Enterprise Platform and Integration Concepts Research at HPI

Michael Perscheid, Hasso Plattner, Daniel Ritter, Rainer Schlosser, Ralf Teusner
Hasso Plattner Institute, University of Potsdam, Germany

ABSTRACT

The Hasso Plattner Institute (HPI), academically structured as the independent Faculty of Digital Engineering at the University of Potsdam, unites computer science research and teaching with the advantages of a privately financed institute and a tuition-free study program. Founder and namesake of the institute is the SAP co-founder Hasso Plattner, who also heads the Enterprise Platform and Integration Concepts (EPIC) research center which focuses on the technical aspects of business software with a vision to provide the fastest way to get insights out of enterprise data. Founded in 2006, the EPIC combines three research groups comprising autonomous data management, enterprise software engineering, and data-driven decision support.

1. INTRODUCTION

In the past, data was kept in separate database systems — specializing either in transaction processing (OLTP) or analytical query execution (OLAP) — for performance reasons. As a result, maintaining tedious materialized aggregates and time-consuming, error-prone ETL processes underpinned business applications and made them complex, slow, inflexible as well as difficult to scale and extend. In particular, it was not possible to ask adhoc analytical questions about the current business state without lengthy preparation and data collection steps, and obtaining results required hours instead of seconds.

In 2006, these shortcomings, together with recent advances in hardware such as multi-core CPUs and large main-memory capacities, motivated Hasso Plattner, cofounder of SAP, to question the status quo and to rethink database architectures. He founded the Enterprise Platform and Integration Concepts (EPIC) research center at the HPI and, together with his PhD students and SAP, he created a novel class of database systems that exclusively store and process all data in main-memory [34]. At the core of this idea is the adoption of the columnar data layout allowing for extremely fast scans that are capable of processing billions of records per second even on a single CPU core; cutting down storage requirements with highly efficient in-memory compression; and reducing the need for additional database constructs such as materialized aggregates [35]. With this in-memory database technology, the traditional separation of data processing systems into transactional and analytical could be overcome and the benefits are a simplified system landscape, removal of data replication, increased transactional throughput and business applications that perform real-time reporting on fresh data [36].

Based on the EPIC's research and its open-source, inmemory database prototype — called *Hyrise* — SAP started to develop a professional version with columnar storage for OLTP and OLAP workloads — *SAP HANA* — which is now the foundation of nearly all SAP solutions. Furthermore, the joint research work led to more than 200 peer-reviewed publications and fundamentally changed the approach to data management in enterprises.

Today, EPIC is still researching the next-generation of database management and enterprise software with the goals of maximizing performance as well as improving cost efficiency and business value. Within the three research groups, we focus on (i) autonomous data management on modern in-memory and cloud hardware, (ii) rethinking development of business applications, and (iii) models and quantitative methods for data-driven decision support. For all of our research, we apply our methods to real-world scenarios and showcase future enterprise applications by developing and evaluating prototypes closely together with industrial partners.

In this work, we further motivate the aforementioned research topics, highlight our key contributions, and conclude with collaboration activities across our research projects; for more details see `https://epic.hpi.de`.

2. AUTONOMOUS DATA MANAGEMENT

The trend toward storing more and more data in the cloud requires database providers to focus on self-optimization and cost-efficiency. At the same time, the limited capacity and high costs of main memory are the driving force behind storing less frequently accessed data on cheaper disk-storage, e. g., in so called data lakes.

We address these trends in two connected lines of research, by sharing certain base concepts. First, we research a self-driving (i. e., autonomous / unsupervised) in-memory database management system, called Hyrise, that finds (near-)optimal configurations for important in-memory database concepts like (i) data compression and tiering, (ii) replication and scale-out, (iii) index selection, and (iv) data dependencies for query optimizations, as well as (v) approaches for joint optimizations (e.g., for spatio-temporal data management).

Second, a more recent line of research addresses the efficient in-situ processing of massive amounts of infrequently accessed data stored in data lakes for interactive, analytical workloads. For this, we are developing Skyrise, which leverages advances in modern serverless technologies like Function-as-a-Service (FaaS). We develop Skyrise along pure serverless / FaaS principles, making it a seminal, cost-efficient, and fully functional data processing-as-a-service system.

Subsequently, we discuss our research contributions in the areas of autonomous databases and cloud data processing systems, which includes Hyrise and Skyrise.

2.1 Autonomous In-Memory Database Hyrise

The initial version of Hyrise [12] was publicly presented in 2010, featuring the concept of hybrid row- and column data layouts for in-memory databases that was coincidentally at the core of other database systems at that time (e. g., SAP HANA [11]). Through subsequent research efforts in the aforementioned areas of autonomous databases, Hyrise was re-engineered starting from 2016 [9] and is available open source[1] as an in-memory database system for reproducible research and education. The foundation of Hyrise is comprised of typical storage and query processing components like SQL parser, logical (LQP) and phyiscal query plans (PQP) that are cached for optimization, and an executor that runs PQP operators. The Hyrise system is accessible via PostgreSQL wire protocol, making it comparable with other database systems. Together with a plugin concept that allows for rapidly building prototypes and performing experiments, the competitive performance of Hyrise (cf. [2, 10]), compared to state-of-the-art in-memory systems like HyPer [26] and analytical databases like DuckDB [37], makes it suitable for systems research.

To that foundation, we added self-driving components as plugins (e. g., data tiering), which leverage runtime parameters (e. g., data, workload) for decision making, to predict the impact of potential configuration changes and automatically apply those changes that are deemed beneficial. The changes are associated with tuning options (e. g., creation of secondary indexes, (re-)encoding of columns), which we describe subsequently. Many of the tuning options have shared requirements (e. g., access counters for encoding, tiering).

[1]Hyrise, visited 04/2022: `https://github.com/hyrise/`

Finally, Hyrise is also relevant for teaching upcoming generations of database developers about the intricacies of in-memory computing. In our In-Memory Research Laboratory, students find a suitable environment to experiment and conduct studies on novel main-memory, multi-core and GPU processor technology. Since 2017, over 130 HPI master students have contributed as part of our database seminar, master projects, master theses, or as student assistants to Hyrise.

2.1.1 Data Compression & Tiering

Data compression and tiering are powerful methods to address the memory bottleneck and cost inefficiencies for in-memory databases. The automatic decision on which data compression technique to use in in-memory column stores is challenging due to trade-offs and non-obvious impacts on large workloads. In [2], we propose a solution for an automatic selection of a budget-constraint encoding in Hyrise, based on linear programming (LP) and greedy heuristics. The encoding configurations are robust with respect to runtime performance, adaptable and workload-aware. To ensure performance robustness, LP techniques are applied to achieve equally distributed performance gains over all queries. The results show the potential of significant memory budget reductions without a deterioration of runtime performance.

Similarly, data tiering promises to reduce the amount of data in main memory by moving infrequently used data to cheaper and more elastic lower memory and secondary storage tiers. The challenge is to find an optimal balance for the trade-off between performance and costs. We propose an automatic tiering for Hyrise in [4], using LP, that addresses this challenge. Our approach tracks frequency and pattern of data accesses to identify rarely used data, which are moved to secondary memory tiers (e. g., NVM / SSDs). This method is applicable to column selection problems in general and ensures Pareto-efficiency for varying memory budgets. Since, aspects like selectivity, size and frequency of queries are taken into account, the resulting performance is optimized and outperforms other heuristics.

2.1.2 Replication & Scale-out

Database replication and query load-balancing are important mechanisms to scale query throughput. The analysis of workloads allows load-balancing queries to replica nodes according to their accessed data. As a result, replica nodes must only store and synchronize subsets of the data. However, evenly balancing the load of large-scale workloads while minimizing the memory footprint is complex and challenging. Moreover, state-of-the-art allocation approaches are either time consuming or the resulting allocations are not memory-efficient.

In our work, we used LP-based decomposition techniques to determine optimized data placements and workload distributions [15, 16]. We extended these solutions considering potential node failures [18, 17]. Further, we derived a heuristic solution to compute robust solutions for large, real-life workload instances providing a competitive performance for different potential as well as uncertain workload scenarios [50].

2.1.3 *Index Selection*

For complex workloads, secondary indexes become highly relevant for database performance, but current index selection algorithms are either not fast or not highly competitive, as we found in our survey which evaluates state-of-the-art approaches [27]. To overcome the observed limitations of existing approaches, we developed two new index selection algorithms, serving different purposes: EXTEND [51] determines (near-)optimal solutions with an iterative heuristic. The produced solutions outperform others in most evaluated cases while the selection runtime is up to 10× lower. SWIRL [30] is based on reinforcement learning (RL) and — after training — delivers solutions instantaneously. SWIRL decreases selection runtimes by orders of magnitude, while the solution quality is within 2% of the best solutions. While EXTEND is universally applicable with a high solution quality, SWIRL requires training, but reduces runtimes. In further approaches, we consider different stochastic potential workload scenarios and aim at finding index selections under risk-averse criteria [49].

2.1.4 *Data Dependencies*

Efficient query optimization is usually based on metadata such as cardinalities and other basic statistics. More advanced techniques consider data dependency types such as functional, uniqueness, order, or inclusion constraints / dependencies. In our recent survey [28], we identified 60 query optimization techniques for application areas like join, selection, sorting and set operations in the literature that are based on data dependencies.

Toward an efficient implementation and integration into commercial database systems, we laid out a vision in [29] for a workload-driven discovery system for query optimization. The dependency discovery is considered "lazy" since only those data dependency candidates are considered that are relevant for the observed workload. Our prototypical implementation in Hyrise identifies relevant data dependency candidates based on executed query plans and dynamically validates the candidates against the database, leading to performance improvements.

2.1.5 *Joint Tuning & Spatio-Temporal Decisions*

Challenges for self-driving database systems, which tune their physical design and configuration autonomously, are manifold: such systems have to anticipate future workloads, find robust configurations efficiently, and incorporate knowledge gained by previous actions into later decisions. We present a theoretical, component-based framework for self-driving database systems that enables database integration and development of self-managing functionality with low overhead, by relying on separation of concerns [31]. In [38], we started to implement joint tuning approaches in Hyrise, accounting for combined indexing, sorting, and compression configurations for spatio-temporal applications.

2.2 Serverless Data Analysis in Skyrise

Enterprises are moving their applications to public cloud environments to benefit from the resource elasticity, and cost efficiency that their infrastructures provide. The resulting collocation of applications provides an opportunity to explore application data within and across organizations with integrated analytics. Current database systems, however, lack either cost-efficiency, or in-memory performance for real-time analytical data processing inside an organization on large data.

With Skyrise [1], we explore a data analysis architecture that exploits modern, serverless cloud infrastructure such as Function-as-a-Service (FaaS) to achieve cost-efficiency and low latency through massive, ad-hoc resource elasticity and auto-scaling. While the technical aspects of FaaS have to further evolve and mature for efficient data processing [19], we started employing FaaS for data processing in the cloud [1] by targeting the current sweet spot for sporadic, interactive, analytical, in-situ processing of large amounts of infrequently used data (e. g., stored in data lakes). Skyrise builds on a number of components used in Hyrise (e. g., SQL parsing, query plan translation and optimization), and introduces new solutions for query plan optimization and distributed plan execution on FaaS infrastructure.

3. ENTERPRISE SOFTWARE ENGINEERING

Built on a fast database layer, it is time to question how enterprise applications are developed on top. Therefore, we explore current shortcomings in enterprise software engineering to create methods and tools for new development practices and software architectures. While our current research focuses on the process side of software development, we are shifting our focus to the cloud-native engineering side.

3.1 Software Development Processes

Enterprise software development usually requires several collaborating teams working on a shared codebase. In popular agile process frameworks, such as Scrum, work processes are maintained through iterative process improvement cycles and retrospection meetings. In our research, we show how existing agile methods can be supplemented with process improvement steps based on software engineering team data. Our approach includes gathering empirical data on the perceptions of team members, as well as deriving insights from teams' project data [33]. The underlying artifacts, such as source code, commits, or work documentation, are already being produced during regular development work. By aggregating, linking, analyzing and visualizing the available data, we enable teams to gain actionable insights into their own development processes [32].

Traditionally, software development teams collaborate with other disciplines such as design, management, or sales. As multiple disciplines work differently, such teams may experience problems in their collaboration. Additionally, companies often implement different process frameworks for these disciplines, leaving collaborating teams with a variety of options and tools without guidance on how to integrate them. In our research, we develop InnoDev [7], a comprehensive framework for such teams to work jointly on innovative software products from idea to implementation and market growth. InnoDev integrates three modern frameworks from complementary disciplines: (i) Design Thinking from the field of Design focuses on understanding the users' problems and developing desirable solutions, (ii) Lean Startup from the field of Entrepreneurship focuses on customer and business development, and (iii) Agile practices from the field of Software Development focus on organizing software development collaboratively and flexibly. InnoDev thus serves as a common language and improves teamwork. We provide empirical evidence that Agile scaling and project management activities are well suited to support the conceptualization phases of the software development process. We also show that Design Thinking supports various activities during the development process and increases the team's product understanding, empathy within the team, and empathy towards users [6]. With our toolbox and our workshop [8], we offer practical guidance for teams which want to integrate Design Thinking, Lean Startup, and Agile software development for creating business applications.

3.2 Cloud-native Enterprise Architecture

To fully leverage the potential of fast databases and modern cloud-native development approaches, our research group is moving its focus up one level in the software stack onto the architecture of Enterprise Resource Planning (ERP) systems. ERP systems play a vital role for enterprises by providing comprehensive standard solutions. However, they offer only limited opportunities to customize the standard processes according to companies' individual needs. Additionally, unsatisfying integration with third-party software leads to numerous data silos on-premise and the cloud. For these reasons, after tackling the database foundations, we rethink the general architecture of ERP systems. Future enterprise applications should enable companies to build and support their growing business processes, while maintaining a future-oriented data model that allows for adaptation to the evolving needs of an upcoming enterprise. We start research in that direction by analyzing the requirements, the feasibility, and the effects of implementing a new generation of cloud-native enterprise systems based on *executable business processes* on top of their underlying business data. Thereby, we address the resulting engineering challenges, especially concerning the responsibility of services and the scalability of inter-process communication.

4. DATA-DRIVEN DECISION SUPPORT

Data-driven decision support (DDDS) for enterprise applications has become highly relevant in recent years, as firms face the challenge of integrating data-driven automation in their processes. Based on the ability to effectively store, process, and handle data, the DDDS research group investigates how quantitative methods of operations research and data science can be used to improve automated decision-making. Our scope involves (i) the identification of causal relations (Sec. 4.1), (ii) specific revenue management problems (Sec. 4.2), and (iii) resource allocation and database tuning problems in close collaboration with the ADM group, see Sec. 2.

4.1 Causal Structure Learning (CSL)

The key for effective decision support is to understand and to be able to measure the causal interplay of main variables involved in a specific use case. We address challenges in CSL, from data to learned causal structures, by improvements in both the application of statistical concepts and the computational acceleration.

Knowledge of the causal structures within complex systems is crucial to many domains. For example, in manufacturing [23, 13], where causal structural knowledge constitutes the basis of error avoidance in a production process. While domain experts within the company have enough expertise to identify the most common relationships, they will require support in the context of both an increasing amount of observational data and the complexity of large systems. This gap can be closed by algorithms of CSL that derive the underlying causal structures from observational data, e.g., error messages and inline measurements of a production process.

Our research of data-driven causal inference concentrates on several workstreams which, combined, aim to allow an efficient application of CSL. In a first stream, we work on information-theoretic approaches [20] to improve CSL in the context of heterogeneous data characteristics, which are typical for real-world scenarios. Second, we built a modular pipeline (MPCSL [22]) for experimental evaluation of CSL from observational data accompanied with a first benchmark framework (MANM-CS [21]) to generate test data allowing us to compare the strengths and weaknesses of CSL algorithms. Third, we exploit hard- and software acceleration to speed up CSL algorithms, which are numerically intensive. In particular, we leverage GPUs to develop efficient implementations in heterogeneous computing systems [14, 53].

4.2 Revenue Management (RM)

Over the years, RM applications have become increasingly difficult to administrate. The number of decisions to control business processes have become too complex to be managed manually. As currently established rule-based solutions are far from optimal, automated decision-making has enormous potential. However, it is challenging to derive optimized decisions, as most RM applications lead to complex stochastic dynamic problems. Nevertheless, the overall vision is to develop self-driving decision support systems, which automatically analyze market data and optimize operational business decisions.

One stream of our work aims at analytical solutions of dynamic pricing problems. In contrast to solely numerical solutions, such approaches allow us to gain structural results and general insights regarding optimal decision-making, which allow us to infer managerial recommendations. Results have been derived for pricing applications with time-dependent demand [39], risk aversion [40], joint advertising [41], oligo-poly competition [43], online vs. offline markets [47], and towards a circular economy [5] (cf. sustainability).

Besides studying analytical solutions, we also look for improved and extended methodologies, especially in the area solving Markov decision processes (MDP). For example, we developed approaches to solve MDPs under risk aversion [44] and designed approximate dynamic programming (ADP) techniques to solve, e.g., challenging multi-product pricing problems [42, 45].

In a further project, we partnered with a top 10 seller of used books on Amazon in Germany. The firm has an inventory of over 100,000 distinct books. The challenge was to optimize (expected) profits and the number of sales. The established pricing strategy of our project partner was a rule-based system that has been developed over the past years by carefully adjusting rules to lessons learned from selling books on Amazon. We developed a data-driven pricing strategy based on a relaxed Markov model with estimated sales probabilities. In a real-world comparison, our strategy clearly outperformed the merchant's benchmark strategy with respect to both profitability *and* speed of sales [46].

Another stream of the DDDS group is to study the interplay of self-adaptive pricing strategies in simulated environments. While RL-based strategies require less knowledge and can be applied in highly complex environments, they are also extremely data hungry. Hence, before applying them in practice, they have to be tested in simulated markets. Our group is working on both frontiers, i. e., RL-based pricing strategies [24, 25, 52] as well as flexible simulation frameworks [3, 48, 54].

5. COOPERATION OF THE GROUPS

To reach the vision of building the next generation of enterprise software, the EPIC's research groups are mutually supportive: At the technical bottom, the ADM group works on the fastest data processing approaches, the ESE builds on top of this and rethinks business applications in a cloud-native world, and, finally, the DDDS group benefits from the work of both groups by being able to conveniently access and handle business data. The DDDS group also supports the other groups by providing models, quantitative methods, and solution techniques for their various kinds of resource allocation problems. Automated database tuning problems, in particular, require not only sophisticated mathematical concepts but are also clearly developing in a direction where dynamic aspects, stochastic or uncertain workloads as well as robust solutions — to be able to guarantee good performances in different potential scenarios — are key.

Acknowledgements We would like to thank the EPIC's current and former, very talented PhD students and Post-Doc researchers. Further, we acknowledge our various industry partners. First of all, we thank SAP for funding us and interesting research questions, especially SAP HANA, New Ventures and Technologies, and the entire Technology and Innovation board area, for bringing research results to innovative products. Besides SAP, we are collaborating with Seagate, Porsche, Heidelberger Druckmaschinen, Hilti, AWS, Zalando, and many others on leveraging technologies for enterprise software. Thanks for providing tough research challenges, believing in us, and transferring the results back to business.

6. REFERENCES

[1] T. Bodner. Elastic query processing on function as a service platforms. In *VLDB PhD Workshop, CEUR 2652*, 2020.
[2] M. Boissier. Robust and budget-constrained encoding configurations for in-memory database systems. *Proc. VLDB Endow.*, 15:780–793, 2022.
[3] M. Boissier et al. Data-driven repricing strategies in competitive markets: An interactive simulation platform. In *RecSys*, pages 355–357. ACM, 2017.
[4] M. Boissier, R. Schlosser, and M. Uflacker. Hybrid data layouts for tiered HTAP databases with Pareto-optimal data placements. In *ICDE*, pages 209–220, 2018.
[5] R. Chenavaz, W. Klibi, and R. Schlosser. Dynamic pricing with reference price effects in integrated online and offline retailing. *International Journal of Production Research, to appear*, 2021.

[6] F. Dobrigkeit and D. de Paula. Design thinking in practice: Understanding manifestations of design thinking in software engineering. In *ESEC/FSE*, pages 1059–1069, 2019.

[7] F. Dobrigkeit, D. de Paula, and M. Uflacker. InnoDev - A software development methodology integrating design thinking, scrum and lean startup. In *Design Thinking - Research Looking Further*, pages 199–228. Springer, 2018.

[8] F. Dobrigkeit et al. InnoDev Workshop: A one day introduction to combining design thinking, lean startup and agile software development. In *CSEE&T*, pages 1–10, 2020.

[9] M. Dreseler et al. Hyrise re-engineered: An extensible database system for research in relational in-memory data management. In *EDBT*, pages 313–324, 2019.

[10] M. Dreseler et al. Quantifying TPC-H choke points and their optimizations. *Proc. VLDB Endow.*, 13(8):1206–1220, 2020.

[11] F. Färber et al. The SAP HANA database – an architecture overview. *IEEE Data Eng. Bull.*, 35(1):28–33, 2012.

[12] M. Grund et al. HYRISE - A main memory hybrid storage engine. *Proc. VLDB Endow.*, 4(2):105–116, 2010.

[13] C. Hagedorn et al. Understanding unforeseen production downtimes in manufacturing processes using log data-driven causal reasoning, journal of intelligent manufacturing. *Journal of Intelligent Manufacturing, to appear*, 2022.

[14] C. Hagedorn and J. Huegle. GPU-accelerated constraint-based causal structure learning for discrete data. In *SDM*, pages 37–45. Society for Industrial and Applied Mathematics, 2021.

[15] S. Halfpap and R. Schlosser. A comparison of allocation algorithms for partially replicated databases. In *ICDE*, pages 2008–2011, 2019.

[16] S. Halfpap and R. Schlosser. Workload-driven fragment allocation for partially replicated databases using linear programming. In *ICDE*, pages 1746–1749, 2019.

[17] S. Halfpap and R. Schlosser. Exploration of dynamic query-based load balancing for partially replicated database systems with node failures. In *CIKM*, pages 3409–3412, 2020.

[18] S. Halfpap and R. Schlosser. Memory-efficient database fragment allocation for robust load balancing when nodes fail. In *ICDE*, pages 1811–1816, 2021.

[19] J. M. Hellerstein et al. Serverless computing: One step forward, two steps back. In *CIDR*, 2019.

[20] J. Huegle. An information-theoretic approach on causal structure learning for heterogeneous data characteristics of real-world scenarios. In *IJCAI*, pages 4891–4892, 2021.

[21] J. Huegle et al. MANM-CS: Data generation for benchmarking causal structure learning from mixed discrete-continuous and nonlinear data. In *WHY-21 at NeurIPS 2021*, 2021.

[22] J. Huegle et al. MPCSL - a modular pipeline for causal structure learning. In *SIGKDD*, pages 3068–3076, 2021.

[23] J. Huegle, C. Hagedorn, and M. Uflacker. How causal structural knowledge adds decision-support in monitoring of automotive body shop assembly lines. In *IJCAI*, pages 5246–5248, 2020.

[24] A. Kastius and R. Schlosser. Dynamic pricing under competition using reinforcement learning. *Journal of Revenue and Pricing Management*, 21:50–63, 2022.

[25] A. Kastius and R. Schlosser. Towards transfer learning for revenue and pricing management. In *Operations Research Proceedings, OR2021, to appear*, 2022.

[26] A. Kemper and T. Neumann. HyPer: A hybrid OLTP & OLAP main memory database system based on virtual memory snapshots. In *ICDE*, pages 195–206, 2011.

[27] J. Kossmann et al. Magic mirror in my hand, which is the best in the land? An experimental evaluation of index selection algorithms. *Proc. VLDB Endow.*, 13(11):2382–2395, 2020.

[28] J. Kossmann et al. Data dependencies for query optimization: A survey. *VLDB Journal*, 31:1–22, 2022.

[29] J. Kossmann et al. Workload-driven, lazy discovery of data dependencies for query optimization. In *CIDR*, 2022.

[30] J. Kossmann, A. Kastius, and R. Schlosser. SWIRL: Selection of workload-aware indexes using reinforcement learning. In *EDBT*, pages 155–168, 2022.

[31] J. Kossmann and R. Schlosser. Self-driving database systems: A conceptual approach. *Distributed and Parallel Databases*, 38(4):795–817, 2020.

[32] C. Matthies and F. Dobrigkeit. Experience vs data: A case for more data-informed retrospective activities. In *Lean and Agile Software Development*, pages 130–144. 2021.

[33] C. Matthies, J. Huegle, T. Dürschmid, and R. Teusner. Attitudes, beliefs, and development data concerning agile software development practices. In *ICSE-SEET*, pages 158–169, 2019.

[34] H. Plattner. A common database approach for OLTP and OLAP using an in-memory column database. In *SIGMOD*, pages 1–2, 2009.

[35] H. Plattner. *A Course in In-Memory Data Management: The Inner Mechanics of In-Memory Databases*. Springer, 2013.

[36] H. Plattner and B. Leukert. *The In-Memory Revolution: How SAP HANA Enables Business of the Future*. Springer, 2015.

[37] M. Raasveldt and H. Mühleisen. DuckDB: An embeddable analytical database. In *SIGMOD*, pages 1981–1984, 2019.

[38] K. Richly et al. Joint index, sorting, and compression optimization for memory-efficient spatio-temporal data management. In *ICDE*, pages 1901–1906, 2021.

[39] R. Schlosser. Dynamic pricing with time-dependent elasticities. *Journal of Revenue and Pricing Management*, 14(5):365–383, 2015.

[40] R. Schlosser. A stochastic dynamic pricing and advertising model under risk aversion. *Journal of Revenue and Pricing Management*, 14(6):451–468, 2015.

[41] R. Schlosser. Joint stochastic dynamic pricing and advertising with time-dependent demand. *Journal of Economic Dynamics and Control*, 73:439–452, 2016.

[42] R. Schlosser. Stochastic dynamic multi-product pricing with dynamic advertising and adoption effects. *Journal of Revenue and Pricing Management*, 15(2):153–169, 2016.

[43] R. Schlosser. Stochastic dynamic pricing and advertising in isoelastic oligopoly models. *European Journal of Operational Research*, 259(3):1144–1155, 2017.

[44] R. Schlosser. Risk-sensitive control of markov decision processes: A moment-based approach with target distributions. *Computers and Operations Research*, 123:104997, 2020.

[45] R. Schlosser. Scalable relaxation techniques to solve stochastic dynamic multi-product pricing problems with substitution effects. *Journal of Revenue and Pricing Management*, 20:54–65, 2021.

[46] R. Schlosser and M. Boissier. Dynamic pricing under competition on online marketplaces: A data-driven approach. *SIGKDD*, pages 705–714, 2018.

[47] R. Schlosser, R. Chenavaz, and S. Dimitrov. Circular economy: Joint dynamic pricing and recycling investments. *International Journal of Production Economics*, 236(108117):1–13, 2021.

[48] R. Schlosser et al. Data-driven inventory management and dynamic pricing competition on online marketplaces. In *IJCAI*, pages 5856–5858, 2018.

[49] R. Schlosser and S. Halfpap. A decomposition approach for risk-averse index selection. In *SSDBM*, pages 16:1–16:4, 2020.

[50] R. Schlosser and S. Halfpap. Robust and memory-efficient database fragment allocation for large and uncertain database workloads. In *EDBT*, pages 367–372, 2021.

[51] R. Schlosser, J. Kossmann, and M. Boissier. Efficient scalable multi-attribute index selection using recursive strategies. In *ICDE*, pages 1238–1249, 2019.

[52] R. Schlosser and K. Richly. Dynamic pricing under competition with data-driven price anticipations and endogenous reference price effects. *Journal of Revenue and Pricing Management*, 18:451–464, 2019.

[53] C. Schmidt et al. Order-independent constraint-based causal structure learning for gaussian distribution models using GPUs. In *SSDBM*, pages 19:1–19:10, 2018.

[54] S. Serth et al. An interactive platform to simulate dynamic pricing competition on online marketplaces. In *EDOC 2017*, pages 61–66, 2017.