1 Embedded sql

SQL statements kunnen in een programma in een klassieke programmeertaal opgenomen worden, bijvoorbeeld C. Hiervoor zijn twee redenen.

- Bepaalde queries kunnen niet met behulp van pure SQL geformuleerd worden (bijv. recursieve queries). Er is dus behoefte aan een taal met een expressievere kracht.
- De gegevens uit een databank moeten beschikbaar gesteld worden in een applicatie die in één of andere hogere programmeertaal geschreven is. Bijvoorbeeld een vliegtuigreservatiesysteem met een grafische user interface in C waarbij de informatie van de bezetting van de vliegtuigen in een databank opgeslagen wordt.

1.1 Een eenvoudig voorbeeld

In het volgende voorbeeld zitten gewone C statements en embedded SQL (ESQL) statements. De ESQL statements worden met behulp van een *precompiler* (bijv. `proc` of `ecpg`) omgezet in statements van de gasttaal.

```
1 #include <stdio.h>
  #include <sys/semaphore.h>
3 int main(int argc, char *argv[])
  {
5       EXEC SQL BEGIN DECLARE SECTION;
            char gebruiker[12] = "naam/pasw";
7           char geg[5] = "L3";
            char stad[16];
9           int  rang = 0;
        EXEC SQL END DECLARE SECTION;
11
        EXEC SQL include sqlca;
13      if ( argc > 1 )
        {
15          strcpy(geg, argv[1]);
        }
17      memset(stad, '\0', 16);
        EXEC SQL CONNECT :gebruiker;
19      EXEC SQL SELECT lstat, pla INTO :rang, :stad
                FROM L
21              WHERE lnr = :geg;
        if ( sqlca.sqlcode != 0)
23          printf("select fout? : %d %s \n",
                    sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
25      else
            printf("   %s   --> %d %-16.16s\n", geg,  rang, stad);
27 }
```

- Ingebedde SQL statements worden voorafgegaan door `EXEC SQL`.
- Ingebedde SQL statements kunnen overal waar andere statements voorkomen, geplaatst worden.
- In de `DECLARE SECTION` worden de *shared* variabelen gedeklareerd. In de gasttaal (hier C) worden deze variabelen met de naam aangesproken. In ingebedde SQL statements worden deze variabelen voorafgegaan door een dubbele punt (`:`).
- Tijdens de uitvoering van bovenstaand voorbeeld krijgen de variabelen `rang` en `stad` een waarde: opvragen van informatie uit de databank.

- Aanpassingen, verwijderen en toevoegen zijn ook mogelijk.

Voor het doorgeven van een foutcode en andere statusvariabelen vanuit de database server (bijv. *Oracle*) naar het C programma wordt de variabele `sqlca` (SQL Communication Area) gebruikt.

```
struct sqlca
{
    char        sqlcaid[8];
    long        sqlabc;
    long        sqlcode;                                    /* fout code */
    struct
    {
        int         sqlerrml;
        char        sqlerrmc[70];             /* beschrijving van de fout */
    }           sqlerrm;
    char        sqlerrp[8];
    long        sqlerrd[6];                 /* element [2]: aantal tuples */
    char        sqlwarn[8];
    char        sqlext[8];
}  sqlca;
```

Wanneer het veld `sqlcode` gelijk is aan 0, dan is de aktie zonder fout verlopen. Bij een waarde kleiner dan nul is er een ernstige fout opgetreden, bijvoorbeeld (-1034) geeft aan dat geprobeerd is een connectie te maken naar een niet bestaande databank. Een waarde groter dan nul geeft een gewone fout aan, de waarde 1403 bijvoorbeeld geeft aan dat er geen data gevonden is of dat de *cursor* uitgeput is.

1.2 Cursor

In het vorige voorbeeld is het resultaat van de query één rij waarvan de waarden van de verschillende attributen toegekend worden aan de verschillende C-variabelen. Het resultaat van een query is meestal een verzameling van waarden. De meeste gasttalen kunnen echter moeilijk overweg met verzamelingen. Er is dus een mechanisme nodig om elke tuple uit een verzameling tuples afzonderlijk te behandelen. Deze aangepaste manier om SQL met de gasttaal te koppelen, gebeurt met behulp van een *cursor* die doorheen de rijen van een relatie loopt. In het volgende voorbeeld wordt de creatie en het gebruik van een cursor geïllustreerd.

```
1 #include <stdio.h>
  int main(int argc, char *argv[])
3 {
      EXEC SQL BEGIN DECLARE SECTION;
5             char gebruiker[12]  = "naam/pasw";
              char nr[5] ;
7             char naam[20] ;
              char stad[20] = "Peulis";
9             int  rang=0;
      EXEC SQL END DECLARE SECTION;
11
      EXEC SQL include sqlca;
13    memset(&sqlca, '\0', sizeof(sqlca) );
      if ( argc > 1 )
15    {
          strcpy(stad, argv[1]);
17    }
      EXEC SQL CONNECT :gebruiker;
19    EXEC SQL DECLARE x CURSOR FOR
```

```
                SELECT lnr, lnaam, lstat FROM L WHERE pla = :stad;
21      if ( sqlca.sqlcode != 0 )
        {
23          printf("cursor fout?  : %d %s \n",
                    sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
25          EXEC SQL ROLLBACK WORK RELEASE;
            exit (2);
27      }
        EXEC SQL OPEN x;
29      if ( sqlca.sqlcode != 0 )
        {
31          printf("open fout?  : %d %s \n",
                    sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
33          EXEC SQL ROLLBACK WORK RELEASE;
            exit (3);
35      }
        while ( 1 )
37      {
            EXEC SQL FETCH x INTO :nr, :naam, :rang;
39          if ( sqlca.sqlcode != 0)
            {
41              printf("fetch fout? : %d %s aantal %d \n",
                     sqlca.sqlcode,sqlca.sqlerrm.sqlerrmc,sqlca.sqlerrd[2]);
43              break;
            }
45          printf("%-5.5s %-16.16s %d %-16.16s\n", nr, naam, rang, stad);
        }
47      EXEC SQL CLOSE x;
    }
```

Op lijnen 15 en 16 wordt de cursor gedeclareerd. Lijn 24 initialiseert de cursor op een positie zodat de eerste rij uit de tabel kan opgehaald worden. In lijn 34 wordt de volgende rij uit de tabel gehaald waarover de cursor gedefinieerd is. Lijn 43 koppelt de cursor los van de tabel.
De output van bovenstaand programma:

```
L1    Jan             20 Peulis
L4    Luc             20 Peulis
fetch fout? : 1403 ORA-01403: no data found
                                                    aantal 2
```

De derde lijn wordt geproduceerd door de `printf` van lijnen 37 en 38. In wezen is er geen echte fout opgetreden, maar zijn alle tuples uit de cursor behandeld geweest. Dit wordt aangegeven door de variabele `sqlca.sqlerrd[2]`: de waarde hiervan is het aantal behandelde tuples in de cursor.

In het volgende voorbeeld wordt geïllustreerd dat naast `SELECT` ook `UPDATE` statements mogelijk zijn.

```
  #include <stdio.h>
2 EXEC SQL include sqlca;
  void sqlprint(void)
4 {
          printf("****** fout?  : %d %s \n",
6                   sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
                  exit(3);
8 }
```

```
10 EXEC SQL WHENEVER SQLERROR do sqlprint();
   int main(int argc, char *argv[])
12 {
       int i;
14     EXEC SQL BEGIN DECLARE SECTION;
          char gebruiker[12]  = "naam/pasw";
16        char  art1[20];
          char  art2[20];
18        int   res;
       EXEC SQL END DECLARE SECTION;
20
       if ( argc < 2 )
22     {
           fprintf(stderr, "Gebruik: la3 artnr artnr\n");
24         exit(1);
       }
26     strcpy(art1, argv[1]);
       strcpy(art2, argv[2]);
28     EXEC SQL CONNECT :gebruiker;
       EXEC SQL  UPDATE a
30               SET gewicht = gewicht * 2
                 WHERE anr IN ( :art1, :art2 );
32     printf("aantal in artikel : %d\n", sqlca.sqlerrd[2]);
       EXEC SQL  DECLARE z CURSOR FOR
34               SELECT anr,gewicht FROM a;
       EXEC SQL OPEN z;
36     printf("aantal in artikel : %d\n", sqlca.sqlerrd[2]);
       EXEC SQL   WHENEVER NOT FOUND DO BREAK;
38     for ( i=1; i<10; i++ )
       {
40         EXEC SQL   FETCH z INTO :art1, :res;
           printf("Res: %-10.10s %4d (reeds %2d)\n",
42                                art1, res, sqlca.sqlerrd[2]);
       }
44     EXEC SQL  CLOSE z;
       EXEC SQL  COMMIT WORK RELEASE;
46 }
```

```
Resultaat bij ./la3 A2 A4 : aantal in artikel : 2
                            aantal in artikel : 0
                            Res: A1          26 (reeds  1)
                            Res: A2          62 (reeds  2)
                            Res: A3          17 (reeds  3)
                            Res: A4          48 (reeds  4)
                            Res: A5          12 (reeds  5)
                            Res: A6          29 (reeds  6)
```

Database aanpasingen door middel van INSERT, UPDATE of DELETE statements moeten geCOMMIT worden, voordat het programma beëindigd wordt.