Versions: Unity 3.0.0f5; MySQL 5.2.28

Author: Jonathan Wood

Alias: Zumwalt

Date: 10/8/2010

Updated: 10/12/2010 to include JS code

Document Version 1.0.1

Unity and MySQL

Table of Contents

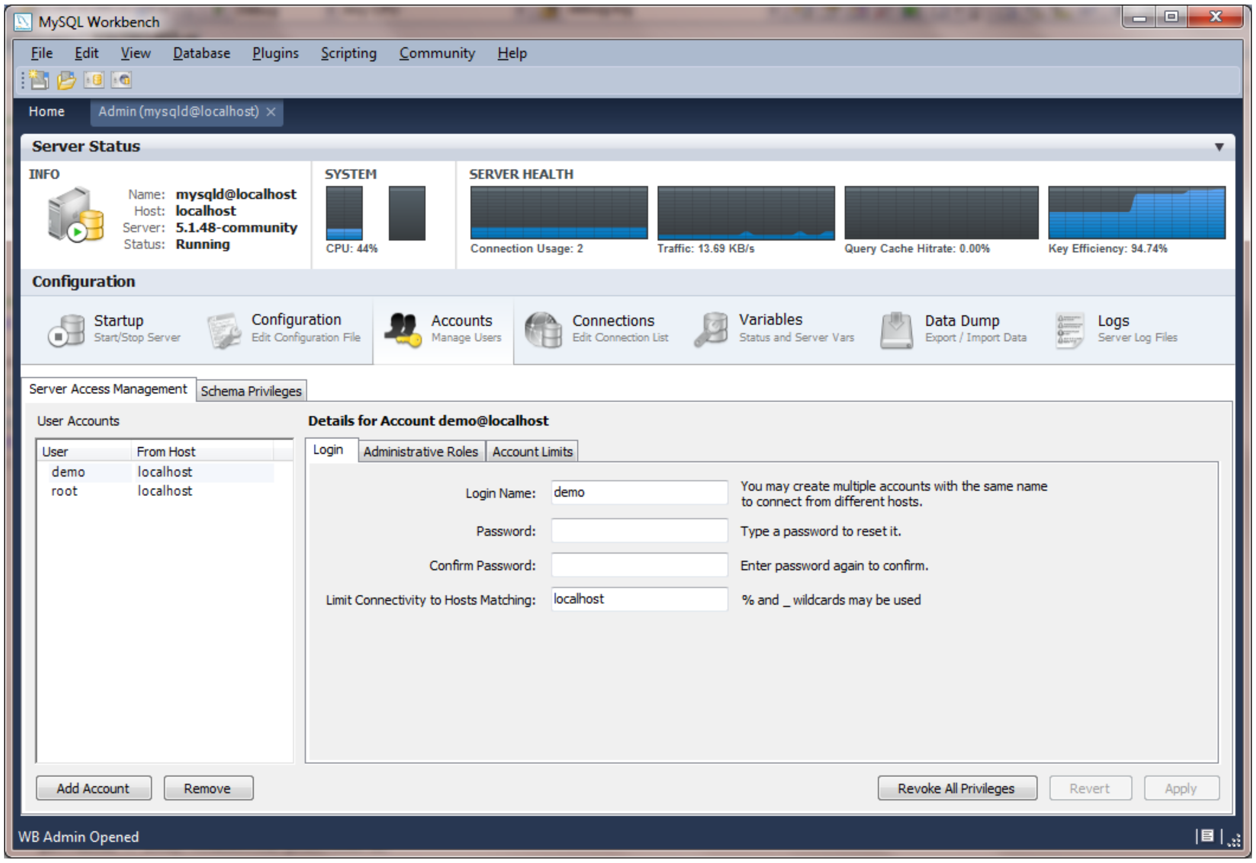
Needed Files

Community Server and Workbench which can be obtained from:

http://dev.mysql.com/downloads/

Server Configuration

Open the server and add a demo user:



Name: mysqld@localhost
Host: localhost
Server: 5.1.48-community
Status: Running

CPU: 44%

Connection Usage: 2 Traffic: 13.69 KB/s Query Cache Hitrate: 0.00% Key Efficiency: 94.74%

User	From Host
demo	localhost
root	localhost

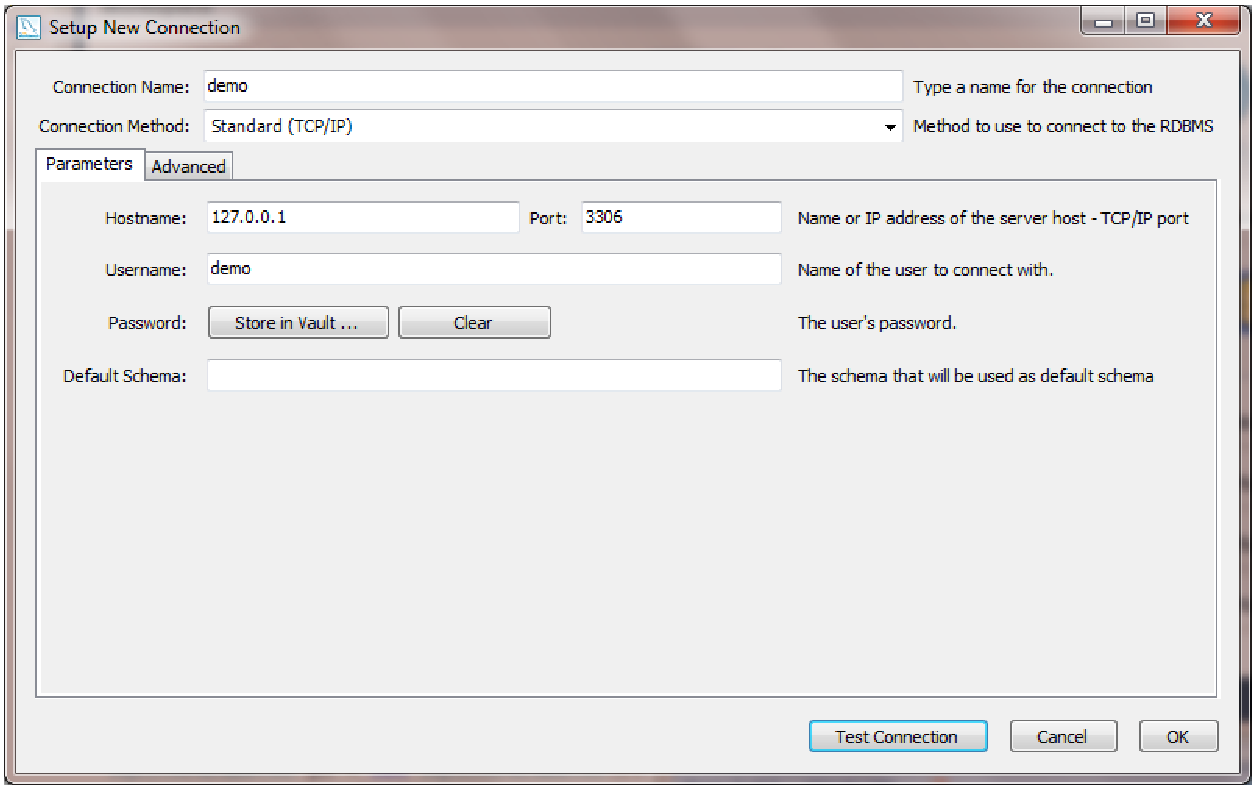
Details for Account demo@localhost

Login Name:	demo	You may create multiple accounts with the same name to connect from different hosts.
Password:		Type a password to reset it.
Confirm Password:		Enter password again to confirm.
Limit Connectivity to Hosts Matching:	localhost	% and _ wildcards may be used

Simply make the login **demo**, the password **demo** and set the connection to localhost and click on the Apply button. Now you have a demo user in the default instance. If you have no default instance, just create one from the workbench on the localhost instance that is running, information on how to do this is within the MySQL documentation.

Setting up the connection

On the “Home” tab in MySQL Workbench, click on the “New Connection in the lower left, name the connection demo, change the user to the demo user:



Setup New Connection

Connection Name: demo — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

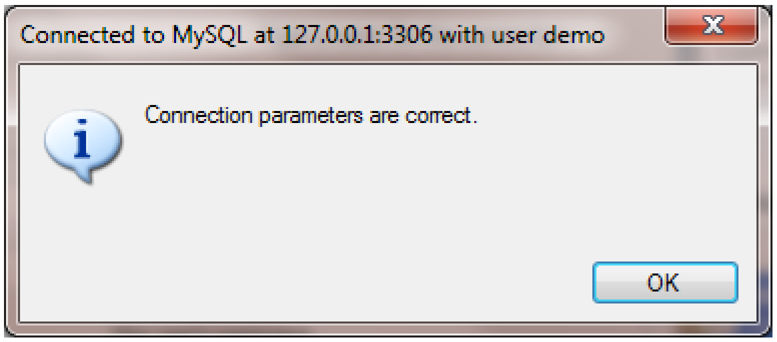
Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host - TCP/IP port

Username: demo — Name of the user to connect with.

Password: — The user's password.

Default Schema: — The schema that will be used as default schema

We have not setup a schema yet so we will do that next, but first click on the test connection and enter the demo users password and click on "OK", if you have setup the demo user correctly, you should see that the connection was correct, the window will be as follows

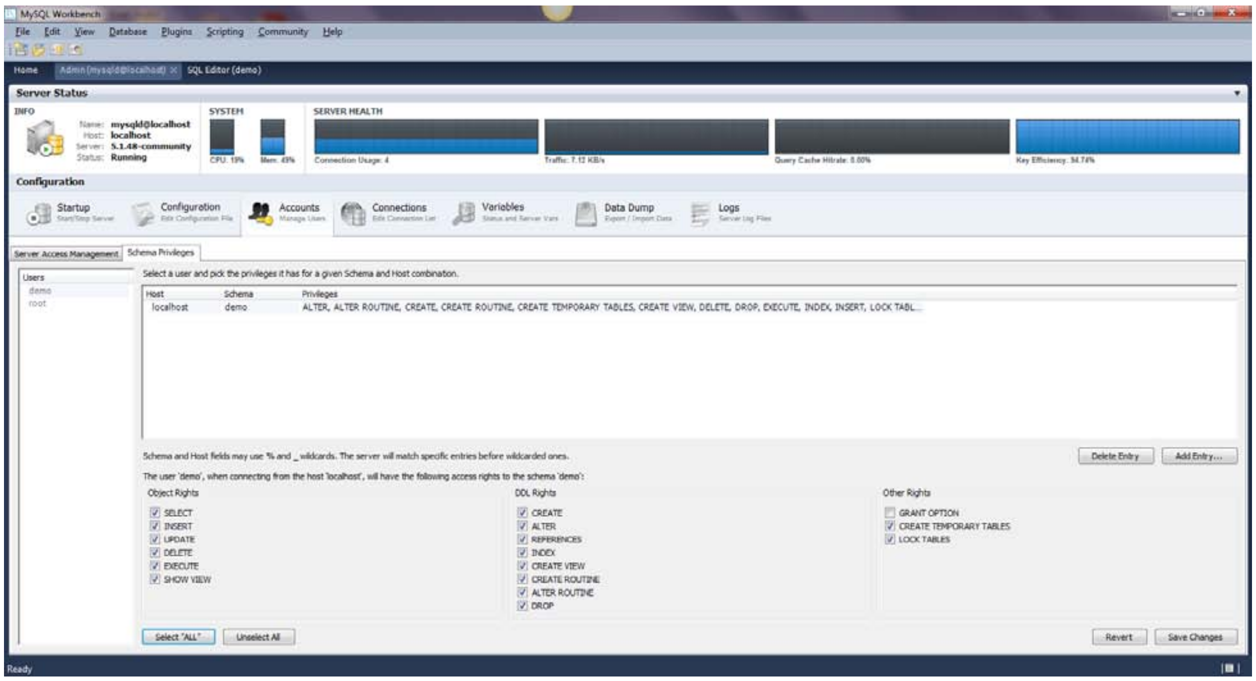


Connected to MySQL at 127.0.0.1:3306 with user demo

Connection parameters are correct.

User Permissions

We now have one problem, the demo user is not allowed to create schemas or have any other permission so we need to set those for that demo user. This is simple to set, click back on our Admin tab as long as it still is on your workbench next to the Home tab, if not then you have to double click on the database instance under Server Administration in the Home tab. You will see the list of users when you click on the Accounts tab. Click on the **demo** user and then you want to click on the **Add Entry** button. At this point, the demo schema might not show up in the list, but that is ok, simply change the Host selection to **Selected host** and choose **localhost** then in the Schema block, click on the radial next to **Schemas matching pattern or name** and type in **demo** then click on **OK**. Now back on the Schema Privileges tab, you will see that demo has appeared and the options at the bottom are available to choose permission sets. For this demo, you can simply click on the **Select All*** button, you will want to review the manual on user rights to understand what each permission does, once you have selected all permissions via that button, simply click on the **Save Changes** button.



Name: mysqld@localhost
Host: localhost
Server: 5.1.48-community
Status: Running

CPU: 19% Mem: 49%

Connection Usage: 4 Traffic: 7.12 KB/s Query Cache Hitrate: 0.00% Key Efficiency: 94.74%

Select a user and pick the privileges it has for a given Schema and Host combination.

Users: demo, root

Host	Schema	Privileges
localhost	demo	ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EXECUTE, INDEX, INSERT, LOCK TABL...

Schema and Host fields may use % and _ wildcards. The server will match specific entries before wildcarded ones.

The user 'demo', when connecting from the host 'localhost', will have the following access rights to the schema 'demo':

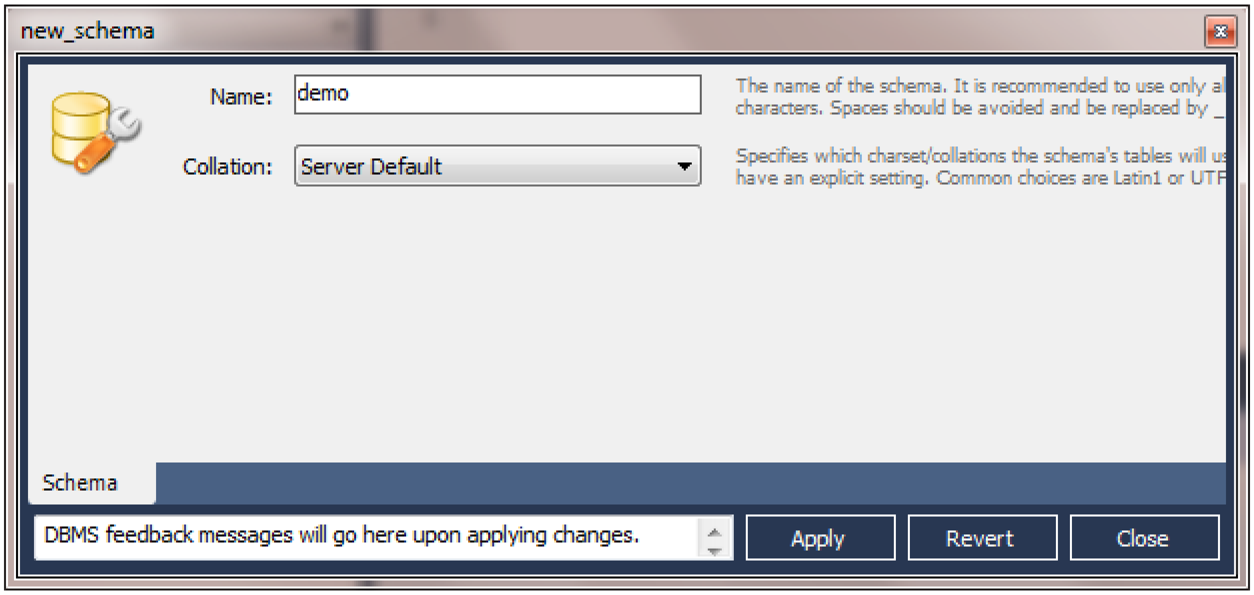
Object Rights	DDL Rights	Other Rights
✓ SELECT	✓ CREATE	☐ GRANT OPTION
✓ INSERT	✓ ALTER	✓ CREATE TEMPORARY TABLES
✓ UPDATE	✓ REFERENCES	✓ LOCK TABLES
✓ DELETE	✓ INDEX	
✓ EXECUTE	✓ CREATE VIEW	
✓ SHOW VIEW	✓ CREATE ROUTINE	
	✓ ALTER ROUTINE	
	✓ DROP	

Our demo user now has the rights to create schemas in the demo database.

Database Editor

Now that we have the demo database setup, double click on the database which will open up our SQL Editor for the Demo database. At the moment we have no schemas, so right click in the area on the left where the Object Browser window is, then left click on "Create Schema".

Give the name of the schema **demo** and click on Apply



Name: demo

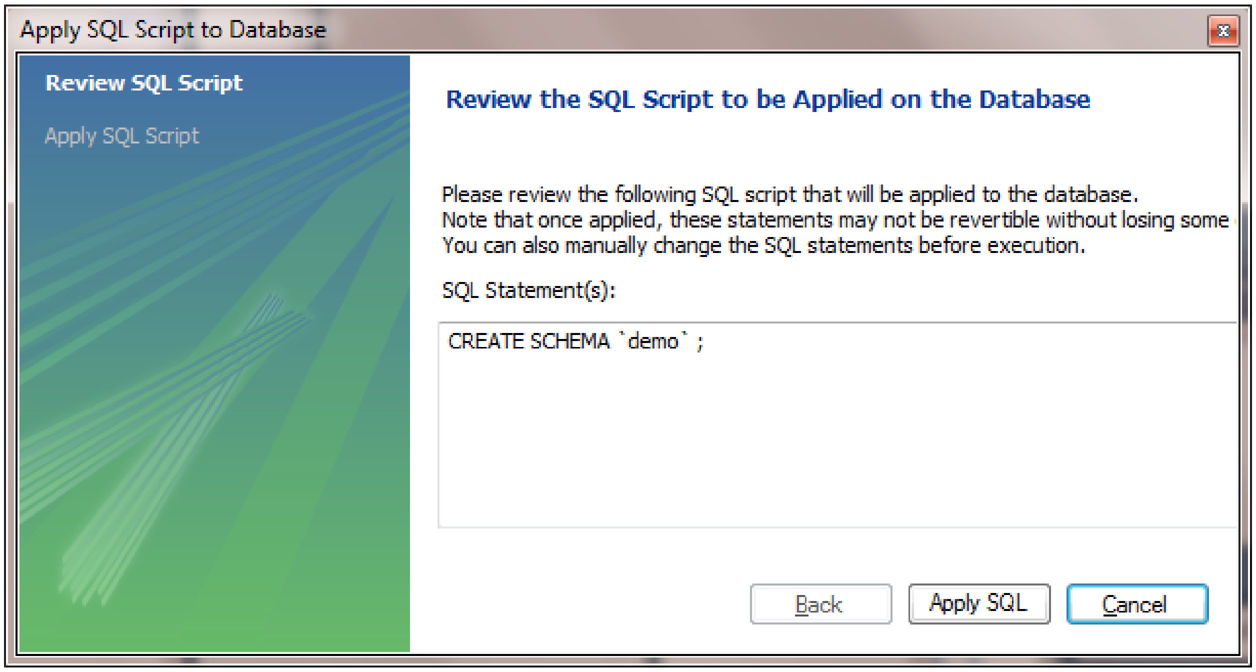
Collation: Server Default

The name of the schema. It is recommended to use only al
characters. Spaces should be avoided and be replaced by _

Specifies which charset/collations the schema's tables will us
have an explicit setting. Common choices are Latin1 or UTF

DBMS feedback messages will go here upon applying changes.

A script window will appear, simply click on Apply SQL



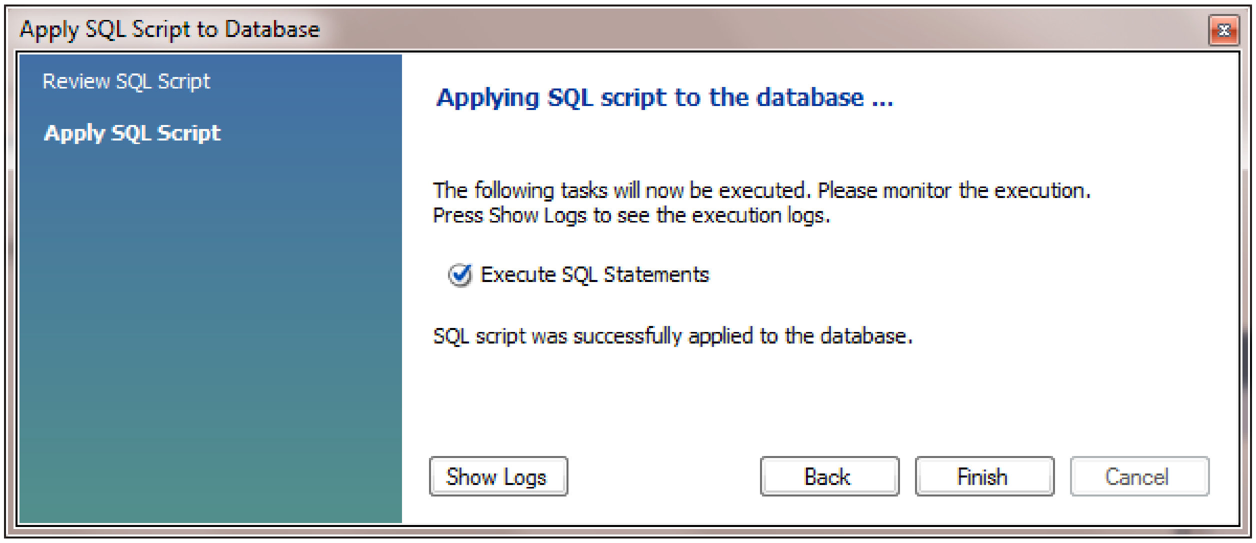
Review the SQL Script to be Applied on the Database

Please review the following SQL script that will be applied to the database.
Note that once applied, these statements may not be revertible without losing some
You can also manually change the SQL statements before execution.

SQL Statement(s):

```
CREATE SCHEMA `demo` ;
```

So far so good as long as you see that the script was executed successfully by the following example


Applying SQL script to the database ...

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Execute SQL Statements

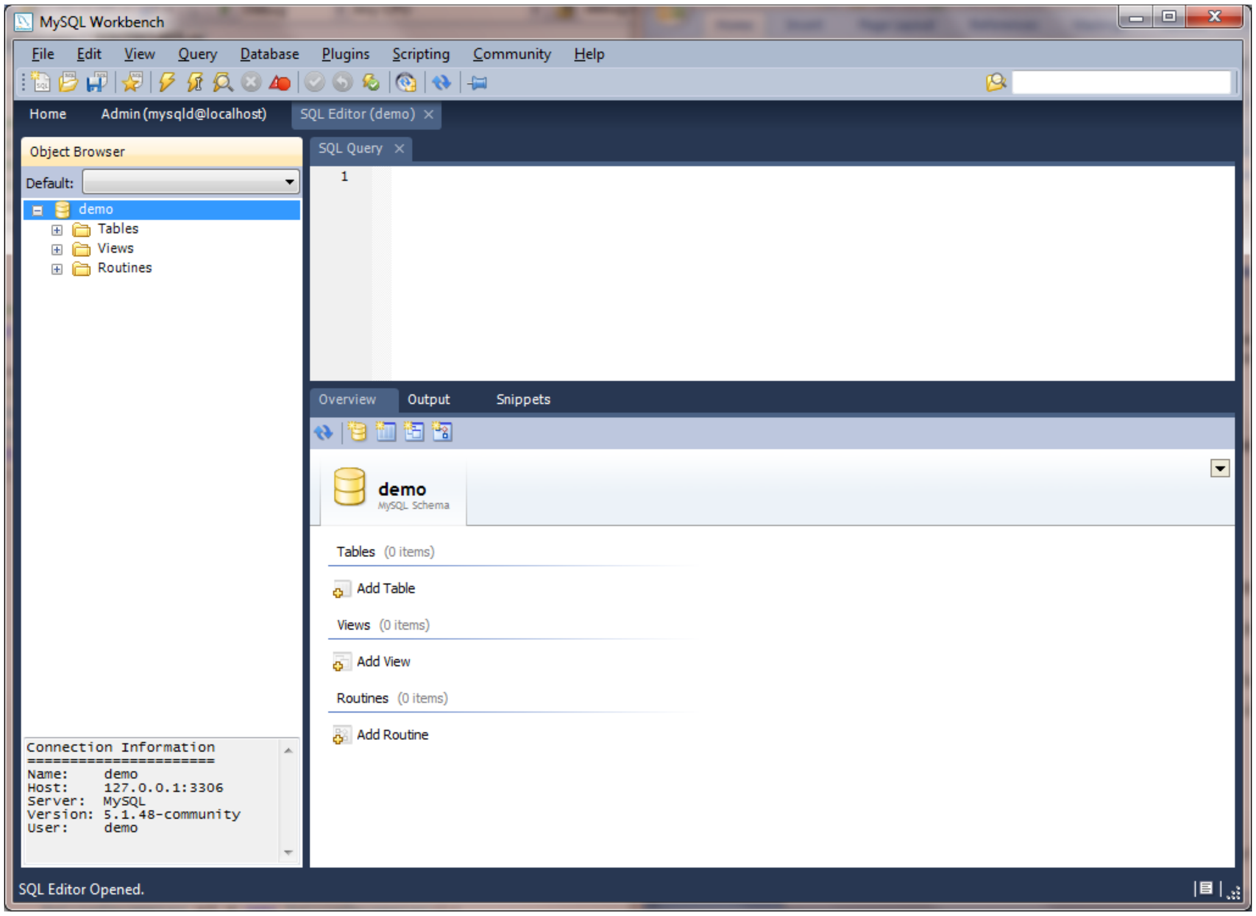
SQL script was successfully applied to the database.

Simply click on **Finish** at this point and close out of the new schema window if it is still open.

You now have your first database and you are ready to add a table that will be used for the Unity demo, we are going to deal with only a single table and a single view in this example. So, let us begin that process.

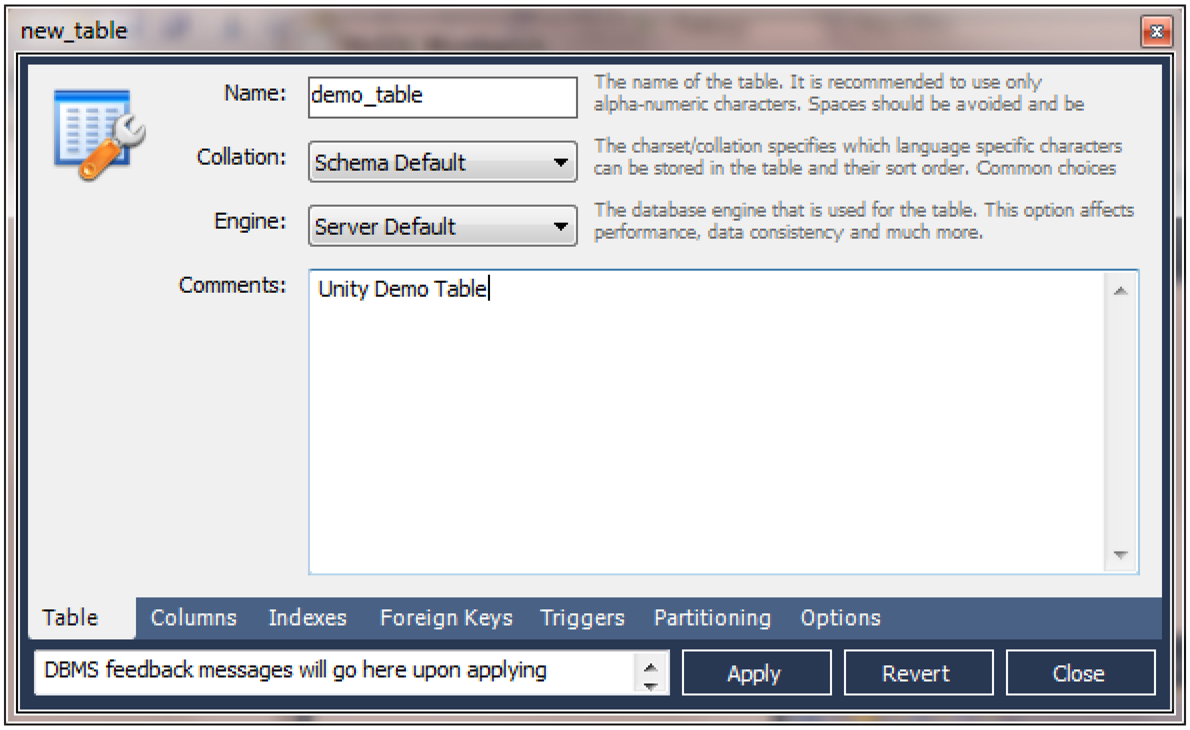
Table Creation

To start out, on our SQL Editor for the demo schema, we have the ability to add tables by simply click on the Add Table button within the demo Schema.



```
Connection Information
======================
Name:     demo
Host:     127.0.0.1:3306
Server:   MySQL
Version:  5.1.48-community
User:     demo
```

Once you double click on the **Add Table** button, a new table window will appear. We will simply keep things moving along by naming this table demo_table with giving it a Comment of Unity Demo Table on the Table tab as follows:


new_table

Name: demo_table

The name of the table. It is recommended to use only alpha-numeric characters. Spaces should be avoided and be

Collation: Schema Default

The charset/collation specifies which language specific characters can be stored in the table and their sort order. Common choices

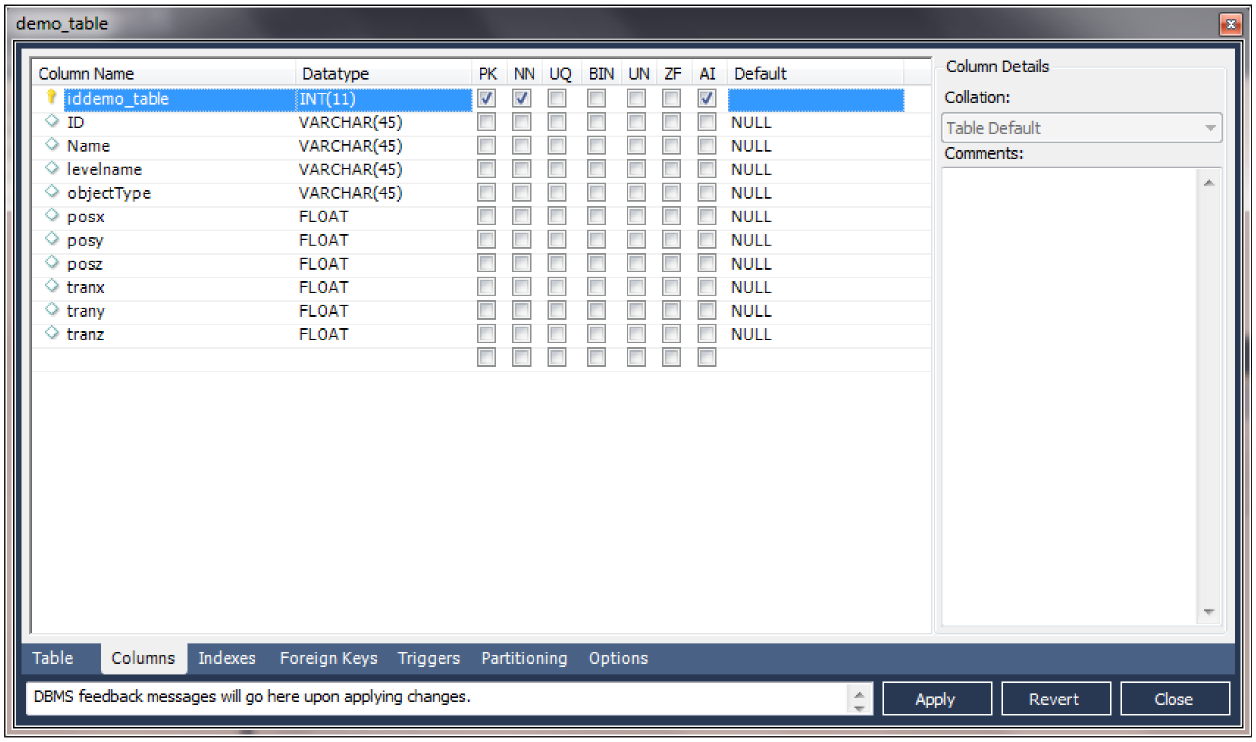
Engine: Server Default

The database engine that is used for the table. This option affects performance, data consistency and much more.

Comments: Unity Demo Table

DBMS feedback messages will go here upon applying

Moving along, we need to add some columns to our table, so click on the Columns tab along the bottom of the new_table window. We want to keep the table ID, this is important for indexing reasons and many other reasons beyond the scope of this document, so leave the first column alone. To add in a new column, double click in the cell directly beneath the iddemo_table and the field will become editable. We want to create the following columns for this demo: (see image)


demo_table

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
iddemo_table	INT(11)	✓	✓					✓	
ID	VARCHAR(45)								NULL
Name	VARCHAR(45)								NULL
levelname	VARCHAR(45)								NULL
objectType	VARCHAR(45)								NULL
posx	FLOAT								NULL
posy	FLOAT								NULL
posz	FLOAT								NULL
tranx	FLOAT								NULL
trany	FLOAT								NULL
tranz	FLOAT								NULL

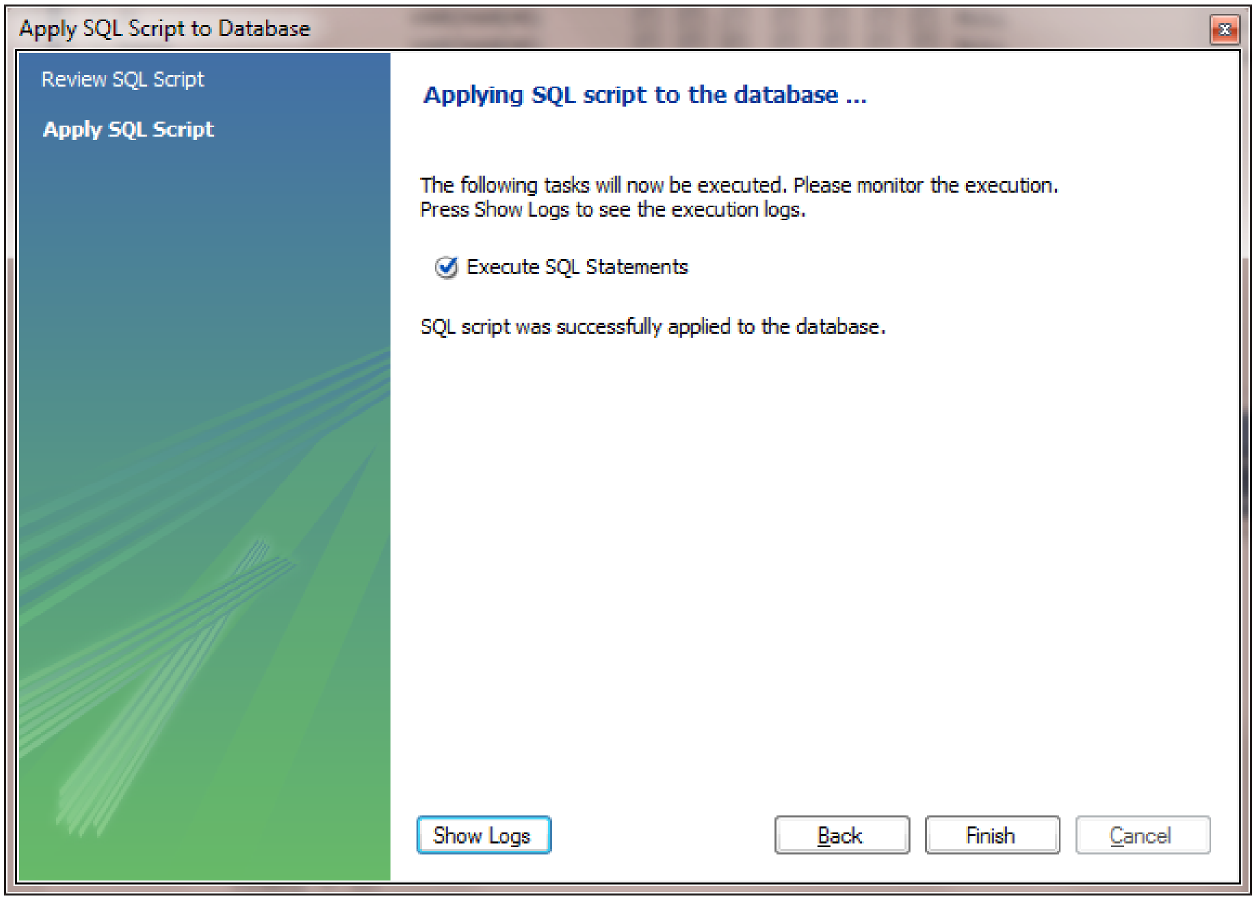
Column Details

Collation: Table Default

Comments:

DBMS feedback messages will go here upon applying changes.

This is really straight forward, NN means not null and PK means primary key, refer to the MySQL manual for each selectable type and also the Datatypes, for now we will not worry about any other options here except making sure that iddemo_table is set with AI which is auto increment. Once you have created that table, simply click on **Apply**. This will bring up a SQL Script to Database window for you to verify how the script will execute, just assume it is right as long as the fields show and click on **Apply SQL,** If all went well, you will have a success window, so simply click on **Finish**



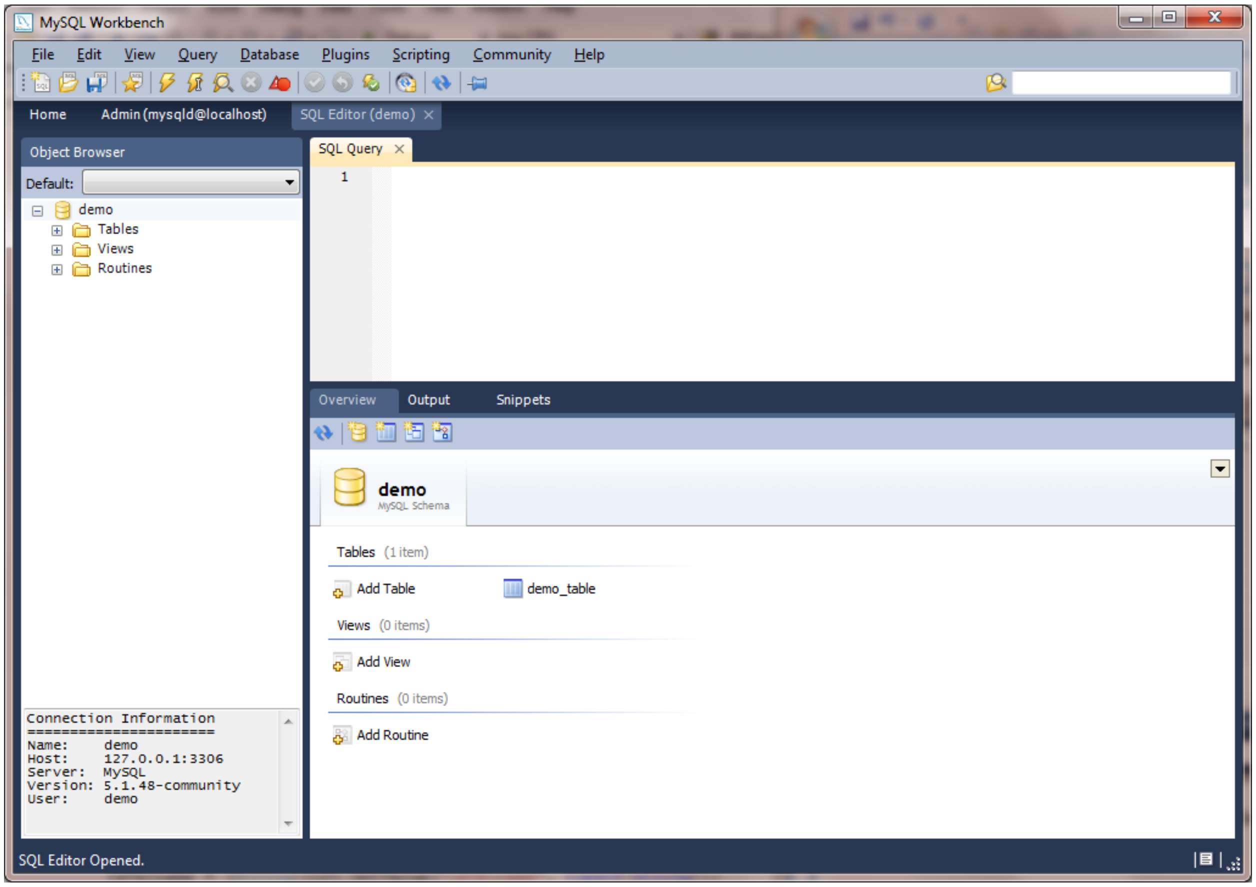
Applying SQL script to the database ...

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Execute SQL Statements

SQL script was successfully applied to the database.

Now you will be back at the new_table window where you can now simply click on the **Close** button, you should be back at the MySQL Workbench and you should see that the demo_table exists



```
Connection Information
======================
Name:     demo
Host:     127.0.0.1:3306
Server:   MySQL
Version:  5.1.48-community
User:     demo
```

View Creation

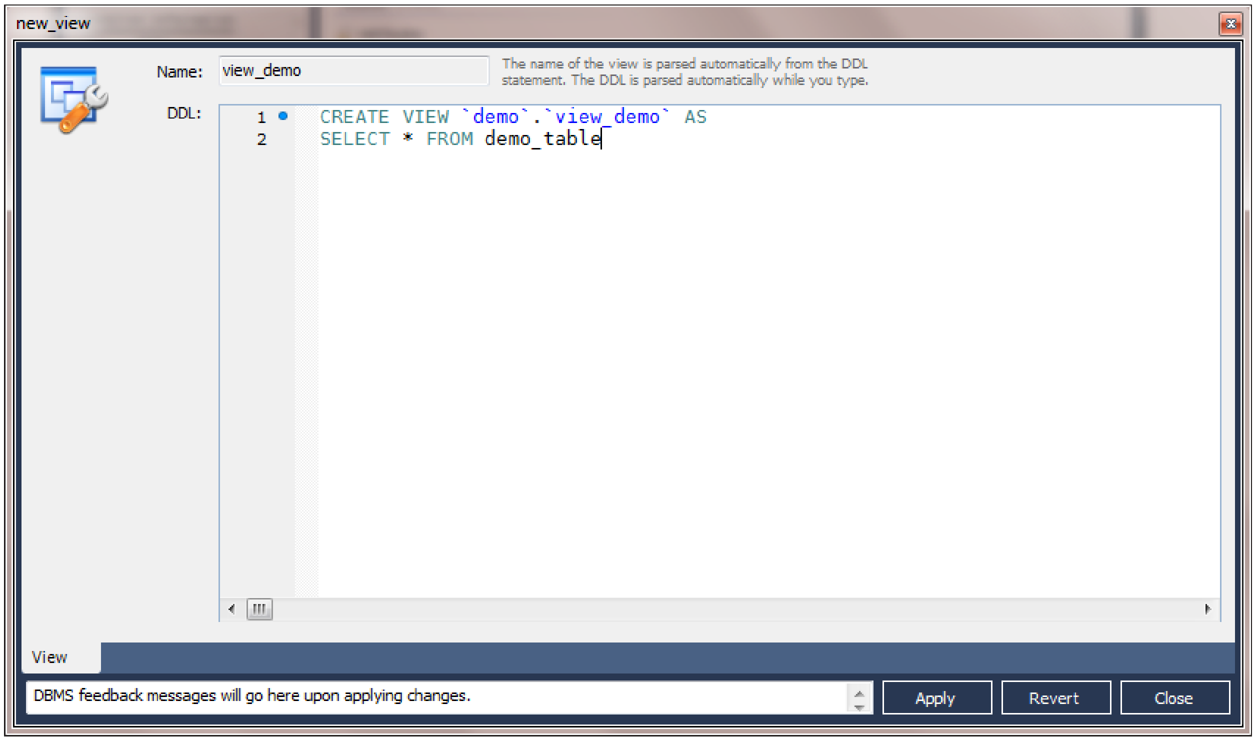
Now we have a table of demo_table and we want to create a view, although you could avoid the view and do select queries directly against the table, you want to get into practice of using views for select queries and leave the insert/update/delete commands to the table direct instead. Best to study up on why to do this through the understanding of database use which is beyond the scope of this document.

Just like the table creation, on the MySQL Workbench, there is a section that has a button for **Add View** and just like the table, you double click on the Add View to begin the creation process. This is a little trickier but to keep this simple, the query text you want is as follows:

CREATE VIEW `demo`.`view_demo` AS

SELECT * FROM demo_table

Once you have that text in the query editor, click on the **Apply** button:



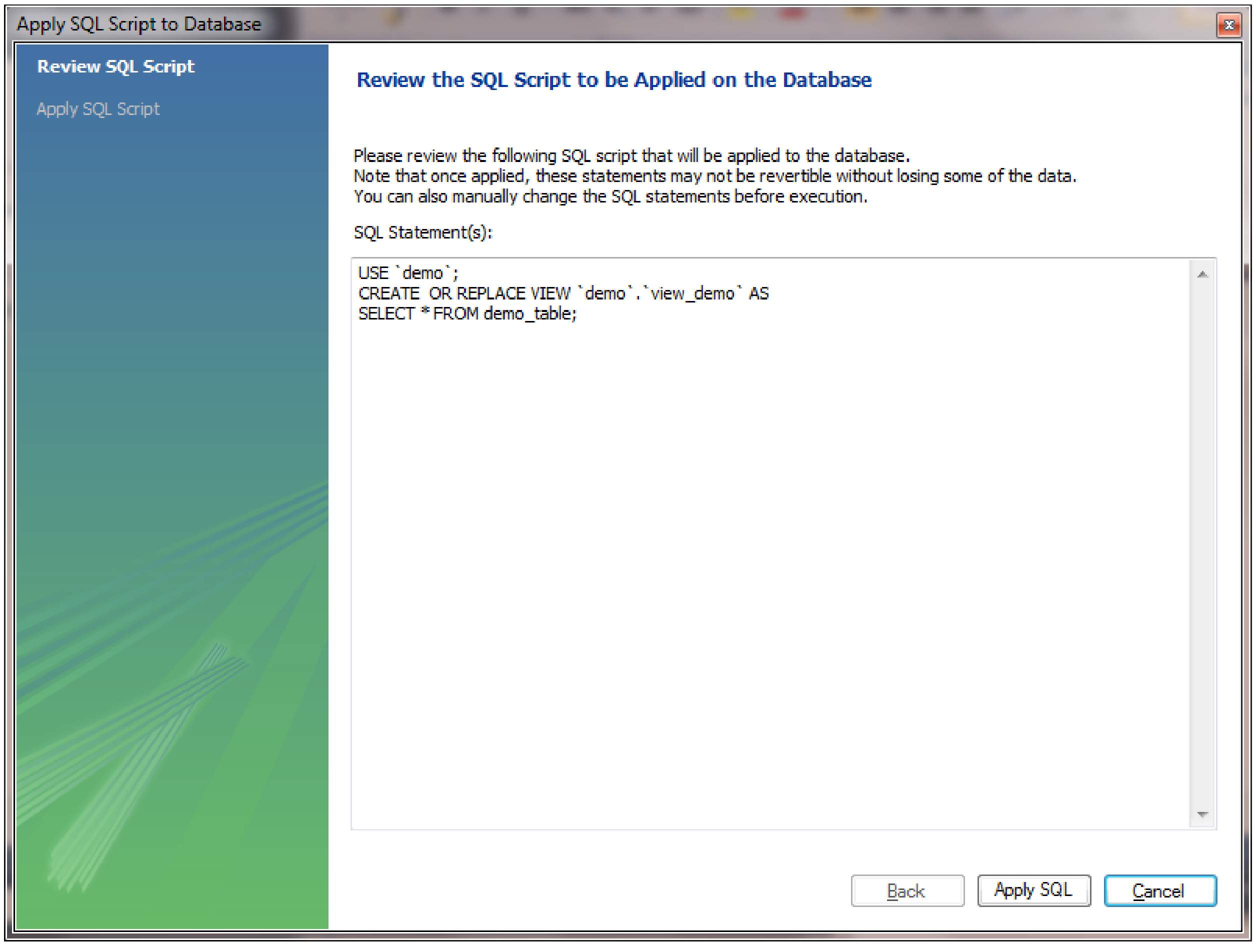
view_demo

The name of the view is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

```
CREATE VIEW `demo`.`view_demo` AS
SELECT * FROM demo_table
```

DBMS feedback messages will go here upon applying changes.

A verify window will appear to show you the syntax, so click on Apply SQL



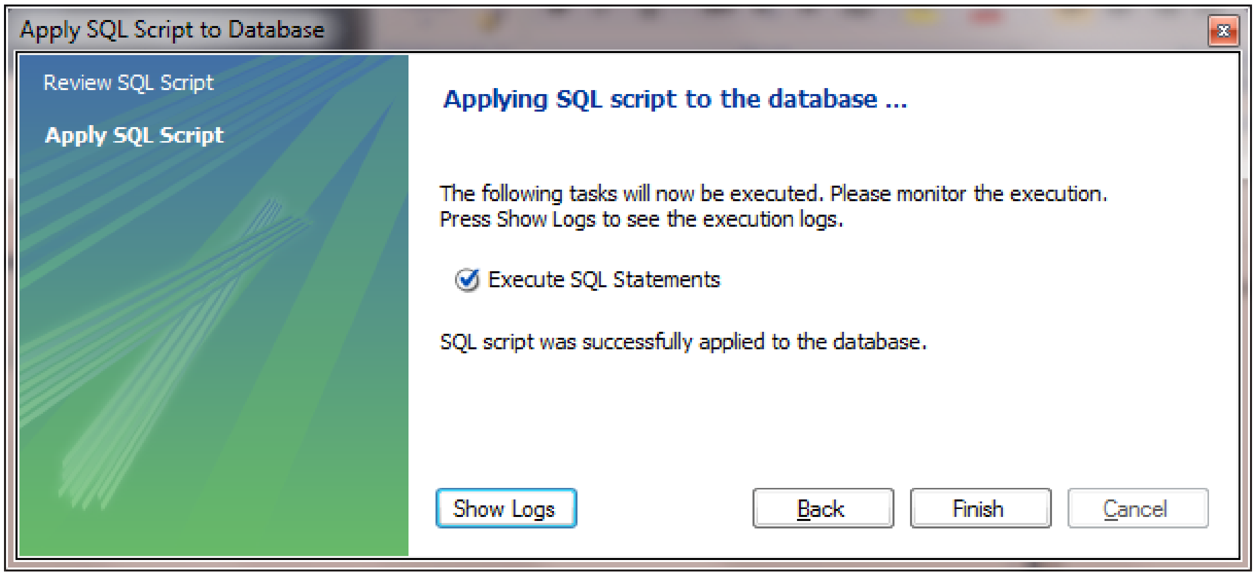
Review the SQL Script to be Applied on the Database

Please review the following SQL script that will be applied to the database.
Note that once applied, these statements may not be revertible without losing some of the data.
You can also manually change the SQL statements before execution.

SQL Statement(s):

```
USE `demo`;
CREATE  OR REPLACE VIEW `demo`.`view_demo` AS
SELECT * FROM demo_table;
```

As long as you have no script errors, the success window will appear, so click on the **Finish** button.



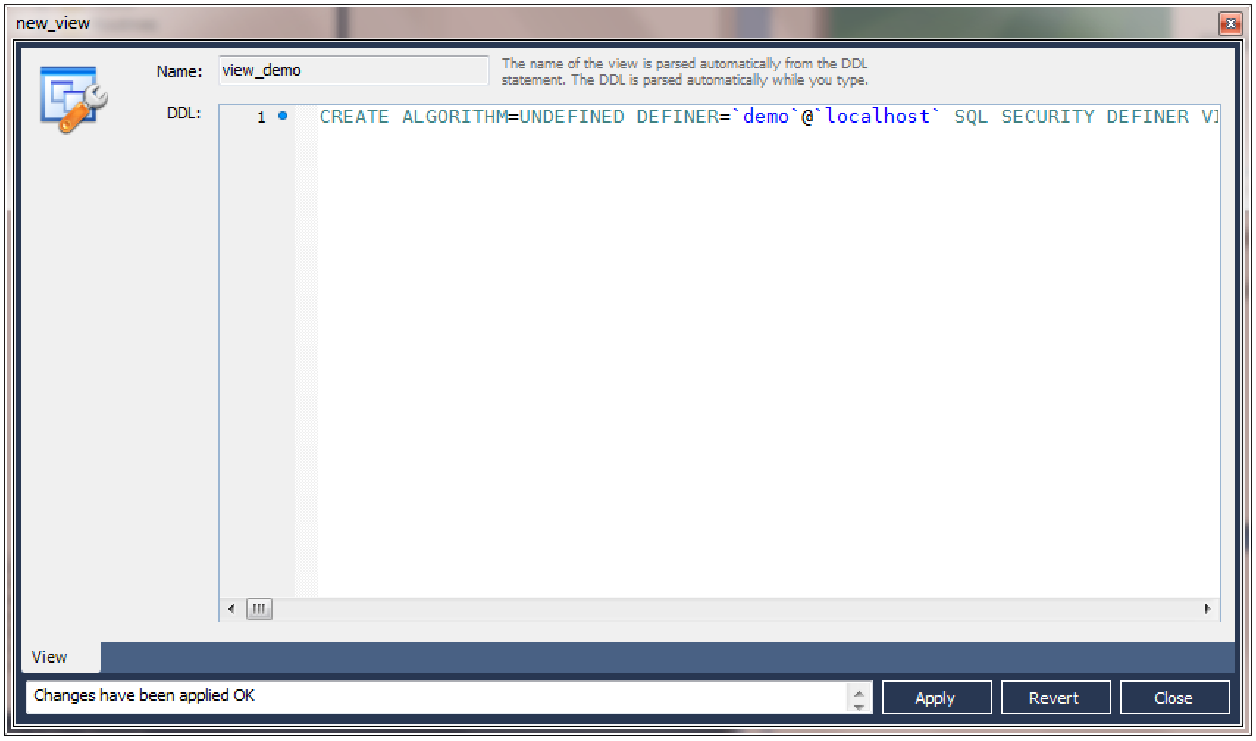
Applying SQL script to the database ...

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Execute SQL Statements

SQL script was successfully applied to the database.

Although the DDL string has now changed in the new_view window, do not worry about that for now, simply click on the **Close** button.



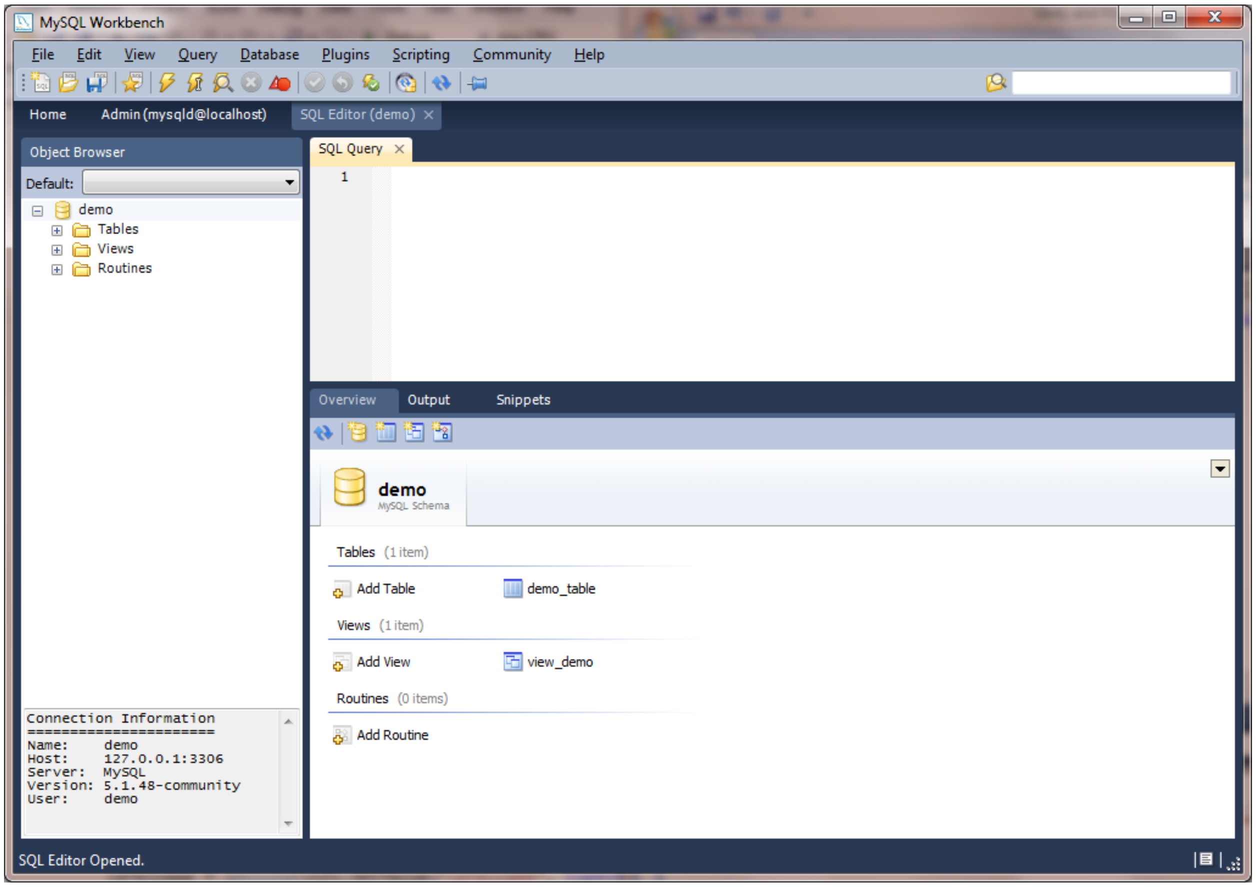
Name: view_demo

The name of the view is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

```
CREATE ALGORITHM=UNDEFINED DEFINER=`demo`@`localhost` SQL SECURITY DEFINER VI
```

Changes have been applied OK

Now, back on our SQL Editor (demo) window, we have a demo_table and a view_demo!



```
Connection Information
======================
Name:     demo
Host:     127.0.0.1:3306
Server:   MySQL
Version:  5.1.48-community
User:     demo
```

SQL Editor Opened.

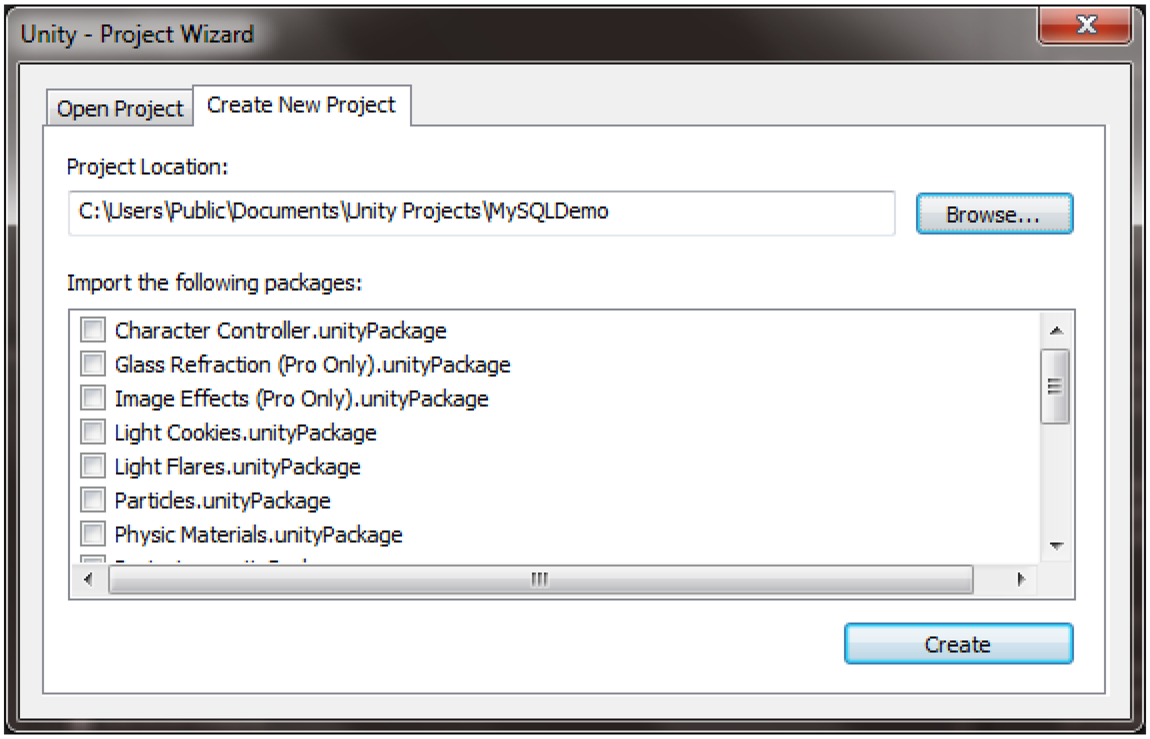
Unity 3.0.0f5

Presuming you are using Unity with version 3.0.0f5, you will want to have several files in your Assets folder for this to work. You need the Mysql.data.dll and the System.Data.dll files, both of which are in the threads on the Unity site:

http://forum.unity3d.com/threads/63008-Connection-to-MySQL-DB-mysql-client-dll

If for some reason the files are no longer in that thread, then you can locate a copy of them on your local installation of Unity within the Editor folder, there are sub folders and you can search through them for an updated copy of those files.

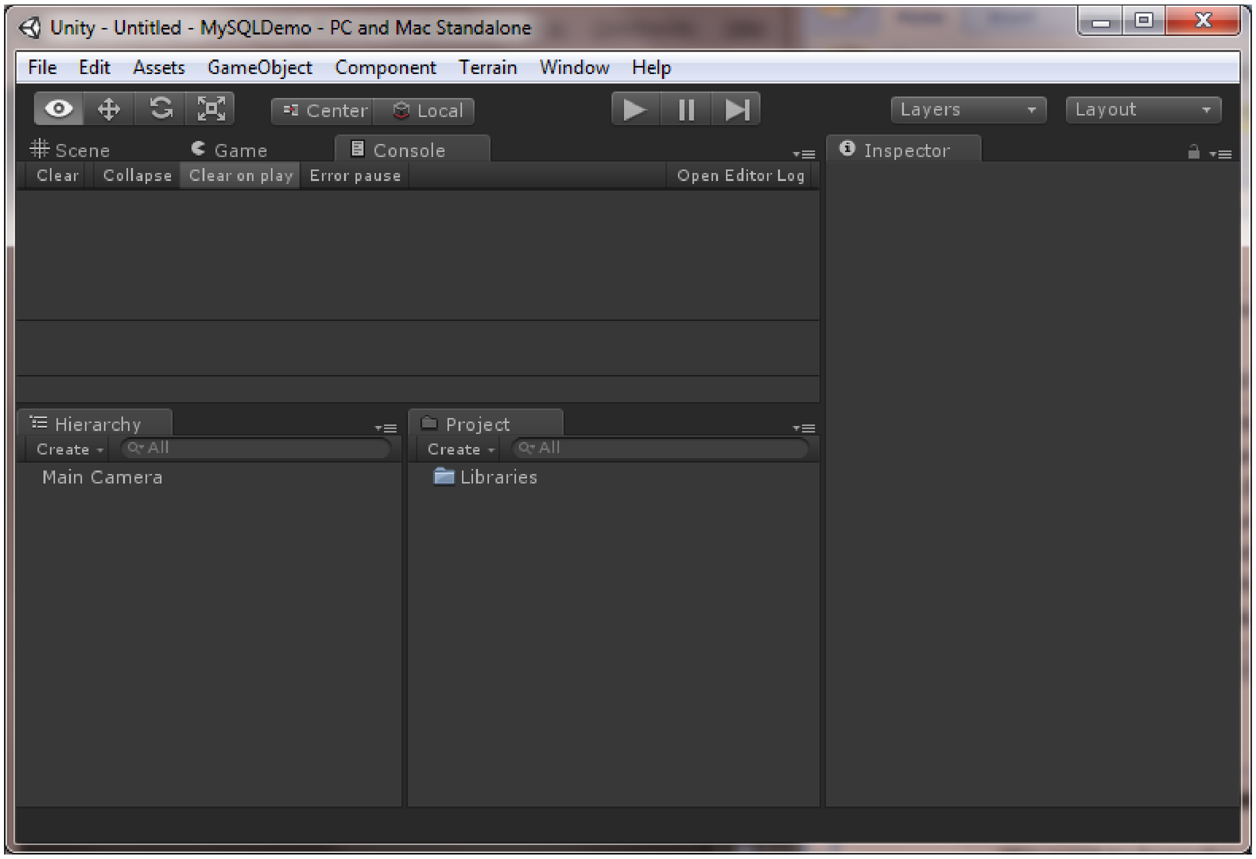
Now, to begin with, we need a brand new blank project, so open Unity, create a new project and store it in a folder called **MySQLDemo** and click on **Create**: (optionally you can use an existing project if you want)



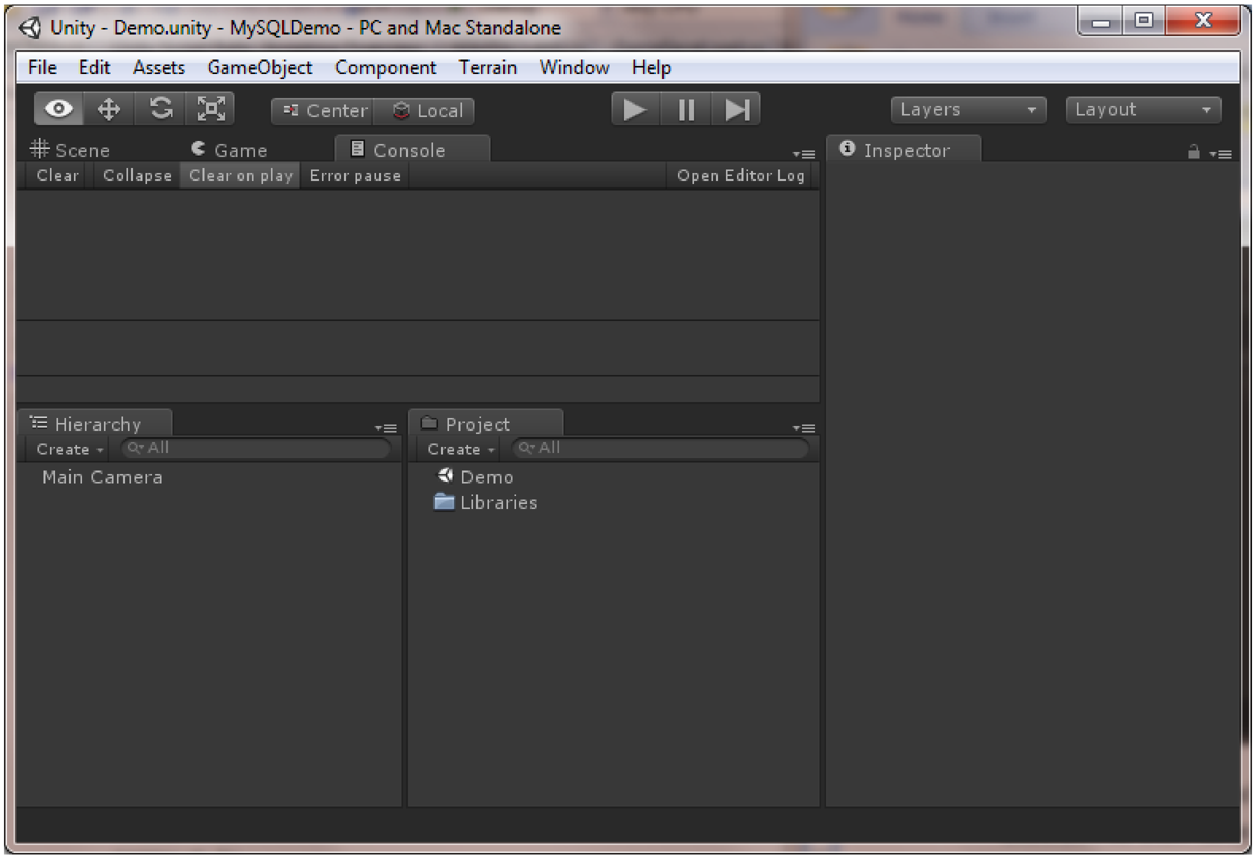
Project Location:
C:\Users\Public\Documents\Unity Projects\MySQLDemo

Import the following packages:
Character Controller.unityPackage
Glass Refraction (Pro Only).unityPackage
Image Effects (Pro Only).unityPackage
Light Cookies.unityPackage
Light Flares.unityPackage
Particles.unityPackage
Physic Materials.unityPackage

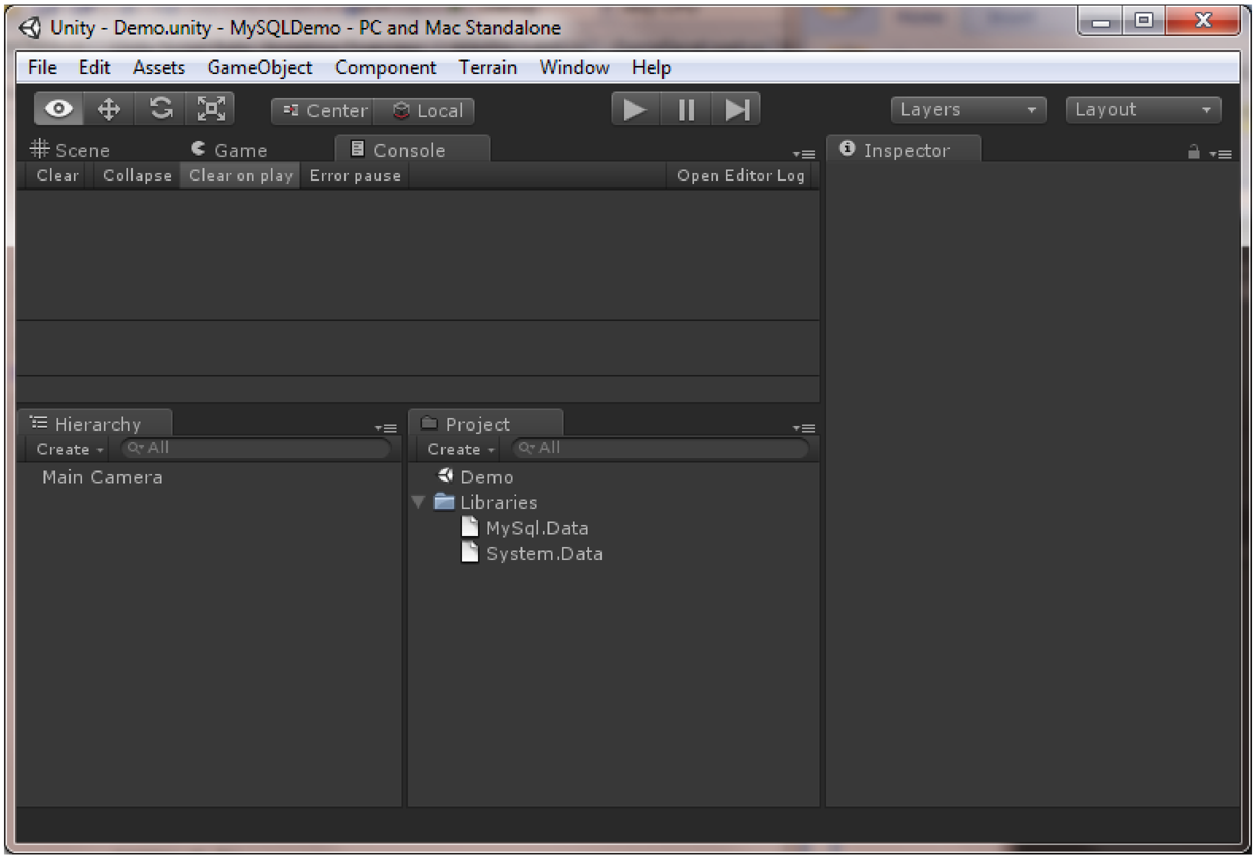
We want to keep this thing clean, so let's put our needed libraries into a folder that we will create and call that folder **Libraries,** simply click on the **Create** under the **Project** tab, choose **Folder** then rename the **New Folder** to **Libraries**, you can click on the **New Folder** so that it is highlighted and hit the **F2** function key then type in the name **Libraries** and hit **Enter**, the end result that we are looking for is as follows:


Unity - Untitled - MySQLDemo - PC and Mac Standalone
File Edit Assets GameObject Component Terrain Window Help
Center Local
Layers Layout
Scene Game Console
Inspector
Clear Collapse Clear on play Error pause Open Editor Log
Hierarchy
Create
Main Camera
Project
Create
Libraries

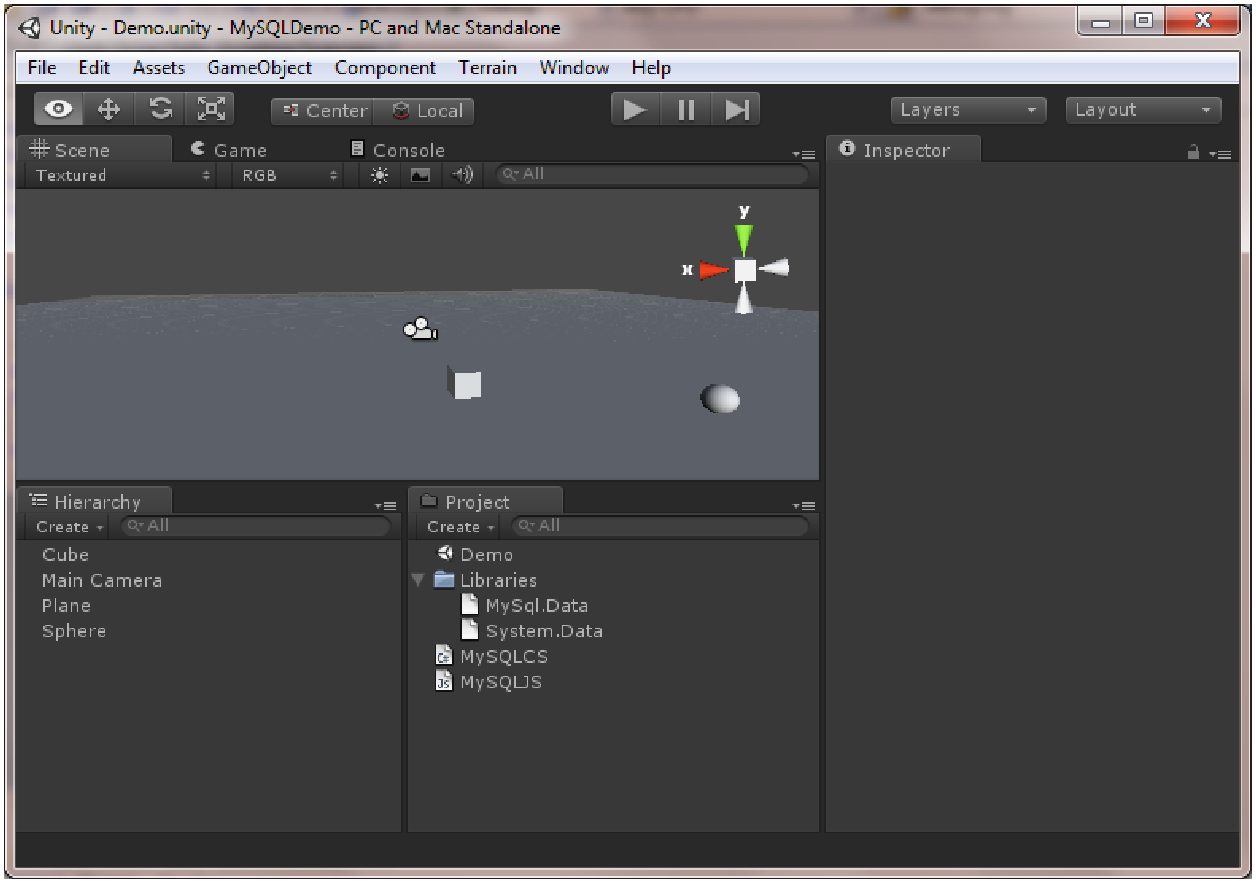
Our current workbench scene has not been saved yet, so let us go ahead and save the scene and project. Click on File then Save Scene, set the File name to Demo and click save, now we will have a Demo scene in the Project, you want to get into the habit early on with saving your work as you move through it and you will also want to get into the habit of backing up your work just in case something goes horribly wrong.



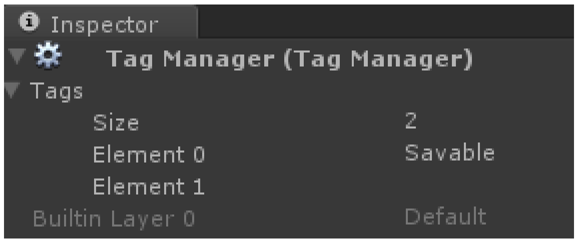
Now we are ready to drag our files into the **Libraries** folder, locate where you have saved your needed **DLL's** and drag them into the **Libraries** folder. Once you have successfully done this, Unity will then do a quick import and read the libraries then they will show up in the hierarchy as follows:


Unity - Demo.unity - MySQLDemo - PC and Mac Standalone
File Edit Assets GameObject Component Terrain Window Help
Center Local
Layers Layout
Scene Game Console
Inspector
Clear Collapse Clear on play Error pause Open Editor Log
Hierarchy
Create All
Main Camera
Project
Create All
Demo
Libraries
MySql.Data
System.Data

We now have our scene, we have our libraries, now we need to throw some random items into the scene, keep it simple, and add a plane, a cube and a sphere however you like. You use the Scene tab and then the menu item for GameObject -> Create Other then you can select the Plane, repeat for the Cube and once again for the Sphere, move them around a bit, see the Unity manual on creating primitives and using the UI. Once you have those created, save the scene so that you are saved in your current working state.



Final step here is that we need to create some way for the system to know what we want to allow to be saved and what we do not. The easiest way to accomplish this task is to create a new tag with the value of "Savable". Simply click on the cube object, and then click on the drop down for Tag then click on "Add Tag". We will make Element 0 called "Savable". Any object that we want to have saved, we will set it to that Tag.



Size	2
Element 0	Savable
Element 1	
Builtin Layer 0	Default

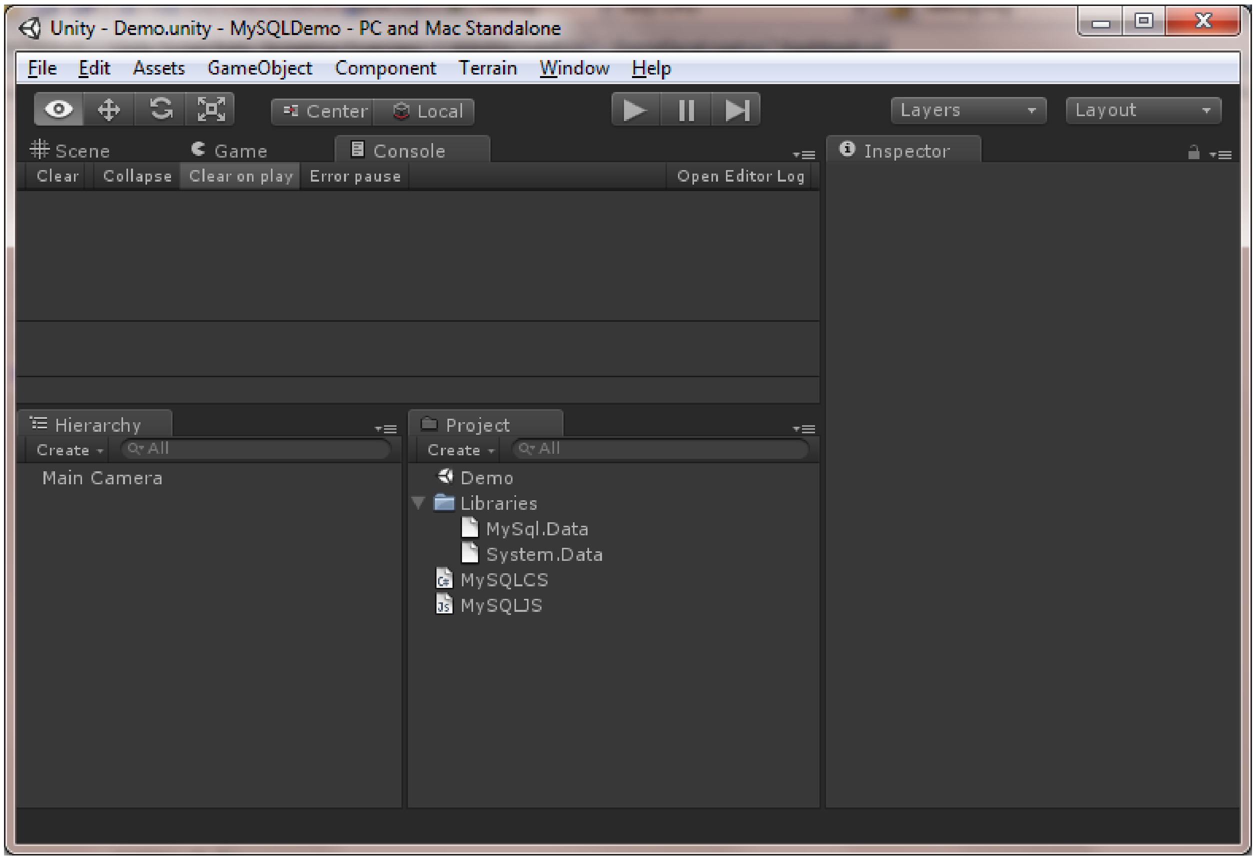
Make sure to change the Sphere and Cube Tag to the Savable before you continue this way when we use the "Save" button, it will save the values of the Cube and Sphere's position and rotation to the database.

Let's now move on to scripting so we can save the data about this scene to the database.

Unity Scripting

Remember back when we created our Schema, we had a few columns we created? This is the data we need to collect and send or retrieve from MySQL. Keeping with the theme of keeping things simple, we will only create 1 script file and call that script file MySQL, you can choose either JavaScript or C# script when creating it.

Click on Create under the Project tab then click on the script type you want to work with, again, either C Sharp Script or Javascript, now you will want to rename that script file you just created, so click on it so that it is highlighted and hit the F2 function key and rename the script to MySQL and hit enter. For this example I have created 2 scripts so I can show the code for either C# or JS, you only need 1 and call it MySQL.



Now it is time to populate the code…

Unity C Sharp Script

Double click on the MySQL script that you just created; this will then open in your favorite script editing program! (See Unity manual on how to change the editor preferences, for C# I use VS 2008). One major difference between C# and Javascript, is that C# uses Classes and you have to make sure your class matches the file name, this is a must, see the Unity manual on why.

[BEGIN CODE SNIPPET FOR C#]

```
using UnityEngine;
using MySql.Data;
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Collections;
using System.Collections.Generic;

public class MySQLCS : MonoBehaviour
{
    // In truth, the only things you want to save to the database are dynamic objects
    // static objects in the scene will always exist, so make sure to set your Tag
    // based on the documentation for this demo

    // values to match the database columns
    string ID, Name, levelname, objectType;
    float posx, posy, posz, tranx, trany, tranz;

    bool saving = false;
    bool loading = false;

    // MySQL instance specific items
    string constr = "Server=localhost;Database=demo;User ID=demo;Password=demo;Pooling=true";
    // connection object
    MySqlConnection con = null;
    // command object
    MySqlCommand cmd = null;
    // reader object
    MySqlDataReader rdr = null;
    // error object
    MySqlError er = null;
    // object collection array
    GameObject[] bodies;
    // object definitions
    public struct data
    {
        public int UID;
        public string ID, Name, levelname, objectType;
        public float posx, posy, posz, tranx, trany, tranz;
    }
    // collection container
    List<data> _GameItems;
    void Awake()
    {
        try
        {
            // setup the connection element
            con = new MySqlConnection(constr);

            // lets see if we can open the connection
            con.Open();
            Debug.Log("Connection State: " + con.State);
        }
        catch (Exception ex)
        {
            Debug.Log(ex.ToString());
        }

    }

    void OnApplicationQuit()
    {
        Debug.Log("killing con");
        if (con != null)
        {
            if (con.State != ConnectionState.Closed)
                con.Close();
            con.Dispose();
        }
    }
    // Use this for initialization
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }


    // gui event like a button, etc
    void OnGUI()
    {
        if (GUI.Button(new Rect(10, 70, 50, 30), "Save") && !saving)
        {
            saving = true;
            // first lets clean out the databae
            DeleteEntries();
            // now lets save the scene information
            InsertEntries();
            // you could also use the update if you know the ID of the item already saved

            saving = false;
        }
        if (GUI.Button(new Rect(10, 110, 50, 30), "Load") && !loading)
        {
            loading = true;
            // lets read the items from the database
            ReadEntries();
            // now display what is known about them to our log
            LogGameItems();
            loading = false;
        }
    }

    // Insert new entries into the table
    void InsertEntries()
    {
        prepData();
        string query = string.Empty;
        // Error trapping in the simplest form
        try
        {
            query = "INSERT INTO demo_table (ID, Name, levelname, objectType, posx, posy, posz, tranx, trany, tranz) VALUES (?ID,
?Name, ?levelname, ?objectType, ?posx, ?posy, ?posz, ?tranx, ?trany, ?tranz)";
            if (con.State != ConnectionState.Open)
                con.Open();
            using (con)
            {
                foreach (data itm in _GameItems)
                {
                    using (cmd = new MySqlCommand(query, con))
                    {
                        MySqlParameter oParam = cmd.Parameters.Add("?ID", MySqlDbType.VarChar);
                        oParam.Value = itm.ID;
                        MySqlParameter oParam1 = cmd.Parameters.Add("?Name", MySqlDbType.VarChar);
                        oParam1.Value = itm.Name;
                        MySqlParameter oParam2 = cmd.Parameters.Add("?levelname", MySqlDbType.VarChar);
                        oParam2.Value = itm.levelname;
                        MySqlParameter oParam3 = cmd.Parameters.Add("?objectType", MySqlDbType.VarChar);
                        oParam3.Value = itm.objectType;
                        MySqlParameter oParam4 = cmd.Parameters.Add("?posx", MySqlDbType.Float);
                        oParam4.Value = itm.posx;
                        MySqlParameter oParam5 = cmd.Parameters.Add("?posy", MySqlDbType.Float);
                        oParam5.Value = itm.posy;
                        MySqlParameter oParam6 = cmd.Parameters.Add("?posz", MySqlDbType.Float);
                        oParam6.Value = itm.posz;
                        MySqlParameter oParam7 = cmd.Parameters.Add("?tranx", MySqlDbType.Float);
                        oParam7.Value = itm.tranx;
                        MySqlParameter oParam8 = cmd.Parameters.Add("?trany", MySqlDbType.Float);
                        oParam8.Value = itm.trany;
                        MySqlParameter oParam9 = cmd.Parameters.Add("?tranz", MySqlDbType.Float);
                        oParam9.Value = itm.tranz;
                        cmd.ExecuteNonQuery();
                    }
                }
            }
        }
        catch (Exception ex)
        {
            Debug.Log(ex.ToString());
        }
        finally
        {
        }
    }

    // Update existing entries in the table based on the iddemo_table
    void UpdateEntries()
    {
        prepData();
        string query = string.Empty;
        // Error trapping in the simplest form
        try
        {
            query = "UPDATE demo_table SET ID=?ID, Name=?Name, levelname=?levelname, objectType=?objectType, posx=?posx,
posy=?posy, posz=?posz, tranx=?tranx, trany=?trany, tranz=?tranz WHERE iddemo_table=?UID";
            if (con.State != ConnectionState.Open)
                con.Open();
            using (con)
            {
                foreach (data itm in _GameItems)
                {
                    using (cmd = new MySqlCommand(query, con))
                    {
                        MySqlParameter oParam = cmd.Parameters.Add("?ID", MySqlDbType.VarChar);
                        oParam.Value = itm.ID;
                        MySqlParameter oParam1 = cmd.Parameters.Add("?Name", MySqlDbType.VarChar);
                        oParam1.Value = itm.Name;
                        MySqlParameter oParam2 = cmd.Parameters.Add("?levelname", MySqlDbType.VarChar);
                        oParam2.Value = itm.levelname;
                        MySqlParameter oParam3 = cmd.Parameters.Add("?objectType", MySqlDbType.VarChar);
                        oParam3.Value = itm.objectType;
                        MySqlParameter oParam4 = cmd.Parameters.Add("?posx", MySqlDbType.Float);
                        oParam4.Value = itm.posx;
                        MySqlParameter oParam5 = cmd.Parameters.Add("?posy", MySqlDbType.Float);
                        oParam5.Value = itm.posy;
                        MySqlParameter oParam6 = cmd.Parameters.Add("?posz", MySqlDbType.Float);
                        oParam6.Value = itm.posz;
                        MySqlParameter oParam7 = cmd.Parameters.Add("?tranx", MySqlDbType.Float);
                        oParam7.Value = itm.tranx;
                        MySqlParameter oParam8 = cmd.Parameters.Add("?trany", MySqlDbType.Float);
                        oParam8.Value = itm.trany;
                        MySqlParameter oParam9 = cmd.Parameters.Add("?tranz", MySqlDbType.Float);
                        oParam9.Value = itm.tranz;
                        MySqlParameter oParam10 = cmd.Parameters.Add("?UID", MySqlDbType.Int32);
                        oParam10.Value = itm.UID;

                        cmd.ExecuteNonQuery();
                    }
                }
            }
        }
        catch (Exception ex)
        {
            Debug.Log(ex.ToString());
        }
        finally
        {
        }
    }

    // Delete entries from the table
    void DeleteEntries()
    {
        string query = string.Empty;
        // Error trapping in the simplest form
        try
        {
            // optimally you will know which items you want to delete from the database
            // using the following code and the record ID, you can delete the entry
            //-----------------------------------------------------------------------
            // query = "DELETE FROM demo_table WHERE iddemo_table=?UID";
            // MySqlParameter oParam = cmd.Parameters.Add("?UID", MySqlDbType.Int32);
            // oParam.Value = 0;
            //-----------------------------------------------------------------------
            query = "DELETE FROM demo_table WHERE iddemo_table";
            if (con.State != ConnectionState.Open)
                con.Open();
            using (con)
            {
                using (cmd = new MySqlCommand(query, con))
                {
                    cmd.ExecuteNonQuery();
                }
            }
        }
        catch (Exception ex)
        {
            Debug.Log(ex.ToString());
        }
        finally
        {
        }
    }

    // Read all entries from the table
    void ReadEntries()
    {
        string query = string.Empty;
        if (_GameItems == null)
            _GameItems = new List<data>();
        if (_GameItems.Count > 0)
            _GameItems.Clear();
        // Error trapping in the simplest form
        try
        {
            query = "SELECT * FROM view_demo";
            if (con.State != ConnectionState.Open)
                con.Open();
            using (con)
            {
                using (cmd = new MySqlCommand(query, con))
                {
                    rdr = cmd.ExecuteReader();
                    if(rdr.HasRows)
                    while (rdr.Read())
                    {
                        data itm = new data();
                        itm.UID = int.Parse(rdr["iddemo_table"].ToString());
                        itm.ID = rdr["ID"].ToString();
                        itm.levelname = rdr["levelname"].ToString();
                        itm.Name = rdr["Name"].ToString();
                        itm.objectType = rdr["objectType"].ToString();
                        itm.posx = float.Parse(rdr["posx"].ToString());
                        itm.posy = float.Parse(rdr["posy"].ToString());
                        itm.posz = float.Parse(rdr["posz"].ToString());
                        itm.tranx = float.Parse(rdr["tranx"].ToString());
                        itm.trany = float.Parse(rdr["trany"].ToString());
                        itm.tranz = float.Parse(rdr["tranz"].ToString());
                        _GameItems.Add(itm);
                    }
                    rdr.Dispose();
                }
            }
        }
        catch (Exception ex)
        {
            Debug.Log(ex.ToString());
        }
        finally
        {
        }
    }

    /// <summary>
    /// Lets show what was read back to the log window
    /// </summary>
    void LogGameItems()
    {
        if (_GameItems != null)
        {
            if (_GameItems.Count > 0)
            {
                foreach (data itm in _GameItems)
                {
                    Debug.Log("UID: " + itm.UID);
                    Debug.Log("ID: " + itm.ID);
                    Debug.Log("levelname: " + itm.levelname);
                    Debug.Log("Name: " + itm.Name);
                    Debug.Log("objectType: " + itm.objectType);
                    Debug.Log("posx: " + itm.posx);
                    Debug.Log("posy: " + itm.posy);
                    Debug.Log("posz: " + itm.posz);
                    Debug.Log("tranx: " + itm.tranx);
                    Debug.Log("trany: " + itm.trany);
                    Debug.Log("tranz: " + itm.tranz);
                }
            }
        }
    }

    /// <summary>
    /// This method prepares the data to be saved into our database
    ///
    /// </summary>
    void prepData()
    {
        bodies = GameObject.FindGameObjectsWithTag("Savable");
        _GameItems = new List<data>();
        data itm;
        foreach (GameObject body in bodies)
        {
            itm = new data();
            itm.ID = body.name + "_" + body.GetInstanceID();
            itm.Name = body.name;
            itm.levelname = Application.loadedLevelName;
            itm.objectType = body.name.Replace("(Clone)", "");
            itm.posx = body.transform.position.x;
            itm.posy = body.transform.position.y;
            itm.posz = body.transform.position.z;
            itm.tranx = body.transform.rotation.x;
            itm.trany = body.transform.rotation.y;
            itm.tranz = body.transform.rotation.z;
            _GameItems.Add(itm);
        }
```

```
        Debug.Log("Items in collection: " + _GameItems.Count);
    }
}
```

[END CODE SNIPPET FOR C#]

Unity Javascript

Double click on the MySQL script that you just created; this will then open in your favorite script editing program! (See Unity manual on how to change the editor preferences, for Javascript I use MonoDevelop for Unity).

[BEGIN CODE SNIPPET FOR JS]

```
    // In truth, the only things you want to save to the database are dynamic objects
    // static objects in the scene will always exist, so make sure to set your Tag
    // based on the documentation for this demo

    // values to match the database columns
    var ID;
    var Name;
    var levelname;
    var objectType;
    var posx;
    var posy;
    var posz;
    var tranx;
    var trany;
    var tranz;

    var saving = false;
    var loading = false;

    // MySQL instance specific items
    var constr = "Server=localhost;Database=demo;User ID=demo;Password=demo;Pooling=true";
    // connection object
    var con;
    // command object
    var cmd;
    // reader object
    var rdr;
    // object collection array
    var bodies:GameObject[];
    // object definitions
    class data
    {
        var UID;
        var ID;
        var Name;
        var levelname;
        var objectType;
        var posx;
        var posy;
        var posz;
        var tranx;
        var trany;
        var tranz;
      }
    // collection container
    var _GameItems;
    function Awake()
    {
        try
        {
            // setup the connection element
            con = new MySql.Data.MySqlClient.MySqlConnection(constr);

            // lets see if we can open the connection
            con.Open();
            Debug.Log("Connection State: " + con.State);
        }
        catch (ex)
        {
            Debug.Log(ex.ToString());
        }

    }

    function OnApplicationQuit()
    {
        Debug.Log("killing con");
        if (con != null)
        {
            con.Close();
            con.Dispose();
        }
    }

    // Use this for initialization
    function Start()
    {

    }

    // Update is called once per frame
    function Update()
    {

    }


    // gui event like a button, etc
    function OnGUI()
    {
        if (GUI.Button(new Rect(10, 70, 50, 30), "Save") && !saving)
        {
            saving = true;
            // first lets clean out the databae
            DeleteEntries();
            // now lets save the scene information
            InsertEntries();
            // you could also use the update if you know the ID of the item already saved

            saving = false;
        }
        if (GUI.Button(new Rect(10, 110, 50, 30), "Load") && !loading)
        {
            loading = true;
            // lets read the items from the database
            ReadEntries();
            // now display what is known about them to our log
            LogGameItems();
            loading = false;
        }
    }

    // Insert new entries into the table
    function InsertEntries()
    {
        prepData();
        var query;
        // Error trapping in the simplest form
        try
        {
            query = "INSERT INTO demo_table (ID, Name, levelname, objectType, posx, posy, posz, tranx, trany, tranz) VALUES (?ID,
?Name, ?levelname, ?objectType, ?posx, ?posy, ?posz, ?tranx, ?trany, ?tranz)";
            if (con.State.ToString() != "Open")
                con.Open();

                for (itm in _GameItems)
                {
                        var cmd = new MySql.Data.MySqlClient.MySqlCommand(query, con);
                        var oParam = cmd.Parameters.AddWithValue("?ID", itm.ID);
                        var oParam1 = cmd.Parameters.AddWithValue("?Name", itm.Name);
                        var oParam2 = cmd.Parameters.AddWithValue("?levelname", itm.levelname);
                        var oParam3 = cmd.Parameters.AddWithValue("?objectType", itm.objectType);
                        var oParam4 = cmd.Parameters.AddWithValue("?posx", itm.posx);
                        var oParam5 = cmd.Parameters.AddWithValue("?posy", itm.posy);
                        var oParam6 = cmd.Parameters.AddWithValue("?posz", itm.posz);
                        var oParam7 = cmd.Parameters.AddWithValue("?tranx", itm.tranx);
                        var oParam8 = cmd.Parameters.AddWithValue("?trany", itm.trany);
                        var oParam9 = cmd.Parameters.AddWithValue("?tranz", itm.tranz);
                        cmd.ExecuteNonQuery();
                    }
        }
        catch (ex)
        {
            Debug.Log(ex.ToString());
        }
    }

    // Update existing entries in the table based on the iddemo_table
    function UpdateEntries()
    {
        prepData();
        var query;
        // Error trapping in the simplest form
        try
        {
            query = "UPDATE demo_table SET ID=?ID, Name=?Name, levelname=?levelname, objectType=?objectType, posx=?posx,
posy=?posy, posz=?posz, tranx=?tranx, trany=?trany, tranz=?tranz WHERE iddemo_table=?UID";
            if (con.State.ToString() != "Open")
                con.Open();

                for(itm in _GameItems)
                {
                        var cmd = new MySql.Data.MySqlClient.MySqlCommand(query, con);
                        var oParam = cmd.Parameters.AddWithValue("?ID", itm.ID);
                        var oParam1 = cmd.Parameters.AddWithValue("?Name", itm.Name);
                        var oParam2 = cmd.Parameters.AddWithValue("?levelname", itm.levelname);
                        var oParam3 = cmd.Parameters.AddWithValue("?objectType", itm.objectType);
                        var oParam4 = cmd.Parameters.AddWithValue("?posx", itm.posx);
                        var oParam5 = cmd.Parameters.AddWithValue("?posy", itm.posy);
                        var oParam6 = cmd.Parameters.AddWithValue("?posz", itm.posz);
                        var oParam7 = cmd.Parameters.AddWithValue("?tranx", itm.tranx);
                        var oParam8 = cmd.Parameters.AddWithValue("?trany", itm.trany);
                        var oParam9 = cmd.Parameters.AddWithValue("?tranz", itm.tranz);
                        var oParam10 = cmd.Parameters.AddWithValue("?UID", itm.UID);
                        cmd.ExecuteNonQuery();
                }
        }
        catch (ex)
        {
            Debug.Log(ex.ToString());
        }
    }

    // Delete entries from the table
    function DeleteEntries()
    {
        var query;
        // Error trapping in the simplest form
        try
    {
        // optimally you will know which items you want to delete from the database
        // using the following code and the record ID, you can delete the entry
        //-----------------------------------------------------------------------
        // query = "DELETE FROM demo_table WHERE iddemo_table=?UID";
        // MySqlParameter oParam = cmd.Parameters.Add("?UID", MySqlDbType.Int32);
        // oParam.Value = 0;
        //-----------------------------------------------------------------------
        query = "DELETE FROM demo_table WHERE iddemo_table";
        if (con.State.ToString() != "Open")
            con.Open();

            var cmd = new MySql.Data.MySqlClient.MySqlCommand(query, con);
            cmd.ExecuteNonQuery();
    }
    catch (ex)
    {
        Debug.Log(ex.ToString());
    }
}

// Read all entries from the table
function ReadEntries()
{
    var query;
    if (_GameItems == null)
        _GameItems = new Array();
    if (_GameItems.Count > 0)
        _GameItems.Clear();
    // Error trapping in the simplest form
    try
    {
        query = "SELECT * FROM view_demo";
        if (con.State.ToString() != "Open")
            con.Open();
             var cmd = new MySql.Data.MySqlClient.MySqlCommand(query, con);

                rdr = cmd.ExecuteReader();
                if(rdr.HasRows)
                while (rdr.Read())
                {
                    var itm = new data();
                    itm.UID = int.Parse(rdr["iddemo_table"].ToString());
                    itm.ID = rdr["ID"].ToString();
                    itm.levelname = rdr["levelname"].ToString();
                    itm.Name = rdr["Name"].ToString();
                    itm.objectType = rdr["objectType"].ToString();
                    itm.posx = float.Parse(rdr["posx"].ToString());
                    itm.posy = float.Parse(rdr["posy"].ToString());
                    itm.posz = float.Parse(rdr["posz"].ToString());
                    itm.tranx = float.Parse(rdr["tranx"].ToString());
                    itm.trany = float.Parse(rdr["trany"].ToString());
                    itm.tranz = float.Parse(rdr["tranz"].ToString());
                    _GameItems.Add(itm);
                }
                rdr.Dispose();
    }
    catch (ex)
    {
        Debug.Log(ex.ToString());
    }
}

/// <summary>
/// Lets show what was read back to the log window
/// </summary>
function LogGameItems()
{
    if (_GameItems != null)
    {
            if (_GameItems.Count > 0)
            {
                for(itm in _GameItems)
                {
                    Debug.Log("UID: " + itm.UID);
                    Debug.Log("ID: " + itm.ID);
                    Debug.Log("levelname: " + itm.levelname);
                    Debug.Log("Name: " + itm.Name);
                    Debug.Log("objectType: " + itm.objectType);
                    Debug.Log("posx: " + itm.posx);
                    Debug.Log("posy: " + itm.posy);
                    Debug.Log("posz: " + itm.posz);
                    Debug.Log("tranx: " + itm.tranx);
                    Debug.Log("trany: " + itm.trany);
                    Debug.Log("tranz: " + itm.tranz);
                }
            }
        }
    }

    /// <summary>
    /// This method prepares the data to be saved into our database
    ///
    /// </summary>
    function prepData()
    {
        bodies = GameObject.FindGameObjectsWithTag("Savable");
        _GameItems = new Array();
        var itm;
        for (body in bodies)
        {
            itm = new data();
            itm.ID = body.name + "_" + body.GetInstanceID();
            itm.Name = body.name;
            itm.levelname = Application.loadedLevelName;
            itm.objectType = body.name.Replace("(Clone)", "");
            itm.posx = body.transform.position.x;
            itm.posy = body.transform.position.y;
            itm.posz = body.transform.position.z;
            itm.tranx = body.transform.rotation.x;
            itm.trany = body.transform.rotation.y;
            itm.tranz = body.transform.rotation.z;
            _GameItems.Add(itm);
        }
        Debug.Log("Items in collection: " + _GameItems.Count);
    }
```

[END CODE SNIPPET FOR JS]