A Layered Architecture Approach for Large-Scale Data Warehouse Systems

Conference Paper · May 2012

DOI: 10.1007/978-3-642-38370-0_20

Veit Köppen
Zentral- und Landesbibliothek Berlin

Andreas Lübcke
Ultra Tendency LLC

Gunter Saake
Otto-von-Guericke-Universität Magdeburg

A Layered Architecture Approach for Large-Scale Data Warehouse Systems

Thorsten Winsemann[1,2], Veit Köppen[2], Andreas Lübcke[2], and Gunter Saake[2]

[1] SAP Deutschland AG & Co. KG, Großer Grasbrook 17, 22047 Hamburg, Germany
Thorsten.Winsemann@t-online.de
[2] Otto-von-Guericke-Universität, Universitätsplatz 2, 39106 Magdeburg, Germany
{Veit.Koeppen,Andreas.Luebcke,Gunter.Saake}@ovgu.de

Abstract. A proper architecture is significant to cope with complex requirements of today's large-scale Data Warehouses. We compare the assignment of so-called reference architectures with an architecture of dedicated layers to satisfactorily face those requirements. Moreover, we point out additional expenses and resulting advantages of this layered approach.

1 Introduction

We define Enterprise Data Warehouses (EDW) as large-scale Data Warehouses (DW) for supporting decision-making on all organizational levels and in all divisions. EDW are an important basis for applications such as Business Intelligence, planning, and Customer Relationship Management. As they are embedded in an enterprise-wide system landscape, EDW must provide a common view on centralized, accurate, harmonized, consistent, and integrated data. They cover all areas of an enterprise, including the collection and distribution of huge amounts of data with a multitude of heterogeneous sources. World-wide range of use means that data from different time-zones must be integrated. Frequently, 24x7-hours data availability has to be guaranteed, facing the problem of concurrent loading and querying. There are high requirements to data: ad-hoc access, near real-time availability, high quality, fine-granular levels of detail, and a long time horizon. Moreover, new or changing needs for information must be flexibly and promptly satisfied. Such complex requirements for EDW need enhancements and refinements to the architecture, compared to traditional ones, which we outline in this paper.

2 Traditional vs. Layered Architecture

In literature, DW reference architectures are build with three to five rather rough levels [1, 2, 3, 4]; Figure 1 shows a simplified model. Data are extracted from sources into the staging area and transformed to a common schema. Afterwards, they are loaded into the basis database and stored usually at the extracted level of detail. Based on this, data marts are built, i.e. redundant copies of data that are defined according to the users' requirements. The operational data store is used for special business cases. Practical experiences show that this rough architectural model is insufficient. Data

H.C. Mayr et al. (Eds.): UNISCON 2012, LNBIP 137, pp. 199–201, 2013.

access, e.g., for analyses, does not only occurs at data mart level, but also in the basis database, and data transformation is not restricted to the staging area, but takes place in all DW areas; cf. [5]. Besides, complexity of data processing is not expressed adequately – although it is a big problem when building and operating a DW [3, 4]. Complex processing and transformation requires a comprehensive classification within a suitable, layered architecture. It describes levels of data alteration; i.e., layers become more detailed, dedicated, and purposeful. An example is SAP's "Layered, Scalable Architecture" [5, 6, 7]; see Figure 1. We describe the layered approach based on this example, even though it is not limited to SAP-based systems.

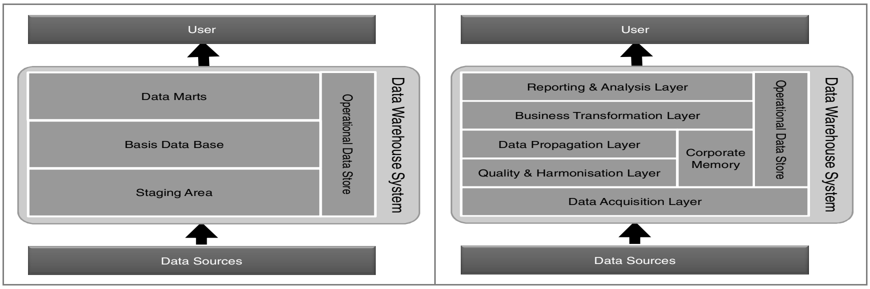

Fig. 1. Traditional DW Reference Architecture vs. Layered, Scalable Architecture

Just as the traditional approach, the layered one is a reference for designing architectures according to individual and actual requirements. However, it is more dedicated, as each layer represents a designated area, in which data are changed w.r.t. their actual format and usage. Extracted data are stored without any transformations in the *Data Acquisition Layer*. The source system is decoupled immediately and no longer strained after loading. Successfully processed data are deleted after a certain time. Yet, if they are stored in the long term, a *Corporate Memory* can be built which enables access to all loaded data. For easier reuse, administrative information can be added, e.g., data origin or timestamp. As the use is rare, storage mediums can be slower and cheaper ones. In the *Quality & Harmonisation Layer*, all operations to integrate data are made, cf. [8]. The result is harmonized and integrated data, which are stored in the *Data Propagation Layer*. It represents a single source of truth and is basis for any further data usage. Therefore, the meaning of data in here must have a clear, common, and company-wide understanding. Data are kept as granular as possible with an adequate time horizon, and the absence of business logic offers high flexibility for further data usage. In the *Business Transformation Layer*, data are transformed regarding business needs; e.g., key figures' computation or combination of data from different areas. The *Reporting & Analysis Layer* offers data "ready-to-use". Data persistence can be necessary to enhance query performance or due to complex data transformation. Yet, data can also be read from layers below. The *Operational Data Store* is mainly dedicated for special needs of data with near real-time availability.

3 Efforts and Benefits of the Layered Approach

A layered architecture initially covers complete business areas and combines data according to current needs in a subsequent step. In contrast, the traditional approach defines DW based on actual users' requirements. This leads to higher complexity and higher volume of extracted data. More conceptual work is necessary, so that duration and costs of the implementation are affected. However, we illustrate that several advantages justify a layered architecture. Transformation rules have to be changed as the transformation of loaded data was erroneous; e.g., currency conversions based on overaged rates. In traditional architectures, data are usually not available in a reusable format, and must be reloaded from sources. In our approach, the propagation layer holds data in a format that enables rebuilding into the overlying levels. Further, data have to be remodeled for new business requirements, e.g., sales regions are restructured. Again, data reloading from sources is usually the solution in traditional architecture, whereas computation can start from propagation layer in the layered approach. Business often requires data that are not included in the initial concept of DW's design. Even in case such data are part of already connected sources, they are not included in data extraction in traditional architectures. Hence, the data flow from source to DW must be enhanced and all data must be loaded again. In the layered architecture, we extract all potentially relevant data into the DW when a new source is extracted. Most likely, data are already kept in the propagation layer or the corporate memory, including previous ones, so that remodeling data flows from sources and extracting missing data is not necessary. Hereby, organizational problems are avoided, e.g., downtimes of source systems during extraction. As data are kept free of business logic up to the propagation layer, they must be loaded only once into the DW and are deployed several times. Hereby, system load, due to redundant extraction, staging, and storage of data, is avoided. Moreover, data remain in the corporate memory even when they are already deleted or changed in the source system. Defining a single source of truth of data supports a common, company-wide view on integrated and trustable data, w.r.t. data governance. Thereby, local or departmental data marts contain reliable information, too. Besides, data load and data supply for reporting is decoupled, and data from different time zones can be released simultaneously. Moreover, a layered architecture eases scalability. Due to openness, modularity, and flexibility of this concept, the system can easily be enhanced by integrating data streams or copying applications, e.g., in case a new business area is defined. Relevant data are transferred to the DW instantly as extraction is not limited to existing areas. As there is no business-related data modification up to the propagation layer, the DW model is not affected until necessary enhancements in the transformation and reporting layer. Here, initial additional expenses pay off.

References

1. Poe, V.: Building a Data Warehouse for decision support. Prentice Hall PTR (1996)
2. Muksch, H., Behme, W. (eds.): Das Data-Warehouse-Konzept. Gabler (2000)
3. Chamoni, P., Gluchowski, P. (eds.): Analytische Informationssysteme. Springer (2006)
4. Köppen, V., Saake, G., Sattler, K.-U.: Data Warehouse Technologien. Mitp (2012)
5. Winsemann, T., Köppen, V.: Persistence in Enterprise Data Warehouses (2012)
6. SAP AG: PDEBW1 – Layered, Scalable Architecture (LSA) for BW (2009)
7. Winsemann, T., Köppen, V., Saake, G.: A layered architecture for enterprise data warehouse systems. In: Bajec, M., Eder, J. (eds.) CAiSE Workshops 2012. LNBIP, vol. 112, pp. 192–199. Springer, Heidelberg (2012)
8. Leser, U., Naumann, F.: Informationsintegration. Dpunkt (2007)