In-Memory Data Management Research

Anja Bog

Benchmarking Transaction and Analytical Processing Systems

The Creation of a Mixed Workload Benchmark and its Application

Springer

In-Memory Data Management Research

Series Editor:
Prof. Dr. Dr. h.c. Hasso Plattner
Hasso Plattner Institute
Potsdam, Germany

For further volumes:
http://www.springer.com/series/11642

Anja Bog

Benchmarking Transaction and Analytical Processing Systems

The Creation of a Mixed Workload Benchmark and its Application

Springer

Anja Bog
Hasso-Plattner-Institute
University of Potsdam
Potsdam
Germany

Dissertation at the University of Potsdam, Hasso Plattner Institute for IT-Systems Engineering

ISBN 978-3-642-38069-3 ISBN 978-3-642-38070-9 (eBook)
DOI 10.1007/978-3-642-38070-9
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013943472

© Springer-Verlag Berlin Heidelberg 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Acknowledgements

I would like to express my deep gratitude to Professor Hasso Plattner as my primary supervisor for sharing his inspiring views and providing me with guidance as well as valuable and constructive suggestions during the planning and development of this research work. With his passion and engagement, he created a unique setting to spark and foster new ideas and pursue targets off the beaten track.

The advice and related work provided by Professor Felix Naumann have been a great help refining my research goals. I would also like to thank my reviewers Professor Marco Vieira and Professor Alejandro Buchmann for their feedback and very useful critiques of this research work.

I want to offer special thanks to Vishal Sikka and Cafer Tosun from SAP for their support of my research without which this work would not have been possible, providing professional feedback, and connecting me with the right people to advance my work. Thanks to Kai Sachs from SAP, who helped me to stay on track through numerous discussions and encouraged me to always look forward, rework my rejected papers, and try again.

I wish to thank Franziska Häger for writing her master thesis in the context of mixed workload benchmarking in close relation to my work and thus offering her perspective on the topic inspiring valuable discussions. Many thanks go to Ralph Kühne, Massimiliano Marcon, and Markus Steiner, who spent countless hours reading preliminary versions of this thesis, listening to me talk about my research, and giving feedback. My fellow researchers at the Enterprise Platform and Integration Concepts group also deserve a big thanks for their unbiased feedback and fruitful conversations.

Last but by no means least, I like to thank my family for supporting years of study and the opportunity to start this research project. Many thanks to you and Peter Weigt for the encouragement, patience, and support you have given me throughout the time I spent on this work.

Potsdam, Germany — Anja Bog
November 2012

Contents

List of Figures

List of Tables

Chapter 1
Introduction

A wider usage of computers in the business setting started in the 1960s [185, Chap. 1]. The first task companies used computers for was the automation of simple routine tasks like payroll amount calculation. More business areas such as financial accounting, order entry, and billing soon followed. At that time, many of these tasks were semi-automatic with users entering information interactively and the system completing the task at a later point in time.

As hardware and software advanced, more and more business users started to interact with the electronic business data to process business transactions. Rockart [175] states that by the end of the 1970s, the more effective and increasingly efficient technology resulted in systems that were oriented at the end user, like online data inquiry and analysis. These were rapidly becoming widespread in major organizations [175] and system interaction with instantaneous processing of business data emerged. Instead of collecting all business transactions and processing them as a bundle in the evening or at night (batch processing), requests were now processed immediately and results were provided directly. The term Online Transaction Processing (OLTP) refers to this instant interaction with the business systems.

> Today, online transaction processing powers the record keeping systems that drive today's commerce, services, and government. – Lindsay 2008 [130]

As quoted, OLTP systems are the backbone of today's information systems, supporting the daily operations of companies. All incoming business requests, such as the creation of sales orders, requests for information of customers, suppliers, and other business partners, incoming payments, as well as internal production processes are recorded, monitored, and processed using these systems. For OLTP systems performance is a key element. If the OLTP system was not able to keep up with business operations, results may not be readily available and incoming order may be delayed, resulting in loss of profit.

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_1,
© Springer-Verlag Berlin Heidelberg 2014

> Business enterprises prosper or fail according to the sophistication and speed of their information systems, and their ability to analyze and synthesize information using those systems. – Codd et al. 1993 [43]

Codd et al. [43] coined the term online analytical processing (OLAP) during a period in which a lot of discussion took place around the coexistence of transaction processing systems and systems to support decision making. They define OLAP as the "dynamic enterprise analysis required to create, manipulate, animate, and synthesize information." From a technical perspective, "[t]his includes the ability to discern new or unanticipated relationships between variables, the ability to identify the parameters necessary to handle large amounts of data, to create an unlimited number of dimensions (consolidation paths), and to specify cross-dimensional conditions and expressions." [43]

According to Codd et al. [43] the success of an enterprise is directly linked to the quality, efficiency, effectiveness and pervasiveness of its OLAP capability and the number of individuals within an enterprise, that have a need to perform more sophisticated analysis is growing. However, the tools developed at that time lacked the ability to analyze data in multiple dimensions, which is the feature that is the most useful to enterprise analysts [225].

Corporate data is growing consistently and rapidly and the need for sophisticated analyses with it. Since its introduction in the 1990s realizing the need and importance for advanced enterprise analytics, the work around OLAP has been established as an own application area next to transaction processing and its applications. Although the data that is central to both application areas is the same, i.e., the business data created during the daily operations of an enterprise, applications of both areas rely on their own systems that are optimized for the respective requirements of these applications.

Implications of Separating Analytics from Transaction Processing

The separation of the two areas obviously offers the advantage that analytical processing does not affect the performance of transaction processing. Furthermore, the specialized systems for OLAP and OLTP allow optimizations specific for their applications to achieve maximum performance. While normalization techniques are popular in OLTP database design [30], traditional OLAP systems rely on summary data and pre-computed aggregates to speed up data analysis in multiple dimensions and hierarchies [229]. Deciding what to prepare and how much to pre-compute is difficult because it can result in an unwanted explosion of storage space necessary to store the data [188]. Consequently, the data present in the OLAP system contains a selected subset of the transactional data identified as necessary for strategic analysis and prepared for fast reporting. The up-front decision which data to choose and prepare limits the flexibility of reporting at a later point in time. "Pre-aggregation is fast, but inflexible" [60] and works best in environments where the user's options to define new queries are limited. New reporting requirements based on data not

present or not prepared in the analytical environment have to wait until the respective data is prepared and the needed structures are established.

Data preparation based on previously defined reporting requirements entails that data in the analytical environment is aggregated or summarized. Thus, data is available on a lower level of detail compared to the former "raw" transactional data. The advantage is that storage requirements are reduced. The disadvantage is that reporting queries can only rely on the pre-defined data. As a result, queries that require information that is more detailed cannot be answered.

Although differently optimized, the data is stored redundantly in the different systems and has to be synchronized. Data is extracted from one or several source systems, transformed into the OLAP-optimized structures, enriched (e.g., with external data), cleansed, and loaded into the OLAP system. This Extraction, Transformation and Load (ETL) process that keeps the OLAP data synchronized with the OLTP data is the most expensive and time-consuming part of setting up an analytical system [120, Chap. 11] and 55 % of the running costs of the analytical system go into refreshing the data from the operational environment [107]. The process of refreshing the data is launched at intervals. These intervals may vary from minutes to days or weeks, meaning that reports, which run in the analytical system, have to necessarily be based on data that is no longer up-to-date.

An advantage of keeping the OLAP system separate from the OLTP system is the integration aspect. Larger companies may have multiple OLTP systems and use the OLAP system for consolidation and strategic analysis of data across several of their systems or they may even include data from external sources.

Why Unite OLTP and OLAP Systems Now?

The above question can be tackled from two sides. The first side is the hardware perspective. The advance of technology enables the unification. The second is the perspective of the applications that benefit from the unification. As companies are continuously adapting to market requirements, their workloads are changing accordingly. Some of the changes in operational and analytical applications over the last decade are so fundamental that the validity of the separation of analytical processing from transaction processing needs to be reevaluated.

The results of a survey concerning the reach, trends and future directions of data warehousing targeted at 421 data managers and professionals conducted in 2011 [139], confirm the above statement to reconsider the current system setup of transactional and analytical processing. McKendrick [139] summarizes from the results of the survey that data needs to be available in real-time to be of value and a larger set of users, besides the traditional analysts, as well as critical business applications need to gain access to analytical functionality. Furthermore, 42 % of the respondents stated that their data warehousing platform is going to be replaced within 5 years to stay in competition on analytics and manage the growing data stores. With efficiency and cost cutting as the main goals, organizations feel the need to consolidate OLTP and OLAP applications onto a single platform. The survey

provides detailed statistics that 40 % of the respondents claim to be already using a single server for some instances of their OLTP and OLAP applications and 37 % are considering the consolidation of OLTP and OLAP applications on the same server in the future.

The Hardware Perspective

> [W]hat is economical to put on disk today will be economical to put in RAM in about 10 years. – Gray and Shenoy 2000 [75]

Hardware is developing rapidly. According to Gartner [70], in-memory data management approaches will enter the mainstream as cost and availability of memory intensive hardware platforms reach tipping points in 2012 and 2013. Harizopoulos et al. [90] stated in 2008, that many OLTP systems already fit or will soon fit into main memory. Keeping enterprise data completely in main memory allows direct optimization of access for this storage and casting aside the disk access strategies. Accessing data directly from main memory instead of having to pass it up from disk, results in a speed-up of operations as well as a simplification of architectures and applications. The reason is that the disk access itself can be dropped and caching as well as pre-fetching strategies so far used to optimize disk access can be discarded. Nonetheless, the value of disks for back up, logging and archiving remains unchanged.

Multi-core processing and multi-processor systems are another advance of hardware introduced in the 2000 decade. By now, they are widely used. The first step towards multi-core processing was the Intel Hyper-Threading Technology [136]. It lets a physical processor appear as two identical logical ones and allows for two tasks to be executed in parallel increasing processor utilization [48]. Koch [122] states that multi-core processing extends Moore's Law [145] into the future despite the fact that clock speed increase has stopped in 2003 [201]. Multi-core processing and multi-processor systems provide the basis for high-volume and massively parallel processing that is necessary for enterprise OLTP systems with their large number of users and also builds the foundation for OLAP systems to process large data volumes faster through parallelism.

Another important advance was the introduction of on-chip memory controllers. This advance strives against the data supply bottleneck between main memory and the increased number of processing units by reducing the memory latency and allowing the memory bandwidth to scale with the number of processing units [83]. HyperTransport [4] and QuickPath Interconnect [110] were introduced as serial point-to-point links for fast inter-processor and chipset communication. According to Loos et al. [133], the complex database designs so far used in analytics, for example, indexes and materialized views become obsolete as it becomes affordable to scan through the data all the time. The ability to remove those specialized structures that were previously necessary for fast OLAP further encourages the notion of bringing the data for analytical processing and transaction processing together.

The Workload Perspective

Nambiar and Poess [149] analyze Moore's law with respect to transaction processing systems and show that even on this higher level, involving not just the processor perspective, but the total system perspective, a close resemblance to Moore's predictions is achieved. Transaction processing 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. This study emphasizes the ability of current and future systems to meet today's and tomorrow's transaction processing requirements. On the contrary, OLTP systems are not growing exponentially like Moore's predictions because the real world entities, they record information about, do not scale with Moore's law [90]. Thus, the study underlines the potential of OLTP systems to handle the increased load created by the addition of OLAP given the fact that the need of materializing structures for OLAP disappears and the structures of the data sets underneath both workloads converge.

Applications that have further evolved in the past decades and which do not clearly fit into the OLTP or OLAP world are another aspect worth examination. Business application examples here are dunning and available-to-promise (ATP) Plattner [167]. These applications are running in the OLTP system, as they require up-to-date data. However, to compute the results they need to process large volumes of data, for example in the case of dunning, all or a set of customers of a company, their invoices and payments to determine which customers have overdue payments and previous actions to find out their dunning level. Workarounds have been introduced to reduce the impact such applications so far have on the currently used OLTP systems. Examples for these workarounds are running the applications during low system load times, e.g., at night, and/or splitting up the data set involved like scheduling dunning runs for selected groups of customers. ATP is a business function that computes the availability of products and their delivery due dates based on resource availability and provides a response immediately upon a customer inquiry. Here, scheduling the computation at a later point in time is not applicable. Instead, aggregates are pre-computed, materialized, and updated during transaction processing to reduce the response time of ATP checks in current systems. This pre-computation is very similar to the approach taken in current analytical systems to support reporting. Plattner [167] illustrates how dunning and ATP can be changed when leveraging the technological potential of a database system developed to serve transactions and analytics based on the same data store.

Krüger et al. [125] show in their analysis of enterprise applications and the resulting workloads that in OLTP systems more than 80 % and in OLAP systems more than 90 % of all queries are read-only. The results for the OLAP systems is in accordance with their expectations, but the high share of read-only queries with more than 20 % range selects in the OLTP systems has not been expected as traditional workload definitions assume a larger share of write access and simple queries with highly selective query terms [62]. Krüger et al. [125] conclude that this query distribution leads to the concept of using a read-optimized database that

supports the required amount of insert and update operations for both transactional and analytical systems.

Raden [174] states that analytical systems are becoming more proactive, real-time, and operational. Thus, analytical applications emerge whose characteristics diverge from the so far accepted "typical" characteristics of OLAP applications for which OLAP systems are optimized. OLAP systems are being adapted accordingly. Besides the fact that OLTP and OLAP systems are coming together in the sense that either side adopts structures and optimizations typical for the other side, the interest in hybrid OLTP and OLAP systems has also grown in the database research community spawning several prototypes, e.g., HyPer [117], OctopusDB [55], and HYRISE [80].

The recent developments in hardware, OLTP and OLAP workloads, their underlying systems, the growing interest in the research community, related research activities and product developments in the industry, in particular those introducing new database systems and prototypes, show the necessity for this work and strengthen its importance.

1.1 Problem Statement

With the reunification of OLTP and OLAP and the development of hybrid systems for this combined workload, the need for means to study and evaluate these combined systems arises. Benchmarks are the standard method to evaluate and compare database systems and support their development. They help to assess the performance and cost-performance ratio of existing database products for analytical and transactional processing. Due to the separation of OLTP and OLAP systems, existing benchmarks are only focused on either domain so far. With the unification of the two workloads and the rise of hybrid database systems, benchmarks to assess these systems are required as well. This thesis introduces a benchmark for hybrid OLTP and OLAP systems and the first research question is concerned with the development of this benchmark.

The implementation of a chosen database product in a productive environment is only the first step. Afterwards follow the tasks of configuring the database for the detailed requirements of the environment and optimizing the database design according to the specific access patterns of the business applications used by the company. The second research question addresses these tasks in the context of logical database design. Reuniting OLTP with OLAP affects the entire design of the database starting at the logical layer. Database schemas as part of the logical database design layer are optimized for one or the other workload. The new benchmark is applied in quantifying the impact of selected logical database design optimizations typically employed in OLTP and OLAP systems under a mixed workload. This evaluation serves as an application and validation of the proposed new benchmark.

Question 1: What is a suitable method to evaluate and compare hybrid systems?

A benchmark for hybrid OLTP and OLAP systems is an important pillar in the development of these systems. Benchmarks play an important role in the development of databases for business applications. On the one hand, companies are able to assess if functionality and the cost-performance ratio are a fit before deciding on a solution and implementing it. On the other hand, database vendors advance their products based on the requirements provided by database benchmarks that reflect the requirements of real companies.

- *What to use as the foundation for a new benchmark?* There are three alternatives to create a benchmark for mixed workloads. These are (i) combining two existing benchmarks, that is, merging an OLTP benchmark with an OLAP benchmark, (ii) extending an existing benchmark with the other workload, or (iii) creating a completely new benchmark. A new mixed workload benchmark must adhere to the workload characteristics of current enterprise systems to provide reliable and relevant results.
- *How to simulate hybrid workloads?* For a combined workload of OLTP and OLAP, workload mix as a new parameter has to be introduced. In a benchmark setting of either transactional or analytical processing, the workload that is simulated closely resembles a typical company. A hybrid setting has to comprise both analytical and transactional operations. Transactional and analytical processing, however, should not only be simulated in equal shares as different companies have different workload characteristics and varying the workload shares impacts resource usage and design decisions for the database. Thus, workload mix, as the variation of the transactional and analytical share in the entire workload, has to be introduced.

Question 2: What is the impact of database schema optimizations in a mixed OLTP and OLAP environment?

Database design of OLTP and OLAP systems so far has contradicting optimization goals. In addition, both workloads are subject to variation. According to day, week, month, or period-end reporting the OLAP workload can peak at certain times. The same is true for OLTP, for example, corresponding to seasonal peaks in customer orders. For such variations the automatic adjustment of the physical database design elements is already subject to research, which is expressed, e.g., in the creation or deletion of indexes according to current workload situations [3]. The focus of this research question lies in understanding the impact of logical database optimizations that need to be adjusted besides physical optimizations like indexes or views.

- *What is the impact of mixing OLAP with OLTP?* Before determining the impact of specific optimizations in logical database design, the behavior of databases when combining the two workloads has to be analyzed.

- *What are typical logical database optimizations?* The optimized database schemas of transactional and analytical processing represent the opposite ends of a scale. While transaction processing schemas are optimized for high throughput of write operations and highly selective read operations, analytical systems are optimized for company-wide information retrieval requirements reading large amounts of data. In a hybrid setting, the database schema has to be optimized according to the workload mix and prioritized operations.
- *What is the influence of database schemas that utilize different optimizations?* This influence on queries has to be determined to support making a decision for specific schema optimizations. A certain schema optimization can have a positive impact on the execution of a subset of queries of the entire mixed workload while impeding the performance of other queries. The quantification of this impact and its understanding is the first step towards making a decision on using a particular optimization in a mixed workload scenario.
- *What are additional factors that influence the performance of database design in a mixed OLTP and OLAP environment?* Workload mix is not the only factor influencing the decision of which database schema is best suited in a specific setting. The type of database, that is, its storage layout, is another important factor of influence. In the OLAP environment, both column and row-oriented physical storage layouts are common, while row-oriented storage is still the prevailing layout of databases underneath OLTP applications. The same schema optimizations on top of different storage layouts may have different impact on queries.

1.2 Contribution and Scope

According to the questions identified in the previous section, the contributions of this thesis are in the areas of benchmarking and understanding logical database design decisions in mixed OLTP and OLAP workload scenarios. The contributions and adjacent works have been published in international conferences, workshops, and journals.

Benchmarking

Existing benchmarks targeting relational databases in OLTP or OLAP environments are compared and discussed with regard to observations from current enterprise workloads. Enterprise workloads have evolved since the creation of the benchmarks widely in use today. Yet, the benchmark workloads remained the same since their specification with minor changes to requirements.

The review of existing benchmarks and observations of current enterprise systems provide the foundation for the definition of the composite benchmark for transactions and reporting (CBTR) proposed in this thesis. CBTR is based on a real database design and it provides realistic queries to simulate different workloads.

The scenario for CBTR, its conceptual entities for the definition of the data set and operations are completely specified and a set of analytical queries that can easily be extended is provided based on examinations of current OLAP systems. Workload mix has been introduced in CBTR as an additional parameter besides data set size and load (number of parallel users) to control the shares of the operational and analytical workload parts. The specific database schema used in CBTR is identical to what can be found in a widely used and current OLTP system.

Early concepts for CBTR have been published at the Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE) in [14]. This work includes the conceptual design of the benchmark, database schema, and the actions. A more detailed specification of the scenario, transactions and queries, and the unique properties of the new benchmark were published in the Information Systems Frontiers Journal [16].

A prototypical implementation of a tool chain to run the benchmark is also part of this thesis. This tool chain provides the driver to run benchmark tests, supports the evaluation of benchmark results, and supplies a monitor to run benchmarks interactively to test different workload mixes and immediately observe the impact on query and transaction performance. Using this tool chain, the impact of mixing OLTP and OLAP can be quantified for any database system based on realistic data structures and workloads.

Details about the implementation of the benchmark driver for CBTR have been published at the International Conference on Industrial Engineering and Engineering Management (IE&EM) in [15]. The paper proposes a configurable simulation framework that can be used to validate existing database systems and their ability to handle the transactional and analytical workload in parallel. The interactive performance monitor that utilizes the benchmark driver developed for CBTR has been demonstrated at the International Conference on Management of Data (SIGMOD) [19]. This performance monitor allows the direct observation of the impact of changing shares within the workload and to interactively assess behavioral characteristics of different database systems under changing mixed workload conditions.

Logical Database Optimizations in Mixed Workloads

Data models used in the past and currently used in transaction and analytical processing are reviewed and discussed to create a deeper understanding of where the optimizations within today's OLTP and OLAP systems originated.

On the base of the previous review of data models in transaction and analytical processing, logical database designs are analyzed. The focus lies on database schema optimizations employed in typical operational and analytical environments. This analysis contributes an overview of the key differentiators employed today that set apart OLTP and OLAP optimized database schemas. Based on the differentiators several variants of CBTR's database schema are defined that employ particular optimizations or avoid them.

The implemented benchmark driver as part of the previously mentioned tool chain is designed in such a modular way that only little effort is necessary to exchange the database schema and adapt the transactions and queries accordingly. The performance impact of different database schema variants is evaluated for selected database systems and workload mixes as part of this work. This evaluation serves as a validation of CBTR as assumptions about the results are met and can be explained with the aid of implementation knowledge of the databases. The results, furthermore, provide valuable insights into the behavior of existing database systems under mixed workload conditions and show that the variation of database schemas is an important factor that has to be considered when changing the workload.

Insights from the analysis and evaluation of logical database design optimizations have been published at the International Conferences on Performance Engineering (ICPE) [17] and Performance Evaluation and Benchmarking (TPC TC) [18]. In [17], initial insights on the impact of different database designs on query response times in various workload mixes have been published. Details on the database schema variants focusing on normalization are provided in [18]. These database schema variants have been analyzed regarding their potential for performance improvements under differing mixed OLTP and OLAP workloads. The results underline the expectation that different database types (disk-based and row-oriented vs. disk-based and column-oriented vs. in-memory and column-oriented) react differently and, thus, a method to test and configure a given database in line with the workload it is supposed to handle is highly relevant.

Further research conducted during the work on this thesis has targeted hybrid architectures and topics derived from the combination of transactional and analytical processing (see Appendix A).

The scope of this thesis comprises the development of a hybrid workload benchmark and the analysis and evaluation of logical database schema optimizations in mixed workload scenarios of enterprise OLTP and OLAP workloads. The proposal and definition of the benchmark in this thesis provides the foundation for a specification as compared to the specifications of established benchmarks such as the TPC benchmarks.

With the actual application of hybrid database systems in business processing, the database schema is bound to evolve to accommodate a mixed OLTP and OLAP workload. The definition of final database schemas optimized for specific mixes of OLTP and OLAP shares and specific database systems is beyond the scope of this thesis.

A benefit of current analytical systems especially for large enterprises is to provide analytics across several OLTP systems and potentially external data sources. The need for reporting across multiple data sources will not dissolve by introducing mixed OLTP and OLAP systems. It is an orthogonal research topic out of scope of this thesis. Research is already active in that area, e.g., in federated query processing efforts.

Other application areas for databases exist. These are, for example, bibliographic databases as digital collections of references to published literature [152], statistical

databases that store parameter data and measured data of experiments for statistical analyses [186], scientific databases that share characteristics of statistical databases but are subject to additional stages of data collection and analysis [187], search engines, or service repositories. However, the focus of this work lies on data management for business processing. Results achieved in this area are not necessarily applicable to the other areas named above. An evaluation of such applicability is out of scope for this work.

1.3 Thesis Organization

This thesis is divided into eight chapters in three parts. The first chapter has presented the motivation for mixed OLTP and OLAP workload benchmarking. It stakes out the scope of this work, which lies in creating a mixed workload benchmark as well as applying and evaluating this new benchmark in the context of understanding the impact of logical database design optimizations for mixed workloads.

Part I: Chapter 2 gives an overview of the most relevant and mainly used data models and database design alternatives within the areas of transactional and analytical processing. Examples of applications that cannot be classified as either typical transactional or analytical and further arguments driving the reunification of transactional and analytical processing are given in addition to related research initiatives that aim at building hybrid OLTP and OLAP data management systems. This chapter provides the basis for the later discussion and evaluation of logical database designs for hybrid OLTP and OLAP workloads. An overview of existing benchmarks in the area of data management and a discussion on their applicability in the context of mixed workload processing is offered in Chap. 3. It also points out key criteria that a new benchmark should fulfill and closes with a discussion on the existing benchmarks and their relation to real workloads.

Part II: Chapter 4 introduces CBTR along with its scenario, the underlying business process, the used data and database design, OLTP and OLAP queries. The concept of workload mix as an additional parameter is introduced in this chapter. The chapter closes with an examination of how the benchmark criteria from the previous chapter are reflected in CBTR. Chapter 5 focuses on the variation of database schemas within logical database design and proposes several variants that contain optimizations relevant for transactional and analytical processing. These schema variants are used in the following evaluation using CBTR to determine the impact of the optimizations in mixed workload scenarios and to gain experience how different database types behave under varying workloads.

Part III: Chapter 6 gives an overview of the tools that have been created around CBTR. It presents all steps that are needed for a complete benchmark run, starting from the preparation of a benchmark run, through running the benchmark, and

finishing with the evaluation of the results. Chapter 7 presents the application of CBTR in evaluating the impact of database schema optimizations under mixed workload scenarios and in different database systems. The tested traditional database systems behave as expected. This shows that the behavior modeled in CBTR matches experience from real systems and it provides a first validity check of CBTR.

Finally, Chap. 8 concludes this thesis with a discussion of the contributions and areas of future work.

Part I
Background of Transactional and Analytical Systems in Logical Database Design and Benchmarking

Chapter 2
Enterprise Data Management for Transaction and Analytical Processing

Enterprise data management has to support all business processes of a company from storing and providing data during daily operations, for example, sales, purchasing, and payroll accounting, to analyzing data for strategic decision making. Daily operations are supported by transaction processing systems. A transaction processing system contains applications that automate business activities [9, Chap. 1]. Sales order processing is a typical example for such an application. Analytical processing is provided by decision support systems. With their help, decision makers within companies gain insights into daily operations from which they defer information for making strategic decisions regarding the future of the company. Decision support systems have experienced an explosive growth in the 1990s [29].

According to Haderle [84], in the 1960s and 1970s the focus of commercial data management systems lay on batch and transaction processing environments. Batch processing means the definition of jobs that execute multiple operations within one request and that can run completely without user interaction. The response time requirements of batch jobs are flexible meaning that a response does not need to be produced immediately and they can be scheduled as resources become available, for example, during non-peak periods. This facilitates the management of the limited computing resources [9, Chap. 1]. Thus, batch processing was well suited to the goal of data management systems at that time, which was to manage concurrent read and write access to data, while minimizing resource utilization. Therefore, batch processing was the major application model and for some application areas, it is still today. In companies with a large customer base, dunning runs to determine the customers whose payments are late are an example for which batch jobs are scheduled in non-peak periods.

Transaction processing applications started to evolve at this time, too. Gray [72] defines a transaction as a mechanism that queries and transforms the state of the accessed data. A transaction is a fixed sequence of operations that have to be completely processed or not at all. It can contain several statements clustering these operations. The transaction concept emerged with four properties that incorporate essential features needed by business applications. These are atomicity, consistency, isolation, and durability (ACID [86]).

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_2,
© Springer-Verlag Berlin Heidelberg 2014

The isolation property means that events happening during the execution of a transaction are hidden from other transactions that are running concurrently. Gray et al. [76] originally introduced isolation in four *degrees of consistency*: (1) protecting other transactions from updates of a transaction, (2) additional protection from losing updates, (3) additional protection from reading incorrect data items, and (4) additional protection from reading incorrect relationships among data items. The other three properties were later defined in [73]. Atomicity ensures that all operations within a transaction are completed or none is. Consistency means that the data is left in a consistent state after the normal end of a transaction has been reached. Durability ensures that the results of a transaction upon successful completion are preserved by the system and that they survive any subsequent malfunction.

Early transaction processing systems already provided the functionality to interactively execute very short transactions, in a so-called on-line processing mode. This is where the term online transaction processing (OLTP) stems from. In online transaction processing, the changes and results are immediately visible as opposed to batch processing. In the beginning of the 1980s, the first commercial relational databases that enabled interactive processing appeared. Similar to OLTP, the term online analytical processing (OLAP) expresses that decision makers interactively work with the system.

This chapter presents the foundations that this thesis builds on. It describes the most relevant and widely used database design alternatives within the areas of transactional and analytical processing. The understanding of database design decisions taken in the past and the reasons behind facilitate building tomorrow's hybrid systems and the benchmarks to evaluate and compare them.

Section 2.1 starts with the description of the data models used within the databases for enterprise data management, explaining their characteristics. Figure 2.1 presents an overview of past and today's most commonly used data models and underlying database designs in OLTP and OLAP systems. Early data management was handled via flat files. To avoid implementing the access and data modification logic in each application, database management systems emerged as an abstraction layer that provides standard interfaces to manipulate data. A database management system (DBMS) is a collection of interrelated data and a set of programs to access that data. The term database refers to the collection of interrelated data, which contains information relevant to, e.g., an enterprise [191]. In OLAP systems dimensional modeling appeared along with the multidimensional and hybrid data models to better match the users needs. Concerning the relational data model, different database designs were introduced that are used in OLTP, OLAP, or in both kinds of systems.

The relational data model emerged as today's most prevalent data model used in business data processing in the areas of transaction processing as well as analytical processing. Therefore, the scope of the subsequent sections is narrowed down to the relational data model and the diversity of database designs within that area.

Section 2.2 covers database design alternatives and optimization opportunities in the area of relational databases. For later discussions, it includes relevant aspects

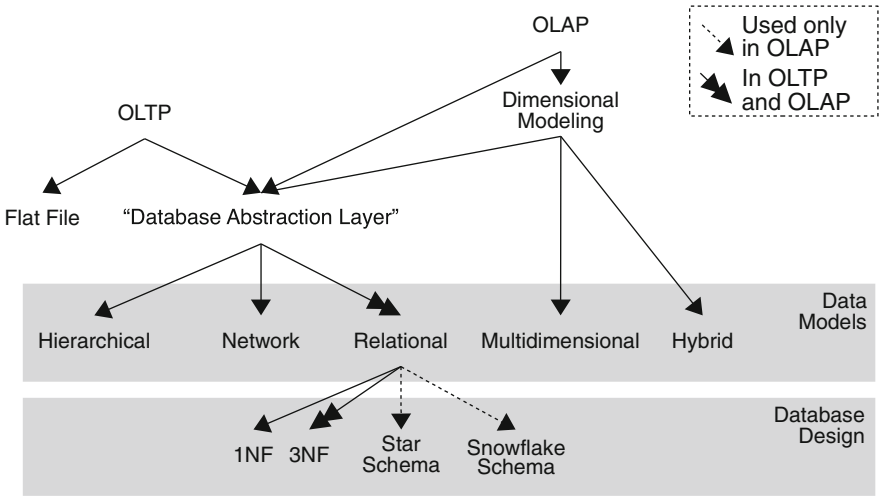

Fig. 2.1 Data management alternatives in OLTP and OLAP systems

from all database design levels: These are normalization on the logical level, optimization of access paths on the physical database design level, for example, indexes and views, and specific physical storage layouts used in database systems for transactional and analytical processing.

2.1 Data Models for Transaction and Analytical Processing

Codd [40] gives a definition of the term data model as the combination of

1. A collection of data structure types,
2. A set of operators and rules to retrieve or modify data from any part of the aforementioned data structure types, and
3. A number of general integrity rules that define consistent database states and changes of state.

Codd [40] emphasizes that all three parts of this definition of data models are equally important and argues that 2 and 3 are essential to understand how a structure based on a certain data model behaves. Without a defined set of operations any application using a structure has to be made aware of the structures internal workings regarding, for example, the conjunction of parts of the structure.

All examples in this section refer to a customer order scenario. Figure 2.2 presents the conceptual overview of the entities and relations for this scenario where customers order products and the respective products are being shipped to the customers. Each customer can place as many orders for as many products as needed

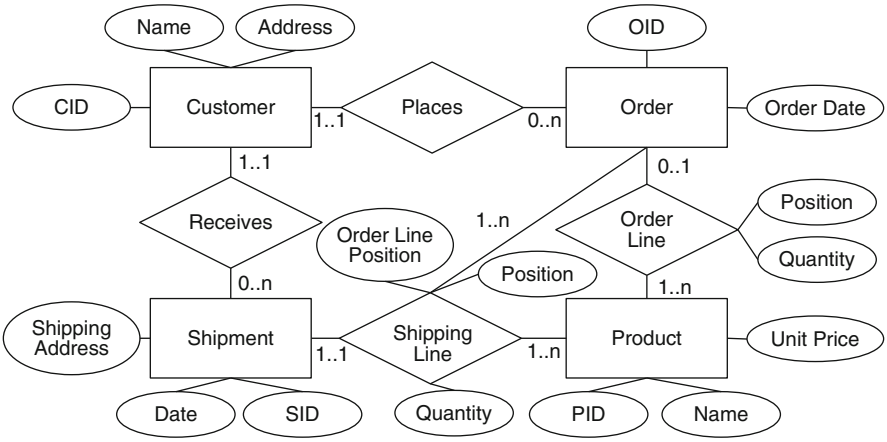

Fig. 2.2 Customer order entity relationship (E/R) diagram (Chen's Notation [33])

and can receive the ordered products in any number of shipments. According to, for example, availability, products within an order are assigned to shipments.

In the following, the data models for transaction and analytical processing systems that have been used at some point in the history or are still used are introduced. Their characteristics, which led to the development of new data models, are discussed, and finally they are compared according to their application areas.

2.1.1 Transaction Processing Systems

With the development of the first transaction processing systems in the 1960s, respective data models to store and manipulate data through applications in a standard way were introduced. Until the mid 1970s, most organizations used file systems to organize their data in so-called flat files. Only a small number used database systems [34]. The first database systems, developed in the 1960s were hierarchical and network systems. The hierarchical and network data models were afterwards defined as an abstraction from the real systems that already existed [40]. In contrast, Codd [38] introduced the relational data model before the implementation of relational database systems.

Flat File Model

The first step to manage data in a structured way was a simple organization of data in records (sets of fields) that were stored in files. The name flat file model or file management system stems from exactly this organization where the database consists of one or more files containing records.

To give a simple example, a company wants to store data about its customers and the orders of its customers, i.e., the company's sales orders. Using flat files, two alternatives for storing such data are immediately conceivable: (a) one file that includes records, with each record containing customer as well as sales order data, (b) two files with one including records of sales order data and the other including records of customer data. In alternative (a) the relationship between customers and their sales orders is explicit within each record, but all records for different sales orders of the same customer contain the same data concerning this customer. This makes updates to customer data a hazard as all records concerning the updated customer have to be changed. In alternative (b), this problem does not arise, but applications have to handle the relationships between customers and their orders themselves as flat files provide no concept for managing relationships.

Haderle [84] states that the flat file data organization already offered limited concurrency and recovery functionality. These are a basis for transaction processing. Moreover, since management of simple records did not include relationships between records or different kinds of records, the knowledge about data organization and determination of data semantics was left to the applications, i.e., no unified way of accessing data and standard data operations were defined as part of this data organization.

The drawback of this approach is that data access directly manipulates the physical data organization. If several applications access the same data, the logic for data access, interpretation, and operations has to be implemented in all of them. These redundant snippets of code increase maintenance overhead. In addition, advances in storage hardware technology introduce changes in the physical data organization to utilize new features and these entail changes to the code within each of the affected applications.

Hierarchical Model

An early work that separated the definition and layout of the data files from their access was a program called Generalized Data Update Access Method (GUAM) [163]. Access logic concerning data organization, maintenance, and data integrity was moved from the applications into a separate layer. Consequently, application code was simplified and the direct dependence of applications on physical data storage was removed [163].

GUAM built on the concept that larger components are created from the combination of smaller components resulting in a hierarchical structure [46, p. 24]. Thus, the hierarchical data model appeared in the 1960s out of the need to manage tremendous amounts of data created by complex manufacturing projects, such as the Apollo rocket project [47, p. 36]. In this context, IBM's Information Management System (IMS) was developed as a joint project with the North American Rockwell Space Division based on the GUAM program. The joint project ended in 1968, but IBM continued to develop IMS as a database and released it as a product called IMS/360 Version 1 in September 1969 [138].

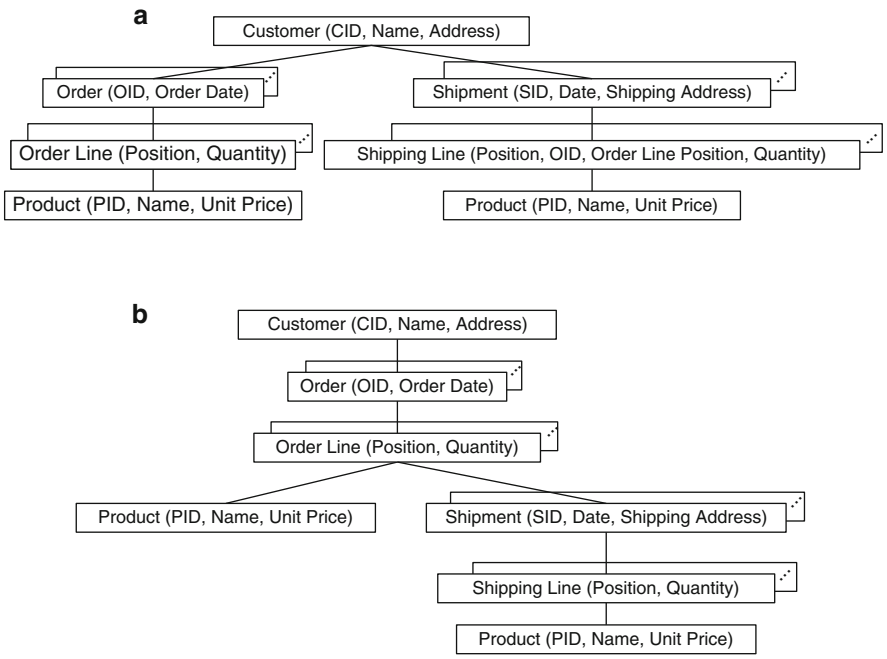

Fig. 2.3 Logical structures of the hierarchical customer order hierarchy. (**a**) No order/shipment mapping. (**b**) Direct mapping of shipments to order lines

The hierarchical model contains several levels starting from a root node, each level with nodes that act as the parents of the nodes on the next lower level if there is a relation between them. Consequently, the hierarchical model is a set of *one-to-many* relationships between parent and child nodes. Each node is stored as a record that maintains links to the node's parent and its children. Different types of nodes can be modeled in order to map diverse types of data entities, for example, a customer entity that contains the name of the customer and his contact data, or a product entity containing a description and pricing information. Applications retrieve data from hierarchical databases by finding the root node of a tree and then following the pointers stored in the records.

Figure 2.3 shows two alternative logical structures to model the customer order hierarchy. Depending on frequent access paths one or the other alternative is of advantage. Considering applications that frequently access shipments with no regard of the associated orders, the structure in Fig. 2.3a is beneficial as the depth of the tree that needs to be traversed is kept at a minimum and the number of branches correlates with the number of shipments. In contrast, shipments are scattered across orders in Fig. 2.3b, which benefits applications that rely on the relation between orders and shipments, such as, reporting orders that have not been shipped (completely).

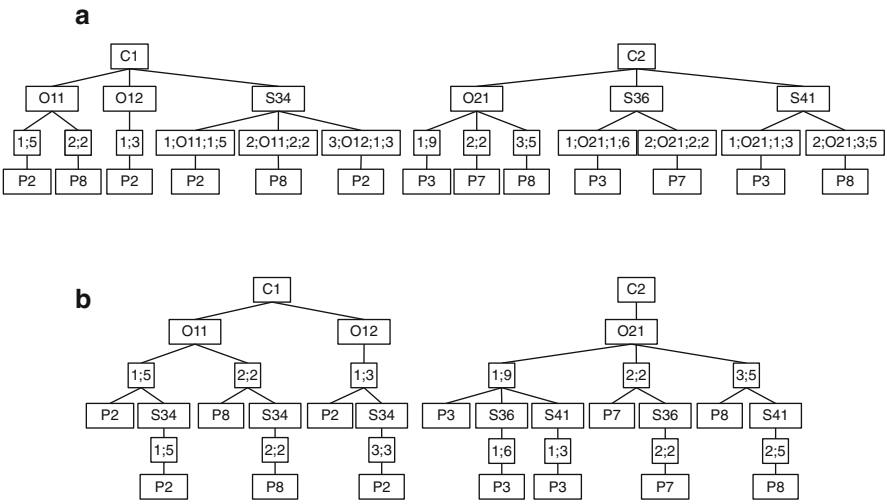

Fig. 2.4 Example hierarchy instances for the customer order hierarchy. (**a**) No order/shipment mapping. (**b**) Direct mapping of shipments to order lines

Figure 2.4 exemplifies the aforementioned alternatives for the hierarchical customer order hierarchy. Nodes in the hierarchical model are instances of entities, e.g., a customer named $C1$. Listing 2.1 presents the orders and shipments depicted in Fig. 2.4. Two customers $C1$ and $C2$ place orders for the products $P2$, $P3$, $P7$, and $P8$. The numbers provided in the order lines and shipment lines represent the ordered and shipped quantities.

The different modeling alternatives introduce differing levels of redundancy. While in the alternative in Fig. 2.4a only the product instances are repeated and an overview of complete orders and shipments per customer is simply achieved by scanning the respective subtrees, the alternative in Fig. 2.4b stores redundant shipment data for each order line besides the redundant product data, but it simplifies queries associating order lines with their shipments. For example, it is easy to determine that the order for product $P3$ of customer $C2$ is split across two shipments, that is, $S36$ and $S41$, but to analyze the contents of an entire shipment, a scan of the complete subtree of this customer is needed.

Logical relationships were added in IMS/360 2.0 to efficiently handle many-to-many relationships [138]. They interrelate segments from different physical hierarchies building a logical hierarchy. Logically connected physical hierarchies may constitute a network data structure, although application data is still stored in one or more physical hierarchies [132]. Concerning the previous example, logical relationships can help to avoid the redundant storage of product records: An own hierarchy for products can be introduced and logical paths from the order line records of the customer order hierarchy to these product records can be created.

```
Customers = {(C1,"Werkstatt Hein","Rosenthaler Grenzweg
    5, Berlin"),
             (C2,"Mc Tools","361 Peachtree Street,
                Atlanta")}
Products = {(P2,"Integral Panel Clamp C4.a",19.99),
            (P3,"Split Point Drill 35mm",4.99),
            (P7,"Machinist's Chest XK044",209.00),
            (P8,"Hot Ring Plier Set T68",27.00)}
Orders(C1) = {(O11,2011-01-05),(O12,2011-01-07)}
Orders(C2) = {(O21,2011-01-27)}
OrderLines(O11) = {(5,P2),(2,P8)}
OrderLines(O12) = {(3,P2)}
OrderLines(O21) = {(9,P3),(2,P7),(5,P8)}
Shipment(C1) = {(S34,2011-01-14,"Rosenthaler Grenzweg
   5, Berlin")}
Shipment(C2) = {(S36,2011-02-03,"1055 Ashbury Rd.,
   Fulton"),
                (S41,2011-02-17,"1055 Ashbury Rd.,
                   Fulton")}
ShippingLines(S34) = {(5,P2),(2,P8),(3,P2)}
ShippingLines(S36) = {(6,P3),(2,P7)}
ShippingLines(S41) = {(3,P3),(5,P8)}
```

Listing 2.1 Order and shipment example

IBM's IMS as the prevalent representative of the hierarchical data model is still in usage for enterprise data management today and widely spread in banking and insurance [121, 155].

Network Model

Similar to the hierarchical model, the network model has been defined after it was already used in database system implementations. Network models have been developed by the Conference on Data Systems Languages (CODASYL) Data Base Task Group (DBTG) [37]. On that account, they are also called CODASYL database models or DBTG database models.

Record types in the network model represent entities and set types represent relationships between entities. Each set type has exactly one owner record type and one or more member record types. The term record denotes a specific instance of an entity. A specific relation of this record to other records is called set. Each set contains exactly one owner record and zero or more member records of its defined member record types. In contrast to the hierarchical model, many-to-many associations can directly be represented by creating a new entity that represents this association [202].

Figure 2.5 presents the network model of the customer order example introduced in Fig. 2.2 according to the data structure diagram notation by Bachman [7]. Record

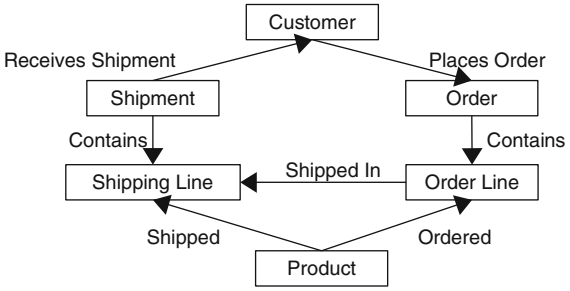

Fig. 2.5 Network model for customer orders and shipments

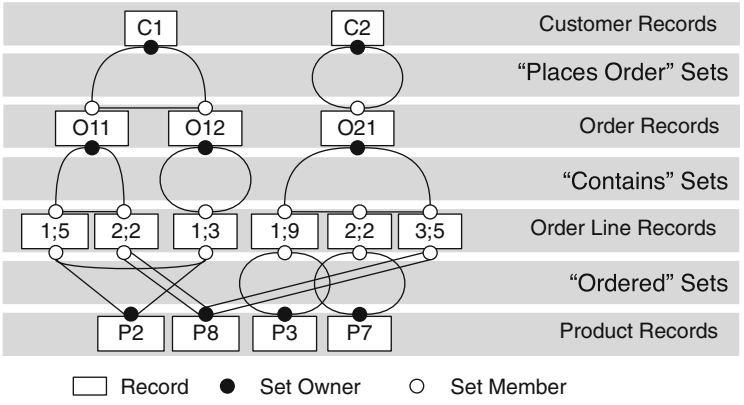

Fig. 2.6 Navigational routes through the customer order network model

types are depicted as rectangles and set types are depicted as arrows between record types. The owner record type of a set type is connected to the tail of the set types' arrow and its member record type is connected to the head.

In addition to defining the data structure, the network model sets up navigational routes through this data structure to access data. This is similar to the pointers in the hierarchical model. The navigational routes for a part containing customers, orders, order lines, and products of the network model depicted in Fig. 2.5 are shown in Fig. 2.6.

Through the network structure, entire contexts around entities can easily be retrieved along the modeled navigational paths. A query that exemplifies the advantage of the network structure over the hierarchical structure, assuming no logical relationships have been defined within the hierarchy, is: *Retrieve the top ten customers with respect to the sales amount of the best-selling product of customer* $C1$. In the hierarchical model above this requires traversing all nodes in the subtree of customer $C1$ to find the most-bought product and then traversing the entire tree to find the top ten other customers as product data is stored in the leaf nodes. In the network model the first step to find the best-selling product of customer $C1$ is the

same, but then the *Ordered* set associated to that product can be used to find and evaluate the orders of other customers. In this case, only customers having actually ordered that product are considered.

To achieve a similar behavior in the hierarchical model, IBM IMS provides bidirectional logical relationships [97] that are supported by a subset of its hierarchical database types. With these, logical paths from order line nodes of one physical hierarchy can be modeled that point to nodes of ordered products stored in an another physical hierarchy and from the product nodes back to the first hierarchy to all order line nodes that contain this product. In contrast to the network model, this approach needs additional logical structures and pointers, increasing the overhead for the DBMS.

Relational Model

In the late 1970s, IBM developed "System R" as an experimental prototype for a DBMS that uses the relational database model (RDBMS – relational database management system). According to Astrahan et al. [6] System R was intended to demonstrate that a relational system can be used in real environments with a comparable performance to the existing systems of that time. The structured query language (SQL), initial called "SEQUEL" (structured English query language) [27], was developed as a query language to retrieve and manipulate stored data in System R. To test and evaluate System R, several installations at internal locations of IBM were set up. The project ended in 1979 with the result that the relational model is the basis for a viable database technology with commercial potential [93, p. 85]. Chamberlin et al. [28] came to the conclusion that databases on the basis of the relational model are able to support many concurrent users, who perform repetitive transactions and ad hoc queries. They argue that the high-level user interface enabled by the relational database model positively effects user productivity for developing new applications.

Along with System R, INGRES (Interactive Graphics and Retrieval System) was one of the first projects to provide a proof of concept for a practical DBMS based on the relational model. INGRES was started as a research project at the University of Berkeley, California. The motivation behind INGRES was the utilization of two basic characteristics of the relational model, namely, the high degree of data independence and the possibility for a high level query language [199]. INGRES included an own language called "QUEry Language" (QUEL). When SQL evolved as the standard database language [50], INGRES was converted to supporting SQL [93, p. 86], while continuing to support QUEL.

IBM's System R and INGRES, however, were not the first to spawn a commercial relational database for the market. Greenwald et al. [77] claim that Oracle V.2, released in 1979, was the worlds first commercial relational database. According to Hernandez and Viescas [93], the first commercial version of INGRES entered the market in 1981. IBM announced its own RDBMS called SQL/Data System in 1981 [41] and shipped it starting in 1982. In 1983, IBM announced Database 2

Product

PID	Name	Unit Price
P2	Intergrip Panel Clamp C4.a	19.99
P3	Split Point Drill 35mm	4.99
P7	Machinist's Chest XK044	209.00
P8	Hog Ring Plier Set T68	27.00

Order

OID	Order Date	Buyer
O11	2011-01-05	C1
O12	2011-01-07	C1
O21	2011-01-27	C2

Customer

CID	Name	Address
C1	Werkstatt Hein	Rosenthaler Grenzweg 5, Berlin
C2	Mc Tools	361 Peachtree Street, Atlanta

Order Line

LID	Order	Product	Quantity
1	O11	P2	5
2	O11	P8	2
1	O12	P2	3
1	O21	P3	9
2	O21	P7	2
3	O21	P8	5

Shipment

SID	Date	Shipping Address	Receiver
S34	2011-01-14	Rosenthaler Grenzweg 5, Berlin	C1
S36	2011-02-03	1055 Ashbury Rd., Fulton	C2
S41	2011-02-17	1055 Ashbury Rd., Fulton	C2

Shipping Line

Ship.	Pos.	Order	OL Pos.	Prod.	Quantity
S34	1	O11	1	P2	5
S34	2	O11	2	P8	2
S34	3	O12	1	P2	3
S36	1	O21	1	P3	6
S36	2	O21	2	P7	2
S41	1	O21	1	P3	3
S41	2	O21	3	P8	5

Fig. 2.7 Relational structure for the customer order example

(DB2), which was shipped starting in 1985. DB2 evolved from System R, IMS and technological advances made since then [85].

Compared to the network model, in the relational model an entity is represented as one or more tables. Single instances of an entity or parts of that entity are represented as a tuple within the tables that make up the entity. Relationships are also represented as tables, with their tuples associating the tuples of other tables (the instances of other entities) [143]. A tuple is a set of values, one for each characteristic of the entity, also called attribute. To uniquely identify each tuple of a table, a primary key can be defined as a subset of a table's attributes such that no combination of their values is encountered more than once within the table. In contrast to the record (a tuple's physical representation on the storage medium), a tuple does not declare an order on the attributes it contains. The attribute values of a tuple are accessed by their attribute names. Additionally, no order of tuples of a table exists on the logical abstraction layer.

Figure 2.7 shows the relational structure for the customer order example given in Fig. 2.2. In this structure, each entity is modeled as one table and the two relations *OrderLine* and *ShippingLine* are modeled as tables as well. They include references to those instances (tuples) of the entities taking part in the relation. For example, the *Product* and *Order* attributes of the *OrderLine* table connect the respective order and product instances. The relations *Places* and *Receives* are modeled as additional attributes in the tables of those entities referring to them. For the *Places* relation this means the attribute *Buyer* in the *Order* table and for *Receives* an attribute *Receiver* is added to the *Shipping* table. Such references to existing tuples within other tables

are called foreign keys if their elements are values of the primary key of another table [38].

Basic operations for data retrieval in the relational structure are set operations, such as, union, intersection, and difference and further typical operation are [69]:

- Selection: the restriction of the returned tuples to a subset defined by a condition (also called filter). A term often used in context with selection is selectivity. It is a measure of the size of the returned subset of tuples. High selectivity means that the condition applies to only a small number of tuples, which are returned. Low selectivity, in contrast, means that the condition is applicable to many tuples, which are then returned in the result set.
- Projection: returns a specified subset of attributes from a table or result set for the selected tuples.
- Joins: a conjunction of one table with itself or between different tables based on conditions defined on attributes.
- Grouping: collocation of tuples in the result set based on the values of specified attributes.
- Aggregation: summary of specified attributes according to a given function, such as, average or summation, and a given grouping.
- Sorting: ordering of tuples in the result set according to the values of specified attributes.

Entry point for data retrieval can be any table defined in the database. Navigation is possible through any compatible pair of attributes of one table or different tables rather than through predefined sets linking the tables.

Discussion of Data Models in Transaction Processing

Similar to file system models, the early hierarchical and network data models and applications relying on them were connected too closely. Data was manipulated via standard host programming languages. As a result, changes to the data storage in order to leverage latest advances in hardware technology induced changes in the applications to preserve functionality. This process endangered a companies investment in existing applications, that would have to be changed over and over again, causing additional costs [84].

Hierarchical models are very efficient for large amounts of data and provide high transaction throughput, but do not allow very flexible relations and require extensive application programming to use the database. Network models are more flexible regarding data access paths and many-to-many relationships are easier to implement than in the hierarchical model, but their structure can easily become very complicated [151, Chap. 20]. In both, network as well as hierarchical models, the database structure has to be designed around the access paths that applications use most often. Logical relationships within hierarchical databases provide the possibility to define additional access paths if needed, incurring some overhead through extra pointers, but reducing redundancy.

In contrast, a relational database is not restricted to specific application access paths, but rather provides the means to model a data set in a logical database design on which applications can work using a high-level query language. Through high level query languages, applications are completely decoupled from the physical representation of data on the storage medium [38]. Consequently, they are not affected any longer by changes in physical database design caused by advancements in hardware technology. The foundation for this independence from hardware changes is the complete specification of the relational model including its operators on top of which the high level interfaces are defined. The operators defined with the relational model provide access capabilities for single entity instances as well as for multiple instances at a time (called set processing). In the hierarchical and network models, set processing in the sense of mathematical sets had to be handled by the application programmers, who were forced to implement iterative loops to navigate through the data structures and collect all data [42]. An example query is the *selection of the combined value of all sales orders in January*. Some transaction processing applications, for example, dunning, and especially analytical use cases rely on set processing, though.

After the development of RDBMS, new database systems called object databases emerged. They targeted the encapsulation of structure and behavior along the lines of object-oriented programming. The object data model is not discussed in more detail as object databases are less relevant for business systems but rather established niche markets for application areas that rely on the management of complex objects. In response to the development of object databases, RDBMS vendors added extensions to their systems. Object-relational DBMS were the result, offering similar functionality as the object databases on top of a relational model, for example, access to the relational database through object-oriented routines using the given standard interfaces (e.g., SQL). Object databases have established a market for applications with special requirements directly benefiting from the functionality offered by object databases, such as, computer-aided design, telecommunication systems, or geographic information systems [53, Chap. 1]. Yet, they never set up a strong foothold in enterprise data management [12].

Some database vendors optimized their data models for specific applications in the 1980s to minimize computing resources [84]. An example was IBM's IMS/VS 1.1.4 fast path feature for critical banking applications that increased throughput by trading generality for specialized application program scheduling and data storage techniques [138]. Restrictions included, for example, no support for inserts in transaction mode or only providing one data access path [84]. Today, RDBMs are widely accepted and used in transaction processing scenarios [92, Chap. 1]. Database systems based on the other models are still in use, however, for special application areas that pose requirements that directly exploit their advantages. For example, IBM IMS with its hierarchical database model allows for a very high transaction throughput in settings that require complex data structures with many hierarchy levels. Thus, use cases for the IMS database are found in the finance, retail, and telecommunications industries, where management of highly complex records and high transaction throughputs are a essential [155].

Independently and in parallel with the development of hybrid databases based on the relational approach, another movement has originated that promotes a new kind of database systems that follow a non-relational approach. It is called NoSQL, standing for "not only SQL" or "not relational". According to Cattell [24], these systems sacrifice some dimensions such as consistency, durability, availability, or query support to achieve others, e.g., higher availability and scalability. Increasingly variable document types and scalability issues in Web applications have created a push for NoSQL in the Web developer community [127]. Yet, their application in the enterprise environment is questionable. Stonebraker [198] states that NoSQL databases are most often considered for update- and lookup intensive OLTP workloads and not query intensive analytical workloads. This covers only a part of today's enterprise workloads.

While early data management systems possessed transaction processing functionality as well as a wide range of decision support tools, developments in the late 1990s have resulted in an optimization of systems according to either the OLTP or the OLAP application area. Specialized data models that emerged to support analytical functionality efficiently will be discussed in the following.

2.1.2 Analytical Processing Systems

As transactional database systems became more sophisticated and stored more data, analysis needs also became more demanding. The extensive supply of data being stored in the transactional database was expected to be available for analysis. Database manufacturers in the mid 1990s attempted to address the transactional as well as the analytical side of business and even simple analytical queries took hours to run, creating the problem of "Too Much Data, Not Enough Information" [61]. Another challenge that systems providing analytical functionality face is the rate at which the needs and accordingly the requirements change. Since markets are becoming increasingly fast-paced, enterprises have to plan and react similarly fast, demanding the flexibility to serve new reporting needs, and include as much and as detailed data as possible, for both historical and up-to-date data analyses.

Dimensional Modeling in Analytical Processing

In contrast to transaction processing with insert, select and update behavior, analytical query processing can be classified as a read-only workload considering the actual user interaction. Bulk inserts to update data are executed in the background or during low system load times to avoid affecting user interaction performance. Queries are composed of complex data selection. The following query exemplifies where the complexity in the selection lies:

> Display the number of purchases per month, product category, and city for the first quarter of 2011.

For this query, the entire set of sales data for the first quarter of the year 2011 is scanned to compute the number of purchases. In a typical database design for transaction processing based on the relational model, this would entail several joins, for example, between sales order header and item tables, product data tables, and tables containing region data. Especially the join of sales header and item tables is expensive because these are tables where transaction data is collected resulting in millions of entries according to business throughput. The dimensional modeling approach [119] typically used in analytical database design avoids joins between large tables and also reduces the number of joins compared to a normalized transactional database design.

In the dimensional modeling approach, data is separated into dimensions and facts. This separation stems from the classification of data into transaction and master data. Transaction data is accumulated during daily operations of a company and includes sales orders, payments, purchases, and production data to name just a few. Master data is changed only infrequently [221] compared to transaction data, that is changed with every business transaction. Examples for master data are customer, supplier, product, or location data.

Dimensions contain master data clustered into groups, for example all product specific data is stored in the product dimension, or time data in the time dimension [119, Chap. 1]. Dimensions are used for filtering and for creating the context by means of which facts are aggregated according to selections within the dimensions. Facts are numeric values from the transaction data [119, Chap. 1].

In the example query above, dimensions are *Time*, *Product*, and *Location*. The fact to be aggregated is *number of purchases*. The time dimension is used for filtering (*first quarter of 2011*) and the granularity level to aggregate the fact on is *month* of the time dimension, *category* of the product dimension, and *city* of the location dimension. Figure 2.8 shows an extract of the reporting result as a table for this query. The level of granularity defines how much detail is provided about the composition of the fact aggregate. Aggregating by product category and brand would increase the level of detail, whereas the aggregation by country would remove details from the report.

Typical operations based on the dimensional model include roll-up, drill-down, slice-and-dice, and pivot [30]. The names for these operations stem from imagining the dimensional data structure as a cube (in the case of three dimensions). Figure 2.9a shows the cube for the example report given in Fig. 2.8. The roll-up operation decreases the level of granularity, for example, showing the number of purchased products per quarter instead of per month (see Fig. 2.9b). Drill-down is an operation in the opposite direction, increasing the level of detail, for example, products per week, instead of per month. Slice-and-dice restricts the result set or relaxes restrictions on the result set according to further criteria on dimensions, for example, only showing the top ten products sold in January (slicing operation, see Fig. 2.9c), or only showing certain cities and categories (dicing operation, see Fig. 2.9d). Pivoting is the operation of rearranging the multidimensional view of the data. It is a table operation, for example, exchanging the columns and rows.

		Month			
City	**Category**	**01/11**	**02/11**	**03/11**	**Total**
Hanover	**Pliers**	2	9	4	15
	Clamps	13	15	12	40
	Drills	9	7	11	27
	Subtotal	24	31	27	82
Munich	**Pliers**	23	21	43	87
	Clamps	31	29	35	95
	Drills	54	97	67	218
	Subtotal	108	147	145	400
Berlin	**Pliers**	15	7	12	34
	Clamps	25	37	22	84
	Drills	23	19	15	57
	Subtotal	63	63	49	175
Total		195	241	221	657

Fig. 2.8 Reporting result for the number of purchases per city, category, and month

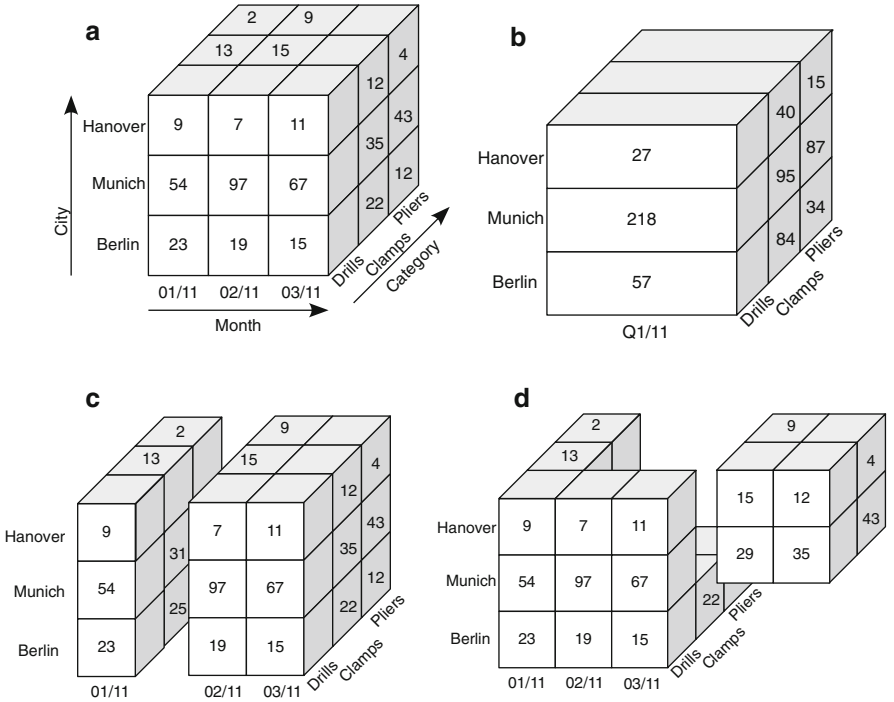

Fig. 2.9 Cube and operations for the number of purchases per city, category, month. (**a**) Cube structure. (**b**) Roll-up operation. (**c**) Slicing operation. (**d**) Dicing operation

From an application usage perspective, transactional and analytical workloads differ largely. Transactional applications automate clerical data processing tasks. These are structured and repetitive in their nature [30] and are concerned with single or a small number of instances of an entity, for example, the sales order of a specific customer and its associated line items. Transactions require up-to-date data. It is essential that changes once committed are preserved, and that they are not restricted by throughput or other resource scheduling constraints. Failures of customer transactions, for example, due to throughput limitations can have a direct impact on a company's revenue and, thus, have to be avoided.

Analytical systems support employees, e.g. executives, managers, and analysts to make better and faster decisions [30]. Much more data is accessed in one report compared to a transaction, especially including historical data, like sales data of a specific region for the entire past year. As analytical reports include many dimensions by means of which facts are aggregated, the underlying queries can become very complex, that is, contain many join, grouping, and sorting operations. Using optimized multidimensional data structures to store data already prepared for analytical needs reduces this complexity. Such data structures will be discussed in the following. The nature of analytics lies in retrieving data from the system, interpreting the results, and inquiring the reasons for unexpected results. Because any inquiry is conceivable, a considerable share of analytical queries is composed of ad-hoc queries. In many cases, they emerge from repetitive reports, for example, from questioning irregularities in the results of a report to create the annual financial statement.

Analytical Reporting Categories

Different categories of analytical reporting exist. Parameters for categorization are the period of source data that is taken into account, the business focus for which the analysis results are needed, or the primary users of the reports. Inmon [104] introduces two categories of reporting: operational and informational. Operational reporting analyzes the detailed transaction data of an enterprise with up-to-the-second accuracy. Inmon names daily production records, or flight-by-flight traveler logs as examples. The goal of informational reporting, in contrast, is strategic analyses and longer-term decisions. Therefore, informational reporting focuses mainly on summary data, such as, monthly sales trends or annual revenue by region, and less on details. The main difference between the two is the freshness of the data analyzed and the time window sizes that are analyzed. For the operational side this means maximum freshness and minimal time slices (hours, days). For the informational side it means less fresh data and longer periods to analyze, e.g. months, quarters, or years.

In contrast to Inmon, White [220] distinguishes three categories of reporting: strategic, tactical, and operational. Strategic reporting supports the execution of long-term business plans and measuring the progress toward organizational goals, such as growing market share or increasing revenues, based on high-level business

performance metrics. A simple, but typical example for such a metric in the context of sales order processing would be setting target net sales revenues per region per quarter and monitoring their fulfillment in comparison to actual sales numbers. The aim of tactical reporting is to monitor business initiatives, such as marketing campaigns that are set up to ensure reaching the long-term goals. Tactical reporting analyzes business operations within a time window of days, weeks, or months. Compared to Inmon, strategic and tactical reporting are subcategories of informational reporting, based on historical data, analyzing time frames of days to weeks to months (tactical reporting) and months to years (strategic reporting). Operational reporting, according to White and similar to Inmon is concerned with daily business and is based on intra-day analyses. Credit card fraud detection and inventory management are typical examples of operational reporting.

Data Storage for Analytical Processing

In the late 1990s, special OLAP systems evolved that solely focused on providing decision makers with information. These systems organize data according to the usage requirements of decision makers using a multidimensional paradigm suited to model the natural structure of decision support problems [54].

According to the implementation of their data storage, analytical processing engines can be distinguished into two basic types: These are relational OLAP (ROLAP) and multidimensional OLAP (MOLAP) engines [32]. MOLAP engines store multidimensional data directly in spatial data structures, for example arrays. Queries from the front end are directly mapped to these special data structures. In ROLAP engines, data is stored in a relational database with the engine implementing efficient algorithms for analytical operations that translate between the above logical multidimensional model and the relational storage model below. MOLAP engines provide exceptional performance in accessing data since the dimensional value combination of a fact is implicitly given by its address. However, as many value combinations may not exist, for example, some products are not sold in some regions resulting in empty values for facts in the associated array cells, MOLAP structures can become sparse leading to a poor utilization of storage space [30]. ROLAP does not suffer from this problem. If facts do not exist for a combination of dimension values, nothing is stored. Consequently, hybrid OLAP (HOLAP) approaches have been introduced to utilize the advantages of both storage techniques. Dense regions of a cube could be stored using MOLAP and sparse regions using ROLAP. Normalization of a data cube, that is choosing an appropriate ordering of attribute values (dimension values), aims at distributing data of a cube into dense and sparse regions, thereby increasing storage efficiency [115].

Relational databases are still the dominant choice as the underlying technology for analytical processing systems [139]. Thus, this work will further focus on ROLAP engines, instead of MOLAP and hybrid approaches.

Data Warehouse, Data Mart and Operational Data Store

A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data to support decision-making [108, Chap. 2]. Instead of clustering data according to the processes like in operational systems, data is structured according to the areas relevant for decision-making, leading to subject-orientation. For a retailer, subject areas of interest may be sales, products, or vendors. Integration means that data from several sources, for example, operational systems and external services, is collected and provided in a single source. The nonvolatile characteristic means that data already in the data warehouse is never updated. New data is added in snapshots, resulting in a historical record of the operational data. Time-variance in this context means that each data element is valid as of a certain point in time. Data records may be given a time stamp or validity interval to determine their validity period.

To create and update the data warehouse ETL tools are used. They transform the source data to fit the analytical business needs and support loading of the transformed data into the warehouse [214].

In many industries data warehouses are successfully employed. Their use cases include order shipment and customer support in manufacturing, user profiling and inventory management in the retail industry, claims, risk, and credit card analysis in the financial sector, call analysis and fraud detection in telecommunications to name but a few [30]. Although the idea is that a company has one data warehouse that is managed centrally as a source for all company-wide reporting needs, it is rarely achieved. Because of, for example, acquisitions and mergers the system landscape of companies is heterogeneous in the analytical as well as the operational domain.

In contrast to the data warehouse, the data mart contains customized data [101]. Inmon [102] defines a data mart as a data structure that is dedicated to serving the analytical needs of one department within a company. Data marts are differentiated into dependent and independent data marts. Dependent data marts are built from data coming from the data warehouse. Independent data marts are built from data coming from the operational systems. They are inexpensive to build, as organizational groups can collect their requirements and set up a data mart based on data in the operational systems without consulting other organizational groups. Inmon states that the drawback of this inexpensive setup of independent data marts can be their uncontrolled growth in the long term. For example, each department tries to service its reporting needs. Yet, none of the existing data marts of other departments fits their specific needs completely resulting in the creation of a new data mart. Thus, it can happen that the same detailed operational data is redundantly stored in many data marts that differ only slightly in their structure and context. Building dependent data marts requires a larger amount of foresight, as requirements of different groups have to be collected to build the basis for these data marts in the data warehouse. However, while independent data marts service instant reporting needs, dependent data marts provide a sound long-term foundation for information decisions.

The different approaches to designing a data warehouse should not be disregarded. Inmon's approach mentioned above is one option. Another option is Kimball's approach [109] to design a data warehouse as a collection of dimensionally modeled data marts. Yet, another approach is using independent data marts to design a data warehouse. This approach, however, is seen as inappropriate in the data warehouse community as ETL steps are repeated unnecessarily and they lack cross-department analysis and communication capabilities [114]. Kimball's approach achieves results quicker and simpler than Inmon's, but a common criticism is that it lacks enterprise-wide focus. Jukic [114] describes the different approaches and outcomes of Inmon's and Kimball's methodologies as a trade-off between extensiveness and power versus quickness and simplicity. A detailed comparison of the Inmon and Kimball approaches can be found in [21].

Another structure in the analytical landscape that is complementary to the data warehouse according to Inmon [108, Chap. 16] is the operational data store (ODS). It is a subject-oriented, integrated, volatile, current-valued, detailed-only collection of data to support a companies need of reporting on up-to-the-second, integrated, operational data [100]. Subject-orientation and integration are similarities between the data warehouse and the ODS. Differences lie in the freshness of data, the amount of data that is kept in the ODS, and that data in the ODS is updated by overwriting existing entries instead of adding another snapshot. Due to the updates, the ODS contains no historical data. In the ODS only detailed data is kept, whereas a data warehouse contains detailed and summary data. ODSs are categorized into different classes according to the length of their update intervals affecting the freshness of data they contain.

Figure 2.10 gives an example of how the different OLAP data stores can be associated and shows possible flows of data between them. Sources of data can be enterprise resource planning systems, files like spreadsheets, or external services to name just a few. The ETL process between the data sources and the data warehouse is not explicitly shown in this overview. A detailed comparison of data warehousing methodologies including all of the three mentioned analytical structured is given in [184].

Inmon [100] initially defined three classes of ODSs. Class I ODSs are kept synchronized with the operational systems they retrieve data from, so that data is available for reporting only seconds after it is inserted in the operational systems. Class II ODSs are updated periodically every hour or in similar time intervals. Updates to the operational systems are stored in an intermediate file, which is then loaded into the ODS. Class III ODSs are subject to the same process, but the time interval between data updates in the ODS can be 24 h or more. The business case for class II and III ODSs is the most common [100]. Operational costs for class I ODSs are much higher because of the immediate synchronization and business cases justifying this class of ODS are rare [100]. A fourth ODS type (class IV) was introduced later on. This class of ODS holds results of reports from the data warehouse [105]. The rationale for the creation of this ODS class was that the data warehouse did not provide responses in real-time. The Class IV ODS is able to do so

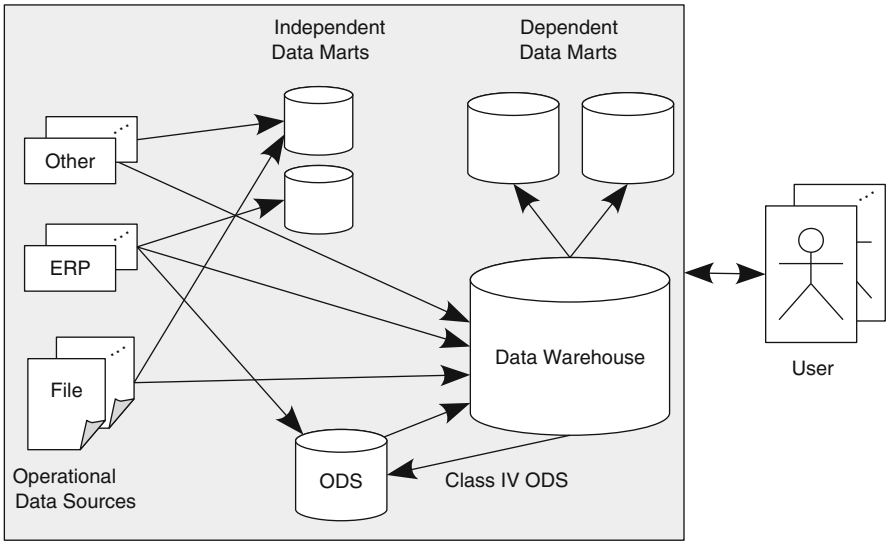

Fig. 2.10 OLAP data stores and data flow in between

for a limited number of prepared reports, thus trading off fast response times against freshness of data [106].

2.2 Relational Database Design

Database design incorporates the definition of the involved business entities, a database schema, and additional structures to accelerate access to data. It is divided into logical and physical database design.

> [T]hink of the logical database design as the architectural blueprints and the physical database implementation as the completed home. – Hernandez [92, p. 29], 2003

Figure 2.11 presents the steps taken in the process of database design. Logical database design starts with the creation of the conceptual model. It specifies what data is included from the business perspective and which relationships exist between business entities [204]. The next step is the creation of the database-specific implementation model, that is, a database schema. While a data model is a set of mechanisms to structure data, a database schema is the definition of the structure of a specific data set with the help of a data model. The database schema as the logical design of the database has to be differentiated from the database instance, which is a snapshot of the data in the database at a given point in time [191, Chap. 2]. For relational databases, the database schema includes the definition of tables and the

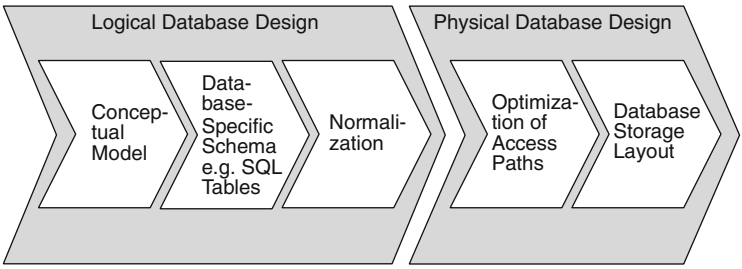

Fig. 2.11 Database design steps

normalization of those tables corresponding to the semantics of data relationships, which will be discussed in Sects. 2.2.1 and 2.2.2.

In 1980, Teorey and Fry [203] have classified logical database design into *Information Structure Design* (ISD) and *Information Structure Refinement* (ISR). Independent of the data model to be used, information structure design gathers all application requirements into a preliminary high-level schema. This phase results in entities, associated attributes that hold detailed information about the entities, and relationships between entities. In the ISR phase, the results of the ISD phase are further developed into a database-processable schema, for example, SQL table definitions. This basic distinction has remained valid in later work. Only a change of naming has taken place from ISD to conceptual modeling and ISR to comprise the creation of the database-specific implementation model, for example, the relational database schema.

Physical database design includes the definition of further access methods to optimize data access, that is, indexes and views, partitioning of tables, and data clustering. This amounts to structuring data with respect to specific machines and optimization of performance regarding a particular workload. Physical database design structures are presented in Sect. 2.2.3. Section 2.2.4 gives an overview of database-specific storage alternatives that have a strong impact on transactional and analytical processing.

2.2.1 Relational Database Schemas in Transaction and Analytical Processing

Transactional relational database schemas follow the principles of normalization. These were developed to achieve a database design with little or no redundancy to avoid false relationships and inconsistencies [84]. In [39], Codd introduces the objectives of normalization, that include obtaining a powerful retrieval capability by simplifying the set of relations. Thus, undesirable insertion, update, and deletion anomalies are removed. Additionally, the need for restructuring when adding new types of data is reduced, which increases the life span of applications.

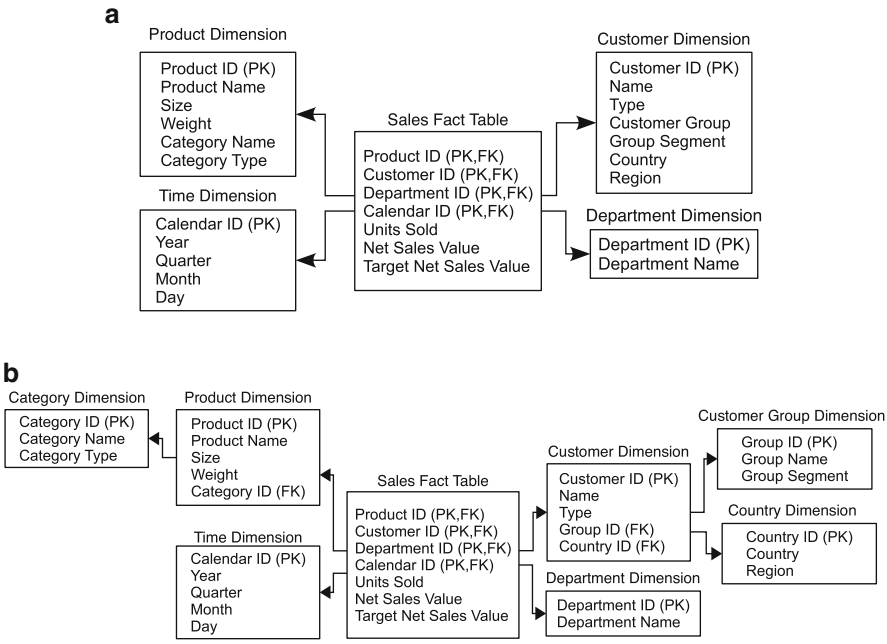

Fig. 2.12 Example ROLAP structures for the sales order example. (**a**) Star schema. (**b**) Snowflake schema

Analytical relational database schemas follow the dimensional modeling approach and the according relational database schemas are the star schema and the snowflake schema [54]. Figure 2.12 shows the star and snowflake schemas according to the sales order example. The star schema (see Fig. 2.12a) is composed of a central fact table and multiple dimension tables. The fact table contains data from business transactions or a snapshot summary of that data. It is connected through a many-to-one relationship to each dimension table.

To speed up joins between the dimensions and the fact table, which is relevant for every OLAP query, join and bitmap indexes are used [123]. The snowflake schema is an extension to the star schema (see Fig. 2.12b) to model hierarchies explicitly, thus normalizing the dimensions. Thereby, redundant data storage within the dimensions is reduced, but additional joins are needed if data from the outer dimensions is required.

Even normalized schemas, for example, in third normal form (3NF) [39] are employed in some analytical use cases, backing away from the dimensional model. The different analytical database schemas are used in different kinds of data stores that exist in the OLAP environment to cater for specific reporting needs. Martyn [137] states that it is reasonable to use the star schema in a data mart, the snowflake schema in a data warehouse and the 3NF schema in an ODS.

2.2.2 Normalization

While reducing redundancy and avoiding false relationships and inconsistencies, an increased level of normalization of a set of relations entails a penalty towards data retrieval. 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 [118] acknowledges, that the highest level of normalization does not need to be enforced, where performance requirements have to be taken into account. Mullins [146, Chap. 12] advocates 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 systems. Bock and Schrage [13] give an overview of denormalization techniques that are utilized in transaction processing schemas. Likewise, Mullins [146] discusses different types of denormalization and describes where they can be useful. Of these, the types *redundant data*, *repeating groups*, and *derivable data* can still be observed in today's transactional systems, mostly however because of legacy issues. Redundant data means including additional columns of another table in one table if they are always queried. This is only advisable if the included columns contain data that does not change frequently. Repeating groups comprises adding more columns for an attribute that can have multiple, but a limited and small number of values, instead of normalizing it into an own table with a foreign key relationship. A typical example is storing up to three telephone numbers for a customer. Thus, three *telephone number* columns are used instead of three rows in a second table. Derivable data comprises precomputing 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, *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 statements. Thus, the report results can be obtained using simple SQL queries. The star and snowflake schemas are members of this category. According to Martyn [137], the denormalization of OLAP schemas, that is, 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. Furthermore, for data warehouses where query performance is paramount, the snowflake schema is generally not recommended because of the reduction in query performance due to additional joins [171, Chap. 11]. Kimball and Ross also write 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." [119, p. 11] Date [51, Chap. 7] 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.

2.2.3 Physical Database Design

Physical database design is the technical specification of tables and the definition of additional structures to support query processing. The technical specification includes table names, column names, data types, primary keys, and foreign keys. Agrawal et al. [3] introduce physical design structures as any access path that is supported by a database server. These include indexes, views, (multidimensional) clustering, and partitioning of tables.

Indexes

Indexes provide alternative access paths to data items besides sequentially scanning a table and searching for a specific value. At the cost of write performance and storage space, indexes, thus, speed up data retrieval. Typical index structures are B-Tree, hash, and bitmap indexes with B-Tree being the most common [172, Chap. 8]. An index can be built on one or more columns and a table can have any number of indexes. The trade-off between index maintenance and fast data access in an ever-changing environment has to be considered. Some database engines support storing tables directly in an index structure, for example, Oracle8i index-organized tables [192]. The users or applications are typically not aware of indexes as the query optimizer decides if an existing index is used for a query or not.

Views

Views are defined as virtual relations [46] or single-relation images of queries [135]. Views are dynamically created through a stored query from the underlying data when required and may span one or more relations providing a subset of the entire data. A typical use case for views is security, where specific users should only have access to a subset of data, e.g. employees may view their own contract data, but not contract data of their colleagues whereas the manager has access to all contract data of his staff. In this case, users only have access to their dedicated view instead of to the tables the view is based on.

Special cases of views are materialized views. Gupta and Mumick [82] describe the materialized view as a view with its tuples being stored in the database. Thus, a materialized view is like a cache that stores a copy of data or derived data for quick access. A typical use case for materialized views in the OLTP as well as the OLAP environment is to store pre-computed data [189]. A typical use case in the OLAP environment is to efficiently implement the multidimensional structures for

analytical cubes [87] or the aforementioned report tables. Depending on their type, materialized views are updated upon changes to the base tables, which introduces overhead to the write operation. The overhead of the update process, also called view maintenance, should not outweigh the benefits achieved by creating and using the materialized view.

Clustering

Clustering of tables is another approach to improve data retrieval for a particular use case. Similar to storing a table in an index-like structure as described above, clustering influences the actual storage of a table. Records with the same value for a chosen attribute (or expression) are stored consecutively. Using more than one attribute as the basis for clustering is called multi-dimensional clustering [161]. In this case, the chosen attributes have to be orthogonal to each other to allow the creation of clusters. According to Lightstone and Bhattacharjee [128], multi-dimensional clustering is a powerful technique that offers significant performance benefits, especially for OLAP systems. OLAP queries directly profit from reduced I/O and CPU costs through their multidimensional nature.

Partitioning

Partitioning refers to cutting a table into pieces: either horizontally into subsets of complete tuples or vertically into subsets of complete columns. Horizontal partitioning has been introduced to improve the manageability for tables with many tuples. If partitions of a table are located on different machines, an additional benefit is increased availability as only part of a table becomes unavailable if a machine fails [57]. Vertical partitioning improves the performance of read access queries that need only a small set of the columns of a table. For databases that use a row-oriented physical storage layout partitioning according to query access reduces the overhead of reading columns that are not necessary for a query.

All of the above mentioned physical database design alternatives have been subject to research regarding the automation of design decisions. The question of how to determine an optimal storage scheme reaches back almost as far as the development of the relational model itself. Choenni et al. [36] define an optimal storage scheme as “the storage scheme which has the lowest cost in processing the workload defined on the database.” Because of the exponential complexity of determining an optimal storage scheme, research has shifted to determining a “good” storage scheme. A storage scheme is defined as “good” if an experienced human database designer would produce the same or a worse scheme with the same information content [36].

A variety of prototypes and tools exist that propose configurations for indexes and materialized views, for example, AutoAdmin for Microsoft SQL Server [31], or that automate partitioning, see Autopart for large scientific databases [162]. Zilio

et al. [227] introduced DB2 Design Advisor, which takes into account all of the above aspects of physical database design.

2.2.4 Database Storage

In addition to physical database design, aspects related to the data storage of relational databases influence query performance. Data storage is specified by where and how the data is stored. Aspects are the medium where the primary data set is located and the resulting data transport path to the processor as well as the layout of data in the primary location. The primary data location is the space where data is stored. During query processing, data is read from the primary location if it is not cached in a different possibly faster location. If changes occurred, data is written back to the primary location or is cached and written back later. Today's research and productive OLTP and OLAP database systems can be categorized into disk-resident and in-memory databases (IMDBs) [68] referring to the primary data location and two major storage layouts, the row-oriented and column-oriented storage layout, have evolved [2].

Disk-Based versus In-Memory Data Storage

IMDBs are less complex than disk-resident databases the main reason being that disk I/O as a mechanical process is avoided [71]. On disk, data is stored in blocks. If data is requested in a query, the block(s) containing the data need to be loaded from disk into main memory from where further processing takes place. Blocks that are read from disk are buffered temporarily in main memory. As a result, subsequent accesses to data already in memory avoid accessing the disk, thus speeding up operations. Other strategies are in place to avoid or reduce disk I/O. For example, if a block is accessed, other blocks that are likely to be accessed in the future can be pre-fetched from disk. Thus, subsequent queries for data from these blocks do not require any disk access.

Simply avoiding the disk access step by putting a disk-resident database into main memory can speed up operations. Yet, this strategy does not lead to the same speedup a pure IMDB achieves [68]. In the case of using a faster primary storage, such as RAM disks, the reason for the lesser speedup is that the algorithms tuned for disk access are still in place. For example, data is still accessed as though it was located on disk and is copied to a main memory area for further processing. This step is superfluous as the database already resides in main memory, but the entire process is faster nonetheless, as data is loaded from main memory and not from disk.

In the case of a completely cached disk-based database, data resides in main memory in the same structure as if it was just loaded from disk, that is, in blocks. However, data structures optimized for in-memory accesses instead of disk accesses achieve further improvement fully cached disk-based databases do not benefit from

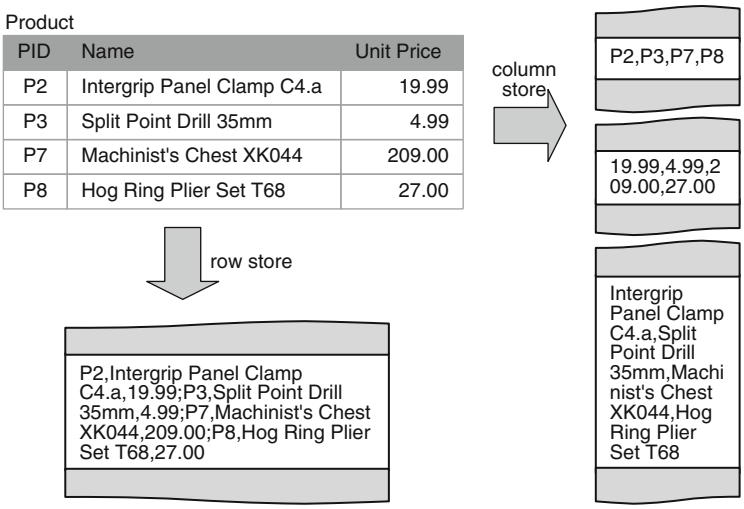

Fig. 2.13 Example for the layout of data in row and column stores

without modification. An example is the storage of data consecutively instead of in blocks. In IMDBs, logs are kept in non-volatile storage, such as, flash storage. Snapshots of the database are also written to this non-volatile storage in regular intervals to avoid loss of data.

Row versus Column-Orientation

The records of a table are stored consecutively in row-orientation. In column-orientation, the attribute values of a column are stored consecutively, meaning that each column of a table is stored separately. Figure 2.13 shows the *Product* table from the customer order example from Fig. 2.7 in row and column-oriented storage.

The benefit of row-orientation lies in operations such as adding new tuples to a table and look-ups of a set of attributes of a single tuple or a small number of tuples. Adding an entire tuple in a row store entails an address look-up for the location to store the record and a sequential write of the record data. In a column store an address look-up and a write is needed for each attribute value of the record [89]. The behavior is similar during a look-up of a set of attributes from a tuple or from a small number of tuples. The set of requested attributes for each tuple has to be reconstructed from the separately stored columns in a column store, whereas they can simply be read sequentially in a row store. Column stores benefit operations that access only a subset of columns from a wide table over a large set of tuples [1]. Examples of such behavior are summary operations such as *displaying the number of purchases per month, product category and city for the first quarter of 2011*, which are typical for analytical applications. In this operation, only the columns actually needed have to be transmitted to the CPU for processing in a column

store [2]. In the row store, data from columns that are not needed is read if the requested attributes are not stored consecutively in the record. Reading data in chunks the size of the CPU cache lines that are usually larger than single attribute values causes this.

Column stores are advantageous in scenarios where the schema has to adapt to changing requirements, that is, adding a new column to an exiting table or changing a column, for example to fit larger values. Columns can be added to a table or existing columns can be changed, for example increasing the length of a text field, without having to touch data in the other columns. In contrast, increasing the size of an existing column or inserting a new column in a row store requires rearranging the storage of the entire table to adapt to the changed record.

Compression is another aspect where column stores prevail over row stores. The values in a column are of the same data type, which is beneficial for compression. Additionally, Krueger et al. [124] have shown in their analyses of enterprise systems that frequent occurrences of the same value are common for the majority of columns in a table. As a result, a better compression ratio in column stores is possible [1]. This facilitates (a) better utilization of bandwidth constraints when transporting data to the processor if results can be computed on compressed data or data is decompressed as late as possible and (b) higher utilization of storage space, which is especially relevant for IMDBs as memory space is still more expensive than disk space.

Until recently, the vast majority of commercial databases used for transaction processing as well as analytical processing followed the row-oriented storage model. By now, column stores and hybrid engines that offer column and row-oriented table layouts have taken hold in the OLAP domain (cf. Feinberg and Beyer [59]).

2.3 Summary

In this chapter, the basic concepts underlying transactional and analytical processing from the data model and database design perspectives were provided. Several data models to support transaction processing were highlighted. Experiences from different data models have led to the development of the relational model and relational databases with it, which are today's most prevalent model in enterprise data management for transactional as well as analytical purposes. According to Lindsay, "their [relational databases] adoption in business, government, and education has enabled the progress we've made in productivity and without these tools to manage the complex affairs of government, business, education, and science our progress would be really much slower." [223] As has been discussed in this chapter, relational databases are the dominant choice for OLTP as well as OLAP. Thus, the remainder of this work focuses on relational databases as the foundation for OLTP and OLAP. Relational database design variants and optimization strategies commonly applied in OLTP and OLAP have been presented and discussed.

Chapter 3
Benchmarks for Transaction and Analytical Processing Systems

As presented in Chap. 1, the goal of this thesis is to analyze and compare the behavior of databases in mixed workload scenarios as a basis to evaluate logical database design decisions. Benchmarks provide a method for this. A benchmark is "a standardized problem or test that serves as a basis for evaluation or comparison (as of computer system performance)." [141]

This chapter provides related work concerning the creation of a mixed OLTP and OLAP benchmark. It gives an overview of existing benchmarks in the area of databases and discusses their applicability in the context of this work and the simulation of mixed workloads.

Section 3.1 discusses the workload characteristics along which transactional and analytical processing workloads have been differentiated from each other. The borders of this classification are blurring because of the continuous adaption of business operations to stay in competition or keep a competitive advantage. As an example, the time elapsed between data being entered into a system and needed for analytical purposes is decreasing, while the amount of data processed is constantly increasing. Transaction and analytical processing systems are required to keep pace and are thus continuously improved. With the change of the domains they are designed to model, benchmarks have to be adapted as well.

Section 3.2 starts with a classification of benchmarking activities in computer science and continues with the focus on database benchmarking for the areas of transaction processing and analytical processing. Recent developments in hybrid benchmarks for OLTP and OLAP are discussed afterwards. In Sect. 3.3, key criteria that are the basis to create valuable benchmarks are presented and discussed in relation to the established benchmarks. This chapter closes with an overview of measures used in benchmarks to compare database systems.

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_3,
© Springer-Verlag Berlin Heidelberg 2014

3.1 Transaction Processing Versus Analytical Processing

Modeling and optimizing a database depends solely on the workload the database is servicing. Transaction and analytical processing workloads have been serviced by the same system until the early 1990s. When both processing workloads became more sophisticated and started to conflict each other's operation, applications were classified into operational and analytical [61] and separate systems started to evolve specializing in either workload. Transaction and analytical processing and the systems servicing them have diverged largely, but recent developments bring forth efforts that again focus on both workloads in the same system. In the following, the characteristics of applications of both workloads will be described. What spurred the development of reuniting them and its implications will be discussed.

3.1.1 OLTP and OLAP Workload Characteristics

Business applications are distinguished by their workload into operational and analytical applications. The type of workload is seen as the key consideration in tuning a database system [58]. Therefore, many efforts have been undertaken to classify operational and analytical workloads along comprehensible attributes.

French [61] starts a high-level comparison of operational and analytical workloads by differentiating from an application usage perspective: Operational applications tend to be static over a long period of time (several years) once they are programmed and configured if legal requirements do not change. Thus, database access is determined by pre-defined queries that are embedded in the program code and are parameterized according to the business case. Analyzing sales order processing as a typical operational application indicates that the structure of operational queries is simple. Selects from the database only contain *select-from-where* clauses. *Insert* queries are composed of selecting information of one or a small number of specific products that are ordered and the actual insertion of new tuples that make up a sales order into the database. Operational queries are highly selective. A closer look at current database workloads shows that more than 50 % of all database operations are lookups [125], which indicates that only a very small number of tuples is needed from a table, for example, the specific products ordered, the address of the customer currently ordering. Due to their pinpoint access behavior, operational queries are very short concerning their run time. Since new data is constantly added to the database (new sales orders, incoming payments, recordings of outgoing deliveries), the operational workload is characterized by a mixed read-write access. Furthermore, operational applications require up-to-date data, for example, to determine if an ordered product is in stock to make a promise.

Analytical processing, according to French [61], is characterized by ad-hoc queries. Standard reports that are based on pre-defined queries, for example, period-end reporting, can entail further inquiries if results deviate from expectations.

These inquiries are often unforeseen and new queries have to be answered by the system. The queries in analytical reporting are complex in comparison to operational queries, including sub-selects, grouping, aggregation, and ordering clauses. An example of such a complex query is determining the top 10 % best-selling products concerning revenue by region and quarter of the last year, ordered by their share. This query also exemplifies that a larger set of data is accessed than in operational processing. Here, all sales orders of an entire year are touched in addition to general data of the top 10 % of all products that have been sold during that year. Selectivity is comparatively low. Because of additional operations like grouping and ordering, analytical queries can have longer run times compared to operational queries. Fresh data is inserted into analytical systems via batch jobs in time windows where interactive processing load is low and system resources can be utilized without affecting end users, i.e., at night. In a later work, French [62] adds other characteristics for the comparison of operational and analytical workloads. These are the number of concurrent users, which is comparatively high (1,000 users and more) in operational workloads and one order of magnitude less for analytical workloads.

Elnaffar et al. [58] propose a more technical set of attributes to distinguish operational from analytical workloads and use these to automatically classify a workload. Table 3.1 gives a summary of the distinguishing attributes of operational and analytical workloads according to French [62] and Elnaffar et al. [58].

The time dimension plays a significant role for analytics, whereas for transaction processing the current state is most important. Prepared data sets for analytics keep track of historical developments and changes applied to the data. Keeping track of changes to data is done in the OLTP environment as well. According to Helland [91], data in large-scale systems is not updated, but only new data is added or a new version is created. If data items were only updated without further processing, information would be lost, which is not a viable outcome in business processing.

3.1.2 Blurring the Border Between OLTP and OLAP

Applications exist that show access patterns typical for one domain, but that demand data and structures of the other. The border between OLTP and OLAP becomes increasingly blurred, as reports require more up-to-date data and transactions process larger sets of data. In the case of OLAP, reports exist that require up-to-date and detailed data for short term analyses and have a low selectivity. Examples are reporting on sales of the day at the end of business or monitoring of promotional activities. OLTP application examples are ATP, dunning runs [167], or payment runs. ATP has to scan through stock data, plan data, and promised data concerning a specific product to determine if the desired quantity of a product is deliverable at a given date. Dunning runs need up-to-date transaction data to determine outstanding payments and scan through a large set of data, that is, sales, billing, payments, and previous dunning records. To avoid strain on the OLTP system dunning and payment

Table 3.1 Operational and analytical workload characteristics (Based on [58, 62])

	OLTP	OLAP
Queries	Selects, inserts, updates	Mainly selects
	• Share of write access operations is higher in OLTP than in OLAP • OLTP requires higher locking effort, because of many write operations	• User interaction consists of selects • Fresh data from transactional systems is entered via batch updates
	Pre-determined	Ad-hoc
	Simple statements	Complex statements including sorts, grouping operations, and aggregations
	Short run times	Long run times
Freshness of data	Real-time updates, most current view on business operations	Varying degrees of freshness, depending on configuration of updates (e.g., real-time, daily, weekly)
Processed data	Small number of tuples	Large numbers of tuples are scanned and selected, e.g., because of loose time constraints like overview of an entire year
	High selectivity – OLTP queries have a high index usage ratio because of their selective access	Low selectivity
Load	Large number of concurrent users (1,000+) and thus large number of concurrent transactions, high requirements on throughput and timeliness in OLTP	Fewer concurrent users (100+) and a smaller number of concurrent queries

runs are executed as batch jobs dissecting the large set of data into subsets of business partners and processing them in parallel to speed up execution. The above application examples are processed in either the OLTP or OLAP system by introducing additional optimized storage or workarounds like the ODS or batch jobs. The ODS introduces another source of redundant data that needs to be managed and time windows for batch jobs become shorter in global enterprises calling for alternatives approaches to deal with these applications of mixed characteristics. These examples show that an exact border where OLTP ends and OLAP begins cannot be drawn.

In contrast to the classification and typical characteristics of OLTP and OLAP workloads given in Sect. 3.1.1, Krueger et al. [124] have found that 90 % of all queries executed in a commercial enterprise resource planning system, a widely employed OLTP system, are selects. The foundations of this analysis have been database log files of 65 companies that employ this particular system. The rationale behind this is that before each insert an application typically selects data from the database to create the context of a particular transaction. To create a new customer

sales order, product and customer data are, for example, selected from the database, an ATP check is triggered to determine the delivery date, etc., before the actual insert takes place. Thus, while write throughput has to be optimized in OLTP systems, the read-only performance is equally important.

IMDBs are widely used in analytical processing. Yet, they were not solely developed for this purpose and according to Garcia [67], database systems like SAP HANA [179], IBM solidDB [98], Oracle TimesTen [159], VoltDB [219], and eXtremeDB [140] are multipurpose in-memory databases that achieve faster response times for analytical applications, but also have the potential to change the way enterprises distinguish between operational and analytical data. Sikka et al. [190] state that the separation of OLTP and OLAP no longer reflects the state-of-the-art requirements of modern business applications and they demonstrate that a column store is suitable for high-scale transactional processing.

Research in database technology has also brought forth prototypes concerned with processing OLTP as well as OLAP workloads in one database system. Röhm [176] 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 [65, 116] handles OLTP and OLAP using a hardware-assisted replication mechanisms. This achieves consistent and up-to-date snapshots of the transaction data. Other prototypes such as OctopusDB [55] and HYRISE [80] aim at avoiding to keep multiple copies of data and instead adapt the storage layout of the data to the usage patterns in the workload.

3.2 Benchmark Classification

Benchmarks can be classified based on what is being evaluated. They can be divided into hardware and software benchmarks. Hardware benchmarks can further be divided into ones that evaluate *systems* in their entirety, *component benchmarks*, and *micro-benchmarks*. The system benchmarks take the system to be tested as a black box and measure based on inputs and the outputs as the reaction. Component benchmarks test isolated but complex parts of a system, e.g., a hard drive. Micro-benchmarks analyze basic components of a system, for example, the performance of the floating-point operation of a CPU.

Micro-benchmarks are also found in software benchmarking, for example, to compare alternative implementations of an algorithm. Further software benchmark categories are *system software benchmarks* and *application software benchmarks*. Application software benchmarks compare the performance and/or functionality of applications or application bundles. System software benchmarks provide the basis for evaluation and comparison of software that services applications and is not necessarily directly used by an end user. Database benchmarks, which are the basis for the elaborations and methodologies presented in this thesis, belong

Table 3.2 Categories for computer system benchmarking

Main category	Sub category	Description
Hardware	Component	Assessment of specific parts of a computer system, i.e. microprocessor, hard disk drive, or graphical processing unit
	System	Assessment of a system in its entirety
	Micro-benchmark	Measure the performance of a very small and specific piece of hardware
Software	Application software	Compare performance and functionality of specific applications or application bundles, e.g. word processing, mail clients/servers, business software
	System software	Compare performance and functionality of software that services applications, e.g. operating systems, virtualization software, database systems
	Micro-benchmark	Measure the performance of a very small and specific piece of software

to this category. Table 3.2 gives an overview of the mentioned benchmarking categories for computer systems.

Two major organizations that offer standardized benchmarks in the area of enterprise applications today are the Transaction Processing Performance Council (TPC) [213] and the Standard Performance Evaluation Corporation (SPEC) [195]. Both are non-profit organizations that comprise hardware and software vendors and research organizations as members, who can influence the development of the benchmarks. SPEC provides hardware as well as software benchmarks, but none are directly concerned with databases or transaction processing systems. TPC aims its attention directly at database benchmarks in the area of software benchmarks. Some vendors develop benchmarks specifically for their products to evaluate different configurations of the software and hardware to help their customers making a decision which hardware and software configuration to use in their specific cases. Examples are the Oracle Applications Benchmark [160] and SAP Standard Application Benchmarks [178].

3.2.1 Transaction Processing System Benchmarks

During the early development of RDBMS, only a few application-specific benchmarks existed instead of standard database benchmarks. These were unsuitable to evaluate the performance of the major components of relational database systems in general. Furthermore, they were difficult to understand and, thus, not straightforward to use [52]. From an effort to measure the speedup characteristics of their own RDBMS, Bitton et al. [11] developed a benchmark that became known as the Wisconsin Benchmark. This benchmark was the first widely used method

to compare RDBMS and it changed the market by pointing out weaknesses of database systems and forcing vendors to significantly improve their products.

The TPC has developed several benchmarks. At the present, the benchmarks TPC-C [209] and TPC-E [210], targeted at transaction processing systems, are widely used by database and hardware vendors to compare their products in that domain. The Telecommunication Application Transaction Processing (TATP) Benchmark [99] is an open source workload also targeting vendors of RDBMS and designed for high-throughput applications.

TPC-C

TPC-C was introduced in 1992 and comprises a set of basic operations representative of complex OLTP applications environments to exercise system functionality [209]. It includes different transaction types, a more complex database structure. The simulated business scenario is that of a wholesale supplier company that uses geographically distributed sales districts and associated warehouses.

The TPC-C database schema consists of 9 tables that contain between 3 and 21 attributes with 92 attributes in total. To evaluate systems according to the requirements of companies of different sizes, TPC-C scales along throughput, which is driven by the activity of customers connected to each warehouse. If larger systems need to be evaluated, more warehouses and their associated customers have to be configured.

TPC-C provides a mixture of read-only and update-intensive transactions that range from entering, delivering, and checking the status of orders, simulating payments, and monitoring warehouse stock levels. All transactions are executed in a pre-defined mix. Order status, delivery, and stock level each make up at least 4 % of the workload and the payment transaction at least has a share of 43 %. No such share exists for the order transaction as the benchmark metric is based on the throughput of this transaction in addition to the workload created by the other transactions in the background. Furthermore, the system should achieve specified response time goals. 90 % of all transactions should complete in 5 s or less, except stock level look-up, which may take up to 20 s.

The benchmark metric for performance comparison is number of orders processed per minute (tpmC). The second metric, called *$/tpmC*, is used for system cost comparison related to performance based on a 3 year cost of ownership.

TPC-E

TPC-E, released in 2007, is the successor of TPC-C with the goal of providing an enhanced and more complex schema and workload to better simulate the characteristics of modern OLTP systems [94]. TPC-C and TPC-E currently coexist. The reason is that TPC-C is still very popular because of its wide distribution and long usage. However, the adoption rate of TPC-E is increasing [150].

TPC-E simulates a brokerage firm and its interactions with customers and business partners. Two types of transactions are defined that comprise ten business transactions to simulate these relationships. The schema contains 33 tables with the widest table having 24 attributes, 21 of these tables contain only 5 attributes or less and the schema includes 191 attributes overall. The tables are divided into four sets to describe the customers, brokers, markets, and general data such as locations.

Similar to TPC-C, TPC-E scales along the number of customers for a brokerage firm. The metrics to compare results are a performance metric, i.e., transactions per second, called tpsE, also referring to the throughput of a specific transaction and a price performance metric like in TPC-C.

Telecommunication Application Transaction Processing Benchmark

The goal of the relatively new TATP benchmark is to evaluate database transaction systems in an extremely high throughput scenario. This benchmark is not a standard benchmark provided by a council of industry and research partners as is the case with the TPC benchmarks, but is used by industry and research already.

The workload of this open source benchmark simulates a home location register database of a mobile carrier. The database consists of 4 tables with the widest table having 34 attributes and the schema having 51 attributes overall.

With the help of seven transactions, subscribers and their actions are imitated. In the basic setting, the seven transactions are issued by ten independent clients. The transaction mixture is the same for all clients and composed of 80 % read transactions and 20 % write transactions. Read transactions include the retrieval of subscriber data with a share of 35 %, retrieval of access data (35 %), and getting the new destination of a subscriber (10 %). The write transactions include location updates (14 %), subscriber data updates (2 %), call forwarding insertions (2 %), and call forwarding deletions (2 %).

After a run, TATP yields the mean qualified throughput, that is, the number of transactions per time unit and response time distributions for all seven transactions.

3.2.2 Analytical Processing System Benchmarks

The TPC provides TPC-H [212] as its standard for testing analytical systems. TPC-DS [208] as a new benchmark has been proposed to replace TPC-H. The Star Schema Benchmark (SSB) by O'Neil et al. [156] takes the decision support benchmark TPC-H a step further in the direction of optimized analytical processing by deriving a pure star schema from the schema layout of TPC-H. Its goal is to evaluate the performance of data management systems built for pure data warehouse environments that require matching database schemas. Like TATP, SSB is not a standard benchmark.

TPC-H

Since 1999, TPC-H is in use to assess hardware systems and databases regarding their decision support capabilities. Similar to the scenario of TPC-C, TPC-H simulates the activities of a wholesale supplier. The database schema consists of 8 tables with the widest having 16 attributes and 61 attributes in total. TPC-H uses 22 analytical style queries to exercise the database and 2 refresh functions, which keep the analytical database up-to-date and clean up old data.

The metrics reported in TPC-H are composite query per hour, called QphH@Size, as the performance metric and TPC-H Price/Performance ($/QphH @Size) as the price-performance metric. Composite query per hour is computed from the results of two runs. The first run is the power test where queries are executed serially in a single session to determine the raw processing power of the system under test for each query. The second run is the throughput test where queries are submitted to the system in parallel sessions to examine the throughput boundaries.

Star Schema Benchmark

To create a pure star schema for the evaluation of star schema data warehouse queries, the two tables of TPC-H that contain data about the orders (*Lineitem* and *Order*) are denormalized into one fact table. Furthermore, text attributes are dropped from this new fact table, because in the denormalized form it takes significant storage space and cannot be used in aggregations. Three more tables are dropped to create the star schema out of the snowflake schema and to avoid periodic snapshots that are updated and that invalidate historical data. Thus, the SSB database schema comprises 1 fact table and 4 dimension tables with 58 attributes in total and the widest table having 17 attributes.

The SSB queries are based on some of the TPC-H queries, however many of those queries cannot be translated to the adapted database schema. The SSB queries are assigned to four groups, called query flights. Each query flight answers a specific business question and includes three to four statements with increasing selectivity. The performance metrics reported are the elapsed time for the query flights, CPU time, and storage consumption broken down for the different structures (tables, indexes, views).

TPC-DS

The rationale behind the work on TPC-DS (Decision Support) was to create a benchmark that is considerably more realistic than earlier benchmarks for analytical processing systems [170]. It provides a star schema with multiple fact tables and shared dimensions. The generated data is skewed and scales sub-linearly for non-fact tables. It provides a wealth of analytical queries of different complexity

classes to simulate a diverse workload including ad-hoc queries. TPC-DS has been released in January 2012 as the new decision support benchmark standard.

The business environment modeled in TPC-DS is that of a retail product supplier including customer, order, and product data. Its database schema consists of 7 fact tables and 17 dimension tables that are shared among the fact tables. The widest table contains 34 attributes and the database schema accounts for 425 attributes distributed across 24 tables.

Similar to TPC-H, TPC-DS includes user queries to simulate analytics and data maintenance operations to simulate updating the data warehouse from the operational sources. The user queries comprise 99 queries of 4 categories, which are reporting, ad hoc, iterative OLAP, and data mining queries [208]:

- Reporting includes queries that run periodically to answer well-known and predefined business questions.
- Ad-hoc queries are similar to reporting queries. The nature of ad-hoc queries in the real world is that they are not known beforehand and, thus, database optimizations are not possible because of this lack of knowledge. To simulate the lack of knowledge any optimizations that are specific to the ad-hoc queries in the benchmark are forbidden.
- Iterative OLAP queries include sequences of simple and complex statements that lead from one to the next. The statements contained in iterative queries are handled as ad-hoc queries and the database must not be optimized for them.
- Data mining queries analyze large sets of data, contain joins and aggregations and produce large result sets.

The metrics to report results are, like in TPC-H, query throughput (queries per hour at a specific scaling factor – QphDS@SF) as the performance metric and the price-performance metric ($/QphDS@SF). The scaling factor is a number by which the initial database sizes are multiplied to create a larger data set.

Although first ideas about TPC-DS have already been published in 2002 [170], it took 10 years until its announcement as the new decision support benchmark standard in January 2012 by the TPC.

3.2.3 Mixed Workload Benchmarking

In the database benchmark area, as can be seen above through the assignment of benchmarks to the OLTP or OLAP domain, benchmarks are further divided into domain-specific benchmarks. Gray [74, Chap. 1] stated that benchmarks needed to be domain-specific because application workloads are too diverse to optimize a system for all domains at once. However, the separation of OLTP and OLAP into two domains is subject to reevaluation, as motivated in Chap. 1.

Workload of a resource can be defined as the amount of work and the types of requests that are assigned to the resource in a given period. Concerning OLTP and OLAP, this can be broken down to the types of queries that take place and their frequency of execution in the database.

A mixed workload in this thesis is defined as the total workload composed of diverse sub-workloads. The diverse sub-workloads in this case are an OLTP and an OLAP workload. In a mixed workload of OLTP and OLAP, the two contribute to the total workload and the share of each sub-workload has to be defined. As currently no productive system exists that is subject to a mixed workload, only the sub-workloads can be defined realistically, e.g., by using traces from real systems.

Existing Mixed Workloads Related to OLTP and OLAP

Mixed workload has been a subject in benchmarking as well as in productive systems even after the separation of OLTP and OLAP into two domains. TPCs transactional web e-Commerce benchmark TPC-W [207], which has been marked as obsolete since 2005, but is still in use, explicitly models different workloads of its basic queries to reproduce diverse user behavior. TPC-W models an Internet commerce environment and simulates the queries encountered in a web shop. Three profiles are provided that vary the ratio between browsing and ordering products, that is, between read-mostly access and write-intensive access. The given profiles are named browsing, shopping, and ordering. In browsing 95 % of the queries are read-only (browsing) and 5 % cover ordering activities. The shopping mix is comprised of 80 % browsing and 20 % ordering, while the ordering mix contains equal shares of ordering and browsing activities. Similar to the mix of OLTP and OLAP queries these mixes cover conflicting optimization goals, e.g. fast access of a large amount of data during browsing versus providing fast insertion of data during ordering.

Like in the example of TPC-W, the contribution of the OLTP and OLAP sub-workloads to the total workload should not be constant in a mixed OLTP and OLAP benchmark as is the case with queries contributing to the workloads of the current benchmarks. This is the case because transactional and analytical processing systems follow conflicting optimization goals and consequently the share of OLTP and OLAP-style queries has an impact on the decisions to optimize a combined system.

The development of ODSs shows how a mixed workload of simplified OLTP and OLAP operations is handled by productive systems. As described in Sect. 2.1.2 there are four classes of ODS, with three of them being copies of transactional data that can be categorized according to data freshness and the fourth type of ODS additionally including strategic information created by a report in the data warehouse and copying the results to the ODS. On top of the ODS different types of users produce a mixed workload. Inmon characterizes the users as “farmers” and “explorers” [103]. Whereas farmers perform the same task repeatedly and exactly know what they are looking for, 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 is that farmers and explorers operate on only a limited amount of data copied

into the ODS to produce reports. They are concerned with only a short period of time, e.g. total sales of the day or the number of new customers. The focus of composite OLTP and OLAP systems, however, lies on systems that contain the entire operational data set of a company for transactional processing and analytics and not just a snapshot.

A Hybrid OLTP and OLAP Benchmark

Latest activities in benchmarking include the creation of benchmarks for composite OLTP and OLAP systems to assess their performance. Funke et al. [66] have recognized the lack of a hybrid benchmark and introduced TPC-CH in March 2011. It is a combination of the standard benchmarks TPC-C and TPC-H. TPC-CH includes the complete database schema and queries of TPC-C without modifications and adds the *Nation*, *Region* and *Supplier* tables from TPC-H to accommodate all of the TPC-H queries. The TPC-H queries have been adapted to fit the new schema without changing their semantics from the business point of view. As a result, TPC-H consists of 22 OLAP queries and 5 transactions, 12 tables with 106 attributes in sum. According to the authors [66], TPC-CH results are comparable with those of TPC-C. TPC-CH has been renamed to CH-benCHmark in subsequent works [45, 64].

Table 3.3 gives an overview of the previously described benchmarks, their database schema, data set, queries, domains, and their year of introduction for non-standard benchmarks respectively the year of standardization for the standard benchmarks. In comparison with each other, the complexity regarding the number of tables, table width and number of queries varies enormously. However, no conclusion should be drawn from this alone, as depending on the area of application a more or less complex benchmark may be desirable.

3.2.4 Other Database Benchmarks

Benchmarks exist that work with the productive systems directly and thus incorporate realistic workloads, for example, the SAP Standard Application Benchmarks. These analyze the entire database and application stack and they are not focused on direct and independent assessment of the database management system. Nonetheless, they present a basis to extract close to reality table structures and workloads for a composite benchmark.

Other database systems exist that augment or compete with relational databases, but relational databases continue to dominate what most people think of as a database system [131]. Examples are object databases, document databases, graph databases, or spatial network databases (cf. [131] for a detailed description of these databases). Along with the development of these databases, benchmarks have been established, e.g., the OO1 [25], OO7 [23], and the HyperModel [5]

Table 3.3 Overview of existing benchmarks

	TPC-C	TPC-E	TATP	TPC-H	TPC-DS	SSB	CH-benCHm.
Domain	OLTP	OLTP	OLTP	OLAP	OLAP	OLAP	Mixed OLTP & OLAP
Standard	Yes	Yes	No	Yes	Yes	No	No
Scenario	Wholesale supplier	Brokerage firm	Mobile carrier	Wholesale supplier	Retail product supplier	See TPC-H	Wholesale supplier
Schema	3NF	3NF	3NF	3NF	Star schema	Star schema	3NF
#Tables	9	33	4	8	24	5	12
Widest table	21	24	34	15	34	17	21
Narrowest table	3	2	5	3	3	7	3
#Columns	92	191	51	61	425	58	106
#Tables updated	8	20	3	2	16	1	8
Data generation	Non-uniform	Uniform	Uniform	Uniform	Skewed	Uniform	Non-uniform
Data set	Synthetic	Pseudo-real	Synthetic	Synthetic	Mostly realistic	Synthetic	Synthetic
#Queries	5	12	7	22	99	13	27
Year	1992	2007	2004	1999	2012	2007	2011

benchmarks as de facto standards for object databases [49], or are being established like in the case of graph databases [56]. Benchmarks for other database types than relational databases are not further discussed in the context of this thesis as the focus lies on enterprise data management. In this area, relational databases are prevalent.

Other work streams are present in connection to relational databases where existing benchmarks or self-made micro-benchmarks are applied to determine if and how the implementation of database components has to be changed to exploit technological advances. For instance according to Petrov et al. [164], flash solid state disks (SSDs) are a disruptive technology that has the potential of changing the established principles in database system architecture. In recent work, they evaluated the sizing of database pages [165] and the performance of database query processing algorithms [8] against the background of using SSDs as primary database storage. Instead of using the aged TPC-C benchmark or self-made micro-benchmarks, the benchmark proposed as part of this thesis provides a complex business processing workload that can serve as a basis for further analyses in these directions.

ETL as the process to synchronize fresh data from the operational system to the analytical system is part of the OLAP landscape. Proposals of benchmarks and implementations have arisen, e.g., the so-called “data intensive integration process benchmark” for evaluating the performance of integration systems [20]

or the RODIN High Performance Extract/Transform/Load Benchmark [44] that mainly measures load performance, but no industry standard benchmarks exist. The need has been acknowledged [215] and the TPC formed a working group to assess the purpose and scope of an ETL benchmark [224]. ETL benchmarks will not be discussed further in this thesis since the traditional ETL process as such is not required in a hybrid OLTP and OLAP system and data integration from several sources for analytics is beyond the scope of this thesis.

According to Cecchet et al. [26], database performance evaluation is about more than peak throughput. The necessity arises to assess performance in the presence of failures, in degraded modes of operation, or under low loads. Benchmarks need a notion of the difference of peak workloads and regular workloads as systems react to failures very differently when resources are fully utilized. Since operational database systems do not usually run under high loads all the time, existing benchmark workloads are not applicable for testing failure behavior. A benchmark that goes beyond measuring peak performance throughput has been introduced by Vieira and Madeira [216]. The so-called DBench-OLTP is an OLTP benchmark and it is based on TPC-C. It focuses on measuring performance and availability in the presence of failures by inducing faults that are based on typical administrator mistakes. Furthermore, Vieira and Madeira [217] provide a comprehensive overview of research activities and working groups in the area of dependability benchmarking. For this thesis, the simulation and handling of failures is an orthogonal dimension and not in scope.

3.3 Key Criteria for the Value of Benchmarks

In his benchmark handbook, Gray [74, Chap. 1] introduces four criteria that make a domain-specific benchmark useful. These are:

- **Relevancy** – Comprises measuring the performance of systems with the help of typical operations that a system in a specific domain is exposed to.
- **Portability** – It means that the benchmark should be easy to implement on a variety of different systems and architectures.
- **Scalability** – To measure evolving systems and different system sizes a benchmark has to be scalable.
- **Simplicity** – A benchmark has to be easily understandable.

Newer work revisited this question of what the overall requirements for benchmarks are. Huppler [96] lists the following five key aspects and expresses that not all of the criteria have to be fulfilled in perfection, but most successful benchmarks are strong in one or two of these aspects.

- **Relevancy** – The results should reflect important issues of the domain.
- **Repeatability** – The benchmark should be able to run multiple times yielding the same result.
- **Fairness** – All compared systems are able to participate equally.

- **Verifiability** – The result of the benchmark is perceived as real.
- **Economic** – The benchmark run should be affordable.

Compared to Gray's initial criteria, being economic and repeatable are new aspects. Fairness can be interpreted to cover scalability and portability. Verifiability can be seen as a further specification of simplicity: Focusing on the results of the benchmark and for them to be perceived as real, the scenario of the benchmark has to be understandable (see simplicity) and relevant for the domain.

Sachs et al. [177] and Vieira et al. [218] summarize based on these criteria that to be useful and reliable a benchmark must fulfill the following criteria: It must be based on a workload that is representative of real-world applications. It has to stress all key aspects of the target platform. To be fair it must not be tuned or optimized for specific database products. Its results have to be reproducible and its scalability must not be limited. Stonebraker [197] argues that to create a benchmark (1) a pressing need and (2) a simple application should be found and that (3) the focus should be on the vendor community to provide better systems.

3.3.1 Established Benchmarks and the Benchmark Properties

As companies continuously adapt to stay ahead of competition, their workloads are adjusting as well. If benchmarks drop behind these developments, they lose relevance [94]. An issue of existing benchmarks is that they do not represent the complexity of the actual database schemas used in today's systems recording daily operations, but an ideal version: According to Nambiar and Poess [148], real database schemas comprise a considerably larger number of tables and attributes. As was previously shown in Table 3.3 the tables in the existing benchmarks are relatively small compared to what Krueger et al. [124] observe in their study of enterprise system databases. They make note of database tables having 100 attributes and more. Even if data is sparsely distributed over such a large amount of attributes and 75 % of all attributes have a small number of distinct values, depending on the database architecture the mere existence of a column in a table – filled or not – can influence query performance.

Hsu et al. [95] argue that workloads in production exhibit a wider range of behavior than is reflected by the TPC benchmarks. Later, a report by Forrester Research further emphasized this line of argumentation with the statement that "TPC benchmarks no longer reflect the complex workloads of today's real-world deployments." [226] Although they still exercise critical database functionality, their simple table structure and the aforementioned objection against the realism of the simulated workload are exclusion criteria that oppose the usage of TPC benchmarks as the basis for creating a new method for the evaluation of mixed workload systems.

The hybrid CH-benCHmark, described in Sect. 3.2.3 as an approach to benchmark mixed OLTP and OLAP systems, uses the TPC-C tables with two additional tables from TPC-H and thus shares their weakness of relying on the comparatively

narrow tables provided by these benchmarks. Furthermore, both benchmarks are relatively old: TPC-C was first introduced in 1992, TPC-H in 1999. The age of a benchmark in itself is not a negative criterion if the benchmark is continuously adapted. In the almost two decades between the introduction of TPC-C, applications and their characteristics have changed. This observation led to the introduction of TPC-E as a new OLTP benchmark and supposed successor of TPC-C. Yet, despite the fact of TPC-C's aging database schema and workload, it is still widely used for database system and hardware evaluation and comparison.

Regarding the economy criterion, Poess and Floyd [169] state that the cost in resources to produce a benchmark can easily become substantial, making it a difficult or even impossible undertaking for a small company. This statement refers to the TPC-W and TPC-C benchmarks, but the other benchmarks are no less complex. The member companies of the TPC belong to the largest database and hardware vendors in the industry. According to DeWitt [222], benchmark results for specific database products are usually produced by the vendor and/or hardware partner producing a performance result that is guaranteed to not be exceeded by the actual system implemented at a user's site. Another issue is the narrow-minded focus of vendors on optimizing their products for particular benchmarks while there are database product vendor's restrictions on third parties to publish own benchmark results [222].

3.3.2 Benchmark Measures

An important aspect constituting to the value of benchmark results are the measures that are actually reported from a benchmark. The main measure in existing benchmarks, as can be seen from Sect. 3.2, is the maximum throughput at which a certain percentile of transactions does not exceed a given maximum response time. In addition, the cost of the system hardware and maintenance for a certain period is specified in a second measure based on the throughput measure. These metrics are generally accepted and used in today's standard benchmarks.

According to Thomasian [205], more detailed measures are needed to compare database systems, because the key performance measures in relational database benchmarks only reflect the ability of the benchmarking team to tune the system, the speed of hardware, and the efficiency of the software. Yet, they are missing lower level parameters such as page references or buffer hit ratio, to break down the performance according to database components. In early work, Bitton and Turbyfill [10] went into a similar direction proposing more fine-grained measures to compare database systems regarding processing times and system utilization and using them in the Wisconsin benchmark [11]. The measures are classified into speed indices and utilization indices. Speed indices, according to Bitton and Turbyfill include:

- Response time of queries
- Response time of queries as a function of the workload
- Throughput of the system as a function of the workload

The second class, utilization indices, includes measures such as CPU utilization and communication line utilization. The distinction between the two classes of measures is that the speed indices provide an absolute measure of the performance of database systems and utilization indices provide a basis to understand where the bottlenecks are. The proposal was that these indices should be evaluated against a range of well-defined workloads with workload of a database system being a function of available indexes, the size of the produced output, and the ratio of read-only and update queries.

In today's benchmarks, however, only a single workload is tested and the system is tuned to maximum performance. In contrast to the just-mentioned fine-grained metrics, only high-level measures are reported for comparison, which are the throughput and price-performance metrics.

Energy prices continue to escalate, but energy efficiency presents an often-untapped opportunity to increase profits. In data centers, for example, the life-cycle energy costs exceed the costs of purchasing the equipment [144]. The SPEC and TPC benchmark organizations have reacted and another measure has been added reporting on energy consumption in relation to performance. SPEC's energy benchmark "SPECpower_ssj2008" [193] is the first industry-standard benchmark to measure power and performance characteristics of server-class compute equipment. It provides an overview of a server system's energy efficiency based on simulating Java server applications. SPEC also introduced the Server Efficiency Rating Tool (SERT) for the evaluation of overall energy efficiency not focusing on capabilities of computer servers in specific application areas or business models like benchmarks [194]. The TPC introduced the TPC-Energy specification [211] based on SPECpower. It is applicable to all its current benchmarks and provides a measure composed of the consumed energy in relation to the throughput and time needed. The TPC energy specification is a supplement to the existing TPC benchmarks to report the extra measure.

3.4 Summary

The characterization of OLTP and OLAP workloads at the start of this chapter leads to the insight that the border that separates the two workloads is increasingly blurring. Many applications already exist that comprise characteristics of both workloads according to the previous characterization. These suffer from the separation of both domains and workarounds are undertaken to enhance their execution, for example, pre-computing data, or execution in batch mode during low system load times to reduce their impact on the rest of the system. Based on the assumption that the trends of real-time, mass data processing, on-the-fly computation and ad-hoc reporting will not break off, the number of applications not cleanly fitting into either domain will only increase in the future. This calls for a change in enterprise data management and its traditional segmentation of transactional and analytical processing systems.

In this chapter, existing benchmarks in the area of transaction and analytical processing were reviewed. Standardized benchmarks, ad-hoc standards, and research benchmarks exist for both application domains. The separation of the two domains has led to the fact that the focus of the existing benchmarks lies on either OLTP or OLAP.

With the CH-benCHmark, a research initiative started investigating hybrid OLTP and OLAP benchmarks for the evaluation of the new database systems in March 2011. This new benchmark is based on the existing TPC-C and TPC-H benchmarks. A benchmark has to adhere to a set of criteria to be of use. It has to be relevant, repeatable, fair, verifiable, and economical. Companies adapt to market situations to remain competitive resulting in changes to their workload. If benchmarks are not adapted according to these changes, their relevancy diminishes. Several researchers have already expressed their concern that existing benchmarks are too simple or do not reflect the current enterprise workloads any longer. Against this background, a new hybrid OLTP and OLAP benchmark developed based on existing benchmarks manifests the same weakness if the changing character of enterprise workloads is neglected.

Part II
Towards a Benchmark for Mixed Workloads and Its Application in Evaluating Database Schemas

Chapter 4
Combined Transaction Processing and Reporting Benchmark

Only limited statements can be made concerning the ability of existing data management systems to handle a mixed OLTP and OLAP workload since they have so far been treated as separate domains and separate benchmarks were created. Existing benchmarks could be applied to a combined architecture for OLTP and OLAP by simply running the benchmarks in parallel. This would only lead to a partial picture of the actual performance of such a system measuring the effects of hardware resource contention as the benchmarks are running on their own distinguished sets of tables. Furthermore, conflicts and opportunities arising from data access of the different operations on the same tables are of particular interest in mixed workload situations. The load on the system obviously increases in a mixed workload scenario compared to two single workload systems. Yet, the data accessed in OLTP and OLAP operations is the same business data just in differently optimized structures. Thus, operations accessing this same data in a common structure can benefit from each other, for example, through multi-query optimization [183], simultaneous pipelining [88], or cooperative scans [126, 228]. Usage of such strategies does not improve performance unrestrictedly and models have to be created to find an optimum like proposed by Johnson et al. [113]. Similar to the creation of new hybrid systems, a relevant and representative workload to test, adapt, and tune new strategies and models is needed here as well. A hybrid benchmark based on observations from current enterprise systems can provide such a workload.

In this chapter, such a benchmark to analyze and compare the performance of systems under mixed workload situations is introduced. Section 4.1 presents the steps taken to create the hybrid benchmark. According to the first step, Sect. 4.2 starts with a discussion on relevant scenarios for OLTP as well as OLAP in the enterprise context and choosing the order-to-cash process as the underlying scenario of the benchmark. The order-to-cash process will be examined in further detail in this section and the underlying data model is provided and explained. In addition to the data, a benchmark needs a representative set of queries to exercise the database. These are presented in Sect. 4.3. Afterwards, details are provided on

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_4,
© Springer-Verlag Berlin Heidelberg 2014

how the benchmark can be adapted to analyze different system sizes and workloads. Section 4.5 relates the newly introduced benchmark to the benchmark criteria of creating a valuable benchmark according to the discussion in Sect. 3.3.

4.1 Creation of a Hybrid Benchmark

Three alternative ways to create a mixed OLTP and OLAP workload benchmark come to mind: (1) the combination of two existing benchmarks like the approach taken in the CH-benCHmark (see Sect. 3.2.3), (2) the extension of an existing benchmark with the respective other workload, and (3) the creation of a completely new benchmark.

Regarding the first approach, the underlying data schemas of the different benchmarks differ and, thus, would need to be integrated to create a data schema for both workloads. As shown in the CH-benCHmark efforts this approach is feasible. However, as was discussed in Sect. 3.3, the tables of the used standard benchmarks are relatively simple compared to the tables in real enterprise systems. Workloads in enterprises adapted to market situations and thus evolve and change. If benchmarks are not adapted accordingly, they lose relevancy because they do not reflect the real workloads any longer. This has to be born in mind when relating benchmark results to real world systems. The question how relevant a new benchmark, which is based on aged database schemas and workloads, is remains. The same arguments apply to the second approach. Based on the observation that established benchmarks have aged and do not reflect current enterprise workloads any longer, the third alternative is chosen to create the benchmark proposed in this chapter.

Figure 4.1 provides an overview of the steps that followed the initial decision to create an entirely new benchmark. The steps taken to develop the new benchmark are based on the requirement to comply with the previously discussed benchmark criteria (Sect. 3.3).

The simplicity criterion is covered by implementing a scenario that is easily understandable as it is part of every-day life. At the same time, this scenario has to fulfill the relevancy criterion, i.e., it has to provide enough complexity to model realistic workloads as observed in today's enterprise systems.

As the next step once the scenario is fixed for the benchmark, the database schema underlying the chosen scenario needs to be defined. To achieve a realistic setting, the database schema is extracted from a real OLTP enterprise system that is widely used. For this database schema, a pseudo-realistic data set is created based on the workload of a model company and analyses of data distributions within several companies.

SQL traces conducted in real OLTP systems and their adaptation to the chosen scenario lead to the definition of the OLTP workload part of the new hybrid benchmark. The OLTP queries are extracted including their inter-transaction dependencies. For example, an invoice can usually only be sent to the customer if

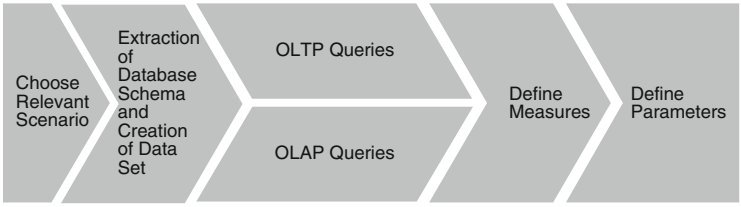

Fig. 4.1 Steps of benchmark creation

all ordered items have been delivered. Therefore, the queries recording the delivery of all the line items that will be invoiced have to be executed before the transaction creating the invoice is executed in the system.

In the same way, the reporting queries relevant in the order-to-cash process are extracted from the data warehouse and the ODS, which is a part of the OLAP system of the enterprise. Queries from the ODS are extracted to include the so-called operational reporting queries, which in contrast to the analytical summary queries need data on a very detailed level, most suitably line item level. These queries are using similar structures as the OLTP system. As a result, only little adjustment in query design is needed to create the operational reporting queries on top of the extracted OLTP database schema. The analytical summary queries are defined based on interviews with managers, who are responsible for strategic decision-making.

The benchmark measures are specified in accordance with the established benchmarks. Finally, data set size, load, and workload mix are defined as the variable parameters of the new benchmark that can be used to adapt the benchmark to the requirements of a specific company or test scenario. As a result, the Composite Benchmark for Transactions and Reporting (CBTR) introduced in this thesis closely resembles a real-world scenario using pseudo-realistic enterprise data.

4.2 The Benchmark Scenario

A large variety of scenarios exists in the enterprise world that can be the basis for a new benchmark. Existing benchmarks relied on scenarios that are part of everyday life, such as order processing and stock trading, and as a result are easily comprehensible as proposed in Gray's benchmark criteria (cf. Sect. 3.3) to gain wide acceptance and credibility. From the variety of application areas within the enterprise world, e.g., marketing, sales, distribution, or accounting, one needs to be chosen for a benchmark scenario according to this simplicity requirement. Figure 4.2 gives an overview of the application areas for transactional enterprise systems. From left to right, the systems are depicted according to their place in the value chain. Some application areas, such as finance, accounting, or warehousing are spanning across other transactional systems with business processes

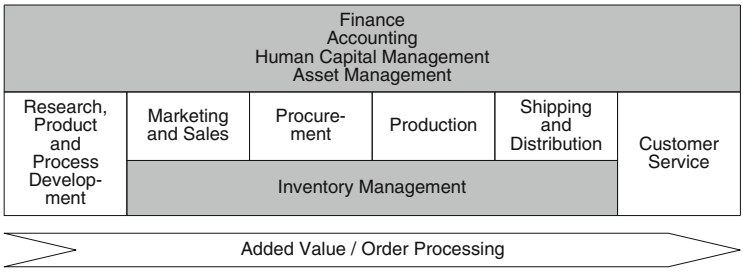

Fig. 4.2 Application areas in transactional systems (Based on [142, Chap. 1])

transcending the borders of application areas. Typical application areas used in existing benchmarks are sales and inventory management.

In a company, the process cycles order-to-cash, purchase-to-pay, and order-to-distribution can be differentiated [166]. Order-to-cash focuses on selling a companies products and purchase-to-pay deals with obtaining all materials or products needed to keep the company running, e.g., purchasing raw material for production or furniture and office material. The first manages revenues and the second manages the expenses. Order-to-distribution is concerned with the management of the supply chain, which includes all activities involved in delivering a product to a customer starting from the raw material [134].

Order-to-cash and purchase-to-pay include parts of financial accounting. Financial accounting collects all financial transactions of a company to retain an overview of operations and to create the financial reports needed, e.g., year-end closing reports. Therefore, the actions in order-to-cash and purchase-to-pay also find their representation in accounting. Both processes, order-to-cash as well as purchase-to-pay, including financial accounting provide a good basis for modeling representative queries from the OLTP as well as the OLAP domain. Order-to-distribution comprises sourcing, manufacturing, inventory, and distribution activities and is a core part of business operations. Strategic planning of future revenues and the order-to-cash process it is based on is the main lever companies use to encourage their business. The operations within the order-to-cash and purchase-to-pay processes are similar, but targeted in opposite directions. Consequently, one of the two scenarios is sufficient for usage within a new benchmark.

For the proposed benchmark, the order-to-cash process and the related part of accounting are used as the basis. This scenario is important for daily operations, e.g., incoming orders, outgoing invoices, incoming payments, and finding late payments to trigger dunning. It is equally important for reporting, e.g., analyzing the levels of order fulfillment, determining all open orders of customers to ensure timely deliveries, or calculating the average processing time of an order for validating the own service levels and keeping customer satisfaction at a high level. Furthermore, it is easily comprehensible since the process of sales, invoicing, and payment is omnipresent in everyday life.

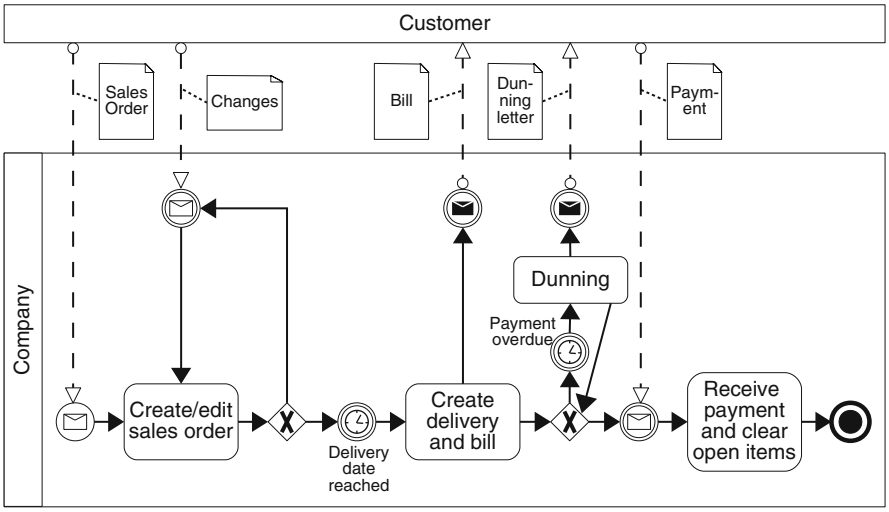

Fig. 4.3 Simplified order-to-cash process

4.2.1 The Order-to-Cash Process

From the transactional processing side, the process underneath the order-to-cash process consists of several steps. Figure 4.3 gives an overview of the simplified order-to-cash process in Business Process Modeling Notation [154]. Multiple actors take part in the process with the main actors being the customer ordering products and the company selling the products and processing the order. In the first step the customer announces his desire to order products of the company, thus a sales order is created. In this sales order the ordered items, their quantity and further information such as delivery date and shipping address of the customer are specified.

The second step contains the creation of the delivery documents for the ordered items and delivering them once the delivery date is reached. After the delivery, or in most cases at the point of delivery, the items will be billed and billing documents are created accordingly, which is the third step. In the last step, either incoming payments are recorded in time of the given payment period and the respective open items are cleared, or the time allowed for payment is up and a dunning letter will be sent. Dunning letters can be created several times, each time increasing the dunning level according to the overdue time and previous dunning actions. The process of an order ends when the payment is received.

From the analytical point of view, various questions are of strategic interest in this scenario. These contain analyses of the success of sales such as the comparisons of products in different categories, sales districts, and periods, or the sales performance of teams and individual sales people. Sales analysis provides the basis for forecasting of future sales based on what customers wanted in

the past. Another aspect includes process optimization and customer satisfaction questions, for example, end-to-end order processing time, order completion, that is, the percentage of orders completed in time, and reducing the average amount of accounts receivable by analyzing and speeding up cash collection. In the following, the conceptual data model containing the most important entities that take part in the order-to-cash process is introduced.

4.2.2 Conceptual Data Model and Database Schema

The entities taking part in the order-to-cash process can be classified according to the process step they are primarily taking part in. The process steps are (a) ordering, (b) delivery, (c) billing, and (d) accounting. Figure 4.4 gives an overview of the entities within this scenario.

Master data, which is depicted in gray shading includes data about the products offered, sales organizations as administrative units of the company that offer products, for example in a specific area or line of business, business partner data, location and contact information of business partners and sales organizations. Business partner is a general term for parties connected with the sales process at the customer's side. Sold-to-party and ship-to-party are typical examples. All other data entities contain transaction data, which is changed frequently during business processing. This includes the sales orders that reference specific products, their deliveries, invoices for delivered orders and accounting information, which covers the financial view of the sales process.

Figure 4.5 depicts an overview of the database schema as taken from a real enterprises transactional system. It shows the excerpt of the tables used in the order-to-cash process, their basic relationships, and the most important attributes. The number of columns within the tables of this original schema is quite large, varying between 5 and 327 columns. The total number of columns is 2,316, at least an order of magnitude more than in the standard benchmarks discussed in Sect. 3.2. One reason for this large amount of attributes is the occurrence of redundant values and pre-computed aggregates in the original data set. This is an optimization so far used to speed up data access and to avoid joins. Because of the large number of columns in the database schema, only those accessed most often in the transactions and queries, which are going to be introduced in the following sections, are depicted in Fig. 4.5.

A general pattern, which can be derived from this database schema, is that each set of tables contains a header and line item table. This conforms to the setup of a database in second normal form. Header tables contain only general information, e.g., creation date and information applicable to the entire set of line items belonging to it like the buyer who initiated the sales order. General information of the header depends on an identifier, e.g. *order ID* as the identifier of a sales order header. Detailed information stored in the line item tables is fully dependent on the entire

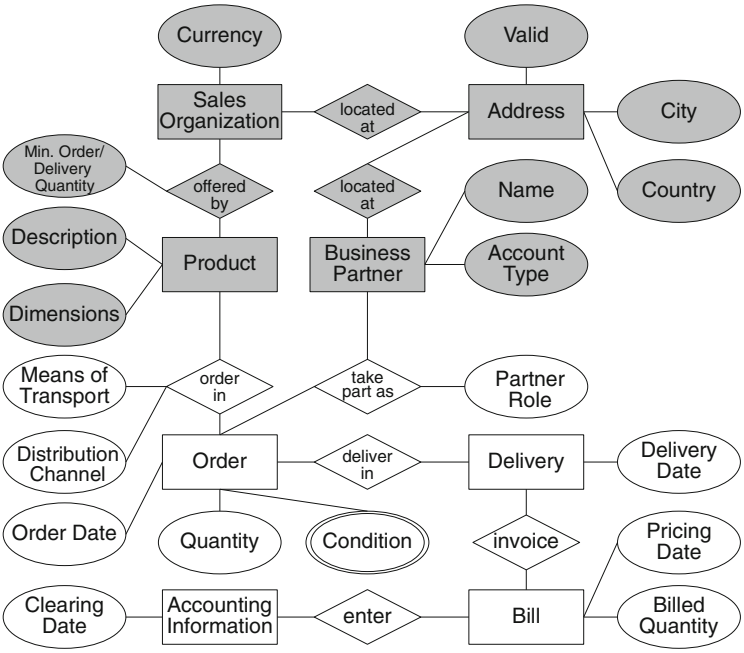

Fig. 4.4 Conceptual data model

primary key, which in this case would be a composite key of the header identifier the item belongs to (e.g. *order ID*) and a position identifier (e.g. *order item ID*).

What breaks the third normal form in this schema is the fact that header as well as item tuples contain non-key columns that are mutually dependent. To give an example, the sales header table contains a reference to the customer via the identifier of the customer, but also additional data about the customer's assignment to specific groups. This additional data solely depends on the customer identifier and is independent of the sales order itself. Similar behavior can be observed in the sales item table that contains references to products via their identifiers and a description of the product. Finally, the header tables are not in second normal form either because they contain summary data about their line items such as the summarized net value of a sales order or the aggregated weight, and volume of ordered products. Thus, the database schema of CBTR is in accordance with the first normal form (1NF).

The header and item pattern mimics the structure of documents, e.g., a sales order document, or a bill. Such documents consist of a document header and the line items. A line item belongs to exactly one header and a header contains at least one line item. In the following, the tables and their most important attributes will be described.

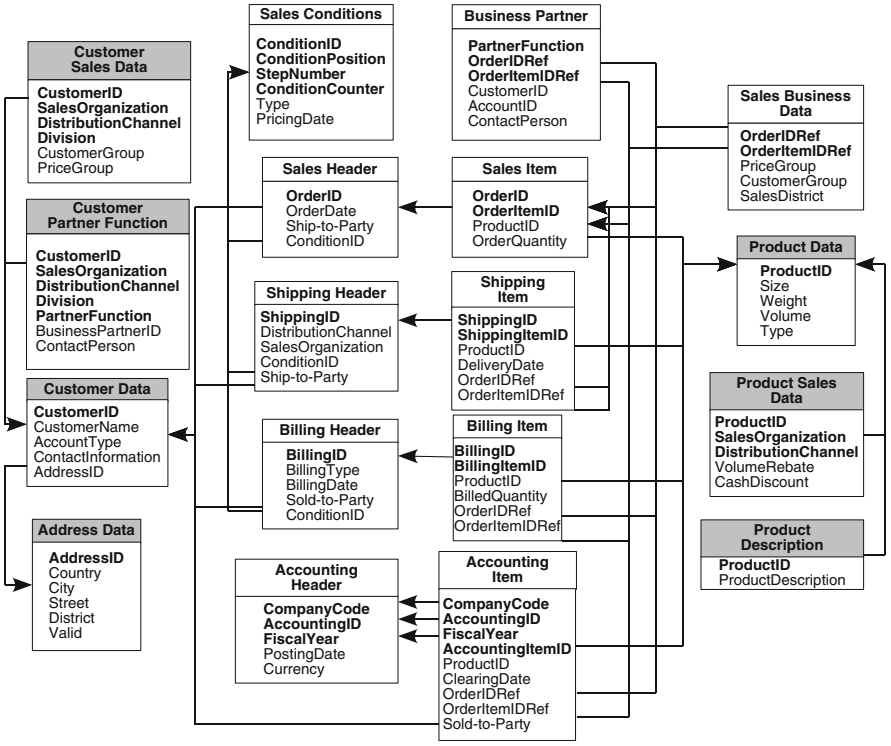

Fig. 4.5 Database schema in first normal form

Information stored in the sales document header table is the order date, requested delivery date, sold-to-party, and sales organization. A record in the sales document line item table includes the ordered material, ordered quantity, allowed deviation from the ordered quantity, weight, and volume information. The business partner relation contains additional information about parties with specific roles and their contact information. Different roles can be sold-to-party, ship-to-party, payer, or vendor of a material. Consequently, if the ship-to-party is given on line item level, differing from the information given in the header, the header will be overridden for this case. The sales business data table holds data for each line item only if a deviation from the data given in the header occurs, such as pricing, billing, invoice dates, and payment methods.

Records of the delivery header table include shipping and receiving point data, what type of delivery will be applied, when the goods are going to be picked, moved and loaded, and ship-to-party to mention the most important. Additional information stored in the delivery line item table is the material group, plant, storage location, the actual quantity delivered, material availability date, and redundant weight and volume information when compared to the sales order tables.

Redundancy such as the weight and volume information being stored again within the different sets of tables referring to the different process steps can be observed at many points of this particular database design. Chmura and Heumann [35] describe that a reason for denormalizing a normalized database is to improve performance. Joins add a significant overhead to computing performance and denormalization reduces the number of joins.

Billing document headers hold information about billing type, billing category, shipping conditions, customer group, payment method, and destination country. Billing types are for example invoice, cash sale, rebate correction. If a bill is created from a previously created sales order, billing category is, for example, "order related billing document". Another example is periodic billing documents that are automatically triggered. Billing document line item data contains the actual billed quantity, business area, date for pricing and exchange rate, and references to the according sales document and delivery item. The condition table, amongst others, has information about condition type, and condition pricing date. For example, the condition type specifies, whether during pricing, a price, a discount, a surcharge, or other pricing elements, such as freight costs and sales taxes are applied.

Incoming payments and clearings of items in open bills (open items) are recorded in the financial accounting tables. Accounting document header data includes the company code, fiscal year and period of the booking, posting date, an inter-company posting procedure number, currency key, and exchange rate to name a few. The accounting document line item, also called accounting document segment, includes the clearing date (if cleared, else empty), account type, debit or credit indicator, and tax code.

The different table sets are connected via their line items. The path of a line item can be traced from the sales order, over the delivery to the final payment. As a result, no one-to-one relationship exists between a sales order document, a delivery document, and the billing document. In fact, a delivery document either may span several sales order documents or may be only a partial delivery, meaning that the line items of one sales order document can be split over several deliveries. This is also true for billing documents where a customer may trigger several orders but wants to pay in one go after all of the orders have been delivered. The alternative, where a billing document covers only part of a delivery is rare but also possible.

Master data is referenced throughout the scenario in all transactional tables. Master data tables are customer detail and product detail tables. Customer detail tables contain the name information of the customer, his contact data, e.g., addresses, and contact persons in case of companies, account type, e.g., global or local. Customers or business partners may play different roles in the order-to-cash process. They, for example, act as buyer, sold-to-party, ship-to-party, consultant, carrier, or intermediary. Product master data contains specifics about the sold products, e.g., weight, volume, dimension data, references to constituents or ingredients, the plant where production takes place, but also about the sales organization offering that product, and the minimum order and delivery quantity (minimum package size) if applicable.

4.3 The Benchmark Queries

In this section, the OLTP and OLAP queries that contribute to the mixed workload of CBTR will be introduced. Furthermore, statistics about the contributions of the OLTP transactions as they are based on the operations of a real enterprise system and the table access profiles of all queries are provided.

4.3.1 Transactions

The transactional side of the benchmark consists of queries with mixed read and write access, as well as queries with read-only access. Transactions with mixed access are the following:

1. Create new sales order
2. Create shipping document
3. Create billing document
4. Record payments and clear open line items

The read-only OLTP transactions are:

- Show detailed sales orders
- Display sales orders of a given period
- Show open items in a given period
- Display customer details
- Display product details

Create New Sales Order

For the creation of a new sales order, the following data has to be entered:

- Order type, commonly used types are standard order, returns, delivery free of charge
- Sold-to party, ship-to party and related data like shipping address
- Order date
- Requested delivery date
- Payment method

Some of this data, for example, order date, shipping address or payment method is selected from master data or default values are initially chosen that can be changed by the user entering the sales order into the system. Thus, data entered by the user is completed by lookups in the general customer data table and general material data table. Descriptions, weight, volume and dimension information is retrieved from the material table and customer contact information is fetched from the customer table. In case a customer is not maintained in the system, a new entry has to be created.

```
Order header:
 Selection from
  Customer data on customer ID
  Customer sales data on customer ID
  Customer partner function on customer ID, division,
      sales organization, distribution channel
  Address data on address ID
 One row insertion into sales header
For each line item:
 Selection from
  Product data on product ID
  Product description on product ID
  Product sales data on product ID
 One row insertion into sales item
 Optional insertion into sales business data
 For each specified role:
  One row insertion into business partner
```

Listing 4.1 Transaction profile for sales order transaction

For each sales order line item, the ordered product and the order quantity have to be entered. To complete the sales order, a sales order number is assigned automatically, the creation date is saved, and the default value for a delivery date, e.g., the next business day after the sales order has been created if no other date has been specified, is inserted.

The transaction profile for creating a new sales order is shown in Listing 4.1. It includes the selection of context data from the customer and product master data tables, the creation of a new row for the order in the sales header table and the creation of a row for each ordered item in the order item table. Business partner data about the parties taking part in this transaction and optional information per sales item about the sales process, for example, customer group, pricing conditions, means of transport, or rebate can be stored as well.

Create Shipping Document

Shipping documents can either be created automatically based on sales orders (one-to-one relationship) or manually. If created manually, a check against the sales order tables has to succeed, that all items to be delivered have been ordered beforehand and are sent to the same receiving party. Line items in a shipping document may have been ordered in different sales orders. Therefore, a reference is inserted to the according sales document line item in the shipping item table.

The transaction profile for the shipping transaction is shown in Listing 4.2. The transaction comprises selecting context data from the sales item table and inserting new data into the shipping header and shipping item tables.

```
Shipping header:
 One row insertion into shipping header
For each shipping item:
 Selection from sales item for referencing the sales
    order line item on order ID and order item ID
 One row insertion into shipping item
```

Listing 4.2 Transaction profile for the shipping transaction

Create Billing Document

Billing is processed in multiple alternative ways: a billing due list can be created and processed automatically, e.g., at a certain time each day, or a billing document can be processed manually based on a work list. The creation of billing documents includes a check if all referenced line items have been delivered before issuing the bill. In case of the manual creation, the billing type has to be entered and if necessary, the reference to a sales order document has to be provided.

Furthermore, the creation of billing documents triggers the creation of the respective documents in the accounting system. Initially, the line items are marked as open items. Open line items are those items that have been delivered and billed, but not yet been paid for. The transaction profile of the billing transaction, including forwarding information to financial accounting is specified in Listing 4.3.

Again, context data is retrieved from the transaction data tables of sales and shipping that has been inserted in previous process steps and new rows are inserted into billing and accounting header and item tables. If additional conditions are specified, for example a discount for a good customer or an alternative price is provided, new rows are added to the sales conditions table including references to all associated items of the sales, shipping and billing tables.

Record Payment and Clear Open Line Items

Clearing of open line items includes checking open line items from the database and finding those that are referenced by an incoming payment document. The respective line items are closed by insertion of the clearing date and the clearing document number. The query is composed of the selection of the items to be paid and updates of them, see Listing 4.4 for the transaction profile.

Show Detailed Sales Orders

Since displaying complete documents is similar for sales orders, delivery, billing, and accounting documents, the necessary steps are described using sales order documents as an example. Sales order documents to be shown can be selected

```
Billing header:
 One row insertion into billing header
 For each specified condition:
  One row insertion into sales conditions
  One row insertion into accounting header
For each billing item:
 Selection from
  Sales item on order ID reference and order item ID
     reference
  Shipping item on shipping ID reference and shipping
     item ID reference
 One row insertion into billing item
 One row insertion into accounting item
```

Listing 4.3 Transaction profile for billing transaction

```
Selection from accounting header on accounting ID,
   company code, fiscal year
For each accounting item:
 One row update of accounting item, adding the clearing
     date and billing ID of the clearing document where
     fiscal year, accounting ID and company code match
    the selected accounting header tuple
```

Listing 4.4 Transaction profile for clearing open items

```
SELECT OrderID, OrderItemID, ProductID, ProductText,
   OrderQuantity, ConfirmedQuantity, SalesUnit,
   NetValue, Currency, SoldToParty
FROM SalesHeader AS sdh, SalesItem AS sdi
WHERE sdh.OrderID = @DocumentNumber
 AND sdi.OrderID = sdh.OrderID;
```

Listing 4.5 Show sales order by key

by key, which is the document number. Listing 4.5 provides an example for the SQL statement. Here, the sales header and sales item tables are joined on document number to retrieve attributes of the sales order residing in the two tables.

Display Sales Orders of a Given Period

Access by key, however, is a rather infrequently used case and may happen directly after a sales order has been created and the automatically created sales document number is still shown. A case more often used is displaying today's, yesterday's

```
SELECT OrderID, SoldToParty, OrderDate, NetValue,
    Currency
FROM SalesHeader
WHERE CustomerID = @CustomerID
 AND OrderDate BETWEEN "01.10.2011" AND "31.10.2011";
```

Listing 4.6 Show sales orders by period

```
SELECT ah.CustomerID, ah.AccountingID, ah.CompanyCode,
    ah.FiscalYear
FROM AccountingItem AS ai, AccountingHeader AS ah
WHERE ah.PostingDate BETWEEN "01.09.2011" AND
 "30.09.2011"
 AND ah.CompanyCode = ai.CompanyCode
 AND ah.AccountingID = ai.AccountingID
 AND ah.FiscalYear = ai.FiscalYear
 AND AccountType = "Debitor"
 AND ClearingDate= "␣"
ORDER BY ah.CustomerID, ah.AccountingID;
```

Listing 4.7 Show list of open items

or last week's documents, or showing recent documents for a certain customer. Listing 4.6 gives an example SQL statement that retrieves sales orders created in October 2011. After this query, the provided identifiers are used to retrieve details of the selected sales orders.

Show Open Items in a Given Period

For dunning, a list of delivered and billed, but not cleared items has to be created from the database, from which the items to be dunned can be chosen. Selection criteria if applied (one may also show all open items) may be the date, sales regions, or certain customer groups. The dunning level is determined referring to the payment due date and previous dunning actions. See Listing 4.7 for an example SQL statement showing all open items of September 2011. Items, for which payment has not yet been received have an empty clearing date by which in addition to *Debitor* as account type can be filtered. The result is ordered by customer and accounting document identifier.

Display Customer and Product Details

Displaying customer and product details, selecting customers by name, and products by name or description includes a simple join over the respective master data

```
SELECT cd.CustomerID, cd.CustomerName, ad.City, ad.
    Country
FROM CustomerData AS cd, AddressData AS ad
WHERE cd.CustomerID = @CustomerID
 AND cd.AddressID = ad.AddressID;
```

Listing 4.8 Display customer details

```
SELECT p.ProductID, p.ProductType, p.IndustrySector,
   pd.ProductDescription
FROM ProductData AS p, ProductDescription AS pd
WHERE pd.Language = "English"
 AND p.ProductID = @ProductID
 AND p.ProductID = pd.ProductID;
```

Listing 4.9 Display product details

tables. An example for the SQL statements is given in Listing 4.8, selecting customer data by ID using a join of the customer data and address data tables, and Listing 4.9, selecting product data by ID from the product data table and product description table.

4.3.2 Analytical Queries

For analytics, the benchmark includes queries that are used for strategic reporting and queries showing operational reporting behavior. Through parameterization, mainly the period of the analysis, the queries can be varied between operational reporting and strategic reporting. The set of queries for the benchmark includes:

1. "Daily flash" reporting
2. Average order processing time
3. Order delivery fulfillment
4. Days sales outstanding

Daily flash reporting is composed of two queries that are executed intra-daily and can be seen as purely operational reporting. The second and third queries collect statistics that provide the basis to ensure customer satisfaction. If these numbers are known, a company can set thresholds to provide early interventions in case improvement is needed, heading in the direction of proactive instead of reactive decision making according to Frolick and Ariyachandra [63]. Days sales outstanding is a strategic reporting query about customer payment behavior.

```
SELECT sdh.OrderDate, cd.Country, sdh.Currency, SUM
   (sdh.NetValue) AS SalesRevenue
FROM SalesHeader AS sdh, CustomerData AS cd
WHERE sdh.OrderDate = @today
 AND sdh.SoldToParty = cd.CustomerID
GROUP BY sh.OrderDate, cd.Country, sdh.Currency
ORDER BY SalesRevenue DESC;
```

Listing 4.10 Daily flash query of sales revenue by country

```
SELECT sdh.OrderDate, sdh.SalesGroup, sdh.
   DistributionChannel, sdh.Division, SUM(sdi.Quantity)
    AS OrderQuantity, sdi.SalesUnit
FROM SalesHeader AS sdh, SalesItem AS sdi
WHERE sdh.OrderDate = @today
 AND sdh.OrderID = sdi.OrderID
GROUP BY sdh.OrderDate, sdh.SalesGroup, sdh.
   DistributionChannel, sdh.Division, sdi.SalesUnit
ORDER BY OrderQuantity DESC;
```

Listing 4.11 Daily flash query of sales quantity by sales unit

Daily Flash Reporting Query

Daily flash reporting is of increased importance, for example, in the seasonal retail business, where managers constantly need to keep track of daily sales in order to see how seasonal sales are progressing and to introduce last-minute marketing campaigns if necessary. Daily flashes include sales revenues, number of incoming sales orders, and comparisons, e.g., by region, district, and department. Example queries for sales revenues and sales order number queries are given in Listings 4.10 and 4.11.

The query in Listing 4.10 compares sales revenue by customer region and retrieves the revenue of sales orders from the sales header table and groups the results by region using location data stored in the customer data table. Listing 4.11 shows a query that compares order quantities by sales unit using data from the sales header table for the grouping and aggregating the quantities from the sales item table. The results are ordered by sales revenue in case of Listing 4.10 and by order quantity in case of the query in Listing 4.11.

Average Order Processing Time Query

This query determines the average time spent for the entire order processing starting from the date, the sales order is created till the last item has been delivered to the

```
SELECT T.SalesGroup, T.DistributionChannel, T.Division,
    AVG(DATEDIFF(T.DeliveryDate, T.OrderDate)) AS
   ProcessingTimeDays
FROM (
 SELECT DISTINCT sdh.SalesGroup, sdh.
    DistributionChannel, sdh.Division, sh.DeliveryDate,
     sdh.OrderDate, sdh.OrderID, sh.ShippingID
 FROM SalesHeader AS sdh, SalesItem AS sdi,
    ShippingHeader AS sh, ShippingItem AS si
 WHERE sdh.OrderDate BETWEEN "01.07.2011" AND
   "30.09.2011"
  AND sdh.OrderID = sdi.OrderID
  AND sdi.OrderID = si.OrderIDReference
  AND sdi.OrderItemID = si.OrderItemIDReference
  AND si.ShippingID = sh.ShippingID)T
GROUP BY T.SalesGroup, T.DistributionChannel,
   T.Division
ORDER BY ProcessingTimeDays DESC;
```

Listing 4.12 Average order processing time

customer over a given time window. This query can either be computed for sales orders of a specified period, e.g. created in the last quarter, or on the entire data set. In the example in Listing 4.12, the average order processing time is determined for the third quarter of 2011 with the help of a built-in function that retrieves the number of days between two dates. The inner select retrieves and filters the tuples that are the basis for the aggregation, grouping the result by sales group. Four transaction tables are joined to create the connection between order date and delivery date. The result is finally sorted by the computed order processing time.

Order Delivery Fulfillment Query

Order delivery fulfillment calculates the percentage of sales orders that have been shipped completely and on-time for a specified period grouped, for example, by sales group or region. The query is similar to the previous one regarding the accessed tables. An example is given in Listing 4.13.

The outer selection counts the number of delivery line items for sales orders created in the specified period, whose actual delivery date is equal or before the requested delivery date stated in the sales order header. The inner selection counts all delivery items that exist for orders in the previously specified period. In the outer query, four transaction tables are joined to create the connection between the order and the final shipment of the items. For the count of total ordered items, the inner selection accesses the two main sales order tables. The result is sorted by the delivered quantity.

```
SELECT sdh.SalesGroup, sdh. DistributionChannel, sdh.
    Division, sdi.SalesUnit, SUM(si.DeliveredQuantity)
    AS Delivered, (
 SELECT SUM(sdi.OrderQuantity)
 FROM SalesItem AS i, SalesHeader as h
 WHERE h.OrderDate BETWEEN "01.07.2009" AND
   "30.09.2009"
  AND sdh.SalesGroup = h.SalesGroup
  AND sdh.DistributionChannel = h.DistributionChannel
  AND sdh.Division = h.Division
  AND sdi.SalesUnit = i.SalesUnit
  AND i.OrderID = h.OrderID) AS Expected
FROM SalesHeader AS sdh, SalesItem AS sdi,
    ShippingHeader AS sh, ShippingItem AS si
WHERE sdh.OrderDate BETWEEN "01.07.2009" AND
  "30.09.2009"
 AND sh.DeliveryDate <= sdh.DeliveryDate
 AND sdh.OrderID = sdi.OrderID
 AND sdi.OrderID = si.OrderIDReference
 AND sdi.OrderItemID = si.OrderItemIDReference
 AND si.ShippingID = sh.ShippingID
GROUP sdh.SalesGroup, sdh. DistributionChannel, sdh.
    Division, sdi.SalesUnit
ORDER BY Delivered DESC;
```

Listing 4.13 Order delivery fulfillment query

Days Sales Outstanding

Stewart [196] describes days sales outstanding (DSO) as a measure that captures the ratio of accounts receivable to daily sales. DSO provides a measure in "days" for the outstanding receivables, thus characterizes customer payment behavior. The DSO report query is executed for a specific period, e.g. for a quarter, where the average amount of time that elapses between the billing of items until the payment is collected from the customer is determined. The formula to calculate DSO is[1]:

$$DSO = \frac{Total\ receivables\ i.p.}{Total\ credit\ i.p.} * Period\ days$$

With $Total\ receivables\ i.p. = Total\ credit\ i.p. - Received\ payments\ i.p.$, DSO can be calculated as follows directly, without the need to store intermediate results:

$$DSO = \left(1 - \frac{Received\ payments\ i.p.}{Total\ credit\ i.p.}\right) * Period\ days$$

[1] in period = i.p.

```
SELECT ai.Currency, (1 - SUM(ai.Amount) / (
 SELECT SUM(bh2.NetValue + bh2.TaxAmount)
 FROM BillingHeader AS h
 WHERE h.BillingDate BETWEEN "01.07.2011" AND
   "30.09.2011"
  AND h.Currency = ai.Currency)*91 AS DSO
FROM AccountingItem AS ai, BillingHeader AS bh
WHERE bh.BillingDate BETWEEN "01.07.2011" AND
  "30.09.2011"
 AND ai.ClearingDate <> "␣"
 AND ai.AccountType = "Debitor"
 AND ai.Indicator = "Debit"
 AND ai.AccountingID = sh.BillingID
GROUP BY ai.Currency
ORDER BY DSO DESC;
```

Listing 4.14 Days sales outstanding query

Table 4.1 CBTR query statistics

Query	Share	Ratio of header to line items
Sales order	30 %	98 % of sales orders have between 1 and 4 line items
Shipping	27 %	98 % of shipping documents have between 1 and 4 shipping lines
Billing	25 %	96 % of bills have between 1 and 4 line items, and 95 % of all bills have between 1 and 20 additional sales conditions defined with 11 conditions per bill occurring in 65 % of all cases
Clearing	18 %	98 % of accounting documents have between 2 and 12 line items

To select the factors to calculate DSO, the billing and financial accounting tables have to be accessed. The example for the query in Listing 4.14 selects received payments (outer selection) and total credit (inner selection) and computes DSO, where a quarter is chosen as the period of interest (91 days). The result is sorted by the computed value for DSO.

4.3.3 CBTR Query Shares and Database Access

To provide a realistic scenario, the queries that modify the database have been recorded from a real enterprise system and are simulated based on that workload. An overview of the shares of each transaction in this workload and coarse statistics regarding the characteristics of the underlying data set based on real company data are given in Table 4.1.

Table 4.2 provides an overview of accesses of all queries to the tables of CBTR. For each existing access to a table the type is given (Insert – “I”, Select – “S”,

Table 4.2 Database access of CBTR queries

			CBTR queries												
			Mixed OLTP				Read-only OLTP					OLAP			
			Sales order	Shipping	Billing	Payment	Sales order by key	Sales order by period	Open items	Customer details	Product details	Daily flash	Order processing time	Order del. fulfillment	DSO
Database tables	Transaction data	Sales header	I*				S5	S6				S5	S5	S6	
		Sales item	I*	S*	S*		S6					S3	S2	S4	
		Sales business data	I*												
		Business partner	I*												
		Shipping header		I*									S2	S2	
		Ship. item		I*	S*								S3	S3	
		Billing header			I*										S5
		Bill. item			I*										
		Sales conditions			I*										
		Accounting header			I*	S*			S5						
		Accounting item			I*	U2			S5						S6
	Master data	Customer data	S*							S3					
		Customer sales data	S4												
		Cust. partner funct.	S5												
		Address data	S*							S2					
		Product data	S*								S3				
		Product sales data	S6												
		Prod. description	S4								S3				

Update – "U") and the number of columns accessed ("*" means that all columns are retrieved for further processing). As an example, a complete row is inserted into the sales header table by the sales order transaction, marked by "I*", and five columns are read from the sales header table by the order processing time query, marked by "S5".

4.4 CBTR Measures and Parameters

Two performance measures are provided in CBTR, these are throughput and average response time. The two measures will be specified in more detail in this section. Afterwards, the scaling and workload mix parameters to adapt the benchmark to specific company requirements are described.

4.4.1 The Throughput and Response Time Measures

Throughput reports the number of queries completed within a given time window over a time span. As introduced in Sect. 4.4.3, CBTR comprises a mixed workload of transactional and analytical queries that is controlled via shares that can be configured. The throughput measure is divided into OLTP and OLAP throughput, reporting a measured result for both sub-workloads. This is necessary as response times vary widely between OLTP and OLAP and an average of both would obscure the actual behavior.

Response time is defined by the time that passes between the moment a client starts to send a request and the moment that the response is completely received from the system under test. The average response time can then be computed per query, query type (mixed, read-only OLTP, or OLAP), or workload type providing a detailed measure to assess and compare all components of the workload. In contrast to the throughput measure, a general response time summarizing both OLTP and OLAP is not recommended as the actual behavior of the different queries and their reaction to varying database and load configurations will just be smoothed away.

The price/performance and energy measures as are provided by the standard benchmarks are not discussed here and out of scope for this work. The proposed benchmark is primarily used to assess system performance and evaluate the impact of database design decisions and database architectures on the performance of OLTP and OLAP queries.

4.4.2 Scaling

The results of a benchmark should be relevant for and applicable to small companies and local business as well as globally operating companies with thousands of employees using the IT systems. One size for a benchmark does not fit or simulate the requirements of companies of all sizes. Scaling is an important aspect of benchmarking to solve this problem. Through scaling, databases of different sizes can be created. Using scaling, systems of different sizes can be assessed and workloads of companies of varying dimensions can be simulated. Two aspects surface when looking at database scaling. The first is scaling the size of the database and the second is scaling the load within the database. In the following, both aspects will be discussed.

Scaling the Database Size

Scaling the size of the database refers to varying the size of the tables within the database. To provide a realistic benchmark, a method is needed that ensures realistic scaling of the database size. Looking at companies of different sizes and comparing

these, leads to the result that several strategies of scaling the data set exist. The larger the company, the more sales departments, for example, depending on regions, customer groups, or product categories, a company may have. Additionally, a larger company for example in retail has a higher turnover than a smaller company in the same branch of business. As a result, one strategy can be to increase the order throughput. Another strategy is adding more sales departments, while keeping sales per department constant and, thus, increasing the order throughput for the entire company.

The above remarks indicate that the tables within the data set should not be scaled equally. One difference can be seen between transaction data tables and master data tables. While transaction data tables grow with each business transaction, this is not necessarily true for master data tables. Business transactions refer to existing master data in most cases. For example, the sale of a product does not necessarily impose any change in its product master data. Only if new products are offered, master data has to be created accordingly. Likewise, the transaction tables grow at different rates. In Sect. 4.3.3 the shares of the queries within the proposed benchmark as observed from a real company system are shown. From these shares can be seen that the sales header table, for example, grows faster in its number of records than does the billing header table during normal transaction processing. The reason is that 30 % of all transaction insert one row into the sales header table, while only 25 % of all transactions insert one row into the billing header table. This imbalance in table sizes increases with an increasing database size and it should be modeled as well in the scale-up process.

For scaling up the database size and to keep it simple, CBTR relies on increasing the sales throughput without adding more sales departments, products, or customers. This results in the company having sold more products within the same period increasing the size of the transaction tables, while the size of the master data tables remains constant. Consequently, the sales header table is used as the base unit of scaling and the cardinality of all other transaction data tables is a function of the number of orders in the sales header table. Table 4.3 summarizes the cardinalities, number of columns, average row lengths and sizes of the tables of CBTR in their initial size as taken from a real system without scaling. Lengths and sizes given are examples that can vary with the implementation of data types in the database. The reference database, the numbers were taken from was MySQL MyISAM engine [147, Chap. 13] using UTF8 encoding.

Scaling the Load

Compared to scaling the database size, scaling the load is achieved in a simpler way. Instead of manipulating the database tables, increased load can either be generated by decreasing think times of clients or by increasing the number of clients accessing the database concurrently. Think time defines the time between receiving the result of one query from the database and sending the next query to the database. A reduction of think times increases the frequency with which clients send requests

Table 4.3 Initial database table cardinalities, average row lengths and initial table sizes

Table name	Number of rows	Average row length (bytes)	Table size (MB)	Number of columns
Sales header	4,566	2,066	9.4	124
Sales item	10,238	3,247	33.2	224
Sales business data	4,673	1,511	7.1	90
Business partner	23,147	477	1.1	24
Shipping header	9,729	2,963	28.8	169
Shipping item	16,718	4,014	67.1	281
Billing header	28,939	1,600	46.3	98
Billing item	42,717	2,804	119.8	194
Sales conditions	361,251	612	221.1	65
Accounting header	33,668	1.730	58.2	99
Accounting item	129,888	4,489	583.1	327
Customer data	7,881	5,342	42.1	165
Customer sales data	23,341	665	15.5	77
Cust. partner function	16,009	230	3.7	13
Address data	18,692	3,774	70.5	85
Product data	14,404	3,278	47.2	208
Product sales data	9,957	526	5.2	68
Product description	161,476	302	48.7	5
Total	917,294	–	1,408.1	2,316

to the database while the number of clients is constant. In CBTR, load scaling is achieved by varying the number of clients taking part in the simulation. Think times are not configured for the clients in CBTR. Clients concurrently send requests to the database, wait for the response and immediately send the next request upon receiving the result for the previous request.

4.4.3 Workload Mix

Data processing requirements for analytical processing are different compared to transactional data processing as discussed in Sect. 3.1 and can be seen in the CBTR queries introduced in Sect. 4.3. OLTP queries are characterized by simple, mixed read and write operations. More than 50 % of these operations are lookups [125]. These retrieve subsets of the columns of a table together. For example, descriptive attributes for a specific sales order and its line items with details are displayed or selected for updates. Furthermore, they are highly selective, meaning that one or only a small number of sales order documents and their line item entries are touched, instead of a huge set of them, e.g., showing all sales orders of a specific customer in January 2011. Figure 4.6 depicts an excerpt from a sales header table. The access pattern typical for a lookup transaction is shown in gray shading. Here the sales orders of the customer with the identifier 1001 are selected.

Order ID	Order Date	Order Category	Net Value	Cur-rency	Distri-bution Channel	Sales Group	Requested Delivery Date	...	Customer Telephone	Cus-tomer ID
4969	01/02/2011	Order	5.500,00	EUR	10	103	01/03/2011		0241-77729	1390
4970	01/03/2011	Order	32.838,00	EUR	12	110	01/08/2011		0365-2251-0	1175
4971	01/07/2011	Order	12.200,00	EUR	12	101	01/09/2011		069-467653-0	1001
4972	01/21/2011	Order	28.604,00	EUR	12	110	01/22/2011		0511-123400	2200
4973	01/21/2011	Order	19.719,00	EUR	12	130	01/22/2011		089-456897-0	1033
4974	01/21/2011	Order	46.686,00	EUR	12	101	01/22/2011		069-987654-9	2140
...										
5010	01/27/2011	Order	13.020,00	EUR	10	101	01/28/2011		0221-454321-0	2130
5011	01/27/2011	Order	12.156,00	EUR	10	110	01/28/2011		0351-5423-00	1460
5013	01/28/2011	Order	667.700,00	EUR	12	111	01/30/2011		069-467653-0	1001

Fig. 4.6 Simplified comparison of access patterns of OLTP and OLAP queries

Analytical queries, in contrast, are more resource and computing intensive including groupings and orderings, for example, by region or sales group. Range selects and table scans make up more than 50 % of all query types [125]. Thus, the data sets that are touched are much larger compared to those of OLTP queries. OLAP queries follow a column-oriented pattern, selecting only a very small subset of columns from a table to compute the results for a query. Compared to OLTP queries, OLAP queries are relatively long running due to their low selectivity and only a small number of columns is usually retrieved. For example, a query computing the *net revenue of sales in January 2011 grouped by sales group* only needs the revenue, sales group and date columns of the sales header table. This query selects an entire block of the sales order table according to the selected date. This pattern is depicted in black shading in Fig. 4.6 selecting the aggregated net value of sales orders grouped by sales group.

The task of mixing the workload is to run these different queries in a predefined and reproducible manner concurrently on the system under test either in realistic shares or according to evaluation of specific scenarios. An example for such a specific scenario is measuring the impact of adding analytical queries to an OLTP workload. Here, starting from a pure OLTP workload that remains constant through the entire measurement period an increasing share of OLAP clients could be added to evaluate how much OLTP response times are affected by the changed workload.

A realistic mixed workload of OLTP and OLAP queries cannot be observed, as no such system exists, yet. Therefore, the workload is configurable in CBTR with regard to:

- Share of the OLTP (S_T) and OLAP (S_A) sub-workloads, with $S_T := \{x \in \Re | 0 \leq x \leq 1\}$ and $S_A = 1 - S_T$.
- Share of read-only OLTP queries (S_{rT}) and mixed OLTP queries (S_{mT}) within the OLTP sub-workload, analogously defined as $S_{rT} := \{x \in \Re | 0 \leq x \leq 1\}$ and $S_{mT} = 1 - S_{rT}$.

Table 4.4 gives an overview for the computation of the shares in the workload for all queries based on the above parameters. The individual shares for the mixed OLTP queries have been introduced in Sect. 4.3.3. For example, the share of the

Table 4.4 Computation of query shares

Type	Query	Share
Mixed OLTP	Sales order	$S_{mSales} = 0.3 * (1 - S_{rT}) * S_T$
	Shipping	$S_{mShip} = 0.27 * (1 - S_{rT}) * S_T$
	Billing	$S_{mBill} = 0.25 * (1 - S_{rT}) * S_T$
	Clearing	$S_{mClear} = 0.18 * (1 - S_{rT}) * S_T$
Read-only OLTP	$S_{rTi} = p_i * S_{rT} * S_T$, with $p_i := \{x \in \Re \mid 0 \leq x \leq 1\}$ and $\sum_{i=1}^{n} p_i = 1$, with n as the number of defined read-only OLTP queries	
OLAP	$S_{Ai} = p_i * (1 - S_T)$, with $p_i := \{x \in \Re \mid 0 \leq x \leq 1\}$ and $\sum_{i=1}^{n} p_i = 1$, with n as the number of defined OLAP queries	

sales order query is computed based on its share within the mixed OLTP queries (30 %) times the probability of occurrence of mixed OLTP queries within the OLTP part of the workload (S_{mT}) times the probability of the OLTP part of the workload itself (S_T) within the entire workload.

4.5 CBTR and the Benchmark Properties

In this section, a closer look is taken at how the key criteria summarized in Sect. 3.3 are reflected in CBTR and discusses the approach taken to create this new benchmark. The key criteria that are the basis for the creation of "good" benchmarks are relevancy, repeatability, fairness, verifiability, and being economical. Not all of these criteria have to be fulfilled in perfection to create a good benchmark, yet they should not be neglected.

Relevance

In Sect. 4.2, an overview of application areas in transactional systems was given and the selection of one area as the setting for CBTR was discussed. The order-to-cash process was chosen, as it is a relevant part of business for each company, regardless whether it offers consumer products or services. This process is essential to keep the core business of companies running and it is the basis for many questions of strategic interest. The order-to-cash process is easily understandable as it is a part of people's everyday life as consumers. Nevertheless, the chosen process has been cut out of a business system composed of many highly integrated processes interacting with each other. Thus, it models only part of the real world and simulates the observed behavior. This simulation results in repeatability, simplicity, and manageability. CBTR is more realistic than other benchmarks, because its data structures are based on observations of currently used real systems, e.g. providing a larger number of attributes, that the database has to handle during processing like it is the case in the real systems.

Repeatability

A CBTR benchmark run is composed of the setup, the actual run, and the evaluation of the results. In the setup the database schema is created and data is loaded into the tables. For the run, a configuration file specifies the types and number of clients to simulate and the duration of the run. Afterwards, the collected result logs can be retrieved and analyzed. As a database is newly created for each run and a configuration can be specified, a run can be repeated.

The server running the database is separate from the one running the benchmark driver and simulating the clients. Resources used for the simulation are not taken from the database server and, thus, do not affect its operation. Thereby measured response times are kept clean from interferences. An assumption is that the network and the systems running the database and the benchmark driver are stable.

Fairness

The CBTR benchmark is fair as its only requirement for the database systems is the usage of SQL as query language. CBTR targets relational database systems. Therefore, this prerequisite is plain, as SQL is the standard language in this area.

Verifiability

Throughput and response time as the benchmark output can be verified with the system under test manually on the one hand to gain an understanding of the basic speed of operation. On the other hand, in addition to the clients' performance, system performance is monitored during the benchmark run and data is logged for analysis afterwards.

Economical

The benchmark exercises standard functionality of the database system via a standard query language (SQL). Extensive adaptation on the side of the database system, for example in terms of programming, is not required. Normal tuning of operating system and database server parameters according to the expected workload, however, is recommended.

Another important aspect for a benchmark to be of value is its distribution and actual usage and thus a growing community to share and compare results. Here, the benefit of using existing benchmarks to establish a new benchmark is their already established acceptance making it easier for the new benchmark to be accepted. Furthermore, the community is already familiar with how they work. Thus, depending on the extent of the changes, the effort of becoming familiar with the workings of the new benchmark is modest. To facilitate understanding,

the scenario for CBTR has been chosen based on the observation from existing benchmarks that use scenarios from daily life. The order-to-cash process modeled in CBTR is omnipresent in everyday life and as a result easy to understand. Yet, it provides a setting that is complex enough to simulate sophisticated transaction processing and reporting.

4.6 Summary

The focus of this chapter lies on the introduction and definition of a hybrid benchmark that incorporates transactional processing as well as analytical processing and variable mixed workloads thereof. This benchmark is used in the remainder of this thesis to simulate mixed transactional and analytical workloads for the analysis of the impact of logical database schema optimizations on transaction and query performance. Before this work, no methods to simulate mixed transaction and analytical processing workloads and compare database systems under such mixed workloads have been available.

CBTR, the new hybrid benchmark proposed in this chapter, is based on the table structures and queries observed in a real enterprise system. Its scenario has been cut out of the business world that comprises many highly integrated processes that interact with each other. The used order-to-cash process is intuitive and easy to comprehend, as it is part of everyday life. Yet, the benchmark database schema is closer to that of enterprise systems increasing the overhead for the benchmark queries similar to the overhead real-world queries encounter.

Chapter 5
Database Schema Variants for Mixed OLTP and OLAP

As discussed in Sect. 2.2, a database design optimized for OLAP degrades the performance of OLTP queries and vice versa. If OLTP and OLAP workloads are to be consolidated onto one system, the question of how to design the database arises. Optimizations of the dedicated OLTP and OLAP systems that have evolved in the past might still be applicable to tune the performance of systems under mixed OLTP and OLAP workloads. In this chapter, the typical optimizations in logical database design of OLTP and OLAP systems, summarized in Sect. 2.2, are applied to the order-to-cash scenario of the CBTR benchmark to create the basis of determining their impact in mixed workload scenarios. The change of the database schema according to particular optimizations results in the need to adjust the queries as well. The query adaptations are summarized into rules in this chapter. If the impact of specific optimizations under distinct workload mixes within a database system has been understood and workload changes can be identified that lead to according modifications in the database schema, these rules could provide a starting point for automatic adaptation of queries as well.

Figure 5.1 illustrates different types of database schemas that can be observed in OLTP and OLAP systems. From left to right the database schema types are ordered according to increase of data redundancy and decrease of the number of join operations. The dotted areas named OLTP and OLAP include those database schema types that have been observed in the respective domains.

In OLAP database schemas tables can be pre-joined like in the star schema. As a result, queries contain fewer joins. Yet, pre-joining of tables can increase the redundancy within the data set. In the case of the star schema, redundancy is incurred in the dimension tables if hierarchies are mapped to one table. In the snowflake schema, hierarchies are normalized into several tables. OLTP schemas, through normalization, do not incur high levels of redundancy. Here, the goal is to avoid insert anomalies and update dependencies. The disadvantage of this approach is that a larger number of joins is needed to reproduce the context of an entity that may become spread over several tables with a higher degree of normalization. Yet, for transactional processing this is not a serious challenge as transactions exhibit

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_5,
© Springer-Verlag Berlin Heidelberg 2014

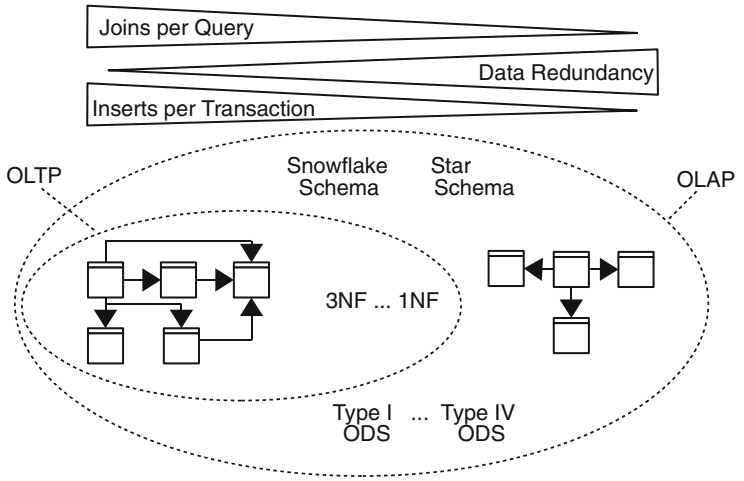

Fig. 5.1 OLTP and OLAP database schema types

a high selectivity compared to analytical queries. Thus, joined sets have a small number of entries, e.g., the specific products ordered in a sales order, or the business partners involved in the deal.

The shown schemas do not belong to distinct classes, but overlap. For example, the snowflake schema contains normalized dimension tables and the fact table is a special structure to collect business data of a specific context, e.g., sales revenue. The class I ODS, which is relevant for real-time reporting, is a copy of a subset of the transactional data kept in sync with the transactions and therefore contains a normalized schema as well. The schemas employed in the OLAP domain are thus a mixture of normalized tables like in the OLTP schemas and additional structures such as fact tables or pre-computed report tables.

This chapter starts with a comprehensive review of the levels of database design in Sect. 5.1 to bring database schema variation as the topic of interest for the application of the proposed benchmark into line with alternative optimization strategies, which have already been presented in Chap. 2. In Sect. 5.2, a selection of database schema variants will be presented. These are chosen according to their relevance in transactional and analytical processing.

5.1 Database Design Variation Levels

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 queries, the task of choosing an optimal database schema and optimizations according to a specific mix of queries becomes vital due

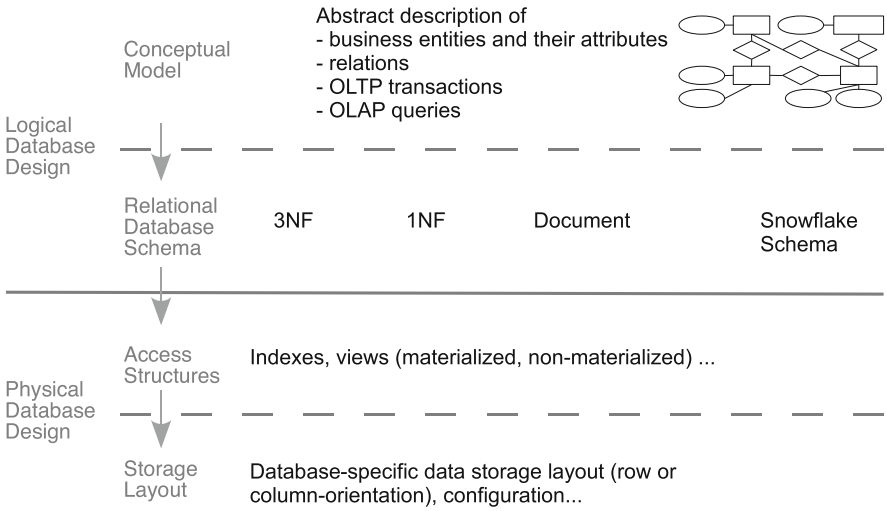

Fig. 5.2 Database design variation levels

to the wide range of data access behavior of the individual queries in the combined OLTP and OLAP scenario.

Figure 5.2 gives an overview of the levels relevant in database design where variation through optimization can take place. The top layer represents the logical database design step. It contains the conceptual definition of involved entities, attributes and their relations and the mapping of the conceptual design to a database schema that includes the concrete definitions of the structures as managed by the database, that is, relational tables. In the physical layer, additional access structures such as indexes to optimize data access and the actual layout of data on the storage medium are encapsulated.

The definitions of entities and their characteristics and relations between them given on the conceptual layer are the basis for applications to work on to achieve logical data independence. If OLTP and OLAP queries in applications were defined based on these conceptual definitions instead of referencing the structures directly in the database schema, they would be oblivious to adaptations of the database schema on the logical design layer and of course independent of changes on the layers below.

Both, the logical and physical database design layers, represent starting points for optimization. Normalization as a transformation of the tables resides in the logical design layer and results in different database schemas. Especially for a composite OLTP and OLAP system, where the workload can tend toward OLTP or OLAP periodically, autonomous database schema variation according to the current workload on the logical level similar to autonomous physical database optimization [81] is of interest.

The logical layer in Fig. 5.2 depicts database schema variation examples according to normalization. 1NF is the extracted part of a database schema of

a real enterprise system, which is modeled in CBTR. 3NF, document-oriented, and snowflake schema are its variations to analyze the impact of normalization under varying mixes of OLTP and OLAP in the workload and in different types of databases. These will be introduced in Sect. 5.2.

The physical design layer comprises the configuration of tables according to the storage media, data types, and access structures used and supported by a database system. Physical database design optimizations and database storage alternatives were introduced in Sects. 2.2.3 and 2.2.4. As discussed there, much research in the area of autonomous physical database design optimization has already been conducted and successfully applied in productive database systems, e.g., DB2 Design Advisor [227]. In the context of composite OLTP and OLAP systems distinguishing characteristics are already found on the logical layer of database design, which is an earlier step than physical database design.

The focus of this work lies in finding optimizations in this earlier step. It is the prerequisite for fine-tuning the database on the lower levels in subsequent steps. The impact on the performance of queries in mixed workloads of typical database schemas used in transactional and analytical processing is evaluated.

5.2 Database Schema Variants

This section focuses on variants for the database schema of the order-to-cash process. One aspect of schema variation in this section is concerned with normalization, the other with the specific structures used in analytical processing. The denormalization techniques that are analyzed regarding their impact on query performance in mixed workload scenarios are pre-joining the tables, derivable data, and redundant data. These techniques have been described in Sect. 2.2.2. While the increase in the level of denormalization increases redundancy within the data set and OLTP transactions writing to the database are impeded, the reduced join complexity can be advantageous for OLAP queries. If reduced join complexity and, thus, denormalization is of advantage in mixed workload scenarios is a question of particular interest in column-oriented and hybrid databases. For these, existing optimizations as used in row-oriented databases are not necessarily applicable or they lead to different results.

In this section, three schema variants are introduced in addition to the transactional database schema in first normal form already used in CBTR. The steps taken to define new schema variants are to start with the original schema in first normal form as defined in CBTR and denormalizing this schema step-by-step by pre-joining the transaction data tables. Two variants are created in this manner, which are the document-oriented schema described in Sect. 5.2.1 and a snowflake schema described in Sect. 5.2.2. The third schema variant is created by removing derivable data and redundant data creating a normalized schema in third normal form (3NF) as shown in Sect. 5.2.3. Changing the database schema entails changing

the queries defined on top of them. The query changes will also be presented in the following.

5.2.1 Document-Oriented Schema

In the first variant, header and item tables are joined, because header and item information is mostly requested in combination. To create a schema variant based on pre-joined tables, the typical queries taking place in a system (the CBTR queries for the order-to-cash process in the case of this thesis) have to be taken into account to identify entities to be joined. From a business perspective, the data about the line items of a sales order is never queried without accessing the sales order header. Even in contexts such as *top n sold products* the header information is necessary to provide the time dimension, as these queries always reference a certain time frame, for example last quarter. This schema is called "document-oriented", as it reflects a complete print-out document of, e.g., a sales order.

Figure 5.3 illustrates the document-oriented schema derived from the schema introduced in Sect. 4.2.2. From the eight transaction tables comprising header and item data of sales orders, shipments, bills, and accounting documents, four tables are obtained, which in each case pre-compute the join between the header and the item table via the document identifier. As an example, the sales facts table is produced from joining the sales header table to the sales item table using the primary key order ID from the sales header table and its pendent in the sales item table for the equijoin condition. Here, and also for the other three sets an inner join is sufficient in a sound order-to-cash database to produce the new tables, as a header is invalid without line items and a line item always belongs to a header. All other tables remain unchanged.

In addition to changing the schema, the queries have to be adapted. The changes to the queries are classified as follows:

Join-A When replacing the join of tables in the *FROM* clause of a selection by a pre-joined table that exactly represents the join in the statement, only the name of the table in the *FROM* clause is replaced by the new fact table and the join conditions used for the pre-computed join in the fact table are removed from the *WHERE* clause of the statement. A typical example for this is the pre-computed join of header and item tables.

Join-B1 For all selections and computations solely based on columns that have been part of tables where tuples have been multiplied during the pre-join, a sub-select with *DISTINCT* command is added to filter out the added redundancy. This redundancy has been created when the tuples were duplicated for each of their associated tuples, e.g., header tuples were copied for each of their associated line item tuples in the header item pre-joined table. The selected distinct columns in the sub-select match the selected columns in the main select with the addition of the primary key columns of the former tables that have been multiplied and on which the selection or aggregation is based (if they are not included already). This

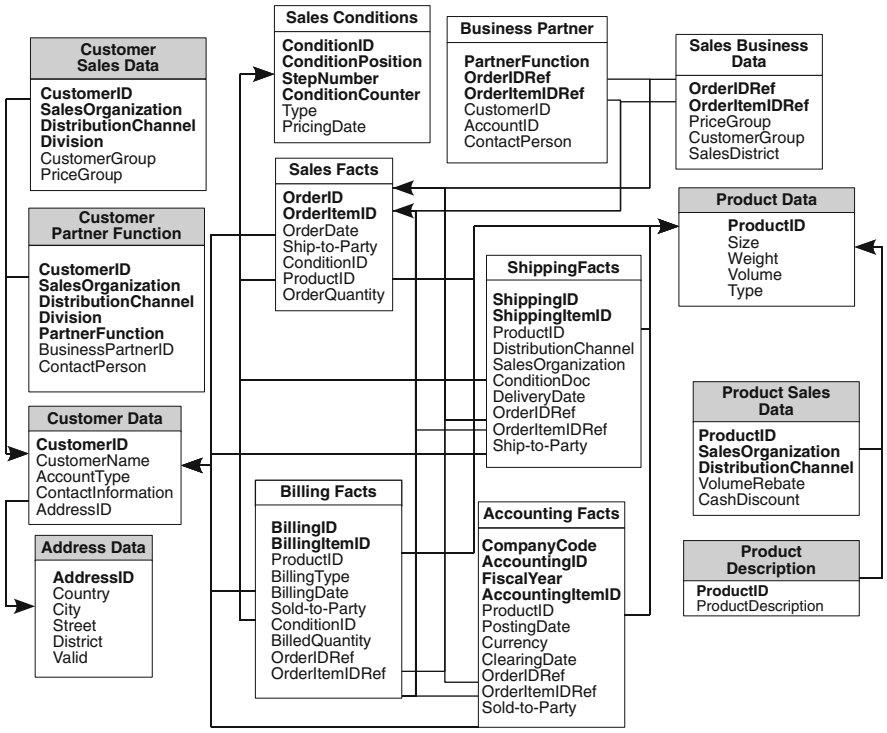

Fig. 5.3 Document-oriented schema

ensures the correct restoration of the cardinality of the former table according to the *WHERE* conditions.

Join-C For inserts to a pre-joined table that presents a one-to-many or parent-to-child relationship the tuple from the former parent table is duplicated for each child tuple and a concatenated tuple comprised of all parent and child columns is inserted. A typical example is the insertion of a new tuple to a pre-joined header and item table, where header data is duplicated for each of its items.

The orders by period query is a special case of type Join-B1. The results have to be filtered regarding distinct values because a used table has been expanded, however, no aggregates are computed. For this query, it is sufficient to only introduce a *DISTINCT* in the original *SELECT* clause without introducing a sub-select to filter the distinct values. This leads to a fourth type of query change as a special case of Join-B1:

Join-B2 For all selections solely based on columns that have been part of tables where tuples have been multiplied during the pre-join and no computation takes place, a *DISTINCT* command is added to the *SELECT* clause to filter out the added redundancy.

Table 5.1 gives an overview of the changes in the SQL statements of the queries. Only the changed parts of the SQL statements are presented and unmodified parts are abbreviated with "`[...]`". Queries not mentioned in the table are not changed at all. This applies to all accesses to master data.

5.2.2 Snowflake Schema

The next schema variant further increases the level of denormalization by completely merging the sales and delivery entities 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 queries making up the workload of CBTR. This schema variant resembles a snowflake schema with shared dimension tables. It represents the maximum level of denormalization that is appropriate for the set of OLAP queries given in CBTR. No changes are applied to the master data tables to create a star schema, because the CBTR OLAP queries would not benefit from joins between dimension tables that represent the master data.

Figure 5.4 illustrates the snowflake schema variant. As can be seen two fact tables have been created. The two fact tables share the product and customer dimension tables. The sales-shipping facts table is produced by a left outer equijoin of the sales facts table to the shipping facts table introduced in Sect. 5.2.1. The join is computed on the order ID reference and order item ID reference foreign keys, which are removed in the pre-joining step. The left outer join is used, because shipping data may not be present for some sales orders, but a sales order always exists for shipments. However, a shipping may cover several sales orders, or a sales order may be split over several shipments. By the left outer join, this relation is retained. Similarly, the billing-accounting facts table is created by a left outer equijoin of the billing facts table to the accounting facts table using the billing ID reference and billing item ID reference foreign keys of the accounting table.

Table 5.2 gives an overview of the changes of the CBTR queries for the snowflake schema variant. Only the differences to Table 5.1 are given as the snowflake schema variant builds upon the document-oriented schema variant. The change column lists all types of changes according to the above classification that have to be applied when starting from the original benchmark schema in first normal form. In the daily flash query only the accessed table is exchanged with the new pre-joined table, but no other changes are included as the selection refers to the former sales header and the original tuples are filtered by the distinct selection on order ID, which is already included in the document-oriented schema variant. A similar behavior can be observed for the orders by period and open items transactions.

For the write-access OLTP queries, another type of change has to be introduced in the case of the snowflake schema. The fact tables now consist of transaction tables where new tuples are added in different phases of the order-to-cash process. For example, the sales-shipping fact table combines data that is created during sales order processing and shipment processing. Shipment data may not be

Table 5.1 CBTR query changes for the document-oriented schema variant

Query	Change	Statement (shortened)

Order by key	Join-A	`SELECT [...]`
		`FROM SalesFacts`
		`WHERE OrderID = @DocumentNumber;`

Orders by period	Join-B2	`SELECT DISTINCT OrderID, SoldToParty, OrderDate, NetValue, Currency`
		`FROM SalesFacts`
		`WHERE CustomerID = @CustomerID`
		`AND (OrderDate BETWEEN "01.10.2011" AND "31.10.2011");`

Open items	Join-A	`SELECT [...]`
		`FROM AccountingFacts`
		`WHERE PostingDate BETWEEN "01.09.2011" AND "30.09.2011"`
		`AND AccountType = "Debitor"`
		`AND ClearingDate= "␣"`
		`ORDER BY [...];`

Sales order	Join-C	For each sales item, a concatenated sales header-item tuple is inserted
Shipping	Join-C	For each shipping item, a concatenated shipping header-item tuple is inserted
Billing	Join-C	For each billing and accounting item, a concatenated billing and accounting header-item tuple is inserted
Clearing	–	No major changes, only adapt the tables to be accessed

Delivery fulfillment	Join-A	`SELECT [...], (SELECT [...]`
		`FROM SalesFacts sf2`
		`WHERE sf2.OrderDate BETWEEN "01.07.2009" AND "30.09.2009"`
		`AND sf.SalesGroup = sf2.SalesGroup`
		`AND sf.DistributionChannel = sf2.DistributionChannel`
		`AND sf.Division = sf2.Division`
		`AND sf.SalesUnit = sf2.SalesUnit) AS Expected`
		`FROM SalesFacts AS sf, ShippingFacts AS shf`
		`WHERE (sf.OrderDate BETWEEN "01.07.2009" AND "30.09.2009")`
		`AND shf.DeliveryDate <= sf.DeliveryDate`
		`AND sf.OrderID = shf.OrderIDReference`
		`AND sf.OrderItemID = shf.OrderItemIDReferenc`
		`GROUP BY [...] ORDER BY [...];`

DSO	Join-B1	`SELECT [...]/ (SELECT [...]`
		`FROM (SELECT DISTINCT [...], bf2.BillingID`
		`FROM BillingFacts AS bf2`
		`WHERE [...]))*91 AS DSO`
		`FROM (SELECT DISTINCT [...], af.AccountingID, bf.BillingID`

(continued)

Table 5.1 (continued)

Query	Change	Statement (shortened)
		`FROM AccountingFacts AS af, BillingFacts AS b WHERE b.BillingDate BETWEEN "01.07.2011" AND "30.09.2011" AND af.ClearingDate <> "␣" AND af.AccountType = "Debitor" AND af.Indicator = "Debit" AND b.BillingID = af.BillingIDReference GROUP BY [...] ORDER BY [...];`
Processing time	Join-A	`SELECT [...] FROM (SELECT DISTINCT [...] FROM SalesFacts AS sf, ShippingFacts AS shf WHERE sf.OrderDate BETWEEN "01.07.2011" AND "30.09.2011" AND sf.OrderID = shf.OrderIDReference AND sf.OrderItemID = shf.OrderItemIDReference) GROUP BY [...] ORDER BY [...];`
Daily flash	Join-B1	`SELECT S.OrderDate, S.Country, S.Currency, SUM(S.NetValue) AS SalesRevenue FROM (SELECT DISTINCT [...], sf.OrderID FROM SalesFacts AS sf, CustomerData AS cd WHERE [...]) AS S GROUP BY [...] ORDER BY [...];`

available during the creation of the sales order. Thus, the according columns of the fact table, which represent the shipping part, remain empty. They are updated as soon as shipments are processed. This leads to two more classes of query changes:

Join-D Inserts to fact tables where data becomes available at a later point in time result in inserts of incomplete tuples to the database where the missing information is marked by empty values.

Join-E Inserts to fact tables where part of the tuple already exists in the database are converted to updates of the referenced tuples. Before updating a tuple with the additional data, an evaluation is necessary if data has been added already to a tuple to avoid overwriting data. If a complete tuple is already contained in the database, a copy-on-update is performed, leaving the previously updated tuple intact. An example where data would be overwritten without a check is a sales order that is split over two shipments. In case of an update on the condition of

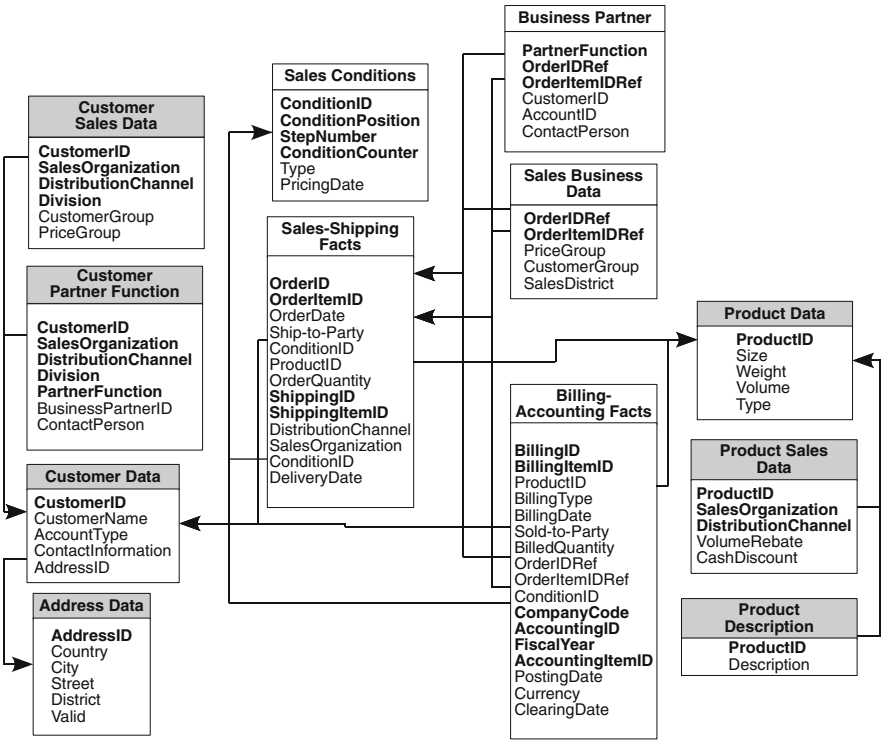

Fig. 5.4 Order-to-cash tables as a snowflake schema

equal order ID references and order ID item references, data would be lost when the second shipment updates the sales-shipping facts tuple already containing the data of the first shipment. Here, at the second insert of shipment data, referencing the same sales order item, the sales data is copied and a completely new tuple including the shipment data is inserted.

5.2.3 Third Normal Form Schema Variant

This schema variant is relevant for a subset of the transactions and queries that utilize pre-computed (derived) and redundant fields in the database. Derived data and redundant data are semantic concepts that are encapsulated in the application logic as opposed to pre-joins of tables applied in the prior two schema variants. An example for a pre-computed field in CBTR is the net value provided in the sales header table. It sums up the price of a sales item multiplied by the ordered quantity for all of a sales orders line items. Similar aggregates can be found in the shipping header table and billing header table. Examples for redundant data

Table 5.2 CBTR query changes for the snowflake schema variant

Query	Change	Statement (shortened)
Daily flash	Join-B1	`SELECT [...] FROM (SELECT DISTINCT [...] FROM SalesShippingFacts, CustomerData WHERE [...]) GROUP BY [...] ORDER BY [...];`
Processing time	Join-A	`SELECT [...] FROM (SELECT DISTINCT [...] FROM SalesShippingFacts WHERE OrderDate BETWEEN "01.07.2011" AND "30.09.2011") GROUP BY [...] ORDER BY [...];`
Delivery fulfillment	Join-A	`SELECT [...], (SELECT [...] FROM SalesShippingFacts WHERE [...]) AS Expected FROM SalesShippingFacts AS sf WHERE (sf.OrderDateBETWEEN "01.07.2009" AND "30.09.2009") AND sf.DeliveryDate <= sf. DeliveryDate GROUP BY [...] ORDER BY [...];`
DSO	Join-A, Join-B1	`SELECT [...] / (SELECT [...] FROM (SELECT DISTINCT [...] FROM BillingAccountingFacts WHERE [...]))*91 AS DSO FROM (SELECT DISTINCT [...] FROM BillingAccountingFacts AS bf WHERE (bf.BillingDate BETWEEN "01.07.2011" AND "30.09.2011") AND bf.ClearingDate <> "␣" AND bf.AccountType = "Debitor" AND abf.Indicator = "Debit") GROUP BY [...] ORDER BY [...];`
Order by key	Join-A, Join-B2	`SELECT DISTINCT [...] FROM SalesShippingFacts WHERE [...];`
Orders by period	Join-B2	`SELECT DISTINCT [...] FROM SalesShippingFacts WHERE} [...];`
Open items	Join-A, Join-B2	`SELECT DISTINCT [...] FROM BillingAccountingFacts WHERE [...] ORDER BY} [...];`
Sales order	Join-C, Join-D	Add empty values according to the columns needed for the shipping header and item tuples

(continued)

Table 5.2 (continued)

Query	Change	Statement (shortened)
Shipping	Join-C, Join-E	Update the already created sales order with the new data for shipping header and item tuples, selecting the tuples to update by order ID reference and order ID item reference contained in the shipment
Billing	Join-C	Add tuples composed of billing and accounting header and item data, joined by the billing and accounting identifiers, order ID reference, and order item ID reference
Clearing	–	Again, no major changes, only adapt the tables to be accessed

to be removed in the first normal form schema are product identifier, sales order item description, product hierarchy information, and sales unit for a product. These are stored in the sales order tables, shipping tables, and billing tables. Shipping and billing entries, however, reference the sales order lines from which they are created. Thus, the above information could be retrieved through this link instead of redundantly storing the data. The omission of derivable data and redundant data is applied to the original schema variant in first normal form by dropping the respective columns and rewriting the queries to

- Compute the previously pre-computed data on the fly for derivable data and
- Retrieve the previously redundant data through additional joins.

Table 5.3 presents the removed attributes, the origin table of the data, the table and column that replaces the removed attribute, and the affected queries. The removed attributes are classified into derivable data and redundant data. Concerning derivable data, the net value aggregates within the header transaction tables are removed, so that queries using these now need to compute the values based on the net value attributes of the corresponding items and a grouping on the foreign key linking the header and the item tuples.

Removed redundant data includes the division and material type attributes stored in the product data table, which have been copied to the transaction header or item tables. The country specification stored in the customer data table has been removed from the billing header transaction table. From the shipping and billing transaction tables, data has been removed that is already stored in the sales transaction tables like sales item description, the ordered product (product ID), sales unit, and sales organization, to name a few. The document currency data stored redundantly in sales header and item has been removed.

The adaption of write-access OLTP queries is limited to removing those values in the insert statements corresponding to columns that have been dropped. Selection parts of the write-access OLTP queries are not affected as data is retrieved either from master data tables or from tuples created in earlier steps of the process flow.

Table 5.3 CBTR schema changes for the third normal form schema variant

Origin table	Removed column	Replacement table	Replacement column	Affected queries
Derivable data				
Sales header	Net value	Sales item	Sum of all net values	Sales order, orders by period, daily flash
Shipping header	Net value	Shipping item	Sum of all net values	Shipping
Billing header	Net value	Billing item	Sum of all net values	Billing, DSO
Redundant data				
Sales header	Division	Product data	Division	Processing time, delivery fulfillment, sales order
Sales item	Division	Product data	Division	Sales order
	Document currency	Sales header	Document currency	Order by key, sales order
Shipping header	Document currency	Sales header	Document currency	Shipping
	Sales organization		Sales organization	
Shipping item	Sales organization	Sales header	Sales organization	
	Sales item description	Sales item	Sales item description	
	Product ID		Product ID	
	Product hierarchy		Product hierarchy	
	Sales unit		Sales unit	
	Material type	Product data	Material type	
	Division		Division	
Billing header	Billing date	Shipping header	Billing date	DSO, billing
	Division	Product data	Divison	Billing
	Country of destination	Customer data	Country	
Billing item	Sales item description	Sales item	Sales item description	
	Product ID		Product ID	
	Product hierarchy		Product hierarchy	
	Sales unit		Sales unit	
	Division	Product data	Division	

Of the read-only OLTP queries, only order by key and orders by period have to be adapted. A simple exchange of the selection of currency from the sales item table to the same-titled column of the sales header table is carried out. Both tables are already used in the statement in its original form. In the orders by period query, a join to the sales item table via the order ID and a grouping by order ID are added to compute the sales order net value. The changes to the syntax for the query statement

```
SELECT sh.[...], SUM(si.NetValue)
FROM SalesHeader AS sh, SalesItem AS si
WHERE sh.CustomerID=[...] AND (sh.OrderDate BETWEEN
    [...])
 AND sh.OrderID = si.OrderID
GROUP BY sh.OrderID,sh.SoldToParty,sh.OrderDate,sh.
    Currency;
```

Listing 5.1 3NF version of show sales orders by period

```
SELECT [...]
FROM (SELECT DISTINCT [...], pd.Division
 FROM [...], ProductData AS pd
 WHERE [...] AND pd.ProductID = sdi.ProductID)T
GROUP BY [...] ORDER BY [...];
```

Listing 5.2 3NF version of average order processing time

are presented in Listing 5.1, based on the complete statement given in Listing 4.6 on page 78. The remaining OLTP queries rely on master data tables or on those columns of transaction tables that have not been changed.

All OLAP queries need to be adapted to this schema variant. The daily flash query is subject to the same adaption as the orders by period query, except that the GROUP BY clause is already contained in the original statement. In the average order processing time query, the selection of the division from the sales header table is exchanged for the selection from the product data table and the corresponding join via the product ID is added. The changes to the statement are presented in Listing 5.2 based on the statement given in Listing 4.12 on page 81. Similar to the average order processing time query, the order delivery fulfillment query is changed. Here, in the outer, as well as the inner selection, a join to the product data table is added for the access to the Division value.

In the days sales outstanding query, two attributes have to be exchanged. One is the aggregated net value so far taken from the billing header table, which has to be computed based on the billing item table now. The other is the billing date so far taken from the billing header that was a copy of the billing date stored in the shipping header. For the net value computation, a join to the billing item table is added via the billing ID. The filter of the billing date from the billing header is replaced by a SELECT clause, which joins the billing item table to the shipping header table via the shipping ID reference and returns all billing IDs in the period of interest. In the outer SELECT, the join to the billing header table becomes superfluous and it is removed including the corresponding join condition. The changes to the SQL statement based on the original statement given in Listing 4.14 on page 83 are presented in Listing 5.3.

```
SELECT [...] / (SELECT SUM(bi2.NetValue)+SUM(DISTINCT
    bh2.TaxAmount)
 FROM [...], BillingItem AS bi2
 WHERE bh2.Currency = [...]
  AND bh2.BillingID = bi2.BillingID
  AND bi2.BillingID IN (SELECT DISTINCT bi3.BillingID
   FROM BillingItem AS bi3, ShippingHeader AS h
   WHERE bi3.ShippingIDReference = h.ShippingID AND
    h.BillingDate BETWEEN "01.07.2011" AND "30.09.2011"
       ))*91 AS DSO
FROM AccountingItem AS ai, BillingHeader AS bh
WHERE ai.ClearingDate <> [...] AND ai.BillingID IN
  (SELECT DISTINCT bi3.BillingID FROM BillingItem AS
     bi3, ShippingHeader AS h
  WHERE bi3.ShippingIDReference = h.ShippingID
   AND h.BillingDate BETWEEN "01.07.2011" AND "
      30.09.2011")
GROUP BY [...] ORDER BY [...];
```

Listing 5.3 3NF version of the days sales outstanding query

5.3 Summary

In this chapter, a document-oriented schema, a snowflake-style schema, and a 3NF schema were defined as three variants for CBTR's database schema. The goal of defining these schema variants is to quantify the impact of typical optimizations applied in database schemas of OLTP and OLAP systems under mixed workload conditions. The optimizations taken into account here are part of logical database design. These include or avoid specific database optimizations within the area of (de-) normalization to analyze their impact.

The original schema of CBTR in 1NF includes redundant data in the form of master data within the transaction tables to reduce master data look-ups and derivable data in the form of aggregates. The document-oriented schema introduces pre-joined tables on top of redundant data and derivable data. The former header and item tables of the 1NF schema have been pre-joined based on the observation of frequent joined accesses to those tables. The snowflake-style schema additionally introduces report tables preparing the data for the OLAP queries. The variants mentioned until now are increasingly advantageous for OLAP queries. The last variant in 3NF is favorable for OLTP queries. For this variant, all redundant and derivable data, pre-joined and report tables were removed.

If the database schema is changed, the queries on top have to be adapted as well. Rules that define how to change the SQL code of the queries have been defined based on the table changes in the schema variants.

Part III
Implementation, Evaluation, and Discussion

Chapter 6
The CBTR Tool Chain

The tool chain to assess database systems according to mixed workloads is composed of three parts. Figure 6.1 provides an overview of these parts. The first comprises database specific scripts to create the database schema (tables and indexes) and to load the benchmark base data into the database system that is going to be tested. The second part covers the simulation of a specific workload configuration – a benchmark run. The last part relates to the analysis of the log files from one or several benchmark runs and the visualization of results.

Section 6.1 describes the preliminaries before running the benchmark and what is needed for cleanup afterwards. This section also presents how the workload is configured, how the benchmark driver is implemented, and what output is produced during the benchmark runs. In Sect. 6.2, examples are given what reports and visualizations are provided by the implemented tool chain. Section 6.3 closes this chapter with an outlook on possible further work concerning the tool chain.

6.1 The Benchmark Run

A benchmark run is controlled by the CBTR driver. The CBTR driver is implemented as a multi-threaded Java [158] application that communicates with databases via JDBC [157]. To avoid interferences, e.g., through shared resources, between the CBTR driver and the database during the tests, the CBTR driver should reside on an own server instead of being executed on the same server that runs the database.

Figure 6.2 illustrates the components of the CBTR driver. One of the threads manages the general driver functionality (*benchmark controller*) including client setup, instantiation and buffering of query queues, and the setup of the *logger* and *result collector*. Through the *DB service* component, each client receives an own JDBC connection to the database that is used to send requests.

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_6,
© Springer-Verlag Berlin Heidelberg 2014

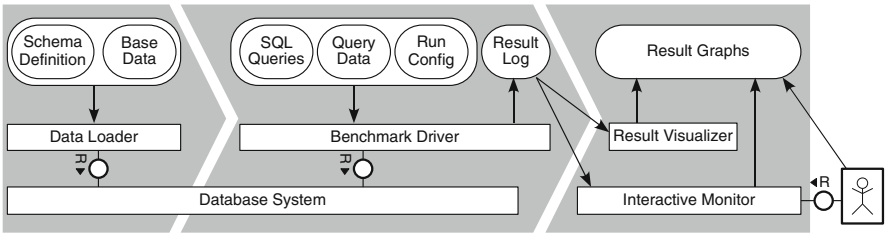

Fig. 6.1 The CBTR tool chain

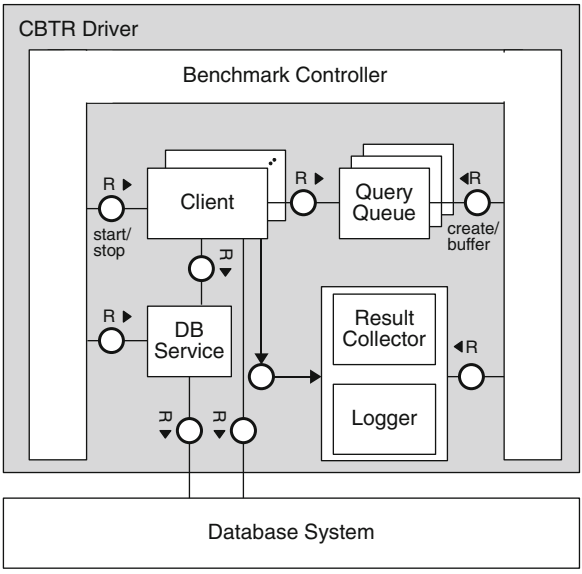

Fig. 6.2 Components of the CBTR driver

Benchmark Run Preparation and Tear-Down

The database setup includes the creation of the tables and the optional creation of indexes. The schema definitions are provided as SQL commands in files that are read by the *data loader*. Loading of a base data set is also part of the database setup, as the database should be pre-filled with data before the benchmark starts, so the read-only queries do not encounter empty responses. For loading the data, the databases own client tools are used as most databases provide clients that allow a high-performance bulk load of data from text files.

During a benchmark run, the data set is modified through the mixed access queries. Before each benchmark run the database has to be reset to its initial state so that benchmark results are reproducible and always the same data set is queried for a given scaling factor. Such a reset can either be executed by undoing all write queries

Table 6.1 Client types and workload components

		Workload type		
		Mixed OLTP	Read-only OLTP	OLAP
Client type	Mixed OLTP/OLAP	X	X	X
	OLTP	X	X	
	OLAP			X

performed during the benchmark run or by destroying and recreating the database. Furthermore, the database caches need to be cleared, e.g., by restarting the database server.

Workload Configuration

The workload is controlled via the number and types of clients simulated during the benchmark run. The client setup is retrieved from a configuration file.[1] It specifies the types of clients to simulate and how many of them to run. The configuration file furthermore includes the server and database parameters to set up the connections. Three types of clients can be specified. These are (i) OLTP clients, (ii) OLAP clients, and (iii) mixed OLTP/OLAP clients. Mixed clients behave according to the workload shares as defined in CBTR (Sect. 4.4.3). OLTP and OLAP clients are additional client types that allow for the specification of pure OLTP and OLAP workloads to run concurrently. Here, the OLTP and OLAP shares of the entire workload depend on the number of clients defined.

The configuration file furthermore contains two parameters to adjust the shares for the workload components these clients execute. The parameters control the behavior of clients in the sense of what queries are chosen during the benchmark run. Parameters are the *read-only OLTP share* and the *OLAP share*. Both can be defined within the value range of $[0 : 100]$. The three workload components controlled by these parameters are mixed OLTP, read-only OLTP, and OLAP. Table 6.1 provides an overview of the client types and which workload components they can potentially execute (depending on the configuration of the read-only OLTP share and the OLAP share).

The read-only OLTP share parameter is relevant for OLTP clients and mixed OLTP/OLAP clients and specifies the portion of read-only OLTP queries from the OLTP part of the workload. A share of 100 % has the effect that no mixed OLTP queries are executed. Thus, no new data is written to the database. The workload in this case is purely read-only. Any number $n \in [0 : 100]$ states that n % of the OLTP queries executed are of type read-only OLTP. The OLAP share parameter is relevant for mixed OLTP/OLAP clients only and controls the portion of OLAP-style queries.

[1] See Listing B.1 for the complete contents of the configuration file.

An OLAP share of 0 % turns a mixed OLTP/OLAP client into a pure OLTP client. Reciprocally, an OLAP share of 100 % turns the client into a pure OLAP client.

According to the configured shares, clients obtain their execution instructions from the query queues. *Query queues* contain the OLTP queries and OLAP queries. There are three query queues, one for each of the above mentioned workload components:

- Mixed-access OLTP query queue
- Read-only OLTP query queue
- OLAP query queue

The mixed-access OLTP query queue contains complete insert, select, and update statements that can directly be executed by a client. The reason for this is that a real OLTP workload, which has been retrieved from execution traces and the respective data in the database, is replayed during the benchmark run. The mixed OLTP query queue provides a stream of those queries for OLTP clients that are ready to execute the next mixed-access OLTP query. The read-only OLTP query and the OLAP query queue contain SQL query skeletons that are parameterized by the clients during runtime.

Workload Simulation

In the configuration file, the length of the benchmark run and an optional warm-up time are given. The goal of the warm-up is to let the database pre-load data and fill its caches. During this period, query run times and results are not logged. A warm-up time has to be determined and configured for each database system under test.

After its creation, each client runs in an own thread with its own database connection to simulate an independent user. The database connection is set up by the benchmark controller and managed by the *DB service*. The run of a client consists of two steps: (i) getting the next query to execute and (ii) executing this query.

To get the next query to execute, a client, according to its type, polls the respective query queue. In the case of read-only OLTP and OLAP queries, SQL query skeletons are retrieved from the query queue, which need to be parameterized. Concrete parameter values are chosen from a pre-defined set that contains valid values included in the database. Concrete values to fill the SQL skeletons are chosen randomly.

Once the SQL statement for the query is complete, it is send to the database. The time between the moment of starting to send the query until a response is received from the database is measured and is sent to the *result collector* as an event to be recorded. Meanwhile, the client can immediately proceed with retrieving the next query from a query queue and executing it.

The execution of a client stops if one of the following conditions is reached:

- The maximum number of cycles per client (see run count as specified in the configuration file in Appendix B.1) is reached or the benchmark run time is up,

- This client or another one encounters an error, or
- No more queries are available from the OLTP queue in a workload containing mixed-access OLTP queries.

Once all clients terminate, the benchmark run is over.

Benchmark Log Files

The *result collector* and *logger* receive events from the clients to record measurement results and status log file entries. The results of a benchmark run are stored in a file[2] for later analyses. The result log file contains a header line storing general information about the benchmark run, including the database under test, workload configuration, and individual comments. The rest of the result log file contains one line for each statement executed during the benchmark run. Log file entries for queries that contain multiple statements, such as, mixed-access OLTP queries, occupy several lines. Such a line contains details about the executed query (kind, parameters, statement), statistics about its execution (start time, end time, and resulting run time), the results (number of columns and rows in the result set), and which client executed the query.

The status log files are mainly used to retrieve additional information about a benchmark run, e.g., exceptions that do not disrupt the entire run, but are important for the final benchmark statistics. An example are lock wait timeouts, which abort the execution of a statement after a pre-configured interval. These queries find their representation in the result log file as well. The given response time then, however, has to be treated differently as the execution was interrupted and no results were returned.

In addition to the log files by the benchmark driver, the performance of the database server is monitored throughout the entire benchmark run. For this purpose, analysis tools provided by the operating system of the server running the database are used.

6.2 Visualization of Results

Depending on the length of the benchmark run, the number of configured clients, and the response times achieved by the tested database, result log files can become large with several thousand lines. Two tools are offered as part of the current implementation to visualize results. The first tool produces predefined graphs from the benchmark log files. The second is a monitor to observe and control benchmark runs interactively. Both tools utilize Gnuplot [112] for graph creation.

[2] See Listing B.2 for an excerpt of a result log file.

Result Analysis Reports

The result visualizer component of the CBTR tool chain includes a set of Python [173] scripts to create summary reports from the benchmark result log files. From one log file or a set of log files, graphs and overview tables are created to support evaluation. Available reports are distinguished into ones containing the results of a single benchmark run and others that provide an overview over multiple runs, e.g., with different workload configurations:

- Response time development is the analysis of a single run that gives an overview of the development of the response time of a query over the entire benchmark run as a graph.
- Multi-run statistic reports provide key figures, e.g., median, average, standard deviation etc. of response times, for all given benchmark runs aggregated by statement, query, or workload component. See Table B.1 for an excerpt of this report.
- Series comparison is based on the multi-run statistics report and provides a graphical representation of a chosen key figure from this report (e.g. average response time) over several client configurations. The chosen key figure is marked on the y-axis and the client configuration on the x-axis. This report needs to be customized according to a classification of runs with the same client configuration. The different classes of benchmark runs are shown as series in the graph.
- Throughput is similar to series comparison. It provides an overview of multiple benchmark runs with different workload configurations and series for the comparison of different classes of runs of the same configuration. Classes in this context are, for example, data set size or database schema variants. The throughput is denoted on the y-axis, workload mix is given on the x-axis.

These reports are an excerpt of possible analysis variants and represent the ones that have been used in this thesis. More reports can be added as necessary. Figure 6.3 provides examples for the above introduced report variants. In Fig. 6.3a, the response time development of the days sales outstanding query of one particular client configuration is depicted. Figure 6.3b shows the shipping query of the same client configuration. Each point in the graph represents the run time of one query of the given type. In both cases, it shows that executions of the chosen queries are distributed evenly over the entire run of the benchmark that the distribution of response times stays constant over the entire benchmark run, and at which values response times cluster as a basis for further analyses.

Figure 6.3c, d show examples for two variants of the series comparison. Figure 6.3c displays an example benchmark run where the response times of OLTP requests decrease when adding OLAP clients to a constant baseline OLTP workload. Different baseline OLTP workload configurations are depicted as series in the graph varying from one to several OLTP clients executing queries in parallel. Response times are normalized to the $[0 : 100]$ interval with 100 marking the longest response time in the result set. In Fig. 6.3d, the performance of OLAP requests utilizing different database schemas (series) is shown using a constant number of 100 OLAP clients as the base load and varying the number of OLTP clients running in parallel.

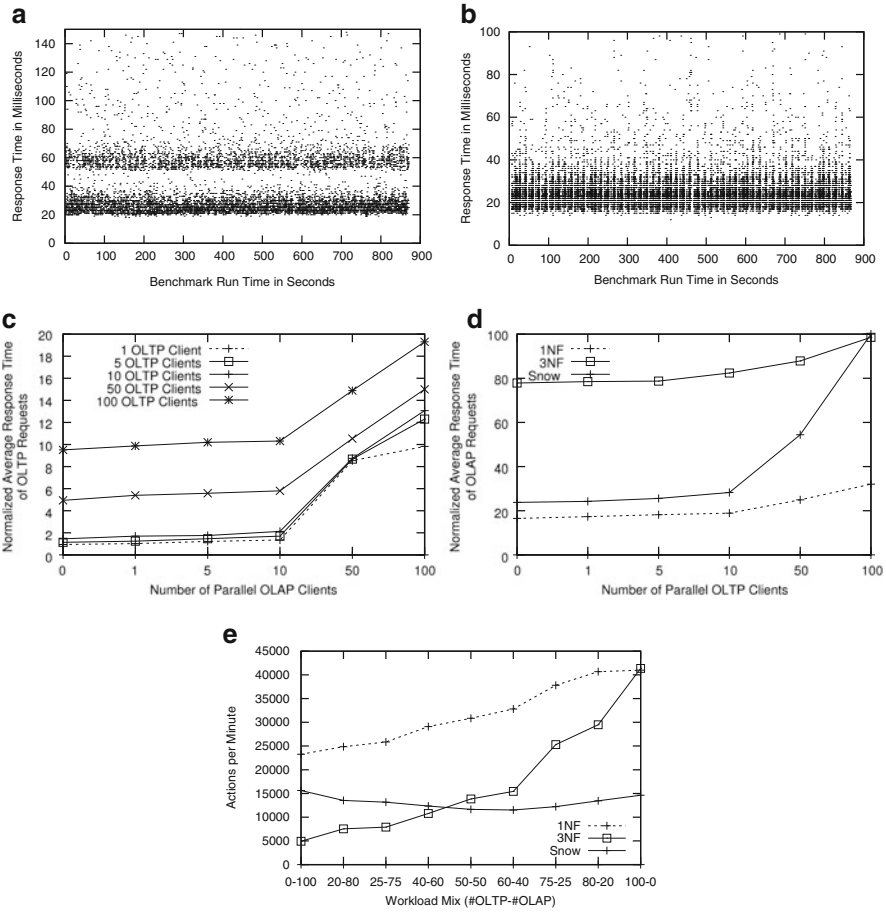

Fig. 6.3 Result analysis report examples. (**a**) Response time development of days sales outstanding. (**b**) Response time development of shipping. (**c**) Series comparison – impact of adding OLAP to OLTP workload. (**d**) Series comparison – schema variation. (**e**) Throughput

Figure 6.3e exemplifies a throughput report. It shows the performance of different database schemas as series depending on varying OLTP and OLAP shares of 100 clients that are running in parallel.

Interactive Monitor

To allow interactive monitoring of system performance, the CBTR driver is extended with a user interface through which the configuration of clients can be changed during run time. New clients can be added and running clients can be stopped and removed. Queries can be selected for which the graphs are drawn.

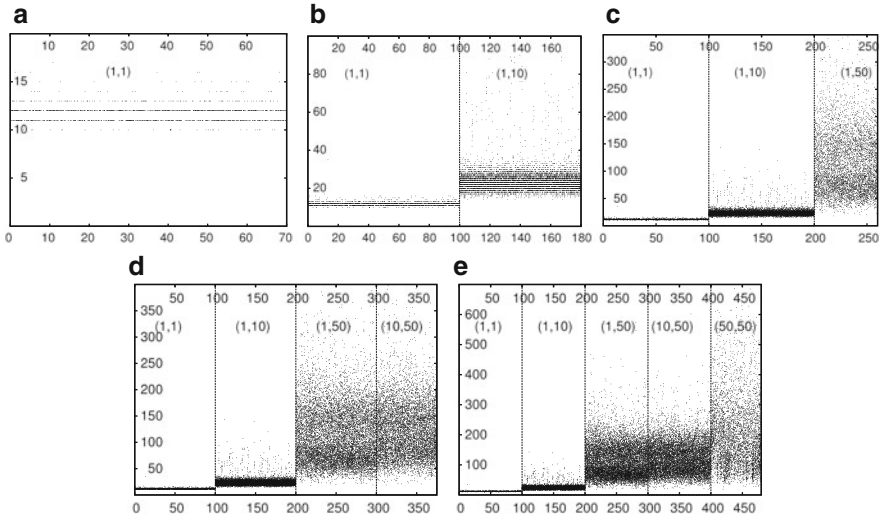

Fig. 6.4 Average order processing time observation (x-axis: benchmark time in seconds, y-axis: response time in milliseconds)

Similar to operating system performance monitoring, the graphs are updated in intervals (configurable length) during the benchmark run. Consequently, the influence of changes in the workload on response times can be experienced immediately.

Figures 6.4 gives an overview of an example benchmark run with interactive monitoring. The workload is modified manually by changing the number of clients every 100 s. Figure 6.4a shows the performance of the average order processing time query under the first workload mix of one OLTP client and one OLAP client, labeled in the graph as "(1, 1)". In Figure 6.4b nine OLAP clients have been added to the mix after 100 s. 40 OLAP clients have then been added at 200 s (Fig. 6.4c) and afterwards nine OLTP clients at 300 s (Fig. 6.4d). The last graph in Fig. 6.4e shows the completed simulation with a final workload of 50 OLTP and 50 OLAP clients. As it can be seen, response times degrade and their variation intensifies with increasing load. Furthermore, through the simulation of different workload mixes, it can be observed that the rate at which performance degrades depends on the types of the added clients. Adding OLTP clients degrades performance to a lesser extent than the addition of OLAP clients.

6.3 Limitations and Opportunities

With the help of the benchmark driver and the additional tools described in this chapter, database systems can be evaluated and compared regarding their performance in mixed workload scenarios. The current implementation of the benchmark

driver constitutes a reactive system for performance evaluation and profiling. In further work, a combination of these benchmark tools with the results from the data schema variation discussed in Chap. 5 can be the basis for automated proposition of database schemas in specific workload situations leading to monitoring of productive systems and adapting database schemas on the fly.

Another future work stream can be found in the area of interactive monitoring. The tool could be extended to interactively observe the impact of data schema variation or changes to physical database design structures, e.g., creation of indexes. Alternative presentations of the results, for example, moving averages, are also conceivable.

Think times of clients between queries are currently not simulated. This keeps the number of requests sent to the database constantly at the configured level to allow throughput measurements. However, to simulate peak and average load times, the benchmark tools could be extended with think times. The multi-tenant setting is an application area for variable load profiles during the simulation. CBTR and its tools could provide the workload to evaluate distribution strategies for tenants based on their loads for automated cluster management, like in [181].

6.4 Summary

The tool chain developed for the CBTR benchmark consists of the driver to run a test, result analysis reports, and the interactive monitor. For the benchmark run, a configuration of the numbers of OLTP, OLAP, or mixed clients can be given. Furthermore, the shares of read-only and mixed OLTP queries have to be defined. Based on the configuration, a specific workload mix is simulated in the benchmark run.

The result analysis reports are a collection of proposals how to analyze the results of one or several benchmark runs, for example, providing an overview of how response times develop during a run, or the comparison of throughput in different workload mixes. The goal of the interactive monitor is to allow database experts direct observation of the impact of workload changes on query response time.

Chapter 7
Evaluation of Mixing the Workload and Variation of the Database Schema

Three fundamental techniques for performance analysis are listed by Lilja [129, Chap. 1]. These are simulation, analytical modeling, and measurements of existing systems. While the last provides the most precise results, it is the most difficult and time-consuming of the three techniques in addition to a limitation of the validity of the derived statements to only the measured system. Additionally, measurements are only possible if the proposed system already exists, otherwise simulation or analytical modeling have to be used [111, Chap. 3].

A simulation is a program that models important features of the analyzed system. An analytical model describes the system mathematically focusing on key aspects. Jain [111] states that to be more convincing, analytical modeling or simulation should be based on previous measurement. The evaluation of the behavior of different database types in this chapter is based on a simulation of the workload, applying the benchmark proposed in Chap. 4, and measurements of response time and throughput metrics for the simulated workload from the real database systems.

The following section gives an overview of the general test setup. Section 7.2 provides evaluation results of the impact of adding OLAP requests to a constant baseline OLTP workload. As the next step, Sect. 7.3 details the behavior of the DBMS under test when different database schemas are used to evaluate if a switch to a different database schema can reduce the negative impact on performance when the load and workload mix changes. Finally, in Sect. 7.4, database schema variants are evaluated under a constant load of 100 clients sending requests in parallel, but shifting workload shares from OLTP-dominated to OLAP-dominated. The chapter closes with a summary of the results.

7.1 General Test Setup

The test landscape for the benchmark runs is composed of two machines. The first is the client machine, which runs the workload simulator that is the benchmark driver. On the second machine, the database server is running. To eliminate influences

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_7,
© Springer-Verlag Berlin Heidelberg 2014

from other applications on both machines, only the basic processes needed by the operating system, network, I/O, and the processes needed for the benchmark are running.

The databases analyzed in the following section, are commercial products used in industry for daily business in different areas. Three DBMS with different data storage characteristics (as discussed in Sect. 2.2.4) are tested: column-oriented in-memory (CM), row-oriented disk-based (RD), and column-oriented disk-based (CD).

Results shown in this chapter are normalized to [0 : 1] per DBMS with 1 marking the longest response time in the result set or the highest throughput. The reason for this normalization is that the focus of the evaluation lies in the application of the proposed benchmark to make observations on behavioral aspects of database types under the variation of workload and database schemas. It is not in the scope of this evaluation to compare the actual performance of different database systems with each other based on the results. Basic optimizations are applied. These include standard indexes on primary key attributes, on foreign key attributes, and on further attributes used in *joins*, *where*, and *group by* conditions.

7.2 Impact of Adding OLAP to OLTP

The goal of this analysis is to evaluate the impact of combining the transactional with the analytical workload. The following questions are posed and answered in this section:

1. What is the impact on query performance when adding analytical queries to the workload, that is, how does the performance of transactions change and to what degree does it decrease?
2. Do databases show different behavior depending on their storage types (row/column-orientation, in-memory/disk-based)?

Simulated workload mixes contain the following combinations of the number of OLTP clients and OLAP clients:

$$\{1, 5, 10, 50, 100\} \times \{1, 5, 10, 50, 100\}$$

Figure 7.1 provides an excerpt of the results of this analysis for the tested DBMS. These tests use the original database schema variant of CBTR in 1NF. The charts on the left depict how throughput behaves when adding OLTP and OLAP clients. The ones on the right show how the response time of mixed OLTP queries changes. The series in both graphs mark the configured base workload of chosen configurations that include between 1 and 100 OLTP clients (legend given below the graphs) running in parallel. The number of OLAP clients specified on the x-axis is added to these. See Table C.1 for a tabular overview of the results.

The measured throughput in all systems increases with the addition of OLTP clients because of sending a larger number of short requests to the database.

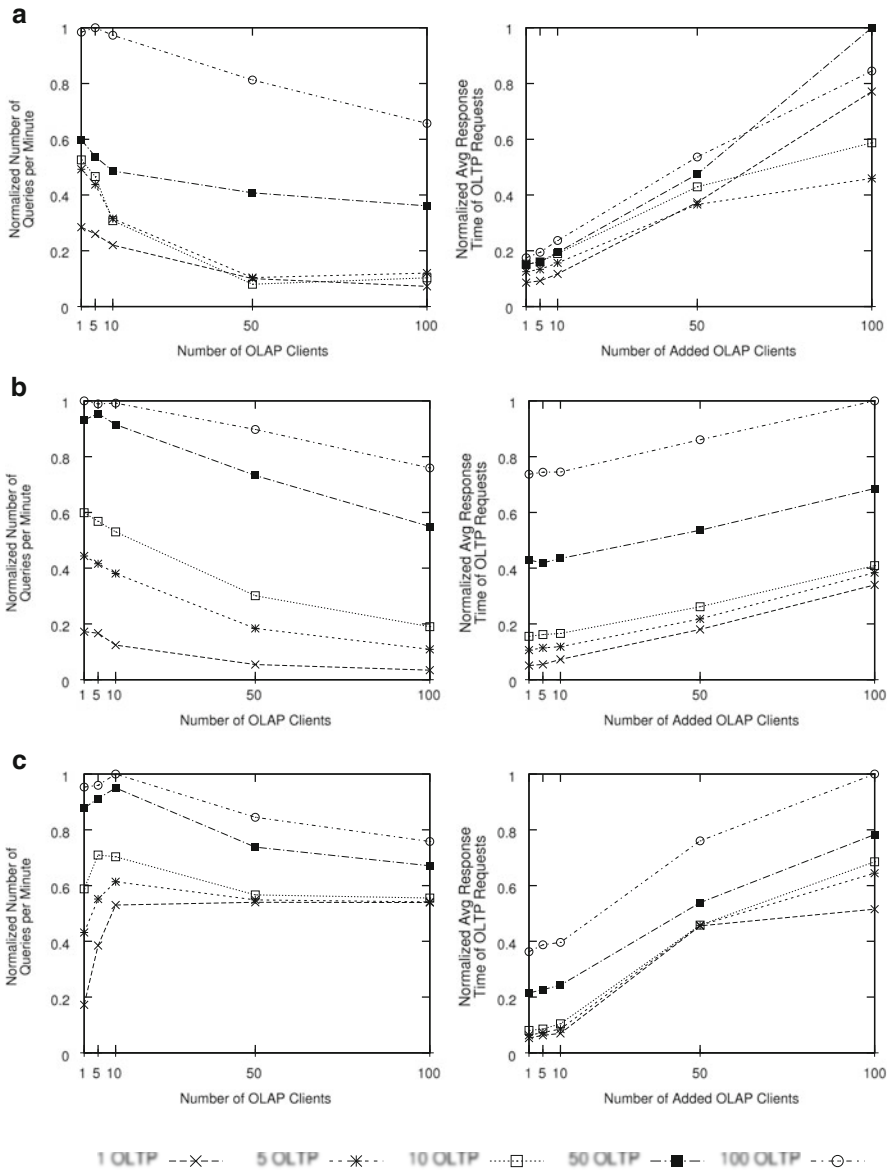

Fig. 7.1 Impact of adding OLAP to OLTP on throughput and response time. (**a**) Row-oriented disk-based system (RD). (**b**) Column-oriented disk-based system (CD). (**c**) Column-oriented in-memory system (CM)

However, in both disk-based systems (Figs. 7.1a and 7.1b), the addition of OLAP clients with their more complex requests that access a larger set of data saturates the system and immediately reduces the throughput of the system. In case of the column-oriented in-memory database (Fig. 7.1c), the addition of a small number

of OLAP clients[1] to the OLTP workload adds to the throughput, but again a larger number of OLAP clients saturates the system and negatively impacts throughput. Compared to the other systems, the throughput of CM decreases to a lesser extent. While throughput reduction of up to 80 % is encountered in CD and RD, in CM the throughput is reduced by 30 % at most within the series of OLTP clients that encounters the largest deviation.

Concerning the response time, the addition of OLTP clients in system CD has a larger impact compared to the other two. In CD, executing 100 OLTP clients increases the average response time of OLTP requests by a factor up to 15. The response times of the same requests increase by factors less equal three in case of RD and factors less equal seven in case of CM.

The reason behind the response time decreasing so drastically in case of CD is that the observed operations include many single inserts of tuples that contain a large number of attributes. The seen behavior can be explained by the fact that in disk-based column-stores the cost of tuple insertion increases with the width of the tuple as distinct locations on disk have to be updated for each attribute contained in the inserted tuple [1]. In comparison, row-oriented systems have to update the one location on disk where the tuple is added to the table and write the new data sequentially.

A similar effect, but less pronounced through the advantage of in-memory random access can be observed in in-memory column-stores. In case of CM, the database system uses a mechanism for the insertion of fresh data, which does not immediately compress the new data and defers the updates of the columns. The storage is separated into a write-optimized differential buffer and the read-optimized main store, like introduced by Stonebraker et al. [200]. The differential buffer collects the inserts and updates. It is merged into the main store periodically or if pre-defined thresholds are met. Therefore, the column-oriented IMDB under test here does not suffer as heavily from the insertion of wide tuples.

From the perspective of adding OLAP clients, an increase of response time by factors less equal seven (CD), factors less equal eleven (RD), and factors less equal ten (CM) can be observed. This increase is much higher in situations with low OLTP load where OLTP response times are at their optimum than in situation with high OLTP load where response times are already increased. In high OLTP load situations, increases by factor less equal two for CD and CM, and factor less equal six for RD are observed.

7.3 Impact of Database Schema Variation

The question to be answered in this evaluation is whether the variation of the database schema can reduce the negative impact on query performance when mixing OLTP and OLAP. The database schemas analyzed here are the original schema

[1] Determination of the exact turning point has not been the focus of these measurements.

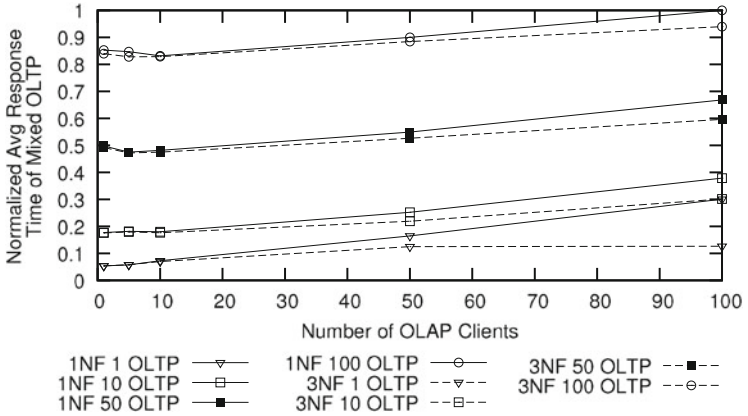

Fig. 7.2 Impact of change from 1NF to 3NF in CD for mixed OLTP requests

in 1NF introduced for CBTR and the three schema variants specified in Sect. 5.2. These include two OLAP-style variants, called document and snowflake and one OLTP-style variant in 3NF that is of a higher normalization level compared to the database schema provided in CBTR.

Figure 7.2, depicts the same context as the graphs in the previous section with the addition of a second database schema focusing on the mixed OLTP workload component. It provides a first glimpse of the impact of changing the database schema in system CD from the perspective of mixed OLTP requests. The second database schema depicted as additional series (dashed lines) is the variant with a higher level of normalization (3NF). As can be seen, for mixed OLTP requests the database schema in 3NF performs just as well or slightly better than the 1NF database schema in low OLAP load situations, but outperforms it in situations with OLAP load. For example, by using 3NF, mixed OLTP request response times are by factor 2.6 lower for 1 OLTP and 100 OLAP clients running in parallel.

To make a decision if the database schema should be kept or changed, the other components of the workload have to be considered as well. The conclusion that 3NF is optimal for mixed OLTP queries from the previous chart is not sufficient to make a decision on one or the other database schema as it completely neglects the OLAP part of the mixed workload.

Figure 7.3 provides an excerpt of the performance measurement results for three DBMS of different types. See Table C.2 for a tabular overview of the results. The measurement results for each DBMS are clustered into four graphs. The graphs on the left side show performance changes from the perspective of mixed OLTP requests, the ones on the right show the impact on OLAP response times. The x-axis provides the number of OLAP clients (left part) and OLTP clients (right part) running concurrently and the y-axis provides the average response times of mixed OLTP request (left part) and OLAP requests respectively for the right part normalized per DBMS. The upper graph for each database shows a configuration

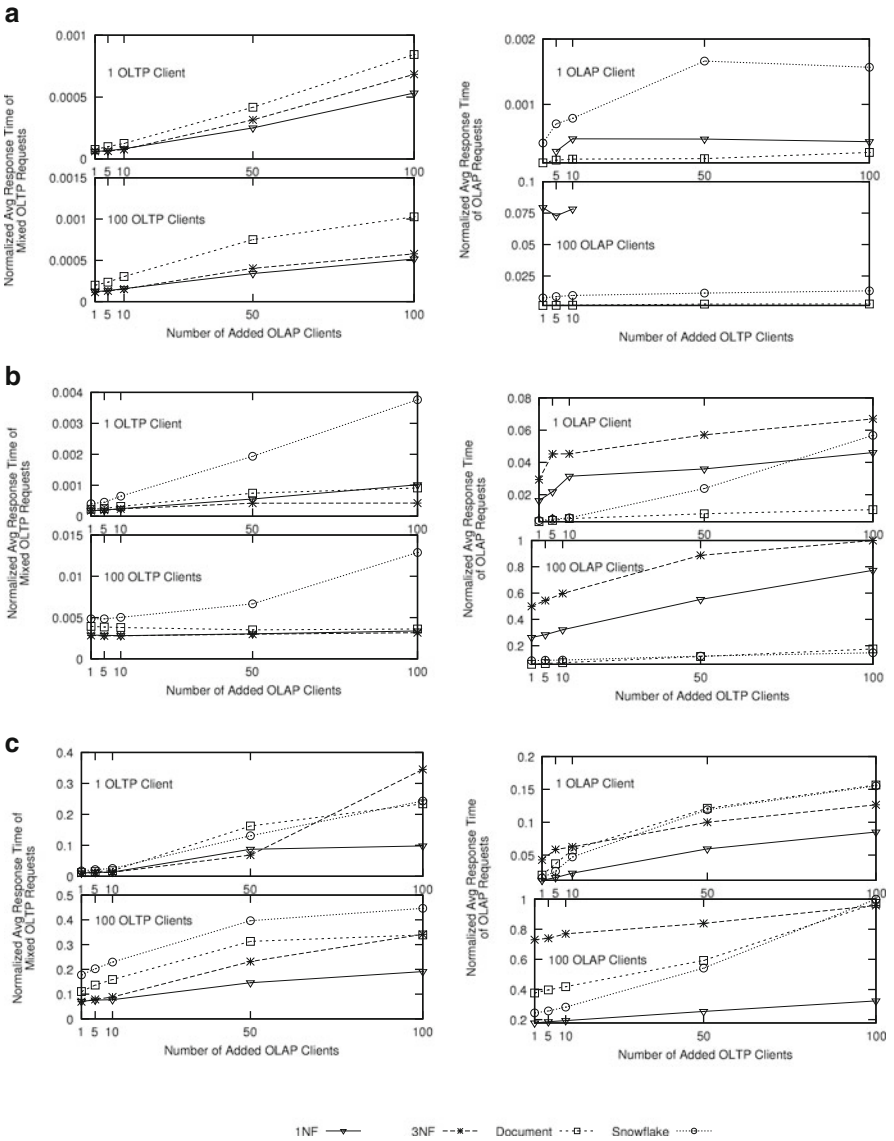

Fig. 7.3 Impact of changing the database schema on response time. (**a**) Row-oriented disk-based system (RD). (**b**) Column-oriented disk-based system (CD). (**c**) Column-oriented in-memory system (CM)

with low OLTP base load (low OLAP base load on the right). Here, only one OLTP client runs in the background. The lower graphs display a configuration with high OLTP base load, that is, 100 OLTP clients running in the background (high OLAP base load respectively on the right side). The different database schemas are depicted as series in the graphs.

As previously mentioned, the 3NF database schema is optimal for the mixed OLTP workload in system CD. However, when analyzing the OLAP part of the same workload (see Fig. 7.3b), 3NF yields the worst performance. As can be seen the queries benefit from database schemas that include special structures to support analytics, like the pre-joined tables. Obviously, a change to OLAP-style database schemas impairs the performance of OLTP requests. Depending on, e.g., priorities of workload components a slight increase of response times for the requests of one component might be reasonable if the benefits for the requests of another component outweigh these drawbacks. Regarding these specific measurements, a switch to the document database schema would benefit OLAP operations resulting in the decrease of average response time by factor 6 compared to the 1NF database schema and factor 11 compared to the 3NF database schema. For the mixed OLTP requests, the respective change to the document database schema would entail an increase of average response times by factor 1.2 if switched from 1NF and factor 1.4 if switched from 3NF. This increase might be negligible in comparison, depending on priorities as previously mentioned.

In case of the row-oriented disk-based DBMS RD (Fig. 7.3a), the document database schema is also a good compromise for mixing the workloads. For requests of the mixed OLTP component, performance of requests using this schema is close to the performance based on the normalized database schemas 3NF and 1NF. Switching to the document database schema would entail an increase of average response times by factor 1.7 on average compared to 1NF and 1.6 on average compared to 3NF for mixed OLTP requests. The performance of the snowflake schema is so low that the measurement results are omitted from the mixed OLTP graphs for reasons of clarity and comprehensibility. On the OLAP side, switching to the document database schema achieves a decrease of response times. OLAP requests are sped up by factors ranging between 2 and 50 when switching the database schema from 1NF to document-style, depending on the workload mix (on average by a factor of 22). For enhanced readability of the graph, OLAP request response times for the 3NF database schema and some results for the 1NF database schema have been omitted because they are up to two orders of magnitude lower.

Regarding the in-memory column-oriented DBMS CM (Fig. 7.3c), 1NF is the favored database schema for the mixed OLTP workload component as well as the OLAP workload component. Because of its architecture, this DBMS is able to manage transactional workloads as well as analytical workloads and pre-computed reporting tables become obsolete.

Figure 7.4 shows the impact of changing the database schema from the perspective of the throughput in pure OLTP (charts on the left side) and OLAP workloads (charts on the right side). See Table C.3 for a tabular overview of the data. The throughput given on the y-axis is normalized per DBMS.

The throughput of systems CD (Fig. 7.4b) and CM (Fig. 7.4c) is much higher in pure OLTP scenarios as a result of the short and simple queries contained in this part of the workload (compared to the OLAP queries). In accordance with the previous observations on response times, the 1NF and 3NF database schemas provide the highest throughput in the OLTP case. In all cases, the saturation throughput

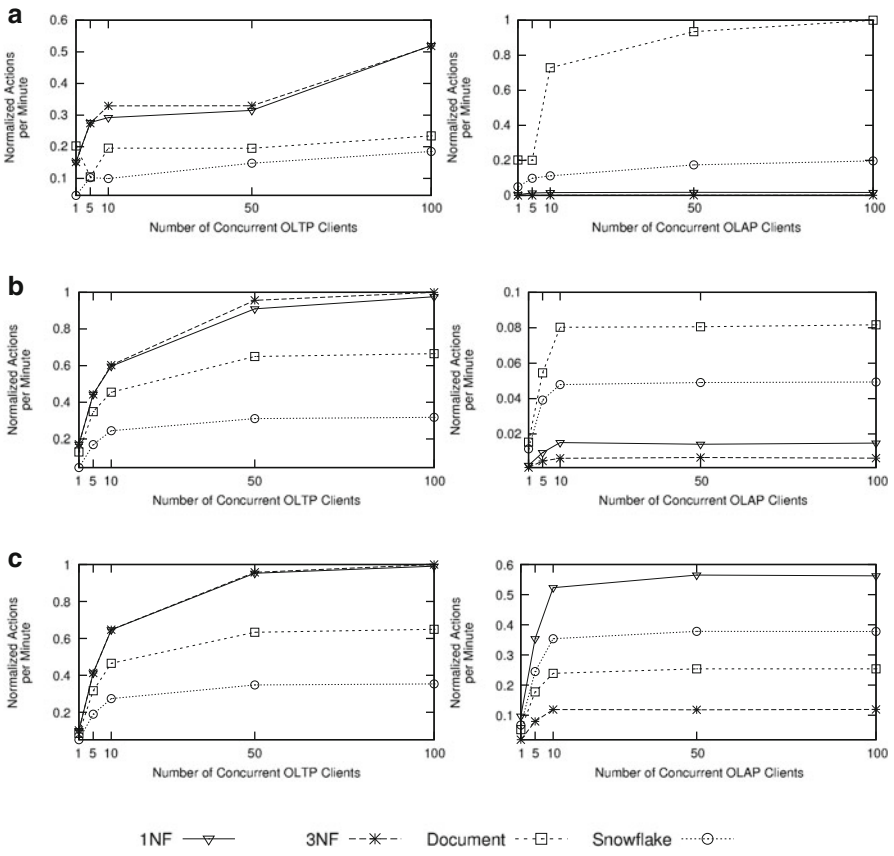

Fig. 7.4 Impact of changing the database schema on throughput. (**a**) Row-oriented disk-based system (RD). (**b**) Column-oriented disk-based system (CD). (**c**) Column-oriented in-memory system (CM)

is reached earlier for pure OLAP workloads as queries are more complex, e.g., they include aggregations and sorting operations, and the computations are based on a larger set of data. In contrast to CD and CM, RD has a higher throughput under the OLAP workload. The particular observation here is that the Clearing OLTP query impairs the performance of the entire system due to the update operation it includes and the locks used on the respective table during the update. Furthermore, this system profits from the pre-joined tables of the document-style database schema.

The snowflake database schema, which also provides pre-joined tables, however, trails behind the document schema in all measurements although it provides the best fit of pre-joined tables with respect to the most complex queries contained in the OLAP workload component. Its disadvantage is that the simple queries that would only access one or two tables also need to access these large pre-joined tables and, thus, they are decelerated, negatively affecting the entire mixed workload.

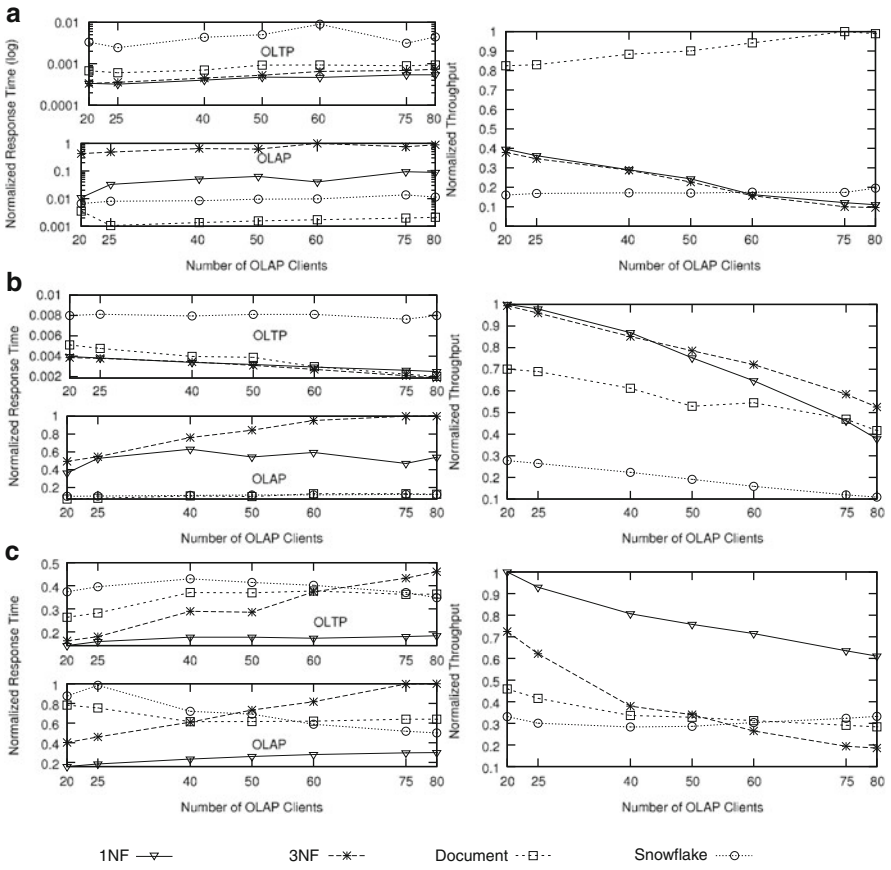

Fig. 7.5 Database schema variation in mixed workloads. (**a**) Row-oriented disk-based system (RD). (**b**) Column-oriented disk-based system (CD). (**c**) Column-oriented in-memory system (CM)

7.4 Database Schemas Under Varying Workload Mixes

In this analysis, the workload mix is varied with a constant number of total clients. The goal of this analysis is to observe how throughput and response times behave on top of different database schemas when workload shares are varied but the total load stays constant (in contrast to the previous section).

Figure 7.5 provides an excerpt of the measurement results. In this test, 100 clients are simulated in total. The x-axis specifies the number of OLAP clients and the difference to the total number of 100 clients is filled up with OLTP clients. The graphs on the left side show the normalized average response times of requests including all three workload components split up into the OLTP requests (upper part) and OLAP requests (lower part). The graphs on the right show

the normalized throughput (depicted results are normalized per DBMS). Tables C.4 and C.5 provide a tabular overview of the data.

The 1NF database schema produces the lowest response times for OLTP as well as OLAP requests and the highest throughput in DBMS CM for all tested workload mixes (see Fig. 7.5). Response times are only slightly deteriorating by factor 1.3 at most for OLTP requests and factor 1.9 at most for OLAP requests with the increasing share of OLAP requests in the workload. Throughput decreases by almost 40 % with the shift to an OLAP dominated workload, which is natural, as the requests by themselves process more data and are longer running. A client waits longer for the completion of a request and as a result, new requests enter the database less frequently. Consequently, a reduction of throughput can be observed. The reason is the constant number of clients. Reaching any system boundaries does not cause the throughput reduction. CBTR can be used for the assessment of system boundaries, but it was not in the scope of this evaluation.

Despite the fact that the 1NF database schema is favored for this DBMS, the behavior of requests on top of the other database schema variants should be noted. The performance of the OLTP-style database schema 3NF decreases with an increasing share of OLAP requests. In both OLAP-style database schemas, response times of requests even decrease when tilting the workload toward OLAP.

In system CD (Fig. 7.5b), from the OLTP perspective the performance of the document-style database schema improves compared to 1NF and 3NF with an increasing share of OLAP requests in the workload. Furthermore, it outperforms both OLTP-style database schemas for the OLAP part of the workload. The reduction in throughput when increasing the OLAP share, while the response times for the document schema are only slightly changing is again caused by the greater number of longer running OLAP requests occupying the clients.

In the row-oriented disk-based system, varying the workload mix has no impact on the choice of the database schema. The OLTP-style schemas remain favored from the OLTP perspective even with an increased share of OLAP requests. The same applies to the OLAP-style database schemas for the OLAP perspective. No switchover from one database schema performing well to another at a certain turning point in workload shares can be observed.

The throughput graph for DBMS RD, however, shows the document schema as a clear front-runner from the perspective of the entire workload. For this database schema, throughput even increases with higher OLAP shares in the workload mix. The reason lies in the extreme benefit on response times that OLAP requests gain from a switch to this schema, while it is not the favored schema for OLTP workloads. With a declining number of OLTP requests in the workload the document schema approaches its sweet spot, which lies in OLAP dominated workloads. Again, the snowflake-style database schema lags behind even in OLAP dominated workloads because the OLAP part of the workload also contains simpler requests that only access parts of the large pre-joined tables and are thus hindered.

7.5 Summary

The first evaluation of (i) adding an increasing number of OLAP clients to a base workload of OLTP clients and (ii) increasing this base line that is the number of OLTP clients shows that the negative performance impact of increasing the baseline number of OLTP clients is higher for the column-oriented DBMS than in the row-oriented DBMS. All three tested DBMS show a similar behavior concerning the negative impact on response times through the addition of OLAP clients.

The next question analyzed in the evaluation was if the negative impact can be decreased or can even be avoided through a switch to a different database schema. The results show that this question cannot simply be answered with "yes" or "no", but OLTP and OLAP as the two components of the workload have to be considered individually. While the switch to a different database schema can benefit the requests of one workload component, it may impair the requests of the other component and yet another schema may be the optimum for this component. This leads to the question how a solution can be found based on conflicting observations.

The results for the row-oriented disk-based database system, for example, identify the document database schema as a compromise for the mixed workload resulting in an increase of response time by factor 1.7 on average for OLTP requests, but a decrease of response times by a factor of 22 on average for OLAP requests when switching from 1NF. However, considering workload mixes with high OLTP shares with thousands of requests per minute and low OLAP shares in comparison, a response time increase by a factor of less than two per OLTP transaction results in a larger degradation than the benefit caused by the larger factor speed-up of a small number of OLAP requests.

A quantifier needs to be included in considering the benefits for the entire workload. Priorities could provide such a quantifier to allow making a decision for or against a specific database schema. Those priorities could be defined by the domain experts of a company who are familiar with the critical business paths and know which workload components (or queries on the lowest level) have to be assigned a high priority. Alternatively, priorities could be assigned automatically by monitoring the workload and determining queries that are responsible for the largest share of the load.

Chapter 8
Conclusion

Benchmarks are the standard method to evaluate, compare, and support the development of database systems. With the unification of transaction and analytical processing systems and the development of new database systems, benchmarking has to evolve as well. On the one hand, existing benchmarks have so far focused on either transaction or analytical processing because of the separation of the two domains. CBTR is the first benchmark to simulate mixed workloads and analyze the behavior of database systems under these conditions. On the other hand, the traditional separation of OLTP and OLAP along the characteristics of static, write-intensive OLTP with its many short transactions and read-only OLAP with its long complex queries is foundering. Applications have emerged that do not fit into either category and the workloads themselves are changing, as companies want to remain competitive. This change in workloads has to be reflected in benchmarks as well to stay relevant.

The unification of OLTP and OLAP entails the need to reevaluate optimizations so far applied in analytical and operational systems in mixed workload situations. These optimizations have conflicting goals. An assessment of their impact on all workload parts is needed to make a decision on which optimizations to employ in a specific database and operating a specific workload mix.

The contribution of this thesis is twofold. In the first place, the new benchmark CBTR has been introduced. Its goal is the assessment of hybrid OLTP and OLAP database systems. Along with the new benchmark, a prototypical implementation to run the benchmark and evaluate its results has been developed. Second, logical database design optimizations so far applied in OLTP and OLAP environments were examined and their effects were analyzed in different database systems using CBTR. Thus, the new benchmark provides means to assist database experts to test and choose optimizations with respect to specific workload requirements.

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9_8,
© Springer-Verlag Berlin Heidelberg 2014

Benchmark Creation

For the creation of CBTR as a benchmark for mixed OLTP and OLAP workloads, existing benchmarks have been assessed in the context of today's transactional and analytical workloads. The benchmarks used widely in research and industry today do not reflect current enterprise workloads. In respect thereof, relevance as one important criterion to make a benchmark valuable for research and industry is not fulfilled any longer.

CBTR was created based on the realistic data structures, operations and workload of a widely used enterprise system, instead of using an existing benchmark as the foundation, to gain relevancy. The queries within CBTR are based on observations from companies. Experiences and lessons learned from the existing benchmarks have been taken into consideration throughout the development of the new benchmark, e.g., simplicity of the scenario to ensure comprehensibility, verifiability, repeatability, and fairness.

To enable testing of varying workloads of OLTP and OLAP shares, workload mix has been introduced as an additional benchmark parameter besides data set size and load.

Analysis of Logical Database Schema Optimizations

The benchmark has been applied in evaluating database design decisions so far used to optimize for one or the other dedicated workload. As a prerequisite to evaluate the impact of database schema variants as part of logical database design, the key differentiators facilitating today's optimized database schemas for OLTP and OLAP have been analyzed and summarized. These are mainly in the area of (de-) normalization, that is, redundant data, derivable data, repeating groups, pre-joined tables, or report tables.

Three database schema variants have been defined utilizing or avoiding the found differentiators. These database schema variants have been tested using CBTR to determine the performance impact of the found optimizations in different OLTP and OLAP workload mixes.

Impact Evaluation

The observed behaviors of databases in the evaluation facilitate the validation of the benchmark. As the workload mix is varied, databases start to behave differently. This change of behavior can be explained based on the query processing strategies of those databases and further knowledge about their data storage internals.

The application of CBTR yielded results that provide insights into database behavior from different perspectives and show that the found database schema optimizations have a significant impact on query and transaction performance with up to two-digit speed-up factors. The different evaluation scenarios were:

- Addition of different OLAP workloads to a constant base OLTP workload to determine the performance impact of increasing the load of an OLTP system.
- Different database schema variants to evaluate the impact of database optimizations such as normalization, aggregates, or pre-joining of tables.
- Different database storage types to analyze if this factor is relevant for choosing database schema optimizations.
- Variation of the OLAP and OLTP shares within a constant number of clients to examine if database schema optimizations have different effects in different workload situations.
- Response time and throughput filtered by the OLTP and OLAP perspective to investigate if an optimization impacts individual components of a mixed workload differently.

The evaluation of adding an OLAP workload to an existing OLTP workload has shown a similar behavior for all tested database systems. Performance degrades compared to a pure OLTP workload scenario. Yet, the benefits outweigh the drawbacks as discussed and performance degradation is less pronounced in settings with an already high OLTP workload. For instance, OLTP response times increase by factors between 2 and 6 in a setting with 100 concurrent OLTP clients as the base load, whereas response times of OLTP requests in low load situations increase by factors up to eleven.

Concerning database design optimizations, different behaviors were observed (i) in the different database systems tested and (ii) under diverse workload mixes. The same database schema affects query performance differently (positive or negative) depending on the workload mix and the DBMS used. For example, in the tested disk-based column-oriented DBMS the normalized database schemas yielded best performance for OLTP queries, while the document database schema accelerated the OLAP queries. In the in-memory column-oriented DBMS, the same database schema (1NF) provided best results for the OLTP and OLAP queries compared to the given alternatives. This behavioral variance underlines the value of the developed benchmark, which provides the method to test and analyze a database system regarding a specific workload mix and creates a basis for optimization decisions.

8.1 Discussion

The results of the evaluation have shown that to make a decision for or against a specific database system or optimization for a workload, all aspects of the mixed workload have to be considered in their combination. An average speed-up by factor 22 for the OLAP requests of a defined scenario that accepts a slight impairment of OLTP request response times by factors less than two sounds promising from the perspective of single requests. This argumentation may become flawed. The “may” in the previous statement and whether the decision to speed up OLAP at the cost of

OLTP performance really is wrong depends on the user requirements of the OLTP system and the actual response times of OLTP requests.

As an example, a company whose representatives rely on OLTP response times because of being in direct customer contact, e.g., on the telephone, is considered. In this situation, the possibility to speed up OLAP at the cost of OLTP performance has to be judged carefully. As long as the increase of response times stays imperceptible, e.g., is in the range of micro up to milliseconds, the potential given by the threshold of human perception can be used to enhance OLAP run times. For example, the increase of response time for the entire execution of an OLTP request and its display to the representative is acceptable for the representatives as long as it stays between half a second and at most a second after an optimization for OLAP requests is employed. A huge benefit for the OLAP users is created if such an optimization pushes response times of OLAP requests below the threshold of one second. "Delays of longer than one second will seem intrusive on the continuity of thought." [22]. Nielsen [153, Chap. 5] explains, that the threshold of one second is the limit for a user's flow of thought to stay uninterrupted, although the users notices the delay. Yet, at this threshold the OLAP users can start to work interactively without the system itself being the reason that they become distracted.

As soon as the speed-up of OLAP causes side effects that diminish OLTP throughput or violate response time requirements, the decision has to be reevaluated keeping in mind the business consequences, for example, stalled business processes or reduced customer satisfaction if responses are not received immediately. The raw query performance of specialized database systems that are developed for either OLTP or OLAP currently prevails compared to the approach of a combined database system. At one point, however, the decision has to be made if a landscape with dedicated systems will be set up or if an integrated one is desired.

CBTR provides a basis to assess emerging systems regarding their ability to cope with given workloads. Which workload share is actually defined for a measurement depends on the party responsible for the measurements. In CBTR, workload shares can be flexibly configured and thus they can be adapted to the requirements of any company interested in assessing a specific scenario. To further guide the decision-making process, a number of factors have been discovered in the discussions throughout this thesis and during the application of the benchmark in the evaluation of database schema optimizations. These factors have to be weighed to reach the right decision according to given requirements.

Having a single source of truth like in a hybrid system restricts the introduction of optimizations within data structures because an optimization positively affects either OLTP or OLAP performance. Thus, optimizations have to be carefully chosen, e.g., based on performance thresholds and workload shares as discussed above.

Data extraction and synchronization is a necessary step to update data in an analytical system that is separate from the operational system. Data preparation as used in current analytical systems allows the introduction of optimizations to speed up reporting. Finally, a separate analytical system and its ETL creates redundant data that has to be managed. Increased resource usage through keeping redundant data sets and running the ETL tasks is the smallest portion of the costs this introduces.

Setting up the ETL process, operating, and maintaining it, causes the major share of costs. In a unified OLTP and OLAP system, ETL is no longer necessary.

Data latency is the delay between the creation of data in the operational system and its availability for analytical processing. In a hybrid OLTP and OLAP system, this latency does not exist, as the actively used data source for both workload parts is the same. In current analytical systems, data latency can range from real-time availability of data to minutes, hours, days, or weeks. Real-time availability of data in the analytical environment is bought dearly in the operational environment. One strategy is to extend the transactions in the operational systems to include the synchronization of data to the analytical system to achieve up-to-date data. Naturally, this introduces overhead to transaction processing. Another strategy is to synchronize the data immediately upon the commit of a transaction. Yet, the interval to transport data and each data preparation step to enhance data for improved reporting performance introduces a delay in which data ages.

Analytical flexibility is restricted in current analytical systems as data is prepared to serve specific reporting needs. If new needs arise, existing structures have to be adapted or new ones have to be created to serve them. The single source of truth in a hybrid system that efficiently processes transactional as well as analytical requests readily allows for the usage of any data item for reporting.

8.2 Future Work

The benchmark proposed in this work and its application in the evaluation of database schema variation provides a foundation for the elaboration of strategies to adjust database optimizations in mixed workload scenarios. Here, more research is necessary to determine the factors that influence the decision if an optimization in a specific workload mix is introduced or not. Basic factors such as the perceptibility of changes that greatly improve the performance of one workload part, but slightly impede the performance of the other workload part, or the number of users in the workload shares and the impact of changes as seen from the sum of all these users have been discussed in this work. Using these factors and priorities for them, rules could be defined that control the usage of optimizations and the adaptation of database schemas according to changes in the mixed workload going into the direction of self-tuning databases for varying mixed OLTP and OLAP workloads.

The database schema of the proposed benchmark is in first normal form in its current version, which is based on observations from real enterprise systems. A lower degree of normalization implies additional overhead for OLTP to avoid anomalies, e.g., through triggers that automatically update redundant data or aggregates. Anomaly avoidance has not been considered in the current version of the benchmark and no triggers have been defined so far. The logic to update redundant data and aggregates is part of the business logic included in the transactions as taken from the enterprise systems at the moment. These are keeping the data set valid. This strategy has to be reevaluated and the benchmark has to be adapted if necessary.

One approach is to define the triggers and introduce conformance requirements that every benchmark run has to adhere to. Another approach is to exchange the database schema with one of a higher degree of normalization to avoid data anomalies.

During the course of this work, the idea to include database schema variation as another dimension besides data set scaling and workload mix surfaced. Consequently, the tool chain was designed in such a modular way that database schema, transactions, and queries can be adapted or extended easily. Furthermore, the database schema at a higher degree of normalization and the adaptation of queries has already been defined as part of the evaluation of the impact of database schema optimizations and the benchmark has been run on top of this database schema. Thus, the exchange of the database schema has been implemented and could be applied immediately.

CBTR uses throughput and response time as its current metrics and, thus, provides performance measurement results. Performance is not the only dimension of interest, though, and more measures, e.g., in the context of energy consumption like in the TPC benchmarks, have to be presented to achieve a more complete picture of a system.

As shortly raised above, restriction of optimizations is a topic that has to be elaborated concretely. If results of benchmark runs are supposed to be fair and comparable, especially if the tests are undertaken by different companies or research groups, rules are necessary that restrict the available optimizations and methods are needed to control their adherence, like the detailed reports in existing standard benchmarks.

A research challenge that is not covered by hybrid or mixed workload OLTP and OLAP systems themselves is reporting across several transactional systems and other or even external data sources. This is a feature of major importance in today's analytical systems in large enterprises. Strategies, e.g., federated query processing, might be introduced on top to simulate information extraction and integration from multiple source systems.

CBTR and its tool chain developed as part of this thesis have been used already in two industry projects to determine the overhead of virtualization in raw database performance. In another industry project, CBTR is currently extended to simulate and evaluate multi-tenancy mixed workload environments. Thus, the work in this thesis has already provided a basis for evaluation projects in the industry and sparked new work in the area of benchmarking.

Appendix A
Related Activities and Publications

Hybrid Architecture for OLTP and Operational Reporting

In [180], published at the International Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE), a first architecture that services operational reporting on top of up-to-date transactional data was proposed. This architecture is based on a row-oriented database that serves the conventional OLTP load and a connected column-oriented data store, which holds a subset of the data in the row store that is needed for operational reporting. Both data stores have the same logical structure and to achieve up-to-date data in the column store it is updated within the transaction. To allow for consumption of the transactional data in the column store virtual cubes are proposed as a method for conventional warehousing environments. Based on this work, a patent was filed in September 2008 [168].

Combining OLTP with OLAP

In [182], a detailed discussion motivates the use of unified, enterprise-wide data stores, which allow for ad-hoc operational reporting on real-time business data without the need of data extraction and loading into dedicated systems. An analysis of the characteristics of real-time ATP as an example for a typical enterprise application that can leverage in-memory column-oriented data storage to provide enhanced functionality, e.g., real-time order rescheduling through on-the-fly aggregation for every processed customer request, is provided in [206]. The idea of navigational SQL, a language that provides constructs specifically tailored for data retrieval in the context of operational reporting, such as a navigation operator that has explicit

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9,
© Springer-Verlag Berlin Heidelberg 2014

knowledge about join paths in a particular schema and gives the user an application level view of the underlying tables has been introduced in [78]. A foray into a different but adjacent direction of enterprise data management, looking into typical operations in the area of multi-tenant databases and an analysis of how they are wired together, was published in [79].

Appendix B
Implementation

Table B.1 gives an overview of the multi run statistics report. The response times are provided in milliseconds. The key figures are:

- Schema: benchmark run variant for schema impact evaluation
- Clientconfig: *a-b* with a as the number of OLTP clients and b the number of OLAP clients
- Lquart/uquart: lower quartile/upper quartile
- Max95: the 95th percentile
- Avg95: average without the 5 % highest values

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9,

© Springer-Verlag Berlin Heidelberg 2014

Table B.1 Reported key figures over multiple runs

Schema	Clientconfig	Min	Uquart	Median	Oquart	Max95	Avg	Avg95	Stddev	Variance
1NF	20–80	7	116	178	254	393	194.63	179.55	110.39	12,187.02
1NF	40–60	7	103	157	228	371	176.84	161.39	109.91	12,079.72
1NF	50–50	8	84	139	211	345	158.47	144.36	101.86	10,375.36
1NF	60–40	7	67	113	183	322	137.66	123.07	101.34	10,270.06
1NF	80–20	6	35	60	112	279	99.16	79.73	139.93	19,581.33
3NF	20–80	9	154	327	828	1,639	554.02	475.37	534.75	285,959.14
3NF	40–60	8	130	230	565	1,257	402.42	342.25	402.05	161,640.27
3NF	50–50	9	110	186	352	1,197	334.53	270.22	383.75	147,264.04
3NF	60–40	9	96	151	231	832	230.58	176.19	281.28	79,116.91
3NF	80–20	10	78	119	176	287	136.20	124.46	84.03	7,060.67
Star	20–80	12	181	306	468	822	355.86	319.93	243.70	59,389.95
Star	40–60	12	212	342	519	910	401.52	360.28	273.56	74,835.96
Star	50–50	15	246	387	583	1,096	461.65	410.32	324.37	105,216.24
Star	60–40	17	235	376	566	1,101	454.51	398.33	345.46	119,343.25
Star	80–20	23	289	453	651	1,038	500.18	456.03	311.16	96,823.63

```
# database meta information
host = <host ip>
dbname= <name of the database>
dbuser = <database user>
dbpassword = <password for database user>
database = <name of the database product, e.g. mysql>

# location of the query skeletons and parameters
trans_dir = <directory>

# workload configuration
number_mixed_clients = <0..>
number_OLAP_clients = <0..>
number_OLTP_clients = <0..>
OLAP_share = <0..100>
rOLTP_share = <0..100>

# benchmark run configuration
run_count = <number of maximum queries per client>
early_terminate = <true/false>
warm_up_time = <time in seconds, 0 if to be skipped>
benchmark_run_time = <time in seconds, if 0 run_count
                      terminates the benchmark run>

# date and time format that the database uses
date_format = <yyyy-MM-dd>
time_format = <hh:mm:ss>

# comments to include in the result output
data_set = <...>
comments = <...>
```

Listing B.1 Benchmark driver configuration

```
#NumClients[Mix-Trans-Analyt];Database;OLAPShare;ROLTPShare;NumRuns;runID;
     DataSet;
          IndexComments
*2[0-1-1];xdb;0;50;1000000;2011-12-11_08-45-23;
          1NF;
#testRunID; clientID; actionName; starttime;
          endtime; runtime; columns; rows; statementtype; actionID;
          actionParams
2011-12-11_08-45-23;0;[Action];OLAP;DailyFlash;120313;
          120317;4;4;2;SELECT;8509;
          '2003-11-26'
2011-12-11_08-45-23;1;[NewSalesOrder_2004-01-07_04:05:31];wOLTP;NewSalesOrder
     ;120308;
          120320;12;0;1;INSERT;0;
          INSERT INTO VBPA VALUES ('800','210000009720','000000',...
2011-12-11_08-45-23;0;[Action];OLAP;OrderDeliveryFullfillment;120317;
          120331;14;6;4;SELECT;8510;
          '2001-04-01' and '2001-04-30'
2011-12-11_08-45-23;1;[NewSalesOrder_2004-01-07_04:05:31];wOLTP;NewSalesOrder
     ;120332;
          120337;5;165;1;SELECT;0;
          SELECT * FROM KNA1 WHERE KUNNR = '0000001033'
2011-12-11_08-45-23;1;[NewSalesOrder_2004-01-07_04:05:31];wOLTP;NewSalesOrder
     ;120337;
          120338;1;3;1;SELECT;0;
          SELECT VKORG,VTWEG,SPART FROM KNVV WHERE KUNNR= 0000001033'
[...]
2011-12-11_08-45-23;1;[Action];rOLTP;ShowOpenItems;121215;
          121220;5;4;15;SELECT;865;
          '2003-05-01' and '2003-05-31'
[...]
```

Listing B.2 Excerpt from a benchmark run result log

Appendix C
Evaluation Results

The evaluation results of the impact of adding OLAP to OLTP are provided in Table C.1. The throughput is normalized per DBMS and average response time is normalized per DBMS and workload component.

Table C.2 provides an excerpt of the results of the impact of database schema variation regarding response time. The response time is normalized per DBMS. Table C.3 provides an excerpt of the results concerning the throughput. Tables C.4 and C.5 give an overview of response times and throughput respectively for different workload mixes.

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9,
© Springer-Verlag Berlin Heidelberg 2014

Table C.1 Normalized throughput and average response times

		Throughput					Response time				
	# OLAP	1 OLTP	5 OLTP	10 OLTP	50 OLTP	100 OLTP	1 OLTP	5 OLTP	10 OLTP	50 OLTP	100 OLTP
RD1	1	0.28	0.49	0.53	0.60	0.98	0.09	0.12	0.15	0.15	0.17
	5	0.26	0.44	0.47	0.54	1.00	0.09	0.13	0.16	0.16	0.19
	10	0.22	0.32	0.31	0.49	0.97	0.12	0.16	0.19	0.19	0.24
	50	0.10	0.10	0.08	0.41	0.81	0.37	0.37	0.43	0.47	0.54
	100	0.07	0.12	0.10	0.36	0.66	0.77	0.46	0.59	1.00	0.84
CD1	1	0.17	0.44	0.60	0.93	1.00	0.05	0.11	0.16	0.43	0.74
	5	0.17	0.42	0.57	0.95	0.99	0.05	0.11	0.16	0.42	0.74
	10	0.12	0.38	0.53	0.92	0.99	0.07	0.12	0.17	0.43	0.74
	50	0.05	0.18	0.30	0.73	0.90	0.18	0.22	0.26	0.54	0.86
	100	0.03	0.11	0.19	0.55	0.76	0.34	0.38	0.41	0.69	1.00
CM1	1	0.17	0.43	0.59	0.88	0.95	0.05	0.06	0.08	0.21	0.36
	5	0.39	0.55	0.71	0.91	0.96	0.06	0.07	0.09	0.23	0.39
	10	0.53	0.61	0.70	0.95	1.00	0.07	0.09	0.10	0.24	0.40
	50	0.54	0.55	0.57	0.74	0.84	0.46	0.46	0.46	0.54	0.76
	100	0.54	0.54	0.56	0.67	0.76	0.52	0.64	0.69	0.78	1.00

Table C.2 Normalized average response times per database schema

			1NF	3NF	Document	Snowflake	1NF	3NF	Document	Snowflake
RD1	Mixed OLTP	# OLAP	1 OLTP				100 OLTP			
		1	0.0001	0.0001	0.0001	0.0003	0.0001	0.0001	0.0002	0.0018
		5	0.0001	0.0001	0.0001	0.0004	0.0001	0.0001	0.0002	0.0020
		10	0.0001	0.0001	0.0001	0.0004	0.0002	0.0002	0.0003	0.0031
		50	0.0003	0.0003	0.0004	–	0.0003	0.0004	0.0008	0.0039
		100	0.0005	0.0007	0.0008	–	0.0005	0.0006	0.0010	0.0047
	OLAP	# OLTP	1 OLAP				100 OLAP			
		1	–	–	0.0001	0.0004	0.0793	0.8253	0.0016	0.0076
		5	0.0003	0.0509	0.0001	0.0007	0.0728	1.0000	0.0017	0.0087
		10	0.0005	0.0405	0.0002	0.0008	0.0783	0.8647	0.0018	0.0096
		50	0.0005	0.0536	0.0002	0.0017	0.0233	0.8473	0.0027	0.0114
		100	0.0004	–	0.0003	0.0016	0.1157	0.7635	0.0028	0.0132
CD1	Mixed OLTP	# OLAP	1 OLTP				100 OLTP			
		1	0.0002	0.0002	0.0002	0.0004	0.0029	0.0028	0.0039	0.0048
		5	0.0002	0.0002	0.0002	0.0005	0.0029	0.0028	0.0039	0.0048
		10	0.0002	0.0002	0.0003	0.0006	0.0028	0.0028	0.0038	0.0050
		50	0.0006	0.0004	0.0007	0.0019	0.0030	0.0030	0.0035	0.0067
		100	0.0010	0.0004	0.0009	0.0038	0.0034	0.0032	0.0036	0.0129
	OLAP	# OLTP	1 OLAP				100 OLAP			
		1	0.0163	0.0293	0.0032	0.0039	0.2627	0.4987	0.0583	0.0858
		5	0.0218	0.0452	0.0040	0.0048	0.2836	0.5434	0.0640	0.0898
		10	0.0314	0.0453	0.0052	0.0056	0.3209	0.5967	0.0702	0.0906
		50	0.0360	0.0570	0.0081	0.0238	0.5517	0.8873	0.1175	0.1209
		100	0.0461	0.0669	0.0107	0.0567	0.7730	1.0000	0.1749	0.1459

(continued)

Table C.2 (continued)

			1NF	3NF	Document	Snowflake	1NF	3NF	Document	Snowflake
CM1	Mixed OLTP	# OLAP	1 OLTP				100 OLTP			
		1	0.0105	0.0102	0.0121	0.0170	0.0712	0.0697	0.1105	0.1776
		5	0.0124	0.0115	0.0154	0.0216	0.0759	0.0786	0.1365	0.2025
		10	0.0136	0.0129	0.0180	0.0250	0.0775	0.0887	0.1589	0.2295
		50	0.0870	0.0682	0.1625	0.1307	0.1459	0.2314	0.3128	0.3965
		100	0.0980	0.3453	0.2341	0.2431	0.1910	0.3413	0.3389	0.4467
	OLAP	# OLTP	1 OLAP				100 OLAP			
		1	0.0116	0.0426	0.0195	0.0152	0.1776	0.7300	0.3768	0.2446
		5	0.0158	0.0584	0.0368	0.0255	0.1859	0.7405	0.3972	0.2573
		10	0.0224	0.0625	0.0579	0.0472	0.1932	0.7697	0.4186	0.2825
		50	0.0595	0.0999	0.1213	0.1190	0.2538	0.8389	0.5926	0.5421
		100	0.0851	0.1265	0.1567	0.1553	0.3240	0.9577	0.9723	1.0000

Table C.3 Normalized throughput per database schema

	OLTP (0 OLAP clients)					OLAP (0 OLTP clients)				
	# OLTP	1NF	3NF	Document	Snowflake	# OLAP	1NF	3NF	Document	Snowflake
RD1	1	0.1513	0.1515	0.2025	0.0463	1	0.0041	0.0001	0.2025	0.0494
	5	0.2769	0.2746	0.1050	0.1024	5	0.0138	0.0003	0.2007	0.0981
	10	0.2923	0.3294	0.1958	0.0995	10	0.0165	0.0005	0.7289	0.1116
	50	0.3150	0.3298	0.1955	0.1483	50	0.0173	0.0006	0.9343	0.1746
	100	0.5196	0.5184	0.2344	0.1850	100	0.0172	0.0005	1.0000	0.1970
CD1	1	0.1710	0.1701	0.1288	0.0439	1	0.0023	0.0010	0.0154	0.0116
	5	0.4436	0.4432	0.3489	0.1693	5	0.0095	0.0048	0.0545	0.0391
	10	0.5969	0.6031	0.4550	0.2448	10	0.0152	0.0063	0.0803	0.0479
	50	0.9102	0.9567	0.6502	0.3108	50	0.0142	0.0067	0.0806	0.0490
	100	0.9764	1.0000	0.6654	0.3179	100	0.0148	0.0064	0.0816	0.0494
CM1	1	0.1011	0.1018	0.0859	0.0515	1	0.0972	0.0189	0.0523	0.0674
	5	0.4129	0.4127	0.3163	0.1893	5	0.3541	0.0797	0.1774	0.2448
	10	0.6467	0.6468	0.4654	0.2739	10	0.5229	0.1191	0.2391	0.3541
	50	0.9525	0.9574	0.6331	0.3478	50	0.5652	0.1179	0.2540	0.3784
	100	0.9912	1.0000	0.6490	0.3535	100	0.5622	0.1192	0.2537	0.3780

Table C.4 Normalized response time of workload mixes

		OLTP				OLAP			
	# OLAP clients	1NF	3NF	Document	Snowflake	1NF	3NF	Document	Snowflake
RD1	20	0.0003	0.0003	0.0007	0.0033	0.0108	0.4241	0.0035	0.0066
	25	0.0003	0.0004	0.0006	0.0024	0.0328	0.4866	0.0011	0.0080
	40	0.0004	0.0005	0.0007	0.0044	0.0512	0.6521	0.0013	0.0084
	50	0.0005	0.0005	0.0009	0.0050	0.0640	0.6211	0.0016	0.0095
	60	0.0005	0.0007	0.0009	0.0090	0.0399	1.0000	0.0017	0.0097
	75	0.0005	0.0007	0.0009	0.0031	0.0937	0.7587	0.0020	0.0137
	80	0.0005	0.0007	0.0009	0.0044	0.0889	0.8862	0.0021	0.0114
CD1	20	0.0040	0.0039	0.0051	0.0080	0.3627	0.4935	0.0709	0.1012
	25	0.0038	0.0038	0.0048	0.0081	0.5272	0.5452	0.0801	0.1044
	40	0.0034	0.0035	0.0040	0.0080	0.6301	0.7612	0.1061	0.1122
	50	0.0032	0.0031	0.0039	0.0081	0.5428	0.8434	0.0980	0.1163
	60	0.0029	0.0027	0.0030	0.0081	0.5941	0.9518	0.1299	0.1177
	75	0.0026	0.0021	0.0022	0.0076	0.4701	1.0000	0.1299	0.1236
	80	0.0025	0.0019	0.0020	0.0080	0.5393	0.9999	0.1252	0.1208
CM1	20	0.1408	0.1615	0.2635	0.3738	0.1579	0.4029	0.7832	0.8761
	25	0.1581	0.1797	0.2819	0.3958	0.1845	0.4587	0.7540	0.9836
	40	0.1773	0.2895	0.3704	0.4302	0.2336	0.6097	0.6138	0.7194
	50	0.1772	0.2856	0.3692	0.4143	0.2604	0.7325	0.6162	0.6907
	60	0.1729	0.3736	0.3773	0.4019	0.2800	0.8174	0.6194	0.5879
	75	0.1800	0.4328	0.3628	0.3707	0.2971	0.9978	0.6401	0.5177
	80	0.1832	0.4609	0.3637	0.3483	0.3006	1.0000	0.6398	0.5006

Table C.5 Normalized throughput of workload mixes

	# OLAP clients	1NF	3NF	Document	Snowflake
RD1	20	0.1453	1.0000	0.2512	0.2044
	25	0.1700	0.1511	0.5278	0.1602
	40	0.1698	0.1882	0.2902	0.3467
	50	0.0559	0.0042	0.3388	0.1043
	60	0.1008	0.0332	0.1885	0.9443
	75	0.1930	0.2897	0.5282	0.3541
	80	0.8213	0.7844	0.1534	0.3222
CD1	20	1.0000	0.9941	0.7010	0.2776
	25	0.9793	0.9599	0.6895	0.2643
	40	0.8693	0.8514	0.6122	0.2232
	50	0.7526	0.7858	0.5291	0.1911
	60	0.6468	0.7219	0.5454	0.1589
	75	0.4615	0.5847	0.4694	0.1194
	80	0.3784	0.5256	0.4163	0.1090
CM1	20	1.0000	0.7248	0.4592	0.3307
	25	0.9293	0.6216	0.4156	0.3003
	40	0.8064	0.3793	0.3364	0.2832
	50	0.7579	0.3403	0.3281	0.2859
	60	0.7149	0.2648	0.3127	0.3037
	75	0.6354	0.1942	0.2908	0.3234
	80	0.6109	0.1853	0.2838	0.3324

Bibliography

1. D.J. Abadi, Column stores for wide and sparse data, in *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research*, Asilomar, 2007, pp. 292–297. Online Proceedings. Retrieved from http://www.cidrdb.org. Last accessed 15 June 2012
2. D.J. Abadi, S.R. Madden, N. Hachem. Column-stores vs. row-stores: how different are they really? in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, Vancouver (ACM, New York, 2008), pp. 967–980
3. S. Agrawal, E. Chu, V. Narasayya, Automatic physical design tuning: workload as a sequence, in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06*, Chicago (ACM, New York, 2006), pp. 683–694
4. AMD, HyperTransport technology I/O link, a high- bandwidth I/O architecture. White paper, Advanced Micro Devices, Inc., #25012, July 2001
5. T.L. Anderson, The hypermodel benchmark, in *Proceedings of the 2nd International Conference on Extending Database Technology: Advances in Database Technology, EDBT '90*, Venice (Springer, New York, 1990), pp. 317–331
6. M.M. Astrahan, M.W. Blasgen, D.D. Chamberlin, K.P. Eswaran, J.N. Gray, P.P. Griffiths, W.F. King, R.A. Lorie, P.R. McJones, J.W. Mehl, G.R. Putzolu, I.L. Traiger, B.W. Wade, V. Watson, System R: relational approach to database management. ACM Trans. Database Syst. **1**(2), 97–137 (1976). ACM, New York
7. C.W. Bachman, Data structure diagrams. ACM SIGMIS Database **1**(2), 4–10 (1969). ACM, New York
8. D. Bausch, I. Petrov, A. Buchmann, On the performance of database query processing algorithms on flash solid state disks, in *Proceedings of the 2011 22nd International Workshop on Database and Expert Systems Applications, DEXA '11*, Toulouse (IEEE Computer Society, Washington D.C., 2011), pp. 139–144
9. P.A. Bernstein, E. Newcomer, *Principles of Transaction Processing*, 2nd edn. (Morgan Kaufmann, Burlington, 2009)
10. D. Bitton, C. Turbyfill, Design and analysis of multi-user benchmarks for database systems. Technical report, Cornell University, Ithaca, 1984
11. D. Bitton, D.J. DeWitt, C. Turbyfill, Benchmarking database systems a systematic approach, in *Proceedings of the 9th International Conference on Very Large Data Bases*, Florence (Morgan Kaufmann, San Francisco, 1983), pp. 8–19
12. R. Bloor, The failure of relational database, the rise of object technology and the need for the hybrid database, Baroudi Bloor International Inc. (2004). Retrieved from http://www.intersystems.com/cache/whitepapers/pdf/baroudi_bloor.pdf. Last accessed 15 June 2012
13. D.B. Bock, J.F. Schrage, Denormalization guidelines for base and transaction tables. ACM SIGCSE Bull. **34**(4), 129–133 (2002). ACM, New York

A. Bog, *Benchmarking Transaction and Analytical Processing Systems*,
In-Memory Data Management Research, DOI 10.1007/978-3-642-38070-9,

© Springer-Verlag Berlin Heidelberg 2014

14. A. Bog, J. Schaffner, J. Krüger, A composite benchmark for online transaction processing and operational reporting, in *IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE'08)*, Tianjin, 2008, pp. 1–5
15. A. Bog, M. Domschke, J. Müller, A. Zeier, A framework for simulating combined OLTP and OLAP workloads, in *16th International Conference on Industrial Engineering and Engineering Management, IE&EM '09*, Beijing, 2009, pp. 1675–1678
16. A. Bog, H. Plattner, A. Zeier, A mixed transaction processing and operational reporting benchmark. Inf. Syst. Front. **13**(3), 321–335 (2011). Kluwer Academic, Hingham
17. A. Bog, K. Sachs, A. Zeier, Benchmarking database design for mixed OLTP and OLAP workloads, in *Proceedings of the Second Joint WOSP/SIPEW International Conference on Performance Engineering, ICPE '11*, Karlsruhe (ACM, New York, 2011), pp. 417–418
18. A. Bog, K. Sachs, A. Zeier, H. Plattner, Normalization in a mixed OLTP and OLAP workload scenario, in *Third TPC Technology Conference on Performance Evaluation & Benchmarking (TPCTC)*, Seattle, 2011
19. A. Bog, K. Sachs, H. Plattner, Interactive performance monitoring of a composite OLTP and OLAP workload, in *Proceedings of the 2012 International Conference on Management of Data, SIGMOD '12*, Scottsdale (ACM, New York, 2012), pp. 645–648
20. M. Böhm, D. Habich, W. Lehner, U. Wloka, Dipbench toolsuite: a framework for benchmarking integration systems, in *ICDE*, Cancun, ed. by G. Alonso, J.A. Blakeley, A.L.P. Chen (IEEE, 2008), pp. 1596–1599
21. M. Breslin, Data warehousing battle of the giants: comparing the basics of the Kimball and Inmon models. Bus. Intell. J. **9**(1), 6–20 (2004). Winter
22. S.K. Card, G.G. Robertson, J.D. Mackinlay, The information visualizer, an information workspace, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology, CHI '91*, New Orleans (ACM, New York, 1991), pp. 181–186
23. M.J. Carey, D.J. DeWitt, J.F. Naughton, The 007 benchmark. ACM SIGMOD Rec. **22**(2), 12–21 (1993). ACM, New York
24. R. Cattell, Scalable SQL and NoSQL data stores. ACM SIGMOD Rec. **39**(4), 12–27 (2011). ACM, New York
25. R.G.G. Cattell, J. Skeen, Object operations benchmark. ACM Trans. Database Syst. **17**(1), 1–31 (1992). ACM, New York
26. E. Cecchet, G. Candea, A. Ailamaki, Middleware-based database replication: the gaps between theory and practice, in *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, Vancouver (ACM, New York, 2008), pp. 739–752
27. D.D. Chamberlin, R.F. Boyce, SEQUEL: a structured English query language, in *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control, SIGFIDET '74*, Ann Arbor (ACM, New York, 1974), pp. 249–264
28. D.D. Chamberlin, A.M. Gilbert, R.A. Yost, A history of system R and SQL/data system, in *Proceedings of the Seventh International Conference on Very Large Data Bases*, Cannes. Volume 7 of VLDB '1981 (VLDB Endowment, 1981), pp. 456–464
29. S. Chaudhuri, U. Dayal, Data warehousing and OLAP for decision support, in *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD '97*, Tucson (ACM, New York, 1997), pp. 507–508
30. S. Chaudhuri, U. Dayal, An overview of data warehousing and OLAP technology. ACM SIGMOD Rec. **26**(1), 65–74 (1997). ACM, New York
31. S. Chaudhuri, V.R. Narasayya, An efficient cost-driven index selection tool for Microsoft SQL server, in *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, Athens (Morgan Kaufmann, San Francisco, 1997), pp. 146–155
32. S. Chaudhuri, U. Dayal, V. Ganti, Database technology for decision support systems. Computer **34**(12), 48–55 (2001). IEEE Computer Society, Los Alamitos
33. P.P.-S. Chen, The entity-relationship model – toward a unified view of data. ACM Trans. Database Syst. **1**(1), 9–36 (1976). ACM, New York

34. P. Chen, Entity-relationship modeling: historical events, future trends, and lessons learned, in *Software Pioneers: Contributions to Software Engineering*, ed. by M. Broy, E. Denert (Springer, New York, 2002), pp. 296–310
35. A. Chmura, J.M. Heumann, *Logical Data Modeling: What It Is and How to Do It* (Springer, New York, 2005)
36. S. Choenni, H.M. Blanken, T. Chang, On the automation of physical database design, in *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice, SAC '93*, Indianapolis (ACM, New York, 1993), pp. 358–367
37. CODASYL, Data base task group. Report, Apr 1971
38. E.F. Codd, A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970). ACM, New York
39. 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*, San Diego (ACM, New York, 1971), pp. 1–17
40. E.F. Codd, Data models in database management, in *Proceedings of the 1980 Workshop on Data Abstraction, Databases and Conceptual Modeling*, Pingree Park (ACM, New York, 1980), pp. 112–114
41. E.F. Codd, The significance of the SQL/data system announcement. Computerworld **15**(7), 27–30 (1981). ACM, New York
42. E.F. Codd, Relational database: a practical foundation for productivity. Commun. ACM **25**(2), 109–117 (1982). ACM, New York
43. E.F. Codd, S.B. Codd, C.T. Salley, Providing OLAP (on-line analytical processing) to user-analysis: an IT mandate. Arbor Software, Hyperion Solutions Corp., White paper, 1993
44. Coglin Mill, RODIN data asset management high performance extract/transform/load benchmark. Report, June 2002. Retrieved from http://www.coglinmill.com/pdf_files/RODIN%20i890%20Benchmark%20June%202002.pdf. Last accessed 15 June 2012
45. R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. Kuno, R. Nambiar, T. Neumann, M. Poess, K.-U. Sattler, M. Seibold, E. Simon, F. Waas, The mixed workload CH-BenCHmark, in *Proceedings of the Fourth International Workshop on Testing Database Systems, DBTest '11*, Athens (ACM, New York, 2011), pp. 8:1–8:6
46. T.M. Connolly, C.E. Begg, *Database Systems: A Practical Approach to Design, Implementation, and Management* (Pearson, Harlow, 2005)
47. C. Coronel, P. Rob, *Database Systems: Design, Implementation, and Management* (Cengage Learning, Boston 2009)
48. R.L. Cross, ITJ foreword Q1, 2002. Intel Technol. J. Hyper-Threading Technol. **6**(1), 3 (2002)
49. J. Darmont, M. Schneider, Object-oriented database benchmarks, in *Advanced Topics in Database Research*, vol. 1, ed. by K. Siau (IGI, Hershey, 2002), pp. 34–57
50. C.J. Date, A critique of the SQL database language. ACM SIGMOD Rec. **14**(3), 8–54 (1984). ACM, New York
51. C.J. Date, *Database in Depth: Relational Theory for Practitioners* (O'Reilly, Sebastopol, 2005)
52. D.J. DeWitt, The Wisconsin benchmark: past, present, and future, in *Database and Transaction Processing System Performance Handbook*, ed. by J. Gray (Morgan-Kaufman, San Mateo, 1991)
53. S.W. Dietrich, S.D. Urban, *Fundamentals of Object Databases: Object-Oriented and Object-Relational Design* (Morgan & Claypool, San Rafael, 2011)
54. B. Dinter, C. Sapia, G. Höfling, M. Blaschka, The OLAP market: state of the art and research issues, in *Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP, DOLAP '98*, Bethesda (ACM, New York, 1998), pp. 22–27
55. J. Dittrich, 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)*, Asilomar, 2011. Online Proceedings. Retrieved from http://www.cidrdb.org/cidr2011/program.html. Last accessed 15 June 2012

56. D. Dominguez-Sal, N. Martinez-Bazan, V. Muntes-Mulero, P. Baleta, J.L. Larriba-Pay, A discussion on the design of graph database benchmarks, in *Proceedings of the Second TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems, TPCTC'10*, Singapore (Springer, Berlin/Heidelberg, 2011), pp. 25–40
57. G. Eadon, E.I. Chong, S. Shankar, A. Raghavan, J. Srinivasan, S. Das, Supporting table partitioning by reference in Oracle, in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, Vancouver (ACM, New York, 2008), pp. 1111–1122
58. S. Elnaffar, P. Martin, B. Schiefer, S. Lightstone, Is it DSS or OLTP: automatically identifying DBMS workloads. J. Intell. Inf. Syst. **30**(3), 249–271 (2008). doi:10.1007/s10844-006-0036-6
59. D. Feinberg, M.A. Beyer, Magic quadrant for data warehouse database management systems. Gartner RAS core research note G00209623, Jan 2011
60. D. Fisher, Incremental, approximate database queries and uncertainty for exploratory visualization, in *Proceedings of the 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, Providence, 2011, pp. 73–80
61. C.D. French, "One size fits all" database architectures do not work for DSS, in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, SIGMOD '95*, San Jose (ACM, New York, 1995), pp. 449–450
62. C.D. French, Teaching an OLTP database Kernel advanced data warehousing techniques, in *Proceedings of the Thirteenth International Conference on Data Engineering, ICDE '97*, Washington, D.C. (IEEE Computer Society, 1997), pp. 194–198
63. M. Frolick, T.R. Ariyachandra, Business performance management: one truth. Inf. Syst. Manag. **23**(1), 41–48 (2006). Winter
64. F. Funke, A. Kemper, S. Krompass, H. Kuno, T. Neumann, A. Nica, M. Poess, M. Seibold, Metrics for measuring the performance of the mixed workload CH-BenCHmark, in *Proceedings of the 3rd TPC Technology Conference on Performance Evaluation and Benchmarking (TPC TC)*, Seattle, 2011
65. F. Funke, A. Kemper, T. Neumann, HyPer-sonic combined transaction AND query processing. PVLDB **4**(12), 1367–1370 (2011)
66. F. Funke, A. Kemper, T. Neumann, Benchmarking hybrid OLTP & OLAP database systems, in *GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, Kaiserslautern, ed. by T. Härder, W. Lehner, B. Mitschang, H. Schöning, H. Schwarz. Volume 180 of LNI, (GI, 2011), pp. 390–409
67. J. Garcia, Role of in-memory analytics in big data analysis. Technology Evaluation Centers (TEC) Article, Mar 2012
68. H. Garcia-Molina, K. Salem, Main memory database systems: an overview. IEEE Trans. Knowl. Data Eng. **4**(6), 509–516 (1992). IEEE Educational Activities Department, Piscataway
69. H. Garcia-Molina, J.D. Ullman, J. Widom, *Database System Implementation* (Prentice Hall, Upper Saddle River, 1999)
70. Gartner, Gartner identifies the top 10 strategic technologies for 2012. Press Release, Oct 2011. Retrieved from http://www.gartner.com/it/page.jsp?id=1826214. Last accessed 15 June 2012
71. S. Graves, In-memory database systems. Linux J. **2002**(101), 10 (2002). Belltown Media, Houston
72. J. Gray, A transaction model, in *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, Noordweijkerhout (Springer, London, 1980), pp. 282–298
73. J. Gray, The transaction concept: virtues and limitations (invited paper), in *Proceedings of the 7th International Conference on Very Large Data Bases*, Cannes (IEEE Computer Society, 1981), pp. 144–154
74. J. Gray (ed.), *The Benchmark Handbook for Database and Transaction Systems*, 2nd edn. (Morgan Kaufmann, San Mateo, 1993)
75. J. Gray, P. Shenoy, Rules of thumb in data engineering, in *Proceedings of the 16th International Conference on Data Engineering*, San Diego (IEEE Computer Society, Washington, D.C., 2000), pp. 3–12

76. J. Gray, R. Lorie, G.F. Putzolu, I.L. Traiger, Granularity of locks and degrees of consistency in a shared data base, in *Modeling in Data Base Management Systems*, ed. by G.M. Nijssen (North Holland, Amsterdam/New York, 1976), pp. 364–394
77. R. Greenwald, R. Stackowiak, J. Stern, *Oracle Essentials: Oracle9i, Oracle8i & Oracle8*, 2nd edn. (O'Reilly Media, Sebastopol, 2001), pp. 1–29
78. M. Grund, J. Krüger, J. Schaffner, M.-P. Schapranow, A. Bog, Operational reporting using navigational SQL, in *IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE)*, Tianjin, 2008
79. M. Grund, J. Schaffner, M.-P. Schapranow, J. Krüger, A. Bog, Shared table access pattern analysis for multi-tenant applications, in *IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE'08)*, Tianjin, 2008
80. M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, S. Madden, HYRISE: a main memory hybrid storage engine. Proc. VLDB Endow. **4**(2), 105–116 (2010)
81. M. Grund, P. Cudré-Mauroux, J. Krüger, S. Madden, H. Plattner, An overview of HYRISE – a main memory hybrid storage engine. IEEE Data Eng. Bull. 35(1), 52–57 (2012)
82. A. Gupta, I.S. Mumick, Maintenance of materialized views: problems, techniques, and applications. IEEE Data Eng. Bull. 18(2), 3–18 (1995)
83. D. Hackenberg, D. Molka, W.E. Nagel, Comparing cache architectures and coherency protocols on x86-64 multicore SMP systems, in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 42*, New York (ACM, New York, 2009), pp. 413–422
84. D.J. Haderle, Database role in information systems: the evolution of database technology and its impact on enterprise information systems, in *IBM Symposium: Database Systems of the 90s*, ed. by A. Blaser. Volume 466 of Lecture Notes in Computer Science (Springer, New York, 1990), pp. 1–14
85. D.J. Haderle, R.D. Jackson, IBM database 2 overview. IBM Syst. J. **23**(2), 112–125 (1984)
86. T. Haerder, A. Reuter, Principles of transaction-oriented database recovery. ACM Comput. Surv. **15**(4), 287–317 (1983). ACM, New York
87. V. Harinarayan, A. Rajaraman, J.D. Ullman, Implementing data cubes efficiently. ACM SIGMOD Rec. **25**(2), 205–216 (1996). ACM, New York
88. S. Harizopoulos, V. Shkapenyuk, A. Ailamaki, QPipe: a simultaneously pipelined relational query engine, in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD '05*, Baltimore (ACM, New York, 2005), pp. 383–394
89. S. Harizopoulos, V. Liang, D.J. Abadi, S. Madden, Performance tradeoffs in read-optimized databases, in *Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06*, Seoul (VLDB Endowment, 2006), pp. 487–498
90. S. Harizopoulos, D.J. Abadi, S. Madden, M. Stonebraker, OLTP through the looking glass, and what we found there, in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, New York (ACM, 2008), pp. 981–992
91. P. Helland, If you have too much data, then "good enough" is good enough. Commun. ACM, **54**(6), 40–47 (2011). ACM, New York
92. M.J. Hernandez, *Database Design for Mere Mortals: A Hands-on Guide to Relational Database Design*, 2nd edn. (Addison-Wesley, 2003)
93. M.J. Hernandez, J.L. Viescas, *Go to SQL* (Pearson Education, München, 2001)
94. T. Hogan, Overview of TPC benchmark E: the next generation of OLTP benchmarks, in *Performance Evaluation and Benchmarking*, ed. by R. Nambiar, M. Poess. Volume 5895 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2009), pp. 84–98
95. W.W. Hsu, A.J. Smith, H.C. Young, Characteristics of production database workloads and the TPC benchmarks. IBM Syst. J. **40**(3), 781–802 (2001)
96. K. Huppler, The art of building a good benchmark, in *Performance Evaluation and Benchmarking*, ed. by R. Nambiar, M. Poess. Volume 5895 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2009), pp. 18–30
97. IBM, Database administration, IMS version 11. Technical document, number SC19-2434-01, 2010

98. IBM, IBM solidDB. Product Website, n.d. Retrieved from http://www-01.ibm.com/software/data/soliddb/soliddb/. Last accessed 15 June 2012
99. IBM Software Group Information Management, Telecommunication application transaction processing (TATP) benchmark description. Version 1.0, Mar 2009. Retrieved from http://tatpbenchmark.sourceforge.net. Last accessed 15 June 2012
100. W.H. Inmon, The operational data store. InfoDB 9, 21–24 (1995)
101. W.H. Inmon, The data warehouse and data mining. Commun. ACM **39**(11), 49–50 (1996). ACM, New York
102. W.H. Inmon, Data mart does not equal data warehouse. Data Manag. Rev. (1998). http://www.information-management.com/issues/19980701/469-1.html
103. W.H. Inmon, The operational data store. Designing the operational data store. Inf. Manag. Mag. (1998)
104. W.H. Inmon, Operational and informational reporting, information management: charting the course. Inf. Manag. Mag. (2000). http://www.information-management.com/issues/20000701/2349-1.html
105. W.H. Inmon, ODS types, information management: charting the course. Inf. Manag. Mag. (2000). http://www.information-management.com/issues/20000101/1749-1.html
106. W.H. Inmon, ODS: on the road to maturity, information management: charting the course. Inf. Manag. Magaz. (2000). http://www.information-management.com/issues/20000201/1855-1.html
107. B. Inmon, Budgeting for the data warehouse. BeyeNetwork Article, June 2004. Retrieved from http://www.b-eye-network.com/view/141. Last accessed 15 June 2012
108. W.H. Inmon, *Building the Data Warehouse*, 4th edn. (Wiley, Indianapolis, 2005)
109. B. Inmon, A tale of two architectures. White paper, 2010. Retrieved from http://www.inmoncif.com/products/A%20TALE%20OF%20TWO%20ARCHITECTURES.pdf. Last accessed 18 June 2012
110. Intel, An introduction to the Intel QuickPath Interconnect. White paper, Jan 2009. Document Number: 320412-001US
111. R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling: Techniques for Experimental Design, Measurement, Simulation and Modelling* (Wiley, New York, 1991)
112. P.K. Janert, *Gnuplot in Action: Understanding Data with Graphs* (Manning Publications, Greenwich, 2009)
113. R. Johnson, S. Harizopoulos, N. Hardavellas, K. Sabirli, I. Pandis, A. Ailamaki, N.G. Mancheril, B. Falsafi, To share or not to share? in *Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB '07*, Vienna (VLDB Endowment, 2007), pp. 351–362
114. N. Jukic, Modeling strategies and alternatives for data warehousing projects. Commun. ACM **49**(4), 83–88 (2006). ACM, New York
115. O. Kaser, D. Lemire, Attribute value reordering for efficient hybrid OLAP, in *Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP, DOLAP '03*, New Orleans (ACM, New York, 2003), pp. 1–8
116. A. Kemper, T. Neumann, HyPer: a hybrid OLTP & OLAP main memory database system based on virtual memory snapshots, in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE '11*, Hannover (IEEE Computer Society, Washington, D.C., 2011), pp. 195–206
117. A. Kemper, T. Neumann, One size fits all, again! The architecture of the hybrid OLTP & OLAP database management system HyPer, in *Enabling Real-Time Business Intelligence*, ed. by M. Castellanos, U. Dayal, V. Markl, W. Aalst, J. Mylopoulos, M. Rosemann, M.J. Shaw, C. Szyperski. Volume 84 of Lecture Notes in Business Information Processing (Springer, Berlin/Heidelberg, 2011), pp. 7–23
118. W. Kent, A simple guide to five normal forms in relational database theory. Commun. ACM **26**(2), 120–125 (1983). ACM, New York

119. R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* (Wiley, New York, 2002)
120. R. Kimball, M. Ross, *The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence* (Wiley, Indianapolis, 2010)
121. B. Klein, IMS 10 for on demand business. IBM Systems Magazine, Jan 2008. Retrieved from http://www.ibmsystemsmag.com/mainframe/administrator/ims/IMS_10_for_on_demand_business/. Last accessed 15 June 2012
122. G. Koch, Discovering multi-core: extending the benefits of Moores law. Technology@Intel Magazine. Intel Corporation, Tech. Rep. (2005)
123. Y. Kotidis, N. Roussopoulos, An alternative storage organization for ROLAP aggregate views based on cubetrees, in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, SIGMOD '98*, Seattle (ACM, New York, 1998), pp. 249–258
124. J. Krueger, M. Grund, A. Zeier, H. Plattner, Enterprise application-specific data management, in *Proceedings of the 2010 14th IEEE International Enterprise Distributed Object Computing Conference, EDOC '10*, Vitoria (IEEE Computer Society, Washington, D.C., 2010), pp. 131–140
125. J. Krüger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, H. Plattner, P. Dubey, A. Zeier, Fast updates on read-optimized databases using multi-core CPUs. Proc. VLDB Endow. **5**(1), 61–72 (2011) VLDB Endowment.
126. C.A. Lang, B. Bhattacharjee, T. Malkemus, S. Padmanabhan, K. Wong, Increasing buffer-locality for multiple relational table scans through grouping and throttling, in *Proceedings of the 23rd International Conference on Data Engineering, ICDE'07*, Istanbul, ed. by R. Chirkova, A. Dogac, M.T. Özsu, T.K. Sellis (IEEE, 2007), pp. 1136–1145
127. R.M. Lerner, At the forge: NoSQL? I'd prefer SomeSQL. Linux J. **2010**(192) (2010). Article No. 5, Belltown Media, Houston
128. S.S. Lightstone, B. Bhattacharjee. Automated design of multidimensional clustering tables for relational databases, in *Proceedings of the 30th International Conference on Very Large Data Bases*, Toronto, ed. by M.A. Nascimento, M.T. Özsu, D. Kossmann, R.J. Miller, J.A. Blakeley, K. Bernhard Schiefer. Volume 30 of VLDB '04 (VLDB Endowment, 2004), pp. 1170–1181
129. D.J. Lilja, *Measuring Computer Performance: A Practitioner's Guide* (Cambridge University Press, Cambridge/New York, 2000)
130. B.G. Lindsay, Jim Gray at IBM: the transaction processing revolution. ACM SIGMOD Rec. **37**(2), 38–40 (2008). ACM, New York
131. L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*. (Springer, New York/London, 2009)
132. R. Long, M. Harrington, R. Hain, G. Nicholls, IMS primer. IBM, International Technical Support Organization, Jan 2000
133. P. Loos, J. Lechtenbörger, G. Vossen, A. Zeier, J. Krüger, J. Müller, W. Lehner, D. Kossmann, B. Fabian, O. Günther, R. Winter, In-memory databases in business information systems. Bus. Inf. Syst. Eng. **3**(6), 389–395 (2011)
134. R.R. Lummus, R.J. Vokurka, Defining supply chain management: a historical perspective and practical guidelines. Ind. Manag. Data Syst. **99**(1), 11–17 (1999). MCB University Press
135. C. Madeiros, F. Tompa, Understanding the implications of view update policies, in *Proceedings of the International Conference of Very Large Databases*, Stockholm, vol. 11 (VLDB Endowment, 1985), pp. 316–323
136. D.T. Marr, F. Binns, D.L. Hill, G. Hinton, D.A. Koufaty, J.A. Miller, M. Upton, Hyper-threading technology architecture and microarchitecture. Intel Technol. J. Hyper-Threading Technol. 06(1), 4–15 (2002)
137. T. Martyn, Reconsidering multi-dimensional schemas. ACM SIGMOD Rec. **33**(1), 83–88 (2004). ACM, New York
138. W.C. McGee, The information management system (IMS) program product. IEEE Ann. Hist. Comput. **31**, 66–75 (2009). IEEE Computer Society
139. J. McKendrick, A new dimension to data warehousing: 2011 IOUG data warehousing survey. Produced by Unisphere Research, a Division of Information Today, Inc, Sept 2011. Research Analyst Report.

140. McObject, eXtremeDB real-time embedded database. Product Website, 2012. Retrieved from http://www.mcobject.com/embedded_database_products. Last accessed 15 June 2012
141. Merriam-Webster, Benchmark. In Merriam-Webster.com, 2011. Retrieved from http://www.merriam-webster.com/dictionary/benchmark. Last accessed 15 June 2012
142. P. Mertens, *Integrierte Informationsverarbeitung 1*, 17th edn. (Gabler, Wiesbaden, 2009)
143. A.S. Michaels, B. Mittman, C. Robert Carlson, A comparison of the relational and CODASYL approaches to data-base management. ACM Comput. Surv. **8**, 125–151 (1976). ACM, New York
144. E. Mills, G. Shamshoian, M. Blazek, P. Naughton, R. Seese, W. Tschudi, D. Sartor, The business case for energy management in high-tech industries. Energy Effic. **1**, 5–20 (2008)
145. G.E. Moore, Cramming more components onto integrated circuits. Electronics **38**(8), 114–117 (1965)
146. G. Mullins, *Database Administration: The Complete Guide to Practices and Procedures* (Addison-Wesley, Boston, 2002)
147. *MySQL 5.5 Reference Manual, Revision: 26779*. Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065, July 2011
148. R.O. Nambiar, M. Poess, The making of TPC-DS, in *Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06*, ed. by U. Dayal, K.-Y. Whang, D.B. Lomet, G. Alonso, G.M. Lohman, M.L. Kersten, S.K. Cha, Y.-K. Kim (VLDB Endowment, Seoul, 2006), pp. 1049–1058
149. R. Nambiar, M. Poess, Transaction performance vs. Moores law: a trend analysis, in *Performance Evaluation, Measurement and Characterization of Complex Systems*, ed. by R. Nambiar, M. Poess. Volume 6417 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2011), pp. 110–120
150. R. Nambiar, N. Wakou, F. Carman, M. Majdalany, Transaction processing performance council (TPC): state of the council 2010, in *Performance Evaluation, Measurement and Characterization of Complex Systems*, ed. by R. Nambiar, M. Poess. Volume 6417 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2011), pp. 1–9
151. R. Narang, *Database Management Systems* (Prentice-Hall, New Delhi 2006)
152. M.L. Neufeld, M. Cornog, Database history: from dinosaurs to compact discs. J. Am. Soc. Inf. Sci. **37**(4), 183–190 (1986). Wiley, New York
153. J. Nielsen, *Usability Engineering* (Academic, London, 1993)
154. Object Management Group (OMG), Business process model and notation (BPMN). Specification, version 2.0, Jan 2011. Retrieved from http://www.omg.org/spec/BPMN/2.0/PDF. Last accessed 15 June 2012
155. C.W. Olofson, A platform for enterprise data services: the proven power and flexibility of IMS from IBM. White paper, IDC, sponsored by IBM, Oct 2009
156. P.E. O'Neil, E.J. O'Neil, X. Chen, The star schema benchmark (SSB), Jan 2007. Retrieved from http://www.cs.umb.edu/~poneil/StarSchemaB.pdf. Last accessed 15 June 2012
157. Oracle, Java SE technologies – database. Technology Webpage, n.d.. Retrieved from http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-136101.html. Last accessed 15 June 2012
158. Oracle, Oracle technology network – Java. Technology Webpage, n.d.. Retrieved from http://www.oracle.com/technetwork/java/index.html. Last accessed 20 June 2012
159. Oracle, Oracle TimesTen in-memory database. Product Website, n.d.. Retrieved from http://www.oracle.com/us/products/database/timesten-066524.html. Last accessed 15 June 2012
160. Oracle, Oracle applications benchmark. Benchmark Website, n.d.. Retrieved from http://www.oracle.com/us/solutions/benchmark/apps-benchmark/index-166919.html. Last accessed 15 June 2012
161. S. Padmanabhan, B. Bhattacharjee, T. Malkemus, L. Cranston, M. Huras, Multi-dimensional clustering: a new data layout scheme in DB2, in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD '03*, San Diego (ACM, New York, 2003), pp. 637–641

162. E. Papadomanolakis, A. Ailamaki, Autopart: automating schema design for large scientific databases using data partitioning, in *Proceedings of the 16th International Conference on Scientific and Statistical Database Management, SSDBM '04*, Santorini Island (IEEE Computer Society, Washington, D.C., 2004), pp. 383–392
163. R.L. Patrick, IMS @ conception. IEEE Ann. Hist. Comput. **31**, 62–65 (2009)
164. I. Petrov, G. Almeida, A. Buchmann, U. Graef, Building large storage based on flash disks, in *Proceeding of the First International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures, ADMS 2010*, Singapore, 2010
165. I. Petrov, R. Gottstein, T. Ivanov, D. Bausch, A. Buchmann, Page size selection for OLTP databases on SSD RAID storage. J. Inf. Data Manag. **2**(1), 11 (2011)
166. D. Pfaff, B. Skiera, T. Weitzel, Financial-Chain-Management: Ein generisches Modell zur Identifikation von Verbesserungspotenzialen. Wirtschaftsinformatik **46**(2), 107–117 (2004)
167. H. Plattner, SanssouciDB: an in-memory database for processing enterprise workloads, in *14. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, ed. by T. Härder, W. Lehner, B. Mitschang, H. Schöning, H. Schwarz. Volume 180 of LNI, Kaiserslautern (GI, 2011), pp. 2–21
168. H. Plattner, A. Bog, J. Schaffner, J. Krüger, A. Zeier, ETL-less zero-redundancy system and method for reporting OLTP data. Publication number: US 2009/0240663 A1, 2009
169. M. Poess, C. Floyd, New TPC benchmarks for decision support and web commerce. ACM SIGMOD Rec. **29**(4), 64–71 (2000). ACM, New York
170. M. Poess, B. Smith, L. Kollar, P. Larson, TPC-DS, taking decision support benchmarking to the next level, in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, SIGMOD '02*, Madison (ACM, New York, 2002), pp. 582–587
171. P. Ponniah, *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*, vol. 1 (Wiley, New York, 2001)
172. G. Powell, *Beginning Database Design* (Wrox, New York 2005)
173. Python, Python programming language. Official Website, 2012. Retrieved from http://www.python.org/. Last accessed 20 June 2012
174. N. Raden, Business intelligence 2.0: simpler, more accessible, inevitable. InformationWeek. Intelligent Enterprise 1 (2007)
175. J.F. Rockart, The changing role of the information systems executive: a critical success factors perspective. Sloan Manag. Rev. **24**(1), 3–13 (1982). Fall
176. U. Röhm, OLAP with a database cluster, in *Database Technologies: Concepts, Methodologies, Tools, and Applications*, ed. by J. Erickson (IGI Global, Hershey, 2009), pp. 829–46
177. K. Sachs, S. Kounev, J. Bacon, A. Buchmann, Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark. Perform. Eval. **66**(8), 410–434 (2009)
178. SAP, SAP standard application benchmarks, n.d.. Retrieved from http://www.sap.com/solutions/benchmark/index.epx. Last accessed 15 June 2012
179. SAP, SAP in-memory computing. Product Website, n.d.. Retrieved from http://www.sap.com/solutions/technology/in-memory-computing-platform/hana/overview/index.epx. Last accessed 15 June 2012
180. J. Schaffner, A. Bog, J. Krüger, A. Zeier, A hybrid row-column OLTP database architecture for operational reporting, in *Informal Proceedings of the Second International Workshop on Business Intelligence for the Real-Time Enterprise, BIRTE 2008*, Auckland, 2008
181. J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, A. Zeier, Predicting in-memory database performance for automating cluster management tasks, in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE '11*, Hanover (IEEE Computer Society, Washington, D.C., 2011), pp. 1264–1275
182. M.-P. Schapranow, M. Grund, J. Krüger, J. Schaffner, A. Bog, Combining advantages – unified data stores in global enterprises, in *IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE'08)*, Tianjin, 2008
183. T.K. Sellis, Multiple-query optimization. ACM Trans. Database Syst. **13**(1), 23–52 (1988). ACM, New York

184. A. Sen, A.P. Sinha, A comparison of data warehousing methodologies. Commun. ACM **48**(3), 79–84 (2005). ACM, New York
185. M.G. Shields, *E-business and ERP: rapid implementation and project planning* (Wiley, New York, 2001)
186. A. Shoshani, Statistical databases: characteristics, problems, and some solutions, in *Proceedings of the 8th International Conference on Very Large Data Bases, VLDB '82*, Mexico (Morgan Kaufmann, San Francisco, 1982), pp. 208–222
187. A. Shoshani, F. Olken, H.K.T. Wong, Characteristics of scientific databases, in *Proceedings of the 10th International Conference on Very Large Data Bases, VLDB '84*, Singapore (Morgan Kaufmann, San Francisco, 1984), pp. 147–160
188. A. Shukla, P. Deshpande, J.F. Naughton, K. Ramasamy, Storage estimation for multidimensional aggregates in the presence of hierarchies, in *Proceedings of the 22th International Conference on Very Large Data Bases, VLDB '96*, Bombay (Morgan Kaufmann, San Francisco, 1996), pp. 522–531
189. A. Shukla, P. Deshpande, J.F. Naughton, Materialized view selection for multidimensional datasets, in *Proceedings of the 24rd International Conference on Very Large Data Bases, VLDB '98*, New York (Morgan Kaufmann, San Francisco, 1998), pp. 488–499
190. V. Sikka, F. Färber, W. Lehner, S.K. Cha, T. Peh, C. Bornhövd, Efficient transaction processing in SAP HANA database: the end of a column store myth, in *Proceedings of the 2012 International Conference on Management of Data, SIGMOD '12*, Scottsdale (ACM, New York, 2012), pp. 731–742
191. A. Silberschatz, H.F. Korth, S. Sudarshan, *Database System Concepts*, 6th edn. (Mc Graw Hill, New York, 2010)
192. J. Srinivasan, S. Das, C. Freiwald, E.I. Chong, M. Jagannath, A. Yalamanchi, R. Krishnan, A.-T. Tran, S. DeFazio, J. Banerjee, Oracle8i index-organized table and its application to new domains, in *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB '00*,Cairo (Morgan Kaufmann, San Francisco, 2000), pp. 285–296
193. Standard Performance Evaluation Corporation, SPEC – power and performance, user guide, SPECpower_ssj2008 V1.11, Sept 2011. Retrieved from http://www.spec.org/power/docs/SPECpower_ssj2008-User_Guide.pdf. Last accessed 15 June 2012
194. Standard Performance Evaluation Corporation (SPEC), Server efficiency rating tool (SERT) design document beta-1, Sept 2011. Retrieved from http://www.spec.org/sert/. Last accessed 15 June 2012
195. Standard Performance Evaluation Corporation, Corporation website, 2012. Retrieved from http://www.spec.org/. Last accessed 15 June 2012
196. G. Stewart, Supply chain performance benchmarking study reveals keys to supply chain excellence. Logist. Inf. Manag. **8**(2), 38–44 (1995)
197. M. Stonebraker, A new direction for TPC?, in *Performance Evaluation and Benchmarking*, ed. by R. Nambiar, M. Poess. Volume 5895 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2009), pp. 11–17
198. M. Stonebraker, SQL databases v. NoSQL databases. Commun. ACM **53**(4), 10–11 (2010). ACM, New York
199. M. Stonebraker, G. Held, E. Wong, P. Kreps, The design and implementation of INGRES. ACM Trans. Database Syst. **1**(3), 189–222 (1976). ACM, New York
200. M. Stonebraker, D.J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, S. Zdonik, C-store: a column-oriented DBMS, in *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB '05*, Trondheim (VLDB Endowment, 2005), pp. 553–564
201. H. Sutter, The free lunch is over: a fundamental turn toward concurrency in software. Dr. Dobb's J. **30**(3), 202–210 (2005)
202. R.W. Taylor, R.L. Frank, CODASYL data-base management systems. ACM Comput. Surv. **8**, 67–103 (1976). ACM, New York
203. T.J. Teorey, J.P. Fry, The logical record access approach to database design. ACM Comput. Surv. **12**, 179–211 (1980). ACM, New York

204. T.J. Teorey, S. Lightstone, T. Nadeau, *Database Modelling and Design: Logical Design*, 4th edn. (Morgan Kaufmann, Boston, 2005)
205. A. Thomasian, Performance analysis of database systems, in *Performance Evaluation: Origins and Directions*, ed. by G. Haring, C. Lindemann, M. Reiser. Volume 1769 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2000), pp. 305–327
206. C. Tinnefeld, J. Krüger, J. Schaffner, A. Bog, A database engine for flexible real-time available-to-promise, in *IEEE Symposium on Advanced Management of Information for Globalized Enterprises, AMIGE'08*, Tianjin, 2008
207. Transaction Processing Performance Council, TPC benchmark W (Web commerce). Specification, version 1.8, Feb 2002. Retrieved from http://www.tpc.org/tpcw/. Last accessed 15 June 2012
208. Transaction Processing Performance Council, TPC benchmark DS (decision support). Draft specification, revision 32, 2005. Retrieved from http://tpc.org/tpcds. Last accessed 15 June 2012
209. Transaction Processing Performance Council, TPC benchmark C. Standard specification, revision 5.11, Feb 2010. Retrieved from http://tpc.org/tpcc. Last accessed 15 June 2012
210. Transaction Processing Performance Council, TPC benchmark E. Standard specification, version 1.12.0, June 2010. Retrieved from http://tpc.org/tpce. Last accessed 15 June 2012
211. Transaction Processing Performance Council, TPC-energy specification. Standard specification, version 1.2.0, June 2010. Retrieved from http://www.tpc.org/tpc_energy. Last accessed 15 June 2012
212. Transaction Processing Performance Council, TPC benchmark H (decision support). Standard specification, revision 2.14.0, Feb 2011. Retrieved from http://tpc.org/tpch. Last accessed 15 June 2012
213. Transaction Processing Performance Council, Council website, 2012. Retrieved from http://tpc.org. Last accessed 15 June 2012
214. P. Vassiliadis, A. Simitsis, S. Skiadopoulos, Conceptual modeling for ETL processes, in *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, DOLAP '02*, McLean (ACM, New York, 2002), pp. 14–21
215. P. Vassiliadis, A. Karagiannis, V. Tziovara, A. Simitsis, Towards a benchmark for ETL workflows, in *Proceedings of the Fifth International Workshop on Quality in Databases, QDB*, Vienna, 2007, ed. by V. Ganti, F. Naumann, pp. 49–60
216. M. Vieira, H. Madeira, A dependability benchmark for OLTP application environments, in *Proceedings of the 29th International Conference on Very Large Databases, VLDB'03*, ed. by J.C. Freytag, P.C. Lockemann, S. Abiteboul, M.J. Carey, P.G. Selinger, A. Heuer (VLDB Endowment, Berlin, 2003), pp. 742–753
217. M. Vieira, H. Madeira, From performance to dependability benchmarking: a mandatory path, in *Performance Evaluation and Benchmarking*, ed. by R. Nambiar, M. Poess (Springer, Berlin/Heidelberg, 2009), pp. 67–83
218. M. Vieira, H. Madeira, K. Sachs, S. Kounev, Resilience benchmarking, in *Resilience Assessment and Evaluation of Computing Systems*, ed. by A. Avritzer, A. van Moorsel, M. Vieira, K. Wolter (Springer, Berlin/Heidelberg 2012), pp. 283–301
219. VoltDB, The NewSQL database for high velocity applications. Product website, n.d. Retrieved from http://voltdb.com/products-services. Last accessed 15 June 2012
220. C. White, The next generation of business intelligence: operational BI. White paper, BI research, sponsored by Sybase, July 2006
221. A. White, D. Newman, D. Logan, J. Radcliffe, Mastering master data management. Gartner research, Jan 2006. Retrieved from http://kona.kontera.com/IMAGE_DIR/pdf/MDM_gar_060125_MasteringMDMB.pdf. Last accessed 15 June 2012
222. M. Winslett, David DeWitt speaks out: on rethinking the CS curriculum, why the database community should be proud, why query optimization doesn't work, how supercomputing funding is sometimes very poorly spent, how he's not a good coder and isn't smart enough to do DB theory, and more. ACM SIGMOD Rec. **31**(2), 50–62 (2002). ACM, New York

223. M. Winslett, Bruce lindsay speaks out: on system R, benchmarking, life as an IBM fellow, the power of DBAs in the old days, why performance still matters, Heisenbugs, why he still writes code, singing pigs, and more. ACM SIGMOD Rec. **34**(2), 71–79 (2005). ACM, New York
224. L. Wyatt, B. Caufield, D. Pol, Principles for an ETL benchmark, in *Performance Evaluation and Benchmarking*, ed. by R. Nambiar, M. Poess (Springer, Berlin/Heidelberg, 2009), pp. 183–198
225. P. Youngworth, OLAP spells success for users and developers. Data Based Advis. **13**(11), 38–49 (1995). Data Based Advisor, Escondido
226. N. Yuhanna, M. Gilpin, D. D'Silva, TPC benchmarks don't matter anymore, features and cost are key factors when choosing a DBMS. Forrester research, Mar 2009
227. D.D. Zilio, D.C. Zilio, J. Rao, S. Lightstone, G. Lohman, A. Storm, C. Garcia-arellano, S. Fadden, DB2 design advisor: integrated automatic physical database, in *Proceedings of the Thirtieth international conference on Very Large Data Bases*, Toronto, ed. by M.A. Nascimento, M.T. Özsu, D. Kossmann, R.J. Miller, J.A. Blakeley, K.B. Schiefer. Volume 30 of VLDB '04 (VLDB Endowment, 2004), pp. 1087–1097
228. M. Zukowski, S. Héman, N. Nes, P. Boncz, Cooperative scans: dynamic bandwidth sharing in a DBMS, in *Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB '07*, Vienna, ed. by C. Koch, J. Gehrke, M.N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C.Y. Chan, V. Ganti, C.-C. Kanne, W. Klas, E.J. Neuhold (VLDB Endowment, 2007), pp. 723–734
229. Thomas Zurek and Markus Sinnwell, Datawarehousing has more colours than just black & white, in *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99*, Edinburgh, ed. by M.P. Atkinson, M.E. Orlowska, P. Valduriez, S.B. Zdonik, M.L. Brodie (VLDB Endowment, 1999), pp. 726–729