```
static char sccsid[]="@(#) di_ora.pc 1.2 98/07/29 17:20:20";
/*****************************************************/
/*                                                   */
/*      Database Interface Program for ORACLE7/8     */
/*                                                   */
/*           coded by A.Kobayashi    93.12.10        */
/*         updated by A.Kobayashi  2000.01.05        */
/*                                                   */
/*****************************************************/
#ifndef SUNOS
#define SUNOS
#endif
#ifndef SUNOS5
#define SUNOS5
#endif

#include    "cpunix.h"
#include    "dip.h"

static int di_sellist_ora();
static int di_longraw_set();

#define Malloc  malloc
#define Free    free

#define PORA(x)

static char *nullstring=" ";

static char *userid;
static char *username;
static char *password;
static char *db_string;
static char *statement;

EXEC SQL INCLUDE sqlca;
EXEC SQL INCLUDE sqlda;
extern void sqlprc();
extern void sqlnul();
extern SQLDA *sqlald();

/*  Declare the bind and select descriptors.    */
static SQLDA *bind_dp   = NULL;
static SQLDA *select_dp = NULL;

static int   precision, scale;
static int   null_ok;
static short *indicator = NULL;
static char  errmsg[512];
static long  gcol_max, gsel_col_max, gbnd_col_max;

/*************************************/
/*                                   */
/*************************************/
static int di_getw(p, wd, lmax)
char *p, *wd;
int  lmax;
{
    char ch;
    int  l = 0, lmax1;

    if ((lmax1 = lmax - 1)<0) return -2;

    while(((ch=*p)=='¥n') || (ch==' ') || (ch=='¥t')) p++;

    while(((ch=*p)!='¥n') && (ch!=' ') && (ch!='¥t') && (ch!='¥0')) {
        if (l<lmax1) {
            *wd++ = ch;
            l++;
        }
        p++;
    }
    *wd = '¥0';
    return(l);
}
```

```
/**********************************/
/*                                */
/**********************************/
static int di_stricmp(s1,s2)
unsigned char *s1,*s2;
{
    unsigned char c1,c2;
    int d=0;

    for (;;) {
        c1 = *s1++;
        c2 = *s2++;
        if (!c1 && !c2) break;
        if (d=toupper(c1)-toupper(c2)) break;
    }

    return d;
}

/**********************************/
/*                                */
/**********************************/
static int di_sqlerrm_ora(pdipCT)
qDipCT *pdipCT;
{
    int  msglen,len,ret=0;

    if ((msglen=sizeof(errmsg)) > 71) {
        sqlglm(errmsg,&msglen,&len);
        errmsg[len] = '¥0';
    }
    else {
        memcpy(errmsg,sqlca.sqlerrm.sqlerrmc,sqlca.sqlerrm.sqlerrml);
        errmsg[sqlca.sqlerrm.sqlerrml] = '¥0';
    }
    pdipCT->sqlcode = sqlca.sqlcode;
    if (sqlca.sqlcode) ret = ERROR_ORACLE;
    return ret;
}

/**********************************/
/*                                */
/**********************************/
int di_init_ora(pdipCT)
qDipCT  *pdipCT;
{
    return di_init2_ora(pdipCT,MAX_COLUMN,MAX_COLUMN);
}

/**********************************/
/*                                */
/**********************************/
int di_init2_ora(pdipCT,sel_col_max,bnd_col_max)
qDipCT  *pdipCT;
long sel_col_max,bnd_col_max;
{
    qParm *p;
    qDist *iitmn;
    int i;

    if (sel_col_max < 1) sel_col_max = 1;
    if (bnd_col_max < 1) bnd_col_max = 1;

    pdipCT->errormsg = errmsg;

    p = pdipCT->pparm;
    p->max_sel_col = gsel_col_max = sel_col_max;
    p->max_bnd_col = gbnd_col_max = bnd_col_max;
    if ((gcol_max=sel_col_max) < bnd_col_max) gcol_max = bnd_col_max;

    if (!(bind_dp = sqlald(gbnd_col_max,MAX_BND_NAME_SIZE,MAX_BND_NAME_SIZE)))
        return(ERROR_MALLOC);
    bind_dp->N = gbnd_col_max;
```

```
    if (!(select_dp = sqlald(gsel_col_max,MAX_SEL_NAME_SIZE,0)))
        return(ERROR_MALLOC);
    select_dp->N = gsel_col_max;

    if (pdipCT->piitmn) memset(pdipCT->piitmn,0,sizeof(qDist)*gsel_col_max);
    if (pdipCT->pidist) memset(pdipCT->pidist,0,sizeof(qDist)*gsel_col_max);
    if (pdipCT->podist) memset(pdipCT->podist,0,sizeof(qDist)*gcol_max);

    if (!(indicator = Malloc(sizeof(short)*gcol_max))) return ERROR_MALLOC;
    memset(indicator,0,sizeof(short)*gcol_max);

    if (iitmn = pdipCT->piitmn) {
        for (i=0;i<gsel_col_max;i++) {
            iitmn[i].attr.dtype = TYPE_ORA_CHAR;
            iitmn[i].cpdat = select_dp->S[i];
            iitmn[i].len = MAX_SEL_NAME_SIZE;
        }
    }

    for (i=0;i<gbnd_col_max;i++) {
        bind_dp->I[i] = &indicator[i];
    }
    for (i=0;i<gsel_col_max;i++) {
        select_dp->I[i] = &indicator[i];
        select_dp->V[i] = NULL;
    }

    return(0);
}

/***********************************/
/*                                 */
/***********************************/
int di_connect_ora(pdipCT)
qDipCT *pdipCT;
{
    qParm *p;
    int ret=0;

    p = pdipCT->pparm;
    username = p->logname;
    password = p->password;
    userid   = p->userid;
    db_string= p->netname;
    if (p->netname && *(p->netname)) {
        if (p->userid) {
            EXEC SQL CONNECT :userid USING :db_string;
        }
        else {
            if (!p->logname || !(p->password)) return (ERROR_LOGIN);
            EXEC SQL CONNECT :username IDENTIFIED BY :password
                     USING :db_string;
        }
    }
    else {
        if (p->userid) {
            EXEC SQL CONNECT :userid;
        }
        else {
            if (!p->logname || !(p->password)) return (ERROR_LOGIN);
            EXEC SQL CONNECT :username IDENTIFIED BY :password;
        }
    }

    return di_sqlerrm_ora(pdipCT);
}

/***********************************/
/*                                 */
/***********************************/
int di_pre_bind_ora(psql,pdipCT)
char *psql;
qDipCT *pdipCT;
{
    int i,ret;
```

```
    bind_dp->N = gbnd_col_max;
    pdipCT->nodist = 0;

    EXEC SQL PREPARE S FROM :psql;

    EXEC SQL DECLARE C CURSOR FOR S;
    if (sqlca.sqlcode) goto SqlErr;

    EXEC SQL DESCRIBE BIND VARIABLES FOR S INTO bind_dp;
    if (sqlca.sqlcode) goto SqlErr;
/*
printf("¥nbind variable = %d¥n",bind_dp->F);
*/
    if (bind_dp->F < 0) {
        PORA(printf("Too many bind variables for desciptor.¥n");)
        bind_dp->N = 0;
        return(ERROR_TOO_MANY);
    }
    bind_dp->N = bind_dp->F;
    pdipCT->nodist = bind_dp->N;
    return 0;
SqlErr:
    bind_dp->N = 0;
    return di_sqlerrm_ora(pdipCT);
}

/************************************/
/*                                  */
/************************************/
int di_pre_ora(psql,pdipCT)
char *psql;
qDipCT *pdipCT;
{
    int ret;
    char wd[10];

    if (!(ret=di_pre_bind_ora(psql,pdipCT))) {
        if (!(ret=di_openc_ora(pdipCT))) {
            di_getw(psql, wd, sizeof(wd));
            if (!di_stricmp(wd,"SELECT")) {
                ret = di_sellist_ora(pdipCT);
            }
            else
                select_dp->F = 0;
        }
    }
    return ret;
}

/************************************/
/*                                  */
/************************************/
int di_openc_ora(pdipCT)
qDipCT *pdipCT;
{
    int i;
    qDist *odist;

    odist = pdipCT->podist;

    for (i=0;i<bind_dp->N;i++) {
/*
printf("di_ora:bind_name[%d]=%.*s,type=%d¥n",i,bind_dp->C[i],bind_dp->S[i],
bind_dp->T[i]);
*/
        bind_dp->L[i] = odist[i].len;
        bind_dp->V[i] = odist[i].cpdat;
        /* Set the indicator variable's value. */
        if (!bind_dp->L[i])
            *bind_dp->I[i] = -1;
        else
            *bind_dp->I[i] = 0;
        bind_dp->T[i] = odist[i].attr.dtype;
/*
```

```
printf("di_ora:L=%d, T=%d¥n",bind_dp->L[i], bind_dp->T[i]);
*/
    }
    EXEC SQL OPEN C USING DESCRIPTOR bind_dp;

    return di_sqlerrm_ora(pdipCT);
}

/**********************************/
/*                                */
/**********************************/
static int di_sellist_ora(pdipCT)
qDipCT *pdipCT;
{
    int i,longraw, extern_char_type;
    qDist *idist, *odist, *iitmn;

    idist = pdipCT->pidist;
    odist = pdipCT->podist;
    iitmn = pdipCT->piitmn;

    select_dp->N = gsel_col_max;

    EXEC SQL DESCRIBE SELECT LIST FOR S INTO select_dp;
    if (sqlca.sqlcode) goto SqlErr;
/*
printf("¥nselect list %d¥n",select_dp->F);
*/
    if (select_dp->F < 0) {
        PORA(printf("Too many select-list items for: %d¥n",-(select_dp->F));)
        return(ERROR_TOO_MANY);
    }
    select_dp->N = select_dp->F;
    pdipCT->nidist = select_dp->N;

/*********************************************************************/
/* CHARのときは、左詰めになり、 残りは、半角スペースになる          */
/* (注)NUMBER属性ときは右詰めになるとマニュアルには書いてあるが、   */
/*     実際には、左詰めになる                                        */
/*********************************************************************/
/*  extern_char_type = TYPE_ORA_CHAR;   */
/*********************************************************************/
/* VARCHAR2のときは、定義属性により右詰めか左詰めになる。           */
/*   NUMBER : 右詰め(後詰め)                                         */
/*   その他 : 左詰め(前詰め)                                         */
/* 残りは、半角スペースになる                                        */
/*********************************************************************/
/*  extern_char_type = TYPE_ORA_VARCHAR2;   */
/*********************************************************************/
/* STRINGのときは、左詰めでデータの後ろかL[i]バイト目に¥0が入る、   */
/* したがって、L[i]を定義長にしておくと、データ長がL[i]のときには、*/
/* データの最後が削られてしまう。                                    */
/* さらに、この場合、標識変数(I[i])が正の値になる(NULLのときは-1)   */
/*********************************************************************/
    extern_char_type = TYPE_ORA_STRING;

    longraw = -1;
    for (i=0;i<select_dp->N;i++) {
        sqlnul(&(select_dp->T[i]),&(select_dp->T[i]),&null_ok);
        idist[i].attr.pre   = 0;
        idist[i].attr.scale = 0;
        idist[i].len = select_dp->L[i];
        odist[i].attr.flag = select_dp->T[i];
        iitmn[i].len = select_dp->C[i];
PORA(printf ("i=%d type=%d L=%d",i,select_dp->T[i],select_dp->L[i]);)
        switch (select_dp->T[i]) {
            case TYPE_ORA_VARCHAR2:
                select_dp->T[i] = TYPE_ORA_VARCHAR;
                select_dp->L[i] += 2;
            /*  idist[i].len = select_dp->L[i]; */
                break;
            case TYPE_ORA_CHAR:
                select_dp->T[i] = extern_char_type;
            /*  idist[i].len = select_dp->L[i]; */
                break;
```

```
            case TYPE_ORA_NUMBER:
                sqlprc(&(select_dp->L[i]),&precision,&scale);
                odist[i].attr.pre   = precision;
                odist[i].attr.scale = scale;
                if (precision == 0) precision = 40;
                select_dp->L[i] = precision + 2;
                if (scale < 0) select_dp->L[i] += -scale;
                select_dp->T[i] = extern_char_type;
                idist[i].attr.pre   = precision;
                idist[i].attr.scale = scale;
            /*  idist[i].len = select_dp->L[i]; */
                break;
            case TYPE_ORA_LONG:
                select_dp->T[i] = TYPE_ORA_VARCHAR;
                select_dp->L[i] = MAX_VCHAR_SIZE + 2;
            /*  idist[i].len = select_dp->L[i]; */
                break;
            case TYPE_ORA_ROWID:
                select_dp->T[i] = extern_char_type;
                select_dp->L[i] = 18;
            /*  idist[i].len = select_dp->L[i]; */
                break;
            case TYPE_ORA_DATE:
                select_dp->T[i] = extern_char_type;
                select_dp->L[i] = 75;
            /*  idist[i].len = select_dp->L[i]; */
                break;
        /*****
            case TYPE_ORA_MLSLABEL:
                select_dp->T[i] = TYPE_ORA_VARCHAR;
                select_dp->L[i] = 128;
                idist[i].len = select_dp->L[i];
                break;
        *****/
            case TYPE_ORA_RAW:
                select_dp->T[i] = TYPE_ORA_VARCHAR;
                select_dp->L[i] *= 2;
                select_dp->L[i] += 2;
            /*  idist[i].len = select_dp->L[i]; */
                break;
            case TYPE_ORA_LONGRAW:
                select_dp->T[i] = TYPE_ORA_LONGVARRAW;
                select_dp->L[i] = 12;
            /*  idist[i].len = select_dp->L[i]; */
                longraw = i;
                break;
        /****
            case TYPE_ORA_VARCHAR:
                select_dp->L[i] = MAX_VCHAR_SIZE + 2;
                idist[i].len = select_dp->L[i];
                break;
            case TYPE_ORA_VARRAW:
                select_dp->T[i] = TYPE_ORA_VARCHAR;
                select_dp->L[i] *= 2;
                select_dp->L[i] += 2;
                idist[i].len = select_dp->L[i];
                break;
        ****/
        }
        idist[i].len = select_dp->L[i];
        /* ¥0のエリア用に +1 する */
        if (select_dp->T[i] == TYPE_ORA_STRING) select_dp->L[i]++;
PORA(printf (" --> type=%d L=%d¥n",select_dp->T[i],select_dp->L[i]);)

        if (select_dp->V[i]) Free(select_dp->V[i]);
        if (!(select_dp->V[i] = Malloc(idist[i].len+1))) return ERROR_MALLOC;
        idist[i].attr.dtype = select_dp->T[i];
        idist[i].cpdat = select_dp->V[i];
/*
printf("i=%d, attr.dtype=%d, attr.pre=%d, attr.scale=%d, attr.flag=%d, cpdat=%08x, len=%d¥n",i,idist[i].
attr.dtype,idist[i].attr.pre,idist[i].attr.scale,idist[i].attr.flag,idist[i].cpdat,idist[i].len);
*/
    }
    if (longraw>=0)
        if (di_longraw_set(longraw,pdipCT)) goto SqlErr;
```

```
    return (0);
SqlErr:
    return di_sqlerrm_ora(pdipCT);
}

/**********************************/
/*                                */
/**********************************/
int di_fetch_ora(pdipCT)
qDipCT *pdipCT;
{
    int i, t, ind;
    qDist *idist;

    idist = pdipCT->pidist;
    EXEC SQL FETCH C USING DESCRIPTOR select_dp;
    pdipCT->sqlcode = sqlca.sqlcode;
    if (!sqlca.sqlcode) {
        for (i=0;i<select_dp->N;i++) {
            idist[i].attr.flag = 0;
            t = select_dp->T[i];
            ind = *(select_dp->I[i]);
            if (ind == -1) {    /* null */
                idist[i].attr.flag = 1;
                if (t == TYPE_ORA_CHAR || t == TYPE_ORA_VARCHAR2)
                    *(select_dp->V[i])='¥0';
            }
            else {
                if (ind > 0) idist[i].attr.flag = 2;    /* over flow */
                if (t == TYPE_ORA_CHAR || t == TYPE_ORA_VARCHAR2)
                    *(select_dp->V[i]+select_dp->L[i])='¥0';
            }
        }
    }
PORA(printf ("di_fetch_ora:sqlcode=%d¥n",sqlca.sqlcode);)
    di_sqlerrm_ora(pdipCT);
    return (sqlca.sqlcode);
}

/**********************************/
/*                                */
/**********************************/
int di_eof_ora(pdipCT)
qDipCT *pdipCT;
{
    int i;
    qDist *idist, *odist;
    qDist *iitmn;

    idist = pdipCT->pidist;
    odist = pdipCT->podist;
    iitmn = pdipCT->piitmn;
    for (i=0;i<pdipCT->nidist;i++) {
        iitmn[i].cpdat[0] = '¥0';
        if (idist[i].cpdat) Free(idist[i].cpdat);
        idist[i].cpdat = NULL;
        idist[i].len = 0;
        odist[i].attr.dtype = '¥0';
        select_dp->V[i] = NULL;
    }

    PORA(printf("¥nNumber of rows processed = %d¥n",sqlca.sqlerrd[2]);)
    EXEC SQL CLOSE C;
    di_sqlerrm_ora(pdipCT);
    return (sqlca.sqlerrd[2]);
}

/**********************************/
/*                                */
/**********************************/
int di_end_ora()
{
    sqlclu(bind_dp);
    sqlclu(select_dp);
```

```
    if (indicator) Free(indicator);
    return (0);
}

/************************************/
/*                                  */
/************************************/
int di_commit_rollback_ora(pdipCT,cmd)
qDipCT *pdipCT;
int cmd;
{
    int ret = 0;

    if (cmd == 2)
        EXEC SQL COMMIT   WORK;
    else if (cmd == 3)
        EXEC SQL ROLLBACK WORK;
    else
        ret = -1;

    return di_sqlerrm_ora(pdipCT);
}

/************************************/
/*                                  */
/************************************/
int di_commit_ora(pdipCT)
qDipCT *pdipCT;
{
    return di_commit_rollback_ora(pdipCT,2);
}

/************************************/
/*                                  */
/************************************/
int di_rollback_ora(pdipCT)
qDipCT *pdipCT;
{
    return di_commit_rollback_ora(pdipCT,3);
}

/************************************/
/*                                  */
/************************************/
int di_start_tr_ora(pdipCT)
qDipCT *pdipCT;
{
    return 0;
}

/************************************/
/*                                  */
/************************************/
int di_disconnect_ora(pdipCT,rollback)
qDipCT *pdipCT;
int rollback;
{
    int ret=0;

    if (rollback)
        EXEC SQL ROLLBACK WORK RELEASE;
    else
        EXEC SQL COMMIT   WORK RELEASE;

    return di_sqlerrm_ora(pdipCT);
}

/************************************/
/*                                  */
/************************************/
int di_exec_sql_ora(psql,pdipCT)
char *psql;
qDipCT *pdipCT;
{
    return di_pre_ora(psql,pdipCT);
```

```
}

/**********************************/
/*                                */
/**********************************/
static int di_longraw_set(longraw,pdipCT)
int longraw;
qDipCT *pdipCT;
{
    int ret,max,len;
    char *p,*pdat;
    qDist *idist;
    long lLen, lw;

    idist = pdipCT->pidist;
    max = 0;
    pdat = select_dp->V[longraw];
    p = pdat + 4;
    ret = di_fetch_ora(pdipCT);
    while (!ret) {
        memcpy(&lLen,pdat,4);
PORA(printf ("di_longraw_set:lLen=%d¥n",lLen);)
        if (lLen>=8) {
            if (p[0] == 'A' && p[1] == 'T' && p[2] == 0x50 && p[3] == 0x00) {
                memcpy((char *)&lw,p+4,4);
                len = ntohl(lw);
                if (len>max) max = len;
            }
            else {
                if (lLen > max+8) max = lLen - 8;
            }
        }
        ret = di_fetch_ora(pdipCT);
    }
    if (ret != 0 && (ret != 100 && ret != ERROR_SQL_EOF)) return ret;
/*
    di_eof_ora(pdipCT);
    if (pdipCT->sqlcode) return (pdipCT->sqlcode);
*/
PORA(printf ("di_longraw_set:max=%d¥n",max);)
    if (max) max += 8;
    else max = MAX_LVRAW_SIZE;
    select_dp->L[longraw] = max + 4;
    idist[longraw].len = select_dp->L[longraw];
    if (select_dp->V[longraw]) Free(select_dp->V[longraw]);
    if (!(select_dp->V[longraw] = Malloc(idist[longraw].len+1)))
         return ERROR_MALLOC;
    idist[longraw].cpdat = select_dp->V[longraw];
    EXEC SQL OPEN C USING DESCRIPTOR bind_dp;
PORA(printf ("di_longraw_set:sqlcode=%d¥n",sqlca.sqlcode);)
    if (sqlca.sqlcode) return (-1);
    return 0;
}
```