TÉCNICO LISBOA

Programação por Objetos e Bases de Dados

Object-Oriented Programming and Data Base

Alexandra Moutinho

Instituto Superior Técnico, Universidade de Lisboa
Dep. of Mechanical Engineering, Center of Intelligent Systems, IDMEC/LAETA
Pav. Mecânica III, 1049-001 Lisbon, Portugal
alexandra.moutinho@tecnico.ulisboa.pt

MySQL / C++

- *Create a MySQL database using IST resources.*
- *Overview how to connect a MFC Application in C++ with this database.*

INSTITUTO SUPERIOR TÉCNICO

IST Database

- IST supplies all students with a MySQL database.
- The following steps will explain in detail how to use this database.
- By using this database and not a local host database you can work in the same database as your group members.
- It can also be accessed by end users of your MFC application.

1. Activate Services

a) Activate the Database Service (*shell*) at: https://ciist.ist.utl.pt/servicos/self_service/

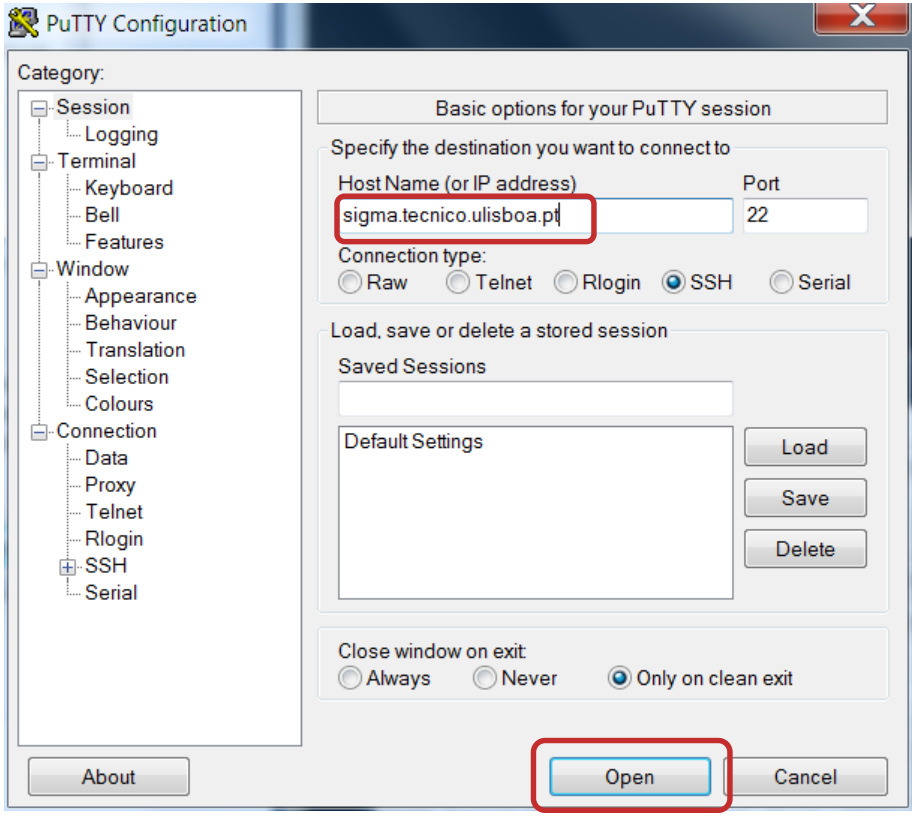
	Serviço	Estado	Acção
✓	wifi	Activo	Como aceder?
✓	proxy	Activo	Como aceder?
✓	afs	Activo	Como aceder?
✓	shell	Activo	Como aceder?
✓	web	Activo	Como aceder?
✓	cgi	Activo	Como aceder?
✓	mail	Activo	Como aceder?

1. Activate Services

- Connecting to the DB requires it to be activated via a SSH tunnel. PUTTY does just that.

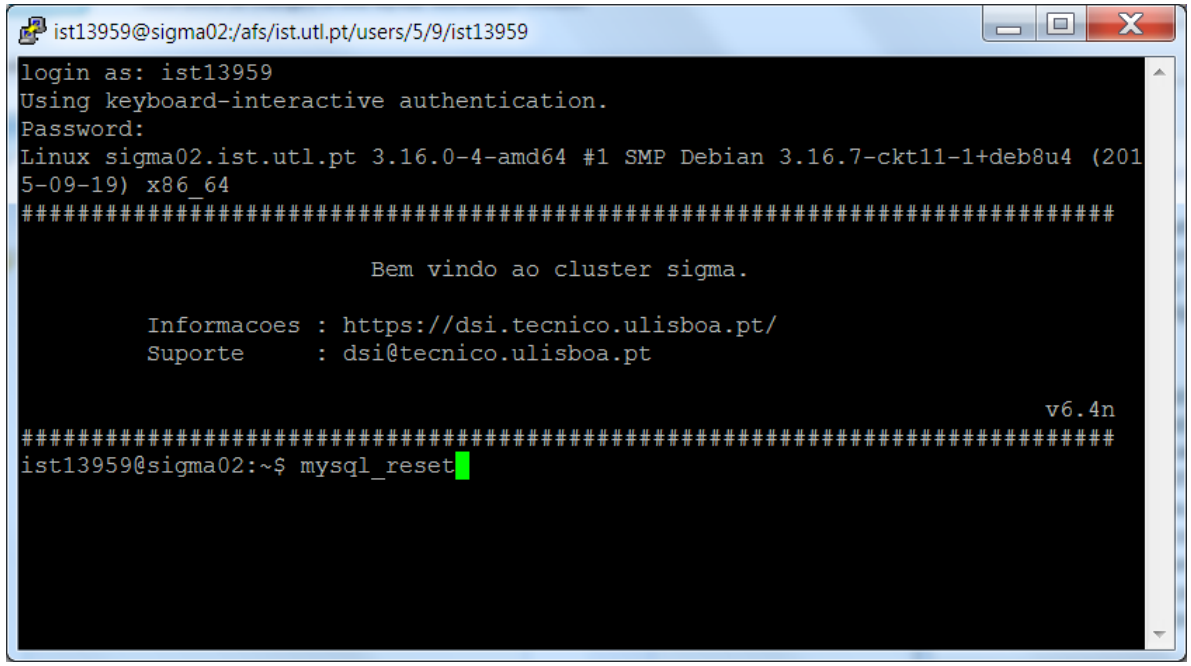
b) Download PUTTY at: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

c) Connect to sigma.tecnico.ulisboa.pt



1. Activate Services

d) Connect with your IST Username and Password and run the mysql_reset command.



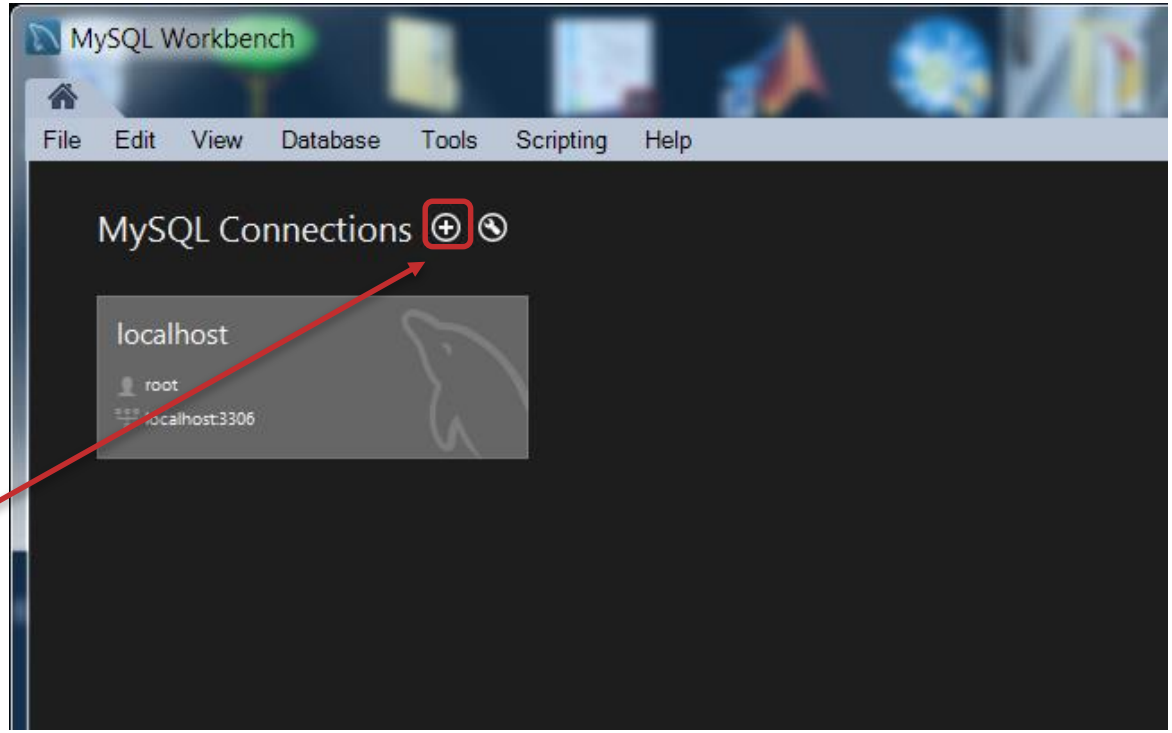
- This command creates a database with a name equal to the username, and a password displayed on screen. **Write down the provided password!**

2. Connect to MySQL Database

- The DB server runs in the machine db.tecnico.ulisboa.pt

a) Open MySQL WorkBench

b) Create a *New Connection*



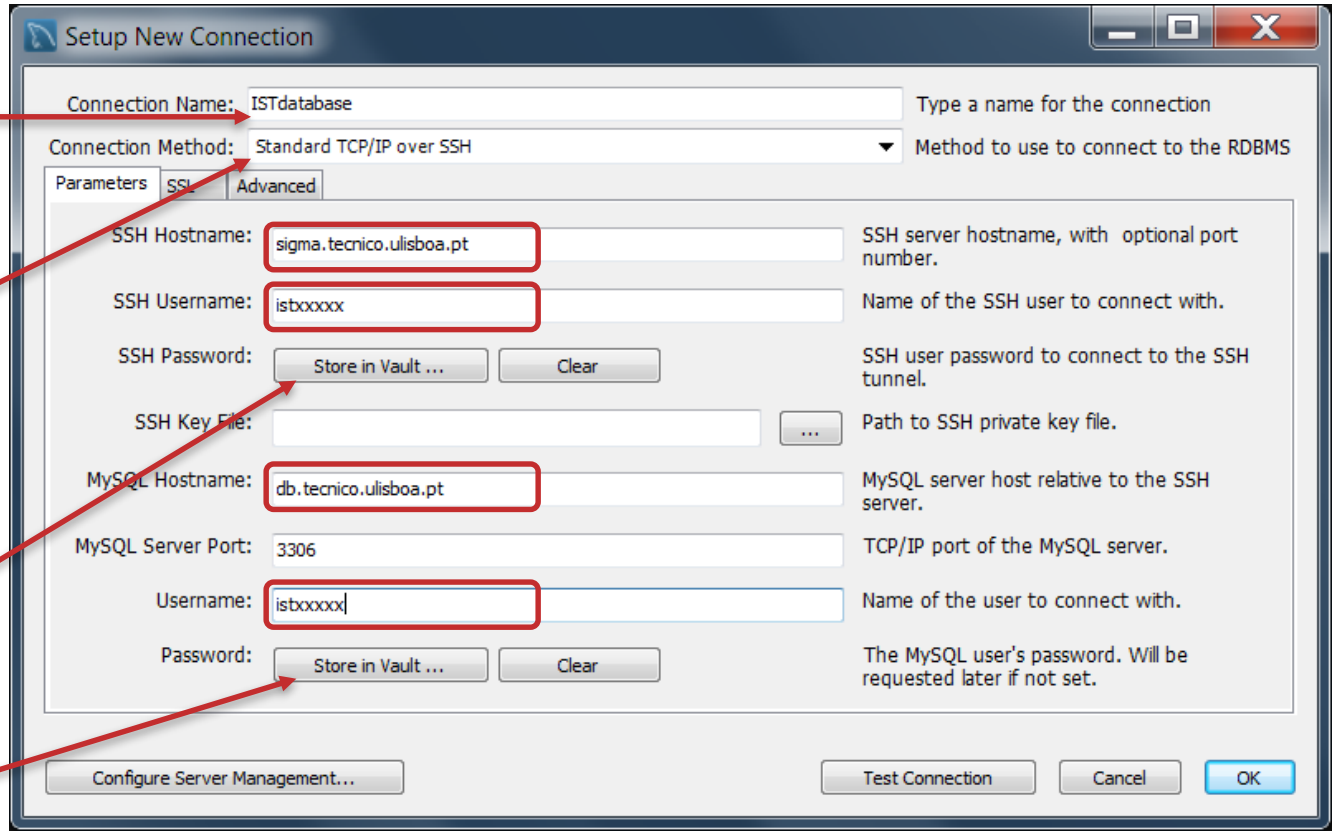
2. Connect to MySQL Database

c) Name your connection

d) Choose a Standard TCP/IP over SSH Connection Method

e) Connect using both your IST password and Putty's password.

f) Username is istxxxxx.



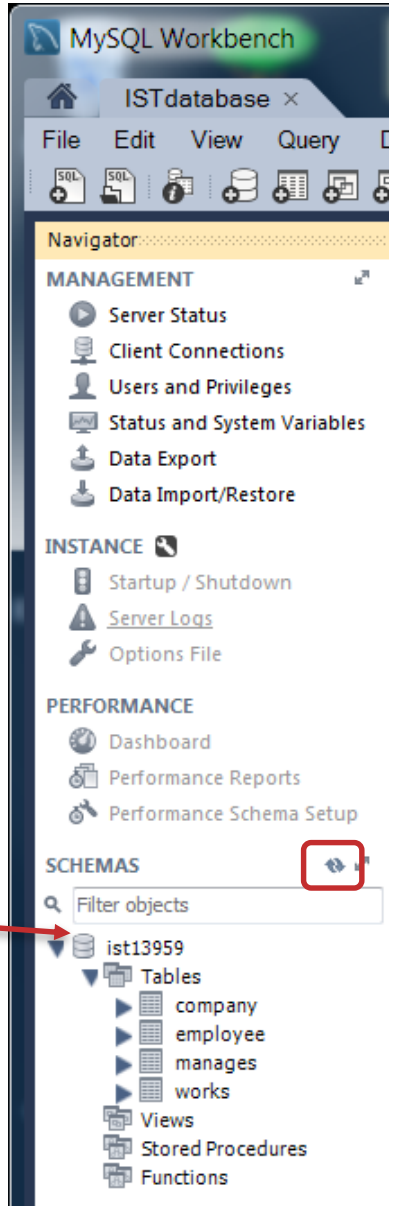
2. Connect to MySQL Database

- You should now be able to connect to your IST Database.
- Note that:
  - You only have one schema (istxxxxx schema). Make sure to add your project to that schema.
  - This database is hosted in Linux! Linux is Case Sensitive. Table "Users" and "USers" is NOT the same thing.
  - The DB is no longer in your PC. It will take some extra time to do big tasks.

3. Load a database

- Try to load the employee database, making these changes to the employee.sql script:

```
DROP DATABASE IF EXISTS
istxxxxx;
CREATE SCHEMA IF NOT EXISTS
istxxxxx;
USE istxxxxx;
-- CREATE DATABASE employee;
-- USE employee;
CREATE TABLE company (
    company_name varchar(255),
    city varchar(255),
    PRIMARY KEY (company_name)
);
```



C++ Application

- We now want to make a MFC C++ application that is connected with our database.
- A library of connector functions is supplied by MySQL to connect C++ Apps with MySQL servers.
- The official MySQL Visual Studio connector is outdated and does not work with VS2013.

4. C++ Connector

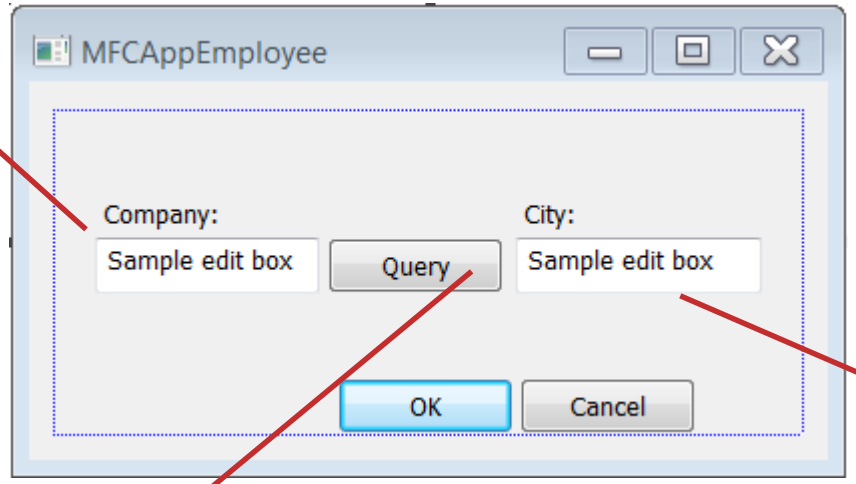
a) Get the C Connector Libraries:
   - Download Windows (x86, 32-bit), ZIP Archive from http://dev.mysql.com/downloads/connector/c/
   - Here the C libraries are used instead of the C++ libraries.

b) Extract the content of the zip file to the Visual Studio folder in My Documents, and rename it, for simplicity, to MySQLconnector (My Documents\Visual Studio20??\MySQLconnector).
   - This folder contains all the include files your program needs to be linked to, so the connector can work.
   - It also includes a compiled library your App needs to be shipped with.

5. MFC C++ Application

a) Create a Dialog based MFC Application. Compile your app at least once before continuing.

Rename IDC_EDITCompany
Add a value control variable, type Cstring, named company



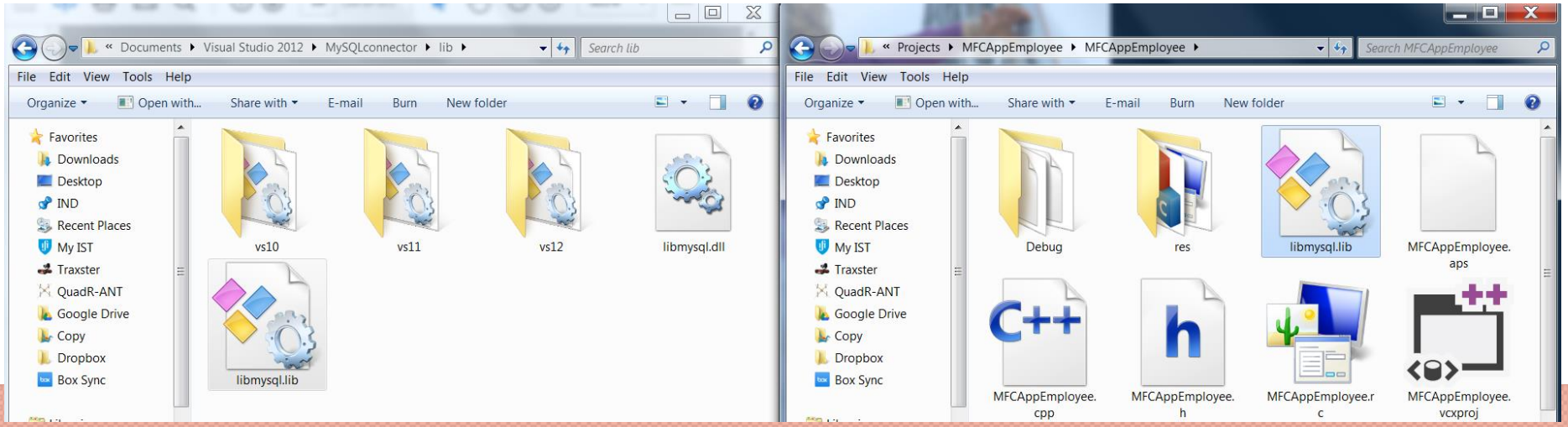
Rename IDC_EDITCity
Add a value control variable, type Cstring, named city

Rename IDC_BUTTONQuery
Add an event handler, named OnBnClickedButtonQuery()

5. MFC C++ Application

b) Copy libmysql.lib to your Project

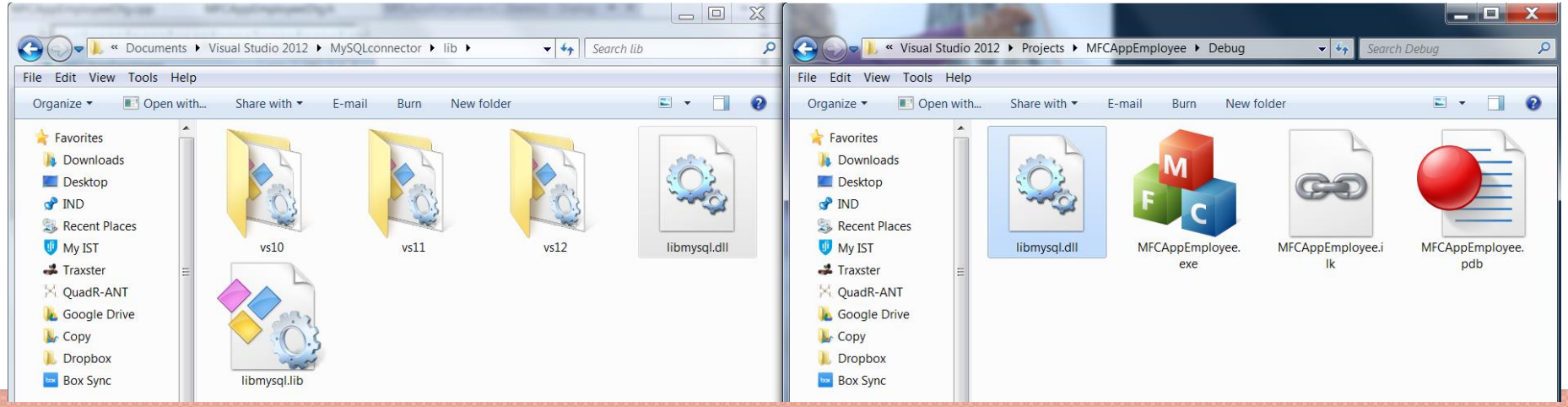
- Copy the *static* library from the MySQLconnector/lib folder to your project <u>main</u> folder Documents/Visual Studio 20??/Projects/projectname/projectname (in this example, the project is called MFCAppEmployee)



5. MFC C++ Application

c) Copy libmysql.dll to your Project

- Copy the *dynamic* library from the MySQLconnector/lib folder to your project <u>debug</u> folder Documents/Visual Studio 20??/Projects/projectname/Debug (in this example, the project is called MFCAppEmployee)



5. MFC C++ Application

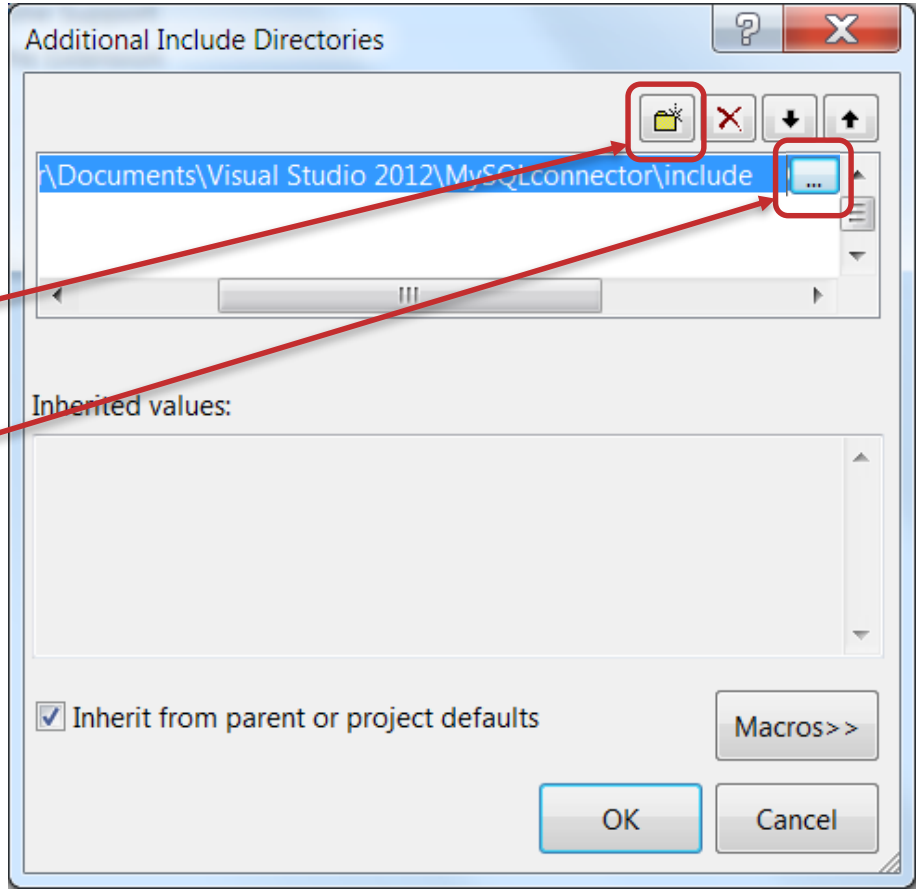
d) Link header files:

i. Right-click your project in the Solution Explorer and choose Properties

ii. Under Configuration Properties → C/C++, edit Additional Include Directories.

iii. Browse to \Documents\Visual Studio 20??\MySQLconnector\include and click OK.

iv. Click OK twice.



5. MFC C++ Application

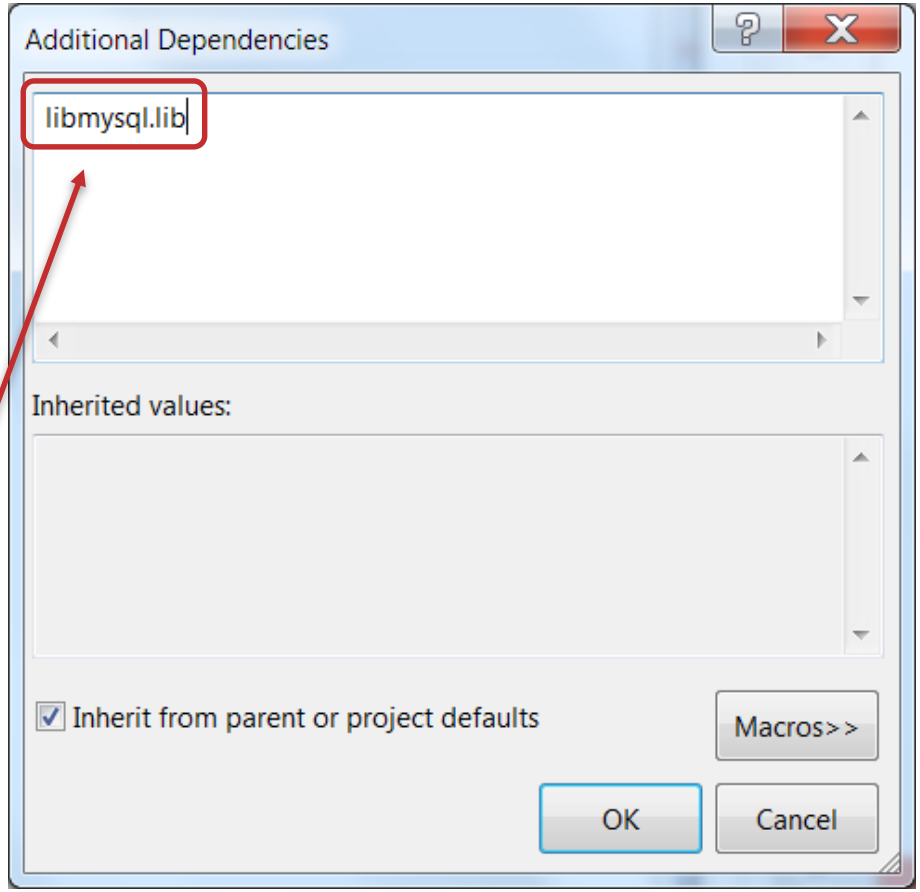
e) Add a dependency library:

i. Right-click your project in the Solution Explorer and choose Properties

ii. Under Configuration Properties → Linker → Input edit Additional Dependencies.

iii. Write libmysql.lib.

iv. Click OK twice.



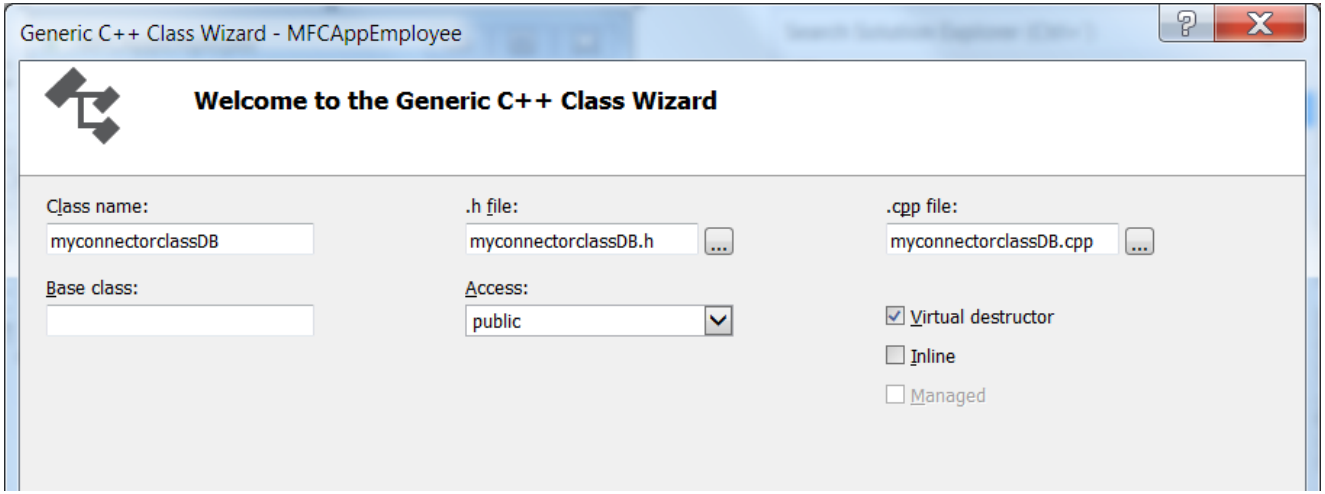
5. MFC C++ Application

- With this setup your project is ready to create the connector class.
- Remember, steps 5.b) to 5.e) have to be repeated for every project.
- Step 5.d) added the header references to the project (mysql.h, etc...).
- Step 5.e) indicated the dependency from our project to the library added in the debug folder in step 5.c).

5. MFC C++ Application

f) Right-click your Project in the Solution Explorer, and add a C++ Class

- Call this something like myconnectorclassDB. Avoid names as MySQLconnector because C++ already has similar named functions and you might run into problems.



g) Edit your myconnectorclassDB.h header file like this:

myconnectorclassDB.h

```
#pragma once
#include "my_global.h" // The included headers we need
#include "mysql.h"

class myconnectorclassDB
{
private:
  #define SERVER "db.ist.utl.pt"
  #define USER "istxxxxx" // Your IST number
  #define PASSWORD "puttypassword" // NOT IST password
  #define DATABASE "istxxxxx" // Your IST number

  void Query(CString query); // Main query function
  MYSQL *connection; // Pointer allocation for a connection.
  // This is an object creation of an existing class in
  // The mySQL libraries we added.
```

Notice that this data is defined as private: you do not want your end client to see this data. Edit it with your information.

INSTITUTO SUPERIOR TÉCNICO

myconnectorclassDB.h

g) Edit your myconnectorclassDB.h header file like this (cont.):

```
public:
    MYSQL_ROW row; // Another object based on the existing
    // library, this one stores a single row from queries
    MYSQL_RES *result; // and this one the entire result
    // from a query
    void connect(); // Connection function. Notice how
    // this is public, but the connection itself is
    // private.
    CString CPtoUnicode(const char* CPString, UINT
CodePage);
    // Converts data from MySQL format to MFC's CString.
    // My Queries
    CString CheckCity(CString company);
    myconnectorclassDB(void);
    virtual ~myconnectorclassDB(void);
};
```

You will need to create your own queries and add them to the header.

INSTITUTO SUPERIOR TÉCNICO

myconnectorclassDB.cpp

h) Add the connect function to your myconnectorclassDB.cpp source file:

```
void myconnectorclassDB::connect()
{
  connection = mysql_init(NULL); // Initialise the instance
  connection = mysql_real_connect(connection, SERVER,
USER,                                          PASSWORD,
DATABASE, 0, NULL, 0);
  // The command mysql_real_connect is included in the
libraries
  if (connection){
  // Add debug code here
  }
  else{
  // and here
  }
}
```

6. Test your connection

- Your program has now enough code to test if you can connect to your database.

a) Include this class in the project (add #include "myconnectorclassDB.h“ in your MFCAppEmplyeeDlg.cpp file).

b) In your OnBnClickedButtonQuery() function, create a new myconnectorclassDB object named MyConnection (“myconnectorclassDB MyConnection;") and call MyConnection.connect() to launch the connection.

INSTITUTO SUPERIOR TÉCNICO

6. Test your connection

d) Connection is by default null. You can debug by calling "MyConnection.connection == NULL". If this statement is false, you are now connected!

- Change your myconnectorclassDB::connect(void) function to provide the apropriate message depending on the success or failure of the connection.

```
CString message;
    ...
    if (connection==NULL){
        message.Format(_T("Unable to connect!"));
        AfxMessageBox(message);
    }
    else{
        message.Format(_T("Connection successful!"));
        AfxMessageBox(message);
    }
```

6. Test your connection

- If you run into linkage problems:
  - Check you downloaded the 32-bits version of the C Connector Libraries as described in step 4;
  - Check you copied the correct files to the directories indicated in steps 5.b) and 5.c);
  - Check you executed correctly steps 5.d) and 5.e).

7. Create a query

- Let us create a simple query (CheckCity):
  - What is the city of a given company?
- The query will send a company name (CString), and return the city (CString) from the sent company.
- These variables have already been defined and added to the respective edit boxes in step 5.a).

a) Add the code of your CString myconnectorclassDB::CheckCity(CString company) function in the next slide to your myconnectorclassDB.cpp

INSTITUTO SUPERIOR TÉCNICO

7. Create a query

```
CString myconnectorclassDB::CheckCity(CString company)
{
   CString value; // Create the object to receive the answer to the query
   CString query = _T("SELECT company.city FROM company WHERE
   company.company_name = '") + company + _T("'");
   //Create a query by combining CStrings, including an input CString
   Query(query); // Pass the query. The result will be stored in the result                    // object.
   while ((row = mysql_fetch_row(result)) != NULL)
   // Method to fetch rows from result
   {
      value = CPtoUnicode(row[0], 1251);
   }
   return value;
}
```

7. Create a query

- mysql_fetch_row(result) will read a line from the result table.
- row[0] corresponds to the first entry from the row. In this operator the first entry is always numbered 0.
- CPtoUnicode(. ,1251) is another translator function, this time from MySQL format to CString.
- For this to work we will first need to define the required Query and CPtoUnicode functions

b) Copy the code provided in the next slide relative to the Query function to your myconnectorclassDB.cpp file

7. Create a query

```
void myconnectorclassDB::Query(CString query)
{
    wchar_t *p = query.GetBuffer();
    char bufUTF8[MAX_PATH];
    WideCharToMultiByte(CP_UTF8, 0, p, -1, bufUTF8,
sizeof(bufUTF8), NULL, NULL);
    /* MySQL uses a different character set from MFC's in
VS. A weird conversion has to be done. The good news is
    you only have to copy and paste this code once. */
    mysql_query(connection, bufUTF8); // Send a query
    result = mysql_store_result(connection); // Store the result
}
```

7. Create a query

- The previous code is complex and it is full of uncommon C functions and classes.
- However, you only need to implement it once. From now on, you call a query by passing a Cstring to the Query function, and your result will be stored in the result variable.

c) Copy the code provided in the next slide relative to the CPtoUnicode function to your myconnectorclassDB.cpp file

7. Create a query

```
CString myconnectorclassDB::CPtoUnicode(const char* CPString,
UINT CodePage)
{
    CString retValue;

    int len = MultiByteToWideChar(CodePage, 0, CPString, -1,
NULL, 0);
    if (len == 0) {return retValue;}
    LPWSTR buffer = retValue.GetBuffer(len);
    MultiByteToWideChar(CodePage, 0, CPString, -1, buffer, len);
    retValue.ReleaseBuffer();
    return retValue;
}
```

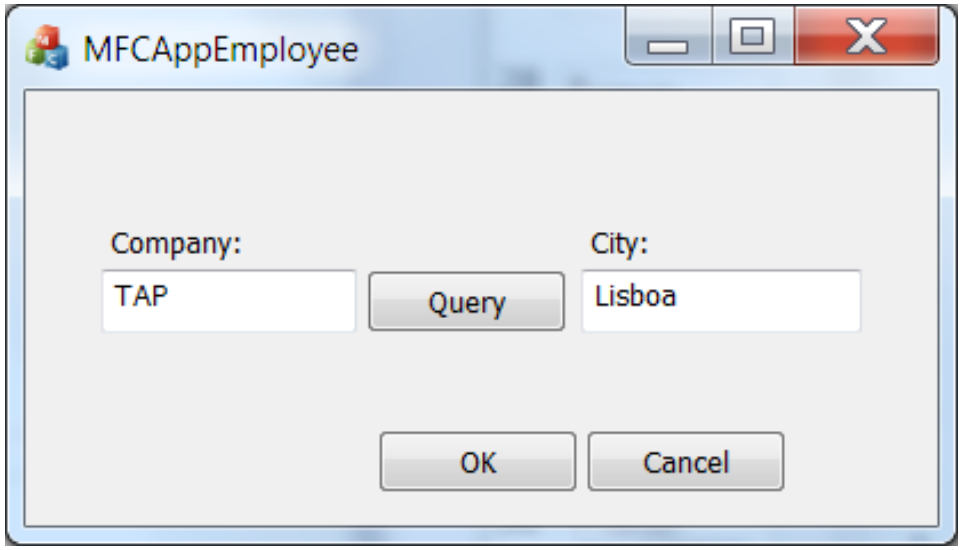
7. Create a query

d) Edit the OnBnClickedButtonQuery() event handler in your MFCAppEmployeeDlg.cpp file:

```
void
CMFCAppEmployeeDlg::OnBnClickedButtonQuery()
{
    myconnectorclassDB MyConnection;
    MyConnection.connect();
    UpdateData(TRUE);
    city = MyConnection.CheckCity(company);
    UpdateData(FALSE);
}
```

8. Run your application

- Your application should run like this:



- Make the appropriate changes so you will not be bothered with the "Connection successful!" message every time you run the query.

Final Notes

- You are now able to create a query that gets information from MySQL Database located at IST
- A series of operators (connect, query, ...) were created so that you do not have to repeat these lines of code for every query.
- There are a LOT more functions available to communicate with MySQL database, but the most important ones, and the ones vital for the project were discussed in this presentation.

Extra Material and Sources

- This procedure was based on the tutorial available at: http://www.nitecon.com/tutorials-articles/develop/cpp/c-mysql-beginner-tutorial/. In this link you will find a simple script to connect to a database using a WIN32 App.
- You can learn more about the IST Database at: http://dsi.tecnico.ulisboa.pt/servicos/servidores-e-dados/bases-de-dados/.
- A Full list of all the functions available from the MySQL connector is available at: http://dev.mysql.com/doc/refman/5.1/en/dynindex-cfunction.html.
- This tutorial was based on the project final presentation of Pedro Sá da Costa and Eduardo Poço.