Valuation Factors for the Necessity of Data Persistence in Enterprise Data Warehouses on In-Memory Databases

Author: Thorsten Winsemann
Otto-von-Guericke Universität Magdeburg, Germany
Kanalstraße 18, D-22085 Hamburg
E: thorsten.winsemann@t-online.de

Supervisor: Prof. Dr. rer. nat. habil. Gunter Saake
Faculty of Computer Science
Departement for Technical & Operational Information Systems
Otto-von-Guericke Universität Magdeburg, Germany
Universitätsplatz 2, D-39106 Magdeburg
E: gunter.saake@ovgu.de

Abstract. ETL (extraction, transformation, and loading) and data staging processes in Enterprise Data Warehouses have always been critical due to their consumption of time and resources. Mostly, the staging processes are accompanied with persistent storage of transformed data to enable a reasonable performance when accessing for analysis and other purposes. The persistence of – often redundant – data requires high effort regarding maintenance, such as for updating and for guaranteeing consistency. Concurrently, flexibility of data usage and speed of data availability is delimited. Especially column-based in-memory databases (IMDB) enable to query high data volumes with good response times. Progress in in-memory technology leads to the question how much persistence is necessary in such warehouses. The answer cannot only be based on cost models, but also has to take other aspects into account. The presented thesis project deals with the problem of defining valuation factors for supporting the decision whether to store data in Enterprise Data Warehouses based on in-memory databases.

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

1 Research Topic

1.1 Prologue: Enterprise Data Warehouse Systems

In this work, the author focuses on *Enterprise Data Warehouses* (EDW); that is a Business Data Warehouse [1], thus supporting management's decisions and covering all business areas. Beyond, EDW are 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 data of the company. That means, there is a common view on centralized, accurate, harmonized, consistent, and integrated data to treat information as corporate asset. An EDW covers all domains of a company or even a corporate group, including the collection and distribution of data with heterogenous source systems and a huge amount of data. 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. Despite of this, there are further very complex requirements to the data: ad-hoc access, near real-time actuality, and often applications, such as CRM, that require very detailed and granular data with a long time horizon. Moreover, new or changing needs for information have to be satisfied flexibly and promptly, and, last but not least, the access to data has to be secured by a powerful authorization concept. Therefore, an efficient usage of an EDW requires a maximum regarding data consistency, data granularity, historical data, flexibility in usage, actuality of data, speed of data provision, availability of data, and system stability.

1.2 The Problem

Persistence often means redundant data, because source and transformed target data sets are stored. The resulting effort does not only concern hardware aspects; for instance, maintenance for keeping data consistently, et cetera, has to be taken into account to calculate the total costs.

The operation of productive EDW systems necessarily means potential for conflicts, which arises from requirements of the data usage – such as reporting and analysis – and time and effort to create the necessary requirements. The requirements and consequences in more detail are as follows.

As mentioned above, EDW represents a data source for several applications; the huge volume of data is a result of the manifold needs that have to be satisfied. The need for detailed information requires lots of data because of fine granularity. The need for historical information requires lots of data from the past, of course. Finally, the broad range of data being collected, for example for data mining scenarios, causes huge data volumes. This amount of data has to be prepared for its intended use; for instance, good reporting performance has to be ensured.

Data, extracted into the EDW, are usually transformed over several layers. Firstly, data are integrated by means of data cleaning, identification of duplicates, data fusion,

and information quality [2]. Subsequently, data are transformed according to technical and business needs, such as combining sales order with invoice data. Finally, data are transformed to enhance the performance of access. Each transformation involves less flexibility in usage and delays availability. 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. [3]).

Data Warehouse architectures usually base on relational databases (RDBMS) with star, snowflake, or galaxy schema as its main logical data model; see for instance [3,4]. 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 summarization levels for enabling reporting and analysis within passable response times.

Column-oriented database systems (cf. [5,6]) came into special focus in the Data Warehouse community (e.g., [7,8]) because of their advantages regarding data compression and read access [9,10]. Since some years, there are column-oriented in-memory techniques used in commercial Data Warehouse products (e.g., [11,12]), for speeding up system's response time. Such techniques lead to the ability to 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 memory [13,14].

Those changes within technology lead to the question, in which degree persistence in IMDB-based EDW is required or necessary. This discussion includes the question, in which format the data have to be stored (i.e., as raw data, syntactically integrated, aggregated data, etc.). In this context, we discuss databases that fully support ACID, including durability; e.g., [14,15,16]. Hence, persistence is clearly to be distinguished from data storage in volatile memory. A decision for persisting data cannot just made by comparing disk space and updating costs versus gain in performance. At first, the purpose of the data's storing (see Section 4) must be taken into account, as well. Such considerations are less important in RDBMS-based EDW, as simply the lower performance of the database motivates storage.

1.3 The Problem in Examples

Often, the problem is to decide whether to store transformed data or to transform raw data repeatedly. In the following, four brief examples point out this problem.

A query result, which is created by a JOIN or UNION of two tables, could easily be redone each time the query is executed. The costs for accessing and combining the data seem to be less than the costs for storing and updating the data set. However, if the query is called regularly and the data do not change (e.g., such as in closed posting periods), the storing of the data might be preferable.

Figure 1 shows a more complex example: the determination of so-called RFM attributes (recency, frequency, and monetary). These attributes are a common way of customer's categorization in CRM business; see [17] for more details. The determination is based on the customer master data (from CRM system), point-of-sale receipts (from POS system in e.g. boutiques), and order and invoice data (from ERP system). In short: RFM attributes have to be determined and checked for new

customers and whenever new receipts, orders, or invoices flow into the system. The RFM determination is done by an application program, including selections, computations, and currency conversions, with lookups to (control) data, et cetera. The computed attributes are not only used in the Data Warehouse, but also transferred back to the CRM system. Talking about 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.

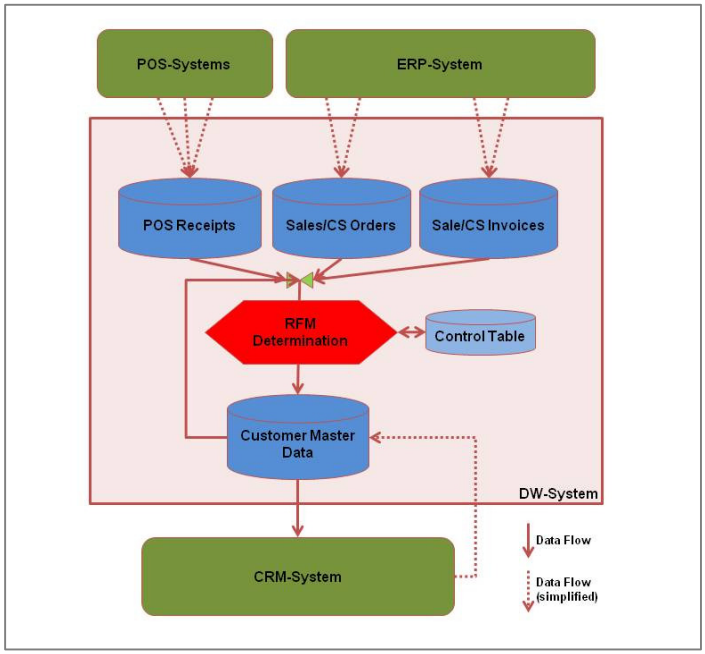

Figure 1: Data-Flow-Example "Determination of RFM Attributes"

Another example is data that cannot – or only with huge effort – be reproduced, as for example data from internet or data already being archived in the source system. Here, storing is the only way to prevent data loss.

Finally, there are also constraints for not persisting data; for instance changing data sets for which average values are computed.

2 Current Status of Research

This thesis project covers several working areas, so that we list products and researching people for each of these areas.

In the field of column-based databases, there are products as C-Store, Monet DB, Coldbase, and Brighthouse, as well as Sybase IQ as a commercial one. Researching people are M. Stonebraker, D.J. Abadi, S.R. Madden, P. Boncz, D. Bößwetter, and A. Lübcke. Relational in-memory databases are offered by IDM (solidDB) [15] and Altibase (Hybrid DB); among others, J. Guisado-Gámez researches in this area.

Today, commercially offered Data Warehouse Systems (DWS) are mainly based on relational database systems. In addition, there are so-called appliances (e.g., "SAP Netweaver Business Warehouse Accelerator", [18]), which receive data from the Data Warehouse and enable fast and flexible analyses on huge data volumes using column-based in-memory technology. This also counts for massively parallel processing (MPP) techniques (e.g., Teradata, HP Neoview, IBM). Researching in this area are for instance F. Färber et al., W. Lehner, H. Plattner et al.

People publishing in the field of (Enterprise) Data Warehouse architecture are for example B. Devlin, W. Inmon, R. Kimball, T. Ariyachandra, H.P. Watson, and T. Zeh. Next to the common Data Warehouse reference architecture, SAP's "Layered, Scalable Architecture" [19] is to note.

The topics ETL and transformation in DWS are examined by P. Vassiliadis, A. Simitsis, H.-J. Lenz, B. Thalheim, A. Shoshani, and P. Gray. Regarding information integration, U. Leser and F. Neumann are to mention.

Today, tools for databases used for commercial data warehousing, offer cost-based models for optimizing data access, supporting decisions whether to create indexes and materialized views. As far as the author knows, the question of decision support indicators for data persistence in Data Warehouses has not been discussed apart of those models. This question will become more important for EDW based on IMDB.

3 Preliminary Results

Defining the reason, for which data are stored, is an important criterion to define whether data persistence is still necessary in IMDB-based EDW. The author presents a collection of persistence reason and describes them in the following.

Source system decoupling: In order to reduce the load of the source system, data are stored immediately in the warehouse's inbound layer after successful extraction; usually, there is no or less transformation done (e.g., data of origin are added).

Data availability: In many cases, data are no longer available, or have been changed; in addition, network connections can be jammed so data are not available when needed. This applies to internet data, data from files that are overwritten regularly or data from disconnected legacy systems, for instance.

Complex data transformation: Due to complexity, transformation is very expensive regarding time and resources – data are stored to avoid repeated transformations.

Addicted transformation: Often, data are transformed using additional data from different sources. As a correct transformation is addicted to these data, they have to be stored until the transformation takes place to guarantee their availability.

Changing transformation rules: In some cases, the rules of transformation are changing. For instance, tax values change at some point of time; unless a timestamp is not part of the data, it will not be possible to transform the same way as before.

Extensive data recreation: Recreation of data that are no longer in the warehouse (e.g., archived data) is extensive; the data are stored transformed or aggregated.

Data access performance: Still one of the most important reasons for storing data is faster access for reporting, analyses et cetera. Data are stored redundantly and usually in a more aggregated format.

Constant data base: There are several applications – analysis and especially planning – where users have to trust in a constant data set; as it must not change for a selected period of time (hours or even days), it is stored separately.

„En-bloc data supply“: Usually, new data trickle into the Data Warehouse from several sources (world-wide); 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 to ensure a selected and defined data set for further processing.

„Single version of truth“: Data are stored after transformation according to the company's general rules – without any business logic or similar. That means, these data build the basis of any further use, meaningless whether it is a report for finance or logistics, for instance.

„Corporate data memory“: Data being extracted into the warehouse are stored in its original state. Hereby, an utmost autarky and flexibility from source systems can be achieved (e.g., when data are deleted or archived in source systems).

Complex, different data: Some data are often very complex and – semantically and syntactically – very different to the warehouse's data; hence, they are stored in order to transform it stepwise for its diverse usage.

Data lineage: Data in reports often result from multi-level processes of transformations; for later validations, backtracking of data origin is essential; source data are stored to enable or to ease this data lineage (cf. [20]).

Complex authorization: Instead of defining complex authorizations for data access, creating dedicated data marts for just the relevant data is often easier.

“Information warranty”: Many warehouses have to guarantee a defined degree of data availability and access speed; hence, data are stored additionally.

Corporate governance: Data are stored according to compliance rules (corporate governance); for example, to understand previous management decisions, the data set on which these decisions have been taken must be stored separately.

Laws and provisions: Finally, data persistence can also be necessary because of laws and provisions; for example, in Germany regulations exist in the areas of finance (German Commercial Code etc., [21]) and product liability [22].

Individual safety: The need for a more secure feeling is also a reason for storing data – redundantly or slightly changed – mostly in the data mart area.

The reasons are classified into three main groups to define persistence's necessity.

Mandatory persistence applies to data that have 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 (at least not in this format) or cannot be reproduced due to changed transformation rules. Lastly, data that are required for other data's transformation must be stored if simultaneous availability is not ensured.

Essentially persistent data can be classified into certain groups. 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 are 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 because of the warehouse design: single version of truth and corporate data memory. Fourthly, responsibility for

guaranteeing information leads to data storage for safety reasons. Finally, there are data redundantly stored for performance purposes – often the biggest volume.

Supporting data include mainly data stored for subjective safety needs.

Figure 2 shows a simple decision diagram for data persistence. The first three steps deal with mandatorily stored data, which are not under consideration when deciding about persistence, nor in in-memory based EDW. For all other data (both, essential and supporting), indicators to come to a decision are much diversified. For instance, if the warehouse design includes a single version of truth of data, it has to be stored. In case the reproduction or transformation of data is complex (in time and/or resources), frequency of access and warranties for availability have to be taken into account.

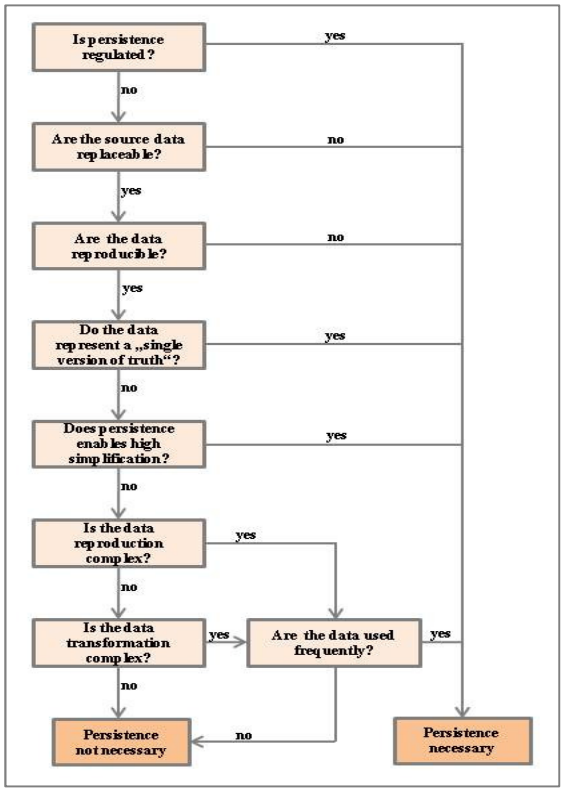

Figure 2: “Persistence Decision Diagram”

Further indicators that have to be examined for this decision are:

Data Volume: Is the amount of data so big that its preparation for usage (reporting, analyses, and others) cannot be done on-the-fly any longer?

Frequency of data usage: Are data accessed so often (for reporting, analyses, and others), that the benefit of additional materialization outweighs its cost.

Frequency of data changes: Are data changed so often (by updating, inserting, or deleting), that the maintenance effort for keeping the downstream views consistent is less than their returns in performance gains. And: How complex are these changes?

4 Goals, Benefits, Methodology

One aspect of this work is the focus on Enterprise Date Warehouses that base on IMDB systems, because of their dedicated requirements. The goal is to develop basics and models for supporting decisions for following questions:

- In which transformation format shall data be stored?
- Will data be transformed when it is requested ("on-the-fly"), will transformed data held in volatile memory or will data be stored persistently?
- On base of which technical/non-technical reasons must/shall data be stored?

Moreover, indicators shall be defined for supporting decisions whether to store transformed data or not, such as:

- Effort of transformation (logical complexity, effort of time/resources)
- Data volume, number of records
- Frequency of data usage/access (for analyses, data mining, etc.)
- Frequency of data modification (by updates, inserts, deletes)
- Requirements for data availability (query performance, "real-time reporting")

This contains both, measureable and non-measurable indicators – for instance, comparisons of runtime and maintenance effort of data stored in certain states to find out if a decision to store data is computable or at least supportable.

Possible scenarios for measuring runtime are as follows. A given raw data base (R) is queried for analysis (A), via multiple steps of transformation (Tn; n={1,2,3}). Comparisons are taken to find out whether it is more effective to store the data persistently (P) after each transformation, to keep it in volatile memory (V), or to calculate it "on-the-fly":

(1) R ➔ T1 + P ➔ T2 + P ➔ T3 + A
(2) R ➔ T1 + P ➔ T2 + V ➔ T3 + A
(3) R ➔ T1 + P ➔ T2 ➔ T3 + A
(4) R ➔ T1 + V ➔ T2 ➔ T3 + A
(5) R ➔ T1 ➔ T2 ➔ T3 + A

In here, factors such as frequency of data usage and changes as well as data volume have to be taken into account. Exemplary, the RFM determination (see Section 1.3) is a real-world scenario on which runtime comparisons can be made.

The expected outputs will support in areas of Data Warehouse development, design, build-up, and operation. For instance, one result could be the development of stored procedures for certain types of transformation. Besides, a guideline for deciding which data will be stored in which format could be used when operating a warehouse.

Methods used in this work will be:

- Literature research for topics as transformation, IMDB, and legislation
- Measurement of runtime and performance tests (e.g., as described above)
- Interviews/questionnaires for finding out practice in Data Warehouse development and operation (e.g., are there any other reasons for persistence, what is the percentaged distribution of the reasons)

References

1. B.A. Devlin, P.T. Murphy: “An architecture for a business and information system”; in: IBM Systems Journal 27(1), pp. 60-80; 1988
2. U. Leser, F. Naumann: „Informationsintegration“; dpunkt-Verlag, Heidelberg; 2007
3. W. Lehner: “Datenbanktechnologie für Data-Warehouse-Systeme“; dpunkt-Verlag, Heidelberg; 2003
4. R. Kimball, M. Ross: “The Data Warehouse Toolkit”; Wiley Publishing Inc., Indianapolis, 2nd edition; 2002
5. G.P. Copeland, S.N. Khoshafian: “A Decomposition Storage Model”; in: SIGMOD’85, pp.268-275; 1985
6. M.J. Turner et al.: “A DBMS for large statistical databases”; in: 5th VLDB’79, pp. 319-327; 1979
7. M. Stonebraker et al.: “C-Store: A Column-oriented DBMS”; in: 31st VLDB’05, pp.553-564; 2005
8. D. Slezak et al.: “Brighthouse: An Analytic Data Warehouse for Ad-hoc Queries“; in: PVLDB 1(2), pp.1337-1345; 2008
9. D.J. Abadi et al.: “Integrating Compression and Execution in Column-Oriented Database Systems“; in: SIGMOD’06, pp.671-682; 2006
10. D.J. Abadi: “Query Execution in Column-Oriented Database Systems”; Dissertation, MIT; 2008
11. T. Legler et al.: “Data Mining with the SAP NetWeaver BI Accelerator“; in: 32nd VLDB’06, pp.1059-1068; 2006
12. J.A. Ross: “SAP NetWeaver® BI Accelerator”; Galileo Press Inc., Boston; 2009
13. H. Plattner: “A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database”; in: SIGMOD’09, pp.1-2; 2009
14. H. Plattner, A. Zeier: „In-Memory Data Management“; Springer-Verlag, Berlin; 2011
15. IBM: IBM solidDB™; Fact Sheet, IBM Corporation; www.ibm.com/software/data/soliddb {18.05.2011}; 2010
16. Oracle: “Extreme Performance Using Oracle TimesTen In-Memory Database”; White Paper, Oracle Corporation; www.oracle.com/technetwork/database/timesten/overview/wp-timesten-tech-132016.pdf {18.05.2011}; 2009
17. J. Stafford: “RFM: A Precursor of Data Mining”; White Paper; www.b-eye-network.com/view/10256 {18.05.2011}; 2009
18. J.A. Ross: „SAP NetWeaver® BI Accelerator”; Galileo Press Inc., Boston; 2009
19. SAP: “PDEBW1 - Layered Scalable Architecture (LSA) for BW“; Training Material, SAP AG; 2009
20. Y. Cui, J. Widom: “Lineage Tracing for General Data Warehouse Transformations”; in: The VLDB Journal 12(1), pp.41-58; 2003
21. §§239,257 Handelsgesetzbuch (01.03.2011); §25a Kreditwesengesetz (01.03.2011); §147 Abgabenordnung (08.12.2010)
22. §13 Produkthaftungsgesetz (19.07.2002)