ODBC i C/C++

Denne artikel beskriver hvordan man bruger ODBC i C/C++.

Der er beskrivelse af build med forskellige compilere.

Den forudsætter lidt kendskab til C/C++ og SQL.

Skrevet den **04. Feb 2009** af **arne_v** I kategorien **Programmering / C/C++** | ★★★☆☆

Historie:
V1.0 - 07/03/2004 - original
V1.1 - 14/12/2007 - close cursor
V1.2 - 26/12/2008 - små ændringer

ODBC

Mange kender ODBC (Open Database Connectivity) og har brugt det, men meget få er klar over at det i virkeligheden er et C API.

ODBC er opfundet for at give et database uafhængigt API.

Det fungerer som:

applikation----ODBC driver manager----foobar ODBC driver----foobar database

ODBC definerer nogle standard C funktioner som man kan kalde uanset hvad databasen er.

De fleste kender nok ODBC som:

applikation----X----ODBC driver manager----foobar ODBC driver----foobar database

hvor det mellem liggende lag X gør det nemt at bruge ODBC fra Access, ASP etc., men det er stadig de samme C funktioner der kaldes. Det er bare koden i X som kalder dem. Og slut brugeren får et lidt nemmere interface.

Denne artikel vil beskrive hvordan man lave simple forespørgsler og opdateringer i en database via ODBC.

Man kan også få ODBC til ikke Windows platforme. Men de er ikke så meget brugte (embedded SQL er mere udbredt). Og jeg har aldrig prøvet det. Så det vil jeg ikke komme ind på. Jeg formoder at API'et er det samme.

ODBC API dokumentationen er online her:
http://msdn.microsoft.com/en-us/library/ms710154.aspx

Fordele og ulemper

Fordele:
- virker med alle database (næsten alle databaser har en ODBC driver)
- virker med alle compilere (kræver ikke C++, MFC, ATL eller andet)
- rimeligt simpelt at forstå

Ulemper:
- et lidt klodset API som godt kan give meget kode

Jeg er forbløffet over at ODBC kald ikke bruges mere i C/C++.

Der er kommet nyere teknologi i form af OLE DB og ADO. Men de har mindre support fra database leverandørerne og kræver kommercielle compilere.

Det er almindeligt kendt at OLE DB / ADO performer bedre end ODBC i VBA/VBS/VB. Men jeg er ikke overbevist om at det samme gælder i C/C++. Min hypotese er at OLE DB er designet og dermed optimeret til at skulle bruges i VBA/VBS/VB, mens ODBC er designet og dermed optimeret til at skulle bruges i C/C++, og at det er X laget der gør ODBC langsomt. Men jeg kan ikke bevise min hypotese.

Eksempler

Eksemplerne vil bruge denne meget simple database:

```
CREATE TABLE t1 (
  f1 INTEGER;
  f2 VARCHAR(50),
  PRIMARY KEY(f1)
);
```

Eksemplerne vil være i C d.v.s. kunne bruges både i C og C++.

Eksemplerne er meget simple, men bør nemt kunne udvides til noget mere avanceret udfra dokumentationen på ovennævnte link.

Eksemplerne er testet mod en Access database, men ODBC er som sagt database uafhængig. For at bruge MySQL skal man kun ændre DSN eller connection string.

Forespørgsel

Med DSN:

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

```
/* nødvendige header filer for at bruge ODBC */
#include <windows.h>
#include <sql.h>
#include <sqlext.h>

/*
* åben connection til:
*    DSN = "TestMSAccess"
*    username = "" (bliver opfattet som defaut "Admin" i Access)
*    password = ""
*/
char *dsn = "TestMSAccess";
char *un = "";
char *pw = "";

int main()
{
   /* ODBC environment */
   SQLHENV Environment;
   /* ODBC connection */
   SQLHDBC DataBaseConnect;
   /* ODBC statement */
   SQLHSTMT stmt;
   /* ODBC return status */
   SQLRETURN stat;
   /* query */
   char *sqlstr = "SELECT * FROM T1";
   /* integer værdi (f1) */
   int i;
   /* string værdi (f2) */
   char s[50];
   /* længder af integer og string værdi */
   int il,sl;
   /* alloker environment = initialiser ODBC */
   stat = SQLAllocEnv(&Environment);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocEnv\n");
   }
   /* connect til database */
   stat = SQLAllocConnect(Environment,&DataBaseConnect);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocConnect\n");
   }
   stat = SQLConnect(DataBaseConnect,
                     (SQLCHAR *)dsn,(SQLSMALLINT)strlen(dsn),
                     (SQLCHAR *)un,(SQLSMALLINT)strlen(un),
                     (SQLCHAR *)pw,(SQLSMALLINT)strlen(pw));
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in Connect\n");
   }
   /* alloker statement */
   stat = SQLAllocStmt(DataBaseConnect,&stmt);
```

```
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocStmt\n");
   }
   /* udfør statement */
   stat = SQLExecDirect(stmt,(SQLCHAR *)sqlstr,strlen(sqlstr));
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in ExecDirect\n");
   }
   /* bind kolonne 1 (f1) til i (integer værdi) og kolonne 2 (f2) til sv
(string værdi) */
   stat = SQLBindCol(stmt,1,SQL_C_LONG,&i,sizeof(i),(SQLINTEGER *)&il);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in BindCol\n");
   }
   stat = SQLBindCol(stmt,2,SQL_C_CHAR,s,sizeof(s),(SQLINTEGER *)&sl);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in BindCol\n");
   }
   /* hent alle rækker */
   for(;;)
   {
      stat = SQLFetch(stmt);
      if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO)) break;
      s[sl] = '\0';
      printf("%d %s\n",i,s);
   }
   /* luk cursor så statement kan genbruges til ny forespørgsel */
   SQLCloseCursor(stmt);
   /* frigør statement */
   SQLFreeStmt(stmt,SQL_DROP);
   /* disconnect og frigør connection */
   SQLDisconnect(DataBaseConnect);
   SQLFreeConnect(DataBaseConnect);
   /* frigør environment */
   SQLFreeEnv(Environment);
   return 0;
}
```

Uden DSN:

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

/* nødvendige header filer for at bruge ODBC */
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
```

```
/*
* åben connection til:
*    driver = "{Microsoft Access Driver (*.mdb)}"
*    database = "D:\\Database\\MSAccess\\Test.mdb"
*    username = "Admin"
*    password = ""
*/
char *constr = "Driver={Microsoft Access Driver
(*.mdb)};Dbq=D:\\Database\\MSAccess\\Test.mdb;Uid=Admin;Pwd=;";

int main()
{
   /* ODBC environment */
   SQLHENV Environment;
   /* ODBC connection */
   SQLHDBC DataBaseConnect;
   /* ODBC statement */
   SQLHSTMT stmt;
   /* ODBC return status */
   SQLRETURN stat;
   /* returned connection string */
   char outconstr[1024];
   /* length of returned connection string */
   int outconlen;
   /* query */
   char *sqlstr = "SELECT * FROM T1";
   /* integer værdi (f1) */
   int i;
   /* string værdi (f2) */
   char s[50];
   /* længder af integer og string værdi */
   int il,sl;
   /* alloker environment = initialiser ODBC */
   stat = SQLAllocEnv(&Environment);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocEnv\n");
   }
   /* connect til database */
   stat = SQLAllocConnect(Environment,&DataBaseConnect);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocConnect\n");
   }
   stat = SQLDriverConnect(DataBaseConnect,NULL,
                           (SQLCHAR *)constr,(SQLSMALLINT)strlen(constr),
                           (SQLCHAR *)outconstr,
(SQLSMALLINT)sizeof(outconstr),
                           (SQLSMALLINT *)&outconlen,SQL_DRIVER_COMPLETE);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in Connect\n");
   }
   /* alloker statement */
   stat = SQLAllocStmt(DataBaseConnect,&stmt);
```

```
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocStmt\n");
   }
   /* udfør statement */
   stat = SQLExecDirect(stmt,(SQLCHAR *)sqlstr,strlen(sqlstr));
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in ExecDirect\n");
   }
   /* bind kolonne 1 (f1) til i (integer værdi) og kolonne 2 (f2) til sv
(string værdi) */
   stat = SQLBindCol(stmt,1,SQL_C_LONG,&i,sizeof(i),(SQLINTEGER *)&il);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in BindCol\n");
   }
   stat = SQLBindCol(stmt,2,SQL_C_CHAR,s,sizeof(s),(SQLINTEGER *)&sl);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in BindCol\n");
   }
   /* hent alle rækker */
   for(;;)
   {
      stat = SQLFetch(stmt);
      if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO)) break;
      s[sl] = '\0';
      printf("%d %s\n",i,s);
   }
   /* luk cursor så statement kan genbruges til ny forespørgsel */
   SQLCloseCursor(stmt);
   /* frigør statement */
   SQLFreeStmt(stmt,SQL_DROP);
   /* disconnect og frigør connection */
   SQLDisconnect(DataBaseConnect);
   SQLFreeConnect(DataBaseConnect);
   /* frigør environment */
   SQLFreeEnv(Environment);
   return 0;
}
```

Opdatering

Med DSN:

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

/* nødvendige header filer for at bruge ODBC */
#include <windows.h>
```

```
#include <sql.h>
#include <sqlext.h>

/*
* åben connection til:
*    DSN = "TestMSAccess"
*    username = "" (bliver opfattet som defaut "Admin" i Access)
*    password = ""
*/
char *dsn = "TestMSAccess";
char *un = "";
char *pw = "";

int main()
{
   /* ODBC environment */
   SQLHENV Environment;
   /* ODBC connection */
   SQLHDBC DataBaseConnect;
   /* ODBC statement */
   SQLHSTMT stmt;
   /* ODBC return status */
   SQLRETURN stat;
   /* insert */
   char sqlstr[200];
   /* counter */
   int i;
   /* alloker environment = initialiser ODBC */
   stat = SQLAllocEnv(&Environment);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocEnv\n");
   }
   /* connect til database */
   stat = SQLAllocConnect(Environment,&DataBaseConnect);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocConnect\n");
   }
   stat = SQLConnect(DataBaseConnect,
                     (SQLCHAR *)dsn,(SQLSMALLINT)strlen(dsn),
                     (SQLCHAR *)un,(SQLSMALLINT)strlen(un),
                     (SQLCHAR *)pw,(SQLSMALLINT)strlen(pw));
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in Connect\n");
   }
   /* alloker statement */
   stat = SQLAllocStmt(DataBaseConnect,&stmt);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocStmt\n");
   }
   /* indsæt 10 rækker */
   for(i=0;i<10;i++)
```

```
   {
      sprintf(sqlstr,"INSERT INTO t1 VALUES(%d,'%s')",100+i,"Dette er en
test");
      stat = SQLExecDirect(stmt,(SQLCHAR *)sqlstr,strlen(sqlstr));
      if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
      {
         printf("Error in ExecDirect\n");
      }
   }
   /* frigør statement */
   SQLFreeStmt(stmt,SQL_DROP);
   /* disconnect og frigør connection */
   SQLDisconnect(DataBaseConnect);
   SQLFreeConnect(DataBaseConnect);
   /* frigør environment */
   SQLFreeEnv(Environment);
   return 0;
}
```

Uden DSN:

```

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

/* nødvendige header filer for at bruge ODBC */
#include <windows.h>
#include <sql.h>
#include <sqlext.h>

/*
* åben connection til:
*    driver = "{Microsoft Access Driver (*.mdb)}"
*    database = "D:\\Database\\MSAccess\\Test.mdb"
*    username = "Admin"
*    password = ""
*/
char *constr = "Driver={Microsoft Access Driver
(*.mdb)};Dbq=D:\\Database\\MSAccess\\Test.mdb;Uid=Admin;Pwd=;";

int main()
{
   /* ODBC environment */
   SQLHENV Environment;
   /* ODBC connection */
   SQLHDBC DataBaseConnect;
   /* ODBC statement */
   SQLHSTMT stmt;
   /* ODBC return status */
   SQLRETURN stat;
   /* returned connection string */
   char outconstr[1024];
```

```
   /* length of returned connection string */
   int outconlen;
   /* insert */
   char sqlstr[200];
   /* counter */
   int i;
   /* alloker environment = initialiser ODBC */
   stat = SQLAllocEnv(&Environment);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocEnv\n");
   }
   /* connect til database */
   stat = SQLAllocConnect(Environment,&DataBaseConnect);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocConnect\n");
   }
   stat = SQLDriverConnect(DataBaseConnect,NULL,
                           (SQLCHAR *)constr,(SQLSMALLINT)strlen(constr),
                           (SQLCHAR *)outconstr,
(SQLSMALLINT)sizeof(outconstr),
                           (SQLSMALLINT *)&outconlen,SQL_DRIVER_COMPLETE);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in Connect\n");
   }
   /* alloker statement */
   stat = SQLAllocStmt(DataBaseConnect,&stmt);
   if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
   {
      printf("Error in AllocStmt\n");
   }
   /* indsæt 10 rækker */
   for(i=0;i<10;i++)
   {
      sprintf(sqlstr,"INSERT INTO t1 VALUES(%d,'%s')",100+i,"Dette er en
test");
      stat = SQLExecDirect(stmt,(SQLCHAR *)sqlstr,strlen(sqlstr));
      if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO))
      {
         printf("Error in ExecDirect\n");
      }
   }
   /* frigør statement */
   SQLFreeStmt(stmt,SQL_DROP);
   /* disconnect og frigør connection */
   SQLDisconnect(DataBaseConnect);
   SQLFreeConnect(DataBaseConnect);
   /* frigør environment */
   SQLFreeEnv(Environment);
   return 0;
}
```

Build

Nedenfor angives hvordan man builder command line med forskellige compilere.

Hvis man bruger en IDE skal man lave noget tilsvarende.

Microsoft VC++:

```
cl query.c odbc32.lib
cl insert.c odbc32.lib
```

Borland C++ Builder:

```
bcc32 query.c \borland\bcc55\lib\PSDK\odbc32.lib
bcc32 insert.c \borland\bcc55\lib\PSDK\odbc32.lib
```

(den gratis command line BCB 5.5 kommer tilsyneladende ikke med ODBC32.LIB, men den kan man få ved at installere Microsoft Platform SDK - og jeg er sikker på at betalings versionerne har den)

GCC/MinGW32:

```
gcc query.c -lodbc32 -o query.exe
gcc insert.c -lodbc32 -o insert.exe
```

Fejl

Hvis man får en fejl på en SQL sætning og gerne vil vide årsagen kan man:

```
int errptr,msglen;
char state[5],msg[1000];
...
SQLGetDiagRec(SQL_HANDLE_STMT, stmt, 1, (SQLCHAR*)state, (SQLINTEGER*)&errptr,
(SQLCHAR*)msg, sizeof(msg), (SQLSMALLINT*)&msglen);
printf("Error text = %s\n", msg);'
```

Kommentar af mathiash d. 01. Jan 2006 | 1

Kommentar af looppen d. 09. Aug 2005 | 2

Kommentar af brhino d. 26. Mar 2004 | 3