An Integrated Data Management for Enterprise Systems

Martin Boissier, Jens Krueger, Johannes Wust, Hasso Plattner
Hasso Plattner Institute for Software Engineering, University of Potsdam, Potsdam, Germany
{f_author, s_author}@hpi.uni-potsdam.de

Keywords: Shearing Layers, Integrated Data Management, Enterprise Architectures

Abstract: Over past decades, higher demands on performance for enterprise systems have led to an increased architectural complexity. Demands as real-time analytics or graph computation add further complexity to the technology stack by adding redundancy and distributing business data over multiple components. We argue that enterprises need to simplify data management and reduce complexity as well as data redundancy. We propose a structured approach using the shearing layer concept with a unified data management to improve adaptability as well as maintainability.

1 INTRODUCTION

Enterprise systems are complex compositions of multiple subsystems, which provide crucial business capabilities as financials, material management, business intelligence, customer relationship management, and much more. The architecture of enterprise systems has undergone many changes over the past decades ranging from first systems that ran on a single database to current setups with arrays of parallel systems and data stores. Such architectures consisting of multiple components, each storing its own (redundant) data set and with a very particular scope of duties, is what we call a *Separated Enterprise Architecture* (SEA). Many different demands led to such an architecture, e.g., the demand for analytical capabilities that introduced data warehouses into enterprise systems or more recent demands as graph computation capabilities or closed-loop concepts.

Meeting demands by adding separate components facilitates high adaptability in contrast to modifying or even re-engineering existing systems. This way, new requirements are met in time while avoiding modifications on business related and mission critical systems.

But in the long run, not only maintainability of redundant systems becomes increasingly expensive. Also adaptability is impaired as data exchange between components and data consistency gets increasingly complex in separated architectures. The diversity of data models and the inherent data redundancy between transactional and analytical systems therefore results in economical penalties through higher maintenance costs and reduced flexibility. In the end, it decelerates the progression of the system, because every additional component and layer increases data latency and data redundancy.

This paper discusses the problems of separated enterprise architectures and proposes a re-engineering of enterprise systems with focus on adaptability and long-term maintainability. Using the shearing layer concept, components are identified by their rate of change and integrated into the architecture accordingly. Such architectures lay the foundation for fast iterating adaptations, while keeping mission critical systems stable and achieving a consistent view on business relevant data. We call data management based on such an architecture *Integrated Data Management*, in which a single database system is employed for all components of the enterprise system.

The remainder of the paper is structured as follows. The progression of enterprise systems is discussed in Section 2, showing how different demands and requirements affected their architectures. Related work proposing different approaches to the increasing demands is presented in Section 3. Section 4 presents the idea of an integrated data management using the shearing layer concept. Presenting recent achievements in the field of database research, Section 5 discusses the feasibility of an integrated data management for enterprise systems. Section 6 discusses how the shearing layer concept can be applied to enterprise systems' using the presented technologies. In Section 7, this paper closes with a conclusion.

ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems, Volume 3, Lisbon, Portugal, 27-30 April, 2014
The final publication is available at SCITEPRESS via https://doi.org/10.5220/0004897204100418.

2 CURRENT ENTERPRISE ARCHITECTURES

The design of enterprise systems changed steadily over the last decades. First enterprise systems deployed a single database to run all business applications, cover business processes, and persist data. With the increasing demand for long running queries in analytical tasks, enterprise systems have been separated. To handle different workloads the applications were separated into transactional and analytical systems. Hereby the transactional system is run using write-optimized databases (usually traditional RDBMS), while data required for analytical queries has been transferred to additional systems, e.g., Enterprise Data Warehouses (EDW). The increasing demand for fast analytical reports and the ability to process long-running analytical queries have made EDWs an indispensable component in business environments and also have vastly effected their design. The first generation of enterprise data warehouses were implemented using traditional row-oriented RDBMSs with aggregated data to process analytical queries, simply to offload work from the transactional system. Later EDWs persist data using adapted data schemes and storage engines optimized for analytical workloads (e.g. star schemes and column-oriented databases).

The data transfer between transactional and analytical systems is done via ETL-processes (Extract, Transform and Load). The ETL-process usually processes only a predefined subset of the transactional data since many attributes of business entities are not relevant for analytical workloads. This subset of data is then batch-copied, filtered, and aggregated to optimize the data for analytical queries. Consequently the data timeliness in an EDW is dependent on the characteristics of the ETL-process, especially the intervals in which the process is run

Besides transactional and analytical systems, additional intermediate systems have been integrated. Increasing demands for new sources of information (e.g., caused by new components as online shops) and reporting on transactional data (e.g., best selling product of the last three days) introduced Operational Data Stores (ODS) (Inmon, 1999) as an intermediate system between the transactional system and the EDW. Operational Data Stores integrate different sources using data cleansing, ETL, and integrity checking. Furthermore, they provide the capability to run rather simple analytics on transactional data in right- or even real-time, usually on much smaller data volumes as in EDWs. With the competence of data integration and short-term analytics, operational data stores became a central component in business environments.

We refer to such architectures as *Separated Enterprise Architectures* due to the separation of data and business logic among different components (i.e. transactional systems, operational data stores, data warehouses, et cetera).

2.1 Advantages of Separated Enterprise Architectures

The need for the component based design of today's separated architectures arose due to insufficient performance for analytical workloads and timely data integration. De-normalized data structures, pre-computed aggregations, and narrowed tables are the foundation for analytical systems, which were able to process complex analytical queries with sufficient performance in contrast to transactional systems.

Another concern has been the integration of heterogeneous data sources. Besides the traditional transformation of transactional sources into analytics-optimized data, new data sources as point of sales data, online stores, or external vendor data needed to be integrated. With analytical components running at full load, a timely integration of data from several different sources has negative effects on query performance.

The most important advantage of a separated architecture is the comparatively easy addition of functionality without the need to modify mission critical systems.

2.2 Drawbacks of Separated Enterprise Architectures

Over the years the difference between transactional and analytical systems has steadily grown. With their analytics-optimized data characteristics EDWs are not capable of processing the majority of operational tasks. The reason is the irrelevance of certain transactional data for analytical tasks, which are required for operational tasks but neglected during ETL-processes. While this optimization improves analytical performance significantly, it hinders the data flow between components due to different data models in different granularities.

The demand for high-performance analytics on transactional data – a matter ODSs were initially designed for – is one of the main challenges of enterprise systems and a reason for the increasing complexity. One example is credit card fraud detection that relies on large data sets to accurately recognize fraudulent patterns. Such approaches require fast access to vast amounts of data while, in parallel, requiring access to the most recent transactional data to recognize fraud as early as possible. These requirements effect the usage of ODS since narrow time constraints are prohibiting any delay caused by intermediate systems. Talking about Business Activity Monitory (BAM, see Section 3) Golfarelli et al. wrote:

> [...]a tool capable of right-time [...] data streams. In practice, in most cases this requires to abandon the ODS approach typically pursued in DW systems and to adopt on-the-fly techniques, which raises serious problems in terms of data quality and integration.

(Golfarelli et al., 2004)

Closed-Loop Concepts A development that makes operational data stores dispensable is the prevalence of closed-loop concepts. With the horizontal as well as vertical connection of business components in modern systems, central messaging and data integration components present a potential bottleneck for data timeliness. Hence ODS are counterproductive for loose coupled and flexible architectures which are leveraging closed-loop approaches.

Furthermore, the evolution of separating business components with separated data and responsibilities contradicts the rather modern concept of close-loop functions, which cyclically integrate data between different data sources (Mangisengi and Huynh, 2008) (e.g., to enrich transactional systems with analytical insights from business intelligence systems).

Data Redundancy Another challenge is data redundancy and complexity added by evading the drawbacks of transactional systems. One reason is that traditional relational database management systems (RDMBSs) have significant shortcomings concerning the ability to process analytical tasks. Complex tasks as dunning or available-to-promise (ATP) checks require fine-grained data (i.e., transactional data) and perform long-running OLAP-style queries (Krueger et al., 2010b). To provide sufficient performance for such long-running queries in transactional systems additional data structures as materialized views or indices are used to compensate for the shortcomings of write-optimized databases used in transactional systems. But those data structures introduce new challenges to the system. Materialized aggregates lead to a higher synchronization effort as multiple write operations are performed for each modification of a business entity, similar to the drawbacks of indices.

Looking at the evolution of separated architectures a problem of adding functionality by adding components becomes apparent. Additional components provide advantages in respect to timeliness of data and performance only as long as the added systems – including the additional network latency – perform better than the originating source system. In the long run, the source systems will converge to the performance of the added components due to technological progress while automatically having advantages in respect to data timeliness. At this point the costs for any additional layer and its inherent latency being introduced further slows down the system. This has been the case for operational data stores as well as increasingly for data warehouses, where ETL-process run times are often no longer acceptable when analytical data is required in (near) real-time.

3 RELATED WORK

Several publications discussed the drawbacks of current enterprise architectures, each proposing different approaches to provide closed-loop functions and timely analytics.

Golfarelli et al. presented an approach to business intelligence (BI) called *Business Performance Management* (BPM) (Golfarelli et al., 2004). BPM is a framework including standardized components as data warehouses as well as reactive components. Such reactive components monitor time-critical business processes so that operational decision support can be steadily tuned and improved according to the company strategy. Thus, it can be seen as a combination of Business Activity Monitoring (BAM) (Dresner, 2003) with a full closed-loop approach. The architecture proposed by Golfarelli et al. is depicted in Figure 1. Especially the BAM component reveals the high complexity of real-time data flows in interleaved separated systems where vertical as well as horizontal data sources are integrated. Components as the user interface have to handle two additional data streams to enrich the data warehouse data with additional information.

Seufert et al. suggest an architecture with a central *Sense & Response (S&R) system* (Seufert and Schiefer, 2005). This S&R system communicates with the internal and external business components and improves business intelligence by linking business processes with BI/DSS (Decision Support Systems) and reducing latencies. The approach of Seufert et al. shows several problems concerning the complexity of SEAs. Besides the fact that the EAI (Enterprise Application Integration) component is not well integrated and lacks standardization for the analytical sources that are streaming data back to operational systems, the approach uses an operational data store for "local closed-loops". By using two differ-

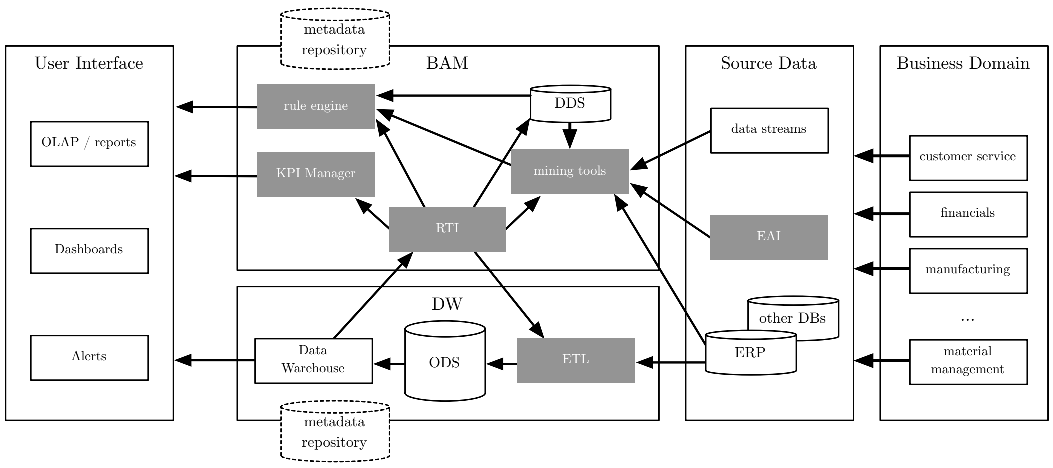


Figure 1: Architecture for Business Performance Management. Adapted with permission from (Golfarelli et al., 2004).

ent closed-loop approaches, the proposal of Seufert et al. introduces additional complexity and redundancy to the architecture. Furthermore, they propose the use of an operational data store that is considered inept in modern real-time systems (see Section 2.2).

Such complex systems have inherently sophisticated communication designs. Another issue is the absence of common communication channels, while a highly interleaved and sophisticated linkage between components is common. This results in an architecture in which communication channels potentially differ between each component.

To sum up, it can be said that both discussed approaches use additional components to evade the drawbacks of currently deployed technologies; adding redundancy and complexity to the architectures.

4 AN INTEGRATED ENTERPRISE DATA MANAGEMENT

The technological conditions for database designs changed starkly, both software and hardware wise. Looking at the technical improvements in the field of databases – e.g. in-memory computing or highly scalable NoSQL databases – the question arises why business environments are to a large extend untouched by the latest achievements in hardware and software. The separation of business concerns into transactional and analytical tasks, using write- and read-optimized databases respectively, is decades old and has never changed significantly.

Using an integrated data management approach only one database is deployed which we assume to have the following capabilities:

- A single database is deployed to store the entire data of the enterprise.
- The database is able to handle transactional as well as analytical workloads.
- Queries are processed directly on transactional data without any pre-aggregated views or cubes, allowing to answer analytical questions with sufficient performance.

Such a system does not persist any redundant data and provides unified data models that cover the complete enterprise system. Such unified data models for all business entities covering all internal components have the advantage of a notably simplified integration of peripheral components. This advantage is of particular interest as an efficient data integration is one of the major bottlenecks in current systems in which ETL-processes hinder analytical operations on current data.

Eliminating redundancy provides additional advantages. Implementing functionality on an integrated data management system means accessing a single database using a single global set of data models. The data used across components is inherently modeled consistently and can be directly joined. This finally enables the goal of having a “single source of truth” (Watson and Wixom, 2007). Furthermore, less data redundancy means less application complexity to keep different data sources consistent. Amongst others, this results in lower costs for maintaining, upgrading, updating and a faster return of investment when implementing new processes and functions (Plattner, 2009).

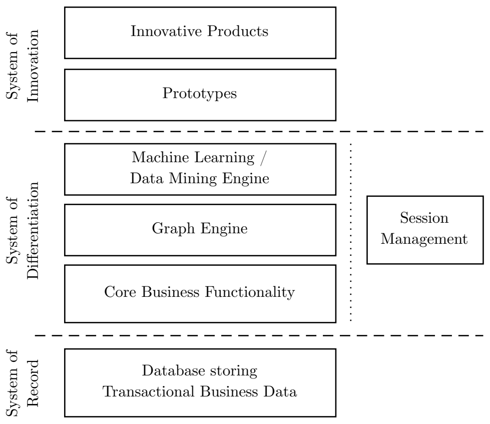
4.1 Adaptable Enterprise Architectures

However, combining all systems into an integrated data management architecture is not sufficient. Upcoming demands are still easier to meet by adding components – with copied data – instead of modifying the underlying system. Here, a structured approach is required that incorporates the need of frequent progression and adaption. With fast growing companies of the new economy and quickly changing markets enterprises are required to be adaptable and able to react to emerging trends or critical situations quickly. The past decade has shown that emerging startups can disrupt the businesses of multi-billion dollar companies, forcing those companies to quickly adjust to thwart customer loss. To handle these fast changing surroundings and markets an enterprise has be to adaptable in respect to its business processes and its technical environment.

For many companies the development focus is on sustaining mission critical systems ("never touch a running system") and implementing new functionality by adding new components. Truex et. al in contrast argue that the main focus should not be on organizational stability and low maintenance, but rather on "continuous analysis, negotiated requirements, and a large portfolio of continuous maintenance activities" (Truex et al., 1999). Here the system is expected to be constantly evolving and improving and never reaches a final state of functional completeness. This is a contradiction to the complexity and redundancy seen in separated architectures as enterprises show very different rates of changes. While the transactional system with its business related importance changes rather slowly, data warehouses evolved from batch-updated systems to increasingly integrated systems that are updated in real-time, sometimes even being part of the transactional system itself. Here an approach is required that takes these different levels of adaption into account to create an architecture that is prepared for upcoming changes.

A concept which describes the structure of evolving objects in respect to the different functions each component has is the shearing layer concept (Brand, 1994). The shearing layer concept originally described six different layers of a building, which evolve in different timescales. The fundamental layer, which is the site of the building, may not change for several generations. The next level – the building structure – may change at rates between 30 and 300 years. The last and most frequently changing layer is called stuff that are parts such as furniture, which move or are exchanged daily to monthly.

The concept has been adopted in many areas, e.g.,


System of Innovation
Innovative Products
Prototypes
System of Differentiation
Machine Learning / Data Mining Engine
Graph Engine
Core Business Functionality
Session Management
System of Record
Database storing Transactional Business Data
Figure 2: Exemplary architecture adapting pace-layered application strategy.

computer science (Simmonds and Ing, 2000) or human computer interaction (Papantoniou et al., 2003). Simmonds and Ing discuss the shearing layer concept in respect to information systems. They propose *agent* and *conversation* constructs as means to discuss the functional design of systems that employ the shearing layer concept.

A concept similar to the shearing layer concept was proposed by Gartner, called the Pace-Layered Application Strategy. This strategy differentiates three different layers (Gartner, 2012): 1.) *Systems of Record* - slowly changing layer that is highly standardized due to regulatory requirements and is usually mission critical, e.g. storing transactional data; 2.) *Systems of Differentiation* - for example business functionality that is customized to the enterprise but does not change steadily; and 3.) *Systems of Innovation* - new applications and recent technologies that are e.g. used to access new markets or requirements.

We use the idea of the shearing layer concept to formulate a structured approach to identify different rates of change of the system's components and how these can be orchestrated to allow a high adaptability while providing a consistent and reliable (slowly changing) foundation.

An exemplary architecture using Gartner's pace-layered notion is depicted in Figure 2. Here data and functionality are placed in the corresponding layers based on their rate of change and their relevance for the business. The base layer is the database storing transactional data that is highly relevant for all business related functions and can be seen as the slowly adapting foundation for the whole system. The base layer forms the core of the integrated data management.

On top of that, core business functionality that creates and processes transactional data is placed. This layer is adapting faster than the actual data store. Furthermore, fast changing or new components are placed in this layer such as engines for graph computation or data mining. Also additional data stores that do not store business relevant data (e.g. caching servers for websites or volatile data stores for session management).

Functionality that is still in the testing phase, demos, or prototypes are placed in the system of innovation layer.

4.2 Feasible Applications

Besides reduced complexity and data redundancy, an integrated data management also enables new applications. The following paragraphs discuss approaches that are not possible – or only with high efforts – in separated systems.

Forecasting The growing forecasting capabilities of enterprise systems are an important improvement of current and upcoming enterprise systems. Forecasting is dominated by long running analytic-style workloads. It is mostly processed on data warehouses or analytical systems, therefore calculations are limited to the data available at execution time. Outliers and abnormalities in the data can be an important factor for forecasts to be as precise as possible. This requirement is not fulfilled by traditional analytical systems, whose data is limited to crucial subsets needed for analytical reports and which is aggregated by predefined data integration methods. Thus forecasting is often limited to long time views, because critical short time forecasts are not possible due to the lack of transactional (near) real-time data.

Techniques as data mining to find new (or not yet known nor precisely modeled) facts can be used in combination with forecasts. The main benefit here is that all these services are not any longer bound to their own data sets, but can easily interact with each other since they are all running on the same data sets.

An example for such an application is the available to promise (ATP) check as presented in (Tinnefeld et al., 2011). Here, the ATP check is not only run when an order arrives or resources have to be reassigned, but also to check whether reassignments might cause problems at a later point in time (e.g., the production of too many items depends on the reliability of a single contractor).

Event-Driven Process Optimization Besides proactive tasks as forecasting, event-driven systems can also be enhanced by improved reactive components especially with the possibility of analytical queries on transactional data. As Seufert et al. wrote "Collecting and reconciling all operational data related to business processes, enables the measurement of process performance and helps to identify opportunities for process improvement" (Seufert and Schiefer, 2005). This advantage becomes subsequently possible because analytical insights, which are usually obtained from warehouses via complex OLAP-queries, are available directly in the source system. It can be propagated immediately to any component in the system.

5 MODERN DATABASE SYSTEMS

The field of database management systems has been widely disrupted in recent years. While traditional RDBMS vendors have dominated the database market for decades, in-memory databases for analytical applications as well as highly scalable NoSQL databases show the potential of recent database technologies.

Besides database software, several hardware achievements have greatly effected the design of current database technologies. On the one hand, increasing main memory capacities allow in-memory database to keep the whole database in memory. In many cases, in-memory databases show significantly improved performance over their disk-based counterparts. On the other hand, current server systems achieve high parallelism using multi-core CPUs or many-node setups, providing both means to scale up as well as to scale out.

These hardware improvements did not only influence the development of new database designs, but also allowed improvements of existing read-optimized databases. While databases for transactional workloads stagnate providing acceptable performance for analytical workloads, analytical databases are increasingly able to handle transactional workloads (MacNicol and French, 2004; Krueger et al., 2010a).

5.1 A "One size fits all" Database

The question whether a single database can be capable of handling both transactional and analytical workloads has been discussed in several publications, from a theoretical as well as from a practical point of view. Today, it is common understanding amongst most researchers that no single database can cover all scenarios and requirements for all applications (Stonebraker and Çetintemel, 2005; French, 1995). Conn states that no system could ever be able to handle both workloads with the same data model (Conn, 2005). Also (Howard, 2002) and (Barbusinski, 2002) conclude, that a common database model can not cover the constraints and characteristics of both - transactional and analytical - systems. Whereby Howard and Conn do not negate the possibility that a single database engine might be capable of such a mixed workload using different storage models.

Recent publications discuss the question, what type of database engine could close the gap between transactional and analytical workloads. This includes topics as adopting columnar characteristics in row stores (Bruno, 2009) as well as improving the write-capabilities of column stores (Krueger et al., 2010a). Also several hybrid approaches - combining row- and column-oriented storage schemata - have been proposed (Kemper and Neumann, 2011; Grund et al., 2010). Database technologies as NoSQL databases appear to be a good fit for enterprise systems as they are highly flexible and scalable, but constraints on transactionality hamper their adaption in productive transactional systems. Whether non-ACID databases can be efficiently used in business systems is not object of this paper, but recent technologies such as the RDBMS F1 show that even NoSQL pioneers as Google reintegrate SQL and ACID compliance in their mission critical systems (Shute et al., 2013).

Not proposing a "one size fits all" solution, but a solution particularly optimized towards enterprise systems, has been proposed by Plattner (Plattner, 2009). This so-called common database approach has been adopted in the commercial database *SAP HANA*, a columnar in-memory database optimized for mixed workloads (Sikka et al., 2012). Another in-memory database is the H-Store-based *VoltDB* (Kallman et al., 2008; Stonebraker and Weisberg, 2013). VoltDB is row-based and optimized for OLTP workloads using sharding and lock-free transactions. All major enterprise database vendors as IBM, Microsoft, and Oracle are currently developing in-memory solutions on top of their disk-based databases (Lindstroem et al., 2013; Larson et al., 2013; Lahiri et al., 2013).

5.2 Databases for Integrated Data Management Systems

Seeing the broad field of databases raises the question which database system might be able to handle an integrated data management and provide the foundation for a architecture as proposed in Section 4.1. As shown in Figure 2, such an architecture relies on a database that stores all business relevant data. This does not exclude a combination of multiple data storages (i.e., polyglot persistence (Sadalage and Fowler, 2012, pp. 133–140)) as there are situations in which the addition of specialized storage engines can be advantageous, e.g., for session management or website caching purposes. But it is important to understand that any business related data should be stored only once and as detailed as possible. If pre-aggregated views or additional technologies (e.g., Oracle's in-memory layer (Lahiri et al., 2013) for analytical workloads or Microsofts's Heckathon (Larson et al., 2013) for transactional workloads) are used to provide a sufficient analytical performance, the views should to be consistent with the primary transactional data source (i.e., no interval-based updates) and handled by the database only, not the application layer. The data storage layer might consist of an array of technologies and use redundancy to provide sufficient performance as long as the such performance additions are completely transparent to applications to reduce maintenance efforts and complexity. According to Plattner it is already possible to deploy a single database system for that matter (Plattner, 2009).

From a conceptual point of view technologies as in-memory databases might be able to close the gap between transactional and analytical systems. Developments that put additional layers on top of transactional data provide graph-computation capabilities without added data redundancy (Rudolf et al., 2013). The same has been shown for data mining and machine learning (Grosse et al., 2013). Consequently, we expect that relational databases using latest achievements in the area of in-memory computing will be able to handle transactional as well as analytical workloads with sufficient performance.

6 APPLICABILITY FOR EXISTING ENTERPRISE ENVIRONMENTS

Discussing integrated data management based on the shearing layer concept raises the question of how to map the different layers to the current software and hardware. Nowadays, most ERP systems use three-tier architectures consisting of database(s), application servers to provide scalability, and the front-end. Amongst others, this setup has several shortcomings in respect to performance and maintenance aspects to keep the application servers consistent.

In contrast, architectures based on the shearing layer concept can deploy a single DBMS to provide scalability without complex application servers. We propose to put both lower layers (i.e., the system of record and the system of differentiation) on the lowest hardware system, the database server. This way the performance wisdom of *bringing the code to the data* can be adhered. In contrast, most current architectures using the three-tier architecture bring data to the code to take computations off the database system. But as presented in the previous Section, nowaday's highly parallel DMBSs are capable of handling much larger workloads compared to the workloads and database performance of the time when the three-tier architecture was initially developed.

The proposed architecture shown in Section 4 can be achieved using different means. Most current database vendors favor stored procedures, which are written in SQL-like languages and are directly executed at the database level. The disadvantage of using SQL-like languages is that they are neither optimized for the expression of complex business logic nor aware of business objects used on the application side. Besides SQL, there is an array of languages to program databases. Haas et al. proposed a system called Starburst to enable object aware programming near the database: "We chose to stay with the relational model because of its great power and simplicity, [...] give users the support they need for their objects through extensions, while preserving the advantages of a relational DBMS" (Haas et al., 1989). A similar approach has recently been proposed by Frber et al. (Faerber et al., 2011). Here, the database is aware of application specific business objects – called *semantic models* – to run business logic directly on the database. Their architecture incorporates several aspects of the system of differentiation layer into the DBMS, e.g., a text analysis engine, a graph engine, and a calculation engine for computation intensive processes such as data mining. Furthermore, a specialized library provides common business functionality, which can be accessed by external applications and is optimized to execute directly inside the database core. Such functionalities include, e.g., currency conversion or disaggregation as used in many planning operations (Jaecksch et al., 2010).

An example of the possible performance improvements of re-engineered business processes has been shown by Tinnefeld et al. (Tinnefeld et al., 2011) with the available-to-promise (ATP) check. The ATP check has been implemented on a in-memory column store pushing down the business logic to the database. Even though this process is usually is considered transactional, it is comparably complex and includes long running queries (Krueger et al., 2010b). This prototype shows that transactional processes can be sped up significantly by pushing down logic to the database level.

However, adapting existing enterprise architectures is challenging. While it is partially possible to avoid rewriting code by using dynamic database views (Plattner, 2013), the costs to adapt complete systems are still rather high. Another challenge to solve is the question how logic push-downs to the database layer can be done using non-proprietary standards to avoid potentially expensive vendor lock-ins when adapting existing systems to vendor specific languages and protocols.

7 CONCLUSION

The trend towards more complex systems has been steady for years now, mainly caused by increasing requirements for data timeliness. With a growing number of systems whose data ought to be integrated in real-time, this trend appears likely to continue. The complexity of current enterprise architectures and the recurring pattern of introducing data-redundant components to improve performance reveal the structural problems of most enterprise architectures. In this paper we proposed a two-fold approach using the shearing layers concept to adapt the system architecture and an integrated data management to simplify further simplify the system. The shearing layers concept allows the identification of components with different scales of change to build an architecture that is adaptable and maintainable in the long run. Together with the implications of an integrated data management, our approach does not only enable new possibilities but also lays the foundation for upcoming generations of enterprise systems. Seeing this approach from a technical point of view in context with latest research leads us to the conclusion that an integrated data management for enterprise architectures running on a single database system is feasible. Because technologies such as in-memory technologies have already shown to have a vast impact on performance without the shortcomings of separated enterprise systems.

REFERENCES

Barbusinski, L. (May 2002). Can you use one database model for olap and oltp? *DM Review*.

Brand, S. (1994). *How buildings learn*. Viking, New York.

Bruno, N. (2009). Teaching an old elephant new tricks. In *CIDR*. www.crdrdb.org.

Conn, S. (2005). Oltp and olap data integration: a review of feasible implementation methods and architectures for real time data analysis.

Dresner, H. (2003). Business activity monitoring: Bam architecture. *Gartner Symposium ITXPO (Cannes, France).*

Faerber, F., Cha, S. K., Primsch, J., Bornhoevd, C., Sigg, S., and Lehner, W. (2011). Sap hana database: data management for modern business applications. *SIGMOD Record*, 40(4):45–51.

French, C. D. (1995). "one size fits all" database architectures do not work for dss. *SIGMOD Rec.*, 24:449–450.

Gartner (2012). Pace-layered application strategy. `http://www.gartner.com/it-glossary/pace-layered-application-strategy/`. Accessed 2013-09-27.

Golfarelli, M., Rizzi, S., and Cella, I. (2004). Beyond data warehousing: what's next in business intelligence? In *DOLAP '04: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*.

Grosse, P., Lehner, W., and May, N. (2013). Advanced analytics with the sap hana database. In *DATA 2013*.

Grund, M., Krueger, J., Plattner, H., Zeier, A., Cudre-Mauroux, P., and Madden, S. (2010). Hyrise: a main memory hybrid storage engine. *Proc. VLDB Endow.*, 4:105–116.

Haas, L. M., Freytag, J. C., Lohman, G. M., and Pirahesh, H. (1989). Extensible query processing in starburst. In *SIGMOD Conference*, pages 377–388. ACM Press.

Howard, S. (May 2002). Can you use one database model for olap and oltp? *DM Review*.

Inmon, W. H. (1999). *Building the Operational Data Store*. John Wiley & Sons, Inc., New York, NY, USA.

Jaecksch, B., Lehner, W., and Faerber, F. (2010). A plan for olap. In *EDBT*, volume 426 of *ACM International Conference Proceeding Series*, pages 681–686. ACM.

Kallman, R., Kimura, H., Natkins, J., Pavlo, A., Rasin, A., Zdonik, S. B., Jones, E. P. C., Madden, S., Stonebraker, M., Zhang, Y., Hugg, J., and Abadi, D. J. (2008). H-store: a high-performance, distributed main memory transaction processing system. *PVLDB*, 1(2):1496–1499.

Kemper, A. and Neumann, T. (2011). Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In *ICDE*, pages 195–206. IEEE Computer Society.

Krueger, J., Grund, M., Tinnefeld, C., Plattner, H., Zeier, A., and Faerber, F. (2010a). Optimizing write performance for read optimized databases. In *DASFAA (2)*, volume 5982, pages 291–305. Springer.

Krueger, J., Tinnefeld, C., Grund, M., Zeier, A., and Plattner, H. (2010b). A case for online mixed workload processing. DBTest '10, pages 8:1–8:6, New York, NY, USA. ACM.

Lahiri, T., Neimat, M.-A., and Folkman, S. (2013). Oracle timesten: An in-memory database for enterprise applications. *IEEE Data Eng. Bull.*, 36(2):6–13.

Larson, P.-A., Zwilling, M., and Farlee, K. (2013). The hekaton memory-optimized oltp engine. *IEEE Data Eng. Bull.*, 36(2):34–40.

Lindstroem, J., Raatikka, V., Ruuth, J., Soini, P., and Vakkila, K. (2013). Ibm soliddb: In-memory database optimized for extreme speed and availability. *IEEE Data Eng. Bull.*, 36(2):14–20.

MacNicol, R. and French, B. (2004). Sybase iq multiplex - designed for analytics. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04.

Mangisengi, O. and Huynh, N. T. (2008). Towards a closed-loop business intelligence framework. In *ICEIS*, pages 210–217.

Papantoniou, B., Nathanael, D., and Marmaras, N. (2003). Moving target: designing for evolving practice. In *HCI International 2003*.

Plattner, H. (2009). A common database approach for oltp and olap using an in-memory column database. In *SIGMOD Conference*, pages 1–2. ACM.

Plattner, H. (2013). *A Course in In-Memory Data Management: The Inner Mechanics of In-Memory Databases*.

Rudolf, M., Paradies, M., Bornhövd, C., and Lehner, W. (2013). The Graph Story of the SAP HANA Database. In *BTW*, pages 403–420.

Sadalage, P. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison Wesley Professional.

Seufert, A. and Schiefer, J. (2005). Enhanced business intelligence - supporting business processes with real-time business analytics. In *DEXA Workshops*, pages 919–925. IEEE Computer Society.

Shute, J., Vingralek, R., Samwel, B., Handy, B., Whipkey, C., Rollins, E., Oancea, M., Littlefield, K., Menestrina, D., Ellner, S., Cieslewicz, J., Rae, I., Stancescu, T., and Apte, H. (2013). F1: A distributed sql database that scales. *PVLDB*, 6(11):1068–1079.

Sikka, V., Färber, F., Lehner, W., Cha, S. K., Peh, T., and Bornhövd, C. (2012). Efficient transaction processing in sap hana database: the end of a column store myth. In *SIGMOD Conference*, pages 731–742. ACM.

Simmonds, I. and Ing, D. (2000). A shearing layers approach to information systems development. Technical Report RC 21694, IBM Research.

Stonebraker, M. and Çetintemel, U. (2005). "one size fits all": An idea whose time has come and gone. In *Proceedings of the 21st International Conference on Data Engineering*, pages 2–11.

Stonebraker, M. and Weisberg, A. (2013). The voltdb main memory dbms. *IEEE Data Eng. Bull.*, 36(2):21–27.

Tinnefeld, C., Mueller, S., Zeier, A., and Plattner, H. (2011). Available-to-promise on an in-memory column store. In *Datenbanksysteme in Business, Technologie und Web (BTW 2011), Proceedings*.

Truex, D. P., Baskerville, R., and Klein, H. K. (1999). Growing systems in emergent organizations. *Commun. ACM*, 42(8):117–123.

Watson, H. J. and Wixom, B. H. (2007). The current state of business intelligence. *Computer*, 40(9):96–99.