Database Connectivity using C/C++

SQL (Structured Query Language) is a fourth-generation language (4GL) that is used to define, manipulate, and control an RDBMS (relational database management system).

Before starting the main article, let us get familiar with the used tools.

1. **Compiler:** Code::Blocks IDE with MinGW compiler

**Download Link:** [Binary Download](http://www.codeblocks.org/downloads/5)  
Code::Blocks is a cross compiler (It can run on any platform like Windows, Linux and Mac) and it is free to download. This IDE is specially designed for C and C++ and easy to use.

1. **API:** We are going to use SQLAPI++ Library

**Download Link:** [SQLAPI Download](http://www.sqlapi.com/Download/index.html)

SQLAPI++ is a C++ library (basically a set of header files) for accessing multiple SQL databases (Oracle, SQL Server, DB2, Sybase, Informix, InterBase, SQLBase, MySQL, PostgreSQL, SQLite, SQL Anywhere and ODBC). It is easy to implement and simple.

1. **OCCI:** Oracle C++ Call Interface

**Download Link:**[OCCI C++ Download](http://www.oracle.com/technetwork/database/features/oci/index-090820.html)  
OCCI is an interface defined by the database company ORACLE that defines a comfortable interfacefor the C++ programmer to access the Oracle database with classes using parameters that are reminiscent of SQL statements. The interface exists for ORACLE 9i, ORACLE 10 and comes with the Oracle.

We must download and install the above three (if we don’t have them). Now we are almost ready to start.

**Some settings before starting:**

-> Open the code::blocks IDE and go to or click on **settings** -> **compiler and debugger settings** (You will now see global compiler settings)

-> Now click on “**Linker settings**” in the linker settings click on ADD button and add the following

For **Windows OS** :

**Code:**

C:\SQLAPI\lib\libsqlapiddll.a

C:\Program Files\CodeBlocks\MinGW\lib\libuser32.a

C:\Program Files\CodeBlocks\MinGW\lib\libversion.a

C:\Program Files\CodeBlocks\MinGW\lib\liboleaut32.a

C:\Program Files\CodeBlocks\MinGW\lib\libole32.a

These will be found in your SQLAPI++ (If you have not extracted in C: drive then select the appropriate location and add the mentioned files to linker settings).

The above code is used to add library files to connect C/C++ program with SQLAPI.

Basically, there are 2 steps:

* 1. **Connecting to database (and error handling)**  
     **Code:**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C++ pgroram for connecting to database (and error handling)  #include<stdio.h>  #include<SQLAPI.h>         // main SQLAPI++ header    int main(int argc, char\* argv[])  {      // create connection object to connect to database      SAConnection con;      try      {          // connect to database          // in this example, it is Oracle,          // but can also be Sybase, Informix, DB2          // SQLServer, InterBase, SQLBase and ODBC          con.Connect ("test",    // database name                       "tester",  // user name                       "tester",  // password                       SA\_Oracle\_Client); //Oracle Client          printf("We are connected!\n");            // Disconnect is optional          // autodisconnect will occur in destructor if needed          con.Disconnect();          printf("We are disconnected!\n");      }        catch(SAException & x)      {          // SAConnection::Rollback()          // can also throw an exception          // (if a network error for example),          // we will be ready          try          {              // on error rollback changes              con.Rollback ();          }          catch(SAException &)          {          }          // print error message          printf("%s\n", (const char\*)x.ErrText());      }      return 0;  } |

**Output:**

We are Connected!

We are Disconnected!

1. **Executing a simple SQL Command**  
   Now, we will look out to execute a simple SQL query.Firstly, creating a table for the database:

create table tb1(id number, name varchar(20);

Now, establish the connection to the database then, after your con.connect; method you should use cmd.setCommandText method to pass the query to the database, it is as shown below:

con.Connect("test", "tester", "tester", SA\_Oracle\_Client);

cmd.setCommandText("create table tb1(id number, name varchar(20));”);

and now, to execute the query we have to use the following command:

cmd.Execute();

**Full Code:**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| #include<stdio.h>  #include <SQLAPI.h> // main SQLAPI++ header  int main(int argc, char\* argv[])  {      SAConnection con; // connection object to connect to database      SACommandcmd;    // create command object      try      {          // connect to database (Oracle in our example)          con.Connect("test", "tester", "tester", SA\_Oracle\_Client);            // associate a command with connection          // connection can also be specified in SACommand constructor          cmd.setConnection(&con);            // create table          cmd.setCommandText("create table tbl(id number, name varchar(20));");          cmd.Execute();            // insert value          cmd.setCommandText("Insert into tbl(id, name) values (1,”Vinay”)");          cmd.setCommandText("Insert into tbl(id, name) values (2,”Kushal”)");          cmd.setCommandText("Insert into tbl(id, name) values (3,”Saransh”)");          cmd.Execute();            // commit changes on success          con.Commit();          printf("Table created, row inserted!\n");      }        catch(SAException &x)      {          // SAConnection::Rollback()          // can also throw an exception          // (if a network error for example),          // we will be ready          try          {              // on error rollback changes              con.Rollback();          }          catch(SAException &)          {          }          // print error message            printf("%s\n", (const char\*)x.ErrText());      }      return 0;  } |

As we know, Oracle is not auto committed (committing is making permanent reflection of data in the database) so, we have to commit it.

con.Commit();

and similarly we can roll back the transactions when an exception occurs, so to do that we use:

con.Rollback();

For deleting a row, we use this command.

cmd.setCommandText("delete from tb1 where id= 2");

Thus, by the end of this article, we have learned how to connect your C/C++ program to database and perform manipulations