CPGJDBC

Kapitel 2: Funktion 2000

Der CPG-JDBC-Daemon ermöglicht in CICS-Transaktionen und in Batchanwendungen die direkte Verarbeitung von Datenbanken anderer Rechnerplattformen (mit JDBC- oder ODBC-Schnittstelle).

Der Daemon setzt TCP/IP auf allen Plattformen voraus und kann aus jeder Programmiersprache im CICS mit EXEC CICS LINK und im Batch mit CALL aufgerufen werden.

Prinzip der Verarbeitung im CICS

- Ein SQL-Befehl oder eine SQL-Befehlsfolge wird auf einen Temporary-Storage-Bereich abgestellt.
- Der Name der Datenbank und des Temporary-Storage-Bereichs werden in die Common Area gestellt.
- Der Daemon wird mit LINK (im CPG mit EXPR, im QPG mit PROG) aufgerufen.
- Das Ergebnis wird vom Daemon in einer zweiten Temporary Storage Queue zur Verfügung gestellt.
- Eine dritte Queue enthält zusätzlich zu den Daten Informationen über den Zugriff (wie die Spaltennamen und Spaltenformate der verarbeiteten Tabelle) oder Fehlermeldungen im Klartext. Die Common Area liefert einen zweistelligen Return-Code.

Prinzip der Verarbeitung im Batch

- Ein SQL-Befehl oder eine SQL-Befehlsfolge wird auf eine VSAM-ESDS-Datei abgestellt.
- Der Daemon wird mit CALL aufgerufen. Dabei wird der Name der zu verarbeitenden Datenbank mit PARM übergeben, der Return-Code der Verarbeitung steht nach dem Rücksprung im PARM-Feld.
- Das Ergebnis wird vom Daemon in einer ESDS-Datei zur Verfügung gestellt.
- Eine dritte ESDS-Datei enthält zusätzlich zu den Daten Informationen über den Zugriff (wie die Spaltennamen und Spaltenformate der verarbeiteten Tabelle) oder Fehlermeldungen im Klartext.

Kapitel 3: Regeln 3000

Regeln für die Onlineverarbeitung 3100

1. Name des Storage-Bereichs zur Übergabe des/der SQL-Commands

Der Name dieses Bereichs ist:

Stelle 1-4: Tasknummer im CICS, gepackt
Stelle 5-8: JDBC (C für Command)

2. Übergabe des Datenbank-Namens an den Daemon

Der Daemon erwartet seine Parameter an folgenden Stellen der Common Area:

Stelle 1 – 8: Name der Datenbank
Stelle 9 -10: blank
Stelle11-17: TS=JDBC

3. Aufruf des Daemons

Mit LINK ruft man die Phase **QJDBCGTO** auf.

4. Return-Code in der Common Area

In den Stellen 1-2 der Common Area steht ein Return-Code. Zur Zeit werden folgende Informationen übergeben:

Blank: Erfolgreicher Zugriff

EF: (Für End of File) – Nicht erfolgreicher Zugriff

5. Ergebnisbereiche lesen

Der Daemon liefert seine Ergebnisse auf folgenden Temporary Storage Queues:

Datenstorage: Stelle 1-4: Tasknummer im CICS, gepackt
Stelle 5-8: JDBD (D für Daten)

‚Logfile': Stelle 1-4: Tasknummer im CICS, gepackt
Stelle 5-8: JDBL (L für Logfile)

Regeln für die Batchverarbeitung 3200

1. Datei zur Übergabe des/der SQL-Commands

VSAM-ESDS-Datei mit dem Namen BJDBC und den Eigenschaften: Reuse, variable Satzlänge, Satzlänge von 100 – 2048 Bytes

2. Übergabe des Datenbank-Namens an den Daemon

Der Daemon erwartet seinen Parameter – den Datenbanknamen – in den ersten acht Stellen des Parameter-Felds.

3. Aufruf des Daemons

Mit CALL ruft man die Phase **QJDBCGTC** auf.

4. Return-Code im Parameter-Feld

In den Stellen 1-2 des Parameterfelds steht nach dem CALL ein Return-Code. Zur Zeit werden folgende Informationen übergeben:

Blank: Erfolgreicher Zugriff

EF: (Für End of File) – Nicht erfolgreicher Zugriff

5. Ergebnisbereiche lesen

Der Daemon liefert seine Ergebnisse auf folgenden ESDS-Dateien:

Daten: BJDBD

‚Logfile': BJDBL

mit den Eigenschaften: No Open, variable Satzlänge, Satzlänge von 100 – 2048 Bytes

Aufbau der Dateien 3300

Der Logfile (B)JDBL 3310

Der Logfile (B)JDBL hat folgenden Aufbau:

1. Satz: **READY**
2. Satz: **CONNECTED**

Ansonsten ist die Verbindung fehlgeschlagen. Ab dem 3. Satz stehen auf dem Logfile folgende Informationen:

- **DATATYPES: xxxx** xxxx steht für eine Auflistung der Spalteninformationen, wie die Datenbank sie liefert: Die Namen der Spalten sowie die Informationen über Datentyp und Länge.
- **DEFAULTS:** mit einer Auflistung aller gesetzten Default-Werte
- **DISCONNECT** falls der Programmierer diesen Befehl gegeben hat.
- **ERROR: xxxx** xxxx steht für eine Fehlermeldung im Klartext entweder vom Daemon oder vom JDBC-Treiber. Eine Liste der Daemon-Fehlermeldungen finden Sie im Kapitel Fehlersuche.
- **UNKNOWN STATEMENT** falls der Programmierer einen vom Daemon nicht unterstützten Befehl übergeben hat.
- **++** falls der Daemon einen neuen Befehl empfangen hat.

Aufbau der Ergebnisbereiche - Der Datenbereich (B)JDBD 3320

Der Datenfile JDBD hat im Default-Fall folgenden Aufbau:

```
Beispiel:

0000000000000000004,4 :Test 00005:0000000000000000
FFFFFFFFFFFFFFFFFF6F47E8AA4FFFFF7FFFFFFFFFFFFFFF
0000000000000000004B40A3523000005A0000000000000000
....5...10....5...20....5...30....5...40....5...50

00005 :
FFFFF474444444444444444444444444444444444444444444
000050A000000000000000000000000000000000000000000
....5...60....5...70....5...80....5...90....5..100 ....
```

Die Datenfelder stehen ‚hintereinander' auf einem Datensatz, wie sie von der Datenbank geliefert werden, jeweils getrennt mit einem Doppelpunkt.

Hier im Beispiel sehen Sie Werte aus einer ORACLE-Testtabelle mit drei Spalten: NUMBER(22), VARCHAR2(10), NUMBER(22). Der Daemon füllt die numerischen Formate (9,2 und 7,0) auf die Spaltenlänge 22 richtig auf.

Der Aufbau des Datenfiles (B)JDBD kann vom Programmierer selbst gestaltet werden:

Dazu wird der Befehl **DEFAULTS auf das Command-File (B)JDBC** ausgegeben.

Aufbau der Command-Datei (B)JDBC 3330

DEFAULTS bezieht sich auf die SELECT-Befehle und hat folgende Parameter:

DDOPT= **Y**es oder **N**o: Die Feldstruktur der Tabelle wird aufgelistet oder **n**icht.

DEC=, Als Dezimalzeichen wird vom Daemon das Komma gesetzt (Default).
DEC=. Als Dezimalzeichen wird vom Daemon der Punkt gesetzt.

EXP=Y Die Werte werden vom Daemon auf die Spaltenbreite aufgefüllt (expand): Numerische Werte linksbündig mit Nullen, alphanumerische Werte rechtsbündig mit Blanks (Default).

EXP=N Die Werte werden nicht spaltengerecht erweitert. In diesem Fall muss mit einem Spaltentrennzeichen gearbeitet werden und die Datenfelder dürfen dieses Trennzeichen nicht enthalten.

MAXROWS=x gibt die Anzahl der Sätze an, die maximal als Ergebnis geliefert wird. x ist maximal fünfstellig und im CICS auf 32K begrenzt. **MAXROWS** funktioniert nicht in Verbindung mit CALLs und Stored Procedures.

TRZ=x Für x kann ein beliebiges Sonderzeichen angegeben werden (Default **:**). **N** steht für ‚kein Trennzeichen', die Spalten stoßen in diesem Fall aneinander.

Beispiele:

DEFAULTS DEC=, TRZ=;

Die Parameter des DEFAULT-Commands können in beliebiger Reihenfolge jeweils durch ein Blank getrennt angegeben werden. Der Beispiel-Befehl bewirkt, dass der Daemon numerische Daten mit einem Dezimalkomma an das Programm übergibt und dass die Daten ‚spaltengerecht' auf dem (B)JDBD-File stehen. Jeder Wert ist vom nächsten mit einem Semikolon getrennt.

DEFAULTS

Wird DEFAULTS ohne Parameter angegeben, so gibt der Daemon die gültigen Defaults auf das Ergebnis-File (B)JDBL aus. Bei der Installation können nämlich die hier genannten Standard-Defaults in einem Property File (vergleichbar der Kundenkonfiguration) modifiziert werden.

Aufbau der Command-Datei (B)JDBC 3340

Die Command-Datei (B)JDBC besteht im Wesentlichen aus dem abzusetzenden Befehl.

Dabei ist folgendes zu beachten:

- **+** ist das Folgezeichen. Das Folgezeichen muss gesetzt werden, wenn der SQL-Befehl nicht in einerm Satz steht, sondern in mehreren. CPGJDBC „übersetzt“ bei der Ausführung den Befehl mit Hilfe der Folgezeichen in einen zusammenhängenden Befehl.

- Ein SQL-Befehl darf maximal 32K groß werden. Ist der Befehl größer als 32.000 Zeichen, muss der Programmierer Sorge tragen, dass er in mehrere Teilbefehle zerlegt wird, die dann vom CPGJDBC nacheinander abgearbeitet werden.

Beispiel:

Der abzusetzende Befehl mit etwa 10.000 Values-Zeilen sieht wie folgt aus:

INSERT INTO TABELLE1 (Spalte1,Spalte2,Spalte3,Spalte4) VALUES
(00001,00001,00001,00001),
:
(09999,34567,38541,08001);

Der INSERT-Befehl hat 61 Zeichen, jedes VALUE-Quadrupel hat 26 Zeichen. Bei 10.000 VALUE-Zeilen wird also die zulässige Befehlslänge von 32K mehrfach überschritten. In diesem Fall könnte der Befehl z.B. wie folgt in n INSERTS mit je 500 VALUE-Zeilen zerlegt werden:

```
INSERT INTO TABELLE1 (Spalte1,Spalte2,Spalte3,Spalte4) VALUES  +
(00001,00001,00001,00001),  +
(00002,93975,02947,37464),  +
:                           +
(00500,92502,00235,20958)
INSERT INTO TABELLE1 (Spalte1,Spalte2,Spalte3,Spalte4) VALUES  +
(00501,00001,00001,00001),  +
(00502,92873,93740,92200),  +
:                           +
(01000,92502,00235,20958)
:
```

Kapitel 6: Fehlersuche 6000

CPGJDBC setzt die Fehlermeldungen auf den eigenen Logfile JDBL.

Falls es beim Schreiben der Fehlermeldung auf den Log-Breich zu Fehlern kommt, werden die Meldungen auf der Systemkonsole ausgegeben.

ERROR: database NOTFOUND UNAVAILABLE

database ist der Name der Datenbank und wird in der Phase QJDBCGTB eingetragen.

Prüfen Sie, ob der Name existiert oder ob die LIBDEF in der Partition die Phase QJDBCGTB in der SEARCH Libdef definiert hat.

```
OPEN  IP *O QCFTCPHO  ERROR :  0008 [ | 0016  | 0024  | 0028]
SEND  IP *S QCFTCPHO  ERROR :  0008 [ | 0016  | 0024  | 0028]
RECV  IP *R QCFTCPHO  ERROR :  0008 [ | 0016  | 0024  | 0028]
CLOSE IP *C QCFTCPHO  ERROR :  0008 [ | 0016  | 0024  | 0028]
```

Diese Meldungen betreffen Fehler, die von TCP/IP gemeldet wurden. OPEN IP z.B kann auch auftreten, wenn TCP/IP im VSE nicht gestartet wurde.

Ein Code 0004 ist eine Warning und die Verarbeitung wird normal fortgesetzt. Ansonsten können die Error Codes im TCP/IP Programming Manual nachgelesen werden.

Beispiel: **OPEN IP *O QCFTCPHO ERROR : 0008**

Beim TCP/IP OPEN wurde ein Fehler gemeldet. In der Regel bedeutet das, dass der Listener nicht gestartet ist.

Zusätzlich wird noch die folgende Meldung ausgegeben:

ERROR: TCPIP cannot connect to: 010.000.000.111:4777

Diese Meldung gibt als Information, welcher Host den Listener nicht gestartet hat.

ERROR: TCPIP cannot connect to: ‘ipadr:port’ CPGJDBC not running

zeigt an, dass der Daemon CPGJDBC nicht gestartet ist.

ERROR: TEMPORARY STORAGE PUT ERROR: IOERR | ITEMERR | NOSPACE | OTHER

Bei der Verarbeitung von Temporary Storage wurde ein Fehler vom CICS gemeldet. Wenn NOSPACE auftritt, sollte die Datei DFHTEMP vergrößert oder in einem anderen VSAM Space definiert werden.

ERROR: NO EDSA/ DSA STORAGE AVAILABLE

Für einen GETMAIN von 8K steht kein Speicher zur Verfügung.

Fehler aus den Ergebnisbereichen auslesen __6010

Im Log-File (B)JDBL sind folgende Einträge positive Meldungen:

- READY
- CONNECTED
- ++

Im Daten-File (B)JDBD beginnen die Fehlermeldungen mit:

- Error

Im Programm kann also auf Fehler reagiert werden, z.B. mit einer Liste (online analog):

```
do while cpgfrc = '  '
  read bjdbd.                    * Datenfile auswerten
  if cpgfrc = '  ' and
  fehler = 'Error'.              * Input aus (B)JDBD Stelle 1-5
    list xxxxxx section fehler
  endif.
enddo.
:
do while cpgfrc = '  '
  read bjdbl.                    * Logfile auswerten
  if cpgfrc = '  ' and
  fehler >< 'READY'  and.        * Input aus (B)JDBL Stelle 1-5
  fehler >< 'CONNECTED' and.     * Input aus (B)JDBL Stelle 1-9
  fehler >< '++  '.              * Input aus (B)JDBL Stelle 1-4
    list xxxxxx section fehler
  endif.
enddo.
```

Beispielprogramme 8000

Für CPG5-Anwender: QPG-Beispiel mit Dataset 8010

Für das Ansprechen des Daemons empfehlen wir die Dataset-Technik:

```
options dat
*-------------------------------------------------------------------------*
*  Dataset für Datenbank-Daemon        * §dokname §lib                     *
*-------------------------------------------------------------------------*
file stor type v.                      * Storage mit variablem Namen
-d.
   cpgfrc  2.                          * File Return-Code
   cpgtsn  8.                          * Temporary Storage Name
   dbank   8.                          * Datenbank
   satz  256.                          * Daten
   cpgeds  1. *------------------------*-------------------------------*

-i.
   file stor.
      1 256  satz.
   field cpgsin.
     87  90  tskna.                   * Tasknummer alpha (intern gepackt)
   field cpgcom.
      1   2  cpgfrc.
-c.
 evaluate.                             *
   when cpgfrc = 'N '.                 * WRITE:  Befehl an Daemon übergeben
     write stor.
     fill ' ' satz
   when cpgfrc = 'C '.                 * CLOSE:  Daemon starten
     convert dbank.                    * sicherheitshalber
     edit cpgcom
     progn = 'QJDBCGTO'.               * Konnektor
     prog cicslnk qpg.                 * EXEC CICS LINK
     move  'JDBD' to cpgtsn.           * Namen für die Applikation richtig
                                       * setzen – nächster Schritt ist READ JDBD
     select cpgcom.                    * Return-Code in CPGFRC setzen
   when cpgfrc = 'D '.                 * RANDOM: JDB-Storages löschen
     movel tskna  to cpgtsn.
     move  'JDBC' to cpgtsn.
     purge stor.
     move  'JDBD' to cpgtsn.
     purge stor.
     move  'JDBL' to cpgtsn.
     purge stor.
   when cpgfrc = 'O '.                 * OPEN : JDB-Storages vorbereiten
     select cpgsin.                    * Tasknummer holen
     movel tskna  to cpgtsn.           * Storage-Namen aufbauen
     move  'JDBC' to cpgtsn.           * --       "          --
     cpgfrc = '  '.                    * Init-Wert Blank = ok
 end-evaluate
-o.
   field cpgcom
```

```
      dbank
      17 'TS=JDBC'.
```

Dieses Dataset kann in einer Anwendung wie folgt aufgerufen werden:

```
options dat
*-------------------------------------------------------------------------*
*    Testprogramm für Datenbank-Daemon * §dokname §lib                    *
*-------------------------------------------------------------------------*
file dsdaemon.                         * QPG-Dataset (siehe Vorseite)
file jdbd.                             * Temporary Storage Queue mit vari-
                                       * ablem Namen, im Dataset gesetzt
-d.
   dbank  8.
   satz  256.
 cpgeds  1. *--------------------------*------------------------------*
   cpgtsn  8.                          * Muß hier definiert werden für die
                                       * Kommunikation mit dem Dataset
                                       * cpgtsn = Temporary Storage Name
   f1map  11.                          * Maskeneingabe für F1
   i  3 0
   num70  7 0
   num92  9 2
   page  20 * 79
-i.
   file jdbd
     12  22  F1.                       * Spaltenname der DB, NUMBER(22)
     24  33  F2.                       * Spaltenname der DB, VARCHAR2(10)
     49  56  F3.                       * Spaltenname der DB, NUMBER(22)
-c.
   :
   ',' replace '.' f1map.              * Wert alphanumerisch einlesen,
                                       * Dezimalkomma durch Punkt ersetzen
*-------------------------------------* Befehl absetzen
   dbank = 'ORACLTST'
   edit satz type lesen
   open  dsdaemon.
   write dsdaemon.
   close dsdaemon.
*-------------------------------------* Daten von JDBD lesen
   :
     do while i < 20 and
     while cpgfrc = '  '
       read jdbd.                      * CPGTSN ist vom Dataset richtig gefüllt
       if cpgfrc = '  '
         moven F1 to num92.            * numerische Inhalte in numerische
         moven F3 to num70.            * Felder übertragen
         i = i + 1
         edit page(i).                 * Seite aufbereiten
       endif
     enddo
     mapo tlajdbc
   :
   random dsdaemon
-o.
   field satz type lesen
     'select * from tsttab1 where F1 >= '
     f1map

   field page
     Num92  15 editcode k
     F2     28
```

```
     num70  41 editcode k
```

Im Beispiel wird eine ORACLE-Datenbank unter LINUX verarbeitet.

Für CPG-Anwender: Batch-Beispiel (analog zu 8010) 8020

```
options batch phase jdbcb.
*---------------------------------------------------------------------*
*  Testprogramm für CPGJDBC-Batch      * §dokname §lib                 *
*---------------------------------------------------------------------*
file bjdbc out var 1400 04 rba reuse esds.
file bjdbd inp var 1400 esds no open.
file bjdbl inp var 1400 esds no open.
file reader
file liste
-d.
   f1rdr  11.                         * Readereingabe für F1
   i  3 0
   num70  7 0
   num92  9 2
   parm1  8.                          * Parameterfeld
-i.
   file reader
      1  11 f1rdr.
   file bjdbd
     12  22  F1.                      * Spaltenname der DB, NUMBER(22)
     24  33  F2.                      * Spaltenname der DB, VARCHAR2(10)
     49  56  F3.                      * Spaltenname der DB, NUMBER(22)
-c.
*  :
   read reader
   ',' replace '.' f1rdr.             * Wert alphanumerisch einlesen,
                                      * Dezimalkomma durch Punkt ersetzen
*-------------------------------------* Befehl absetzen
   parm1 = 'ORACLTST'
   excpt bjdbc.                       * Ausgabe über Output-Division
   close bjdbc
   call 'QJDBCGTC'
   parm parm1.
   open bjdbd
*-------------------------------------* Daten von JDBD lesen
*  :
     do while i < 66 and
     while cpgfrc = '  '.              * File Return-Code
       read bjdbd.                    * Daten lesen
       if cpgfrc = '  '
         moven F1 to num92.           * numerische Inhalte in numerische
         moven F3 to num70.           * Felder übertragen
         i = i + 1
         excpt liste.                 * Daten auf Liste ausgeben
       endif
     enddo
*  :
-o.
   file bjdbc add bjdbc
              22  'select * from tsttab1 '
              34  'where F1 >= '
     f1rdr     46
   file liste     liste
     num92  15 editcode k
     F2     28
```

```
     num70  41 editcode k
```

Für COBOL-Anwender: Batch-Beispiel 8030

```
         *   C P G - J D B C - D A E M O N
         *   EINSATZ DARGESTELLT AN EINEM BEISPIEL IN COBOL
         *+++++
         *   DIE SQL-BEFEHLE UND DIE DEFAULTS-ANGABEN WERDEN IN
         *   DIE (COMMAND-) DATEI  BJDBC  AUSGEGEBEN.
         *+++++
         *   DER  C P G J D B C - D A E M O N  (QJDBCGTC)
         *   GIBT DIE GEWUENSCHTEN DATEN IN DIE (DATEN-) DATEI  BJDBD
         *   ERWEITERTE INFORMATIONEN IN DIE (LOGFILE-) DATEI BJDBL  AUS.
         *+++++
         *   DIE DATEIEN BJDBC, BJDBL, BJDBD WERDEN PROTOKOLLIERT.

*************************************************************************
               IDENTIFICATION DIVISION.
               PROGRAM-ID.       B2JDBC.
               AUTHOR.           INGEBORG BIERGANS.
               DATE-WRITTEN.     SEPTEMBER 2004.

*************************************************************************
               ENVIRONMENT DIVISION.
             *-----------------------------------------------------------
               CONFIGURATION SECTION.
               SPECIAL-NAMES.
               DECIMAL-POINT IS COMMA
               C01 IS KANAL01.
             *
               INPUT-OUTPUT SECTION.
               FILE-CONTROL.
                  SELECT COMMAND-DATE  ASSIGN TO SEQUENTIAL-AS-BJDBC
                                       ACCESS        IS SEQUENTIAL
                                       ORGANIZATION  IS SEQUENTIAL
                                       FILE STATUS   IS FC-STATUS.
             *
                  SELECT LOG-DATEI     ASSIGN TO SEQUENTIAL-AS-BJDBL
                                       ACCESS        IS SEQUENTIAL
                                       ORGANIZATION  IS SEQUENTIAL
                                       FILE STATUS   IS FL-STATUS.
             *
                  SELECT DATEN-DATEI   ASSIGN TO SEQUENTIAL-AS-BJDBD
                                       ACCESS        IS SEQUENTIAL
                                       ORGANIZATION  IS SEQUENTIAL
                                       FILE STATUS   IS FD-STATUS.
             *
                  SELECT LISTE         ASSIGN TO SYS020-UR-3211-S-PRINTER.
*************************************************************************
```

```
DATA DIVISION.
*-----------------------------------------------------------------------
          FILE  SECTION.
          FD  COMMAND-DATEI
                 RECORD IS VARYING FROM 100 TO 1400 CHARACTERS
                 DATA RECORD IS COMMAND-PUFFER.
              01 COMMAND-PUFFER.
                    03 DBINF01            PIC X(50).
                    03 DBINF02            PIC X(50).
                    03 FILLER             PIC X(1300).
          FD  LOG-DATEI
                 RECORD IS VARYING FROM 100 TO 1400 CHARACTERS
                 DATA RECORD IS LOG-PUFFER.
              01 LOG-PUFFER.
                    03 LOG1               PIC X(50).
                    03 LOG2               PIC X(50).
                    03 FILLER             PIC X(1300).
          FD  DATEN-DATEI
                 RECORD IS VARYING FROM 100 TO 1400 CHARACTERS
                 DATA RECORD IS DATEN-PUFFER.
              01 DATEN-PUFFER.
                    03 DATEN1             PIC X(50).
                    03 DATEN2             PIC X(50).
                    03 FILLER             PIC X(1300).
          FD  LISTE
                 RECORDING MODE IS F
                 LABEL RECORD   IS OMITTED.
              01 LST-ZEILE              PIC X(132).
WORKING-STORAGE SECTION.
*-----------------------------------------------------------------------
  01  DATENBANK-NAME.
      05 PARM                          PIC X(8).
         01  SEITEN-ZAEHLER             PIC S9(4)  VALUE ZEROS.
         01  ZEILEN-ZAEHLER             PIC S9(2)  VALUE ZEROS.
        *
         01  FC-STATUS                   PIC 99.
               88 FC-ERFOLGREICH  VALUE 00 97.
         01  FL-STATUS                   PIC 99.
               88 FL-ERFOLGREICH  VALUE 00 97.
         01  FD-STATUS                   PIC  99.
               88 FD-ERFOLGREICH  VALUE 00 97.
        *
         01  KOPFZEILE-1.
               03 FILLER                PIC X(17) VALUE SPACES.
               03 FILLER                PIC X(43) VALUE
                         'PROTOKOLL   C P G - J D B C - D A E M O N'.
               03 FILLER                PIC X(2).
               03 DATEI-INFO            PIC X(5).
               03 FILLER                PIC X(56) VALUE SPACES
               03 FILLER                PIC X(5)  VALUE 'SEITE'.
               03 SEITE                 PIC ZZZ9.
          01  KOPFZEILE-2.
               03 FILLER                PIC X(132) VALUE ALL '-'.

         01  POSTENZEILE-1.
               03 A-INFO1               PIC X(50).
        *
```

```
*****************************************************************************
       PROCEDURE DIVISION.
*****************************************************************************
*++++++++  1. TEIL       EINSATZ DES CPGJDBC-DAEMONS                      ++++
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                OPEN OUTPUT  COMMAND-DATEI.
                IF NOT FC-ERFOLGREICH
                   DISPLAY 'OPEN FEHLER BEI DATEI-BJDBC'  FC-STATUS
                   GO TO ENDE
                END-IF.
         *+++++  ANGABEN  DATENFILES
         *
                MOVE 'DEFAULTS DDOPT=YES, MAXROWS=15;'      TO DBINFO1.
                WRITE COMMAND-PUFFER.
         *
         *+++++  ANGABEN  DATENBANK-COMMANDS
         *
                MOVE 'SELECT * FROM CPGKSD ORDER BY KDNRA'  TO DBINFO1.
                WRITE COMMAND-PUFFER.
                CLOSE COMMAND-DATEI.
         *
         *+++++  DATENBANK-NAME UND EINSATZ CPGJDBC-DAEMON
         *
                MOVE 'CPGDB   '  TO  PARM.
                CALL 'QJDBCGTC'  USING PARM.
         *
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*++++++++   2. TEIL       PROTOKOLL DER DATEIEN BJDBC,BJDBL,BJDBD      ++++
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                OPEN OUTPUT LISTE.
         *
                OPEN INPUT COMMAND-DATEI.
                IF NOT FC-ERFOLGREICH
                   DISPLAY 'OPEN FEHLER BEI DATEI-BJDBC'  FC-STATUS
                   GO TO ENDE
                END-IF.
         *
                OPEN INPUT LOG-DATEI.
                IF NOT FL-ERFOLGREICH
                   DISPLAY 'OPEN FEHLER BEI DATEI-BJDBL'  FL-STATUS
                   GO TO ENDE
                END-IF.
         *
                OPEN INPUT DATEN-DATEI.
                IF NOT FD-ERFOLGREICH
                   DISPLAY 'OPEN FEHLER BEI DATEI-BJDBD'  FD-STATUS
                   GO TO ENDE
                END-IF.
         *
                MOVE 61  TO   ZEILEN-ZAEHLER.
                MOVE 'BJDBC' TO  DATEI-INFO.
           LESEN-COMMAND-DATEI.
                IF ZEILEN-ZAEHLER > 60
                   PERFORM KOPFZEILEN.
                   READ COMMAND-DATEI AT END GO TO LOG.
                IF FC-ERFOLGREICH THEN NEXT SENTENCE
                   ELSE DISPLAY FC-STATUS
                   GO TO ENDE.
         *
                MOVE DBINFO1    TO   A-INFO1.
```

```
                MOVE DBINFO2    TO   A-INFO2.
                PERFORM POSTENZEILE.
        *
                GO TO LESEN-COMMAND-DATEI.
        *
           LOG.
        *
                MOVE 61  TO   ZEILEN-ZAEHLER.
                MOVE 'BJDBL' TO  DATEI-INFO.
           LESEN-LOG-DATEI.
                IF ZEILEN-ZAEHLER > 60
                   PERFORM KOPFZEILEN.
                   READ LOG-DATEI     AT END GO TO DAT.
                IF FL-ERFOLGREICH THEN NEXT SENTENCE
                   ELSE DISPLAY FL-STATUS
                   GO TO ENDE.
        *
                MOVE LOG1       TO   A-INFO1.
                MOVE LOG2       TO   A-INFO2.
                PERFORM POSTENZEILE.
        *
                GO TO LESEN-LOG-DATEI.
        *
           DAT.
                MOVE 61  TO   ZEILEN-ZAEHLER.
                MOVE 'BJDBD' TO  DATEI-INFO.
           LESEN-DATEN-DATEI.
                IF ZEILEN-ZAEHLER > 60
                   PERFORM KOPFZEILEN.
                   READ DATEN-DATEI   AT END GO TO ENDE.
                IF FD-ERFOLGREICH THEN NEXT SENTENCE
                   ELSE DISPLAY FD-STATUS
                   GO TO ENDE.
        *
                MOVE DATEN1     TO   A-INFO1.
                MOVE DATEN1     TO   A-INFO2.
                PERFORM POSTENZEILE.
        *
                GO TO LESEN-DATEN-DATEI.
           ENDE.
                CLOSE COMMAND-DATEI.
                CLOSE LOG-DATEI.
                CLOSE DATEN-DATEI.
                CLOSE LISTE.
           STOP  RUN.
*****************************************************************************
                *  POSTENZEILEN PROTOKOLL
                *-------------------------------------------------------------
                 POSTENZEILE.
                     WRITE LST-ZEILE       FROM POSTENZEILE-1 AFTER 1.
                     MOVE  SPACES          TO   LST-ZEILE.
                     ADD   1               TO   ZEILEN-ZAEHLER.
                 POSTENZEILE-ENDE.
                *-------------------------------------------------------------
                *  KOPFZEILEN PROTOKOLL
                *-------------------------------------------------------------
                 KOPFZEILEN.
                     ADD   1               TO   SEITEN-ZAEHLER.
                     MOVE  SEITEN-ZAEHLER  TO   SEITE.
                     WRITE LST-ZEILE       FROM KOPFZEILE-1 AFTER KANAL01.
                     WRITE LST-ZEILE       FROM KOPFZEILE-2 AFTER 2.
                     MOVE  SPACES          TO   LST-ZEILE.
                     MOVE  4               TO   ZEILEN-ZAEHLER.
```

KOPFZEILEN-ENDE.

Für COBOL-Anwender: CICS-Beispiel 8040

```
*****************************************************************
*  NAME       DATUM      ZEIT   DOKUMENTATION                    *
*  ---------- ---------- -----  --------------------------------*
*  MAASSEN    25-03-2004 14.00  001. SAMPLE ODBC GET            *
*  TRANSID    TC09  PGM Q2JDBC  999. LETZTE AENDERUNG
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    Q2JDBC.
 AUTHOR.        MAASSEN.
 DATE-WRITTEN.  25/03/04.
 DATE-COMPILED.
*LAST-MODIFIED. 25/03/04.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 DATA DIVISION.

 WORKING-STORAGE SECTION.

 01  FILLER                          PIC X(40) VALUE
                  'Q2JD WORKING STORAGE BEGINNT HIER =====>'.
 01  PROGRAM-NAME                    PIC  X(8) VALUE 'Q2JDBC  '.
 01  ZERO-BIN                        PIC S9(9) VALUE 0.
 01  LENTSD                          PIC S9(4) BINARY VALUE 500.
 01  LENTSC                          PIC S9(4) BINARY VALUE 64.
 01  LENTSL                          PIC S9(4) BINARY VALUE 200.
 01  PGMNAM                          PIC  X(8) VALUE 'QJDBCGTO'.
 01  TSNAMC.
   02 TSNAMC-TASKNR                 PIC S9(7) COMP-3.
   02 TSNAMC-NAME                   PIC  X(4).
 01  TSNAMD.
   02 TSNAMD-TASKNR                 PIC S9(7) COMP-3.
   02 TSNAMD-NAME                   PIC X(04).
 01  TSNAML.
   02 TSNAML-TASKNR                 PIC S9(9) COMP-3.
   02 TSNAML-NAME                   PIC X(04).
 01  SQLCOM                          PIC  X(64) VALUE
                  'SELECT * FROM CPGKSD ORDER BY KDNRA     '.
 01  SQLDATA                         PIC  X(500).
 01  SQLLOGD                         PIC  X(200).
*---------------------------------------------------------------*
 01  COMMAREA1.
  02  DBNAME                       PIC X(08).
  02  RFREE                        PIC X(02).
  02  TSNAM                        PIC X(07).
 01  WORK-EIBCALEN                  PIC S9(5).
  EJECT
*---------------------------------------------------------------*
* LINKAGE SECTION                                               *
*---------------------------------------------------------------*
 LINKAGE SECTION.
  EJECT
*---------------------------------------------------------------*
* EXEC-INTERFACE-BLOCK   UND                                    *
* COMMUNICATION-AREA ZUM HAUPT-PROGRAMM                         *
*---------------------------------------------------------------*
 01  DFHCOMMAREA.
  02   RETC                        PIC X(2).
  EJECT
*---------------------------------------------------------------*
```

```
 PROCEDURE DIVISION.
*-----------------------------------------------------------------*
 PROG-START.
*-----------------------------------------------------------------*
*
     MOVE EIBCALEN TO WORK-EIBCALEN.
*
     EXEC CICS HANDLE CONDITION ERROR(PROG-ENDE) END-EXEC
*
     PERFORM JDBC-COMMND
     PERFORM JDBC-GET
     PERFORM JDBC-READ.
*-----------------------------------------------------------------*
 PROG-ENDE.
*-----------------------------------------------------------------*
     EXEC CICS RETURN END-EXEC.
*-----------------------------------------------------------------*
  EJECT
*-----------------------------------------------------------------*
 JDBC-BEGINN.
     MOVE EIBTASKN TO TSNAMC-TASKNR
     MOVE EIBTASKN TO TSNAMD-TASKNR
     MOVE EIBTASKN TO TSNAML-TASKNR
     MOVE 'JDBC'   TO TSNAMC-NAME
     MOVE 'JDBD'   TO TSNAMD-NAME
     MOVE 'JDBL'   TO TSNAML-NAME.
*-----------------------------------------------------------------*
 JDBC-COMMND.
     MOVE 64 TO       LENTSC
     MOVE 'SELECT * FROM CPGKSD ORDER BY KDNRA     ' TO SQLCOM
     EXEC CICS WRITEQ TS QUEUE(TSNAMC)
          FROM(SQLCOM) LENGTH(LENTSC) END-EXEC.
*-----------------------------------------------------------------*
 JDBC-GET.
     MOVE 'ORA92NET'  TO DBNAME
     MOVE '  '        TO RFREE
     MOVE 'TS=    '   TO TSNAM
*-----------------------------------------------------------------*
     MOVE 'ORA92NET'  TO DBNAME
     MOVE 'QJDBCGTO'  TO PGMNAM
     EXEC CICS LINK PROGRAM(PGMNAM) COMMAREA(COMMAREA1)
          LENGTH(17) END-EXEC.
*-----------------------------------------------------------------*
 JDBC-READ.
     MOVE 500 TO LENTSD
     EXEC CICS READQ TS QUEUE(TSNAMD)
          INTO(SQLDATA) LENGTH(LENTSD) END-EXEC
     EXEC CICS READQ TS QUEUE(TSNAMD)
          INTO(SQLDATA) LENGTH(LENTSD) END-EXEC.
*-----------------------------------------------------------------*
```