Some Specifics of ODBC with C

5DV119 — Introduction to Database Management

Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

ODBC Documentation for C

- ODBC with C is a mature subject.
- There is an extensive online reference manual maintained by Microsoft (link on the course Web site).
-  It is a *reference manual*, not a tutorial.
- There are also many other Web sites with documentation.
- The only real book on the subject is out of print (and was for C++ and so very heavy on boilerplate).
- These notes and the example programs in C should provide enough to do the ODBC laboratory exercise.
- C is a difficult language to learn.
- Students who have learned Python but do not know C are advised to do the laboratory exercise using Python.
- Although the examples are intended to be generic, they have been tested only with the PostgreSQL (and in some cases MySQL) DBMS, using Debian Linux as the client-side operating system.
- For programming submissions in this course, the reference machine will be *salt*, which runs Debian 7 Linux.

Compiling a C Program with ODBC Calls

- One of the following should work to compile `program.c`.:

```
cc -lodbc program.c
cc -liodbc program.c
```

- Although they are supposed to be functionally equivalent, `odbc` exhibits much better compatibility with MySQL and should be used if possible.
- The libraries `odbc` and `iodbc` cannot normally co-exist on the same system (since they install headers files with the same names.)
- If problems arise, try reversing the argument order:

```
cc program.c -lodbc
```

- Currently, use `odbc` on the Debian 7 Linux systems of the department.
- The actual structure of C programs which contain ODBC calls is illustrated via accompanying example programs, with some basic principles discussed in these slides.

ODBC Drivers and C

- As noted previously, there are two ODBC drivers for PostgreSQL:
  ANSI: The ANSI driver `psqlodbca.so`.
  Unicode: The Unicode driver `psqlodbcw.so`.
- For PostgreSQL under Debian 7 Linux. which uses `SQL_ANSI` server encoding, use the ANSI ODBC driver `psqlodbca.so`.
- For installations which use UTF8 server encoding (*e.g.*, newer versions of Ubuntu), `psqlodbcw.so` may be the driver which works.
- For MySQL, the driver `libmyodbc.so` should work for both encodings.
- If in doubt, try the alternatives and see which one works.
- Keep in mind that a simple connect may work with an incorrect driver while more complex connections involving queries will require the correct one.

Some Basics of ODBC Calls in C

Identifiers:

- Most ODBC identifiers begin with `SQL` (uppercase letters).
- It is therefore good practice to avoid using this sequence of characters as the beginning of user-defined identifiers.

API calls:

- ODBC contains a large number of functions (around 80).
- They have names such as `SQLAllocHandle` and `SQLCloseCursor`.
- Only a few will be used in this course.
- Most (all?) return a value of type `SQLReturn`.
- The returned value is zero if the the execution was normal, and nonzero if it was not.

Includes: To use ODBC API calls, the following two includes must be in the program header:

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

Matching SQL to Host-Language Data Structures in C

- SQL and the host language (in this case `C`) each have their own data types.
- To use ODBC, there must be a mechanism for translation between these types.
- To effect this, the primitive types which occur in SQL are assigned corresponding types in the host language.
- The definitions are found in the header file `sqltypes.h`, which is loaded by `sql.h`.
- A table of some of the principal associations is shown on the next slide.
- For API calls, these types, rather than the associated types of `C`, should be used.

The Principal ODBC-C Data-Type Associations

- Some of the most commonly used associations are shown below.

ODBC Type	C Type
SQLCHAR	char
SQLSCHAR	signed char
SQLINTEGER	long int
SQLUINTEGER	unsigned long int
SQLSMALLINT	short int
SQLUSMALLINT	unsigned short int
SQLREAL	float
SQLDOUBLE,SQLFLOAT	double
SQLDATE	a large struct

- There are many others (*e.g.*, for time).
- The exact associations may vary from system to system.
  - They may also vary between 32- and 64-bit versions of the same system.
- Therefore, for API calls, the ODBC types, rather than the C types, should be used.

Integer Encodings of ODBC Data Types in C

- Each ODBC type has an integer encoding.
- These encodings are used in the arguments to API calls, to indicate which data type is to be used.
- Each encoding also has a symbolic name, so the programmer need not know (and should not use) the actual integer.
- The associations of numbers to symbolic names are found in `sqlext.h`.
- A table of some of the most common ones is shown below.

ODBC Type	Name for Integer Encoding
SQLCHAR	SQL_C_CHAR
SQLSCHAR	SQL_C_STINYINT
SQLINTEGER	SQL_C_SLONG
SQLUINTEGER	SQL_C_ULONG
SQLSMALLINT	SQL_C_SSHORT
SQLUSMALLINT	SQL_C_USHORT
SQLREAL	SQL_C_FLOAT
SQLDOUBLE,SQLFLOAT	SQL_C_DOUBLE
SQLDATE	SQL_C_TYPE_DATE

- These need not be remembered; only the concept is important.

Integer Encodings for SQL Data Types in C

- There is also an integer encoding (and an associated symbolic identifier) for each SQL type.
- These associations are found in sqlext.h.
- The table below gives only typical associations, found in the on-line documentation for ODBC.

SQL Type	Name for Integer Encoding
char(n)	SQL_CHAR
varchar(n)	SQL_VARCHAR
smallint	SQL_SMALLINT
integer	SQL_INTEGER
real	SQL_REAL
numeric(p,s)	SQL_DECIMAL
decimal(p,s)	SQL_NUMERIC
date	SQL_TYPE_DATE

- Consult the local documentation for exact usage.

Declaration and Use of ODBC Handles in C

- Handles are declared using the type `SQLHandle`.
- Handles are allocated using the function `SQLAllocHandle`.
- Handles are freed using the function `SQLFreeHandle`.
- These are all illustrated in the example programs.
- The current version of ODBC is 3.xx.
- The (much) older version 2.xx used a different syntax for handles.
- That syntax is deprecated and should not be used, even though it may still work.

Commit Mode

- The default commit mode is *autocommit*.
- This means that updates are committed automatically.
- The commit mode may be set to *manual* when the ODBC connection is opened.
- In that case, the `SQLEndTrans()` function must be called to commit the results.

Other Key ODBC Directives in C

- The following are just indicators of the key operations in ODBC.
- All are illustrated in the example programs.

`SQLPrepare` Prepare ("compile") an SQL statement for execution.

`SQLExecute` Executed a compiled SQL statement.

`SQLExecDirect` Compile and then execute an SQL statement in one step.

- Appropriate in situations in which an SQL statement is to be executed only once.

`SQLBindParameter` Bind an input-parameter index in an SQL statement to a variable in the program.

`SQLBindCol` Bind a column of a query result (output parameter) to a variable in the program.

`SQLFetchTuple` Fetch the next tuple from the result of a query.

`SQLCloseCursor` Close the cursor on a given query, so that the statement handle may be used to collect the results of a new query.

Other Classes of API Calls for ODBC in C

- A few other major classes of ODBC API calls are the following.

  Catalog queries: Find out which relations are in a given database, what the types of the columns are, what the constraints are, and so forth.

  Optimization directives: Process large sets of queries with efficient batch operations.

  Error management: If something goes wrong, find out what the problem is.

- In all, there are over 80 API calls in ODBC.