A Layered Architecture for Enterprise Data Warehouse Systems

Thorsten Winsemann[1,2], Veit Köppen[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,Gunter.Saake}@ovgu.de

Abstract. The architecture of Data Warehouse systems is described on basis of so-called reference architectures. Today's requirements to Enterprise Data Warehouses are often too complex to be satisfactorily achieved by the rather rough descriptions of this reference architecture. We describe an architecture of dedicated layers to face those complex requirements, and point out additional expenses and resulting advantages of our approach compared to the traditional one.

1 Characteristics of an Enterprise Data Warehouse

A *Business Data Warehouse* (BDW, [1]) is a Data Warehouse (DW) to support decisions concerning the business on all organizational levels. It covers all business areas, such as logistics, finance, and controlling. We define Enterprise Data Warehouses (EDW) as subspecies of BDW systems, which not only cover all activities of an enterprise, but are an important basis for applications, such as Business Intelligence, planning, and Customer Relationship Management. An EDW collects and distributes huge amounts of data from a multitude of heterogeneous source systems. It has to provide a single version of truth for all data of the company. That means that there is one common view on centralized, accurate, harmonized, consistent, and integrated data at a given point of time. The range of use is often world-wide, so that data from different time-zones have to be integrated. Frequently, 24x7-hours data availability has to be guaranteed, facing the problem of loading and querying at the same time. In addition, there are high requirements to the data: ad-hoc access, near real-time availability, high data quality, and the need for very detailed and granular data with a long time horizon. Moreover, new or changing requirements for information have to be flexibly and promptly satisfied, and, last but not least, the data access has to be secured by a powerful authorization concept.

These significant requirements for an EDW also necessitate enhancements and refinements to the DW architecture, compared to the traditional one, which is outlined in this paper. Sections 2 and 3 describe the traditional and the layered DW architecture and show a comparative example of both approaches. In Section 4, we discuss additional expenses and resulting advantages of the layered architecture approach in detail. Section 5 concludes our paper and gives an outlook on future work.

M. Bajec and J. Eder (Eds.): CAiSE 2012 Workshops, LNBIP 112, pp. 192–199, 2012.

2 Traditional Data Warehouse Reference Architectures

In literature, several DW architectures are described [e.g., 2, 3, 4], and DW systems are mostly modeled with three to five areas as *reference architectures* [cf. 5, 6, 7, 8]; see Figure 1 for a simplified model. Data are extracted from data sources into the staging area, where they are transformed to the common DW schema. Afterwards, data are loaded into the basis database of the DW, where they are stored at the required level of granularity. Based on this, data marts are built; that means redundant copies of data that are defined according to the users' requirements, for instance for analysis. The operational data store is used for special business cases, such as near real-time reporting. In other words, one can define three aspects of data handling: data acquisition, data processing, and data provision.

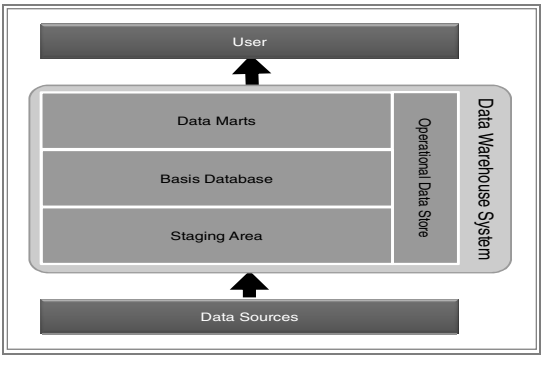

Fig. 1. DW Reference Architecture

In Figure 2, we illustrate this theoretical approach with an example from the business area of sales and distribution in general, and the sales documents of orders and invoices in particular. In an Enterprise Resource and Planning system (ERP), data are kept in four different tables: sales order and sale invoice, header and item data. Data are selected and joined in ERP, extracted to the DW, transformed (i.e., cleansed, harmonized, etc.) in the staging area and loaded into the basis database. Then, data marts are filled that are designed for fast analysis' access.

However, practical experiences show that this rough architectural model is insufficient in several aspects. Data access for analyses does not only take place at the data mart level, but is also applied to the basis database – the boundaries are fluid [9]. ETL (Extraction, Transformation, and Loading) processes in this model consist of different steps of data processing and reach from data source via staging area into the basis database [e.g., 8, 10]. In practice, data marts are often used as sources of the basis database, and data are not only transformed (e.g., enriched with attributes) in the staging area, but also in the basis database and data mart area. When accessing data for analyses, they are usually also transformed (e.g., aggregated or averaged). Therefore, transformation cannot be considered as a process which ends when data are stored in the basis database. It applies over several layers and in all areas of a DW system: data acquisition, data processing, and data provision. Moreover, the complexity of data

processing is not expressed adequately – although it is one of the biggest problems when building and operating a DW [6]. Data transformation processes with more than five levels and execution times of several hours are not rare within a productive EDW environment.

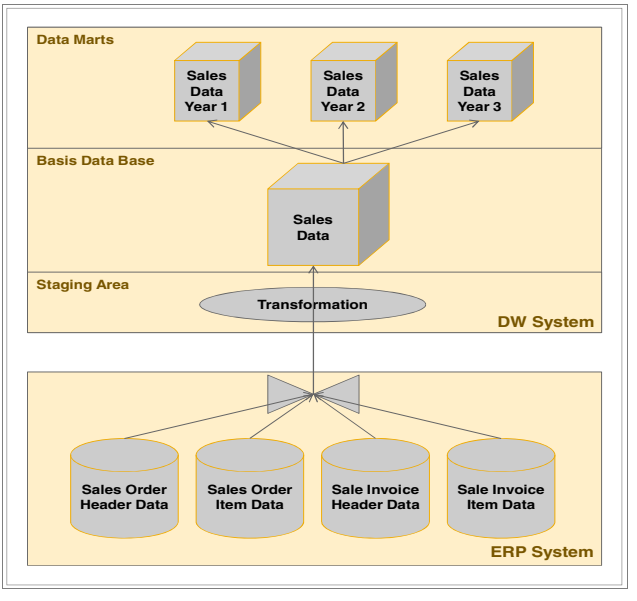

Fig. 2. Data-Flow-Example in a Reference Architecture

3 The Layered Architecture Approach

A layered architecture improves the reference architecture approach with respect to a comprehensive and ordering collection and classification of data transformation. It describes levels of data alteration within a DW from data acquisition to data processing and data provision. Layers become more detailed, dedicated, and purposeful herein.

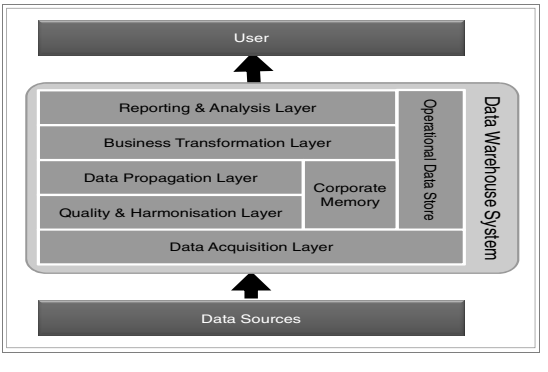

Fig. 3. Layered, Scalable Architecture for EDW [11]

A good example for such architecture is SAP's "Layered, Scalable Architecture (LSA)" for Enterprise Data Warehousing, shown in Figure 3. It is defined as a reference for designing architectures - not limited to SAP-based systems - according to individual and actual requirements.

Extracted data are stored immediately in tables of the *Data Acquisition Layer*, without any transformations. Thereby, the source system is decoupled immediately and no longer strained after successful data transfer.

Data in the acquisition layer are deleted after a certain time when they are successfully loaded to all dedicated data targets. Yet, if it is stored in the long term, it can be defined as *Corporate Memory* and enables access to all loaded data. For easier re-use, some administrative information can be added (e.g., data origin and timestamp). Hence, data are available in the DW, even when they are already deleted or changed in the source system; usually, data cannot be recovered and are lost. Furthermore, re-loading data causes organizational problems, such as necessary down-times. It is advisable to extract not only actual necessary data from one source, but all possibly useful data. Thus, the data base represents a reservoir for unpredictable requirements – without re-modeling data flows from source systems and extracting data that are missing in the DW. By definition, the use is rare, so that storage mediums can be slower and cheaper ones.

In the *Quality & Harmonisation Layer*, necessary integration of data is done; this includes syntactical and semantical integration, schema mapping, consolidation, cleansing, quality-checks, data validation, and de-duplication [cf. 10, 12]. The result is harmonized and integrated data, stored in the overlying layer.

The *Data Propagation Layer* contains harmonized, integrated data. It represents the data's single point of truth as the basis for any further usage. Therefore, it has to be the layer, where a clear, common, and company-wide understanding of the meaning of the data is defined (e.g., "what really does working day mean?", "how is revenue defined?"). Data are kept as granular as possible with an adequate time horizon and free of any business logic. Data are loaded only once into the DW, and are deployed several times, so that further system load, due to redundant extraction, staging, and storage of data, is avoided.

The absence of business logic in the propagation layer offers utmost flexibility for the use of data in the upper layers. Data are transformed regarding the business needs in the *Business Transformation Layer*. Such business needs can be currency conversions with month's end exchange rate, computation of key indicators, or business-defined combination of data from different areas. Our example shows a simple transformation of merging document header and item data. However, one can imagine complex combinations of data from different business areas, such as purchasing and inventory management, or logistics and finance.

The main task of the *Reporting & Analysis Layer* is to offer data as "ready-to-use". Due to a better query performance or complex transformation of data, a further materialization can be necessary (as illustrated for the sales data years 1 – 3 in Figure 4). Yet, the data can also be read from layers below and are presented as a virtual cube for reporting (as shown for "Special Sales Data").

The *Operational Data Store* – which is not implemented in our example – is mainly dedicated for special needs of application-specific, granular data with near real-time availability.

A rough mapping of these layers to the three DW's aspects of data handling can be done: data acquisition is covered within the Data Acquisition and the Quality & Harmonisation Layer, data processing in the Data Propagation and Business Transformation Layer, and data provision in the Reporting & Analysis Layer.

This architecture is a conceptual matter. Each layer represents a designated area, in which data are changed with respect to their actual format and their dedicated usage. The integration of each layer into the individual DW architecture mainly depends on the need for such changes. For instance, a data set does not have to be lifted to the data mart layer if it is already usable on a lower one. Further aspects result from design decisions or DW architects' preferences; for example whether to build a dedicated corporate memory or to combine it with the acquisition layer. The assignment of transformations and layers illustrates the increase of the *data's excellence* – the added value of the data that is reached on each layer.

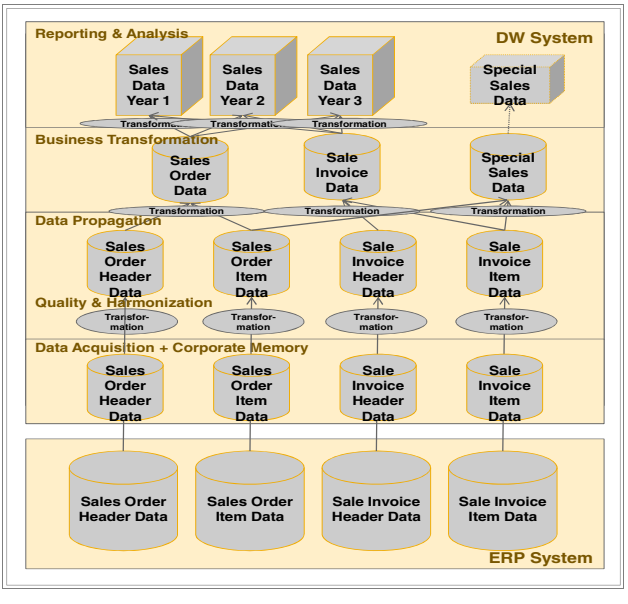

Fig. 4. Data-Flow-Example in a Layered Architecture

We make the difference to the traditional architecture clearer by adapting the data flow shown in Figure 2 to the layered architecture in Figure 4. Data are selected in ERP and extracted to the DW system, usually one data flow per source. In the DW, data are stored unchanged in the acquisition layer, which is also defined as corporate memory. Subsequently, the data are cleaned, quality checked, and harmonized. Afterwards, they are stored into tables in the propagation layer, free of business-related

changes. Such changes are made in the business transformation layer, where the two data sets ("Sales Order" and "Sale Invoice") are merged and a "Special Sales" set is created, in which order and invoice items' data are combined. On top, several data marts (one for each year) are defined for better performance. The "Special Sales Data" mart represents a virtual access point without data and clarifies that reporting is possible on data of each layer.

4 Architectural Differences in Detail

Comparing Figures 2 and 4, the higher complexity of the layered architecture is obvious. It initially covers complete business areas and combines data according to actual requirements in a subsequent step. This leads to a higher volume of data to be extracted to the DW. Besides, more conceptual work is necessary than in the traditional approach of defining the DW on base of the actual users' requirements. Hereby, duration and costs of the implementation project are affected. However, there are several advantages which justify this kind of architecture. We illustrate these advantages by exemplarily present some common activities in an EDW, such as change of transformation rules, change of data, and need for new data.

The transformation of data loaded into the DW can be erroneous: incorrect computation of key indicators, currency conversion based on over aged rates, or assignment of countries to regions has changed. In the traditional architecture, data are usually not available in a re-usable format and must be re-loaded from source systems. In contrast, the propagation layer holds data in a format that enables a smooth re-building into the overlying levels. Occasionally, data must be combined differently due to new business requirements; for instance, sales regions are restructured or new key indicators must be computed. Again, data re-loading is usually done in traditional architecture, whereas computation can be started from the propagation layer in the layered architecture. Often, business requires data that are not included in the initial concept of the DW's design. Even in case these new data are part of already connected sources, they are not included in the extraction. Hence, the only way in the traditional architecture is an enhancement of the data flow from the source to the DW, followed by a complete new loading. As pronounced above, the approach of the layered architecture is to get all possibly relevant pieces of information into the DW when a new source is extracted. Therefore, the probability is high, that such data are already kept in the corporate memory or in the propagation layer, including previous data.

The above mentioned examples clarify that the layered architecture offers fast ways to satisfy new or changing needs for information, which are frequent use cases. Furthermore, defining data in the propagation layer as single point of truth supports a common, company-wide view on integrated and trustable data with respect to data governance. Based on this, even local or departmental data marts, or such created for special purposes, contain reliable information. Moreover, the data stream of loading and using for reporting is decoupled. By this means, it compensates extraction of data from different time zones, which are released into the DW's basis database at one defined point in time. Continuing, the data availability with detailed previous data is a highlight, too. Moreover, the layered architecture offers easy possibilities of

scalability, especially when a proper naming convention has been established. Due to openness and flexibility of the layer concept, the system can easily be enhanced by integrating data streams or copying of applications. Here, the initial additional expenses pay off. This becomes necessary, whenever data from another region, country, business area or company have to be absorbed into the EDW. For example, a new business area is defined in ERP. As data extraction is not limited to existing ones, such data are transferred to the DW instantly. As there is no business-related data modification up to the propagation layer, the DW is not affected by this enhancement until modeling is necessary in the business transformation and reporting layer.

[2] points out several "metrics for assessing architecture success", such as *information quality* (i.e., accuracy, completeness, and consistency of data) and *system quality* (i.e., flexibility, scalability, and integration of the system). In view of EDW's complexity, a layered architecture offers good means to cope with it.

5 Conclusion and Outlook

We introduce the main characteristics of an EDW and explain the traditional reference architecture for DW systems. We illustrate the layered architecture on base of SAP's "Layered, Scalable Architecture", which represents a common and field-tested example of such an approach. It defines layers as designated areas, in which data are changed with regard to their actual format and their dedicated usage. Compared to the traditional approach, building a DW with a layered architecture needs more conceptual work and additional expenses. Certainly, it enables a much clearer, dedicated assignment of data transformation to each layer. The advantages of the layered architecture outbalance the traditional one, regarding consistency of data, flexibility, reusability, and scalability – especially with respect to EDW's complexity. This shall not only be in mind when building a new DW, but even lead to think about re-designing existing ones.

Future work will include detailed, exemplary illustrations as well as validations of the described layered approach by comparing it to the traditional reference architecture on base of SAP and non-SAP DW platforms.

References

1. Devlin, B.A., Murphy, P.T.: An architecture for a business and information system. IBM Systems Journal 27(1), 60–80 (1988)
2. Watson, H.J., Ariyachandra, T.: Data Warehouse Architectures: Factors in the Selection and the Success of the Architectures (2005), `http://www.terry.uga.edu/~hwatson/DW_Architecture_Report.pdf` (call April 01, 2012)
3. Hajmoosaei, A., Kashfi, M., Kailasam, P.: Comparison plan for data warehouse system architectures. In: 3rd ICMiA Proceedings, pp. 290–293 (2011)
4. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill (2009)
5. Poe, V.: Building a Data Warehouse for decision support. Prentice Hall (1996)

6. Chamoni, P., Gluchowski, P. (eds.): Analytische Informationssysteme. Springer (2006)
7. Muksch, H., Behme, W. (eds.): Das Data-Warehouse-Konzept. Gabler (2000)
8. Zeh, T.: Referenzmodell für die Architektur von Data-Warehouse-Systemen (Referenzarchitektur) (2008), http://www.tzeh.de/doc/gse-ra.ppt (call April 01, 2012)
9. Lehmann, P.: Meta-Datenmanagement in Data-Warehouse-Systemen. PhD Thesis, University of Magdeburg (2001)
10. Leser, U., Naumann, F.: Informationsintegration. dpunkt (2007)
11. SAP AG: Layered, Scalable Architecture (LSA) for BW, Training Material (2009)
12. Lehner, W.: Datenbanktechnologie für Data-Warehouse-Systeme. dpunkt (2003)