Performance analysis framework for a complex financial data warehouse

Citation for published version (APA):

Document status and date:
Published: 28/09/2016

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.
Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Performance Analysis Framework for a Complex Financial Data Warehouse

Vladimir Mikovski Iotov
September 2016
Performance Analysis Framework for a Complex Financial Data Warehouse

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

The design described in this report has been carried out in accordance with the TU/e Code of Scientific Conduct.

Partners

Rabobank
Eindhoven University of Technology

Steering Group
Mark Verwijs (Rabobank)
Jurgen Krosse (Rabobank)
Peter Onvlee (Rabobank)
Dr. Serguei Roubtsov(TU/e)

Date
September 2016

Document status
Public
Abstract

This report describes the project that analyzes the performance of a data warehouse used for liquidity risk data aggregation in Rabobank. Furthermore, a methodological approach to performance evaluation is suggested as part of the testing phase. This method considers three sources of information: the hardware running the programs, the processes that support the execution of the system, and the system’s processes. Following that, a framework was outlined and built to support the method suggested. In conclusion, the results demonstrate the benefits of using the framework, bringing valuable new insights into the system’s performance.

Keywords

performance, analysis, financial, data, warehouse, evaluation, testing, methodological, framework, Rabobank, PDEng, TUe, software, technology

Preferred reference

PERFORMANCE ANALYSIS FRAMEWORK FOR A COMPLEX FINANCIAL DATA WAREHOUSE, SAI Technical Report, September 2016. (Eindverslagen Stan Ackermans Instituut ; 2016/056)

Partnership

This project was supported by Eindhoven University of Technology and Rabobank.

Disclaimer

Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology or Rabobank. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology or Rabobank, and shall not be used for advertising or product endorsement purposes.

Disclaimer Liability

While every effort will be made to ensure that the information contained within this report is accurate and up to date, Eindhoven University of Technology makes no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks

Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Copyright

Copyright © 2016. Eindhoven University of Technology. All rights reserved.

No part of the material protected by this copyright notice may be reproduced, modified, or redistributed in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the Eindhoven University of Technology and Rabobank.
Foreword

In the wake of the financial crisis in 2008, many of the external, regulatory bodies have – raised the bar – on the number, granularity and time windows of risk and finance disclosures. At Rabobank, particularly within the Risk & Finance domain, this growing regulatory pressure is becoming a serious challenge for the data warehouses that form the basis of regulatory reporting. The vast amounts of data that must be processed and prepared are hitting the boundaries of the data warehouse performance.

The project “Performance Analysis Framework for a Complex Financial Data Warehouse” is aiming to deliver a sound, scientifically based framework to evaluate the performance of a data warehouse. Besides testing the performance, the framework should also help the test manager to identify and analyze the most likely bottlenecks; please note that it should give insight in all areas, covering hardware resources, network activity, database performance, and coding of the transformation or business logic.

Vladimir’s work already led to identifying some of the bottlenecks, as well as the technical debt in one of our live data warehouses whilst he was building the framework. With those findings, Vladimir already demonstrated the benefits of a thorough, structured approach to performance analysis and testing.

We see the framework delivered by Vladimir as an important step towards the standardization of executing performance testing in the Risk & Finance department, an effort that is not to be underestimated. Now, it is up to Rabobank to adopt the framework and reap its benefits.

Jurgen Krosse
Architect
IT Systems and Development, Risk and Finance
September, 2015
Preface

This report describes the project conducted by Vladimir Mikovski Iotov as part of the Professional Doctorate in Engineering program in Software Technology. The project has been completed in nine months as a collaboration between Eindhoven University of Technology and Rabobank.

The project, named "Performance Analysis Framework for a Complex Financial Data Warehouse", aims to produce a framework in order to facilitate the performance evaluation of a data warehouse system.

This report is intended for readers with a technical background Software Technology, however other non-technical readers may find it interesting. The following table relates the different chapters with the types of readers.

<table>
<thead>
<tr>
<th>Reader</th>
<th>Chapters</th>
</tr>
</thead>
</table>
| Software Technology Engineer | Introduction  
|                       | Stakeholder Analysis  
|                       | Problem Analysis  
|                       | Domain Analysis  
|                       | Feasibility Analysis  
|                       | System Requirements  
|                       | System Architecture  
|                       | System Design  
|                       | Implementation  
|                       | Verification & Validation  
|                       | Deployment  
|                       | Project Management  
|                       | Project Retrospective |
| Performance Analysts  | Introduction  
|                       | Domain Analysis  
|                       | System Requirements  
|                       | System Architecture  
|                       | Implementation  
|                       | Deployment |
| Project Managers      | Introduction  
|                       | Stakeholder Analysis  
|                       | Problem Analysis  
|                       | Feasibility Analysis  
|                       | System Requirements  
|                       | Project Management  
|                       | Project Retrospective |

September 2016
Acknowledgements

I would like to thank to my supervisors from Rabobank and TU/e for their continuous dedication to me and my project, without them it wouldn't be possible to finish successfully. In particular I want to thank to Jurgen Krosse for his support and guidance that were crucial for me in order to find my way inside Rabobank. Peter Onvlee for his feedback and support on how to manage the project and Mark Verwijs for his sponsorship that made possible the project. Also I want to acknowledge the help from the specialists and colleagues from Rabobank, which inspired me and gave me the opportunity to work with many wonderful people.

Next, from the TU/e, I would like to express my gratitude to Dr. Serguei Roubtsov for his dedication and support, being always available to meet me and give me valuable advices. As well, I want to thank the help received from my colleagues from PDEng and the project director Ad Aerts. Especially to my colleague Dmitriy Ilin, for his technical and personal advices and the moments that we shared traveling to and within Rabobank.

Last but not least, I would like to thank my family for their moral support during the project. To my brother, Ivan Mikovski Iotov, for always being there for me when I needed to talk and discuss with someone. Finally, to my friends Johan Bertrand Bonnemason, Trajche Masinov, and Erikos Alkalai for putting up with my moods when I was frustrated with my work and the project.

September 2016
Executive Summary

At present, the banking sector needs to implement changes for the increasing number of new regulations as a consequence of the international financial crisis in 2007-08. It represents a challenge for traditional banks, such as Rabobank, that need to improve their existing IT landscape. Recently, the requirements for liquidity risk management have increased and the existing Rabobank’s solution needs to be upgraded and extended.

The main focus of this project is the system used for liquidity risk data aggregation in Rabobank, which is currently being enhanced to extend and improve its capabilities. These improvements could be measured with a continuous analysis on its performance. Furthermore, the performance analysis has to become part of the evaluation and testing of the system before its release to production. In order to achieve it, this project applied a methodological approach to performance evaluation that can be adopted as part of the testing phase.

An extended domain analysis was carried out on performance analysis for software intensive systems in this project. As a result of it, a systematic method to analyze a system like the one used for liquidity risk data aggregation in Rabobank was proposed. This method for performance analysis consider three sources of information: the hardware running the programs, the processes that support the execution of the system, and the system’s processes. Additionally, an investigation of the system was done to identify requirements and constraints for its later evaluation. Following that, a framework was outlined with the output of the domain and problem analysis to support the performance evaluation suggested.

The design of the performance analysis framework emphasizes the usability and genericity attributes of the solution. It follows a distributed architecture divided into building blocks that makes it flexible and extensible. An implementation of the framework, a prototype, was developed as part of this project. The prototype was used to produce a performance analysis of the system for liquidity risk data.

The prototype was used in a test scenario where Rabobank’s system and the framework are deployed in the same machine. The performance information was produced in the test scenario using synthetic data as test case. A wide range of metrics were used to collect data validating the functionality and flexibility of the framework. The data was analyzed with an interactive interface that allows drawing theories based on the behavior and performance of the analyzed system. In conclusion, the results obtained demonstrate the benefits of using the framework as part of the testing procedure, bringing valuable new insights into the system’s performance.

I recommend to test the framework in a production-like environment as a proof of concept. With some additional development, the framework implementation will be mature enough to be adopted as part of the test cycles of the system for liquidity risk data aggregation. Furthermore, the framework can be implemented for other systems. I suggest the system for credit risk data aggregation because of its similarities with the system for liquidity risk data.
# Table of Contents

Foreword ......................................................................................................................... i
Preface .............................................................................................................................. iii
Acknowledgements .......................................................................................................... v
Executive Summary ......................................................................................................... vii
Table of Contents ........................................................................................................... ix
List of Figures ................................................................................................................ xiii
List of Tables .................................................................................................................. xv

1. Introduction .................................................................................................................. 17  
   1.1 Context ..................................................................................................................... 17  
   1.2 Risk Data Aggregation and Reporting ................................................................. 17  
   1.3 Liquidity and LICWID program ............................................................................ 18  
   1.4 Performance analysis framework for a complex financial data warehouse .......... 18  
   1.5 Outline .................................................................................................................... 19

2. Stakeholder Analysis .................................................................................................. 21  
   2.1 Rabobank ............................................................................................................... 21  
   2.2 Eindhoven University of Technology (TU/e) .................................................... 21

3. Problem Analysis ....................................................................................................... 23  
   3.1 Context ..................................................................................................................... 23  
   3.2 Liquidity risk reporting ......................................................................................... 23  
   3.3 LiQDW design ......................................................................................................... 26  
   3.4 Performance Challenges ...................................................................................... 28  
   3.5 Design Opportunities ............................................................................................ 29

4. Domain Analysis ....................................................................................................... 31  
   4.1 Introduction ............................................................................................................. 31  
   4.2 Performance analysis ............................................................................................ 31  
      4.2.1. A systematic approach to performance evaluation .............................. 32  
      4.2.2. USE Method ............................................................................................. 33  
   4.3 LiQDW system analysis ......................................................................................... 33  
   4.4 Data Vault modeling .............................................................................................. 34

5. Feasibility Analysis .................................................................................................... 37
5.1 Issues and challenges ........................................37
  5.1.1. Change of project definition ............................37
5.2 Risks .....................................................................37
  5.2.1. Environment unavailable for performance analysis ..........37
  5.2.2. Test data unavailable for performance analysis ..........37
  5.2.3. Critical changes in LiQDW ................................37

6. System Requirements ..............................................39
  6.1 Introduction ....................................................39
  6.2 Main features and use cases ..................................39
  6.3 Functional requirements .....................................40
  6.4 Software quality attributes ..................................41
  6.5 Constraint requirements .....................................41

7. System Architecture ..............................................43
  7.1 Introduction ....................................................43
  7.2 Architectural reasoning .....................................43
  7.3 Technological constraints ..................................45

8. System Design .....................................................47
  8.1 Introduction ....................................................47
  8.2 Scenarios .......................................................47
  8.3 Logical view ...................................................47
  8.4 Process view ...................................................51
  8.5 Development view ..........................................54
  8.6 Deployment view ............................................55

9. Implementation ....................................................57
  9.1 Introduction ....................................................57
  9.2 Data Storage ...................................................57
  9.3 Data Collector ..................................................58
    9.3.1. Data Collector Manager .................................60
  9.4 Data Viewer .....................................................60
  9.5 Technology overview .......................................63

10. Verification & Validation ........................................65
  10.1 Introduction ...................................................65
  10.2 Local performance evaluation of LiQDW ....................65
  10.3 Validation ......................................................68

11. Deployment .......................................................71
  11.1 Test scenario deployment ...................................71
  11.2 Installation manual .........................................71
List of Figures

Figure 1 Reference Architecture simplified of LiQDW ........................................24
Figure 2 Architecture of LiQDW .................................................................24
Figure 3 Logical Data Architecture .............................................................25
Figure 4 SQLSpec stored procedure of Performance Analysis Framework ..........26
Figure 5 SQLSpec stored procedure of Performance Analysis Framework ..........27
Figure 6 SQL Dependency Tracker. Stored procedures LiQDW Metadata........27
Figure 7 SQL Dependency Tracker. Tables in Raw Vault, "Trade" selected........28
Figure 8 Systematic approach in ten steps for performance evaluation ..........32
Figure 9 Example of Data Vault model for Customer, Product, and Transaction (source: www.bluesoft.net.pl) .................................................................35
Figure 10 System level use cases view ..........................................................40
Figure 11 High-level abstract components model of the system ......................43
Figure 12 High-level deployment architecture ..............................................44
Figure 13 High-level architecture with Hot (red) and Cold (blue) spots ............45
Figure 14 Diagram with objects inside PerfDB ............................................48
Figure 15 Sequence diagram of PerfData structure generation ......................49
Figure 16 Logical structure to capture performance data ...............................50
Figure 17 Sequence diagram of collecting DMV data ....................................50
Figure 18 Sequence diagram of collecting LiQDW logging data .......................51
Figure 19 Logical structure to analyze performance data ..............................51
Figure 20 Activity diagram to capture performance data ................................52
Figure 21 Activity diagram to capture performance data asynchronously ........53
Figure 22 Activity diagram to analyze performance data ...............................53
Figure 23 Component diagram of the performance analysis framework ..........54
Figure 24 Schema diagram with an example of tables and views generated ......55
Figure 25 Deployment diagram of the performance analysis framework ..........56
Figure 26 SQL Server Management Studio (SSMS) object explorer on PerfDB .58
Figure 27 Windows Performance Monitor (PerfMon) with some metrics displayed .................................................................59
Figure 28 SQL Server Management Studio (SSMS) object explorer on the Agent with the jobs .................................................................60
Figure 29 Tableau chart comparing multiple processors usage and its average highlighted .................................................................61
Figure 30 Tableau chart with multiple metrics over one time line ..................62
Figure 31 Tableau charts with multiple metrics using dual axis ......................63
Figure 32 Tableau Gantt chart with the executions of LiQDW system .............63
Figure 33 Deployment diagram with the technology chosen .........................64
Figure 34 Chart of executions obtained from the experiment .........................65
Figure 35 Chart of processors' usage obtained from the experiment ...............66
Figure 36 Chart of processors' privileged time obtained from the experiment ...66
Figure 37 Charts of CPU information obtained from the experiment ...............67
Figure 38 Charts of RAM information obtained from the experiment .............67
Figure 39 Deployment diagram on a single machine .....................................71
Figure 40 Work-breakdown structure of the project .....................................75
Figure 41 Initial project planning .................................................................77
Figure 42 Final project planning .................................................................77
# List of Tables

Table 1 Rabobank's Stakeholders ................................................................. 21
Table 2 TU/e's Stakeholders ....................................................................... 21
Table 3 Functional requirements ................................................................. 40
Table 4 Non-functional requirements .......................................................... 41
Table 5 Constraint requirements ................................................................. 41
Table 6 Functional requirements status ....................................................... 68
Table 7 Non-functional requirements status ............................................... 68
Table 8 Glossary .......................................................................................... 81
1. Introduction

This chapter introduces the project, its context, and the ecosystem where it lives. The scope of the project and its goals are briefly mentioned. The outline section of this chapter gives a brief overview of what is discussed in the following chapters.

1.1 Context

At present, the banking sector in The Netherlands is being regulated by many governing bodies like Basel Committee on Banking Supervision (BCBS), De Nederlandsche Bank (DNB), European Central Bank (ECB), and European Banking Authority (EBA). In the last eight years, as a consequence of the international financial crisis in 2007-08, the number of regulations and their speed of adoption are continuously increasing. All this represents a challenge for traditional banks that need to improve their IT landscape.

The “Performance Analysis Framework for a Complex Financial Data Warehouse” project was conducted by the trainee, as part of his Professional Doctorate in Engineering (PDEng) program. The PDEng degree program in Software Technology is provided by the Department of Mathematics and Computer Science of Eindhoven University of Technology in the context of the 4TU.School for Technological Design, Stan Ackermans Institute.

The project was initiated by Rabobank, a Dutch multinational banking and financial services company. Rabobank is a global leader in agro-financing and sustainability-oriented banking. It has various software-related programs with the objective to improve existing and build new data warehouses. “Performance Analysis Framework for a Complex Financial Data Warehouse” project lies under the program Liquidity (LICWID) which was started in 2013 by Finance and Group Risk Data unit. A more detailed description and context of that project is stated below.

1.2 Risk Data Aggregation and Reporting

The Basel Committee on Banking Supervision (BCBS) provides a forum for regular cooperation on banking supervisory matters. Its objective is to enhance understanding of key supervisory issues and improve the quality of banking supervision worldwide. The Basel Committee created the Basel III framework [1][2] and the BCBS239 [3] principles to regulate the risk data management.

The BCBS239, "Principles for Effective Risk Data Aggregation and Reporting," is a set of principles to strengthen banks’ risk data aggregation capabilities and internal risk reporting practices. A robust and smart risk data aggregation enables high quality risk management reports. BCBS239 consist of 14 principles separated into four sections: "Overarching governance and infrastructure," "Risk data aggregation capabilities," "Risk reporting practices," and “Supervisory review, tools and cooperation.”

The term risk data aggregation means “defining, gathering and processing risk data according to the bank’s risk reporting requirements. It enables the bank to
measure its performance against its risk tolerance/appetite. Data aggregation includes sorting, merging and breaking down sets of data." [3] The key attributes of the risk data aggregation proposed by the BCBS239 are accuracy, integrity, completeness, timeliness, and adaptability.

Basel III is an international regulatory framework to strengthen the regulation, supervision, and risk management of the banking sector. Its scope is on bank capital adequacy, stress testing, and market liquidity risk and it complements the previous Basel II and Basel I frameworks. One of the main components is the Liquidity Coverage Ratio (LCR) aiming to ensure that a bank has an adequate stock of high quality liquid assets (HQLA), which consist of cash or assets that can be converted into cash easily. The other component is the Net Stable Funding Ratio (NSFR) that requires banks to maintain a stable funding profile in relation to the composition of their assets and off-balance sheet activities.

1.3 Liquidity and LICWID program
Liquidity risk is present in many different ways, it can appear in financial markets or in accounting, but in essence, it is the ability to quickly convert an asset into cash without affecting the asset's price. Thus, liquidity risk management is the control over the firm to meet its current and future cash flow (cash inflows vs. outflows). Financial firms are especially sensitive to liquidity risk based on their nature and activities. In response to this well-known risk, Rabobank initiated the LICWID program with the final objective to provide daily liquidity data on trade level for the entities' full balance sheets. Later, with the publication of BCBS239 principles and Basel III, the program had to adjust to the new guidelines.

Several projects are being performed under the umbrella of the LICWID program. Those projects are assigned to one or more teams responsible to realize them. The main system of LICWID program is LiQDW, a large data warehouse that aggregates financial data, cash-flows, and market data among others. LiQDW was designed following the Data Vault modeling technique in order to have a flexible and scalable data model that is aligned with BCBS239 principles of risk data aggregation.

The purpose of LiQDW is to support the analysis and reporting of liquidity risk. The financial analysts are able to access and analyze data from LiQDW in order to answer specific questions or drill deeper for more details given a result from a report. This type of analysis is called ad hoc analysis. The main components that use the aggregated data from LiQDW are NSFR monthly and LCR monthly and daily as stated in Basel III.

The LiQDW system relies on several source systems to obtain the necessary data for the data aggregation. The complexity of LiQDW resides in the variety of the availability and format of its source data and source systems. Following the principles of BCBS239, extra steps are introduced on top of the aggregation to improve the quality of the data received. The implementation of all these requirements has an important impact on the performance of the system.

1.4 Performance analysis framework for a complex financial data warehouse
The idea behind the project is to analyze the performance of LiQDW system, a data warehouse designed following the Data Vault modeling techniques. With Basel III and BCBS239, reports have to be produced faster, more often, and have to comply with new requirements, therefore LiQDW system has to upgrade and enhance its aggregation capabilities. Since the project's main focus is LiQDW system, it falls under the LICWID program.
The main goal of this project is to define a disciplined approach to analyze a data warehouse system like LiQDW. Building a framework that can be used during the performance testing will facilitate the analysis. The framework will allow to gather performance data that will help to identify possible bottlenecks and prevent other future challenges. Furthermore, the framework will enable the combination and visualization of the internal-process-execution metadata (LICWID processes) with the computing-platform-performance data (hardware, OS, and database engine).

Besides the main goal, the projects aims to produce recommendations based on the results from the analysis of LiQDW Data Vault. Additionally, since this is the first project of such nature carried out in Rabobank, the internal processes and standards are challenged by the trainee as part of the assignment.

1.5 Outline

The reports starts introducing the stakeholders involved in this project, their motivation and goals are identified. Next, the problem analysis (Chapter 3) is explained with the details about the problem to be solved during this project. Following that, the domain analysis (Chapter 4) present the research on the different topics derived from the previous chapter. The domains studied are performance analysis, data vault modeling, and LiQDW system design. As result, the feasibility analysis (Chapter 5) is presented where the risks are identified with their mitigation strategy.

The next six chapters explain the solution of the problem in a structured way. First, the system requirements (Chapter 6) are derived from the problem and domain analysis. Functional requirements, quality attributes, and constraint requirements are specified for the design of the solution. Then, the architecture (Chapter 7) is introduced where a system is delineated as the solution to the problem stated before. Following it, the design in more details of the system is explained in Chapter 8. Next, the implementation (Chapter 9) of the system is presented with the technological decisions. Chapter 10 discusses the verification and validation of the system design and its implementation. Finally, the deployment (Chapter 11) of the system is presented with the installation manual.

The conclusion about the project is given in the Chapter 12 together with the results and future work for the project. The Chapter 13 addresses the project management process followed during the nine months of the project. Last, Chapter 15 gives the author’s retrospective and reflection on the project.
2. Stakeholder Analysis

In the previous chapter, the project and its context was introduced. In this chapter, the stakeholders involved in the project are analyzed with their goals and interests.

2.1 Rabobank

Rabobank is a Dutch multinational banking and financial services company. It is a global leader in agro-financing and sustainability-oriented banking. The main stakeholders and initiators of the project are from Rabobank. They have several points of interest in the project.

First, they are interested in getting involved with educational institutions (in this case TU/e) and bringing an alternative opinion to their present challenges. They want a project that follows a more theoretical/methodological approach. They expect to obtain a different solution to the problem with an alternative process and an independent point of view.

Second, they want to improve the performance of the reporting system since it has to cope with new requirements. Therefore it is crucial to understand the current design, evaluate its performance, and ultimately bring new ideas and an alternative design.

Third, based on the first point and as an extension, they are interested in how a PDEng trainee will adapt to the company’s environment. They want the trainee to challenge the organization and its internal processes.

Table 1 Rabobank’s Stakeholders

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jurgen Krosse</td>
<td>Solution Architect</td>
<td>Company supervisor</td>
</tr>
<tr>
<td>Peter Onvlee</td>
<td>Project Manager</td>
<td>Project Manager</td>
</tr>
<tr>
<td>Mark Verwijs</td>
<td>Delivery Manager</td>
<td>Project Owner</td>
</tr>
</tbody>
</table>

Jurgen is the main stakeholder and the one fully involved with the project. Although he is not directly involve with LiQDW and LICWID program, his experience and guidance were very useful. Peter is an experienced project manager previously involve in LICWID program. During the project, Peter had a health issue that kept him away from participating in the last months of the project. Mark is part of the management team in charge of Finance and Group Risk Data. He is the sponsor of the project.

With Jurgen and Peter, a weekly update was organized in order to keep them informed with the progress of the project. With Mark (with the others) a monthly meeting was used to update him with the status of the project and to plan the future action points.

2.2 Eindhoven University of Technology (TU/e)

The Eindhoven University of Technology is responsible for the educational aspect of this project. Certain standards need to be met in order to fulfill the necessary requirements for a PDEng project. The TU/e is concerned with the problem analysis, design process, its implementation, and project management.

Table 2 TU/e’s Stakeholders

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serguei Roubtsov</td>
<td>TU/e supervisor</td>
</tr>
<tr>
<td>Vladimir Mikovski Iotov</td>
<td>PDEng Candidate</td>
</tr>
</tbody>
</table>
Dr. Serguei is a docent at TU/e in the Department of Mathematics and Computer Science and he was researcher in the Laboratory for Quality Software (LaQuSo). His expertise is focused in quality software and relational databases. Serguei is the mentor and supervisor of the project. He participate in weekly updates and the monthly meetings. Vladimir is the trainee (PDEng candidate) responsible to carry out the project.
3. Problem Analysis

This chapter focuses on the problem to be solved in the project. First, an introduction to the context and risk reporting is given, followed by the current system in place, and finally, the reasons why a performance analysis is needed.

3.1 Context

Banks need to generate not only internal reports but also several other reports to satisfy the requirements of the different external regulators. Reporting plays a key role in keeping control of the organization and its current situation. It is used to diagnose results and audit the information inside the organization.

Risk management is an essential part of financial institutions like banks. After the crisis in 2007-08, reporting risk data became a critical subject. The introduction of new reports and the ever-changing requirements of these reports is a reality where legacy systems struggle to keep pace on both data quality and timeliness aspect.

The reporting speed and frequency is increasing repeatedly. In order to cope with the new challenges, the current systems need to improve their performance. Therefore, the performance analysis becomes part of the steps to re-design and further develop systems that are prepared for the new and future requirements.

3.2 Liquidity risk reporting

Currently, the LiQDW system is responsible for the data aggregation to support reporting on LCR and NSFR, and the European Bank Authority's (EBA) reports. In order to produce the liquidity reporting, new data is received from multiple source systems. The data received varies in format and time since it comes from different countries and time zones. This situation conforms to a diverse and complex IT landscape.

The current architecture is shown in Figure 1. It is important to notice that LiQDW is one of the systems that supply data directly to the Reporting Solution, which is in charge of producing the reports, using the EBA accounting interface. Also the number of source system is significant, there can be more than one of them per entity/location (e.g. Utrecht, Hong Kong, and London). In order to aggregate and have the data ready for reporting, all the source systems must have delivered their data.
The architecture of LiQDW system is depicted in Figure 2. It is divided into six principal blocks: Staging, Data Vault, Data mart, Data consumers, Workflow application, and Orchestration. These blocks are aligned with the Logical Data Architecture standard followed in Rabobank (Figure 3).
The Staging block is used to receive the data incoming from the source systems. In order to receive new information, LiQDW have to extract, transform and load data from those source systems. This is called the ETL (Extract, Transform, and Load) process. The Staging is mainly used by ETL processes to support consolidation, alignment, and cleansing data. This block represents the Pre-validation Layer of the Logical Data Architecture.

The Data Vault block contains all the data historically organized. It is the core block of LiQDW and it is designed following the Data Vault modeling techniques. It is the end step of an ETL process and represents the Data Persistency Layer of the Logical Data Architecture.

It is important to add that Data Vault includes three components: Raw Vault, Rule Vault and Meta Vault. The Raw Vault keep the data as it is loaded from the source systems while the Rule Vault contains the results of the business rules applied to Raw Vault. Meta Vault is used to create the Data Vault processes. The Generic Business View, which is also in the Logical Data Architecture is also part of the Data Vault block.

On top of the Data Vault, LiQDW Metadata is responsible to define and handle the internal execution. Logging stores the status and information of the processes currently running/scheduled or already finished.

The Data mart block represents the Specific Business View Layer of the Logical Data Architecture. It acts as interface between the final users and the data inside the Data Vault. It is built to facilitate the delivery of data to the Data consumers.

The last block in the chain, Data consumers, represents the users of the system. Although they are mainly out of the system, some components remain in LiQDW. The Cube is used by the financial analysts through the dashboard to consolidate and verified the data delivered by LiQDW.

The Workflow application block facilitate the users to pull new data or to trigger business rules in the Data Vault block. It connects the Data consumers with the Data Vault through an interactive web portal. Last but not least, the Orchestration block contains several small components that have the responsibility to keep the flow of execution in LiQDW.
3.3 **LiQDW design**

A challenge that arose while studying LiQDW is that it is a system in continuous development. In order to keep the design aligned with the current version, the available information is only in a high level of abstraction. Further analysis is needed to understand the system’s details and its behavior. Some tools for automatic reverse engineering were used as a first approach.

SQLSpec [4] is a proprietary software able to analyze the objects inside databases and produce documentation about their structure, interfaces, and dependencies. Figure 4 and Figure 5 are examples of typical results using SQLSpec. Important to notice that in order to get the maximum amount of information, the code should be commented in a certain way.

![Figure 4: SQLSpec stored procedure of Performance Analysis Framework](image)

---

**Figure 4** SQLSpec stored procedure of Performance Analysis Framework
Red Gate's tools [5] for SQL Server contains a broader set of functionalities. It is possible to generate documentation on the objects inside the database as well as interact with the object dependencies (SQL Dependency Tracker, Figure 6 - Figure 7). Another useful tool is the SQL monitor, where most common performance metrics are available.

Figure 5 SQLSpec stored procedure of Performance Analysis Framework

Figure 6 SQL Dependency Tracker. Stored procedures LiQDW Metadata
Although these tools provide instant information on the structure of the collection of objects inside the databases, it is also important to understand the behavior inside the system. The collection of objects outside the databases (on server level) can’t be analyzed by those tools and there is a significant interaction between the two collections, inside and outside.

The SSIS packages are widely employed for ETL processes. The SSIS stands for SQL Server Integration Services and it is a component used mainly for data migration (ETL for example). Therefore, manual review of the code was necessary to connect the information from the SSIS packages and the database objects. Most of the SSIS packages are generated using the metadata from the Meta Vault.

Finally, looking deeper into the Data Vault block, which contains the main databases, there is a massive number of objects inside (tables, views, stored procedures, and others). The reason for the high number of tables is that Raw Vault and Rule Vault databases are designed with the Data Vault modeling techniques. In order to understand the system better, a research was carried out on the Data Vault modeling technique (Section 4.4, Data Vault modeling).

### 3.4 Performance Challenges

The execution to obtain a valid output from LiQDW to the Reporting Solution involve automatic and manual work. At present, the automatic work is the most time consuming (more than 75% of the total time). A full execution implies multiple loads of source data (for each entity/location source system). Each loaded entity will be processed twice before is approved (sign-off) its output. An overview of the current...
situation and target performance is available in the Appendix X1 (Not available in public version).

Potentially, a minute saved in the automatic processing, could save 50 minutes on the whole execution. Therefore it is very important to minimize the LiQDW processing. The goal is to decrease it up to 90%, making the execution from LiQDW to Reporting Solutions 75% faster. In order to fulfill such ambitious objective, it is necessary to observe, understand, and finally improve LiQDW processes.

A performance analysis framework helps to observe and measure the improvements achieved with each update of LiQDW. Since each update can focus on different aspect of LiQDW execution, the framework allows to customize it for specific needs.

3.5 Design Opportunities
During the analysis of the LiQDW system and the problem analysis, the design opportunities identified are usability and genericity. Although these two are related to the end product (the framework), as part of the process the project focuses on documentation and on following a methodological approach.
4. Domain Analysis

In the previous chapter, the problem analysis opened interesting topics which need to be further investigated in the domain analysis. These topics are the performance analysis as methodology, LiQDW system analysis and Data Vault modeling technique.

4.1 Introduction

Performance analysis is necessary to determine the improvements needed by a system to fulfill its requirements. Furthermore, it allows system administrators, designers, and developers to compare the efficiency between different alternatives, and to detect and prevent possible bottlenecks. Every performance analysis requires an intimate knowledge of the system analyzed in order to select the right techniques, metrics and workloads used in its evaluation.

Obtaining information of large and complex system is always challenging and time consuming. Often the knowledge is spread among different people that are not available or don't have the time to transfer their information. However, there are tools that may help to generate an overview and to understand better the system.

The study of the standards followed to develop the system can provide also valuable information. One of the main design principles used is the Data Vault modeling. This modeling method offers a flexible model and long-term historical storage of data coming from multiple source systems.

4.2 Performance analysis

The performance analysis is an extensive subject applied in many different fields. In this project, the performance analysis is a procedure to evaluate a system or application using techniques, metrics, and tools in a known environment with a defined workload. A performance analysis may completely differ from one problem to another. Therefore, it is not possible to produce mechanically the successful performance evaluation suitable for all situations.

As stated by Raj Jain [6]: "Performance evaluation is an art, and like an artist, each analyst has a unique style. Given the same problem, two analysts may choose different performance metrics and evaluation methodologies. In fact, given the same data [collected], two analysts may interpret them differently."

Despite the anarchism that may inspire the uniqueness of each performance problem, there are several methodologies proposed that provide guidance. Since the goal of this project is to follow a theoretical approach, it is crucial to examine in depth these methodologies. A collection of the most common methodologies is briefly summarized by Brendan Gregg in his blog [7]. This project focuses in the ten steps performance evaluation proposed by Raj Jain and the USE method by Brendan Gregg [8].

Besides these methods, there are several practices that is important to not use during the project. These practices, named as anti-methods by Brendan Gregg, usually are counterproductive. Among them, most relevant are the Streetlight, the Drunk Man, and the Traffic Light anti-methods. The Streetlight, based on an observational bias called the streetlight effect [9], consist of using a familiar, or found on the Internet, monitoring tool and seeing whether anything obvious shows up. It is the absence of any deliberate methodology but a hit-or-miss approach that can overlook many types of performance issues.

The Drunk Man anti-method is based on guessing game. It consist on changing randomly parameters or code in order to make the performance problem go away. With-
out good luck, it can take a considerable time and may not find the root of the problem. The Traffic Light anti-method consist of using a dashboard with the results of a monitoring tool. If all results are "green" then no problems. Without a further analysis, it is possible to miss a problem or misinterpret a false positive.

4.2.1. **A systematic approach to performance evaluation**

This systematic approach defined by Raj Jain [6] is based on his experience as analyst. It consists of ten common steps, which should be taken into account in any performance analysis. These steps are shown in Figure 8 and each step is explained below.

![Figure 8 Systematic approach in ten steps for performance evaluation](image)

1. **State the performance requirements and system definition:** the performance analysis is based upon the goals of the study case, the problems to be solved. These goals are described as requirements that after the experiment, it will be possible to reflect if the results satisfy them. Also the goals will delineate the system boundaries and abstract the unnecessary elements. The choice of the system boundaries affects the complexity of the experiment’s design and the metrics used.

2. **List services, interfaces, and outcomes of the system:** the system provides a set of services to the users. With each user request, different services may be involved and therefore it is possible to have different outcomes. The knowledge of services and their outcomes will influence on selecting the metrics and factors.

3. **Select the metrics and criteria for the performance evaluation:** the metrics selected will generate data (results) that can answer the goals stated in the first step. The metrics can be separated into three categories: responsiveness, productivity, and utilization.

4. **Identify the available parameters:** a list with all the possible parameters that may affect the performance is important. The parameters can be divided into system parameters (software or hardware) and workload parameters. This list will be revised during the analysis and new parameters can be added.
5. Select the factors from the parameters: from the list of parameters, some of them will be varied. These parameters are factors and their values are called levels. The parameters with high impact on the system should be preferably selected as factors. While selecting factors, it is important to consider the economic, political, and technological constraints that exist as well as the limitations imposed.

6. Select the evaluation technique: the three broad techniques for performance analysis are analytical modeling, simulation, and measuring a real system. The selection of the right technique depends upon the time and resources available to solve the problem and the desired level of accuracy.

7. Select the workloads: a workload consists of a list of service requests to the system. Depending upon the evaluation technique chosen, the workload may be expressed in different forms. It is essential that the workload is a representative usage of the system in real life.

8. Design experiment: with previous steps ready and the list of factors with their levels, next is to decide on the sequence of experiments (executions) to apply. It should offer maximum information with minimal effort.

9. Analyze the data (results) from the experiment: it is necessary to take into account the variability of the data and understand that the experiment only produces data and not conclusions. The data provide the basis on which the analysts can draw conclusions.

10. Present results: it is important to present the result of the performance analysis in a manner that is easily understood. This usually requires presenting the results in a graphical form and without statistical jargon. Now it should be possible to answer the requirements from step one, if not, then the analysis starts over again from a previous step.

### 4.2.2. USE Method

The USE (Utilization, Saturation, and Errors) method is used in the early stage of a performance investigation. This method was created by Brendan Gregg [8] and it can be summarized as: for every resource, check utilization, saturation, and errors. A resource is any hardware component like CPUs, RAM, and disks, also it is possible to examine software resources if it is required.

The idea is to have a list of metrics for the three categories on each resource. The utilization is the percentage of busy (working) time or the used capacity of a resource. The saturation is the pending work that a resource has in a point-in-time, usually represented by queues. The errors is the number of error events occurred in the resource. Recoverable failures are also part of the errors if it is significant for the performance analysis. In all the metrics is important to specify the time interval of the measurements, a long period of average normal value can hide short burst of high values.

In contrast to the Streetlight anti-method, the USE method iterates over system resources instead of starting with available or known tools. This creates a complete list of questions to ask, and only then selects the tools to answer them. If there are questions that is not possible to answer them with the current tools, it will be documented and taken into account when making conclusions after the evaluation.

### 4.3 LiQDW system analysis

LiQDW system is a data warehouse build based on the Data Vault technique and Rabobank's guidelines for data warehouses. Furthermore, there are several elements reused from Rabobank's Foundation ToolKit (FTK) for data warehouses. This toolkit provides functionality that is common for data warehouses currently in Rabobank. The tools available and used by LiQDW are the following.

The Metadata Driven ETL Generator provides BIML (Business Intelligence Markup Language) based ETL generator for SSIS packages. It is integrated with the tool.
PowerDesigner to generate the XAML (Extensible Application Markup Language) business rules. It uses the metadata and templates to generate the final ETL packages.

MetaVault tool offers validation, processing, and orchestration for ETL packages and stored procedures. Also provides support to the other tools of the FTK. The Screening Framework allows double validation on the data quality. First time when loading data from the source systems to the staging area and another extra validation when storing data from staging to the data vault.

Orchestration is responsible for the execution of the processes. It handles the logistics of the data flow. On the other hand, Static Data Maintenance (MDS) offers traceability, auditability, and security.

Business Rule Framework provides business rule's repository, editor, and engine for the execution of the rules inside the data warehouse. Quality Vault stores information on the quality of the data received from the sources. Workflow framework is the backbone for users to request data from the Data Vault.

The performance analysis of LiQDW starts with understanding the system and the services that it provides. However, it is also important to obtain metrics from the processes executed. There is information available in the Logging schema of the Metadata database. In order to understand this information, it is necessary to investigate how it was produced and what it represents.

An analysis of the available code of LiQDW produced several diagrams. The analysis carried out focuses in the logic behind the logging information and how it is produced. A block diagram and several sequence diagrams represents the interaction between the central SSIS packages and the stored procedures with the logging tables. Furthermore, since there is a priority queue for the ETL execution, also there is available a sequence diagram showing the priorities and locks of it. All the diagrams are available in the Appendix X2 (Not public) and correspond to April's version of LiQDW.

### 4.4 Data Vault modeling

Data Vault modeling is a specific data modeling technique for designing highly flexible, scalable, and adaptable data structures for enterprise data warehouse repositories. The Data Vault fulfills the role of a centralized enterprise data warehouse (EDW) which provides data to star-schema data marts, exploratory marts, and flat (denormalized) report tables. The Data Vault contains the facts while the data marts have their interpretation.

The formal definition is as follows: "The Data Vault is a detail oriented, historical tracking and uniquely linked set of normalized tables that support one or more functional areas of business. It is a hybrid approach encompassing the best of breed between 3rd normal form (3NF) and star schema. The design is flexible, scalable, consistent, and adaptable to the needs of the enterprise. It is a data model that is architected specifically to meet the needs of today's enterprise data warehouses."[10][11]

The target of a Data Vault data warehouse is to integrate disparate data from many different sources, and to link it all together while maintaining source systems' context. A Data Vault uses mainly three basic entity types (tables): Hubs, Links, and Satellites. An example is shown in Figure 9.
Hubs are tables that represent business objects. They contain a unique list of business keys with metadata such as load date timestamp, source system record, and surrogate (technical) unique key. Its job is to save the first time a business key arrives to the Data Vault and what is its origin.

Links are tables that represent relationships between hubs. Links are created if two or more business objects interact and they keep track of the past and present of the association. The relationships are always “many-to-many” because it will not restrict future changes in the model, making the design flexible. A link table contains its own unique identifier and as foreign keys, the identifiers of the connected hubs.

Satellites are tables that represent the attributes of the hubs and links. Each satellite contains the identifier of the hub or link and the column that indicates the date when the satellite data was loaded. A record in a satellite table represents a version of a hub’s or link’s satellite data. For a hub identifier there could be multiple records stored in a satellite table, but only one record is valid at present, the rest represent the past of that business object.

Although Data Vault modeling brings many benefits, there are some drawbacks. The complexity of the model in a Data Vault warehouse is very high, there are much more objects than in conventional data warehouses. This is a challenge from the point of view of maintainability and usability. It is not recommended/possible to directly use the Data Vault model by the business users. An abstraction layer is needed in order to interpret the data available.

Another challenge is the performance of the resulting Data Vault. The process to load data inside the Data Vault usually is not a problem but querying data is a completely different story. A query normally involves multiple tables with several joins that have as conditions date ranges. This is already a challenge for the current system LiQDW that needs to improve. Currently there is an initiative in LICWID to tackle this challenge.
5. Feasibility Analysis

After explaining the problem and the domain of the project, a feasibility analysis is performed to identify the issues and risks that may arise or exist. This chapter covers the issues identified during the project, and the risks with its mitigation strategies.

5.1 Issues and challenges

This section describes the challenges encountered during the project lifetime. There is a timeline chart available in the Appendix X3 (not public) that helps to map when and how these issues happened.

5.1.1. Change of project definition

At the beginning, the project was stated as "Create a performing user access layer for a large and complex Data Vault data warehouse". However the project was overlapping with another internal project in Rabobank with the exactly same purpose. Although at the beginning a collaboration was initiated, it didn't produce the expected results. Therefore in the third month, a change in the project definition was decided with the agreement of all the stakeholders. This implied re-scoping and changing the goals of the project in order to fit in the new project "Performance Analysis Framework for a Complex Financial Data Warehouse".

5.2 Risks

The most significant risk identified during the project with their mitigation and/or avoidance strategy. When a risk cannot be influenced by the trainee, it becomes a shared risk with the supervisors of the project.

5.2.1. Environment unavailable for performance analysis

The project needs a platform with the system to be analyzed. The performance results are more meaningful if the platform and system are as identical as possible to the production environment. The environment to be tested needs to be controlled and measured only by the performance evaluation, no other users are allowed because it can invalidate the experiment.

Since this risk was most likely to happen, the mitigation measure is to test the system in a smaller environment within the projects time life constrain. Later it can be tested when the target environment is available for a performance evaluation.

5.2.2. Test data unavailable for performance analysis

Similar to the previous one, in order to obtain meaningful performance information, a data set like the production environment should be available. The system to be analyzed uses source systems as input data. Those source systems needs to be prepared and ready for the performance evaluation.

In order to mitigate this risk, synthetic data can be used. This data is available for developers in order to make functional tests. However, the data is very small in size and the results obtained are not comparable to the production system.

5.2.3. Critical changes in LiQDW

LiQDW is under continuous development. There are teams making changes in the system that can block and constrain the project, even invalidate the performance analysis framework. It is not possible to keep up with all the updates and carry out the performance analysis.
The mitigation strategy adopted is to use a stable version of the system during the development. Keep low the amount of dependencies with LiQDW and at the end make the necessary changes to use the latest version. If not possible, an old version will be used for the performance analysis.
6. System Requirements

In this chapter, the functional and non-functional requirements are presented. They are based on the problem and domain analysis, and have to be satisfied in the project.

6.1 Introduction

With the problem analysis and the domain analysis finished, next step is to match them with the project goals in a design of a new system. The new system has to support the performance analysis of LiQDW system with the quality attributes of usability and genericity previously selected as design opportunities.

In order to accomplish a successful performance analysis, three main features are identified. First, it is important to capture performance data and shouldn't be constrained to a single tool. As explained in the domain analysis, the performance evaluation is not based on a known tool but in the required metrics. Second, the data needs to be stored and aggregated in order to get the best interpretation of this information. Third and the last step is to visualize and share the performance data so that it can be used to draw conclusions and support analyst’s theories.

6.2 Main features and use cases

There are three main features that define the behavior expected from the system, and the functional requirements are classified based on them. Those features are the following:

- **Capture performance data** allows the user to gather information on the behavior and status of a system. It measures the grade of usage, saturation and events that arises. This feature should produce data that can be stored for its later visualization.

- **Data storage repository** provides a centralized solution to store performance data. It should be flexible to accept data from different sources. Also it should keep the data historically organized and aggregated. It allows access to the data from other applications for its visualization.

- **Visualization of performance data** present the performance information to the users. It should be interactive in order to allow in depth analysis of the data available. Furthermore, the views and analysis of the data should be possible to share it with other users.

Only one role, User, can operate the whole system. There is no need to differentiate between the user that triggers the process of capturing data with the user that analyze the data produced since both can be the same person. This is align with the new trend to join the operational and development roles together in what is called DevOps engineer. The intention is that the system will help not only to monitor for bottlenecks but also in the development process.

As many other systems, the need of an administrator is required. The Administrator, which is recommended to be a Database Administrator (DBA), is a special user responsible of the system's configuration. Since the performance data gathered in an experiment can grow exponentially, a role that ensure the optimal parameters and supervise the system's usage is needed. A DBA is expert in databases and data related issues, and for that reason it is expected to fulfill correctly the intended administrator role.

In Figure 10, the relation between the uses cases and the actors is represented. Two of the use cases are extensions, which means that not necessary are used by the user.
Administrator actor is an extended role of the User, a role with more privileges that can configure the system.

![uc Use Cases](image)

**Figure 10 System level use cases view**

### 6.3 Functional requirements

With the main use cases identified, the functional requirements are derived. The functional requirements define the behavior, input, and output of the system. They describe what the system is supposed to achieve.

From the three main features, a set of functional requirements are derived (Table 3). The requirements are classified with their use case and a priority is assigned instead of the options must, should, and could. All the requirements have to be fulfilled up to certain degree at the end of the project.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Use case</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_01</td>
<td>The system should allow visualization of performance data based on the capture time</td>
<td>View</td>
<td>High</td>
</tr>
<tr>
<td>FR_02</td>
<td>The system should aggregate performance data based on the capture time</td>
<td>Capture</td>
<td>High</td>
</tr>
<tr>
<td>FR_03</td>
<td>The system should be able to gather performance data of the tested system and its processes</td>
<td>Capture</td>
<td>High</td>
</tr>
<tr>
<td>FR_04</td>
<td>The system should integrate and adapt to different performance input data and tools</td>
<td>Configure</td>
<td>High</td>
</tr>
<tr>
<td>FR_05</td>
<td>The system should be able to share the performance analysis based on the data</td>
<td>Share</td>
<td>High</td>
</tr>
<tr>
<td>FR_06</td>
<td>The system should execute the capturing process based on a schedule predefined</td>
<td>Schedule</td>
<td>High</td>
</tr>
<tr>
<td>FR_07</td>
<td>The system should allow access to historical performance data based on capture time</td>
<td>View</td>
<td>High</td>
</tr>
<tr>
<td>FR_08</td>
<td>The system should read, import, and export simple data format of performance information</td>
<td>Capture</td>
<td>High</td>
</tr>
<tr>
<td>FR_09</td>
<td>The system should allow templates to visualize the</td>
<td>View</td>
<td>Medium</td>
</tr>
</tbody>
</table>
performance information
FR_10 The system should be able to compare two sequences of performance information View Low

6.4 Software quality attributes

Besides the functional requirements, a number of quality attributes (non-functional requirements) are identified from the project’s goals and the design opportunities selected (Table 4). Those attributes fall under the usability and flexibility categories.

As part of the usability characteristics of the system, operability and attractiveness are the main focuses. For operability, the system should provide an intuitive interface where the user should be able to perform different analyses based on the data collected. The interface should be adaptable and interactive. As for attractiveness, the system should produce precise, clear and complete charts in order to increase the salability of the results.

The characteristic of flexibility is related to the ability to change and use different components and tools. The system should be able to adapt its functionality based on the requirements of each performance analysis and system to be analyzed. Different tools and metrics can be used to measure the performance and it is important to facilitate its integration.

Table 4 Non-functional requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR_01</td>
<td>The system should support more than one performance capturing tool (Flexibility)</td>
<td>Capture</td>
</tr>
<tr>
<td>NFR_02</td>
<td>The system should allow more than one visualization tool (Flexibility)</td>
<td>View</td>
</tr>
<tr>
<td>NFR_03</td>
<td>The system should provide an interactive visualization. The charts should be real-time modifiable by the user (Usability)</td>
<td>View</td>
</tr>
<tr>
<td>NFR_04</td>
<td>The system should adapt to different performance input data and tools automatically or with minimal manual work (Flexibility)</td>
<td>Configure</td>
</tr>
<tr>
<td>NFR_05</td>
<td>The system should support charts with time precision of at least second (Usability)</td>
<td>View</td>
</tr>
</tbody>
</table>

6.5 Constraint requirements

Besides the functional requirements and the quality attributes, a constraint requirement is identified. This constraint is based on infrastructure and organizational limitations, more specifically the LiQDW system, which is the target system to be analyzed.

Table 5 Constraint requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR_01</td>
<td>The system should share the same technology stack as LiQDW when possible</td>
</tr>
</tbody>
</table>
7. System Architecture

In the previous chapter, the requirements of the system are defined. These requirements shape the architecture and design of the solution. In this chapter, the architecture is presented with a technological decision.

7.1 Introduction

The system architecture is the backbone of the system's design. It is selected based on the functional requirements and non-functional requirements (quality attributes) defined already. The architecture will expose the high level structure of the solution (system) that realizes the main use cases. It is important to early adopt the most suitable architecture style or pattern in order to avoid future challenges and constraints.

7.2 Architectural reasoning

The system follows a distributed architecture. The three main features of the system are to capture performance data, to store and aggregate it, and to visualize the data and share its analysis. Derived from these three features the main blocks identified are: Data Collector, Data Storage, and Data Viewer (Figure 11). Each block has as responsibility one of the main features and they can be implemented differently as long as they comply with the established interfaces. This contributes to the flexibility of the system and can potentially reduce cost of implementation if a component is replaced by a third-party software.

![Figure 11 High-level abstract components model of the system](image)

The Data Storage component needs to store and manipulate data as well as supply it to the Data Viewer’s users. It provides two interfaces, one to store data and one to select it. The Data Collector component represents the different tools used in the performance testing. This component changes based on the needs of the performance analysis. The Data Viewer component allows users to visualize the performance information and supports an active data analysis with an interactive interface. As different users are interested in specific data for their analysis, customized views on the data are possible.

Client/Server architecture is used as deployment architecture style. The system is deployed into two servers and the client (Figure 12). With this approach, the data can be centralized in a data storage server. The data collector’s tools are application or services in the server (tested server) that contains the system to be evaluated. That way the data storage processes don't interfere with the performance information obtained by the data collectors.

The client contains the application that allows the visualization of data. Additionally an interface to manage the data collector’s functionality is needed. Both applications
(interfaces) are located in the same client since there is no reason to separate them, however it is possible to deploy them separately.

![High-level deployment architecture](image)

*Figure 12 High-level deployment architecture*

An important decision is to have the system in the form of software framework to support the performance analysis. With this approach, the system’s flexibility is enhanced. The system will provide an implementation to carry out the performance analysis of LiQDW, however the intention is to keep open the design for other implementations in different environments.

An application framework models a specific domain or an important aspect as a reusable design containing a set of implementations. The framework extensions and the collaboration between its components are vital for its effectiveness. A framework is a skeleton that determines how the internal components relate to each other. The main advantage is that design and code reuse are higher and therefore productivity is increased while time-to-market is decreased. In addition, the implementations tend to look more homogenous reducing maintainability efforts.

“Application frameworks consist of ready-to-use and semi-finished building blocks. The overall architecture (the composition and interaction of building blocks) is predefined as well. Producing specific applications usually means to adjust building blocks to specific needs by overriding. In general, an application framework standardizes applications for a specific domain.”[12] And in this project, this specific domain is performance analysis.

Following the work of W. Pree [12], software frameworks can be separated into frozen spots and hot spots. Frozen spots are parts of the system that remain unchanged and therefore reused in each instantiation of the application framework. These frozen blocks define the overall architecture of the system. Hot spots represent the parts of the framework where software engineers most likely will change or extend with their own implementation to add the specific functionality needed in their project.
In Figure 13, the hot and cold spots are represented. Data Collector is a hot spot because it can change depending on the metrics selected for the performance evaluation. Remember that the performance analysis is guided first by metrics rather than known tools (see Section 4.2). Data Viewer is also a hot spot since each user may require a specific visualization tool for a unique analysis. The Data Collector Manager is yellow since it should not change, however it is dependent on the Data Collector and it may need some adjustments. On the other hand, the Data Storage is a cold spot because it stores and aggregates simple data format that is generic for any tool.

### 7.3 Technological constraints

The constraint requirements from Section 6.5 of the report lead to an early choice in the technologies used in the solution. These constrains are related to LiQDW, the target system to be analyzed.

LiQDW is developed for Microsoft SQL Server, a relational database management system, and as consequence of the constraint to share technological stack when possible, the Data Storage component will use the same technology. Furthermore, it will enable the option to deploy it in the same server as LiQDW.
In this chapter, the system design is presented. It describes in more detail the solution delineated in the previous chapter, the system architecture.

### 8.1 Introduction

The system design defines in more detail the components that conform the system. The “4+1” view model [13] is used to describe the system design. It consist of multiple concurrent views or perspectives which address different concerns separately to various stakeholders and handle the functional and non-functional requirements. The model is made up of five views:

- **The logical view** which contains the object model or structure of the design.
- **The process view** which contains the concurrency and synchronization aspects of the design.
- **The development view** which describes the static organization of the software in its development environment.
- **The deployment view** which contains the mapping of the software onto the hardware and its distribution aspect.

The fifth view, the *scenarios*, is a special view which combines all the previous views in a use case or scenario. The scenarios view is closely related with the use cases already presented in the previous Section 6.2.

### 8.2 Scenarios

The scenarios used in all the four views are to *capture performance data* and to *analyze performance data*. These are the typical actions that a user will execute and they correspond to the use cases in Section 6.2 with same names.

In the scenario to capture performance data, the user triggers the process of collecting performance data in the tested server with the system to be analyzed its performance. The data collected is stored and sent to the data storage.

In the scenario to analyze performance data, the user opens the visualization tool in the client and connects to the data storage server to select the data desired. Once received the data, the user can interact with it as graphical representation (charts).

A third scenario is used only in some of the four views. It is to *configure the data structure*. The administrator generates the internal structure of the data storage in order to support the desired data collector output. This scenario is only relevant in the logical and development views.

### 8.3 Logical view

The logical view decompose the system into a set of abstractions and elements that supports the functional requirements. Following the three scenarios, the diagrams will break down the main components focusing on its logical structure and behavior.

In order to configure and store the performance data, the Data Storage component is a database, PerfDB, where data is stored in tables. Inside PerfDB, there are 3 schemas, used as a logical separation of the internal objects. PerfMeta contains the stored procedures and tables necessary to generate the tables and views for the performance data. These tables and views are created dynamically in the PerfData schema. PerfCapture schema contains stored procedures used to collect performance data from the database side. The diagram in Figure 14 shows the structure of PerfDB, a
non-standard modeling technique is used [14] (for clarity each object is represented with its stereotype and the operations without parameters).

Figure 14 Diagram with objects inside PerfDB

The generation of the performance structure (inside PerfData schema) is a crucial step before being able to capture or visualize data. In the following sequence diagram, Figure 15, the logic behind the scenario of configuring data structure is illustrated. Important to notice that there is an option to drop old structure, which will delete any previous performance data stored. It is recommended to make a backup of PerfDB before using this option.

If an interface is specified, a new view for each table will be created following the specifications of the interface. It is possible to create more than one set of views as interface for different applications. In this example, the interface for Tableau is created and mainly it is formatting some of the columns from the source tables.
In the scenario to capture performance data, the components involved are the Data Collector and the Data Storage (see Figure 16). To initiate the capturing of performance data, the concept of a job is introduced. A job can be scheduled to run at a certain moment. It consists of a series of steps that can be executed non-sequentially, which opens different possibilities. There are mainly two types of jobs identified. First type executes only once its steps and finishes. It is useful for single tasks or for asynchronous executions. Under this category are PerfMon jobs.

The other type of jobs will execute continuously until they are stopped. The order of their steps is configured to be an infinite loop. One of the steps is a wait interval that can be modified depending on the requirements of the performance analysis. These types of jobs are suitable for repetitive short tasks.
The logical behavior of collecting performance data is represented in the sequence diagrams of Figure 17 and Figure 18. In the first one, performance data is extracted from the Dynamic Management Views (DMVs) of SQL Server and stored in PerfDB. Then, a sleep interval is executed and the job waits until it finishes and the loop starts again with the first step. This loop continues until the job is stopped.

The second sequence (Figure 18) shows how to extract the logging data from LiQDW. In this case, the job executes only once and finishes, it doesn't need intervention to stop. Important to notice that if the job doesn't exist, the user needs to
create it first. This is only necessary the first time, the Agent will keep track of the jobs.

![Sequence diagram of collecting LiQDW logging data](image)

**Figure 18** Sequence diagram of collecting LiQDW logging data

The last scenario, to analyze performance data, involves Data Viewer and Data Storage components. In the data viewer, the concept of Workbook is used. A workbook can connect to PerfDB and contains charts that are built based on the performance data stored in PerfData schema. Figure 19 represents the logical structure. A workbook has a start and end date in order to tailor the performance analysis. After creating a workbook, it can be published and shared with other users.

![Logical structure to analyze performance data](image)

**Figure 19** Logical structure to analyze performance data

### 8.4 Process view

The process view takes into account the non-functional requirements, it addresses issues of concurrency and distribution. For the two scenarios (to capture and to analyze performance data), the actions involved are shown with activity diagrams. In this view, the third scenario (to configure) is ignored since it doesn't bring relevant information.
For the capture of performance data, there are two options. The first one is when the job never ends and it executes an infinite loop of tasks that collects and stores data (Figure 20). There are two ways to finish such job, either by an asynchronous request to stop or when the schedule of the job is finished. The data collected is directly stored to PerfDB. If a connection to PerfDB is not possible or doesn't perform as expected, it is possible to use a file which later will be uploaded to PerfDB.

![Activity diagram to capture performance data](image)

The second option is when the job only executes once and as a task, it activates asynchronously different process to collect performance data. In that case, the user has to trigger a new job to stop the process initiated previously (see Figure 21). The main difference with the first sequence is that the interruption occurs within the Tested Server while in the other one is in the Agent. Also, the Agent stays inactive while the process of collecting data is running.
Figure 21 Activity diagram to capture performance data asynchronously

Figure 22 Activity diagram to analyze performance data
In the analysis of performance data scenario, the concurrency of requesting data from PerfDB is handled by SQL Server (technology already chosen in Section 7.3). Many users can access to the data since their request is read-only. A workbook can contain local data inside that will minimize or avoid continuous pulling from PerfDB. Figure 22 represents the activity diagram.

### 8.5 Development view

The development view focuses on the actual software module organization within the software development environment. It presents the interfaces and the dependencies between modules. The modules with its components are represented in Figure 23. For the scenario to capture performance data, the modules involved are the Agent, Data Collector, and PerfTables (from PerfData schema) of PerfDB. To analyze performance data scenario, the modules involved are Data Viewer and the PerfViews (from PerfData schema) of PerfDB. The package view (Figure 24) represents the schemas and the objects inside PerfDB as a result of the scenario to configure data structure.

![Component Model](image)

*Figure 23 Component diagram of the performance analysis framework*
8.6 Deployment view

The deployment view describes where the system will execute and the hardware that will support its execution. The intended deployment model (see Figure 25) distributes the functionality over three different devices, following the client/server architecture. With a fully distributed system, the performance impact when visualizing and storing data is minimal in the system to be evaluated (that resides in the Tested Server).
Figure 25 Deployment diagram of the performance analysis framework
9. Implementation

In the previous chapters the design is explained. In this chapter, its realization together with the technology choices that conforms the framework implementation are discussed.

9.1 Introduction

The prototype’s implementation is a proof of concept that validates the functional requirements and quality attributes of the system’s design and architecture. The choices made in the implementation are based on the constraint requirements of Section 6.5 and the possibility to use the performance analysis framework for the system LiQDW. The end result should yield measurements on the performance of LiQDW.

The Appendix 1: Performance Analysis of LiQDW has the analysis steps applied to LiQDW. It contains the metrics chosen for the performance analysis, therefore has a direct impact on the implementation of the Data Collector component. It is important to check the appendix because it is closely related to the implementation of the framework.

The following sections divide the choices made within the main components identified in Section 7.2 (System Architecture).

9.2 Data Storage

Early in the Section 7.3, the first technological choice was made for Data Storage component by selecting Microsoft SQL Server 2012 as database manager system (DBMS) for the database PerfDB. This choice is based on the constraint requirement to use the same technology stack used in LiQDW. Other possibilities could be one of the other well-known vendors of database technologies such as MySQL, LUW DB2, and Oracle Database. However, these are not used in LiQDW and therefore they are discarded.

The Figure 26 shows the tables and stored procedures created in PerfDB. The tool used is Microsoft SQL Server Management Studio (SSMS), very popular for SQL Server development. The object explorer allows you to connect to an instance deployed and explore its databases.

Windows Server 2012 is selected as operation system for the database server because SQL Server 2012 is constrained to Microsoft Windows operation systems.
Data Collector

The performance analysis specify the metrics needed before selecting the tools to use as data collectors. The metrics are available in Appendix 1: Performance Analysis of LiQDW (step 3 pg. 86). With the metrics decided, the tools already available in the Tested Server are the first possible choice to look at. This way it reduces the technology stack necessary for the performance analysis.

Since LiQDW uses Windows Server, the tool already available is the well-known Windows Performance Monitor (PerfMon, Figure 27). It allows to examine how running programs affect the computer's performance. With PerfMon, the metrics covered are from id 1 to id 49. These metrics are related to the usage of hardware components by the currently executing programs in the server. Some of them are specifically to SQL Server 2012 software. In order to capture data, first a data collector set is registered with its output (as part of the installation, see Section 11.2 ). Although it offers the option as output to store data directly to SQL Server, a binary file is selected because it has lesser performance impact.
With PerfMon is not enough to cover all the metrics desired, another tool is needed. The Dynamic Management Views and Functions (DMVs) of SQL Server are the perfect candidate to satisfy the rest of metrics (from id 50 to id 63). They are applied to obtain server and database state information that can be used to monitor the health of a SQL Server instance, diagnose problems, and tune performance. The metrics needed are closely related to the performance of SQL Server. In order to access the data, stored procedures with the Transact-SQL are used. They are available in PerfCapture schema of PerfDB.

Last but not least, a dedicated stored procedure is created to copy the metadata of LiQDW processes. The data is extracted from Logging schema and combines multiple tables with information about start/end date times, number of row (selected, changed, dropped, and skipped), and hierarchy of the processes.
9.3.1. Data Collector Manager

The manager of the Data Collector component is the interface for the user to interact with the data collectors. It is the Agent that executes the jobs related to capture performance data. Since LiQDW uses Windows Server and SQL Server, two options are available without the need of extra software.

Windows Task Scheduler is an option included in Windows Server that allows to execute jobs like starting applications or scripts. Jobs can be scheduled to start at a specific time or when an event occurs. This option was discarded since it is not used by LiQDW development team and it requires elevated privileges on Windows Server.

SQL Server Agent is the other option already available from SQL Server 2012. It also allows to execute scheduled, triggered by event, and on demand jobs. A job contains several steps and can run on a local server or on multiple remote servers. This option was chosen since it is widely used by LiQDW team and allows direct execution of SQL Database objects, like stored procedures.

In order to start and stop PerfMon, two batch scripts are created that are called from the corresponding jobs in SQL Server Agent (the manager). For the stored procedures which collect the rest of metrics, jobs with the task to execute them are created. Figure 28 shows the jobs created in SQL Server Agent.

![Figure 28 SQL Server Management Studio (SSMS) object explorer on the Agent with the jobs](image)

9.4 Data Viewer

For the visualization, Tableau software is selected. It provides an interactive way of looking at the data. It allows to connect to multiple sources, which among them is the SQL Server 2012. It works based on workbooks that can be published in Tableau Server to share it with other users.

Tableau is already used by the business users of LiQDW to analyze its financial output and validate it. Tableau offers an intuitive interface based on drag and drop where information is automatically filled from the sources. Additionally, the attractiveness
of the data visualization is superior to other tools like PerfMon (Figure 27) or Microsoft Excel.

With Tableau interface is possible to compare multiple values if all of them have the same scale, otherwise the information may be deceptive. An example is shown in Figure 29. In order to create such chart, Measure Values has to be drag and drop to Rows and then select the measurements wanted (bottom left). Having multiple measures can be messy, therefore there is a possibility to highlight them as needed.

Another possibility with Table is to represent multiple charts sharing the same timeline axis. This way it is easier to compare, analyze, and correlate performance data. To achieve this effect, the Columns will contain only one date time while the Rows can have multiple values. Figure 30 is an example where context switches per second, processor queue length, batch requests per second, and (re)compilations per second are displayed timely align.
Another alternative used to compare two metrics that have different scale is to have a chart with dual axis. On the left side is one metric scale and on the right is the other one. This effect can be achieved when two metrics in Rows are combined. Figure 31 represents an example with a few metrics in dual axis and sharing the same time line.

For the execution processes of LiQDW, a Gantt chart is used. This style of charts are well suited to represent task that are time constrained. An example can be seen in Figure 32.

The workbook used in an analysis can be saved and shared with other users if it is published. In order to be able to do that, a Tableau Server is needed. Users can access and download workbooks through Tableau Server and later reuse them as templates for their own analysis. This stimulates users to share knowledge and reuse work.
The overview of all the technologies used in the implemented solution are shown in Figure 33 as deployment diagram. Since PerfDB and the system to be analyzed (LiQDW) are implemented in SQL Server, both can be deployed in the same device (server). Same situation with the SQL Server Agent, being part of SQL Server allows it to combine with PerfDB and LiQDW. All devices share Microsoft Windows operation system, which minimize the stack of technologies, however the servers must use Windows Server edition.
Figure 33 Deployment diagram with the technology chosen
10. Verification & Validation

In the previous chapters the design and implementation is explained. To ensure that the prototype developed fulfills the intended functionality and behavior, a representative test scenario is produced. This chapter introduces the test scenario and its results.

10.1 Introduction

In order to test the performance analysis framework, a test environment is needed. The stakeholders are interested to evaluate the performance of LiQDW system, as stated in the project goals. The risks identified in the Feasibility Analysis (Section 5) also shape the design of the performance test environment.

A test scenario with a local deployment is carried out in order to test the functionality of the framework (more information in Section 11.1, Deployment). This scenario consists of a LiQDW installation with a test case that uses synthetic data. All the components of the framework are deployed in one machine sharing the resources available. In this test scenario all risks identified are avoided but the results of the performance evaluation are not useful rather than for testing the functionality of the framework.

10.2 Local performance evaluation of LiQDW

The test scenario is a performance analysis experiment of LiQDW using the framework. The whole experiment is available in the Appendix 1: Performance Analysis of LiQDW. The experiment follows the methodological approach of USE method and the systematic approach of the ten steps from Section 4.2 (Domain Analysis).

In the performance analysis, first the information about LiQDW system is explained. It also includes the objective of the experiment, which is to baseline the performance of LiQDW. Then the metrics desired are listed with its explanation, based on the USE method. The parameters that affect the experiment are identified but none of them is selected to be varied because the goal is to only baseline the performance.

The evaluation technique selected is to measure, and the workload is the test case that developers use to verify the functionality of LiQDW. It uses a small amount of synthetic data. Finally, the output result with the analyzed data is produced. As example the following conclusions can be drawn from the data obtained.

The execution of the test started at 2:36 PM and it consisted of two requests to the Rule Vault (MasterRuleVault) that will extract data to the Data mart and the Cube (Figure 34).

Figure 34 Chart of executions obtained from the experiment
Figure 35 Chart of processors' usage obtained from the experiment

Figure 36 Chart of processors' privileged time obtained from the experiment
During these two requests, it is possible to see how the processors’ usage increases (Figure 35). All the processors are well used which suggests that there is no parallelization problem. Also from the time spend in privileged mode, except three spikes, everything is normal (Figure 36).

Looking closely to the number of context switches (Figure 37), it may raise a warning, however with such low values of queue length means that many short tasks are executed at that moment. Since the synthetic data used is very small amount, many of the jobs are not heavy processes.

For the Memory values, all parameters seems to be normal (Figure 38). Again, it is possible to see when the test started because of the increase in the amount of memory used by the SQL Server (pink).

![Figure 37 Charts of CPU information obtained from the experiment](image1)

![Figure 38 Charts of RAM information obtained from the experiment](image2)
Many other charts were created in the workbook after the test execution. No particular interesting result or red flags were found due to the test scenario and the synthetic data used.

### 10.3 Validation

The framework designed and developed has to comply with the functional, non-functional (quality attributes) requirements from Section 6 (System Requirements). Once the test scenario is used, it is possible to answer how well the framework satisfy them. The answers for each requirement is satisfied, not satisfied or not validated.

Table 6 Functional requirements status

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_01</td>
<td>The system should allow visualization of performance data based on the capture time</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_02</td>
<td>The system should aggregate performance data based on the capture time</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_03</td>
<td>The system should be able to gather performance data of the tested system and its processes</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_04</td>
<td>The system should integrate and adapt to different performance input data and tools</td>
<td>Not validated</td>
</tr>
<tr>
<td>FR_05</td>
<td>The system should be able to share the performance analysis based on the data</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_06</td>
<td>The system should execute the capturing process based on a schedule predefined</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_07</td>
<td>The system should allow access to historical performance data based on capture time</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_08</td>
<td>The system should read, import, and export simple data format of performance information</td>
<td>Not validated</td>
</tr>
<tr>
<td>FR_09</td>
<td>The system should allow templates to visualize the performance information</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR_10</td>
<td>The system should be able to compare two sequences of performance information</td>
<td>Not validated</td>
</tr>
</tbody>
</table>

The FR_04 and FR_08 are not proven with the current test scenario. It is necessary to create more experiments in order to validate them. The FR_10 is not validated since it was not possible to finish an example of it within the time frame of the project. However, the workbook of Tableau allows to compare sequences of information which suggest that the requirement can be satisfied.

The quality attributes (non-functional requirements) are evaluate the same way as the functional requirements (Table 7). Satisfied, no satisfied or not validated are the three options. The NFR_04 is not validated since more test scenarios are needed with different toolsets for Data Collector component.

Table 7 Non-functional requirements status

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR_01</td>
<td>The system should support more than one performance capturing tool (Flexibility)</td>
<td>Satisfied</td>
</tr>
<tr>
<td>NFR_02</td>
<td>The system should allow more than one visualization tool (Flexibility)</td>
<td>Satisfied</td>
</tr>
<tr>
<td>NFR_03</td>
<td>The system should provide an interactive visualization. The charts should be real-time modifiable by the user (Usability)</td>
<td>Satisfied</td>
</tr>
<tr>
<td>NFR_04</td>
<td>The system should adapt to different performance input data and tools automatically or with minimal manual work (Flexibility)</td>
<td>Not validated</td>
</tr>
</tbody>
</table>
The constraint requirement (CR_01) from Section 6.5 is satisfied since the technologies used in the implemented framework are also employed in LiQDW. More information on the Implementation chapter (Section 9).
11. Deployment

In this chapter, the deployment and installation of the system is explained. The deployment is based on the design and the choices made in the implementation together with the test scenario.

11.1 Test scenario deployment

The deployment of the framework shown in the architecture and system design is adapted for the test scenario. In order to minimize the requirements of a full deployment of the framework, all the components are installed locally in a single machine.

In the implementation of the framework, all components are chosen to execute on Windows operation system. This facilitates the deployment on a single operation system without establishing communication channels. Also SQL Server installation can be reused by LiQDW, PerfDB, and the Agent. The deployment used in the test scenario is represented in Figure 39, Tableau Server is an optional server that allows to share workbooks.

![Deployment diagram on a single machine](image)

**Figure 39 Deployment diagram on a single machine**

11.2 Installation manual

A manual for the single machine installation is available as Appendix 2: Installation Manual. It contains the steps to install and uninstall the framework with the implementation for the test scenario. The general steps are as follows:

1. Create home directory used by the framework and copy installation files
2. Create PerfDB database in Microsoft SQL Server 2012
3. Create/Grant user rights and permissions on PerfDB and Windows OS
4. Install scripts for the objects inside PerfDB
5. Configure SQL Server Agent and create jobs
6. Configure PerfMon data collectors

Since the installation only involves one machine, there is no need to configure communications and less users and roles are needed than a distributed installation.
12. Conclusions

This chapter presents the results achieved in the project and the future work. It also elaborates on the lessons learned during the project.

12.1 Results

Data warehouses such as LiQDW will continue to face new challenges in the near future with the upcoming regulations. In order to improve the aggregation capabilities of LiQDW, continuous updates are being developed. With each new version, the throughput has to be monitored to identify any possible problem. A performance analysis framework can help with this task and also bring new insights on where the system can be improved for the next update.

A disciplined approach to analyze LiQDW’s performance is the main goal of this project. This goal was realized with the framework for performance analysis and its implementation for LiQDW. An extensive domain analysis on performance analysis was done in order to conform to a systematic/methodological approach (Section 4.2) as demanded by the stakeholders. Furthermore, the framework was designed and implemented following the system requirements derived from the problem and domain analysis.

The framework was tested in a scenario as a performance analysis experiment, obtaining tangible results (illustrated in Section 10.2 and Appendix 1: Performance Analysis of LiQDW). These results combine the logging data from LiQDW with the performance data of the hardware and software. Unfortunately the data obtained from the experiment cannot be used to produce recommendations on LiQDW system since it is a test scenario where synthetic input data was used.

12.2 Future work

With the time constraint of the project, there is always possibilities for future extensions or improvements on the solution. The identified future work on the framework is:

- The current use of the framework is as a local installation, all the components are installed in a single machine. This can be changed to a distributed deployment as designed in the Section 8.6. This way, the framework will not interfere with the performance data gathered from LiQDW. However, the installation manual (Appendix 2: Installation Manual) will need an update and it will become more complex.

- Applying the performance analysis in production environment, where the results of the analyzed data can be used for future improvements and to detect problems. This step will be only possible after the framework implementation is mature enough to be able to cope with the quality standards required by Rabobank.

- The flexibility of the framework to support different tools in Data Collector and input data in Data Storage was not fully tested in the prototype implementation. More investigation and testing will delineate the borders of the current designed framework, and it may suggest new improvements.

- Comparing two traces of performance data was not implemented. Although it is a functional requirement, due to time limitations on the project it was not implemented. With this feature, it will be possible to compare the performance of two different versions of LiQDW. This can be used as guidance
for the developers that are solving issues or extending the system with new functionality.

12.3 Lessons learned

The lessons learned during this project can be separated into two types, the organizational and the technical. On the organizational side, for both Rabobank and the trainee, it is the first time that they supervise/perform such project. This is a challenge that can only be overcome by having continuous update meetings with clear communication. The trainee is ultimately responsible for the project, however, the decision-making process should be shared with the supervisors. In the middle of the project, an important decision (change the project’s goals) has to be made and only with the guidance of the supervisors was possible to successfully bring this change. The lesson learned is that it is important to know when to be guided and when to take the initiative.

On the more technical side, it is important to identify the technical risks early and use the knowledge and experts from the organization to solve them or at least avoid them. The risks identified in the Feasibility Analysis were avoided with the right implementation and test scenario. The help received from Rabobank colleagues were key in order to overcome these challenges. The lesson learned is that the trainee is not alone in the organization and that he/she should explore and find the knowledge that he/she needs.
13. Project Management

In this chapter, an analysis on how the work described in the previous chapters was managed. The planning and execution of the project are explained.

13.1 Introduction

In this project, the project management was carried out using an agile approach [15], iterative with fast feedback, while borrowing some PRINCE2 [16] principles, like manage on stages and with a project initiation documentation.

This is a project where it is important to have a delivery that fulfills the goals of the stakeholders (prototype and technical report). In order to identify these goals, the PRINCE2 Project Initiation Documentation (PID) was produced. It contains a collection of documents that provide a reference point throughout the project for both the customer and the project team (in this case only the trainee). The documents used during the project are project goals, deliverables, scope, stakeholders, risks, and context.

Since the project is time-constrained, a general strategy or plan was created dividing the time into stages. These stages are described in the Project execution section. Each stage focuses in a particular activity and the next stage is not started until the previous one is finished. This is appropriate for this project because the team is only one person, the trainee, and these activities are mostly dependent each other.

At the beginning, the requirements of the project are not define, and most likely they will change during the project. So an agile approach of management is needed. Short loops of feedback with the main stakeholders is important, and normally weekly updates were used for that. In order to show the new progress, an iterative approach was used during the software development. In general, the intention is to minimize the micromanagement while maximizing the available time for research and development.

13.2 Work-Breakdown Structure (WBS)

The work-breakdown structure of the project is define with three stages: Project Initiation, Domain Analysis, and the development of the Performance Analysis Framework. This work can be mapped to the project plan overview in the next section.

![Figure 40 Work-breakdown structure of the project](image)
The project initiation will produce the Project Initiation Documentation that describes the motivation and supports the project planning.

The domain analysis is separated into three main chunks of work. The performance analysis domain that will produce the practices and theories that can be applied for LiQDW evaluation. Data Vault modeling research that will contribute to understand the LiQDW system and how it is build. LiQDW system corresponds to the investigation and the reverse engineering of LiQDW. At the end, the domain analysis will produce the knowledge necessary to implement the framework for LiQDW system.

For the performance analysis framework, the three main features can be developed separately. Data Collector will focus on facilitating the integration of tools that will capture the performance data needed for the LiQDW analysis. Data Storage will deal with handling the data produced by the collectors. Data Viewer will connect to the Data Storage and visualize the performance information.

### 13.3 Project planning

The project planning is based on the Project Initiation Documentation (from PRINCE2) produced at the beginning of the project. Several adjustments in the plan were needed during the project. Two different plans are presented, first the initial plan (Figure 41) that includes a stage for testing the framework with LiQDW, while the final plan (Figure 42) the testing part is minimal and both the domain analysis and implementation are extended.

### 13.4 Project execution

During the project, every month a Project Steering Group (PSG) meeting with the main stakeholders was held either in Rabobank or TU/e. The topics presented in these meetings are: the tasks completed last month, demonstration of the results, current progress, risk and challenges identified, and the tasks planned for the next iteration. The stakeholders can give their feedback during the whole session. At the end of the meeting, other topics that may affect the project are discussed. After the meeting, minutes are produced by the trainee. Every PSG meeting is an iteration in the project that may change its direction.

The execution of the project can be split into five stages as shown in Figure 42. Each stage has different length and internally they were organized taking into account the tasks. In the first month, meetings with the stakeholders were organized and research on the context was carried out while the necessary resources were requested (laptop, accounts, security badge).

In the project initiation, the documentation about the project was produced. It provided a solid base from where the project can be developed. Stakeholders, goals, scope, risks, and deliverables are identified and the project plan was produced.

The domain analysis was mostly research and meetings with experts from Rabobank and TU/e. During this stage, the project suffered a change of direction that affected the whole planning. Therefore, the domain analysis took longer than expected.

Around June, the design and implementation of the framework started. Although the intention was to have a long testing period where a time window to use LiQDW system would be available, the implementation was not finished on time. The local deployment scenario was used instead of a production-like environment.

The last stage was part of the wrap up process and the preparation for the defense of the project. The handover of the system and documentation was organized.
Figure 41 Initial project planning

Figure 42 Final project planning
14. Project Retrospective

In this chapter, the reflections on the project is given and the design opportunities chosen are reviewed.

14.1 Reflection

The project provided enough challenges for its nine months of duration. Both technical and organizational aspects of the project were hard. On the organizational part, extra attention and communication was needed since the supervisors and Rabobank as organization were not experienced with such projects. This was an important aspect that allowed to develop and consolidate the communication skills of the trainee.

The division between external staff and internal staff was an extra difficulty that played a decisive role in project. It was needed from the trainee to explore and understand the motivations under each stakeholder. This brought several learning points on how to deal with and behave in big organizations like Rabobank.

On the technical side, the domain of finance is new for the trainee. Most of the terminology used by the experts was unknown, therefore a broad context and problem analysis was needed. Likely, it was possible to abstract most of the problem from the financial terminology and kept it in the software domain.

The performance analysis was also something new for the trainee. One of the biggest concerns was to not be able to find a methodical approach for performance analysis that can be generic enough for the framework. However, with the help of the experts in Rabobank and an extensive research the first results were achieved. Additionally, it was the first time that the trainee was designing a framework, it has its differences with designing and implementing an application.

Overall, the project was a great experience and an opportunity to exercise the abilities of the trainee. Soft skills like communication and managing expectations were repeatedly used while the hard skills like design and research were widely needed.

14.2 Design opportunities revisited

During the analysis of the LiQDW system and the problem analysis, the design opportunities identified were usability and genericity. The usability of the framework is achieved with the Tableau software. Its interface excel in the operability and attractiveness, allowing the user a smooth and nice experience.

The genericity of the solution can be measured by the amount of artifacts and practices re-usable. The framework was built to be re-usable for any performance analysis of a data warehouse system. The artifacts can be reused if they fulfill the same internal function as originally intended. And for the practices, the systematic approach can be applied to any performance analysis, therefore it can be reused in the whole domain. In general, it is possible to reuse the framework in another data warehouse by configuring or extending some of the internal components.
# Glossary

This table contains the terminologies used throughout this report with their definition.

Table 8 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCBS</td>
<td>Basel Committee on Banking Supervision, authority body for regular cooperation on banking supervision</td>
</tr>
<tr>
<td>DBA</td>
<td>Database Administrator. Expert in databases and data related issues in charge of its administration</td>
</tr>
<tr>
<td>DMV</td>
<td>Data Management Views and Functions, as part of SQL Server, supply database and server state information</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract, Transform, and Load. Process to copy data from one or multiple data sources</td>
</tr>
<tr>
<td>FTK</td>
<td>Foundation ToolKit. Developed by Rabobank, contains several tools reused in Rabobank’s data warehouses.</td>
</tr>
<tr>
<td>HQLA</td>
<td>High Quality Liquid Assets</td>
</tr>
<tr>
<td>LCR</td>
<td>Liquidity Coverage Ratio, ratio of high quality liquid assets of an entity like a bank</td>
</tr>
<tr>
<td>LICWID</td>
<td>Rabobank’s program responsible for the data warehouse, LiQDW, system for liquidity risk data aggregation</td>
</tr>
<tr>
<td>LiQDW</td>
<td>Liquidity Data Warehouse, main system of LICWID</td>
</tr>
<tr>
<td>Liquidity</td>
<td>Ability to quickly convert an asset into cash without affecting the asset’s price</td>
</tr>
<tr>
<td>NSFR</td>
<td>Net Stable Funding Ratio</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PDEng</td>
<td>Professional Doctorate in Engineering</td>
</tr>
<tr>
<td>PerfMon</td>
<td>Windows Performance Monitor, as part of Windows, it is a tool to examine computer’s performance</td>
</tr>
<tr>
<td>PRINCE2</td>
<td>Projects IN Controlled Environments, it is a project management methodology</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System, an application that define, create, query, update and administrate relational databases</td>
</tr>
<tr>
<td>SQL Server</td>
<td>RDBMS developed by Microsoft. It is a database server with the primary function to store and retrieve data</td>
</tr>
<tr>
<td>SSIS</td>
<td>SQL Server Integration Services, component of SQL Server that perform a broad range of data migration tasks</td>
</tr>
<tr>
<td>SSMS</td>
<td>SQL Server Management Studio, developed by Microsoft, is a development tool for SQL Server’s databases</td>
</tr>
<tr>
<td>TU/e</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>USE</td>
<td>Usage, Saturation, and Errors. It is a method to collect data of a system resource in the three dimensions</td>
</tr>
</tbody>
</table>
Bibliography

References


Additional Reading


Appendix 1: Performance Analysis of LiQDW

The performance analysis will be carried out following the next steps:

1. State the performance requirements and system definition

**Performance requirements**
The goal of this performance analysis is to baseline LiQDW system. The requirement is to map the execution of the system with the basic performance information. The main focus is the responsiveness of the system and the timeliness of its internal processes.

**System boundaries**
The boundaries will be the server where LiQDW components are running. The source systems are out of scope, as well the data consumers outside the server. The Cube inside data consumers is part of the evaluation boundaries since it resides in the server.
2. List services, interfaces, and outcomes of the system

**Services**
The services available for the user are:
- to read data from the source systems when it is available
- to calculate and extract data from the Data Vault

**Interfaces**
The possible interfaces with the user are:
- Jobs available in SQL Server Agent that can execute both services
- Web portal that triggers the calculation and extraction of data from LiQDW

**Outcomes**
For the purpose of the first performance analysis, only a successful outcome will be considered for any request of a service. Failures and incorrect outcomes are out of the scope of this analysis, and therefore ignored.

3. Select the metrics and criteria for the performance evaluation

**Metrics**
From the goals stated before, the main interest is the responsiveness of the system. In order to baseline the system, the USE (Utilization, Saturation, and Errors) method is used. It focuses on three main characteristics for each hardware and software component (resource). The metrics are:

<table>
<thead>
<tr>
<th>Id</th>
<th>Metric</th>
<th>Type - Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>% Processor Time</td>
<td>U - CPU</td>
<td>% of time the processor spends to execute a non-Idle thread</td>
</tr>
<tr>
<td>02</td>
<td>% Processor Privileged</td>
<td>U - CPU</td>
<td>% of time the process threads spent executing code in privileged mode</td>
</tr>
<tr>
<td>03</td>
<td>Context switches/sec</td>
<td>U - CPU</td>
<td>Average number of switches that all processors have done in order to change from one thread to another</td>
</tr>
<tr>
<td>04</td>
<td>Processor Queue Length</td>
<td>S - CPU</td>
<td>Number of threads ready to execute in the processors queue</td>
</tr>
<tr>
<td>05</td>
<td>Available Mbytes</td>
<td>U - RAM</td>
<td>Amount of memory available to be used by the processes</td>
</tr>
<tr>
<td>06</td>
<td>Pages Input/sec</td>
<td>S - RAM</td>
<td>Number of pages read from disk to resolve a hard page fault</td>
</tr>
<tr>
<td>07</td>
<td>Pages Output/sec</td>
<td>S - RAM</td>
<td>Number of pages written to disk in order to free up space in the RAM</td>
</tr>
<tr>
<td>08</td>
<td>Transition Faults/sec</td>
<td>S - RAM</td>
<td>Number of soft page faults. Each page fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>(Physical) % Disk Time</td>
<td>U - Disk</td>
<td>% of time the disk was busy servicing a request</td>
</tr>
<tr>
<td>10</td>
<td>(Physical) Avg. Disk sec/Transfer</td>
<td>U - Disk</td>
<td>Average time that takes a disk transfer operation</td>
</tr>
<tr>
<td>11</td>
<td>(Physical) Disk Transfer/sec</td>
<td>U - Disk</td>
<td>Rate of read and write operations on disk</td>
</tr>
<tr>
<td>12</td>
<td>(Logical) % Free Space</td>
<td>U - Disk</td>
<td>% of free usable space on a logical disk drive</td>
</tr>
<tr>
<td>13</td>
<td>(Physical) Avg. Disk Queue Length</td>
<td>S - Disk</td>
<td>Average number of read/write requests queued to be served</td>
</tr>
<tr>
<td>14</td>
<td>(Physical) Current Disk Queue Length</td>
<td>S - Disk</td>
<td>Number of read/write requests queued at a point in time</td>
</tr>
<tr>
<td>15</td>
<td>Bytes Received/sec</td>
<td>U - Network</td>
<td>Number of bytes received over the network adapter</td>
</tr>
<tr>
<td>16</td>
<td>Bytes Sent/sec</td>
<td>U - Network</td>
<td>Number of bytes sent over the network adapter</td>
</tr>
<tr>
<td>17</td>
<td>Output Queue Length</td>
<td>S - Network</td>
<td>Outbound packets queued to be sent</td>
</tr>
<tr>
<td>18</td>
<td>Packets Outbound Discarded</td>
<td>S - Network</td>
<td>Number of outbound packets chosen to be discarded even though no errors had been detected</td>
</tr>
<tr>
<td>19</td>
<td>Packets Received Discarded</td>
<td>S - Network</td>
<td>Number of inbound packets chosen to be discarded even though no errors had been detected</td>
</tr>
<tr>
<td>20</td>
<td>Packets Outbound Errors</td>
<td>E - Network</td>
<td>Number of outbound packets not transmitted because of errors</td>
</tr>
<tr>
<td>21</td>
<td>Packets Inbound Errors</td>
<td>E - Network</td>
<td>Number of inbound packets not delivered to higher-layer protocol because of errors</td>
</tr>
<tr>
<td>22</td>
<td>(sqlservr) % Processor Time</td>
<td>U - CPU (SQL)</td>
<td>% of processor time spent on SQL Server process threads</td>
</tr>
<tr>
<td>23</td>
<td>(msmdsrv) % Processor Time</td>
<td>U - CPU (SQL)</td>
<td>% of processor time spent on SSAS process threads</td>
</tr>
<tr>
<td>24</td>
<td>(MsDtsSrv) % Processor Time</td>
<td>U - CPU (SQL)</td>
<td>% of processor time spent on SSIS process threads</td>
</tr>
<tr>
<td>25</td>
<td>(SQLAGENT) % Processor Time</td>
<td>U - CPU (SQL)</td>
<td>% of processor time spent on SQL Agent process threads</td>
</tr>
<tr>
<td>26</td>
<td>Batch requests/sec</td>
<td>U - CPU (SQL)</td>
<td>Number of SQL batch requests received by the server</td>
</tr>
<tr>
<td>27</td>
<td>Compilation/sec</td>
<td>U - CPU (SQL)</td>
<td>Number of SQL Server compilations</td>
</tr>
<tr>
<td>28</td>
<td>Recompilation/sec</td>
<td>U - CPU (SQL)</td>
<td>Number of SQL Server recompilations</td>
</tr>
<tr>
<td>29</td>
<td>Total Server Memory (KB)</td>
<td>U - RAM (SQL)</td>
<td>Total amount of dynamic memory the server is currently consuming</td>
</tr>
<tr>
<td>30</td>
<td>Target Server Memory (KB)</td>
<td>U - RAM (SQL)</td>
<td>Ideal amount of dynamic memory the server is willing to consume</td>
</tr>
<tr>
<td>31</td>
<td>Buffer Cache Hit Ratio</td>
<td>U - RAM (SQL)</td>
<td>% of page requests satisfied by data pages already in the buffer pool</td>
</tr>
<tr>
<td>32</td>
<td>Memory Grants Pending</td>
<td>S - RAM (SQL)</td>
<td>Current number of processes waiting for a workspace memory grant</td>
</tr>
<tr>
<td>33</td>
<td>Free list stalls/sec</td>
<td>S - RAM (SQL)</td>
<td>Number of requests that had to wait for a free page</td>
</tr>
<tr>
<td>34</td>
<td>Lazy Writes/sec</td>
<td>S - RAM (SQL)</td>
<td>Number of times per second SQL Server relocates dirty pages from buffer pool (memory) to disk</td>
</tr>
<tr>
<td>35</td>
<td>Page Life Expectancy</td>
<td>S - RAM (SQL)</td>
<td>Number of seconds a page will stay in the buffer pool without refer-</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Source</td>
<td>Notes</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>36</td>
<td>Checkpoint Pages/sec</td>
<td>S - RAM (SQL)</td>
<td>Number of pages that are moved from buffer to disk per second during a checkpoint process</td>
</tr>
<tr>
<td>37</td>
<td>Lock Requests/sec</td>
<td>U - Locks (SQL)</td>
<td>New locks or conversions made by the lock manager. (scope: Total, RID and Metadata)</td>
</tr>
<tr>
<td>38</td>
<td>Lock Waits/sec</td>
<td>S - Locks (SQL)</td>
<td>Lock requests that had to wait for the lock to be granted</td>
</tr>
<tr>
<td>39</td>
<td>Average Wait Time (ms)</td>
<td>S - Locks (SQL)</td>
<td>Average amount of time spent in a wait lock requests</td>
</tr>
<tr>
<td>40</td>
<td>Lock Wait Time (ms)</td>
<td>S - Locks (SQL)</td>
<td>Total wait time for locks ended only in the last second</td>
</tr>
<tr>
<td>41</td>
<td>Number of Deadlocks/sec</td>
<td>E - Locks (SQL)</td>
<td>Number of lock requests resulted in a deadlock</td>
</tr>
<tr>
<td>42</td>
<td>Workfiles Created/sec</td>
<td>U - Disk (SQL)</td>
<td>Number of work files created per second, usually as a part of tempdb processing with hashing operations</td>
</tr>
<tr>
<td>43</td>
<td>Full Scans/sec</td>
<td>U - Disk (SQL)</td>
<td>Number of full scans on tables or indexes</td>
</tr>
<tr>
<td>44</td>
<td>Page splits/sec</td>
<td>S - Disk (SQL)</td>
<td>Number of times SQL Server had to split a page when updating or inserting data due to fragmentation</td>
</tr>
<tr>
<td>45</td>
<td>Databases Data File(s) Size (KB)</td>
<td>U - Disk (SQL)</td>
<td>Cumulative size of all the data files in the databases</td>
</tr>
<tr>
<td>46</td>
<td>Databases Log File(s) Size (KB)</td>
<td>U - Disk (SQL)</td>
<td>Cumulative size of all the log files in the databases</td>
</tr>
<tr>
<td>47</td>
<td>SQL Server Errors/sec</td>
<td>E - SQL Server</td>
<td>Number of errors/sec, it combines different types of errors</td>
</tr>
<tr>
<td>48</td>
<td>SQL Misguided plan executions/sec</td>
<td>E - SQL Server</td>
<td>Number of plan executions in which a plan could not be honored during plan generation. Normal compilation was used to generate the executed plan</td>
</tr>
<tr>
<td>49</td>
<td>SQL Attention rate/sec</td>
<td>E - SQL Server</td>
<td>Number of cancels and query timeouts occurring per second</td>
</tr>
<tr>
<td>50</td>
<td>Database transaction count</td>
<td>U - SQL Server</td>
<td>Amount of transactions in a specific state (grouped by current state)</td>
</tr>
<tr>
<td>51</td>
<td>Cached pages count</td>
<td>U - SQL Server</td>
<td>Amount of cached pages per database</td>
</tr>
<tr>
<td>52</td>
<td>Context switches count</td>
<td>U - SQL Scheduler</td>
<td>Number of context switches that have occurred on a scheduler</td>
</tr>
<tr>
<td>53</td>
<td>Current tasks count</td>
<td>U - SQL Scheduler</td>
<td>Number of current tasks that are associated with a scheduler</td>
</tr>
<tr>
<td>54</td>
<td>Work queue count</td>
<td>S - SQL Scheduler</td>
<td>Number of tasks in the pending queue waiting to be picked by a worker</td>
</tr>
<tr>
<td>55</td>
<td>Runnable tasks count</td>
<td>S - SQL Scheduler</td>
<td>Number of workers with tasks waiting to be scheduled</td>
</tr>
<tr>
<td>56</td>
<td>Pending disk IO count</td>
<td>S - SQL Scheduler</td>
<td>Number of pending IOs per scheduler that are waiting to be completed</td>
</tr>
<tr>
<td>57</td>
<td>Waiting tasks count</td>
<td>S - SQL Server</td>
<td>Number of waits on SQL Server (grouped by type)</td>
</tr>
<tr>
<td>58</td>
<td>Wait time ms</td>
<td>S - SQL Server</td>
<td>Total wait time for this wait type in milliseconds (grouped by type)</td>
</tr>
<tr>
<td>59</td>
<td>Max wait time ms</td>
<td>S - SQL Server</td>
<td>Maximum wait time on a wait type</td>
</tr>
<tr>
<td>60</td>
<td>Total Size MB</td>
<td>U - SQL</td>
<td>Total amount of preempted memory</td>
</tr>
</tbody>
</table>
database

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Unit</th>
<th>SQL database</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>Allocated extent size MB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>Unallocated extent size MB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>Lock request status</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Identify available parameters

**Parameters**

The system parameters that influence the experiment are:

- CPU Characteristics (amount of processors, clock speed, cache size, shared cache, host-bus speed)
- Disk (HDD or SSD, local or SAN, access time, max. IO operations/sec)
- RAM (capacity, timings)
- Network (Dev or Prod network, response time, package lost)
- DB configuration (SQL Server administration)
- OS configuration (Windows Server administration)
- Other processes running in parallel (Virus scanners, other products)

The workload parameters can be separated in the next categories:

- Number of users
- Order of the requests
- Size of the requests

5. Select factors from the parameters

**Factors**

No factors will be varied in this experiment. After the first results, some parameters will be selected as factors to be varied within the next experiment.

6. Select evaluation technique

The criteria for selecting an evaluation technique for the experiment is available in the table below:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Analytical Modeling</th>
<th>Simulation</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage of system</td>
<td>Any</td>
<td>Any</td>
<td>Prototype</td>
</tr>
<tr>
<td>Time required</td>
<td>Small</td>
<td>Medium</td>
<td>Varies</td>
</tr>
<tr>
<td>Tools</td>
<td>Programs</td>
<td>Instruments</td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Trade-off factor/levels</td>
<td>Easy</td>
<td>Moderate</td>
<td>Difficult</td>
</tr>
<tr>
<td>Cost</td>
<td>Small</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Salability</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

Taking into account that LiQDW is already deployed in production, the first row doesn't exclude the measurement technique. Although analytical modeling (Queuing theory) seems to have many advantages, the accuracy and salability is a considerable downside and a big reason to avoid it for the first experiment. Furthermore, the description of the system available should be consistent and detailed for an analytical model or the results will deviate from reality. This is definitely a problem with the current amount (and level of abstraction) of the information accessible.

The simulation is also avoided because the real system is already developed and no benefits are foreseen developing a new simulation. Finally the Measurement is the choice for this experiment since there are many benefits like salability and accuracy of the result against few disadvantages as the trade-off that doesn't affect because no
factors are selected in the previous step five (there are no factor values to compare performance in this experiment).

7. Select workloads

Workloads
The workload of this experiment is the test case used by the developers to check the correct integration of all features within LiQDW that are used when a user requests data. This test contains a minimal set of synthetic data to execute all the modules necessary to extract data from the Data Vault.

The test, BR_IntegrationFeature, is part of the test suite for integration testing. The first part of the test is to insert the artificial data inside the Data Vault and Staging. Then, it triggers two requests in the Rule Vault that will extract data to the Data mart and the Cube. The first request is for a monthly reporting example and the second is for a daily reporting example.

8. Design experiment

Environment
The environment consists of a development machine where the framework is installed locally with the LiQDW system. A more detailed list of the hardware and software factors is represented with the next table.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Development Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Dell OptiPlex 9020</td>
</tr>
<tr>
<td>CPU type</td>
<td>Intel i7 – 4790</td>
</tr>
<tr>
<td>CPU speed</td>
<td>3600MHz</td>
</tr>
<tr>
<td>CPU amount</td>
<td>1</td>
</tr>
<tr>
<td>CPU cores</td>
<td>4</td>
</tr>
<tr>
<td>RAM</td>
<td>16GB</td>
</tr>
<tr>
<td>RAM speed</td>
<td>1600MHz</td>
</tr>
<tr>
<td>RAM modules</td>
<td>2x 8GB</td>
</tr>
<tr>
<td>Disk</td>
<td>256 GB (SSD)</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 7 Enterprise x64 SP1</td>
</tr>
<tr>
<td>SQL Server version</td>
<td>2012 Enterprise SP2</td>
</tr>
<tr>
<td>Instances</td>
<td>1x SSDB, 1x SSAS, 1x SSRS</td>
</tr>
</tbody>
</table>

Instruments
The tools used with the framework to collect data for the intended metrics are:
- Windows Performance Monitor (PerfMon) for the metrics 1 to 49.
- Dynamic Management Views and Functions (DMV) for the metrics 50 to 63
- Logging schema of LiQDW for the details of the executed processes inside LiQDW.

The scenarios of before, during, and after the experiment are explained and represented in the next three figures.
First, in the pre-execution of the experiment, several elements need to be configured. Those are highlighted in green in the figure below. An admin user with the necessary rights is needed. The SQL Agent Server needs to have the jobs to execute LiQDW, execute PerfMon, the stored procedures to retrieve DMV’s information, and stored procedure to retrieve logging information from LiQDW. PerfMon needs to have registered the data collectors to capture data for the metrics selected. PerfDB must have the internal structure (tables) generated to support the data sent by the data collectors.
Once everything is setup, the execution of the experiment begins. The figure below shows the flow of control and data during the test. The SQL Agent will trigger the execution of PerfMon, the DMV collectors, and LiQDW execution (and in that order).

Later, once the execution of the test is finished, it is necessary to collect the accumulative data from LiQDW (Logging) and from DMV. Last step is to import the data from the binary files to PerfDB. Then the user can access to the performance data centralized in PerfDB. A workbook from Tableau can be used to visualize the performance information.
9. Analyze the data from the experiment

With the data obtained in the experiment and Tableau software, the analysis can be carried out. Following this, some of the data analysis is presented with a workbook from Tableau. It will contain a dashboard where data can be easily visualized and it is composed of different views (charts) of the system. In step 10 the example result.

10. Present results
Appendix 2: Installation Manual

Steps to install and configure the performance framework. Scripts are available to automate most of the steps, however they are not included in this report.

1. **Home directory**
   Create an empty folder "\%systemdrive\%\PAF\". Check requirements of growth in step 2 if not sure how much space is needed in the drive selected. No script provided for this task.

2. **Create a new Database, PerfDB**
   Create a new database with name “PerfDB”. It will keep all data centralized and allow an easy uninstallation. With the performance analysis of Appendix 1: Performance Analysis of LiQDW, the parameters of growth are:
   - PerfDB (without inserting PerfMon data): 8MB/h (Using interval of 2 sec)
   - PerfMon Binary log files: 15MB/h (1800 captures/h – 1 capture per 2 sec)
   - All values are rounded up, so 3 days non-stop = 1.6 GB. Recommended to reserve at least 3GB.
   No script is provided. This should be handled by the Database Administrator (DBA).

3. **Users**
   In order to configure and run the tools installed, the user in charge of executing the performance analysis experiment needs several rights.
   - User must be dbowner or admin of PerfDB to allow installation
   - User must have VIEW SERVER STATE permission on SQL Server instance (for DMVs)
   - User must have VIEW DATABASE STATE permission on LiQDW databases (for DMVs)
   - User must have SQL Server Agent operator role (for the jobs)
   - User must belong to the Performance Log Users and Performance Monitor Users group.

4. **Install objects inside PerfDB**
   Create the objects of PerfDB that will support the storage and aggregation of the data collected.
   - Use *MasterScriptInstall.sql*: It will prepare the metadata for the metrics specified in Appendix 1: Performance Analysis of LiQDW. First it will create the Schemas PerfMeta and PerfData. Then it will insert the metadata in PerfMeta.PerfTable and PerfMeta.TableauIf.
   - Stored procedures to support basic functionality of the framework (several scripts)
   - Last use *GeneratePerfStructure.sql*: generates the internal structure (tables and views) of PerfData.

5. **SQL Server Agent jobs**
   The jobs inside SQL Server Agent can be created using SQL scripts. First it is important to check the @server_name (set to "(local)") and @owner_login_name (set to "sa")
   - *CollectDMVsContinuos*: copies data from the DMVs every interval, the job run in an infinite loop. It requires manual stop or a schedule.
   - *CollectDMVsAccumulated*: copies data from the DMVs single time. The job finishes after executing once.
   - *CollectLiQDWMetaData*: copies the logging data from LiQDW system. The job finishes after executing all the steps.
   - *StartPerfMonDataCollector_ALL*: starts

6. **PerfMon Data Collectors**

7. **Bat files**
8. Typical scenarios
9. Uninstallation
About the Author

Vladimir Mikovski received his MSc degree in Computer Science and Software Engineering (2012) from Complutense University of Madrid. His master thesis “Muphic: Image-Based Automatic Musical Composition” studies the relationship between images and sound through synesthesia using algorithmic composition. It was awarded third place in Sopra Group competition. After his graduation he worked for two years as a software engineer for a Spanish company specialized in database systems. He has published a few android applications on the market as independent projects.
4TU.School for Technological Design, Stan Akkermans Institute offers two-year postgraduate technological designer programmes. This institute is a joint initiative of the four technological universities of the Netherlands: Delft University of Technology, Eindhoven University of Technology, Universty of Twente and Wageningen University. For more information please visit: www.4tu.nl/sai