Eindhoven University of Technology

MASTER

Human resources analytics at Viggo
warehousing solutions for CSV data

Triantos, K.

Award date:
2016

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

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

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 14. Sep. 2026

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Web Engineering Group

Human Resources analytics at Viggo: warehousing solutions for CSV data

Master Thesis

Konstantinos Triantos

Supervisors:
dr. G.H.L. Fletcher (Eindhoven University of Technology)
ir. G. Oerlemans (Viggo Eindhoven Airport B.V.)
M.Sc. J. Melissen (Viggo Eindhoven Airport B.V.)

Examination Committee:
dr. P.M.E. De Bra
dr. G.H.L. Fletcher
ir. G. Oerlemans
dr. A. Serebrenik

Eindhoven, November 2016

Abstract

This master thesis focuses on the design, the implementation and the analysis of a data warehouse, which is specialized in HR analysis, within Viggo Eindhoven Airport, the largest service provider in Eindhoven Airport. More specifically, within Viggo there is a spreadsheet overuse, which prevents the HR and BA departments to make fast and accurate decisions, regarding the development of Viggo's manpower. Because of the fact that it is difficult for Viggo to shift to a completely new software system, which will provide better management of the stored to the spreadsheets information, an alternative solution should be investigated. This work recommends a data warehouse, which converts the existing spreadsheets into a fully centralized data repository, focused on HR department business analysis, as this alternative. Precisely, within this thesis project, two problems are discussed. The former is the data warehouse design, based on HR analysis and the latter is regarding ETL solutions for spreadsheet-based sources. Regarding the ETL solutions, an ETL framework, which uses a query mechanism for CSV spreadsheets, has been designed. The introduced ETL framework is called CSVQL and is a potential query language. The work load is split into two major parts. The former focuses on Viggo's data warehouse design and the latter investigates ETL solutions according to Viggo's data sources. With the use of the aforementioned data warehouse system, which transforms the raw information into meaningful charts, Viggo can easily produce solutions regarding decision making with low cost and without the need to switch into a new software system. Moreover, the available OLAP servers in the market, can utilize the views of the stored information, in a manner, which can unmask problems or give answers to difficult dilemmas, with respect to Viggo's requirements.

Preface

This report concludes the results of my graduation project, regarding the "Computer science & engineering" M.Sc. program at the department of "Computer Science & Mathematics" in Eindhoven University of Technology. This project was performed within the "Web Engineering" group of Eindhoven University of Technology, in collaboration with Viggo Eindhoven Airport B.V. In completing this graduate project, I have been fortunate to receive help, support and encouragement from many people and I would like to acknowledge them for their cooperation.

First of all, I would like to thank my graduation supervisor dr. G.H.L. Fletcher for guiding me through, not only on this project, but during my whole studies in the Netherlands. The collaboration during the lectures, the seminars and my master project has been a pleasant and a great learning experience for me.

In addition, I would like to express my gratitude to my supervisors ir. G.Oerlemans and M.Sc. J.Melissen, who gave me the opportunity to not only start working on this project, within Viggo, but also for their assistance and support on a daily basis all these months. I am truly grateful for the fact that you were my first supervisors in the beginning of my career.

Furthermore, I would like to thank all my friends, my classmates and my colleagues for the help and the positive energy I received during my first years in the Netherlands. Being far away from home is difficult, but you made me overcome all the difficulties I faced and to consider Eindhoven as my new home.

In conclusion, I would like to thank my parents for their unconditional love and support, all these 27 years. All the care they have provided me over the years is the greatest gift that anyone has ever given me. Last but not least, I would like to say thanks to pumba, my half-dog, half-human best friend, who made me realize my own potential as a person and as a friend.

Konstantinos Triantos,
Eindhoven,
November 2016

Contents

Contents vii

List of Figures xi

List of Tables xiii

1 Introduction 1
- 1.1 Current state of art . . . 1
- 1.2 Problem statement . . . 2
- 1.3 Thesis contribution . . . 4
- 1.4 Thesis outline . . . 4

2 Preliminaries 5
- 2.1 Database management systems . . . 5
- 2.2 Data warehouse concepts . . . 6
- 2.3 Data warehouse architecture . . . 7
  - 2.3.1 Data sources . . . 8
  - 2.3.2 Back-end tier . . . 8
  - 2.3.3 Data warehouse tier . . . 8
  - 2.3.4 OLAP tier . . . 9
  - 2.3.5 Front-end tier . . . 9
- 2.4 Management of Comma-Separated Values data . . . 9
  - 2.4.1 Definition . . . 9
  - 2.4.2 Data querying . . . 10

3 Data warehouse overview 13
- 3.1 Solution . . . 13
- 3.2 Benefits of data warehouses . . . 13
  - 3.2.1 Data warehouses as spreadsheets overuse cure . . . 14
  - 3.2.2 Data warehouses as data repositories . . . 15
- 3.3 Viggo data warehouse . . . 16
  - 3.3.1 Architecture . . . 16
  - 3.3.2 Data sources . . . 16
  - 3.3.3 Back-end tier . . . 17
  - 3.3.4 Data warehouse tier . . . 17
  - 3.3.5 OLAP tier . . . 17
  - 3.3.6 Front-end tier . . . 17

4 Data sources analysis 19
- 4.1 Viggo Shift Rosters . . . 19
  - 4.1.1 Functionality . . . 19
  - 4.1.2 Schema . . . 19
- 4.2 Viggo Training Managers . . . 20

4.2.1 Functionality . . . 20
4.2.2 Schema . . . 20
4.3 Youforce database . . . 21
4.3.1 Functionality . . . 21
4.3.2 Schema . . . 21

5 Back-end tier **23**
5.1 ETL process . . . 23
5.1.1 Global problems and constraints . . . 23
5.1.2 Extraction & transportation . . . 24
5.1.3 Transformation & cleaning . . . 24
5.1.4 Loading . . . 25
5.1.5 ETL in Viggo . . . 25
5.2 CSVQL . . . 26
5.2.1 Clauses overview . . . 26
5.2.2 SELECT . . . 27
5.2.3 FROM . . . 27
5.2.4 WHERE . . . 29
5.2.5 OUTPUT . . . 30
5.2.6 Querying Viggo Training Manager . . . 31
5.3 CSVQL Implementation . . . 35
5.3.1 Concept idea . . . 35
5.3.2 Implementation architecture . . . 36
5.3.3 Data Load script . . . 36
5.3.4 Query MySQL mapping . . . 36
5.4 Experimental study . . . 38
5.4.1 Scenario . . . 38
5.4.2 Experiment set-up . . . 38
5.4.3 Implementation . . . 40
5.4.4 Experiment results . . . 40

6 Data warehouse tier **43**
6.1 Design assumptions . . . 43
6.1.1 Dimensions . . . 44
6.1.2 Measures . . . 45
6.2 Conceptual level . . . 45
6.2.1 Basic concepts . . . 45
6.2.2 ER diagram . . . 46
6.2.3 Entities . . . 47
6.2.4 Relationships . . . 49
6.3 Logical level . . . 50
6.3.1 Mapping rules . . . 51
6.3.2 ER diagram mapped to relational schema . . . 51
6.4 Physical level . . . 51

7 OLAP & Front-end tier **53**
7.1 IcCube server . . . 53
7.1.1 Communication with Data warehouse tier . . . 53
7.1.2 Cube builder . . . 54
7.1.3 Cube querying . . . 54
7.2 Web reporting tools . . . 55
7.2.1 Bar chart . . . 56
7.2.2 Line chart . . . 56
7.2.3 Bullet chart . . . 56

7.2.4 Heat map 57

8 Conclusions **59**
8.1 Organizational 59
8.2 Academic 59
8.3 Future Work 60
8.3.1 Organizational 60
8.3.2 Academic 60

Bibliography **63**

Appendix **65**

A Data Load script **65**
A.1 CSVQL.java 65
A.2 LoadCSV.java 66

B CSVQL experiment results **70**
B.1 Primary index 70
B.2 Secondary index 70

List of Figures

2.1 A three-dimensional cube regarding sales . . . 6
2.2 A data warehouse architecture . . . 7
2.3 Samples obtained by a 20 Hz GPS sensor traces. Each line of this file is a time-stamped record. The first line is the header line. . . . 10

3.1 Viggo data warehouse architecture. . . . 16

4.1 A sample of Viggo Shift Rosters' schema. . . . 20
4.2 A sample of Viggo Training Managers' schema. . . . 20
4.3 A sample of Youforce export spreadsheets' schema. . . . 22

5.1 The logo of CSVQL framework. . . . 26
5.2 Select clause example 1 . . . 27
5.3 Select clause example 2 . . . 27
5.4 Select clause example 3 . . . 27
5.5 From clause example 1 . . . 28
5.6 From clause example 2 . . . 28
5.7 From clause example 3 . . . 28
5.8 From clause example 4 . . . 28
5.9 Where clause example 1 . . . 30
5.10 Where clause example 2 . . . 30
5.11 Output clause example 1 . . . 30
5.12 Output clause example 2 . . . 31
5.13 The relational structure of Viggo training.csv file, after CSVQL application. . . . 31
5.14 Querying Viggo training.csv file . . . 32
5.15 The f1 row-by-row path within Viggo training.csv file. . . . 32
5.16 The f2 row-by-row path within Viggo training.csv file. . . . 33
5.17 The f3 row-by-row path within Viggo training.csv file. . . . 33
5.18 The f4 row-by-row path within Viggo training.csv file. . . . 34
5.19 CSVQL implementation Achritecture . . . 36
5.20 Experiment's input as a CSV grid . . . 38
5.21 Experiment's input after Data Load Script operation. . . . 39
5.22 Benchmark query with one join on Primary key . . . 39
5.23 Benchmark query with one join on Secondary key . . . 39
5.24 Benchmark query with eight joins on Primary key . . . 40
5.25 Benchmark query with eight joins on Secondary key . . . 40
5.26 CSVQL performance for joins on primary key per experiment . . . 41
5.27 CSVQL performance for joins on primary key per join . . . 41
5.28 CSVQL performance for joins on secondary key per experiment . . . 42
5.29 CSVQL performance for joins on secondary key per join . . . 42

6.1 The hierarchy of calendar dimension. . . . 44

6.2 The hierarchy of human resources dimension. 44
6.3 The hierarchy of skills training dimension. 44
6.4 The entity-relationship diagram of Viggo data-warehouse. 46
6.5 The entity-relationship diagram of Viggo data-warehouse after reverse engineering process on physical level. 52

7.1 MDX query example 55
7.2 Web reporting tool overview. 55
7.3 Bar chart regarding payment in Viggo Cleaning department. 56
7.4 Bar chart regarding payment in Viggo Cleaning department, per function. 56
7.5 Line chart regarding payment in Viggo Cleaning department. 56
7.6 Line chart regarding payment in Viggo Cleaning department, per function. 56
7.7 Bullet chart regarding payment in Viggo Cleaning department. 57
7.8 Bullet chart regarding payment in Viggo Cleaning department, per function. 57
7.9 Heat map regarding payment in Viggo Cleaning department, per date. 57

List of Tables

5.1 CSVQL Clauses 26
5.2 Results of CSVQL scalability experiment with Primary index 41
5.3 Results of CSVQL scalability experiment with Primary index 42

B.1 Results of CSVQL scalability experiment with Primary index 70
B.2 Results of CSVQL scalability experiment with Secondary index 70

Abbreviations

Abbreviations	Meaning	First occurrence
BA	Business Analysis	p.2
CSV	Comma-Separated Values	p.4
CSVQL	Comma-Separated Values Query Language	p.4
DBMS	Database Management Systems	p.4
ETL	Extract, Transformation and Loading	p.4
HR	Human Resources	p.2
MDX	MultiDimensional eXpressions	p.17
OLAP	On-line Analytical Processing	p.1
OLTP	On-Line Transaction Processing	p.6
RDBMS	Relational Database Management Systems	p.5
SQL	Structured Query Language	p.4
W3C	World Wide Web Consortium	p.10
XML	Extensible Markup Language	p.8

Chapter 1

Introduction

1.1 Current state of art

Nowadays, business industry is encountering complex and demanding challenges, in terms of data analysis. More specifically, each organization impels its workforce to analyze even the minor detail, which may give either a slight either a critical advantage against its competitors. Consequently, people, in this kind of organizations, struggle with data analysis tools and try to utilize each decision, which will lead them to meet the demands of their operational goals. The process, which is described above is the application of "business intelligence" in practice.

Business intelligence encompasses all the methodologies, the processes, and the technologies, which can transform raw data into variable information for decision making. More specifically, business intelligence helps managers of numerous organizational levels by supporting them on analyzing information. A business intelligence system collects big amounts of raw data and transforms them into a form, which can be used for business analysis. The process of this data transformation is split into a set of smaller processes, which extract the source data and after applying transformation, integration, and cleansing tasks on them, they store the output in a database, which is called "data warehouse" [6].

A data warehouse is a database, which has been implemented as an integral part of a business intelligence system. Data warehouses are data stores, where the data are stored in a structure, which facilitates the user to receive fast and accurate answers to complex queries, regarding these data. In recent times, business community offers a wide variety of mechanisms and systems that operate on retrieval, analysis and visualization of the data, which are stored inside data warehouses. The most popular among all of them is the "On-Line Analytical Processing" (OLAP) [6].

OLAP systems offer the power to its users to query the data of a data warehouse and to study them from multiple perspectives. In other words, OLAP systems contribute on the decision making process by enabling their users to obtain and analyze the information that they need, in a multilevel manner. Because of the fact that business intelligence is a new born science, it grows exponentially, day by day, compared with the rest of scientific community. As a result, new business intelligence techniques appear and are applied in industrial field. This rapid evolution of business intelligence can offer powerful tools, which can be applied in the data analysis field [6]. However, this work focus only on the current state of business intelligence and more specifically, on data management challenges regarding data analysis.

Data management is a broad term, but within this work it indicates all the means, which operate on the data, as a resource of information for data analysis. More specifically, in this thesis, a research takes place regarding the design, the implementation and the use of a data warehouse for human resource analysis. In the following chapters, a real life case study with reference to human resource management is introduced and according to this case a quest on the data warehouse world begins. Precisely, in this report, all the steps, which are followed to build a data warehouse are introduced and an attempt has been made to contribute to the existing

knowledge and technology.

1.2 Problem statement

Viggo Eindhoven Airport B.V. is the largest service provider in Eindhoven Airport, with 42-years history, and offers total solutions in ground handling, cargo, security and cleaning services. More specifically, every year, Viggo handles around 28 thousand flights and guarantees comfort and security for more than 4 million passengers inside Eindhoven Airport. For this purpose, Viggo employs more than 6 hundred operators, who cleverly integrate the existing links in the service chain, on a daily basis. As every company that wants to excel, Viggo collects and analyzes big amount of data regarding its employees, in order to improve its services and to keep the leadership among its competitors.

Subsequently, Human Resources (HR) and Business Analysis (BA) departments apply analytic processes, in order to improve employee performance and, therefore, getting a better return on their investment. According to this ambition, Viggo administration should not just deal with gathering data, but should aim to provide insights and metrics, which can be retrieved from these sources of information. These specific metrics are crucial, because they unravel all the meaningful information, which is concealed inside the data and will assist Viggo to make relevant decisions.

The process, which will produce the metrics that the BA department tends to analyze, requires an efficient structure, in which the data sources are formulated. However, this structure does not exist and the reason is the fact that data sources in Viggo are escalated inside spreadsheets. More specifically, there is a spreadsheet overuse that concludes files with enormous amount of information, which is difficult or impossible to be analyzed. This spreadsheet overuse is supported by the fact that, inside Viggo, employees from different departments and with different background seem to be eager to interact with spreadsheets, rather than any other form of data representation. Precisely, within Viggo, the "Export to Excel" joke, which "is the third most common button in data and business intelligence apps... after OK and Cancel" is not far away from reality [5]. As a result, Viggo has a huge amount of data, but at the same time, there is a lack of knowledge about its employees. The main reason is the fact that spreadsheets may demonstrate the information via a graphical user-friendly interface with direct manipulation tools, but they lack essential database functionality. More specifically, spreadsheets assert the following disadvantages:

Difficult to keep up-to-date Keeping a spreadsheet up to date is often a commotion, because there are no automatic mechanisms, which guarantee that the enclosed information is always the appropriate. This commotion can be exponentially worse as the volume of stored data enlarges and especially when more than one users manipulate the same spreadsheet. The only solution, which can secure that the current information is the most recent is the manually inspection. Nowadays, this process is assisted by macros - automated input sequences, which imitate keystrokes or mouse actions. However, there is no audit technique, which secures that data in a spreadsheet are always synchronized [11]. As a result, taking into account that Viggo operates on thousands of passengers and hundreds of flights on daily basis, it is concluded that unnecessary time and effort are wasted updating spreadsheets in order to keep data up to date.

Vulnerable to fraud Since, this work has been taken place inside the business industry, this spreadsheet disadvantage has the potential to be the most damaging, among all of the spreadsheet disadvantages, which are listed. Nowadays, the most popular system to design, edit and maintain spreadsheets is the Microsoft Excel software. Deceitful manipulations in a company's Microsoft Excel files have resulted in multimillion-dollar losses [10]. The main reason, which causes this vulnerability is the lack of mechanisms, which can control who alters formulas, values, or dependencies. As a result, a user can modify crucial modules without being detected [11].

Prone to human errors Apart from the fraud threat, there is another more significant threat, which can conclude the same disaster as fraud. This threat is the built-in inefficiency of

spreadsheets to prevent even trivial human errors. More specifically, spreadsheet systems do not support functions for validation and, thus, human mistakes can occur critical errors. A missed negative sign or a misaligned row might sound harmless, but they may affect a whole payment plan and cause a loss of thousands of euros. Therefore, valuable time is consumed to validate and to track information, especially when the spreadsheet contains big volumes of data. Moreover, a hardware failure or an external threat can potentially disorganize or even steal all the work and destroy a whole business analysis [11].

Unfit for agile business practices Since there are no standards regarding spreadsheet development and because of the fact that spreadsheets are normally created by individuals, who have not any knowledge about data management and software documentation, the majority of the business world faces the same problem: personalized spreadsheets. More specifically, when a company do not provide spreadsheet templates and directions for spreadsheet development to its employees, the spreadsheet files become highly personal and individual. As a result, each spreadsheet is built independently and is affected by the background or the applications that its developer uses. Therefore, a new user, who tries to take over this kind of spreadsheets, should start from scratch, in order to understand the structured information, which these spreadsheets enclose [11]. Inspecting Viggo case, there are around 20 different spreadsheet structures, which have been resulted during the years, because of the different needs between company's departments and after Microsoft Excel system's recommendations.

Not designed for collaborative work A spreadsheet consists a powerful and efficient tool for every professional. However, spreadsheets behave efficient until more users start operating on and editing it. Precisely, after a spreadsheet becomes shared, horrific chaos and confusion often occurs. Since, there is no system to indicate who commit any change and when this change occurred, anyone can be blamed for any mistake, which eventuates. In Viggo, employees cooperate every day in many projects, regarding planning, forecasting, budgeting, and reporting. All these activities are collaborative by nature, because they require information from different individuals, who belong to different departments. Experience will tell that these collaborating processes tend to duplicate erroneous data. Consequently, team members will face difficulties on keeping track of similar files and sometimes even end up sending or working on a wrong version [11].

Scale poorly As an organization, like Viggo grows, data, which are stored in spreadsheet-based systems, become more distributed, according the time. This means that the outlined disadvantages affect the work in a higher rate than normally. Moreover, spreadsheet systems are not designed for big volume of data and as a result their performance is affected is a negative manner, resulting lower efficacy and higher response time. Hence, it could be concluded that it is not prudent for a large organization to keep relying on spreadsheets, for business and data analysis in general [11].

Hard to consolidate As it mentioned above, spreadsheets are highly favored by end users, when the requirements are bounded by simple data entries or quick ad-hoc data analysis tasks. Precisely, this is the reason that spreadsheets consist of one of the most popular office tools in business field. Consequently, data in spreadsheet-based systems are distributed throughout companies and organizations. This fact results a slow consolidation process, when there is a need to generate a report, which requires data from multiple data sources. More specifically, in most of the cases, end users have to collect data from multiple files, to summarize them and, finally, to submit them through emails, portable storage media or by upload them to a commonly shared network folder. This process must take place as many times as it is needed, till the information reaches the department or the company top decision makers. Throughout this entire consolidation process, data is subjected to numerous error-prone and filtering activities such as copy-pasting, cell entry, and range specification, which can conclude different than the actual results [11].

This concludes that spreadsheet overuse prevents the HR and BA departments to make fast and accurate decisions, regarding the development of Viggo's manpower. Moreover, there is no ambition, within the company, to shift to appropriate contemporary software solutions, which will provide better management of the stored to the spreadsheets information. As a result, at this moment, it is impossible for Viggo to analyze data and make decisions.

1.3 Thesis contribution

The aforementioned set of problems can be treated by importing into and performing analytics within Database Management Systems (DBMS). These systems are able to collect big amounts of data and organize them in a structure, which can be used, in order to analyze organizational behavior, from a business analyst perspective. This result is achieved by the use of a set of tasks, which retrieve data from sources and, by means of extraction, transformation and integration processes, store the data in a centralized DBMS, which is called as data warehouse.

This master thesis focuses on the design, the implementation and the analysis of a data warehouse, which is specialized in HR analysis, within Viggo. More specifically, in this work, **we have designed a data warehouse, which converts the existing spreadsheets into a fully centralized data repository, focused on HR department business analysis**. This specific data warehouse integrates information from Viggo's internal data sources (spreadsheets and databases) and will facilitate business intelligence operations on its stored data.

Precisely, within this thesis project, two problems are discussed. The former is the data warehouse design, based on HR analysis and the latter is regarding Extract, Transformation and Loading (ETL) solutions for spreadsheet-based sources. Both of the topics focus on HR assistance for Viggo. However, the ETL solutions focus on an academic research regarding querying spreadsheets. More specifically, regarding the ETL solutions, **we have designed an ETL framework, which uses a querying mechanism for Comma-Separated Values (CSV) spreadsheets. This ETL framework that we introduce in this thesis is called CSVQL (Comma-Separated Values Query Language) and is a potential query language**. Precisely, CSVQL is in a Structured Query Language (SQL) form and has been built to query information from CSV files, no matter file's structure. The proof of thesis' concept idea will validated via a set of tools. Precisely, the efficiency of CSVQL will be certified by a MySQL implementation and the capabilities of Viggo's data warehouse by icCube, an OLAP server, which operates over data warehouses.

1.4 Thesis outline

As it mentioned in Section 1.3, within this project, the work load is split into two major parts. The former focuses on Viggo's Data Warehouse design and the latter investigates ETL solutions for Viggo's data sources. Both parts are presented in the following chapters which have the structure of data warehouse modules:

Chapter 2. Preliminaries Background knowledge.

Chapter 3. Data warehouse overview Concept idea behind Viggo's Data Warehouse.

Chapter 4. Data sources analysis Viggo's data internal sources analysis.

Chapter 5. Back-end tier Introduction to CSVQL and its attributes.

Chapter 6. Data warehouse tier Design and implementation of Viggo data warehouse.

Chapter 7. OLAP & Front-end tier Introduction to icCube utilities and its reporting tools.

Chapter 8. Conclusions Discussion and suggestions.

Chapter 2

Preliminaries

This chapter is served as the recommended background for the project, by providing the prior knowledge for the material, which is presented in the following chapters. More specifically, in this chapter the basic concepts of data warehouses are introduced. Section 2.1 consists of an introduction Database Management Systems and in Section 2.2, the need of a data warehouse is elaborated and its building blocks are introduced. In Section 2.3, the architecture of data warehouse systems is described in detail. Precisely, the basic components of the architecture of a typical data warehouse are presented. For each of these components, there is an presentation of their characteristics.

2.1 Database management systems

Since the beginning of computer age, the most crucial and essential component regarding computer's performance has been the memory. More specifically, the manner in which the information is organized inside the memory, can conclude either the success or the disaster of any computation attempt. Nowadays, data management is a hot topic, since almost any application relies on data, which size usually increases exponentially according the time. From simple web sites to complex business analyst tools, different types of data should be processed, stored and retrieved, in a efficient and accurate way. As a result, the systems, which operate over data must be adaptive, responsive, efficient and accurate as well.

Any system, which operates on data management is called Database Management System (DBMS). This kind of systems is characterized as a high-level software, which cooperates with low-level interfaces, aiming on data storage and querying. More specifically, a DBMS assists the user to handle enormous collections of data, which are stored in the hard drive. Because of the fact that a data collection does not have standard shape or size, a lot of DBMS have been developed, during the years in order to assist in dealing with this variety in data format. Consequently, there are dozens of solutions, which have been developed, though the years and offer different approaches in data management, according to user's needs. However, only a relatively small set became popular and stay in use for a longer time. Precisely, the most popular kind of DBMS all this time is the Relational DBMS (RDBMS).

Relational databases are the primary data storage mechanism in recent times. As its name indicates, a relational database is based on the relational algebra schema, which organizes the data into tables with specific structures and attributes. Nowadays, there are a lot of versions of a RDBMS - commercial and open source products- such as Oracle Database, Microsoft SQL Server, MySQL, PostgreSQL and SQLite. These specific versions offer various functions and tools regarding data management and consists of the biggest piece of the pie chart of popular database systems.

2.2 Data warehouse concepts

As it mentioned in the introduction, the necessity and the priority of data analysis increases steadily, as every organization in business field tries to utilize its decision-making mechanisms, which will maintain or increase any advantage against its market competitors. However, the traditional DBMS cannot fulfill the obligations of a data analysis project. The reason is the fact that these database systems are built to support data management, in terms of fast and accurate information retrieval on daily basis. This transactional approach indicates that this kind of systems offers data consistency and security for everyday transactions and as a result they are described as On-Line Transaction Processing (OLTP) database systems [6].

As it is mentioned above, OLTP database systems focus on transactional operations. A transaction could be an insertion of a new order inside the database, an update of the status of a placed order or the deletion of this specific order. In addition to these transactional actions, an OLTP database system is characterized by strong indexing capabilities, in order to support fast and efficient data querying. Moreover, in order to support heavy transaction loads and to prevent update anomalies, OLTP systems are able to achieve a high level of normalization. However, these characteristics affect their performance in a negative manner, when complex queries and aggregation tasks should be applied to a huge amount of stored data. Furthermore, the up-to-date operational mode of OLTP database systems results data structures with detailed but not historical data. According to these needs, which are required for decision making processes, an "on-line analytical processing" (OLAP) database system is a oriented solution [6].

An OLAP database system builds its data structure model in order to support heavy queries, which probably require joins of multiple tables and data aggregation. These tasks usually demand the traversing all the records of the database and because of that, they are expensive for an OLTP database's schema, in terms of processing power and memory space. As a result, there should be a schema that utilize this expensive process. In contrast, with OLTP databases, OLAP database systems do not tend to spread their data into tables to fulfill normalization and indexing obligations, since the reconstruction of the information requires joins, which are the most expensive query. A database system, which fits into the OLAP profile is called "data warehouse" [6].

A data warehouse can be described as a particular database, which is designed and optimized in order to support OLAP queries. More specifically, they are large repositories, which store data from multiple sources and consist of the main core of any business intelligence application. The data model of a data warehouse is based on multidimensional model, which offers a data view from multiple perspectives, since the data are stored in an multidimensional space. The building block of this specific model is the cube (hyper-cube for four or more dimensions) [6].

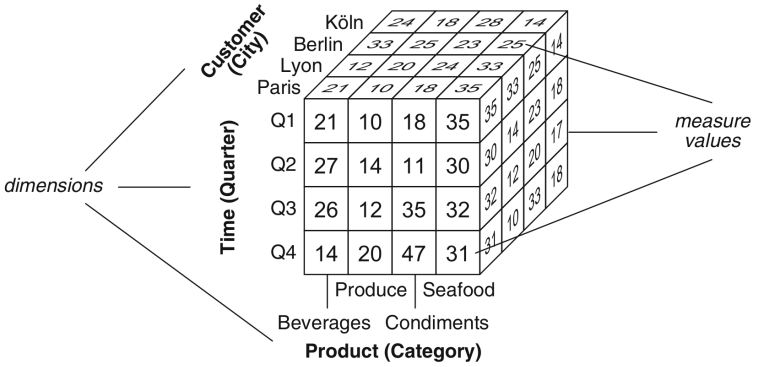


Figure 2.1: A three-dimensional cube regarding sales
[6]

As it is presented in Figure 2.1, a cube is a structure which presents the stored information via

multiple perspectives. More specifically, a cube is defined by its dimensions and its facts. A dimension of a data cube represents a perspective, in which the information of this data is described. In Figure 2.1, there are three dimensions: "Product", "Time" and "Customer". Moreover, each dimension can have hierarchy levels, which represent the level of detail, in which the information is described via this specific dimension. For example, the hierarchy levels of "Time" dimension can be: "Day", "Week", "Month", "Quarter" and "Year". On the other hand, a fact represents the measures, which are stored in a data cube. These measures can be numeric values or not. In Figure 2.1, the fact of the data cube, represents the measure "Quantity". A data cube can have more than one facts [6].

2.3 Data warehouse architecture

In the market, there are a lot of data warehouse systems, which differ among each other. However, all of them follow a general architecture, which is introduced by several tiers. These are the following:

Data sources represents the external sources, which supply the data warehouse with data.

Back-End tier represents the process of data insertion in the data warehouse.

Data warehouse tier represents the way in which the data are stored inside the data warehouses

OLAP tier represents the multidimensional model in which the stored data are analyzed.

Front-end tier represents the output of the data analysis inside OLAP tier.

The introduced tiers are represented by the Figure 2.2.

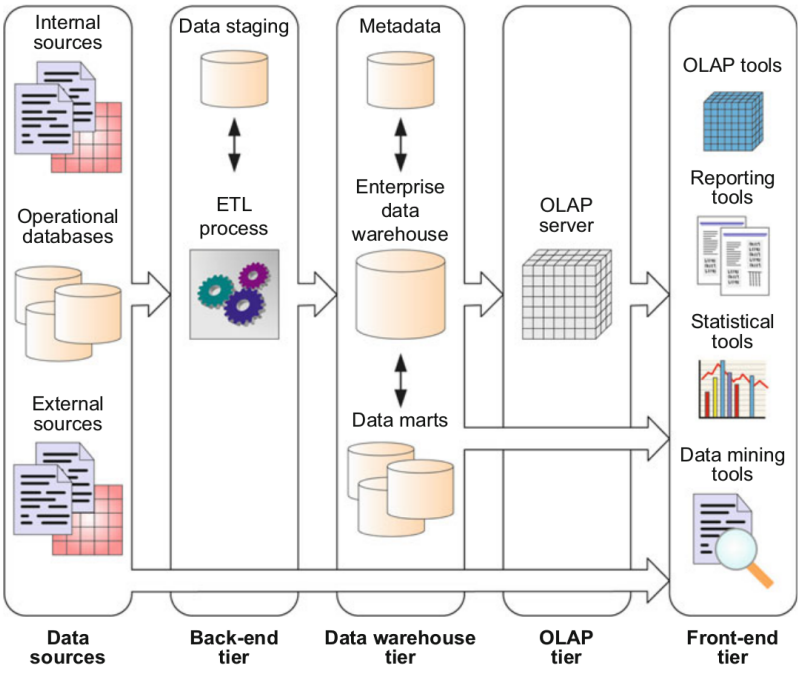


Figure 2.2: A data warehouse architecture [6]

2.3.1 Data sources

A data source of a data warehouse can be any supplier of information, which is intended to be stored inside this specific data warehouse. In Figure 2.2, the data sources are internal and external files, such as spreadsheets or Extensible Markup Language (XML) files, and databases.

2.3.2 Back-end tier

The back-end tier indicates the process, which retrieves the data from the external sources. This process is also well-known as Extraction, Transformation, Loading (ETL) process. As its name indicates, ETL is a three-step process and each of these step is described bellow:

Extraction During the first step, data from multiple and heterogeneous external sources, are obtained. These external data sources can be operational databases but also may be files with various formats, such as spreadsheets, XML files or even document files. These files can be internal and/or external to the organization.

Transformation During transformation step the obtained data are modified from the format that they had in their data source to a new one that fits to warehouse data model. The transformation process encloses sub-tasks such as:

- Cleaning, which fixes errors and inconsistencies in the data, which will be converted into a standardized format;
- Integration, which adjusts the data from the data sources, according to data warehouse schema;
- Aggregation, which performs calculations and create measurements regarding the obtained data

Loading During last step the data warehouse is filled with the transformed data. Consequently, this task requires also to refresh the data warehouse and to trigger updates regarding the stored data.

ETL process usually uses an extra database in order to provide an extra filter and back-up layer. This database is called data staging area or operational data store and stores the obtained data, until the transformation step finishes.

2.3.3 Data warehouse tier

The data warehouse tier is associated with the actual storing structures of the data warehouse. Figure 2.2 illustrates an enterprise data warehouse and several data marts. Data marts are small parts of the data warehouse, because usually a data warehouse stores data from not only a department, but from a whole organization. As a result, data marts aim only to a specific set of data (regarding only one department for example), while an enterprise data warehouse encompasses the entire organization. From a different perspective, a data mart can be considered as a smaller local data warehouse. An additional component to data warehouse tier is the metadata repository. A metadata repository usually contains the following information:

- Metadata regarding the structure of the data warehouse and/or the data marts, at conceptual, logical and physical level. Furthermore, these metadata may contain security and monitoring information.
- Metadata regarding the data sources and their schemas. In addition, these metadata can have descriptive additional information such as ownership, update frequencies and intervals, limitations, and access rights and methods.
- Metadata regarding ETL processes, including data origin, rules about extraction, cleaning and transformation and algorithms for aggregation.

2.3.4 OLAP tier

The OLAP tier handles the stored data and presents them via a multidimensional perspective. In Figure 2.2 the OLAP tier is represented by an OLAP server, which reconstructs the data from data warehouses and/or data markts into multidimensional data (cubes). In general, the concept idea of a data warehouse is based on the multidimensional model. As it is presented in Section 2.2, in a multidimensional model, the information is represented via cubes. These cubes contain measures and these measures have levels of abstraction, which are called dimensions and they are important to the business community. The measures inside the cube cells contain measures for business analysis, such as salaries, dates and any possible aggregations. Regarding the building of these cubes, OLAP components operate on Data warehouse tier and query the stored data in a manner, which outputs the aforementioned cubes [6]. As a result, OLAP tier consists a higher level than data warehouse tier, regarding the representation of the stored data. This means that with the use of this tier, the actual structure of the stored data and the specifications of the underlying DBMS are hidden.

2.3.5 Front-end tier

The front-end tier consists of the visualization end of a data warehouse and offers tools that can help the user to project the contents of this data warehouse. As it is presented in Figure 2.2, the typical client tools are the following:

OLAP tools OLAP tools achieve interactive navigation within the data warehouse OLAP tier, offering multiple views and levels of detail.

Reporting tools Reporting tools support the construction of paper-based or interactive reports, where the output of the OLAP queries is presented.

Statistical tools Statistical tools offer data cubes analysis and visualization, with the use of statistical theory.

Data mining tools Data mining tools identify patterns inside the data and support decisions by predicting trends.

2.4 Management of Comma-Separated Values data

Because of the fact that the majority of the data sources within Viggo are CSV files, a extended study should take place regarding the main concepts and characteristics of this type of data structures. The reason is that these files are potential data sources for a data warehouse and, as a result, any information retrieval process must take into consideration the manner in which the information is stored.

2.4.1 Definition

A Comma-Separated Values (CSV) file contains tabular data in a plain text form. This means that every line of a CSV file can be considered as a table line. Moreover, in case of relational data, a line of a CSV file can be considered as a data record. According to its definition, every record consists of one or multiple columns, which are separated by commas. The use of the comma, as a column separator, defines the name of this type of file format [22].

However, the CSV file format has not been standardized, yet. The concept idea of separating columns gets complicated, when the values within the columns are in form of a string and contain also commas or even embedded line-break characters. This kind of problems are treated with the use of quotation marks around the value of each column. Nevertheless, quotation does not solve all the aspects of this issue 100%, because it is possible that the value of a column may need embedded quotation marks. This concludes that despite of the fact that CSV idea is simple and

clear, CSV implementation has major issues regarding the "separators" between the values [22]. Figure 2.3 presents, a sample of some recorded values in a CSV format.

```
'timestamp','tag_id','x_pos','y_pos','heading','direction','energy','speed','total_distance'
    ...
'2013-11-03 18:30:00.000612',31278,34.2361,49.366,2.2578,1.94857,3672.22,1.60798,3719.61
'2013-11-03 18:30:00.004524',31890,45.386,49.8209,0.980335,1.26641,5614.29,2.80983,4190.53
'2013-11-03 18:30:00.013407',0918,74.5904,71.048,-0.961152,0,2.37406,0,0.285215
'2013-11-03 18:30:00.015759',109,60.2843,57.3384,2.19912,1.22228,4584.61,8.14452,4565.93
'2013-11-03 18:30:00.023466',909,45.0113,54.7307,2.23514,2.27993,4170.35,1.76589,4070.6
    ...
```

Figure 2.3: Samples obtained by a 20 Hz GPS sensor traces. Each line of this file is a time-stamped record. The first line is the header line.

On February 2016, World Wide Web Consortium (W3C) published a note of 43 pages, regarding CSV standards, since the use of this format is very popular on the web. More specifically, the "CSV on the Web Working Group" developed standard mechanisms to express metadata regarding CSV files and similar tabular data [22].

2.4.2 Data querying

Recalling ETL process, the need of information retrieval from data sources is the most interesting field of research within the Back-end tier of a data warehouse. Precisely, when these data sources are in CSV format, then in addition with the rest of any ETL issues, the CSV data structure requires also the attention of the data warehouse engineer. During the years, a lot of research has been made regarding CSV format, but because to the fact that this format is not a standard yet, every work has been developed based on different assumptions, regarding the data schema and the metadata which a CSV file stores. Since the assumptions among the scientific works do not meet the same standards, within this work, the related work is taken into consideration only as a motivation for research.

In the paper "A framework for annotating CSV-like data", Marcelo Arenas et al. conclude that CSV grids cannot be queried by SQL mechanisms. The reason is that CSV grids usually are not in the form of relational tables. Since, the annotation systems of these mechanisms are based on a row-based model, such as the relational schema, they cannot support their functionality for CSV grids, where the data can be stored in any possible manner [18]. Thus, in terms of SQL querying, the information is not enough to be presented via tabular data, but it must follow the relational model.

The above conclusion is supported by the work of Juliana Freire et al. with title "The Exception that Improves the Rule". Within this work, the paradox of spreadsheets overuse is investigated. In contrast with the great development of database community, in terms of solutions and tools, spreadsheets usage is still high. More specifically, a proposal has been made that indicates the creation of a hybrid spreadsheet/relational system, which can use the advantages of both of these means of data storage [17]. As a result, an alternative to this "expensive" proposal, could be the creation of a mechanism, which can transform the "unstructured" CSV grids into relational tables. In this way, the spreadsheet user could still interact with the familiar to them CSV files and at the same time the querying process would be still feasible.

However, the aforementioned process is not trivial and it cannot be fully automated. In the paper "Integrating Spreadsheet Data via Accurate and Low-Effort Extraction", Zhe Chen and Michael Cafarella, advocate that this process can be implemented as a two-phase semiautomatic mechanism, which outputs relational data. The disadvantage of this approach is the required metadata, which must be included in every CSV file. Moreover, the data in the CSV files must be in a semi-relational form, in a sense that they must be tabular but with multiple headers [23]. Therefore, there is no "one fit to all" solution, since each CSV set of data should contain a small set of metadata, regarding its schema. Because of the fact that this small piece of information is not always available, then it is difficult or even impossible to create an adaptive to every possible CSV schema system, which can retrieve data automatically or semiautomatically.

On the other hand, an adaptable system such as an SQL-like query language could meet some of the discussed demands. In papers "A Spreadsheet Algebra for a Direct Data Manipulation Query Interface" of Bin Liu et al. and "Expressive Query Construction through Direct Manipulation of Nested Relational Results" of Eirik Bakke and David R. Karger, the possibility of an SQL-like expressiveness framework is introduced. More specifically, a query system could allow the user to express his query preferences and then these preferences would be translated in terms of CSV parameters [9] [13]. In this manner, the user would be able to query the CSV files as SQL tables, with the only difference that she should recognize the pattern, in which the data are stored on his own.

Chapter 3

Data warehouse overview

This chapter illustrates the concept idea, which consists the solution to Viggo's problem. More specifically, in Section 3.1, the solution according Viggo's problem statement is introduced and its benefits are elaborated. This specific solution advocates the implementation of a data warehouse, which stores all the information related to Viggo's needs, will be the centralized repository of the whole organization. In Section 3.2, the advantages of this choice are elaborated. Precisely, the benefits are split into two major parts. The former includes the advantages of a data warehouse use according to spreadsheet overuse and the latter introduces the advantages of using a data warehouse as a centralized repository. In Section 3.3 the architecture of Viggo data-warehouse is presented and the modules of this specific data warehouse are described.

3.1 Solution

According to the problem statement, within Viggo, HR and BA departments usually struggle to take decisions regarding the development of company's manpower. The main reasons are the problems that spreadsheets introduce and the lack of a centralized repository, which can facilitate business intelligence operations on the stored data. The solution which treats this problem in an efficient manner is the design and the implementation of a data warehouse. This data warehouse will be the centralized data repository, which quickly integrate data from Viggo's internal data sources, such as spreadsheets and assisting database systems. More specifically, this specific product will designed in order to fulfill demands regarding:

- decision-support
- analytical-reporting
- ad-hoc queries
- data mining

3.2 Benefits of data warehouses

The importance of the data warehouses in business community is supported by Forbes. More specifically, during 2010, a Forbes Insights study concluded that "data-related problems cost the majority of companies more than 5 million annually, and for the 20 % of the companies costs more than 20 million annually" [19]. Thus, a proper data warehouse can cause a serious positive impact on the expenses of a company or an organization.

Precisely, data warehouses not only save time, resources and effort by stockpiling data in only one centralized storage, but they also guarantee the quality of the provided information. More specifically, the provided information from data warehouse is more accurate and reliable than from

spread and usually unprotected data sources, such as spreadsheets. Consequently, data warehouses can help Viggo to minimize or even to eliminate threats, caused by spreadsheet overuse.

3.2.1 Data warehouses as spreadsheets overuse cure

Regarding the issues that Viggo faces because of its spreadsheet overuse, a data warehouse treats the following threats:

"Difficult to keep up to date" threat By definition, a database is a centralized data-store [6]. As a result, the information is not spread along multiple sources and, consequently, there is no need for multiple updates when there is a data update. Moreover, the update process can be performed automatically, by update triggers and overnight tables maintenance.

"Vulnerable to fraud" threat A data warehouse can provide an environment, in which the users can interact with the data with safety. This safety concerns the use and the manipulation of the stored information. More specifically, to a database, a broad set of controls and layers, which can prevent fraud, can be applied, including [16]:

- Access control
- Auditing
- Authentication
- Encryption
- Integrity controls
- Backups
- Application security
- Database Security applying Statistical Method

"Prone to human errors" threat A basic characteristic of a relational data store is the restriction regarding the type of its data. More specifically, each column within a database table is required to have a specific name and a specific data type. As a result, a data type can be considered, practically, as a label or as a guideline to the user, in order to understand what type of data is expected inside of each column. This mechanism will secure the data-set against human errors [16].

"Unfit for agile business practices" threat Relational data stores are built with a predefined schema. More specifically, insides a relational data store the information is organized in a specific form of tables, which have specific columns with specific data-types. As a result, the users of this specific data store, always know the structure of the stored data and they can utilize their applications according to this specific schema [15]. On the other hand, when the data are stored into individual files, then the structure is not predefined and as a result, no assumptions can be made regarding the manner in which the data are stored. Consequently, the operation on these data cannot be utilized.

"Not designed for collaborative work" threat A RDBMS is a centralized data store, which have one or multiple users. These users can have concrete rights regarding database usage and operate on the data at the same time. In addition, there is a log file, which stores all the actions that took place. In addition, there are back-up option when there is a need of rolling-back actions, which caused data loss or other inconsistencies [3].

"Scale poorly" threat According to the nature of the data and the hardware, a relational data stores can scale millions of records. On the other hand, a spreadsheet can scale some hundreds of records with huge latency and poor performance [21].

"Hard to consolidate" threat The centralized character of an RDBMS erases this threat, since all the data are stored inside one source and they are not spread in multiple sources. By definition, a database is a centralized data-store [6]. As a result, the information is not spread along multiple sources and, consequently, there is no need of sources combination in order to consolidate data.

3.2.2 Data warehouses as data repositories

Instead of evaluate a data warehouse, as a medicine for spreadsheet overuse, it would be more beneficial to investigate a data warehouse as a DBMS. The reason is the fact that a comparison between a data warehouse and a spreadsheet can easily suggest the data warehouse as the best option, but, on the other hand, it hides data warehouse's real potential, as a business analysis tool. In this section, an attempt is made to investigate the benefits of data warehouses.

Fundamentally, a data warehouse is a kind of data store, which stores information for business analytics. Business analytics require calculations and aggregations on data, which are important for the organization that interacts with them. However, this kind of operations are expensive, in terms of time and storage, when they are implemented as ad-hoc queries. Data warehouses are used in order to solve these expenses. More specifically, a data warehouse is able to interact with transactional systems, extract their data and convert them into usable information for business analysis, in quick and efficient manner. In addition, a data warehouse, according to its structure, can process and execute complex and demanding queries with a highly-efficient mode. Thus, a data warehouse can offer to business community various improvements and practical gains.

From early 1990s, the need of data warehouses has been identified, More specifically, the explosion of Internet and of digital marketing created enormous amounts of data that were growing exponentially and the current database technology couldn't manage successfully. Trying to solve the arisen problem, during the 1990s, Bill Inmon published a book with title "Building the Data Warehouse". This book is a introduction a modern concept of storing data. According to the author, "data warehouses provided a much-needed strategy for organizations to collect, store, and analyze vast amounts of data. As businesses expand both brick-and-mortar and online activities, the field of data warehousing has become increasingly important". Trying to simplify, these words, a data warehouse can provide a structure in which the information is provided, according to business needs. Consequently, a business analyst should spend less effort and time to extract patterns and results, regarding customers or products and, as a result, to define business decisions. In general, the main reasons, that a data warehouse can be beneficial for a company, are the following:

First, data warehouses help employees not to waste valuable effort and time, by allowing them to store all the information at the same place. In contrast, an employee, who keeps his data in multiple locations, need to spend a serious amount of time to retrieve and maintain the actual information and its structure. Thus, data warehouses can assist business analysts to conclude and improve strategic decisions, without needing to combine different data sources. In addition, data warehouses can reduce the expenses of a company, since the business analysts of the company can retrieve and inspect data, without expensive actions or extensive assistance from company's Information and Communications Technology (ICT) department.

Second, the format of the stored information within a data warehouse can be a standard for the company, which uses this specific data warehouse. More specifically, companies can benefit themselves from storing their data, according to a specific format. The reason is the fact that if all the departments, within an organization, standardize their data in the same format, then each data set can be synchronized with the rest, without any modification, no matter the origin department. This data quality and uniformity can lead business analysts to feel more confident, regarding their data, in terms of accuracy, and, as a result, to produce more informed business decisions.

Third, business intelligence can be affected in a positive manner with the use of data warehouses. Because of the fact that a data warehouse combines information from multiple sources and sectors of the company, business analysts can feel more confident, about their decisions, since they have been made based on a broad set of information. More specifically, a data warehouses can determine a full picture of a company, in terms of marketing factors, financial logs, inventory and sales history. This broad view allows business analysts to make their decisions, based not only on one aspect, but on various conditions.

3.3 Viggo data warehouse

The following data warehouse is the solution to Viggo's problem statement. This data warehouse is a centralized data repository, which integrates data from Viggo's internal data sources via an ETL framework, which uses CSVQL query language. The data, which are extracted with the use via the ETL process, are stored into a MySQL database and on top of this specific database, an icCube OLAP server operates, by creating multidimensional views of these stored data. One level above, these specific multidimensional views are presented via the icCube web reporting tool. More specifically, this web reporting tool provides graphs and charts, which consist a diagrammatical illustration of the stored data and are required for HR and BA analytics. The architecture of the introduced data warehouse and its modules are presented by the following sub-sections.

3.3.1 Architecture

The architecture of Viggo data-warehouse is presented by the Figure 3.1. The modules of this data-warehouse are described by the following subsections.

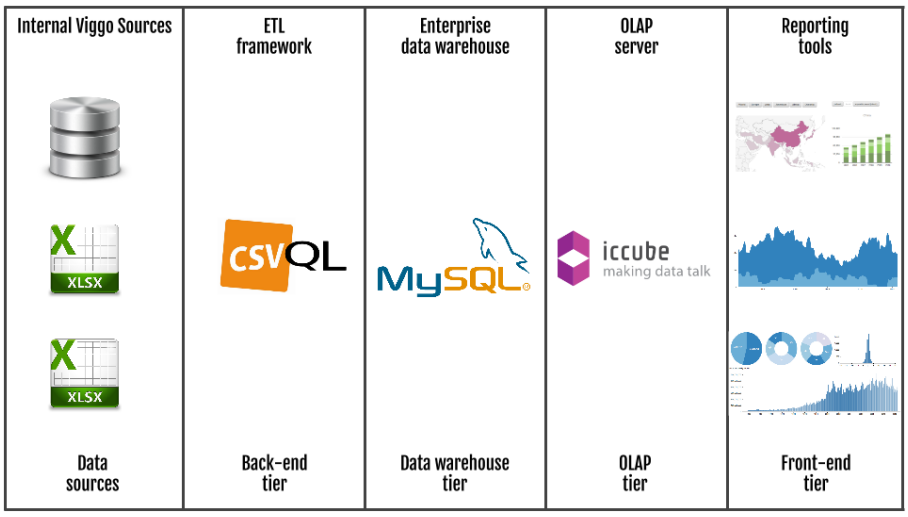


Figure 3.1: Viggo data warehouse architecture.

3.3.2 Data sources

The data sources of Viggo data-warehouse are spreadsheets, XML files and relational databases. However, within this work the data sources are only spreadsheets and relational databases. Because of the fact that some of these databases are inside DBMS's, which are used via product license, there is no option for direct data retrieval. The reason is that the providers of these systems, offer to Viggo only the option to extract data in spreadsheets. As a result, the data sources in practice can be only spreadsheets, of which some are in relational form already and some others are not.

3.3.3 Back-end tier

The back-end tier refers to the ETL process of the data warehouse and is implemented by a newborn framework. This tier receives the most of the attention, during this work and is the main focus of research. More specifically, this specific framework uses a querying mechanism for CSV spreadsheets, which is called CSVQL (Comma-Separated Values Query Language). CSVQL is a potential query language, which is in a SQL form and is built to query information from CSV files, no matter if the file is in a relational form or not. CVSQL in addition with the import/ bulk functions of the SQL DBMS implements the ETL process and consequently the back-end tier of Viggo data warehouse.

3.3.4 Data warehouse tier

The data warehouse tier is implemented by a SQL DBMS. This specific database system will store in a relational structure, all the information, which is retrieved by the Viggo's internal data sources via the ETL process. More specifically, a MySQL 5.7.13 server, which runs on an Ubuntu 16.04.2 operating system, is used. MySQL is an open-source RDBMS, which has been released for the first time in 1995, and at this moment, it is the second most popular database engine after Oracle, according to DB-Engines Ranking [1].

3.3.5 OLAP tier

The OLAP server, which operates over the Data warehouse tier, is an icCube Server. This product is an in-memory OLAP server, which supports its users with business intelligence and web reporting tools. Within this project, the icCube Suite 5.2, which is released in May 2016, is used [2]. More specifically, the OLAP tier is implemented via MultiDimensional eXpressions (MDX) queries, which build the OLAP cubes for business analysis. As a result, all the OLAP cubes, which are required, will be built via MDX queries to MySQL DBMS through icCube platform.

3.3.6 Front-end tier

The front-end tier is implemented by icCube Web reporting tools. More specifically, icCube platform offers a variety of web reporting tools and dashboards, which are presented via a unique Web interface [2]. Precisely, at this layer all the graphs and dashboards, which will unravel the hidden information from data sources, are implemented. As a result, this layer is the actual solution to Viggo's problem, since it gives useful insights for Viggo's manpower and implements the requirements of BA and HR departments. Moreover, the OLAP and the Front-end tiers consist the proof of the solution, which is proposed in Section 3.1.

Chapter 4

Data sources analysis

This chapter focuses on the first tier of Viggo data warehouse; the data sources. Viggo's internal data sources provide Viggo data warehouse with data, are introduced. Viggo has a big variety of spreadsheets, but within this work three types of spreadsheets will be used; the Shift Rosters, the Training Managers and the Youforce exported spreadsheets. Each section represents one type of spreadsheets and analyzes its type, in terms of functionality and data schema. Section 4.1 describes Viggo Shift Rosters, Section 4.2 describes Viggo Training Managers and Section 4.3 describes the exported spreadsheets from Youforce database.

4.1 Viggo Shift Rosters

Viggo Shift Rosters are spreadsheets, which contain a list of employees, and associated information regarding their tasks (e.g. working times) for a given time period (e.g. week). These specific spreadsheets are created by the operation manager on a weekly basis, with the use of Microsoft Excel. Since 2010, within Viggo there is one Shift Roster per department with duration one semester (6 months). Each of these spreadsheets, contains 3 - 4 tabs, more than 200 rows and around 550 columns, regarding employee's shift scheduling.

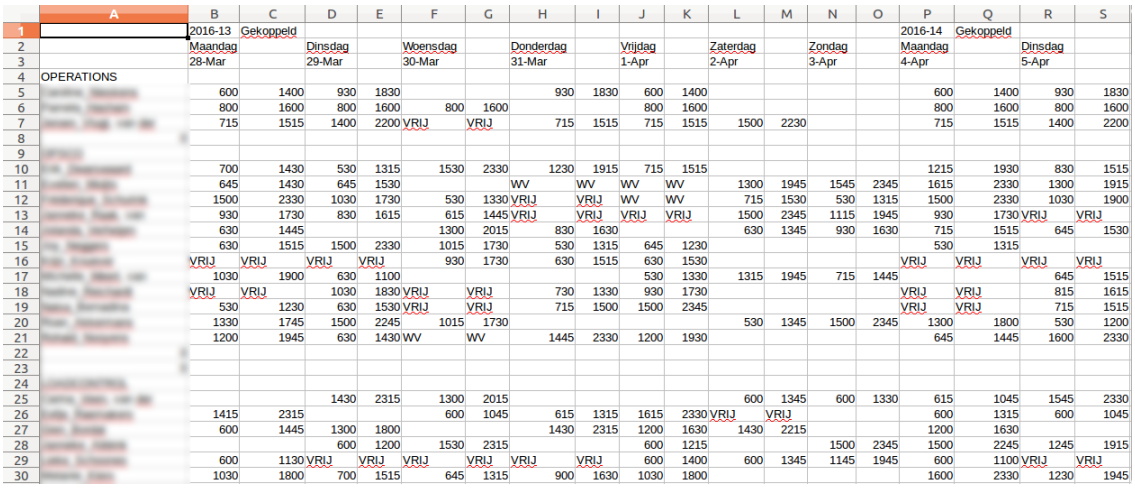
4.1.1 Functionality

Shift Rosters are necessary for the day-to-day operations. More specifically, they contain the scheduling of the employees in Eindhoven Airport during a certain period of time. Thus, every employee can access the rooster, in order to get informed about her shifts, during the period of time that this specific rooster indicates.

4.1.2 Schema

The schema of a Shift Roster is not relational, in a sense that it does not follow the relational model. As it can be seen in Figure 4.1, the schema of a rooster is not organized as a set of records, with concrete attributes and with a unique key identifying each record. In contrast, a Shift Roster is a time series schedule, in the following form.

The schedule is represented by columns, which are ordered chronologically. Each employee is represented by rows, which are ordered alphabetically. Employees are listed on the left hand side of a grid, with the days of the week on the top of the grid. Each day has two data points per employee; the starting time and ending time of her shift in HH:MM format. When an employee does not work a day, both of the data points of this specific day for this specific employee have the value "VRIJ".



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		2016-13	Gekoppeld													2016-14	Gekoppeld		
2		Maandag		Dinsdag		Woensdag		Donderdag		Vrijdag		Zaterdag		Zondag		Maandag		Dinsdag	
3		28-Mar		29-Mar		30-Mar		31-Mar		1-Apr		2-Apr		3-Apr		4-Apr		5-Apr	
4	OPERATIONS																		
5	[redacted]	600	1400	930	1830			930	1830	600	1400					600	1400	930	1830
6	[redacted]	800	1600	800	1600	800	1600			800	1600					800	1600	800	1600
7	[redacted]	715	1515	1400	2200	VRIJ	VRIJ	715	1515	715	1515	1500	2230			715	1515	1400	2200
8																			
9	[redacted]																		
10	[redacted]	700	1430	530	1315	1530	2330	1230	1915	715	1515					1215	1930	830	1515
11	[redacted]	645	1430	645	1530			WV	WV	WV	WV	1300	1945	1545	2345	1615	2330	1300	1915
12	[redacted]	1500	2330	1030	1730	530	1330	VRIJ	VRIJ	WV	WV	715	1530	530	1315	1500	2330	1030	1900
13	[redacted]	930	1730	830	1615	615	1445	VRIJ	VRIJ	VRIJ	VRIJ	1500	2345	1115	1945	930	1730	VRIJ	VRIJ
14	[redacted]	630	1445			1300	2015	830	1630			630	1345	930	1630	715	1515	645	1530
15	[redacted]	630	1515	1500	2330	1015	1730	530	1315	645	1230					530	1315		
16	[redacted]	VRIJ	VRIJ	VRIJ	VRIJ	930	1730	630	1515	630	1530					VRIJ	VRIJ	VRIJ	VRIJ
17	[redacted]	1030	1900	630	1100					530	1330	1315	1945	715	1445			645	1515
18	[redacted]	VRIJ	VRIJ	1030	1830	VRIJ	VRIJ	730	1330	930	1730					VRIJ	VRIJ	815	1615
19	[redacted]	530	1230	630	1530	VRIJ	VRIJ	715	1500	1500	2345					VRIJ	VRIJ	715	1515
20	[redacted]	1330	1745	1500	2245	1015	1730					530	1345	1500	2345	1300	1800	530	1200
21	[redacted]	1200	1945	630	1430	WV	WV	1445	2330	1200	1930					645	1445	1600	2330
22																			
23																			
24	[redacted]																		
25	[redacted]			1430	2315	1300	2015					600	1345	600	1330	615	1045	1545	2330
26	[redacted]	1415	2315			600	1045	615	1315	1615	2330	VRIJ	VRIJ			600	1315	600	1045
27	[redacted]	600	1445	1300	1800			1430	2315	1200	1630	1430	2215			1200	1630		
28	[redacted]			600	1200	1530	2315			600	1215			1500	2345	1500	2245	1245	1915
29	[redacted]	600	1130	VRIJ	VRIJ	VRIJ	VRIJ	VRIJ	VRIJ	600	1400	600	1345	1145	1945	600	1100	VRIJ	VRIJ
30	[redacted]	1030	1800	700	1515	645	1315	900	1630	1030	1800					1600	2330	1230	1945

Figure 4.1: A sample of Viggo Shift Rosters' schema.

4.2 Viggo Training Managers

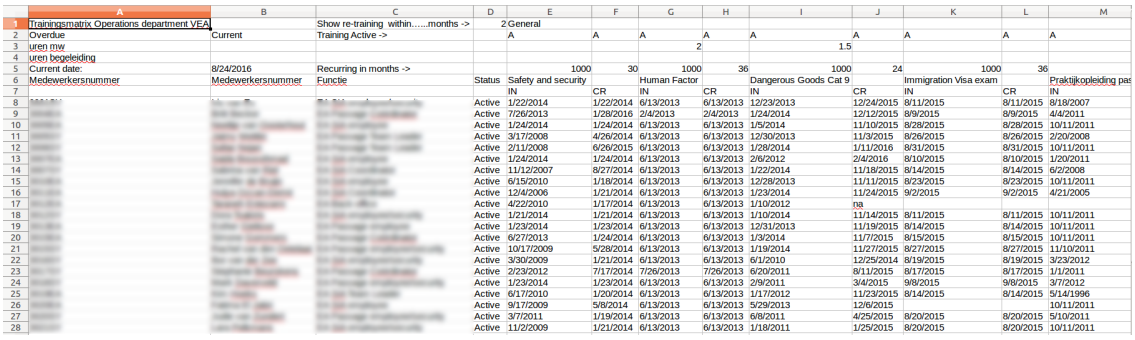
Training Managers are spreadsheets, which contain a list of employees, and associated information regarding their skills (e.g. expired data of a trained skill such as "Towbar Pushback"). These specific spreadsheets are created and updated by the trainers within Viggo Academy on a monthly basis, with the use of Microsoft Excel. Since 2010, within Viggo there are around 3 Training Managers, per department. Each of these spreadsheets, contains 6 to 7 tabs, more than 200 rows and more than 50 columns, regarding employee's training.

4.2.1 Functionality

Training Managers are necessary for the classification of the employees into function categories (e.g "EA Platform Employee 1", "EA Platform Employee 2") and for task assignments. More specifically, each employee can perform only the task for which she has the required skills. As a result, Training Managers are the official logs of employees' capabilities. Thus, every operation manager should access the Training managers, in order to get informed about her shifts, during the period of time that this specific rooster indicates.

4.2.2 Schema

As in Roosters case, the schema of a Training Manager is not relational, in a sense that it is not in a relational table. The schema of a Training Manager can seen by Figure 4.2.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Trainingsmatrix Operations department VEA		Show re-training within.....months ->	2	General								
2	Overdue	Current	Training Active ->		A	A	A	A	A	A	A	A	A
3	uren mw					2			1.5				
4	uren begeleiding												
5	Current date:	8/24/2016	Recurring in months ->		1000	30	1000	36	1000	24	1000	36	
6	Medewerkersnummer	Medewerkersnummer	Functie	Status	Safety and security		Human Factor		Dangerous Goods Cat 9		Immigration Visa exam		Praktijkopleiding pas
7	[redacted]	[redacted]	[redacted]		IN	CR	IN	CR	IN	CR	IN	CR	IN
8	[redacted]	[redacted]	[redacted]	Active	1/22/2014	1/22/2014	6/13/2013	6/13/2013	12/23/2013	12/24/2015	8/11/2015	8/11/2015	8/18/2007
9	[redacted]			Active	7/26/2013	1/28/2016	2/4/2013	2/4/2013	1/24/2014	12/12/2015	8/9/2015	8/9/2015	4/4/2011
10	[redacted]			Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	1/5/2014	11/10/2015	8/28/2015	8/28/2015	10/11/2011
11	[redacted]			Active	3/17/2008	4/26/2014	6/13/2013	6/13/2013	12/30/2013	11/3/2015	8/26/2015	8/26/2015	2/20/2008
12	[redacted]			Active	2/11/2008	6/26/2015	6/13/2013	6/13/2013	1/28/2014	1/11/2016	8/31/2015	8/31/2015	10/11/2011
13	[redacted]			Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	2/6/2012	2/4/2016	8/10/2015	8/10/2015	1/20/2011
14	[redacted]			Active	11/12/2007	8/27/2014	6/13/2013	6/13/2013	1/22/2014	11/18/2015	8/14/2015	8/14/2015	6/2/2008
15	[redacted]			Active	6/15/2010	1/18/2014	6/13/2013	6/13/2013	12/28/2013	11/11/2015	8/23/2015	8/23/2015	10/11/2011
16	[redacted]			Active	12/4/2006	1/21/2014	6/13/2013	6/13/2013	1/23/2014	11/24/2015	9/2/2015	9/2/2015	4/21/2005
17	[redacted]			Active	4/22/2010	1/17/2014	6/13/2013	6/13/2013	1/10/2012	na			
18	[redacted]			Active	1/21/2014	1/21/2014	6/13/2013	6/13/2013	1/10/2014	11/14/2015	8/11/2015	8/11/2015	10/11/2011
19	[redacted]			Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	12/31/2013	11/19/2015	8/14/2015	8/14/2015	10/11/2011
20	[redacted]			Active	6/27/2013	1/24/2014	6/13/2013	6/13/2013	1/3/2014	11/7/2015	8/15/2015	8/15/2015	10/11/2011
21	[redacted]			Active	10/17/2009	5/28/2014	6/13/2013	6/13/2013	1/19/2014	11/27/2015	8/27/2015	8/27/2015	11/10/2011
22	[redacted]			Active	3/30/2009	1/21/2014	6/13/2013	6/13/2013	6/1/2010	12/25/2014	8/19/2015	8/19/2015	3/23/2012
23	[redacted]			Active	2/23/2012	7/17/2014	7/26/2013	7/26/2013	6/20/2011	8/11/2015	8/17/2015	8/17/2015	1/1/2011
24	[redacted]			Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	2/9/2011	3/4/2015	9/8/2015	9/8/2015	3/7/2012
25	[redacted]			Active	6/17/2010	1/20/2014	6/13/2013	6/13/2013	1/17/2012	11/23/2015	8/14/2015	8/14/2015	5/14/1996
26	[redacted]			Active	9/17/2009	5/8/2014	6/13/2013	6/13/2013	5/29/2013	12/6/2015			10/11/2011
27	[redacted]			Active	3/7/2011	1/19/2014	6/13/2013	6/13/2013	6/8/2011	4/25/2015	8/20/2015	8/20/2015	5/10/2011
28	[redacted]			Active	11/2/2009	1/21/2014	6/13/2013	6/13/2013	1/18/2011	1/25/2015	8/20/2015	8/20/2015	10/11/2011

Figure 4.2: A sample of Viggo Training Managers' schema.

As it can be seen in Figure 4.2 the expire dates of the skills are represented by a grid. The header rows of the grid contain the skills by name and the rest of the rows represent the employees. The names of the employees are listed on the left side of a grid, with the skills on the top of the grid. For each skill, every employee has two data points; the starting date and the ending date of the period that the skill is active for this specific employee. In case that an employee is not trained for a skill, the data points of this specific skills are empty.

4.3 Youforce database

Youforce is a Database Management System (DBMS), which is used by Viggo for HR purposes and is provided by Raet B.V. Raet's manpower is more than 1000 employees and it provides services in business industry, for more than 50 years. Raet's most popular product is Youforce. Youforce DBMS stores around 1.7 million employees worldwide and provides insights over HR affairs [4]. Within, Viggo, Youforce DBMS is used for payroll and contract administration. Since 2010, within Viggo there are Shift Rosters. Each of these spreadsheets, contains tabs, rows and columns, regarding employee's training.

4.3.1 Functionality

As the nature of the product indicates, Viggo uses Youforce for payroll purposes. More specifically, by using payroll, Viggo "will always pay the right amounts to employees, pension funds and other institutions, since payroll let its users easily calculate and change salary data on line. Any consequences of a change on the net payment and deductions will be automatically calculated according to the applicable laws and regulations and collective labor agreement" [4].

4.3.2 Schema

As a data source, Youforce is only available in spreadsheet mode. Raet does not provide any API for information retrieval and, as a result, the only way to import data from Youforce is to extract them in spreadsheet mode and re-import them again into Viggo data warehouse. The spreadsheets which are produced have relational structure, with the following columns:

Medewerkercode : The working Code of the employee
e.g. 0001VC

Naam : The name of the employee
e.g. John Smith

Datum uitdienst : The last date that the employee worked for Viggo (in case she is fired)
e.g. 31/12/2015

Leeftijd : The age of the employee
e.g. 18

Functie : The function of the employee within Viggo
e.g. VC Cleaning Supervisor

Afdeling : The department in which the employee belongs
e.g. Cleaning Viggo Eindhoven Airport

Salaris ft schaal : The salary of the employee according to her scale (Dutch financial system)
e.g. 1101.01

Salaris full time : The full-time salary of the employee
e.g. 1201.01

Salaris periodiciteit : The periodicity of the salary
e.g. per maand

Salaris uurloon : The salary of the employee per hour
e.g. 11.01

Schaalbedrag uurloon : The salary of the employee according to her scale (Dutch financial system) per hour
e.g. 11.01

Salaristrede : The salary overview
e.g. 48

Salarisschaal : The salary scale in which the employee belongs (Dutch financial system)
e.g. 03

Student : Indication if the employee is a student
e.g. Ja

Afloopdatum : The termination date of employee's contract
e.g. 31/12/2015

Arbeidsovereenkomst soort : The type of employee's contract
e.g. Stagiair(e)

Contractomschrijving : The description of employee's contract
e.g. Basiscontract

Datum indienst : The date when the employeed started working for Viggo
e.g. 01/01/2000

Soort contract : The type of employee's payment
e.g. Bepaalde tijd

The overview of the above description can be presented by Figure 4.3.

	A	B	C	D	E	F	G	H
1	Ad-hoc rapportage							
2	Bedrijf	Viggo Security B.V.						
3	Peildatum	23-05-2016						
4	Rapportdefinitie	Konstantimos						
5								
6	Medewerkercode	Naam	Datum uitdienst	Leeftijd	Functie	Afdeling	Salaris ft schaal	Salaris full time
7	[illegible]	[illegible]		[illegible]	[illegible]	Operations Viggo Security	[illegible]	
8	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
9	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
10	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
11	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
12	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
13	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
14	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
15	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
16	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
17	[illegible]	[illegible]	1/11/2016	[illegible]	[illegible]	Passage Viggo Security	[illegible]	
18	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security	[illegible]	
19	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
20	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
21	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
22	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
23	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
24	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
25	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		
26	[illegible]	[illegible]		[illegible]	[illegible]	Passage Viggo Security		

Figure 4.3: A sample of Youforce export spreadsheets' schema.

Chapter 5

Back-end tier

This chapter describes the Back-end tier of Viggo data warehouse. More specifically, Section 5.1, highlights the hottest topic, regarding ETL process. During this section, the phases of ETL process are described with emphasis on the constraints and the requirements. In addition, there is an overview of the ETL process in Viggo. In Section 5.2, the CSVQL is introduced. The initial clauses of this new born framework are presented in a form of a tutorial and a real life example of CSVQL is presented with the use of Viggo's data sources. Section 5.3 describes the process of the implementation CSVQL framework and its concepts into an actual query system. In Section 5.4, an experimental study takes place. This study validates the performance of the implemented system, according to virtual data sources.

5.1 ETL process

As it mentioned in Chapter 2, data warehouse operational processes consist of a specific work flow, where tasks, such as the extraction, the cleaning, the transformation, and the loading of data takes place, in order to populate the information inside the warehouse. To deal with this set of tasks the specialized set of actions should take place, under the general title "ETL" (Extraction Transformation Loading) process. More specifically, ETL process is consisted of inner tasks responsible for the Extraction of data from one or several sources, the Transformation of the extracted data, in order to fit business needs and database schema, and the Loading of these specific into the data warehouse.

During all the phases of the ETL process, particular issues can arise and alter data warehouse refreshment into a troublesome task. In this work, there is an attempt to clarify the complexity and the common issues of ETL processes. Thus, the following sub-sections are brief presentation of several issues and constraints, which use to arise in each phase of ETL process [7].

5.1.1 Global problems and constraints

Oracle states that the 90% of all the problems, inside a data warehouse, are caused by the nightly batch cycles [8]. During this period, the administrators of the data warehouse should deal with various difficulties such as:

(a) efficient data loading

(b) simultaneous job mixture and/or dependencies

In addition, ETL process in industry use to have global time constraints, according to the initiation time and to market deadlines. Practically, in the most the industries, there is a narrow and inflexible "time window", usually during the night, when the data warehouse can be maintained and be updated. The main reason is the fact that the most of the companies do not use their

systems or, at least the systems are in their pick in terms of usage. Consequently, the first and the major problem is regarding the scheduling of the whole process.

Regarding scheduling, there is a need of an execution plan, which utilize the ETL process by taking into account the available data sources, the DBMS's limitation in terms of performance and the available time space. In case that a data warehouse is used for business analytics and strategic purposes and the "time window" for updating is not so tight, then the ETL plan can be executed easier, since long-term reporting/planning cannot be affected by small delays [7].

5.1.2 Extraction & transportation

As the acronym ETL indicates, one of the very first tasks is the Extraction of the information, which is related to the data warehouse. This specific information, which is stored in the data sources, should be imported into the data warehouse, in order to have always the most recent and up to date information for analysis. Consequently, on daily, weekly or monthly basis, any new piece of information has to be obtained and be added to with the rest of the facts. However, this task requires the detection of the data that have not been imported yet, in order to minimize the overall processing time and to avoid duplicates. The most common technique to achieve that is a physically detection, which is performed by comparing two snapshots of the extraction task output. A snapshot regarding the previous extraction and another one regarding the current one. Moreover, there are efficient algorithms which can utilize this task, such as snapshot differential algorithms. Another popular technique focuses on log "sniffing". More specifically, there are log files, which can reconstruct the changes that took place, since the last extraction. Rarely, there are triggers, which are activated when a change detection occurs. However, this solution is difficult to be implemented because of the fact that usually the data sources can be either legacy systems either flat files in the most of the cases. On the other hand, when the data sources are relational systems, the use of triggers can slow down the performance of the whole process or even after the structure of the data warehouse. Another crucial issue involves the transportation of the extracted information. More specifically, security and speed are aspects that matter and, thus, FTP, encryption/decryption and compression/decompression can possibly concern [7].

5.1.3 Transformation & cleaning

Regarding Transformation, two kind of problems can be identified:

1. Mismatch between data sources and data warehouse schema
2. Data format

More specifically, according to schema level there can be naming conflicts. This means that the same header is used for different kind of entities (homonyms) or that each schema names the same entity differently (synonyms). In addition, on schema level there can be structural conflicts too, where there is different different representations of the same entity across multiple sources.

According to data format, there can be a lot of conflicts across the data sources. Some of this kind of conflicts are the following:

1. contradicting records,
2. different value representations (e.g. for marital status),
3. different interpretation of the values (e.g. measurement units Dollars against Euros),
4. different aggregation levels (e.g. sales per product against sales per product group),
5. reference to different points in time (e.g. current sales as of yesterday for a certain source against as of last week for another source)

This list can be enriched by low-level technical problems, such as of conflicts are the following:

- data type conversions,
- applying format masks,
- assigning fields to a sequence number,
- substituting constants,
- setting values to NULL or DEFAULT based on a condition,
- using simple SQL operators (e.g., UPPER , TRUNC , SUBSTR , etc.)

Thus, the transformation tasks, which will operate over the extracted the data, should cure the upper issues. As a result, a set of activities should be performed, such as:

1. reformatting,
2. recalculating,
3. modifying key structures,
4. adding an element of time,
5. identifying default values,
6. supplying logic to choose between multiple sources,
7. summarizing,
8. merging data from multiple sources etc

5.1.4 Loading

Finally, the Loading phase indicates the loading of the extracted and transformed information into the data warehouse. As the aforementioned phases, this phase has its own challenges. The major issue is the discrimination between the records, which are totally new, and the ones that should be updated. Modern ETL techniques provide some mechanisms regarding this problem but these solution are based on language prediction. Furthermore, SQL queries are not sufficient, because of the fact that when records are inserted one by one in a loop which scan the whole data warehouse, the performance becomes extremely low in terms of time, especially for the vast volumes of data. Another problem is the usage of the rollback segments and/or log files during the loading tasks. From a performance perspective, an possible solution is to deactivate this log files. On the other hand, this option contains some risk, especially in case of a loading failure. So far, it seems that the best solution is to use the batch loading tools, which are offered by most RDBMSs. Alternative solutions that can utilize the loading task suggest the creation of temporary tables, the reduction of inter-process wait states, and/or the maximization of concurrent the hardware [7].

5.1.5 ETL in Viggo

According to Viggo needs, the ETL tasks will be defined as follow:

Extraction Data extraction from Viggo Shift Rosters, Viggo Training Managers and Viggo Youforce database. More specifically, these data will fill the fact tables of Viggo data warehouse regarding the training, the shifts, the contracts and the payments.

Transformation Transformation of data schema, which are retrieved from Viggo Shift Rosters and Viggo Training Managers, into a relational structure. Precisely, the output of this phase will be set of records that will be ready to be loaded into the RDBMS, which implements Viggo data warehouse tie.

Loading Loading of Transformation phase outputs into Viggo data warehouse, with the use of RDBMS's batch functions.

5.2 CSVQL

As it has been discussed in Chapter 2, ETL work for CSV data sources requires a mechanism, which can query these CSV files in an SQL-like manner. In order to fulfill the ETL obligations in Viggo data warehouse, with respect to related work suggestions, a query framework for CSV querying, has been designed. More specifically, this framework is called CSVQL (Comma-Separated Values Query Language) and, as its name indicates, consists a query language, which allows the user to extract information from CSV (Comma-Separated Values) data sources. Precisely, this query language's feature is the ability to retrieve and transform data from CSV spreadsheets into relational tables. These manipulation and formatting tasks can achieve the Extraction and Transform requirements of ETL process, within Viggo's data warehouse. The Loading task can be solved by the bulk functions of the DBMS (Database Management System), which implements the "Data warehouse" tier. The logo of CSVQL framework is presented by Figure 5.1.

Figure 5.1: The logo of CSVQL framework.

The syntax of CSVQL is similar to SQL and the reason is the intention to keep the productivity ratio of the framework at a high level. As productivity, it is defined how handy and useful a query language is. More specifically, with the term productivity, the functions that a query language offers to its users to assist them, are indicated. Productivity consists of a major metric of evaluation, because a more friendly-user language is more preferable than another one, even with more utilities and better performance [21]. In this case, developers familiar with SQL are able to quickly learn and use CSVQL in a very short period of time. However, there are some significant differences, between these two query languages, because of the fact that they operate on data with different structure. SQL operates over relational tables and CSVSQL over unstructured CSV grids.

5.2.1 Clauses overview

The syntax of the query language is composed of the following clauses:

Table 5.1: CSVQL Clauses

Clause	*Usage*
SELECT	Selects which information will be written in the output file, and in which order.
FROM	Indicates the source of a cells-set to be operated upon.
WHERE	Filters the cells that match a condition.
OUTPUT	Indicates in which file the query result will be stored.

As the Table 5.1 presents, each clause starts with a keyword and it is separated by spaces from its arguments. All the clauses are mandatory, except from the clause WHERE, which is optional. Each clause is described in details in the following subsections.

5.2.2 SELECT

Description

The SELECT clause is used to specify the information, which will be written in the output file in the OUTPUT clause. More specifically, this information will be consisted by data, which are retrieved from data sources, and by strings, which the user inputs. The aforementioned information will be written to the output file, in a order that is defined in the SELECT clause.

Examples

- The Query 5.2 duplicates the file "employees.csv". If * is used, all of the data of the data sources will be retrieved and will be written in their original order (row by row and column after column).

```
SELECT  *

FROM    employees.csv
```

Figure 5.2: Select clause example 1

- The Query 5.3 outputs the cell [1,A] of the file "employees.csv". More specifically, the data, which is stored inside cell [1,A] of query's data source, will be retrieved and will be written to the output file, in a column with header name "Employee".

```
SELECT  [1,A] AS Employee

FROM    employees.csv
```

Figure 5.3: Select clause example 2

- The Query 5.4 outputs the cell [1,A] of the data source "filename" and the string "Active". More specifically, the data, which is stored inside cell [1,A] from data source "filename" will be retrieved and will be written in a column with header name "Employee". Next this column, there will be another column with header name "Status" and this column will contain the string "Active".

```
SELECT  filename.[1,A]  AS Employee, "Active" AS Status

FROM    employees.csv   AS filename
```

Figure 5.4: Select clause example 3

5.2.3 FROM

Description

The FROM clause is used to specify the data sources of the query. A data source can be an entire CSV file or a row-set from a specific CSV file. More specifically, in FROM clause the user can create one or multiple data sources from a CSV file. For better understanding, a data source can be considered as a traditional SQL table. However, a data source is not a table, but it is a row-by-row path of cells inside a specific CSV file. This means that a specific data source defines a specific path, which the query parser will follow inside a CSV file, in order to obtain the required information.

Examples

- The Query 5.5 duplicates the file "employees.csv". The data source is a row-by-row path of the whole CSV file "employees.csv". The data of all its cells will be retrieved, row by row and column after column, and be written to the output file.

```
SELECT  *

FROM    employees.csv
```

Figure 5.5: From clause example 1

- The Query 5.6 outputs the cell [1,A] of the data source, which is named as "file", which is the whole CSV file employees.csv. The data of its cell [1,A] will be retrieved and be written to the output file, in a column with header name "Employee".

```
SELECT  file.[1,A]     AS Employee

FROM    employees.csv AS file
```

Figure 5.6: From clause example 2

- The Query 5.7 outputs the cell [1,A] of the data source "file1", the cell [2,B] of the data source "file1" and the string "Active". The data source, which is named as "file1", is a row-by-row path of the whole file "employees.csv". The data of its cell [1,A] will be retrieved and be written to the output file with header name "Employee". In the same manner the data source, which is named as "file2", is a row-by-row path of the whole file "functions.csv". The data of its cell [2,B] will be retrieved and be written to the output file next to column "Employee", with header name "Function". Finally, next to column "Function", there will be another column with name "Status" and this column will have the string "Active".

```
SELECT  file1.[1,A] AS Employee,
        file2.[2,B] AS Function,
        "Active"    AS Status

FROM    employees.csv AS file1
        functions.csv AS file2
```

Figure 5.7: From clause example 3

- The Query 5.8 outputs some specific cells of the data source "employee" and some other cells of the data source "function".

```
SELECT  employee.[ROW, COLUMN]   AS Employee,
        function.[1, COLUMN]     AS Function

FROM    employees.csv.[1:10 WITH STEP 1, A] AS employee,
        functions.csv.[ *,B:J WITH STEP 2]  AS function
```

Figure 5.8: From clause example 4

The data source, which is named as "employee", is a cell-by-cell path within the CSV file employees.csv. This path contains the column A of the rows 1,2,3 . . . 10, and this cell-by-cell path is retrieved row by row. This means that the cells of the 1^{st} row will be retrieved

first, then the cells of the 2^{nd} row, etc. According to this definition, the data of the cells, which are included in this specific path are the following:

1. cell[1,A]
2. cell[2,A]
3. cell[3,A]
4. cell[4,A]
5. cell[5,A]
6. cell[6,A]
7. cell[7,A]
8. cell[8,A]
9. cell[9,A]
10. cell[10,A]

When a cell-by-cell path is retrieved, there are two variables, which indicate the coordinates of the current position of this specific path. The variable ROW indicates the current row and the variable COLUMN indicates the current column of the file, which this specific path operates on. Thus, the variables ROW and COLUMN of an expression in SELECT clause indicate the index of the current position of a specific row-by-row path, which is defined in the FROM clause. In this example the data of the cells of the path "employee" will be retrieved and will be written to of the output file. More specifically, they will be written in first column with header name "Employee".

In the same manner, the data source, which is named as "function", is a cell-by-cell path within the CSV file functions.csv. The "function" path contains the columns B,D,F,H,J of all(*) the rows of the CSV file functions.csv (assuming that this CSV file has 15 rows). The data of the cells, which are included in this specific path are the following:

1. cell[1,A]
2. cell[2,A]
3. cell[3,A]
4. cell[4,A]
5. cell[5,A]
6. cell[6,A]
7. cell[7,A]
8. cell[8,A]
9. cell[9,A]
10. cell[10,A]
11. cell[11,A]
12. cell[12,A]
13. cell[13,A]
14. cell[14,A]
15. cell[15,A]

5.2.4 WHERE

Description

The WHERE clause is used to return only information that match a specified condition. More specifically, WHERE clause is used to specify conditions while fetching the data from a data-source or joining with multiple data-sources. If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.

The simple comparison operators are

- <=
- >
- =
- <>
- <
- >=
- !=

Both comparison operators != and <> mean not-equal. Strings are compared by lexicographic value. Note that equality is indicated by =, not == as in most computer languages. Comparing to null is done using is null or is not null. Multiple conditions can be joint by using the logical operators AND, OR, and NOT. Parentheses can be used to define explicit precedence. The WHERE clause also supports some more complex string comparison operators. These operators take two strings as arguments; any non-string arguments (for example, dates or numbers) will be converted to strings before comparison. String matching is case sensitive (you can use upper() or lower() scalar functions to work around that).

contains - A sub-string match. whole contains part is true if part is anywhere within whole.

starts_with - A prefix match. value starts with prefix is true if prefix is at the beginning of value.

ends_with - A suffix match. value ends with suffix is true if suffix is at the end of value.

matches - A (preg) regular expression match. haystack matches needle is true if the regular expression in needle matches haystack.

like -A text search that supports two wildcards: %, which matches zero or more characters of any kind, and _ (underscore), which matches any one character.

Examples

- The Query 5.9 outputs only the employees with name that starts with "Jo". For example, an employee with name "Joost", will be retrieved and will be written in a column with header name "Employee".

```
SELECT   employee.[ROW,COLUMN] AS Employee
FROM     employees.csv.[1:10 WITH STEP 1, A] AS employee
WHERE    employee.[ROW,COLUMN] starts_with("Jo")
```

Figure 5.9: Where clause example 1

- The Query 5.10 outputs only the salaries that are higher than 4000 euros. For example, a salary with which is 3000 euros, will not be presented.

```
SELECT   salaries.[ROW,COLUMN] AS HighSalaries
FROM     salaries.csv.[1:10 WITH STEP 1, C]  AS salaries
WHERE    salaries.[ROW,COLUMN] > 4000
```

Figure 5.10: Where clause example 2

5.2.5 OUTPUT

Description

The OUTPUT clause is used to specify where the results of the query will be stored. More specifically, OUTPUT indicates where the result of SELECT clause will be stored.

Examples

- The Query 5.11 creates a duplication of data source "employees.csv" with the name "duplicateEmployees.csv".

```
SELECT   *
FROM     employees.csv
OUTPUT   duplicateEmployees.csv
```

Figure 5.11: Output clause example 1

- The Query 5.12 creates files named "listHighSalaries.csv", which contains a list with all the high salaries.

```
SELECT   salaries.[ROW,COLUMN] AS HighSalaries

FROM     salaries.csv.[1:10 WITH STEP 1, C]  AS salaries

WHERE    salaries.[ROW,COLUMN] > 4000

OUTPUT   listHighSalaries.csv
```

Figure 5.12: Output clause example 2

In this example, only the employees with name that starts with "Jo", e.g. Joost, will be retrieved and will be retrieved and will be written in a column with header name "Employee".

5.2.6 Querying Viggo Training Manager

The objective of this example is to extract information regarding the training of Viggo employees. More specifically, there is a CSV file with name "training.csv" which contains all the necessary information but not in a relational structure, as it is presented by the Figure **??**. However, there is a need to create a new CSV file, which has a relational structure as it is presented by the Figure 5.13. This specific structure will make the data more suitable for Viggo's data warehouse.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Trainingsmatrix Operations department VEA		Show re-training within.....months ->	2	General								
2	Overdue	Current	Training Active ->		A	A	A	A	A	A	A	A	A
3	uren mw						2		1.5				
4	uren begeleiding												
5	Current date:	8/24/2016	Recurring in months ->		1000	30	1000	36	1000	24	1000	36	
6	Medewerkersnummer	Medewerkersnummer	Functie	Status	Safety and security		Human Factor		Dangerous Goods Cat 9		Immigration Visa exam		Praktijkopleiding pas
7					IN	CR	IN	CR	IN	CR	IN	CR	IN
8	[redacted]	[redacted]	EA SIA employee/security	Active	1/22/2014	1/22/2014	6/13/2013	6/13/2013	12/23/2013	12/24/2015	8/11/2015	8/11/2015	8/18/2007
9	[redacted]	[redacted]	EA Passage Coördinator	Active	7/26/2013	1/28/2016	2/4/2013	2/4/2013	1/24/2014	12/12/2015	8/9/2015	8/9/2015	4/4/2011
10	[redacted]	[redacted]	EA SIA employee	Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	1/5/2014	11/10/2015	8/28/2015	8/28/2015	10/11/2011
11	[redacted]	[redacted]	EA Passage Team Leader	Active	3/17/2008	4/26/2014	6/13/2013	6/13/2013	12/30/2013	11/3/2015	8/26/2015	8/26/2015	2/20/2008
12	[redacted]	[redacted]	EA Passage Team Leader	Active	2/11/2008	6/26/2015	6/13/2013	6/13/2013	1/28/2014	1/11/2016	8/31/2015	8/31/2015	10/11/2011
13	[redacted]	[redacted]	EA SIA employee	Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	2/6/2012	2/4/2016	8/10/2015	8/10/2015	1/20/2011
14	[redacted]	[redacted]	EA SIA Coordinator	Active	11/12/2007	8/27/2014	6/13/2013	6/13/2013	1/22/2014	11/18/2015	8/14/2015	8/14/2015	6/2/2008
15	[redacted]	[redacted]	EA SIA employee	Active	6/15/2010	1/18/2014	6/13/2013	6/13/2013	12/28/2013	11/11/2015	8/23/2015	8/23/2015	10/11/2011
16	[redacted]	[redacted]	EA SIA Coordinator	Active	12/4/2006	1/21/2014	6/13/2013	6/13/2013	1/23/2014	11/24/2015	9/2/2015	9/2/2015	4/21/2005
17	[redacted]	[redacted]	EA Back-office	Active	4/22/2010	1/17/2014	6/13/2013	6/13/2013	1/10/2012	na			
18	[redacted]	[redacted]	EA SIA employee/security	Active	1/21/2014	1/21/2014	6/13/2013	6/13/2013	1/10/2014	11/14/2015	8/11/2015	8/11/2015	10/11/2011
19	[redacted]	[redacted]	EA Passage employee	Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	12/31/2013	11/19/2015	8/14/2015	8/14/2015	10/11/2011
20	[redacted]	[redacted]	EA Passage Coördinator	Active	6/27/2013	1/24/2014	6/13/2013	6/13/2013	1/3/2014	11/7/2015	8/15/2015	8/15/2015	10/11/2011
21	[redacted]	[redacted]	EA Passage employee/security	Active	10/17/2009	5/28/2014	6/13/2013	6/13/2013	1/19/2014	11/27/2015	8/27/2015	8/27/2015	11/10/2011
22	[redacted]	[redacted]	EA SIA employee/security	Active	3/30/2009	1/21/2014	6/13/2013	6/13/2013	6/1/2010	12/25/2014	8/19/2015	8/19/2015	3/23/2012
23	[redacted]	[redacted]	EA Passage Coördinator	Active	2/23/2012	7/17/2014	7/26/2013	7/26/2013	6/20/2011	8/11/2015	8/17/2015	8/17/2015	1/1/2011
24	[redacted]	[redacted]	EA Passage employee/security	Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	2/9/2011	3/4/2015	9/8/2015	9/8/2015	3/7/2012
25	[redacted]	[redacted]	EA SIA Team Leader	Active	6/17/2010	1/20/2014	6/13/2013	6/13/2013	1/17/2012	11/23/2015	8/14/2015	8/14/2015	5/14/1996
26	[redacted]	[redacted]	EA SIA employee	Active	9/17/2009	5/8/2014	6/13/2013	6/13/2013	5/29/2013	12/6/2015			10/11/2011
27	[redacted]	[redacted]	EA Passage employee/security	Active	3/7/2011	1/19/2014	6/13/2013	6/13/2013	6/8/2011	4/25/2015	8/20/2015	8/20/2015	5/10/2011
28	[redacted]	[redacted]	EA SIA employee/security	Active	11/2/2009	1/21/2014	6/13/2013	6/13/2013	1/18/2011	1/25/2015	8/20/2015	8/20/2015	10/11/2011

However, there is a need to create a new CSV file which will have the following structure:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	tid	skill	employee	startdate	enddate														
2	id	Safety and security	[redacted]	1/22/2014	1/22/2014														
3	id	Human Factor	[redacted]	6/13/2013	6/13/2013														
4	id	Dangerous Goods Cat 9	[redacted]	12/23/2013	12/24/2015														
5	id	Immigration Visa exam	[redacted]	8/11/2015	8/11/2015														
6	id	Praktijkopleiding passage	[redacted]	8/18/2007															
7	id	PRM Passage	[redacted]	11/12/2012	9/28/2015														
8	id	Check in	[redacted]																
9	id	Pre scan	[redacted]																
10	id	Gate	[redacted]																
11	id	Opzetten	[redacted]																
12	id	Preparateuren	[redacted]																
13	id	Drop&Go	[redacted]																
14	id	Servicebalie opleiding	[redacted]	1/8/2014	1/8/2014														
15	id	Informatiebalie opleiding	[redacted]	1/8/2014	1/8/2014														
16	id	Lost and Found opleiding	[redacted]	11/10/2011	1/8/2014														
17	id	Refreshment beveiliging	[redacted]	3/27/2012	11/22/2012														
18	id	BHV	[redacted]	3/1/2012	2/24/2015														
19	id	ASM test Transavia (OM->	[redacted]																
20	id	Train de trainer	[redacted]	10/22/2013	12/3/2014														
21	id	Auditor training	[redacted]																
22	id	Safety and security	[redacted]	7/26/2013	1/28/2016														
23	id	Human Factor	[redacted]	2/4/2013	2/4/2013														
24	id	Dangerous Goods Cat 9	[redacted]	1/24/2014	12/12/2015														
25	id	Immigration Visa exam	[redacted]	8/9/2015	8/9/2015														
26	id	Praktijkopleiding passage	[redacted]	4/4/2011															
27	id	PRM Passage	[redacted]	11/8/2012	9/28/2015														
28	id	Check in	[redacted]	8/18/2015	8/18/2015														
29	id	Pre scan	[redacted]	8/18/2015	8/18/2015														
30	id	Gate	[redacted]	8/18/2015	8/18/2015														
31	id	Opzetten	[redacted]																
32	id	Preparateuren	[redacted]																
33	id	Drop&Go	[redacted]	8/18/2015	8/18/2015														
34	id	Servicebalie opleiding	[redacted]																
35	id	Informatiebalie opleiding	[redacted]																
36	id	Lost and Found opleiding	[redacted]																
37	id	Refreshment beveiliging	[redacted]																

Figure 5.13: The relational structure of Viggo training.csv file, after CSVQL application.

This result can be achieved by executing the 5.14 query:

```
SELECT   "id"                  AS tid,
         f1.[ROW,COLUMN]       AS skill,
         f2.[ROW,A]            AS employee,
         f3.[ROW,COLUMN]       AS startdate,
         f4.[ROW,COLUMN]       AS enddate

FROM     training.csv.[6, E:AR WITH STEP 2]                    AS f1,
         training.csv.[8:28 WITH STEP 1, {A,D}]                AS f2,
         training.csv.[8:28 WITH STEP 1, E:AR WITH STEP 2]     AS f3,
         training.csv.[8:28 WITH STEP 1, F:AS WITH STEP 2]     AS f4

WHERE    f1.COLUMN = f3.COLUMN      AND
         f2.ROW=f3.ROW              AND
         f1.COLUMN = f4.COLUMN + 1  AND
         f2.ROW=f4.ROW              AND
         f2.[ROW,D] = "Active"

OUTPUT   relational.csv
```

Figure 5.14: Querying Viggo training.csv file

FROM Clause

Inside FROM clause, 4 cell-by-cell paths are created:

- **f1 (source file: training.csv)**, which indicates that
  from row 6:
  the columns E to AR with step 2 (E, G, I, K, . . ., AP, AR) are retrieved.
  As a result, the cells which are inside this path are

  1. cell[6,E]
  2. cell[6,G]
  3. cell[6,I]
  4. cell[6,K] . . .

  The visualization of this path is presented by Figure 5.15.

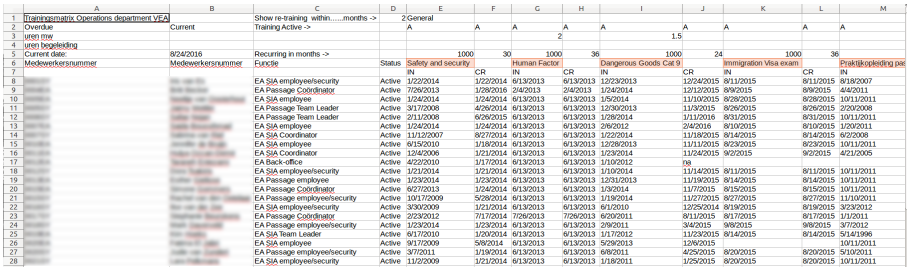
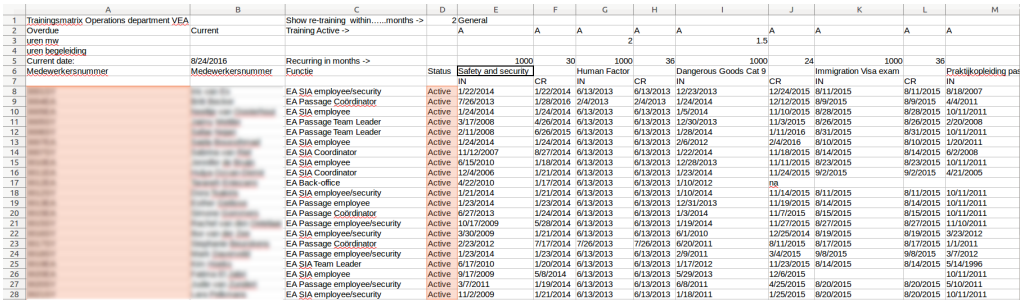


Figure 5.15: The f1 row-by-row path within Viggo training.csv file.

- **f2 (source file: training.csv)**, which indicates that
  from rows 8 to 28 with step 1 (8, 9, 10, 11, . . ., 27, 28):
  the columns A, D are retrieved.
  As a result, the cells which are inside this path are

1. cell[8,A]
2. cell[8,D]
3. cell[9,A]
4. cell[9,D] . . .

The visualization of this path is presented by Figure 5.16.



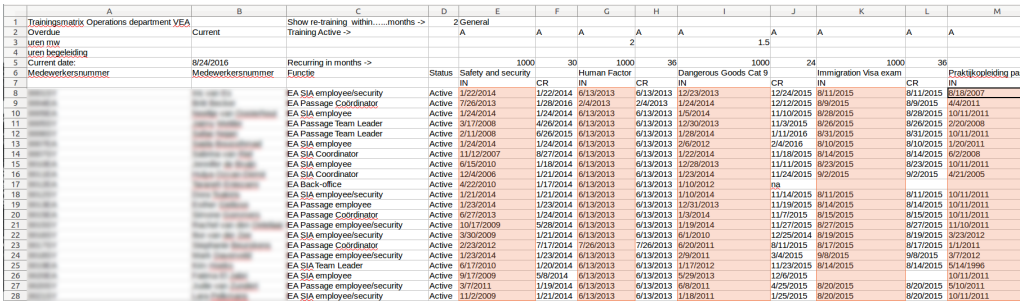
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Trainingsmatrix Operations department VEA		Show re-training within.....months ->	2	General								
2	Overdue	Current	Training Active ->		A	A	A	A	A	A	A	A	A
3	uren mw						2		1.5				
4	uren begeleiding												
5	Current date:	8/24/2016	Recurring in months ->		1000	30	1000	36	1000	24	1000	36	
6	Medewerkersnummer	Medewerkersnummer	Functie	Status	Safety and security		Human Factor		Dangerous Goods Cat 9		Immigration Visa exam		Praktijkopleiding pas
7					IN	CR	IN	CR	IN	CR	IN	CR	IN
8	[redacted]	[redacted]	EA SIA employeee/security	Active	1/22/2014	1/22/2014	6/13/2013	6/13/2013	12/23/2013	12/24/2015	8/11/2015	8/11/2015	8/18/2007
9	[redacted]	[redacted]	EA Passage Coördinator	Active	7/26/2013	1/28/2016	2/4/2013	2/4/2013	1/24/2014	12/12/2015	8/9/2015	8/9/2015	4/4/2011
10	[redacted]	[redacted]	EA SIA employee	Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	1/5/2014	11/10/2015	8/28/2015	8/28/2015	10/11/2011
11	[redacted]	[redacted]	EA Passage Team Leader	Active	3/17/2008	4/26/2014	6/13/2013	6/13/2013	12/30/2013	11/3/2015	8/26/2015	8/26/2015	2/20/2008
12	[redacted]	[redacted]	EA Passage Team Leader	Active	2/11/2008	6/26/2015	6/13/2013	6/13/2013	1/28/2014	1/11/2016	8/31/2015	8/31/2015	10/11/2011
13	[redacted]	[redacted]	EA SIA employee	Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	2/6/2012	2/4/2016	8/10/2015	8/10/2015	1/20/2011
14	[redacted]	[redacted]	EA SIA Coordinator	Active	11/12/2007	8/27/2014	6/13/2013	6/13/2013	1/22/2014	11/18/2015	8/14/2015	8/14/2015	6/2/2008
15	[redacted]	[redacted]	EA SIA employee	Active	6/15/2010	1/18/2014	6/13/2013	6/13/2013	12/28/2013	11/11/2015	8/23/2015	8/23/2015	10/11/2011
16	[redacted]	[redacted]	EA SIA Coordinator	Active	12/4/2006	1/21/2014	6/13/2013	6/13/2013	1/23/2014	11/24/2015	9/2/2015	9/2/2015	4/21/2005
17	[redacted]	[redacted]	EA Back-office	Active	4/22/2010	1/17/2014	6/13/2013	6/13/2013	1/10/2012	na			
18	[redacted]	[redacted]	EA SIA employeee/security	Active	1/21/2014	1/21/2014	6/13/2013	6/13/2013	1/10/2014	11/14/2015	8/11/2015	8/11/2015	10/11/2011
19	[redacted]	[redacted]	EA Passage employee	Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	12/31/2013	11/19/2015	8/14/2015	8/14/2015	10/11/2011
20	[redacted]	[redacted]	EA Passage Coördinator	Active	6/27/2013	1/24/2014	6/13/2013	6/13/2013	1/3/2014	11/7/2015	8/15/2015	8/15/2015	10/11/2011
21	[redacted]	[redacted]	EA Passage employee/security	Active	10/17/2009	5/28/2014	6/13/2013	6/13/2013	1/19/2014	11/27/2015	8/27/2015	8/27/2015	11/10/2011
22	[redacted]	[redacted]	EA SIA employeee/security	Active	3/30/2009	1/21/2014	6/13/2013	6/13/2013	6/1/2010	12/25/2014	8/19/2015	8/19/2015	3/23/2012
23	[redacted]	[redacted]	EA Passage Coördinator	Active	2/23/2012	7/17/2014	7/26/2013	7/26/2013	6/20/2011	8/11/2015	8/17/2015	8/17/2015	1/1/2011
24	[redacted]	[redacted]	EA Passage employee/security	Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	2/9/2011	3/4/2015	9/8/2015	9/8/2015	3/7/2012
25	[redacted]	[redacted]	EA SIA Team Leader	Active	6/17/2010	1/20/2014	6/13/2013	6/13/2013	1/17/2012	11/23/2015	8/14/2015	8/14/2015	5/14/1996
26	[redacted]	[redacted]	EA SIA employee	Active	9/17/2009	5/8/2014	6/13/2013	6/13/2013	5/29/2013	12/6/2015			10/11/2011
27	[redacted]	[redacted]	EA Passage employeee/security	Active	3/7/2011	1/19/2014	6/13/2013	6/13/2013	6/8/2011	4/25/2015	8/20/2015	8/20/2015	5/10/2011
28	[redacted]	[redacted]	EA SIA employeee/security	Active	11/2/2009	1/21/2014	6/13/2013	6/13/2013	1/18/2011	1/25/2015	8/20/2015	8/20/2015	10/11/2011

Figure 5.16: The f2 row-by-row path within Viggo training.csv file.

- **f3 (source file: training.csv)**, which indicates that
  from rows 8 to 28 with step 1 (8, 9, 10, 11, . . ., 27, 28):
  the columns E to AR with step 2 (E, G, I, K, . . ., AP, AR) are retrieved.
  As a result, the cells which are inside this path are

  1. cell[8,E]
  2. cell[8,G] . . .
  3. cell[9,E]
  4. cell[9,G] . . .

  The visualization of this path is presented by Figure 5.17.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Trainingsmatrix Operations department VEA		Show re-training within.....months ->	2	General								
2	Overdue	Current	Training Active ->		A	A	A	A	A	A	A	A	A
3	uren mw						2		1.5				
4	uren begeleiding												
5	Current date:	8/24/2016	Recurring in months ->		1000	30	1000	36	1000	24	1000	36	
6	Medewerkersnummer	Medewerkersnummer	Functie	Status	Safety and security		Human Factor		Dangerous Goods Cat 9		Immigration Visa exam		Praktijkopleiding pa
7					IN	CR	IN	CR	IN	CR	IN	CR	IN
8	[redacted]	[redacted]	EA SIA employeee/security	Active	1/22/2014	1/22/2014	6/13/2013	6/13/2013	12/23/2013	12/24/2015	8/11/2015	8/11/2015	8/18/2007
9	[redacted]	[redacted]	EA Passage Coördinator	Active	7/26/2013	1/28/2016	2/4/2013	2/4/2013	1/24/2014	12/12/2015	8/9/2015	8/9/2015	4/4/2011
10	[redacted]	[redacted]	EA SIA employee	Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	1/5/2014	11/10/2015	8/28/2015	8/28/2015	10/11/2011
11	[redacted]	[redacted]	EA Passage Team Leader	Active	3/17/2008	4/26/2014	6/13/2013	6/13/2013	12/30/2013	11/3/2015	8/26/2015	8/26/2015	2/20/2008
12	[redacted]	[redacted]	EA Passage Team Leader	Active	2/11/2008	6/26/2015	6/13/2013	6/13/2013	1/28/2014	1/11/2016	8/31/2015	8/31/2015	10/11/2011
13	[redacted]	[redacted]	EA SIA employee	Active	1/24/2014	1/24/2014	6/13/2013	6/13/2013	2/6/2012	2/4/2016	8/10/2015	8/10/2015	1/20/2011
14	[redacted]	[redacted]	EA SIA Coordinator	Active	11/12/2007	8/27/2014	6/13/2013	6/13/2013	1/22/2014	11/18/2015	8/14/2015	8/14/2015	6/2/2008
15	[redacted]	[redacted]	EA SIA employee	Active	6/15/2010	1/18/2014	6/13/2013	6/13/2013	12/28/2013	11/11/2015	8/23/2015	8/23/2015	10/11/2011
16	[redacted]	[redacted]	EA SIA Coordinator	Active	12/4/2006	1/21/2014	6/13/2013	6/13/2013	1/23/2014	11/24/2015	9/2/2015	9/2/2015	4/21/2005
17	[redacted]	[redacted]	EA Back-office	Active	4/22/2010	1/17/2014	6/13/2013	6/13/2013	1/10/2012	na			
18	[redacted]	[redacted]	EA SIA employeee/security	Active	1/21/2014	1/21/2014	6/13/2013	6/13/2013	1/10/2014	11/14/2015	8/11/2015	8/11/2015	10/11/2011
19	[redacted]	[redacted]	EA Passage employee	Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	12/31/2013	11/19/2015	8/14/2015	8/14/2015	10/11/2011
20	[redacted]	[redacted]	EA Passage Coördinator	Active	6/27/2013	1/24/2014	6/13/2013	6/13/2013	1/3/2014	11/7/2015	8/15/2015	8/15/2015	10/11/2011
21	[redacted]	[redacted]	EA Passage employee/security	Active	10/17/2009	5/28/2014	6/13/2013	6/13/2013	1/19/2014	11/27/2015	8/27/2015	8/27/2015	11/10/2011
22	[redacted]	[redacted]	EA SIA employeee/security	Active	3/30/2009	1/21/2014	6/13/2013	6/13/2013	6/1/2010	12/25/2014	8/19/2015	8/19/2015	3/23/2012
23	[redacted]	[redacted]	EA Passage Coördinator	Active	2/23/2012	7/17/2014	7/26/2013	7/26/2013	6/20/2011	8/11/2015	8/17/2015	8/17/2015	1/1/2011
24	[redacted]	[redacted]	EA Passage employee/security	Active	1/23/2014	1/23/2014	6/13/2013	6/13/2013	2/9/2011	3/4/2015	9/8/2015	9/8/2015	3/7/2012
25	[redacted]	[redacted]	EA SIA Team Leader	Active	6/17/2010	1/20/2014	6/13/2013	6/13/2013	1/17/2012	11/23/2015	8/14/2015	8/14/2015	5/14/1996
26	[redacted]	[redacted]	EA SIA employee	Active	9/17/2009	5/8/2014	6/13/2013	6/13/2013	5/29/2013	12/6/2015			10/11/2011
27	[redacted]	[redacted]	EA Passage employeee/security	Active	3/7/2011	1/19/2014	6/13/2013	6/13/2013	6/8/2011	4/25/2015	8/20/2015	8/20/2015	5/10/2011
28	[redacted]	[redacted]	EA SIA employeee/security	Active	11/2/2009	1/21/2014	6/13/2013	6/13/2013	1/18/2011	1/25/2015	8/20/2015	8/20/2015	10/11/2011

Figure 5.17: The f3 row-by-row path within Viggo training.csv file.

- **f4 (source file: training.csv)**, which indicates that
  from rows 8 to 28 with step 1 (8, 9, 10, 11, . . ., 27, 28):
  the columns F to AS with step 2 (F, H, J, L, . . ., AQ, AS) are retrieved.
  As a result, the cells which are inside this path are

1. cell[8,F]
2. cell[8,H] . . .
3. cell[9,F]
4. cell[9,H] . . .

The visualization of this path is presented by Figure 5.18.

	C	D	E	F	G	H	I	J	K	L	
1	Show re-training within.....months ->	2	General								
2	Training Active ->		A	A	A		A	A	A	A	A
3	Recurring in months ->		1000	24	1000		1000	30	1000	36	
4	Functie	77	Platform Safety		Introductietraining		Safety & Security		Human Factor training		Dangero
5		Status	IN	CR	IN	CR	IN	CR	IN	CR	IN
6	EA Platform Coördinator	Active	12/21/2012	12/19/2014	NA		5/18/2010	1/21/2014	1/21/2013	12/16/2015	12/1/201
7	EA Assistant Platform Supervisor	Active	9/28/2012	10/6/2014	NA		11/28/2006	5/1/2015	1/9/2013	12/4/2015	11/4/201
8	EA Platform Coördinator	Active	6/1/2010	4/14/2014	NA		7/5/2010	1/20/2014	4/22/2013	4/22/2013	12/10/20
9	EA Assistant Platform Supervisor	Active	2/3/2011	3/31/2015	NA		3/19/2009	1/20/2014	1/22/2013	12/4/2015	12/9/201
10	EA Assistant Platform Supervisor	Active	12/2/2010	1/29/2015	NA		5/18/2010	1/20/2014	2/1/2013	1/13/2016	12/1/201
11	EA Bagage Coördinator	Active	11/9/2012	6/12/2014	NA		5/20/2010	4/7/2014	2/17/2013	1/13/2016	6/15/201
12	EA Platform Coördinator	Active	12/9/2010	4/1/2015	NA		11/29/2006	3/13/2014	1/21/2013	12/16/2015	12/9/201
13	EA Platform Coördinator	Active	2/10/2010	4/1/2015	NA		7/27/2010	3/10/2014	2/19/2013	1/21/2016	12/20/20
14	EA Platform Coördinator	Active	3/17/2011	3/3/2015	NA		11/30/2006	4/14/2015	3/13/2013	3/8/2016	8/25/201
15	EA Platform Coördinator	Active	11/4/2010	12/10/2014	NA		7/20/2010	1/21/2014	1/19/2013	1/4/2016	8/30/201
16	EA Platform Employee 1	Active	1/1/2011	3/31/2015	NA		12/6/2006	5/28/2015	2/28/2013	1/4/2016	9/8/2011
17	EA Platform Coördinator	Active	3/10/2011	3/31/2015	NA		12/5/2006	10/11/2015	4/11/2013	4/11/2013	11/16/20
18	EA Assistant Platform Supervisor	Active	11/17/2010	4/1/2015	4/9/2014		12/6/2006	2/25/2014	2/18/2013	1/17/2016	8/15/201
19	EA Platform Employee 3	Active	12/7/2008	5/17/2014	NA		12/4/2006	12/19/2014	4/22/2013	4/22/2013	9/25/201
20	EA Platform Employee 1	Active	11/22/2010	4/29/2014	NA		1/16/2008	8/27/2015	2/22/2013	1/22/2016	11/24/20
21	EA Platform Employee 1	Active	3/3/2011	3/31/2015	NA		6/9/2009	4/14/2014	2/4/2013	1/18/2016	3/3/2011
22	EA Assistant Platform Supervisor	Active	12/15/2014	12/15/2014	NA		3/27/2013	9/27/2015	1/30/2013	12/14/2015	10/3/200
23	EA Platform Employee 1	Active	12/12/2010	4/9/2015	NA		12/1/2006	4/14/2014	5/14/2013	5/14/2013	12/12/20
24	EA Bagage Coördinator	Active	4/6/2011	3/31/2015	NA		3/30/2010	1/20/2014	3/8/2013	3/7/2016	9/8/2011
25	EA Bagage Coördinator	Active	8/10/2011	7/29/2015	NA		6/17/2010	1/20/2014	5/14/2013	5/14/2013	12/14/20
26	EA Bagage Coördinator	Active	7/1/2010	3/13/2015	NA		6/17/2010	1/18/2014	1/30/2013	12/14/2015	12/7/201
27	EA Bagage Coördinator	Active	6/1/2011	3/13/2015	NA		11/12/2009	1/18/2014	1/31/2013	12/15/2015	7/26/201
28	EA Platform Employee 1	Active	5/4/2011	4/5/2015	NA		1/24/2011	1/20/2014	1/31/2013	12/16/2015	8/11/201
29	EA Assistant Platform Supervisor	Active	1/1/2011	12/9/2014	4/9/2014		12/5/2006	11/4/2014	4/10/2013	4/10/2013	12/9/201
30	EA Bagage Coördinator	Active	7/5/2011	3/13/2015	NA		2/21/2012	7/21/2014	4/22/2013	4/22/2013	9/6/2011

Figure 5.18: The f4 row-by-row path within Viggo training.csv file.

WHERE Clause

As it mentioned above, the data sources f1, f2, f3 and f4 can be considered as SQL tables which participate in a SQL query. Thus, if there is a need to combine these data sources, there is an obligation to apply a mapping pattern. This mapping filter will filter unwanted combinations of information. In case there is no filter, the result will be a Cartesian product of all the cells from these specific data-sources, which means that every possible combination of cells will be in the output. As a result, in the WHERE clause there are conditions, which extract only those records that fulfill a specified criterion, by synchronizing the variables of the row-by-row paths and/or approve only the cells with the desired data.

The 5 conditions, which consist the filter in this CSVQL query are

- **f1.COLUMN = f3.COLUMN**
  This condition looks like a SQL condition, which joins two tables on a column. However, this condition indicates that data-source f1 and data-source f3 are joined on their columns. This means that the query will process data-sources f1 and f3 and will combine only the cells, which have the same column coordinate.

- **f2.ROW = f3.ROW**
  In the same manner, the query will process data-sources f2 and f3 and will combine only the cells, which have the same column coordinate.

- **f1.COLUMN = f4.COLUMN+1**
  This condition looks like a SQL condition, which joins two tables on a column. However, this condition indicates that data-source f1 and data-source f4 are joined on their columns. This means that the query will process data-sources f1 and f4 and will combine only the cells, which have the same column coordinate, as it is defined by the condition f1.COLUMN = f4.COLUMN+1. This means that column of data source f4 is shifted right by one position then column of data source f1. For example if f1.column = A then f4.column = B

- **f2.ROW = f4.ROW**
  In the same manner, the query will process data-sources f2 and f4 and will combine only the cells, which have the same column coordinate.

- **f2.[ROW,D] = "Active"**
  This condition specifies that a cell from a row of data-source f2 can be retrieved, only if the column D of this specific line is equal with "Active".

These condition are introduced with the AND operator and, as a result, all of them should be fulfilled.

SELECT Clause

In this example, the query output strings and specific cells from the data-sources in a specific order. The output the column of the output are presented bellow:

- **"id" AS tid**
  The string "id" is the first column, which has the name "tid".

- **f1.[ROW,COLUMN] AS skill**
  The second column, which has the name "skill", contains all the cells, which the cell-by-cell path f1 indicates.

- **f2.[ROW,A] AS employee**
  The third column, which has the name "employee", contains the cells which are in column A inside the cell-by-cell path f2 indicates and the column D on their row is equal with "Active".

- **f3.[ROW,COLUMN] AS startdate**
  The forth column, which has the name "startdate", contains the cells which which the cell-by-cell path f3 indicates.

- **f4.[ROW,COLUMN] AS enddate**
  The fifth column, which has the name "enddate", contains the cells which which the cell-by-cell path f4 indicates.

5.3 CSVQL Implementation

As it mentioned above, CSVQL is an ETL framework, in a form of a query language, which allows the user to extract information from CSV (Comma-Separated Values) data sources. In order to apply CSVQL queries to real data sources, such as the Viggo's ones, this framework must be implemented. In this section, the implementation of CSVQL is described in terms of existence technology.

5.3.1 Concept idea

Since CSVQL is in a form of SQL query language, the most similar to it technology is an actual SQL query language of a DBMS. According to DB-Engines Platform, nowadays, the most popular non-commercial SQL DBMS is MySQL. Therefore, MySQL has been chosen as the query language, which implements the CSVQL, because of its popularity, which results a lot of available forums and libraries for support. More specifically, a CSVQL query will be translated into a MySQL one and then it will be applied to the data source that concerns. However, each data source must be uploaded first inside the MySQL database, before any query application. This is achieved with the use of a script, which imports the CSV data source as a MySQL table into the MySQL database, which implements CSVQL.

5.3.2 Implementation architecture

The implementation of CSVQL is a two components system. The main component of this system is a MySQL database, which stores the CSV data sources in a form of relational tables. Any CSVQL query regarding a data source, will be translated first into a MySQL one and then it will be applied to the relational table, which stores this specific data source.

The second component of this system is a script, which imports the CSV data sources into the MySQL database in a form of relational tables. Assuming that every data source is a .csv file, a relational table represents exactly one .csv file. The aforementioned script is implemented with the use of Java programming language and can be found in Appendix A with name "Data Load Script".

The architecture of the whole implementation is presented on Figure 5.19 :

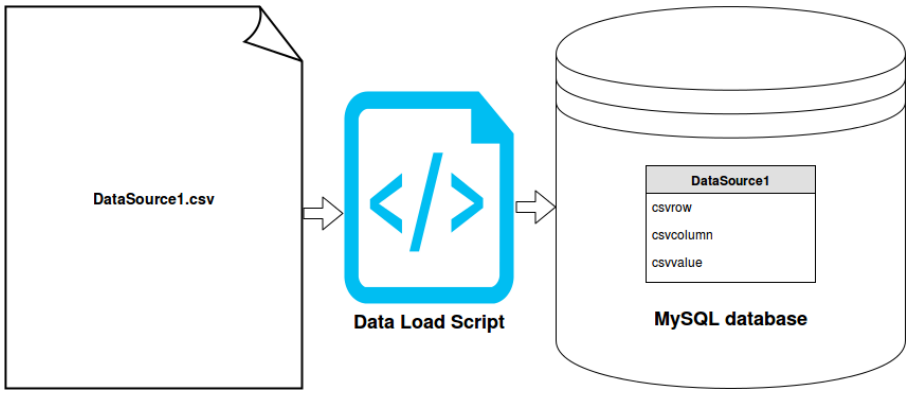


Figure 5.19: CSVQL implementation Achritecture

5.3.3 Data Load script

The role of "Data Load Script" is to convert a CSV grid into a MySQL table. More specifically, the input of the script is one or multiple CSV files. For each of the input files, the following procedure is performed.

1. Retrieves the input file line by line and imports its cells into an 2-Dimensional array. At the end, this 2-Dimensional array is identically same with the CSV grid, which is stored into the input file.
2. Connects to the MySQL database
3. Creates a table with the name of the input file or overwrites any existing table with the same name.
4. Imports the 2-Dimensional array, which has been created during the first step into the table with the input file's name. Each row of this table is in the following relational form: $(csvrow, csvcolumn, csvvalue)$. Every

   csvrow indicates the row of the stored cell, inside the input file

   csvcolumn indicates the column of the stored cell, inside the input file

   csvvalue indicates the value of the stored cell

5.3.4 Query MySQL mapping

Because of the fact that CSVQL has similar structure with MySQL, the transformation from the former query language to the latter one, is a trivial process. More specifically, the following transformations should take place:

FROM Clause

As it is presented in Section 5.2, with the use of CSVQL the query's data sources are represented as Figures 5.5, 5.6, 5.7 and 5.8 indicate. When a CSVQL query is translated into a MySQL one, then every query's data source is represented by a MySQL sub-query. This sub-query has the same data source and the same parameters that the CSVQL query has. Regarding, the coordinates of the data source, the "csvrow" and "csvcolumn" columns are used. The "WITH STEP" command is implemented with the "MOD" function of MySQL. Only in case of A,D set, there is a need of JOIN, otherwise the implementation in MySQL is not feasible. In practice, the contents in FROM clause of Query 5.14 will be transformed into

```
( SELECT *
  FROM training
  WHERE csvrow=6 AND csvcolumn>=5 AND csvcolumn MOD 2 = 1 ) f1 ,

( SELECT trainingA.csvrow AS csvrow, trainingA.csvcolumn AS csvcolumn, trainingA.
    cscvalue AS A, trainingD.cscvalue AS D
  FROM training AS trainingA
    INNER JOIN training AS trainingD
    ON trainingA.csvrow = trainingD.csvrow
  WHERE trainingA.csvrow>=8 AND trainingA.csvrow<=28 AND trainingA.csvcolumn = 1
      AND trainingd.csvcolumn = 4)) f2 ,

( SELECT *
  FROM training
  WHERE csvrow>=8 AND csvrow<=28 AND csvcolumn>=5 AND csvcolumn MOD 2 = 1 ) f3 ,

( SELECT *
  FROM training
  WHERE csvrow>=8 AND csvrow<=28 AND csvcolumn>=6 AND csvcolumn MOD 2 = 1 ) f4
```

WHERE Clause

As it is presented in Section 5.2, the statements, inside the WHERE clause are classified into two major categories. The former concerns the constraints regarding the values of the data (e.g. f2.[ROW,D] = "Active") and its translation is a straight forward process. The latter accomplishes the join of the FROM clause data sources (e.g. f2.ROW=f3.ROW). This kind of statements can be translated either straight forward either with the use of JOIN. In this example, the statements are translated straightforward. In practice, the contents in WHERE clause of Query 5.14 will be transformed into

```
f1.csvcolumn = f3.csvcolumn      AND
f2.csvrow=f3.csvrow              AND
f1.csvcolumn = f4.csvcolumn + 1 AND
f2.csvrow=f4.csvrow              AND
f2.D = ''Active''
```

SELECT Clause

In the same manner with the WHERE Clause,the contents in SELECT clause of Query 5.14 will be transformed into

```
"id"          AS tid ,
f1.csvvalue AS skill ,
f2.A          AS employee ,
f3.csvvalue AS startdate ,
f4.csvvalue AS enddate
```

5.4 Experimental study

The implementation of CSVQL must be efficient in terms of productivity and performance. The former requirement is fulfilled by the uses of MySQL syntax, which is the most popular non commercial language within SQL family. The latter must be examined. Within this section, the performance of CSVQL implementation will be examined. More specifically, an experiment has been implemented about the ability of the system to response quick to benchmark queries, which use data sources similar to Viggo's ones.

5.4.1 Scenario

The scenario of the experiment focus on scalability analysis of specific MySQL queries, which implement the CSVQL ones. According to this scenario, a set of benchmark queries, has been built, which perform a set of joins on CSV grid data sources. Join is the most expensive operation in a relational algebra implementation. The reason is the fact that it has a quadratic complexity $O(n^2)$ in both time and space. As a result, as the number of records increases, the join become also larger and, since the space complexity is also quadratic, large joins often either run out of memory either need to be stored on disk which makes processing even slower [21].

The arity of joins, inside the benchmark queries, starts from one (single join) until eight. The number eight is not a random number. Eight is the highest number of columns that a table of Viggo's data warehouse has. Therefore, assuming that at the worst case there will be needed a data source per attribute, the max number of joins will be eight.

5.4.2 Experiment set-up

Data sources

The data source of the experiment is a CSV grid of integer number, which has been imported into the MySQL database. Figure 5.20 illustrates this data source as a CSV grid. This grid has 1747 rows and 71 columns.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	759762	3243	61170	114249	821266	832323	902720	359303	784833	40336	512938	891384	177551	270637	286248	418295	303314	777370	537961	168677	448818	644844
2	866492	106493	510793	649092	721085	579026	908304	769317	409371	459039	481942	535321	65965	986516	431559	295651	266113	874028	764460	764084	499946	287055
3	849198	208064	700983	437875	788548	799580	696759	67946	140329	19687	986426	282716	976777	271739	581735	706600	559417	216260	263876	370383	655862	254298
4	270997	107247	658429	124014	701385	484129	6010	38843	332682	923420	508791	194604	865171	822490	628564	517276	539457	701785	682171	894896	127317	648862
5	402013	265852	228733	924468	462049	818248	260788	14299	166389	946101	491724	937541	269895	340094	530443	289405	922171	754841	444605	757308	793789	781354
6	199248	597072	342611	989563	561953	949223	467963	331574	172453	857983	933582	949885	674920	199553	234290	914443	971567	735190	913793	106389	374628	346763
7	595262	880956	872609	807336	854030	379034	631005	150803	856705	319560	707564	463669	662145	245345	36840	256231	11670	162142	782588	534220	137229	139085
8	645170	73489	447247	35047	454782	135003	786184	396402	334609	273895	420272	406429	977634	854327	209259	11646	982854	53114	161193	100844	574917	371820
9	646409	766671	862190	193891	557920	716076	743735	102977	685265	232783	493895	111447	799015	964391	51179	641739	11690	943550	191558	418366	458811	492631
10	62362	124270	737566	884241	787734	309877	837901	347110	621467	728179	575180	839808	939218	325379	45978	869304	220671	506473	820228	711835	864036	966353
11	987264	601242	762978	574233	416906	752894	85030	718882	165349	350516	99144	302563	961727	441521	287309	403966	396334	349931	447446	656424	239119	331346
12	846367	407744	813244	290703	235138	717952	410138	157912	157862	823420	934455	566749	872366	371369	730890	439647	301688	529293	672258	331269	541577	527455
13	435091	29444	469934	142142	771298	31031	479737	510648	936110	272956	575795	913957	630922	929459	29705	343728	342611	366148	945817	75778	689297	1283
14	976331	112356	749336	196507	724423	176138	46368	479463	385862	859913	511594	973613	616482	19549	399195	907451	576169	843628	147616	245506	639817	287747
15	668698	789706	926236	101770	650514	88348	357766	599397	116439	446046	225025	977574	797608	430485	326437	861370	351332	490012	220471	679383	620467	614382
16	906882	891257	579796	378162	442775	867244	54738	567416	698740	601707	590323	557270	417995	887631	570496	315580	339327	67061	625707	33308	209301	98009
17	200996	618146	383678	497141	942522	117397	956925	691581	320032	617176	20886	186536	601644	655371	440741	950368	761845	945287	427885	3760	931375	747856
18	185516	959079	118052	785010	321	491680	84171	806473	604565	302	730396	823950	487949	543680	65871	130386	989001	515996	581171	811399	26033	361773
19	493461	251919	461795	734792	473557	301567	99993	574602	166651	703148	584134	800473	774833	680984	311248	258618	47908	291184	992019	596497	703200	750134
20	884368	126322	842148	768156	649250	511769	344267	10844	210444	276447	263789	255617	280507	993873	809166	648402	574649	575738	690503	450784	64824	536174
21	899795	611815	677102	883697	541643	989148	3123	467620	702537	32898	295860	256640	923178	388548	25694	540422	637697	587193	641237	969532	550072	610977
22	285954	709102	831371	439310	677320	347004	225208	85683	308119	374896	993468	194125	211233	556769	325269	653299	538773	658164	163462	762775	154239	909606
23	335648	731125	228533	537655	318095	542363	377428	403895	439339	327344	668211	532483	276960	694102	935815	124315	356676	604273	940371	809019	429191	17462
24	287755	453765	46540	102955	332843	368156	790968	241855	929259	737137	936760	853697	185409	342184	88377	249683	121217	534322	613186	685507	37704	239598
25	323677	635268	91233	178503	675336	794534	546936	908358	5337	45776	401497	789492	121150	445836	611163	189392	734041	854981	373671	136291	962039	517580
26	212219	657473	583831	969887	341660	918598	541518	120218	285481	82566	665070	366593	656133	899222	66786	324226	775834	148585	5385	376693	304050	902743
27	282245	968081	367829	842408	902498	457957	245098	137568	971163	195390	885941	726105	798258	683186	567050	310410	230673	122105	503900	527524	410776	757372
28	324959	32094	431301	405302	100596	451549	600401	416632	755340	164492	49679	595849	97233	193508	302582	937391	619896	747365	262272	521651	575934	386628
29	117236	770855	717764	682572	743306	931052	195681	510648	830195	44153	783481	65239	752706	168163	664445	567350	259805	765075	948626	309070	213425	726820
30	806511	679629	654609	45299	322307	836831	478710	368889	639008	819131	633042	577784	751020	380208	383821	706433	556763	59483	564125	528405	609450	84920

Figure 5.20: Experiment's input as a CSV grid

With the use of "Data Load Script", this CSV is imported into the MySQL data as a relational table, which has the form that Figure 5.21 presents. This relational table has 124037 records.

This relational table has as Primary index the set of "csvrow" and "csvcolumn" columns. Moreover there are two Secondary indexes; one on "csvrow" column and one on "csvcolumn" column. According to the number of joins, the use of this data source is repeated.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	csvrow	csvcolumn	csvvalue															
2	1	1	759762															
3	1	2	3243															
4	1	3	61170															
5	1	4	114249															
6	1	5	821266															
7	1	6	832323															
8	1	7	902720															
9	1	8	359303															
10	1	9	784833															
11	1	10	40336															
12	1	11	512938															
13	1	12	891384															
14	1	13	177551															
15	1	14	270637															
16	1	15	286248															
17	1	16	418295															
18	1	17	303314															
19	1	18	777370															
20	1	19	537961															
21	1	20	168677															
22	1	21	448818															
23	1	22	644844															
24	1	23	213720															
25	1	24	411674															
26	1	25	844338															
27	1	26	516780															
28	1	27	223236															
29	1	28	480108															
30	1	29	750015															
31	1	30	571786															
32	1	31	883102															
33	1	32	323978															
34	1	33	988658															

Figure 5.21: Experiment's input after Data Load Script operation.

Benchmark queries

As it mentioned above, the set of the benchmark queries focuses on the on the performance analysis of CSVQL implementation. More specifically, there are 14 queries, which operate on the input data-set. These queries project all the columns of the output, which is the result of data sources' join. Every join operates on 124037 records per data source. There are two query families; one which joins on the primary key and one which joins on the Secondary key. It could be a third one, which joins on a column which is not an key of an index. However, this approach will never be the case in Viggo and as it mentioned in Sub-Section 5.4.1 it will be always $O(n^2)$ (where n: the number of the records), in terms of time at the worst case.

The 7 queries that join data sources on the primary key, output the data, which are at the same position inside these specific data sources, next to each other. For instance they output all the values of all the cells [1,A] of all the data sources next to each other. This approach can be used within Viggo in order to inspect versions of the same kind of CSV files, such as the ones, which have been introduced in Chapter 4.

The other 7 queries that join data sources on the secondary key, output the data, which are at the same row or at the same column inside these specific data sources, next to each other. For instance they output all the values of all the cells which are at the column A of all the data sources next to each other. Practically, this means that all the values, which are stored in column A such as employee's names, will be presented. This approach can be used within Viggo in order to inspect or combine data of the same or different kind of CSV files, such as the ones, which have been introduced in Chapter 4.

Some sample queries are presented by the Figures 5.22, 5.24, 5.23 and 5.25.

```
Select *
From CSVQL.Sample1 as F1
  Inner Join CSVQL.Sample2 as F2
    on F1.csvrow=F2.csvrow and F1.
       csvcolumn=F2.csvcolumn;
```

Figure 5.22: Benchmark query with one join on Primary key

```
Select *
From CSVQL.Sample1 as F1
  Inner Join CSVQL.Sample2 as F2
    on F1.csvrow=F2.csvrow and F1.
       csvcolumn=F2.csvcolumn;
```

Figure 5.23: Benchmark query with one join on Secondary key

```
Select *
From CSVQL.Sample1 as F1
  Inner Join CSVQL.Sample2 as F2
    on F1.csvrow=F2.csvrow and F1.
       csvcolumn=F2.csvcolumn
  Inner Join CSVQL.Sample3 as F3
    on F1.csvrow=F3.csvrow and F1.
       csvcolumn=F3.csvcolumn
  Inner Join CSVQL.Sample2 as F4
    on F1.csvrow=F4.csvrow and F1.
       csvcolumn=F4.csvcolumn
  Inner Join CSVQL.Sample3 as F5
    on F1.csvrow=F5.csvrow and F1.
       csvcolumn=F5.csvcolumn
  Inner Join CSVQL.Sample2 as F6
    on F1.csvrow=F6.csvrow and F1.
       csvcolumn=F6.csvcolumn
  Inner Join CSVQL.Sample3 as F7
    on F1.csvrow=F7.csvrow and F1.
       csvcolumn=F7.csvcolumn
  Inner Join CSVQL.Sample1 as F8
    on F1.csvrow=F8.csvrow and F1.
       csvcolumn=F8.csvcolumn
  Inner Join CSVQL.Sample1 as F9
    on F1.csvrow=F9.csvrow and F1.
       csvcolumn=F9.csvcolumn;
```

Figure 5.24: Benchmark query with eight joins on Primary key

```
Select *
From CSVQL.Sample1 as F1
  Inner Join CSVQL.Sample2 as F2
    on F1.csvrow=F2.csvrow and F1.
       csvcolumn=F2.csvcolumn
  Inner Join CSVQL.Sample3 as F3
    on F1.csvrow=F3.csvrow and F1.
       csvcolumn=F3.csvcolumn
  Inner Join CSVQL.Sample2 as F4
    on F1.csvrow=F4.csvrow and F1.
       csvcolumn=F4.csvcolumn
  Inner Join CSVQL.Sample3 as F5
    on F1.csvrow=F5.csvrow and F1.
       csvcolumn=F5.csvcolumn
  Inner Join CSVQL.Sample2 as F6
    on F1.csvrow=F6.csvrow and F1.
       csvcolumn=F6.csvcolumn
  Inner Join CSVQL.Sample3 as F7
    on F1.csvrow=F7.csvrow and F1.
       csvcolumn=F7.csvcolumn
  Inner Join CSVQL.Sample1 as F8
    on F1.csvrow=F8.csvrow and F1.
       csvcolumn=F8.csvcolumn
  Inner Join CSVQL.Sample1 as F9
    on F1.csvrow=F9.csvrow and F1.
       csvcolumn=F9.csvcolumn;
```

Figure 5.25: Benchmark query with eight joins on Secondary key

5.4.3 Implementation

The SQL data store have been implemented by the MySQL DBMS for Linux. The system's specifications are the following:

OS Linux 4.4.0-28-generic, Ubuntu 16.04 LTS

CPU 4x Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz (862.589MHz)

RAM 11.72GB

SQL DBMS MySQL 5.7.12 and Workbench 6.3 have been installed.

The experiment has 14 phases. Every phase refers to one benchmark query and consists of ten executions of this specific query. All the experiments are executed via Workbench platform. Something that must be mentioned is that Workbench calculates the duration of each query as a summary of *Query duration + Fetching time* . Fetching time measures how long transferring fetched results take, which has nothing to do with query execution. As a result, this parameter cannot be considered as an SQL query metric, since fetching time depends on network connection.

5.4.4 Experiment results

In this section the results of the experiment are presented. The full tables with all the results can be found in Appendix B with name "CSVQL Experiment Results"; one table regarding the eight join queries on primary key and one table about the eight join queries on secondary key. The analysis of the results is split into two categories; one for the queries, which join data sources on primary key and one for the queries, which join data sources on secondary key.

Join on primary key

As it can be seen in Figures 5.2 and 5.27, for one until seven joins on table's primary key, the average time of a CSVQL benchmark query via the aforementioned implementation is no more than 0.0027 seconds. For eight joins the average time increases to 0.0037 seconds. Moreover Figure 5.26 indicates that the worst time among this category of queries has been recorded during the eight joins benchmark query and it is 0.0059 seconds. For the rest of the queries the worst time is not higher than 0.0034 seconds. On the other hand, all the queries have best execution time around 0.002 seconds. Practically, this means that when a user wants to compare or present data which are at the same position in a CSV grid, she doesn't need more than 40 milliseconds, or in other words 8 honey bee's wing flaps according to Michael S. Engel in "The taxonomy of recent and fossil honey bees (Hymenoptera: Apidae: Apis)" [20].

Table 5.2: Results of CSVQL scalability experiment with Primary index

	One Join	Two Joins	Three Joins	Four Joins	Five Joins	Six Joins	Seven Joins	Eight Joins
Average time	0.0027	0.002	0.0021	0.0022	0.0022	0.0021	0.0023	0.0037
Worst time	0.004	0.0024	0.003	0.0029	0.0026	0.0028	0.0034	0.0059
Best time	0.0021	0.0018	0.002	0.0019	0.002	0.0019	0.002	0.0021

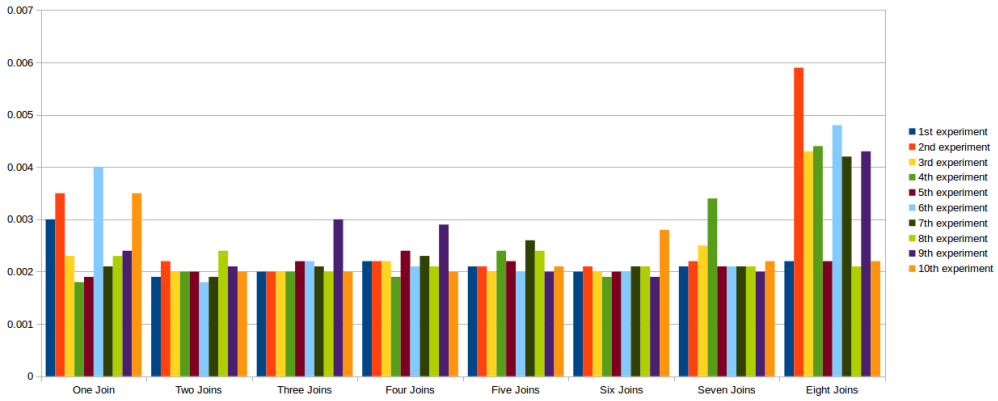


Figure 5.26: CSVQL performance for joins on primary key per experiment

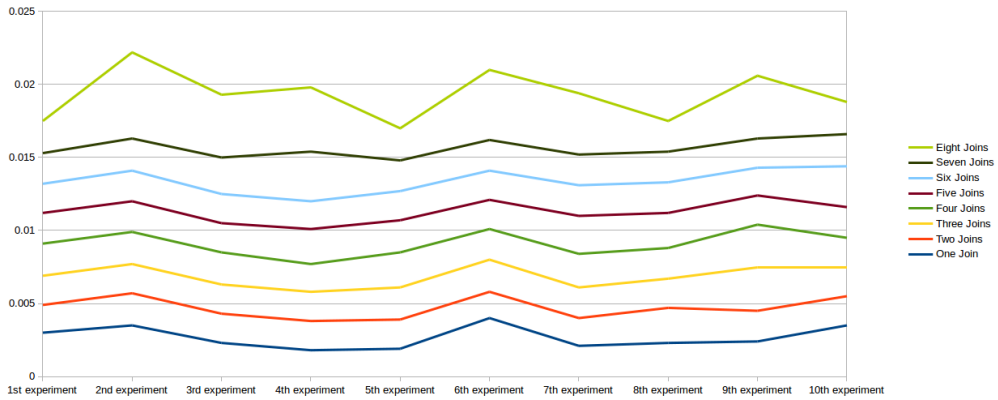


Figure 5.27: CSVQL performance for joins on primary key per join

Join on secondary key

As it can be seen in Figures 5.3 and 5.29, for one until four joins on table's secondary key, the average time of a CSVQL benchmark query via the aforementioned implementation is no more than 0.0025 seconds. However, after the fifth join the time grows exponentially. More specifically, for five joins the average time is 0.0065 seconds, for six joins 0.023 seconds, for seven joins 0.014 and for eight joins 1.06 seconds (460 times more than the average time of one join's average time). Something that must be mentioned is the fact that in every experiment, the biggest difference between the performance results are at most 60 milliseconds, which means that the behavior of CSVQL in terms of expected performance time is constant and reliable. In Figure 5.29, the difference between the benchmark queries can be inspected. This view emphasizes on the comparison between the benchmark queries performance. After a closer inspection, it can be remarked that even the worst performance time which is 1087 milliseconds is less than 3 blinks of an eye according to Krishna, G.V.Siva, and K. Amarnath in "Anovel approch of eye tracking and blink detection with a human machine" [14].

Table 5.3: Results of CSVQL scalability experiment with Primary index

	One Join	Two Joins	Three Joins	Four Joins	Five Joins	Six Joins	Seven Joins	Eight Joins
Average time	0.0023	0.0028	0.0017	0.0025	0.0065	0.023	0.14	1.06
Worst time	0.0035	0.0062	0.0024	0.0044	0.011	0.0042	0.172	1.087
Best time	0.0015	0.0017	0.00077	0.001	0.0025	0.0014	0.108	1.028

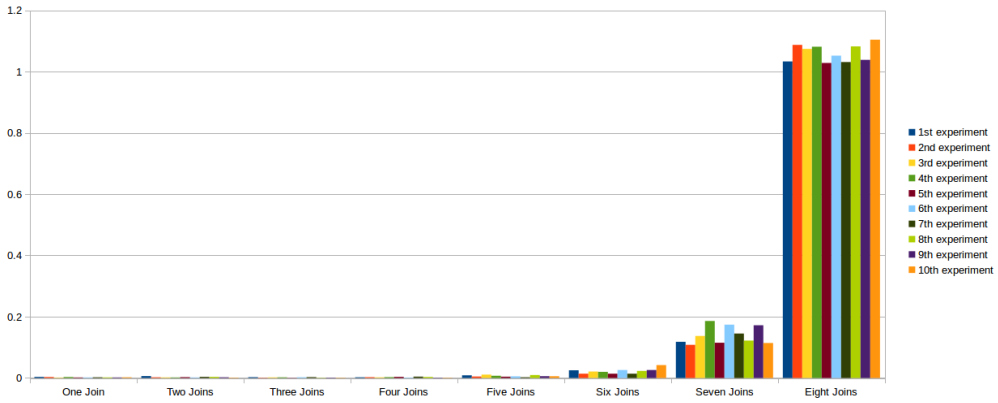


Figure 5.28: CSVQL performance for joins on secondary key per experiment

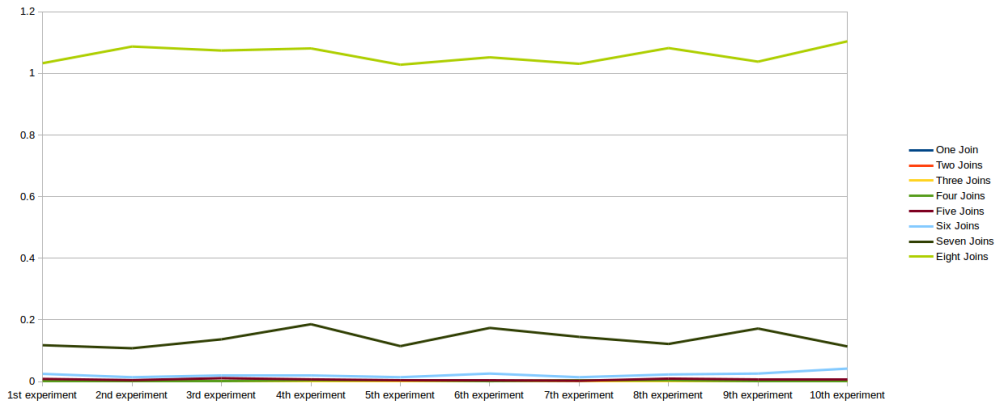


Figure 5.29: CSVQL performance for joins on secondary key per join

Chapter 6

Data warehouse tier

This chapter introduces the data-warehouse, which stores the all the data, regarding the employees of Viggo Airport Eindhoven BV. This specific data-warehouse is named Viggo data-warehouse and is a SQL database, which follows the basic relational database concepts as concerns design and implementation. By the theory, the basic relational database concepts are split into three major levels - the conceptual, the logical, and the physical level - and, as a result, this data-warehouse is described according to these specific levels. More specifically, Section 6.1 clarifies the assumptions, which have been made regarding the data-warehouse design, and introduces the framework in which this data-warehouse is built. Section 6.2 introduces the conceptual design level, which produces the initial model of the data-warehouse, by focusing especially on entity-relationship model. Section 6.3 elaborates the logical design level, which develops a relational implementation of the conceptual level's output, by introducing the mapping process, which translates an entity-relationship model to relational schema. Section 6.4 presents the physical design level, which optimizes the logical level's product with respect to a particular database technology or implementation platform.

6.1 Design assumptions

The implemented data-warehouse follows the rules of a traditional OLAP database design. The traditional OLAP database design suggests a data model based on the fact and the dimension tables. More specifically, the data are organized into tables and each table is either a fact either a dimension table. The whole data model is built in a way, which should support heavy queries in terms of time execution. This kind of queries usually implies aggregation and, as a result, the processing time involves the traversing of all the records.

According to the above analysis, the data-warehouse, which has been created, fulfills the following obligations:

Fact tables The fact tables focus on the following aspects:

1. **Shifts**, in terms of duration and time windows.
2. **Payment**, in terms of salary.
3. **Contract**, in terms of duration and type of agreement.
4. **Skills training**, in terms of duration and type of specialization.

Dimensions There will be three dimensions, with hierarchy "*child with exactly one parent*" and they focus on the following aspects:

1. **Time**
2. **Human resource**
3. **Skills training**

6.1.1 Dimensions

There are three dimensions in which the cubes of Viggo data-warehouse are organized:

Calendar

This dimension refers to the time and its hierarchy is organized with a leveling, which is presented in Figure 6.1.

Figure 6.1: The hierarchy of calendar dimension.

Human resource

This dimension refers to the employees and its hierarchy is organized with a leveling, which is presented in Figure 6.2.

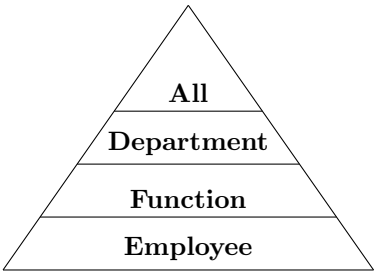


Figure 6.2: The hierarchy of human resources dimension.

Skill training

This dimension refers to the skills that Viggo employees possess and its hierarchy is organized with a leveling, which is presented in Figure 6.3.

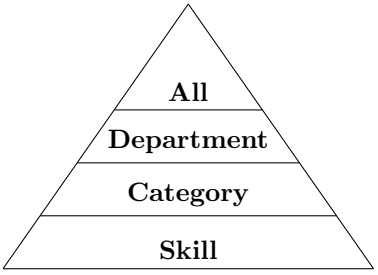


Figure 6.3: The hierarchy of skills training dimension.

6.1.2 Measures

There are two kind of measures that Viggo data-warehouse focuses on:

Additive measures The attributes, which can be meaningfully summarized along all the dimensions are the following:

- Employee's salary
- Shift's duration

Nonadditive measures The attributes, which cannot be meaningfully summarized across any dimension are the following:

- Contract's duration
- Contract's scale
- Training's duration

6.2 Conceptual level

The design of a database at the conceptual level aims to create a user-oriented illustration of the database. This user-oriented representation behaves in a high level of abstraction, since it does not take any implementation requirements into consideration. This level of abstraction is achieved by using a conceptual model, which can identify the relevant concepts of the database system at hand. In this work, the conceptual model, which is used is the entity-relationship model, one of the most popular conceptual models for database design [6]. The role of an entity-relationship model is to demonstrate the conceptual view of Viggo data-warehouse. More specifically, it specifies the real-world entities, which participate in, and the relationships among them. The basic concepts of an entity-relationship model are presented in Section 6.2.1.

6.2.1 Basic concepts

The building blocks of the entity-relationship model are the following:

Entity

As an entity, any animate or inanimate real-world object can be defined. For instance, in a airport database, passengers, aircrafts and scheduled flights can be considered as entities. All these entities have some attributes, which define their identity. In a the same manner, a set of entities is a collection of the same type of entities. More specifically, a set of entities contains homogeneous entities, with same attributes and similar values. For instance, a set of passengers contains all the passengers of an airport and flights contains all the flights that operate in this specific airport.

Attributes and Keys

As it mentioned above, entities are characterized by their properties, which are called attributes and these attributes have values. For instance, a passenger entity may have a name, a nationality, and an age as attributes. Within a set of entities, every entity must be unique and be uniquely identified. For this purpose, one or more attributes consist a key value, which uniquely identifies a specific entity among its entity set. For instance, an attribute, which can identify a passenger, within the set of passengers, is the ID-number of her passport, which is unique world-wide.

Relationship

Two or more entities can be associated with each other and their association is named relationship. For instance, a passenger "takes" a flight, an aircraft "performs" a flight. In this case, "takes" and "performs" are called relationships. In the same manner, a set of relationships is a collection of the same type of relationships. Likewise entities, a relationship can also have attributes. These specific attributes are called descriptive attributes.

Mapping Cardinalities

The term cardinality defines the arity of entities in an entity set, which can be related with a number of entities of another entity set via a relationship set.

One-to-one One entity of an entity set A can be affiliated with at most one entity of entity set B and vice versa.

One-to-many One entity of an entity set A can be affiliated with more than one entities of entity set B. On the other hand, an entity of entity set B, can be affiliated only with at most one entity of entity set A.

Many-to-many One entity of an entity set A can be affiliated with more than one entity of entity set B and vice versa.

6.2.2 ER diagram

The entity-relationship diagram and its concepts are presented below:

Diagram

Figure 6.4 represents the entity-relationship diagram of Viggo data-warehouse.

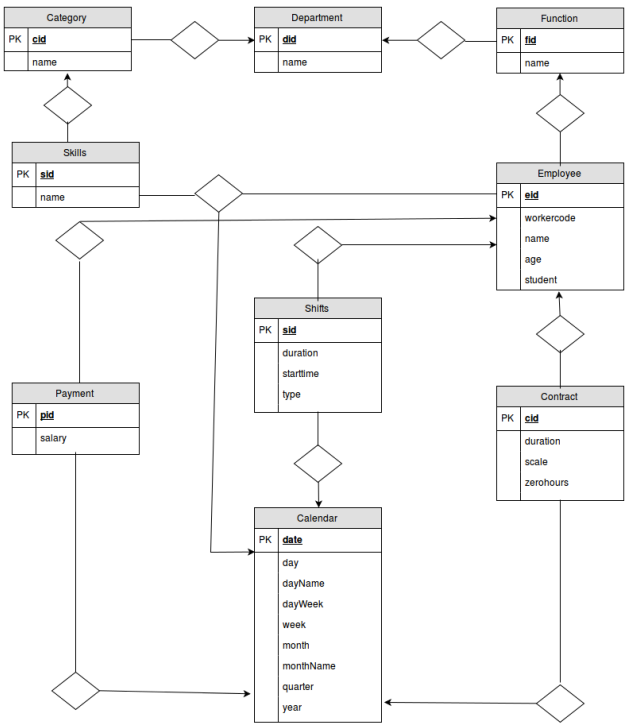


Figure 6.4: The entity-relationship diagram of Viggo data-warehouse.

Explanation

The Figure 6.4 illustrates the entity-relationship model of Viggo data-warehouse model. The aforementioned basic concepts are represented via the following visualization:

Entity Entities (or entity sets) are symbolized by rectangles. These rectangles are in the shape of a table with one column and two rows. The former row indicates the name of the entity and the latter, the attributes of this specific entity.

Relationship Relationships (or relationship sets) are symbolized by diamond-shaped boxes, which induce the name of this specific relationship, inside. The entities (rectangles) that participate in this relationship are attached to diamond-shaped box of this specific relationship by lines. The nature of these lines varies and is explained below.

Relationship and Cardinality Relationships between entities have two potential types; Binary or Multiple. Binary relationship is a relationship with only two entities participating. Multiple relationship is a relationship with three or more entities participating. Regarding binary relationships, the most important factor is the cardinality. More specifically, according to a binary relationship, cardinality indicates the arity of an entity that can be associated with the related to it entity, via this specific relationship.

One-to-one In case when at most one instance of an entity is affiliated with its associated entity via a binary relationship relationship, then it is marked Arrow Diamond Arrow.

One-to-many When more than one instance of an entity is associated with a relationship, it is marked Arrow Diamond Line

Many-to-many The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. Line Diamond Line

6.2.3 Entities

More specifically, inside the diagram of Figure 6.4 there are 9 set of entities, which are presented bellow. The underline attributes consist of the primary key of the entity.

Department Indicates the departments of Viggo Eindhoven Airport. Department is a Level table.
(e.g. Platform Viggo Eindhoven Airport, Cleaning Viggo Eindhoven Airport)
Attributes:

did : Primary key

name : Department's name (e.g "Platform Viggo Eindhoven Airport")

Employee Indicates the employees of Viggo Eindhoven Airport. Employee is a Level table.
(e.g. John Smith, John Doe)
Attributes:

eid : Primary key

workercode : Working code of Viggo employees (e.g 0001VC)

name : Employee's name (e.g "John Smith")

age : Employee's age (e.g 18)

student : Employee's student state (e.g "Nee")

Function Indicates the functions that the employees of Viggo Eindhoven Airport possess. Function is a Level table.
(e.g. VC Assistant Cleaning Supervisor, EA Platform Employee 2)
Attributes:

fid : Primary key

name : Function's name (e.g "EA Platform Employee 2")

Skill Indicates the skills that the employees of Viggo Eindhoven Airport are trained for. Skill is a Level table.
(e.g. Towbar Pushback, Bagage tracing procedures course FR)
Attributes:

sid : Primary key

name : Skill's name (e.g "Towbar Pushback")

Category Indicates the categories that the skills of set of entities "Skill" belong to. Category is a Level table.
(e.g. Platform Employee 1, Passage Security)
Attributes:

cid : Primary key

name : Category's name (e.g "Platform Employee 1")

Calendar Indicates the calendar dates from 1/1/2010 to 31/12/2020. Calendar is a Level table.
(e.g. 1/1/2010, 3/1/2010)
Attributes:

date : Primary key - the date in DD/MM/YYYY format (e.g 1/1/2010)

day : Day's number in year (e.g 365)

dayName : Day's name (e.g "Monday")

dayWeek : Day's number in week (e.g 7)

week : Week's number in year (e.g 52)

month : Month's number in year (e.g 12)

monthName : Month's name (e.g "January")

quarter : Quarter's number in year (e.g 4)

year : Year's number (e.g 2010)

Contract Indicates the contracts that employees of Viggo Eindhoven Airport have signed. Contract is a Fact table.
(e.g. Contract with Cleaning Viggo Eindhoven Airport department, contract Platform Viggo Eindhoven Airport department)
Attributes:

cid : Primary key

duration : Contract's duration in days till expire date (e.g 365)

scale : Contract's salary scale (e.g 3)

zerohours : Contract's type (boolean) (e.g True)

order : Contract's order (e.g "2^{nd} contract")

Payment Indicates the payment that employees of Viggo Eindhoven Airport receive. Payment is a Fact table.
(e.g. Payment of John Smith on 29^{th} July 2010, Payment of John Smith on 29^{th} June 2010)
Attributes:

pid : Primary key

salary : Payment's amount (e.g 2010€)

Shifts Indicates the shifts that the employees of Viggo Eindhoven Airport worked or planned to work. Shifts is a Fact table.
(e.g. 12:00-20:00, 8:30-16:30)
Attributes:

sid : Primary key

duration : Shift's duration in minutes (e.g 60)

starttime : Shift's start time as timestamp (e.g 15:30)

type : Shift's type (e.g "Planned")

6.2.4 Relationships

In addition, there are 12 set of relationships between these 9 entities:

Function - Department The set of entities "Function" is related to the set of entities "Department" with the following cardinality:

- A function belongs exactly to one department.
- A department has at least one or more functions.

Employee - Function The set of entities "Employee" is related to set of entities "Function" with the following cardinality:

- An employee possesses exactly one function.
- A function can be assigned to zero or more employees.

Category - Department The set of entities "Category" is related to set of entities "Department" with the following cardinality:

- A category of skills belongs exactly to one department.
- A department can be have zero or more categories of skills.

Skill - Category The set of entities "Skill" is related to set of entities "Category" with the following cardinality:

- A skill belongs exactly to one category of skills.
- A category of skills has at least one or more skills.

Category - Function The set of entities "Category" is related to set of entities "Function" with the following cardinality:

- A category of skills supports zero or more functions.
- A function requires zero or more categories of skills.

Payment - Calendar The set of entities "Payment" is related to set of entities "Calendar" with the following cardinality:

- A payment took place on exactly one calendar date.
- Om a calendar date, zero or more payments can be done.

Payment - Employee The set of entities "Payment" is related to set of entities "Employee" with the following cardinality:

- A payment refers exactly to one employee.
- An employee can have already received zero or more payments.

Shifts - Calendar The set of entities "Shifts" is related to set of entities "Calendar" with the following cardinality:

- A working shift of hours takes place on exactly one calendar date.
- On a calendar date, zero or more working shifts can take place.

Shifts - Employee The set of entities "Shifts" is related to set of entities "Employee" with the following cardinality:

- A working shift refers exactly to one employee.
- An employee can have zero or more working shifts.

Contract - Calendar The set of entities "Contract" is related to set of entities "Calendar" with the following cardinality:

- A contract has exactly one calendar date of start.
- On a calendar date, zero or more contracts can start.

Contract - Employee - Department The sets of entities "Contract", "Employee" and "Department" are related to each other with the following cardinality:

- A contract is singed by exactly one employee.
- A contract is singed by exactly one department.
- An employee has at least one or more contracts signed with a department.
- A department can have zero or more contracts with an employee.
- A contract is valid, until an exactly one specific expire date.

Employee - Skill - Calendar The sets of entities "Employee", "Skills" and "Calendar" are related to each other with the following cardinality:

- An employee can have zero or more skills.
- A skill can belong to zero or more employees.
- A skill belongs to an employee, until an exactly one specific expire date.

6.3 Logical level

Conceptual models are beneficial, in terms of database design, since they present the basic concepts of the designed data model, in a simplified manner. However, conceptual models must be transformed into logical ones, in order to implement a data model view, which is closer to the database schema. This process takes place at the logical level. At logical level, database design focus on translating the conceptual model (entity-relational model in this case) into an appropriate logical model, which can be implemented by a database management system. The most popular logical model, at present, is the relational model, which is introduced by Edgar F. Codd in 1970 [6] [12]. More specifically, in the relational model of a database, the information is organized and represented in form of relations (tables), and every relation is in form of tuples (rows with columns). In order to guarantee a proper logical representation, a set of mapping rules will be declared. More specifically, by applying these specific rules to a conceptual model, an accurate logical model, which represents all the aspects that the translated conceptual model indicates, will be created.

6.3.1 Mapping rules

The rules by which, the conceptual model is translated into a logical one, are the following:

Rule 1 Each entity is translated into a table, with columns all the attributes of this specific entity.

Rule 2 A relationship between two entities (binary relationship) is translated by the following rules, depending on its cardinality:

- **Rule 2a** If the relationship is one-to-one, the table corresponding to the fact or to the child level is extended with all the attributes of the dimension level or the parent level, respectively.
- **Rule 2b** If the relationship is one-to-many, the table corresponding to the fact or to the child level is extended with the primary key of the table corresponding to the dimension level or to the parent level, respectively.
- **Rule 2c** If the relationship is many-to-many, then this relationship is translated into a table, with columns all the attributes of the relationship plus the primary keys of the participants to this relationship tables.

Rule 3 A relationship between three or more entities (multiple relationship) is translated by splitting this specific relationship into binary ones and then by applying the rules 1 and 2.

6.3.2 ER diagram mapped to relational schema

By applying the mapping rules of Section 6.3.1, the following relational model is obtained:

Department (did, name)

Function (fid, name, *Department.did*)

Employee (eid, workercode, name, age, student, *Function.fid*)

Category (cid, name, *Department.did*)

Skill (sid, name, *Category.cid*)

Calendar (date, day, dayName, dayWeek, week, month, monthName, quarter, year)

Contract (cid, duration, scale, zerohours, order, *Calendar.date*, *Department.did*, *Employee.eid*)

Payment (pid, salary, *Employee.eid*, *Calendar.date*)

Shift (sid, duration, starttime, type, *Employee.eid*, *Calendar.date*)

Requirements (*Function.fid*, *Category.cid*)

Training (*Skills.sid*, *Employee.eid*, *Calendar.date*)

6.4 Physical level

As it has been mentioned in the introduction of this chapter, the database design specifies the physical storage of the database and all the aspects, which secure a certain level of performance of this specific database application [6]. Within this report, the physical design will focus on the application, which builds the database and how database is filled. The reason is the fact that, the database, which is illustrated, is not an OLTP database but an OLAP and, as a result, there is no need for analysis in terms of query and transaction processing.

The DBMS which is chosen to store the Viggo data is an instance of **MySQL 5.7.12**, with the following system characteristics:

OS Linux 4.4.0-28-generic, Ubuntu 16.04 LTS

CPU 4x Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz (862.589MHz)

RAM 11.72GB

The platform, which is used to implement and manage the DBMS is the Workbench 6.3. More specifically, MySQL Workbench 6.3(Version 6.3.6 build 511 CE, 64 bits) Community edition has been installed.

The final form of the Viggo's data warehouse is presented in Figure

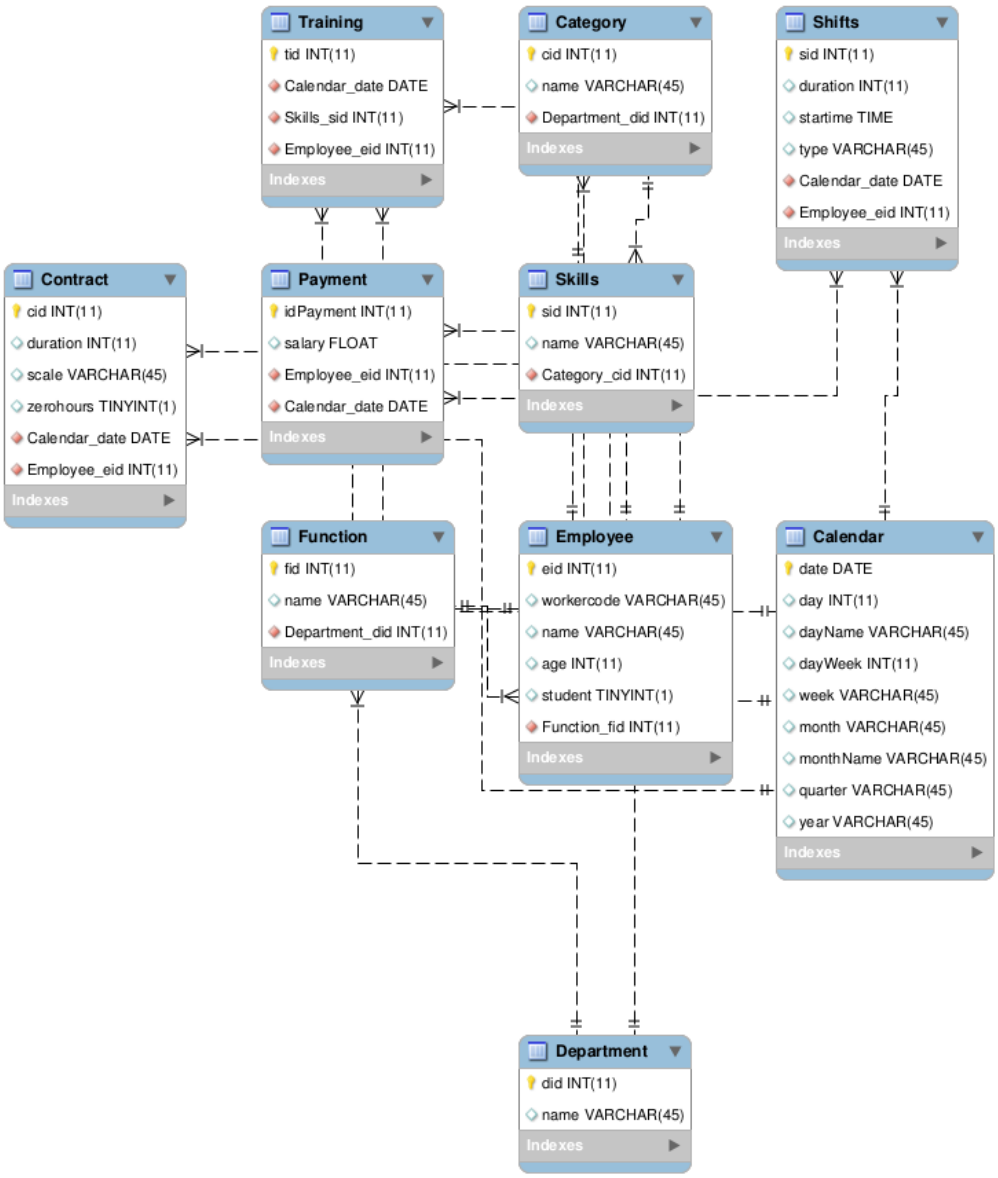


Figure 6.5: The entity-relationship diagram of Viggo data-warehouse after reverse engineering process on physical level.

Chapter 7

OLAP & Front-end tier

This chapter illustrates the OLAP and the Front-end tiers of Viggo data-warehouse. The OLAP tier is consisted of an OLAP server, which builds a multidimensional view of the data inside the data warehouse. On the other hand, the Front-end tier consists of the visualization end of the OLAP tier and offers tools that can help the user to project the contents of the data warehouse. Within this project, the implementation of the OLAP and the Front-end tiers is achieved with the use of icCube 5.2. Thus, in this chapter, an attempt has been made to introduce icCube server. More specifically, in Section 7.1, the basic characteristics of icCube server are presented. In Section 7.1.1, the communication of icCube server with the data warehouse tier is elaborated and Section 7.1.2 explains the manner in which the OLAP cubes are built inside icCube server. Section 7.1.3 is a gentle introduction to MDX query language and its main concepts. Regarding front-end tier, Section 7.2 illustrates icCube's web reporting tools and Sub-Section **??** presents some outputs, which prove the concept idea of Viggo's data warehouse according to the problem statement.

7.1 IcCube server

The platform icCube is an engine for data analysis and data visualization purposes. This engine is characterized by high performance and, as it mentioned above, is in a form of a multidimensional OLAP server. The most common use case of icCube lies in the field of business intelligence. More specifically, icCube can be used, as an analysis tool, to get insights from data that are stored across multiple sources, such as databases and spreadsheets. In addition to the OLAP server, icCube features an additional Web Reporting server, which operates over the OLAP and implements the front-end tier of the data warehouse, by providing interactive charts and dashboards [2].

7.1.1 Communication with Data warehouse tier

The icCube server provides multiple options to implement the OLAP tier. No matter the approach, the implementation consists of selecting one or multiple data sources, such as databases or flat files. Because of the fact that relational databases are the most popular data sources in icCube, the platform has no limitation about the number of database sources that can be used for the creation of this OLAP tier [2]. This means that data from multiple databases can be combined to create a cube.

Within this project only one DBMS is used, so there is no use of this feature. Regarding the supported technology, icCube supports any database with a jdbc driver. As a result, DBMS's such as Oracle, SQL Server, Postgres, Sybase ASE and MySQL are supported. Apart from that, the output of SQL queries on these specific databases can be used as a data source. This feature of icCube can be used in combination with the multiple databases use, in the case of Youforce data source, if this product is supported with a jdbc driver in the future.

7.1.2 Cube builder

As it mentioned in Chapter 2, the core of the OLAP tier is the cube. In relation with this, icCube server can considered as a container of cubes. These cubes can be created with the use of the "Cube Builder". This module provides assistance during the creation of the OLAP schema via a modern graphical interface. More specifically, via "Cube Builder", the user can define all the necessary needed components of a schema, such as the data sources, the dimensions, the hierarchies and the calculated members. Among these components, the most important are the dimensions and the facts [2].

7.1.3 Cube querying

This section describes the cube querying process with the use of icCube server. More specifically, for cube querying there is a query language for Multidimensional Expressions (MDX) and it is in form of SQL. This specific language supports also calculations in a similar manner as spreadsheet formulas. The following paragraphs consist a gentle introduction regarding MDX and its main concepts [2].

Overview

The name "MDX" stands for "Multi-Dimensional Expressions" and is the standard querying language for querying OLAP servers, defined by Microsoft. The syntax of the MDX queries look similar to SQL. However, MDX is a totally different language. SQL has been designed in order to query relational data structures, where information is organized into rows and columns. On the other hand, in OLAP structures, information is organized into measures, dimensions, hierarchies, and levels. As a result, there is a need of a different approach. MDX is a query language, which can operate over OLAP structures and it includes a broad set of functions for statistical analysis. In contrast with SQL, MDX cannot manipulate information but only can read and analyze data [2].

Concepts

Because of the fact that OLAP servers usually import their data from relational databases, SQL can be used sometimes to describe MDX functionality. As a table and its columns are the building blocks of SQL queries, in the same manner, dimensions, hierarchies, and levels, are the main concepts in MDX [2].

Let's consider the following example: Within Viggo, there is a need of some charts for the HR and the financial departments and the information we have is as follows:

- John Smith received 5000 euros in January 2016 as a EA Platform Employee.
- John Smith received 4000 euros in February 2016 as a EA Platform Employee.
- John Doe received 3000 euros in January 2016 as a VC Assistant Cleaning Supervisor.
- John Doe received 4000 euros in February 2016 as a VC Assistant Cleaning Supervisor.

With this example, the concepts of 'Payment', 'Employee', 'Function' and 'Time' are introduced. Within an OLTP database, this model can designed with one table per concept and then the foreign keys can be related into another tables. In MDX 'Employee', 'Function' and 'Time' are modeled as dimensions and the 'Payment' as a measure. It must highlighted that measures in MDX are different than dimensions and they have numeric values. The concrete value of a dimension (e.g. "EA Platform Employee") is a member of this specific dimension and it is similar to the value of an SQL table. Thus, an OLAP cube can be considered as a collection of dimensions, which index a list of measures. In this example there are three dimensions ('Employee', 'Function' and 'Time') and one numeric measurable quantity ('Payment'). It obvious that in real life business problems in Viggo there will be many more dimensions and measures.

To elaborate the upper information an MDX query is presented in Figure 7.1, which results a table with the amount of payments per function and per month. In order to compute some statistics with this data, the information is enriched with additional details. Precisely, the functions are organized into a graph (Department, Role, Function) the years into semesters and months (Year, Semester, Month).

```
SELECT
        ([Department].[Role].[Function].members) ON 0,
        ([Year].[Semester].[Month].members ) ON 1
FROM
        [Payment]
```

Figure 7.1: MDX query example

In this manner, the cubes regarding HR and BA processes are retrieved, in order to be presented via icCube reporting tools.

7.2 Web reporting tools

The front-end tier of Viggo's Data warehouse has been implemented also with the use of icCube. More specifically, icCube provides a web reporting tool, which is allows the user to create graphs and charts, regarding the cubes of OLAP server. This reporting tool is in a form of a web application and therefore is physically accessible via a URL of OLAP server's domain. Inside this web application, there are lot of graphs, which can visualize the data of the OLAP server. Within this work, the charts, which are used are implemented with the use of D3.js library. This JavaScript library makes available the use of CSS3, HTML, and/or SVG, in order to manipulate the input data-set in terms of visualization. In addition, the represented elements of the data-set can be interactive through the use of D3.js data-driven transformations and transitions. D3.js is written in JavaScript and uses a functional style, which means that the code can be reused in the future with specific functions to heart's content. In the following sub-sections the charts that are used in the front-end tier of Viggo data warehouse are presented. Something that must be mentioned is the fact that the numbers on the charts are fictional and do not represent real facts. The overview of icCube web reporting tool is presented by the Figure 7.2.

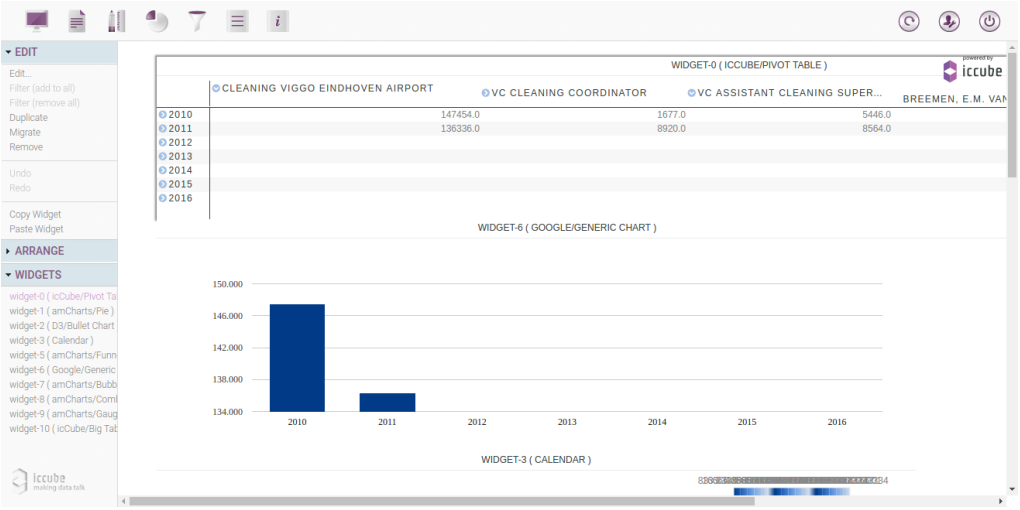


Figure 7.2: Web reporting tool overview.

7.2.1 Bar chart

Bar charts are one of the most common ways to visualize data. Their advantage is the fact that they are quick in terms of information comparison and revealing highs and lows at a glance. In addition, bar charts are especially effective, when there are numerical data, which are split nicely into different clusters and, as a result, trends about the data can be easily identified.

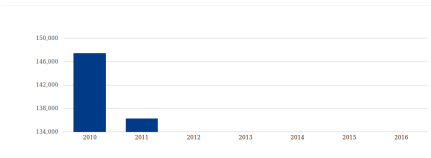


Figure 7.3: Bar chart regarding payment in Viggo Cleaning department.

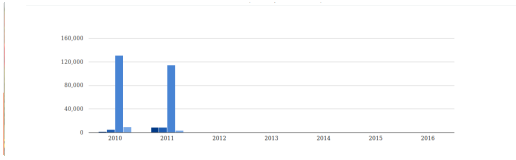


Figure 7.4: Bar chart regarding payment in Viggo Cleaning department, per function.

Advantages:

- Comparing data across clusters. Examples: Amount of employees per function, amount of payments per department, contract renewals per year.
- Multiple bar charts on a dashboard help the viewer to quickly compare related information instead of flipping through a bunch of spreadsheets or slides to answer a question.

7.2.2 Line chart

Line charts are one of the most frequently used chart type. Line charts connect individual numeric 2D data points. The view is a simple, straightforward way to visualize a sequence of values as a trend, over a period of time.

Advantages:

- Viewing trends in data over time. Examples: expenses for employees over a year period, contract renewals during a five years period, revenue growth by quarter.

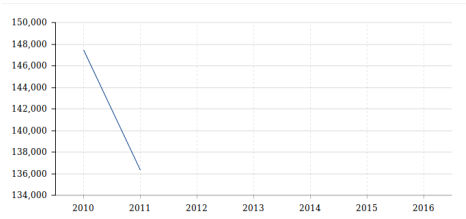


Figure 7.5: Line chart regarding payment in Viggo Cleaning department.

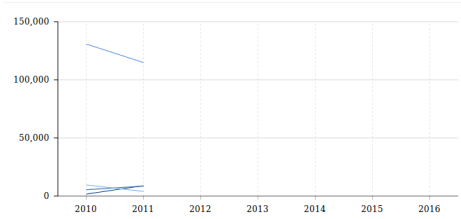


Figure 7.6: Line chart regarding payment in Viggo Cleaning department, per function.

7.2.3 Bullet chart

Bullet charts are ideal for progress tracking according to a set goal. At its heart, a bullet graph is a variation of a bar chart and it has been It was designed to replace dashboard gauges, meters and thermometers, because these means of visualization do not display sufficient information.

Advantages:

- Evaluating performance of a metric against a goal. Examples: Evaluating contract renewals against a goal of a specific amount of renewals, actual payments against budget.

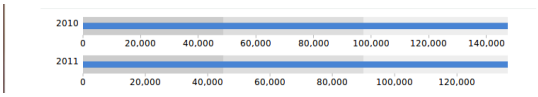


Figure 7.7: Bullet chart regarding payment in Viggo Cleaning department.

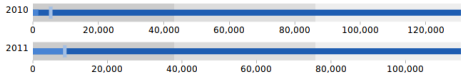


Figure 7.8: Bullet chart regarding payment in Viggo Cleaning department, per function.

7.2.4 Heat map

Heat maps are a great way to compare data across two categories using color. The effect is the quick observation of the intersection of the categories is strongest and weakest and of a distribution of events over a time or spatial area.

Advantages:

- Observation of event distribution. Examples: Segmentation analysis of payments per semester, segmentation analysis of training events over a semester.

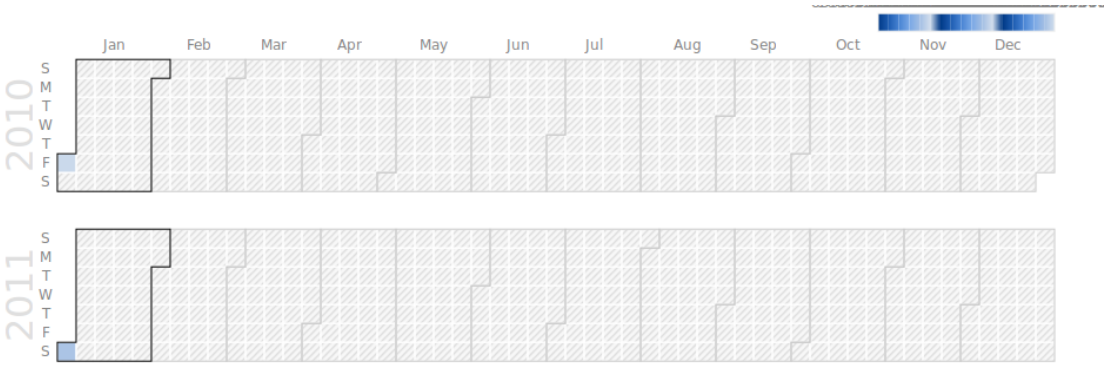


Figure 7.9: Heat map regarding payment in Viggo Cleaning department, per date.

Chapter 8

Conclusions

This work aims the design and the implementation of a fully centralized data repository for HR and BA analytics. More specifically, the concept idea of this thesis is the production of a data warehouse solution for Viggo. This data warehouse has been built after a 7 month research, which consists of theoretical and experimental studies. During this period of time, some conclusions have been made, regarding data warehousing solutions, terms of proof of concepts and potential work. In the following sections, these conclusions are presented from an organizational and from an academic perspective.

8.1 Organizational

From an organizational perspective, the state of Viggo will be analyzed. Viggo is a multilevel company, which operates in airport handling business field. This field requires scheduling at the finest level, otherwise the expenses for the company can sum into multi-thousands euros. As a result, logistics play an important role in Viggo's everyday life. The tools that assist scheduling in the market are many, but Microsoft Excel is the most popular among all. The use of this specific tool produces hundreds of spreadsheet per year and because of the reasons, which has been introduced in Chapter 2, HR and BA departments face difficulties to make fast and accurate decisions, regarding the development of Viggos manpower.

With the use of a data warehouse system, which transforms the raw information into meaningful charts, Viggo can easily produce solutions regarding decision making with low cost and without the need to switch into a new scheduling system. Moreover, the available OLAP server in the market, can utilize the views of the stored information, in a manner, which can unmask problems or give answers to difficult dilemmas.

8.2 Academic

From an academic perspective, data warehousing is a very broad topic and, as a result, there are a lot of issues, which can be discussed. For this reason, the analysis will be clustered according to the tiers of a data warehouse. Regarding the data sources of a data warehouse and the ETL process, the main conclusion is the need of standards for CSV spreadsheets. Spreadsheets are very popular inside business community, because of their availability and simplicity. However, this simple model causes a lot of issues when there is a need of data retrieval. The reason is the freedom of the user to store the information, without following any schema rules, such as relational model and this makes any assumption of data structure impossible. Latest technology can offer solution with the use of Data Mining and Query Languages, but even none of these solution can fit to any ETL requirement.

In terms, of data storage, relational database can fulfill the most of the obligations. The advantages when a relational database is used are the following:

Well-known Each computer and data scientist is trained and aware of using a relational database. More specifically, they know the fundamentals of a typical SQL database and they can use it with almost no training in the most of the cases.

Mature Relational databases have already passed the rough tests of a new technology. There are plenty of drivers and tools, which can connect them with the rest of the applications and communities which can support the users. Moreover, there are no political issues which should be taken into account regarding their use.

Still popular Taking into account that Facebook in year 2015 still uses a SQL database for the storage of millions of records, we understand that a relational database can still support the needs of an application, which intends to deal with big data.

At the highest tiers of a data warehouse system, OLAP servers can provide decent and efficient solutions for Multidimensional structures. More specifically, the implementation of the OLAP and the Front-end tiers have already become a trivial process and the newest versions of these tools provide solutions even for the Back-end tier in terms of ETL process.

8.3 Future Work

Since this work is a newborn product, there are a lot of parameters, which could be improved, in order to offer a better overview to business and academic community.

8.3.1 Organizational

From an organizational perspective, the future steps of Viggo will be analyzed. Viggo can set short-term and long-term targets. A short term target, which should be implemented is the set of standards, regarding CSV spreadsheets, within the company. More specifically, there is a big amount of spreadsheets, which have been implemented for the same purpose but because of the fact that the editor are different employees, their structure is different. As mentioned in Chapter 1, since it is difficult for Viggo the shift to appropriate contemporary software solutions and because of the fact that spreadsheets are normally created by individuals, who have different educational background and not any knowledge about data management, within Viggo there is a big amount of personalized spreadsheets. If Viggo provides spreadsheet templates and directions for spreadsheet development to its employees, then there will be at least a standard per spreadsheet and a potential for some automated processes regarding ETL process. A long-term target can be the shift to a decent piece of software, which will provide better management of the stored to the spreadsheets information. One step further, can be the total elimination of spreadsheets from Viggo's processes and their use will be limited only for personal or inner group use. Instead of spreadsheets, there can be an investment on systems, which are specified into specific tasks, such as scheduling, human resource etc. This option is more expensive, but, on the other hand, is more functional and it will assist the employees. Moreover, the data management of this systems can assist Viggo's data warehouse to provide better insights for the organization.

8.3.2 Academic

Regarding academic community, the future steps of this work should focus on Back-end and on Data warehouse tiers. Regarding Back-end tier, CSVQL should become an ordinary query language and not remain a MySQL implementation. This decision requires the design of a grammar, which will implement the syntax of the language and will define query's clauses. The evolution of CSVQL can be split into phases.

Phase A Design of Extraction tasks.

Phase B Design of Transform tasks.

Phase C Design of Loading tasks.

Therefore, at the end of Phase C, CSVQL will be an ETL tool, which can implement the Back-end tier of Viggo's data warehouse.

Regarding Data warehouse tier, a study can take place about NoSQL data stores and data warehousing. A lot of research has been published for the advantages of NoSQL data stores as OLTP data stores. Thus, it would be an interesting future project, a research regarding the capability of NoSQL to fulfill OLAP requirements. The requirements of data warehousing evolve and the evolution of needs demands the evolution of solutions.

Bibliography

[1] DB-Engines Ranking. http://db-engines.com/en/ranking/, Retrieved in September 2016. 17

[2] icCube Suite. http://www.iccube.com/iccube-suite/, Retrieved in September 2016. 17, 53, 54

[3] MySQL User Account Management. http://dev.mysql.com/doc/refman/5.7/en/user-account-management.html, Retrieved in September 2016. 14

[4] Raet official website. http://www.raet.nl/, Retrieved in September 2016. 21

[5] The 3rd most common button in data apps is... http://www.powerpivotpro.com/2012/03/the-3rd-most-common-button-in-data-apps-is/, Retrieved in August 2016. 2

[6] Alejandro Vaisman & Esteban Zimanyi. *Data Warehouse Systems*. Springer, 2014. 1, 6, 7, 9, 14, 15, 45, 50, 51

[7] Alkis Simitsis. Modeling and Optimization of Extraction-Transformation-Loading (ETL) Processes in Data Warehouse Environments, 2004. Dissertation thesis. 23, 24, 25

[8] B. Scalzo. Oracle DBA Guide to Data Warehousing and Star Schemas. *Prentice Hall PTR*, 2003. 23

[9] Bin Liu, H.V. Jagadish. A Spreadsheet Algebra for a Direct Data Manipulation Query Interface. *ICDE '09 Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 417–428, 2009. 11

[10] Denizon Team. Spreadsheet Risks in Banks. *https://www.denizon.com/*, 2016. Retrieved in August. 2

[11] Denizon Team. Top 10 Disadvantages of Spreadsheets. *https://www.denizon.com/*, 2016. Retrieved in August. 2, 3

[12] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communication of the ACM*, 13(6):377–387, 1970. 50

[13] Eirik Bakke, David R. Karger. Expressive Query Construction through Direct Manipulation of Nested Relational Results. *SIGMOD '16 Proceedings of the 2016 International Conference on Management of Data*, pages 1377–1392, 2016. 11

[14] G.V.Siva Krishna. Anovel approch of eye tracking and blink detection with a human machine. *International Journal of Advancements in Research and Technology*, 2(7), 2013. 42

[15] Henryk Rybinski. On first-order-logic databases. *ACM Transactions on Database Systems (TODS)*, 12(3):325–349, 1987. 14

[16] Hugo Shebbeare - Microsoft MVP, SQL Server: Systems Administration. Database Security Best Practices for the Vigilant Database Administrator and Developer. *https://technet.microsoft.com/*, Retrieved in September 2016. 14

[17] Juliana Freire, Boris Glavic, Oliver Kennedy, Heiko Mueller. The Exception that Improves the Rule. *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, (7), 2016. 10

[18] Marcelo Arenas, Francisco Maturana, Christian Riveros, Domagoj Vrgoc. A framework for annotating CSV-like data. *Proceedings of the VLDB Endowment*, 9(11):876–887, 2016 . 10

[19] Mark Everett Hall. Managing Information in the Enterprise: Perspectives for Business Leaders. Forbes Insights, 2010. 13

[20] Michael S Engel. The taxonomy of recent and fossil honey bees (Hymenoptera: Apidae: Apis). *Journal of Hymenoptera Research*, 7-8, 1992. 41

[21] Triantos Konstantinos. NoSQL database systems for professional football analytics. 2016. Eindhoven University of Technology, Capita Selecta Seminar. 14, 26, 38

[22] W3C. CSV on the Web: A Primer. http://www.w3.org/, Retrieved in September 2016. 9, 10

[23] Zhe Chen, Michael Cafarella. Integrating Spreadsheet Data via Accurate and Low-Effort Extraction. *KDD*, 20:1126–1135, 2014. 10

Appendix A

Data Load script

In this section, the script, which transforms a CSV file into a MySQL table, is presented. The functionality of this script, which is implemented in JAVA and is consisted of two classes, is presented in Section 5.3.

A.1 CSVQL.java

```
package csvql;

/**
 *
 * @author Konstantinos Triantos
 */


public class CSVQL {

    /**
     * @param args the command line arguments
     * @throws java.lang.Exception
     */
    @SuppressWarnings("resource")

    public static void main(String[] args) throws Exception {

        LoadCSV loadCSV1 = new LoadCSV("CSVQL/files/Samples/Sample1.csv");
    }
}
```

Listing A.1: The public class CSVQL

A.2 LoadCSV.java

```
package csvql;

/*****************************************************************************/
/** import libraries *********************************************************/
/*****************************************************************************/

import com.opencsv.CSVReader;
import java.io.File;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


/*****************************************************************************/
/** class LoadCSV ************************************************************/
/*****************************************************************************/

public class LoadCSV {

    /*************************************************************************/
    /** Variables ****************************************************************/
    /*************************************************************************/

    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost:3306/CSVQL?useSSL=false";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "root";

    private final File csvfile;
    private List<String[]> cells;


    /*************************************************************************/
    /** Constructor **************************************************************/
    /*************************************************************************/

    LoadCSV(String inputfile) throws Exception {

        System.out.println("Loading from CSV file into database just started...");

        this.csvfile = new File(inputfile);

        inputData();

        connectDatabase();

        System.out.println("Loading from CSV into database finished.\n");
    }

    /*************************************************************************/
    /** Import data from CSV *****************************************************/
    /*************************************************************************/

    private void inputData() throws Exception {

        System.out.println("Reading from input file "+ csvfile.getPath().toString()
            +" ...");
```

```
        cells = new ArrayList<String[]>();

        /* For normaliaztion of the table's row counting (start from 1 and not from
             0), we add this line in the begining */
        String normalization[] = {"normalization","normalization"};
        cells.add(normalization);

        /*
        * the 0(zero) indicates the line which the retrieval starts
        * CSVReader(Reader reader, char separator, char quotechar, int line)
        */
        CSVReader reader = new CSVReader(new FileReader(csvfile.toPath().toString()
            ), ',', '"', 0);

        /* Read CSV line by line and use the string array as you want */
        String[] nextLine;
        while ((nextLine = reader.readNext()) != null) {
            if (nextLine != null) {

                /* For normaliaztion of the table's columns counting (start from 1
                    and not from 0), we add this extra string in the begining */
                String record = "normalization,"+Arrays.toString(nextLine).
                    replaceAll("[\\[\\]]", "");
                cells.add(record.split(","));
            }
        }

        System.out.println("Reading from input file "+ csvfile.getPath().toString()
            +" finished.");
    }

    /**************************************************************************/
    /** Connect to Database ***************************************************/
    /**************************************************************************/

    private void connectDatabase() {

        Connection conn = null;
        Statement stmt = null;

        try {

            //STEP 2: Register JDBC driver
            Class.forName("com.mysql.jdbc.Driver");

            //STEP 3: Open a connection
            System.out.println("Connecting to CSVQL database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connected to CSVQL database successfully...");

            //STEP 4: Execute a query DELETE TABLE
            System.out.println("Deleting table "+csvfile.getName().substring(0,
                csvfile.getName().length() - 4)+" in CSVQL database...");
            stmt = conn.createStatement();

            String sql = "DROP TABLE IF EXISTS "+csvfile.getName().substring(0,
                csvfile.getName().length() - 4);

            stmt.executeUpdate(sql);
            System.out.println("Table "+csvfile.getName().substring(0, csvfile.
                getName().length() - 4)+" is deleted in CSVQL database...");

            //STEP 5: Execute a query CREATE TABLE
            System.out.println("Creating table "+csvfile.getName().substring(0,
                csvfile.getName().length() - 4)+" in CSVQL database...");
            //stmt = conn.createStatement();
```

```
            sql = "CREATE TABLE "+csvfile.getName().substring(0, csvfile.getName().
                length() - 4)+" " +
                   "(csvrow INTEGER not NULL, " +
                   " csvcolumn INTEGER not NULL, " +
                   " csvvalue VARCHAR(255), " +
                   " PRIMARY KEY ( csvrow, csvcolumn ))";

            stmt.executeUpdate(sql);
            System.out.println("Created table "+csvfile.getName().substring(0,
                csvfile.getName().length() - 4)+" in CSVQL database...");

            //STEP 6: Execute a query INSERT RECORDS
            System.out.println("Inserting records into the table "+csvfile.getName
                ().substring(0, csvfile.getName().length() - 4)+" ...");
            for (int i = 1; i < cells.size(); i++) {
                for (int j = 1; j < cells.get(i).length ; j++) {

                    sql = "INSERT INTO "+csvfile.getName().substring(0, csvfile.
                        getName().length() - 4)+" " +
                    "VALUES ("+i+", "+j+", '"+cells.get(i)[j].trim()+"')";
                    stmt.executeUpdate(sql);

                    //System.out.println(i + "," +j+ "," +cells.get(i)[j]);
                }
            }
            System.out.println("Inserted records into the table "+csvfile.getName()
                .substring(0, csvfile.getName().length() - 4)+" ...");

            //STEP 7: Execute a query CREATE INDEXES
            System.out.println("Creating index rowINDEX on "+csvfile.getName().
                substring(0, csvfile.getName().length() - 4)+" table in CSVQL
                database...");

            sql = "CREATE INDEX rowINDEX\n" +
                  "ON "+csvfile.getName().substring(0, csvfile.getName().length() -
                       4)+" (csvrow)";

            stmt.executeUpdate(sql);
            System.out.println("Created index rowINDEX on "+csvfile.getName().
                substring(0, csvfile.getName().length() - 4)+" table in CSVQL
                database...");

            System.out.println("Creating index columnINDEX on "+csvfile.getName().
                substring(0, csvfile.getName().length() - 4)+" table in CSVQL
                database...");

            sql = "CREATE INDEX columnINDEX\n" +
                  "ON "+csvfile.getName().substring(0, csvfile.getName().length() -
                       4)+" (csvcolumn)";

            stmt.executeUpdate(sql);
            System.out.println("Created index columnINDEX on "+csvfile.getName().
                substring(0, csvfile.getName().length() - 4)+" table in CSVQL
                database...");


        } catch (SQLException | ClassNotFoundException se) {
        } finally {
            //finally block used to close resources
            try {
                if (stmt != null) {
                    conn.close();
                }
            } catch (SQLException se) {
            }// do nothing
            try {
```

```
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException se) {
            }
        }
        System.out.println("Goodbye!");
    }
}
```

Listing A.2: The public class LoadCSV

Appendix B

CSVQL experiment results

This section presents the results of the experimental study in Section 5.4. More specifically, there are two tables: the former regarding the results of CSVQL scalability experiment with primary index and the latter regarding the results of CSVQL scalability experiment with secondary index.

B.1 Primary index

Table B.1: Results of CSVQL scalability experiment with Primary index

	One Join	Two Joins	Three Joins	Four Joins	Five Joins	Six Joins	Seven Joins	Eight Joins
1st experiment	0.003	0.0019	0.002	0.0022	0.0021	0.002	0.0021	0.0022
2nd experiment	0.0035	0.0022	0.002	0.0022	0.0021	0.0021	0.0022	0.0059
3rd experiment	0.0023	0.002	0.002	0.0022	0.002	0.002	0.0025	0.0043
4th experiment	0.0018	0.002	0.002	0.0019	0.0024	0.0019	0.0034	0.0044
5th experiment	0.0019	0.002	0.0022	0.0024	0.0022	0.002	0.0021	0.0022
6th experiment	0.004	0.0018	0.0022	0.0021	0.002	0.002	0.0021	0.0048
7th experiment	0.0021	0.0019	0.0021	0.0023	0.0026	0.0021	0.0021	0.0042
8th experiment	0.0023	0.0024	0.002	0.0021	0.0024	0.0021	0.0021	0.0021
9th experiment	0.0024	0.0021	0.003	0.0029	0.002	0.0019	0.002	0.0043
10th experiment	0.0035	0.002	0.002	0.002	0.0021	0.0028	0.0022	0.0022

B.2 Secondary index

Table B.2: Results of CSVQL scalability experiment with Secondary index

	One Join	Two Joins	Three Joins	Four Joins	Five Joins	Six Joins	Seven Joins	Eight Joins
1st experiment	0.0035	0.0062	0.0027	0.0023	0.0086	0.025	0.118	1.033
2nd experiment	0.0032	0.0023	0.001	0.0027	0.0047	0.014	0.108	1.087
3rd experiment	0.0015	0.002	0.002	0.0022	0.011	0.021	0.137	1.074
4th experiment	0.0032	0.0017	0.0021	0.0029	0.0072	0.02	0.186	1.081
5th experiment	0.0018	0.0028	0.00077	0.0037	0.0043	0.014	0.115	1.028
6th experiment	0.0017	0.0017	0.0024	0.0014	0.0054	0.026	0.174	1.052
7th experiment	0.0023	0.0038	0.0028	0.0044	0.0025	0.014	0.145	1.031
8th experiment	0.0019	0.0037	0.00083	0.0033	0.0096	0.023	0.122	1.082
9th experiment	0.0017	0.0024	0.0012	0.001	0.0058	0.026	0.172	1.038
10th experiment	0.00223	0.0013	0.001	0.001	0.0057	0.042	0.114	1.104