Catalog Functions in ODBC Pro

By Noreddine Margoum, QA Engineer, 4D S.A.
TN 06-10

Introduction

The purpose of this technical note is to demonstrate the use of the ODBC Pro plug-in commands that describe the catalog of a database.

Catalog functions

Note: the term 'column' will be used as a synonym of 'field'.

Databases all have a structure that allows access to information such as the list of tables, privileges for a given table, etc. This structure is commonly referred to as the catalog of the database.

Using the ODBC Pro commands of the catalog, you can describe a structure in its entirety.

The main use for these commands is to:

- build dynamic SQL requests while the application is executing, because the list of tables and columns can be obtained through the catalog functions ***ODBC_SQLTables*** and ***ODBC_SQLColumns***

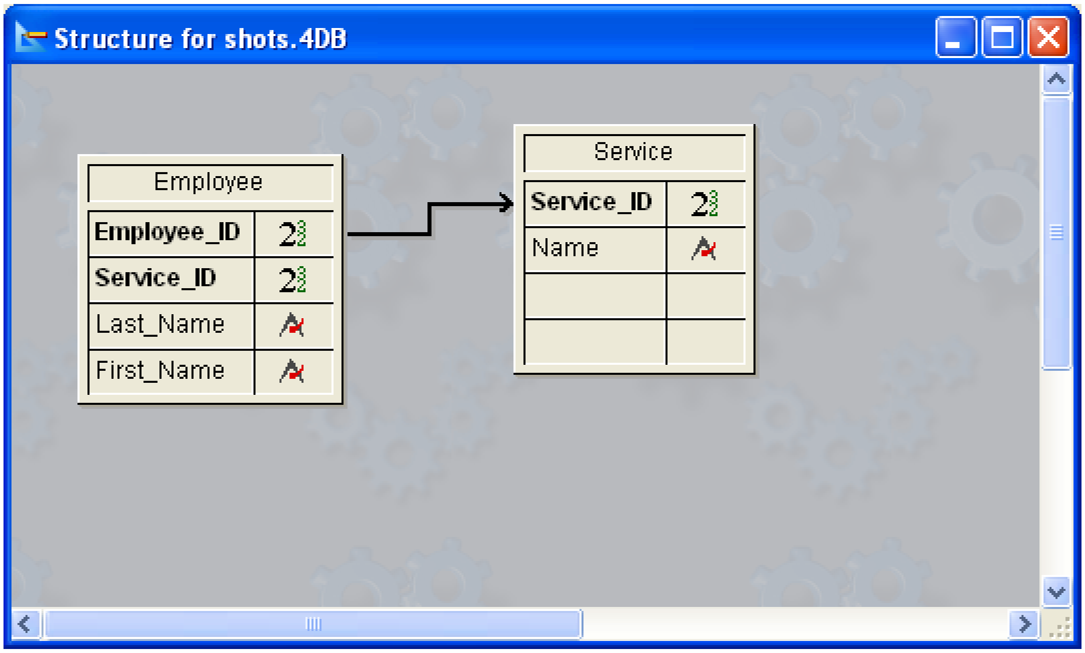
- make SQL requests more efficient by taking advantage of relations between tables and indexes, via the commands ***ODBC_SQLPrimaryKeys***, ***ODBC_SQLForeignKeys***, and ***ODBC_SQLStatistics***

Below is the list of catalog commands, as well as their description:

Commands	**Description**
ODBC_SQLTables	Returns the list of catalog, schemas, tables and table types in the data source;
ODBC_SQLColumns	Returns the list of columns for one or several tables;
ODBC_SQLStatistics	Returns a list of statistics (number of records of the table, number of unique value of the index, etc.) for one table or returns the list of index associated to the table;
ODBC_SQLSpecialColumns	Returns the list of columns that identify a record in a unique manner. Returns also, for that table, the list of

	columns that are automatically updated when any column is updated through a transaction;
ODBC_SQLPrimaryKeys	Returns the list of columns that are part of a primary key for a given table. As a reminder a primary key is a column or a combination of columns that allows you to uniquely identify a record in a table;
ODBC_SQLForeignKeys	Returns the list of foreign keys* for a given table or returns the list of foreign keys that are pointing towards a given table. As a reminder, a foreign key for a given table points to the primary key in another table;
ODBC_SQLTablePrivileges	Returns the list of tables and their privileges, the user that assigns the privilege as well as the user to which the privilege is assigned;
ODBC_SQLColumnPrivileges	Returns, for a given table, the list of columns and the privileges that are associated to them.;
ODBC_SQLProcedures	Returns the list of stored procedures in the data source. The term of stored procedures is generic and covers both procedures and functions;
ODBC_SQLProcedureColumns	Returns the list of input and output parameters as well as columns that may constitute the result of the execution of the procedure. This command determines also the type of each parameter (Input, Input/Output, Output) and, if the if the parameter is a column;
ODBC_SQLGetTypeInfo	Returns the list of data types supported by the data source. These data types are used in SQL requests that define data such as CREATE TABLE and ALTER TABLE.

* To better understand the notion of foreign key and primary key, here is an example:



In this simple case, the [Employee] table contains a primary key Employee_ID.
The table [Service] contains the primary key Service_ID. The primary key is used to identify each record of the table in a unique manner. There are data systems where the primary key consists of several columns.

In this case, there is only one foreign key, Service_ID, contained in the [Employee] table. This foreign key references the table [Service] in the table [Employee]. It defines the relation of the employee 'to' the service. There are data systems where the foreign key consists of several columns.

All the catalog commands return their results as a sorted list of columns. To retrieve the data if those columns, you will need to associate a field or a 4D variable to the column, based on the position of the column.

Using the functions of the catalog

General use

```
C_LONGINT($0) `return code of the method
C_LONGINT($1;$vl_ConnectionID) `Connection ID
C_TEXT($2)
C_POINTER(${3}) ` pointers to array to retrieve data resulting from request
C_LONGINT($vl_StmtID) `statement handle
C_TEXT($vt_CatalogName) `Catalog name
C_TEXT($vt_SchemaName) `Schema name
C_TEXT($vt_TableName) `Table name
C_TEXT($vt_ColumnName) `Column name

$vl_ConnectionID:=$1
$vt_CatalogName:=" "
$vt_SchemaName:=" "
$vt_TableName:=$2
$vt_ColumnName:=" "

  `allocatign the statement handle
vl_Status:=ODBC_SQLAllocStmt ($vl_ConnectionID;$vl_StmtID)
vl_Status:=ODBC_SQLColumnPrivileges
($vl_StmtID;$vt_CatalogName;$vt_SchemaName;$vt_TableName;$vt_ColumnName)
If ((vl_Status=SQL_SUCCESS ) | (vl_Status=SQL_SUCCESS_WITH_INFO ))
  vl_Status:=ODBC_SQLBindCol ($vl_StmtID;4;$3) `column name
  vl_Status:=ODBC_SQLBindCol ($vl_StmtID;5;$4) `name of assigner
  vl_Status:=ODBC_SQLBindCol ($vl_StmtID;6;$5)
  vl_Status:=ODBC_SQLBindCol ($vl_StmtID;7;$6) `privilège
  While ((vl_Status#SQL_NO_DATA ) & (vl_Status#SQL_ERROR ))
    vl_Status:=ODBC_SQLFetch ($vl_StmtID)
  End while
End if
```

```
vl_Status:=ODBC_SQLFreeStmt ($vl_StmtID;SQL_CLOSE )
vl_Status:=ODBC_SQLFreeStmt ($vl_StmtID;SQL_DROP )
```

Calling the commands of the catalog follows the structure of the code above.

1. Allocating a statement handle (longint) using the connection handle(***ODBC_SQLAllocStmt***);

2. Calling the catalog command with the statement handle as the first parameter, followed by the parameters specific to the command;

3. The execution of the catalog command, when successful, results in the creation of the sorted list of columns;

4. To each column in the result, we associate a variable or a field. The position of the column in the result is critical and needs to be indicated at the time of the bind. The bind Column/Field or variable is set by the following call:
ODBC_SQLBindCol(HandleRequete;PositionColonne;->VariableouChamp4D) ;

5. Data resulting from the command are retrieved using the command ***ODBC_SQLFetch***.
The conditions for this command to stop are either an error escalation or the end of the data retrieval;

6. Closing and freeing the request.

Implementation example

```
C_LONGINT($0) `return parameter
C_LONGINT($1;$vl_ConnectionID) `Connection ID
C_TEXT($2) `Table name
C_POINTER(${3}) `pointer to the arrays of the column names, and their types and sizes
C_LONGINT($vl_StmtID) `statement handle
C_TEXT($vt_CatalogName) `catalog name
C_TEXT($vt_SchemaName) `schema name
C_TEXT($vt_TableName) `table name
C_TEXT($vt_ColumnName) `column name


$vl_ConnectionID:=$1
$vt_CatalogName:=" "
$vt_SchemaName:=" "
$vt_TableName:=$2
$vt_ColumnName:=" "


   `allocating the statement handle
vl_Status:=ODBC_SQLAllocStmt ($vl_ConnectionID;$vl_StmtID)
```

```
vl_Status:=ODBC_SQLColumns
($vl_StmtID;$vt_CatalogName;$vt_SchemaName;$vt_TableName;$vt_ColumnName)
If ((vl_Status=SQL_SUCCESS ) | (vl_Status=SQL_SUCCESS_WITH_INFO ))
  vl_Status:=ODBC_SQLBindCo   ($vl_StmtID;4;$3)  `column   names
  vl_Status:=ODBC_SQLBindCo   ($vl_StmtID;6;$4)  `column   types
  vl_Status:=ODBC_SQLBindCo   ($vl_StmtID;8;$5)  `column   sizes
  While ((vl_Status#SQL_NO_DATA ) & (vl_Status#SQL_ERROR ))
    vl_Status:=ODBC_SQLFetch   ($vl_StmtID)
  End while
Else
  $0:=SQL_ERROR
End if
vl_Status:=ODBC_SQLFreeStm   ($vl_StmtID;SQL_CLOSE )
vl_Status:=ODBC_SQLFreeStm   ($vl_StmtID;SQL_DROP )
```

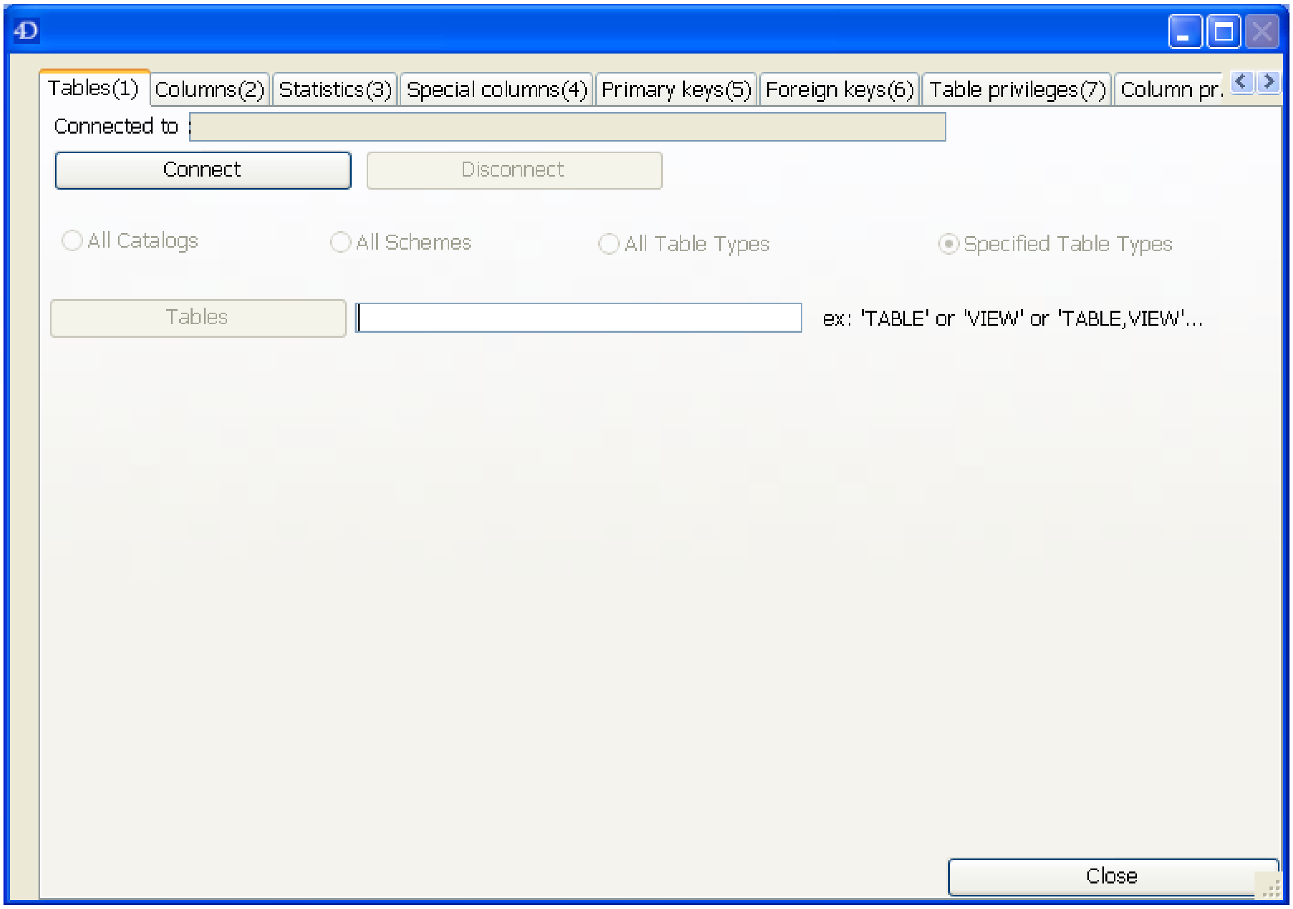
Comments on the method:

This method returns the list of columns for a table, as well as their type and size. It uses the command ***ODBC_SQLColumns***. This method follows the structure indicated earlier.

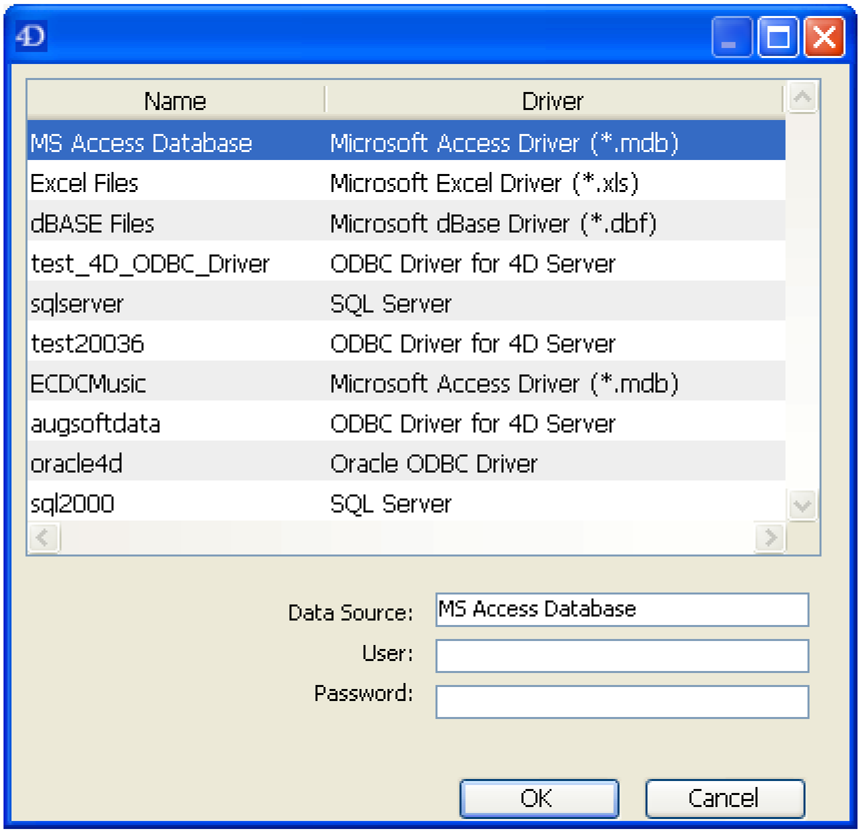
Using the Test database

The test database includes the implementation of all the commands of the catalog.

At the launch of the database, the main dialog is displayed (you can also access it through the File/Demo menu):



By clicking the Connect button, you can select the data source to which you want to connect. The data source has to be configured in the ODBC data source administrator window.



Name	Driver
MS Access Database	Microsoft Access Driver (*.mdb)
Excel Files	Microsoft Excel Driver (*.xls)
dBASE Files	Microsoft dBase Driver (*.dbf)
test_4D_ODBC_Driver	ODBC Driver for 4D Server
sqlserver	SQL Server
test20036	ODBC Driver for 4D Server
ECDCMusic	Microsoft Access Driver (*.mdb)
augsoftdata	ODBC Driver for 4D Server
oracle4d	Oracle ODBC Driver
sql2000	SQL Server

Data Source: MS Access Database
User:
Password:

Now that the connection is established with the data source, you can use the different features of the dialog.

Conclusion

This technical note demonstrated the use of the ODBC catalog commands to describe the structure of a database.