Technical report

Nr.: FIN-002-2012

Persistence in Enterprise Data Warehouses

Thorsten Winsemann, Veit Köppen

Arbeitsgruppe Datenbanken

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Nr.: FIN-002-2012

Persistence in Enterprise Data Warehouses

Thorsten Winsemann, Veit Köppen

Arbeitsgruppe Datenbanken

Technical report (Internet)
Elektronische Zeitschriftenreihe
der Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg
ISSN 1869-5078

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Thorsten Winsemann
Postfach 4120
39016 Magdeburg
E-Mail: thorsten.winsemann@t-online.de

http://www.cs.uni-magdeburg.de/Technical_reports.html

Technical report (Internet)
ISSN 1869-5078

Redaktionsschluss: 01.03.2012

Bezug: Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Dekanat

Persistence in Enterprise Data Warehouses

Thorsten Winsemann, Veit Köppen

School of Computer Science
Otto-von-Guericke University Magdeburg, Germany
thorsten.winsemann@t-online.de, veit.koeppen@ovgu.de

Abstract. Yet, persistence of redundant data in Data Warehouses is often simply justified with an achievement of better performance when accessing data for analysis and reporting. Especially in Enterprise Data Warehouse systems, data management via multiple persistence levels is necessary to condition the huge amount of data into an adequate format for its final usage. However, there are further reasons to store data persistently, which are often not recognized when designing such warehouses. As processing and maintenance of data is complex and requires huge effort, less redundancy will downsize effort. Latest in-memory technologies enable good response times for data access, so that the question arises, what data for what purposes really need to be stored persistently - and how can this be efficiently decided. We present a compendium of purposes for data persistence and use it as a basis for decision-making whether to store data or not. As an outlook, we expand this discussion on Enterprise Data Warehouses based on in-memory databases.

Keywords: Enterprise Data Warehouse, Persistence, In-Memory Database

1 Introduction

Today‘s Data Warehouses (DW) are often characterized by enormous data volumes [Wi08]. When designing and operating a DW, there are high requirements regarding data provision, such as performance, granularity, flexibility, and actuality. Besides, restrictions call for additional, redundant data. For example, materialized views and summarization levels are commonly used for enhancing speed of data access, such as for reporting and analysis. Performance is still the main reason for storing data redundantly, but there are several other reasons which lead to additional data persistence, as for instance design decisions, usability, or safety. Such reasons are often underestimated or even not considered, but frequently motive for storing data. Yet, additional storage of data always requires a huge effort to guarantee consistency and limits prompt data availability.

Latest announcements pledge technology, based on in-memory databases (IMDB), to get rid of any redundant data without any loss of performance [P+09]. Assuming that performance of data access – as the main reason for storing data – is much less important in IMDB, the question arises, which data persistence is really necessary in DW? Is it possible to do any kind of reporting on raw data that is transformed on-the-fly, according to its usage? Do any reasons for storing data remain? In order to answer these questions, we list purposes for persistence, which we clarify by examples, and point out potential conflicts between data storage and usage. Beyond, we define

indicators for supporting decisions on whether to store data or not. This paper is a first step to discuss data persistence in changing DW environments. Note, such considerations are also valid, but less important for DW based on relational databases (RDBMS), as simply the lower performance of the database (DB) motivates additional data storage.
This paper is organized as follows: Section 2 introduces briefly to specialities of Enterprise Data Warehouses (EDW) and a layered architecture. In Section 3, we present a compendium of reasons for data persistence in today's EDW, grouped and amended by examples, and define a diagram for supporting the decision whether to store data. Moreover, we describe potential conflicts arising from requirements of data usage and tasks to fulfill them. In Section 4, we present prospects of data persistence in in-memory based EDW. Section 5 gives an overview on related work, and Section 6 concludes the paper with an outlook on future work.

2 A Layered Architecture for Enterprise Data Warehouses

An Enterprise Data Warehouse is a Business Data Warehouse [P+09], thus supporting management's decisions and covering all business areas. Beyond, EDW are an important basis for several applications, such as Business Intelligence (BI), planning and Customer Relationship Management (CRM). As they are embedded in an enterprise-wide system landscape, an EDW has to provide a single version of truth for all analytical data of a company. That means, there is a common view on centralized, accurate, harmonized, consistent, and integrated data to treat information as corporate assets. An EDW covers all domains of a company or even a corporate group, including collection and distribution of data with heterogenous source systems and a huge amount of data. The range of use is often world-wide, so data from different time-zones have to be integrated. Frequently, 24x7 data availability has to be guaranteed, lacking of regular off-peak-hours and facing the problem of loading and querying at the same time. Despite this, there are other quite complex requirements on data: ad-hoc access, near real-time actuality, and often applications, such as CRM, that require very detailed and fine-granular data with a long time horizon. Furthermore, there has to be a flexible and prompt way to satisfy new or changing needs for information, and, last but not least, the access to sensitive data has to be secured by a powerful authorization concept. Therefore, several reasons for persistence are quite particular for EDW.

Persistence in a Data Warehouse is closely connected to its architecture. That means, the decision of storing data comes along with the data's format and the area or layer, where data are stored. Excluding the data sources, common reference architectures – see for instance [PR96], [MB00], [GC06], and [Ze08] – define three main areas, representing three aspects of data handling: data acquisition in the staging area, data processing in the basis data-base, and data provision in the data marts. Within this rather rough model, persistence on each level is implicit [De09]. Regarding EDW, a layered architecture as introduced by [SA09] refines this approach (cf. Figure 1). Herein, the layers become more detailed and dedicated. Each of the five layers represents an area for increasing the data's value with respect to the usage – if necessary. That means, for instance, that a data set does not have to be lifted to the

highest level of data marts (and stored there), if it is already usable (for reporting etc.) on a lower level. Yet, a layer does not imply data storage by definition. One has to consider the data format and where to store data. Though, persistence has to be decided prior based on purposes for data needs.

A layered architecture (s. Figure 1) is divided in several areas, which we describe briefly in the following.
The *Data Acquisition Layer* represents the extraction phase, the "inbox" of the Warehouse, where incoming data are accepted usually without modification.
In the *Quality & Harmonisation Layer*, data are integrated technically and semantically, including de-duplication, aspects of information integration (cf. [LN07]) etc.; that is, transformation within "conventional" ETL process.
At the *Data Propagation Layer*, the company's data are kept as a single version of truth of harmonized and integrated data, without any business logic; therefore, it defines a common data-base for all applications.
In the *Business Transformation Layer*, data are transformed due to business' needs, which can be dependent on different departement requirements; for instance, order and invoice data are combined for computing open orders' information.
At the *Reporting & Analysis Layer*, data are transformed mainly according to requirements of usage (e.g., computing rolling periods' values) and to enable fast access to the data.
Within the *Operational Data Provider*, data is simply transformed for specific business cases (e.g., near real-time reporting).
Although the boundaries are shifting, a rough classification of these layers to the three warehouse's aspects of data handling can be done: Data acquisition is covered within the Data Aquisition and the Quality & Harmonisation Layer, data processing in the Data Propagation and Business Transformation Layer, and data provision in the Reporting & Analysis Layer.
[W+11] gives a more detailed description of layered architectures and its comparison.

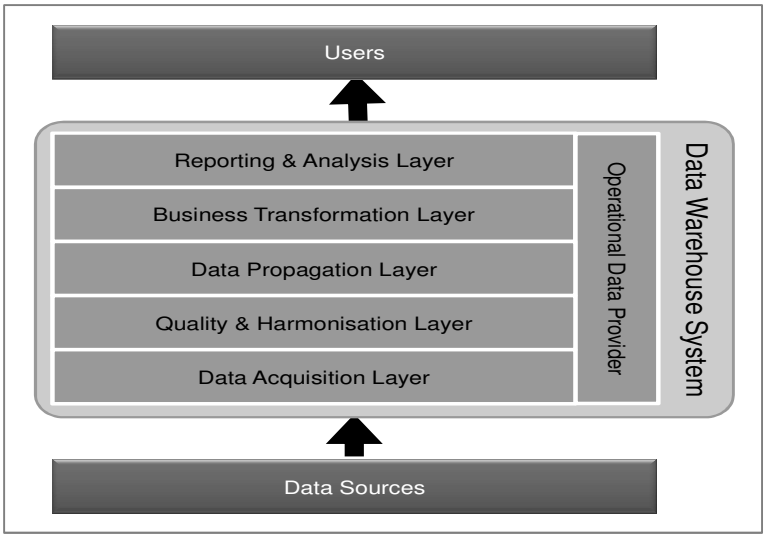

Figure 1: Layered Architecture for EDW (based on [SA09])

3 Reasons for Data Persistence

In this section, we present reasons as well as requirements for data persistence in EDW. In the first part, persistence's purposes are addressed; in the following sub-sections, we group such reasons and define decisions' indicator for data persistence.

3.1 Purposes for Persistence

In literature, mainly two reasons for persistence in DW are mentioned: the storage of data in an already transformed state in the basis data-base and the storage of redundant, aggregated data in the data mart layer. However, there is a broad range of further reasons for storing data in a DW system. They can be based on technical conditions, the company's terms of governance, legal restrictions The ease of data maintenance is another purpose as well as simply subjective needs for safety. As far as we know, reasons for persistence are rarely mentioned in the literature. Hence, we miss a collection as we present and describe in the following. Additionally, we illustrate each reason with one or more examples in detail. Of course, data persistence can also result of the combination of two or several reasons. The listing arranges the purposes in five areas: data acquisition, data modification, data management, data availability, and laws and provisions.

3.1.1 Data Acquisition

Source system decoupling: Enterprise Resource Planning (ERP) systems, as common sources for EDW, have constant, heavy system load: transactional processing during business hours and batch jobs running in the off-hours. Therefore, additional processing due to data extraction has to be minimized. In order to reduce the load, data are stored immediately in the warehouse's inbound layer after successful extraction. Figure 2 illustrates this process in exemplarily. After the extraction is triggered by the DW (1), the data is determined in the ERP (2) and send to the DW, bundled as sets of data packages (3). The last package is accompanied by end-of-extraction information (4); after all data are successfully stored in the staging area of the DW, the process is closed in the ERP system (5). The extracted data are typically stored without any transformation to enhance the time for processing. In some cases, data are enriched by control or informative data, such as date of origin and timestamp.

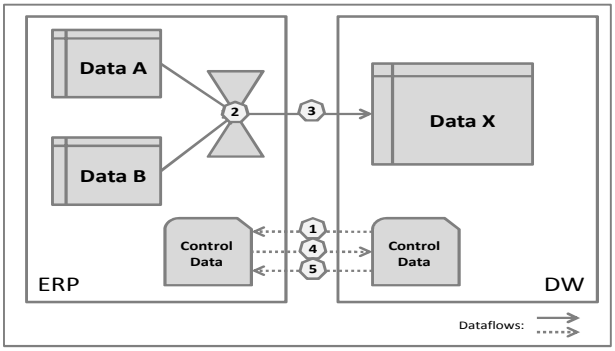

Figure 2: Exemplary (Control) Data Flow while Data Extraction

Data availability: Often, data are no longer available or have been changed in their sources. For instance, this applies to data that are extracted from the internet (e.g., stock exchange quotations), where volatileness of data is common [FS07]. Other examples are: data from files that are overwritten regularly, data from disconnected legacy systems, or data of changing documents, for which no change logs exist (cf. also next purpose). Moreover, this also applies for temporary non-availability; for instance, network connections can be jammed, so data is not accessible when needed.

Extensive data recreation: The recreation of data that are no longer available directly is extensive, both in time and resource consumption. Therefore, such data are stored in DW, usually in an aggregated format. This especially belongs to data in ERP that are no longer used for daily business. Nonetheless, they must be kept and have to be archived, that means copied to external and slower accessed mediums. Another example illustrates changes in documents, in which change logs data are stored differently (e.g., in other tables and/or differing formats) than the actual status of the document.

Data Lineage: The need for reliable data requires verification – often as recurring tasks. Transformed data are very difficult to verify, as usually one or many input data lead to one ore many output data. It becomes nearly impossible, if source data do not exist; only data transformed by filters are reproducible without origins. Moreover, report data often result from multi-level transformation processes; for later validations, backtracking of data origins is essential. Source data or intermediate results are stored to enable or to ease data lineage; see [CW03] for detailed information.

3.1.2 Data Modification

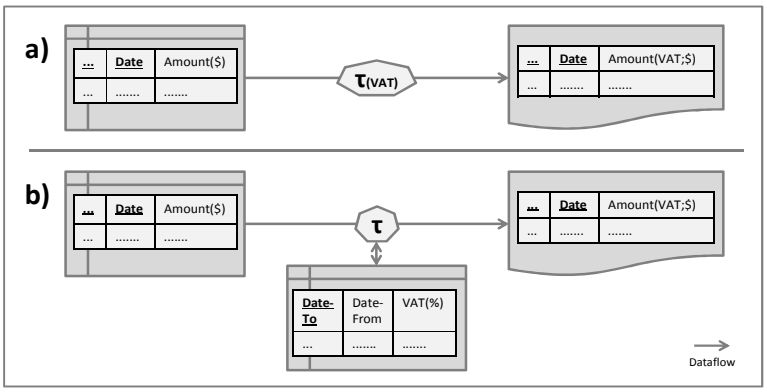

Figure 3: Simple Example for "Changing Transformation Rules"

Changing transformation rules: Transformation rules can change, for instance due to modified business requirements. Unless a timestamp is not part of the data (and the history of the transformation rules is not kept), it will not be possible to transform source data the same way as before. As simple example, imagine changes in sales tax values (VAT). As long as they are a fix part of the transformation rule (s. Figure 3a) as just a scalar data transformer, information is lost when rules are changed, and former results are not reproducible. Here, an alternative solution is rather simple: tax

values are stored separately with their validity periods, and are read during transformation (s. Figure 3b). Nonetheless, transformation rules can be much more complex, including several lines of code, and therefore require extensive logic for variable usage. In our example, we illustrate only one piece of information to access during transformation. Imaging, a transformation needs several pieces of information from different tables of an external system. For each of these tables, an extractor has to be defined, including loading procedures and administration. Often, the figures are hard-coded in the transformation to avoid such a complex setup.

Addicted transformation: As "addicted", we define transformations that need further data from different sources to transform data properly. For instance, the averaged distribution of employees' bonuses requires the total staffs' number (per department and/or period). Therefore, this information has to be available in DW; in Figure 4, the addicted transformation is marked as "τ_a". Correct processing is guaranteed by storing all data that must be available at the time of transformation.

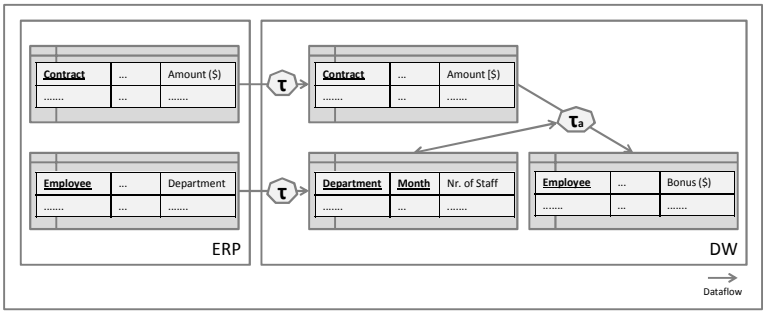

Figure 4: Simplified Example for "Addicted Transformation (τ_a)"

Complex data transformation: Due to complexity, transformation of data can be very expensive regarding computational time and resources. Therefore, data are stored to avoid repeated transformations. In Figure 5, data transformation from data set A to C contains both, complex, time-consuming operations, and processing that needs all tuples for a specified character combination (e.g., percentage and average computation). As complex operations can be done on single tuples, the transformation is split into two parts: "τ_c" includes all complex operations, the result is stored in data set B, and "$\tau(\Sigma,\%,\emptyset)$", in which all tuples are required. Through this, we avoid repeated complex transformation.

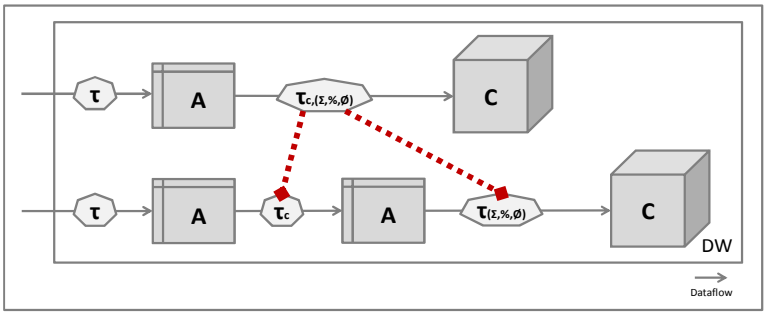

Figure 5: Simplified Example for "Complex Transformation (τ_c)"

Complex, different data: Sometimes, data are very complex and – semantically and syntactically – different to the warehouse's data; hence, they are stored in order to transform it stepwise for its diverse usage. Figure 6 shows an example, in which data is transformed stepwise and stored in each case subsequently. Here, the transformations are: data harmonization and schema integration (τ_H), de-duplication (τ_D), data enrichment (τ_E), and data aggregation ($\tau_{(\Sigma)}$).

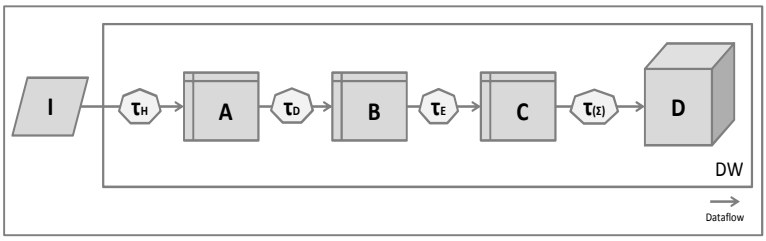

Figure 6: Simplified Example for "Complex, Different Data"

For a more detailed illustration, consider the extraction and processing of point-of-sales (POS) receipts; the example with (an extract of) figures:

Relation of incoming data I:
I : <recno[numc10]; dates[date10]; buyer[numc10]; bname[char60]; amodc[dec10,2]; dcurr[char2]>
{<12345; 31.10.2011; 7410000; Meier; 25,00; EU>;
<12347; 31.10.2011; 6820000; Schulz; 10,00; EU>}

τ_H: recno[numc10] ➔ docnr[char10], dates[date10]➔[date8], buyer[numc10] ➔ custom[char10], bname ➔ cname, amodc[dec10,2]➔[dec10,2], dcurr[char2]➔[char3]

Relation of harmonized and integrated data A:
A : <docnr[char10]; dates[date8]; custom[char10]; cname[char60]; amodc[dec15,2]; dcurr[char3]>
{<0000012345; 20111031; 0007410000; Meier; 25,00; EUR>;
<0000012347; 20111031; 0006820000; Schulz; 10,00; EUR>}

τ_D: 'Meier' ➔ 'Mayr', 'Schulz' ➔ 'Schultz'

Relation of deduped data B:
B = A
{<0000012345; 20111031; 0007410000; Mayr; 25,00; EUR>;
<0000012347; 20111031; 0006820000; Schultz; 10,00; EUR>}

τ_E: Determination of customer class (cuscl[char3])

Relation of enhanced data C:
C: <docnr[char10]; dates[date8]; custom[char10]; cname[char60]; cuscl[char3]; amodc[dec15,2]; dcurr[char3]>
{<0000012345; 20111031; 0007410000; Meier; 015; 25,00; EUR>;
<0000012347; 20111031; 0006820000; Schulz; 015; 10,00;EUR>}

$\tau_{(\Sigma)}$: Aggregation on level month + customer class

Relation of aggregated data D:
D: <month[date6]; cuscl[char5]; amodc[dec15,2]; dcurr[char3]>
{<201110; 015; 35,00; EUR>}

3.1.3 Data Management

Constant data-base: In EDW, updating of data sets with new data is frequent and usually happens up to several times per hour. There are several applications – analysis and especially planning – where users rely on constant data, which must not change for the analysis or planning phase (i.e., a given period of time), and therefore is stored separately. Besides storing, there are other options to separate data, such as versioning or using timestamps. However, such methods enhance all general data with information for a dedicated usage, and lead to rising data volume. Moreover, for example a planning data-base is typically a subset of available data; separation offers faster processing on a smaller data set, and makes data verification easier, too.

En-bloc data supply: Usually, new data trickle into the DW from several sources and temporary shared; especially for EDW, there are several source systems, often external (i.e., non-in-house), world-wide, and situated in different time zones. As a result, data flow continuously and sometimes unsteady into the DW. After integrating them syntactically and semantically, all new data are kept separately and released to the warehouse's basis data-base at a certain point of time. Thus, a "plausibility gate" [Zu11] is established, this ensures defined and plausible data sets for reporting, analysis, and further processing. Figure 7 illustrates a simplified, but common scenario within a sales and distribution application. The DW extracts data from a (company-owned) system for order and invoice management ("OIS"), the deliveries are provided by an external delivery system ("eDS"). Delivery data are sent to the DW once a day during night, whereas order and invoice data flow in every two hours. However, delivery data can be prevented for any reasons. Now, imagine the calculation of "open order quantity" according to following formula:

Open Order Quantity = Order Quantity – Delivery Quantity

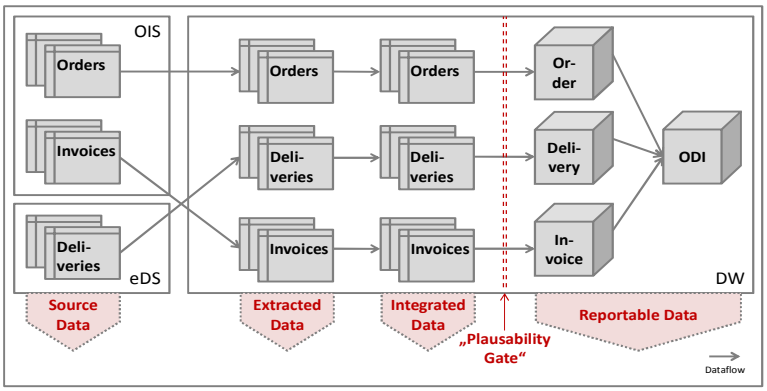

Figure 7: Simplified Example for "En-bloc Data Supply"

If data are reportable as soon as they have been integrated technically, open orders would grow during day, as no adequate delivery data come in. "Open order quantity" would not be plausible in this case; therefore, a "plausibility gate", which enables to release comparable data, is installed with (additional) data storage. Only data that have passed this gate are reportable (i.e., data in the four data cubes in Figure 7).This, of course, is not limited to a system landscape with external systems connected; it

would also be valid even for one-source-systems, if the processing time for the involved data sources differed in execution and processing time.

Complex authorization: DW contain vast amount of information which have to be kept secured for forbidden access – and authorization administration is a sensitive topic and very sensitive (e.g., [W+01]). For instance, it is possible to define access right on dedicated dimensions (e.g., sales organization) and facts (e.g., revenue), and on values within those fields (e.g., sales organization = "North"). Moreover, authorization can be included and excluded. As a result, authorization profiles become quite complex and often confusing. Instead of defining such detailed authorizations, creating dedicated data marts with relevant authorization is often easier.

Single version of truth (SVoT): Data are stored in the basis data-base after being harmonized, syntactically and semantically integrated, according to the company's general rules. As these data remain basically unchanged and free of any explicit business logic, they represent the single version of truth of the company's data, and represent a set of integrated, homogeneous, comparable, and trustable data – with respect to data governance, too. For example, a DW is provided with key-figure "revenue" from both, sales (REV_S) and finance (REV_F) application. REV_S value is in document currency, whereas REV_F is in local currency. Even if, for instance, REV_S is converted into local currency in the DW, both figures might slightly vary, due to different rates or booking dates. Unambiguousness can be reached by defining one of the two figures as "the company's revenue", or denoting them clearly (e.g., "turnover/sales" vs. "revenue/finance"). Thus, a clear, common, and company-wide understanding of the meaning of the data is defined (e.g., "how is revenue defined?", "what really does mean working day?"). Data are kept as granular as possible with an adequate time horizon, and absence of business logic offers utmost flexibility. Therefore, they are basis for further use of any application – meaningless whether it is a report for finance or logistics, for instance. Based on this, even local or departmental data marts, or such created for special purposes, contain reliable information; compare [SA09] and [W+11]. In Figure 8, a simple example is shown. Invoice header and item data are extracted and processed for further usage in the areas of sales and distribution (SD) and controlling (CO). Although the data are processed differently, according to their unlike usage, they refer to the same data-base, the "Invoices" relation.

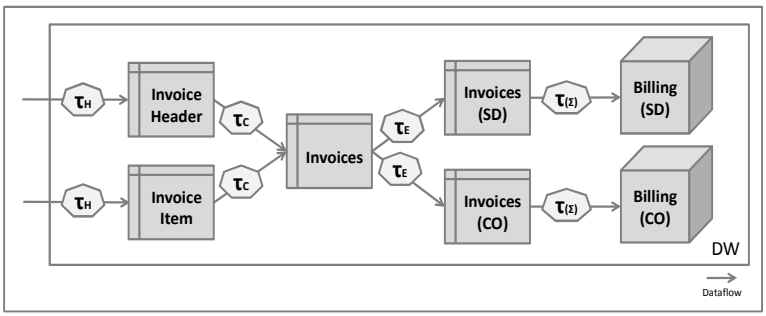

Figure 8: Simplified Example for "Single Version of Truth"

The example is illustrated with figure in the following:

"Invoice Header" relation:
<invnr[char5]; dates[date8]; dateb[date8]; custom[char]
{<12345; 20111031; 20111101; 74100>}

"Invoice Item" relation:
<invnr[char5]; itmnr[char3]; matnr[char6]; quasu[dec15,3]; sunit[char2]; modc[dec15,2]; dcurr[char3]>
{<12345; 001; 000471; 2,000; PC; 20,00; EUR>;
<12345; 002; 000473; 1,000; PC; 5,00; EUR>}

τ_C: Determination of customer class (cuscl[char3])
τ_C: Determination of material class (matcl[char3])

"Invoices" relation:
<invnr[char5]; itmnr[char3]; dates[date8]; dateb[date8]; cuscl[char3]; custom[char5]; matcl[char3]; matnr[char6]; quasu[dec15,3]; sunit[char3]; amodc[dec15,2]; dcurr[char3]>
{<12345; 001; 20111031; 20111101; 003; 74100; 047; 000471; 2,000; PC; 20,00; EUR >
<12345; 002; 20111031; 20111101; 003; 74100; 047; 000473; 1,000; PC; 5,00; EUR>}

τ_E: Determination of sales organization (slorg[char2]), conversion of amount into local currency according to sales date

"Invoices (SD)" relation:
<invnr[char5]; itmnr[char3]; dates[date8]; slorg[char2]; cuscl[char3]; custom[char5]; matcl[char3]; matnr[char6]; quasu[dec15,3]; sunit[char3]; amolc[dec15,2]; lcurr[char3]>
{<12345; 001; 20111031; 01; 003; 74100; 047; 000471; 2,000; PC; 28,00; CHF >
<12345; 002; 20111031; 01; 003; 74100; 047; 000473; 1,000; PC; 7,00; CHF>}

τ_E: Determination of controlling area (conar[char2]), conversion of quantity into base unit + amount into controlling currency according to booking date

"Invoices (CO)" relation:
<invnr[char5]; itmnr[char3]; dateb[date8]; conar[char2]; cuscl[char3]; custom[char5]; matcl[char3]; matnr[char6]; quabu[dec15,3]; bunit[char3]; amocc[dec15,2]; ccurr[char3]>
{<12345; 001; 20111101; C3; 003; 74100; 047; 000471; 1,000; PK; 32,00; USD >
<12345; 002; 20111101; C3; 003; 74100; 047; 000473; 1,000; PK; 8,00; USD>}

$\tau_{(\Sigma)}$: Aggregation on level month (according to sales date) + customer

"Billing (SD)" relation:
<month[date6]; slorg[char2]; cuscl[char3]; custom[char5]; amolc[dec15,2]; lcurr[char3]>
<201110; 01; 003; 74100; 35,00; CHF >}

$\tau_{(\Sigma)}$: Aggregation on level month (according to booking date) + customer

"Billing (CO)" relation:
<month[date6]; conar[char2]; cuscl[char3]; custom[char5]; amocc[dec15,2]; ccurr[char3]>
{<201111; C3; 003; 74100; 40,00; USD >}

Corporate data memory (CDM): Any data being extracted into the DW are stored in their original state; for easier re-use, some administrative information can be added (e.g., data origin, timestamp). Hence, data are available in the DW, even when they are already deleted, changed or archived, and thus not recoverable in the source system. Hereby, an utmost autarky and flexibility from source systems is achieved. Moreover, re-loading data causes organizational problems, such as necessary down-times; now, sets of data can be rebuilt without accessing the source system. It is advisable to extract not only actual necessary data from one source, but all data that are possibly needed and useful for analysis. 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 DW. By definition, use of these data is rare, so that storage mediums can be slower and cheaper ones (s. [SA09], [W+11]).

3.1.4 Data Availability

Information warranty: Many EDW have to guarantee a defined degree of data availability (often even 24 hours a day) for the users; hence, critical data are stored redundantly (next to the cases described as "data availability" above). In general, techniques for ensuring high availability of DB systems, such as backups and mirrored systems, are valid for DW, too[1].
Moreover, a special scenario to increase availability by avoiding inconsistent (i.e. implausible) data is the following. Two sets of data (A, B) are updated equally, yet subsequently with identical data. Due to the amount of new data, the update processes the data in packages, setting a commit after each package; however, the new data is plausible only as a whole. As soon as data set A is updated, set B is base of reporting. Thereby, an older, but plausible set of data is offered. After being successfully updated, set A is reporting base again, while set B is updated.
A special case poses *individual safety*; that means, individual needs for a more secure feeling causes data storing – redundantly or slightly changed – mostly in the data mart area. Yet, as this is rather a psychological than a technical or design issue, we will not go into details.

Better performance for data access: Still one of the most common and important reasons for storing data is faster access for reporting, analyses, et cetera. Data are stored redundantly and usually in a more aggregated format. Higher data access performance is mostly enabled by reducing the volume of data, which can be achieved by creating additional materialized views or by transforming data onto a more aggregated summarization level. For instance, data on detail level day can be reduced by about factor 30 when aggregating it on level month. A main problem, next to the verification of consistence, is to create materialized views that fit to users' behavior and requirements. Disregarding fixed reports, it is very difficult to anticipate how data are used; analyses vary, extend, and change according to business requirements. See also for related topics in Section 5.

3.1.5 Laws and Provisions

Corporate governance: Data are stored according to compliance rules of corporate governance. Managerial decisions usually base on pieces of information that are generated from DW data, distributed via reports and analyses. As such decisions often have expansive outcome for the company, traceability of decisions is of enormous importance. Therefore, data, on which decisions have been taken, must be stored separately to discharge duties of care. Additionally, not only the analyses' results can be important, but also the numbers and indicators on which the result has been created. In this case, the whole set of data has to be stored, too.

Laws and provisions: Data persistence can also be caused by laws and provisions.

[1] Although, such measures are usually more complex, due to the size of DW.

In Germany, legal obligations to retain data in the finance area are defined in the Commercial Code (§§257, 239 HGB), the Fiscal Code (§§147,146 AO), and other provisions [Law1]. For instance, balance sheet, profit and loss account, and annual report are listed, with safekeeping periods of ten years. Moreover, organizational liabilities for financial institutes base on the German Banking Act (§25a KWG) and related provisions [Law2], and also affects corporate governance. Another aspect is product liability, which includes test reports, for instance. In the Product Liability Act, obligations to retain data is not explicitly defined, but result from stated safekeeping periods of ten years (§13 ProdHaftG; [Law3]). Further details, including compilations of safekeeping periods for different business areas, can be found for instance in [BW07], [HH11], and [ZE09]. In Austria and Switzerland, similar legal obligations with slightly different periods of safekeeping exist (cf., [Law4], [Law5]). We limit our investigations to these three countries, and are convinced that similar provisions are valid in other countries, too.

Persistence often means redundant data, because source and transformed target data sets are stored. As transformations are usually not unique, keeping only the target data means loss of information. The resulting effort does not only concern hardware aspects; for example, maintenance for keeping data consistently and so on, has to be taken into account to calculate total costs. The operation of productive DW necessarily means potential for conflicts which arises from requirements of data usage – such as reporting and analysis – and time and effort to create the necessary prerequisites. In the following, we briefly describe these requirements and their consequences. As mentioned above, EDW are data sources for several applications. Huge data volume results from manifold requirements that have to be satisfied; to mention are mainly three: detailed information needs finest granularity, historical information requires lots of previous data, and the broad range of data being collected (e.g., for data mining). This amount of data has to be prepared for its intended use, which includes data redundancy. The management of redundant data not only requires disk space, but also additional effort for keeping the aggregated data up-to-date and consistently regarding to the raw data (cf. [Le03]). As this effort takes time, the data‘s availability is limited. Moreover, the defined data set constrains the flexibility of data concerning new needs of analysis or usage in general. A complex staging process with several layers of persistent data is in opposite of a fast availability of data. This topic also includes the question of how to combine a „near-real-time“ concept into a DW system, where particularly the updating of master data has to be taken into account (cf. [La04]).

3.2 Grouping of Persistence Purposes

DW architectures usually base on RDBMS with star, snowflake, or galaxy schema as its main logical data model; see for instance [Le03], [KR02]. Basically, these models offer an adequate performance when accessing data for on-line analytical processing (OLAP). However, the amount of data enforces to build up materialized views and aggregated summarization levels for enabling reporting and analysis within passable response times – including consequences we describe in Section 3.1.
A decision for persisting data cannot just be made by “disk space and updating costs versus gain in performance”. One has to take the purpose of the data’s storing into

account, to define whether it is only helpful, rather essential, or even mandatory. In order to identify the necessity of persistence, we classify such reasons into two groups: mandatory and essentially persistence.

Mandatory persistence applies to data that has to be stored according to laws and regulations of corporate governance. It also holds for data that cannot be replaced because they are not available any longer or cannot be reproduced due to changes of transformation rules. Lastly, data that are required for other data's transformation must be stored if simultaneous availability is not ensured.

Essential persistence can be classified into certain sub-categories. Firstly, data those are available or reproducible in principle. However, the effort for reproducing - in time and/or resources - is too high (e.g., for archived or costly transformed data). Of course, the definition of "too high" is very subjective and needs clarified. Another group is data which is stored to simplify the maintenance or operation of the warehouse or defined applications - such as planning data and data marts for specially authorized users. A third group of data is persistent due to the warehouse's conceptual design: single version of truth and corporate memory. Fourthly, responsibility for guaranteeing information leads to data storage for safety reasons. Finally, there is data redundantly stored for performance purposes – often the biggest volume.

For a complete grouping of persistence purposes by necessities, we refer to Table 1 (note: column "DS" (decision step) values are commented in the next sub-section).

Table 1. Persistence purposes, grouped by necessities

Purpose	Necessity	Category	DS
Data availability	Mandatory	-	2
Changing transformation rules	Mandatory	-	3
Addicted transformation	Mandatory	-	1
Corporate governance	Mandatory	-	1
Laws and provisions	Mandatory	-	1
Source system decoupling	Essential	High effort	5
Extensive data recreation	Essential	High effort	6
Complex data transformation	Essential	High effort	7
Constant data-base	Essential	Simplification	5
Data lineage	Essential	Simplification	5
Complex, different data	Essential	Simplification	5
En-bloc data supply	Essential	Simplification	5
Complex authorization	Essential	Simplification	5
Single version of truth	Essential	Design	4
Corporate data memory	Essential	Design	4
Information warranty	Essential	Safety	5
Data access performance	Essential	Performance	7

3.3 Decision for Data Persistence

A decision whether to store data must take into consideration the reason for persistence and its necessity. A regulation drives into persistence, for instance; basically, this is the case for all mandatorily stored data. The decision is more difficult for essentially stored data, as the reason cannot be quantified clearly. This is valid for categories high effort, simplification, safety, and performance. Solely design reasons are distinct identifiers for data storage. Figure 9 shows a decision flow for persistence of any data. It is simplified, because rather fuzzy terms, such as "complex" and

"frequently", have to be specified dependent on the domain and application scenario.

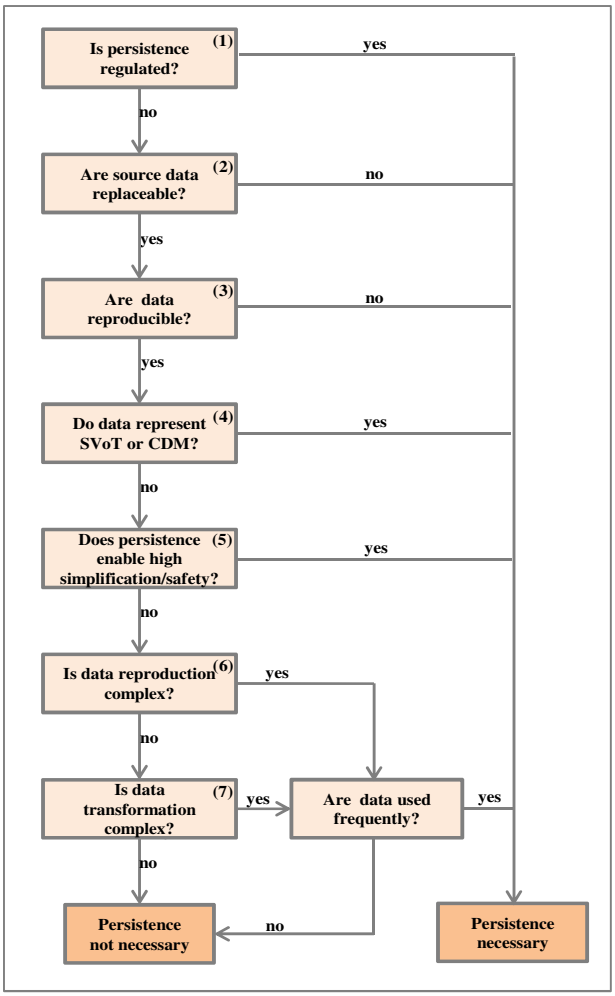

Figure 9: Persistence Decision Flow

The number of each step (1–7, in brackets) is used to assign purposes, shown in column DS (decision step) of Table 1. The first three steps deal with mandatorily stored data, which are not under consideration when deciding about persistence. For essentially stored data, indicators to come to a decision are much diversified. For instance, when the DW design includes a corporate data memory, these data have to be stored. In case the reproduction or transformation of data is complex (in time and/or resources), factors such as data volume, frequency of data access and data changes, and warranties for availability have to be taken into account. Moreover, the basis for the decision changes, due to the fact that data persistence has more than one reason. For example, as the connection to one source system is excellent, data would not be stored solely due to source system decoupling. Yet, as the transformation is partly complex, data is stored anyhow.

4 Outlook: Persistence in In-Memory

The approach of column-oriented DB systems is not new (cf. [CK85], [T+79]), and came into special focus in the DW community (e.g., [S+05a], [S+08]) because of their advantages regarding data compression and read access ([A+06], [Ab08]). Since some years, there are column-oriented in-memory-techniques used in commercial Data Warehouse products (e.g., "SAP NetWeaver® Business Warehouse Accelerator" [L+06], [Ro09] and "ParAccel Analytical Database™" [PA10]), for speeding up system's response time. Such techniques lead to the ability to both, load and query on data in terabyte volumes with good performance. Latest announcements even propagate installations embodying both, on-line transactional processing (OLTP) and OLAP - with up to 50 TB data in main memory [Pl09]; SanssouciDB [PZ11] is to mention here.
Those changes within technology lead to the question, in which degree persistence in IMDB-based EDW is required or necessary. It is suggested that no data additional to the source data have to be stored, but any data (e.g., for analysis) are computed on-the-fly ([Pl09], [PZ11]). Yet, this is only valid for some of the above-mentioned reasons. In this context we discuss databases that fully support ACID, including durability – such as IBM's solidDB [IC10], TimesTen by Oracle [OC09], SAP's HANA [SA11], or MonetDB [MD11]. Hence, persistence is clearly to be distinguished from data storage in volatile memory, where data are lost when the DB shuts down. As mentioned, such considerations are less important in RDBMS-based DW, as lower performance of the DB motivates storage.

4.1 Evaluation of Persistence in IMDB

All data which are not stored mandatorily come into focus for the discussion of persisting data in IMDB-based DW. Especially, this concerns data that have been additionally stored for enhancing access performance or due to complex transformation. That does not mean that all additional persistence is obsolete due to processing speed in such systems – as to en-bloc data supply or the creation of a constant data-base for planning, it will be even more valid. IMDB snapshot mechanisms, as defined in [KN10], do not keep data constantly for a required period of about hours or days. The essential here is not fast supply with new data, but the creation of an unchanged data-base over a certain period of time. Timestamps in in-memory concepts (cf. [PZ11] and [KN10]), can be an approach that requires deeper examination. However, it is not applicable for replacing a CDM if data come from different source systems – as usual for EDW. Moreover, reasons for persistence such as complex authorizations and data lineage are still valid.
Experiences show that technical limits come almost always earlier than expected, so system resources will not be sufficient for the given tasks. Performance that was satisfactory once will not be enough soon. The potential to access huge data volumes will raise new needs; new requirements will arise, the amount of data will grow. Under this assumption, it is to validate in IMDB-based DW, whether it is more reasonable to store data in an adequate format rather than computing it repeatedly on-the-fly. This is especially true for data that are often accessed and change rarely, such as closed periods of end-of-year-/quarter/month in financial accounting.

Another question in this context is the format, in which data are stored; which level of transformation is optimal for storage, can persistence layers be avoided? Hereby, the format has to be determined, that offers flexible usage and avoids repeated, identical transformation. This can be found out by cost-based scenarios for measuring runtime, such as:
Given a set of raw data (R), that must be processed via multiple transformations (τ_n; n={1,2,3}) to make it ready for analysis (A). We have to compare for finding out whether it is more effective to store data persistently (p_x) after each transformation, to keep it in volatile memory (v_x), or to compute it on-the-fly:

(1) R ➔ τ_1 + p_1 ➔ τ_2 + p_2 ➔ τ_3 + A
(2) R ➔ τ_1 + p_1 ➔ τ_2 + v_2 ➔ τ_3 + A
(3) R ➔ τ_1 + p_1 ➔ τ_2 ➔ τ_3 + A
(4) R ➔ τ_1 + v_1 ➔ τ_2 ➔ τ_3 + A
(5) R ➔ τ_1 ➔ τ_2 ➔ τ_3 + A

Further indicators that have to be examined for deciding whether transformed data have to be materialized are:
Data Volume: Is the amount of data this big, that its preparation for usage (reporting, analyses, and others) cannot be done on-the-fly any longer?
Frequency of data usage: Is data so often accessed (for reporting, analyses, and others), that the benefit of additional materialization outweighs its cost.
Frequency of data changes: Is data changed so often (by updating, inserting, or deleting), that the maintenance effort for keeping downstream views consistently is less than their returns. Furthermore, how complex are these changes?
Evaluations in this area are valid in RDBMS-based DW, too. However, we expect that performance of an IMDB leads to on-the-fly transformation rather than redundant persistence.
Some IMDB, as for instance [OC09] and [IC10]), enable to specify different requirements for durability for certain DB area, so that non-mandatorily persistent data are available in volatile memory only. We interpret this as a two level concept of data storage. Of course, such data are available only unless the DB is shut down. As this happens quite rarely, the data maintenance costs are low. Exemplary, this scenario can be preferable for determination of customers' RFM attributes (recency, frequency, and monetary). These attributes are a common way of customer categorization in CRM business; see for example [St09] for more details. The determination (shown in Figure 10) is based on customer master data (from CRM systems), point-of-sale receipts (from POS systems as boutiques), and order and invoice data (from ERP systems). In short: RFM attributes have to be determined and checked for new customers and whenever new receipts, orders, or invoices flowing into the system. The RFM determination is done by an application, including selections, computations, and currency conversions, with lookups to control data, et cetera. Computed attributes are not only used in the DW, but also transferred back to the CRM system. In practice, a fast changing, but reproducible data-base of several millions of customers with tens of transactions each, one can realize that storing the results in volatile memory is preferable.

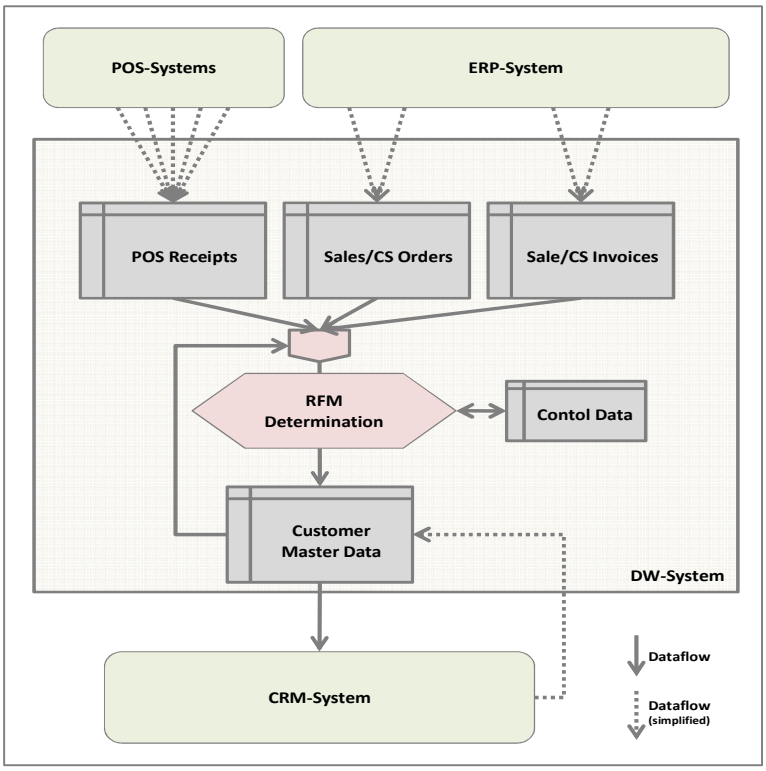

Figure 10: Determination of RFM Attributes

We expect that data, that is stored redundantly due to performance aspects in RDBMS, will vanish. Higher access speed in in-memory allows transformation of data on-the-fly, especially data with simple transformation logic (aggregations, joins, etc.). In consequence, materialized views will become virtual views.

4.2 Bases of Decision-Making – an Evaluation Approach

As mentioned above, decision-making on data persistence is a complex process. Moreover, it is often ambiguous and the basis for decisions changes frequently. In order to cope with such problems, we develop an approach that we briefly describe in the following. Our approach contains data stored of any reason for persistence, including mandatorily (cf. Sections 3.2f).
Our rating system, as basis for decision-making, consists of indicators and measurements that are results of combinations of quantifiers (i.e., criteria, factors, and key-figures):

- A criterion c is a natural number, validated with 0 or 1: $c \in \{0,1\}$.
- A factor f is a real number between 0 and 1: $f \in [0, 1] := \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$.
- A key-figure k is a real number: $k \in \mathbb{R}$.

Such validation indicators are classified in definable and designated ones. The first category is sub-divided into calculable, measurable, appreciable, and assessable, the

latter into external and internal figures. The subcategories again are defined as distinct (calculable, measurable, and external) and fuzzy (appreciable, assessable, and internal). See Figure 11 for a complete overview.

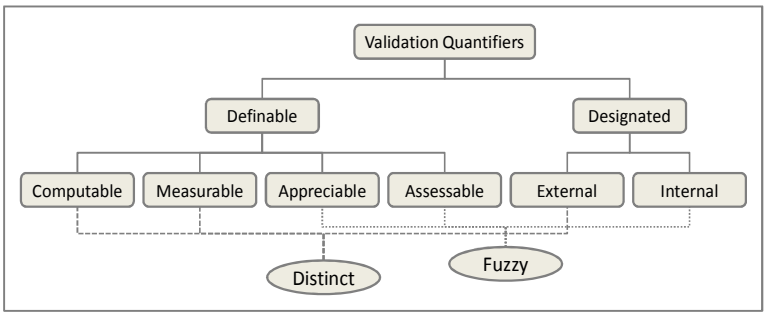

Figure 11: Classification of Validation Quantifiers

Definable in our context means, that the figures can be defined related to the particular case. For instance, computable figures are data volume in a data cube or frequency of data changes of a table, whereas the frequency of usage of data in a cube is rather a measurable one. Yet, calculations in this field have to deal with approximate values. Increase of data volume is an appreciable figure, examples for assessable ones are the operational availability of reports (e.g., "99.9%"), or the warranty of reporting performance (e.g., "< 5s for 95% of reports").
In contrast, designated figures are fixed to a specific value; for example, that means externally set by laws and provisions or internally by the requirement for a CDM or a SVoT of data.

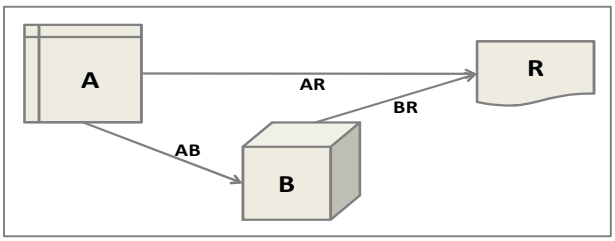

Figure 12: Performance/Effort Comparison Example

As mentioned before, the format, in which the data is stored best, is another problem in our discussion. Figure 12 shows such an example; the question is whether to access data of table A directly for report R, or to define another level of persistence (data cube B). Both options have their pros and cons. Data access from cube B will probably be faster than the access from A, but it requires more cost for additional update. The most important factors in the comparison of data access (i.e., arrows AR and BR) are time of data selection (T_σ), time of data transformation (T_τ), and frequency of data querying (F_Q). Disregarding additional factors, such as indexing, one can define the following formulas:σ

- $AR = \sum_{i=1}^{n} (T_{\sigma(AR)} + T_{\tau(AR)}) \rightarrow (T_{\sigma(AR)} + T_{\tau(AR)}) * F_Q \Leftrightarrow R = 1$
- $BR = \sum_{i=1}^{n} (T_{\sigma(BR)} + T_{\tau(BR)}) \rightarrow (T_{\sigma(BR)} + T_{\tau(BR)}) * F_Q \Leftrightarrow R = 1$

Data supply from A to B (AB) is characterized by the frequency of loading new data (F_Δ), time of updating (T_U), and time and frequency of cube's reorganization (T_R, F_R):

- $AB = \sum_{i=1}^{n} (T_{S(AB)} + T_{\tau(AB)} + T_U) + (T_R * F_R)$

The comparison of the results (i.e. indicators) has to be validated with respect to reporting performance and its guaranteed factor.
Such models are exemplarily for computable indicators. Next to the ones mentioned above, there are others to name in this technical context, such as data volume (with impact on disc space and processing time) and time for indexing and creating materialized views. Further indicators are requirements for data, such as timeliness, quality, consistency, and plausibility. Figures like effort for maintenance and for data validation are rather hard to quantify. This is also valid for questions that are brought up before: what does "high", "complex", and "frequently" mean (s. Figure 9)? Moreover, the flexibility of a data set has to be considered; that means, how manifold is its possible use regarding reporting and other applications.
Within this discussion, approaches of the multi-criteria decision analysis (MCDA) seem to offer good foundations to face such problems; see [Sc91] for an introductive compendium. Especially its variants, namely classical utility analysis (cf. [De08]) and Saaty's Analytic Hierarchy Process (AHP; [Sa80]), pledge fruitful adoption here (cf. [BK10]). Moreover, the use of Balanced Scorecard (BSC) methods (s. [KN96]) is a further step towards an evaluation framework.

5 Related Work

Due to the broadness of the discussed questions in this paper, related works comes from several research areas; we define four main categories:

- DW architecture and design,
- ETL, data transformation and information integration,
- maintenance and cost-based evaluation of materialized views, and
- in-memory databases.

In the field of *DW architecture and design*, [WA05] provides an overview and comparison between main architectural approaches. Many papers deal with reference architectures for DW systems. As mentioned in Section 2, such architectures define three main areas that represent the aspects of data handling: data acquisition in the staging area, data processing in the basis data-base, and data provision in the data cubes or marts. While naming differs, the meaning remains the same. [DM88] expects data in a declared format and differentiates its design into layers of raw data and enhanced data on a detailed and a summary level. [PR96] defines a generic DW architecture, where a separated DW is filled with transformed and integrated data; as a variation, the DW is split into a central EDW and data-mart-like Business DW. [MB00], [CG06], [Ze08], and [BG09] pursue the three-layer approach of staging area, basis data-base, and data marts with a slightly different naming, yet without going too deep into details of data consistence. With regard to DW, [In02] defines an architecture with three detail levels: atomic DW (current detailed data), departmental

or data mart (slightly summarized data), and individual (highly summarized data); besides, older detailed data can be accessed. In contrast, the approach of [KR02] defines DW as a collection of process-based data marts, filled directly from the staging area. Moreover, [SA09] introduces a "Layered, Scalable Architecture" (cf. Section 2), which explicitly deals with questions of what type of data processing and transformation is done on which layer. Although data persistence is directly influenced by the system's layout, it is not discussed deeply in those publications – mainly due to the reason that lower performance of RDBMS motivates storage. That means, due to the database's limited performance for data processing, persistence on each layer, or even after each level of transformation, is implicit.

Much work in the area of *ETL* is done for instance by [V+02], [Si03], [S+05b], who define conceptual and logical workflow-models for ETL processes in order to optimize them. Compared to this, our work does not focus on the "E" (extraction) part from external source systems. Hence, problems in this area are not discussed deeply here, such as data cleansing, and DW refreshment anomalies. The mentioned work, however, only takes batch job processing as a basis, and limits its examinations on the first target in the DW basis data base, considering it as the final one. Yet, much processing time is necessary during succeeding "TL" (transforming, loading) jobs via several layers of data transformation with regard to business requirements, for instance (cf. explanations for SVoT in Section 3.1.3). In our understanding, this is an important factor within the overall processing effort in today's EDW applications. [P+09] propagates ETL-less on-the-fly transformation for a system that covers both, transactional processing and reporting. Thereby, data integration is disregarded, which is however an explicit raison d'être of DW. *Data transformation* within OLAP databases is researched by [G+97], [LS97], and [LT09], who describe aggregation operators, summarizability and formal frameworks for aggregation. Such findings can act as an indicator for decision-making for our work. [LN07] covers the complex field of *information integration*. However, none of the mentioned works deeply outline complexity in terms of consuming time and resources within their studies, and effects and necessity for persistence are not considered.

Data persistence in DW is closely related to *materialized views* and their incremental maintenance, in conjunction with incremental loading of DW. Several authors work on these topics. [GL95], [P+02], and [GM06] work on incremental maintenance of materialized views within a single DB system. In their work, updates are usually defined as pairs of delete and insert, whereby a clear possibility for data history, another raison d'être of DW, is prevented. In reference to work about view maintenance in DW environments (e.g., [Z+95], [G+96], [Qu96], [A+97], and [L+01]), we have to agree with [JD08], as the conception of data warehousing, taken as a basis in these papers, varies from real-world ones in DW refreshment cycles, data sources' query capabilities, and significance of previous data in DW applications. [JD08], [JD09], and [BJ10] work on incremental view maintenance with focus on DW, where [JD08] points out the affinity of incremental loading of DW and incremental maintenance of materialized views, as both deal with "incremental updates of physically integrated data". As such techniques are still important in data warehousing, they are also to be considered in our work. However, similar to mentioned works on ETL, the authors do not pay attention to the reasons for persistence and they limit themselves to the first data storage in DW. A *cost-based*

model to identify optimal materialized views is proposed in [Ac04]. Such a model can be used as a foundation, but has to be combined with and enhanced by ambiguous figures.
In consideration of *in-memory databases*, only those are considered which fully support ACID criteria. In this area, mainly the MonetDB project [MD11] as well as works by [Pl09], [P+09], [KN10], and [PZ11] are to mention. Moreover, there are already some commercially offered IMDB: TimesTen by Oracle [OC09], IBM's solidDB [IC10], and SAP HANA [SA11]. As processing power of such systems in first place allows investigating reasons for persistence deeply, progress in this technology is directly influencing our research.
Today, databases that are used for commercial data warehousing, offer tools for cost-based modeled data access optimization that support decisions whether to create indexes and materialized views. As far as we know, the question of decision support indicators for data persistence in DW has not been discussed apart of those models. Nevertheless, this question will become more important for EDW based on IMDB.

6 Conclusion and Outlook

Enterprise Data Warehouses are complex systems with specific requirements on data, which can be met best by a dedicated, layered architecture. The need for data persistence in such systems has to be defined by the data's purpose. We classify reasons for persistence into mandatory, essential, and helpful. Based on this, we come up to decide whether to persist data or not in such installations. As this perception becomes more important regarding EDW on in-memory databases, we give an outlook on possible scenarios.
Persistent data also exist in EDW systems running on IMDB, not only mandatory but also essential data. However, a large amount of data is only stored in volatile memory or computed on-the-fly. Furthermore, the question arises, in which format data are stored. The answer cannot easily be found; in fact, it is a multidimensional decision of several facts, such as efforts of transformation, storing, and updating, number and time of calls, and number and time of updates.
Future work will include the definition of figures and indicators that support decisions whether to store data persistently in IMDB-based EDW. This contains both, distinct and fuzzy ones; for instance, comparisons of runtime and maintenance effort of data stored in certain format to find out if a decision to store data is computable or at least supportable. Our aim is to provide a set of definable, quantifiable, and weighted indicators to specify formulas that support practical decision making in the field of EDW data persistence in IMDB. Within this context, we expect that methods of MCDA, such as AHP, and the BSC approach seem suitable tools to adapt.

References

[Ab08] D.J. Abadi: “Query Execution in Column-Oriented Database Systems”; PhD Thesis, Massachusetts Institute of Technology; 2008.

[Ac04] T.L. Achs: “Optimierung der materialisierten Sichten in einem Datawarehouse auf der Grundlage der aus einem ERP-System übernommenen operative Daten”; PhD Thesis, Wirtschaftsuniversität Wien; 2004.

[A+97] D. Agrawal, A. El Abbadi, A. Singh et al.: “Efficient View Maintenance at Data Warehouses“; in: SIGMOD’97 Proceedings, pp.417-427; 1997.

[A+06] D.J. Abadi, S.R. Madden, M.C. Ferreira: “Integrating Compression and Execution in Column-Oriented Database Systems“; in: SIGMOD`06 Proceedings, pp.671-682; 2006.

[BG09] A. Bauer, H. Günzel (eds.): “Data-Warehouse-Systeme”; d-punkt, Heidelberg, 3rd edition; 2009.

[BJ10] A. Behrend, T. Jörg: “Optimized Incremental ETL Jobs for Maintaining Data Warehouses”; in: IDEAS’10 Proceedings, pp.216-224; 2010.

[BK10] B. Berendt, V. Köppen: “Improving Ranking by Respecting the Multidimensionality and Uncertainty of User Preferences”; in: G. Armano, M. de Gemmis, G. Semeraro et al. (eds.): “Intelligent Information Access”, pp. 39-56; Springer, 2010.

[BW07] Bundesministerium für Wirtschaft und Technologie (BMWI): “Handlungsleitfaden zur Aufbewahrung elektronischer und elektronisch signierter Dokumente“; on: www.bmwi.de/BMWi/Redaktion/PDF/Publikationen/Dokumentationen/doku-564, property=pdf,bereich=bmwi,sprache=de,rwb=true.pdf {01.10.2011}; 2007.

[CG06] P. Chamoni, P. Gluchowski (eds.): „Analytische Informationssysteme“, Springer, Berlin, 3rd edition; 2006.

[CK85] G.P. Copeland, S.N. Khoshafian: “A Decomposition Storage Model”; in: SIGMOD`85 Proceedings, pp.268-279; 1985.

[CW03] Y. Cui, J. Widom: “Lineage Tracing for General Data Warehouse Transformations”; in: The VLDB Journal 12(1), pp.41-58; 2003.

[De08] D. Delić: “Ein multiattributives Entscheidungsmodell zur Erfolgsbewertung nicht-kommerzieller Webportale”; PhD Thesis; Freie Universität Berlin; 2008.

[De09] B.A. Devlin: “Business Integrated Insight (BI²)”; on: www.9sight.com/bi2_white_paper.pdf {01.10.2011}; 2009.

[DM88] B.A. Devlin, P.T. Murphy: “An architecture for a business and information system”; in: IBM Systems Journal 27(1), pp.60-80; 1988.

[FS07] O. Fischer, M. Semrock: “Datenbank Internet“; on: http://wi.f4.htw-berlin.de/users/morcinek/DWH-Arbeiten/student/Datenbank_Internet.pdf {01.10.2011}; 2007.

[GL95] T. Griffin, L. Libkin: “Incremental Maintenance of Views with Duplicates”; in: SIGMOD’95 Proceedings, pp.328-339; 1995.

[GM06] H. Gupta, I.S. Mumick: Incremental Maintenance of Aggregate and Outerjoin Expressions”; in: Information Systems (31/6), pp.435-464; 2006.

[G+96] A. Gupta, H.V. Jagadish, I.S. Mumick: “Data Integration using Self-Maintainable Views”; in: EDBT’96 Proceedings, pp.140-144; 1996.

[G+97] J. Gray, S. Chaudhuri, A. Bosworth et al.: “Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals”; in: Data Mining and Knowledge Discovery 1(1), pp.29-53; 1997.

[HH11] Handelskammer Hamburg: “ABC der Aufbewahrungsfristen von Geschäftsunterlagen in Steuerrecht und Handelsrecht”; on: www.hk24.de/recht_und_fair_play/steuerrecht/abgabenordnung/365506/aufbewahrungsfristen.html {01.10.2011}; 2011.

[IC10] IBM Corporation: IBM solidDB™; on: www.ibm.com/software/data/soliddb {01.10.2011}; 2010.

[In02] W.H. Inmon: “Building the Data Warehouse”; Wiley Inc., New York, 3rd edition; 2002.
[JD08] T. Jörg, S. Deßloch: “Towards Generating ETL Processes for Incremental Loading”; in: IDEAS’08 Proceedings, pp.101-110; 2008.
[JD09] T. Jörg, S. Deßloch: “Near Real-Time Data Warehousing Using State-of-the-Art ETL Tools”; in: BIRTE’09 Proceedings, pp.100-117; 2009.
[KN96] R.S. Kaplan, D.P. Norton: “The Balanced Scorecard: Translating Strategy into Action”; Harvard Business Press, Cambridge; 1996.
[KN10] A. Kemper, T. Neumann: “HyPer: Hybrid OLTP&OLAP High PERformance Database System“; on: www3.in.tum.de/research/projects/HyPer/HyperTechReport.pdf {01.10.2011}; 2010.
[KR02] R. Kimball, M. Ross: “The Data Warehouse Toolkit”; Wiley Publishing Inc., Indianapolis, 2nd edition; 2002.
[La04] J. Langseth: “Real-Time Data Warehouses: Challenges and Solutions“; on: www.dssresources.com {01.10.2011}; 2004.
[Law1] Handelsgesetzbuch (HGB; 01.03.2011); Abgabenordnung (AO; 08.12.2010); “Grundsätze ordnungsmäßiger DV-gestützter Buchführungssysteme“ (AO_GoBS; 07.11.1995); “Grundsätze zum Datenzugriff und zur Prüfbarkeit digitaler Unterlagen (GDPdU; 16.07.2001).
[Law2] Kreditwesengesetz (KWG, 01.03.2011); “Mindestanforderungen an das Risikomanagement (MaRisk, 15.12.2010).
[Law3] Produkthaftungsgesetz (ProdHaftG, 19.07.2002).
[Law4] Bundesabgabenordnung (BAO, 07.05.2008)
[Law5] Obligationenrecht (OR, 01.01.2011); Mehrwertsteuergesetz (MWSTG, 01.06.2011); Geschäftsbücherverordnung (GeBüV, 18.06.2002)
[Le03] W. Lehner: “Datenbanktechnologie für Data-Warehouse-Systeme“; dpunkt, Heidelberg; 2003.
[LN07] U. Leser, F. Naumann: “Informationsintegration“; dpunkt, Heidelberg; 2007
[LS97] H.-J. Lenz, A. Shoshani: “Summarizability in OLAP and Statistical Data Bases”; in: SSDBM’97 Proceedings, pp. 132-143; 1997.
[LT09] H.-J. Lenz, B. Thalheim: “A Formal Framework of Aggregation for the OLAP-OLTP Model”; in: Journal of Universal Computer Science 15(1), pp. 273-303; 1997.
[L+01] K.Y. Lee, J.H. Son, M.H. Kim: “Efficient Incremental View Maintenance in Data Warehouses”; in: CIKM’01 Proceedings, pp.349-357; 2001.
[L+06] T. Legler, W. Lehner, A. Ross: “Data Mining with the SAP NetWeaver BI Accelerator“; in: VLDB`06 Proceedings, pp.1059-1068; 2006.
[MB00] H. Muksch, W. Behme (eds.): “Das Data Warehouse-Konzept“; Gabler, Wiesbaden, 4th edition; 2000.
[MD11] MonetDB B.V.: “MonetDB – Column Store Features”; on: http://www.monetdb.org/Home/Features {01.10.2011}; 2011.
[OC09] Oracle Corporation: “Extreme Performance Using Oracle TimesTen In-Memory Database”; on: www.oracle.com/technetwork/database/timesten/overview/wp-timesten-tech-132016.pdf {01.10.2011}; 2009.
[PA10] ParAccel: “PARACCEL ANALYTIC DATABASE™“; on: www.paraccel.com/wp-content/uploads/2010/07/PA_DS.pdf {01.10.2011}; 2011.
[Pl09] H. Plattner: “A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database”; in: SIGMOD’09 Proceedings, pp.1-2; 2009.
[PR96] V. Poe, L.L. Reeves: “Building a Data Warehouse for decision support”; Prentice Hall PTR, Upper Saddle River; 1996.
[PZ11] H. Plattner, A. Zeier: “In-Memory Data Management“; Springer, Berlin; 2011.
[P+02] T. Palpanas, R. Siddle, R. Cochrane et al.: “Incremental Maintenance for Non-Distributive Aggregate Functions”; in: VLDB’02 Proceedings, pp.802-813; 2002

[P+09] H. Plattner, A. Bog, J. Schaffner et al.: “ETL-less Zero Redundancy System and Method for Reporting OLTP Data” (US 2009/0240663 A1); US Patent Application Publication; 2009.

[Qu96] D. Quass: “Maintenance Expressions for Views with Aggregation”; in: VIEWS’96 Proceedings, pp.110-118; 1996.

[Ro09] J.A. Ross: “SAP NetWeaver® BI Accelerator”; Galileo Press Inc., Boston; 2009.

[Sa80] T. Saaty: “The Analytic Hierarchy Process for Decisions in a Complex World“; McGraw-Hill, New York; 1980.

[SA09] SAP AG: “PDEBW1 - Layered Scalable Architecture (LSA) for BW“; Training Material; 2009.

[SA11] SAP AG: “SAP® In-Memory Appliance (SAP HANA™); on: www.sap.com/platform/in-memory-computing/in-memory-appliance/index.epx {01.10.2011}; 2011.

[Sc91] C. Schneeweiß: “Planung 1: Systemanalytische und entscheidungstheoretische Grundlagen”; Springer, Berlin; 1991.

[Si03] A. Simitsis: “Modeling and managing ETL processes”; in: VLDB’03 PhD Workshop; 2003.

[S+05a] M. Stonebraker, D.J. Abadi, A. Batkin et al.: “C-Store: A Column-oriented DBMS”; in: VLDB`05 Proceedings, pp.553-564; 2005.

[S+05b] A. Simitsis, P. Vassiliadis, T. Sellis: “Optimizing ETL Processes in Data Warehouses”; in: ICDE’05 Proceedings, pp.564-575; 2005.

[S+08] D. Slezak, J. Wróblewski, V. Eastwood et al.: “Brighthouse: An Analytic Data Warehouse for Ad-hoc Queries“; in: PVLDB 1(2), pp.1337-1345; 2008.

[St09] J. Stafford: „RFM: A Precursor of Data Mining”; on: www.b-eye-network.com/view/10256 {01.10.2011}; 2009.

[T+79] M.J. Turner, R. Hammond, P. Cotton: “A DBMS for large statistical databases”; in: VLDB`79 Proceedings, pp.319-327; 1979.

[V+02] P. Vassiliadis, A. Simitsis, S. Skiadopoulos: “Conceptual Modeling for ETL Processes”; in: DOLAP’02 Proceedings, pp.14-20; 2002.

[WA05] H.J. Watson, T. Ariyachandra: “Data Warehouse Architectures: Factors in the Selection and the Success of the Architectures”; on: www.terry.uga.edu/~hwatson/DW_Architecture_Report.pdf {01.10.2011}; 2005

[Wi08] R. Winter: “Why Are Data Warehouses Growing So Fast?”; on: www.b-eye-network.com/print/7188 {01.10.2011}; 2008.

[W+01] E. Weippl, O. Mangisengi, W. Essmayr et al.: “An Authorization Model for Data Warehouses and OLAP“; in: „Workshop on Security in Distributed Data Warehousing“, New Orleans; 2001.

[W+11] T. Winsemann, V. Köppen, G. Saake: “Advantages of a Layered Architecture for Enterprise Data Warehouse Systems“; in: CSDM’11 Proceedings, accepted; 2011.

[Ze08] T. Zeh: “Referenzmodell für die Architektur von Data-Warehouse-Systemen (Referenzarchitektur)“; on: www.tzeh.de/doc/gse-ra.ppt {01.10.2011}; 2008.

[ZE09] Zentralverband Elektrotechnik- und Elektronikindustrie e.V. (ZVEI): “Archivierung von Dokumenten”; on: www.zvei.org/fachverbaende/electronic_components_and_systems/publikationen/druckansicht.html?tx_ZVEIpubFachverbaende_pi1%5Bpointer%5D=1&cHash=86e89cd8796b7229ef81e63c237dc321&type=1 {01.10.2011}; 2009.

[Zu11] T. Zurek: “NoSQL Options in Analytics and Data Warehousing”; on: www.analytic-processing.blogspot.com/2011/04/nosql-options-in-analytics-and-data.html, {01.10.2011}; 2011.

[Z+95] Y. Zhuge, H. Garcia-Molina, J. Hammer et al.: “View Maintenance in a Warehousing Environment”; in: SIGMOD’97 Proceedings, pp.417-427; 1995.