Framework for Composing Aggregate Reports in Bi Modules

Kostadin Nevrokopliev and Svetoslav Enkov

April 9, 2022

Framework for Composing Aggregate Reports in Bi Modules

Kostadin Nevrokopliev[1] and Svetoslav Enkov [1]

[1] *Paisii Hilendarski University Of Plovdiv, 24 Tsar Assen Str., Plovdiv, Bulgaria*

Abstract

This paper describes a framework that allows DTO objects to be grouped by data from certain fields. The result is a collection of objects containing summed and averaged data. The goal of this framework is to enable the composition of analytical reports that have high efficiency in making important managerial decisions. The framework is created in the C# programming language using the .net platform. All the bottlenecks and their solutions are described. Sample reports composed by it are presented, with emphasis on how each is useful to managers and which questions it can answer.

Keywords

Business Intelligence, Aggregate Reports, Data Analysis, Software Framework

Introduction

Business intelligence is a set of software products and technologies, that are crucial for the decision-making process in corporate companies. They support managers in multiple business processes that include “pricing”, “product positioning”, “customer acquisition”, “discovery of liabilities”, “trimming unnecessary expenses”.

To achieve this, BI software producers have multiple tools at their disposal. Some of the methodologies that those tools use include:

- Reporting;
- Online analytical processing;
- Data mining;
- Predictive analysis and many others.

Grouping reports give a summary of business data that helps managers track changes in the company's business over time. The resulting information is often described as aggregated data.

What is aggregated data?

“Aggregate data”, is the result of the secondary processing of individual data. It is obtained by grouping some fields and processing the data of others so that they can be summarized. This is achieved by applying “aggregate functions”. Some of the most commonly used are:

- Average
- Count
- Max
- Min
- Range
- Sum

Proceedings Name, Month XX–XX, YYYY, City, Country
EMAIL: nevrokopliev@gmail.com (K. Nevrokopliev); svetoslav.enkov@gmail.com (S. Enkov);

Aggregated data is higher-level data that is the result of processing lower-level data. In most cases, it finds application in decision making, comparing data from different time periods, examining trends, statistical analysis. Aggregated reports have the ability to reveal information insights, that may be observable without being grouped.

Developing a Framework for composing aggregate reports

The aim of this paper is to describe the process of developing a framework that converts a business layer object into aggregated data. The language which will be used is C#. For that reason, the framework will be aimed at the Microsoft®.NET platform.

The framework is part of the Business Logic Layer(BLL) of a software application and is meant to process data that is either from the Data Access Layer(DAL) or Data Transfer Object(TDO) entities. It will be able to group certain fields in an entity while applying different aggregate functions to others. The resulting data would be susceptible to further processing like sorting.

The resulting set will be composed of a collection of objects from an anonymous class. They can be passed directly to the presentation layer by serialization.

The name of the framework is "Granary" (Grouping, analyzing, arranging information).

3.1. Challenges of the development

C# is a class-based object-oriented language. Most of its objects are instances of a class. Because of the nature of those reports, the resulting data of an aggregated report can have different structures depending on the grouping fields and aggregate functions applied to it. For that reason, they should return objects of an anonymous class. The most fitting data structure in C# for those objects is "dynamic".

To compose aggregated data from business objects it is necessary to specify which fields will be grouped and which will be passed to aggregate functions. To achieve this functionality we use the builder design pattern. In the builder class, there is a collection of grouped fields and multiple collections for fields that need to be processed by different aggregate functions. The fields are stored in the following data structure - *IEnumerable<Expression<Func<TBusinessObject, object>>>*. This makes it possible to add properties from the business object to be processed.

The process of composing aggregated data from the business object requires the composition of a dynamic LINQ query in which the fields that will be processed and need to be described at runtime. To achieve this behavior we use "Dynamic LINQ" library which gives us the ability to compose the query into a string and then apply it to an object of type IQueryable.

3.2. Architecture

The core of the model consists of three classes.

The first one - "BusinessObjectSet" encapsulates the collection of business objects. The following functions help manage the set:

- void AddBo(TBo);
- void AddBoRange(IEnumerable<TBo>);
- TBo Remove(TBo);
- TBo RemoveAt(int);
- int Count;
- void Clear();

The main objective of "*BusinessObjectSet*" is to compose an aggregated report. This is performed by the *GroupBy()* method. As an input parameter, it receives an object of type *GroupFieldBuilder<TDto>*.

The second important class is *GroupFieldBuilder<TDto>*. It is developed using the builder software pattern. It consists of multiple collections of get properties of the DTO object. The first collection *_groupFields* contains those fields that will be grouped. The remaining collections are:

- *_sumFields;*
- *_avgFields;*
- *_countFields;*

As it can be seen from the above list, the current version of the framework supports aggregate functions for - summing, averaging, and counting. Additional functionalities can be added in future versions without changing the contents of the class itself. This way the "open-closed" principle will be respected.

Applying the framework to a real software

The framework will be implemented into "X3M ERP" business software. Its Business Intelligence module (BI) contains reports with individual-level and aggregated data. We will be using the framework in the following reports:

- Sales;
- Sold Products
- Payments and Income

4.1. Sales Report

The "Sales" report carries information on document turnover and how it affects revenue. It helps to explore the state of the market and detect declines in turnover during certain periods. This makes it possible to analyze and eliminate the reasons for their occurrence.

This sample report is set to visualize the following fields of the Sale business object:

- Sale ID;
- Date And Time;
- Customer Name;
- Amount (VAT not included);
- Total (VAT included).

This is an example of data exported from it.

Table 1
Sales Filtered by Date

Date	Client	Amount	Total
12.11.21	Sunny	10.00	12.00
12.11.21	Mambo 5	16.50	19.80
13.11.21	GigaF	13.45	16.14
13.11.21	Sunny	120.30	144.36
14.11.21	Mambo 5	15.50	18.60

The report can be grouped by: "Date", "Month", "Year" and "Customer". The "Amount" and "Total" fields will be summarized and averaged.

4.1.1. Grupped by Date

Here is an example of an aggregated report grouped by date and selecting the summed and averaged “Total” field.

Table 2
Sales Grouped by Date

Date	Count	Amount	Total
12.11.21	2	31.80	15.90
13.11.21	2	160.50	80.25
14.11.21	1	18.60	18.60

This way the manager can track the best and worst turnover days. This will enable him to analyze the reasons for their occurrence and optimize processes.

Additionally, the sprite can be filtered to affect a given time span or sum / average total range. This way it can monitor:

- sales turnover movements over a period of time;
- most / least profitable days.

4.1.2. Grouped by Customer

Alternately the report can be grouped by client. This is the result of aggregating the report from Table 1:

Table 3
Sales Grouped by Clients

Client	Count	Amount	Total
GigaF	1	16.14	16.14
Mambo 5	2	19.12	38.40
Sunny	2	78.18	156.36

The report described in Table 3 gives managers the opportunity to learn about their customers' behavior and spot those that bring in the most revenue. Having a field for average value and number of sales helps to answer the question of whether large but fewer in number transactions result in higher profits than small but multiple in number. This enables managers to best profile their customers.

4.2. Report on Items Sold

The reports concerning the goods that are sold serve to analyze how successful they are sold. They help managers and marketing experts to analyze the quantitative and financial turnover of each commodity as well as their movement over time. With their help, it can be determined which commodities are experiencing increases and decreases in demand during certain periods.

The fields that must be present in such a report are: sale number, item name, quantity, unit price and total price. A customer field has also been added to the report in Table 4. It allows us to analyze which customers are interested in which of our products. This enables managers to manage a customer's trade discounts for certain goods against the quantity they have purchased. Such analyses are applicable in companies that have the practice of registering their clients.

Table 4
Items Sold Filtered by Date

Goods	Customer	Quantity	Unit Price	Total
Sugar	Waffle World	1000	0.43	430.00
Cocoa	Choco Life	5200	2.78	14456.00
Cocoa	Waffle World	1200	2.78	3336.00
Sugar	Delta Candy	1320	0.43	567.60

Note that the data in Figure 5 were extracted over the period of one day. For this reason, the date field has been suppressed.

4.2.1. Sold Items Grouped by Product

By grouping the report by commodity, the quantities sold and the turnover of each commodity becomes clear. Most often in this report, a sales time interval filter is applied. This makes it possible to monitor how demand for the commodity is moving over time. This gives managers insights on the following issues:

- Most frequently sold products. This indicator helps new firms to best define their product niche. For firms that have already established themselves in the market, it may indicate an increase in demand for some goods that have not been in demand before.
- Commodities with the highest monetary turnover. There are many cases in which items with low quantity turnover result in higher financial returns than those with higher quantities sold. Many managers find it difficult to compare the two types of goods. This type of reference is useful for just that analysis. To be maximally effective it should be done over a minimum period of two months.
- Goods with small quantities sold.

All those reports are different variations of “Sold Items Grouped by Product”. They can be created simply by sorting on the correct field.

One of the most effective methods for such analysis is the so-called "Previews Period" Report. It can compare sales by commodity for previous periods such as quarters, past years, previous month, etc.

Table 5
Items Sold Filtered by Date

Goods	Sum Quantity	Avg. Unit Price	Sum Total
Cocoa	6400	2.78	17792.00
Sugar	2300	0.43	997.60

New Features Planned for Future Versions

These functionalities are sufficient to publish an initial version of the framework. This will comply with the "Minimum Valuable Product" publishing rule described in the book "The Lean Startup".

This is a list of features planned for the next sprint:

- Serialization module supporting XML and JSON.
- Additional date aggregation options, including group by week, month, and year.

Conclusion

The framework was successfully implemented into the BI module of “X3M ERP” and improved the architectural design of its business logic layer. Many other aggregated reports can be implemented with its help. This reduces the time required to implement new requirements by the “Product owner”.

One of the main goals of software frameworks is to achieve the so-called "inversion of control". By doing so, they create constraints and rules that protect programmers from errors during the development of a module. The framework described in this paper takes business objects as input and creates aggregated data from them. A Builder object is used to describe which items are to be grouped and which aggregated. This prevents incorrect composition of the resulting structure.

References

[1] K. Cwalina, J. Barton, B. Abrams, Framework Desing Guidelenes: Conventions, Idioms, and Patterns for Reusable. NET, 3rd edition, Addison-Wesley Professional, 2008.

[2] Dr. D. Delen, E. Turban, R. Sharda, Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th edition, Pearson, 2017.

[3] J. Albahari, “C# 9.0 in a Nutshell: The Definitive Reference”, 1st edition, O'Reilly Media, 2021.

[4] W. McKinney, Python for Data Analysis & Data Wragling with Pandas, NumPy, and IPython, 2nd edition, O'Reilly Media, 2017.

[5] Sherman R., Business Intelligence Guidebook: From Data Integration to Analytics, 1st edition, Morgan Kaufmann, 2016.

[6] C. Albright, W. Wilson, Business Analytics: Data Analysis & Decisions Making, 6th edition, Cengage Learning, 2014.

[7] G. Blokdyk, Aggregate Data A Complete Guide, 2020 edition, 5STARCooks, 2021.

[8] B. Perkins, Beginning C# and .NET, 2nd edition, Wrox, 2021.

[9] G. Baptista, F Abbruzzese, Software Architecture with C# 10 and .NET 6: Develop software solutions using microservices, DevOps, EF Core, and desing patterns for Azure, 3rd edition, Packt Publishing, 2022.

[10] en.wikipedia.org, Business intelligence, URL:https://en.wikipedia.org/wiki/Business_intelligence