECUTE ON PLAN PR1 TO PUBLIC
000040 /*
***** ***** Bottom of Data *****
Command ==>          Scroll ==> PAGE
F1=Help      F3=Exit      F5=Rfind      F6=Rchange  F12=Cancel

```

Abbildung 17: Das JCL-Script (Panel #3)

Panel #3: Die Referenz auf Tabelle (Plan) "PR1" taucht nochmals auf.

Sie können für ähnliche Aufgaben dieses JCL-Script immer wieder verwenden, indem Sie lediglich an den durch Pfeile gekennzeichneten Stellen ihre eigenen Programmnamen eingeben.

```

//CICS PRE JOB( ), CLASS=A, MSGCLASS=H, MSGLEVEL=(1,1), NOTIFY=&SYSUID,
//          TIME=1440
//*****
//* TRANSL/COMP/LINKEDIT
//*****
//COMP      EXEC PROC=CTOCICS, REG=0M
//TRN.SYSIN DD DISP=SHR, DSN=PRAKT20.CICSDB2.TEST01(OUT)
//LKED.SYSIN DD *
//          INCLUDE DB2LOAD(DSNCLI)
//          NAME TESTPR1(R)
//*****
//* BIND
//*****
//BIND      EXEC PGM=IKJEFT01
//STEPLIB   DD DISP=SHR, DSN=DSN510.SDSNEXIT
//          DD DISP=SHR, DSN=DSN510.SDSNLOAD

//DBRMLIB   DD DISP=OLD, DSN=PRAKT20.DBRMLIB.DATA(TESTPR1)
//SYS PRINT DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSTSIN   DD *
//          DSN S(DBA1)
//          BIND PLAN(PR1) MEMBER(TESTPR1) ACTION(REP) RETAIN ISOLATION(CS)
END
//*****
//* GRANT
//*****
//GRANT     EXEC PGM=IKJEFT01
//STEPLIB   DD DISP=SHR, DSN=DSN510.SDSNLOAD

```

```
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(DBA1)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA51) -

LIBRARY('DSN510.RUNLIB.LOAD')
  END
//SYSIN DD *
GRANT EXECUTE ON PLAN PR1 TO PUBLIC
/*
```

Wir geben "SUB" auf der Kommandozeile ein, warten, bis JES den Job ausgegeben hat und bestätigen anschließend mit der Eingabetaste.

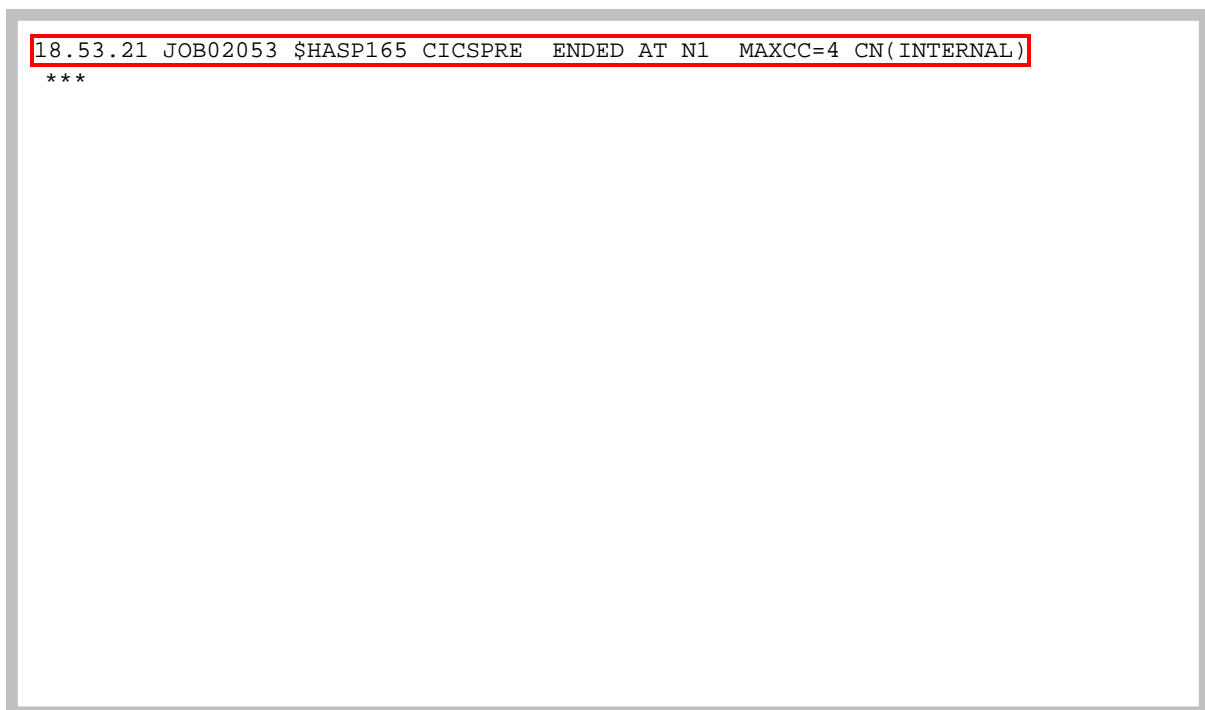


Abbildung 18: Ausgabe der Jobverarbeitung

"MAXCC=4" bedeutet, dass der Compile- und Link-Lauf erfolgreich durchgeführt wurde.

Aufgabe: Erstellen Sie einen neuen Member, legen Sie das JCL-Script an und führen Sie es aus.

Wir haben nun alle Programme für unsere CICS - DB2-Transaktion erstellt. Als nächsten Schritt müssen sie in dem CICS-Subsystem installiert werden. Hierzu öffnen wir eine weitere OS/390-Session.

```

TCPIP MSG10 ==> SOURCE DATA SET = SYS1.LOCAL.VTAMLST(USSTCPIP)

03/01/01                W E L C O M E   T O                19:20:39

          SSSSSS   //   3333333  9999999  0000000
        SS       //   33  33  99  99  00  00
       SS        //           33  99  99  00  00
      SSSS       //   33333  9999999  00  00
     SS         //           33           99  00  00
    SS          //   33  33  99  99  00  00
   SSSSSS      //   3333333  9999999  0000000

YOUR TERMINAL NAME IS :                YOUR IP ADDRESS IS : 217.002.103.061

          APPLICATION DEVELOPMENT SYSTEM
          OS/390 RELEASE 2.7.0

==> ENTER "L " FOLLOWED BY THE APPLID YOU WISH TO LOGON TO.  EXAMPLE "L TSO"
    FOR TSO/E OR "L C001" FOR THE CICSC001 CICS APPLICATION.

L C001
    
```

Abbildung 19: Der Login-Screen

Wir loggen uns mit "L C001" ein (s. Abbildung 19) und bestätigen mit der Eingabetaste.

```

CICS C001 A06C001      IBM DEMONSTRATION SYSTEM 24:00:00

          *****\ *****\ *****\ *****\
          *****\ *****\ *****\ *****\
          **\\\\\\**\ **\\\\\\ **\\\\\\**\ **\\\\\\**\
          **\      \\\ **\      \\\ **\      \\\ **\      \\\
          **\      **\      **\      *****\
          **\      **\      **\      *****\
          **\      **\      **\      \\\\\\\**\
          **\      **\      **\      **\      **\
          *****\ *****\ *****\ *****\
          *****\\ *****\\ *****\\ *****\\
          \\\\\\\ \\\\\\\ \\\\\\\ \\\\\\\ TM

DFHAC2001 03/01/01 21:25:38 A06C001 Transaction '' is not recognized. Check
that the transaction name is correct. CEDA DISPLAY GROUP(*)
    
```

Abbildung 20: CICS

Das CICS Entry Panel erscheint. Wir drücken nochmals die Eingabetaste. Eine Fehlermeldung erscheint. Wir geben den "CEDA DISPLAY GROUP(*)"-Befehl ein und bestätigen mit der Eingabetaste.

```

CEDA DEFINE MAPSET(S01SET) GROUP(PRAKT20)
ENTER COMMANDS
GROUP
AOR2TOR
ARTT
ATC
CBPS
CEE
CICREXX
CSQ
CSQCKB
CSQSAMP
CTA1TCP
C001EZA
C001TCP
DAVIN4
DAVIN8
DAVIN85
DAVIN9
+ DAVIN94

RESULTS: 1 TO 17
PF 1 HELP          3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

SYSID=C001 APPLID=A06C001
TIME: 00.00.00 DATE: 01.060

```

Abbildung 21: Die bestehenden Gruppen

Der "CEDA DISPLAY GROUP(*)"-Befehl zeigt alle bisher vorhandenen Gruppen an (s. Abbildung 21).

Wir definieren zunächst unser BMS-Programm mit dem Namen "S01SET1" für die neue Group "PRAKT20" und bestätigen anschließend mit der Eingabetaste (s. Abbildung 22).

Der Group-Name kann beliebig gewählt, aber immer bloß einmal vergeben werden. Der Übersichtlichkeit wegen ist es sinnvoll, den Login-Namen zu verwenden.

```

CEDA DEFINE MAPSET(S01SET) GROUP(PRAKT20)
OVERTYPE TO MODIFY
CEDA DEFine Mapset( S01SET )
Mapset      : S01SET
Group       : PRAKT20
Description ==>
RESident    ==> No           No | Yes
USAge       ==> Normal       Normal | Transient
USElpacopy  ==> No           No | Yes
Status      ==> Enabled      Enabled | Disabled
RS1         : 00             0-24 | Public

CICS RELEASE = 0530

I New group PRAKT20 created.

DEFINE SUCCESSFUL
PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

SYSID=C001 APPLID=A06C001
TIME: 00.00.00 DATE: 01.060

```

Abbildung 22: Definition von S01SET

Die Installation war erfolgreich, und die neue Gruppe wurde erstellt.

```

CEDA DEFINE PROG(TESTPR1) GROUP(PRAKT20)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE Mapset( S01SET )
  Mapset      : S01SET
  Group       : PRAKT20
  Description  ==>
  RESident    ==> No                No | Yes
  USAge       ==> Normal            Normal | Transient
  USElpacopy  ==> No                No | Yes
  Status      ==> Enabled           Enabled | Disabled
  RSl         : 00                  0-24 | Public

I New group PRAKT20 created.

                                           SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                          TIME: 00.00.00 DATE: 01.060
PF 1 HELP 2 COM 3 END                      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 23: Bestätigung der Definiton

Als nächstes wird das C-Programm definiert. Dazu drücken wir die Eingabetaste.

```

CEDA DEFINE PROG(TESTPR1) GROUP(PRAKT20)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE PROGRAM( TESTPR1 )
  PROGRAM     : TESTPR1
  Group       : PRAKT20
  Description  ==>
  Language     ==> Le370            CObol | Assembler | Le370 | C | Pli
  REload      ==> No                No | Yes
  RESident    ==> No                No | Yes
  USAge       ==> Normal            Normal | Transient
  USElpacopy  ==> No                No | Yes
  Status      ==> Enabled           Enabled | Disabled
  RSl         : 00                  0-24 | Public
  CEDf        ==> Yes                Yes | No
  Datalocation ==> Below            Below | Any
  EXECKey     ==> User              User | Cics
  COncurrency ==> Quasirent         Quasirent | Threadsafe
REMOTE ATTRIBUTES
  DYNAMIC     ==> No                No | Yes
+ REMOTESystem ==>

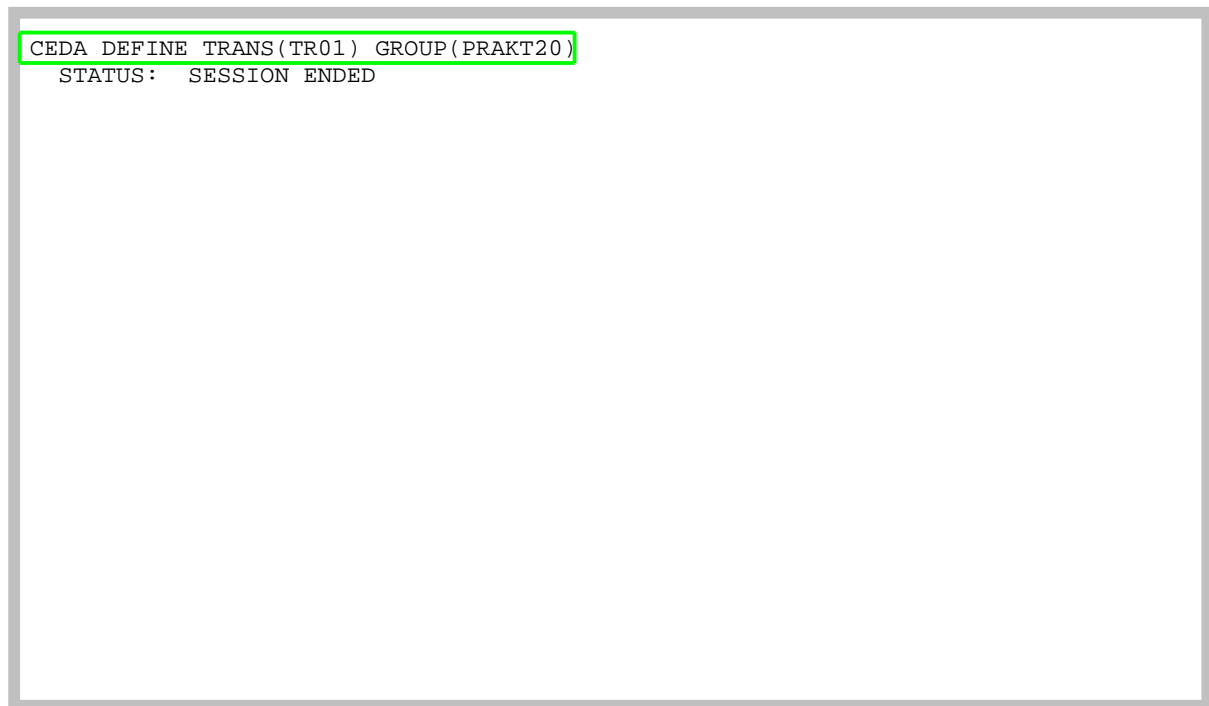
                                           SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                          TIME: 00.00.00 DATE: 01.060
PF 1 HELP 2 COM 3 END                      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 24: Auswahl der Parameter

Le370 wird als Language eingegeben; sie ist aber eigentlich eine Entwicklungsumgebung (s. Abbildung 24)

Auch hier bestätigen wir mit der Eingabetaste.
Die Nachricht "DEFINE SUCCESSFUL" erscheint, wir waren also erfolgreich und beenden diesen Screen mit der F3-Taste.

A screenshot of a terminal window with a grey border. The text inside is:

```
CEDA DEFINE TRANS (TR01) GROUP (PRAKT20)
STATUS: SESSION ENDED
```

 The first line is highlighted with a green rectangular box.

Abbildung 25: Sitzung beendet

Als letztes müssen wir die Bezeichnung der neuen Transaktion definieren. Wir wählen auch hierfür den Namen "TR01". Es könnte natürlich auch ein beliebiger anderer Name sein, solange er aus 4 Zeichen besteht. Wir geben das Kommando "CEDA DEFINE TRANS(TR01) GROUP(PRAKT20)" ein (s. Abbildung 25) und bestätigen mit der Eingabetaste.

```

DEFINE TRANS(TR01) GROUP(PRAKT20)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE TRANSAction( TR01 )
TRANSAction ==> TR01
Group       ==> PRAKT20
Description ==>
PROGRAM    ==> TESTPR1
TWasize    ==> 00000                0-32767
PROFile    ==> DFHCICST
PARTitionset ==>
STatus     ==> Enabled              Enabled | Disabled
PRIMedsize : 00000                0-65520
TASKDATAloc ==> Below              Below | Any
TASKDATAKey ==> User               User | Cics
STorageclear ==> No                No | Yes
RUNaway     ==> System              System | 0 | 500-2700000
SHutdown    ==> Disabled            Disabled | Enabled
ISolate     ==> Yes                 Yes | No
Brexite     ==>
+ REMOTE ATTRIBUTES
S PROGRAM OR REMOTESYSTEM MUST BE SPECIFIED.
                                           SYSID=C001 APPLID=A06C001

PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 26: Auswahl des Programms

In die Zeile "PROGram" geben wir nun "TESTPR1" ein (s. Abbildung 26) und bestätigen dies mit der Eingabetaste.

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE TRANSAction( TR01 )
TRANSAction : TR01
Group       : PRAKT20
Description ==>
PROGRAM    ==> TESTPR1
TWasize    ==> 00000                0-32767
PROFile    ==> DFHCICST
PARTitionset ==>
STatus     ==> Enabled              Enabled | Disabled
PRIMedsize : 00000                0-65520
TASKDATAloc ==> Below              Below | Any
TASKDATAKey ==> User               User | Cics
STorageclear ==> No                No | Yes
RUNaway     ==> System              System | 0 | 500-2700000
SHutdown    ==> Disabled            Disabled | Enabled
ISolate     ==> Yes                 Yes | No
Brexite     ==>
+ REMOTE ATTRIBUTES
                                           SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                               TIME: 00.00.00 DATE: 01.060
PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 27: Definition der Transaktion

"DEFINE SUCCESSFUL" erscheint, also war die Definition erfolgreich, wir beenden sie mit der F3-Taste (s. Abbildung 27).

```

CEDA INSTALL GROUP(PRAKT20)
STATUS:  SESSION ENDED

```

Abbildung 28: Installation der Gruppe

Nachdem die BMS-MAP, das C-Programm und die Transaktionsbezeichnung definiert worden sind, wird nun alles in unserer Gruppe "PRAKT20" installiert. Dazu geben wir den Befehl "CEDA INSTALL GROUP(PRAKT20)" ein (s. Abbildung 28) und bestätigen mit der Eingabetaste.

```

INSTALL GROUP(PRAKT20)
OVERTYPE TO MODIFY
CEDA Install
All
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Engmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROcesstype ==>
PROFile ==>
PROGram ==>
+ Requestmodel ==>

SYSID=C001 APPLID=A06C001
INSTALL SUCCESSFUL TIME: 00.00.00 DATE: 01.060
PF 1 HELP 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 29: Installation war erfolgreich

Die erfolgreiche Installation der Gruppe "PRAKT20" zeigt die Ausgabe "INSTALL SUCCESSFUL" (s. Abbildung 29) an. Wir beenden diese Installation, indem wir die F3-Taste drücken.



Abbildung 30: Aufruf der Transaktion

In Tutorial 3 waren wir mit der Definition und Installation unserer Transaktion fertig. Wir versuchen es einmal, indem wir unsere Transaktion mit der Bezeichnung "TR01" aufrufen. Dazu tragen wir den Namen in die CICS-Kommandozeile ein (s. Abbildung 30) und bestätigen mit der Eingabetaste.

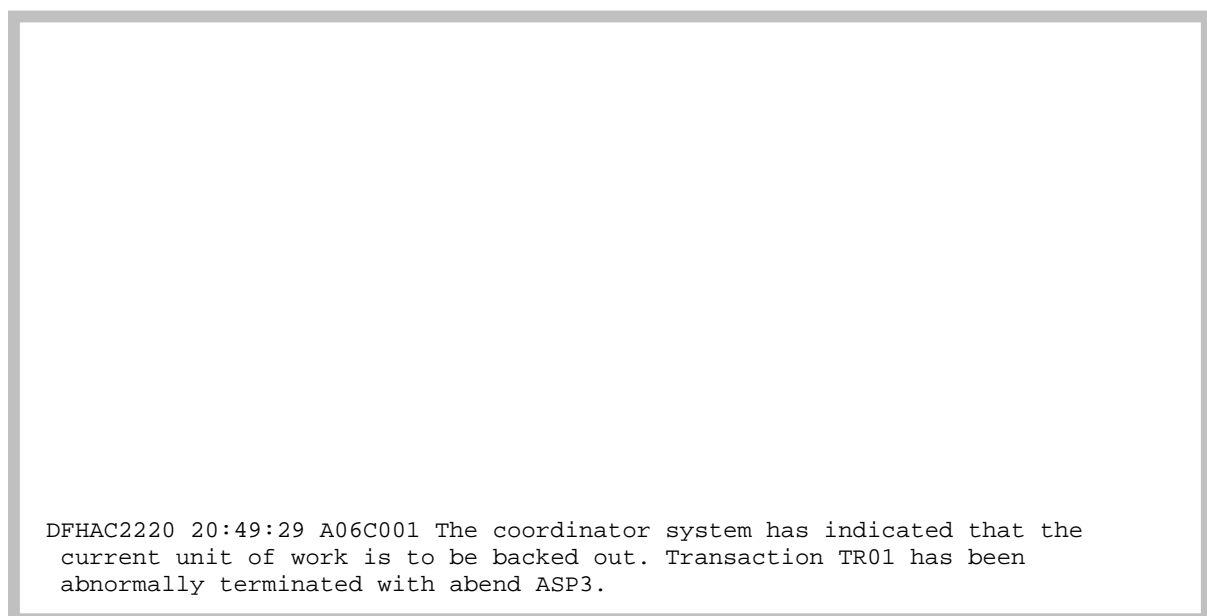


Abbildung 31: Fehlermeldung

Wir erhalten eine Fehlermeldung (s. Abbildung 31).

Die Beschreibung zur Fehlermeldung ASP3 findet sich in einem Online-Handbuch:

<http://www.s390.ibm.com/bookmgr-cgi/bookmgr.cmd/BOOKS/DFHWG400/2%2e3%2e606?ACTION=MATCHES&REQUEST=asp3&TYPE=FUZZY&SHELF=&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT>

Wir suchen nach dem Fehlercode ASP3 und finden den folgenden Eintrag:

Explanation: The abnormal termination occurs because a remote system on which the unit of work depends fails to take a syncpoint. The transaction cannot commit its changes until all coupled systems to which function has been transmitted also commit. This may be because the syncpoint protocol for transaction to transaction has been violated by failing to be in send mode for all sessions for which syncpoint has not been received.

User Response:

Check why the remote system failed to respond to the request.

Wir erinnern uns: TSO, CICS und DB2 sind alle separate OS/390-Subsysteme, die in getrennten virtuellen Adressräumen laufen. Die CICS-Group "PRAKT20" benötigt eine Definition unserer Datenbank und Datenbanktabelle.

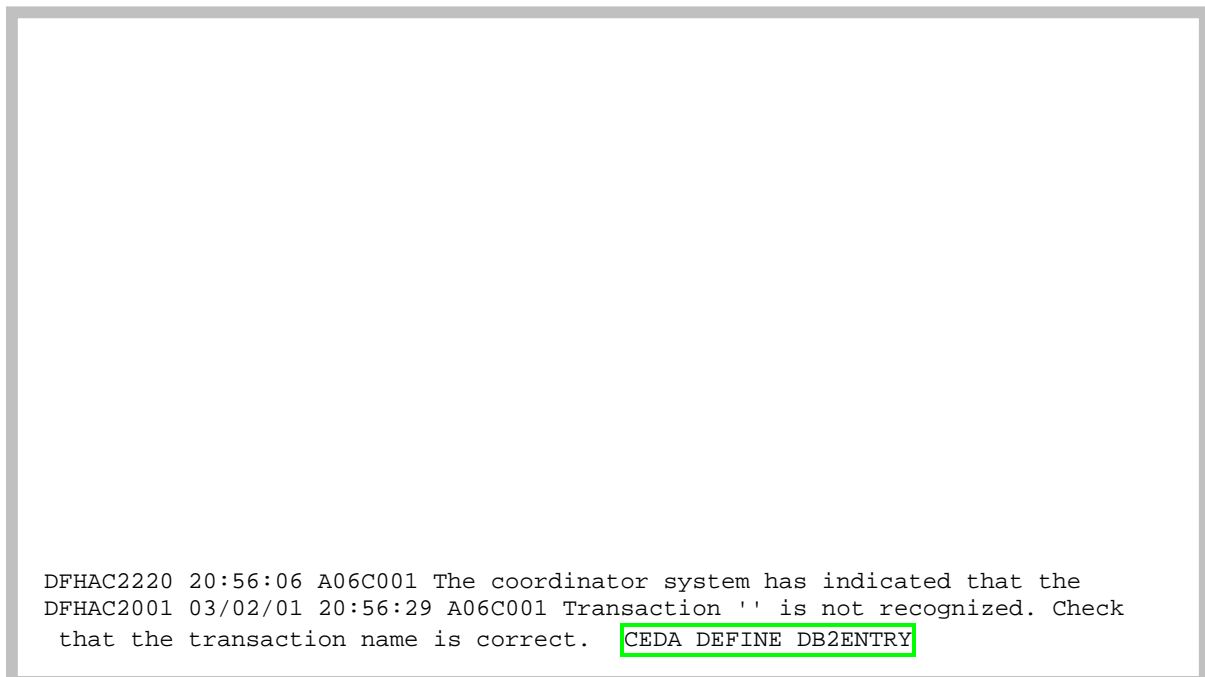


Abbildung 32: Aufruf der Definition der Datenbank

Die Definition erfolgt mit dem Kommando "CEDA DEFINE DB2ENTRY" (s. Abbildung 32).

```

DEFINE DB2ENTRY
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE DB2Entry(                               )
  DB2Entry    ==>
  Group       ==>
  Description  ==>
THREAD SELECTION ATTRIBUTES
  TRansid     ==>
THREAD OPERATION ATTRIBUTES
  ACcountrec  ==> None           None | TXid | TAsk | Uow
  AUTHid      ==>
  AUTHType    ==>               Userid | Opid | Group | Sign | TTerm
                                   | TX
  DRollback   ==> Yes           Yes | No
  PLAN        ==>
  PLANExitname ==>
  PRIority    ==> High           High | Equal | Low
  PROtectnum  ==> 0000           0-2000
  THREADLimit ==>               0-2000
  THREADWait  ==> Pool           Pool | Yes | No
MESSAGES: 2 SEVERE
                                           SYSID=C001 APPLID=A06C001

PF 1 HELP 2 COM 3 END                      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 33: DEFINE DB2ENTRY-Panel

Nachdem wir die Eingabetaste gedrückt haben, erscheint das "DEFINE DB2ENTRY-Panel" (s. Abbildung 33). Wir müssen die fehlenden Angaben eintragen (s. Abbildung 34 und 35).

```

define DB2ENTRY
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE DB2Entry(                               )
  DB2Entry    ==> TR01
  Group       ==> PRAKT20
  Description  ==>
THREAD SELECTION ATTRIBUTES
  TRansid     ==> TR01
THREAD OPERATION ATTRIBUTES
  ACcountrec  ==> None           None | TXid | TAsk | Uow
  AUTHid      ==>
  AUTHType    ==>               Userid | Opid | Group | Sign | TTerm
                                   | TX
  DRollback   ==> Yes           Yes | No
  PLAN        ==> PR1
  PLANExitname ==>
  PRIority    ==> High           High | Equal | Low
  PROtectnum  ==> 0000           0-2000
  THREADLimit ==>               0-2000
  THREADWait  ==> Pool           Pool | Yes | No
MESSAGES: 2 SEVERE
                                           SYSID=C001 APPLID=A06C001

PF 1 HELP 2 COM 3 END                      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 34: Eingabe der Parameter

Wir bezeichnen den DB2- Zugriff (DB2Entry) mit dem Namen "TR01". Das Ganze wird Teil der Gruppe "PRAKT20". Unsere Transactions-ID (TRansID) ist "TR01". Erinnern wir uns: Wir hatten ein JCL-Script "STARTSQL" erstellt, das unser C-Programm übersetzte. In diesem Script definierten wir an zwei Stellen einen Zeiger auf unsere Datenbanktabelle (Plan)

mit dem Namen "PR1". Hier wird jetzt für CICS die Verknüpfung zu der Datenbanktabelle hergestellt.

```

define DB2ENTRY
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE DB2Entry(                             )
  DB2Entry    ==> TR01
  Group       ==> PRAKT20
  Description  ==>
THREAD SELECTION ATTRIBUTES
  TRansid     ==> TR01
THREAD OPERATION ATTRIBUTES
  ACcountrec  ==> TXid          None | TXid | TAsk | Uow
  AUTHid      ==>
  AUTHType    ==> Sign          Userid | Opid | Group | Sign | TTerm
                                   | TX
  DRollback   ==> Yes          Yes | No
  PLAN        ==> PR1
  PLANExitname ==>
  PRiority    ==> High        High | Equal | Low
  PROtectnum  ==> 0000        0-2000
  THREADLimit ==> 0003        0-2000
  THREADWait  ==> Yes         Pool | Yes | No
MESSAGES: 2 SEVERE
                                           SYSID=C001 APPLID=A06C001

PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 35: Eingabe der Parameter

Wir geben weiterhin die gekennzeichneten 4 Parameter ein und bestätigen zum Schluß mit der Eingabetaste.

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE DB2Entry( TR01                         )
  DB2Entry    : TR01
  Group       : PRAKT20
  Description  ==>
THREAD SELECTION ATTRIBUTES
  TRansid     ==> TR01
THREAD OPERATION ATTRIBUTES
  ACcountrec  ==> TXid          None | TXid | TAsk | Uow
  AUTHid      ==>
  AUTHType    ==> Userid        Userid | Opid | Group | Sign | TTerm
                                   | TX
  DRollback   ==> Yes          Yes | No
  PLAN        ==> PR1
  PLANExitname ==>
  PRiority    ==> High        High | Equal | Low
  PROtectnum  ==> 0000        0-2000
  THREADLimit ==> 0003        0-2000
  THREADWait  ==> Yes         Pool | Yes | No
                                           SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                               TIME: 00.00.00 DATE: 01.061
PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 36: Bestätigung der gelungenen Definition

Die Definition war erfolgreich und wird bestätigt durch die Ausschrift: "DEFINE SUCCESSFUL" (s. Abbildung 36).

Wir verlassen die Definition mit der F3-Taste.

```
CEDA INSTALL GROUP(PRAKT20)
STATUS:  SESSION ENDED
```

Abbildung 37: Installation der Gruppe

Diese Änderung muß wieder installiert werden. Dazu geben wir wieder den Befehl "CEDA INSTALL GROUP(PRAKT20)" (s. Abbildung 37) ein und bestätigen mit der Eingabetaste.

```
INSTALL GROUP(PRAKT20)
OVERTYPE TO MODIFY
CEDA Install
All
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROcesstype ==>
PROfile ==>
PROGram ==>
+ Requestmodel ==>

SYSID=C001 APPLID=A06C001
INSTALL SUCCESSFUL
TIME: 00.00.00 DATE: 01.061
PF 1 HELP 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 38: Installation der Gruppe

Die Ausschrift in der Abbildung 38 sagt aus, dass die Installation erfolgreich war. Wir verlassen diesen Screen wieder mit F3.



Abbildung 39: Starten der Transaktion

Wir geben den Namen unserer Transaktion "TR01" ein, um diese aufzurufen (s. Abbildung 39) und bestätigen mit der Eingabetaste.

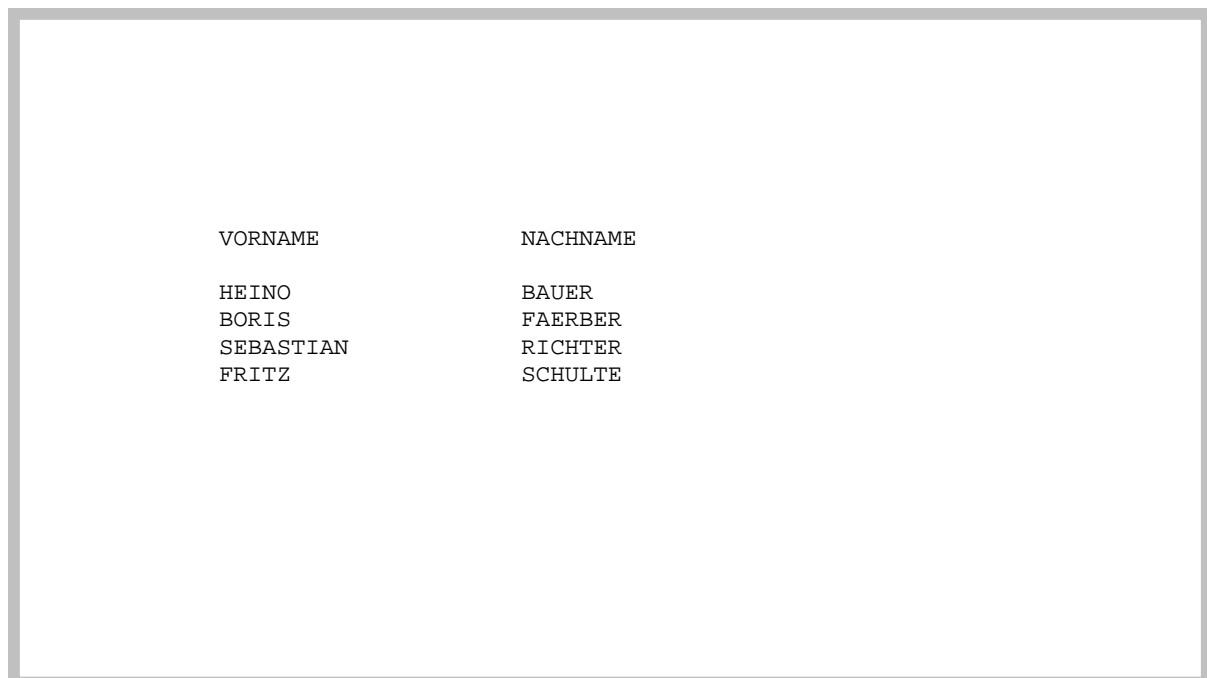


Abbildung 40: Ausgabe der Datenbanktabelle

Die korrekte Ausgabe der Datenbank erscheint auf dem Bildschirm (s. Abbildung 40). Damit ist die Aufgabe gelöst.

Aufgabe: Bereiten Sie unter CICS die Transaktion vor, die auf die DB2-Datenbank zugreift und führen Sie diese aus.