

# Normalization in a Mixed OLTP and OLAP Workload Scenario

Anja Bog<sup>1</sup>, Kai Sachs<sup>2</sup>, Alexander Zeier<sup>1</sup>, and Hasso Plattner<sup>1</sup>

<sup>1</sup> Hasso Plattner Institute, University of Potsdam, 14482 Potsdam, Germany  
{anja.bog,alexander.zeier,hasso.plattner}@hpi.uni-potsdam.de

<sup>2</sup> SAP AG, 69190 Walldorf, Germany  
kai.sachs@sap.com

**Abstract.** The historically introduced separation of online analytical processing (OLAP) from online transaction processing (OLTP) is in question considering the current developments of databases. Column-oriented databases mainly used in the OLAP environment so far, with the addition of in-memory data storage are adapted to accommodate OLTP as well, thus paving the way for mixed OLTP and OLAP processing. To assess mixed workload systems benchmarking has to evolve along with the database technology. Especially in mixed workload scenarios the question arises of how to layout the database. In this paper, we present a case study on the impact of database design focusing on normalization with respect to various workload mixes and database implementations. We use a novel benchmark methodology that provides mixed OLTP and OLAP workloads based on a real scenario.

**Keywords:** Combined OLTP and OLAP Workloads, Database Design, Normalization, Workload Mix, Benchmarking

## 1 Introduction

Online transaction processing (OLTP) systems are the backbone of today's enterprises for daily operation. They provide the applications for the business processes of an enterprise and record all business movements, e.g. sales and purchase orders, production, billing, and payments. For strategic insights to make business decisions for the future and to monitor the performance of their business, enterprises utilize online analytical processing (OLAP) systems. Since OLTP and OLAP present very different challenges for database architectures and transaction throughput is essential for OLTP, they have been separated into different systems in the mid 1990s to avoid impairments. The business data used in both domains is the same, however, stored in differently optimized structures. The separation implies several drawbacks, for example, keeping redundant versions of the same data, data staleness for analytics, as data in the OLAP system is updated only periodically, or having only a selected data set available for analytics.

Hardware development, that is, multi-core technology and systems with large main memory provide more powerful systems as a basis for business processing. In [26], Nambiar and Poess analyze Moore's law [24] directly with respect to

transaction processing systems and show that it holds true even on this higher level. They show that systems have been (and most probably will be) capable of running ever larger transaction processing workloads with a growth that closely resembles Moore’s predictions. In combination with recent developments of database systems, especially column-oriented storage in addition to lightweight compression techniques, the question arises if the historically grown separation of the two domains is still necessary. Plattner [28] states that in-memory column-oriented databases are capable of running OLTP and OLAP applications in one and the same system in a mixed mode.

Running OLTP and OLAP in a mixed mode directly leads to the question of how to design the database. Database design is a major distinguishing characteristic between the OLTP and OLAP domains and has conflicting goals for optimization. So far, efforts to optimize database design have been focused on one or the other domain, but not considering a mixed workload of both. Using a methodology to benchmark composite OLTP and OLAP systems that enables the simulation of mixed workloads, we present a case study of the impact of database design for different workload mixes and different database storage architectures. Regarding the database design, our focus in this study lies on different degrees of normalization of the database schema. In this paper, we report our results for the behavior of databases with primary storage on disk, in main memory as well as row and column-orientation. A brief announcement of this case study was presented in [4]. In this paper, we give a complete introduction to the schema variants analyzed in the case study and present results for a wide range of mixed workloads and database architectures.

In the following section, we discuss the background of our study by giving examples for applications that do not clearly fit either into the OLTP or OLAP domain. Further, we provide a short introduction of OLTP and OLAP database design and its normalization in particular and we review related work in the area of database benchmarking and optimizing database design for OLTP and OLAP. In Section 3 we introduce our database design variants related to normalization. We apply these variants in a comprehensive case study in Section 4, discuss the results of the study and finally, we summarize our findings.

## 2 Background

In this section we present applications which do not clearly fit either into the OLTP or OLAP domain, we observe that database architectures exist that already cater for mixed OLTP and OLAP workloads and we present an overview of the benchmarks for OLTP and OLAP systems. Further, we discuss database design in general and optimization of database design in detail and we point out work related to database optimization as our case study aims at optimizing databases regarding mixed workloads.

### 2.1 Applications Spanning Across Domains

Applications are classified to belong to one or the other domain based on their data processing characteristics. French [10] characterizes the OLTP workload as

simple mixed queries of inserts, updates, and deletes that are relatively short running, retrieve a large number of columns of a table and touch only a small set of rows. In contrast, OLAP operations are characterized as complex, read-mostly queries that touch a small number of columns, but a large number of rows. According to Abolhassani [1] applications exist in either domain that show the application characteristics typical for the other domain. As a result, although a line is drawn between OLTP and OLAP, it is not always obvious from the characteristics of an application which domain it belongs to.

Krueger et al. [21] point out several examples, e.g., dunning and available-to-promise (ATP), where this classification is not clear. Dunning and ATP are categorized as OLTP operations, which need the latest business data in order to determine the right results. Dunning runs, for example, are triggered for entire sales areas or customer groups, and therefore touch a relatively large data set. To avoid long run times in relation to typical OLTP operations and to avoid queuing other OLTP operations, workarounds have been introduced. For dunning runs it means that they are triggered only periodically, e.g. during times of low system load. To further reduce the impact of long response times through reading large amounts of data on the OLTP system, secondary tables, e.g., holding information about unpaid bills for dunning or materialized aggregates, i.e., time lines for availability-to-promise are created and kept up-to-date in synchronization with the business transactions.

## 2.2 Combined OLTP and OLAP Database Architectures

Because of the need of OLAP to analyze ever fresher and more complete OLTP data, database architectures and systems that allow a combined workload of OLTP and OLAP are proposed. Röhm [29] introduced his unified architecture for OLTP and OLAP to allow OLAP clients the access to up-to-date data. He proposes a middleware-based database cluster with OLTP and OLAP nodes side by side where each OLAP node holds its own copy of the data with a varying degree of freshness. In contrast to this middleware-based approach, HyPer [18] handles OLTP and OLAP using hardware assisted replication mechanisms. This achieves consistent and up-to-date snapshots of the transaction data. Other prototypes such as OctopusDB [8] and Hyrise [12] aim to avoid keeping multiple copies of data and instead adapting the storage layout of the data according to usage patterns in the workload.

## 2.3 Related Work in Benchmarking OLTP and OLAP

The methodology we use in our case study is targeted as a new benchmark for mixed OLTP and OLAP workloads. However, it is not a standard and we briefly discuss why it is desirable to introduce a new benchmark and note where its benefits lie regarding our case study.

As transactional and analytical benchmarking have been treated as separate domains so far, being able to use the existing benchmarks only limited statements can be made concerning the ability of data management systems to handle

a mixed workload. Running an OLTP and an OLAP benchmark in parallel, is a valid approach to create a mixed workload. However, with the current benchmarks only a partial picture of the actual performance of a hybrid system will be achieved, measuring the effects of hardware resource contention. Because the benchmarks still run on their dedicated data sets, conflicts arising from data access to the same data set are not observed. Harmonizing the different requirements for the design of the data set for optimal access are a characteristic of particular interest for a mixed workload.

Regarding combined OLTP and OLAP workloads, we are aware of two non-standard benchmarking approaches, one is the composite benchmark for transaction processing and operational reporting (CBTR) [3] and the other is TPC-CH [11]. CBTR is based on the order-to-cash process of a real enterprise system that is widely applicable in many industries and TPC-CH is derived from the standard benchmarks TPC-C [31] and TPC-H [32].

Both benchmark proposals include workload mix as a new parameter that is of importance in the mixed OLTP/OLAP scenario. Workload is defined by the types of actions that take place and their frequency of execution. The contribution of the OLTP and OLAP-style workloads to the total workload should not be constant because transactional and analytical processing follow conflicting optimization goals and consequently the share of OLTP and OLAP-style actions has an impact on the decisions to optimize a combined system. TPC's transactional web e-Commerce benchmark TPC-W [30], which has been marked as obsolete since 2005, but is still in use, explicitly models different workloads of its basic transactions in order to reproduce diverse user behavior. These are browsing, shopping and ordering. Similar to the mix of OLTP and OLAP actions these mixes cover conflicting optimization goals, e.g. fast access of a large amount of data during browsing versus providing hassle-free insertion of data during ordering. For our study of the impact of database design, workload mix is of particular interest.

In our case study we used CBTR because its data schema and the included data are the most realistic concerning data found in real enterprise systems and we believe our results concerning the impact of database design decisions under varying workloads to be more accurate. In [3], we provide a detailed description of the scenario, database design and statements used in CBTR.

## 2.4 OLTP and OLAP Database Design

For both, OLTP and OLAP, basic rules for the creation of optimized database designs exist. The resulting database schemas, however, differ to a great extent and the design goals for optimizing OLTP and OLAP systems are in conflict, meaning that a design which is optimized for OLAP performance degrades OLTP performance and vice versa [9].

The borders between the OLTP and OLAP domain are increasingly blurring when observing the characteristics of the involved data set. The data set of OLAP systems, though part of it is highly optimized for multi-dimensional queries and aggregation, increasingly bears similarity with the OLTP data set.

According to Inmon [15] operational data stores (ODS), which are part of the OLAP environment hold almost fresh OLTP data on the transactional level of detail, i.e., the same or a very similar logical schema as in the OLTP system. The freshness of the data within the ODS ranges from updates appearing in the ODS only seconds later to 24 hours or more, depending on the ODS class. As data in the ODS ages, it passes into the data warehouse.

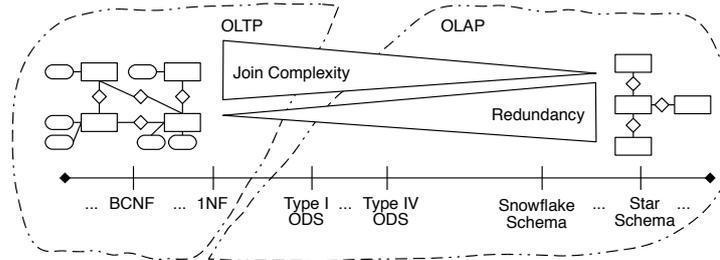


Fig. 1. Analytical vs. Transaction Processing Schemas

In Figure 1 we illustrate the database schema variants used in OLTP and OLAP systems. They are ordered according to the join complexity, a fixed query occurs if the underlying schema is varied. With the decreasing complexity of joins, the level of redundancy within the data set increases.

### 2.5 Normalization in Database Design

In this section we focus on normalization as this is the main differentiating factor when comparing OLTP and OLAP database schemas.

To achieve a sound logical database design and in particular to reduce redundancy within data that can cause false relationships and inconsistencies, the principles of normalization were developed [13]. In [6], Codd introduces the objectives of normalization, that include obtaining a powerful retrieval capability by simplifying the set of relations, removing undesirable insertion, update and deletion anomalies, and reducing the need for restructuring when adding new types of data, thus increasing the life span of application programs.

However, for an increased level of normalization of a set of relations a penalty towards data retrieval has to be accepted. Data that could have been retrieved from one tuple in a denormalized design may have to be retrieved from several tuples in a normalized design. Kent [19] acknowledges, that the highest level of normalization need not be enforced, where performance requirements have to be taken into account. Mullins [25] enforces normalization by pointing out that a normalized schema should never be denormalized unless a performance need arises which cannot be solved in any other way.

Denormalization can be observed in the data schemas of productive systems for analytical processing as well as in transaction processing. Bock and Schrage

[2] discuss and give examples how denormalization can be used to improve system response times for transaction processing, while avoiding redundant storage and without incurring prohibitive data maintenance. Furthermore, Mullins [25] names different types of denormalization techniques, of which the types *redundant data*, *repeating groups*, and *derivable data* can still be observed in today’s transactional systems. Redundant data means adding columns of a table  $B$  to table  $A$  if they are always queried in combination with table  $A$ . This is only advisable if the included columns contain data that does not change frequently. Repeating groups comprises of adding more columns for an attribute that can have multiple, but a limited and small number of values, instead of normalizing this attribute into an own table with a foreign key relationship. A typical example is storing up to three telephone numbers for a customer. Here, three *telephone number* columns are used instead of three rows in a second table. Derivable data comprises of pre-computing data and storing it in a column. An example is an extra column storing the *net sales value* of an order instead of aggregating it from the order line items each time it is requested.

In analytical systems, the types *pre-joined tables* and *report tables* are common. In pre-joined tables, as the name says, two or more tables are stored in their joined form, omitting redundant columns. Report tables are tables that already represent the report based on the data that is otherwise stored in several tables and can only be retrieved via complex SQL statement. Thus, the report can be built using simple SQL queries. The star and snowflake schemas are members of this category. In our study, we particularly focus on the denormalization type pre-joined tables under mixed workload conditions.

According to Martyn [23], the denormalization found in the star schema in the data warehouse is acceptable because of the read-only character of the data warehouse and the opportunity of addressing potential update anomalies during the ETL process. Kimball expresses a similar opinion, writing that the “use of normalized modeling in the data warehouse presentation area defeats the whole purpose of data warehousing, namely, intuitive and high-performance retrieval of data.” [20] In contrast, Date [7], when discussing relational models in general which applies to OLTP as well as OLAP schemas, views denormalization in a critical fashion and believes “that anything less than a fully normalized design is strongly contraindicated” and that denormalization should only be used as a last resort if all other strategies to improve performance fail.

## 2.6 Related Work in Optimizing the Database Design for OLTP and OLAP

Structures for optimized query performance like indexes, views, or precomputed aggregates introduce redundancy, which adds overhead to the insertion of new data, updates, and deletes [22]. They are relativized with a growing share of read access or the increasing size of tables while access patterns, e.g. queried time windows, are constant. A variety of prototypes and tools exist that propose configurations for indexes and materialized views, for example, AutoAdmin for Microsoft SQL Server [5], or to automate partitioning, see Autopart for large

scientific databases [27]. Zilio et al. [33] introduced DB2 Design Advisor, which additionally takes into account clustering and partitioning. All optimization approaches are, however, focused on either OLTP or OLAP workloads. Furthermore, they utilize only structures of physical database design, while a difference between OLTP and OLAP data schemas can also be found on the logical level of database design, that is, normalization. Our case study provides insights on the performance of different database schemas with respect to normalization under a mixed workload.

The development of ODSs shows how a mixed workload can be handled by one system. There are four types of ODS (cf. [17]). Three of them are copies of transactional data and can be categorized according to data freshness. The fourth type of ODS additionally includes strategic information. Data created from a report in the data warehouse environment is transported back into the ODS [14]. The different types of users producing the mixed workload are characterized by Inmon as “farmers” and “explorers” [16]. Whereas farmers repeat the same task over and over again and exactly know what they are looking for, i.e. benefit from clearly structured data like it is the case in OLTP, explorers exhibit unpredictable behavior. They skim through a large data set to search for patterns and relationships similar to OLAP workloads.

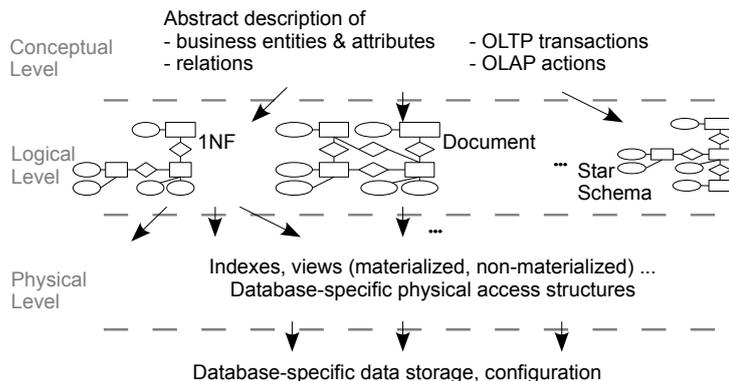
The reason why these diverse loads could be integrated within the ODS as one system are that farmers and explorers operate on only a limited amount of data copied into the ODS to produce reports concerned only with a short period of time, e.g. totals sales of the day or the number of new customers. Our focus, however, lies on systems which contain the entire operational data set of a company for analytics and not just a snapshot.

### 3 Database Design Variants Focusing on Normalization

In this section, we exemplify database design variation in the case of normalization. Database systems expose a very individual behavior, which heavily depends on the specific queries, underlying data schema, and physical optimizations. For a hybrid workload of OLTP and OLAP-like queries the task of choosing an optimal database design and optimizations according to a specific mix of queries becomes vital due to the wide range of data access behavior of the operations in a combined OLTP and OLAP system.

Figure 2 gives an overview of the levels relevant in database design. The top layer represents the abstract or conceptual definition of involved entities, attributes and their relations. OLTP and OLAP queries are defined based on these definitions and should be oblivious to changes of the database design on the lower layers. The lower two layers, logical and physical, show the possible variations of the database design. Normalization as a transformation of the tables resides in the logical layer.

The logical layer in Figure 2 depicts the schema variations applied in our case study. 1NF is the extracted part of a data schema of a real enterprise system, which is modeled in CBTR. Document and star schema are its variations to



**Fig. 2.** Data Model Variation Levels

analyze the impact of normalization under varying workloads and in different types of databases. These will be introduced in Section 3.2.

### 3.1 The Base Schema

Our starting point is an original database schema taken from a real-world OLTP database, modeled in CBTR [3]. Figure 3 depicts the database schema that is the basis for an order-to-cash scenario. This scenario covers sales order processing, deliveries, billing and payment processing as the related functions of accounting.

18 tables support this scenario, seven of them containing master data and 11 tables for transactional data. Master data is visualized in gray shading and transactional data is shown with a black header. Primary keys of tables are marked in bold face. Foreign keys are displayed by additional connections that start at the foreign key and point to the table and attribute that is referenced. In this schema only the most important attributes and relationships are illustrated for easier understanding. The complete schema contains 2316 columns that are distributed over the shown tables. The columns are not evenly distributed. Thus, table widths vary between 5 and a maximum of 327 columns.

The mapping of the conceptual design to the tables of this schema is not trivial. Business partner data is represented by three customer master tables and product data respectively by three product master tables. The order entity is split up into sales header, sales item, sales conditions, and sales partner. The delivery, billing, and accounting information entities are each split up in header and item tables respectively.

This schema is optimal for the OLTP operations taking place in the order-to-cash scenario, i.e., order entry, delivery scheduling, billing, and payment recording. Only a small set of transactional tables have to be touched during any of these actions and no joins are needed. However, the access pattern looks completely different for the OLAP operations. Customer payment behavior, order processing time, and order completion access data from several transactional

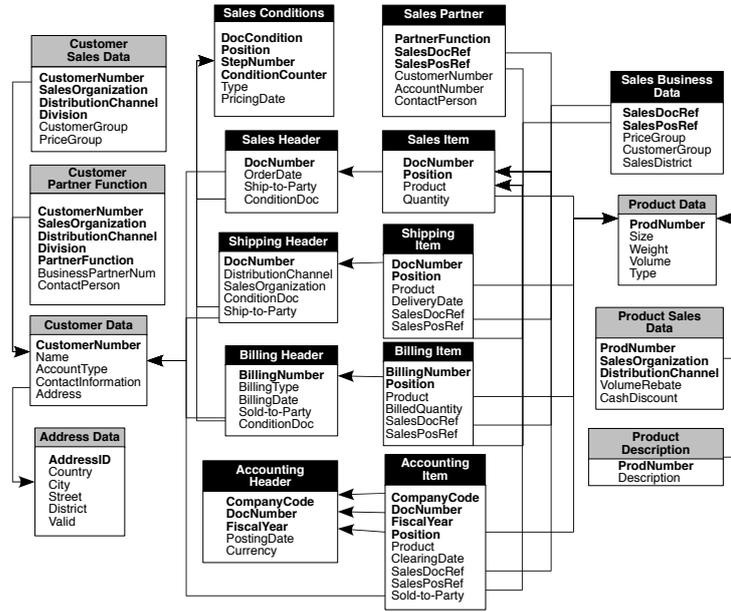


Fig. 3. Original OLTP Schema

tables and joining over four header and item tables is common. These joins are particularly problematic since the header and item tables have the largest cardinality. For the measurements, the OLTP and OLAP actions as defined in the CBTR benchmark are used. Table 1 gives an overview of the transaction profiles for the benchmark actions. We refer to [3] for the detailed profiles and SQL statements.

Table 1: Overview of CBTR Transaction Profiles and SQL Statements

Type	Action	Profile/SQL Statement
wOLTP	Sales Order	Header: Select [Cust. Data, Sales Data, Partner Function, Address Data], Insert [Sales Header]; Per Item: Select [Product Data, Description, Sales Data], Insert [Sales Item, Business Data, Business Partner]
	Shipping	Header: Insert [Shipping Header]; Per Item: Select [Sales Item], Insert [Shipping Item]
	Billing	Header: Insert [Billing & Accounting Header, Sales Conditions]; Per Item: Select [Sales & Shipping Item], Insert [Billing & Accounting Item]
	Payment	Select [Accounting Header]; Per Item: Update [Accounting Item]
rOLTP	Sales Order by Key	<b>SELECT</b> [...] <b>FROM</b> SalesHeader, SalesItem <b>WHERE</b> OrderID = @DocNum <b>AND</b> [...];

Table 1: (continued)

Type	Action	Profile/SQL Statement
	Sales Order by Period	<b>SELECT</b> [...] <b>FROM</b> SalesHeader <b>WHERE</b> CustomerID = @CustomerID <b>AND</b> (OrderDate <b>BETWEEN</b> ‘01.10.2011’ <b>AND</b> ‘31.10.2011’);
	Open Items	<b>SELECT</b> [...] <b>FROM</b> AccountingItem, AccountingHeader <b>WHERE</b> [...] <b>AND</b> AccountType = “Debitor” <b>AND</b> ClearingDate= “” <b>ORDER BY</b> CustID, AccID;
	Customer Details	<b>SELECT</b> [...] <b>FROM</b> CustomerData, AddressData <b>WHERE</b> [...];
	Product Details	<b>SELECT</b> [...] <b>FROM</b> ProductData, ProductDescription <b>WHERE</b> [...];
OLAP	Daily Flash	<b>SELECT</b> [...] <b>SUM</b> (Quantity) <b>FROM</b> SalesHeader, SalesItem <b>WHERE</b> [...] <b>GROUP BY</b> [...] <b>ORDER BY</b> [...];
	Order Processing Time	<b>SELECT</b> [...], <b>AVG</b> (DATEDIFF(T.DeliveryDate, T.OrderDate)) <b>AS</b> Days <b>FROM</b> ( <b>SELECT</b> <b>DISTINCT</b> [...] <b>FROM</b> SalesHeader, SalesItem, ShippingHeader, ShippingItem <b>WHERE</b> OrderDate <b>BETWEEN</b> “01.07.2011” <b>AND</b> “30.09.2011”, [...])T <b>GROUP BY</b> [...] <b>ORDER BY</b> Days <b>DESC</b> ;
	Order Delivery Fulfillment	<b>SELECT</b> [...] <b>SUM</b> (DeliveredQuantity), ( <b>SELECT</b> <b>SUM</b> (OrderQuantity) <b>FROM</b> SalesItem, SalesHeader <b>WHERE</b> [...]) <b>AS</b> Expected <b>FROM</b> SalesHeader <b>AS</b> sdh, SalesItem, ShippingHeader <b>AS</b> sh, ShippingItem <b>WHERE</b> sh.DeliveryDate j= sdh.DeliveryDate <b>AND</b> [...] <b>GROUP BY</b> [...] <b>ORDER BY</b> DeliveredQuantity <b>DESC</b> ;
	Days Sales Outstanding	<b>SELECT</b> [...], (1 - <b>SUM</b> (Amount) / ( <b>SELECT</b> <b>SUM</b> (NetValue + TaxAmount) <b>FROM</b> BillingHeader <b>WHERE</b> [...]))*91 <b>AS</b> DSO <b>FROM</b> AccountingItem, BillingHeader <b>WHERE</b> ClearingDate <> “” <b>AND</b> AccountType = “Debitor” <b>AND</b> [...] <b>GROUP BY</b> Currency <b>ORDER BY</b> DSO <b>DESC</b> ;

### 3.2 Normalization Variants of the 1NF Schema

Figures 4 and 5 illustrate schema variations to optimize the performance of OLAP operations by avoiding the joins. Through pre-joining the tables, that is denormalization, the level of redundancy within the data set is increased. This adds effort for the OLTP operations that insert data.

To assess the impact of normalization for the specific OLAP operations the tables are denormalized as follows: In Figure 4 header and item tables are joined, because header and item information is mostly requested in combination by the OLAP operations. From a business perspective the data about the line items of an order is never queried without accessing the order header. Even in contexts where *top n sold products* are analyzed the header information is necessary to provide the time dimension, as such queries always reference a certain time frame, such as last quarter. This schema variant will be called *Document* in the following because it provides the join between header and item information as would be reflected in a complete order document.

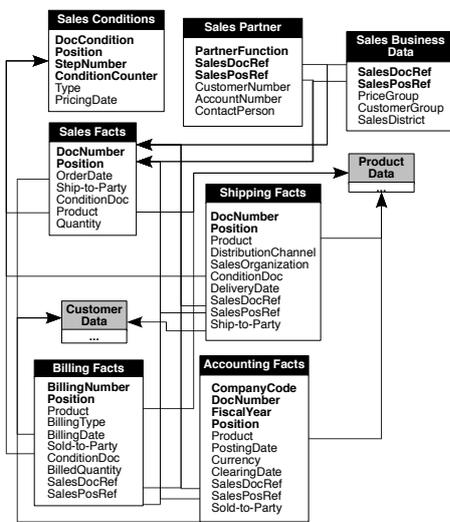


Fig. 4. Header-Item Join

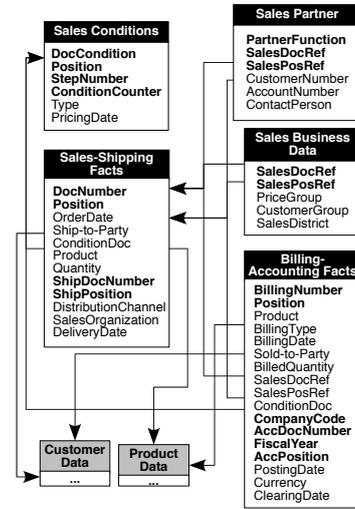


Fig. 5. Invoice-Accounting and Order-Shipping Fact Tables

The schema variant in Figure 5 further increases the level of denormalization by joining together the sales and delivery information on one hand and the billing and accounting entities on the other hand. Thus, joins between transactional data tables can be completely avoided in the OLAP operations. This schema variant resembles a star schema with shared dimension tables and will be called *Denormalized* in the following. It represents the maximum level of denormalization that fits the set of OLAP operations given in CBTR. No changes are applied to the master data, therefore only stubs are depicted that represent the same master data tables and relations between them as shown in Figure 3.

According to the changes in the tables the actions needed to be adapted. This means for the OLAP queries and read-only OLTP transactions mainly exchanging the tables to be accessed and removing joins for the Document and Denormalized schemas. Insert OLTP transactions become bulkier with a higher degree of denormalization. For example, new sales order transactions insert one sales order header line and one or more sales order item lines into the tables depending on the number of items ordered. In the Document schema, the single header insert is removed, instead the header data is inserted together with each line item, redundantly storing the information of the sales order header. Besides increased storage space consumption additional complexity is introduced to updates of header information as multiple line item tuples have to be updated.

## 4 Case Study

In this section we present the setup and the results of our normalization case study.

### 4.1 Setup

We analyzed the behavior of four databases of the following configuration:

1. in-memory column-oriented data storage
2. disk-based column-oriented storage (System A)
3. disk-based column-oriented storage (System B)
4. disk-based row-oriented storage

For our workload we simulated 100 concurrently running clients that fire requests to the database in parallel. Each client upon receiving the answer for its request, immediately fires the next query. Thus, clients do not simulate any thinking time between requests. The OLTP and OLAP workload mix is controlled via the share of client types within the set of the 100 clients. Clients can be of type OLAP and OLTP, with OLAP clients sending only OLAP requests to the database, OLTP clients behave respectively. OLTP clients execute either read-only (rOLTP) or modifying statements (wOLTP) each with a share of approximately 50%. For example, in the workload  $(\text{OLTP}, \text{OLAP}) = (80, 20)$  the share of workload types is  $(\text{wOLTP}, \text{rOLTP}, \text{OLAP}) = (40\%, 40\%, 20\%)$ .

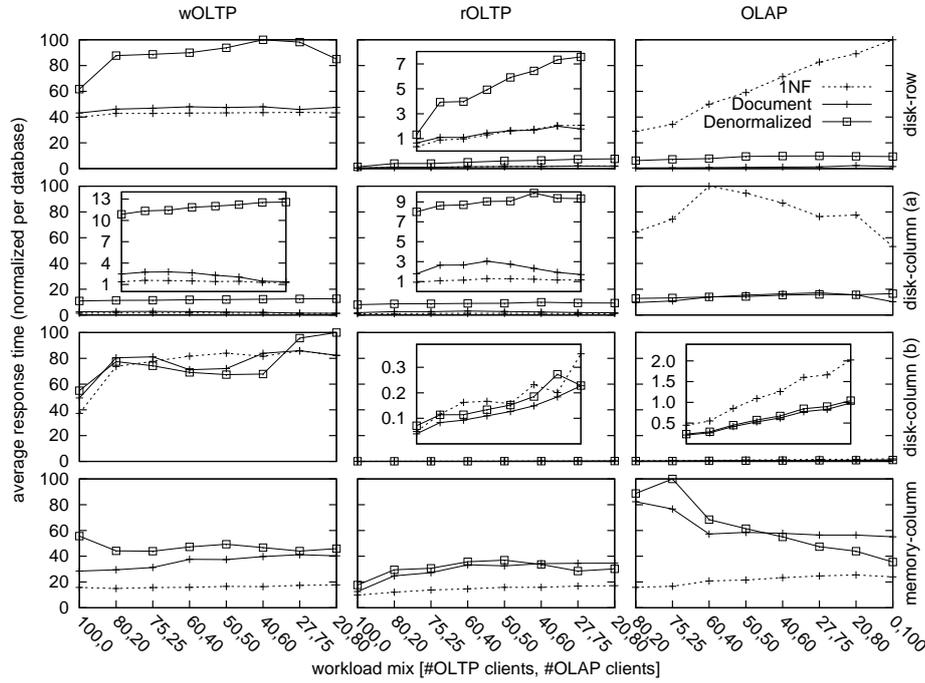
For each database, nine workload configurations  $(\text{OLTP}, \text{OLAP}) \in ((100, 0), (80, 20), (75, 25), (60, 40), (50, 50), (40, 60), \dots, (0, 100))$  were tested for the above introduced three schema variants (1NF, Document, and Denormalized).

### 4.2 Results

The results of our experiments are shown in Figure 6. Each row depicts the results for one database. The workload configuration is shown on the x-axis<sup>3</sup>, and the schema variants are depicted as series. The average response time is depicted for each workload type on a scale, which is normalized per database for easier comparison of the schemas for one database. For better comparison of those schema variants with very fast response times, some graphs are magnified.

From the tests we can see that independent of the the database the performance of some schemas varies when the workload mix is changed, particularly in the cases of the read-only statements. Especially in the OLAP cases, increasing response times are an indicator for the system becoming clogged with too many requests touching large datasets and being processed independent of each other's context. In the memory-column case we can observe that response times are decreasing with increasing OLAP workload for the Denormalized schema. This shows the impact of reusing intermediate results. As requests concentrate

<sup>3</sup> For better readability in some graphs the axis description is omitted. The x-axes are the same for all graphs of a workload type and the y-axes are the same for all non-magnified graphs. Thus, axis descriptions on the bottom/left are applicable.



**Fig. 6.** Database Schema Performance for Varying Database Implementations and Workloads

on few tables in the Denormalized schema, the probability of reusing existing results increases.

Depending on the workload, the schema which allows top performance differs. As was expected, for example, for the OLAP cases, the pre-joined schemas are advantageous, except in the case of the memory-column database, which is the only system tested that is being developed for a mixed workload. Here we can see, that 1NF as the schema with the highest degree of normalization tested, is the top candidate for all workload types. Thus, Date’s view on normalization mentioned in Section 2.5 is enforced, which emphasizes denormalization as a strategy to be used for performance improvement as the last resort.

Depending on the database, different schemas allow top performance. In the disk-row case, we can see that for OLTP 1NF offers the best query performance closely followed by the Document schema. However, for OLAP the 1NF schema performs increasingly worse than Document or Denormalized with greater numbers of OLAP clients running concurrently. The conclusion for this database would be to use the Document schema as it achieves the best response times for all workload types. Taking a look at the results for the disk-column (a) database, the same conclusion in favor of the Document schema can be drawn. The average response times of the schemas for the disk-column (a) database are

lowest for 1NF for OLTP, but clearly not for OLAP. Since 1NF is in favor for OLTP, but not in the OLAP case, a decision in favor of the Document schema would be made, which is close to the optimum for OLTP, is the optimum in the OLAP case, and performs much better for the OLAP queries compared with 1NF. For the disk-column (b) database making a decision between Document and Denormalized is hard without further knowledge of priorities and workload specifications. Only in the case of the memory-column database, one schema, i.e., 1NF outperforms all other variants regardless of the workload type.

## 5 Conclusion

The increasing need of up-to-date and detailed data for enterprise analytical processing and strategic decision making counteracts the current separation of OLTP and OLAP into two separate domains. Drawing them back into one domain, however, requires new system architectures that are able to mediate between the conflicting demands of OLTP and OLAP. The resulting new domain of a combined workload requires benchmarking activities to evolve in order to accompany and drive the further development of database systems and they need more effort than just extending existing benchmarks, as workload mix as a new parameter is brought into play.

Next, the question of database design has to be answered. Efforts so far are focused on OLTP or OLAP and mainly on optimizing physical database design using indexes, partitioning, clustering or views. Another aspect, however, is added when uniting OLTP and OLAP, which is their difference on the logical database design layer. Normalization is the main distinguishing factor when comparing typical OLTP and OLAP database designs. To our knowledge no study exists on the variation of database designs on the logical level under a mixed OLTP and OLAP workload. Therefore, we measured the impact of logical database design with the focus on normalization on various workload mixes. We analyzed four databases: one mainly used in the OLTP domain, two from the OLAP domain and one designed to handle mixed workloads and found that only for the database developed for a mixed workload one schema is optimal for all workload types under all tested workload mixes, which is the one with the highest degree of normalization. Thus, denormalization as the highly disputed weapon of last resort for performance improvement is not required.

Normalization is only one dimension of logical schema variation as mentioned in Section 3. More redundant data can be observed in real enterprise data models, i.e. materialized aggregates. In future work the impact of removing these and computing them on-the-fly will be analyzed. Further future work includes the consideration of basic service levels for operations, e.g., *99% of all billing insertion OLTP operations should have a response time of less than 5ms*. With a set of such rules and priorities the database design could be adapted automatically.

## Acknowledgements

The authors would like to thank Franziska Häger for her contribution on improving the benchmark definitions and her support in running the database tests.

## References

1. M. Abolhassani. *Managing Information Technology in a Global Environment (IRMA-2001)*, chapter Business Objects and Enterprise Applications, pages 240–242. Idea Group Publishing, Hershey, London, January 2001.
2. D. B. Bock and J. F. Schrage. Denormalization Guidelines for Base and Transaction Tables. *ACM SIGCSE Bulletin*, 34(4):129–133, December 2002.
3. A. Bog, H. Plattner, and A. Zeier. A mixed transaction processing and operational reporting benchmark. *Information Systems Frontiers*, 13:321–335, October 2010. Springer Netherlands, <http://dx.doi.org/10.1007/s10796-010-9283-8>.
4. A. Bog, K. Sachs, and A. Zeier. Benchmarking database design for mixed oltp and olap workloads. In *Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering, ICPE '11*, pages 417–418, New York, NY, USA, 2011. ACM.
5. S. Chaudhuri and V. R. Narasayya. An efficient cost-driven index selection tool for microsoft sql server. In *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB '97*, pages 146–155, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
6. E. F. Codd. Normalized data base structure: a brief tutorial. In *Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control, SIGFIDET '71*, pages 1–17, New York, NY, USA, 1971. ACM.
7. C. J. Date. *Database in Depth: Relational Theory for Practitioners*. O'Reilly, 2005.
8. J. Dittrich and A. Jindal. Towards a One Size Fits All Database Architecture. In *Outrageous Ideas and Vision track, 5th Biennial Conference on Innovative Data Systems Research (CIDR 11)*, January 9–12 2011.
9. C. D. French. “One size fits all” database architectures do not work for DSS. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 449–450, New York, NY, USA, 1995. ACM.
10. C. D. French. Teaching an OLTP Database Kernel Advanced Data Warehousing Techniques. In *ICDE '97: Proceedings of the 13th International Conference on Data Engineering*, pages 194–198, Washington, DC, USA, 1997. IEEE Computer Society.
11. F. Funke, A. Kemper, and T. Neumann. Benchmarking hybrid oltp&olap database systems. In T. Härder, W. Lehner, B. Mitschang, H. Schöning, and H. Schwarz, editors, *14. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, volume 180 of *LNI*, pages 390–409. GI, 2011.
12. M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden. Hyrise: a main memory hybrid storage engine. *Proceedings of the VLDB Endowment*, 4(2):105–116, November 2010.
13. D. J. Haderle. Database Role in Information Systems: The Evolution of Database Technology and its Impact on Enterprise Information Systems. In A. Blaser, editor, *IBM Symposium: Database Systems of the 90s*, volume 466 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 1990.
14. C. Imhoff. A New Class of Operational Data Store. Intelligent Solutions. *Information Management Magazine*, July 2000.
15. W. H. Inmon. The Operational Data Store. *InfoDB*, pages 21–24, February 1995.
16. W. H. Inmon. The Operational Data Store. Designing the Operational Data Store. *Information Management Magazine*, July 1998.
17. W. H. Inmon. ODS Types. Information Management: Charting the Course. *Information Management Magazine*, January 2000.

18. A. Kemper and T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In S. Abiteboul, K. Böhm, C. Koch, and K.-L. Tan, editors, *IEEE 27th International Conference on Data Engineering (ICDE)*, pages 195–206, Hannover, April 2011. IEEE Computer Society.
19. W. Kent. A simple guide to five normal forms in relational database theory. *Communications of the ACM*, 26(2):120–125, February 1983.
20. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, 2002.
21. J. Krueger, C. Tinnefeld, M. Grund, A. Zeier, and H. Plattner. A Case for Online Mixed Workload Processing. In *3rd International Workshop on Testing Database System*, 2010.
22. S. S. Lightstone, T. J. Teorey, and T. Nadeau. *Physical Database Design: The Database Professional's Guide to Exploiting Indexes, Views, Storage, and more*. Morgan Kaufmann Publishers, 2007.
23. T. Martyn. Reconsidering Multi-Dimensional schemas. *ACM SIGMOD Record*, 33(1):83–88, March 2004.
24. G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, Apr. 1965.
25. G. Mullins. *Database Administration: The Complete Guide to Practices and Procedures*. Addison-Wesley, 2002.
26. R. Nambiar and M. Poess. Transaction performance vs. moores law: A trend analysis. In R. Nambiar and M. Poess, editors, *Performance Evaluation, Measurement and Characterization of Complex Systems*, volume 6417 of *Lecture Notes in Computer Science*, pages 110–120. Springer Berlin / Heidelberg, 2011.
27. E. Papadomanolakis and A. Ailamaki. Autopart: Automating schema design for large scientific databases using data partitioning. In *In Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 383–392. IEEE Computer Society, 2004.
28. H. 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*, SIGMOD '09, pages 1–2, New York, NY, USA, 2009. ACM.
29. U. Röhm. *Database Technologies: Concepts, Methodologies, Tools, and Applications*, chapter OLAP with a Database Cluster, pages 829–46. Hershey: IGI Global, 2009.
30. TPC. TPC-W, Version 1.8. Technical report, Transaction Processing Performance Council, February 2002.
31. TPC. TPC Benchmark C, Standard Specification, Revision 5.9. Technical report, Transaction Processing Performance Council, June 2007.
32. TPC. TPC Benchmark H (Decision Support), Standard Specification, Revision 2.7.0. Technical report, Transaction Processing Performance Council, February 2008.
33. D. C. Zilio, J. Rao, S. Lightstone, G. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. Db2 design advisor: integrated automatic physical database design. In *Proceedings of the 30th international conference on Very large data bases - Volume 30*, VLDB '04, pages 1087–1097. VLDB Endowment, 2004.