

Pro*C から組み込み SQL 移行ガイド (入門編)

Pro*C から組み込み SQL 移行ガイド (入門編)

日本アイ・ビー・エム株式会社
ソフトウェア事業部
ソフトウェア・テクニカルサポート DM 技術部
開発技術支援グループ

更新履歴

2002/08/22 初版

目次

1	はじめに	4
2	基本事項	5
2-1	SQLCA 構造のインクルード	5
2-2	ホスト変数の宣言	5
2-3	データタイプのマッピング	6
2-4	接続	6
2-5	切断	7
2-6	WHENEVER ステートメントを用いたエラー処理	7
2-7	カーソル処理	7
2-8	標識変数	8
2-9	エラーメッセージの取得	8
2-10	サンプル・プログラム	9
3	ホスト変数の扱い方	13
3-1	ホスト変数が構造体の場合	13
3-2	サンプル・プログラム	13
4	動的 SQL (EXECUTE IMMEDIATE を使った方法)	16
4-1	EXECUTE IMMEDIATE を使った SQL の実行	16
4-2	サンプル・プログラム	16
5	動的 SQL その2 (PREPARE を使った方法)	16
5-1	PREPARE を使った SQL の実行	16
5-2	サンプル・プログラム	16
6	ストアド・プロシージャの呼び出し	16
6-1	ストアド・プロシージャの呼び出し	16
6-2	サンプル・プログラム	16

1 はじめに

本ガイドでは、Oracle の Pro*C の経験者を対象に DB2 での組み込み SQL を紹介します。Pro*C でのコーディング例と、DB2 でのコーディング例を掲載し、ステートメントを比較できるようにしました。

Pro*C で作成した埋込み SQL を DB2 の組み込み SQL に移行する場合の参考にもなります。また、本ガイドは、DB2 で組み込み SQL を始めたい方のガイドとして利用することもできます。

本ガイドのサンプル・プログラムは MEMBER 表を使用しています。MEMBER 表の構造は次の通りです。

```
CREATE TABLE MEMBER  
(  
    ID INT,  
    NAME CHAR(20)  
)
```

組み込み SQL のコンパイル方法については、「組み込み SQL の始め方 (C 言語編)」を参考にしてください。

なお、第 2 章からは、Oracle および DB2 は次の意味で使っている場合があります。

- ・ Oracle = Pro*C で作成した埋込み SQL
- ・ DB2 = DB2 で C 言語をつかった組み込み SQL

記載の会社名と製品名はそれぞれ各社の登録商標または商標です。

2 基本事項

この章では、テーブルをカーソルを使って照会している簡単なサンプル・プログラムをもとに、Oracle と DB2 を比較しながら、データベースの接続から切断までの基本的な手続きを見てきます。

サンプル・プログラム全体はこの章の最後に掲載しています。

2-1 SQLCA 構造のインクルード

Oracle、DB2 とともに SQLCA 構造が必要になります。名称は同じですが、構造が違う事に注意してください。

インクルードは次のように行います。

2-1-1 Oracle の場合

```
EXEC SQL INCLUDE SQLCA.H;
```

2-1-2 DB2 の場合

```
EXEC SQL INCLUDE SQLCA; .H が必要ない
```

2-2 ホスト変数の宣言

組み込み SQL 文が参照する変数の宣言は、EXEC SQL BEGIN DECLARE SECTION ~ EXEC SQL END DECLARE SECTION の間に定義します。

Oracle、DB2 とともに、同じステートメントを使っています。

Oracle の場合、proc オプションによってはこのステートメントは省略可能ですが、DB2 の場合は省略する事が出来ません。

2-2-1 Oracle の場合

```
EXEC SQL BEGIN DECLARE SECTION;  
  VARCHAR user[80];          /* ユーザー */  
  VARCHAR password[20];     /* パスワード */  
  VARCHAR dbname[8];       /* データベース名 */  
  int      id;              /* ID */  
  short id_ind;            /* ID インジケータ変数 */  
EXEC SQL END DECLARE SECTION;
```

2-2-2 DB2 の場合

```
EXEC SQL BEGIN DECLARE SECTION;  
  char user[9];             /* ユーザー */  
  char password[9];        /* パスワード */  
  char dbname[9];          /* データベース名 */
```

```
long id; /* ID */
short id_ind; /* ID 標識変数 */
EXEC SQL END DECLARE SECTION;
```

2-3 データタイプのマッピング

Oracle でよく使われている VARHCAR 型について考えてみます。

Oracle 上で VARCHAR 型をプリコンパイルすると、

VARHCAR abc[n];

は

```
struct {
    unsigned short len;
    unsigned char arr[n];
} abc;
```

と展開されます。

DB2 への移行を考えた場合、この struct 型を移行すればよいのですが、文字列として扱い難いので、単純に

char abc[n+1];

に置き換えます。

2-3-1 Oracle の場合

```
VARCHAR user[8];
```

2-3-2 DB2 の場合

```
char user[9];
```

その他のデータタイプについては「IBM DB2 ユニバーサル・データベース移植ガイド Oracle から DB2 V7.1」等を参考にしてください。

2-4 接続

接続は、Oracle、DB2 とともにデータベース名、ユーザー、パスワードを指定する事は同じですが、ステートメントが異なります。

2-4-1 Oracle の場合

```
EXEC SQL CONNECT :user IDENTIFIED BY :password USING :dbname;
```

2-4-2 DB2 の場合

```
EXEC SQL CONNECT TO :dbname USER :user USING :password;
```

DB2 では、データベース名、ユーザー、パスワードをハードコードする事も可能です。

```
EXEC SQL CONNECT TO sample USER db2dev USING ibmdb2;
```

2-5 切断

切断も、ステートメントが異なります。Oracle は、トランザクション管理と同時に行いますが、DB2 は、切断処理単独のステートメントとなります。

2-5-1 Oracle の場合

```
EXEC SQL COMMIT WORK RELEASE;  
または  
EXEC SQL COMMIT ROLLBACK RELEASE;
```

2-5-2 DB2 の場合

```
EXEC SQL CONNECT RESET;
```

2-6 WHENEVER ステートメントを用いたエラー処理

Oracle では、WHENEVER のアクションに関数を指定しますが、DB2 ではラベルを指定します。

2-6-1 Oracle の場合

```
EXEC SQL WHENEVER SQLERROR DO SqlError();
```

2-6-2 DB2 の場合

```
EXEC SQL WHENEVER SQLERROR GOTO SqlError;
```

なお、DB2 は WHENEVER ステートメントには以下の 3 つの基本形式があります。

```
EXEC SQL WHENEVER SQLERROR action  
EXEC SQL WHENEVER SQLWARNING action  
EXEC SQL WHENEVER NOT FOUND action
```

action には、CONTINUE または、GO TO label を設定することができます。

2-7 カーソル処理

ブレイク処理が異なります。

NOT FOUND 時の sqlca.sqlcode が異なっている事に注意してください。

2-7-1 Oracle の場合

```
EXEC SQL WHENEVER NOT FOUND DO break;
```

これは、if (sqlca.sqlcode == 1403) break; として展開されています。

2-7-2 DB2 の場合

```
if(SQLCODE == 100){  
    break;  
}
```

SQLCODE は、sqlca.sqlcode に展開されます。

2-8 標識変数

Oracle ではインジケータ変数と言っています。
NULL 値を受け取ることができるホスト変数です。
ステートメントはどちらも同じです。

2-8-1 Oracle の場合

```
EXEC SQL FETCH memcur INTO :id INDICATOR :id_ind, :name  
INDICATOR :name_ind;  
または  
EXEC SQL FETCH memcur INTO :id:id_ind, :name:name_ind;
```

2-8-2 DB2 の場合

```
EXEC SQL FETCH memcur INTO :id INDICATOR :id_ind, :name  
INDICATOR :name_ind;  
または  
EXEC SQL FETCH memcur INTO :id:id_ind, :name:name_ind;
```

標識変数が負の場合、NULL 値であることを示します。

2-9 エラーメッセージの取得

Oracle では、sqlglm を使って、エラーメッセージの取得を行います。
DB2 では、sqlaintp を使います

2-9-1 Oracle の場合

```
emsgbuff_len = sizeof(emsg);  
/* Oracle からのエラーメッセージの取得 */  
sqlglm(emsg, &emsgbuff_len, &emsg_len);  
printf("%.*s¥n", emsg_len, emsg);
```

2-9-2 DB2 の場合

```
emsgbuff_len = sizeof(emsg);  
/* DB2 からのエラーメッセージの取得 */
```

```
sqlaintp(msg, msgbuff_len, 0, pSqlca);  
printf("%s¥n", msg);
```

2-10 サンプル・プログラム

この章で使用したサンプル・プログラムです。

2-10-1 Oracle の場合 Smp010RA.pc

```
/*  
Smp010RA.pc  
*/  
  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
#include <sqlcpr.h>  
  
EXEC SQL INCLUDE SQLCA.H;  
  
void SqlError(void);  
  
int main()  
{  
  
    /* ホスト変数の宣言 */  
    EXEC SQL BEGIN DECLARE SECTION;  
        VARCHAR user[80];          /* ユーザー */  
        VARCHAR password[20];     /* パスワード */  
        VARCHAR dbname[8];       /* データベース名 */  
        int id;                   /* ID */  
        short id_ind;            /* ID インジケータ変数 */  
        VARCHAR name[20];        /* 名前 */  
        short name_ind;         /* 名前 インジケータ変数 */  
    EXEC SQL END DECLARE SECTION;  
  
    /* 接続する為の準備 */  
    strcpy((char *)user.arr, "db2dev");  
    user.len = (short)strlen((char *)user.arr);  
    strcpy((char *)password.arr, "ibmdb2");  
    password.len = (short)strlen((char *)password.arr);  
    strcpy((char *)dbname.arr, "OraDB");  
    dbname.len = (short)strlen((char *)dbname.arr);  
  
    EXEC SQL WHENEVER SQLERROR DO SqlError();  
  
    /* 接続 */  
    EXEC SQL CONNECT :user IDENTIFIED BY :password USING :dbname;  
  
    /* カーソル宣言 */  
    EXEC SQL DECLARE memcur CURSOR FOR  
        SELECT id, name FROM member;  
  
    /* オープン */  
    EXEC SQL OPEN memcur;  
  
    EXEC SQL WHENEVER NOT FOUND DO break;  
  
    for(;;){  
        /* FETCH */
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
EXEC SQL FETCH memcur INTO :id INDICATOR :id_ind, :name INDICATOR :name_ind;
if(id_ind == -1){
    printf("ID=(NULL)¥t");
}
else{
    printf("ID=(%d)¥t", id);
}
if(name_ind == -1){
    printf("MEMBER=(NULL)¥n");
}
else{
    printf("MEMBER=(%. *s)¥n", name.len, name.arr);
}
}

/* カーソルのクローズ */
EXEC SQL CLOSE memcur;

/* 切断 */
EXEC SQL COMMIT WORK RELEASE;

printf ("正常終了¥n");

exit(EXIT_SUCCESS);
}

void SqlError()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;

    char emsg[128];      /*エラーメッセージ */
    int emsgbuff_len;   /*errmsg のサイズ*/
    int emsg_len;       /*取得したエラーメッセージのサイズ*/

    emsgbuff_len = sizeof(emsg);
    /* Oracle からのエラーメッセージの取得 */
    sqlgln(emsg, &emsgbuff_len, &emsg_len);
    printf("%. *s¥n", emsg_len, emsg);

    /* SQLCA からのエラーコード取得 */
    printf("SQLCODE(%d)¥n", sqlca.sqlcode);

    /* 切断 */
    EXEC SQL ROLLBACK WORK RELEASE;

    exit(EXIT_FAILURE);
}
}
```

2 - 1 0 - 2 DB2 の場合 Smp01DB2.sqc

```
/*****
Smp01DB2.spc
****/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sql.h>

EXEC SQL INCLUDE SQLCA; SQLCA のインクルード

int SqlError(struct sqlca*);
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
int main()
{
    /* ホスト変数の宣言 */
    EXEC SQL BEGIN DECLARE SECTION;
        char user[9];          /* ユーザー */
        char password[9];     /* パスワード */
        char dbname[9];       /* データベース名 */
        long id;              /* ID */
        short id_ind;         /* ID 標識変数 */ 標識変数用 short 型で宣言する
        char name[21];        /* 名前 */  member テーブルの name 列は char(20)なので・
        short name_ind;      /* 名前 標識変数 */
    EXEC SQL END DECLARE SECTION;

    /* 接続する為の準備 */
    strcpy(user, "db2dev");
    strcpy(password, "ibmdb2");
    strcpy(dbname, "sample");

    EXEC SQL WHENEVER SQLERROR GOTO SqlError;  Error の時は、SqlError ラベルに処理が移る

    /* 接続 */
    EXEC SQL CONNECT TO :dbname USER :user USING :password;  DB に接続する

    /* カーソル宣言 */
    EXEC SQL DECLARE memcur CURSOR FOR SELECT id, name FROM member;  カーソルの宣言

    /* オープン */
    EXEC SQL OPEN memcur;  カーソルのオープン

    for(;;){
        /* FETCH */
        EXEC SQL FETCH memcur INTO :id INDICATOR :id_ind, :name INDICATOR :name_ind;
            値と標識変数の取り出し

        if(SQLCODE == 100){  NOT FOUND の時
            break;
        }

        if(id_ind < 0){  標識変数が負の値なので、この時は NULL 値
            printf("ID=(NULL)%t");
        }else{
            printf("ID=(%d)%t", id);
        }
        if(name_ind < 0){  標識変数が負の値なので、この時は NULL 値
            printf("MEMBER=(NULL)%n");
        }else{
            printf("MEMBER=(%s)%n", name);
        }
    }

    /* カーソルのクローズ */
    EXEC SQL CLOSE memcur;  カーソルのクローズ

    /* 切断 */
    EXEC SQL COMMIT;
    EXEC SQL CONNECT RESET;  DB との切断

    printf ("正常終了%n");

    exit(EXIT_SUCCESS);
}
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
SqlError: Error が発生した時はここに処理が移る
          SqlError(&sqlca); エラー処理関数
}

int SqlError(struct sqlca *pSqlca)
{
    EXEC SQL WHENEVER SQLERROR CONTINUE; SQLERROR がある場合でも処理を続ける

    char errmsg[128]; /*エラーメッセージ*/
    short errmsgbuff_len; /*errmsgのサイズ*/

    errmsgbuff_len = sizeof(errmsg);
    /* SQLCA からのエラーメッセージの取得 */
    sqlaintp(errmsg, errmsgbuff_len, 0, pSqlca); sqlaintp を使って SQLCA からエラーコードの取得
    printf("%s\n", errmsg);

    /* SQLCODE の表示 */
    printf("SQLCODE(%d)\n", SQLCODE); SQLCODE の表示

    /* 切断 */
    EXEC SQL ROLLBACK;
    EXEC SQL CONNECT RESET; DB との切断

    exit(EXIT_FAILURE);
}
```

3 ホスト変数の扱い方

この章では、ホスト変数が構造体の場合の扱い方を見ていきます。
サンプル・プログラム全体はこの章の最後に掲載しています。

3-1 ホスト変数が構造体の場合

カーソルで FETCH した値をホスト変数に入れる場合の例です。ホスト変数が構造体になっています。

Oracle の場合と、DB2 の場合で、指標変数の宣言方法の違いに注意してください。

3-1-1 Oracle の場合

```
//宣言の部分
struct member{
    int id; /* ID */
    VARCHAR name[20]; /* 名前 */
} member;
struct member_ind{
    short id_ind; /* ID インジケータ変数 */
    short name_ind; /* 名前 インジケータ変数 */
} member_ind;

//FETCH の部分
EXEC SQL FETCH memcur INTO :member INDICATOR :member_ind;
```

3-1-2 DB2 の場合

```
//宣言の部分
struct member{
    long id; /* ID */
    char name[21]; /* 名前 */
}member;
short member_ind[2]; /* 標識変数 */

//FETCH の部分
EXEC SQL FETCH memcur INTO :member INDICATOR :member_ind;
```

3-2 サンプル・プログラム

この章で使用したサンプル・プログラムです。

3-2-1 Oracle の場合 Smp020RA.pc

```
/*
Smp020RA.pc
*/
```

Pro*C から組み込み SQL 移行ガイド (入門編)

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

EXEC SQL INCLUDE SQLCA.H;

void SqlError(void);

int main()
{
    /* ホスト変数の宣言 */
    EXEC SQL BEGIN DECLARE SECTION;
        VARCHAR user[80];          /* ユーザー */
        VARCHAR password[20];     /* パスワード */
        VARCHAR dbname[8];       /* データベース名 */
        struct member{
            int id;               /* ID */
            VARCHAR name[20];    /* 名前 */
        } member;
        struct member_ind{
            short id_ind;        /* ID インジケータ変数 */
            short name_ind;     /* 名前 インジケータ変数 */
        } member_ind;
    EXEC SQL END DECLARE SECTION;

    /* 接続する為の準備 */
    strcpy((char *)user.arr, "db2dev");
    user.len = (short)strlen((char *)user.arr);
    strcpy((char *)password.arr, "ibmdb2");
    password.len = (short)strlen((char *)password.arr);
    strcpy((char *)dbname.arr, "OraDB");
    dbname.len = (short)strlen((char *)dbname.arr);

    EXEC SQL WHENEVER SQLERROR DO SqlError();

    /* 接続 */
    EXEC SQL CONNECT :user IDENTIFIED BY :password USING :dbname;

    /* カーソル宣言 */
    EXEC SQL DECLARE memcur CURSOR FOR
        SELECT id, name FROM member;

    /* オープン */
    EXEC SQL OPEN memcur;

    EXEC SQL WHENEVER NOT FOUND DO break;

    for(;;){
        /* FETCH */
        EXEC SQL FETCH memcur INTO :member INDICATOR :member_ind;
        if(member_ind.id_ind == -1){
            printf("ID=(NULL)≠t");
        }else{
            printf("ID=(%d)≠t", member.id);
        }
        if(member_ind.name_ind == -1){
            printf("MEMBER=(NULL)≠n");
        }else{
            printf("MEMBER=(%. *s)≠n", member.name.len, member.name.arr);
        }
    }
}
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
    }

    /* カーソルのクローズ */
    EXEC SQL CLOSE memcur;

    /* 切断 */
    EXEC SQL COMMIT WORK RELEASE;

    printf ("正常終了\n");

    exit(EXIT_SUCCESS);
}

void SqlError()
{
    略・・・ Smp010RA.pc を参照
}
```

3 - 2 - 2 DB2 の場合 Smp02DB2.sqc

```
/*
Smp02DB2.sqc
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sql.h>

EXEC SQL INCLUDE SQLCA;

int SqlError(struct sqlca*);

int main()
{
    /* ホスト変数の宣言 */
    EXEC SQL BEGIN DECLARE SECTION;
        char user[9];          /* ユーザー */
        char password[9];     /* パスワード */
        char dbname[9];       /* データベース名 */
        struct member{
            long id;          /* ID */ member テーブルでの integer 型は long 型へ
            char name[21];    /* 名前 */
        }member; member テーブルと同じ構造にする
        short member_ind[2]; /* 標識変数 */ 標識変数を配列で宣言する
    EXEC SQL END DECLARE SECTION;

    /* 接続する為の準備 */
    strcpy(user, "db2dev");
    strcpy(password, "ibmdb2");
    strcpy(dbname, "sample");

    EXEC SQL WHENEVER SQLERROR GOTO SqlError;

    /* 接続 */
    EXEC SQL CONNECT TO :dbname USER :user USING :password;

    /* カーソル宣言 */
    EXEC SQL DECLARE memcur CURSOR FOR
        SELECT id, name FROM member;
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
/* オープン */
EXEC SQL OPEN memcur;

for(;;){
    /* FETCH */
    EXEC SQL FETCH memcur INTO :member INDICATOR :member_ind;
    変数を member 構造体に、標識変数を member_ind 配列に値を取り出す

    if(SQLCODE == 100){
        break;
    }

    if(member_ind[0] < 0){ 配列の 0 番目は id の標識変数
        printf("ID=(NULL)¥t");
    }else{
        printf("ID=(¥d)¥t", member.id);
    }
    if(member_ind[1] < 0){ 配列の 1 番目は member の標識変数
        printf("MEMBER=(NULL)¥n");
    }else{
        printf("MEMBER=(¥s)¥n", member.name);
    }
}

/* カーソルのクローズ */
EXEC SQL CLOSE memcur;

/* 切断 */
EXEC SQL COMMIT;
EXEC SQL CONNECT RESET;

printf ("正常終了¥n");

exit(EXIT_SUCCESS);

SqlError:
    SqlError(&sqlca);

}

int SqlError(struct sqlca *pSqlca)
{
    略・・・ Smp01DB2.sqc を参照
}

```

4 動的 SQL (EXECUTE IMMEDIATE を使った方法)

この章では動的 SQL の実行方法の一つ、EXECUTE IMMEDIATE を使った方法を見ていきます。

サンプル・プログラム全体はこの章の最後に掲載しています。

4-1 EXECUTE IMMEDIATE を使った SQL の実行

EXECUTE IMMEDIATE を使った SQL の実行方法です。
ホスト変数に SQL 文を代入して使用します。

Oracle、DB2 とともに実行方法は同じです。

4-1-1 Oracle の場合

```
strcpy(stmt, "INSERT INTO member VALUES(22, 'SASAKI');  
EXEC SQL EXECUTE IMMEDIATE :stmt;
```

4-1-2 DB2 の場合

```
strcpy(stmt, "INSERT INTO member VALUES(22, 'sasaki');  
EXEC SQL EXECUTE IMMEDIATE :stmt;
```

4-2 サンプル・プログラム

この章で使用したサンプル・プログラムです。

4-2-1 Oracle の場合 Smp030RA.pc

```
/*  
Smp030RA.pc  
*/  
  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
#include <sqlcpr.h>  
  
EXEC SQL INCLUDE SQLCA.H;  
  
void SqlError(void);  
  
int main()  
{  
  
    /* ホスト変数の宣言 */  
    EXEC SQL BEGIN DECLARE SECTION;  
        VARCHAR user[80];          /* ユーザー */  
        VARCHAR password[20];     /* パスワード */  
        VARCHAR dbname[8];        /* データベース名 */
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
        char stmt[256];          /* SQL文 */
EXEC SQL END DECLARE SECTION;

/* 接続する為の準備 */
strcpy((char *)user.arr, "db2dev");
user.len = (short)strlen((char *)user.arr);
strcpy((char *)password.arr, "ibmdb2");
password.len = (short)strlen((char *)password.arr);
strcpy((char *)dbname.arr, "OraDB");
dbname.len = (short)strlen((char *)dbname.arr);

EXEC SQL WHENEVER SQLERROR DO SqlError();

/* 接続 */
EXEC SQL CONNECT :user IDENTIFIED BY :password USING :dbname;

strcpy(stmt, "INSERT INTO member VALUES(22, 'SASAKI')");

EXEC SQL EXECUTE IMMEDIATE :stmt;

strcpy(stmt, "ROLLBACK");

EXEC SQL EXECUTE IMMEDIATE :stmt;

/* 切断 */
EXEC SQL COMMIT WORK RELEASE;

printf ("正常終了\n");

exit(EXIT_SUCCESS);
}

void SqlError()
{
    略・・・ Smp01ORA.pc を参照
}

```

4 - 2 - 2 DB2 の場合

```
/*
Smp03DB2.sqc
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sql.h>

EXEC SQL INCLUDE SQLCA;

int SqlError(struct sqlca*);

int main()
{
    /* ホスト変数の宣言 */
EXEC SQL BEGIN DECLARE SECTION;
        char user[9];          /* ユーザー */
        char password[9];     /* パスワード */
        char dbname[9];       /* データベース名 */
        char stmt[256];

```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
EXEC SQL END DECLARE SECTION;

/* 接続する為の準備 */
strcpy(user, "db2dev");
strcpy(password, "ibmdb2");
strcpy(dbname, "sample");

EXEC SQL WHENEVER SQLERROR GOTO SqlError;

/* 接続 */
EXEC SQL CONNECT TO :dbname USER :user USING :password;

/* SQL 文の実行 */
strcpy(stmt, "INSERT INTO member VALUES(22, 'sasaki')");
EXEC SQL EXECUTE IMMEDIATE :stmt; /* ホスト変数に SQL 文をコピーする
実行 */

/* SQL 文の実行 */
strcpy(stmt, "ROLLBACK");
EXEC SQL EXECUTE IMMEDIATE :stmt; /* Rollback もこのように実行できる */

/* 切断 */
EXEC SQL COMMIT;
EXEC SQL CONNECT RESET;

printf ("正常終了\n");

exit(EXIT_SUCCESS);

SqlError:
    SqlError(&sqlca);
}

int SqlError(struct sqlca *pSqlca)
{
    略・・・ Smp01DB2.sqc を参照
}
```

5 動的 SQL その 2 (PREPARE を使った方法)

PREPARE を使った SQL の実行方法の比較です。

PREPARE は、SQL ステートメントの動的な実行を準備するために、アプリケーション・プログラムによって使用されます。

サンプル・プログラム全体はこの章の最後に掲載しています。

PREPARE を使った CURSOR 宣言も参考にしてください。

5-1 PREPARE を使った SQL の実行

Oracle はダミーのホスト変数 (プレースホルダ) として表されます。

DB2 ではパラメーター・マーカーは疑問符 (?) で表現します。

5-1-1 Oracle の場合

```
/* SQL 文の準備 */
strcpy(stmt, "INSERT INTO member VALUES(:host1, :host2)");
EXEC SQL PREPARE insertstmt FROM :stmt;

/* SQL 文の実行 */
id = 25;
strcpy(name, "matsubara");
EXEC SQL EXECUTE insertstmt USING :id, :name;
```

5-1-2 DB2 の場合

```
/* SQL 文の準備 */
strcpy(stmt, "INSERT INTO member VALUES(?, ?)");
EXEC SQL PREPARE insertstmt FROM :stmt;

/* SQL 文の実行 */
id = 26;
strcpy(name, "tashiro");
EXEC SQL EXECUTE insertstmt USING :id, :name;
```

5-2 サンプル・プログラム

この章で使用したサンプル・プログラムです。

5-2-1 Oracle の場合

```
/*****
Smp04ORA.pc
****/

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

Pro*C から組み込み SQL 移行ガイド (入門編)

```
#include <sqlcpr.h>

EXEC SQL INCLUDE SQLCA.H;

void SqlError(void);

int main()
{
    /* ホスト変数の宣言 */
    EXEC SQL BEGIN DECLARE SECTION;
        char user[81];           /* ユーザー */
        char password[21];      /* パスワード */
        char dbname[9];         /* データベース名 */
        int id;                 /* ID */
        char name[21];          /* 名前 */
        char stmt[256];         /* SQL文 */
        struct member{
            int id;             /* ID */
            char name[21];      /* 名前 */
        } member;
        struct member_ind{
            short id_ind;       /* ID インジケータ変数 */
            short name_ind;     /* 名前 インジケータ変数 */
        } member_ind;
    EXEC SQL END DECLARE SECTION;

    /* 接続する為の準備 */
    strcpy(user, "db2dev");
    strcpy(password, "ibmdb2");
    strcpy(dbname, "OraDB");

    EXEC SQL WHENEVER SQLERROR DO SqlError();

    /* 接続 */
    EXEC SQL CONNECT :user IDENTIFIED BY :password USING :dbname;

    /* SQL文の準備 */
    strcpy(stmt, "INSERT INTO member VALUES(:host1, :host2)");
    EXEC SQL PREPARE insertstmt FROM :stmt;

    /* SQL文の実行 */
    id = 25;
    strcpy(name, "matsubara");
    EXEC SQL EXECUTE insertstmt USING :id, :name;

    /* カーソル処理 */
    strcpy(stmt, "SELECT id, name FROM member WHERE id > :host1");
    EXEC SQL PREPARE curstmt FROM :stmt;

    /* カーソル宣言 */
    EXEC SQL DECLARE memcur CURSOR FOR curstmt;

    /* 10 より大きい id を表示 */
    id = 10;

    /* オープン */
    EXEC SQL OPEN memcur USING :id;

    EXEC SQL WHENEVER NOT FOUND DO break;

    for(;;){
        /* FETCH */
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
        EXEC SQL FETCH memcur INTO :member INDICATOR :member_ind;
        if(member_ind.id_ind == -1){
            printf("ID=(NULL)¥t");
        }else{
            printf("ID=(%d)¥t", member.id);
        }
        if(member_ind.name_ind == -1){
            printf("MEMBER=(NULL)¥n");
        }else{
            printf("MEMBER=(%s)¥n", member.name);
        }
    }

    /* カーソルのクローズ */
    EXEC SQL CLOSE memcur;

    EXEC SQL ROLLBACK WORK;

    /* 切断 */
    EXEC SQL COMMIT WORK RELEASE;

    printf ("正常終了¥n");

    exit(EXIT_SUCCESS);
}

void SqlError()
{
    略・・・ Smp010RA.pc を参照
}

```

5 - 2 - 2 DB2 の場合

```
/*
Smp04DB2.sqc
****/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sql.h>

EXEC SQL INCLUDE SQLCA;

int SqlError(struct sqlca*);

int main()
{
    /* ホスト変数の宣言 */
    EXEC SQL BEGIN DECLARE SECTION;
        char user[9];           /* ユーザー */
        char password[9];      /* パスワード */
        char dbname[9];        /* データベース名 */
        struct member{
            long id;           /* ID */
            char name[21];     /* 名前 */
        }member;
        short member_ind[2];   /* 標識変数 */
        char stmt[256];        /* SQL 文 */
        long id;               /* ID */

```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
        char name[21];                /* 名前 */
EXEC SQL END DECLARE SECTION;

/* 接続する為の準備 */
strcpy(user, "db2dev");
strcpy(password, "ibmdb2");
strcpy(dbname, "sample");

EXEC SQL WHENEVER SQLERROR GOTO SqlError;

/* 接続 */
EXEC SQL CONNECT TO :dbname USER :user USING :password;

/* SQL 文の準備 */
strcpy(stmt, "INSERT INTO member VALUES(?, ?)");
EXEC SQL PREPARE insertstmt FROM :stmt;

/* SQL 文の実行 */
id = 26;
strcpy(name, "tashiro");
EXEC SQL EXECUTE insertstmt USING :id, :name;

パラメーター・マーカを使ったカーソルの宣言方法
/* カーソルの準備 */
strcpy(stmt, "SELECT id, name FROM member WHERE id > ?");
EXEC SQL PREPARE curstmt FROM :stmt;

/* カーソル宣言 */
EXEC SQL DECLARE memcur CURSOR FOR curstmt;

/* 10 より大きい id を表示 */
id = 10;

/* オープン */
EXEC SQL OPEN memcur USING :id;

for(;;){
    EXEC SQL FETCH memcur INTO :member INDICATOR :member_ind;
    if(SQLCODE = 100){
        break;
    }
    if(member_ind[0] = -1){
        printf("ID=(NULL)%t");
    }else{
        printf("ID=(%d)%t", member.id);
    }
    if(member_ind[1] = -1){
        printf("MEMBER=(NULL)%n");
    }else{
        printf("MEMBER=(%s)%n", member.name);
    }
}
}
```

パラメーター・マーカを?で示す。
この例の場合は2つ
PREPAREでSQL文を準備する。

USING句で2つ変数を設定する。
(PREPAREで設定したパラメーター・マーカの数分)
そしてEXECUTEでSQL分を実行する。
ここで実行したSQL文は
INSERT INTO member VALUES(26, 'tashiro');

Pro*C から組み込み SQL 移行ガイド (入門編)

```
/* カーソルのクローズ */
EXEC SQL CLOSE memcur;

EXEC SQL ROLLBACK;

/* 切断 */
EXEC SQL COMMIT;
EXEC SQL CONNECT RESET;

printf ("正常終了\n");

exit(EXIT_SUCCESS);

SqlError:
    SqlError(&sqlca);
}

int SqlError(struct sqlca *pSqlca)
{
    略・・・ Smp01DB2.sqc を参照
}
```

6 ストアド・プロシージャの呼び出し

ストアド・プロシージャの呼び出し方法です。
Oracle、DB2 とともに CALL 文を使用します。
サンプル・プログラム全体はこの章の最後に掲載しています。

6-1 ストアド・プロシージャの呼び出し

6-1-1 Oracle の場合

```
EXEC SQL CALL procIDxX(:id, :msg);
```

6-1-2 DB2 の場合

```
EXEC SQL CALL procIDxX(:id, :msg);
```

6-2 サンプル・プログラム

この章で使用したサンプル・プログラムです。

6-2-1 Oracle の場合

```
/*  
Smp050RA.pc  
*/  
  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
#include <sqlcpr.h>  
  
typedef char msgtype[128];  
  
EXEC SQL INCLUDE SQLCA.H;  
  
void SqlError(void);  
  
int main()  
{  
  
    /* ホスト変数の宣言 */  
    EXEC SQL BEGIN DECLARE SECTION;  
        char user[81];          /* ユーザー */  
        char password[21];     /* パスワード */  
        char dbname[9];       /* データベース名 */  
        int id;               /* ID */  
        EXEC SQL TYPE msgtype IS VARCHAR2(128);  
        msgtype msg;         /* SP からのメッセージ */  
    EXEC SQL END DECLARE SECTION;  
  
    /* 接続する為の準備 */  
    strcpy(user, "db2dev");  
    strcpy(password, "ibmdb2");
```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
strcpy(dbname, "OraDB");

EXEC SQL WHENEVER SQLERROR DO SqlError();

/* 接続 */
EXEC SQL CONNECT :user IDENTIFIED BY :password USING :dbname;

id = 2;
EXEC SQL CALL proclDxx(:id, :msg);

printf("%s\n", msg);

/* 切断 */
EXEC SQL COMMIT WORK RELEASE;

printf ("正常終了\n");

exit(EXIT_SUCCESS);

}

void SqlError()
{
    略・・・ Smp010RA.pc を参照
}

```

6 - 2 - 2 DB2 の場合

```
/*****
Smp05DB2.sqc
*****/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sql.h>

EXEC SQL INCLUDE SQLCA;

int SqlError(struct sqlca*);

int main()
{

    /* ホスト変数の宣言 */
    EXEC SQL BEGIN DECLARE SECTION;
        char user[9];           /* ユーザー */
        char password[9];      /* パスワード */
        char dbname[9];        /* データベース名 */
        char msg[128];         /* メッセージ */
    EXEC SQL END DECLARE SECTION;

    /* 接続する為の準備 */
    strcpy(user, "db2dev");
    strcpy(password, "ibmdb2");
    strcpy(dbname, "sample");

    EXEC SQL WHENEVER SQLERROR GOTO SqlError;

    /* 接続 */
    EXEC SQL CONNECT TO :dbname USER :user USING :password;
}

```

Pro*C から組み込み SQL 移行ガイド (入門編)

```
id = 2;

EXEC SQL CALL proclDxX(:id, :msg);   ストアドプロシージャの呼び出し

printf("%s\n", msg);   ストアドプロシージャからのメッセージの確認

/* 切断 */
EXEC SQL COMMIT;
EXEC SQL CONNECT RESET;

printf ("正常終了\n");

exit(EXIT_SUCCESS);

SqlError:
    SqlError(&sqlca);

}

int SqlError(struct sqlca *pSqlca)
{
    略・・・ Smp01DB2.sqc を参照
}

```