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.

Hinweis: Tutorial 5 baut auf die erfolgreiche Bearbeitung von Tutorial 4 auf.

Unser Business Logic-Programm soll dabei SQL-Aufrufe enthalten.
Diese müssen durch einen SQL-Precompiler in native DB2 API-Aufrufe übersetzt werden, ehe der C-Compiler das Business Logic-Programm übersetzen kann.

Inhalt dieses Tutorials (Vorschau)

Wir werden drei JCL-Scripte erstellen sowie diese mittels "SUB" ausführen. Des Weiteren werden wir ein C-Programm erstellen, welches EXEC SQL-Statements enthält.

CICS trennt strikt Berechnungen und Datenbankzugriffe von dem Layout der Darstellungen auf Panels. Ersteres wird als "Business Logic" und letzteres als "Presentation Logic" bezeichnet.

Das erste zu erstellende sowie auszuführende JCL-Script (Dataset-Name: PRAKT20.CICSDB2.TEST01(BMSJCL) (s. die Abbildungen 5 und 6) behandelt die Presentation Logic. Sie besteht aus genau einem Mapset "S01SET", der genau eine Map "MAP001" enthält. Ein Mapset kann aber auch mehrere Maps enthalten. Diese Map "MAP001" definiert Positionen, Länge sowie weitere Attribute der Darstellung der Daten aus der DB2-Datenbank auf dem Bildschirm.

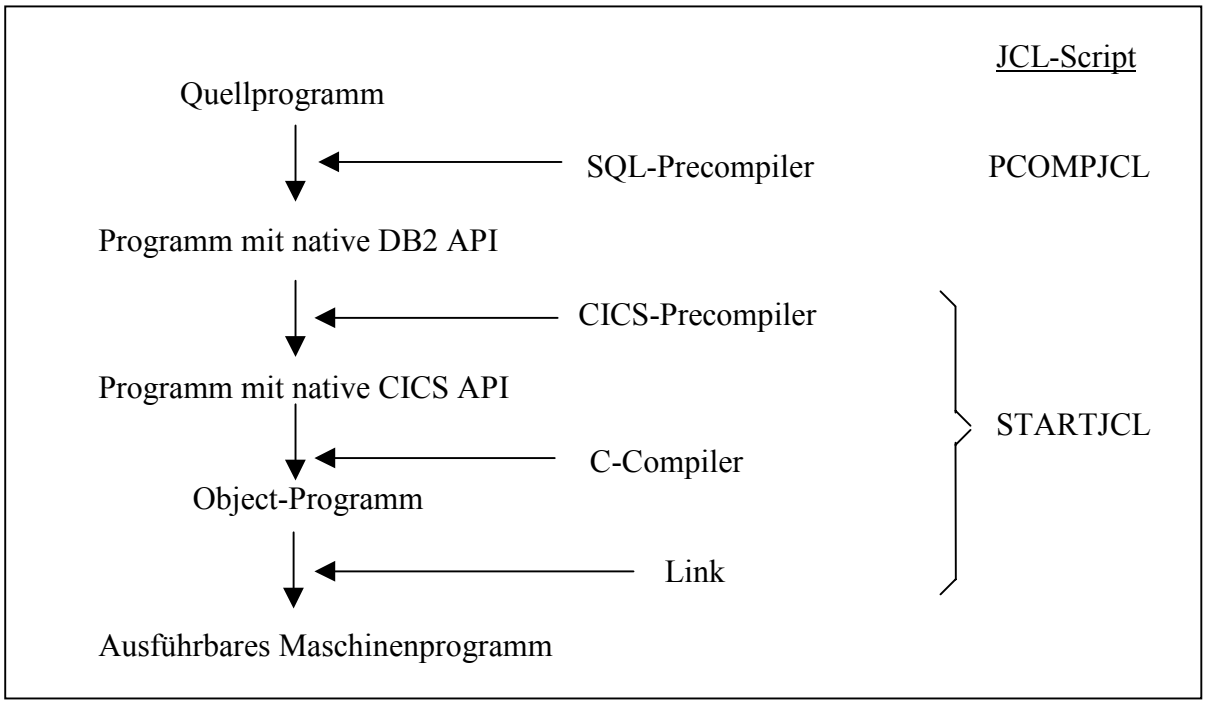


Abbildung 1:
Schritte vom C-Programm mit EXEC SQL-Statements zum ausführbaren Maschinenprogramm

Anschließend erstellen wir das C-Programm "PRAKT20.CICSDB2.TEST01(CPROG020)" (s. die Abbildungen 10 und 11), welches EXEC SQL-Statements enthält.

So wie in Abbildung 1 dargestellt, führt das zweite von uns erstellte JCL-Script "PRAKT20.CICSDB2.TEST01(PCOMPJCL)" einen Precompilerdurchlauf aus. Alle EXEC SQL-Statements im C-Programm werden durch dieses in native DB2 API-Aufrufe konvertiert.

Als drittes JCL-Script wird von uns "PRAKT20.CICSDB2.TEST01(STARTJCL)" erstellt und ausgeführt (s. auch die Abbildungen 16-18). Der CICS-Precompiler generiert aus dem C-Programm mit native *DB2* API-Aufrufen ein C-Programm mit native *CICS* API-Aufrufen. Anschließend wird der nun so entstandene C-Programmcode zu einen Objekt-Programmcode übersetzt, aus welchem der Linker ein ausführbares Maschinenprogramm erzeugt (s. auch Abbildung 1).

Alle diese Schritte werden im TSO ausgeführt. TSO ist ein Subsystem von z/OS und OS/390. Ein weiteres Subsystem von z/OS und OS/390 ist CICS. Der folgende Teil des Tutorials behandelt, über welche Schritte ein CICS-Zugriff auf die Daten unserer DB2-Datenbank möglich wird. Das Ziel ist also, den Zugriff auf die DB2-Daten durch eine selbst geschaffene CICS-Transaktion mit der Transaktions-ID "D020" auszulösen. Folgende Schritte sind dazu notwendig:

1. Definition des Mapsets mittels
   "CEDA DEFINE MAPSET(S01SET) GROUP(PRAKT20)"
2. Definition des C-Programmes mittels
   "CEDA DEFINE PROG(CPROG020) GROUP(PRAKT20)"
3. Definition des Namens der Transaktion-ID mittels
   "CEDA DEFINE TRANS(D020) GROUP(PRAKT20)"
4. Definition unserer Datenbank und Datenbanktabelle mittels
   "CEDA DEFINE DB2ENTRY"

Nach diesen Schritten sind der Mapset S01SET mit der Map MAP001, das ausführbare Maschinenprogramm, das aus CPROG020 generiert wurde, die selbst definierte Transaktion-ID "D020" sowie die Datenbank und Tabelle, aus der ausgelesen werden soll, dem CICS-System bekannt. Ebenfalls ist ihm bekannt, dass alle diese Komponenten der Gruppe "PRAKT20" zugewiesen wurden. Diese Gruppe wird durch Schritt 1. automatisch erstellt. Doch diese *Definitionen* reichen noch nicht aus. Unser Ziel erreichen wir erst, wenn alle Komponenten auch *installiert* werden. Dies geschieht durch

5. "CEDA INSTALL GROUP(PRAKT20)"

Nun haben wir unser Ziel erreicht. Geben wir "D020" unter CICS ein (s. Abbildung 40), so wird unsere selbst definierte Transaktion ausgeführt, welche die Spalten "VORNAME" und "NACHNAME" aus der im Tutorial 4 angelegten DB2-Tabelle ausliest und auf unserem Bildschirm ausgibt (s. Abbildung 41).

Hinweis:

Versuchen Sie, ein mehrmaliges Abarbeiten dieses Tutorials zu vermeiden. Denn da könnte "CEDA INSTALL GROUP(PRAKT20)" die Fehlermeldung "INSTALL UNSUCCESSFUL" produzieren. Sollte dies bei Ihnen auftreten, informieren Sie bitte Ihren Betreuer.

Tutorial einschließlich der Übungsaufgaben

Aufgabe: *Arbeiten Sie nachfolgendes Tutorial durch. Vergrößern Sie den Dataset "PRAKT20.LIB", indem Sie den alten löschen sowie den neuen mit folgenden Parametern anlegen: Primary quantity: 34 Kbyte, Secondary quantity: 8 Kbyte, Directory blocks: 2, Record format: FB, Record length: 80, Block size: 320, Data set name type : PDS*

Wir loggen uns als TSO-Benutzer ein und öffnen das DSLIST-Panel.

```
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.TEST01                                          SMS001
         PRAKT20.CICSDB2.TEST01                                       SMS001
         PRAKT20.DBRMLIB.DATA                                         SMS001
         PRAKT20.ISPF.ISPPROF                                         SMS001
         PRAKT20.LIB                                                  SMS001
         PRAKT20.SPFLOG1.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 ===>                                                 Scroll ===> PAGE
 F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 2: Der DSLIST-Panel

Wir hatten im vorangegangenen Tutorial alle hierfür erforderlichen Partitioned Datasets angelegt. Unser DSLIST-Panel zeigt 11 Partitioned Datasets (s. Abbildung 2). "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-Precompilers automatisch gefüllt. "PRAKT20.CICSDB2.TEST01" nimmt die von uns zu erstellenden Quellprogramme auf.

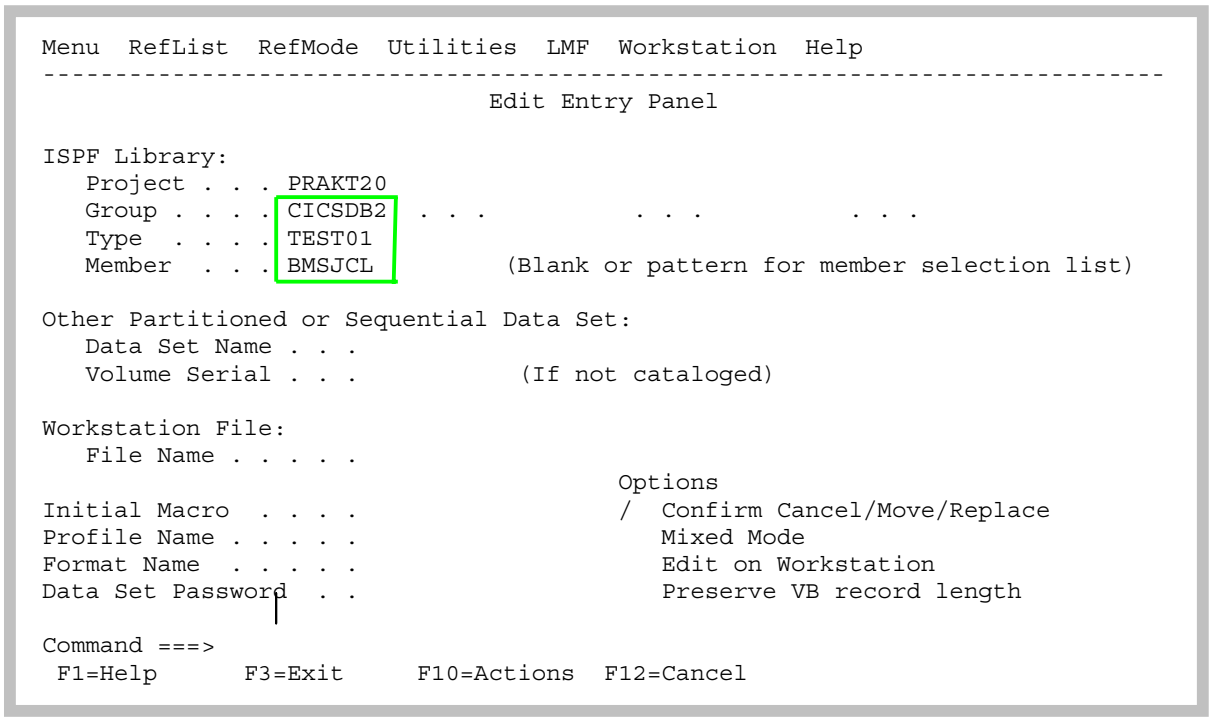


Abbildung 3: Anlegen des Members "BMSJCL"

Mit Hilfe des "Edit Entry Panels" erstellen wir einen neuen Member "BMSJCL" (s. Abbildung 3) und bestätigen anschließend mit der Eingabetaste.

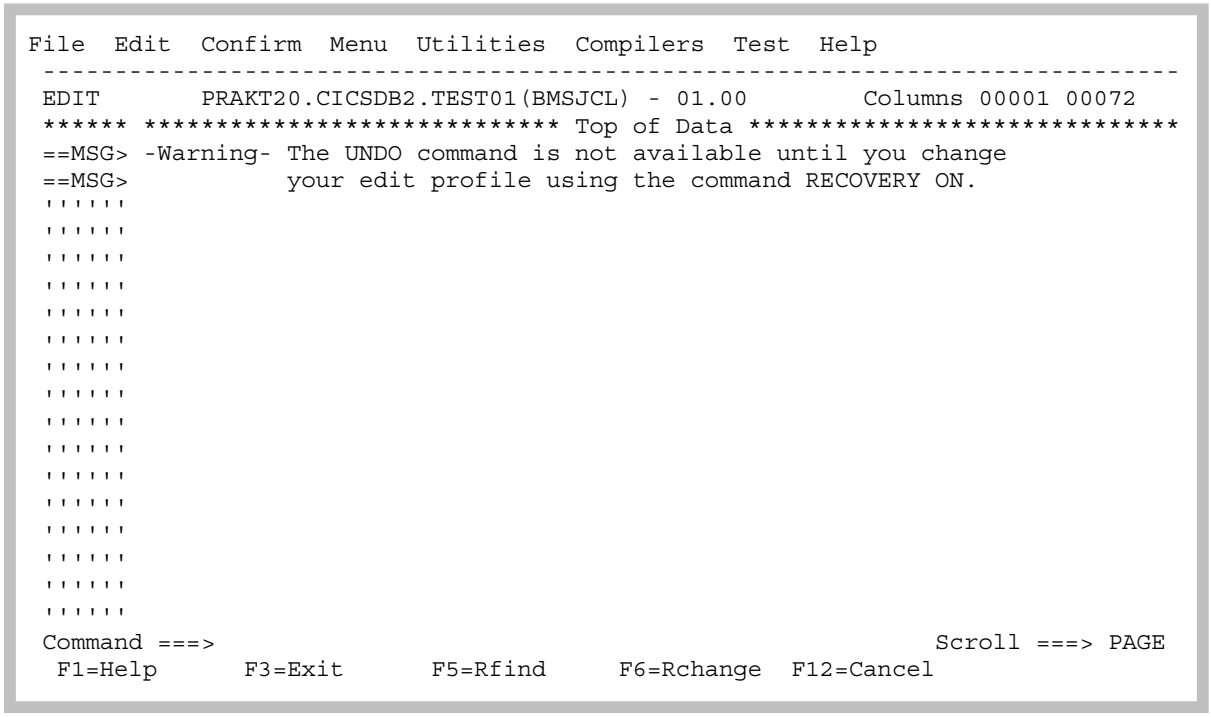


Abbildung 4: Der leere Edit-Entry-Panel

Ein leeres Edit-Entry-Panel erscheint (s. Abbildung 4). 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 (BMSJCL).

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAKT20.CICSDB2.TEST01(BMSJCL) - 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 //PRAKT20B 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 MAP001  DFHMDI 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 VNAM1   DFHMDF POS=(11,13),ATTRB=(ASKIP,NORM),LENGTH=20
000012 NNAM1   DFHMDF POS=(11,34),ATTRB=(ASKIP,NORM),LENGTH=20
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 5: 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(BMSJCL) - 01.01        Columns 00001 00072
000013 VNAM2   DFHMDF POS=(12,13),ATTRB=(ASKIP,NORM),LENGTH=20
000014 NNAM2   DFHMDF POS=(12,34),ATTRB=(ASKIP,NORM),LENGTH=20
000015 VNAM3   DFHMDF POS=(13,13),ATTRB=(ASKIP,NORM),LENGTH=20
000016 NNAM3   DFHMDF POS=(13,34),ATTRB=(ASKIP,NORM),LENGTH=20
000017 VNAM4   DFHMDF POS=(14,13),ATTRB=(ASKIP,NORM),LENGTH=20
000018 NNAM4   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 6: 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.

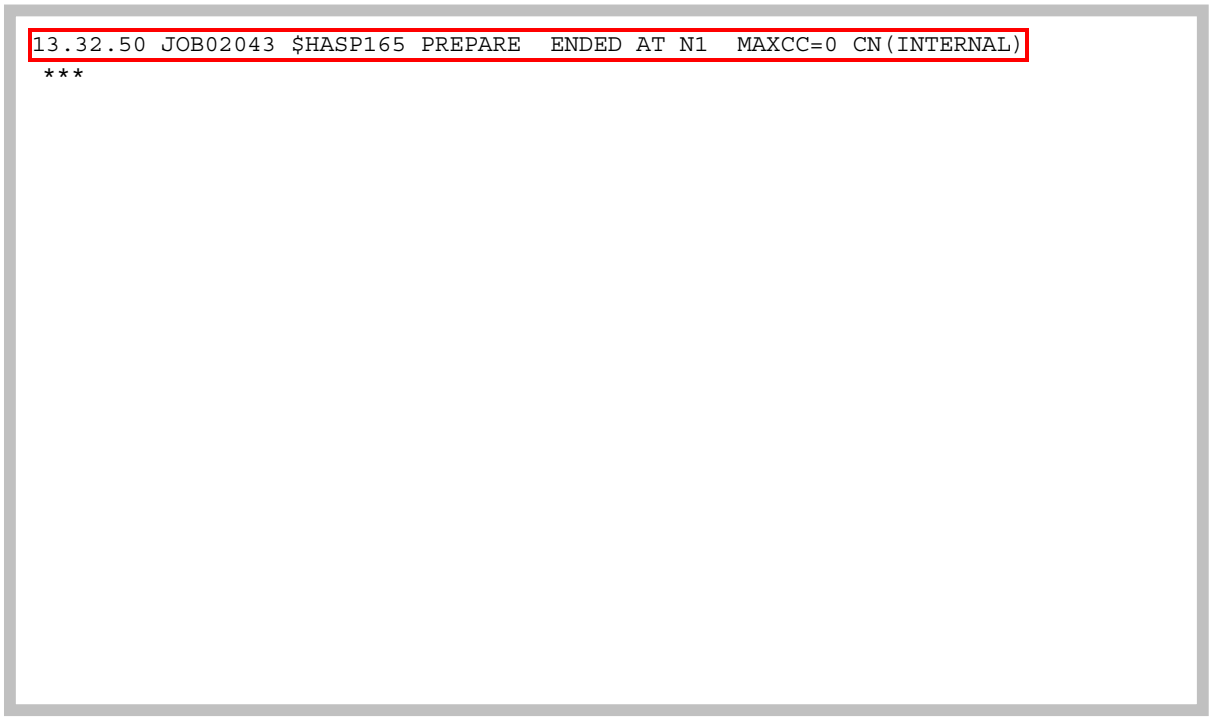


Abbildung 7: Bestätigung der Jobverarbeitung

Wir warten, bis "JES" unser BMS-Programm übersetzt hat (30-60 Sekunden). Durch das Betätigen der Eingabetaste erscheint das hier gezeigte Panel (s.Abbildung 7). "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. Ersetzen Sie "//PRAKT20B" entsprechend ihres Mainframe-Accountnamens.*

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 der während der Übersetzung erstellte Member "PRAKT20.LIB(S01SET)". Wir schauen uns "PRAKT20.LIB(S01SET)" an. Dieser Member könnte, je nach eingestellter Host-Code-Page, auch leicht modifiziert auf dem Bildschirm dargestellt sein.

```
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";
       } map001i;

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";
       } map001o;

} map001;
```

Dies ist der Code von "PRAKT20.LIB(S01SET)". Er erstreckt sich über 4 Panels. Der Member enthält eine "Union", die aus 2 "Structures" besteht. Wir verwenden es als Vorlage (Template) für die von uns als C-Programm zu erstellende Business Logic.

Wir rufen erneut das Edit-Entry-Panel auf.

```
Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel

ISPF Library:
   Project . . . PRAKT20
   Group . . . . CICSDB2  . . .          . . .          . . .
   Type  . . . . TEST01
   Member  . . . CPROG020         (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                        Options
Initial Macro  . . . .                  /  Confirm Cancel/Move/Replace
Profile Name . . . . .                     Mixed Mode
Format Name  . . . . .                     Edit on Workstation
Data Set Password  . .                     Preserve VB record length

Command ===>
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Abbildung 8: Anlegen des Members "CPROG020"

Wir legen ein weiteres Member "PRAKT20.CICSDB2.TEST01(CPROG020)" an (s. Abbildung 8). Es soll unser Business Logic-Programm aufnehmen.

Wir bestätigen mit der Eingabetaste.

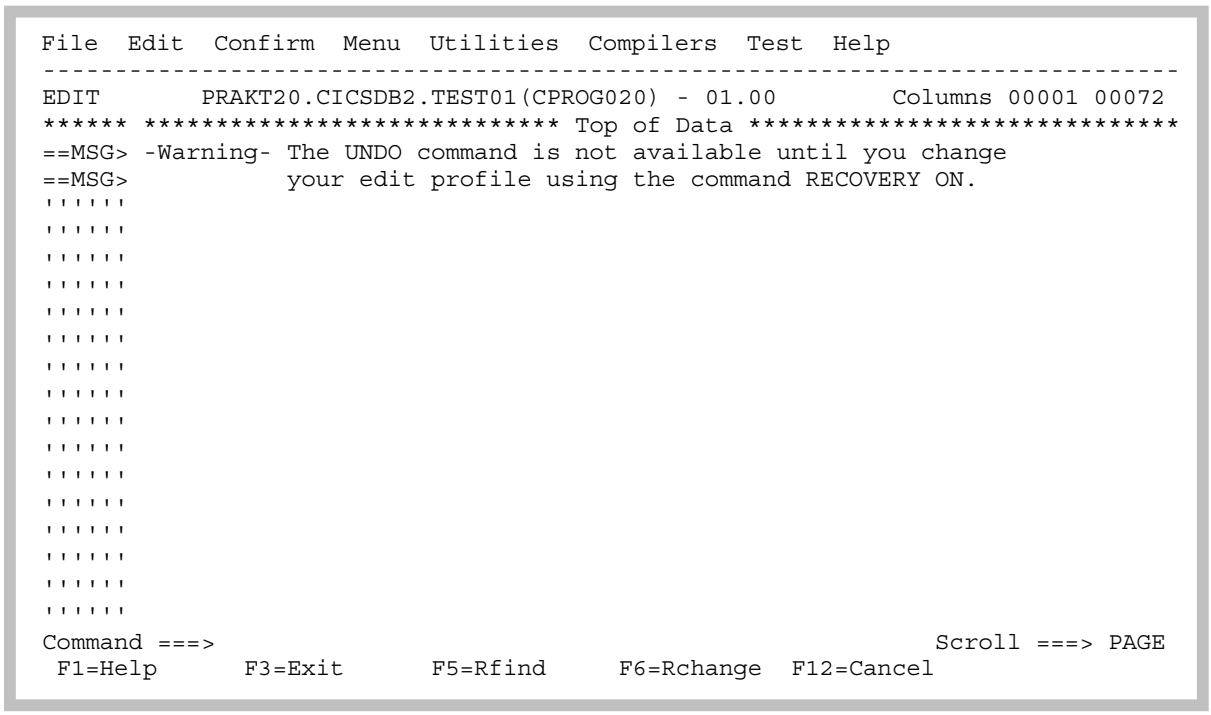


Abbildung 9: Das leere Edit-Entry-Panel

Ein leeres "Edit Entry Panel", wie in Abbildung 9, erscheint.

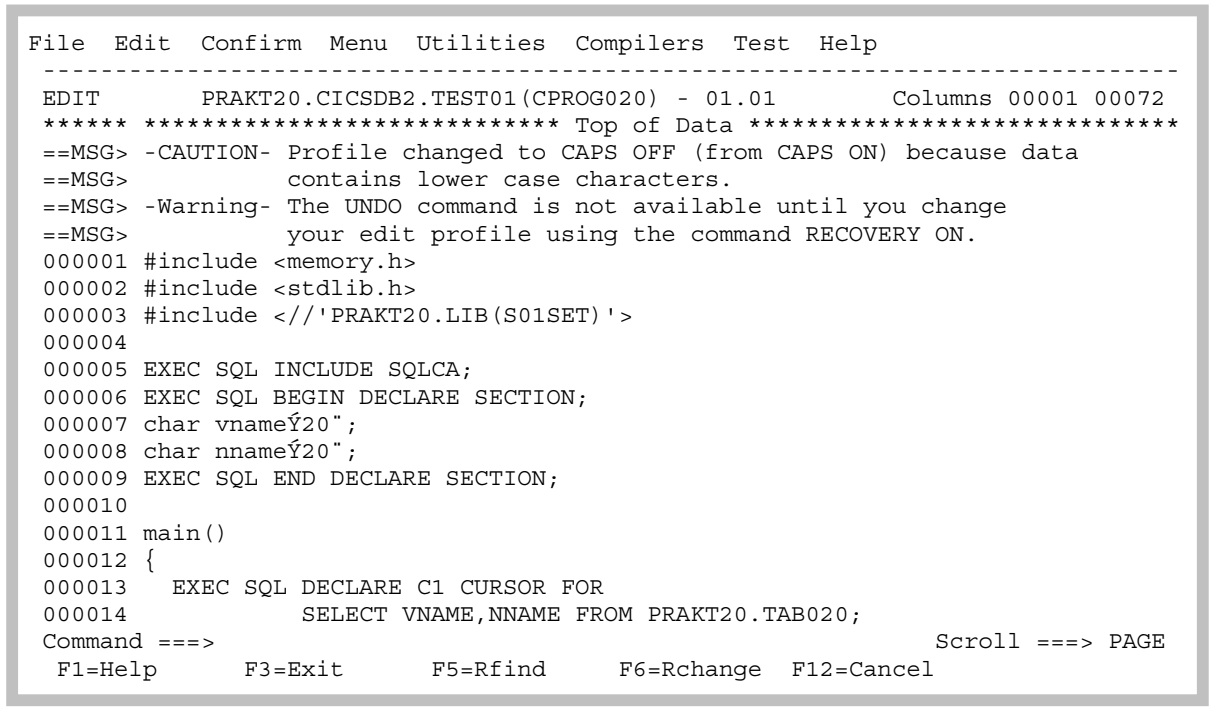


Abbildung 10: Der erste Teil des Business Logic-Programms

Dies ist das vollständige C-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(CPROG020) - 01.01        Columns 00001 00072
000015   EXEC SQL OPEN C1;
000016   EXEC SQL FETCH C1 INTO :vname, :nname;
000017   memcpy(map001.map001i.vnam1i,vname,20);
000018   memcpy(map001.map001i.nnam1i,nname,20);
000019   EXEC SQL FETCH C1 INTO :vname, :nname;
000020   memcpy(map001.map001i.vnam2i,vname,20);
000021   memcpy(map001.map001i.nnam2i,nname,20);
000022   EXEC SQL FETCH C1 INTO :vname, :nname;
000023   memcpy(map001.map001i.vnam3i,vname,20);
000024   memcpy(map001.map001i.nnam3i,nname,20);
000025   EXEC SQL FETCH C1 INTO :vname, :nname;
000026   memcpy(map001.map001i.vnam4i,vname,20);
000027   memcpy(map001.map001i.nnam4i,nname,20);
000028   EXEC SQL CLOSE C1;
000029
000030   EXEC CICS SEND MAP("map001") MAPSET("s01set") ERASE;
000031
000032
000033 }
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 11: Der zweite Teil des Business Logic-Programms

In Zeile 000007 und Zeile 000008 von "PRAKT20.CICSDB2.TEST01(CPROG020)" 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.

Während der Übersetzung des Mapsets PRAKT20.CICSDB2.TEST01(BMSJCL) wurde ein Member "S01SET" im Dataset "PRAKT20.LIB" erstellt, der ein Template für unser C-Programm enthält. Im Tutorial 3 hatten wir das Template manuell in unser C-Programm kopiert. Im vorliegenden Fall gehen wir anders vor.

Zeile 3 unseres C-Programms enthält das Statement:
#include <//'PRAKT20.LIB(S01SET)'>
Es tritt an Stelle des manuellen Kopiervorgangs.

Nach Fertigstellung des Programms kehren wir zum Edit-Entry-Panel zurück.

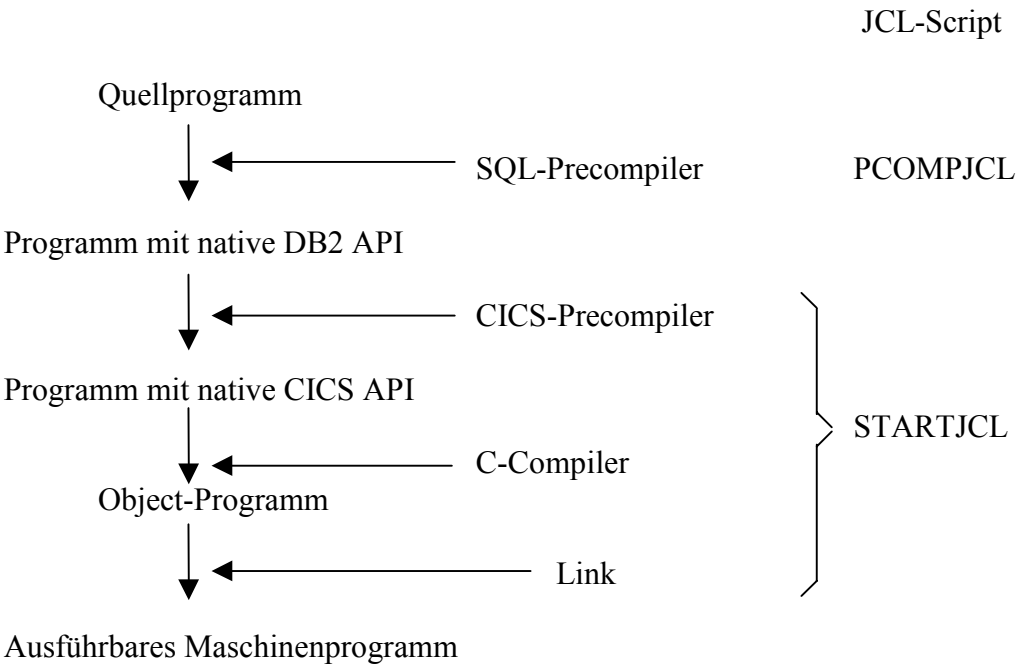
Aufgabe: Erstellen Sie das Member und schreiben Sie das C-Programm hinein.

```
Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel

ISPF Library:
   Project . . . PRAKT20
   Group . . . . CICSDB2  . . .          . . .          . . .
   Type  . . . . TEST01
   Member  . . . PCOMPJCL        (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                       Options
Initial Macro  . . . .                 /  Confirm Cancel/Move/Replace
Profile Name . . . . .                    Mixed Mode
Format Name  . . . . .                    Edit on Workstation
Data Set Password  . .                    Preserve VB record length

Command ===>
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Abbildung 12: Anlegen des Members PCOMPJCL

Ehe dieses Programm mit Hilfe des C-Compilers übersetzt werden kann, sind 2 Precompiler-Läufe erforderlich. Der erste Precompiler-Lauf übersetzt alle EXEC SQL-Statements in native DB2 API-Aufrufe.

Wir erstellen ein neues Member "PCOMPJCL" (s. Abbildung 12) zur Aufnahme eines JCL-Scripts, das den SQL-Precompiler aufruft.



Wir drücken anschließend die Eingabetaste.

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAKT20.CICSDB2.TEST01(PCOMPJCL) - 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 //PRAKT20P 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(CPROG020),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(CPROG020)
****** **************************** Bottom of Data ****************************
Command ===> SUB                                            Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 13: Das JCL-Script

Wir erstellen das in Abbildung 13 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(CPROG020)" 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.

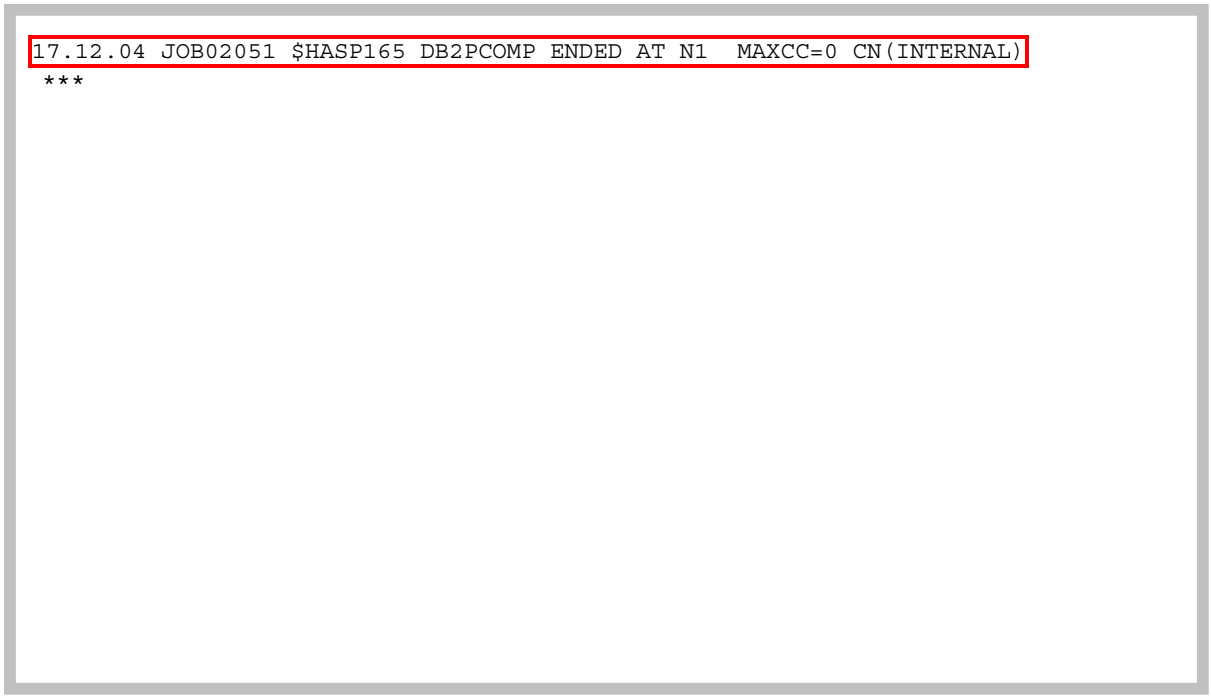


Abbildung 14: Bestätigung der Jobverarbeitung

"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. Denken Sie daran, den Jobnamen ' PRAKT20P ' wieder an ihren Mainframe-Accountnamen anzupassen.*

```
Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel

ISPF Library:
   Project . . . PRAKT20
   Group . . . . CICSDB2  . . .          . . .          . . .
   Type  . . . . TEST01
   Member  . . . STARTJCL        (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                       Options
Initial Macro  . . . .                 /  Confirm Cancel/Move/Replace
Profile Name . . . . .                    Mixed Mode
Format Name  . . . . .                    Edit on Workstation
Data Set Password  . .                    Preserve VB record length

Command ===>
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Abbildung 15: Anlegen des Members "STARTJCL"

Als nächstes erstellen wir ein neues Member "STARTJCL" zur Aufnahme eines JCL-Scripts, das folgende Funktionen aufruft:

- den CICS-Precompiler
- den C-Compiler
- den Linker

Anschließend betätigen wir die Eingabetaste.

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAKT20.CICSDB2.TEST01(STARTJCL) - 01.04       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 //PRAKT20S JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000002 //             TIME=1440
000003 //**************************
000004 //* TRANSL/COMP/LINKEDIT
000005 //**************************
000006 //COMP     EXEC PROC=CTOCICS,REG=0M
000007 //TRN.SYSIN DD DISP=SHR,DSN=PRAKT20.CICSDB2.TEST01(OUT)
000008 //LKED.SYSIN DD *
000009       INCLUDE DB2LOAD(DSNCLI)
000010       NAME CPROG020(R)
000011 //*********************
000012 //* BIND
000013 //*********************
000014 //BIND     EXEC PGM=IKJEFT01
000015 //STEPLIB  DD DISP=SHR,DSN=DSN510.SDSNEXIT
000016 //         DD DISP=SHR,DSN=DSN510.SDSNLOAD
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 16: Das JCL-Script (Panel #1)

"STARTJCL" 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(STARTJCL) - 01.04       Columns 00001 00072
000017 //DBRMLIB  DD DISP=OLD,DSN=PRAKT20.DBRMLIB.DATA(CPROG020)
000018 //SYSPRINT DD SYSOUT=*
000019 //SYSTSPRT DD SYSOUT=*
000020 //SYSUDUMP DD SYSOUT=*
000021 //SYSTSIN  DD *
000022   DSN S(DBA1)
000023   BIND PLAN(PR1) MEMBER(CPROG020) 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 ===>                                                Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 17: Das JCL-Script (Panel#2)

Panel #2: Der SQL-Precompiler-Lauf hat im Dataset "PRAKT20.DBRMLIB.DATA" ein Member "CPROG020" angelegt (Zeile 17).

Die Ausführung unseres Programms "CPROG020" 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(STARTJCL) - 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 18: 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.

```
//PRAKT20S 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 CPROG020(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(CPROG020)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN  DD *
  DSN S(DBA1)
  BIND PLAN(PR1) MEMBER(CPROG020) 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.

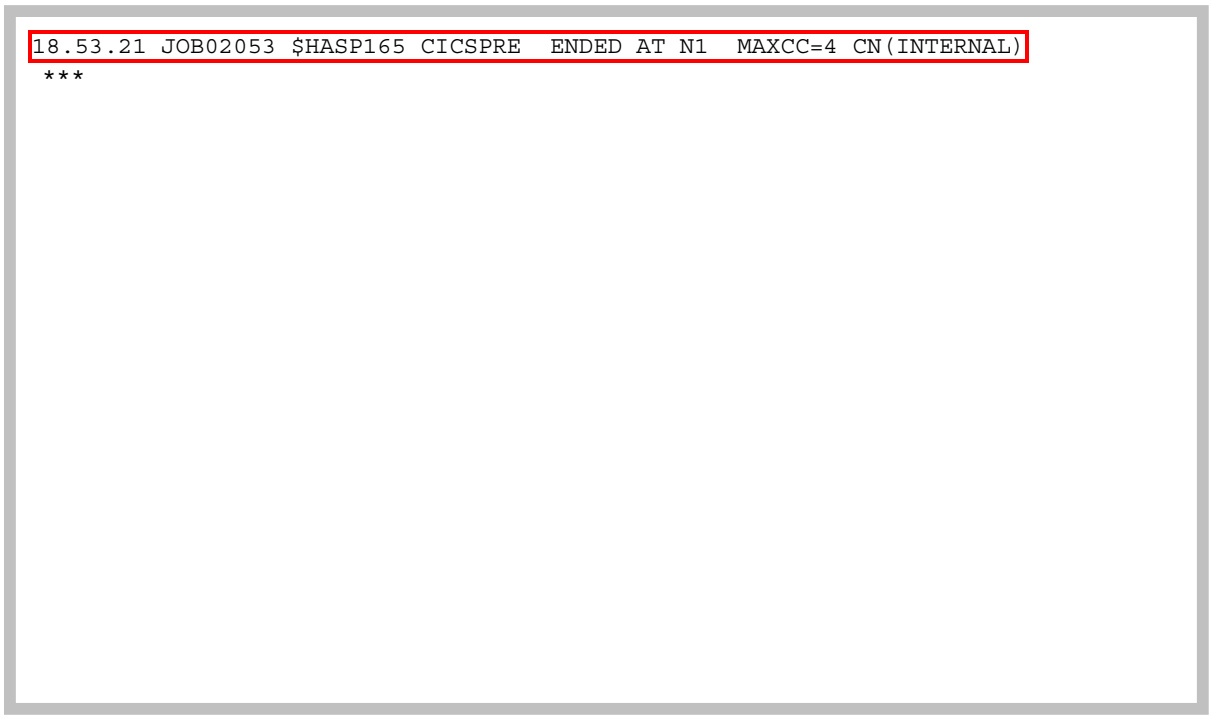


Abbildung 19: Ausgabe der Jobverarbeitung

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

Aufgabe: *Erstellen Sie einen neuen Member, legen Sie das JCL-Script an (mit an Ihren Accountnamen angepaßtem Jobnamen) 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 ME  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
              SSSSSSS  //   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 20: Der Login-Screen

Wir loggen uns mit "L C001" ein (s. Abbildung 20) und bestätigen mit der Eingabetaste.

```
                          Signon to CICS                        APPLID A06C001

--------------- WELCOME AT UNIVERSITY OF LEIPZIG ---------------        -JEDI-
BITTE TRANSAKTION <CESF LOGOFF> ZUM AUSLOGGEN BENUTZEN!                 -CICS-



Type your userid and password, then press ENTER:

         Userid . . . . PRAKT20     Groupid . . .
         Password . . . *******
         Language . . .

     New Password . . .









DFHCE3520 Please type your userid.
F3=Exit
```

Abbildung 20a: Signon to CICS-Screen

Wir müssen uns unter CICS mit der gleichen Userid wie unter TSO einloggen (s. Abbildung 20a). Auch unser TSO-Paßwort ist in dieses Panel einzugeben. Durch das Betätigen der Eingabetaste kommt man in den nächsten Screen.

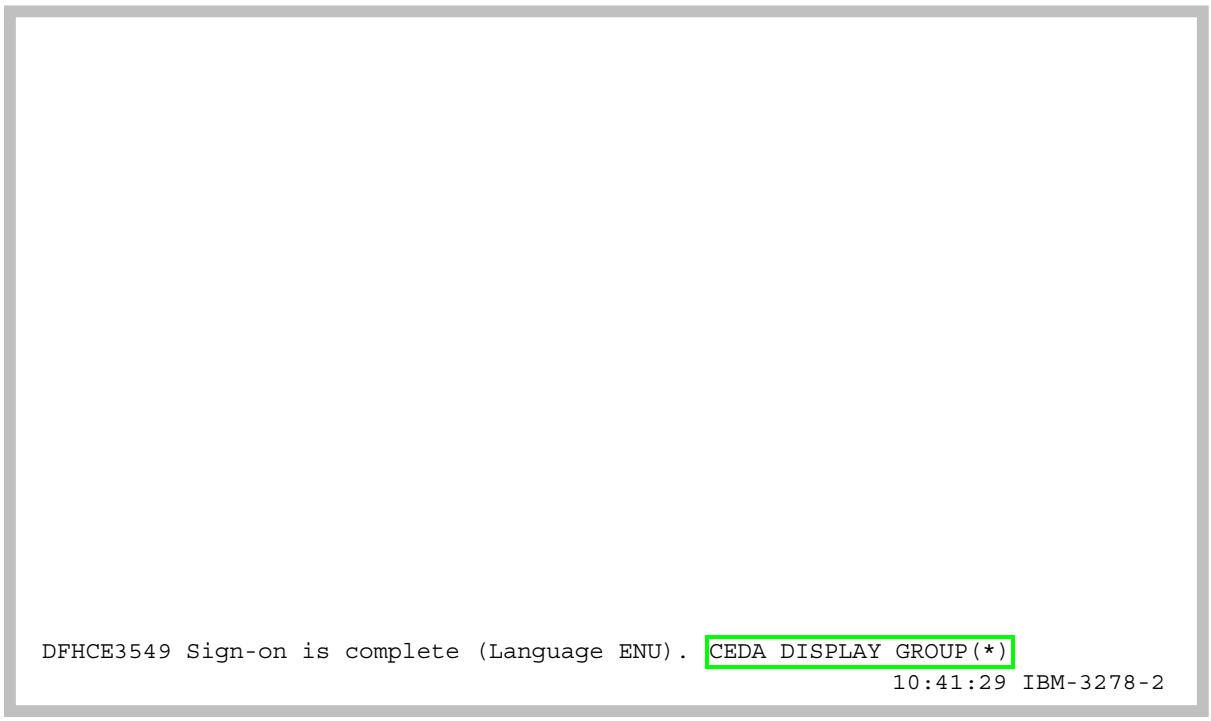


Abbildung 21: Einloggvorgang ist abgeschlossen

Wir betätigen die Tab-Taste, so dass der Cursor auf die letzte Zeile springt.
Hier geben wir den "CEDA DISPLAY GROUP(*)"-Befehl ein und bestätigen anschließend mit der Eingabetaste (s. Abbildung 21).

```
 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

                                                    SYSID=C001 APPLID=A06C001
  RESULTS: 1 TO 17                                TIME:  00.00.00  DATE: 01.060
PF 1 HELP       3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 22: Die bestehenden Gruppen

Der "CEDA DISPLAY GROUP(*)"-Befehl zeigt alle bisher vorhandenen Gruppen an (s. Abbildung 22).

Wir definieren zunächst unser BMS-Programm mit dem Namen "S01SET" für die neue Group "PRAKT20" und bestätigen anschließend dreimal die Eingabetaste.

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

```
CEDA DEFINE MAPSET(S01SET) 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: Definition von S01SET

Die Definition war erfolgreich und die neue Gruppe wurde erstellt.

```
 CEDA DEFINE PROG(CPROG020) 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 24: Bestätigung der Definition

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

```
CEDA DEFINE PROG(CPROG020) GROUP(PRAKT20)
  OVERTYPE TO MODIFY                                      CICS RELEASE = 0530
   CEDA  DEFine PROGram( CPROG020  )
    PROGram        : CPROG020
    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 25: Auswahl der Parameter

Le370 wird bei Language eingeben; sie ist aber eigentlich eine Entwicklungsumgebung (s. Abbildung 25)

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

Abbildung 26: Sitzung beendet

Als letztes müssen wir die Bezeichnung der neuen Transaktion definieren. Wir wählen auch hierfür den Namen "D020". Es könnte natürlich auch ein beliebiger anderer Name sein, solange er aus 4 Zeichen besteht. Wir geben das Kommando "CEDA DEFINE TRANS(D020) GROUP(PRAKT20)" ein (s. Abbildung 26) und bestätigen mit der Eingabetaste.

```
 DEFINE TRANS(D020) GROUP(PRAKT20)
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
  CEDA  DEFine TRANSaction( D020 )
   TRANSaction  ==> D020
   Group        ==> PRAKT20
   DEscription  ==>
   PROGram      ==> CPROG020
   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
   Brexit       ==>
+ 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 27: Auswahl des Programms

In die Zeile "PROGram" geben wir nun "CPROG020" ein (s. Abbildung 27) und bestätigen dies mit der Eingabetaste.

```
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
  CEDA  DEFine TRANSaction( D020 )
   TRANSaction    : D020
   Group          : PRAKT20
   DEscription  ==>
   PROGram      ==> CPROG020
   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
   Brexit       ==>
+ 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 28: Definition der Transaktion

"DEFINE SUCCESSFUL" erscheint, also war die Definition erfolgreich, wir beenden sie mit der F3-Taste (s. Abbildung 28).

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

Abbildung 29: 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 29) 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: 01.060
 PF 1 HELP       3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 30: Installation war erfolgreich

Die erfolgreiche Installation der Gruppe "PRAKT20" zeigt die Ausgabe "INSTALL SUCCESSFUL" (s. Abbildung 30) an. Wir beenden diese Installation, indem wir die F3-Taste drücken.

```
D020
  STATUS:  SESSION ENDED
```

Abbildung 31: 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 "D020" aufrufen. Dazu tragen wir den Namen in die CICS-Kommandozeile ein (s. Abbildung 31) und bestätigen mit der Eingabetaste.

```
DFHAC2220 20:49:29 A06C001 The coordinator system has indicated that the
 current unit of work is to be backed out. Transaction D020 has been
 abnormally terminated with abend ASP3.
```

Abbildung 32: Fehlermeldung

Wir erhalten eine Fehlermeldung (s. Abbildung 32).

Manchmal erscheint auch eine andere Fehlermeldung als die in Abbildung 32 dargestellte.

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.

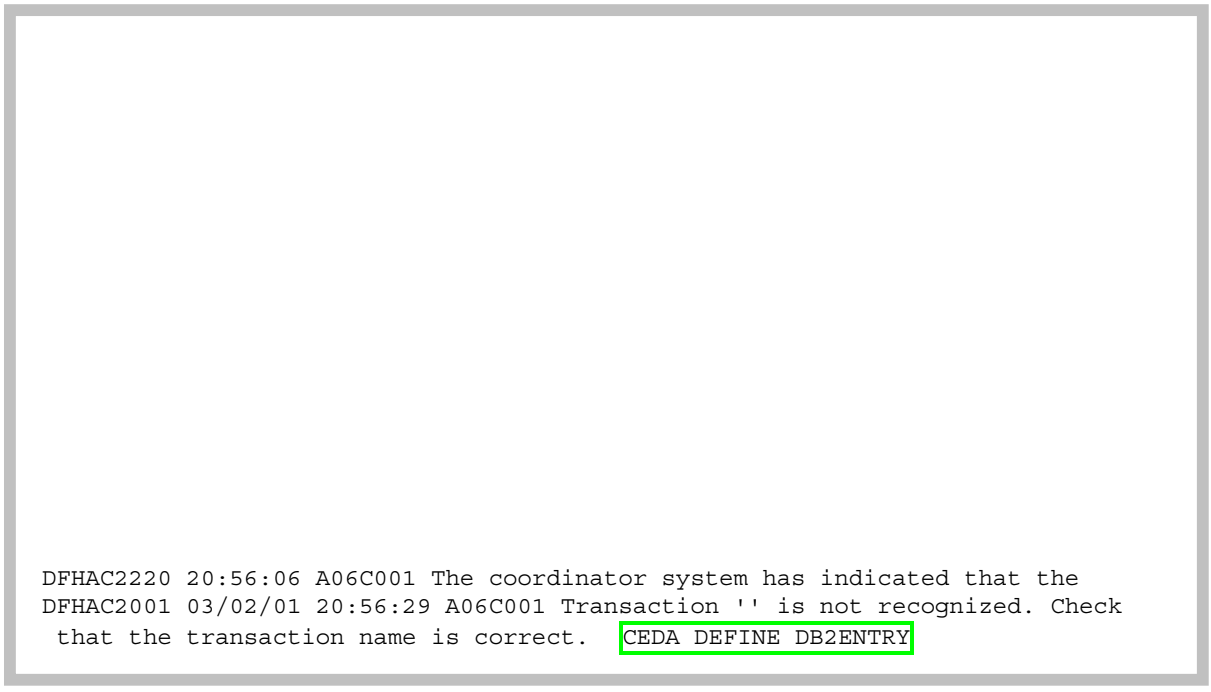


Abbildung 33: Aufruf der Definition der Datenbank

Die Definition erfolgt mit dem Kommando "CEDA DEFINE DB2ENTRY" (s. Abbildung 33).

```
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 | TErm
                                       | 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 34: DEFINE DB2ENTRY-Panel

Nachdem wir die Eingabetaste gedrückt haben, erscheint das "DEFINE DB2ENTRY-Panel" (s. Abbildung 34). Wir müssen die fehlenden Angaben eintragen (s. Abbildung 35 und 36).

```
define DB2ENTRY
  OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
   CEDA  DEFine DB2Entry(          )
    DB2Entry     ==> D020
    Group        ==> PRAKT20
    DEscription  ==>
   THREAD SELECTION ATTRIBUTES
    TRansid      ==> D020
   THREAD OPERATION ATTRIBUTES
    ACcountrec   ==> None              None | TXid | TAsk | Uow
    AUTHId       ==>
    AUTHType     ==>                   Userid | Opid | Group | Sign | TErm
                                       | 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 35: Eingabe der Parameter

Wir bezeichnen den DB2- Zugriff (DB2Entry) mit dem Namen "D020". Das Ganze wird Teil der Gruppe "PRAKT20". Unsere Transactions-ID (TRansid) ist "D020". Erinnern wir uns: Wir hatten ein JCL-Script "STARTJCL" 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     ==> D020
    Group        ==> PRAKT20
    DEscription  ==>
   THREAD SELECTION ATTRIBUTES
    TRansid      ==> D020
   THREAD OPERATION ATTRIBUTES
    ACcountrec   ==> TXid              None | TXid | TAsk | Uow
    AUTHId       ==>
    AUTHType     ==> Sign                  Userid | Opid | Group | Sign | TErm
                                         | 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 36: 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( D020     )
    DB2Entry       : D020
    Group          : PRAKT20
    DEscription  ==>
   THREAD SELECTION ATTRIBUTES
    TRansid      ==> D020
   THREAD OPERATION ATTRIBUTES
    ACcountrec   ==> TXid              None | TXid | TAsk | Uow
    AUTHId       ==>
    AUTHType     ==> Userid            Userid | Opid | Group | Sign | TErm
                                       | 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 37: Bestätigung der gelungenen Definition

Die Definition war erfolgreich und wird bestätigt durch die Ausschrift: "DEFINE SUCCESSFUL" (s. Abbildung 37).

Wir verlassen die Definition mit der F3-Taste.

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

Abbildung 38: Installation der Gruppe

Diese Änderung muss wieder installiert werden. Dazu geben wir wieder den Befehl "CEDA INSTALL GROUP(PRAKT20)" (s. Abbildung 38) 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 39: Installation der Gruppe

Die Ausgabe „INSTALL SUCCESSFUL“ in der Abbildung 39 sagt aus, dass die Installation erfolgreich war. Wir verlassen diesen Screen wieder mit F3.

Abbildung 40: Starten der Transaktion

Wir geben den Namen unserer Transaktion "D020" ein, um diese aufzurufen (s. Abbildung 40) und bestätigen mit der Eingabetaste.

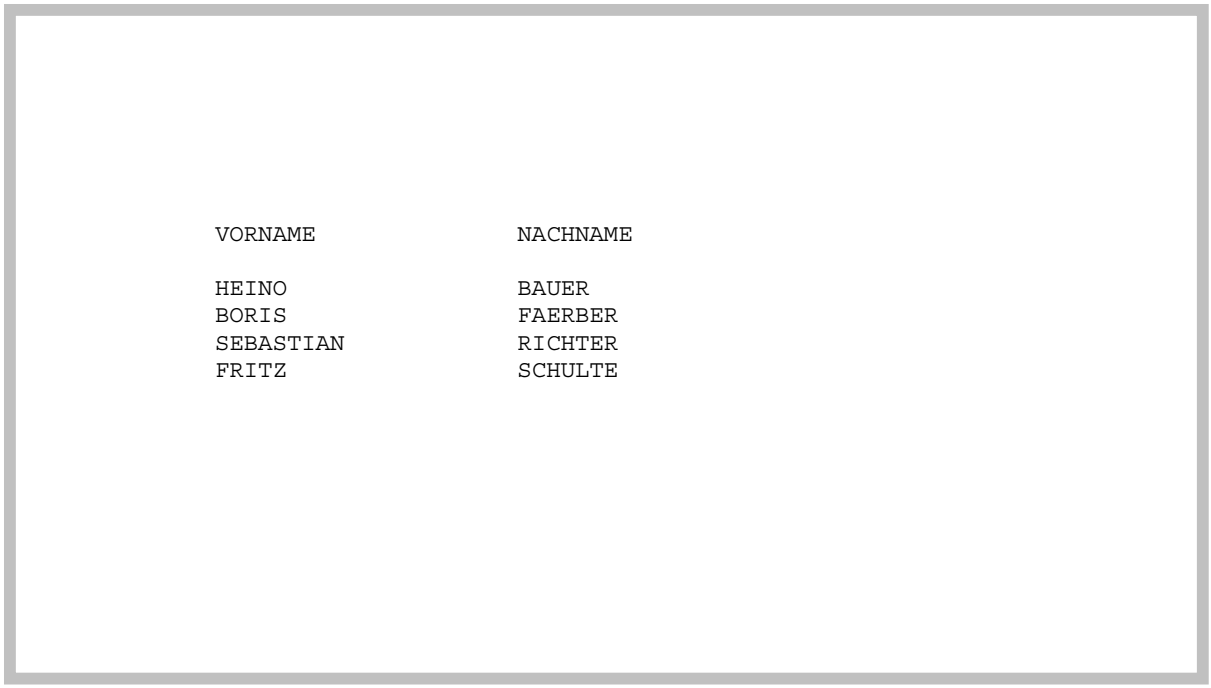


Abbildung 41: Ausgabe der Datenbanktabelle

Die korrekte Ausgabe der Datenbank erscheint auf dem Bildschirm (s. Abbildung 41). Damit ist die Aufgabe gelöst.

Aufgabe: *Bereiten Sie unter CICS die Transaktion vor, die auf die DB2-Datenbank zugreift (benutzen Sie die CICS-Gruppe "prakt<Ihre Gruppen-Nr>") und führen Sie diese aus. Die DB2-Datenbank soll die Namen der Übungsteilnehmer Ihrer Gruppe enthalten. Benutzen Sie als Transaktions-ID "D<Gruppen-Nr>". Sind Sie z.B. Gruppe 20, dann wäre Ihre Transaktion-ID "D020". Erzeugen Sie einen Screenshot (unter Windows durch den Shortcut ALT-Druck) Ihrer Version der Abbildung 40 und schicken Sie diesen ihrem Betreuer per Mail zu. Der Screenshot darf eine Größe von 300 Kbyte nicht überschreiten, benutzen Sie möglichst das JPG-Format, dass mit Dateigrößen unter 90 Kbyte auskommt. Löschen Sie nichts von Ihrer Lösung, so dass Ihr Betreuer Ihre Transaktion aufrufen kann.*

Aufgabe: *Gehen Sie vom CUSTOMPAC MASTER APPLICATION MENU aus in die System Display and Search Facility. Im erscheinenden SDSF PRIMARY OPTION MENU wählen Sie die Option ST. Löschen Sie alle angezeigten Jobs, die sich in der PRINT-Queue befinden, indem Sie links neben einen jeden Jobnamen "p" (purge) eintragen und anschließend die Eingabetaste (mehrfach) drücken. Einen Job dürfen Sie natürlich nicht löschen: Den einen, der sich in der EXECUTION-Queue befindet. Denn das ist der Job, mit dem Sie zur Zeit eingeloggt sind.*