Юбилейна международна научна конференция „Компютърни технологии и приложения“, 15-17 септември 2021 г., Пампорово, България
Anniversary International Scientific Conference “Computer Technologies and Applications”, 15-17 September 2021, Pamporovo, Bulgaria

PROGRAMMING WITH F# LANGUAGE FOR BUSINESS INTELLIGENCE

Veselina Naneva[1]

[1] *Faculty of Mathematics and Informatics,*
University of Plovdiv “Paisii Hilendarski”, Plovdiv, 236 Bulgaria Blvd.
Email: vnaneva@uni-plovdiv.bg

Abstract. Business Intelligence refers to the process of collecting, storing, analyzing and visualizing data, produced by a certain company’s activities. There is a defined five-level architecture, which includes all the tools and methodologies – from the processing of the information to its actual presentation. Additionally, a variety of programming languages, related to data science, can be specified, which fully satisfy each Business Intelligence level requirement. The focus of this paper is on the application of F# programming in data visualization and what is its impact on the end interactive report.

Key Words: *Business Intelligence, BI architecture, functional languages.*

Introduction

In the era of digitalization, each business produces a large amount of information, which needs to be stored and transformed in order both the quality of the product and the service offered to be improved. This in turn requires a well systemized process of data visualization to be followed. The sphere that provides a huge impact on manipulation of the data is Business Intelligence (BI). Plenty of methodologies and software applications satisfy such kinds of BI requirements but their practical usage depends on the size, type or processing level of the data. Due to the fact that each data-driven business has its own structure, firstly the characteristics of this BI architecture need to be determined by the help of specified individual stages. In this case, the general concept of it can be divided into three main ones – transforming, storing and visualizing [1].

The aim of transforming level (TL) is to provide systematized data even if the collecting sources are different. Taking into account what exactly is the business scale, we can expect that the information will be extracted in a structured, semi-structured or instructed way. On the other hand, the corresponding result may depend on the fact what software solutions are implemented and what are their capabilities with respect to the tools for data extraction. For instance, some companies count on hybrid data entering, i.e., simultaneously managing some software products and storing information into local files. In addition, processing steps should be followed to prepare data for the next actions. As an end product from the transforming level, we should be able to have fully structured data with divided information into logical tables. This in fact means that TL is closely related to providing relational databases with formed data type and cardinality. In this segment of BI, a combination of cloud-based solutions and local technologies for data manipulation can be used.

After recalculations and data transformation, the ready DB tables can be moved to an appropriate storage with regard to their size and complexity. Additionally, one of the most common requirements related to BI is data to have as much dynamic as possible. Therefore, an implementation of a cloud-based storage solution is meaningful in every case. Moreover, there are plenty of tools and pre-defined strategies so that the subsequent data loading is consistently on time.

When the business data has been processed, i.e. transformed and load levels have been applied, the next step covers the data visualization. Following commonly used BI principals, in this stage we should specify what kind of business-related questions need to be answered and what is the best way to be presented. As an example, it could be given a bar and column chart. This type of visualization is well known in most BI tools since it is suitable for organizing specific values across different categories.

Although in each mentioned stage many well-known software tools or environments can be used, it is possible that an application of a personalized solution could become obligatory to a successful data analyzation. This additional feature could be related to data science, i.e., by the help of different programming languages to define algorithms for optimized source transformation or to implement a custom visualization in the BI interactive tool.

The end product of every BI process flow is a dynamic report. Regardless of the type of the development, such kind of systemized information is a base tool for data analyzation. The necessity of quality decision making requires the ready dashboard to be accessible not only for the developer, but also the different company levels. Additionally, it should contain a well-structured business story, which needs to satisfy the client requirements and to provide interactions between the data, the time and the user.

Tools and Technologies in BI

To be more precise in describing the importance of customization in the BI end products, it is necessary to define what is the actual impact on it with regard to the most used program languages and tools.

As a first example of data science language, closely related to statistical computing and providing environments for graphical visualization, it could be considered R language. It provides a variety of techniques including linear and nonlinear modeling, time-series analysis and machine learning algorithms solutions.

Additionally, Python is a language which can be used in a combination with R or independently for data science and data visualization. Surely, there are several libraries with specific purposes, which can allow the access to different features such as NumPy – a fundamental Python package, used generally for scientific computing, and Matplotlib – library suitable for creating graphs and visual representation of the data.

The most common data science technology is JavaScript. Along this line of assumption, its library named D3 could be pointed out as a tool for not only creating custom solutions, but also it provides a variety of ready visual elements. Of course, there are many other JS libraries, which are closely related to data manipulation such as TableTop, which is intended for parsing of CSV data and jStat which is suitable for statistical data analysis. Because this technology is well known as web-driven, it is easy to create content and run it live into any browser platform.

Based on the necessity of a user-friendly end product of BI processing and visualization, there are several BI software products created such as Tableau and Power BI. They may provide similar features, but differ in the report creation sequence. For instance, Power BI is a complex environment for data transformation on a structured level and its presentation by the help of handy visual elements. Its core strength is based on the powerful combination of Data expression language (DAX) and M Language for data preparation. The platform is accessible by Desktop, Mobile and Service versions.

On the other hand, Tableau requires all of the features to be divided into working books. The BI tool provides a developing environment and user service one in order to slit the processing flow. The preparation and processing of data rely on Multidimensional Expressions (MDX), which is a query language, and on Basic calculations, Table calculations, and Level of Details expressions with respect to calculations. Tableau comes along with various access points such as Tableau Desktop, Tableau Public, Tableau Server, Tableau Online and Tableau Reader, which provide different features with respect to modification, access and share of the ready reports.

Programming with F#

As it was mentioned in the introduction, in some cases additional changes or development are required in order to create a meaningful BI end product. Surely, there are a lot of programming languages, which cover more than one BI aspects at once, but in this paper some of the advantages of F# language will be emphasized [2].

F# is formed as a functional-first and multi-paradigm programming language which also is strongly typed. Additionally, it is known as a general purpose so it is designed for building a variety of solutions. There are several integrated packages, related to data science that could be used in order to precise data transformation. FSLab can be pointed out as an example of such a feature [3]. This is the first data science package collection, which provides not only access to common data formats, but also the implementation of statistical testing, linear algebra and machine learning processing is available.

Firstly, we may consider the FSharp.Data package [4]. It implements type providers and tools for parsing CSV, HTML and JSON file format. From the BI point of view, parsing through different formats is one of the most important data manipulating steps due to the fact that the information flows could be taken from various sources. For instance, if we need to process a CSV file, we need to take into account that usually the first row in the file is related to the header, i.e. the naming of the columns, and then the actual data records follow. When we want to process the information and as a result to receive structured data, we need to use CsvProvider to get the strongly typed view. It can include several parameters such as location of the CSV file, whether the sample contains the names of the columns as its first line, column delimiter(s), etc.

```
    Type          Weather=          CsvProvider<"../data/Weather.csv",
ResolutionFolder=__SOURCE_DIRECTORY__>
```

Code 1. CsvProvider example

The generated type provides two methods for data loading. The parse method can be used if we have string values or to load not only row data, but also an URL result.

If we suggest using a cloud provider with F# in order to improve the dynamic manipulation of the data, we can do this with Azure Functions in the Azure Environment. This solution gives an easy way to run small pieces of code and the created logic could be connected with the storage services of Azure, such as Azure Blob Storage, Azure File Storage, etc.

Additionally, by the help of this package collection, we can easily implement some libraries for data visualization such as Plotly.NET and others. To illustrate this, we will consider some examples of data preparation based on the FSharp.Stats multipurpose project [5].

```
    open FSharp.Stats
    let fromFileWithSep (separator:char) (filePath) =
        seq {  let sr = System.IO.File.OpenText(filePath)
                while not sr.EndOfStream do
                    let line = sr.ReadLine()
                    let w = line.Split separator//[|',';' ';'\t'|]
                    yield w }
    let lables,data =
        fromFileWithSep ',' (__SOURCE_DIRECTORY__ + "/data/example.csv")
        |> Seq.skip 1
        |> Seq.map (fun arr -> arr.[4], [| float arr.[0]; float arr.[1];
float arr.[2]; float arr.[3]; |])
        |> Seq.toArray
        |> Array.shuffleFisherYates
        |> Array.mapi (fun i (lable,data) -> sprintf "%s_%i" lable i, data)
        |> Array.unzip
```

Code 2. FSharp.Stats example

If we take a look in Code 2, we can use F# with its package FSharp.Stats to implement algorithms for clustering. First of all, it defines the delimiters and the type of file source. After that by predefined methods it specifies the logical sequences for solving a specific business related problem. By the help of evoking **shuffleFisherYates**, it is possible to generate a random permutation of a finite sequence [6]. It continually determines the next element by randomly drawing it from the array until there are no elements left. If we want to visualize data for the business, which is classified by several categories, but there are no unique labels to be used, it is necessary to proceed the data in such a way.

To combine the data preparation with the actual visualization, it could be considered the Plotly.NET package, which provides functions for generating and rendering plotly.js charts in .NET programming languages. The main design philosophy of Plotly.NET is based on the following visualization flow:

- Initialize a GenericChart object from the data that is required to be visualized by using the respective Chart.* function, optionally setting some specific style parameters.
- Further style the chart with fine-grained control such as setting axis titles, tick intervals, and so on.
- Display (in the browser or as cell result in a notebook) or save the chart, and respectively export it in a form so that it can be additionally implemented in an appropriate BI tool.

To visualize a certain data (recall the example.csv file from Code 2) with Plotly.NET, it is necessary to be installed in a similar way as all of the packages in the FsLab collection [7]. Then, it should be followed the flow considered in the following rows:

```
open Plotly.NET
let colnames = ["Example 1";"Example 2";" Example 3";" Example 4"]
let colorscaleValue =
    StyleParam.Colorscale.Electric
    let dataChart =
    Chart.Heatmap(data,ColNames=colnames,RowNames=
(lables |> Seq.mapi (fun i s -> sprintf "%s%i" s i
)),Colorscale=colorscaleValue,Showscale=true)
      |> Chart.withMarginSize(Left=250.)
```

Code 3. Plotly.NET example

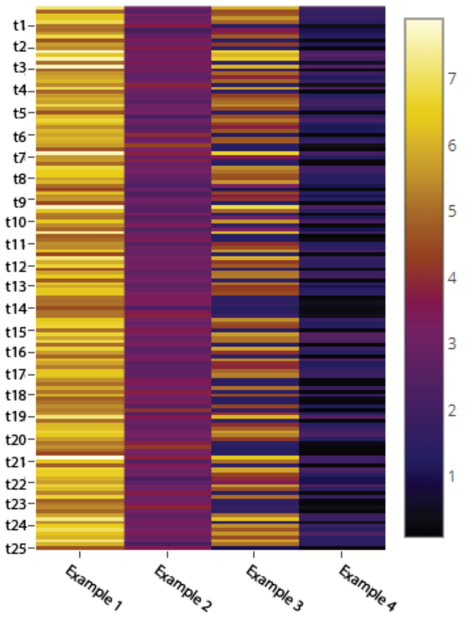

Figure 1. Configured visual example

As it could be seen from Figure 1, there is a defined color scheme by the StyleParam.Colorscale.Electric function, which has an impact on the entire visual element [8]. Additionally, it uses Heatmap, which is a specialized chart that operate with colors to represent data values in a table. If we need to run the end result, different plugins can be installed in the developing environments such as .NET Interactive Notebook. The visualization could be embedded in the interactive report by following the specific tool flow for setting the environment or preparing it for web presentation by low-code dash cloud, provided by Plotly.

Moreover, XPlot is another package of F#, which is powered by the Plotly chart function collection [9]. The XPlot library can be used interactively from F# Interactive, but charts can easily be embedded either in F# applications or in

HTML reports. As opposite to the other library, we can visualize not only Plotly related charts, but also a collection of elements from Google charts and D3.js.

The following example (see Code 4) uses Google Charts library and visualizes the data by the help of Combo and Line charts. Xplot can be manipulated by F# language for graphic configuration. Additionally, functions such as Chart.WithLabels and Chart.WithLegend specify the annotations in the individual chart series. The cross-platform data visualization package works also with predefined style layout, which can be referred by operator "|>" and function Chart.WithLayout.

```
open XPlot.GoogleCharts
let series = [ "bars"; "bars"; "bars"; "lines" ]
let inputs = [ Example1; Example2; Example3; Example4]


let chart2 =
    inputs
    |> Chart.Combo
    |> Chart.WithOptions
        (Options
(title = "Example",
                series = [| for typ in series
-> Series(typ)
|]))
    |> Chart.WithLabels ["Example1"; " Example2"; " Example3"; "
Example4"]
    |> Chart.WithLegend true
    |> Chart.WithSize (600, 250)
```

Code 4. Xplot example

Since F# language is functional based, it provides the possibilities to implement changes not only on data transformation level, but also includes features for visualization. In addition, it is known as a combination of JavaScript packages and C# object-oriented programming methods. That is why it is fully applicable to the BI architecture in each of its stages.

Conclusions

Business Intelligence is usually characterized by simultaneous data preparation and its dynamical visualization. There are a variety of software tools and strategies, closely related to all the sub steps that are connected with transforming, loading and presenting the business data. The focus of this paper is on F# language, which is open-source, cross-platform, and interoperable programming language used for data science and for visualization. Additionally, some code examples for data transformation are considered as well as two data visualization packages are suggested.

Acknowledgments

This paper is partially supported by project MU21-FMI-009 of the Scientific Research Fund of Plovdiv University “Paisii Hilendarski”, Bulgaria.

References

[1] I. Ong, P. Siew, S. Wong, A Five-Layered Business Intelligence Architecture, *Communications of the IBIMA*, Vol. 2011, 2011, Article ID 695619, 11 pages, DOI: 10.5171/2011.695619, ISSN: 1943-7765.

[2] T. Petricek, G. Guerra, D. Syme, Types from data: making structured data first-class citizens in F#, *ACM SIGPLAN Notices*, 51 (6), 2016, 477–490, DOI: http://dx.doi.org/10.1145/.

[3] https://fslab.org/packages.html, retrieved on July 2021.

[4] https://github.com/fsprojects/FSharp.Data/, retrieved on August 2021.

[5] https://fslab.org/FSharp.Stats/, retrieved on July 2021.

[6] https://www.geeksforgeeks.org/shuffle-a-given-array-using-fisher-yates-shuffle-algorithm/, retrieved on August 2021.

[7] https://plotly.net/, retrieved on July 2021.

[8] https://plotly.com/javascript/colorscales/, retrieved on July 2021.

[9] https://fslab.org/XPlot/, retrieved on July 2021.