ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Prof. H.-J. Schek *A. Feßler, U. Röhm, H. Schuldt, R. Weber*

Informationssysteme IS-G SS 1997

Übung 4 Abgabe: 6.5.1997

Aufgabe 1: Programmierung mit "Embedded SQL"

Gegeben sei die Musik-Datenbank der Übung 3. Schreiben Sie ein C-Programm mit eingebettetem SQL, das einem Sachbearbeiter eines Musikarchivs dazu dient, neue Platten zu erfassen. Zur Vereinfachung können Sie dabei die Information über Interpreten und Autoren von Musikstücken vernachlässigen.

Der Sachbearbeiter soll zunächst den vollständigen Titel einer Platte eingeben. Falls dieser Titel noch nicht in der Datenbank existiert, sollen danach die Titel der auf der Platte enthaltenen Musikstücke eingegeben werden, ansonsten ist eine entsprechende Fehlerbehandlung vorzusehen (z.B. kann der Benutzer entscheiden, die Platte trotzdem zu erfassen oder die Erfassung abzubrechen). Die DiskID soll von Ihrem Programm automatisch generiert werden; der Wert der DiskID ist irrelevant, solange er eindeutig ist. Nach diesem Schema soll der Sachbearbeiter mehrere Platten interaktiv erfassen können.

Hinweise:

- Nachdem Sie in der letzten Übung Anfragen an die Datenbank über einen Web-Browser gestellt haben, wollen wir in dieser Übung Anwendungen mit Hilfe des Datenbanksystems ORACLE erstellen. Benutzen Sie dazu Ihr Konto auf dem RIF/RAF Cluster sowie den in der Übung vergebenen ORACLE-Account. Falls Sie noch keinen Zugang zu einem Konto besitzen, kontaktieren Sie bitte Ihren Assistenten. Damit Sie überhaupt mit ORACLE arbeiten können, müssen Sie folgenden Befehl eingeben: `module add oracle/stud` (bitte fügen Sie diesen Befehl nicht in Ihr .cshrc-File ein, da Sie sonst möglicherweise Probleme beim Einloggen bekommen könnten).
- Im Verzeichnis ~infosys/esqlbsp finden Sie ein Beispielprogramm mit zugehörigem Makefile, welches die Einbettung von SQL in C demonstriert (Generieren Sie das Programm mit `make USERID=`*id/passwd* `simpledisk`). Benutzen sie das Programm als Ausgangspunkt für Ihre Programmentwicklung. Das Beispiel finden Sie ebenfalls auf der Rückseite dieser Aufgabenstellung.
- In dieser Übung arbeiten Sie mit der Musikdatenbank, die unter dem User IS0 angelegt ist. Beachten Sie, dass Sie auf die Tabellen des ORACLE-Users IS0

nur lesenden Zugriff haben. Um Ihr Erfassungsprogramm zu testen, müssen Sie sich eigene Tabellen anlegen. Dabei können Sie sich auf das Erzeugen lokaler Tabellen `Disk` und `Musikstück`, die jeweils nicht mit Testdaten gefüllt sein müssen, beschränken. Verwenden Sie hierzu den Editor SQLPLUS. Mit dem Befehl `sqlplus` können Sie diesen starten. Danach werden Sie nach User–Name und Passwort gefragt. Nach richtiger Eingabe können Sie SQL-Befehle eingeben (mit `;` abschliessen !), die danach in der Datenbank ausgeführt werden.
Die Online-Hilfe für ORACLE7 können Sie in `sqlplus` mit dem Befehl `help` anfordern. `help commands`, `help menu`, `help bnf syntax` geben eine gute Übersicht über die verhandenen Hilfstexte. Hilfe für den Befehl `select` erhalten Sie z.B. mit `help select`.
Mit den Befehlen `quit` bzw. `exit` können Sie den Editor wieder verlassen.

- Konzentrieren Sie sich bei dieser Aufgabe auf die SQL-Anweisungen und deren Einbettung. Ver(sch)wenden Sie nicht zu viel Zeit mit der Gestaltung der Dialogschnittstelle!

- **Abgabe-Modus**: Geben Sie bitte einen Ausdruck des lauffähigen ESQL-Programms, mit Ihrem Namen versehen, bei Ihrem Betreuer ab.

Beispielprogramm

Folgendes Beispielprogramm liest Autorennamen und Titel ein und führt eine LIKE–Anfrage auf der Musikdatenbank durch:

```
#include <ctype.h>
#include <stdio.h>
#define TRUE    1
#define ANL    21
#define TL     41

void __cg92_used( void )
{/* needed when linking gcc compiled code with Oracle Embedded SQL */}

/* useful macro */
#define resize(A) A.arr[A.len]=(char)0

/* variable declarations */
char autname[ANL], titel[TL];

/* declaration of hostvariables used in SQL statements */
EXEC SQL BEGIN DECLARE SECTION;
        VARCHAR userid[20];         /* user identification           */
        VARCHAR password[20];       /* password                      */
        VARCHAR catstr[64];         /* concatenated string for query */
        int     DiskID;
        VARCHAR DiskTitel[255];
/* For VARCHAR host_var[40] the precompiler generates the structure
   struct { unsigned short len;     /o number of character o/
            unsigned char  arr[40]  /o string              o/
          } host_var; */
EXEC SQL END DECLARE SECTION;

/* include SQL communication area */
EXEC SQL INCLUDE sqlca.h;

main(int argc, char *argv[])
{
   if (argc!=3)
      {printf("USAGE: %s <account> <password>\n", argv[0]); return;}

   strcpy(userid.arr, argv[1]); strcpy(password.arr, argv[2]);
   userid.len=strlen(userid.arr); password.len=strlen(password.arr);

   /* the program exits if any errors occur in EXEC SQL statements */
   EXEC SQL WHENEVER SQLERROR GOTO errexit;

   /* logon to ORACLE */
   EXEC SQL CONNECT :userid IDENTIFIED BY :password;
```

```
   printf("\nConnected to data base\n");

   /* Ask for input */
   printf("\nAutorenname? ");
   scanf("%20s%*[ -~]", autname );  /* Truncate any longer input */
   getchar();                       /* Consume EOL               */

   printf("Musiktitel? "); scanf("%40s%*[ -~]", titel ); getchar();

   sprintf(catstr.arr, "%%%s%%%s%%", autname, titel);
   catstr.len = strlen(catstr.arr);

   /* define cursor for SQL queries and open cursor */
   EXEC SQL DECLARE DISK CURSOR FOR
        SELECT * FROM ISO.Disk WHERE DiskTitel LIKE :catstr;
   EXEC SQL OPEN DISK;

   /* define label, where to continue, if no more tuple found */
   EXEC SQL WHENEVER NOT FOUND GOTO endloop;

   /* process appropriate disks */
   printf("DiskID\tDiskTitel\n======\t=========\n\n");
   while (TRUE) {                /* endless loop */
      EXEC SQL FETCH DISK INTO :DiskID, :DiskTitel;
      resize(DiskTitel);
      printf("%6.6d\t%s\n", DiskID, DiskTitel.arr);
   }

endloop:
   EXEC SQL CLOSE DISK;

   /* commit transaction and logout from ORACLE */
   EXEC SQL COMMIT WORK RELEASE;
   return;

/* error handling section for EXEC SQL errors */
errexit:
   errrpt();
   EXEC SQL WHENEVER SQLERROR CONTINUE;
   fprintf(stderr,"\nSQL-ERROR  -->  ROLLBACK\n");
   EXEC SQL ROLLBACK WORK RELEASE;
   return;
}

/* errrpt prints the ORACLE error msg and number */
errrpt()
{ printf("\n%.70s (%d)\n", sqlca.sqlerrm.sqlerrmc, sqlca.sqlcode);
  return(0);
}
```