A Case for Online Mixed Workload Processing

Jens Krueger
Hasso Plattner Institute

Christian Tinnefeld
Hasso Plattner Institute

Martin Grund
Hasso Plattner Institute

Alexander Zeier
Hasso Plattner Institute

Hasso Plattner Institute for IT Systems Engineering
University of Potsdam
Prof.-Dr.-Helmer-Str. 2-3
14482 Potsdam, Germany

ABSTRACT

Database systems in the context of business data processing are segmented into two categories: those intended for online transaction processing (OLTP) and those for online analytical processing (OLAP). Over the last 15 years, database management system (DBMS) proposals directly addressing one of those categories were most represented in terms of academic publications and variety of commercial products in the domain of enterprise computing.

In contrast, the most innovative DBMS proposals in this century were invented not by addressing a well-known category but by following a methodology that purely focuses on the application characteristics as practiced by Amazon or Google. This paper applies a part of that methodology to the field of enterprise applications in order to evaluate what extend they are covered by the categories OLTP and OLAP. The evaluation shows that there are enterprise applications that reveal a mix of those characteristics which are usually exclusively associated either with OLTP or with OLAP and therefore cannot be addressed adequately by traditional DBMS.

The paper contributes by pointing out that those applications cause an online mixed workload and by explaining what properties a corresponding specialized DBMS should have and how this category of enterprise applications could benefit from it.

1. INTRODUCTION

The progression of DBMS development has been very technically oriented with the evolution of the underlying data model acting as main guideline. For a long time, it was a given paradigm to try to address many different applications with a very limited set of different database systems. Hereby, relational database management systems dominated the market as companies could use them for their daily business involving online transaction processing [16]. At the beginning of the 1990s, large enterprise companies were no longer able to run reports on their data stored in their relational database management systems. This shortcoming was originated in the inability of existing systems to process the involved complex queries on very large amounts of data in a reasonable time frame. As a result, database systems addressing the category of online analytical processing were invented which emphasize the use of specialized data structures to overcome the previously mentioned problems as described by Codd [3]. Since then, the market for database systems used in the context of enterprise systems is segmented into OLTP and OLAP, while the first classification is characterized by a mix of reads and writes to a few rows at a time, whereas applications belonging to the latter are characterized by bulk updates and large scans of portions of the database.

Based on our work with systems and customers of SAP, we observe that applications frequently do not partition neatly into one of these two categories. In particular, many applications that involve an OLTP-style workload require the ability to frequently compute OLAP-style aggregate queries, for example, to measure stock levels and balances. However, this categorization of workloads is not entirely reflective on modern enterprise computing requirements.

First, there is an increasing need for real-time analytics which is up-to-the-moment reporting on businesses processes that has traditionally been handled by warehousing systems. Although warehouse vendors are doing as much as possible to improve response times (e.g. by reducing load times), the explicit separation between transaction processing and analytics systems introduces a fundamental bottleneck in analytics response times. For some applications directly answering analytical queries from the transactional system is a superior approach.

Secondly, there exist applications such as available to promise (ATP) (that determines if an order can be filled) which need to issue OLTP-style queries to update stock levels and mark orders as filled while processing OLAP-style queries to determine if there are sufficient stock levels or other resources available to fulfill orders.

1.1 Application Characteristics

Application characteristics in the context of computer science describe prominent properties and aspects of a piece of
software. The selection of the different properties may vary in the different fields of computer science. The identification and investigation of those properties can result in motivation, guidance, and validation for the work in the respective fields of computer science.

In contrast, in other domains than enterprise computing the consideration of application characteristics has a continuously rising influence on the design of the DBMS to be used. According to Gilenson et al. [7], application characteristics describe how the application interacts with the database system. Therefore, the examination of application characteristics reveals a requirement description towards a database system for that particular application. One example for applying this methodology is the use of object-oriented DBMS in the field of computer aided design (CAD) engineering: CAD engineers tend to work on and modify one large object exclusively, which makes an object-oriented DBMS a good match. A more recent example for applying this methodology is Amazon which used off-the-shelf relational DBMSs in their early days and suffered from massive database scalability problems which were addressed by short-cycled hardware upgrades. As this procedure did not address the scalability problem at its core, the team at Amazon analyzed their application characteristics and came up with a set of specialized DBMS such as Amazon Dynamo or S3.

1.2 Related Work

According to Vogels et al. [4], Amazon identified their application characteristics by focusing on the database requirements of their main business processes. Here, they identified key characteristics such as the size of an data object, the involved query model (e.g. primary key access only or multi-attribute queries), the need for strong consistency, needed response time or the involved data schema. With those characteristics on hand, they were able to divide their applications and involved processes into discrete groups which then could be used as requirements blueprints for the previously mentioned specialized DBMS.

More focusing on different application areas, Stonebraker and Cetintemel state [15] that the last 25 years of commercial DBMS development can be summed up in a single phrase: One size fits all. Furthermore, they think that traditional DBMS architecture has been used to support many data-centric applications with widely varying characteristics and requirements. As a consequence, they argue that there are several application areas where a database system that directly addresses the application characteristics can improve the performance at least by a factor ten. This claim is backed up by Stonebraker et al. [14] who compare commercial implementations of specialized architectures and relational DBMS. Amongst others, the following application areas are cited by Stonebraker et al. [14] as areas where application characteristic oriented database systems can result in performance gains: text search, scientific applications, data warehousing and stream processing.

Following this assumptions, managing of enterprise data seems to be solved as those applications are traditionally associated with pure transactional or analytical processing. However, to the best of our knowledge, characteristics of enterprise applications have not been revisited the last decade.

2. ENTERPRISE APPLICATIONS

Fowler [5] points out that enterprise applications are data intensive, time critical, and are triggered either by possibly several concurrent human users or automatically e.g. by batch runs. On the other hand, he says that he cannot give a precise definition which applications are considered as enterprise applications but gives some indication of his meaning by listing examples of enterprise applications. The list includes payroll, shipping tracking, supply chain management, accounting, customer service, and foreign exchange trading.

Nowadays, enterprise applications are built on a data management infrastructure that was designed to meet a specific set of requirements for OLTP systems. In the meantime, enterprise applications have become more sophisticated, addressing for example legal regulations, governmental compliance, new accounting principles, and global supply chains. In addition, data set sizes have increased, requirements on the freshness of input data have been strengthened, and the time allotted for completing business processes has been reduced.

In contrast, Section 1 shows that outside of the enterprise scope new DBMS developments were derived from focusing on the application characteristics of the applications to be served. As a consequence, this paper aims at reevaluating the paradigm of dividing enterprise applications into OLTP and OLAP.

From an ideal point of view, it would be desirable to investigate the application characteristics from all the different enterprise applications. As this is not feasible in the context of a paper, one has to decide whether to segment them horizontally or vertically. A strict vertical segmentation would result in picking out one application and analyze this application in a very detailed way. But this will not provide the desired cross section of different application characteristics of enterprise applications. On the other hand, a strict horizontal segmentation would still result in too many applications to be investigated at an adequate level. Furthermore, as one application addresses a large variety of business processes it make sense to just pick out one business process. Consequently, a combined vertical and horizontal segmentation is done based on a scenario. That scenario involves a company producing and selling goods and includes the following aspects:

1. At the end of a year, a company plans the demand for its products in the next year. This plan will be used as input for the production.
2. Throughout the next year, customers contact the company and order products.
3. With every new order, the availability of the requested products has to be checked in order to tell the customer if and when the products can be shipped.
4. Some customers may fall behind on their payments for received products. Therefore, payments must be tracked and payments reminders must be generated for outstanding payments.
5. After the first quarter in the new year, the company wants to analyze its sales in order to see how well they are performing.
This scenario results in the following subset of enterprise applications that will be investigated:

Demand planning in the context of supply chain management, sales order processing in the context of enterprise resource planning, available to promise in the context of supply chain management, dunning in the context of financial accounting, and sales analysis in the context of enterprise resource planning.

**Demand Planning**

Demand planning is used to estimate future sales by combining several sources of information such as previous sales, introduction of new or discontinuation of old products, market forecasts or general events that could have an impact on buying behavior. The resulting outcome influences e.g. production planning which determines how many products must be build within a certain time interval.

Based on our investigation, the following application characteristics of a demand planning application can be identified: a single demand planning run for a company involves a large amount of data (up to 100 gigabytes). This is caused by hundred of thousands of different products and their respective variations and configurations and the fine-grained timely planning level that allows to plan on day or hour basis. The main operations on that data are aggregation and disaggregation as with every planning run all current numbers are aggregated. Then, some high-level changes will be applied which then will be pushed down again to the most fine granular level. The operation usually involves far more read operations then write operations as all numbers will be read but only a fraction of them gets actually changed. The underlying system must be able to perform the involved operation with a sub-second response time as a human planner interactively and continuously works with the system. Due to multi user concurrency, isolated user contexts must be created to guarantee a consistent view on the data while planning. This acts as a multiplicator for the overall storage and computing requirements.

**Sales Order Processing**

The purpose of sales order processing is to capture a customer order. A sales order consists of a general part for each order that includes e.g. the name of the customer, its billing address or the method of payment. Furthermore, details about the individual products are stored as line items. Sales order processing involves read and write operations on transactional data. Read operations are mainly simple select operations, while write operations insert new sales orders and sales order line items. Read operations dominate write operations in total numbers. Figure 1 shows an analysis of sales order processing database logs from over 50 enterprises which undermines the dominance of read operations. All transactions involve small amounts of data processed by highly predictable queries. The application is used interactively by clerks who rely on a fast response time.

**Available to Promise**

The decision to accept an order for a product is predicated on the successful completion of the available to promise check. ATP determines whether sufficient quantities of the product will be available in current and planned inventory levels at the required date. To support flexible operations such as reordering, where a high-priority order overrides previously accepted low-priority orders, ATP must aggregate across pending orders rather than just materializing inventory levels.

ATP encompasses read and write operations on large sets of data. Read operations dominate and result in aggregating the respective time series. Write operations work on fine-granular transactional level e.g. to declare products as promised to customers. Variance in data types and records are small. The amount of data grows by introducing e.g. new products, customers or locations.

**Dunning**

Dunning is the process of scanning through unpaid sales orders to identify ones that are overdue, generating reminder notices for those orders, and tracking which notices have been sent.

Dunning involves read and write operations on large amounts of continuously growing transactional data. Either the read operations are batch-oriented as the list of open items gets processed customer by customer. Or the read operations are analytical queries resulting in data aggregation e.g. when the outstanding payments for a certain customer group in a certain fiscal year is determined. Read operations only access very few attributes of the open items. Inserts are needed to keep track of issued payment reminders. Updates are needed to modify a customer’s master data after a dunning run.

**Sales Analysis**

Sales analysis provides an overview on historical sales numbers. Usually, the analysis is used for decision making within an organization. It performs read-only operations on large amounts of data. The involved data contains data on a transactional level as well pre-aggregated data. The involved data contains recent and historical data. The involved queries are multidimensional. Although there are common queries which can be derived from cyclic asked questions, the overall amount of queries is unpredictable due to the ad-hoc nature of the questions.

3. **ONLINE MIXED WORKLOAD**

From the analysis of the applications in the previous section we now want to draw a combined picture of the inte-
The integrated nature of modern enterprise applications. While each of the investigated application can be seen as an enclosed piece of software, the challenge for modern enterprise applications is to integrate all those application into one system using one single data source.

The main goal of our top-down approach is to identify the characteristics of those applications by a common denominator and we explicitly choose not to investigate the way they are currently implemented, but rather focus on their behavior derived by business functionality.

To avoid misconceptions, we want to clarify on the terms transactional and analytical: Transactional — typical data entry applications — require a certain data storage, as do analytical applications. Typical analytical applications follow distinct patterns and fulfill a special purpose like data cleansing, data consolidation, hierarchy adjustments, and many more. Queries on this data typically selects all data and step by step reduces the amount of processed data — following a certain navigation path or drill-down pattern. In our terms, analytical queries might only fit in the last steps of such patterns, but are of great importance for the corresponding business processes. How such requirements can be met today we will show in the next section.

To summarize for our understanding, analytical queries read relatively large amounts of data but are typically restricted to a certain customer or product.

### 3.1 Application Characteristics Comparison

Table 1 shows the combined evaluation of all applications. To show the diversity of the applications we colored the transactional characteristics in light grey and the analytical characteristics in dark grey. The first deduction is that most of the applications do not fit the common separation between transactional and analytical applications. Even though the sales order processing application carries transactional behavior only, both the sales analysis and dunning depend on the data from this application. This respective applications cause a workload with the following characteristics: they operate on transactional data. The operations include read and write operations. Read operations include complex, analytical queries on large amounts of data.

How transactional and analytical behavior interacts can be determined by the required context of the operation. We define the context of a transaction as a metric that determines the amount of required information to perform a decision in a business process. If the context of the decision is small than only a small amount of data is processed. The larger the context gets the more amount of data is required to be read.

As shown in Table 1, there are single applications in the domain of enterprise computing that reveal the characteristics of both workloads which implies that a mixed workload can be originated by a single application. Therefore,

---

<table>
<thead>
<tr>
<th>Granularity of Data</th>
<th>Demand Planning</th>
<th>Sales Order Processing</th>
<th>Available to Promise</th>
<th>Dunning</th>
<th>Sales Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations on Data</td>
<td>Transactional</td>
<td>Transactional</td>
<td>Transactional</td>
<td>Transactional</td>
<td>Pre-Aggregated</td>
</tr>
<tr>
<td>Preprocessing of Data</td>
<td>Read &amp; Write</td>
<td>Read &amp; Write</td>
<td>Read &amp; Write</td>
<td>Read &amp; Write</td>
<td>Read-Only</td>
</tr>
<tr>
<td>Timeframe of Data</td>
<td>Historical &amp; Recent</td>
<td>Recent Only</td>
<td>Historical &amp; Recent</td>
<td>Historical &amp; Recent</td>
<td>Historical &amp; Recent</td>
</tr>
<tr>
<td>Update Cycles of Data</td>
<td>Always Up-to-Date</td>
<td>Always Up-to-Date</td>
<td>Always Up-to-Date</td>
<td>Always Up-to-Date</td>
<td>Cyclic Updates</td>
</tr>
<tr>
<td>Amount of Data per Query</td>
<td>Large</td>
<td>Small</td>
<td>Large</td>
<td>Large</td>
<td>Large</td>
</tr>
<tr>
<td>Complexity of Queries</td>
<td>High</td>
<td>Standard</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Predictability of Queries</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Response Time of Queries</td>
<td>Seconds</td>
<td>Seconds</td>
<td>Seconds</td>
<td>Seconds to Hours</td>
<td>Seconds to Hours</td>
</tr>
</tbody>
</table>

**Table 1: Comparison of Identified Application Characteristics**

**Table 2: Characteristics of Online Mixed Workload Processing**
the differentiation between OLTP and OLAP applications should be enriched by a third category that describes applications which cause a mixed workload. This category could be labeled as online mixed workload processing and reveals characteristics as shown in Table 2.

### 3.2 Origins of Mixed Workloads

Powley et al. point out [13] that mixed workloads refer to workloads with different characteristics. Usually, the term mixed workloads is associated with two different workloads including the processing of small sets of transactional data at a time including write operations on the one hand and complex read-only queries on large sets of data on the other hand. There are basically two different approaches how to handle them: the by far most popular one is distributing them onto two or more different systems, as described before in this chapter, or by managing the workload on a single DBMS instance.

When addressing the mixed workload by handling them with different systems, one solution is to use a data warehouse. This data warehouse gets its data in a cyclic manner from one or more transactional DBMS via an ETL process as already described throughout the thesis. Another solution in this field are so called operational data stores (ODS) as described by Inmon [9]. Hereby, the data of transactional DBMSs get instantly loaded into an ODS where analytical queries can be executed in two to three seconds.

When handling mixed workloads on one single DBMS, one can distinct between two different approaches, namely dynamic resource allocation and workload adaption, including admission control. Brown et al. [2] propose an algorithm that automatically adjusts memory allocation to achieve a set of response time goals for a complex workload in DBMS. Pang et al.[10] propose an algorithm for mixed workloads that uses admission control, allocating memory and assigning priorities based on current resource usage, workload characteristics and performance statistics. Baryshnikov et al. [1] propose to reduce the amount of memory used for query compilation in a DBMS by blocking compilations at certain periods of time until their execution resources become available. Parekh et al. [11] propose a throttling technique to limit the impact of on-line database utilities such as backup/restore, data re-organization, or automatic statistic collection on user work. Nevertheless, all those proposals can be rather seen as fine-tuning of existing relational DBMS which sacrifice to some extend the aspect of real-time processing as described by Plattner [12]. While main memory databases have the potential to lead to new architectures which may well cope with demands encompassed by a mixed workload as described by Plattner [12]. While main memory databases have significantly faster read and write rates than disk-based variants the sequential access in column stores has been proven to be advantageous for queries focussing on certain attributes over many tuples rather than on single entities.

Besides, a widening gap between the growth rate of CPU speed and memory access speed can be observed. This trend argues for the usage of compression techniques requiring higher effort for de-compression, since CPU power grows mostly-read operations on large sets of data with a projectivity on just a few columns.

### 4. FUTURE WORK

As a mixed workload is already in the interest of researchers with regards of combining transactional and analytical workloads, we see two different ways of carrying out future research in order to address the actual requirements of enterprise applications.

#### 4.1 Need for a Benchmark

Our observations show that there is spreading gap between existing benchmarks and enterprise applications. The goal of a benchmark is to drive industry and technology forward and thus benchmarks have a life-time. Of course, the enterprise software industry moves differently from Web 2.0 startups and their requirements of data storage. However, the requirements changed and can no longer be met alone by the existing benchmarks.

The biggest shortcoming of the existing evaluation possibilities is that they focus on only one workload at a time. As shown in the previous section single enterprise applications cause a mixed workload and only parts could be mapped to existing benchmarks. The most interesting change is that decision making can not only be found at the end of a business process, but is involved in several steps throughout the whole process. A typical example is that already at the beginning of a purchase process in a sales application the first decisions are made. Which price the customer pays is no longer only determined by a fixed configuration, but because the turnover of the last three month is compared and validated against the last year. More and more information is required to build the context of decision making.

The benchmarks TPC-E and TPC-H each claim one part of the process. In TPC-E a brokerage company is simulated performing orders and complex transactions, while in TPC-H its stated that it is a decision making benchmark. TPC-H is based on a separation between operations and decision support and information only arrives via scheduled batch updates. In conclusion, the current TPC suite does not include a benchmark that covers a mixed workload originated by a single application as they rely on conventional workload classification.

#### 4.2 DBMS Draft for Mixed Workloads

The mixed workload characterization defined in the previous section is subject to several publications while the common argumentation is to keep the systems separate as described for example by French [6]. However, recent developments, such as main-memory databases, lightweight compression techniques, and column-oriented storage schemas have the potential to lead to new architectures which may well cope with demands encompassed by a mixed workload as described by Plattner [12]. While main memory databases have significantly faster read and write rates than disk-based variants the sequential access in column stores has been proven to be advantageous for queries focussing on certain attributes over many tuples rather than on single entities.
and resources are wasted for waiting on I/O operations to complete, thus trading CPU power for I/O improvements is feasible while improving the bandwidth utilization by the factor achieved in data amount reduction. Data compression techniques exploit redundancy within data and knowledge about the data domain for optimal results. Compression applies particularly well to columnar storages. Since all data within a column a) has the same data type and b) typically has similar semantics and thus low information entropy, i.e. there are few distinct values in many cases.

Database engines that combine those technologies can rapidly execute analytical queries because they exploit the specific characteristics of a column store by reading most of the data sequentially and compressed from main memory while being aware of memory hierarchies as much as possible. Rapid access to single business entities spread over many tables is possible as well but comes with a slight performance penalty due to the random access. However, the point where this tuple-reconstruction overhead outpaces the advantages of a column-oriented storage is much higher in main-memory based databases due to random access capabilities which makes the usage of such technology appealing for transactional workloads that incorporate access to single instances as well as at the same time complex queries can be executed.

5. CONCLUSION

This paper has investigated the application characteristics of a set of enterprise applications with regards to the data management and performed an evaluation to what extend the characterized applications are covered by the categories OLTP and OLAP. The evaluation shows that there are applications whose characteristics match with those associated with OLTP or OLAP. On the one hand, sales order processing is a typical example of an OLTP application and on the other, sales analytics is a typical example of an OLAP application. In contrast, available to promise, demand planning, and dunning are not covered by this classification as they process small sets of transactional data at a time including write operations as well as complex, unpredictable mostly-read queries on large sets of data. As a consequence, the workloads that adequate DBMSs need to handle diversify more and more. Enterprise applications of the earlier days had to perform OLTP style queries mostly. The trend has moved towards a mix of OLTP and OLAP style queries, with an increasing part of the latter.

The resulting mix of workloads is usually described as originated by different applications and therefore addressed by separating the applications onto different database systems: this solution cannot be applied here as the paper has shown that there are single applications which cause a mixed workload originated by a single application. Therefore, the distinction between OLTP and OLAP should be enriched by a third category that describes the aforementioned online mixed workload processing.

This third category could benefit from a specialized DBMS as this would significantly speed up each single transaction and therefore the overall executed business processes. In addition, new data could be instantly included as a batch updated would become obsolete. Overall, this would enable the execution of complex queries on large datasets in an increasing number of applications, which is necessary to incorporate existing data in all relevant business steps.

6. REFERENCES


