An Architecture for Integrated Operational Business Intelligence

Dr. Ulrich Christ

SAP AG
Dietmar-Hopp-Allee 16
69190 Walldorf
ulrich.christ@sap.com

Abstract: In recent years, Operational Business Intelligence has emerged as an important trend in the Business Intelligence (BI) market. While sharing some obvious features with "traditional" data warehouse based BI, some of the requirements for the operational flavor are strikingly different justifying the question how, or even whether, Operational BI fits into a DW environment. We describe an architecture that, even though optimized for Operational BI, integrates smoothly into the broader data warehousing picture.

1 Introduction

While the term Business Intelligence is typically associated with systems and techniques that support the information need for strategic decisions – often based on historical data – Operational BI refers to the application of BI methods (and technology) to the vast number of low-level decisions to be taken in the daily operations of a business. Ideally, the respective reports or queries are tightly integrated into the business processes of a company to provide users with actionable just-in-time information when executing their task[1].

Typical examples would include CRM or call center applications, but also logistics or real-time analysis of technical processes can be named here. Due to its proximity to transactional processes and applications, Operational BI differs from Strategic BI in several aspects, in particular:

- Operational Integration as opposed to Information Integration – Operational BI reports typically have a narrow focus, supplementing a certain operational application; as a consequence, the need to integrate data from various applications (or even systems) is far less pronounced than in the realm of Strategic BI

[1] [Eck] provides a good overview of the topic

- Target group – due to its nature, a much higher number of users on all levels of an enterprise has access to Operational BI

- Latency – to support business decisions effectively, data has to be available for reporting much faster, ranging from within a few hours to sub second latency.

2 Common Approaches to Operational BI

It is a very natural and widely-spread approach to apply standard BI and data warehousing technology to analyze operational data. Particularly the question of whether to incorporate data for operational analyses into an existing data warehouse or not is probably the most fundamental one in discussions about Operational BI.

2.1 Direct Access to Application Persistences

Maybe the oldest and most straight-forward idea when it comes to Operational BI is to directly access data in the application systems. Today several BI vendors promote this approach as "data federation". While avoiding the need to replicate data, the main drawback are the resulting performance implications due to the mixed workload imposed on the operational system. Moreover, data will typically be stored in a write-optimized and thus highly normalized fashion which increases the complexity of BI queries even further.

2.1 The Data Warehouse

Data warehouses are a well established part of most IT infrastructures. In heterogeneous landscapes they serve as a data hub, integrating data from various application systems, taking care of data quality and providing a dedicated point of access to this data for analysis purposes. However, their role as a data hub also has drawbacks with respect to the requirements of Operational BI. First, data warehouses are generally loosely coupled with operational systems. They extract data from there in typically daily (or longer) intervals leading to significant latency issues in operational scenarios. To overcome these problems, (Near-) Real-time DW technologies like SAP NetWeaver BI Realtime Data Acquisition have been developed which allows a reduction of data latency to the range of one to five minutes. Yet, in many situations the data warehouse approach, which introduces additional data layers and processes that need to be maintained, might be considered too heavyweight.

From our point of view, the need to integrate data from various sources should be the decisive argument for using a data warehouse. In many scenarios, e.g. when the relevant data comes from a single application, a leaner approach would certainly be preferable. However, we would like to emphasize that in order to provide scalability and add flexibility, such a lean architecture should integrate into the large scale system landscape and be able to propagate data into a data warehouse.

2.3 The Operational Data Store

Within W.H. Inmon's concept of a Corporate Information Factory (CIF), Operational Data Stores (ODS) are architectural entities specifically designed for the purposes of Operational BI. They play a dual role, supporting both decision support and operational transaction processing[2]. Like data warehouses they *integrate* data from various systems. In fact, as figure 2.1 shows, they can be used as intermediate layers between operational systems and a data warehouse. However, they differ from data warehouses in three important aspects:

- their contents are *volatile*, meaning that they change at much higher frequencies than contents of a data warehouse
- they keep *detailed*, highly granular data and are
- *current valued* in the sense that they contain no (or little) historical data.

In particular the concentration on current data keeps the size of Operational Data Stores manageable and ensures acceptable query runtimes on fresh operational data.

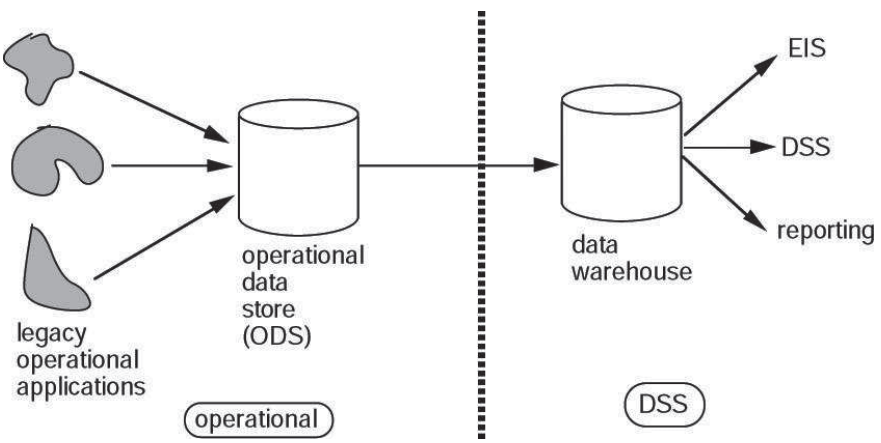

Figure 2.1: Inmon's Operational Data Store[2]

[2] for a detailed discussion of the subject, see [IIB]

3 Operational BI as Part of the Enterprise Data Warehouse Architecture

In this chapter we describe an architecture that allows to combine low data latency with excellent query response times. To this end we place an entity similar to the Operational Data Store inside the operational system and tie it closely to business transactions. In addition we make use of in-memory technology to achieve excellent query response times.

3.1 Motivation and Requirements

There is certainly no one-size-fits-all approach that supports all possible use cases in an optimal way. For relatively simple scenarios with small amounts of data, direct access to the operational sources is most likely appropriate. On the other end of the spectrum, complex scenarios will be best supported by data warehouses with specialized (near-) realtime technology like SAP NetWeaver BI Realtime Data Acquisition. Especially when integration of data from several operational systems is required – and consequently data quality issues tend to arise –, a central data hub fits very well in the overall architecture.

However, in between these extremes lies a number of scenarios that are of modest complexity and do not require integration, yet the underlying data volume makes direct access to operational data impractical. In these cases, replication into a data warehouse for mere performance reasons is certainly undesirable and a more lightweight and flexible solution would be preferable.

Overtime, as a company's IT landscape and processes change, though, it may turn out to be necessary to integrate the accumulated data into a data warehouse. We will take this into account and ensure that, like Inmon's Operational Data Store, our architecture can serve as an access point for extraction into a central data hub.

To summarize the requirements, we focus on a technology that

- enables excellent query response times with minimal data latency
- does not require a full-fledged data warehouse
- can act as an upstream persistence for a data warehouse

3.2 Large Scale Architecture

The architecture we describe in this paper resembles the notion of an Operational Data Store. It shares two of the technical properties in being able to keep volatile and detailed data with expected data latencies in the (sub-) second range. However, it is intended to be able to keep historical data over longer time intervals than Operational Data Stores. It is also closely coupled with operational applications and therefore does not serve as an integration hub.

Figure 3.1 below shows the positioning of our *Integrated ODS* within a typical landscape of operational and decision support systems. From a semantic perspective we would like to point out that the object should not be seen as tied to a specific application, which means that e.g. customer master data could be shared among several applications. We will elaborate on this in Chapter 4 "Reporting View" below.

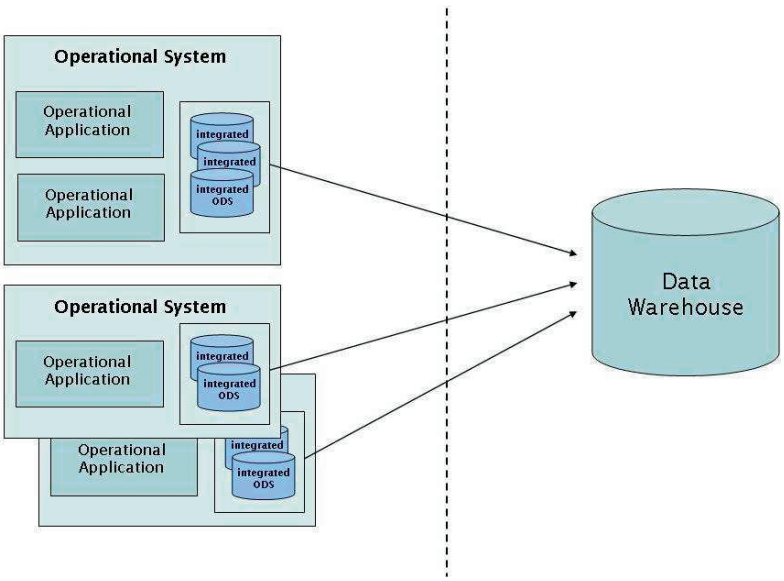

Figure 3.1: Positioning of the Integrated ODS

3.3 ETL Services in SAP Systems

BI experts commonly agree that it is not in general possible to *derive* a meaningful view suitable for BI reporting from a transactional schema automatically; in particular, join operations will generally not suffice to transform operational data into a format that suits reporting needs. Thus one can very well argue that provisioning of (transaction) data in a dedicated reporting format is part of the application logic.

Reflecting this within an SAP landscape, the NetWeaver BI Service API provides services and interfaces that enable SAP applications to pre-process data for BI purposes. *DataSources* provide a metadata description of a view on application data that is specifically designed for BI purposes. Generally they include an extraction program and additional information about the extraction process they support. One of these extraction processes works via the so called *Delta Queue*. When committing transactional changes to the database, an application can write corresponding records for reporting purposes to this container, from which they can be extracted asynchronously and in historical order to the SAP data warehouse. See figure 3.2 for a sketch of this architecture.

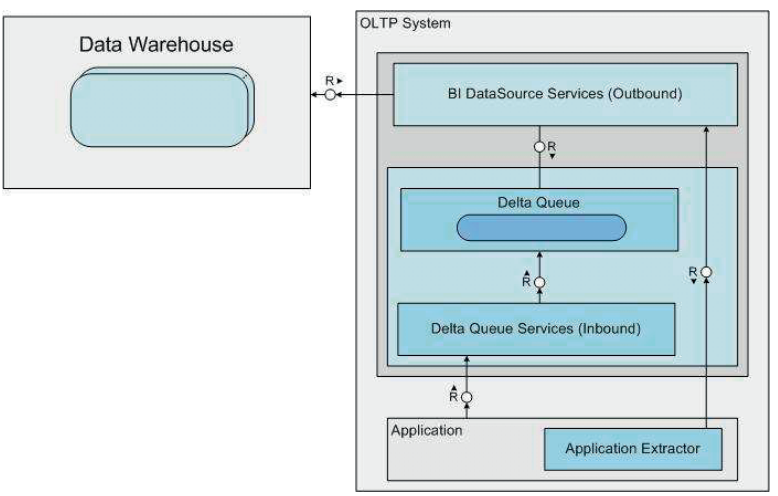

Figure 3.2: NetWeaver BI Delta Queue

3.4 Architecture Details

It is therefore a natural idea to enhance the *Delta Queue* to be capable of executing OLAP queries while keeping its role as an intermediate layer between operational persistence and data warehouse. To be able to hold and query large amounts of data and thus be able to operate independently of a data warehouse, SAP NetWeaver TREX provides excellent column based storage and aggregation technology, which is also used for the SAP BI Accelerator.

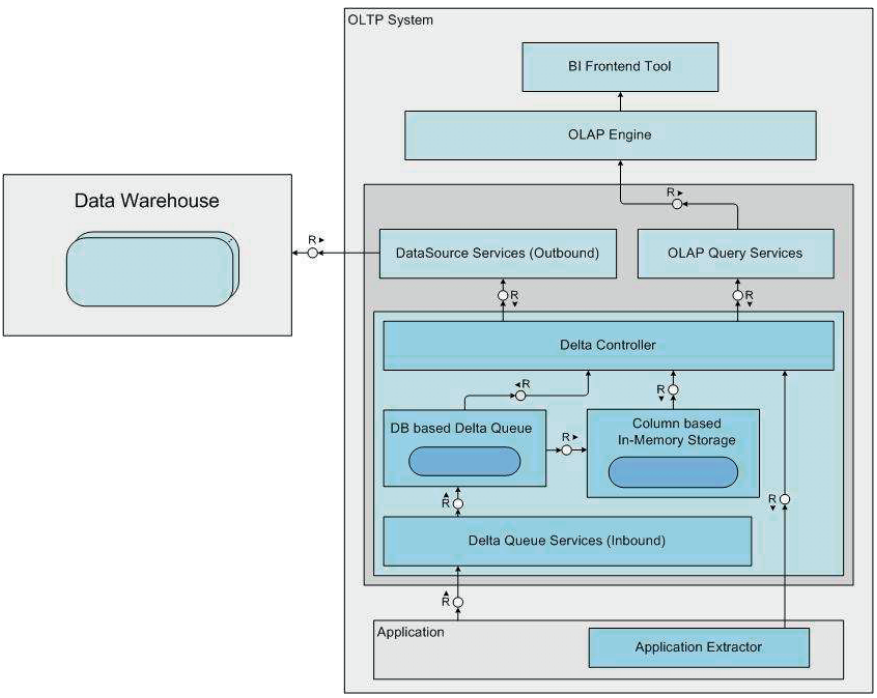

Figure 3.3: The Integrated ODS offers interfaces for OLAP tools as well as for replication into a data warehouse

Column based technologies are optimized for high speed read access and aggregation. However, they are less optimal for typical transactional updates consisting of few records and occurring at high frequencies. Besides the increased costs for continuous updates of the read-optimized structures, a certain communication overhead and a more complicated commit protocol have to be taken into account.

To minimize the effect on transaction runtime, we keep the original Delta Queue as a buffer receiving updates as part of the OLTP transaction. From there data is propagated to TREX's column based storage engine in intervals of about once per minute. An update frequency in that range allows keeping the size of the Delta Queue small and ensures that the communication overhead remains acceptable.

To accomplish both our goals of providing high speed query access and allowing consistent delta replication into the data warehouse, inside TREX data is organized in two layers. One, which will be used for typical Operational BI queries, is keeping the up-to-date data, i.e. the most recent record for every given semantical key. Upon execution of a BI query, the most recent updates from the Delta Queue are merged with the data in this layer to compute the query result. In this way, data latency is reduced to the (sub-) second range. Of course, some sophistication is required for this merge since we cannot simply transfer the cost saved during OLTP commit to the time of query execution. The second layer keeps track of historical changes of records which are needed to supply consistent delta information to connected data warehouses.

4 Reporting View

So far we have discussed processing of data for individual *DataSources*. To explain how to combine the resulting *ODSs* for reporting purposes, we consider a simple scenario involving Sales Order, Business Partner and Product data.

For each of these objects we have a corresponding *DataSource* and *integrated ODS*. To provide a meaningful reporting view combining transaction and master data, we need in addition a thin metadata layer which describes the relation between Sales Orders, Business Partners and Products – basically knowledge of foreign key relationships and specification of Sales Orders as the "transactional facts" suffices[3].

Figure 4.1: Combining master and transaction data to a Reporting View

If master data is harmonized over a suite of applications, the corresponding ODS's like Products or Business Partners can be shared with other reporting views, e.g. in a Delivery scenario.

[3] Recent developments in NW BI's Analytical Engine allow to derive the metadata for NW BI's Olap Processor at runtime from (in our specific case) the DataSource metadata.

5 Summary

The most common approaches to Operational BI, namely using data warehouse or data federation functionality, both are appropriate in a certain range of scenarios. However, they tend to have deficits when it comes to either performance or data latency and they are quite loosely coupled with business transactions. In particular, in scenarios with high data volume produced by a single application an integrated, lean and yet flexible approach is desirable.

We therefore argue in favour of an architecture that allows tight integration of BI functionality into business applications and consequently can resolve the performance vs. latency dilemma to a large extent. It should be considered a lightweight extension of data warehousing functionality in the operational system adding flexibility in scenarios where using a data hub seems inappropriate. However, we also emphasize that in order to achieve true flexibility – e.g. in case of changing requirements concerning data integration or lifecycle –, it is most important to ensure integration of the architecture with the data warehouse.

Bibliography

[Eck] Eckerson, W.W.: Best Practices in Operational BI – Converging Analytical and Operational Processes. TDWI Best Practices Report, 2007

[IIB] Inmon, W.H.; Imhoff, C., Battas, G.: Building the Operational Data Store, Second Edition. John Wiley & Sons, 1999.