Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Department of Information Engineering
Master's Degree Course in Computer Engineering

Analysis of Data Integration and Business Management solutions based on EPM and Data Warehouse technologies

Master Candidate

Francesco Caldivezzi

Student ID 2037893

Supervisor

Prof. Gianmaria Silvello

University of Padova

Academic Year: 2022/2023
Graduation Date: 23/10/2023

To my parents,
relatives and
friends.

Abstract

In recent years, Enterprise Performance Management (EPM) and Data Warehouse (DW) solutions are increasingly gaining ground to handle problems such as Data Integration and Business Management. In this thesis, it will be analyzed in practice how these problems are solved by means of the technologies mentioned, in a practical case, within a company. In particular, the business problems encountered before the solution, the technologies adopted to solve them and the functioning of the tools used will be presented. In conclusion, the advantages and disadvantages of the proposed solutions and possible future improvements will be described.

Sommario

Negli ultimi anni, soluzioni EPM e di DW stanno sempre più prendendo piede per gestire problemi come Data Integration e Business Management. In questa tesi, si andrà ad analizzare all'atto pratico come questi problemi vengono risolti per mezzo delle tecnologie citate, in un caso pratico, all'interno di una realtà aziendale. In particolare verranno presentati i problemi aziendali riscontranti prima della soluzione, le tecnologiche adottate per risolverli ed il funzionamento degli strumenti adoperati. In conclusione si descriveranno vantaggi e svantaggi delle soluzioni proposte e possibili miglioramenti futuri apportabili.

Contents

List of Figures

List of Tables

List of Code Snippets

List of Acronyms

PwC PricewaterhouseCoopers

BOM Bill Of Materials

ERP Enterprise Resource Planning

SCM Supply chain management

CRM Customer Relationship Management

HCM Human Capital Management

PLM Product Lifecycle Management

DW Data Warehouse

BI Business Intelligence

IoT Internet of Things

DSS Decision Support System

EDW Enterprise Data Warehouse

ETL Extract, Transform, Load

OLAP Online Analytical Processing

OLTP Online Transnational Processing

SAP Systemanalyse und Programmentwicklung

SE Societas Europaea

GmbH Gesellschaft mit beschränkter Haftung

SRM	Supplier Relationship Management
SCM	Supply Chain Management
BTP	Business Technology Platform
EPM	Enterprise Performance Management
SDA	Smart Data Access
ODP	Operational Data Provisioning
EIM	Enterprise Information Management
ADSO	Advanced DataStore Object
DTP	Data Transfer Process
BPC	SAP Business Planning and Consolidation
BPF	Business Process Flow
PoC	Proof of Concept
BBP	Business Blueprint
FUT	Functional Unit Testing
SIT	System Integration Testing
UAT	User Acceptance Tests
AFO	Analysis for Microsoft Office
WBS	Work Breakdown Structure
SAC	SAP Analytics Cloud

1 Introduction

This thesis is the final result of an internship carried out at the financial consulting company PricewaterhouseCoopers (PwC), one of the Big Four accounting firms and the second-largest professional services network in the world, inside the Finance Transformation team in Rubano (PD), within a project in a client company, namely, "Aruba S.p.A" an Italian leader company that offers data center, web hosting, email and domain name registration services. The project, for this client company, started in January 2023, four months before I entered PwC, and has an estimated duration of roughly one year. Therefore, because this internship that I performed was part of a large project, my activities were limited to a particular area of the overall intervention, and, because my presence inside the company has a total duration of six months, this work will describe, in detail, only the tasks carried out by me, as well as the technologies used and learned to perform the jobs assigned. In particular, this thesis will analyze the main problems that the client company is facing, the general technologies used in the project solutions for those issues, as well as the strategies adopted and work methodologies for analyzing, understanding, and developing the implementation necessary.

This work starts with the description of the two main classes of technologies used to solve the problems that the company "Aruba S.p.A." is facing, i.e. Data Warehouse Systems and Enterprise Performance Management Software. Nowadays, such types of tools are in rapid expansion and a lot of companies are starting to use them as a solid, simple, and powerful solution to several problems including Business Management and Data Integration.

In particular, the second Chapter of this thesis defines a Data Warehouse System as "a digital storage system that connects and harmonizes large amounts of data from many different sources". Additionally, the branch continues, with a brief overview of the history of Data Warehouse Systems. Then, after defining what Data Warehouse Systems can store, and, their principal characteristics, the Chapter continues with also an overview of several other important aspects of this type of technology. For instance, it defines the difference between the two main types of Data Warehouse Systems "Aggregation Data Warehouse" and "Virtualization Data Warehouse", the key components of a Data Warehouse System: "Extract Transform Load", "Metadata", "Central Database" and "Access Tool", as well as the dissimilarity between a Data Warehouse System, a Database, a Data Lake and a Data Mart. Finally, the Chapter ends with the main advantages that a Data Warehouse System can provide and with a simple yet, updated analysis of the current market of such technology.
Next, in the third section of this paper, the focus will switch to Enterprise Performance Management Software. Again, in the same way, done in Chapter two, after explaining that Enterprise Performance Management Software is a type of software that is used to help developers build a solid implementation for problems that fall under the Business Management class, the portion of this work continues with the analysis of the main purposes for which Enterprise Performance Management Software is needed, for instance, for creating a strategic plan. Finally, the Chapter ends with the reasons why Enterprise Performance Management Software is so useful as well as an overview of the main products present in the market.
Then, after these two introductory sections, the following one, defines the instances of the technologies previously mentioned, used in the "Aruba S.p.A." project, mainly "SAP BW/4 HANA" and "SAP BPC", respectively the Data Warehouse System and the Enterprise Performance Management Software used. However, before technically explaining those two previously mentioned technologies, Chapter Four begins with an introduction to "SAP", namely, the worlds leading company for Enterprise Resource Planning software vendor as well as one of the main producers of Data Warehouse Systems and Enterprise Performance Management Software. Next, the analysis of the "SAP" company continues with a brief description of its history and with a study of its main products. Following, this small outline of the "SAP" company, the Chapter continues with the more technical stuff. Both "SAP BW/4 HANA" Data Warehouse and "SAP

BPC" will be examined from a technical point of view. Specifically, the focus for "SAP BW/4 HANA" will be not only on its working architecture but also on the components that a developer must use to build a solution with it. Among such components, four will be perfectly eviscerated: "InfoObjects", "Advanced Data-Store Objects", "Query", and "Planning Sequence". In particular, the first two, since they represent the two main storage mechanisms of "SAP BW/4 HANA", will be reserved for a space where will be analyzed how data are handled, stored, and managed through their usage. For the other two components instead, the description will be detailed as well but in this case, with a more example-like approach. In fact, will be provided two examples of how queries are shown as well as how to eventually implement a "custom" planning sequence. Finally, Chapter Four ends with a review of "SAP BPC", through a simple description of what it is, and, a focus on its main features, particularly on the "Business Process Flow" one.
The thesis will, then, enter Chapter Five where, here, the analysis will shift toward the project development for the "Aruba S.p.A." company. There, a simple introduction of what the "Aruba S.p.A." company does will break the ice for making some space for the more complex stuff. In fact, after that, the chaotic, yet, effective working method adopted by PwC for this project will be defined. Specifically, not only will be explained the several stages that the development will be subject to, like the consolidation of the requirements through the "Business Blueprint" and the testing that the solution developed will be subject to, but also the interaction between the PwC's teams that developed the solution, as well as the ones on the "Aruba S.p.A." side that dictate the different requirements needed. In particular, the approach of PwC in this case was to fragment this project into three different groups to make the process as fluent as possible. Just to anticipate, the idea was to divide the teams by competence into the "S4" Team, "BPC" Team, and "Change Management" Team. The tasks carried out by them differ in this way by their abilities, respectively, being able to use "SAP S/4HANA" as Enterpise Resource Planning technology, being able to use "SAP BPC" and "SAP BW/4 HANA" technologies, and, being able to manage the client relations and help if difficulties arise with the new solution. However, because I was inserted into the "BPC" Team, only a detailed description of such a Team will be provided, while for the others, will be described only the interaction process with the client and with the latter Team. Next, the Chapter continues with the list and definition of the three main problems that "Aruba

S.p.A." is facing: "Data Integration" i.e. dealing with dissimilar, messy data coming from several different sources; "The creation of the Managerial or Management Consolidation", namely presenting aggregate financial results by the organization's strategic business groups; and, "The computation of costs and revenues of its internal Bill of Materials" i.e. understanding what are the revenues and/or costs of some or all the levels of a multi level Bill of Materials by splitting such information coming from a certain income statement. Finally, the Chapter ends with the solutions developed for those problems, by first explaining the overall architecture developed, consisting of using "SAP S/4HANA" as a single Enterprise Resource Planning tool, that, will be adopted in the future by all the organizations owned by "Aruba S.p.A.", "SAP BW/4 HANA" as the main "Data Integration" solution, i.e., the collector of all the companies data from all the different sources: "SAP S/4HANA", "Onda", "Oceano", etc. as well as the cleaner and aggregator of these data, and finally, "SAP BPC" as final user interface through the "Business Process Flow" functionality, which, with the help of Excel and Analysis for Office, makes this last device usable also for reporting. Additionally, this last portion of the Chapter ends with detailed descriptions of the single solutions to the previously mentioned problems. Specifically, "Data Models" and "Logical Layers" built for "SAP BW/4 HANA" will be the main focus for the first problem, and the "Allocation Engine" will be the main discussion for the problem of "The computation of costs and revenues of its internal Bill of Materials", while how to configure "SAP BPC" will describe how to produce the Consolidation.

Last but not least, Chapter Six, will end the thesis by summarizing what has been done, what should be done in the future for making improvements, and what Aruba S.p.A. could do to increase its productivity eventually.

2
Data Warehouse Systems

This chapter will explore the concept of DW by understanding its main characteristics, types, key components, and architecture. Additionally, a section will be reserved for the definition of what they can store as well as one for the advantages that they bring to the table, and, lastly one will be used for the market analysis.

2.1 Introduction

A DW [19] is a digital storage system that connects and harmonizes large amounts of data from many different sources. Its purpose is to feed Business Intelligence (BI)[1], reporting, and analytics, and support regulatory requirements, so, companies can turn their data into insight and make smart, data-driven decisions. Data flows into a DW from operational systems, databases, and external sources such as partner systems, Internet of Things (IoT) devices, weather apps, and social media, usually, on a regular cadence. The location of the storage of the data, in recent years, has moved away from traditional on-premise infrastructure to multiple locations, including on-premise, private cloud, and public cloud.

[1]BI [5] is the set of strategies and technologies used by enterprises for the data analysis and management of business information.

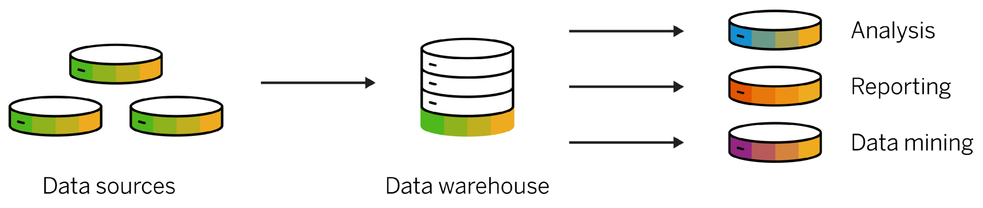

Figure 2.1: What is a Datawarehouse?

2.2 History

DWs first came onto the scene [17] in the late 1980s and, their initial purpose was to help data flow from operational systems into Decision Support System (DSS). These early DWs required an enormous amount of redundancy. Most organizations had multiple DSS environments that served their various users. Although the DSS environments used much of the same data, the gathering, cleaning, and integration of the data was often replicated for each environment. As DWs became more efficient, they evolved from information stores that supported traditional BI platforms into broad analytics infrastructures that support a wide variety of applications, such as operational analytics and performance management. Moreover, DWs have progressed over time to deliver incremental additional value to the enterprise by becoming Enterprise Data Warehouse (EDW). Nowadays, AI and machine learning are also transforming DWs. They are changing DWs requirements and capabilities. For instance, the latest step of the evolution of DWs, also known as autonomous DWs offers enterprises the ability to extract even greater value from their data while lowering costs and improving DWs reliability and performance thanks to AI and machine learning.

2.3 What can they store?

When DWs first became popular in the late 1980s, they were designed to store [19] information about people, products, and transactions. This data, called structured data, was very well organized and formatted for easy access. However, businesses soon wanted to store, retrieve, and analyze unstructured data, like documents, images, videos, emails, social media posts, and raw data from machine sensors. For such a reason a modern DWs can accommodate both structured and unstructured data. By merging these data types and breaking down silos between the two, businesses can get a complete, comprehensive picture of the most valuable insights. So, without data warehousing, it is very difficult to combine data from heterogeneous sources, ensure it is in the right format for analytics, and get both a current and long-range view of data over time.

2.4 Characteristics

One of the important things that must be understood when dealing with DWs is the main features [9] that make them different from other tools. In particular, such characteristics reside on the data that DWs store, and are:

- Subject-Orientation: this means that data in a data warehouse revolves around the subjects of the enterprise;
- Data Integration: this characteristic means that data found within a DWs are integrated, i.e., since it comes from several operational systems, all inconsistencies must be removed;
- Time-Variant: such a feature consists of the fact that data stored in DWs is mostly historical data. Therefore, it can be used for data mining and forecasting;
- Nonvolatile: this means that DWs stores mainly read-only data. So, operations like updates or deletions are not so common;

2.5 Types

Another aspect that must be considered when operating with DWs is how many types of DWs there are. Knowing how many types of DWs exist as well as the difference between them can have a huge impact both on the economic and implementation side. Therefore it is crucial information to remember while working with DWs technologies. So, nowadays, mainly, there are two main types [9] of DWs:

- Aggregation DWs: in this type of DWs data can be aggregated in Data Marts[2] at different levels of abstraction, and, typically when using this type of DWs, the analysis starts at a higher level and drills down to lower levels of detail;

[2]A Data Mart [8] is a structure of DW environments, used to retrieve client-facing data.

- Virtualization DWs: in this case, the data used inside the DWs remains in its original locations and real-time access is established to allow analytics across multiple sources. This can aid in resolving some technical difficulties such as compatibility problems when combining data from various platforms, lowering the risk of error caused by faulty data, and guaranteeing that the newest data is used. Furthermore, avoiding the creation of a new database containing personal information can make it easier to comply with privacy regulations. However, with data virtualization, the connection to all necessary data sources must be operational as there is no local copy of the data, which is one of the main drawbacks of the approach;

2.6 The Key Components and Architecture

The last two important technical topics that will be addressed in this section for a DW are [19] the key components of a DW itself and its architecture. In particular, for what concerns the key components, a DW has mainly four of them: a central database, Extract, Transform, Load (ETL) tools, metadata, and access tools. All of these are engineered for speed, so, results can quickly get and data can be analyzed on the fly.

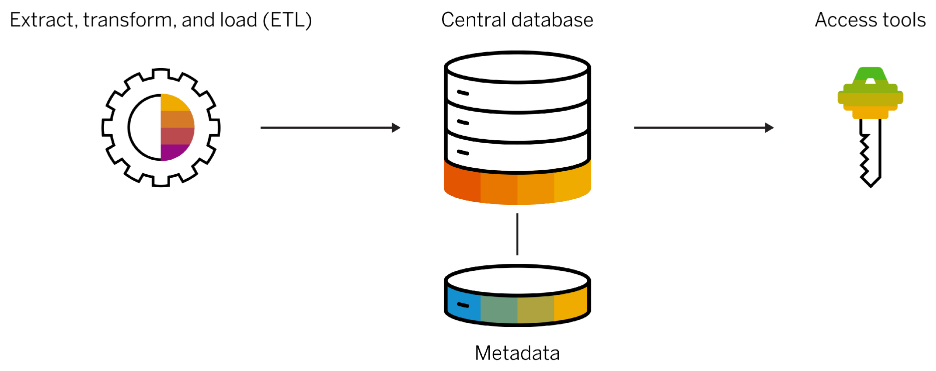

Figure 2.2: Diagram showing the components of a data warehouse.

The central database is the foundation of a data warehouse. Traditionally, they have been standard relational databases running on-premise or in the cloud. But, in the last few years, in-memory databases are rapidly gaining in popularity because of the Big Data phenomenon. ETL tools, instead, are used to feed data to the central database to achieve rapid analytical consumption and integrate data from different sources. Sources correspond to metadata which technically are data about other data. Metadata, specifies, not only the source, but also the usage, values, and other features of the data sets in a data warehouse. Finally, data warehouse access tools allow users to interact with the data in a data warehouse.

On the other hand, the architecture of a data warehouse can be divided into three main layers :

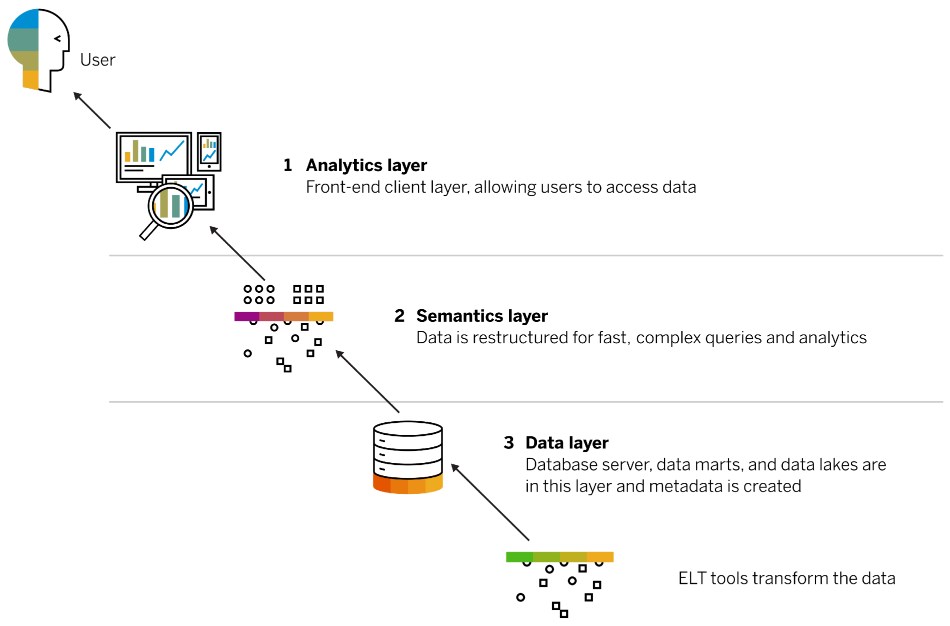
User
1 Analytics layer
Front-end client layer, allowing users to access data
2 Semantics layer
Data is restructured for fast, complex queries and analytics
3 Data layer
Database server, data marts, and data lakes are in this layer and metadata is created
ELT tools transform the data

Figure 2.3: Diagram of data warehouse architecture.

- Data layer: Data is extracted from your sources and then transformed and loaded into the bottom tier using ETL tools. The bottom tier consists of a database server, data marts, and data lakes. In this tier, metadata is created and data integration tools, like data virtualization, are used to seamlessly combine and aggregate data;
- Semantics layer: In the middle tier, Online Analytical Processing (OLAP)[3] and Online Transnational Processing (OLTP)[4] servers, restructure the data for fast, complex queries and analytics;
- Analytics layer: The top tier is the front-end client layer. It holds the data warehouse access tools that let users interact with data, create dashboards and reports, monitor KPIs, mine and analyze data, build apps, and more. This tier often includes a workbench or sandbox area for data exploration and new data model development;

[3]OLAP [15] is an approach to answer multi-dimensional analytical queries, that means that OLAP is part of the broader category of BI.
[4]OLTP [16] is a type of database system used in transaction-oriented applications, such as many operational systems.

2.7 Benefits

A well-designed data warehouse is the building block for any successful BI or analytics program. The reasons behind this statement can be acknowledged by realizing what are the main benefits [19] that a data warehouse brings to the table. In particular, the main ones are:

- Better business analytics: With data warehousing, decision-makers have access to data from multiple sources and no longer have to make decisions based on incomplete information;
- Faster queries: Data warehouses are built specifically for fast data retrieval and analysis. Therefore large amounts of consolidated data can be queried rapidly;
- Improved data quality: Before being loaded into the data warehouse, data cleansing cases are created by the system and entered in a worklist for further processing, ensuring data is transformed into a consistent format to support analytics and decisions, based on high-quality, accurate data;
- Historical insight: By storing rich historical data, a data warehouse lets decision-makers learn from past trends and challenges, make predictions, and drive continuous business improvement;

2.8 Common Misunderstood Concepts

When working with Data Warehouses most of the time there are a lot of concepts that come into place and are often confused [19]. For making this clear, below are listed the main terms that frequently are exchanged together.

- Data warehouse and Database: Databases and Data Warehouses are both data storage systems, but, they serve different purposes. A Database stores data usually for a particular business area. A Data Warehouse stores current and historical data for the entire business and feeds BI and analytics. DW uses a database server to pull in data from an organization's databases and has additional functionalities for data modeling, data life cycle management, data source integration, and more;
- Data Warehouse and Data Lake: Both Data Warehouses and Data Lakes are used for storing Big Data, but they are very different storage systems. A DW stores data that has been formatted for a specific purpose, whereas a Data Lake stores data in its raw, unprocessed state;

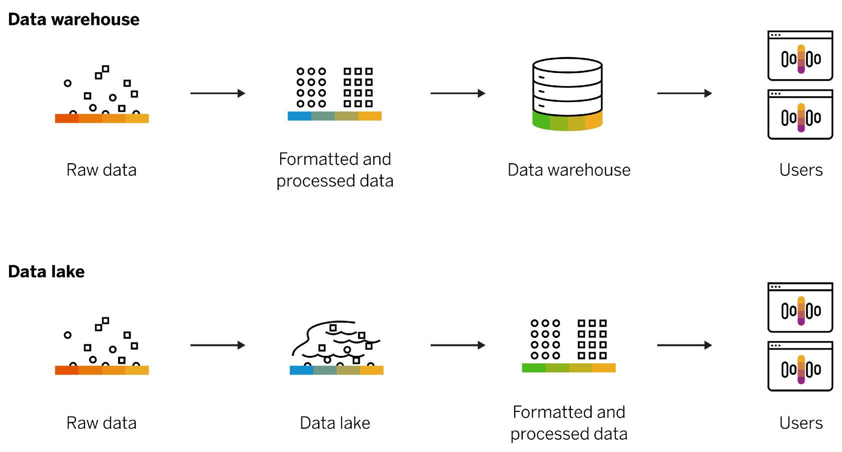

Figure 2.4: Diagram of a Data Warehouse compared with a Data Lake.

- Data Warehouse and Data Mart: A Data Mart is a subsection of a DW, partitioned specifically for a department or line of business like sales, marketing, or finance. Some Data Marts are created for standalone operational purposes as well. While a Data Warehouse serves as the central data store for an entire company, a Data Mart serves relevant data to a select group of users. This simplifies data access, speeds up analysis, and gives them control over their data. Multiple data marts are often deployed within a Data Warehouse;

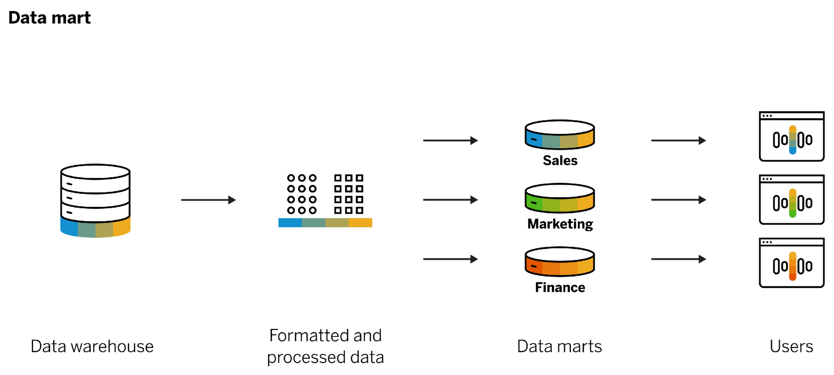

Figure 2.5: Diagram of a Data Mart and how it works.

2.9 Market Analysis

Nowadays the market of Data warehouses is in constant evolution, estimates said [10] that by 2028 the market is projected to reach $51.18 billion in value. The reason for these huge projected revenues can be re-conducted to the fact that there are a lot of competitors [11] that deal with DWs like Snoflakes, Amazon

Redshift, SAP BW, Google BigQuery, and Oracle DW. A confirmation of this comes from the fact that for the project that will be analyzed in the last chapter of this work, will be used by PwC company a Data Warehouse technology i.e. "SAP BW/4 HANA" as main tool.

3 Enterprise Performance Management Software

In this chapter, the main focus will be EPM tools. This segment will explain what EPM software is, its benefits, how the process of using EPM software works, and, finally, will be analyzed the current market share and main products for such software.

3.1 Introduction

EPM is a type [29] of software that specializes in budgeting, forecasting, and financial management. It provides data analytics, reporting, and forecast modeling so organizations can analyze, understand, and plan strategically for the business. With EPM software, companies can align business strategy with business execution. The technology uses feedback drawn from data generated from systems, processes, and activities across the organization. The resulting analytics help to identify business drivers and other insights. Companies can assess new opportunities, augment profitability in existing business, and respond with greater agility in the face of unexpected change and disruption. Enterprise performance management software helps inform future decisions, drive efficiencies, and improve the financial and operational performance of the company.

3.2 How are they used?

As with most traditional software, enterprise performance management systems were initially installed on-premise. Today, more and more EPM software systems run in the cloud. Therefore, their usage [29] process has changed, and here is how:

- Access data across all business units: this means that the data fed to the EPM software can come from different sources like e-commerce systems, front-office and back-office applications, data warehouses, and external data sources. Moreover, it can be accessed from different business units across the same organization;
- Create a strategic plan: with the use of EPM software, it can be created data analytics to build forecast models and ad-hoc simulations across multiple dimensions. That means that a company that uses EPM software, can make data-driven decisions, so that, it maximizes its profitability, performance, etc.;
- Budget: EPM systems can be used to work collaboratively across lines of business to crowdsource plans and build flexible budgets;
- Track and report: because EPMs work with real-time data, such data can be used to assess performance across the enterprise and determine if adjustments are required;
- Assess and analyze: this means that EPMs can help users to identify new areas of opportunity, resolve areas of underperformance, and use the intelligence to inform the next cycle of strategic planning;

3.3 Benefits

Companies nowadays are using a lot of EPM software because they simplify processes as well as they do have a lot of benefits [29]. In particular, the top benefits are:

- Profitability: EPM technology can help increase margins, maximize returns, and identify new opportunities to increase profitability and performance;
- Integrated business strategy: EPM delivers insights based on real-time data, so companies can explore different opportunities and make the most profitable and efficient decisions;
- Modern, automated financial processes: EPM connects processes and data from across the organization for a comprehensive view of the business;

- Regulatory oversight: EPM supports a range of reporting standards with accurate and current information, eliminating the need for multiple reporting systems and accelerating cycle times;
- Faster reconciliation of accounts: EPM provides automated workflows that eliminate risk and securely expedite global account reconciliation;

3.4 Products and Market Share

The market [28] of EPM software is mainly dominated by 10 vendors: Oracle, SAP, Anaplan, IBM, Workday, InsightSoftware, OneStream software, Workiva, Planful, and Infor. These 10 vendors, in 2021 accounted for nearly 75.9% of the the global EPM applications market.

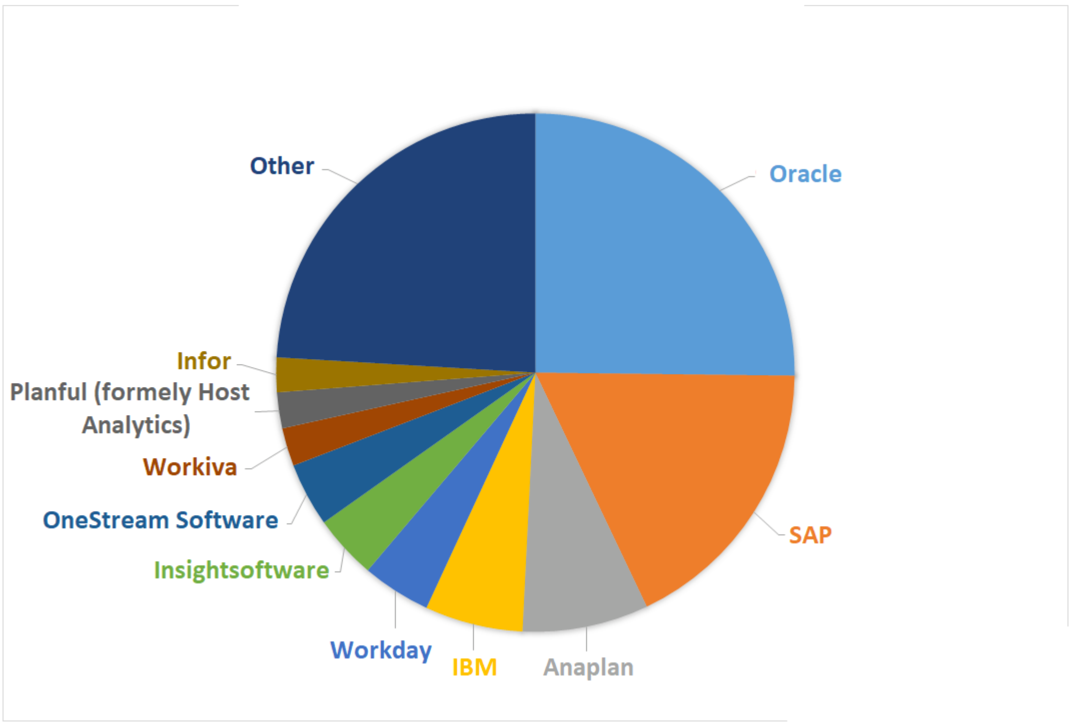

Figure 3.1: Market Share of EPM Software

Moreover, of the top 60 [3] EPM software products, one that appears in the top places among the SAP products is SAP Business Planning and Consolidation i.e. SAP SAP Business Planning and Consolidation (BPC). As will be seen better later, this is the main EPM tool that will be used in the project for the "Aruba S.p.A." company.

4 SAP Technologies

This chapter will present and analyze the two main technologies used for developing the solution for the "Aruba S.p.A." project. In particular, the main focus will be on the technical aspects of SAP BW/4 HANA and SAP BPC i.e. the DW and the EPM software respectively used. Moreover, a section will also be reserved for an introduction to who is SAP and what it does.

4.1 The SAP Company

This section will explain who is Systemanalyse und Programmentwicklung (SAP) and what it does by concentrating on the history and products that, SAP, develop and sell, every year, to the entire world.

4.1.1 Introduction

SAP [18] i.e. System Analysis and Software Development is a German multinational software company based in Walldorf, Baden-Württemberg. It develops enterprise software to manage business operations and customer relations. SAP is the largest non-American software company by revenue, the world's third-largest publicly traded software company by revenue, and the largest German company by market capitalization. Its headquarters are located in Walldorf, Baden-Württemberg, Germany, with offices in 180 countries.

4.1.2 History

The history [18] of the SAP company, dates back to when Xerox exited the computer hardware manufacturing industry in 1971, asking IBM to migrate its business systems to IBM technology. Among the IBM engineers, five of them from the AI department were working on an enterprise-wide system based on this software, only to be told that it would no longer be necessary. Rather than abandoning the project, they decided to leave IBM Tech and start another company. So, in June 1972, they founded the SAP company, as a private partnership under the German Civil Code. Their first client was Imperial Chemical Industries, where they developed mainframe programs for payroll and accounting. The main feature of their work was the fact that they decided to store the data locally in the Electronic System while using a common Logical database for all activities of the Organization. For this reason, they called their software a real-time system, since there was no need to process the punch cards overnight. This first version was also a standalone software that could be offered to other interested parties. In 1976, SAP Gesellschaft mit beschränkter Haftung (GmbH) was founded as a sales and support subsidiary, and, the headquarters moved the following year to Walldorf, Germany. In August 1988, SAP GmbH became SAP AG, and public trading started on 4 November 1988. In 1995, SAP was included in the German stock index DAX and, on 22 September 2003, SAP was included in the STOXX Europe 50. Moreover, the company's official name became SAP AG (a public limited company) after the 2005 annual general meeting. Finally, on 7 July 2014, SAP announced it had changed its legal form to a European Company (Societas Europaea (SE)).

4.1.3 Products

SAP nowadays is famous for several reasons, in particular, the most important one is that it has become the world's leading Enterprise Resource Planning (ERP)[1] software vendor. Several products were developed during its history, for instance: "SAP R/1" (i.e. the first financial accounting system), "SAP R/2", "SAP R/3" (the first ERP developed by SAP based on a client-server architecture with a GUI), "SAP ERP" Central Component (ECC) and finally "SAP S4/HANA" (it

[1]ERP [13] is a category of business management software that an organization can use to collect, store, manage and interpret data from many business activities.

will be the ERP software used for the "Aruba S.p.A." project as will be seen in the next chapter) developed based on "SAP HANA"[2]. The just-mentioned ERP does not take into account the fact that since 2014, thanks to the partnership with IBM, SAP has started to work also on cloud ERP solutions, for instance, Concur is one of the most important cloud-based solutions that SAP has to offer. But the company does not limit itself only to ERPs it is also known for other important software and technologies like DW (like SAP BW/4 HANA etc.), EPM (like SAP BPC, SAP Analytics Cloud, etc.), Human Capital Management (HCM), Customer Relationship Management (CRM), Product Lifecycle Management (PLM), Supplier Relationship Management (SRM), Supply Chain Management (SCM) and Business Technology Platform (BTP) software.

4.2 SAP BW/4 HANA

This section will analyze the SAP BW/4 HANA DW by explaining how it works in terms of its architecture and what are its main components, from a technical point of view.

4.2.1 Introduction

SAP BW/4 HANA [21] is a DW solution with:

- Agile and flexible data modeling;
- SAP HANA-optimized processes;
- State-of-the-art user interfaces highly optimized for the SAP HANA platform;

SAP BW/4 HANA offers a managed approach to data warehousing. This means that prefabricated templates (building blocks) are offered for building a data warehouse in a standardized way.

4.2.2 Architecture

The architecture [21] of SAP BW/4 HANA can be described with the following image:

[2]SAP HANA [26] is an in-memory, column-oriented, relational database management system.

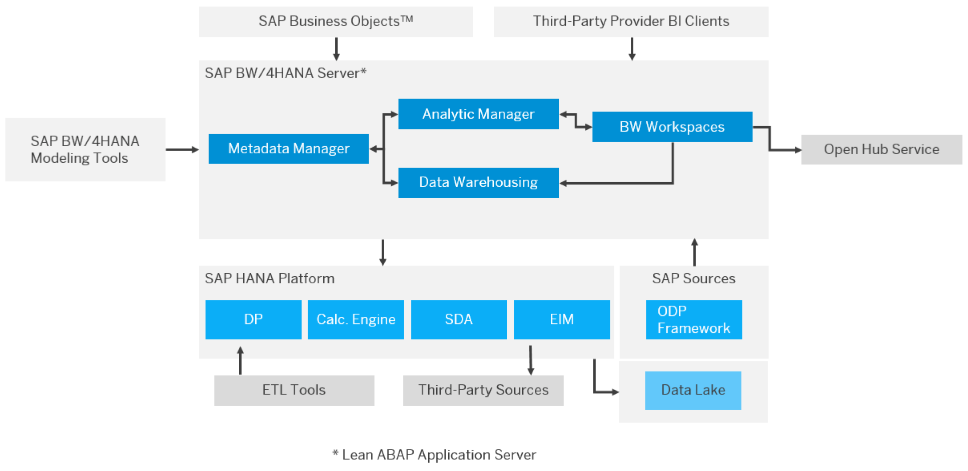

Figure 4.1: Architecture of SAP BW/4 HANA

As can be seen from the above image, there are a lot of features that characterize the architecture of SAP BW/4 HANA. First, this architecture is an open architecture, which means, that it enables the integration of external, non-SAP sources. In addition, it can be observed that for connecting third-party sources the SAP HANA Platform provides Enterprise Information Management (EIM) and Smart Data Access (SDA) interfaces, for allowing such a connection. Moreover, SAP data sources are connected through the Operational Data Provisioning (ODP) Framework, and, the ETL tool, can also be used to transfer data to SAP BW/4 HANA. Finally, the Analytic Manager is used for analytic functions and services while the evaluation can be performed with SAP Business Objects or BI clients also from third-party manufacturers.

4.2.3 Components

In this subsection, it will be presented the most commonly used and most important BW/4 HANA data components as well as logical components.

4.2.3.1 InfoObjects

InfoObjects [21] also known as business evaluation objects are the smallest units in BW/4 HANA. They are used to depict information in a structured form required for creating InfoProviders. They are divided into Characteristics, Key Figures, and Units.

4.2.3.1.1 Characteristics

Characteristics [21] are a type of InfoObjects that exists in three different versions:

- "Normal" Characteristics: "Normal" characteristics specify the granularity at which the key figures are stored in the InfoProvider. The most common examples are company code, product, customer group, fiscal year, period, or region;
- Time characteristics: time characteristics are characteristics such as date, fiscal year, etc.;
- Technical characteristics: technical characteristics are characteristics used for administrative purposes only within BW/4 HANA;

The information that characteristics store is stored in the so-called "Master Data". It represents data that remains unchanged over a long period and is always needed in the same way. Moreover, "Master Data" is divided into three different parts:

- Attributes: identifies the elements stored in the characteristic through ids;
- Texts: represent the string descriptions of the characteristic;
- Hierarchies: structure and contextualize the whole "master data";

The last thing to consider when dealing with master data is how it is updated, modified, and maintained over time. There are two main ways in how master data is updated, in particular with:

- Enhanced master data update: described by the below image, it is used for characteristics with attributes and/or text.

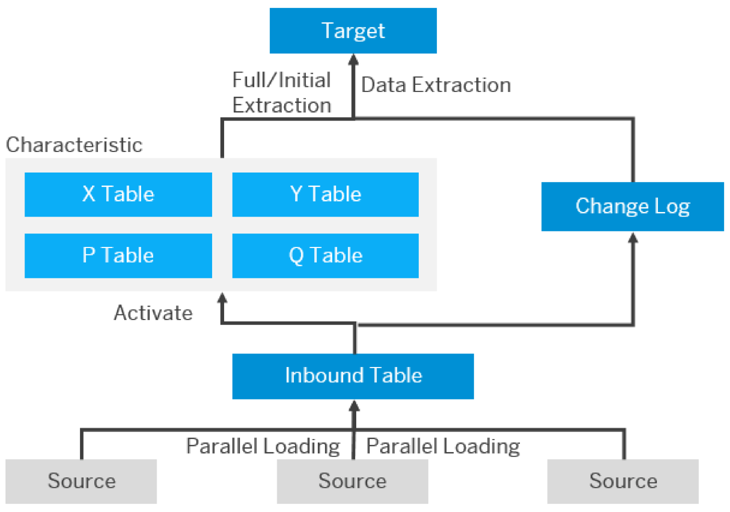

Figure 4.2: Enhanced Master Data Update

The main feature of this master data update is the fact that it enables parallel loading of data;

- Classic master data update: described by the below image, in this case, parallel loading is not possible. Here, data is loaded serially and directly to the tables that represent the characteristic;

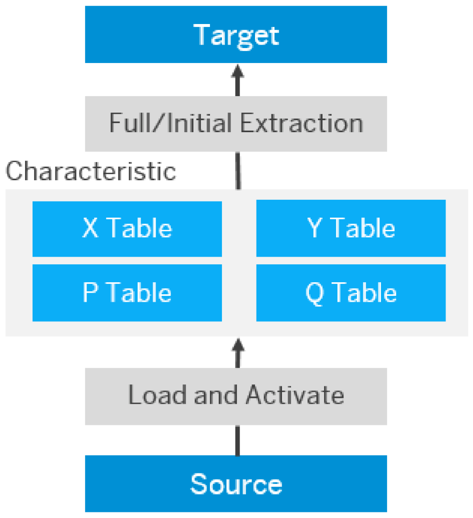

Figure 4.3: Classic Master Data Update

4.2.3.1.2 Key Figures

Key Figures [21] provide the values that are reported in a query. They can be quantity, amount, or number of items. Key figures have some important proprieties that influence the way data is loaded and how a query is displayed, for instance, some of those allow to set aggregation or specify the number of decimal places in a query.

4.2.3.1.3 Units

Units [21] are a type of InfoObjects used to assign meaning to the values of key figures. For instance, key figures of type amount are always assigned a "currency key" as a unit and key figures of type quantity also receive a "measurement" as a unit.

4.2.3.2 Reference InfoObject

A Reference InfoObject [21] is an InfoObject with the same technical properties, therefore, they inherit all the same features of the "normal" InfoObjects, such as the possibility of having different types of reference InfoObjects. To be precise, depending on the type of the "original" InfoObjects from which the reference one is created, there are different types of technical properties that are maintained. For instance:

- Characteristics: maintain the same data type, length, and master data;
- Key Figures: maintain the same key figure type, length, and master data;

So, Reference InfoObjects are nothing else than "pointers" to other InfoObjects.

4.2.3.3 InfoProviders

InfoProviders [24] are BW/4 HANA objects where data is loaded into or which display views of data (that can be analyzed with queries). There are InfoProvider types in which the data is stored physically and InfoProvider types that are only views on the data, in fact, InfoProviders can be either:

- InfoObjects;
- Advanced DataStore Object (ADSO);
- Open ODS Views;
- Composite Providers;
- Aggregation Level InfoProviders;

4.2.3.3.1 InfoObjects

InfoObjects with attributes or texts can be considered as InfoProviders in their own right.

4.2.3.3.2 Advanced DataStore Objects

ADSO [21] is the central object for data storage and data consolidation in the SAP BW/4 HANA system. It can contain both InfoObjects and fields and is particularly well suited to deal with frequent loading and large amounts of data. The structure of an ADSO consists of 3 tables at most: the inbound table, the change log, and the table of active data.

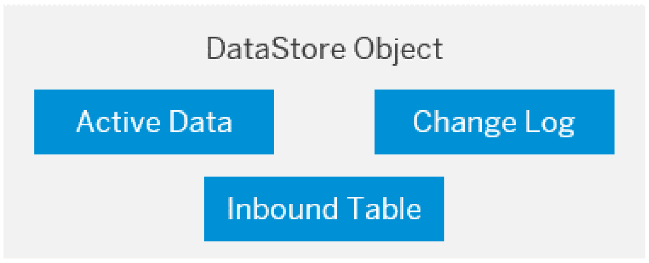

Figure 4.4: ADSO: Tables

The Inbound Table contains the data initially loaded. The Active Data (table) contains data that has been activated and substitute the inbound table for reading purposes when data is active. The Change Log (table) instead, contains the change history for the delta update. Another important aspect to consider when dealing with ADSOs are the so-called "modeling proprieties". They define how an ADSO can be used, and, therefore, a way to classify ADSOs. In particular, such "modeling proprieties" are:

- Standard DataStore Object: this modeling property is used for the majority of application cases and for reporting. Once activated, the ADSO can be represented as follow:

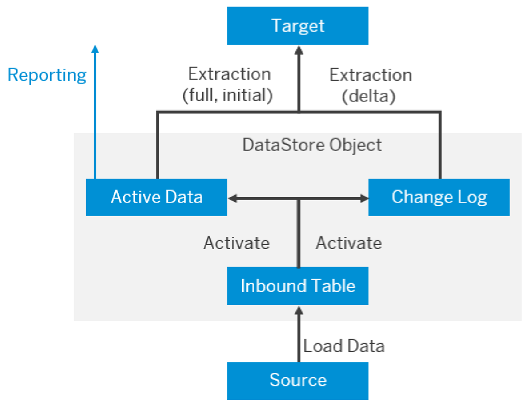

Figure 4.5: Standard DataStore Object

This means that:

  - Data is loaded into the inbound table;
  - Once data is activated it is transferred to the active data (table) and, if the write change log propriety is enabled also the change log (table) is populated;

  - Deletions apply only to requests not activated yet, while, rollbacks are only possible if the "write change log" property is set;
  - Data used for updating a further data target is extracted from active data or changed log data table depending if the extraction is full or delta;
  - When a query is executed the data that is accessed is the one inside the active data;

- Staging DataStore Object: this version of an ADSO has two different structures:
  - If the "Compress Data" property is set:

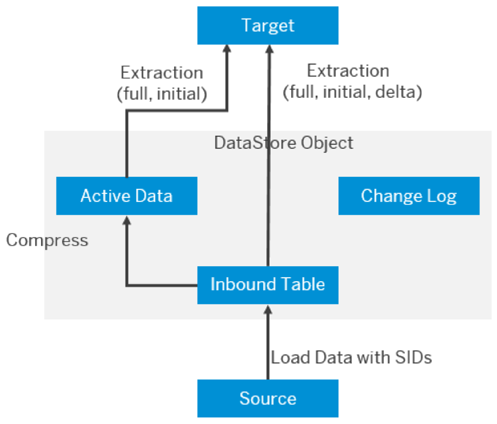

Figure 4.6: Staging DataStore Object: "Compress Data"

  - If the "Reporting-Enable" property is set:

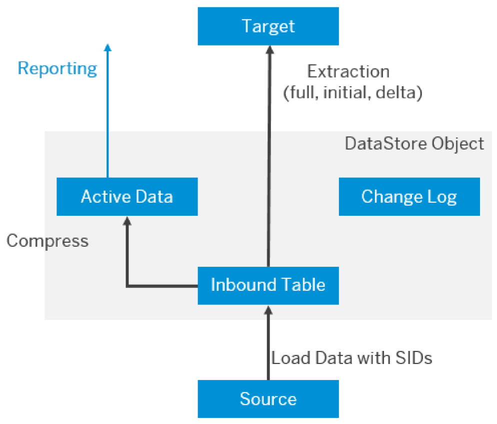

Figure 4.7: Staging DataStore Object: "Reporting-Enable"

The consequences of this are:

- Independently of the version data are always stored in the inbound table;
- The activation of the data makes, for the first case, a transfer of data from the inbound table to the active data table, while, in the second case data is not transferred but copied;
- Deleting data can only be done if it has not been activated yet;
- In the first case, the inbound table and active data table are accessed for extraction, while, in the second case full extraction only goes to the inbound table;
- For reporting in the first case, the ADSO can only be used for a very limited extent, while on the other hand, in the second case reporting only takes place on the active data table;

- Data Mart DataStore Object: this modeling property makes the ADSO optimized for reporting and analysis. The structure of the ADSO becomes the following:

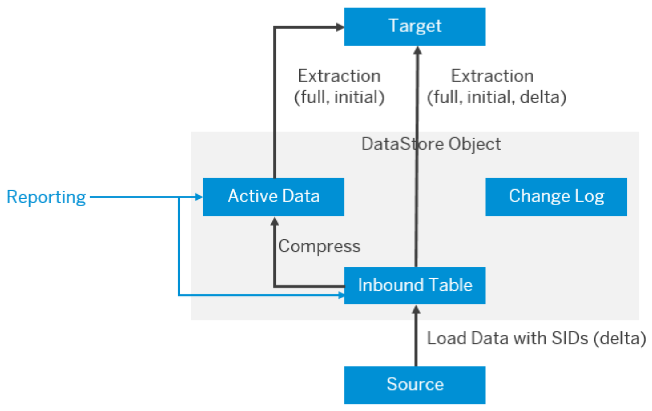
Target
Extraction (full, initial)
Extraction (full, initial, delta)
DataStore Object
Reporting
Active Data
Change Log
Compress
Inbound Table
Load Data with SIDs (delta)
Source

Figure 4.8: Data Mart DataStore Object

The implications are:

- Data is always loaded into the inbound table;
- When data is activated, it is copied to the table of active data and grouped according to the aggregation behavior;
- The full extraction reads data from inbound and active tables, while the delta extraction uses only the inbound table;
- Deletion can happen only for the request that has not been activated;
- For reporting, queries use both the inbound table as well the active data table;

- Update DataStore Object: this final version of an ADSO can be represented as follow:

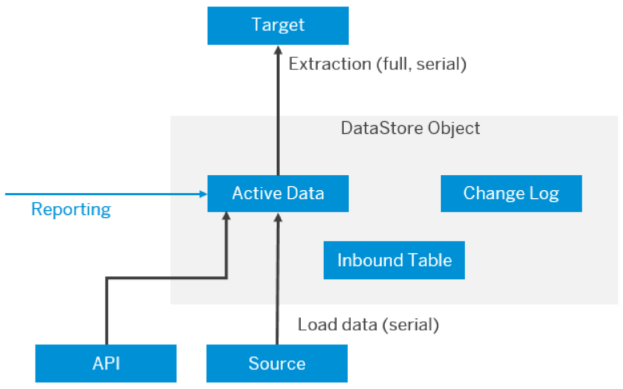

Figure 4.9: Update DataStore Object

The main characteristics of using this type of ADSO are:

- Data is loaded directly into the active data table;
- Activation is not possible;
- Deletion can be made using requests;
- Only full extraction is possible;
- Reporting uses only active data table;

4.2.3.3.3 Open ODS Views

While ADSOs are physical data models, Open ODS Views [21] are instead a type of InfoProviders that can be classified as Virtual Data models. In particular, Open ODS views enable to define data models for objects like database tables, database views, or DataSources (for direct access). Therefore, the main difference with ADSO is the fact that they access directly the sources of the data and they do not store anything, so there is no need to create InfoObjects. To define an Open ODS View the following components need to be specified:

- Semantics: facts, master data, and texts;
- Source type: where the data comes from. It can be either a BW/4 HANA Datasource object, ADSO, Database table, view or Transformation;
- Source Object: the table-shaped object on which the Open ODS View is based;
- Analytic Metadata: it must be specified for each field of the Open ODS View if it should be interpreted as a characteristic or as a key figure;

4.2.3.3.4 Composite Providers

Another InfoProvider that can be said to be a Virtual Data Model is the Composite Provider [21]. A Composite Provider, is, indeed, an InfoProvider that can merge data from InfoProviders with data from HANA views using union and join, or just merge data from InfoProviders and HANA views.

4.2.3.3.5 Aggregation Level InfoProviders

The last type of InfoProvider that is a Virtual Data Model and that will be defined is the so-called Aggregation Level InfoProvider [22]. Such InfoProvider has been designed for planning data manually or changing it using planning functions. An aggregation level represents a selection of characteristics and key figures for the underlying InfoProvider and therefore determines the granularity of the planning. It can be created more than one aggregation level for an InfoProvider, thereby modeling various levels of planning and hierarchical structures for example. Note, however, that, aggregation levels cannot be nested.

4.2.3.4 Datasource

A DataSource [21] is a set of logically-related fields that are provided for accessing and transferring data into BW4 HANA in a flat structure or multiple flat structures. DataSources supply the metadata description of source data. They are used to extract data from a source system and to transfer the data to the BW/4 HANA system. They are also used for direct access to the source data from the BW system. The following types of DataSource exist:

- DataSource for transaction data;
- DataSource for master data;
- DataSource for attributes;
- DataSource for texts;
- DataSources for hierarchies;
- DataSources for segmented;

4.2.3.5 Transformation

A Transformation [21] is a process that allows to consolidate, cleanse, and integrate data. It can be created between a source and a target. The source of a Transformation can be either a BW object or virtual object while the target of a Transformation is always a BW object. A Transformation must define how to convert the fields of the source into the format of the target, and, to do so it uses at least one Transformation Rule. In particular, what is used is the so-called "Rule Group" i.e. a group of Transformation Rules that contains one Transformation rule for each key field of the target. Each single Transformation rule defines how fields of the source of the Transformation will be used to obtain the fields of the target of the Transformation.

4.2.3.6 Data Transfer Process and Process Chains

A Data Transfer Process (DTP) [21] is an object that implements the transfer between two persistent objects (source and target) in the SAP BW/4 HANA system. A DTP is used to transfer data in SAP BW/4 HANA from one persistent object to another object, with "rules" defined with certain Transformations and filters. Data, with a DTP can be transferred either in 2 different ways:

- Full extraction mode: in this case the entire dataset of the source is transferred to the target;
- Delta mode: in this scenario, only data that was posted to the source since the last data transfer is transferred;

DTPs are mostly used with another important BW/4 HANA component which is the so-called Process Chain [21]. A Process Chain is a sequence of processes that are scheduled in the background for an event. Some of these processes trigger a separate event that can, in turn, start other processes. In particular, Process Chains can be integrated with DTP so that, by triggering a process chain, all the DTPs placed inside of it will be executed in the correct order.

4.2.3.7 Planning Sequences

A Planning Sequence [25] is a BW/4 HANA component that is made up of one or more steps. Each step can be either of two different types:

- Planning function type: in this case, a step of the planning sequence needs to define three things:

  - Aggregation Level InfoProvider over which the data, for the step considered, will be used;
  - Planning function to execute, that must be defined over the same previously defined Aggregation level;
  - Filter over the same previously defined Aggregation Level;

- Input template: on the other hand, this type of step of a planning sequence needs to define only two things:

  - Aggregation Level InfoProvider over which the data, for the step considered, will be used;
  - Filter over the same previously defined Aggregation Level;

The difference between the two types of steps is the fact that the first one is a concrete step because it will execute a planning function, while the other is used only for testing a planning function. Therefore, most of the time, a planning sequence is defined as a sequence of planning functions to execute, because, the input template type is rarely used.

4.2.3.7.1 Planning Functions

As seen previously, one of the components [25] of a planning sequence is the planning function. A planning function defines how the transaction data of a specific aggregation level is changed. To define a planning function, it is needed:

- The Aggregation Level: this is needed to identify which data the planning function must use to work;
- The type of the planning function: defines the operation that the planning function will execute over the data previously defined. It can be either user-defined or system-defined;
- The parameter values of the planning function: these are the parameters needed for the planning function to work, and, depend on the type of the planning function used;

In the case it is necessary a planning function with a custom type [23], must be first created such type of planning function, and then, the planning function itself. To do so it must be defined mainly three things:

- the parameters needed for the planning function type to work;
- the features that the function type needs to have, which can be:

  - Reference Data: if enabled this feature, reference data is needed when the planning function is executed;
  - Without Blocks: if this feature is enabled, all characteristic values in the transaction data can be changed;
  - Process Empty Records: if this is enabled all zero-value records are processed;

- the Object Oriented ABAP class that implements the operation defined by this custom planning function type. In particular, such a class needs to have the following characteristics:

  - Define the structures over which it works on. Those structures represent Aggregation Level InfoProviders among which there is always the Aggregation Level InfoProvider previously mentioned;

```
" Aggregation Level InfoProvider over which the Planning
    Sequence is defined
TYPES: BEGIN OF STRUCT_EXAMPLE_1,
    FISCPER TYPE /BI0/OIFISCPER,
    FISCVARNT TYPE /BI0/OIFISCVARN
    INFOPROV TYPE RSINFOPROV,
    ZEXAMPLE1 TYPE /BIC/OIZEXAMPLE1
END OF STRUCT_EXAMPLE_1.
TYPES: TABLE_EXAMPLE_1 TYPE
       STANDARD TABLE OF STRUCT_EXAMPLE_1.

"Additional Aggregation Level InfoProvider
TYPES: BEGIN OF STRUCT_EXAMPLE_2,
    FISCPER TYPE /BI0/OIFISCPER,
    FISCVARNT TYPE /BI0/OIFISCVARN
    INFOPROV TYPE RSINFOPROV,
    ZEXAMPLE1 TYPE /BIC/OIZEXAMPLE1,
    ZEXAMPLE2 TYPE /BIC/OIZEXAMPLE2
END OF STRUCT_EXAMPLE_2.
TYPES: TABLE_EXAMPLE_2 TYPE
       STANDARD TABLE OF STRUCT_EXAMPLE_2.

```

Code 4.1: Defintion of Structures

  - Define the interfaces to be implemented, which are:
    * IF_RSPLFA_SRVTYPE_IMP_EXEC or IF_RSPLFA_SRVTYPE_IMP_EXEC_REF depending if the Reference Data property is enabled or not;
    * IF_RSPLFA_SRVTYPE_TREX_EXEC or IF_RSPLFA_SRVTYPE_TREX_EXEC_R depending if the Reference Data property is enabled or not;
    * IF_AMDP_MARKER_HDB;

```
INTERFACES IF_AMDP_MARKER_HDB.
INTERFACES IF_RSPLFA_SRVTYPE_TREX_EXEC_R.
INTERFACES IF_RSPLFA_SRVTYPE_IMP_EXEC_REF.

```

Code 4.2: Defintion of Interfaces

  - Define, import and export parameters of the function for the IF_AMDP_MARKER_HDB interface;

```
CLASS-METHODS: FUNCTION_EXAMPLE
    IMPORTING
        VALUE(i_view) TYPE TABLE_EXAMPLE_1
        VALUE(i_view_ref_data) TYPE TABLE_EXAMPLE_1
        VALUE(i_EXAMPLE2) TYPE TABLE_EXAMPLE_2
    EXPORTING
        VALUE(e_view)  TYPE TABLE_EXAMPLE_1

```

Code 4.3: Import and Export Parameters

– Implement IF_RSPLFA_SRVTYPE_IMP_EXEC or IF_RSPLFA_SRVTYPE_IMP_EXEC_REF to filter the data coming from the Aggregation Level InfoProvider used to define the planning sequence (through the usage of the filter defined over such Aggregation Level InfoProvider);

```
METHOD IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~INIT_EXECUTION.
ENDMETHOD.
METHOD IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~EXECUTE.
ENDMETHOD.
METHOD IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~ADD_NEW_BLOCKS.
ENDMETHOD.
METHOD IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~GET_REF_DATA_SEL.
" Code for filtering the data of the Aggregation Level
   InfoProvider used to define the Planning Sequence
ENDMETHOD.
METHOD IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~FINISH_EXECUTION.
ENDMETHOD.

```

Code 4.4: IF_RSPLFA_SRVTYPE_IMP_EXEC_REF Implementation

– Implement IF_RSPLFA_SRVTYPE_TREX_EXEC or IF_RSPLFA_SRVTYPE_TREX_EXEC_R to:
    * Filter the data of eventually other structures that correspond to other (not the one over which the planning function type is defined over) Aggregation Level InfoProviders with the parameters defined in the current planning function type;
    * Call the function that will elaborate the input data and will also compute and write the result;

```
METHOD IF_RSPLFA_SRVTYPE_TREX_EXEC_R~INIT_AND_CHECK.
  E_TREX_SUPPORTED  = RS_C_TRUE.
ENDMETHOD.

METHOD IF_RSPLFA_SRVTYPE_TREX_EXEC_R~TREX_EXECUTE.
  DATA:
    L_R_SQL_SCRIPT   TYPE REF TO IF_RSPLS_SQL_SCRIPT,
          L_PROCEDURE_NAME TYPE STRING,
            L_T_IOBJ_PARAM   TYPE IF_RSR_PE_ADAPTER=>
   TN_T_IOBJ_PARAM.
        L_R_SQL_SCRIPT = CL_RSPLS_SESSION_STORE_MANAGER=>
   GET_SQL_SCRIPT_INSTANCE( I_R_STORE = I_R_STORE ).

  L_R_SQL_SCRIPT->GET_PARAMETER_VALUES(
    EXPORTING
      I_R_PARAM_SET = I_R_PARAM_SET
      I_PARA_NAME_FOR_PROCEDURE = 'FUNCTION_EXAMPLE'
    IMPORTING
      E_PROCEDURE_NAME = L_PROCEDURE_NAME
      E_T_IOBJ_PARAM = L_T_IOBJ_PARAM ).

  L_PROCEDURE_NAME = 'CLASS_NAME=>FUNCTION_EXAMPLE'.
```

```

    DATA:
    L_T_AGGR_VIEW     TYPE IF_RSPLS_SQL_SCRIPT=>T_AGGR_VIEW,
    L_S_AGGR_VIEW     TYPE IF_RSPLS_SQL_SCRIPT=>S_AGGR_VIEW,
    L_S_CHARSEL       TYPE RSPLF_S_CHARSEL.

  CLEAR L_S_AGGR_VIEW-T_CHARSEL.

  "Name of the Aggregation Level InfoProvider used to define
     the Planning Sequence
  L_S_AGGR_VIEW-AGGREGATION_LEVEL = 'EXAMPLE_1'.

  APPEND L_S_AGGR_VIEW TO L_T_AGGR_VIEW.

  R_S_VIEW-VIEW = L_R_SQL_SCRIPT->EXECUTE_SQL_SCRIPT(
    I_VIEW = I_VIEW
    I_VIEW_REF = I_VIEW_REF
    I_T_AGGR_VIEW = L_T_AGGR_VIEW
    I_T_IOBJ_PARAM = L_T_IOBJ_PARAM
    I_PROC_NAME = L_PROCEDURE_NAME
    I_R_MSG = I_R_MSG ).
ENDMETHOD.

```

Code 4.5: IF_RSPLFA_SRVTYPE_TREX_EXEC_R Implementation

– Implement IF_AMDP_MARKER_HDB which consists of defining the previously mentioned "called function" that will, through SQL code elaborate the input and write the result correctly;

```
METHOD FUNCTION_EXAMPLE BY DATABASE PROCEDURE FOR HDB
    LANGUAGE SQLSCRIPT.
  "Here must be added the SQL code for developing the logic
    of the Custom Planning Function
ENDMETHOD.

```

Code 4.6: IF_AMDP_MARKER_HDB Implementation

4.2.3.7.2 Filters

Filters [25] are the last planning sequence components that are used to restrict data to a certain business area, certain product groups, or certain periods for example. In particular, a filter is needed for each planning sequence step and therefore for each planning function to define the admitted values that the execution of the planning sequence will produce.

4.2.3.8 Queries

The last and most important component of SAP BW/4 HANA is the so-called Queries [21]. Queries represent table views on the data stored by an InfoProvider. A query can be defined on any InfoProvider and their purpose is to display the data as a navigable Pivot-Table. In particular, to define a query mainly must be defined five things:

- Fixed Values: this is the list of components of the InfoProvider over which the query is defined that must assume a certain value or values. When the query is shown such components cannot display values other than those defined here;

- Default Values: this is the set of components of the InfoProvider over which the query is defined that must assume a default value. That means that once the query is displayed the value or values for such components will be these default ones, but if the user wants to change what to display he or she is allowed to do it;

- Row Data: This section is used to define what is the list of components, that, are shown in the pivot table, to the user, in row mode (that means each row will display a different value);

	Sales	$
Product	**Category**	
Chains	Components	20000
Socks	Clothing	6000
Bid-Shorts	Clothing	4000
Shorts	Clothing	13300
Tights	Clothing	36000
Handlebars	Clothing	2300
Brakes	Components	8800
Mountain Bikes	Bikes	6300
Helmets	Accessories	17000
Lights	Accessories	21600
Locks	Accessories	29800

Table 4.1: Pivot-Table: Category and Product in row-mode

- Column Data: This section is used to define what is the list of components, that , are shown in the pivot table to the user in column mode (that means each column will display a different value);

Sales	$	$	$	$
Category	Components	Clothing	Bikes	Accessories
Product				
Chains	20000			
Socks		6000		
Bid-Shorts		4000		
Shorts		13300		
Tights		36000		
Handlebars	2300			
Brakes	8800			
Mountain Bikes			6300	
Helmets				17000
Lights				21600
Locks				29800

Table 4.2: Pivot-Table: Category in col-mode and Product in row-mode

- Free Data: This section is used to define what is the list of components, that, in the pivot table are not shown, but, can be displayed if the user wants either in a row or column mode;

4.3 SAP Business Planning and Consolidation

This last section will be reserved for a brief technical overview of one of the most important EPM products of SAP: SAP BPC. In practice, this last section other than defining what is BPC, will also describe its most important features, again, from an engineering prospective.

4.3.1 Introduction

BPC [20] provides everything to meet bottom-up and top-down financial and operational planning requirements through a single application and user interface. One of the latest versions of BPC is compatible with SAP BW/4 HANA. That means, BPC can support the HANA database. Moreover, it is also equipped with a user interface based on standard HTML5, and, can be configured in two ways:

- In an embedded configuration, Business Planning and Consolidation are integrated with SAP HANA-optimized processing, through which you can perform SAP HANA-optimized planning;
- In a standard configuration, all the functionality of the embedded version is used but, it is built for data warehouses like SAP NetWeaver that are not based on HANA, so it can work also with them;

4.3.2 Features

The BPC tool of SAP has a lot of features [20], that for simplicity can be summarized mainly into three categories:

- Business Process Flow (BPF): this feature consists of a guide for the final user through a sequence of tasks within a defined business process;
- Run Consolidation: this feature corresponds to the Consolidation module of BPC and enables to generate and manage consolidated data giving an accurate view of the status of the collected and consolidated data within an organization as well as a unified environment to perform consolidation tasks over the web;
- System Reports: this feature provides the default reports available in BPC;

In addition, all of those previously mentioned features are managed within an administration section which can be accessed only by administrators. In particular, in this last section it is possible to perform the following tasks:

- Environment Management: this task allows to manage environments i.e. a container that holds different planning scenarios in the form of models. With this task administrators can create new environments, delete environments and send email notifications to the users or teams defined in the system;
- Dimension Management: this task enables administrators to manage, generally speaking, dimensions i.e. a collection of related data members that represent one aspect of business. In particular, what can be done is the following:
  - Modify the name, description, type, etc. of a dimension;
  - Add, Remove a member of a dimension or Change the property values for an existing member;
  - Edit, Create, or Delete an existing or new dimension;
  - Add or Delete a hierarchy for a dimension;
  - Manage the security of a dimension that means deciding who's going to have access to it;
  - Create or Remove local dimensions;
- Model Management: with this feature, the administrator can manage (create, delete, modify) models. A model is a representation of the relationships, calculations, and data of an organization or business segment. Models can be categorized as 'planning models' or 'consolidation models'. Planning models are used for generic planning-related activities. Consolidation models are designed for performing consolidation tasks. Consolidation-type models must have specific components defined to fulfill legal consolidation requirements;

- Legal Consolidation: this feature allows finance departments of a corporation to consolidate numbers from subsidiaries and produce consolidated financial statements of this group of legal entities at the end of a financial period;
- Security Management: this allows administrators roles and teams so, that, there is controlled access to certain information;
- Management of Controls: this feature allows to define teams and the correct privileges to the final users;
- Set up Rules: this feature allows administrators to set up business rules to perform data calculations;
- Set up of BPF: this function enables to create and manage BPF templates. A process template is the specification of a single enterprise-wide business process, containing tasks that can span across the various modules of BPC. It defines the business processes that are presented to users as a prepackaged set of application tasks. Those tasks can be integrated not only with the built-in features previously mentioned but also with the usage of custom reports implemented with Excel and with the usage of the Analysis for Office plugin;
- Set up Journals: this function enables administrators to set up the journal component of BPC;
- Set up of Work Statuses: this function allows to lock a region or slice of data in a model;

5 The Aruba S.p.A. Project

This chapter will describe what has been done in the "Aruba S.p.A." project, which means: what problems needed to be solved for the company, and what solution has been proposed to them and implemented. In particular, the chapter will begin with a small introduction to what the company does, and who is it. Then it will be followed by a detailed description of the working methodology as well as the main problems that the company is facing with its definitions. Moreover, there will be a description of the solution proposed, and, finally, an illustration of some of the implementations developed to tackle the issues previously mentioned.

5.1 The Aruba S.p.A. Company

Aruba S.p.A. [2] is an Italian leader company that offers data center, web hosting, email, and domain name registration services. Its main products vary from Web hosting with Microsoft or Linux servers to Certified Electronic Mail as well as Cloud Computing servers. The history of Aruba dates back to 1994, when in Florence, "Technorail S.r.l." was founded. A couple of years later, this brand was replaced by "Aruba.it" which first launched free internet dial-up service on the market. In 2003, a web farm in Arezzo was inaugurated as the main center where the web servers are hosted. A year later "Aruba.it" was transformed from a limited liability company to a joint-stock company and changed its name to "Aruba S.p.A." as it is known nowadays. From this date on, its expansion continue to rise by acquiring several other Italian and not Italian companies and by becoming in 2014, thanks to a collaboration with "Ducati Corse", the main sponsor of the Superbike team called "Aruba.it Racing Ducati".

5.2 Working Method

Before introducing the problems related to the "Aruba S.p.A." project and their solutions, let's first analyze and describe how PwC managed to define a working strategy to carry out its tasks with the client smoothly. One of the most difficult things when dealing with a client's project is understanding what the customer wants. Therefore, if a strategy is not used, to get the correct requirements, then, there is the actual risk of misunderstanding the requests of the client and as a consequence, wasting time, energy, and especially money on implementing features that are wrong and not necessary. For these reasons, the PwC company where I worked decided to define mainly three teams inside itself, for this project, that work on different aspects. In particular, those teams were:

- "Change Management" team: this team has the main goal to drive "Aruba S.p.A." to the new solution that will be implemented, and, so, help them to adopt a new way of working. Finally, it has also the goal to describe the impact of certain changes of requirements requested by the client itself;
- "S4" team: this is a more practical team because it is the team that works on the SAP S/4HANA ERP technology. Therefore, their tasks are to understand the problem of the customer, design a solution for those problems, and finally, implement such a solution. In practice, this team worked on how to store data inside the SAP S/4HANA ERP by defining standard conventions, rules, etc. to save the data;
- "BPC" Team: Similar to the "S4" team, also this last one, is a team that works on a technology, mainly the EPM tool SAP BPC. But, because the SAP BPC EPM tool works on top of a Data Warehouse, the "BPC" Team works also on the Data Warehouse part of the project using the SAP BW/4 HANA Data warehouse technology. Again, the goal here, as suggested by the previous technical team, is to collect the requirements, design a solution for the problems defined and finally implement such a solution. Practically speaking, this team has the goal to manage the data obtained from the sources (for instance the SAP S/4HANA ERP) and produce with them useful business results that allow the client to improve its productivity as well as its business;

In addition, all of these teams mainly worked with the "Aruba S.p.A." management control team. So, the interactions between those teams can be visualized simply by the following image:

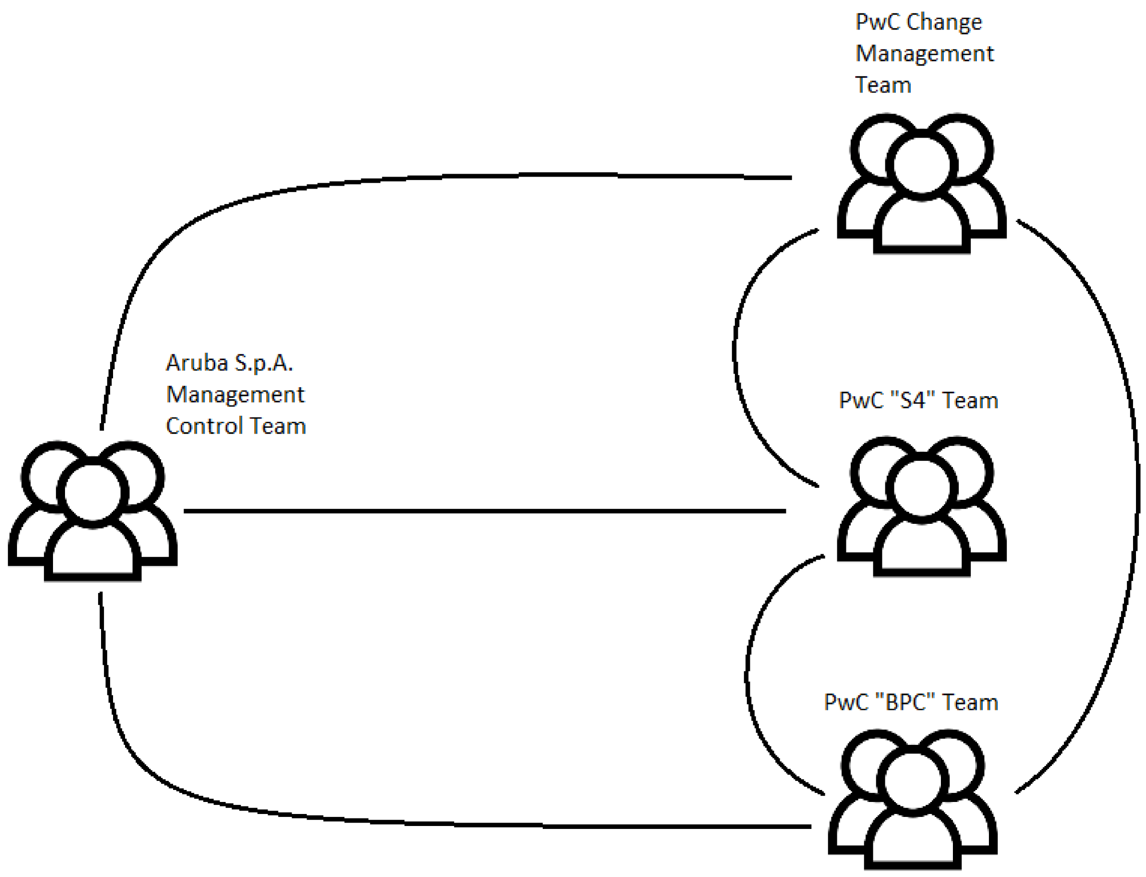
PwC Change Management Team
Aruba S.p.A. Management Control Team
PwC "S4" Team
PwC "BPC" Team

Figure 5.1: "Aruba S.p.A." Project: Main Teams

As can be seen, not only there is communication between the client's team and the PwC's teams, but also there is a conversation between the PwC teams themselves. Furthermore, as will be seen later, another important team of the "Aruba S.p.A." company will interact in the development of the project, in particular in the Allocations part. Such team, called "Team Data Center" will help the construction and design of the Allocation Engine[1] by interacting mainly with the PwC "BPC" Team and the "Aruba S.p.A." management control team as suggested by the following image:

[1]The Allocation Engine is the solution that will be used for solving one of the problems that Aruba is currently facing.

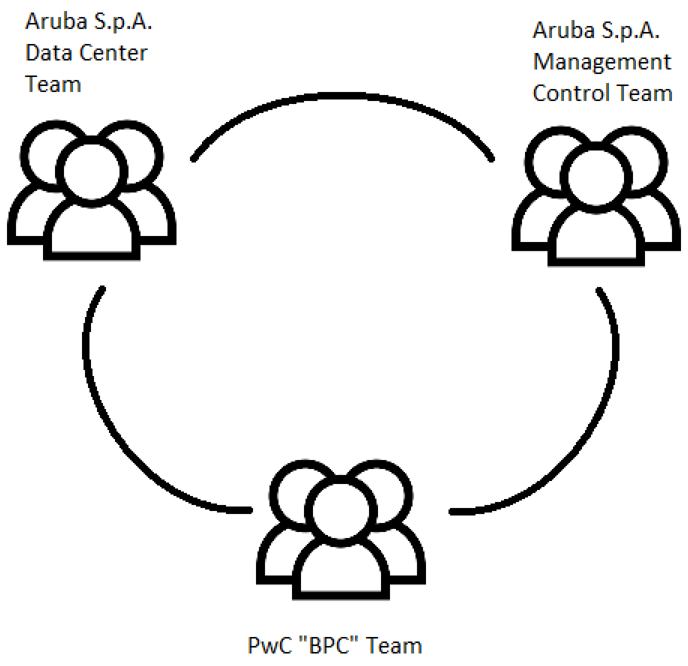

Figure 5.2: "Aruba S.p.A." Project: Allocation Engine Teams

Moreover, for overcoming the issues previously mentioned (waste of time, energy, money, etc.) each team of the PwC side decided to adopt an ad-hoc strategy for their purposes, which for the "BPC" team where I worked, was divided into the following phases:

- Analysis & Proof of Concept (PoC): this phase is used to understand the client's problems and collect all the requirements that are needed to build the solution. To do so, several existing solutions, of very similar projects, will be re-used for building simple mock-ups to provide a PoC to the customer for getting itself familiar with what to expect at the end, and eventually, if the requirements were misunderstood, change the demo to meet the new requirements requested by the client. Finally, during this process will be also written a document called Business Blueprint (BBP) where all the problems, requests, and requirements of the client will be documented;
- BBP Review: this phase starts almost at the end of the previous phase and consists of sharing the BBP document with the client for getting approval to agree on what will be implemented. Eventually, such a document is modified to meet the new requirements of the client or for sharing more details of the solution;
- Development: This phase starts almost at the same time as the first phase and consists of developing the PoC until the BBP is not approved by the

customer and then, developing the actual solution, starting from the PoC when the BBP has been approved;

- Functional Unit Testing (FUT) and System Integration Testing (SIT): this phase consists of testing the application developed. In particular, as suggested by the name two types of tests will be applied:
  - FUT [12]: this type of test is done to ensure that the application's overall functionality meets the business requirements. In particular, each element of an application is tested to ensure that the element works as expected;
  - SIT [27]: this type of test involves the overall testing of a complete system of many subsystem components or elements. They consist, initially, of the "process of assembling the constituent parts of a system in a logical, cost-effective way, comprehensively checking system execution, and including a full functional check-out." And, then, "verifying that the system meets its requirements, and validating that the system performs following the customer or user expectations";
- Training: during the previous test phase, will also start the training for using the application by the client, where will be explained how the application works and how it needs to be used;
- User Acceptance Tests (UAT) [1]: this phase consists of one of the last steps before releasing the final application, where, the so-called User Acceptance Tests will be applied. They consist of tests conducted to determine if the requirements of a specification or contract are met. In particular, they are conducted by the end/final user, in this case by the client itself;
- Review of BBP: in this phase, after all the previous tests have been completed, the BBP document will be reviewed with the client in case something during the development went wrong or must be modified;
- Go Live and Support: This is the last phase and consists of delivering the application to the client and supporting itself, eventually, once the application has been delivered by fixing eventual errors and/or bugs;

Thus, as can be seen, the process is quite complicated. Moreover, this does not consider eventual future improvements. Most of the time what happens is the fact that to elevate the solution proposed to the maximum level possible, there could be the need for the client to upgrade the solution proposed with a new application or eventually add new, ad-hoc functionally that were not discussed during the collection of the requirements of the previous application but where, for instance, only suggested by the consulting company in the BBP document itself. Therefore, if something like this happens, then, this entire process will start again to create a new solution.

5.3 The Main Problems

As already said in one of the previous sections the "Aruba S.p.A." company is an organization that comprises several other groups. Therefore, dealing with data inside a very distributed and fractured environment is not a very easy task, especially when it carries economic value. For these reasons, inconsistencies are a commonality in this scenario, and as a consequence, operations like managerial consolidation or the computation of revenue and costs of a bill of materials are very hard to perform. Thus, the problems that the "Aruba S.p.A." company is facing are:

- Data Integration;
- Business Management which comprises:
  - The creation of the Managerial or Management Consolidation;
  - The computation of costs and revenues of the internal Bill of Materials;

5.3.1 Data Integration

One of the problems that "Aruba S.p.A." is facing is the so-called Data Integration. Data Integration [7], in fact, consists, generally speaking, of the problem of combining data residing in different sources and providing users with a unified view of them. In particular, it is very common in a lot of situations especially in the commercial case when, for example, two or more companies need to merge their data. Moreover, one of the main difficulties in facing Data Integration is combining data from heterogeneous systems and transforming them into a single and coherent data store that provides synchronous data across a network of files for clients.
For what concerns the particular case of the "Aruba S.p.A." company the issue is manifested because the organizations that compose the company itself use different systems to manage the data. In particular, those organizations are using several different ERPs and management systems to manage their data. This makes it difficult to perform simple business operations because, not only, those data, are not unified under a common system but also each distinct ERP and management system, that, the organizations are using, utilize different naming

conventions, template, etc. to store the same data. Moreover, the issue is also so problematic because of the variety of the ERPs and management systems used across the organization, speaking of which, Aruba itself uses the following classification to identify the different organizations depending on the systems that they use:

- companies that uses SAP S/4HANA as ERP;
- companies that use proprietary management systems called "Onda" and "Oceano";
- companies that use do not use either SAP S/4HANA or "Onda" or "Oceano";

Thus, a common, central, and unified solution is needed to maintain all the data consistent.

5.3.2 The Computation of Costs and Revenues of the Internal Bill of Materials

This issue is a Business Management-related problem, but compared to the other one represents a more high-level problem. So, the issue consists of mainly two sub-problems. The first one includes, given the revenues of a sold final product, computing the revenues of the elements that are used to build such a product. On the other hand, the second problem involves, given the costs of the elements that are used to build a final product, computing the costs of the final product itself. In particular, these computations rely on the concept of a Bill Of Materials (BOM) which consists [4] of a list of the raw material, sub-assemblies, intermediate assemblies, sub-components, parts, and the quantities of each needed to manufacture an end product. Speaking of which, in the particular instance of the "Aruba S.p.A." project is a multi-level Bill Of Material, mainly divided into four levels:

- Level 1 (L1) or Packets: which corresponds to the final products that the company sold;
- Level 2 (L2) or Materials: which are the components that are used to build the previous level, but can also be sold alone;
- Level 3 (L3) or Platforms: which corresponds to a base unit for providing a service on the market and which groups cost;
- Level 4 (L4) or Elements of Cost: these are all the materials used for defining the infrastructural costs like maintenance, energy, collocation, etc;

Therefore, given this information, the final goal is to understand mainly:

- How to compute, given the revenues of a certain Accounting Nature and/or Business Line[2], the revenues of either the first or second level of the BOM;
- How to compute, given the costs of an Accounting Nature and/or Business Line the corresponding costs of all the levels of the BOM;

To make things clear let's consider the following examples[3]:

- For the first case: suppose of having the following business data:

Accounting Nature	Business Line	Other Data Model Dimensions	L1 - Packets	L2 - Materials	Value (Revenue)
R00000001	BL01	Other Values	N/A	N/A	2540€

Table 5.1: "Aruba S.p.A." Project: BOM Revenues Problem

  So, the goal is to assign to either both the two dimensions L1 and L2, or only one of them a value to attach the revenue to either one of the two levels or a combination of them, respectively;

- For the second case: suppose of having the following business data:

Accounting Nature	Business Line	Other Data Model Dimensions	L1	L2	L3	L4	Value (Cost)
C00000001	BL01	Other Values	N/A	N/A	N/A	N/A	4890€

Table 5.2: "Aruba S.p.A." Project: BOM Costs Problem

  Again the goal is to land the cost in one or all the levels of the Bill of material by correctly assigning the correct value for each dimension;

So, as can be understood, solving such a problem manually by using "Offline" data, Excel files, and inputting data manually each time, as was done by Aruba, is very complex and requires a lot of time. Thus, an automatic, reliable, consistent solution is required to solve the problem.

[2]Two dimensions of the Data Model.
[3]The Data used here is fake for privacy reasons.

5.3.3 The Creation of the Managerial or Management Consolidation

Another problem that Aruba is facing is the so-called creation of managerial or management consolidation. This is a more business management-like problem compared to the data integration one, but, still, it requires an IT-based solution to be solved. Such a problem, i.e. the creation of the Managerial or Management Consolidation, generally speaking, consists [14] of presenting aggregated financial results by the organization's strategic business groups such as geographical location, business divisions, products, services, and brands. Compared to the legal or statutory consolidation counterpart, management consolidation is prepared to provide users with a different perspective on the company's performance, while on the other hand, legal consolidation is used to present entity-level and group-level performance. Another important aspect of why management or managerial consolidation is useful is the fact that it provides a deeper understanding of the company's business performance by strategic segments, with a level of detail that will be important for decision-making and controlling purposes. In summary, the main purposes of the Managerial or Management Consolidation are:

- Analyze the performance of different segments of the company;
- Analyze the Profitability by business unit or segment;
- Apply guidelines of internal management;
- Perform allocations of revenue or costs;
- Get profitability by segment;

In practical terms, to obtain the Managerial or Management Consolidation some rules must be applied, which makes this operation very difficult if not automated. Most of the time, the rule that needs to be used is the so-called "Elimination of Inter-Unit Transaction". So, to better understand how this work, let's consider the following example [6].
Suppose that a group company has a management consolidation group structure with the following three segments: Trucks, Bus, Cars and, with "E1 - Smart Wagon Europe", "E2 - Smart Wagon US" and "E3 - Smart Wagon UK" as legal entities. In addition, suppose, that exists, also, the following pairs of entities for management consolidation:

	Bus	Truck	Cars
E1	Yes	Yes	
E2		Yes	Yes
E3	Yes		Yes

Table 5.3: Management Consolidation Example: Entities

As a consequence, the management consolidation entities are: E1_BUS, E1_TRUCK, E2_TRUCK, E2_CARS, E3_BUS, E3_CARS. Finally, assume that the next intra-segment sales happen:

Gal Account	Consol Unit	Partner Unit	Time	Amount
IC Sales	E1_BUS	E3_BUS	2020001	1000

Table 5.4: Management Consolidation Example: Intra-Segment Sales

Therefore this will inflate the sales numbers of Segment BUS. So, such intra-segment transactions for BUS Segment must be eliminated.
In the particular instance of the "Aruba S.p.A." the problem consisted of the fact that to consolidate their data they have to go through several different not uniform sources, and then if they were able to put all the data together, they can consolidate them. Speaking, this operation before the PwC intervention was realized only for a very few organizations that compose the "Aruba S.p.A." company, and in particular, only the ones that have the same source for the data. Moreover, the operation was performed "by hand" making it still tedious and annoying. Thus, an automated, more general solution is required to perform such an operation.

5.4 Solution Proposed

Now that all the problems of the "Aruba S.p.A." project have been exposed, it is time to describe the solutions that have been proposed and implemented for each one of them. However, before doing so, let's introduce the commonalities that all of those implementations have. In particular, the most important commonality that all of the following implementations rely on is the system design or infrastructure adopted. What can be guessed, by what has been explained in

Chapter number four, is the fact that all of the problems previously mentioned will be solved by using several common SAP technologies, in particular:

- The ERP "SAP S/4HANA": already used by "Aruba S.p.A." for some of their organizations but now extended to all of them;
- The DW "SAP BW/4 HANA": this is mainly necessary for applying to the data collected: filters, logic, and underneath operations to hide calculations to the final user;
- The EPM "SAP BPC": this tool is mainly necessary for consolidating data, but adapted to be used also for solving other problems, in combination with Microsoft Excel and the "SAP Analysis for Microsoft Office (AFO)"[4] add-in of it, as a final user interface for making the operations necessary;

Therefore, the software infrastructure that those solutions require can be described in the following picture:

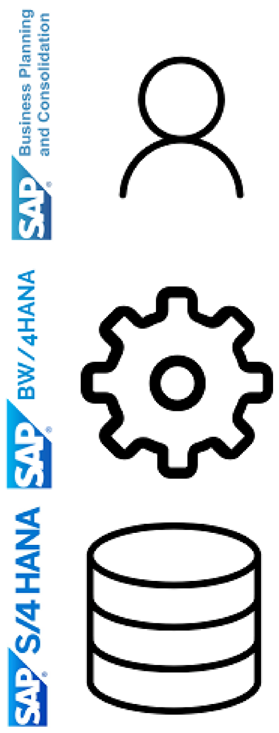

Figure 5.3: "Aruba S.p.A." Project: Software Architecture Solution

So, data flows from the ERP "SAP S/4HANA" to the DW "SAP BW/4 HANA" and finally to the EPM tool "SAP BPC". Moreover, this infrastructure does not

[4]SAP AFO [30], is a Add-In that allows multidimensional analysis of OLAP sources.

limit, as will be seen later, to accept as sources only the ERP itself, but also other external sources, making the DW more flexible, at the cost of elaborating more data.

5.4.1 Data Integration

This subsection will cover the description of the solution for the Data Integration problem from the technical side. Particularly, it will focus on two main aspects of the solution: the Logical Layers and the Data Models adopted.

5.4.1.1 Logical Layers

The solution to the Data Integration problem mainly relies on the use of the SAP BW/4 HANA and SAP S/4HANA as the two DW and ERP technologies used. Particularly speaking, this solution relies on the adoption by all the organizations that form "Aruba S.p.A." of the SAP S/4HANA as a new technology that will, in the future substitute or partially substitute their internal management systems or ERPs. Therefore to facilitate the switch from the current technologies used to the new SAP S/4HANA very hard work on the way data are stored has been done by the "Team S4" of PwC. In practice, this team has worked on remodeling how the data are stored in the SAP S/4HANA ERP, by the organizations that have been using it, to help the transition and loading on the SAP BW/4 HANA DW of not only those remodeled information but also, the data coming from the companies that use other sources to reduce as possible the effort to merge all of those data. However, even if the previously mentioned work has highly reduced the complexity of the solution, still it needs a bit more effort to solve completely the problem. In particular, to integrate correctly the data, of the different sources, into the SAP BW/4 HANA DW, the "Team BPC" developed a solution that consists of using two main logical layers:

- The BPC Layer: this layer, built on the SAP BW/4 HANA DW, is the layer that is used for the final application, and, contains all the correct data for making analysis, allocations, managerial consolidation, and building the simple reporting;
- The Staging Layer: This layer, also built on the SAP BW/4 HANA DW, is the level that collects the data of all the sources, refines them, and then moves them to the BPC layer. To be precise, only the data of the companies that use:
  - SAP S/4HANA as ERP already;

– "Onda" or "Oceano" as their Management Systems;

will be loaded into the staging layer. The third class of companies that instead neither use both of those technologies will load their data directly on the BPC Layer, assuming that they have the data in the correct format and already cleared;

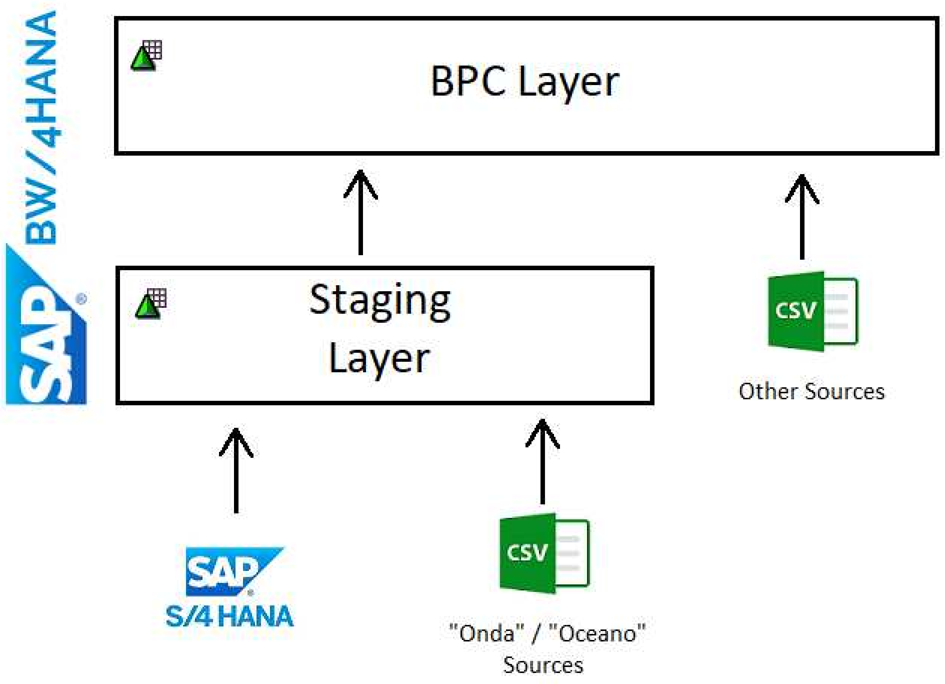

Figure 5.4: "Aruba S.p.A." Project: Logical Layers

Finally, before moving the data from the Staging Layer to the BPC Layer, some mappings will be applied to have a consistent and coherent population of the data in the BPC Layer.

5.4.1.2 Data Models

The description of how the previous two layers interact is not sufficient to have a clear view of what is happening under the scenes and requires a more detailed and "in-practice" overview of what is going on. Each Logical Layer uses one or more Data Models i.e. a collection of technical SAP BW/4 HANA objects to implement the solution for the Data Integration problem. Particularly speaking, three Data Models were used: two for the Staging Layer and one for the BPC Layer.

For the BPC Layer, the data model consists mainly of the following dimensions:

- Level 1 (L1) or Packet: provides details of the final products that the company sold;
- Level 2 (L2) or Material: identifies the components that are used to build the previous level;
- Level 3 (L3) or Platform: used to contain the list and details of the Platforms;
- Level 4 (L4) or Elements of Cost: represent the list and the details of the Platforms;
- Client: used to identify the Clients of "Aruba S.p.A.";
- Business Line: represent the Profit Centers;
- Infrastructure: is the Data Centers Infrastructure managed by the Aruba group;
- Project: identifies the individual orders of various activities;
- Organizational Department: consists of the lowest level for identifying groups and/or teams inside Aruba;
- Legal Entity: contains the SAP S/4HANA ERP, "Onda" or "Oceano" Management Systems and other sources companies;
- Chart of Account Intended: chart of accounts used for management reporting;
- Accounting Nature: used to carry out accounting records;
- Intercompany: the Trading Partners of the Aruba Group;
- Currency: identify what is the currency of the values exposed in reporting;

In addition, other two classes of dimensions are used in this level:

- Technical Dimensions: This set of dimensions are used for technical purposes to make the application (SAP BPC) work;
- Drivers: This set of dimension are a replica of the previous list, and their only purpose is to be used for the Allocations[5];

For the Staging layer, instead ,two different data models are required:

- SAP S/4HANA ERP Data Model, composed of the dimensions:
  - Company Code: represent the all the information regarding a company;

[5]In the Computation of Costs and Revenues of the internal Bill of Materials this concept will be explained better.

  - Cost Center: comprise Platforms, Organization Department and Infrastructure;
  - Profit Center: identifies the Business Lines;
  - Cost Element: represents the Accounting Nature;
  - Customer: identifies the clients of the Aruba group;
  - Material Number with Plant: represent the list of packets and materials used by Aruba;
  - Work Breakdown Structure (WBS): identifies the projects, clients, packets and materials associated to a project;

- "Onda" or "Oceano" Management System composed of the following dimensions: Accounting Nature, Business Line, Client, Intercompany, Organization Department, Project, Level 1 (L1) or Packets, Level 2 (L2) or Materials;

Finally the last thing to explain is how the BPC Layer Data Model is populated given the Data Models of the Staging Layer. So the process can be explained as follow:

- First the data from SAP S/4HANA will be loaded into the dimensions of the corresponding data model;
- Second, the dimensions of the "Onda" or "Oceano" Management Systems will be loaded into its Data Model as well, and, each value stored will be mapped to a corresponding element of the SAP S/4HANA ERP Data Model;
- Third, the Data loaded into the SAP S/4HANA ERP Data Model and the mapped data stored in the "Onda" or "Oceano" Data Model will be transferred correctly into the BPC Layer according to this table:

Source		**Destination**
SAP S/4HANA ERP Data Model	**"Onda" or "Oceano" Management Systems**	**BPC Data Model Dimension**
Material Number with Plant and WBS	Level 1 (L1) or Packets	Level 1 (L1) or Packets
Material Number with Plant and WBS	Level 2 (L2) or Materials	Level 2 (L2) or Materials
Cost Center	N/A	Level 3 (L3) or Platforms
Customer and WBS	Client	Client
Profit Center	Business Line	Business Line
Cost Center	N/A	Infrastructure
WBS	N/A	Project
Cost Center	Organization Department	Organization Department
Company Code	N/A	Legal Entity
Cost Element	Accounting Nature	Accounting Nature

Table 5.5: "Aruba S.p.A." Project: Data Flow

In conclusion, all the other dimensions (Level 4 (L4) or Elements of Cost, Currency, etc.) will be either manually populated directly in the BPC Layer or, generated with ad-hoc mappings.

5.4.2 The Computation of Costs and Revenues of the Internal Bill of Materials

This subsection will be used to present the solution, namely the Allocation Engine, to the problem of Computing the Cost and Revenues of the Internal Bill of Materials of "Aruba S.p.A." as well as for making other simple allocations that the Management Control team of "Aruba S.p.A." may need. The subsection will start with a high-level description and technical representation of how the "Allocation Engine" works. Then, it will also be presented how the Engine has been implemented in practice.

5.4.2.1 Allocation Engine

In this part will be presented the solution for the problem of computing the costs and revenues of the internal bill of materials of the "Aruba S.p.A." company. Particularly speaking, will be presented both a technical and a high-level description as well as the implementation and an execution example.

5.4.2.1.1 High-Level Description

The definition of what is the "Allocation Engine" resides intrinsically in the "Allocation" term. Generally speaking in this context, "Allocation" means to "redistribute values of data". Therefore, the "Allocation Engine" is a tool that allows to do such an operation. However, this definition is quite hard to catch if it is the first time that this term is heard. So, to better understand what the "Allocation Engine" is, let's first introduce with which technologies it is built. In particular, the "Allocation Engine" is designed entirely on the DW SAP BW/4 HANA and, for this project, it is built on top of the BPC Layer, so that it can work with the Data Model dimensions of such layer. Moreover, for what concerns the final user perspective, access to the Allocation engine is done through the EPM tools: SAP BPC and AFO. To be precise, after opening an Excel-AFO- sheet, by accessing through SAP BPC, the "Allocation Engine" can be used in its entire functionalities, if the opened sheet has access to it.

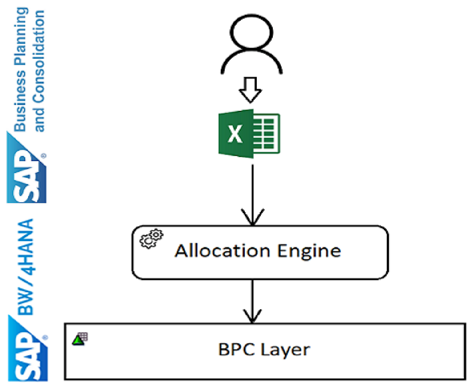

Figure 5.5: "Aruba S.p.A." Project: Allocation Engine High-Level Representation

5.4.2.1.2 Technical Description

Until now it has been described, in a high-level way how the Allocation Engine works. The next step is to understand how it has been implemented. So, the first thing is to analyze what the "Allocation Engine" consists of in practice. In particular, it is an SAP BW/4 HANA technical object i.e. a Planning Sequence. Such a planning sequence is composed of two custom planning function that works with two other SAP BW/4 HANA technical objects, namely two Aggregation Level InfoProviders.
Those two Aggregation Level InfoProviders are built on top of two ADSOs:

- The first one, called "Allocation Engine ADSO" contains two types of data:
  - "Data to be Allocated": A set of data that needs to be redistributed depending on some other data called "Drivers" and the "Allocating Dimension" selected;
  - "Drivers": A set of data that defines in which way, i.e. through percentages, how the "Data to be Allocated" must be redistributed depending on the "Allocating Dimension" selected;

- The second one, called "Allocation Rules ADSO" contains the so-called "Allocation Rules". An "Allocation Rule" defines how a single Allocation operation must be performed;

Instead, the structure of those two ADSOs just described can be represented as follow:

- For the "Allocation Engine ADSO" is:

Data Model Dimensions Section			**SignData Section**
Data Model Dimension 1	**Data Model Dimension i**	**Data Model Dimension n**	**SignData**

Table 5.6: "Aruba S.p.A." Project: Allocation Engine ADSO Structure

where:

- "Data Model Dimensions Section" represent the list of combination of values of the Data Model Dimensions;
- "SignData Section" represents a number that depends on the values of the Data Model dimensions and can be either: a cost, a revenue, a quantity, or a percentage depending on what a single data (row of the table) represents;

Notice that, each row of the above table is either a "Data to be Allocated" or a "Driver". Furthermore, each row of such a table is data of the "Allocation Engine ADSO";

- For the "Allocation Rules ADSO" is:

IDs			**Data to Allocate**		**Drivers**		
Cycle	Rule	Allocating Dimension	Data Model Dimen-sion 1	Data Model Dimen-sion n	Data Model Dimen-sion 1	Data Model Dimen-sion n	Enable

Table 5.7: "Aruba S.p.A." Project: Allocation Rules ADSO Structure

where:

- "IDs Section": this section is used to specify two IDs:
  - The Allocation Cycle ID: this ID is used to group similar Allocation Rules that must be executed, in a certain order, depending on their Allocation Rule ID;
  - The Allocation Rule ID: this id is used to identify an Allocation Rule (i.e. a single row of the previous table or, a single element of the corresponding ADSO) inside the Allocation Cycle considered;
- Allocating Dimension Section: this part is reserved, for specifying, for a single rule, which dimension is going to be allocated, which means, given the set of "Data to be Allocated" and the "Drivers" to use to perform the Allocation, how the "SignData" of the "Data to be Allocated" is broken down depending on the values of the "SignData" of the "Drivers" and the values of the "Drivers" for the Data Model dimension that corresponds to the value inside "Allocating Dimension";

- Filters For Data to be Allocated Section: this part is reserved for specifying which data to retrieve from the "Allocation Engine ADSO", to define the set of data to be allocated. In other words, this specifies the set of filters that must be applied to obtain such a set and so, each column reserved for such a section can assume as value one of the following types:
    * A Master Data value for that dimension, which could also be a node[6] of the hierarchy for that dimension;

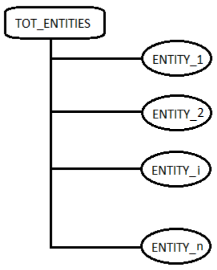

Figure 5.6: "Aruba S.p.A." Project: Hierarchy Nodes Example

    * The Empty value indicated with "#";
    * The Value "XX" which means any value;

- Filters For Drivers Section: this part is reserved for specifying which data to retrieve from the "Allocation Engine ADSO", to define the drivers. In other words, this specifies the set of filters that must be applied to obtain such a set, and so, each column reserved for such a section can assume as value one of the following types:
    * The same values used for the "Filters For Data to be Allocated Section";
    * The Additional Value "PP" which means "Parity" or select all the values for the dimension that specifies "PP" that have a value for the "Allocating Dimension" to use as Drivers[7];

- Enable Section: this last portion is used to identify if the rule is enabled or not;

Notice that, such data contained in the two ADSOs is either collected from "BPC Layer" or can be manually added. In particular, the set of Drivers can be augmented by the final user by manually inserting them, while, on the other hand, the set of "Allocation Rules" is only manually inserted. Finally, for what concerns the Planning sequence that implements the "Allocation Engine" the reason why it is composed of two custom planning functions is the following:

[6]Considering the "Hierarchy Nodes Example" image, it could also be "TOT_ENTITIES".
[7]The concept will be better understood in the "Allocation Engine: Execution Example" subsubsection.

- The first planning function called "Expansion of Rules" is used to re-write the Allocation Rules, which contains hierarchy node values, simpler, by replicating such rules for each leaf under the tree rooted by the node selected [8];
- The second function is the actual "Allocation Engine" whose goal is to, given the set of Data to be Allocated and the Drivers, redistribute the data to be allocated according to the drivers themselves;

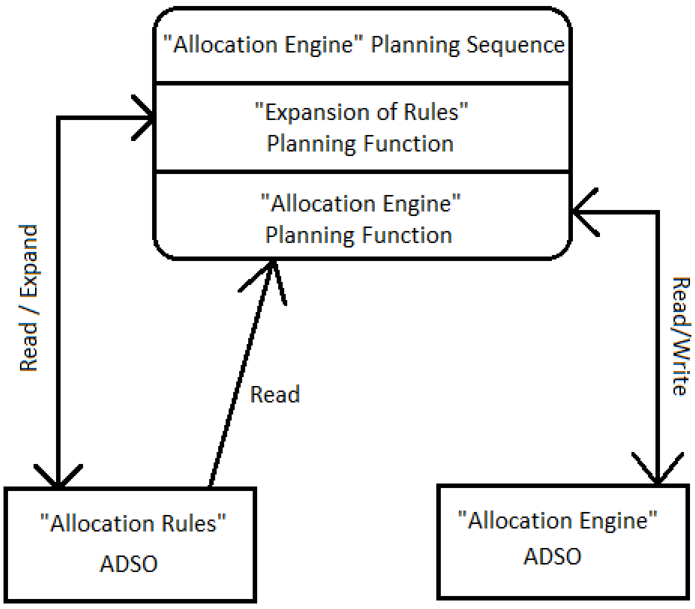
"Allocation Engine" Planning Sequence
"Expansion of Rules" Planning Function
"Allocation Engine" Planning Function
Read / Expand
Read
Read/Write
"Allocation Rules" ADSO
"Allocation Engine" ADSO

Figure 5.7: "Aruba S.p.A." Project: Allocation Engine Schema

5.4.2.1.3 Implementation

This section will describe how the "Allocation Engine" has been implemented. In particular, as said just before, the "Allocation Engine" consists of a custom planning function whose goal is to, given the set of data to be allocated and the drivers, for the current allocation rule selected, perform the opportune re-partitions on the data to be allocated. This means that the "Allocation Engine" works not only with data to be allocated and the drivers, but also with the "Allocation Rules" and, particularly speaking can handle multiple "Allocation Rules", which manage to create multiple: drivers, data to be allocated and allocated data. However, when stating that the "Allocation Engine" represents

[8]In case of multiple nodes selected, for the same rule, a Cartesian product will be applied to match all the possible combinations.

a custom planning function this is not entirely correct. To be precise, it consists of the implementation of the SQL function that the IF_AMDP_MARKER_HDB interface requires. Having said this, the next big thing to analyze is how in practice the "Allocation Engine" works in terms of its implementation. Therefore to make things clear the implemented code will be introduced slowly to explain each detail of how this powerful tool works. So, the first thing that the "Allocation Engine" does is to declare some useful variable that later will be used. In particular, among those variables worth mentioning are:

- RULE_NUM: which contains the number of "Allocation Rules" that must be processed by the "Allocation Engine";
- ALL_DIM: which represents the dimension to be allocated for the currently processed "Allocation Rule";

```
DECLARE RULE_NUM INT;
DECLARE ITERATOR INT;
DECLARE ALL_DIM STRING;
DECLARE RULE_CURR INT;
DECLARE CYCLE_CURR INT;
DECLARE ITERATOR_INCREMENT INT;
```

Code 5.1: "Allocation Engine" Implementation: Variables Declaration

Next, the "Allocation Engine" prepares all the data that it must use before elaborating each single "Allocation Rule" at once. This step will:

- Define the set of all "filters" that must be used for each "Allocation Rule" to identify the set of data to be allocated;
- Define the set of all "filters" that must be used for each "Allocation Rule" to identify the set of drivers;

```
RULES_DRIVERS_FILTERS_FILTER =
    SELECT
        ZACCCONS, ZACCNAT, ZBL, ZCATEGORY, ZCLIENT, ZENTITY,
        ZINFSTR, ZINTRCMP, ZORGREP, ZPROJ, ZSRVPRO1, ZSRVPRO2,
        ZSRVPRO3, ZSRVPRO4, ZSUBTBLS,
        ZDACCCONS, ZDACCNAT, ZDBL, ZDCATEGRY, ZDCLIENT, ZDENTITY,
        ZDINFSTR, ZDINTRCMP, ZDORGREP, ZDPROJ, ZDSRVPRO1, ZDSRVPRO2,
        ZDSRVPRO3, ZDSRVPRO4, ZDSUBTBLS,
        FISCPER, FISCVARNT, INFOPROV, ZALCDIM, ZALLCYCLE, ZALLRULE,
        ZAUDIT, ZCUR, ZDRIVER, ZENABLE, ZGROUP
    FROM :I_ZAL2ALR
    WHERE
```

```
        ZENTITY = :S_ENTITY AND
        ZCATEGORY = :S_CATEGORY AND
        FISCPER = :S_FISCPER AND
        (S_ALLOCATIONCYCLE = '' OR ZALLCYCLE = :S_ALLOCATIONCYCLE);

RULE_FILTERS =
    SELECT
        ZACCCONS, ZACCNAT, ZBL, ZCATEGORY, ZCLIENT, ZENTITY,
        ZINFSTR, ZINTRCMP, ZORGREP, ZPROJ, ZSRVPRO1, ZSRVPRO2,
        ZSRVPRO3, ZSRVPRO4, ZSUBTBLS, ZALLCYCLE, ZALLRULE, ZALCDIM
    FROM
        :RULES_DRIVERS_FILTERS_FILTER;

DRIVER_FILTERS =
    SELECT
        ZDACCCONS, ZDACCNAT, ZDBL, ZDCATEGRY, ZDCLIENT, ZDENTITY,
        ZDINFSTR, ZDINTRCMP, ZDORGREP, ZDPROJ, ZDSRVPRO1, ZDSRVPRO2,
        ZDSRVPRO3, ZDSRVPRO4, ZDSUBTBLS, ZDRIVER, ZALLCYCLE,
        ZALLRULE, ZALCDIM
    FROM
        :RULES_DRIVERS_FILTERS_FILTER;
```

Code 5.2: "Allocation Engine" Implementation: "Filters" Computation

The following step of the "Allocation Engine" is to start a "while" loop that will process each single "Allocation Rule" at once.

```
e_view =
    SELECT
        FISCPER, FISCVARNT, INFOPROV, ZACCCONS, ZACCNAT, ZALLCYCLE,
        ZALLRULE, ZAUDIT, ZBL, ZCATEGORY, ZCLIENT, ZCUR, ZDRIVER,
        ZENTITY, ZGROUP, ZINFSTR, ZINTRCMP, ZORGREP, ZPROJ, ZSRVPRO1,
        ZSRVPRO2, ZSRVPRO3, ZSRVPRO4, ZSUBTBLS, ZSIGNDATA
    FROM
        :i_view_ref_data
    WHERE
        ZBL = 'EMPTY';

SELECT COUNT(*) INTO RULE_NUM FROM :RULE_FILTERS;

SELECT 1 INTO ITERATOR FROM dummy;

WHILE ITERATOR <= RULE_NUM DO
```

Code 5.3: "Allocation Engine" Implementation: "While-Loop" Preparation

Inside the previously mentioned "while" loop several operations will be performed:

- First will be populated some useful variables, in particular, the one that allows stepping into the next "Allocation Rule" as well as the one that contains the current "Allocating Dimension";

```
    SELECT
        ZALCDIM, ZALLCYCLE, ZALLRULE
    INTO
        ALL_DIM, CYCLE_CURR, RULE_CURR
    FROM (
        SELECT
            *
        FROM
            :RULE_FILTERS
        ORDER BY
            ZALLCYCLE ASC,
            ZALLRULE ASC
        LIMIT
            :ITERATOR
    )
    ORDER BY
        ZALLCYCLE DESC,
        ZALLRULE DESC
    LIMIT 1;

    SELECT
        COUNT(*)
    INTO
        ITERATOR_INCREMENT
    FROM
        :RULE_FILTERS
    WHERE
        ZALLCYCLE = :CYCLE_CURR AND
        ZALLRULE = :RULE_CURR;

```

Code 5.4: "Allocation Engine" Implementation: "Loop-Variables"

- Second will be computed the "filters" for selecting both the data to be allocated as well as the data to be used as drivers (for the "Allocation Rule" selected);

```
    RULE_FILTERS_I =
        SELECT DISTINCT
            ZACCCONS, ZACCNAT, ZBL, ZCATEGORY, ZCLIENT,
            ZENTITY, ZINFSTR, ZINTRCMP, ZORGREP, ZPROJ,
            ZSRVPRO1, ZSRVPRO2, ZSRVPRO3, ZSRVPRO4, ZSUBTBLS,
            ZALLCYCLE, ZALLRULE, ZALCDIM
        FROM
            :RULE_FILTERS
        WHERE
            ZALLCYCLE = :CYCLE_CURR AND
            ZALLRULE = :RULE_CURR;

    DRIVER_FILTERS_I =
        SELECT DISTINCT
            ZDACCCONS, ZDACCNAT, ZDBL, ZDCATEGRY, ZDCLIENT,
            ZDENTITY, ZDINFSTR, ZDINTRCMP, ZDORGREP, ZDPROJ,
            ZDSRVPRO1, ZDSRVPRO2, ZDSRVPRO3, ZDSRVPRO4,
            ZDSUBTBLS, ZDRIVER, ZALLCYCLE, ZALLRULE, ZALCDIM
        FROM
            :DRIVER_FILTERS
        WHERE
            ZALLCYCLE = :CYCLE_CURR AND
            ZALLRULE = :RULE_CURR;

```

Code 5.5: "Allocation Engine" Implementation: "Loop-Filters"

- Then, will be computed, given the previous filters, the actual data to be allocated as well as the drivers themselves;

```
    ALLOCATION_DATA =
        SELECT
            :i_view_ref_data.ZACCCONS,
            :i_view_ref_data.ZACCNAT,
            :i_view_ref_data.ZBL,
            :i_view_ref_data.ZCATEGORY,
            :i_view_ref_data.ZCLIENT,
            :i_view_ref_data.ZENTITY,
            :i_view_ref_data.ZINFSTR,
            :i_view_ref_data.ZINTRCMP,
            :i_view_ref_data.ZORGREP,
            :i_view_ref_data.ZPROJ,
            :i_view_ref_data.ZSRVPRO1,
            :i_view_ref_data.ZSRVPRO2,
            :i_view_ref_data.ZSRVPRO3,
            :i_view_ref_data.ZSRVPRO4,
            :i_view_ref_data.ZSUBTBLS,
            '' AS ZDRIVER,
            :i_view_ref_data.zsigndata AS ZSIGNDATA
        FROM
           :i_view_ref_data
        JOIN
           :RULE_FILTERS_I
        ON
            (:RULE_FILTERS_I.ZACCCONS = 'XX'
            OR
            :RULE_FILTERS_I.ZACCCONS =
            :i_view_ref_data.ZACCCONS)
        AND
            (:RULE_FILTERS_I.ZACCNAT = 'XX'
            OR
            :RULE_FILTERS_I.ZACCNAT =
            :i_view_ref_data.ZACCNAT)
        AND
            (:RULE_FILTERS_I.ZBL = 'XX'
            OR
            :RULE_FILTERS_I.ZBL =
            :i_view_ref_data.ZBL)
        AND
            (:RULE_FILTERS_I.ZCATEGORY = 'XX'
            OR
            :RULE_FILTERS_I.ZCATEGORY =
            :i_view_ref_data.ZCATEGORY)
        AND
            (:RULE_FILTERS_I.ZCLIENT = 'XX'
            OR
            :RULE_FILTERS_I.ZCLIENT =
            :i_view_ref_data.ZCLIENT)
        AND
            (:RULE_FILTERS_I.ZENTITY = 'XX'
            OR
            :RULE_FILTERS_I.ZENTITY =
            :i_view_ref_data.ZENTITY)
        AND
            (:RULE_FILTERS_I.ZINFSTR = 'XX'
            OR
            :RULE_FILTERS_I.ZINFSTR =
            :i_view_ref_data.ZINFSTR)
        AND
            (:RULE_FILTERS_I.ZINTRCMP = 'XX'
            OR
            :RULE_FILTERS_I.ZINTRCMP =
            :i_view_ref_data.ZINTRCMP)
        AND
            (:RULE_FILTERS_I.ZORGREP = 'XX'
            OR
            :RULE_FILTERS_I.ZORGREP =
            :i_view_ref_data.ZORGREP)
        AND
            (:RULE_FILTERS_I.ZPROJ = 'XX'
            OR
            :RULE_FILTERS_I.ZPROJ =
            :i_view_ref_data.ZPROJ)
        AND
            (:RULE_FILTERS_I.ZSRVPRO1 = 'XX'
            OR
            :RULE_FILTERS_I.ZSRVPRO1 =
            :i_view_ref_data.ZSRVPRO1)
        AND
            (:RULE_FILTERS_I.ZSRVPRO2 = 'XX'
            OR
            :RULE_FILTERS_I.ZSRVPRO2 =
            :i_view_ref_data.ZSRVPRO2)
        AND
            (:RULE_FILTERS_I.ZSRVPRO3 = 'XX'
            OR
            :RULE_FILTERS_I.ZSRVPRO3 =
            :i_view_ref_data.ZSRVPRO3)
        AND
            (:RULE_FILTERS_I.ZSRVPRO4 = 'XX'
            OR
            :RULE_FILTERS_I.ZSRVPRO4 =
            :i_view_ref_data.ZSRVPRO4)
        AND
            (:RULE_FILTERS_I.ZSUBTBLS = 'XX'
            OR
            :RULE_FILTERS_I.ZSUBTBLS =
            :i_view_ref_data.ZSUBTBLS);

    DRIVERS_DATA =
        SELECT
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDACCCONS <> 'PP' AND
                    :ALL_DIM <> 'ZACCONS'
                THEN
                   'XX'
                ELSE
                    :i_view_ref_data.ZACCCONS
            END AS ZACCCONS,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDACCNAT <> 'PP' AND
                    :ALL_DIM <> 'ZACCNAT'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZACCNAT
            END AS ZACCNAT,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDBL <> 'PP' AND
                    :ALL_DIM <> 'ZBL'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZBL
            END AS ZBL,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDCATEGRY <> 'PP' AND
                    :ALL_DIM <> 'ZCATEGORY'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZCATEGORY
            END AS ZCATEGORY,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDCLIENT <> 'PP' AND
                    :ALL_DIM <> 'ZCLIENT'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZCLIENT
            END AS ZCLIENT,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDENTITY <> 'PP' AND
                    :ALL_DIM <> 'ZENTITY'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZENTITY
            END AS ZENTITY,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDINFSTR <> 'PP' AND
                    :ALL_DIM <> 'ZINFSTR'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZINFSTR
            END AS ZINFSTR,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDINTRCMP <> 'PP' AND
                    :ALL_DIM <> 'ZINTRCMP'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZINTRCMP
            END AS ZINTRCMP,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDORGREP <> 'PP' AND
                    :ALL_DIM <> 'ZORGREP'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZORGREP
            END AS ZORGREP,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDPROJ <> 'PP' AND
                    :ALL_DIM <> 'ZPROJ'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZPROJ
            END AS ZPROJ,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDSRVPRO1 <> 'PP' AND
                    :ALL_DIM <> 'ZSRVPRO1'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZSRVPRO1
            END AS ZSRVPRO1,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDSRVPRO2 <> 'PP' AND
                    :ALL_DIM <> 'ZSRVPRO2'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZSRVPRO2
            END AS ZSRVPRO2,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDSRVPRO3 <> 'PP' AND
                    :ALL_DIM <> 'ZSRVPRO3'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZSRVPRO3
            END AS ZSRVPRO3,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDSRVPRO4 <> 'PP' AND
                    :ALL_DIM <> 'ZSRVPRO4'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZSRVPRO4
            END AS ZSRVPRO4,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDSUBTBLS <> 'PP' AND
                    :ALL_DIM <> 'ZSUBTBLS'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZSUBTBLS
            END AS ZSUBTBLS,
            CASE
                WHEN
                    :DRIVER_FILTERS_I.ZDRIVER <> 'PP' AND
                    :ALL_DIM <> 'ZDRIVER'
                THEN
                    'XX'
                ELSE
                    :i_view_ref_data.ZDRIVER
            END AS ZDRIVER,
            :i_view_ref_data.ZSIGNDATA AS ZSIGNDATA
            FROM
                :i_view_ref_data
            JOIN
                :DRIVER_FILTERS_I
            ON
                (:DRIVER_FILTERS_I.ZDACCCONS = 'PP'
                OR
                :DRIVER_FILTERS_I.ZDACCCONS = 'XX'
                OR
                :DRIVER_FILTERS_I.ZDACCCONS =
                :i_view_ref_data.ZACCCONS)
            AND
                (:DRIVER_FILTERS_I.ZDACCNAT = 'PP'
                OR
                :DRIVER_FILTERS_I.ZDACCNAT = 'XX'
                OR
                :DRIVER_FILTERS_I.ZDACCNAT =
                :i_view_ref_data.ZACCNAT)
            AND
                (:DRIVER_FILTERS_I.ZDBL = 'PP'
                OR
                :DRIVER_FILTERS_I.ZDBL = 'XX'
                OR
                :DRIVER_FILTERS_I.ZDBL =
                :i_view_ref_data.ZBL)
            AND
                (:DRIVER_FILTERS_I.ZDCATEGRY = 'PP'
                OR
                :DRIVER_FILTERS_I.ZDCATEGRY = 'XX'
                OR
```

```
                    :DRIVER_FILTERS_I.ZDCATEGRY =
                    :i_view_ref_data.ZCATEGORY)
                AND
                    (:DRIVER_FILTERS_I.ZDCLIENT = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDCLIENT = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDCLIENT =
                    :i_view_ref_data.ZCLIENT)
                AND
                    (:DRIVER_FILTERS_I.ZDENTITY = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDENTITY = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDENTITY =
                    :i_view_ref_data.ZENTITY)
                AND
                    (:DRIVER_FILTERS_I.ZDINFSTR = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDINFSTR = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDINFSTR =
                    :i_view_ref_data.ZINFSTR)
                AND
                    (:DRIVER_FILTERS_I.ZDINTRCMP = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDINTRCMP = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDINTRCMP =
                    :i_view_ref_data.ZINTRCMP)
                AND
                    (:DRIVER_FILTERS_I.ZDORGREP = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDORGREP = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDORGREP =
                    :i_view_ref_data.ZORGREP)
                AND
                    (:DRIVER_FILTERS_I.ZDPROJ = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDPROJ = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDPROJ =
                    :i_view_ref_data.ZPROJ)
                AND
                    (:DRIVER_FILTERS_I.ZDSRVPRO1 = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO1 = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO1 =
                    :i_view_ref_data.ZSRVPRO1)
                AND
                    (:DRIVER_FILTERS_I.ZDSRVPRO2 = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO2 = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO2 =
                    :i_view_ref_data.ZSRVPRO2)
                AND
                    (:DRIVER_FILTERS_I.ZDSRVPRO3 = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO3 = 'XX'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO3 =
                    :i_view_ref_data.ZSRVPRO3)
                AND
                    (:DRIVER_FILTERS_I.ZDSRVPRO4 = 'PP'
                    OR
                    :DRIVER_FILTERS_I.ZDSRVPRO4 = 'XX'
                    OR
                :DRIVER_FILTERS_I.ZDSRVPRO4 =
                :i_view_ref_data.ZSRVPRO4)
            AND
                (:DRIVER_FILTERS_I.ZDSRVPRO4 = 'PP'
                OR
                :DRIVER_FILTERS_I.ZDSUBTBLS = 'XX'
                OR
                :DRIVER_FILTERS_I.ZDSUBTBLS =
                :i_view_ref_data.ZSUBTBLS)
            AND
                (:DRIVER_FILTERS_I.ZDRIVER = 'PP'
                OR
                :DRIVER_FILTERS_I.ZDRIVER = 'XX'
                OR
                :DRIVER_FILTERS_I.ZDRIVER =
                :i_view_ref_data.ZDRIVER);

```
Code 5.6: "Allocation Engine" Implementation: Drivers and Allocation Data

Notice that here will be used the previously mentioned special characters "PP" and "XX" which perform the "Parity" and "Any" selection respectively[9];

- Next, depending the "Allocating Dimension", will be calculated the percentages, given the drivers, on how to break down the data to be allocated[10];

```
    ....
    ELSEIF( :ALL_DIM = 'ZENTITY' ) THEN
        DRIVERS_DATA =
            SELECT
                *
            FROM
                :DRIVERS_DATA
            WHERE
                :DRIVERS_DATA.ZENTITY <> '';

        SUM_ZSIGNDATA_ENTITY =
            SELECT
                ZACCCONS, ZACCNAT, ZBL, ZCATEGORY, ZCLIENT,
                ZINFSTR, ZINTRCMP, ZORGREP, ZPROJ, ZSRVPRO1,
                ZSRVPRO2, ZSRVPRO3, ZSRVPRO4, ZSUBTBLS, ZDRIVER,
                SUM( ZSIGNDATA ) AS TOT
            FROM
                :DRIVERS_DATA
            GROUP BY
                ZACCCONS, ZACCNAT, ZBL, ZCATEGORY, ZCLIENT,
                ZINFSTR, ZINTRCMP, ZORGREP, ZPROJ, ZSRVPRO1,
                ZSRVPRO2, ZSRVPRO3, ZSRVPRO4, ZSUBTBLS, ZDRIVER
            HAVING
                SUM( ZSIGNDATA ) <> 0 ;

        PERC =
            SELECT
                :DRIVERS_DATA.ZACCCONS, :DRIVERS_DATA.ZACCNAT,
                :DRIVERS_DATA.ZBL, :DRIVERS_DATA.ZCATEGORY,
                :DRIVERS_DATA.ZCLIENT, :DRIVERS_DATA.ZENTITY,
                :DRIVERS_DATA.ZINFSTR, :DRIVERS_DATA.ZINTRCMP,
                :DRIVERS_DATA.ZORGREP, :DRIVERS_DATA.ZPROJ,
                :DRIVERS_DATA.ZSRVPRO1, :DRIVERS_DATA.ZSRVPRO2,
                :DRIVERS_DATA.ZSRVPRO3, :DRIVERS_DATA.ZSRVPRO4,
                :DRIVERS_DATA.ZSUBTBLS, :DRIVERS_DATA.ZDRIVER,
                SUM( :DRIVERS_DATA.ZSIGNDATA /
```

[9]Their effect, as said previously, will be described better with an example.
[10]For space reasons the code will be shown only for the "Legal Entity" Data Model Dimension.

```
                :SUM_ZSIGNDATA_ENTITY.TOT ) AS PERC
            FROM
                :SUM_ZSIGNDATA_ENTITY
            JOIN
                :DRIVERS_DATA
            ON
                :SUM_ZSIGNDATA_ENTITY.ZACCCONS =
                :DRIVERS_DATA.ZACCCONS
            AND
                :SUM_ZSIGNDATA_ENTITY.ZACCNAT =
                :DRIVERS_DATA.ZACCNAT
            AND
                :SUM_ZSIGNDATA_ENTITY.ZBL =
                :DRIVERS_DATA.ZBL
            AND
                :SUM_ZSIGNDATA_ENTITY.ZCATEGORY =
                :DRIVERS_DATA.ZCATEGORY
            AND
                :SUM_ZSIGNDATA_ENTITY.ZCLIENT =
                :DRIVERS_DATA.ZCLIENT
            AND
                :SUM_ZSIGNDATA_ENTITY.ZINFSTR =
                :DRIVERS_DATA.ZINFSTR
            AND
                :SUM_ZSIGNDATA_ENTITY.ZINTRCMP =
                :DRIVERS_DATA.ZINTRCMP
            AND
                :SUM_ZSIGNDATA_ENTITY.ZORGREP =
                :DRIVERS_DATA.ZORGREP
            AND
                :SUM_ZSIGNDATA_ENTITY.ZPROJ =
                :DRIVERS_DATA.ZPROJ
            AND
                :SUM_ZSIGNDATA_ENTITY.ZSRVPRO1 =
                :DRIVERS_DATA.ZSRVPRO1
            AND
                :SUM_ZSIGNDATA_ENTITY.ZSRVPRO2 =
                :DRIVERS_DATA.ZSRVPRO2
            AND
                :SUM_ZSIGNDATA_ENTITY.ZSRVPRO3 =
                :DRIVERS_DATA.ZSRVPRO3
            AND
                :SUM_ZSIGNDATA_ENTITY.ZSRVPRO4 =
                :DRIVERS_DATA.ZSRVPRO4
            AND
                :SUM_ZSIGNDATA_ENTITY.ZSUBTBLS =
                :DRIVERS_DATA.ZSUBTBLS
            AND
                :SUM_ZSIGNDATA_ENTITY.ZDRIVER =
                :DRIVERS_DATA.ZDRIVER
            GROUP BY
                :DRIVERS_DATA.ZACCCONS, :DRIVERS_DATA.ZACCNAT,
                :DRIVERS_DATA.ZBL, :DRIVERS_DATA.ZCATEGORY,
                :DRIVERS_DATA.ZCLIENT, :DRIVERS_DATA.ZENTITY,
                :DRIVERS_DATA.ZINFSTR, :DRIVERS_DATA.ZINTRCMP,
                :DRIVERS_DATA.ZORGREP, :DRIVERS_DATA.ZPROJ,
                :DRIVERS_DATA.ZSRVPRO1, :DRIVERS_DATA.ZSRVPRO2,
                :DRIVERS_DATA.ZSRVPRO3, :DRIVERS_DATA.ZSRVPRO4,
                :DRIVERS_DATA.ZSUBTBLS, :DRIVERS_DATA.ZDRIVER;
    ELSEIF...

```
Code 5.7: "Allocation Engine" Implementation: Percentage Calculation

- Finally as last operations of the "while" loop that ends the entire code for the "Allocation Engine" will be computed:
  - The "Allocated Data", namely, the data to be allocated, divided correctly according to the previous percentages computed;

– The so-called "Cancellations" i.e. the set of data to be allocated, changed in a sign that must be added with both the "Allocated Data" to the "Allocation Engine ADSO" to conclude the current Allocation operation;

– The final data that must be written in the "Allocation Engine ADSO" which consists of the union of "Allocated Data" and "Cancellations" with the value of one of the technical dimensions changed accordingly to when the Allocations were performed (pre or post-consolidation);

```
        ALLOCATED_DATA =
            SELECT
                CASE
                    :ALL_DIM
                WHEN
                    'ZACCCONS'
                THEN
                    :PERC.ZACCCONS
                ELSE
                    :ALLOCATION_DATA.ZACCCONS
                END AS ZACCCONS,
                CASE
                    :ALL_DIM
                WHEN
                    'ZACCNAT'
                THEN
                    :PERC.ZACCNAT
                ELSE
                    :ALLOCATION_DATA.ZACCNAT
                END AS ZACCNAT,
                CASE
                    :ALL_DIM
                WHEN
                    'ZBL'
                THEN
                    :PERC.ZBL
                ELSE
                    :ALLOCATION_DATA.ZBL
                END AS ZBL,
                CASE
                    :ALL_DIM
                WHEN
                    'ZCATEGORY'
                THEN
                    :PERC.ZCATEGORY
                ELSE
                    :ALLOCATION_DATA.ZCATEGORY
                END AS ZCATEGORY,
                CASE
                    :ALL_DIM
                WHEN
                    'ZCLIENT'
                THEN
                    :PERC.ZCLIENT
                ELSE
                    :ALLOCATION_DATA.ZCLIENT
                END AS ZCLIENT,
                CASE
                    :ALL_DIM
                WHEN
                    'ZENTITY'
                THEN
                    :PERC.ZENTITY
                ELSE
                    :ALLOCATION_DATA.ZENTITY
                END AS ZENTITY,
                CASE
                    :ALL_DIM
                WHEN
```

```
                    'ZINFSTR'
                THEN
                    :PERC.ZINFSTR
                ELSE
                    :ALLOCATION_DATA.ZINFSTR
                END AS ZINFSTR,
                CASE
                    :ALL_DIM
                WHEN
                    'ZINTRCMP'
                THEN
                    :PERC.ZINTRCMP
                ELSE
                    :ALLOCATION_DATA.ZINTRCMP
                END AS ZINTRCMP,
                CASE
                    :ALL_DIM
                WHEN
                    'ZORGREP'
                THEN
                    :PERC.ZORGREP
                ELSE
                    :ALLOCATION_DATA.ZORGREP
                END AS ZORGREP,
                CASE
                    :ALL_DIM
                WHEN
                    'ZPROJ'
                THEN
                    :PERC.ZPROJ
                ELSE
                    :ALLOCATION_DATA.ZPROJ
                END AS ZPROJ,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO1'
                THEN
                    :PERC.ZSRVPRO1
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO1
                END AS ZSRVPRO1,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO2'
                THEN
                    :PERC.ZSRVPRO2
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO2
                END AS ZSRVPRO2,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO3'
                THEN
                    :PERC.ZSRVPRO3
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO3
                END AS ZSRVPRO3,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO4'
                THEN
                    :PERC.ZSRVPRO4
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO4
                END AS ZSRVPRO4,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSUBTBLS'
                THEN
                    :PERC.ZSUBTBLS
                ELSE
                    :ALLOCATION_DATA.ZSUBTBLS
                END AS ZSUBTBLS,
                CASE
                    :ALL_DIM
                WHEN
                    'ZDRIVER'
                THEN
                    :PERC.ZDRIVER
                ELSE
                    :ALLOCATION_DATA.ZDRIVER
                END AS ZDRIVER,
                ( :ALLOCATION_DATA.ZSIGNDATA *
                :PERC.PERC ) AS ZSIGNDATA
            FROM
                :ALLOCATION_DATA
            JOIN
                :PERC
            ON
                ((((:PERC.ZACCCONS = 'XX'
                OR
                :PERC.ZACCCONS = :ALLOCATION_DATA.ZACCCONS))
                AND
                ( :ALL_DIM <> 'ZACCCONS'))
                OR
                ( :ALL_DIM = 'ZACCCONS'))
            AND
                ((((:PERC.ZACCNAT = 'XX'
                OR
                :PERC.ZACCNAT = :ALLOCATION_DATA.ZACCNAT))
                AND
                ( :ALL_DIM <> 'ZACCNAT'))
                OR
                ( :ALL_DIM = 'ZACCNAT'))
            AND
                ((((:PERC.ZBL = 'XX'
                OR
                :PERC.ZBL = :ALLOCATION_DATA.ZBL))
                AND
                ( :ALL_DIM <> 'ZBL'))
                OR
                ( :ALL_DIM = 'ZBL'))
            AND
                ((((:PERC.ZCATEGORY = 'XX'
                OR
                :PERC.ZCATEGORY = :ALLOCATION_DATA.ZCATEGORY))
                AND
                ( :ALL_DIM <> 'ZCATEGORY'))
                OR
                ( :ALL_DIM = 'ZCATEGORY'))
            AND
                ((((:PERC.ZCLIENT = 'XX'
                OR
                :PERC.ZCLIENT = :ALLOCATION_DATA.ZCLIENT))
                AND
                ( :ALL_DIM <> 'ZCLIENT'))
                OR
                ( :ALL_DIM = 'ZCLIENT'))
            AND
                ((((:PERC.ZENTITY = 'XX'
                OR
                :PERC.ZENTITY = :ALLOCATION_DATA.ZENTITY))
                AND
                ( :ALL_DIM <> 'ZENTITY'))
                OR
                ( :ALL_DIM = 'ZENTITY'))
```

```
        AND
            ((((:PERC.ZINFSTR = 'XX'
            OR
            :PERC.ZINFSTR = :ALLOCATION_DATA.ZINFSTR))
            AND
            ( :ALL_DIM <> 'ZINFSTR'))
            OR
            ( :ALL_DIM = 'ZINFSTR'))
        AND
            ((((:PERC.ZINTRCMP = 'XX'
            OR
            :PERC.ZINTRCMP = :ALLOCATION_DATA.ZINTRCMP))
            AND
            ( :ALL_DIM <> 'ZINTRCMP'))
            OR
            ( :ALL_DIM = 'ZINTRCMP'))
        AND
            ((((:PERC.ZORGREP = 'XX'
            OR
            :PERC.ZORGREP = :ALLOCATION_DATA.ZORGREP))
            AND
            ( :ALL_DIM <> 'ZORGREP'))
            OR
            ( :ALL_DIM = 'ZORGREP'))
        AND
            ((((:PERC.ZPROJ = 'XX'
            OR
            :PERC.ZPROJ = :ALLOCATION_DATA.ZPROJ))
            AND
            ( :ALL_DIM <> 'ZPROJ'))
            OR
            ( :ALL_DIM = 'ZPROJ'))
        AND
            ((((:PERC.ZSRVPRO1 = 'XX'
            OR
            :PERC.ZSRVPRO1 = :ALLOCATION_DATA.ZSRVPRO1))
            AND
            ( :ALL_DIM <> 'ZSRVPRO1'))
            OR
            ( :ALL_DIM = 'ZSRVPRO1'))
        AND
            ((((:PERC.ZSRVPRO2 = 'XX'
            OR
            :PERC.ZSRVPRO2 = :ALLOCATION_DATA.ZSRVPRO2))
            AND
            ( :ALL_DIM <> 'ZSRVPRO2'))
            OR
            ( :ALL_DIM = 'ZSRVPRO2'))
        AND
            ((((:PERC.ZSRVPRO3 = 'XX'
            OR
            :PERC.ZSRVPRO3 = :ALLOCATION_DATA.ZSRVPRO3))
            AND
            ( :ALL_DIM <> 'ZSRVPRO3'))
            OR
            ( :ALL_DIM = 'ZSRVPRO3'))
        AND
            ((((:PERC.ZSRVPRO4 = 'XX'
            OR
            :PERC.ZSRVPRO4 = :ALLOCATION_DATA.ZSRVPRO4))
            AND
            ( :ALL_DIM <> 'ZSRVPRO4'))
            OR
            ( :ALL_DIM = 'ZSRVPRO4'))
        AND
            ((((:PERC.ZSUBTBLS = 'XX'
            OR
            :PERC.ZSUBTBLS = :ALLOCATION_DATA.ZSUBTBLS))
            AND
            ( :ALL_DIM <> 'ZSUBTBLS'))
```

```
                OR
                ( :ALL_DIM = 'ZSUBTBLS'))
            AND
                ((((:PERC.ZDRIVER = 'XX'
                OR
                :PERC.ZDRIVER = :ALLOCATION_DATA.ZDRIVER))
                AND
                ( :ALL_DIM <> 'ZDRIVER'))
                OR
                ( :ALL_DIM = 'ZDRIVER'));

        CANCELLATIONS =
            SELECT
                CASE
                    :ALL_DIM
                WHEN
                    'ZACCCONS'
                THEN
                    :ALLOCATION_DATA.ZACCCONS
                ELSE
                    :ALLOCATION_DATA.ZACCCONS
                END AS ZACCCONS,
                CASE
                    :ALL_DIM
                WHEN
                    'ZACCNAT'
                THEN
                    :ALLOCATION_DATA.ZACCNAT
                ELSE
                    :ALLOCATION_DATA.ZACCNAT
                END AS ZACCNAT,
                CASE
                    :ALL_DIM
                WHEN
                    'ZBL'
                THEN
                    :ALLOCATION_DATA.ZBL
                ELSE
                    :ALLOCATION_DATA.ZBL
                END AS ZBL,
                CASE
                    :ALL_DIM
                WHEN
                    'ZCATEGORY'
                THEN
                    :ALLOCATION_DATA.ZCATEGORY
                ELSE
                    :ALLOCATION_DATA.ZCATEGORY
                END AS ZCATEGORY,
                CASE
                    :ALL_DIM
                WHEN
                    'ZCLIENT'
                THEN
                    :ALLOCATION_DATA.ZCLIENT
                ELSE
                    :ALLOCATION_DATA.ZCLIENT
                END AS ZCLIENT,
                CASE
                    :ALL_DIM
                WHEN
                    'ZENTITY'
                THEN
                    :ALLOCATION_DATA.ZENTITY
                ELSE
                    :ALLOCATION_DATA.ZENTITY
                END AS ZENTITY,
                CASE
                    :ALL_DIM
                WHEN
                    'ZINFSTR'
                THEN
                    :ALLOCATION_DATA.ZINFSTR
                ELSE
                    :ALLOCATION_DATA.ZINFSTR
                END AS ZINFSTR,
                CASE
                    :ALL_DIM
                WHEN
                    'ZINTRCMP'
                THEN
                    :ALLOCATION_DATA.ZINTRCMP
                ELSE
                    :ALLOCATION_DATA.ZINTRCMP
                END AS ZINTRCMP,
                CASE
                    :ALL_DIM
                WHEN
                    'ZORGREP'
                THEN
                    :ALLOCATION_DATA.ZORGREP
                ELSE
                    :ALLOCATION_DATA.ZORGREP
                END AS ZORGREP,
                CASE
                    :ALL_DIM
                WHEN
                    'ZPROJ'
                THEN
                    :ALLOCATION_DATA.ZPROJ
                ELSE
                    :ALLOCATION_DATA.ZPROJ
                END AS ZPROJ,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO1'
                THEN
                    :ALLOCATION_DATA.ZSRVPRO1
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO1
                END AS ZSRVPRO1,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO2'
                THEN
                    :ALLOCATION_DATA.ZSRVPRO2
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO2
                END AS ZSRVPRO2,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO3'
                THEN
                    :ALLOCATION_DATA.ZSRVPRO3
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO3
                END AS ZSRVPRO3,
                CASE
                    :ALL_DIM
                WHEN
                    'ZSRVPRO4'
                THEN
                    :ALLOCATION_DATA.ZSRVPRO4
                ELSE
                    :ALLOCATION_DATA.ZSRVPRO4
                END AS ZSRVPRO4,
                CASE
                    :ALL_DIM
                WHEN
```

```
                    'ZSUBTBLS'
                THEN
                    :ALLOCATION_DATA.ZSUBTBLS
                ELSE
                    :ALLOCATION_DATA.ZSUBTBLS
                END AS ZSUBTBLS,
                CASE
                    :ALL_DIM
                WHEN
                    'ZDRIVER'
                THEN
                    :ALLOCATION_DATA.ZDRIVER
                ELSE
                    :ALLOCATION_DATA.ZDRIVER
                END AS ZDRIVER,
                ( :ALLOCATION_DATA.ZSIGNDATA *
                :PERC.PERC * (-1.0) ) AS ZSIGNDATA
            FROM
                :ALLOCATION_DATA
            JOIN
                :PERC
            ON
                ((((:PERC.ZACCCONS = 'XX'
                OR
                :PERC.ZACCCONS = :ALLOCATION_DATA.ZACCCONS))
                AND
                ( :ALL_DIM <> 'ZACCCONS'))
                OR
                ( :ALL_DIM = 'ZACCCONS'))
            AND
                ((((:PERC.ZACCNAT = 'XX'
                OR
                :PERC.ZACCNAT = :ALLOCATION_DATA.ZACCNAT))
                AND
                ( :ALL_DIM <> 'ZACCNAT'))
                OR
                ( :ALL_DIM = 'ZACCNAT'))
            AND
                ((((:PERC.ZBL = 'XX'
                OR
                :PERC.ZBL = :ALLOCATION_DATA.ZBL))
                AND
                ( :ALL_DIM <> 'ZBL'))
                OR
                ( :ALL_DIM = 'ZBL'))
            AND
                ((((:PERC.ZCATEGORY = 'XX'
                OR
                :PERC.ZCATEGORY = :ALLOCATION_DATA.ZCATEGORY))
                AND
                ( :ALL_DIM <> 'ZCATEGORY'))
                OR
                ( :ALL_DIM = 'ZCATEGORY'))
            AND
                ((((:PERC.ZCLIENT = 'XX'
                OR
                :PERC.ZCLIENT = :ALLOCATION_DATA.ZCLIENT))
                AND
                ( :ALL_DIM <> 'ZCLIENT'))
                OR
                ( :ALL_DIM = 'ZCLIENT'))
            AND
                ((((:PERC.ZENTITY = 'XX'
                OR
                :PERC.ZENTITY = :ALLOCATION_DATA.ZENTITY))
                AND
                ( :ALL_DIM <> 'ZENTITY'))
                OR
                ( :ALL_DIM = 'ZENTITY'))
            AND
```

```
        ((((:PERC.ZINFSTR = 'XX'
        OR
        :PERC.ZINFSTR = :ALLOCATION_DATA.ZINFSTR))
        AND
        ( :ALL_DIM <> 'ZINFSTR'))
        OR
        ( :ALL_DIM = 'ZINFSTR'))
    AND
        ((((:PERC.ZINTRCMP = 'XX'
        OR
        :PERC.ZINTRCMP = :ALLOCATION_DATA.ZINTRCMP))
        AND
        ( :ALL_DIM <> 'ZINTRCMP'))
        OR
        ( :ALL_DIM = 'ZINTRCMP'))
    AND
        ((((:PERC.ZORGREP = 'XX'
        OR
        :PERC.ZORGREP = :ALLOCATION_DATA.ZORGREP))
        AND
        ( :ALL_DIM <> 'ZORGREP'))
        OR
        ( :ALL_DIM = 'ZORGREP'))
    AND
        ((((:PERC.ZPROJ = 'XX'
        OR
        :PERC.ZPROJ = :ALLOCATION_DATA.ZPROJ))
        AND
        ( :ALL_DIM <> 'ZPROJ'))
        OR
        ( :ALL_DIM = 'ZPROJ'))
    AND
        ((((:PERC.ZSRVPRO1 = 'XX'
        OR
        :PERC.ZSRVPRO1 = :ALLOCATION_DATA.ZSRVPRO1))
        AND
        ( :ALL_DIM <> 'ZSRVPRO1'))
        OR
        ( :ALL_DIM = 'ZSRVPRO1'))
    AND
        ((((:PERC.ZSRVPRO2 = 'XX'
        OR
        :PERC.ZSRVPRO2 = :ALLOCATION_DATA.ZSRVPRO2))
        AND
        ( :ALL_DIM <> 'ZSRVPRO2'))
        OR
        ( :ALL_DIM = 'ZSRVPRO2'))
    AND
        ((((:PERC.ZSRVPRO3 = 'XX'
        OR
        :PERC.ZSRVPRO3 = :ALLOCATION_DATA.ZSRVPRO3))
        AND
        ( :ALL_DIM <> 'ZSRVPRO3'))
        OR
        ( :ALL_DIM = 'ZSRVPRO3'))
    AND
        ((((:PERC.ZSRVPRO4 = 'XX'
        OR
        :PERC.ZSRVPRO4 = :ALLOCATION_DATA.ZSRVPRO4))
        AND
        ( :ALL_DIM <> 'ZSRVPRO4'))
        OR
        ( :ALL_DIM = 'ZSRVPRO4'))
    AND
        ((((:PERC.ZSUBTBLS = 'XX'
        OR
        :PERC.ZSUBTBLS = :ALLOCATION_DATA.ZSUBTBLS))
        AND
        ( :ALL_DIM <> 'ZSUBTBLS'))
        OR
```

```
                ( :ALL_DIM = 'ZSUBTBLS'))
            AND
                ((((:PERC.ZDRIVER = 'XX'
                OR
                :PERC.ZDRIVER = :ALLOCATION_DATA.ZDRIVER))
                AND
                ( :ALL_DIM <> 'ZDRIVER'))
                OR
                ( :ALL_DIM = 'ZDRIVER'));

        TEMP =
            SELECT
                *
            FROM (
                SELECT
                    *
                FROM
                    :CANCELLATIONS
                UNION ALL
                SELECT
                    *
                FROM
                    :ALLOCATED_DATA
                );

        e_view =
            SELECT
                *
            FROM
                :e_view
            UNION ALL
            SELECT
                S_FISCPER AS FISCPER, 'K4' AS FISCVARNT,
                'ZADDM2CON' AS INFOPROV, ZACCCONS, ZACCNAT,
                :CYCLE_CURR AS ZALLCYCLE, :RULE_CURR AS ZALLRULE
    ,
                CASE
                    :S_GROUP
                WHEN
                    'G_NONE'
                THEN
                    'ALL_BPC'
                ELSE
                    'ALL_POC'
                END AS ZAUDIT,
                ZBL, ZCATEGORY, ZCLIENT, S_CURRENCY  AS ZCUR,
                ZDRIVER, ZENTITY, S_GROUP AS ZGROUP, ZINFSTR,
                ZINTRCMP, ZORGREP, ZPROJ, ZSRVPRO1, ZSRVPRO2,
                ZSRVPRO3, ZSRVPRO4, ZSUBTBLS, ZSIGNDATA
            FROM
                :TEMP;

        i_view_ref_data =
            SELECT
                *
            FROM
                :i_view_ref_data
            UNION ALL
            SELECT
                *
            FROM
                :e_view;

        ITERATOR = ITERATOR + :ITERATOR_INCREMENT;
    END WHILE ;
```

Code 5.8: "Allocation Engine" Implementation: Allocatated Data

Therefore, by defining the correct set of "Allocation Rules" and corresponding drivers, the problem of computing the costs and revenues of the internal Bill of Material for the "Aruba S.p.A." company will be easily solved automatically without worrying and relying on manual calculations and assumptions.

5.4.2.1.4 Execution Example

This subsubsection will be reserved to describe how the "Allocation Engine" works, in practice, by executing it on two main examples: "Any / XX Example" and "Parity / PP Example"[11].
In both cases let's assume that:

- The Data to be Allocated are the following:

Legal Entity	Business Line	L1 - Material	SignData
E1	BL1	#	100
E1	BL2	#	100
E2	BL1	#	100
E2	BL3	#	100

Table 5.8: Allocation Engine Execution: Data to be Allocated

- The Drivers for performing the allocation are the following:

Legal Entity	Business Line	L1 - Material	SignData
#	BL1	MAT1	25
#	BL2	MAT1	25
#	BL3	MAT1	50
#	BL1	MAT2	75
#	BL2	MAT2	75
#	BL3	MAT2	50

Table 5.9: Allocation Engine Execution: Drivers

5.4.2.1.4 "Any / XX" Example

In this example, suppose the "Allocation Rules" to be used are the following:

[11]Only three data model dimensions, for simplicity, will be used on those examples. Moreover, fake data will also be utilized.

Allocating Dimension	Legal Entity	Business Line	L1 - Material	Legal Entity	Business Line	L1 - Material
L1 - Material	XX	XX	#	XX	XX	XX

Table 5.10: "Any / XX" Example: Allocation Rules

Then the "Allocation Engine" compute the following percentages, given those "Allocation Rules" and previous drivers:

Legal Entity	Business Line	L1- Material	SignData	Sum	%
#	BL1	MAT1	25	300	1/12
#	BL2	MAT1	25	300	1/12
#	BL3	MAT1	50	300	1/6
#	BL1	MAT2	75	300	1/4
#	BL2	MAT2	75	300	1/4
#	BL3	MAT2	50	300	1/6

Table 5.11: "Any / XX" Example: Percentages

Therefore, the "Allocated Data" are:

Legal Entity	Business Line	L1- Material	SignData
E1	BL1	MAT1	33.33
E1	BL1	MAT2	66.66
E1	BL2	MAT1	33.33
E1	BL2	MAT2	66.66
E2	BL1	MAT1	33.33
E2	BL1	MAT2	66.66
E2	BL3	MAT1	33.33
E2	BL3	MAT2	66.66

Table 5.12: "Any / XX" Example: Allocated Data

5.4.2.1.4 "Parity / PP" Example

In this example, suppose the "Allocation Rules" to be used are the following:

Allocating Dimension	Legal Entity	Business Line	L1 - Material	Legal Entity	Business Line	L1 - Material
L1 - Material	XX	XX	#	XX	PP	XX

Table 5.13: "Parity / PP" Example: Allocation Rules

Then the "Allocation Engine" compute the following percentages, given those "Allocation Rules" and previous drivers:

Legal Entity	Business Line	L1- Material	SignData	Sum	%
#	BL1	MAT1	25	100	1/4
#	BL2	MAT1	25	300	1/4
#	BL3	MAT1	50	300	1/2
#	BL1	MAT2	75	300	3/4
#	BL2	MAT2	75	300	3/4
#	BL3	MAT2	50	300	1/2

Table 5.14: "Parity / PP" Example: Percentages

Therefore, the "Allocated Data" are:

Legal Entity	Business Line	L1- Material	SignData
E1	BL1	MAT1	25
E1	BL1	MAT2	75
E1	BL2	MAT1	25
E1	BL2	MAT2	75
E2	BL1	MAT1	25
E2	BL1	MAT2	75
E2	BL3	MAT1	50
E2	BL3	MAT2	50

Table 5.15: "Parity / PP" Example: Allocated Data

5.4.3 The Creation of the Managerial or Management Consolidation

The solution proposed for the creation of the Managerial Consolidation is at the same time the simplest, among the three problems introduced, but also the longest in terms of the implementation time required. The reasons behind this strong statement can be brought back to the infrastructure used for implementing the overall solutions to the problems. In particular, generating the Managerial or Management Consolidation is pretty easy given the technology tools defined. In fact, in the particular case of the "Aruba S.p.A." company consolidating the data for defining the Managerial Consolidation consists of only eliminating the transactions between the companies that belong to the "Aruba S.p.A." that means, removing records that have as a client one of the organizations that compose Aruba.
To make things more clear consider the following example:

Accounting Nature	**Legal Entity**	**Client**	**Other Data Model Dimensions**	**Value**
C00000001	Aruba S.p.A. Company 1	Aruba S.p.A. Company 2	Other values	590€

Table 5.16: "Aruba S.p.A." Project: Company-Company Exchange Example

As can be seen, this is a record that must be not considered to create the consolidation. Instead, if it is like the following:

Accounting Nature	**Legal Entity**	**Client**	**Other Data Model Dimensions**	**Value**
C00000001	Aruba S.p.A. Company 1	Client 1	Other values	850€

Table 5.17: "Aruba S.p.A." Project: Company-Client Exchange Example

Then, it must have been counted for the consolidation purposes. To do so, however, the EPM tool SAP BPC must be used, which requires that all the underneath infrastructure is implemented as well as a BPF correctly configured for generating the Consolidation. Given that a BPF must be configured to Consolidate the data, a smart choice that was made by the "BPC" Team of PwC was to unify the user access for all the operations: Data Integration and Management, Allocations, Consolidation, Reporting, etc., making it the tool with AFO, only used from the final user side[12] for this application.

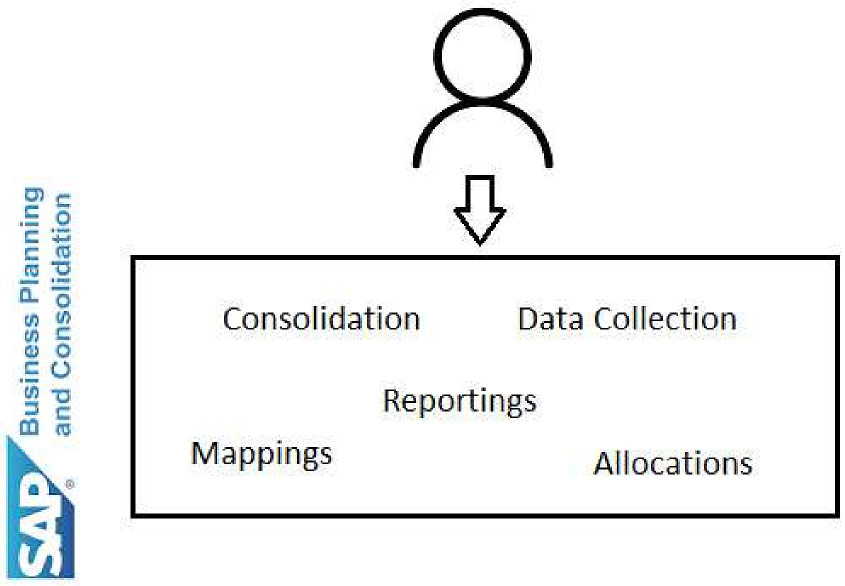

Figure 5.8: "Aruba S.p.A." Project: SAP BPC as Clients Access

Therefore, to make this happen the BPF was organized into macro steps where several operations will be performed. In particular, those macro steps are:

- Initialization: this macro step will update:
  - the values stored in all the dimensions by launching a loading process that generate those data by retrieving from SAP S/4HANA (supposing that all the organization have switched to such ERP);
  - the ownership of the consolidation i.e. one of the technical parameters necessary to generate the managerial or management consolidation;
  - the currency conversion rates necessary for converting the values in one currency to another one;

- Data Collection: here is where all the tasks necessary to solve the Data Integration problem have been placed. In particular:

[12]SAP S/4HANA is also used by some users to record data, but, from the context of the "BPC" Team of PwC only AFO and SAP BPC are the tools used by the final users.

  - Imports of SAP S/4HANA, "Onda" or "Oceano" Management Systems, and other sources data;
  - Mappings necessary for aggregating the data into a single source i.e. populate the BPC Layer correctly;

- Allocations Pre-Consolidation: here is where the allocations either general or for the BOM will happen. In particular, this macro step allows to :
  - Insert Allocation rules: by manually inserting them or copying them from previous periods;
  - Insert Drivers for the Allocation: by manually inserting or loading them through a ".csv" file;
  - Run the Allocation i.e. execute the "Allocation Engine";

- Chart of Accounts Intended population: this macro step is reserved for the population of the "Chart of Accounts Intended" dimension used for BI purposes;
- Consolidation: this section generates the Managerial or Management Consolidation, by applying the correct rules and displaying the data consolidated into reports;
- Allocation Post-Consolidation: this section is the same as the "Allocations Pre-Consolidation" but the data used here are the ones that have been consolidated;
- Reports: this section contains all the BI reports that the Management Control team of Aruba uses to make their business decisions;

6 Conclusions

This paper has tried to describe in detail, from the technical perspective, how a project inside PwC is managed and developed. In particular, this work was focused on developing a solution for the client company "Aruba S.p.A.". In fact, in this internship, several technologies have been explored and used. To be precise, the SAP BW/4 HANA Data warehouse and SAP BPC with Analysis for Office as the main EPM Technologies. Such technologies allowed the development of a stable, highly efficient solution for problems like Data Integration and Business Management. Integrating data from different sources coming from several ERPs and different, heterogeneous Management Systems is a task very common to solve with the usage of a Data Warehouse. Likewise, Business Management is not hard to deal with if a stable EPM solution is implemented. A problem like creating Managerial Consolidation is, in fact, very simple to manage with the help of SAP BPC. Not as easy as the previous two problems, instead, is the computation of costs and revenues of a Bill of Material. This, requires a little more effort to solve it. For such a reason a custom solution was developed, successfully, called "Allocation Engine" able to re-partition the costs and revenues of single components of a Bill of Material. Finally thanks to software like Analysis for Office and SAP BPC several Business Intelligence's reports were designed for "Aruba S.p.A.". So, all of such implementations have worked well with the expected results. However, some space for improvement is still available. The first one is related to the "Allocation Engine" itself. In particular, it is related to its performance. Mainly the solution developed has a bottleneck that resides on the "Expansions of Rules" part of the "Allocation Engine". Simply this has an impact on the number of hierarchy nodes, at most, in a single Allocation Rule can be selected. So, a better, future, solution, for expanding such rules, would be, to parallelize the "expansion part", making it faster. Additionally, but more on the business-related side, another important future improvement is to extend the solution developed also for the Legal Consolidation, making it therefore completed. Finally, the last bit of improvement that can be made is on the Business Intelligence side. At the moment, the Business Intelligence developed is mainly concentrated on simple, highly effective reports based on Excel. A better, robust, and powerful solution is to use for those, another SAP software called SAP Analytics Cloud (SAC) whose goal is to provide a flexible yet simple usage for both developers and final users.

References

[1] *Acceptance testing*. https://en.wikipedia.org/wiki/Acceptance_testing.

[2] *Aruba (company)*. https://it.wikipedia.org/wiki/Aruba_(azienda).

[3] *Best Enterprise Performance Management Software*. https://aimultiple.com/epm-system/.

[4] *Bill of Materials*. https://en.wikipedia.org/wiki/Bill_of_materials.

[5] *Business Intelligence*. https://en.wikipedia.org/wiki/Business_intelligence.

[6] *Consolidation: what is the difference between Legal Management Consolidation*. https://trijotech.com/entries/general/consolidation-series-ii-what-is-the-difference-between-legal---management-consolidation-.

[7] *Data Integration*. https://en.wikipedia.org/wiki/Data_integration.

[8] *Data Mart*. https://en.wikipedia.org/wiki/Data_mart.

[9] *Data Warehouse*. https://en.wikipedia.org/wiki/Data_warehouse.

[10] *Data Warehouse Market*. https://www.alliedmarketresearch.com/data-warehousing-market.

[11] *Data Warehouse Market Share*. https://6sense.com/tech/data-warehousing/sap-data-warehouse-cloud-market-share.

[12] *Difference between Functional Testing and Unit Testing*. https://www.browserstack.com/guide/difference-between-functional-testing-and-unit-testing.

[13] *Enterprise Resource Planning*. https://en.wikipedia.org/wiki/Enterprise_resource_planning.

[14] *Management Consolidation 101.* https://www.lucanet.com/blog/simply-finance/management-consolidation-101.

[15] *OLAP: Online Analytical Processing.* https://en.wikipedia.org/wiki/Online_analytical_processing.

[16] *OLTP: Online transaction processing.* https://en.wikipedia.org/wiki/Online_transaction_processing.

[17] *Oracle : What is a Data Warehouse.* https://www.oracle.com/database/what-is-a-data-warehouse.

[18] *SAP.* https://en.wikipedia.org/wiki/SAP.

[19] *SAP : What is a Data Warehouse?* https://www.sap.com/products/technology-platform/datasphere/what-is-a-data-warehouse.html.

[20] *SAP Business Planning and Consolidation.* https://shorturl.at/gtwLP.

[21] *SAP BW4 HANA.* https://shorturl.at/fixKO.

[22] *SAP BW4 HANA: Aggregation Level InfoProviders.* https://help.sap.com/saphelp_gbt10/helpdata/EN/f8/cab4db576742438c832d7a3d728970/frameset.htm.

[23] *SAP BW4 HANA: Implementing Planning Function Types.* https://shorturl.at/fqI79.

[24] *SAP BW4 HANA: InfoProviders.* https://help.sap.com/saphelp_SCM700_ehp02/helpdata/en/4a/407a10acb51cece10000000a42189b/content.htm?no_cache=true.

[25] *SAP BW4 HANA: Planning Concepts.* https://shorturl.at/eCGM4.

[26] *SAP HANA.* https://en.wikipedia.org/wiki/SAP_HANA.

[27] *System integration testing.* https://en.wikipedia.org/wiki/System_integration_testing.

[28] *Top 10 EPM Software Vendors, Market Size and Market Forecast 2021-2026.* https://www.appsruntheworld.com/top-10-epm-software-vendors-and-market-forecast/.

[29] *What is enterprise performance management (EPM)?* https://www.sap.com/products/technology-platform/cloud-analytics/what-is-enterprise-performance-management.html.

[30] *What is SAP Analysis for Microsoft Office?* https://shorturl.at/tyHIV.

Acknowledgments

The first thing that I would like to say here is "thank you" to everyone who has been on this amazing journey besides me and who I had the pleasure to meet during these years.
The first Person and Institution that I would like to thank in this part are the University of Padua and my supervisor, for this work, Professor Gianmaria Silvello who allowed me to come across PwC and its amazing, responsible and talented teams and persons that I have had the pleasure to know during these last six months. In particular, among those teams, I had to thank all the people at PwC's team in Rubano and Treviso for the warmest welcoming, highest patience, and kindness I have ever seen before. A big thanks also to "Aruba S.p.A." and the PwC team where I worked for helping me with the working aspects and for allowing me to learn a lot of new things. From the university side instead, the persons who deserve to be thanked are all my mates who have known me and worked on several different projects with me and appreciated my hectic work mentality, even if sometimes I have asked them too much.
Coming to the more personal side I would like to thank my family, in particular my parents for the huge opportunity they gave me and the amazing sacrifice and hard work they put in even when things were not going well. I would like to thank them also for the amazing support they gave me because they allowed me to obtain this huge dream without worrying too much about the economic aspects. Additionally, I like to thank my older sister for the support and advice she gave me from her previous experiences. An important and huge thank you also to my group of friends: Federico, Matteo, and Simone during these years they always gave me the energy and motivation to continue and push the limit further to obtain this dream that I wanted so badly. Finally, a huge thank also to myself that against all odds, adversity, bullying, and physical and mental problems you always had your feet on the ground and knew what was better

to do to achieve this accomplishment even if sometimes you have not made the smartest decision. However even if this is the end of a long journey and dream that started way back in 2005 when I first came across a personal computer, I have still unfinished business and goals that I would like to achieve, but, for those the future will reserve me what I need. The thing is that "I am proud but I never will be satisfied with anything".