Tutorial 5

Datenbankzugriff mit CICS (C/C++)

ph v/2010/03

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. Desweiteren 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.

Der Dataset PRAKT20.CICSDB2.TEST01 wurde bereits im Tutorial 4 erstellt und ist bisher noch leer. 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 "SET5020", der genau eine Map "MAP5020" enthält. Ein Mapset kann aber auch mehrere Maps enthalten. Diese Map "MAP5020" definiert Positionen, Länge sowie weitere Attribute der Darstellung der Daten aus der DB2-Datenbank auf dem Bildschirm.

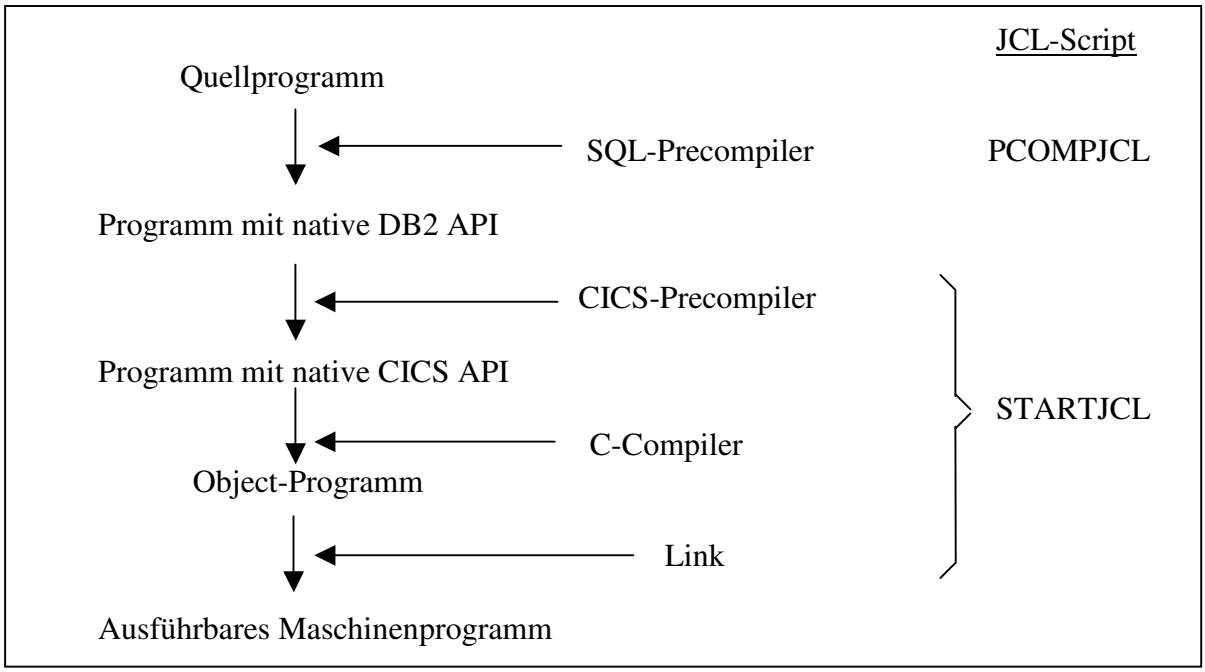


Abbildung 1:
Schritte vom C-Programm mit EXEC SQL-Statements zum ausführbaren Maschinenprogramm

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 dem der Linker ein ausführbares Maschinenprogramm erzeugt (s. auch Abbildung 1).

Alle diese Schritte werden im TSO ausgeführt. TSO ist ein Subsystem von z/OS. Ein weiteres Subsystem von z/OS 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 implementierte CICS-Transaktion mit der Transaktions-ID "X020" auszulösen. Folgende Schritte sind dazu notwendig:

1. Definition des Mapsets mittels
   "CEDA DEFINE MAPSET(SET5020) GROUP(PRAKT20)"
2. Definition des C-Anwenderprogrammes mittels
   "CEDA DEFINE PROG(CPROG020) GROUP(PRAKT20)"
3. Definition des Namens der Transaktion-ID mittels
   "CEDA DEFINE TRANS(X020) GROUP(PRAKT20)"
4. Definition unserer Datenbank und Datenbanktabelle mittels
   "CEDA DEFINE DB2ENTRY"

Nach diesen Schritten sind der Mapset SET5020 mit der Map MAP5020, das ausführbare Maschinenprogramm, das aus CPROG020 generiert wurde, die selbst definierte Transaktion-ID "X020" 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. Das geschieht durch

5. "CEDA INSTALL GROUP(PRAKT20)"

Nun haben wir unser Ziel erreicht. Geben wir "X020" 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).

Warnung:

Ihr DB2ENTRY ist nur ein einziges Mal von Ihnen installierbar. Deshalb könnte z.B. Ihre zweite Anwendung von "CEDA INSTALL GROUP ..." die Fehlermeldung "INSTALL UNSUCCESSFUL" produzieren. Dieser Fehler kann von Ihnen mangels Ihrer CICS-Zugriffsrechte nicht behoben werden. Im Anhang wird dieses Problem erläutert. Informieren Sie deshalb umgehend Ihren Tutor.

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 den 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.

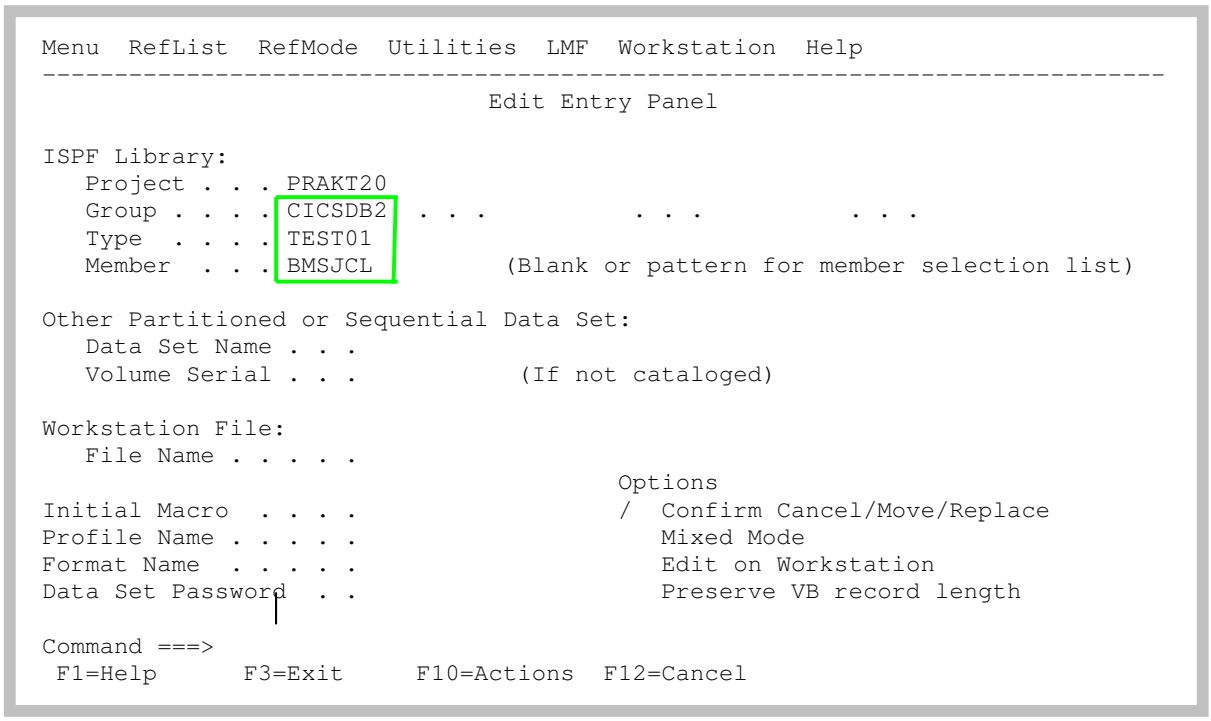


Abbildung 3: Anlegen des Members "BMSJCL"

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

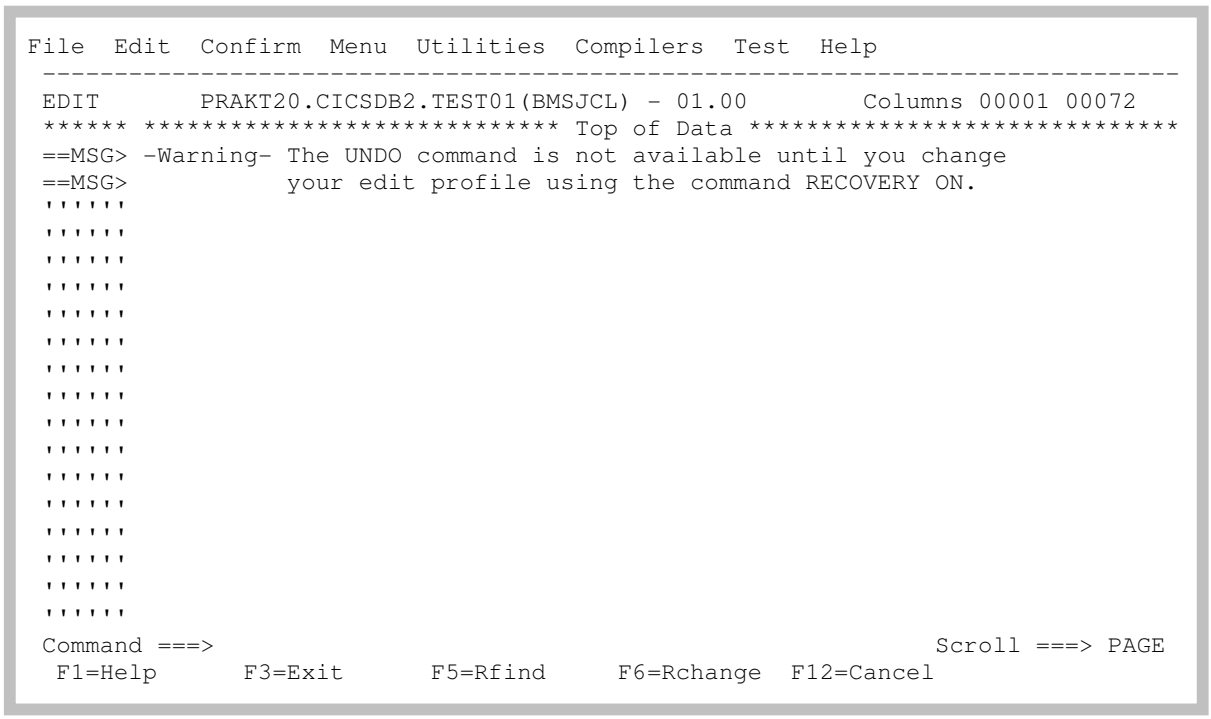


Abbildung 4: Der leere Edit-Entry-Panel

Ein leerer 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.

```
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='SET5020',RMODE=24
000003 //SYSUT1   DD *
000004 SET5020 DFHMSD TYPE=MAP,MODE=INOUT,LANG=C,STORAGE=AUTO,TIOAPFX=YES
000005 *        MENU MAP.
000006 MAP5020 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, die wir aus unserer DB2-Datenbank auslesen.

Wir geben "SUB" auf der Kommandozeile ein. Zusätzlich zu dem übersetzten Programm wird in dem Member "PRAKT20.LIB(SET5020)" ein Template für unser Business Logic-Programm (in C) abgespeichert.

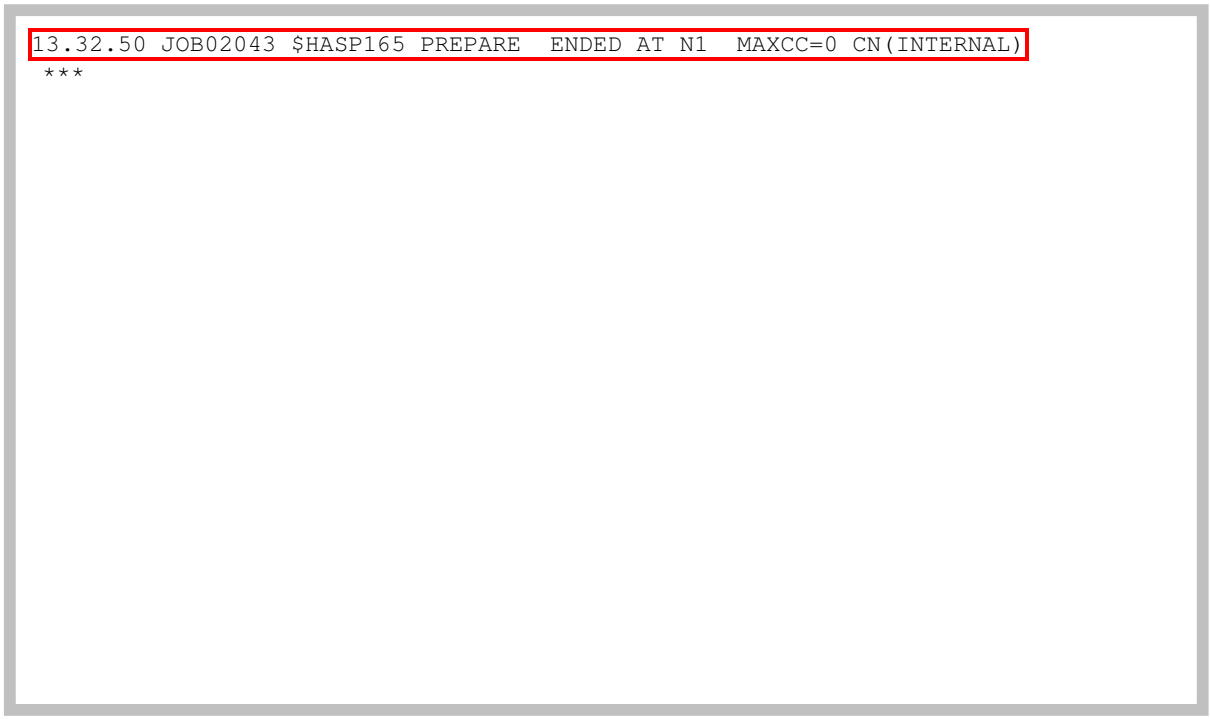


Abbildung 7: Bestätigung der Jobverarbeitung

Wir warten, bis JES unser BMS-Programm übersetzt hat (30-60 Sekunden). Durch das Betätigen der Eingabetaste erscheint der 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.Benutzen Sie MAP5<Ihre Prakt-ID> als Mapnamen sowie SET5<Ihre Prakt-ID> als Mapsetnamen. Haben Sie z.B. den Account PRAK162, so ist Ihr Map-Name MAP5162 und Ihr Mapset-Name SET5162.*

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(SET5020)". Wir sehen uns "PRAKT20.LIB(SET5020)" an. Dieser Member könnte je nach eingestellter Host-Code-Page auch leicht modifiziert auf dem Bildschirm dargestellt sein.

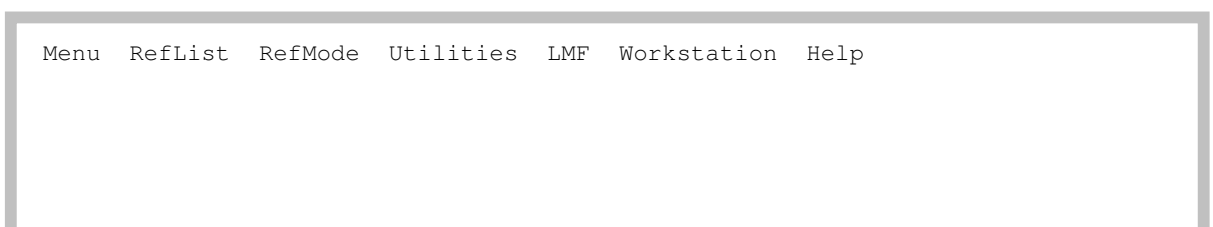
```
union
{
struct {
          char         dfhms1Ý12¨;
          short int    vnam1l;
          char         vnam1f;
          char         vnam1iÝ20¨;
          short int    nnam1l;
          char         nnam1f;
          char         nnam1iÝ20¨;
          short int    vnam2l;
          char         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¨;
        } map5020i;

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¨;
        } map5020o;

} map5020;
```

Dies ist der Code von "PRAKT20.LIB(SET5020)". 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 den Edit-Entry-Panel auf.



```
------------------------------------------------------------------------------
                              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.

```
 File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       PRAKT20.CICSDB2.TEST01(CPROG020) - 01.00        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.
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 ''''''
 Command ===>                                                 Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 9: Der leere Edit-Entry-Panel

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

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       PRAKT20.CICSDB2.TEST01(CPROG020) - 01.01        Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 ==MSG> -CAUTION- Profile changed to CAPS OFF (from CAPS ON) because data
 ==MSG>           contains lower case characters.
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000001 #include <memory.h>
 000002 #include <stdlib.h>
 000003 #include <//'PRAKT20.LIB(SET5020)'>
 000004
 000005 EXEC SQL INCLUDE SQLCA;
 000006 EXEC SQL BEGIN DECLARE SECTION;
 000007 char vnameÝ20¨;
 000008 char nnameÝ20¨;
 000009 EXEC SQL END DECLARE SECTION;
 000010
 000011 main()
 000012 {
 000013   EXEC SQL DECLARE C1 CURSOR FOR
 000014            SELECT VNAME,NNAME FROM PRAKT20.TAB020;
 Command ===>                                                 Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

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(map5020.map5020i.vnam1i,vname,20);
000018   memcpy(map5020.map5020i.nnam1i,nname,20);
000019   EXEC SQL FETCH C1 INTO :vname, :nname;
000020   memcpy(map5020.map5020i.vnam2i,vname,20);
000021   memcpy(map5020.map5020i.nnam2i,nname,20);
000022   EXEC SQL FETCH C1 INTO :vname, :nname;
000023   memcpy(map5020.map5020i.vnam3i,vname,20);
000024   memcpy(map5020.map5020i.nnam3i,nname,20);
000025   EXEC SQL FETCH C1 INTO :vname, :nname;
000026   memcpy(map5020.map5020i.vnam4i,vname,20);
000027   memcpy(map5020.map5020i.nnam4i,nname,20);
000028   EXEC SQL CLOSE C1;
000029
000030   EXEC CICS SEND MAP("map5020") MAPSET("set5020") 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 etwas mit dem "Host Code Page"-Problem zu tun: Je nach eingestellter Host Code Page werden Sonderzeichen, wie z.B. eckige Klammern, ganz verschieden auf dem Bildschirm dargestellt.

Nachfolgend wird dargestellt, welche Hex-Codes vom C-Compiler als eckige Klammern erkannt werden und welche nicht:

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

Der "Proper EBCDIC"-Code wird vom C-Compiler als eckige Klammer erkannt, doch der "Improper EBCDIC"-Code nicht. Und wenn eine entsprechende Host Code Page eingestellt ist, wird der "Proper EBCDIC"-Code der öffnenden eckigen Klammer als "Ý" angezeigt und vom C-Compiler richtig erkannt; doch der "Improper EBCDIC"-Code als "[" korrekt auf dem Bildschirm angezeigt, doch vom C-Compiler nicht als eckige Klammer erkannt.

Es gibt mehrere Möglichkeiten mit diesem Problem umzugehen. Die einfachste ist die folgende:

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 "SET5020" im Dataset "PRAKT20.LIB" erstellt, der ein Template für unser C-Programm enthält. Im Tutorial 3 (C-Version) 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(SET5020)'>
Es tritt an Stelle des manuellen Kopiervorgangs.

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

Aufgabe: *Erstellen Sie den Member und schreiben Sie das C-Programm hinein. Benutzen Sie als Membernamen CPROG<Ihre Prakt-ID>.*

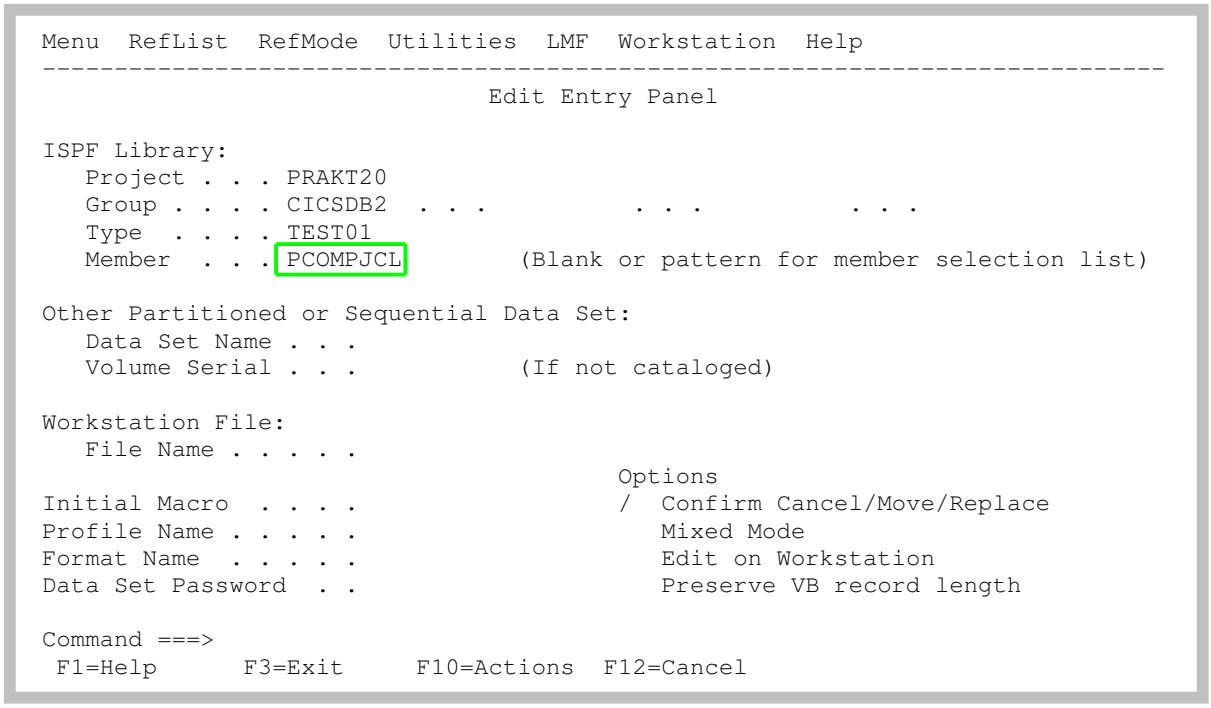
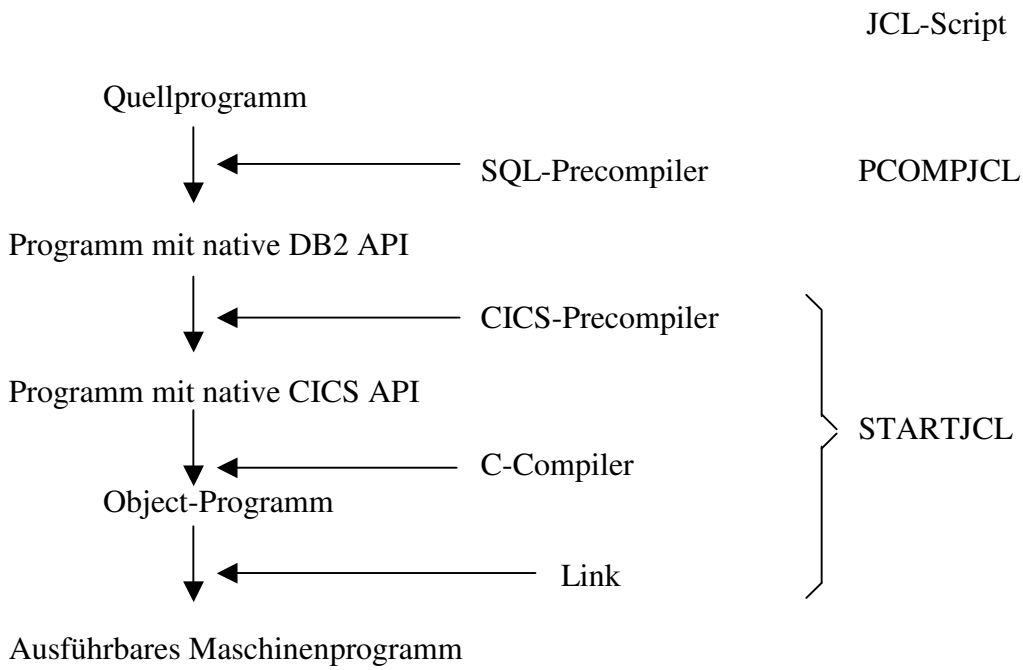


Abbildung 12: Anlegen des Members PCOMPJCL

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" ist bis jetzt noch leer. 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.

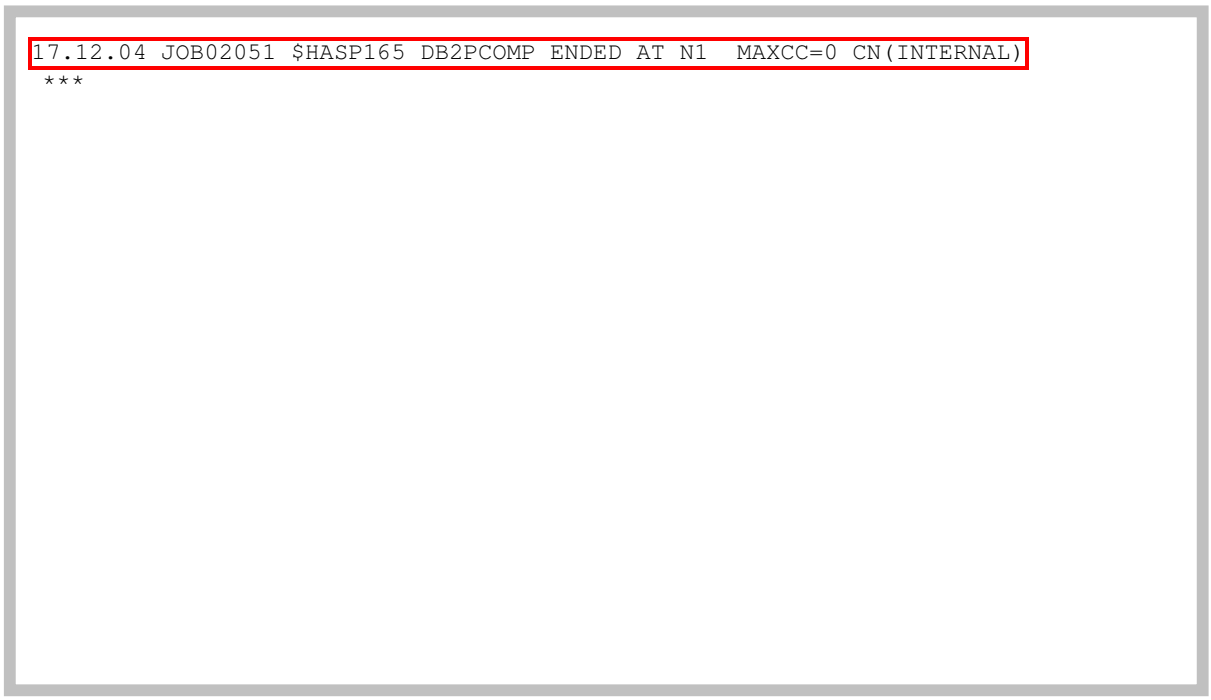


Abbildung 14: Bestätigung der Jobverarbeitung

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

Aufgabe: *Erstellen Sie einen neuen Member und schreiben Sie das JCL-Script, das den Precompiler-Aufruf enthält, hinein. Führen Sie es 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 einen neuen 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=SYS1.DSN.V910.SDSNEXIT
000016 //         DD DISP=SHR,DSN=SYS1.DSN.V910.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(D931)
000023   BIND PLAN(ZGR020) MEMBER(CPROG020) ACTION(REP) RETAIN ISOLATION(CS)
000024 END
000025 //********************
000026 //* GRANT
000027 //********************
000028 //GRANT    EXEC PGM=IKJEFT01
000029 //STEPLIB  DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
000030 //SYSPRINT DD SYSOUT=*
000031 //SYSTSPRT DD SYSOUT=*
000032 //SYSUDUMP DD SYSOUT=*
000033 //SYSTSIN  DD *
000034   DSN SYSTEM(D931)
000035   RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) -
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 im JCL-Script als "PLAN" bezeichnet). Wir geben diesem Zeiger den Namen "ZGR020" (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('SYS1.DSN.V910.RUNLIB.LOAD')
000037   END
000038 //SYSIN    DD *
000039 GRANT EXECUTE ON PLAN ZGR020 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) "ZGR020" taucht nochmals auf (Zeile 39).

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

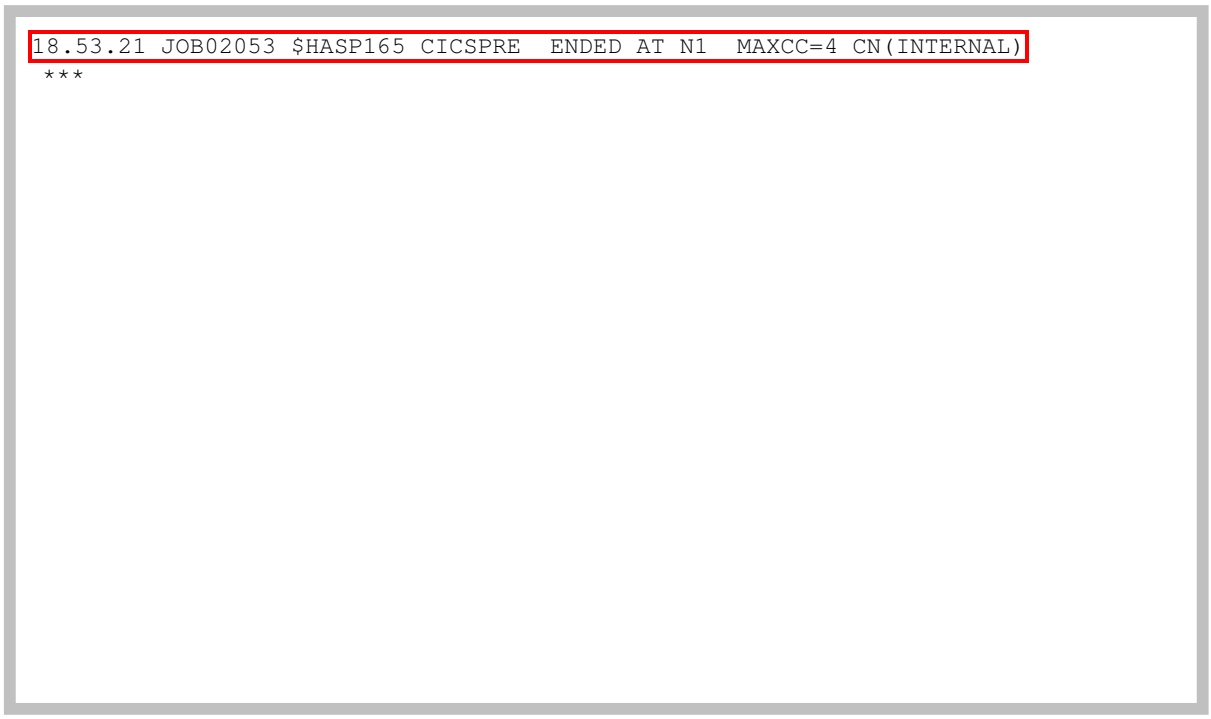


Abbildung 19: Ausgabe der Jobverarbeitung

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

Aufgabe: *Erstellen Sie einen neuen Member, legen Sie das JCL-Script STARTJCL an (mit an Ihren Accountnamen angepaßtem Jobnamen) und führen Sie es aus. Benutzen Sie als Zeiger (Plan) den Bezeichner ZGR<Ihre Prakt-Nr>.*

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 z/OS-Session.

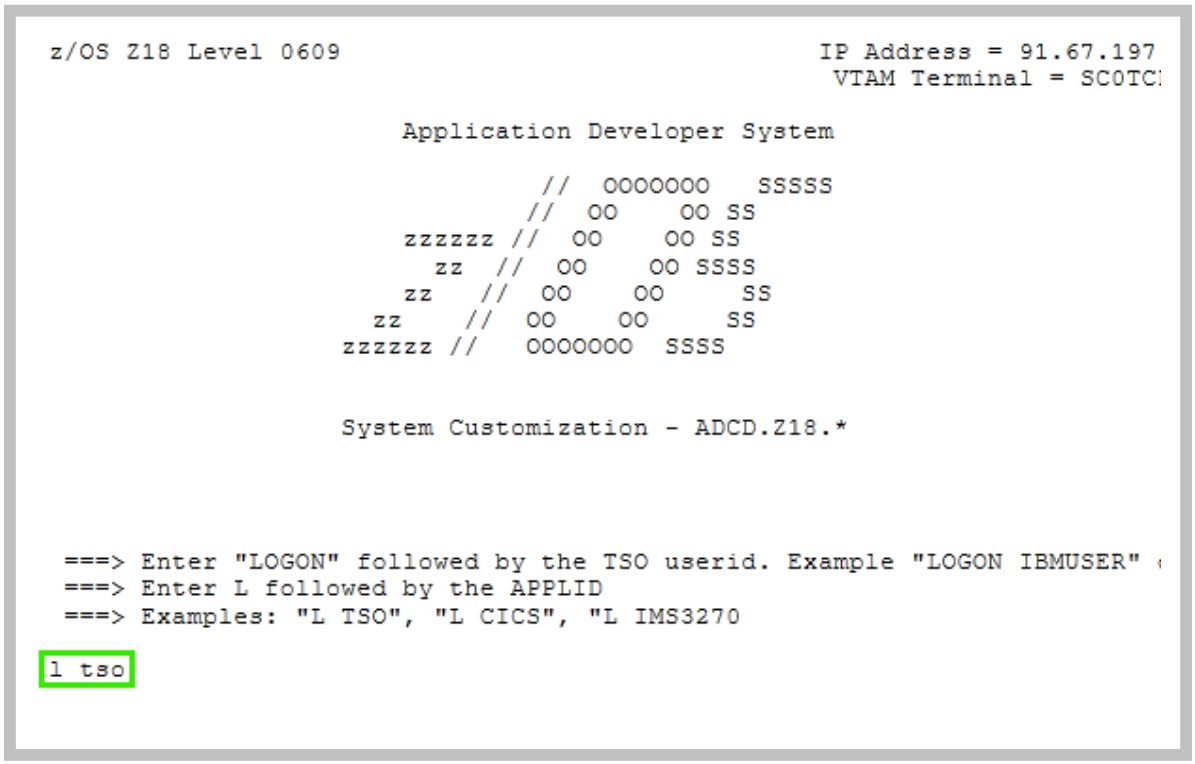


Abbildung 20: Der Logon-Screen

Wir loggen uns anstatt "l tso" mit "L CICS" ein (s. Abbildung 20) und bestätigen mit der Eingabetaste.

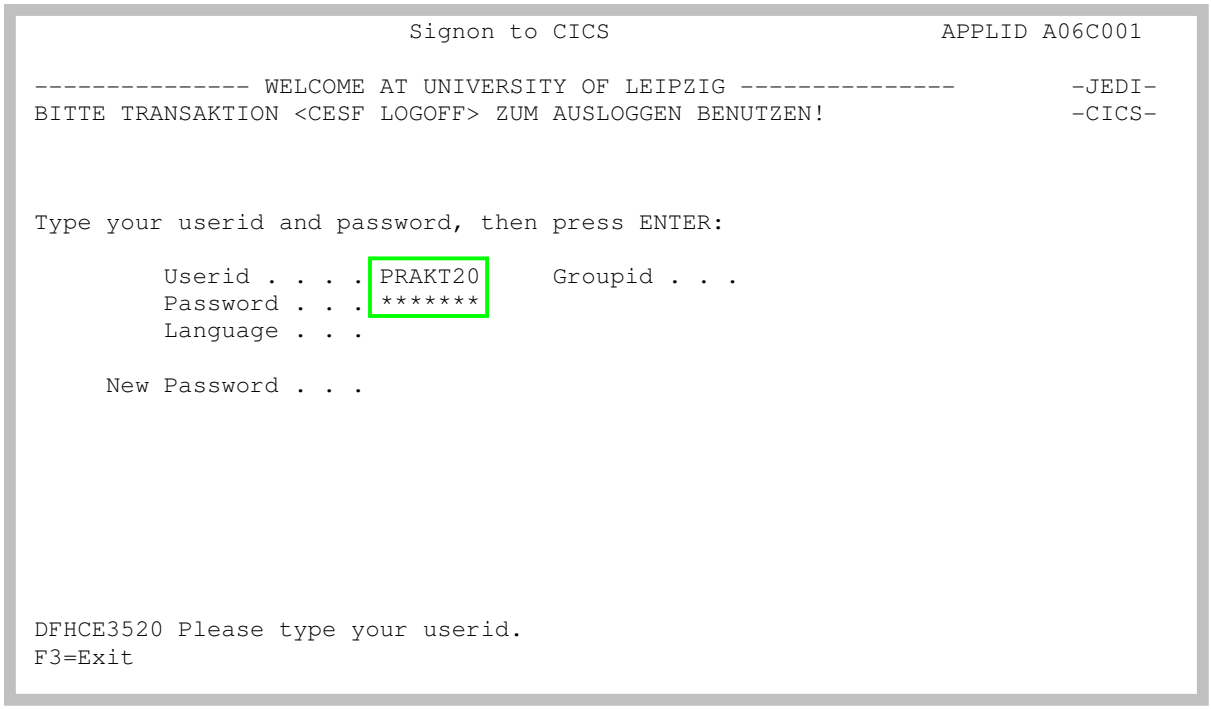


Abbildung 20a: Signon to CICS-Screen

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 kommen wir in den nächsten Screen.

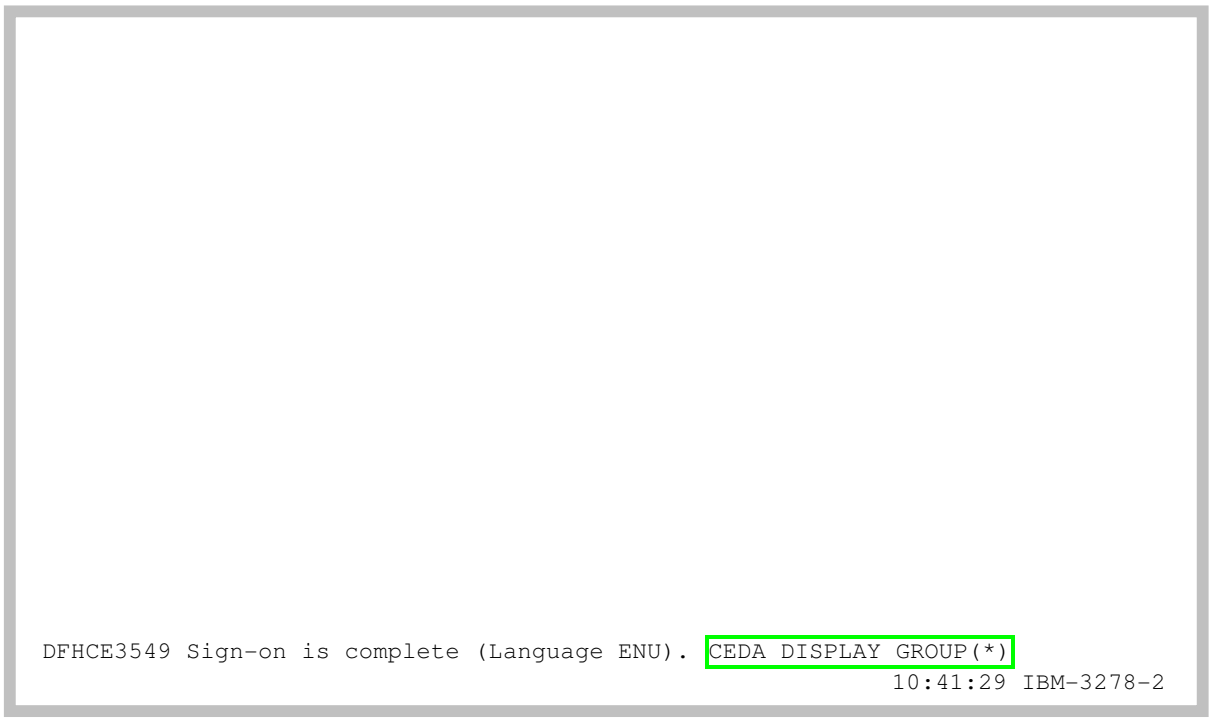


Abbildung 21: Einloggvorgang ist abgeschlossen

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

An dieser Stelle ist es jetzt notwendig, eine neue Gruppe anzulegen.

```
 CEDA DEFINE MAPSET(SET5020) 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).

An dieser Stelle ist es jetzt notwendig, eine neue Gruppe anzulegen. Wir definieren zunächst unser BMS-Programm mit dem Namen "SET5020" für die neue Group "PRAKT20" und betätigen 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. Wir haben aber bereits die Gruppe PRAKT20 in Tutorial 3 verwendet. Wir löschen deshalb die Gruppe PRAKT20 mit dem Befehl

CEDA DELETE ALL GROUP(PRAKT20)

```
CEDA DELETE ALL GROUP(PRAK129)
ENTER COMMANDS
 GROUP
```

Und verifizieren danach mit CEDA DISPLAY GROUP(*) dass dies auch tatsächlich geschehen ist.

```
CEDA DEFINE MAPSET(SET5020) GROUP(PRAKT20)
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
  CEDA  DEFine Mapset( SET5020   )
   Mapset         : SET5020
   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 des Mapsets "SET5020"

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( SET5020   )
   Mapset         : SET5020
   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 eingegeben; Le370 ist aber eigentlich eine Entwicklungsumgebung (s. Abbildung 25)

Auch hier bestätigen wir mit der Eingabetaste.

Die Nachricht "DEFINE SUCCESSFUL" erscheint; wir beenden diese Aktion mit Betätigung der F3-Taste.

```
CEDA DEFINE TRANS(X020) GROUP(PRAKT20)
  STATUS:  SESSION ENDED
```

Abbildung 26: Sitzung beendet

Als letztes müssen wir die Bezeichnung der neuen Transaktion definieren. Wir wählen auch hierfür den Namen "X020". Es könnte natürlich auch ein beliebiger anderer Name sein, solange er aus 4 Zeichen besteht. Wir geben das Kommando "CEDA DEFINE TRANS(X020) GROUP(PRAKT20)" ein (s. Abbildung 26) und bestätigen mit der Eingabetaste.

```
 DEFINE TRANS(X020) GROUP(PRAKT20)
 OVERTYPE TO MODIFY                                      CICS RELEASE = 0530
  CEDA  DEFine TRANSaction( X020 )
   TRANSaction  ==> X020
   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 mit der Eingabetaste.

```
 OVERTYPE TO MODIFY                                      CICS RELEASE = 0530
  CEDA  DEFine TRANSaction( X020 )
   TRANSaction    : X020
   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 (Abbildung 28); also war die Definition erfolgreich, wir beenden sie mit der F3-Taste.

```
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.

Es kann sein, dass an dieser Stelle die Meldung „Install unsuccessful“ erscheint. Eine Erläuterung dieses Fehlers finden Sie im Anhang dieses Tutorials.

```
X020
 STATUS:  SESSION ENDED
```

Abbildung 31: Aufruf der Transaktion

Im Tutorial 3 waren wir mit der Definition und Installation unserer Transaktion fertig. Wir versuchen es einmal, indem wir unsere Transaktion mit der Bezeichnung "X020" 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 X020 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.

TSO, CICS und DB2 sind separate z/OS-Subsysteme, die in getrennten virtuellen Adressräumen laufen. Die CICS-Gruppe "PRAKT20" benötigt eine Definition unserer Datenbank und Datenbanktabelle.

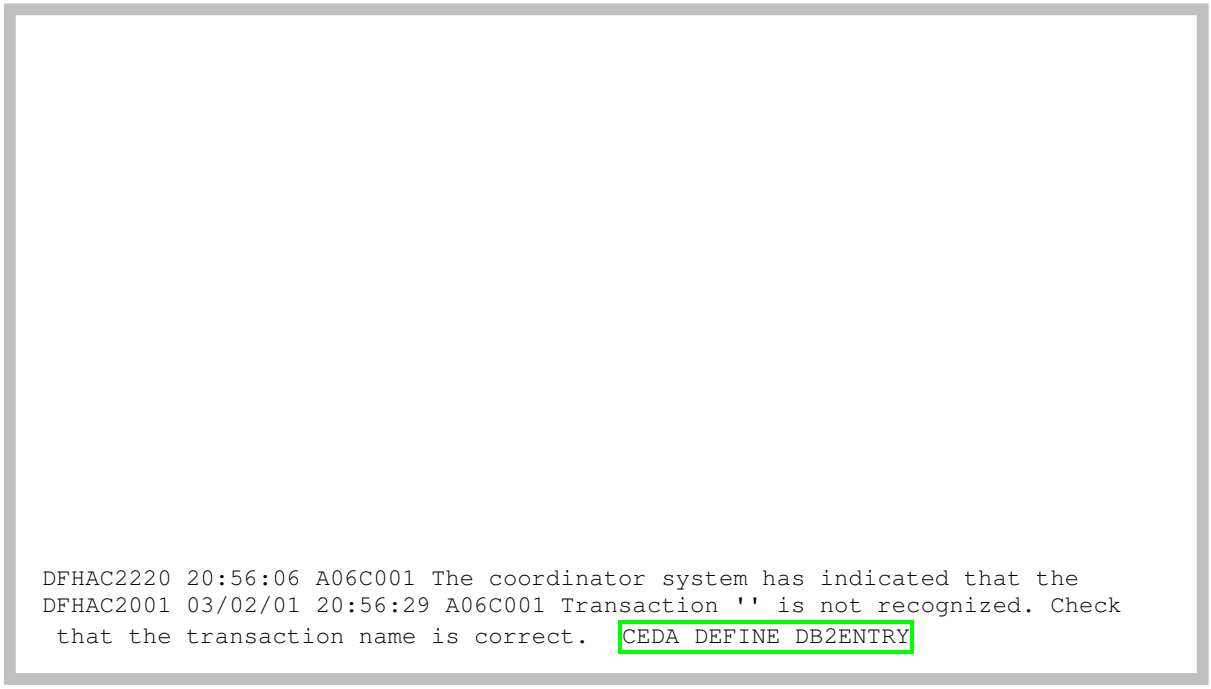


Abbildung 33: Aufruf der Definition der Datenbank

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

Es kann auch sein, dass das System sich an dieser Stelle aufhängt. Resultat: Keine Tastatureingabe ist möglich, und links unten erscheint ein Strich-Männchen. Drücken der F2-Taste behebt dieses Problem.

```
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 der "DEFINE DB2ENTRY-Panel" (s. Abbildung 34). Wir müssen die fehlenden Angaben eintragen und betätigen abschließend die Eingabetaste (s. Abbildung 35).

```
define DB2ENTRY
  OVERTYPE TO MODIFY                                      CICS RELEASE = 0530
   CEDA  DEFine DB2Entry(          )
    DB2Entry     ==> X020
    Group        ==> PRAKT20
    DEscription  ==>
   THREAD SELECTION ATTRIBUTES
    TRansid      ==> X020
   THREAD OPERATION ATTRIBUTES
    ACcountrec   ==> TXid              None | TXid | TAsk | Uow
    AUTHId       ==>
    AUTHType     ==> Sign              Userid | Opid | Group | Sign | TErm
                                       | TX
    DRollback    ==> Yes               Yes | No
    PLAN         ==> ZGR020
    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 bezeichnen den DB2-Zugriff (DB2Entry) mit dem Namen "X020". Das Ganze wird Teil der Gruppe "PRAKT20". Unsere TRansaction-ID (TRansID) ist "X020". 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 "ZGR020". Hier wird jetzt für CICS die Verknüpfung zu der Datenbanktabelle hergestellt.

```
 OVERTYPE TO MODIFY                                          CICS RELEASE = 0530
  CEDA  DEFine DB2Entry( X020     )
   DB2Entry       : X020
   Group          : PRAKT20
   DEscription  ==>
  THREAD SELECTION ATTRIBUTES
   TRansid      ==> X020
  THREAD OPERATION ATTRIBUTES
   ACcountrec   ==> TXid              None | TXid | TAsk | Uow
   AUTHId       ==>
   AUTHType     ==> Userid            Userid | Opid | Group | Sign | TErm
                                      | TX
   DRollback    ==> Yes               Yes | No
   PLAN         ==> ZGR020
   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.

```
X020
 STATUS:  SESSION ENDED
```

Abbildung 40: Starten der Transaktion

Wir geben den Namen unserer Transaktion "X020" 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).

Aufgabe: *Bereiten Sie unter CICS die Transaktion vor, die auf die DB2-Datenbank zugreifen soll und führen Sie diese anschließend aus. Benutzen Sie dabei als CICS-Gruppen-Namen Ihren Accountnamen, also z.B. PRAKT45 oder PRAK162. Die DB2-Datenbank soll Ihren Namen / Ihre Namen enthalten. Benutzen Sie als Transaktions-ID "X<Ihre Prakt-ID>". Bezeichnen Sie den DB2ENTRY identisch zu Ihrer Transaktions-ID.*

Erzeugen Sie einen Screenshot (unter Windows durch den Shortcut ALT-Druck) Ihrer Version der Abbildung 41 und schicken Sie diesen Ihrem Betreuer per Mail zu. Der Screenshot darf eine Größe von 250 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.*

Die Ausführung unserer Transaktion (unseres C-Programms) ist damit abgeschlossen – CICS erwartet jetzt die Eingabe einer neuen Transaktion. Dies könnte z.B. CEDA DISPLAY GROUP(*) sein.

Wenn wir mit unserer CICS Sitzung fertig sind und keine weitere Transaktion durch Eingabe einer TRID starten wollen, geben wir die Logoff-Transaktion "CESF LOGOFF" ein, gefolgt von der Eingabetaste, ein (s. Abbildung 42).

Abbildung 42: Ausloggen aus CICS

Anhang

"CEDA INSTALL GROUP ..." erzeugt den Fehler "install unsuccessful"

Dieses Problem könnte auftreten, wenn jemand mehrmals den Befehl "CEDA INSTALL GROUP ..." eingibt. Als Fehlermeldung wird INSTALL UNSUCCESSFUL zurückgegeben.

Das Problem ist, dass sich die schon einmal per "CEDA INSTALL GROUP ..." installierte DB2ENTRY-Komponente nicht so ohne weiteres überschreiben läßt.

Man muß das Überschreiben erlauben. Dies erfordert aber Administrator Rechte über die Ihre User ID nicht verfügt! Deshalb versuchen Sie bitte, ein mehrfaches Abarbeiten der Tutorials 5 zu vermeiden.

Sollte das Problem trotzdem einmal auftreten, informieren Sie bitte einen Betreuer mit Admin-Rechten.

Zu Ihrer Information:

Das Problem kann mit Administrator Rechten behoben werden durch die Eingabe:

CEMT I DB2E(<DB2E-Name>)

Dies gibt den DB2Entry mit dem Namen <DB2E-Name> auf dem Bildschirm aus.

Im konkreten Beispiel liefert

CEMT I DB2E(A020) die folgende Bildschirmausgabe:

```
I DB2E(A020)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Db2e(A020    ) Txi Sig Ena Poo Hig Pro( 0000 ) Pth(0000)
    Threadl( 0003 ) Threads(0000) Twa Plan( AS5      )
```

Der Wert "Ena" (ENAble) ist auf "Dis" (DISable) zu setzen, um ein Überschreiben zu erlauben. Dazu reicht es, wenn man den Buchstaben "E" von "Ena" mit einem "D" überschreibt sowie die Eingabetaste betätigt. Das Ergebnis dieser Aktion ist im konkreten Beispiel

```
I DB2E(A020)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Db2e(A020    ) Txi Sig Dis Poo Hig Pro( 0000 ) Pth(0000)    NORMAL
    Threadl( 0003 ) Threads(0000) Twa Plan( AS5      )
```

Nun ist ein Überschreiben des DB2Entry-Eintrages "A020" und damit auch eine Neuinstallation der Gruppe PRAKT20 wieder möglich:

CEDA INSTALL GROUP(PRAKT20)

funktioniert fehlerfrei und gibt wieder

INSTALL SUCCESSFUL zurück.