Techniques Display Data in A Visual Database Programming Language

Agung Brastama Putra*, Siti Mukaromah, Mohamad Riza Ramadhani Oding, Muhammad Rizky Rahmatul Anam

Department of Information Systems, Universitas Pembangunan Nasional "Veteran" Jawa Timur, Indonesia

*Corresponding author:
E-mail: agunghp.si@upnjatim.ac.id

ABSTRACT

Many techniques of presenting information in programming languages, mainly for data that amounts to less than 100 record data. Moreover, if the data has reached more than 10,000 data, application performance will slow down. This has several factors, one of which is the process of the way data is presented by the code/algorithm in a programming language. The purpose of this paper is to show the effects of comparisons of code/algorithms in presenting data, expected outcome of this comparison of code/algorithm is to demonstrate increasing code/algorithm can present data faster.

Keywords: Desktop programming, visual database, display data, visual programming

Introduction

Science programming is increasing and evolving. Similarly, visualizing data pro-cessed by the result of the application pro-cess, using drill-down or and what-if analy-sis (Putra et al., 2019; Putra et al., 2019; Putra et al., 2020). Programming is a basic skill in computer science; logical thinking is needed to solve problems. In another paper, data collection was carried out by interview (Mukaromah et al., 2018; 2020). In software requirements, data is an important item because it is used by an analyst for describes and views all relationships of objects and information (Pressman, 2010).

Several software development articles use data or track information contained in databases with a limited amount of data. (Nugroho & Putra, 2016; Priambodo et al., 2017) or using a data dummy. This can im-pact the outcomes of software development implementation, the implication is that if you use less than 1000 records of data, then the application or program does not look at any difference in terms of the data pro-cessing capacity of the application, the data display process, and the overall application performance.

Performance testing is used to detect performance issues that could arise from a lack of server-side resources, insufficient network capacity, insufficient database ca-pacities, and defective or weak operating system capacities, poorly built web or desk-top application features, and other prob-lems with hardware or software that can lead to reduced performance of the client-server. The aim is two namely: (1) to under-stand how the system reacts as loading in-creases (i.e., num-ber of users, number of transactions, or total volume of data) and (2) to collect metrics that will lead to per-formance improvement design changes (Pressman, 2010).

Based on the above, this paper provides creation and input on how to show data over 5000 data track records with the ap-propriate algorithm and will be compared to displaying data below 100, the result ob-tained is the speed at which data above 5000 records/data line is displayed. The first experiment uses data that is less than 100 rows of information in this paper using two experiments, and the volume of data above 5000 is used for the second experiment.

How to cite: Putra, A. B., Mukaromah, S., Oding, M.R.R., & Anam, M.R.R. (2021). Techniques display data in a visual database programming language. 5th International Seminar of Research Month 2020. NST Proceedings. pages 93-99. doi: 10.11594/nstp.2021.0914
Material and Methods

This section will describe software development using prototype methodology and why use it.

Why use prototype methodology

Both the functions and features needed for the software being created are not specified by the client. In other situations, the developer could be uncertain as to the feasibility and ability of the algorithm that applied the adaptability of the operating system, or the shape of the human-machine interaction. A prototyping paradigm could give the best approach in these, and many other situations. Although prototyping can be implemented as a stand-alone process model, it is more commonly used in the sense of some of the process models described in this chapter as a technique that can be implemented. The prototyping model allows you and other stakeholders to better understand what is to be constructed when specifications are fuzzy, regardless of the way it is implemented (Pressman, 2010).

Prototype methodologies

Figure 1 shows the iteration of the methodology of the prototype and a description of each step is given below (Bourque & Fairley, 2014):

Communication

In this stage, identifying the requirements requested, and describing the areas that must be implemented, in this paper the communication stage is displaying data with large amounts of data with fast results.

Quick plan

Quick plan means that the developer must make plans according to user needs and make display designs and iterations quickly.

Modeling and quick design

The focus of the quick design is on portraying the facets of the program that will be used by the end-user. Quick design leads to a prototype being built.

Construction of prototyping

In this stage, start implementing the results of the design into an application. In this paper, the construction stages are divided into 2, namely: 1. conducting experiments with 20 times iteration with the number of data below 20 lines, 2. using data more than 5000 lines.

Deployment, delivery, and feedback

The expected results from experiments 1 and 2 are that the algorithm applied can display data faster than other algorithms.
Results and Discussion

This section will describe the result with experiment 1 and experiment 2:

**Experiment 1**

In the First experiment using 17 rows of data and performed 50 iterations, in Figure 2 we can see time execution Object-2 better than Object-1, with several data Object-1 between Object-2 is the same record. Source code comparison between Object-1 (Figure-3) and Object-2 (Figure-4).
In Figure 3, the source code is indeed less when compared to Figure 4, but the execution time takes much longer. This is because the source code in Figure 3 uses the syntax `ObjectGrid1.DataSource = ds.Tables("table").DefaultView` means that all data is immediately displayed and executed, while in Figure 4 the source code is indeed more but the result of execution time is faster.

Object 2 has an average execution time of 111ms and it's slightly lower than Object 1 (114ms) (See Figure 5). Object 1 tends to increase its execution time when compared to Object 2. Object 1 has a maximum execution time value of 121ms with a minimum value of 110ms, while Object 2 has a maximum execution time value of 119ms and a minimum value of 106ms, for more details can be seen in the Figure 6 about graph time execution.

```vbnet
Function obj1() As Double
    Dim ds As New DataSet
    Dim swatch As New System.Diagnostics.Stopwatch
    swatch.Start()
    System.Threading.Thread.Sleep(100)
    Dim sda As New MySqlDataAdapter(Query, conn)
    sda.Clear()
    sda.Fill(ds, "table")
    ObjectGrid1.DataSource = ds.Tables("table").DefaultView
    swatch.Stop()
    tm1 = swatch.ElapsedMilliseconds.ToString()
    watch.Text = tm1 & " ms"
    Return tm1
End Function
```

Figure 3. Source code Object 1

```vbnet
Function obj2() As Double
    Dim ds As New DataSet
    Dim swatch As New System.Diagnostics.Stopwatch
    swatch.Start()
    System.Threading.Thread.Sleep(100)
    Dim sda As New MySqlDataAdapter(Query, conn)
    sda.Clear()
    sda.Fill(ds, "table")
    DataGridView1.Rows.Clear()
    For k As Integer = 0 To ds.Tables("table").Rows.Count - 1
        ObjectGrid1.Rows.Add("", ).Cells(0).Value = ds.Tables("table").Rows(k).Item(0).ToString()
        ObjectGrid1.Rows[ObjectGrid1.Rows.Count - 2].Cells(1).Value = ds.Tables("pembeilian").Rows(k).Item(1).ToString()
        ObjectGrid1.Rows[ObjectGrid1.Rows.Count - 2].Cells(2).Value = ds.Tables("pembeilian").Rows(k).Item(2).ToString()
        ObjectGrid1.Rows[ObjectGrid1.Rows.Count - 2].Cells(3).Value = ds.Tables("pembeilian").Rows(k).Item(3).ToString()
        ObjectGrid1.Rows[ObjectGrid1.Rows.Count - 2].Cells(4).Value = ds.Tables("pembeilian").Rows(k).Item(4).ToString()
    Next
    swatch.Stop()
    tm2 = swatch.ElapsedMilliseconds.ToString()
    time2.Text = tm2 & " ms"
    End Dispose()
    Return tm2
End Function
```

Figure 4. Source code Object 2
Experiment 2

In the 2nd experiment, using data from more than 1000 lines, to be precise 9,523 rows of data and 16 Columns. The results obtained after running a query on Object-1, with a nested query or subquery, get the results of an execution time of 36.047ms with a total data of 9,523 lines. Whereas, with the same amount of data for Object-2 the execution time is 41733.

Figure 7. Comparison Execution Time Object-1 vs Object-2
select noreg, nomedic, idbuku, patient, address, religion, convert (umur, char) as age, date_format(tglmasuk, "%Y-%m-%d") datein, timein, lkm, clinic, IFNULL (kamarrawat, (select subquery4 where rekening=bb.klinikasal)) kamarrawat, ruang, kelas, keter, (select subquery3 where rekening=bb.klinikasal) namaklinik from (
    select aa.*,
    (select subquery1 where rp.kode=aa.kamar) room,
    (select subquery2 where rp2.kode=aa.lkm) inpatientRoom from
    select query level 1 from masterroom m right join roominap i on m.nomedic=i.nomedic
    join room k on m.noreg=k.noreg where lkm not in('963','064')
)

The query used in this paper is a level 2 query with each level having 2 subqueries (see Figure-8), a detailed explanation of the query can be seen in Figure 9.

Figure 8. Query for Object-1 vs Object-2

![Figure 8. Query for Object-1 vs Object-2](image)

Figure 9. Detailed explanation query to use

```csharp
Sub obj2(ByRef q As String)
    tm2 = 0
    swatch = New System.Diagnostics.Stopwatch()
    swatch.Start()
    System.Threading.Thread.Sleep(100)
   DataGridView2.Rows.Clear()
    Dim strSQLy As String = q
    Dim drFilterRow() As Data.DataRow = dsAll.Tables("patientAll").Select(strQuery, "noreg ASC")
    For i As Integer = 0 To drFilterRow.GetUpperBound(0)
        objectGrid.Rows(objectGrid.Rows.Count - 2),Cells(0).Value = drFilterRow(i).Item(0).ToString()
        objectGrid.Rows(objectGrid.Rows.Count - 2),Cells(1).Value = drFilterRow(i).Item(1).ToString()
        objectGrid.Rows(objectGrid.Rows.Count - 2),Cells(2).Value = drFilterRow(i).Item(2).ToString()
    Next
    swatch.Stop()
    tm2 = swatch.ElapsedMilliseconds.ToString()
    time2.Text = tm2 & " ms"
    rows2.Text = DataGridView2.Rows.Count - 1
End Sub
```

Figure 10. Object-2 source code snippets
Conclusion

The results of the study can be concluded that:
1. Based on experiments 1 and 2, this paper succeeds in displaying data quickly using the source code in Object-2.
2. The source code in Object-2 can be applied and used for amounts of data less than 1000 lines or more.
3. Displaying data more than 5000 rows with a query level of more than 1 and a subquery of more than 2, so it's better to use the algorithm on object-2

Acknowledgment

The author would like to thank LPPM UPN Veteran East Java for providing financial support in textbook grants and thank you for the support of fellow lecturers from the information systems department

References

Bourque, P., & Fairly (Dick), R. E. (2014) Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0).

Mukaromah, S., Kusumanatara, P. M., Putra, A. B., & Pratama, A. (2018). Analysis and Design Logbook Information Systems 1(Icst).

Mukaromah, S., Pratama, A., Ithriah, S. A., & Putra, A. B. (2020). Analysis and design student entrepreneurship information system. Journal of Physics: Conference Series, 1569(2), 113-120. doi: 10.1088/1742-6596/1569/2/022045.

Nugroho, B., & Putra, A. B. (2016). Sistem informasi pengelolaan administrasi tugas akhir di jurusan teknik informatika UPN "Veteran Jawa Timur. Jurnal Sistem Informasi Dan Bisnis Cerdas (SIBC), 9(1), 25–34.

Pressman, R. S. (2010). Software engineering: A practitioner's approach 7th Edition. McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020.

Priambodo, B., Wibowo, N. C., & Putra, A. B. (2017). Sistem pengelolaan data barang berbasis desktop. Jurnal Sistem Informasi Dan Bisnis Cerdas (SIBC), 10(2), 19. Available at: https://www.nesabamedia.com/pengertian-aplikasi/.

Putra, A. B., Hanafi, R., Maulana, A., & Falah, N. (2019). Aplikasi Sistem Informasi Eksekutif Dengan Fasilitas Drilldown Dan Analisis What-If. Jurnal Sistem Informasi dan Bisnis Cerdas, 12(1), 41–50. doi: 10.33005/sibc.v12i1.1587.

Putra, A. B., Gemilang, C. C., Fajar, M. B., & Pasya, A. M. K. (2019). Sistem informasi eksekutif stok daging sapi di Pulau Jawa. Prosiding Nasional Rekayasa Teknologi Industri dan Informasi XIV, 2019(November), pp. 291–295.

Putra, A. B., Mukaromah, S., Agussalim, Lusiarini, Y., Risky, M. I., & Bestari, P. Y. (2020). Design and Development Executiv Information System Application with Drilldown and What-If Analysis features. Journal of Physics: Conference Series, 1569, 022050. doi: 10.1088/1742-6596/1569/2/022050.