Embedded SQL 사용하기

소개: CUBRID 에서 Embedded SQL (이하 ESQL) 을 작성하고, 빌드하는 방법에 대하여 소개한다. LINUX 를 기준으로 작성되었으며 UNIX 계열에서는 compile 옵션이 다를 수 있으며, 소스는 수정없이 사용될 수 있다. 또한 CUBRID2008 이전 버전에서 작성된 프로그램은 모두 재 compile 되어야만 한다.

2008-10-20 기술컨설팅팀 남재우, **CUBRID**

적용 대상: CUBRID2008, LINUX, GCC

목 차

ESQL?

Embedded SQLX 는 일반적인 프로그래밍언어(C)를 통하여 편리하게 큐브리드 데이터베이스에 접근할 수 있도록 해준다. ESQL 프로그래머는 C 언어와 같은 일반적인 프로그래밍 언어 내에서 표준 SQL 명령을 사용하여 데이터베이스 내부의 정보를 검색하거나 갱신 또는 추가 할 수 있게 된다. 데이터베이스 내부에 대한 세부적인 지식이 없더라도 저장 시킬 정보를 준비하거나 추출된 정보를 분석하는데 있어 좀더 자유로운 처리가 가능해진다.

ESQL 작성 규칙

확장자

ESQL 프로그램 파일의 확장자는 .ec 로 만들어야 한다.

데이터베이스 연결

ESQL 을 사용하기 위해서는 일단 초기화 함수(uci_startup())를 먼저 호출하여야 하며, 그 다음 데이터베이스와 연결한다. CUBRID 는 동시에 2 개 이상의 데이터베이스에 연결할 수 없으므로 하나의 데이터베이스와 작업을 완료한 후 다른 데이터베이스에 연결하여야 한다.

초기화 및 데이터베이스 연결은 아래와 같이 수행한다. demodb 라는 데이터베이스에 public 유저로 passwd 라는 암호를 가진 경우의 예이다. 암호가 없는 경우 with 이하는 생략가능하며, 암호없이 public(큐브리드 기본 제공 데이터베이스 사용자)으로 로그인하는 경우 identified by 이하 생략 가능하다.

```
uci_startup("프로그램이름");

EXEC SQL CONNECT 'demodb' identified by 'public' with 'passwd';
```

데이터베이스 종료

데이터베이스 연결 종료는 아래와 같이 간단히 수행한다. 앞서도 언급하였듯이 CUBRID 는 한번에 하나의 데이터베이스와 연결이 가능하므로 연결 종료시에는 별도로 데이터베이스 이름을 주지 않는다.

```
EXEC SQL DISCONNECT;
```

호스트 변수

프로그램에서 사용되어 지는 변수중 SQL 문장에 함께 사용되어 지는 경우, 즉 SQL 문장의 한 부분에 값을 넣기 위해 사용되는 변수는 호스트 변수로 선언되어야 하며, 선언은 아래와 같이 한다.

```
EXEC SQL BEGIN DECLARE SECTION

        int     bus_id;

        char    bus_num[10];

EXEC SQL END DECLARE SECTION;
```

프로그램 내에서 호스트변수 사용시 일반 변수와 똑같이 사용한다. 다만 SQL 문 등 ESQL 명령에 사용될 때는 변수명 앞에 : 를 붙여서 사용한다.

```
EXEC SQL select bus_num from bus where bus_id = :bus_id;
```

문자열

고정길이의 문자열(CHAR type)에 대하여 검색 조건으로 사용시 변수의 빈자리는 문자열로 채워주어야 한다. 예를 들면 고정길이 10 을 가지는 필드에 대하여 검색시 실제 데이터는 5 바이트라면 나머지 5 바이트는 스페이스로 채워주어야 한다.

```
고정길이 문자열 뒷 부분에 스페이이스를 채우는 간단 예제

strcpy(name, ¡홍길동¡);

for (i = strlen(name); i < 10; i++) name[i] = ' ';

name[9] = '\0';
```

반면에 가변길이의 문자열(VARCHAR type)에 대하여는 VARCHAR 라는 구조체를 사용하여야 한다. 검색결과를 받을때는 구조체중 length 부분에 최대길이(필드크기, 즉 넘겨받을 문자열의 최대 길이)를 넣어주어야 하고, 검색에 사용될 경우에는 실제 길이(strlen) 를 넣어주어야 한다.

가변길이의 문자열에 대하여 검색결과를 받는 경우

```
VARCHAR    addr[50];

i

addr.length = sizeof(addr);

EXEC SQL fetch c1 into :addr;
```

가변길이의 문자열에 대하여 검색에 사용하는 경우

```
VARCHAR    addr[50];

i

strcpy(addr.array, '서울시 강남구');

addr.length = strlen('서울시 강남구');

EXEC SQL select ¡  from .. where address = :addr;
```

numeric

numeric 필드를 다루기 위해서는 프로그램내에서는 문자열로 처리하고, 데이터베이스에 입력시나 검색시에는 DB_VALUE 라는 호스트 변수를 사용해야 한다.

입력시에는 문자열 변수에 numeric 값을 넣은 후, numeric 필드의 자리수를 db_value_domain_init() 라는 함수를 이용하여 설정후, db_value_put() 을 이용하여 DB_VALUE 에 값을 넣어주고 insert/update 질의를 수행하게 된다.

반대로 검색시에는 fetch 시 받는 호스트 변수를 DB_VALUE 로 선언된 변수를 이용하고 db_value_get() 을 이용하여 값을 문자열 변수로 받아낸다.

numeric(15,2) 로 설정된 필드에 30.1 을 입력

```
DB_VALUE point;

char       numeric_buf[38];

i

strcpy(numeric_buf, "30.1");

/* numeric(15,2) 로 선언된 경우, DB_VALUE 변수의 numeric 필드의 크기를 설정한다 */

db_value_domain_init(&point, DB_TYPE_NUMERIC, 15, 2);

/* 문자열 변수에 들어가 있는 값을 DB_VALUE 변수에 넣어준다 */

db_value_put(&point, DB_TYPE_C_VARCHAR, (void *)numeric_buf, strlen(numeric_buf));

EXEC SQL insert into ¡  values(¡  :point);
```

numeric 필드에서 값을가져와 numeric_buf 에 저장

```
DB_VALUE point;

char       numeric_buf[38];

i
```

```
EXEC SQL fetch c1 into   :point;



/* numeric 값을 꺼내기 위해 db_value_get() 을 사용한다 */

/* db_value_get(DB_VALUE *value, DB_TYPE db_type, void *buffer, int buffer_length, int *data_length, int *output_length) */

/* 이때 data_length 에는 실제 넘겨진 값의 길이를 넣어주고, 만약 실제 data 가 buffer 의 길이보다 큰 경우
buffer_Length 만큼 잘려져서 들어가고 원래 data 의 길이는 output_Length 에 넣어준다. 돌려지는 data 가 잘리지 않은
경우에는 output_length 의 값은 0 이다 */

db_value_get(&point, DB_TYPE_C_VARCHAR, &numeric_buf, sizeof(numeric_buf), &t, &o);
```

SQL 사용

프로그램 내에서 사용되는 SQL 문장 및 관련 명령들은 반드시 EXEC SQL 로 시작되어야 한다.

```
EXEC SQL CONNECT subway

EXEC SQL select bus_num from bus where bus_id = :bus_id;
```

ESQL compile

ESQL 컴파일은 일단 pre-compile 후, ANSI C 컴파일러를 이용한다. pre-compilier 옵션중 unsafe_null 은 null 값이 넘어올 경우 indicator 변수를 사용하지 않아도 되도록 하는 것이다. 코딩은 쉬워지나 null 이 넘어오는 경우 변수에는 garbage 가 있게 되므로 주의할 필요가 있다. 가장 간단한 방법은 not null default ¡¡ 를 최대한 활용하여 데이터에 null 이 들어가지 않도록 하는 것이다. indicator 변수에 대한 설명은 매뉴얼을 참고한다.

```
cubrid_esql -u esql_sample.ec

gcc -c esql_sample.c -I$(CUBRID)/include   # 또는 gcc -c esql_sample.c -I$CUBRID/include

gcc -o esql_sample esql_sample.o -L$(CUBRID)/lib -lcubridesql -lcubridcs   # 또는 gcc -o esql_sample esql_sample.o -
L$CUBRID/lib -lcubridesql -lcubridcs
```

예제

예제는 esql_sample.ec 로 작성하면 되며, demodb 데이터베이스에 대하여 수행하나 테이블은 아래의 user_info 테이블을 생성하여 사용하였다.

```
create class user_info (

                id              int,

                 name           char(10),

                 address        varchar(50),

                 weight         float,

                 point          numeric(15,2)
```

```
)
```

ESQL 에 대한 자세한 설명은 API 매뉴얼의 ESQL 부분을 참조한다.

입력

다음의 예는 running 테이블에 임의의 데이터를 입력하며, 뒤에 나오는 검색예제 등에서 값들이 사용된다.

```
#include <string.h>
#include <stdio.h>

void sql_error()
{
        printf("ERROR in line %ld : %s \n", SQLLINE, SQLERRMC);

        EXEC SQL rollback work;
        EXEC SQL disconnect;

        exit(-1);
}

main()
{
/* numeric 필드를 다루기 위해서는 DB_VALUE 를 사용해야 한다 */
        EXEC SQL BEGIN DECLARE SECTION;
                int     id;
                float   weight;
                char    name[11];
                VARCHAR address[51];
                DB_VALUE point;
        EXEC SQL END DECLARE SECTION;
        char    numeric_buf[38];

        uci_startup("esql_sample");
```

```
	/*ESQLX 에서 에러 발생시 항상 sql_error 함수를 호출 함 */
	EXEC SQL whenever sqlerror call sql_error;

	/* 문자열을 직접사용시 "가 아닌 '를 사용해야 함. */
	EXEC SQL connect 'demodb';

	id = 1;
	weight = 70;
	strcpy(name, "홍길동      ");
	/* varchar 의 경우 변수명.array 에 값을 복사후, 변수명.length 에 실제 값의 길이를 넣어준다. */
	strcpy(address.array, "서울특별시 강남구");
	address.length = strlen(address.array);

	/* numeric 필드에 입력할 값을 문자열 변수에 넣어준다. */
	strcpy(numeric_buf, "30.1");
	/* numeric(15,2) 로 선언된 경우, DB_VALUE 변수의 numeric 필드의 크기를 설정한다 */
	db_value_domain_init(&point, DB_TYPE_NUMERIC, 15, 2);
	/* 문자열 변수에 들어가 있는 값을 DB_VALUE 변수에 넣어준다 */
	db_value_put(&point, DB_TYPE_C_VARCHAR, (void *)numeric_buf, strlen(numeric_buf));
	EXEC SQL insert into user_info(id,weight,name,address,point) values(:id, :weight, :name, :address, :point);

	EXEC SQL commit work;

	EXEC SQL disconnect;

	return 0;
}
```

검색

앞에서 입력한 데이터를 검색하는 예이다. 문자열 처리부분에 주의한다.

```
#include <string.h>
```

```
#include <stdio.h>


void sql_error()
{
        printf("ERROR in line %ld : %s \n", SQLLINE, SQLERRMC);


        EXEC SQL rollback work;
        EXEC SQL disconnect;


        exit(-1);
}


strcpy_padding(char *dest, char *src, int s_size)
{
        int i, len = strlen(src);


        strcpy(dest, src);
        for(i = len; i < s_size-1; i++)
                dest[i] = ' ';
        dest[i] = '\0';
}


main()
{
        EXEC SQL BEGIN DECLARE SECTION;
                int        id;
                float     weight;
                char       name[11];
                VARCHAR addr[51], address[51];
                DB_VALUE point;
        EXEC SQL END DECLARE SECTION;
        char       numeric_buf[38];
```

```
	int	t, o;


	uci_startup("esql_sample");


	/*ESQLX 에서 에러 발생시 항상 sql_error 함수를 호출 함 */
	EXEC SQL whenever sqlerror call sql_error;


	/* 문자열을 직접사용시 "가 아닌 '를 사용해야 함. */
	EXEC SQL connect 'demodb';


	/* 고정길이형 문자열을 검색조건에서 사용시 데이터베이스의 필드수만큼 스페이스로 채워주어야 >한다. */
	strcpy_padding(name, "홍길동", sizeof(name));
	/* 가변형 문자열을 검색조건에서 사용시 변수명.array 에 검색할 값을 넣어주고, 변수명.length 에 실제 길이를 넣어준다. */
	strcpy(addr.array, "서울특별시 강남구");
	addr.length = strlen(addr.array);


/* 검색결과가 더이상 찾아지지 않으면 not_found 로 이동한다. */
	EXEC SQL whenever not found goto not_found;
	while (1) {
		/* 가변형 문자열의 데이터를 받기 위해 그 length 를 최대값으로 한다. */
		address.length = sizeof(address);
		/* 검색결과를 tuple 단위로 한개씩 꺼낸다 */
		EXEC SQL fetch c1 into :id, :weight, :address, :point;


		/* numeric 값을 꺼내기 위해 db_value_get() 을 사용한다 */
		/* db_value_get(DB_VALUE *value, DB_TYPE db_type, void *buffer, int buffer_length, int *data_length, int *output_length) */
		/* 이때 data_length 에는 실제 넘겨진 값의 길이를 넣어주고, 만약 실제 data 가 buffer 의 길이보다 큰 경우 buffer_Length 만큼 잘려져서 들어가고 원래 data 의 길이는 output_Length 에 넣어준다. 돌려지는 data 가 잘리지 않은 경우에는 output_length 의 값은 0 이다 */
		db_value_get(&point, DB_TYPE_C_VARCHAR, &numeric_buf, sizeof(numeric_buf), &t, &o);
```

```
                printf("%s: %d %f %s %s \ n", name, id, weight, address.array, numeric_buf);
        }
not_found:

        EXEC SQL commit work;

        EXEC SQL disconnect;

        return 0;
}
```

삭제

앞에서 입력한 데이터를 삭제하는 예이다. 역시 문자열 처리부분에 주의한다.

```
#include <string.h>
#include <stdio.h>

void sql_error()
{
        printf("ERROR in line %ld : %s \ n", SQLLINE, SQLERRMC);

        EXEC SQL rollback work;
        EXEC SQL disconnect;

        exit(-1);
}

main()
{
        EXEC SQL BEGIN DECLARE SECTION;
                int     id;
        EXEC SQL END DECLARE SECTION;
```

```
        uci_startup("esqlx_test");



        /*ESQLX 에서 에러 발생시 항상 sql_error 함수를 호출 함 */

        EXEC SQL whenever sqlerror call sql_error;


        /* 문자열을 직접사용시 "가 아닌 '를 사용해야 함. */

        EXEC SQL connect 'subway';


        id = 1;


        EXEC SQL delete from user_info where id = :id;


        EXEC SQL commit work;


        EXEC SQL disconnect;


        return 0;
}
```

(주) 큐브리드 All rights reserved.