MySQL Stored Procedures & Views (Java)

This Document shows you how to create stored Procedures and views in MySQL and use them in Java (first with native Java, later with JPA).

Contents

[Requirements 2](#_Toc476652227)

[Eclipse Setup 2](#_Toc476652228)

[MySQL Java Connector Download 2](#_Toc476652229)

[MySQL Setup 3](#_Toc476652230)

[Create a Stored Procedure 3](#_Toc476652231)

[Create a View 4](#_Toc476652232)

[Project Setup 5](#_Toc476652233)

[Create Project and add the Connector 5](#_Toc476652234)

[Create Packages and Classes 6](#_Toc476652235)

[Create Packages 6](#_Toc476652236)

[Create Class Database 7](#_Toc476652237)

[Create Class StoredProceduresMain 8](#_Toc476652238)

[Create Class ViewsMain 9](#_Toc476652239)

[Combined with JPA 10](#_Toc476652240)

[Expand CountryModel 10](#_Toc476652241)

[Adjust Main 11](#_Toc476652242)

Last Changes: Arno Rohner, 07.03.2017 (12:15)

Added Create Class ViewsMain (9)

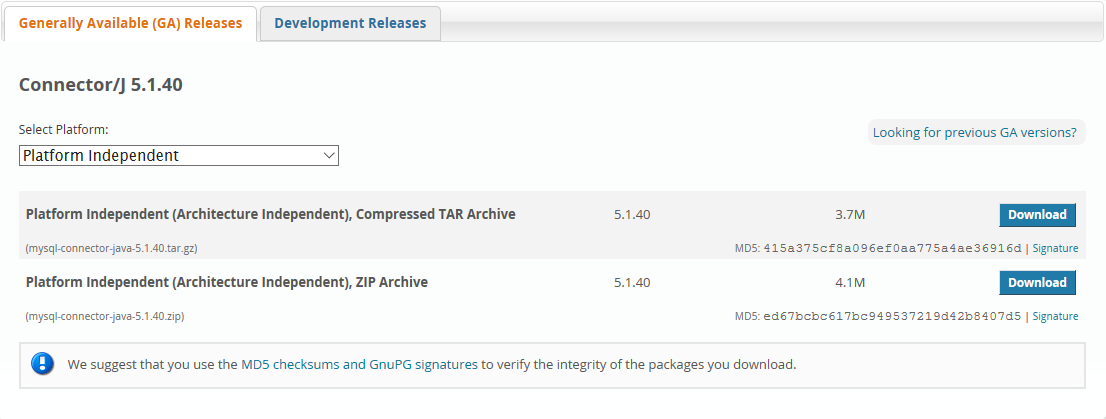
# Requirements

* Eclipse
* MySql Database (Do not delete the Schema “sakila”) with Workbench
* Know how to create Java Projects, Classes and Packages in Eclipse
* For the JPA part: JPA with EclipseLink Guide done

# Eclipse Setup

If you have done the JPA with EclipseLink guide, you can skip this and go to Project Setup.  
(The jar we download here is in eclipse/EclipseLinkJPA)

## MySQL Java Connector Download

If you have already downloaded a MySQL Java Connector (used by Glassfish), locate this file, otherwise download it from [here](https://dev.mysql.com/downloads/connector/j/): 

Screenshot from dev.mysql.com/downloads/connector/j/

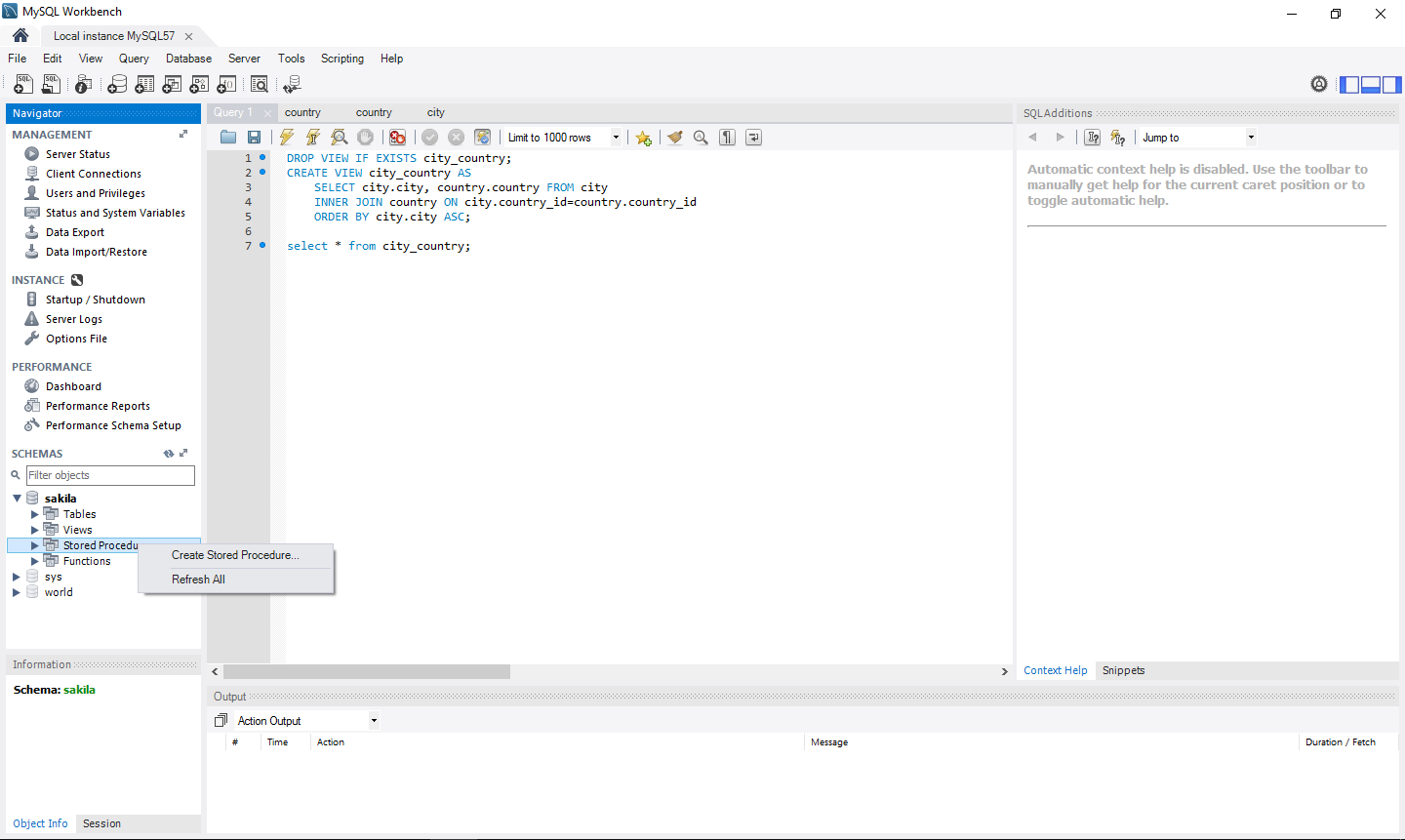
Click “No thanks, just start my download.” to start the download without logging in.

Now move the downloaded jar to the location you want and remember it.

# MySQL Setup

You will now set up a stored procedure und a view in MySQL Workbench. First start the Workbench and connect to an instance.

## Create a Stored Procedure

On the left, expand the schema “sakila” and **right click** Stored Procedures -> create Stored Procedure 

Screenshot from MySQL Workbench (the text on the right side can be ignored)

A new tab will open where you can enter text. Replace the text with the following and then click Apply and in the newly opened window click Apply again, then Finish:

CREATE PROCEDURE `searchCountry`(IN searchQuery VARCHAR(20), IN caseSesitive BOOLEAN)

BEGIN

IF caseSesitive THEN

SELECT country FROM sakila.country WHERE country LIKE BINARY searchQuery;

ELSE

SELECT country FROM sakila.country WHERE country LIKE searchQuery;

END IF;

END

You have now created a stored procedure. It is quite similar to Java Functions, in Java it would look something like this: *public ResultSet searchCountry(String searchQuery, boolean caseSensitive)*Note: stored procedures can return multiple values, you can find a good tutorial [here](http://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx).

## Create a View

Press CTRL+T to open a new Query Tab. Into that tab you paste the following code:

USE sakila;

DROP VIEW IF EXISTS city\_country;

CREATE VIEW city\_country AS

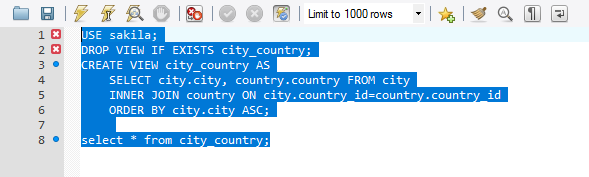
SELECT city.city, country.country FROM city

INNER JOIN country ON city.country\_id=country.country\_id

ORDER BY city.city ASC;

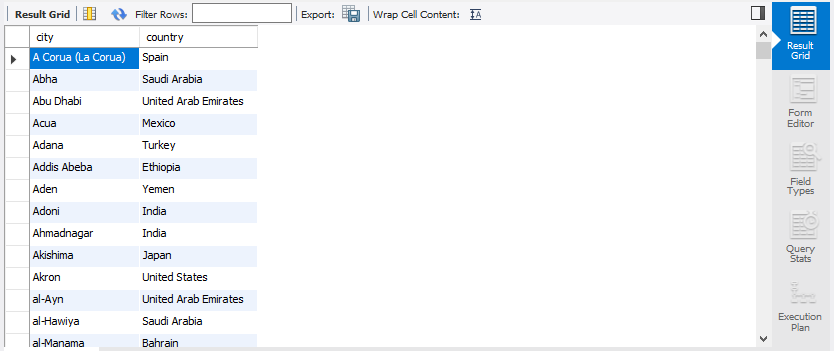
select \* from city\_country;

Now run the code by selecting the text you pasted before and clicking the lighting icon:



Screenshot of the query in MySQL Workbench

The last line will test the newly created view, so you should now see something like this:



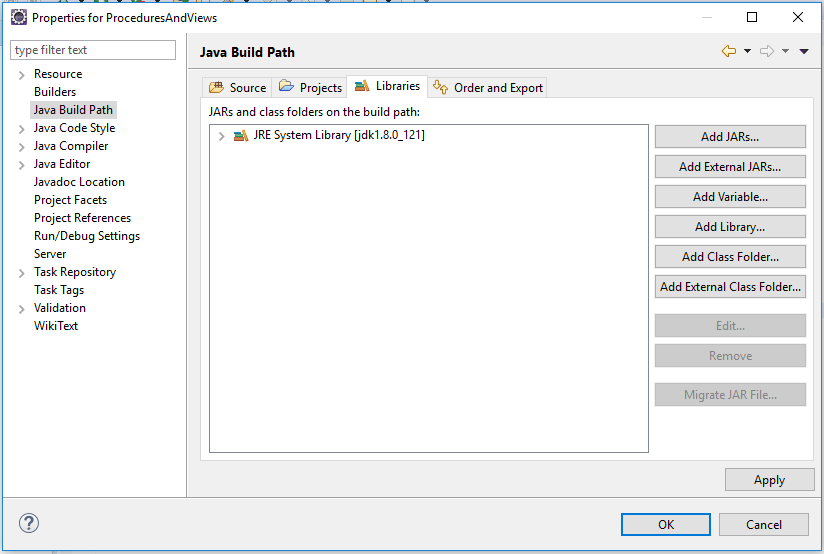
Screenshot of the result that will open when executing the query above

You have now created a view. It is like a normal table, but instead of storing its own data, it uses the data from other tables to return a table formatted for the application. Views are read-only.

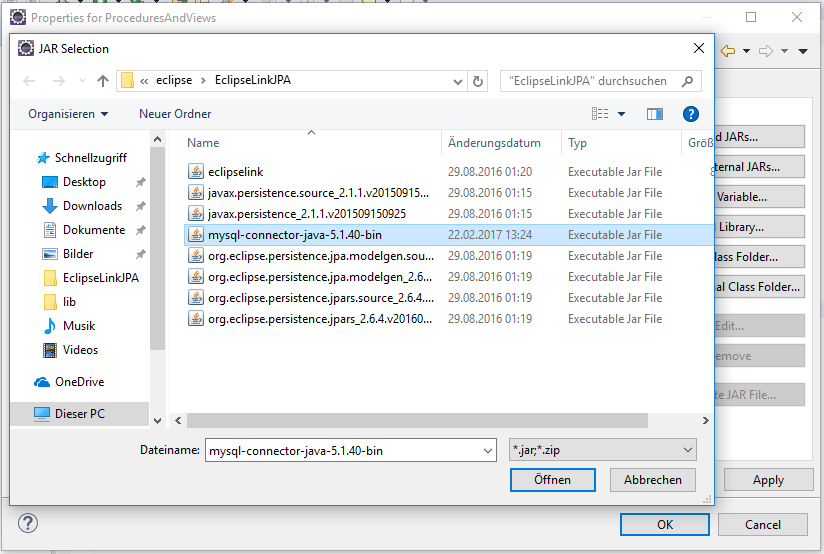
# Project Setup

## Create Project and add the Connector

Create a new Java Project called “ProceduresAndViews”. Now select the new project in the Package Explorer (left side where all the projects are) and press ALT+ENTER.

A new window will open. Click on Java Build Path, then on Add External JARs…

Screenshot of the properties window of the new Java project (opened with ALT+ENTER)

A file chooser will open, navigate to the jar you downloaded earlier and select it. 

Screenshot of the file chooser called when clicking Add External JARs…

Click Open (in the system language, in my case “Öffnen”). The file chooser will close and you will return to the properties window, where you can click OK.

## Create Packages and Classes

### Create Packages

Create two new packages, “database” and “main”.

In the database package the connection to the database will be handled, the course of the program will be defined in the main package

### Create Class Database

In the package database, create a new Class called “Database” and replace its code with the following:

package database;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

public class Database {

private static final String DRIVER = "com.mysql.jdbc.Driver";

private static final String USER = "root";

private static final String PASSWORD = "admin";

private static final String HOST = "localhost";

private static final int PORT = 3306;

public static Connection getMySQLConnection(String dbName) throws SQLException, ClassNotFoundException{

Class.forName(DRIVER);

Properties credentials = new Properties();

credentials.put("user", USER);

credentials.put("password", PASSWORD);

return DriverManager.getConnection("jdbc:mysql://" + HOST + ":" + PORT + "/" + dbName, credentials);

}

}

You may need to **adjust the private static final fields USER and PASSWORD** to fit the credentials from your database. You can adjust the field HOST to connect to a database on a different computer.

### Create Class StoredProceduresMain

In the package main, create a new Class called “StoredProceduresMain” and replace its code with the following:

package main;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import database.Database;

public class StoredProceduresMain {

private static final String DB\_NAME = "sakila";

private static final int COLUMN\_INDEX = 1;

private static final String SEARCH\_QUERY = "sw%";

private static final boolean CASE\_SENSITIVE = false;

public static void main(String[] args) throws SQLException, ClassNotFoundException {

Connection conn = Database.getMySQLConnection(DB\_NAME);

String sql = "CALL searchCountry('" + SEARCH\_QUERY + "', " + CASE\_SENSITIVE + ")";

ResultSet result = conn.createStatement().executeQuery(sql);

while(result.next()){

System.out.println(result.getString(COLUMN\_INDEX));

}

conn.close();

}

}

Run it and it should print out “Sweden” and on a new line “Switzerland” (ignore the red warning above). You can adjust the private static final fields SEARCH\_QUERY and CASE\_SENSITIVE to search for other countries. The “%” stands for anything so now it searches for countries whose name starts with “sw”, ignoring the case (uppercase / lowercase).

### Create Class ViewsMain

A view can be used like any normal table so you can alter the results with keywords like WHERE, the only difference: **Views are read-only**.

In the package main, create a new Class called “ViewsMain” and replace its code with the following:

package main;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import database.Database;

public class ViewsMain {

private static final String DB\_NAME = "sakila";

private static final String VIEW = "city\_country";

private static final int COLUMN\_INDEX = 1;

private static final String COUNTRY = "Switzerland";

public static void main(String[] args) throws SQLException, ClassNotFoundException {

Connection conn = Database.getMySQLConnection(DB\_NAME);

String sql = "SELECT city FROM " + VIEW + " WHERE country = '" + COUNTRY + "'";

ResultSet result = conn.createStatement().executeQuery(sql);

System.out.println("List of cities in " + COUNTRY + ":");

while(result.next()){

System.out.println(result.getString(COLUMN\_INDEX));

}

conn.close();

}

}

Run it and it should print out “Basel”, “Bern” and “Lausanne” (each on a different line). These are the cities from Switzerland in the database. To change the country, just edit the private static final field COUNTRY.

# Combined with JPA

You can only do this part of the Guide if you completed the Guide JPA with EclipseLink. You will still need the Project created there (you also need the things created here).

I will not show you how to use views, because views are treated like normal tables so the code is exactly the same as in the Guide for JPA with EclipseLink. (Note: Views are read-only)

From now on, work in the old project “CountrySearch”.

## Expand CountryModel

In the Class CountryModel, add a new method:

@SuppressWarnings("unchecked")

**public** List<String> searchCountryNames(String searchQuery, **int** searchOption, **boolean** caseSensitive){

List<String> result = **new** ArrayList<String>();

**switch** (searchOption) {

**case** ***BEGINS\_WITH\_OPTION***:

searchQuery = searchQuery + "%";

**break**;

**case** ***ENDS\_WITH\_OPTION***:

searchQuery = "%" + searchQuery;

**break**;

**case** ***CONTAINS\_OPTION***:

searchQuery = "%" + searchQuery + "%";

**break**;

**default**:

**throw** **new** IllegalArgumentException();

}

StoredProcedureQuery storedProcedure = em.createStoredProcedureQuery("searchCountry");

storedProcedure.registerStoredProcedureParameter("searchQuery", String.**class**, ParameterMode.***IN***);

storedProcedure.registerStoredProcedureParameter("caseSensitive", Boolean.**class**, ParameterMode.***IN***);

storedProcedure.setParameter("searchQuery", searchQuery);

storedProcedure.setParameter("caseSensitive", caseSensitive);

storedProcedure.execute();

**for**(Object[] row : (List<Object[]>)storedProcedure.getResultList()){

result.add(row[0].toString());

}

**return** result;

}

The whole code of the class (made small to save space):

package model;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.ParameterMode;

import javax.persistence.Persistence;

import javax.persistence.StoredProcedureQuery;

public class CountryModel {

public static final int BEGINS\_WITH\_OPTION = 1;

public static final int NOT\_BEGINS\_WITH\_OPTION = -1;

public static final int ENDS\_WITH\_OPTION = 2;

public static final int NOT\_ENDS\_WITH\_OPTION = -2;

public static final int CONTAINS\_OPTION = 3;

public static final int NOT\_CONTAINS\_OPTION = -3;

private EntityManager em = Persistence.createEntityManagerFactory("Example\_JPA").createEntityManager();

private List<Country> searchCountry(String sqlCondition){

List<Integer> ids = em.createNativeQuery("Select country\_id from country where " + sqlCondition).getResultList();

List<Country> countries = new ArrayList<Country>();

for(int id : ids){

countries.add(getCountryById(id));

}

return countries;

}

public List<Country> searchCountry(String searchQuery, int searchOption, boolean caseSensitive){

String like = "LIKE";

String condition;

if(caseSensitive) like = like + " BINARY";

if(searchOption < 0){ //If the value is negative, it will be converted to its positive equal and NOT will be used

like = "NOT " + like;

searchOption \*= -1;

}

switch (searchOption) {

case BEGINS\_WITH\_OPTION:

condition = "\"" + searchQuery + "%\"";

break;

case ENDS\_WITH\_OPTION:

condition = "\"%" + searchQuery + "\"";

break;

case CONTAINS\_OPTION:

condition = "\"%" + searchQuery + "%\"";

break;

default:

throw new IllegalArgumentException();

}

return searchCountry("country " + like + " " + condition);

}

public Country getCountryById(int id){

return em.find(Country.class, id);

}

@SuppressWarnings("unchecked")

public List<String> searchCountryNames(String searchQuery, int searchOption, boolean caseSensitive){

List<String> result = new ArrayList<String>();

switch (searchOption) {

case BEGINS\_WITH\_OPTION:

searchQuery = searchQuery + "%";

break;

case ENDS\_WITH\_OPTION:

searchQuery = "%" + searchQuery;

break;

case CONTAINS\_OPTION:

searchQuery = "%" + searchQuery + "%";

break;

default:

throw new IllegalArgumentException();

}

StoredProcedureQuery storedProcedure = em.createStoredProcedureQuery("searchCountry");

storedProcedure.registerStoredProcedureParameter("searchQuery", String.class, ParameterMode.IN);

storedProcedure.registerStoredProcedureParameter("caseSensitive", Boolean.class, ParameterMode.IN);

storedProcedure.setParameter("searchQuery", searchQuery);

storedProcedure.setParameter("caseSensitive", caseSensitive);

storedProcedure.execute();

for(Object[] row : (List<Object[]>)storedProcedure.getResultList()){

result.add(row[0].toString());

}

return result;

}

}

## Adjust Main

In the class Main, comment out the for-loop and add the following code below it (between the for-loop and *in.close()*)

//With Stored Procedures

**for**(String countryName : model.searchCountryNames(searchQuery, option, caseSensitive)){

System.***out***.println(countryName);

}

The whole code of the class:

package main;

import java.util.Scanner;

import model.Country;

import model.CountryModel;

public class Main {

public static void main(String[] args) {

CountryModel model = new CountryModel();

Scanner in = new Scanner(System.in);

System.out.println("Search country:");

String searchQuery = in.nextLine();

System.out.println("Select search option:");

System.out.println(" 1 : Begins with...");

System.out.println("-1 : Does not begin with...");

System.out.println(" 2 : Ends with...");

System.out.println("-2 : Does not end with...");

System.out.println(" 3 : Contains...");

System.out.println("-3 : Does not contain...");

int option = Integer.parseInt(in.nextLine());

System.out.println("Case sensitive? (Y/N):");

String caseSensitiveInput = in.nextLine().toUpperCase();

boolean caseSensitive = false;

switch (caseSensitiveInput) {

case "Y":

caseSensitive = true;

break;

case "N":

caseSensitive = false;

break;

default:

System.out.println("Invalid Input");

System.exit(0);

}

//Without Stored Procedures

// for(Country c : model.searchCountry(searchQuery, option, caseSensitive)){

// System.out.println(c.getCountry());

// }

//With Stored Procedures

for(String countryName : model.searchCountryNames(searchQuery, option, caseSensitive)){

System.out.println(countryName);

}

in.close();

}

}

Run it and the program should still work as it did before. If you are not sure if the results are correct, remove the slashes (comment) from the first for-loop and run it. If it prints out the same result 2 times, it works.