Linux における Embedded SQL

Embedded SQL は、C または C++ ソース・コード内に直接記述する SQL 文で構成されます。これらの SQL 文は、**SQL プリプロセッサ**によって C または C++ ソース・コードに変換されます。SQL プリプロセッサは、コンパイルの前に実行されます。このプリプロセッサは、DBMS ライブラリ関数を参照してデータベース・エンジンでコマンドを実行する適切な C コードに SQL 文を変換します。これらの文は、結果のオブジェクト・コードにハードコードされるため、静的文と呼ばれます。

ASA 用の ESQL アプリケーションに必要な環境変数

埋め込み C アプリケーションを Linux でコンパイルする場合は、ASA の実行に必要な変数のほかに PLATFORM 変数を設定することをおすすめします。Redhat Linux でシステム全体にわたって (すべてのユーザについて) その変数を設定するには、/etc/profile を編集して変更を加えます。

PLATFORM=LINUX

また、次を実行して、常に変数をエクスポートする必要があります。

export PLATFORM

Adaptive Server Anywhere のインストールには、2 つの Embedded SQL サンプルが含まれています。静的カーソルの Embedded SQL サンプルである cur.sqc では、静的 SQL 文を使用しています。動的カーソルの Embedded SQL サンプルである dcur.sqc では、動的 SQL 文を使用しています。これらのサンプル以外にも、特定のプラットフォームで利用できる機能を示したプログラムとソース・ファイルが Adaptive Server Anywhere のインストールの一部として用意されています。

サンプルのソース・コードは、Adaptive Server Anywhere インストールの一部としてインストールされます。これらは、Adaptive Server Anywhere インストール・ディレクトリの cxmp サブディレクトリに配置されます。

サンプル・プログラムには、シェル・スクリプト makeall が付属しています。このスクリプトを使用すると、Linux 用のサンプル・プログラムをコンパイルすることができます。

makeall スクリプトからの抜粋:

```
--------------------------------
# makefile for UNIX SQLAnywhere examples
#
# USAGE:    make -f makeall
# OR:         make -f makeall clean (to clean up object files)
#
# NOTE: this makefile expects the following environment
# variables to be set (see below for details):
# COMPILER compiler to be used
# ASANY location of the ASA6 installation
# (default: /opt/SYBSasa6)
# ODBC location of the ODBC driver manager installation
# (default: /opt/odbc)
```

```
#       PLATFORM define this as: SUN if running on Solaris
# HP if running on HP-UX
# AIX if running on AIX
# LINUX if running on Linux
#
#============================================================
#
# COMPILER variable should be set in the environment,
# otherwise, it defaults to gnu.
# Alternatively, it can be set in the command line for make.
# available compilers are:
# gnu for GNU gcc
# sun_native for SUN Workshop compiler C/C++ 4.2
# hp_native for HP
# aix_native for IBM AIX
#
#============================================================
#
# Compiler defines:
#
# CC name of the C compiler
# _COMMON_FLAGS compiler flags common to both C and C++
# _CFLAGS user-definable flags for the C compiler
# OBJDIR directory where the object files are to be placed
# (this variable can be set in the command line for make)
# _LIBS platform-specific libs
#
# The _CFLAGS macro is optional, and user-definable below.
#
```

簡単な例

ESQL の強力な点は、特別に宣言した C 変数を SQL 文に直接埋め込んで、結果を受け取ったり変数をランタイム・クエリに渡すことができることです。このような変数はホスト変数と呼ばれます。上記のリストでは、ホスト変数を fetch_rows 関数で使用して、クエリの結果を標準 C 変数に直接渡しています。

```
EXEC SQL FETCH RELATIVE 1 C1
INTO :emp_id, :name, :sex, :birthdate;
```

emp_id、name、sex、および birthdate は、クエリからの戻り値を含めるために設計された基本的な C ポインタです。この場合のクエリでは、あらかじめ以下のコードで設定されたカーソルから次のローがフェッチされます。

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT emp_id, emp_fname || ' ' || emp_lname, sex, birth_date
FROM "dba".employee;
```

式 emp_fname || ' ' || emp_lname では、2 つの文字型カラム (emp_fname と emp_lname) が結合されて 1 つの文字型値になります。この値は、クエリから戻されます。

このサンプル・プログラムを機能させるには、サンプル・ソースをファイル ctest.sqc にコピーし、そのファイルに対して SQLC プリプロセッサを実行します。

```
$ sqlpp ctest.sqc
```

これにより、C ファイル ctest.sqc が出力されます。ユーザは、このファイルをコンパイルして、関連ライブラリの指定とディレクトリのインクルードを処理する必要があります。

```
$ g++ -Wall -I/opt/SYBSasa6/include/ -L/opt/SYBSasa6/lib/ -DUNIX -DODBC_UNIX -ldblib6 -o ctest ctest.c
```

これにより、実行プログラム ctest が生成されます。このプログラムを実行する前に、サンプル・データベースを実行しているエンジンがないことを確認してください。もう一度、次のように入力します。

```
$ dbeng6 asademo.db
```

出力は次のようになります。

```
$ ./ctest
(102, Fran Whitney, F, 1958-06-05)
(105, Matthew Cobb, M, 1960-12-04)
(129, Philip Chin, M, 1966-10-30)
(148, Julie Jordan, F, 1951-12-13)
(160, Robert Breault, M, 1947-05-13)
(184, Melissa Espinoza, F, 1939-12-14)
(191, Jeannette Bertrand, F, 1964-12-21)
(195, Marc Dill, M, 1963-07-19)
(207, Jane Francis, F, 1954-09-12)
(243, Natasha Shishov, F, 1949-04-22)
(247, Kurt Driscoll, M, 1955-03-05)
(249, Rodrigo Guevara, M, 1956-11-23)
(266, Ram Gowda, M, 1947-10-18)
(278, Terry Melkisetian, F, 1966-05-17)
(299, Rollin Overbey, M, 1964-03-15)
(316, Lynn Pastor, F, 1962-07-14)
(318, John Crow, M, 1962-04-24)
(390, Jo Ann Davidson, F, 1957-02-17)
(409, Bruce Weaver, M, 1946-04-05)
(445, Kim Lull, M, 1955-01-19)
(453, Andrew Rabkin, M, 1957-08-10)
(467, James Klobucher, M, 1952-11-09)
(479, Linda Siperstein, F, 1967-09-21)
(501, David Scott, M, 1947-03-01)
```

(529, Dorothy Sullivan, F, 1950-04-19)
(582, Peter Samuels, M, 1968-02-28)
(586, James Coleman, M, 1966-03-04)
(591, Irene Barletta, F, 1957-01-30)
(604, Albert Wang, M, 1958-12-25)
(641, Thomas Powell, M, 1951-10-31)
(667, Mary Garcia, F, 1963-01-23)
(690, Kathleen Poitras, F, 1965-09-29)
(703, Jose Martinez, M, 1953-07-22)
(750, Jane Braun, F, 1939-08-09)
(757, Denis Higgins, F, 1968-05-12)
(839, Dean Marshall, M, 1966-05-21)
(856, Samuel Singer, M, 1959-04-07)
(862, John Sheffield, M, 1955-09-25)
(868, Felicia Kuo, F, 1968-07-24)
(879, Kristen Coe, F, 1965-11-11)
(888, Doug Charlton, M, 1966-01-23)
(902, Moira Kelly, F, 1950-08-16)
(913, Ken Martel, M, 1943-04-23)
(921, Charles Crowley, M, 1960-09-11)
(930, Ann Taylor, F, 1962-06-06)
(949, Pamela Savarino, F, 1955-07-28)
(958, Thomas Sisson, M, 1969-10-02)
(992, Joyce Butterfield, F, 1960-04-15)
(1013, Joseph Barker, M, 1969-02-14)
(1021, Paul Sterling, M, 1950-02-27)
(1039, Shih Lin Chao, M, 1969-12-12)
(1062, Barbara Blaikie, F, 1953-11-14)
(1090, Susan Smith, F, 1950-11-30)
(1101, Mark Preston, M, 1966-09-14)
(1142, Alison Clark, F, 1957-05-04)
(1157, Hing Soo, M, 1970-03-07)
(1162, Kevin Goggin, M, 1952-04-18)
(1191, Matthew Bucceri, M, 1944-11-19)
(1250, Emilio Diaz, M, 1936-01-02)
(1293, Mary Anne Shea, F, 1955-03-13)
(1336, Janet Bigelow, F, 1950-07-21)
(1390, Jennifer Litton, F, 1948-04-05)
(1446, Caroline Yeung, F, 1971-06-21)
(1483, John Letiecq, M, 1939-04-27)
(1507, Ruth Wetherby, F, 1959-07-19)
(1570, Anthony Rebeiro, M, 1963-04-12)
(1576, Scott Evans, M, 1960-11-15)
(1596, Catherine Pickett, F, 1959-11-18)
(1607, Mark Morris, M, 1941-01-08)

(1615, Sheila Romero, F, 1972-09-12)
(1643, Elizabeth Lambert, F, 1968-09-12)
(1658, Michael Lynch, M, 1973-01-18)
(1684, Janet Hildebrand, F, 1955-10-31)
(1740, Robert Nielsen, M, 1965-06-19)
(1751, Alex Ahmed, M, 1963-12-12)

サンプルのコードは次のとおりです。

```
#include <stdio.h>
#include <stdlib.h>
/* Set up the SQL Communication Area*/
EXEC SQL INCLUDE SQLCA;
#include "sqldef.h"
/* Set up a struct set up to collect query results */
typedef struct employee_t
   unsigned long emp_id;
   char name[ 41 ];
   char sex[ 2 ];
   char birthdate[ 15 ];
} employee_t;
/* Print out the various fields of the employee struct */
static void print_employee(employee_t emp)
{
 printf("(%li, %s, %s, %s)¥n", emp.emp_id, emp.name, emp.sex, emp.birthdate);
}
/* Declare the Cursor. This must precede all cursor operations in
 the source code
*/
EXEC SQL DECLARE C1 CURSOR FOR
 SELECT emp_id, emp_fname || ' ' || emp_lname, sex, birth_date
 FROM "dba".employee;
/* Fetch a single row and advance the cursor */
static int fetch_row(
   EXEC SQL BEGIN DECLARE SECTION;
   unsigned long *emp_id,
   char *name,
   char *sex,
   char *birthdate
   EXEC SQL END DECLARE SECTION;
   )
{
   EXEC SQL FETCH RELATIVE 1 C1
           INTO :emp_id, :name, :sex, :birthdate;
```

```
  /* SQLCODE is 0 while there are no errors and rows remaining */
 return !SQLCODE;
}
/* A basic SQL error handler */
EXEC SQL WHENEVER SQLERROR
 char buffer[ 200 ];
 printf("SQL error: %s¥n", sqlerror_message(&sqlca, buffer, sizeof(buffer)));
 return(0);
};

/* Now let's do the real work */
int main( int argc, char *argv[] )
{
 employee_t current_employee;
/* Initialize the Sybase client run-time */
if(!db_init(&sqlca))
   {
     printf("Unable to initialize database interface");
     return(-1);
   }
/* Locate the running server engine */
if(!db_find_engine(&sqlca, NULL))
   {
     printf("Database Engine not running");
     db_fini( &sqlca );
     return(-1);
 }
/* Connect to the engine using user/password "DBA"/"SQL" */
EXEC SQL CONNECT "DBA" IDENTIFIED BY "SQL";
 /* Open (initialize) the cursor */
EXEC SQL OPEN C1;
/* Iterate over all the rows, retrieve the data and print them out */
while (fetch_row(current_employee.emp_id,
    current_employee.name,
    current_employee.sex,
    current_employee.birthdate
    ))
   {
     print_employee(current_employee);
   }
/* We're done with the cursor */
EXEC SQL CLOSE C1;
 return 0;
}
```