# Databases – Practical Teamwork Project

A chain of supermarkets holds information about its **products in MySQL database** consisting of the tables like the shown below:

## Products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **VendorID** | **Product Name** | **MeasureID** | **Base Price** |
| 1 | 20 | Beer “Zagorka” | 100 | 0.86 |
| 2 | 30 | Vodka “Targovishte” | 100 | 7.56 |
| 3 | 20 | Beer “Beck’s” | 100 | 1.03 |
| 4 | 10 | Chocolate “Milka” | 200 | 2.80 |
| … | … | … | … | … |

## Vendors

|  |  |
| --- | --- |
| **ID** | **Vendor Name** |
| 10 | Nestle Sofia Corp. |
| 20 | Zagorka Corp. |
| 30 | Targovishte Bottling Company Ltd. |
| … | … |

## Measures

|  |  |
| --- | --- |
| **ID** | **Measure Name** |
| 100 | liters |
| 200 | pieces |
| … | … |

For testing purposes please fill at least 50 products, at least 20 vendors and at least 5 measures. Try to use real-world data.

## Sales Reports

At the central server of the supermarket chain every day a set of new **sales reports** are submitted from the local supermarkets (which are spread in the entire country). Each of these reports is **an Excel 2003 file** that has a single spreadsheet called “Sales” containing information about the daily sales:

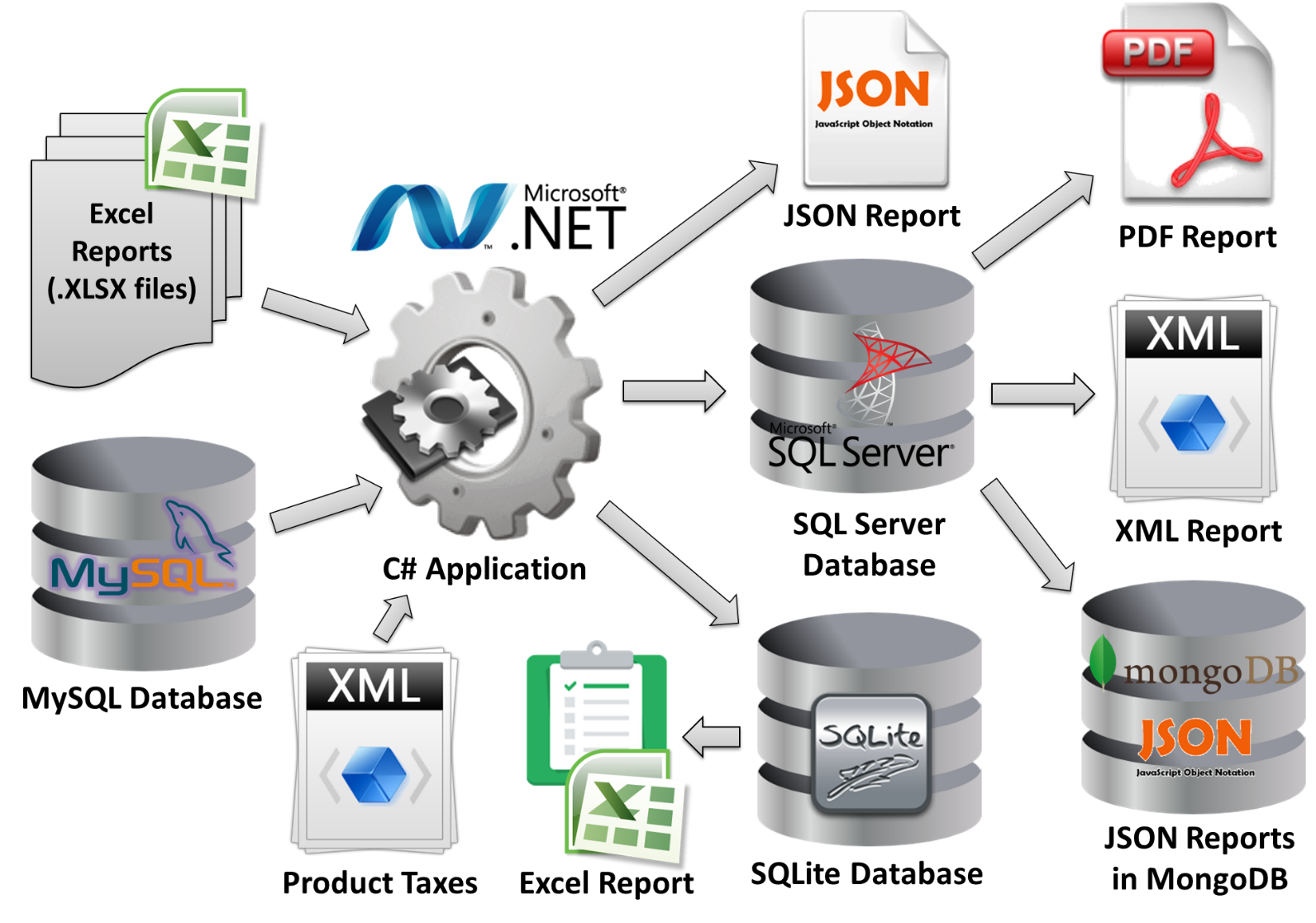
## Sales Report – Sample Excel File

|  |  |  |  |
| --- | --- | --- | --- |
| **Supermarket “Kaspichan – Center”** | | | |
| **ProductID** | **Quantity** | **Unit Price** | **Sum** |
| 3 | 40 | 1.20 | 48.00 |
| 1 | 65 | 0.92 | 59.80 |
| 4 | 12 | 2.90 | 34.80 |
| … | … | … | … |
| **Total sum:** | | | **142.60** |

The unit price could slightly vary in the different supermarkets from the chain due to local marketing and sales policy.

## Assignment

Your assignment is to design, develop and test a C# application for importing Excel sales reports from ZIP file and product data from MySQL into SQL Server, generate PDF aggregated reports and XML sales reports, create product reports as JSON documents and load them into MongoDB, load vendor expenses from XML file, read taxes from SQLite and calculate vendor's total results and write them into Excel file.



## Problem #1 – Load Excel Reports from ZIP File

Suppose you have the **MySQL database “Supermarket”** holding information about the vendors and products and **a set of Excel files** (\*.xls) holding information about the sales in the different super­markets.

Your task is to **load the Excel reports in** **MS SQL Server**. You will need preliminary to move design a database schema to hold all data about products (data from the MySQL database and data from the Excel files). You are not allowed to manually copy the MySQL tables (your C# program should do it).

The Excel files are given inside a **ZIP archive** holding subfolders named as the dates of the report in format **dd-MMM-yyyy** (see the example reports archive [Sample-Sales-Reports.zip](file:///C:\NAKOV\Databases\2013\18.%20Databases%20Team%20Work%20Project\Databases-Teamwork-Practical-Project\Sample-Sales-Reports.zip)).

Note that ZIP file could contain few hundred dates (folders), each holding few hundreds Excel files, each holding thousands of products sold.

## Problem #2 – Generate PDF Aggregated Sales Reports

Your task is to **generate a PDF aggregated report** summarizing the sales from all supermarkets for all available dates from the SQL Server.

The **PDF report** should contain a table like the sample below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aggregated Sales Report** | | | | |
| Date: 20-Jul-2013 | | | | |
| **Product** | **Quantity** | **Unit Price** | **Location** | **Sum** |
| Beer “Beck’s” | 40 liters | 1.20 | Supermarket “Kaspichan – Center” | 48.00 |
| Beer “Zagorka” | 37 liters | 1.00 | Supermarket “Bourgas – Plaza” | 37.00 |
| Chocolate “Milka” | 7 pieces | 2.85 | Supermarket “Bay Ivan” – Zmeyovo | 19.95 |
| Vodka “Targovishte” | 14 liters | 8.50 | Supermarket “Bourgas – Plaza” | 119.00 |
| Chocolate “Milka” | 12 pieces | 2.90 | Supermarket “Kaspichan – Center” | 34.80 |
| Beer “Zagorka” | 65 liters | 0.92 | Supermarket “Kaspichan – Center” | 59.80 |
| Vodka “Targovishte” | 4 liters | 7.80 | Supermarket “Bay Ivan” – Zmeyovo | 31.20 |
| … | … | … | … | … |
| Total sum for 20-Jul-2012: | | | | **349.75** |
| Date: 21-Jul-2013 | | | | |
| **Product** | **Quantity** | **Unit Price** | **Location** | **Sum** |
| Beer “Zagorka” | 11 liters | 1.00 | Supermarket “Bourgas – Plaza” | 11.00 |
| Beer “Zagorka” | 78 liters | 0.92 | Supermarket “Kaspichan – Center” | 71.76 |
| Beer “Zagorka” | 146 liters | 0.88 | Supermarket “Plovdiv – Stolipinovo” | 128.48 |
| Vodka “Targovishte” | 20 liters | 8.50 | Supermarket “Bourgas – Plaza” | 170.00 |
| Vodka “Targovishte” | 67 liters | 7.70 | Supermarket “Plovdiv – Stolipinovo” | 515.90 |
| Vodka “Targovishte” | 3 liters | 7.80 | Supermarket “Bay Ivan” – Zmeyovo | 23.40 |
| Beer “Beck’s” | 43 liters | 1.20 | Supermarket “Kaspichan – Center” | 51.60 |
| Beer “Beck’s” | 75 liters | 1.05 | Supermarket “Plovdiv – Stolipinovo” | 78.75 |
| Chocolate “Milka” | 9 pieces | 2.90 | Supermarket “Kaspichan – Center” | 26.10 |
| Chocolate “Milka” | 5 pieces | 2.85 | Supermarket “Bay Ivan” – Zmeyovo | 14.25 |
| … | … | … | … | … |
| Total sum for 21-Jul-2012: | | | | **1091.24** |
| Date: 22-Jul-2013 | | | | |
| **Product** | **Quantity** | **Unit Price** | **Location** | **Sum** |
| Beer “Zagorka” | 16.00 | 1.00 | Supermarket “Bourgas – Plaza” | 16.00 |
| Beer “Zagorka” | 90.00 | 0.92 | Supermarket “Kaspichan – Center” | 82.80 |
| Beer “Zagorka” | 230.00 | 0.88 | Supermarket “Plovdiv – Stolipinovo” | 202.40 |
| Vodka “Targovishte” | 24.00 | 8.50 | Supermarket “Bourgas – Plaza” | 204.00 |
| Vodka “Targovishte” | 12.00 | 7.70 | Supermarket “Plovdiv – Stolipinovo” | 92.40 |
| Beer “Beck’s” | 18.00 | 1.20 | Supermarket “Kaspichan – Center” | 21.60 |
| Beer “Beck’s” | 60.00 | 1.05 | Supermarket “Plovdiv – Stolipinovo” | 63.00 |
| Chocolate “Milka” | 14.00 | 2.90 | Supermarket “Kaspichan – Center” | 40.60 |
| … | … | … | … | … |
| Total sum for 21-Jul-2012: | | | | **722.80** |
| Grand total: | | | | **2163.79** |

A sample PDF report is also available: <Sample-Aggregated-Sales-Report.pdf>.

## Problem #3 – Generate XML Sales Report by Vendors

Your task is to create a C# program to **generate aggregated sales report by dates in XML format** like the sample below:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8">  <sales>  <sale vendor="Nestle Sofia Corp.">  <summary date="20-Jul-2013" total-sum="54.75" />  <summary date="21-Jul-2013" total-sum="40.35" />  <summary date="20-Jul-2013" total-sum="40.60" />  </sale>  <sale vendor="Targovishte Bottling Company Ltd.">  <summary date="20-Jul-2013" total-sum="150.20" />  <summary date="21-Jul-2013" total-sum="709.30" />  <summary date="20-Jul-2013" total-sum="249.40" />  </sale>  <sale vendor="Zagorka Corp.">  <summary date="20-Jul-2013" total-sum="144.80" />  <summary date="21-Jul-2013" total-sum="341.59" />  <summary date="20-Jul-2013" total-sum="385.80" />  </sale>  <sales> |

Save the report in a file named “**Sales-by-Vendors-report.xml**”.

## Problem #4 – Product Reports

Your task is to write a program to create a **product report** for each product in **JSON format** and save all reports as JSON documents in **MongoDB**. All product reports should look like the sample below and should be saved in the **MongoDB** database as well as in the file system (in a folder called “**Product-Reports**”, in files named “**XX.json**” where **XX** is the product ID).

Sample product report in JSON format:

|  |
| --- |
| [{  "product-id" : 3,  "product-name" : "Beer “Beck’s”",  "vendor-name" : "Zagorka Corp.",  "total-quantity-sold" : 236,  "total-incomes" : 262.95,  },  {  "product-id" : 1,  "product-name" : "Beer “Zagorka”",  "vendor-name" : "Zagorka Corp.",  "total-quantity-sold" : 673,  "total-incomes" : 609.24,  },  {  "product-id" : 4,  "product-name" : "Chocolate “Milka”",  "vendor-name" : "Nestle Sofia Corp.",  "total-quantity-sold" : 47,  "total-incomes" : 135.70,  },  {  "product-id" : 2,  "product-name" : "Vodka “Targovishte”",  "vendor-name" : "Targovishte Bottling Company Ltd.",  "total-quantity-sold" : 144,  "total-incomes" : 1155.90,  }] |

## Problem #5 – Load Vendor Expenses from XML

You are given an **XML file** **Vendors-Expenses.xml** holding the expenses of all vendors by months in the following format:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8">  <sales>  <sale vendor="Nestle Sofia Corp.">  <expenses month="Jul-2013">30.00</expenses>  <expenses month="Aug-2013">40.00</expenses>  </sale>  <sale vendor="Targovishte Bottling Company Ltd.">  <expenses month="Jul-2013">200.00</expenses>  <expenses month="Aug-2013">180.00</expenses>  </sale>  <sale vendor="Zagorka Corp.">  <expenses month="Jul-2013">120.00</expenses>  <expenses month="Aug-2013">180.00</expenses>  </sale>  <sales> |

Your task is to **read the expenses** XML file, parse it and **save the expenses** in the **MongoDB** database and in the **SQL Server**. Please think how your database schema / document model will support expenses.

## Problem #6 – Vendors Total Report

You are given a **SQLite database** holding the taxes for each product in the following format:

|  |  |
| --- | --- |
| **Product Name** | **Tax** |
| Beer “Beck’s” | 20% |
| Beer “Zagorka” | 20% |
| Chocolate “Milka” | 18% |
| Vodka “Targovishte” | 25% |

Write a program to **read the MongoDB database** of product reports, **load the products into SQLite** and generate a **single Excel** file called “**Products-Total-Report.xlsx**” holding the following information **for the current month** (e.g. July 2013):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vendor** | **Incomes** | **Expenses** | **Taxes** | **Financial Result** |
| Nestle Sofia Corp. | 135.70 | 30 | 24.43 | **81.27** |
| Targovishte Bottling Company Ltd. | 1155.90 | 200 | 288.98 | **666.92** |
| Zagorka Corp. | 872.19 | 120 | 174.44 | **577.75** |

You are allowed to connect to the SQL Server or MySQL databases to read information about the vendors.

## Additional Requirements

* Your main program logic should be a C# application.
* Use non-commercial framework to read the ZIP file.
* MySQL should be accessed through OpenAccess ORM.
* SQL Server should be accessed through Entity Framework.
* MongoDB should be accessed through the Official MongoDB C# Driver.
* The Excel files should be accessed through ADO.NET (without ORM or third-party libraries).
* For the PDF export use a non-commercial third party framework.
* The XML files should be read / written through the standard .NET parsers (by your choice).
* The SQLite embedded database should be accesses though its Entity Framework provider.