Manipulation de données avec 4D Open

Par
Aziz ELGHOMARI, Support Technique 4D
Note technique 4D-200003-08-FR
Version 1

Date 1 Mars 2000

Résumé

La présente note technique permet de manipuler des données avec les librairies 4D Open et effectuer des traitements concernant les suppressions et ajouts de données, le tri et la recherche d'informations ainsi que la manipulation des ensembles et des Blobs.

4D Notes techniques

Introduction

La première note technique se rapportant aux librairies de 4D Open, concernait l'accès à la structure d'une base lancée avec 4D Serveur (Note technique 99-33).
La présente note technique traite cette fois-ci de l'accès aux données d'une base tournant sur le serveur.

Les utilisateurs des librairies 4D open peuvent, à partir de cette note technique, construire un client "léger" pouvant traiter les données d'un 4D Serveur (l'application openpack.exe et la librairie 4dopen.DLL réunies pèsent 260 Ko alors qu'un 4D Client fait 8,6 Mo).

Pour réaliser cet exemple, nous allons aborder les sujets suivants :

- Suppression et Ajout d'enregistrements
- Recherche d'enregistrements
- Tri des sélections d'enregistrements
- Manipulation d'ensembles
- Manipulation de Blobs

Supprimer et Ajouter des enregistrements

Avant d'ajouter des enregistrements, nous allons supprimer les éventuels enregistrements de la base, et pour cela utiliser les routines :

_4D_SelectAllRecords
_4D_DeleteSelectedRecords

Les ajouts affecterons les champs suivants : Alpha, Texte, Numérique, Entier, Entier long , Date, Heure, Booléen, Image, BLOB.

La structure contient les éléments suivants :

```
typedef struct DataRec {
        DataType        typ;
        byte            filler;
        union {
        TE4D            text;       /* s_Text32K */
        st80            s;                  /* s_AlphaNumeric */
        Real            r;                  /* s_Number  */
        int2            b;                  /* s_Bool    */
        int2            i;                  /* s_Int16   */
        int4            l;                  /* s_Int32   */
        int4            tim;                /* s_TimeSec */
```

```
        Date4D          d;              /* s_Dates   */
        Pict4D          pic;            /* s_Pictures */
        Blob4D          blob;           /* s_Blob    */
        struct { st80 s2 ; int2 unused; } other ;
        } u;

        } DataRec;
```

Pour les ajouts d'enregistrements, les principales routines utilisées dans l'exemple :

_4D_CreateBuffer(ConnectHandle CIDH,int2 Targetfile)

CIDH : Correspond au Handle de connexion retourné par la routine 4D? 4D_Open4DConnection
Targetfile : Correspond au numéro de fichier sur lequel, on veut créer le Buffer

_4D_AddToBuffer(ConnectHandle CIDH,BufferHandle Buffer, int2 FieldNumber, DataRec *Field)

Buffer	Le handle retourné par la routine _4D_CreateBuffer
FieldNumber	Numéro du champ.
*Field	La valeur du champs contenu dans le DataRec

Rechercher un enregistrement

Dans l'exemple fourni, la recherche porte sur le champ 4 de la table 1 qui est de type entier , et cherche les enregistrements dont le champ 4 est égal à 999.

La routine à utiliser est :

_4D_Search(ConnectHandle CIDH, SearchRecordPtr What, int4 *RecordsFound)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
What : On lui passe un pointeur sur la structure SearchRecord
RecordsFound : Elle retourne ne nombre d'enregistrements trouvés

Les structures SearchLine et SerchRecord :

```
typedef struct SearchLine {
        int2            Field_Number;
        int2            File_Number;
        byte            SOP;            /* search operator */
        byte            unused;
        DataRec         Value;
        byte            LOP;    /* logical operator */
```

```
        byte                    unused2,unused3 ;
                } SearchLine;

typedef struct SearchRecord {
        int2                    NB_Lines;
        int2                    TargetFile;
        SearchLine              lines[1];
        } SearchRecord;


typedef SearchRecord *SearchRecordPtr;
```

Dans le source de la routine de recherche, les lignes suivantes se chargent de remplir la structure :

pSearch->NB_Lines = 1
La recherche concerne le premier niveau de recherche, dans ce cas il existe un seul niveau de recherche

pSearch->TargetFile = 1
La recherche concerne le fichier numéro 1

pSearch->lines[0].Field_Number = 4
La recherche concerne le champ numéro 4 de type Entier

pSearch->lines[0].File_Number = 1
Concerne le fichier contenant le champ retenu pour la recherche

pSearch->lines[0].SOP = Equal
L'opérateur de recherche est égal

pSearch->lines[0].Value = data
La valeur à rechercher dans le champ entier est 999

pSearch->lines[0].LOP = None2
L'opérateur logique est vide car il n'existe qu'une seule ligne de recherche.

Trier les enregistrements

Dans l'exemple de cette note technique, le tri s'effectuera sur le champ 1.

Les structures à remplir pour le tri sont SortLine et SortRecord :

```
typedef struct SortLine {
        int2                            Field_Number;
```

```
        int2                    File_Number;
        byte                    Ascent;
        } SortLine;

typedef struct SortRecord {
        int2                    NB_Lines;
        int2                    TargetFile;
        SortLine        lines[1];
        } SortRecord;

typedef SortRecord *SortRecordPtr;
```

La routine à utiliser est :
_4D_Sort(hConnect, pSort)

hConnect : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
pSort : Correspond au pointeur sur la structure SortRecord

Dans le source de la routine de tri, les lignes suivantes se chargent de remplir la structure :

pSort->NB_Lines = 1
Correspond au numéro du niveau de tri

pSort->TargetFile = 1
Correspond au numéro du fichier sur lequel porte le tri

pSort->lines[0].File_Number = 1
Correspond au numéro de fichier contenant le champs

pSort->lines[0].Field_Number = 1
Correspond au champ sur lequel va s'effectuer le tri. C'est le champ alphanumérique en l'occurrence.

pSort->lines[0].Ascent = 1
Cette ligne permet de fixer l'ordre de tri, 1 correspond au tri ascendant, 0 correspond au tri descendant

Manipuler les ensembles

Pour l'exemple concernant la manipulation des ensembles, nous aurons besoin de créer deux ensembles : MySet1 et MySet2.
Le 1er ensemble contiendra l'enregistrement courant
Le 2ème ensemble contiendra le résultat d'une recherche
On affiche ensuite le nombre d'éléments contenus dans les ensembles MySet1 et MySet2

Les routines qui concernent les ensembles :

_4D_CreateEmptySet(ConnectHandle CIDH, st80 SetName, int2 Targetfile)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4Dconnection
SetName : Nom de l'ensemble vide à créer
Targetfile : ID de la table

Cette commande crée un ensemble vide pour la table spécifié

_4D_CreateSet(ConnectHandle CIDH, st80 SetName, int2 Targetfile)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4Dconnection
SetName : Nom de l 'ensemble à créer.
Targetfile : ID de la table

Cette commande crée un nouvel ensemble pour la table spécifiée et met la sélection courante dans l'ensemble spécifié

_4D_AddToSet(ConnectHandle CIDH, st80 SetName, int2 Targetfile)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4Dconnection
SetName : Nom de l'ensemble
TargetName : Identifiant de la table

Cette commande ajoute la sélection courante de la table spécifiée à l'ensemble spécifié

_4D_RecordsInSet(ConnectHandle CIDH, st80 SetName, int4 *recInSet)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
SetName : Nom de l'ensemble
*recInSet : Nombre d'enregistrements dans l'ensemble

Cette commande retourne dans recInSet le nombre d'enregistrements présents dans l'ensemble spécifié.

_4D_ClearSet(ConnectHandle CIDH, st80 SetName)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
SetName : Ensemble à vider

Cette commande libère la mémoire utilisée par l'ensemble.

Manipulation des blobs

_4D_SelectAllRecords(ConnectHandle CIDH, int2 Targetfile)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
Targetfile : Numéro de fichier sur lequel on applique un tout sélectionner

_4D_GotoSelectedRecord(ConnectHandle CIDH, int2 Targetfile, int4 RecNumberInSel)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
Targetfile : Numéro de fichier sur lequel on sélectionne le RecNumberInSel
RecNumberInSel Détermine la position dans la sélection

_4D_GetFields(ConnectHandle CIDH, ReqFieldRecPtr FieldList, BufferHandle *Buffer, byte *IsLocked)

CIDH Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
FieldList La liste des champs concernés
*Buffer Handle sur les données des champs

Pour déterminer le champ à utiliser , il faut remplir la structure suivante :

```
typedef struct ReqFieldRec {
        int2            NB_Fields;
        int2            TargetFile;
        int2            Fields[1];
        } ReqFieldRec
```

_4D_GetNthField(ConnectHandle CIDH,BufferHandle Buffer, int2 FieldNumber, DataRec *Field)

CIDH : Correspond au Handle de connexion retourné par la routine 4D_Open4DConnection
Buffer : Handle sur les données du champ retourné par la routine _4D_GetFields
FieldNumber : Numéro de champs relatif
*Field : Valeur du champ

La structure permettant de récupérer les valeurs des champs est la suivante :

```
typedef struct DataRec {
        DataType        typ;
        byte                    filler;
        union {
        TE4D            text;   /* s_Text32K */
        st80            s;              /* s_AlphaNumeric */
        Real            r;              /* s_Number  */
        int2            b;              /* s_Bool    */
        int2            i;              /* s_Int16   */
        int4            l;              /* s_Int32   */
        int4            tim;            /* s_TimeSec */
        Date4D          d;              /* s_Dates   */
```

```
Pict4D          pic;            /* s_Pictures */
Blob4D          blob;           /* s_Blob    */
struct { st80 s2 ; int2 unused; } other ;
} u;
 } DataRec;
```

Si on veut accéder à la taille du champ Blob par exemple, il faut utiliser :
data.u.blob.BlobLen

Mode d'utilisation

Cette note technique est fournie avec deux projets : un projet CW Pro 5 sous Macintosh et un projet Visual C++6 sous Windows.

Sous Windows, pour lancer l'exécutable, il faut placer le fichier 4D Open.dll dans le dossier Debug, à côté de l'exécutable openpack.exe.

Si vous voulez recompiler le projet sous Macintosh, il faut ajouter dans le dossier Libs la librairie 4D Open.PPC.Lib
Si vous désirez recompiler le projet sous Windows, il faut ajouter dans le dossier du projet la librairie 4D Open .lib.

Note : vous pouvez installer ces librairies à partir du CD Rom lors de l'installation de 4D Open.

La base doit être lancée sur le réseau, le protocole réseau utilisé est défini au début du programme par la ligne :
Ex const short KNC = NC_TCPIP

Par défaut, le protocole utilisé est TCPIP pour l'environnement Windows et Macintosh.

La base lancée par le 4D Serveur se compose d'une table contenant des champs de types suivants :
Alpha, Texte, Numérique, Entier, Entier long, Date, Heure, Booléen, Image, BLOB.

La base utilisée doit être lancée avec un 4D Serveur version 6.5.x, les composants réseaux utilisés avec notre projet étant ceux de la version 6.5.