A Hybrid OLAP & OLTP Architecture Using Non-Relational Data Components

Thomas Benker

Chair of Information Systems - Systems Engineering
University of Bamberg
An der Weberei 5
96047 Bamberg, Germany
thomas.benker@uni-bamberg.de

Abstract: Relational database systems are still the first choice for the realization of business application systems. They are used as operational databases to persist business process data or as the basis for data warehouse systems. In recent years, the NoSQL-movement proposed a number of new non-relational data stores because of changing requirements of Web 2.0 applications and related limitations of relational systems. We argue that these new systems can also be used in OLTP- and right-time OLAP-scenarios. We demonstrate how enterprise application systems can be realized based on concepts of the NoSQL-movement. We propose an architecture that enables the deployment of an appropriate data model to a certain business domain and the integration of OLTP- and OLAP-functionality. Components of the architecture are suited to enhance business data with analysis-relevant information and to perform analysis tasks. To demonstrate the application of the architecture and its analytic features, we introduce a short case study.

1 Introduction

Since the publication of the relational model of Codd [Cod70] in 1970 most database systems were built upon that theoretical foundation. These systems were the first choice for realizing the persistence layer of business application systems for at least 20 years. Within the last few years, a new generation of non-relational data stores, known as NoSQL (often interpreted as “Not only SQL”) emerged. The reasons for their development are changing requirements, e.g. big data, scalability and fewer schema restrictions in Web 2.0 applications. To fulfill these requirements, they offer different system characteristics compared to traditional relational database systems. NoSQL data stores use non-relational data models and can be categorized as document stores, column-oriented data stores, graph data stores or key/value-stores [Ind12]. They are mostly designed for high availability and partition tolerance [Ind12]. This means, according to Brewer's CAP-Theorem [Bre00], that they are restricted for strong consistency, a key feature of relational database and enterprise application systems. The CAP-Theorem states that just two of the consistency, availability and partition tolerance requirements can be fully satisfied [GL02]. However, we claim that non-relational data stores can also be used in enterprise application systems, despite their specific characteristics. Instead, we argue that alternative data models and varying system characteristics of these data stores offer further benefits. But currently, there are only

few contributions considering these topics (see section 2). For that, we investigate how these non-relational technologies can be used in combined OLTP- (Online Transaction Processing) and OLAP-scenarios (Online Analytical Processing) and contribute to reducing the mismatch between application domains and persistent data models.

Today, object-oriented programming languages are dominant for the realization of business application systems. But storing objects in a relational manner often means breaking up complex hierarchical domain structures into a normalized schema. This phenomenon is known as object-relational impedance mismatch. Currently, this is often treated by an additional mapping layer between application and persistence logic. But these mappers introduce additional complexity in development and programming logic [Gho10]. Further, the relational model is a general data model and not aligned to a certain application domain and its characteristics; e.g. social networks, route planning problems, and business entities, as invoices or orders, are quite distant from relational concepts. Storing them in a normalized schema means violating their natural structures. This manifests a further mismatch with additional complexity in data usage [Gho10]. The authors of [SF12] propose to choose an appropriate data model/store fitting the requirements of each particular domain model. This concept is called polyglot persistence. It implies a movement from integration to application databases. Integration databases are designed to fulfill requirements of a number of application systems. Application databases can be designed and modified just according to the requirements of a single application [SF12]. Those requirements can be technical, concerning availability, consistency, backup strategy, scalability; or they can be structural, concerning data structures (structured vs. semi-structured data, highly linked entities vs. coherent hierarchical encapsulated data). The NoSQL-movement offers a number of alternatives when choosing the appropriate data model/store for a certain business domain.

Enterprise application systems can be categorized as OLTP- and OLAP-systems. OLTP-systems are used to realize the functionality of operational business processes. OLAP-systems perform business analysis and deliver decision-relevant information. Data extracted from operational data structures offers a common and valuable input for those systems. Today, enterprises have to be flexible in order to detect and react to certain operational business events with low latency [ACN05]. In [Hac04] this latency is split up in *capture latency*, describing the time between the occurrence of an event and its preparedness for analysis, *analysis latency*, meaning the interval until the information is delivered, and *decision latency* until an action is taken. The authors of [Rus11] examined the relevance of a concept called operational data warehousing in praxis. Key features are the bi-directional integration of operational business processes with data warehouse capabilities, in order to react to certain business events within an appropriate latency. One result is that almost of two-thirds of the participating organizations practice some form of operational data warehousing, and most of the others are planning to start related projects.

For this, one contribution of that paper is the proposal of a software architecture that enables the integration of OLTP- and operational OLAP-capabilities for right-time decision making. The term “right-time” is used to state, that whenever a decision maker is doing some analyses, these analyses are based on up-to-date data. The second contribution is the identification of concepts for the application of non-relational technologies (NoSQL) in enterprise application systems in order to realize benefits of

polyglot persistence. This work manifests a first step in research. The architectural blueprint should be used as basis to discuss and motivate further research with the objective to refine the presented concepts and to enhance applicability in integrated OLTP- and OLAP-scenarios. For that first step the proposed architecture focuses on a single business process scenario.
With respect to the mentioned contributions, the paper is structured as follows: In section 2 we investigate the state-of-the-art concerning component architectures to integrate OLTP- and OLAP-scenarios and using non-relational data stores. In section 3, we present the blueprint of the hybrid architecture using three types of components to integrate OLTP- and OLAP-scenarios. Section 4 presents the operational, monitoring and analytic data components of the architecture in more detail. In section 5 we present a use case to demonstrate the application of the architectural concept and its analytic capabilities. Section 6 closes the paper with a discussion of the presented architectural blueprint and proposal for future research projects.

2 State-of-the-Art

This section reflects the literature related to the presented work. Few publications discuss the usage of non-relational technologies in order to enable polyglot persistence or to integrate OLAP- and OLTP-scenarios. Component or service-oriented software architectures using non-relational technologies or integrating OLAP and OLTP are not discussed in literature.
In [Gho10] the author presents an infrastructural concept for multiparadigm data storage in enterprise applications. NoSQL data stores are used as cache systems to bridge the gap between the domain model of the application system and relational back-end data storage. The relational database system is used as a centralized data store to integrate data from heterogeneous domain models and application systems. The author also proposes using the relational structures for the generation of reports by using SQL or other relational tasks. The synchronization between cache and back-end databases is done in an asynchronous way. So, the infrastructure is only suitable for systems with eventually consistent characteristics. This means that there may be a short time span in which non-relational cache and relational back-end systems are not consistent. This work focuses solely on the integration of heterogeneous NoSQL data stores in enterprise scenarios.
In [SF12] the authors discuss a wide range of aspects for using NoSQL data stores in enterprise application systems. Among others, they make proposals concerning polyglot persistence, designing non-relational data structures and analyzing them using Map/Reduce [DG08]. They propose pooling data in a common entity type that is used together by the application. Further, they suggest defining services to manage each data type. Each service should encapsulate a data store best fitting the technical and structural requirements of the managed data structures. The results are used as input to the presented work and are integrated in an architectural concept for hybrid OLTP and OLAP usage.
Additionally, some research groups work on a more technical level, e.g. published in [KN11], [KHB+11] or [Pla09]. Their goal is to integrate OLAP and OLTP capabilities based on a single database system with a non-relational data model. The development is

mainly driven by the technical evolution of the last few years. The characteristics of these database systems are in-memory technologies and/or column-oriented data models. The concept of polyglot persistence in enterprise application architectures is not covered.

3 The Hybrid Component Architecture

The term “software architecture” is widely understood as a set of system components and their relations in order to realize a certain functionality of a software system. Nowadays, a huge number of business application systems are realized as service-oriented architectures. For the presented work we use the term *component architecture*. A service is characterized as an interface offering certain functionality while hiding the realization. In this and the future work, the realization of such an interface is of huge relevance and the inner structure of a component has to be discussed; for that we use the term “component”. Furthermore, a component is an independent part of a software system offering a dedicated functionality by its interface. It can be composed with and reused by other components in order to deliver higher functionality [Som11]. With respect to that definition and to reduce the mismatch of data and domain model, as well as integrating OLAP and OLTP in order to reduce latency between recognition of and reaction on business events, the requirements of the architecture can be summarized as follows:

support of polyglot persistence by keeping the business proccess execution (workflow) independent of a concrete data store/model;

enable the integration of reusable data components in higher workflows while hiding their implementation details behind an interface;

enabling the integration of OLTP- and OLAP-functionality in order to deliver up-to-date data for right-time decision situations;

In [SF12] the authors motivate to encapsulate heterogeneous database systems behind service interfaces to foster the reuse of data and to enhance the independence of application systems from specific data models. Based on this idea, we propose a component architecture in order to realize polyglot persistence, as well as right-time business analysis, in the context of a single business process. This paper focuses explicitly on data components supporting business processes, tracking monitoring data at run-time and business analysis. In figure 1 a blueprint of the proposed architecture is provided.
During *business process execution* by a workflow system, several *operational data components* are orchestrated to fulfill a common business functionality. The purpose of an operational data component is to manage the persistent business data. To integrate these data in a higher process, an operational component offers an OLTP-interface mainly supporting CRUD-operations (create, read, update, delete) or some advanced business logic on its managed business entities. The offered operations are implemented on an appropriate non-relational data model, fitting the technical and structural requirements of the business entity and its domain. Therefore, the mismatch between domain and data model can be reduced. To be orchestrated by a workflow, operational

data components also have to deal with the heterogeneity of data models and be prepared to be integrated in a transaction context.

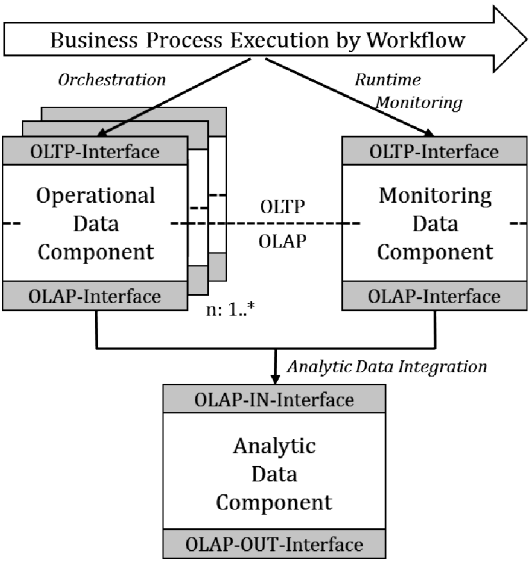

Figure 1: The blueprint of the hybrid OLAP & OLTP architecture.

In order to integrate OLTP- and OLAP-scenarios, each component has to support OLAP-functionality, too. To be flexible in data analysis, data warehouse systems use the multidimensional data model of Codd et al. [CCS93]. To provide analytic capabilities, quantitative measures are analyzed and aggregated following qualitative dimensions and their levels. The multidimensional model is also used for analysis in the proposed architecture. Each operational data component can provide measures and dimensional information for its managed business entities. To provide a complete analytic view of the business process, the analytic information of all operational components has to be integrated. This is done by the *analytic integration process*. The data integration is done in two steps: The first step is internal to the operational data component. It involves extracting the analysis-relevant data and enriching it with multidimensional information, e.g. dimensions and/or pre-calculated measures. Operational data components are able to deliver analytic data without negatively influencing the OLTP-functions. We argue this is due to typical system characteristics as scalability, parallel processing and new concurrency concepts (e.g. Multiversion Concurrency Control) of NoSQL-systems. In the second step, external to the operational components, the data is integrated by the application integration process, triggered by certain events. This process calls the OLAP-interface of the operational data components to get analysis-relevant information, integrates this information and hands it to an analytic data component using the OLAP-IN-interface. Within a single business process, data is assumed to be consolidated and homogeneous. This means it can be integrated with low complexity by using the analytic integration process. The analytic data component is responsible for the realization of

analytic functions. In order to offer these functions and to deliver the results, it is prepared with an OLAP-OUT-interface.
Monitoring data components are introduced to the architecture in order to enhance the analytic capabilities with run-time information. Monitoring data components are also divided in OLTP- and OLAP-functionality. Their purpose can be described as tracking run-time data from the business process executions and enriching it with analytic information.
To be flexible in delivering the right information at the right time when it is required for analysis, we propose four types of events. They are specified to trigger the analytic data integration process: (1) *Temporal events* are formulated as time periods or as certain points in time (e.g. every 10 hours certain process data has to be analyzed). (2) *Business events* show that a certain state of the business process has been reached. An incoming order or payment are examples of business events. (3) *Run-time events* represent more technical events in workflow instances executing the business process, e.g. a process instance is idle for a certain time or a run-time exception occurred. (4) *Consumer events* are related to the analytic data component. They represent ad-hoc consumer requests using the OLAP-OUT-interface.

4 Operational, Monitoring and Analytic Data Components

This section is used to discuss the data components in detail. Operational data components are orchestrated by workflows and manage the business process data while monitoring data are used to keep track of workflow executions. Analytic data components perform the analysis on the integrated business and logging data.

4.1 The Concept of Operational Data Components

The blueprint of an operational data component is shown in figure 2. There are four main parts to be examined in detail: the OLTP-part, realizing the business logic; the OLAP-part, realizing the analytic data enrichment; the interface to configure the component and the non-relational data store at its core.

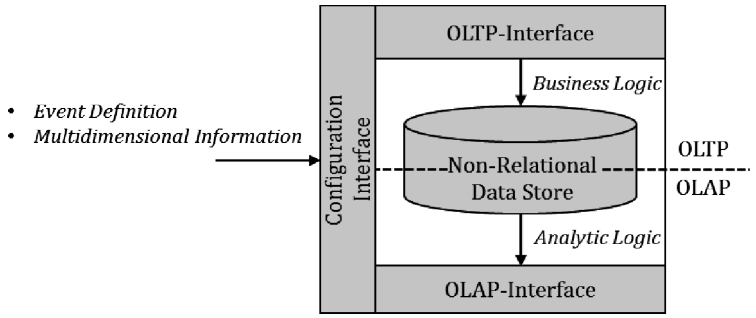

Figure 2: The blueprint of an operational data component.

According to [SF12], a common feature of most non-relational data stores is what they call the aggregate data model. This means that data which is handled (read, written)

together by the application logic is persisted in a single denormalized entity type. We extend this definition to the business context by defining business entities, as they occur in enterprise reality, like order, bill or customer as aggregate data. For each aggregate business entity we suggest using one managing component. It is responsible for offering a manageable set of operations on coherent data without the necessity of calling other components.

The *OLTP-interface* of an operational component defines operations which can be used by workflows in order to realize a certain composed business functionality. This functionality is often realized by the means of transactions. The overall result of a transaction is a successful ending or an abort with rollback to the initial state. All components participating in a transaction have to be consistent with that result. For that it must be possible to integrate an operational component in the transaction context of the orchestrating workflow. This is possible by providing an implementation of the Do-Undo-Redo-protocol as proposed in [GR93]. In case of a failing overall transaction, the local results of an operational data component can be undone. The component's internal consistence can be realized by the aggregate data design. Most non-relational data stores do not offer ACID-transactions (atomicity, consistency, isolation, durability) on more than one item but modify a single aggregate with transactional characteristics [SF12]. In order to manage the heterogeneity between data components, a common data format has to be used. JSON (Java Script Object Notations, [Cro06]) is a data format supported by most of the non-relational data store APIs. Workflow-systems have to be evaluated for their JSON support in order to use that format for persistence and data exchange. Doing so allows programming complexity within the components to be reduced.

The *OLAP-interface* has to offer operations in order to provide data for the analytic integration process. The analytic part of the component has to extract relevant data from the non-relational data store and enhance it with additional multidimensional information, e.g. the aggregation hierarchy of a dimension or pre-calculated measures. Each data component has to be configured in order to be able to prepare its business entities with analytic information. For this reason each data component is designed to have a *configuration interface* where this information can be specified. Two general possibilities are imaginable for this purpose. First, a data component can be parameterized with the location of a repository or a service where additional multidimensional information can be requested at run-time. Second, the multidimensional information is specified directly on each data component before business process executions. Which alternative fits best and the design of a concrete solution are tasks for future work. A further feature of the configuration interface is the ability to define the events (temporal, business, run-time) that trigger the analytic integration process. By using these events, the architecture can be configured to react to certain business events and subsequently reduce analysis and capture latency.

The central part of the component is a *non-relational data store*. To reduce the impedance mismatch and to benefit from polyglot persistence, an appropriate data model has to be chosen. Non-relational candidates are document stores, column-oriented data stores, graph data stores or key/value-stores. Which one to choose depends on technical and structural requirements of the domain model and on the features supported by a concrete data store. In [Hec11] and [Ind12] the authors characterize NoSQL-systems according to their data model and technical characteristics. These publications can be used to support the decision for a dedicated non-relational data store.

4.2 The Concept of Monitoring Data Components

The concept of monitoring data components is quite similar to operational data components. As mentioned earlier, their purpose is to log run-time data of the execution of workflow instances. Examples are entering a certain part of the workflow, calling a certain data component, starting, committing or aborting a transaction, ending a workflow successfully or in failure state. Monitoring components are part of the architecture in order to add further analytic capabilities and to react to run-time events in an adequate manner. The OLTP-interface is different compared with operational components and offers operations to deliver logging information of the execution environment. Monitoring data components also provide a configuration interface in order to define multidimensional information and to specify run-time events. The OLAP-interface is used to add analytically enriched monitoring data to the business data by the analytic integration process.

4.3 The Concept of Analytic Data Components

All analysis-relevant information (measures and dimensions) are integrated by the analytic integration process to a single data set, e.g. to a JSON-document, and given to the *OLAP-IN-interface* of the analytic data component. The blueprint of the analytic data component is visualized in figure 3. The purpose of that component is to perform the analysis based on non-relational data in a NoSQL data store. Using a relational database system would introduce further comlexity, e.g. for transforming non-relational business data to relational analytic data.

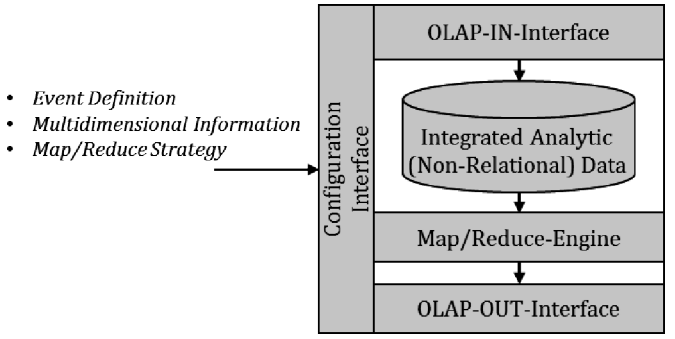

Figure 3: The blueprint of an analytic data component.

The integrated data is stored in the *non-relational data store* of the component and fits the requirements of the used data processing technology – Map/Reduce. Similar to relational OLAP, where data is structured as star or snowflake schema with respect to efficient SQL-queries, the non-relational data is prepared to be evaluated by the *Map/Reduce-engine*. Map/Reduce as proposed in [DG08] is a paradigm for parallel processing of huge data sets. It is often used in context of NoSQL data stores because it fits the characteristics of the non-relational data models. Additionally, the analysis latency can be reduced by easily increasing the number of nodes performing the

Map/Reduce-task. Map/Reduce offers two functions: first, incoming data sets are *mapped* to key-value pairs and second, the values of equivalent keys are aggregated and *reduced* to a single value [SF12]. The analytic data component has to offer analytic capabilities based on business process data. We propose how to use Map/Reduce for this purpose and show, thereby, that it is possible to perform that kind of analysis in the proposed architecture. As mentioned earlier, the analytic data consists of measures and dimensions, as known from multidimensional data structures. To describe the map-function, we extend the proposal in [SF12]. We suggest building up the keys with extensions of the levels of the associated dimensions and separating them by ".". So, a key represents a possible extension of the dimension levels, one from each dimension. The values in those key/value-pairs are the measures which remain untouched during the execution of a map-function. As a result, there is a key/value-pair for each possible dimension level combination. The reduce-function uses the output of the map-function as input and applies the aggregate function to the measures of the key/value-pairs which share the same key. A practical example is given in section 5.2. It is not necessary to apply the map- and reduce-functions to all analytic data sets each time the analytic data component is updated. There are different strategies to apply the map/reduce-pattern, for example temporal decoupling of map and reduce or an incremental application only to updated data [SF12]. These strategies can be defined using the *configuration interface* and contribute to reducing the analysis latency. The configuration interface can also be used to define events in order to trigger further analysis (e.g. on new incoming data or on certain analytic results) or to send a message to certain observers. Using the *OLAP-OUT-Interface*, observers can register to be informed in case of certain results. Furthermore, the OLAP-OUT-interface can be used to trigger ad-hoc analysis functions (defined as consumer event).

5 Case Study

We present a case study in order to demonstrate the application of the hybrid OLAP and OLTP component architecture. We show how it can be used in order to manage and to analyze business process data. For the sake of confirmability, the case study is kept simple and focuses on the introduced concepts of the architectural blueprint.

5.1 Introduction of the Business Process "Procurement of Flights"

Figure 4 shows in a schematic manner the interactions within a business process for the procurement of flights from airlines to customers by using a web portal. The case focuses on the interaction between customer and flight portal, which should be automated using the presented hybrid OLAP and OLTP architecture. To procure a flight, there are two main interaction activities between portal and customer.

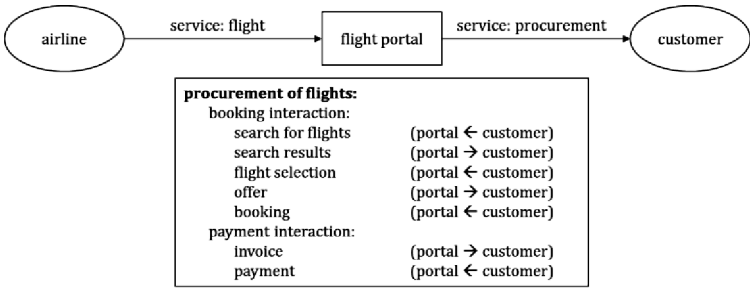

Figure 4: Interaction of the case study "procurement of flights".

First, to book a flight, the customer can search for flights, select one, and after getting a flight offered, book this flight. The second interaction describes the payment. The flight portal sends an invoice to the customer who reacts with the payment. The interaction between flight portal and airline is mainly used for synchronizing data. After a customer has chosen a certain flight, the flight portal requests up-to-date flight information. After the completion of the booking by the customer, the flight portal and the airline synchronize the booking and customer information.

5.2 The Hybrid Component Architecture for the Case Study

Figure 5 shows the proposed architecture to realize the business process "procurement of flights." It is necessary to manage information on flights, the airlines and the customers. We propose one operational component for each of these master data entity types. We propose two further components, one to manage the booking and one to manage the payment information. We argue that data that is treated together by the application workflow should be persisted and managed by a dedicated operational data component. The workflow can be realized using an executable language, e.g. BPMN (Business Process Model and Notation, [OMG11]) and orchestrates the data components by using a REST-based (Representational State Transfer, [Fie00]) OLTP-interface and JSON-documents. Based on this, a document store fits best for persistence. As a result, business entities are managed according to the structures of the domain model and close to the data usage of the orchestrating workflow. The monitoring data component is realized on a key/value-store. It fits best for logging monitoring information. Figure 5 also shows examples of events, defined on operational data components in order to trigger the analytic data integration process (realized as BPMN-workflow). If a booking or a payment is completed, the concerning data components start to create a JSON-document with the measures (volume of the booking, the days sales outstanding of the payment) and hand it to the BPMN-process. During its execution it requests further analytic data from the other data components, e.g. the dimension geography of the customer from the customer component in order to enable aggregate functions on city, state and country.

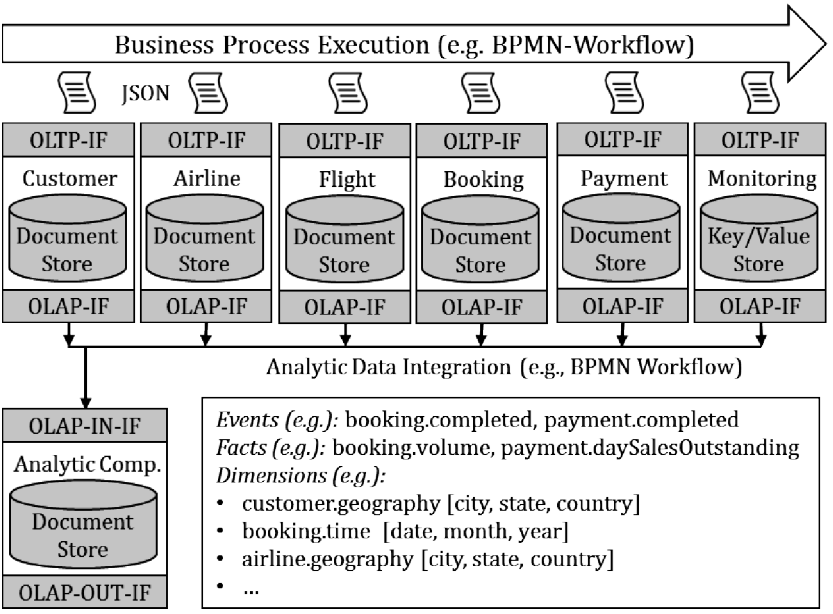

Figure 5: The proposed architecture to automate the business process on procurement of flights.

Figure 6 shows by example how the analytic data component can use the Map/Reduce-paradigm in order to perform business analysis. On the left side, the incoming integrated analytic data is shown. This document has the two named measures and dimensional information for geography and time. The map-function is used to build key/value-pairs. Keys are derived from the extensions of the dimension levels. Each level has to be combined with each level of all other dimensions. The values of the measures remain untouched. By using the reduce-function, the values of documents with the same key are aggregated. To get the average days-sales-outstanding and the total volume of all bookings, done in Bavaria on May 21st, 2013 all documents with key "2_bavaria.1_2013-05-21" are reduced to a single data set.

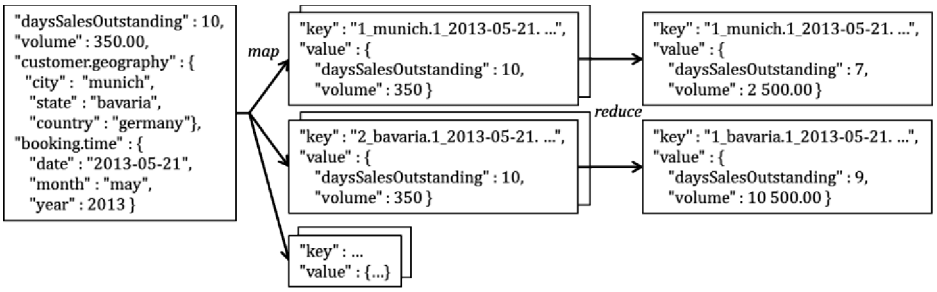

Figure 6: Exemplary application of Map/Reduce on the case study on the procurement of flights.

6 Discussion and Future Work

A new generation of non-relational data stores (NoSQL) has been emerging in recent years. The objective of this paper is to show how these Web 2.0-driven data stores can be used in enterprise application systems. For this, we proposed an architecture that enables the use of non-relational data stores in enterprise applications with respect to the requirements of polyglot persistence and of integrating OLTP- and operational right-time OLAP-scenarios. In the first step of research, the focus is restricted to a single business process. The architecture enables the deployment of the appropriate data model for a certain business entity with respect to its technical and structural characteristics. Furthermore, we propose the concept of operational data components, one for each business entity, to manage the heterogeneous data stores and to integrate their functionality in higher transactional workflows. These components also enrich their business data with multidimensional information. The multidimensional information of all components is integrated and can be analyzed by using the Map/Reduce-paradigm in an analytic data component. We have shown the application of the architectural concept and its analytic capabilities by using a case study on the procurement of flights.
The presented architecture is complementary to traditional data warehouse systems. It can be used in order to deal with the shortcomings because of periodic (long term) data load to support short term tactical and operational business process analysis. By using multidimensional data structures, historical and aggregating data analysis is enabled. Monitoring components track data in order to integrate run-time information in the business process analysis. To conclude, the hybrid architecture offers analytic capabilities within the boundaries of a single business process.
Figure 7 shows the time-value-curve of [Hac04] (adjusted by [Rus11]). Decision situations sharing the characteristics of the curve have the following in common: the more time that elapses following a business event, the more value they lose. The hybrid component architecture can contribute to shortening capture and analysis latency. Of course this is not the case if real-time reaction (within milliseconds) is required. But it is well suited for “right”-time problems. This means, data is delivered when a decision maker needs up-to-date information. We offer the possibility of defining that right point in time by specifying temporal, business, run-time and consumer events in operational and analytic data components. These events trigger the analysis of business process data. Capture latency is mainly reduced by short running, non periodically analytic data integration processes; analysis latency is merely influenced by Map/Reduce-strategies and hardware settings of the Map/Reduce-engine.
For future work we have to refine the architectural concept and enhance it in scenarios spanning more than one business process. We made initial propositions regarding which functionality which type of interface has to offer. Future work will need to address more detail in the design of all interface types. In the case of transactions it has to be discussed whether weaker concepts compared to ACID-transactions are sufficient in some scenarios and how they can be determined and realized. Further, how to realize data consistency in and between data components is a vital area for investigation. Persisting data denormalized and aligned to a single application system may result in redundancies. In order to prevent anomalies and inconsistency, concepts and strategies for data components and workflows have to be developed.

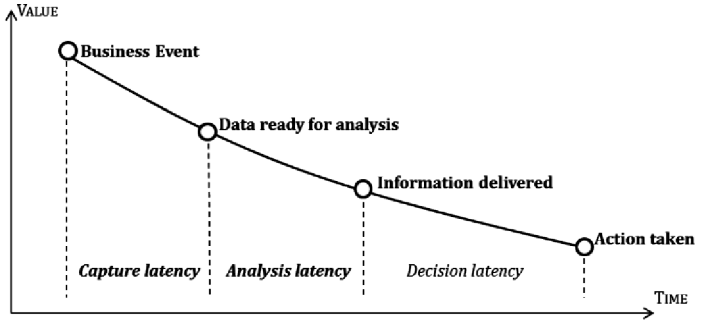

Figure 7: The time-value-curve according to [Hac04](adjusted by [Rus11]).

Parameterizing operational and analytic components, as well as realizing the integration process, are further topics of research. In order to design application systems on the proposed architecture, there has to be a design method, starting at a conceptual business process level and offering integrated design concepts (e.g. meta-models, transformation rules) to derive an application specification that is as complete as possible. It should cover analytic and operational requirements, non-functional requirements to identify data store and transaction concepts, identify data exchange patterns and formats as well as business entities and their managing data components with interfaces and realization. As mentioned earlier, the authors of [Rus11] suggest having bi-directional integration between business and analytic processes. We have to enhance the hybrid architecture to be able to react automatically on certain business events, and therefore shortening the decision latency. Finally, the proposed architecture needs to be evaluated in a more complex case study, including the realization based on a certain technology stack. This should lead to further research topics and substantiate the presented blueprint.

References

[ACN05] B. Azvine, Z. Cui, and D. Nauck. Towards real-time business intelligence. *BT Technology Journal*, 23(3): 214-225, 2005.

[Bre00] Eric A. Brewer. Towards robust distributed systems. (Invited Talk) Principles of Distributed Computing, July 2000.

[CCS93] Edgar F. Codd, Sharon B. Codd, and Clynch T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate, 1993.

[Cod70] Edgar F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[Cro06] Douglas Crockford. Introducing JSON. http://tools.ietf.org/html/rfc4627, 2006. Online accessed May, 21st, 2013.

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[Fie00] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[Gho10] Debasish Ghosh. Multiparadigm Data Storage for Enterprise Applications. *IEEE Software*, 27(5):57–60, 2010.

[GL02] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):51, 2002.

[GR93] Jim Gray and A. Reuter. *Transaction processing: Concepts and techniques*. The Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, San Mateo and Calif, 1993.

[Hac04] Richard Hackathorn. The BI Watch: Real-Time to Real-Value. http://bolder.com/pubs/DMR200401-Real-Time%20to%20Real-Value.pdf, 2004. Online accessed May, 21st, 2013.

[Hec11] Robin Hecht. NoSQL evaluation: A use case oriented survey. In Stefan Jablonski, editor, *Proceedings of the International Conference on Cloud and Service Computing*, pages 336–341, 2011.

[Ind12] Maria Indrawan-Santiago. Database Research: Are We at a Crossroad? Reflection on NoSQL. *15th International Conference on Network-Based Information Systems*, pages 45–51, 2012.

[KHB+11] Jens Krüger, Florian Hübner, Martin Boissier, Johannes Wust, Alexander Zeier, and Hasso Plattner. Main Memory Databases for Enterprise Applications. In *IEEE IE&EM*, 2011.

[KN11] Alfons Kemper and Thomas Neumann. One Size Fits all, Again! The Architecture of the Hybrid OLTP&OLAP Database Management System HyPer. In Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Malu Castellanos, Umeshwar Dayal, and Volker Markl, editors, *Enabling Real-Time Business Intelligence*, volume 84 of Lecture Notes in Business Information Processing, pages 7–23. Springer Berlin Heidelberg, Berlin and Heidelberg, 2011.

[OMG11] OMG. Business Process Model and Notation (BPMN): Version 2.0. http://www.omg.org/spec/BPMN/2.0/PDF, 2011. Online accessed May, 21st, 2013.

[Pla09] Hasso Plattner. A common database approach for OLTP and OLAP using an in-memory column database. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 1–2. ACM, Providence and Rhode Island and USA, 2009.

[Rus11] Philip Russom. TDWI Best Practices Report: Operational Data Warehousing: The Integration of Operational Applications and Data Warehouses. available as PDF on http://tdwi.org/research/2010/10/bpr-q4-operational-data-warehousing.aspx?tc=page0, 2011. Online accessed May, 21st, 2013.

[SF12] Pramod J. Sadalage and Martin Fowler. *NoSQL distilled: A brief guide to the emerging world of polyglot persistence*. Addison-Wesley, Upper Saddle River and NJ, 2012.

[Som11] Ian Sommerville. *Software engineering*. Pearson, Boston and Mass and London, 9 edition, 2011.