# Homework: Entity Framework Performance

This document defines the homework assignments from the ["Database Applications" Course @ Software University](https://softuni.bg/trainings/21/Database-Applications-Mar-2015). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code) of all below described problems.

## Show Data from Related Tables

You are given a **MS SQL Server database "Ads"** holding advertisements, organized by categories, towns and users, available as SQL script.

Using **Entity Framework** write a SQL query to select all **ads** from the database and later print their **title**, **status**, **category**, **town** and **user**. Do not use Include(…) for the relationships of the Ads. Check how many SQL commands are executed with the [SQL ExpressProfiler](https://expressprofiler.codeplex.com) (or a similar tool).

Add Include(…) to select **statuses**, **categories**, **towns** and **users** along with all **ads**. Compare the number of executed SQL statements and the performance before and after adding Include(…).

For this problem you will have to submit the **two versions of the program**: the one with **Include(…)** and the one without **Include(…)**. You will also need to fill **the following table** and submit it as a **.txt** file:

|  |  |  |
| --- | --- | --- |
|  | **No Include(…)** | **With Include(…)** |
| **Number of SQL statements** |  |  |

Here is a tool which can create text tables: <http://ozh.github.io/ascii-tables/>.

## Play with ToList()

Using Entity Framework select all **ads** from the database, then invoke ToList(), then filter the categories whose status is **Published**; then select the ad **title**, **category, town** and **date**, then invoke ToList() again and finally **order** the ads by **publish** **date**. Rewrite the same query in a more optimized way and compare the performance.

Compare the execution time of the two programs. **Hint:** use the **System.Diagnostics.Stopwatch** class. You can see a tutorial on how to use it [here](http://www.dotnetperls.com/stopwatch). Run each program 10 times and write the average performance time in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Run 1** | **Run 2** | **…** | **Run 10** | **Average time** |
| **Non-optimized** |  |  |  |  |  |
| **Optimized** |  |  |  |  |  |

For this problem you will have to submit the **two versions of the program**: the **slow** version and the **optimized** version.

**WARNING:** You may see that the second, third, etc. runs are much faster than the first one. This is because SQL Server caches the executed queries. To make tests valid, **in the beginning of your program**, send the following native SQL query:

**CHECKPOINT; DBCC DROPCLEANBUFFERS;**

## Select Everything vs. Select Certain Columns

Write a program to compare the execution speed between these two scenarios:

* Select **everything** from the Ads table and print only the ad title.
* Select the **ad** **title** from Ads table and print it.

Run the two queries 10 times and write down the average time. Follow all the steps you did for Problem 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Run 1** | **Run 2** | **…** | **Run 10** | **Average time** |
| **Non-optimized** |  |  |  |  |  |
| **Optimized** |  |  |  |  |  |

For this problem you will have to submit the **two versions of the program**: the **slow** version and the **optimized** version, and the table above.

## Test Performance of Order By

Use the **SoftUni** database. Write a program that measure time needed to sort the Employees table **by JobTitle** **ascending** and then **by DepartmentID** **descending**. Add more records to the table and measure it again. Fill the table below to compare the results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Employees**  **Sorting Time** | **293** | **1000** | **10 000** | **100 000** | **1 000 000** |
| **Before calling .ToList()** |  |  |  |  |  |
| **After calling .ToList()** |  |  |  |  |  |