Data Structures for Data-Intensive Applications: Tradeoffs and Design Guidelines

Other titles in Foundations and Trends® in Databases

Consensus in Data Management: From Distributed Commit to Blockchain
Faisal Nawab and Mohammad Sadoghi
ISBN: 978-1-63828-160-3

Multidimensional Array Data Management
Florin Rusu
ISBN: 978-1-63828-148-1

Modern Datalog Engines
Bas Ketsman and Paraschos Koutris
ISBN: 978-1-63828-042-2

Natural Language Interfaces to Data
Abdul Quamar, Vasilis Efthymiou, Chuan Lei and Fatma Özcan
ISBN: 978-1-63828-028-6

Trends in Explanations: Understanding and Debugging Data-driven Systems
Boris Glavic, Alexandra Meliou and Sudeepa Roy
ISBN: 978-1-68083-880-0

Differential Privacy for Databases
Joseph P. Near and Xi He
ISBN: 978-1-68083-850-3

Data Structures for Data-Intensive Applications: Tradeoffs and Design Guidelines

Manos Athanassoulis
Boston University
mathan@bu.edu

Stratos Idreos
Harvard University
stratos@seas.harvard.edu

Dennis Shasha
New York University
shasha@courant.nyu.edu

the essence of knowledge
Boston — Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

M. Athanassoulis *et al.*. *Data Structures for Data-Intensive Applications: Tradeoffs and Design Guidelines*. Foundations and Trends® in Databases, vol. 13, no. 1-2, pp. 1–168, 2023.

ISBN: 978-1-63828-185-6

Editorial Scope

Topics

Foundations and Trends® in Databases publishes survey and tutorial articles in the following topics:

- Data Models and Query Languages
- Query Processing and Optimization
- Storage, Access Methods, and Indexing
- Transaction Management, Concurrency Control and Recovery
- Deductive Databases
- Parallel and Distributed Database Systems
- Database Design and Tuning
- Metadata Management
- Object Management
- Trigger Processing and Active Databases
- Data Mining and OLAP
- Approximate and Interactive Query Processing
- Data Warehousing
- Adaptive Query Processing
- Data Stream Management
- Search and Query Integration
- XML and Semi-Structured Data
- Web Services and Middleware
- Data Integration and Exchange
- Private and Secure Data Management
- Peer-to-Peer, Sensornet and Mobile Data Management
- Scientific and Spatial Data Management
- Data Brokering and Publish/Subscribe
- Data Cleaning and Information Extraction
- Probabilistic Data Management

Information for Librarians

Foundations and Trends® in Databases, 2023, Volume 13, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Contents

Data Structures for Data-Intensive Applications: Tradeoffs and Design Guidelines

Manos Athanassoulis[1], Stratos Idreos[2] and Dennis Shasha[3]

[1]*Boston University, USA; mathan@bu.edu*

[2]*Harvard University, USA; stratos@seas.harvard.edu*

[3]*New York University, USA; shasha@cs.nyu.edu*

ABSTRACT

Key-value data structures constitute the core of any data-driven system. They provide the means to store, search, and modify data residing at various levels of the storage and memory hierarchy, from durable storage (spinning disks, solid state disks, and other non-volatile memories) to random access memory, caches, and registers. Designing efficient data structures for given workloads has long been a focus of research and practice in both academia and industry.

This book outlines the underlying design dimensions of data structures and shows how they can be combined to support (or fail to support) various workloads. The book further shows how these design dimensions can lead to an understanding of the behavior of individual state-of-the-art data structures and their hybrids. Finally, this systematization of the *design space* and the accompanying guidelines will enable you to select the most fitting data structure or even to invent an entirely new data structure for a given workload.

Manos Athanassoulis, Stratos Idreos and Dennis Shasha (2023), "Data Structures for Data-Intensive Applications: Tradeoffs and Design Guidelines", Foundations and Trends® in Databases: Vol. 13, No. 1-2, pp 1–168. DOI: 10.1561/1900000059.

1

Introduction

1.1 Data Structures Are Foundational

Data structures are the means by which software programs store and retrieve data. This book focuses on key-value data structures, which are widely used for data-intensive applications thanks to the versatility of the key-value data model. Key-value data structures manage a collection of *key-value entries*, with the property that a given key maps to only one value but the same value can be associated with many keys. The value part of a data entry may have arbitrary semantics. For example, it may be a record in a relational database or a Pandas DataFrame, or an arbitrary set of fields that the application knows how to parse and use in a NoSQL system. In some settings, such as when systems manage data for social networks, the value may contain a reference to a large object such as an image or video.

Physically, a key-value data structure consists of (1) the data, physically stored in some layout, (2) optional metadata to facilitate navigation over the data, and (3) algorithms to support storage and retrieval operations (Hellerstein *et al.*, 2007; Selinger *et al.*, 1979; Idreos *et al.*, 2018a). Other terms used in the literature for data structures include "access methods," "data containers," and "search structures."

Data systems, operating systems, file systems, compilers, and network systems employ a diverse set of data structures. This book draws examples primarily from the area of large-volume data systems which require secondary storage devices, but the core analysis and design dimensions apply to purely in-memory systems as well, where access to RAM (random access memory) is far slower than access to the cache. In fact, the analysis applies to any setting in which there are two or more levels in the memory/storage hierarchy.

Given the wealth of applications that can be modeled using key-value data, such data structures have enormous general utility. For example, a particular data structure can be used to describe (i) metadata access in files, networks, and operating systems (Bovet and Cesati, 2005; Rodeh, 2008), (ii) data access in relational systems (Hellerstein *et al.*, 2007), (iii) data access in NoSQL and NewSQL systems (Idreos and Callaghan, 2020; Mohan, 2014), and (iv) feature engineering and model structures in machine learning pipelines (Wasay *et al.*, 2021).

Each application, or *workload*, can be represented as a mixture of key-value operations (point queries, range queries, inserts, deletes, and modifications) it supports over its data. In addition, the amount of memory and persistent storage required, along with their cost, shape the requirements of a given application. For example, file systems manage file metadata and contents using data structures optimized for frequent updates. Compilers typically use hash maps to manage variables during the variables' lifespan and use abstract syntax trees to capture the overall shape of a program. Similarly, network devices require specialized data structures to efficiently store and access routing tables.

As data-intensive applications emerge and evolve over time, using efficient data structures becomes critical to the viability of such applications, sometimes resulting in a three orders of magnitude performance change, as shown by Chatterjee *et al.* (2022). The reason is that data movement is the major bottleneck in data-intensive applications. Data movement is largely governed by the way data is stored, i.e., by the data structure. Thus, we expect that there will be an ongoing need for new data structures as new applications appear, hardware changes and data grows. Currently, research in academia and industry produces several new data structure designs every year, and this pace is expected to grow. At the same time, with a growing set of new data structures available, even the task of choosing from an off-the-shelf data structure, that is, one that can be found in textbooks, has become more complex.

This book aims to explain the space of data structure design choices, how to select the appropriate data structure depending on the goals and workload of an application at hand, and how the ever-evolving hardware and data properties require innovations in data structure design. The overarching goal is to help the reader both select the best existing data structures and design and build new ones.

1.2 Tradeoffs in Data Structure Design

Every data structure represents a particular workload- and hardware-dependent performance tradeoff (formalized by Athanassoulis *et al.*, 2016 and Idreos *et al.*, 2018b). In order to choose an existing data structure or to design a new data structure for a particular workload on particular hardware, you should understand the possible design space of data structure design clearly and formally. That is the focus of this book. To motivate that discussion, let us look at a few examples of designs and tradeoffs when considering the workload (Section 1.2.1) as well as the underlying hardware (Section 1.2.2) and how they both evolve over time.

1.2.1 Workload-Driven Designs

Optimizing for a Workload. Consider a workload that consists of a small number of inserts and updates together with a large number of point and range queries. In order to balance the read and the write cost, many applications employ a B^+-tree, originally proposed by Bayer and McCreight (1972) and later surveyed by Graefe (2011). B^+-trees have a high node fanout, so that traversing from root to leaf requires few secondary memory accesses, and their top levels are cached in the faster levels of the memory hierarchy (Section 4.1). Further, a B^+-tree supports range queries by maintaining all the keys sorted in the leaf nodes and by connecting the leaf nodes in a linked list. As the number of insertions and updates increase, however, leaf nodes must be reorganized and maybe even split, which can become a performance bottleneck.

To address workloads having many inserts, a completely different approach is taken by a data structure called the log-structured merge-tree (LSM-tree). LSM-trees were originally introduced by O'Neil *et al.* (1996), and their many variants were surveyed by Luo and Carey (2020). As we will see in Section 4.8, LSM-trees place all updates in a common memory buffer which is flushed to disk when it becomes full. As more buffers accumulate, they are merged to form larger sorted data collection. This design employs an *out-of-place* policy of handling modifications, surveyed in detail by Sarkar and Athanassoulis (2022), in which there can be many key-value pairs in the structure having the same key. (For a given key k, the most recently inserted key-value pair for k has the current value.)

Thus two different workloads – one with more read queries and one with more insert operations – suggest different data structures.

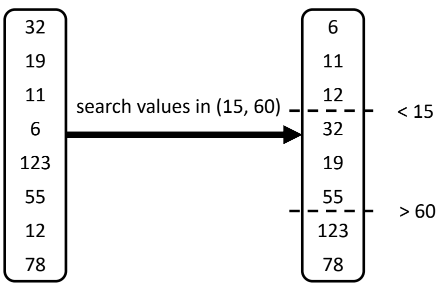

Figure 1.1: Adaptive data organization using the most recent search as a hint. In this example, a range query for values 15 to 60 leads to partitioning the base data in three non-overlapping partitions, one with values less than 15, one with values between 15 and 60, and one with values greater than 60.

Adapting to a Workload. Because the central theme of this book is to design a data structure given the expected workload, we also consider designing data structures that gradually adapt to the ideal design. To illustrate this point, consider that the B$^+$-tree and the LSM-tree, as originally designed, impose a sorted order within disk-resident nodes in order to answer any point or range query. An adaptive data structure may start with one or more unsorted nodes, but sorts them gradually in an opportunistic way, as shown in Figure 1.1. Chapter 5 will explain the concept of database cracking, proposed by Idreos *et al.* (2007a) and further expanded by Idreos *et al.* (2007b) and Idreos *et al.* (2009). Intuitively, cracking uses the access patterns of incoming queries to continuously and incrementally physically reorganize the underlying data with the goal of improving the performance of future queries.

1.2.2 Memory-Driven And Storage-Driven Designs

As a complement to workload-based considerations, hardware advances create new challenges, needs, and opportunities in data structure design. Over the years, the *memory and storage hierarchy* has been enriched with devices such as solid-state disks, non-volatile memories, and deep cache hierarchies. Here, we discuss a few key hardware considerations for data structures, and we expand on them in Chapters 6 and 7.

Optimizing for the Storage/Memory Hierarchy. In a storage hierarchy, the lower levels offer a lot of storage at a low price but at high access latency, and as we move higher, that is, closer to the processor(s), the storage is faster but smaller and more expensive per byte. In the storage/memory hierarchy there is always a level that is the bottleneck for a given application, which depends on the size of the application data relative to the storage capacity available at the different levels of the hierarchy.

Originally, B^+-trees tried to minimize disk accesses by maximizing the fanout. As the memory sizes grew, however, much of the data could fit into random access memory or non-volatile secondary memory. This changed the tradeoffs dramatically. For example, in-memory B^+-trees perform best with small fanout, as shown by Kester *et al.* (2017).

Memory Wall. While the memory hierarchy is expanding with technologies like high-bandwidth memory as outlined by Pohl *et al.* (2020), a key hardware trend for several decades has been the growing disparity between processor speed and the speed of off-chip memory, termed *the memory wall* by Wulf and McKee (1995). Since the early 2000s, operating systems, as discussed by Milojicic and Roscoe (2016), and data management systems, as discussed by Johnson *et al.* (2009), have been carefully re-designed to account for the memory wall by optimizing the use of cache memories.

Storage Devices Evolve. In addition, secondary storage itself has reached a crossover point. Traditional hard disks have long since hit their physical speed limits (Athanassoulis, 2014), and have largely been replaced by shingled disks and flash-based devices (Hughes, 2013). Shingled disks increase the density of storage on the magnetic medium, changing the performance properties of disks because the granularity of reads and writes changes (Hughes, 2013). Flash-based drives offer significantly faster read performance than traditional disks, but suffer from relatively poor write performance. Further, flash-based drives are equipped with a highly functional firmware layer called the Flash Translation Layer (FTL), which can be rewritten to yield dramatic changes in performance. Thus, flash hardware performance depends on both hardware and firmware. Such changes create a need for new data structures to optimize for different hardware/firmware combinations.

1.3 Audience & Prerequisites

This book aims to be used as part of graduate-level classes on designing complex data structures. It assumes at least an undergraduate-level familiarity with data structures.

Specifically, we assume that the reader is already familiar with basic data structures like arrays, linked lists, binary trees, queues, stacks, heaps, and hash tables, all taught in introductory courses for data structures and algorithms (Cormen *et al.*, 2009). Such basic data structures sit at the core of the more complex designs we outline. Most of the use cases and the presented designs are motivated by data-intensive systems, hence building on classical data structures used in database systems like B^+-trees (Bayer and McCreight, 1972), open hashing (Ramakrishnan and Gehrke, 2002), and LSM-trees (O'Neil *et al.*, 1996).

Other than that, the book is self-contained. We will describe all algorithms used in navigating complex data structures and outline how to combine the various design decisions we introduce.

1.4 Learning Outcomes

After reading this book, you will be able to reason about which existing data structure will perform best given a workload and the underlying hardware. In addition, you will be able to design new and possibly hybrid data structures to handle workloads with different composition, locality, and access patterns.

1.5 Overview of the Book

Here is the outline of our book. We recommend that you read the chapters sequentially.

- Chapter 2 introduces the fundamental performance metrics for data structures with respect to the most important key-value operations and hardware properties.
- Chapter 3 presents the core set of design principles, primarily based on workload characteristics, that largely guide the design of key-value data structures.
- Chapter 4 starts by explaining the design and performance characteristics of traditional data structures based on the design principles. We then discuss how to use the design principles to design new data structures for arbitrary workloads.
- Chapter 5 discusses the need for and design principles underlying adaptive data structures. We also illustrate use cases in which adaptivity leads to greatly improved performance.
- Chapter 6 discusses how data structures are utilized in big data applications, including databases, file systems, and machine learning.
- Chapter 7 discusses additional design considerations that can influence the detailed deployment of data structures ranging from deploying data structures in a setting with concurrent execution, in the context of distributed systems, to new hardware, new type workloads, and new application requirements.

References

Abadi, D. J., P. A. Boncz, S. Harizopoulos, S. Idreos, and S. Madden. (2013). "The Design and Implementation of Modern Column-Oriented Database Systems". *Foundations and Trends in Databases*. 5(3): 197–280. DOI: 10.1561/1900000024.

Abadi, D. J., D. S. Myers, D. J. DeWitt, and S. R. Madden. (2007). "Materialization Strategies in a Column-Oriented DBMS". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 466–475. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4221695.

Abu-Libdeh, H., D. Altınbüken, A. Beutel, E. H. Chi, L. Doshi, T. Kraska, Xiaozhou, Li, A. Ly, and C. Olston. (2020). "Learned Indexes for a Google-scale Disk-based Database". In: *Proceedings of the Workshop on ML for Systems at NeurIPS*. URL: http://mlforsystems.org/assets/papers/neurips2020/learned_abu-libdeh_2020.pdf.

Adelson-Velsky, G. and E. Landis. (1962). "An algorithm for the organization of information". *Proceedings of the USSR Academy of Sciences*. 146: 263–266.

Aggarwal, A. and J. S. Vitter. (1988). "The Input/Output Complexity of Sorting and Related Problems". *Communications of the ACM*. 31(9): 1116–1127. DOI: 10.1145/48529.48535.

Ailamaki, A., D. J. DeWitt, M. D. Hill, and M. Skounakis. (2001). "Weaving Relations for Cache Performance". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 169–180. URL: http://dl.acm.org/citation.cfm?id=645927.672367.

Ailamaki, A., D. J. DeWitt, M. D. Hill, and D. A. Wood. (1999). "DBMSs on a modern processor: Where does time go?" In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 266–277. URL: http://128.105.2.28/pub/techreports/1999/TR1394.pdf.

Ailamaki, A., E. Liarou, P. Tözün, D. Porobic, and I. Psaroudakis. (2014). "How to stop under-utilization and love multicores". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 189–192. DOI: 10.1109/ICDE.2015.7113419.

Ailamaki, A., E. Liarou, P. Tözün, D. Porobic, and I. Psaroudakis. (2017). *Databases on Modern Hardware: How to Stop Underutilization and Love Multicores*. *Synthesis Lectures on Data Management*. Morgan & Claypool Publishers. DOI: 10.2200/S00774ED1V01Y201704DTM045.

Ailamaki, A., D. J. DeWitt, and M. D. Hill. (2002). "Data Page Layouts for Relational Databases on Deep Memory Hierarchies". *The VLDB Journal*. 11(3): 198–215. DOI: 10.1007/s00778-002-0074-9.

Alagiannis, I., S. Idreos, and A. Ailamaki. (2014). "H2O: A Hands-free Adaptive Store". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1103–1114. DOI: 10.1145/2588555.2610502.

Alsubaiee, S., Y. Altowim, H. Altwaijry, A. Behm, V. R. Borkar, Y. Bu, M. J. Carey, I. Cetindil, M. Cheelangi, K. Faraaz, E. Gabrielova, R. Grover, Z. Heilbron, Y.-S. Kim, C. Li, G. Li, J. M. Ok, N. Onose, P. Pirzadeh, V. J. Tsotras, R. Vernica, J. Wen, and T. Westmann. (2014). "AsterixDB: A Scalable, Open Source BDMS". *Proceedings of the VLDB Endowment*. 7(14): 1905–1916. DOI: 10.14778/2733085.2733096.

Andersen, D. G., J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. (2009). "FAWN: A Fast Array of Wimpy Nodes". In: *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*. 1–14. DOI: 10.1145/1629575.1629577.

Antoshenkov, G. (1995). "Byte-aligned Bitmap Compression". In: *Proceedings of the Conference on Data Compression (DCC)*. 476–476. URL: http://dl.acm.org/citation.cfm?id=874051.874730.

Arulraj, J., A. Pavlo, and P. Menon. (2016). "Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 583–598. DOI: 10.1145/2882903.2915231.

Arumugam, S., A. Dobra, C. M. Jermaine, N. Pansare, and L. Perez. (2010). "The DataPath System: A Data-centric Analytic Processing Engine for Large Data Warehouses". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 519–530. URL: http://dl.acm.org/citation.cfm?id=1807167.1807224.

Athanassoulis, M. (2014). "Solid-State Storage and Work Sharing for Efficient Scaleup Data Analytics". *PhD thesis*. EPFL. DOI: http://dx.doi.org/10.5075/epfl-thesis-6032.

Athanassoulis, M. and A. Ailamaki. (2014). "BF-Tree: Approximate Tree Indexing". *Proceedings of the VLDB Endowment*. 7(14): 1881–1892. URL: http://www.vldb.org/pvldb/vol7/p1881-athanassoulis.pdf.

Athanassoulis, M., A. Ailamaki, S. Chen, P. B. Gibbons, and R. Stoica. (2010). "Flash in a DBMS: Where and How?" *IEEE Data Engineering Bulletin*. 33(4): 28–34. URL: http://sites.computer.org/debull/A10dec/athanassoulis.pdf.

Athanassoulis, M., S. Chen, A. Ailamaki, P. B. Gibbons, and R. Stoica. (2015). "Online Updates on Data Warehouses via Judicious Use of Solid-State Storage". *ACM Transactions on Database Systems (TODS)*. 40(1).

Athanassoulis, M. and S. Idreos. (2016). "Design Tradeoffs of Data Access Methods". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Tutorial*.

Athanassoulis, M., M. S. Kester, L. M. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan. (2016). "Designing Access Methods: The RUM Conjecture". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 461–466. URL: http://dx.doi.org/10.5441/002/edbt.2016.42.

Athanassoulis, M., S. Sarkar, T. I. Papon, Z. Zhu, and D. Staratzis. (2022). "Building Deletion-Compliant Data Systems". In: *IEEE Data Engineering Bulletin*. 21–36.

Badam, A., K. Park, V. S. Pai, and L. L. Peterson. (2009). "Hash-Cache: Cache Storage for the Next Billion". In: *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 123–136. URL: http://dl.acm.org/citation.cfm?id=1558977.1558986.

Barber, R., P. Bendel, M. Czech, O. Draese, F. Ho, N. Hrle, S. Idreos, M.-S. Kim, O. Koeth, J.-G. Lee, T. T. Li, G. M. Lohman, K. Morfonios, R. Müller, K. Murthy, I. Pandis, L. Qiao, V. Raman, R. Sidle, K. Stolze, and S. Szabo. (2012). "Business Analytics in (a) Blink". *IEEE Data Engineering Bulletin*. 35(1): 9–14. URL: http://sites.computer.org/debull/A12mar/blink.pdf.

Barber, R., G. M. Lohman, V. Raman, R. Sidle, S. Lightstone, and B. Schiefer. (2015). "In-Memory BLU Acceleration in IBM's DB2 and dashDB: Optimized for Modern Workloads and Hardware Architectures". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*.

Bayer, R. (1972). "Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms". *Acta Informatica*. 1: 290–306. DOI: 10.1007/BF00289509.

Bayer, R. and E. M. McCreight. (1970). "Organization and Maintenance of Large Ordered Indices". In: *Proceedings of the ACM SIGFIDET Workshop on Data Description and Access*. 107–141. DOI: 10.1007/BF00288683.

Bayer, R. and E. M. McCreight. (1972). "Organization and Maintenance of Large Ordered Indices". *Acta Informatica*. 1(3): 173–189. DOI: 10.1007/BF00288683.

Becker, B., S. Gschwind, T. Ohler, B. Seeger, and P. Widmayer. (1996). "An Asymptotically Optimal Multiversion B-Tree". *The VLDB Journal*. 5(4): 264–275. DOI: 10.1007/s007780050028.

Bender, M. A., M. Farach-Colton, W. Jannen, R. Johnson, B. C. Kuszmaul, D. E. Porter, J. Yuan, and Y. Zhan. (2015). "An Introduction to Bϵ-trees and Write-Optimization". *White Paper*. URL: http://supertech.csail.mit.edu/papers/BenderFaJa15.pdf.

Bender, M. A., M. Farach-Colton, R. Johnson, R. Kraner, B. C. Kuszmaul, D. Medjedovic, P. Montes, P. Shetty, R. P. Spillane, and E. Zadok. (2012). "Don't Thrash: How to Cache Your Hash on Flash". *Proceedings of the VLDB Endowment*. 5(11): 1627–1637. URL: http://dl.acm.org/citation.cfm?id=2350229.2350275.

Bentley, J. L. (1979). "Decomposable Searching Problems". *Information Processing Letters*. 8(5): 244–251. DOI: 10.1016/0020-0190(79)90117-0.

Bentley, J. L. and J. B. Saxe. (1980). "Decomposable Searching Problems I: Static-to-Dynamic Transformation". *Journal of Algorithms*. 1(4): 301–358. DOI: 10.1016/0196-6774(80)90015-2.

Bentley, J. L. and A. C.-C. Yao. (1976). "An Almost Optimal Algorithm for Unbounded Searching". *Information Processing Letters*. 5(3): 82–87. DOI: 10.1016/0020-0190(76)90071-5.

Bernstein, P. A., V. Hadzilacos, and N. Goodman. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

Bhat, W. A. (2018). "Bridging data-capacity gap in big data storage". *Future Generation Computer Systems*. DOI: https://doi.org/10.1016/j.future.2017.12.066.

Bloom, B. H. (1970). "Space/Time Trade-offs in Hash Coding with Allowable Errors". *Communications of the ACM*. 13(7): 422–426. URL: http://dl.acm.org/citation.cfm?id=362686.362692.

Boncz, P. A., M. L. Kersten, and S. Manegold. (2008). "Breaking the Memory Wall in MonetDB". *Communications of the ACM*. 51(12): 77–85. DOI: 10.1145/1409360.1409380.

Boncz, P. A., S. Manegold, and M. L. Kersten. (1999). "Database architecture optimized for the new bottleneck: Memory access". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 54–65. URL: http://www.vldb.org/conf/1999/P5.pdf.

Boncz, P. A., M. Zukowski, and N. J. Nes. (2005). "MonetDB/X100: Hyper-Pipelining Query Execution". In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*.

Borovica-Gajic, R., S. Idreos, A. Ailamaki, M. Zukowski, and C. Fraser. (2015). "Smooth Scan: Statistics-Oblivious Access Paths". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 315–326. DOI: 10.1109/ICDE.2015.7113294.

Bovet, D. P. and M. Cesati. (2005). *Understanding the Linux Kernel*. 3rd Editio. O'Reilly Media, Inc.

Breslow, A. and N. Jayasena. (2018). "Morton Filters: Faster, Space-Efficient Cuckoo Filters via Biasing, Compression, and Decoupled Logical Sparsity". *Proceedings of the VLDB Endowment*. 11(9): 1041–1055. DOI: 10.14778/3213880.3213884.

Brodal, G. S. and R. Fagerberg. (2003). "Lower Bounds for External Memory Dictionaries". In: *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 546–554. URL: http://dl.acm.org/citation.cfm?id=644108.644201%20http://www.cs.au.dk/~gerth/papers/alcomft-tr-03-75.pdf.

Candea, G., N. Polyzotis, and R. Vingralek. (2009). "A scalable, predictable join operator for highly concurrent data warehouses". *Proceedings of the VLDB Endowment*. 2(1): 277–288. URL: http://dl.acm.org/citation.cfm?id=1687627.1687659.

Candea, G., N. Polyzotis, and R. Vingralek. (2011). "Predictable Performance and High Query Concurrency for Data Analytics". *The VLDB Journal*. 20(2): 227–248. URL: http://dl.acm.org/citation.cfm?id=1969331.1969355.

Chan, C. Y., B. C. Ooi, and H. Lu. (1992). "Extensible Buffer Management of Indexes". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 444–454. URL: http://www.vldb.org/conf/1992/P444.PDF.

Chan, C.-Y. and Y. E. Ioannidis. (1998). "Bitmap index design and evaluation". *ACM SIGMOD Record*. 27(2): 355–366. DOI: 10.1145/276305.276336.

Chan, C.-Y. and Y. E. Ioannidis. (1999). "An efficient bitmap encoding scheme for selection queries". *ACM SIGMOD Record*. 28(2): 215–226. DOI: 10.1145/304181.304201.

Chandramouli, B., G. Prasaad, D. Kossmann, J. J. Levandoski, J. Hunter, and M. Barnett. (2018). "FASTER: A Concurrent Key-Value Store with In-Place Updates". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 275–290. DOI: 10.1145/3183713.3196898.

Chang, F., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. (2006). "Bigtable: A Distributed Storage System for Structured Data". In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 205–218. URL: http://dl.acm.org/citation.cfm?id=1267308.1267323.

Chatterjee, S., M. Jagadeesan, W. Qin, and S. Idreos. (2022). "Cosine: A Cloud-Cost Optimized Self-Designing Key-Value Storage Engine". In: *In Proceedings of the Very Large Databases Endowment*. DOI: 10.14778/3485450.3485461.

Chazelle, B. and L. J. Guibas. (1985). "Fractional Cascading: A Data Structuring Technique with Geometric Applications". In: *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*. 90–100. DOI: 10.1007/BFb0015734.

Chen, L. (2009). "Curse of Dimensionality". In: *Encyclopedia of Database Systems*. Ed. by L. Liu and T. Özsu. Boston, MA: Springer US. 545–546. DOI: 10.1007/978-0-387-39940-9{_}133.

Chen, S., P. B. Gibbons, and T. C. Mowry. (2001). "Improving Index Performance through Prefetching". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 235–246. DOI: 10.1145/375663.375688.

Chen, S., P. B. Gibbons, T. C. Mowry, and G. Valentin. (2002). "Fractal Prefetching B+-Trees: Optimizing Both Cache and Disk Performance". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 157–168. URL: http://dl.acm.org/citation.cfm?id=564691.564710.

Chen, S., P. B. Gibbons, and S. Nath. (2011). "Rethinking Database Algorithms for Phase Change Memory". In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*.

Colantonio, A. and R. Di Pietro. (2010). "Concise: Compressed 'N' Composable Integer Set". *Information Processing Letters*. 110(16): 644–650. DOI: 10.1016/j.ipl.2010.05.018.

Comer, D. (1979). "The Ubiquitous B-Tree". *ACM Computing Surveys*. 11(2): 121–137. DOI: 10.1145/356770.356776.

Copeland, G. P. and S. Khoshafian. (1985). "A Decomposition Storage Model". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 268–279. URL: http://dl.acm.org/citation.cfm?id=318898.318923.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. (2009). *Introduction to Algorithms, 3rd Edition*. MIT Press. URL: http://mitpress.mit.edu/books/introduction-algorithms.

Dayan, N., M. Athanassoulis, and S. Idreos. (2017). "Monkey: Optimal Navigable Key-Value Store". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 79–94. DOI: 10.1145/3035918.3064054.

Dayan, N., M. Athanassoulis, and S. Idreos. (2018). "Optimal Bloom Filters and Adaptive Merging for LSM-Trees". *ACM Transactions on Database Systems (TODS)*. 43(4): 1–16. DOI: 10.1145/3276980.

Dayan, N. and S. Idreos. (2018). "Dostoevsky: Better Space-Time Trade-Offs for LSM-Tree Based Key-Value Stores via Adaptive Removal of Superfluous Merging". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 505–520. DOI: 10.1145/3183713.3196927.

Dayan, N. and S. Idreos. (2019). "The Log-Structured Merge-Bush & the Wacky Continuum". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 449–466. DOI: 10.1145/3299869.3319903.

Dayan, N., Y. Rochman, I. Naiss, S. Dashevsky, N. Rabinovich, E. Bortnikov, I. Maly, O. Frishman, I. B. Zion, Avraham, M. Twitto, U. Beitler, E. Ginzburg, and M. Mokryn. (2021). "The End of Moore's Law and the Rise of The Data Processor". *Proceedings of the VLDB Endowment*. 14(12): 2932–2944. URL: http://www.vldb.org/pvldb/vol14/p2932-dayan.pdf.

Debnath, B., S. Sengupta, and J. Li. (2010). "FlashStore: high throughput persistent key-value store". *Proceedings of the VLDB Endowment*. 3(1-2): 1414–1425. URL: http://dl.acm.org/citation.cfm?id=1920841.1921015.

Debnath, B., S. Sengupta, and J. Li. (2011). "SkimpyStash: RAM space skimpy key-value store on flash-based storage". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 25–36. DOI: 10.1145/1989323.1989327.

Deeds, K., B. Hentschel, and S. Idreos. (2020). "Stacked Filters: Learning to Filter by Structure". *Proceedings of the VLDB Endowment*. 14(4): 600–612. DOI: 10.14778/3436905.3436919.

Deliège, F. and T. B. Pedersen. (2010). "Position list word aligned hybrid: optimizing space and performance for compressed bitmaps". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 228–239. DOI: 10.1145/1739041.1739071.

Ding, B., S. Chaudhuri, and V. R. Narasayya. (2020a). "Bitvector-aware Query Optimization for Decision Support Queries". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2011–2026. DOI: 10.1145/3318464.3389769.

Ding, J., U. F. Minhas, B. Chandramouli, C. Wang, Y. Li, Y. Li, D. Kossmann, J. Gehrke, and T. Kraska. (2021). "Instance-Optimized Data Layouts for Cloud Analytics Workloads". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 418–431. DOI: 10.1145/3448016.3457270.

Ding, J., U. F. Minhas, J. Yu, C. Wang, J. Do, Y. Li, H. Zhang, B. Chandramouli, J. Gehrke, D. Kossmann, D. B. Lomet, and T. Kraska. (2020b). "ALEX: An Updatable Adaptive Learned Index". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 969–984. DOI: 10.1145/3318464.3389711.

Dittrich, J. and A. Jindal. (2011). "Towards a One Size Fits All Database Architecture". In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*. 195–198. URL: http://cidrdb.org/cidr2011/Papers/CIDR11_Paper25.pdf.

Dong, J. and R. Hull. (1982). "Applying Approximate Order Dependency to Reduce Indexing Space". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 119–127. DOI: 10.1145/582353.582375.

Dong, S., M. Callaghan, L. Galanis, D. Borthakur, T. Savor, and M. Strum. (2017). "Optimizing Space Amplification in RocksDB". In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*. URL: http://cidrdb.org/cidr2017/papers/p82-dong-cidr17.pdf.

Dong, S., A. Kryczka, Y. Jin, and M. Stumm. (2021). "RocksDB: Evolution of Development Priorities in a Key-value Store Serving Large-scale Applications". *ACM Transactions on Storage (TOS)*. 17(4): 26:1–26:32. DOI: 10.1145/3483840.

Faerber, F., A. Kemper, P.-Å. Larson, J. J. Levandoski, T. Neumann, and A. Pavlo. (2017). "Main Memory Database Systems". *Foundations and Trends in Databases*. 8(1-2): 1–130. DOI: 10.1561/1900000058.

Falkoff, A. D. and K. E. Iverson. (1973). "The Design of APL". *IBM Journal of Research and Development*. 17(5): 324–334. DOI: 10.1147/rd.174.0324.

Fan, B., D. G. Andersen, M. Kaminsky, and M. Mitzenmacher. (2014). "Cuckoo Filter: Practically Better Than Bloom". In: *Proceedings of the ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT)*. 75–88. DOI: 10.1145/2674005.2674994.

Fang, J., Y. T. B. Mulder, J. Hidders, J. Lee, and H. P. Hofstee. (2020). "In-memory database acceleration on FPGAs: a survey". *The VLDB Journal*. 29(1): 33–59. DOI: 10.1007/s00778-019-00581-w.

Färber, F., S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner. (2011). "SAP HANA Database: Data Management for Modern Business Applications". *ACM SIGMOD Record*. 40(4): 45–51. DOI: 10.1145/2094114.2094126.

Färber, F., N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees. (2012). "The SAP HANA Database – An Architecture Overview". *IEEE Data Engineering Bulletin*. 35(1): 28–33. URL: http://sites.computer.org/debull/A12mar/hana.pdf.

Ferragina, P. and G. Vinciguerra. (2020). "The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds". *Proceedings of the VLDB Endowment*. 13(8): 1162–1175. DOI: 10.14778/3389133.3389135.

Fox, E. A., Q. F. Chen, A. M. Daoud, and L. S. Heath. (1991). "Order-Preserving Minimal Perfect Hash Functions and Information Retrieval". *ACM Transactions on Information Systems (TOIS)*. 9(3): 281–308. DOI: 10.1145/125187.125200.

Francisco, P. (2011). "The Netezza Data Appliance Architecture: A Platform for High Performance Data Warehousing and Analytics". *IBM Redbooks*. URL: http://www.redbooks.ibm.com/redpapers/pdfs/redp4725.pdf.

French, C. D. (1995). ""One size fits all" database architectures do not work for DSS". *ACM SIGMOD Record*. 24(2): 449–450. DOI: 10.1145/568271.223871.

French, C. D. (1997). "Teaching an OLTP Database Kernel Advanced Data Warehousing Techniques". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 194–198. URL: http://dl.acm.org/citation.cfm?id=645482.653422.

Galakatos, A., M. Markovitch, C. Binnig, R. Fonseca, and T. Kraska. (2019). "FITing-Tree: A Data-aware Index Structure". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1189–1206. DOI: 10.1145/3299869.3319860.

Giannikis, G., G. Alonso, and D. Kossmann. (2012). "SharedDB: Killing One Thousand Queries with One Stone". *Proceedings of the VLDB Endowment*. 5(6): 526–537. URL: http://dl.acm.org/citation.cfm?id=2168651.2168654.

Giannikis, G., D. Makreshanski, G. Alonso, and D. Kossmann. (2013). "Workload optimization using SharedDB". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1045–1048. DOI: 10.1145/2463676.2463678.

Gottstein, R., R. Goyal, S. Hardock, I. Petrov, and A. P. Buchmann. (2014). "MV-IDX: indexing in multi-version databases". In: *Proceedings of the Symposium on International Database Engineering & Applications (IDEAS)*. 142–148. DOI: 10.1145/2628194.2628911.

Graefe, G. (2003). "Sorting And Indexing With Partitioned B-Trees". In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*. URL: http://www-db.cs.wisc.edu/cidr/cidr2003/program/p1.pdf.

Graefe, G. (2011). "Modern B-Tree Techniques". *Foundations and Trends in Databases*. 3(4): 203–402. URL: http://dx.doi.org/10.1561/1900000028.

Graefe, G., F. Halim, S. Idreos, H. Kuno, and S. Manegold. (2012). "Concurrency control for adaptive indexing". *Proceedings of the VLDB Endowment*. 5(7): 656–667. URL: http://dl.acm.org/citation.cfm?id=2180918.

Graefe, G., S. Idreos, H. Kuno, and S. Manegold. (2010). "Benchmarking adaptive indexing". In: *Proceedings of the TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*. 169–184. URL: http://dl.acm.org/citation.cfm?id=1946050.1946063.

Graefe, G. and H. Kuno. (2010a). "Self-selecting, self-tuning, incrementally optimized indexes". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 371–381. URL: http://dl.acm.org/citation.cfm?id=1739041.1739087.

Graefe, G. and H. A. Kuno. (2010b). "Adaptive indexing for relational keys". In: *Proceedings of the IEEE International Conference on Data Engineering Workshops (ICDEW)*. 69–74.

Graefe, G., H. Volos, H. Kimura, H. A. Kuno, J. Tucek, M. Lillibridge, and A. C. Veitch. (2014). "In-Memory Performance for Big Data". *Proceedings of the VLDB Endowment*. 8(1): 37–48. DOI: 10.14778/2735461.2735465.

Gray, J. and F. Putzolu. (1986). "The 5 Minute Rule for Trading Memory for Disc Accesses and the 5 Byte Rule for Trading Memory for CPU Time". *Tandem Computers - Technical Report*. URL: http://db.cs.berkeley.edu/cs286/papers/fiveminute-tr1986.pdf.

Grund, M., J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden. (2010). "HYRISE: A Main Memory Hybrid Storage Engine". *Proceedings of the VLDB Endowment*. 4(2): 105–116. DOI: 10.14778/1921071.1921077.

Guttman, A. (1984). “R-Trees: A Dynamic Index Structure for Spatial Searching”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 47–57. DOI: 10.1145/602259.602266.

Haeupler, B., S. Sen, and R. E. Tarjan. (2015). “Rank-Balanced Trees”. *ACM Transactions on Algorithms (TALG)*. 11(4): 30:1–30:26. DOI: 10.1145/2689412.

Halim, F., S. Idreos, P. Karras, and R. H. C. Yap. (2012). “Stochastic Database Cracking: Towards Robust Adaptive Indexing in Main-Memory Column-Stores.” *Proceedings of the VLDB Endowment*. 5(6): 502–513. URL: http://vldb.org/pvldb/vol5/p502_felixhalim_vldb2012.pdf.

Harizopoulos, S. and A. Ailamaki. (2003). “A Case for Staged Database Systems”. In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*.

Harizopoulos, S., V. Shkapenyuk, and A. Ailamaki. (2005). “QPipe: A Simultaneously Pipelined Relational Query Engine”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 383–394. URL: http://dl.acm.org/citation.cfm?id=1066157.1066201.

Hellerstein, J. M., E. Koutsoupias, D. P. Miranker, C. H. Papadimitriou, and V. Samoladas. (2002). “On a Model of Indexability and Its Bounds for Range Queries”. *Journal of the ACM*. 49(1): 35–55. DOI: 10.1145/505241.505244.

Hellerstein, J. M., E. Koutsoupias, and C. H. Papadimitriou. (1997). “On the Analysis of Indexing Schemes”. In: *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*. 249–256. DOI: 10.1145/263661.263688.

Hellerstein, J. M., J. F. Naughton, and A. Pfeffer. (1995). “Generalized Search Trees for Database Systems”. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 562–573. URL: http://dl.acm.org/citation.cfm?id=645921.673145.

Hellerstein, J. M., M. Stonebraker, and J. R. Hamilton. (2007). “Architecture of a Database System”. *Foundations and Trends in Databases*. 1(2): 141–259. DOI: 10.1561/1900000002.

Héman, S., M. Zukowski, and N. J. Nes. (2010). "Positional Update Handling in Column Stores". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 543–554. URL: http://dl.acm.org/citation.cfm?id=1807167.1807227.

Herlihy, M., N. Shavit, V. Luchangco, and M. Spear. (2020). *The Art of Multiprocessor Programming (Second Edition)*. Morgan Kaufmann. DOI: 10.1016/C2011-0-06993-4.

Hilbert, M. and P. López. (2011). "The World's Technological Capacity to Store, Communicate, and Compute Information". *Science*. 332(6025): 60–65. DOI: 10.1126/science.1200970.

Holanda, P., S. Manegold, H. Mühleisen, and M. Raasveldt. (2019). "Progressive Indexes: Indexing for Interactive Data Analysis". *Proceedings of the VLDB Endowment*. 12(13): 2366–2378. URL: http://www.vldb.org/pvldb/vol12/p2366-holanda.pdf.

Huang, G., X. Cheng, J. Wang, Y. Wang, D. He, T. Zhang, F. Li, S. Wang, W. Cao, and Q. Li. (2019). "X-Engine: An Optimized Storage Engine for Large-scale E-commerce Transaction Processing". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 651–665. DOI: 10.1145/3299869.3314041.

Huang, H. and S. Ghandeharizadeh. (2021). "Nova-LSM: A Distributed, Component-based LSM-tree Key-value Store". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 749–763. DOI: 10.1145/3448016.3457297.

Huang, K., Y. He, and T. Wang. (2022). "The Past, Present and Future of Indexing on Persistent Memory". *Proceedings of the VLDB Endowment*. 15(12): 3774–3777. URL: https://www.vldb.org/pvldb/vol15/p3774-wang.pdf%20https://www2.cs.sfu.ca/~tzwang/pmem-index-tutorial-slides.pdf.

Hughes, J. (2013). "Revolutions in Storage". In: *IEEE Conference on Massive Data Storage*. Long Beach, CA. URL: http://storageconference.us/2013/Presentations/Hughes.pdf.

Hutflesz, A., H.-W. Six, and P. Widmayer. (1988). "Globally Order Preserving Multidimensional Linear Hashing". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 572–579. DOI: 10.1109/ICDE.1988.105505.

Idreos, S. and M. Callaghan. (2020). “Key-Value Storage Engines”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2667–2672. DOI: 10.1145/3318464.3383133.

Idreos, S., N. Dayan, W. Qin, M. Akmanalp, S. Hilgard, A. Ross, J. Lennon, V. Jain, H. Gupta, D. Li, and Z. Zhu. (2019a). “Design Continuums and the Path Toward Self-Designing Key-Value Stores that Know and Learn”. In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*.

Idreos, S., F. Groffen, N. Nes, S. Manegold, K. S. Mullender, and M. L. Kersten. (2012). “MonetDB: Two Decades of Research in Column-oriented Database Architectures”. *IEEE Data Engineering Bulletin*. 35(1): 40–45. URL: http://sites.computer.org/debull/A12mar/monetdb.pdf.

Idreos, S., M. L. Kersten, and S. Manegold. (2007a). “Database Cracking”. In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*.

Idreos, S., M. L. Kersten, and S. Manegold. (2007b). “Updating a Cracked Database”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 413–424. DOI: 10.1145/1247480.1247527.

Idreos, S., M. L. Kersten, and S. Manegold. (2009). “Self-organizing Tuple Reconstruction in Column-Stores”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 297–308. DOI: 10.1145/1559845.1559878.

Idreos, S., S. Manegold, H. Kuno, and G. Graefe. (2011). “Merging What’s Cracked, Cracking What’s Merged: Adaptive Indexing in Main-Memory Column-Stores”. *Proceedings of the VLDB Endowment*. 4(9): 586–597. URL: https://www.vldb.org/pvldb/vol4/p586-idreos.pdf.

Idreos, S., K. Zoumpatianos, M. Athanassoulis, N. Dayan, B. Hentschel, M. S. Kester, D. Guo, L. M. Maas, W. Qin, A. Wasay, and Y. Sun. (2018a). “The Periodic Table of Data Structures”. *IEEE Data Engineering Bulletin*. 41(3): 64–75. URL: http://sites.computer.org/debull/A18sept/p64.pdf.

Idreos, S., K. Zoumpatianos, S. Chatterjee, W. Qin, A. Wasay, B. Hentschel, M. S. Kester, N. Dayan, D. Guo, M. Kang, and Y. Sun. (2019b). “Learning Data Structure Alchemy”. *IEEE Data Engineering Bulletin*. 42(2): 47–58. URL: http://sites.computer.org/debull/A19june/p47.pdf.

Idreos, S., K. Zoumpatianos, B. Hentschel, M. S. Kester, and D. Guo. (2018b). “The Data Calculator: Data Structure Design and Cost Synthesis from First Principles and Learned Cost Models”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 535–550. DOI: 10.1145/3183713.3199671.

István, Z., K. Kara, and D. Sidler. (2020). “FPGA-Accelerated Analytics: From Single Nodes to Clusters”. *Foundations and Trends in Databases*. 9(2): 101–208. DOI: 10.1561/1900000072.

Jain, A., A. Phanishayee, J. Mars, L. Tang, and G. Pekhimenko. (2018). “Gist: Efficient Data Encoding for Deep Neural Network Training”. In: *Proceedings of the ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*. 776–789. DOI: 10.1109/ISCA.2018.00070.

Jannen, W., J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. (2015). “BetrFS: A Right-optimized Write-optimized File System”. In: *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*. 301–315. URL: http://dl.acm.org/citation.cfm?id=2750482.2750505.

Jensen, S. K., T. B. Pedersen, and C. Thomsen. (2017). “Time Series Management Systems: A Survey”. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 29(11): 2581–2600. DOI: 10.1109/TKDE.2017.2740932.

Jin, R., S. J. Kwon, and T.-S. Chung. (2011). “FlashB-tree: A Novel B-tree Inex Scheme for Solid State Drives”. In: *Proceedings of the ACM Symposium on Research in Applied Computation (RACS)*. 50–55. DOI: 10.1145/2103380.2103390.

Johnson, R., S. Harizopoulos, N. Hardavellas, K. Sabirli, I. Pandis, A. Ailamaki, N. G. Mancheril, and B. Falsafi. (2007). "To Share or Not to Share?" In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 351–362. URL: http://dl.acm.org/citation.cfm?id=1325851.1325894.

Johnson, R., I. Pandis, N. Hardavellas, A. Ailamaki, and B. Falsafi. (2009). "Shore-MT: a scalable storage manager for the multicore era". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 24–35. URL: http://dl.acm.org/citation.cfm?id=1516360.1516365.

Johnson, T. and D. Shasha. (1989). "Utilization of B-trees with Inserts, Deletes and Modifies". In: *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*. 235–246. DOI: 10.1145/73721.73745.

Johnson, T. and D. Shasha. (1994). "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 439–450. URL: http://dl.acm.org/citation.cfm?id=645920.672996.

Kang, D., D. Jung, J.-U. Kang, and J.-S. Kim. (2007). "mu-tree: an ordered index structure for NAND flash memory". In: *Proceedings of the ACM/IEEE International Conference on Embedded Software (EMSOFT)*. 144–153. URL: http://dl.acm.org/citation.cfm?doid=1289927.1289953.

Kang, Y., R. Pitchumani, P. Mishra, Y.-S. Kee, F. Londono, S. Oh, J. Lee, and D. D. G. Lee. (2019). "Towards building a high-performance, scale-in key-value storage system". In: *Proceedings of the ACM International Conference on Systems and Storage (SYSTOR)*. 144–154. DOI: 10.1145/3319647.3325831.

Karger, D. R., E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin. (1997). "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web". In: *Proceedings of the Annual ACM Symposium on the Theory of Computing (STOC)*. 654–663. DOI: 10.1145/258533.258660.

Kemper, A. and T. Neumann. (2011). “HyPer: A Hybrid OLTP & OLAP Main Memory Database System Based on Virtual Memory Snapshots”. In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 195–206. DOI: 10.1109/ICDE.2011.5767867.

Kennedy, O. and L. Ziarek. (2015). “Just-In-Time Data Structures”. In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*. URL: http://www.cidrdb.org/cidr2015/Papers/CIDR15_Paper9.pdf.

Kester, M. S., M. Athanassoulis, and S. Idreos. (2017). “Access Path Selection in Main-Memory Optimized Data Systems: Should I Scan or Should I Probe?” In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 715–730. DOI: 10.1145/3035918.3064049.

Kim, W., K.-C. Kim, and A. G. Dale. (1989). “Indexing Techniques for Object-Oriented Databases”. In: *Object-Oriented Concepts, Databases, and Applications*. ACM Press and Addison-Wesley. 371–394.

Koltsidas, I. and S. D. Viglas. (2011). “Data management over flash memory”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1209–1212. DOI: 10.1145/1989323.1989455.

Kornacker, M., C. Mohan, and J. M. Hellerstein. (1997). “Concurrency and Recovery in Generalized Search Trees”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 62–72. DOI: 10.1145/253260.253272.

Kraska, T., A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. (2018). “The Case for Learned Index Structures”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 489–504. DOI: 10.1145/3183713.3196909.

Krishna, S., N. Patel, D. Shasha, and T. Wies. (2021). “Automated Verification of Concurrent Search Structures”. *Synthesis Lectures on Computer Science*. 9(1): 1–188. DOI: 10.2200/S01089ED1V01Y202104CSL013.

Kuszmaul, B. C. (2014). “A Comparison of Fractal Trees to Log-Structured Merge (LSM) Trees”. *Tokutek White Paper*.

Lahiri, T., S. Chavan, M. Colgan, D. Das, A. Ganesh, M. Gleeson, S. Hase, A. Holloway, J. Kamp, T.-H. Lee, J. Loaiza, N. Macnaughton, V. Marwah, N. Mukherjee, A. Mullick, S. Muthulingam, V. Raja, M. Roth, E. Soylemez, and M. Zait. (2015). "Oracle Database In-Memory: A Dual Format In-Memory Database". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*.

Lam, M. S., E. E. Rothberg, and M. E. Wolf. (1991). "The Cache Performance and Optimizations of Blocked Algorithms". In: *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 63–74. DOI: 10.1145/106972.106981.

Lamb, A., M. Fuller, and R. Varadarajan. (2012). "The Vertica Analytic Database: C-Store 7 Years Later". *Proceedings of the VLDB Endowment*. 5(12): 1790–1801. URL: http://dl.acm.org/citation.cfm?id=2367518.

Lang, H., T. Mühlbauer, F. Funke, P. A. Boncz, T. Neumann, and A. Kemper. (2016). "Data Blocks: Hybrid OLTP and OLAP on Compressed Storage using both Vectorization and Compilation". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. DOI: 10.1145/2882903.2882925.

Lang, H., T. Neumann, A. Kemper, and P. A. Boncz. (2019). "Performance-Optimal Filtering: Bloom overtakes Cuckoo at High-Throughput". *Proceedings of the VLDB Endowment*. 12(5): 502–515.

Larson, P.-A., C. Clinciu, E. N. Hanson, A. Oks, S. L. Price, S. Rangarajan, A. Surna, and Q. Zhou. (2011). "SQL server column store indexes". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1177–1184. DOI: 10.1145/1989323.1989448.

Larson, P.-A., E. N. Hanson, and S. L. Price. (2012). "Columnar Storage in SQL Server 2012". *IEEE Data Engineering Bulletin*. 35(1): 15–20. URL: http://sites.computer.org/debull/A12mar/apollo.pdf.

Larson, P.-A., E. N. Hanson, and M. Zwilling. (2015). "Evolving the Architecture of SQL Server for Modern Hardware Trends". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*.

Larson, P.-A., R. Rusanu, M. Saubhasik, C. Clinciu, C. Fraser, E. N. Hanson, M. Mokhtar, M. Nowakiewicz, V. Papadimos, S. L. Price, and S. Rangarajan. (2013). "Enhancements to SQL server column stores". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1159–1168. DOI: 10.1145/2463676.2463708.

Leeuwen, J. van and M. H. Overmars. (1981). "The Art of Dynamizing". In: *Proceedings of Mathematical Foundations of Computer Science*. 121–131. DOI: 10.1007/3-540-10856-4{_}78.

Leeuwen, J. van and D. Wood. (1980). "Dynamization of Decomposable Searching Problems". *Information Processing Letters*. 10(2): 51–56. DOI: 10.1016/S0020-0190(80)90073-3.

Lehman, P. L. and S. B. Yao. (1981). "Efficient Locking for Concurrent Operations on B-Trees". *ACM Transactions on Database Systems (TODS)*. 6(4): 650–670. DOI: 10.1145/319628.319663.

Lehman, T. J. and M. J. Carey. (1986). "A Study of Index Structures for Main Memory Database Management Systems". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 294–303. URL: http://www.vldb.org/conf/1986/P294.PDF.

Leis, V., A. Kemper, and T. Neumann. (2013). "The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 38–49. DOI: 10.1109/ICDE.2013.6544812.

Lerner, A. and P. Bonnet. (2021). "Not your Grandpa's SSD: The Era of Co-Designed Storage Devices". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2852–2858. DOI: 10.1145/3448016.3457540.

Levandoski, J. J., D. B. Lomet, and S. Sengupta. (2013). "The Bw-Tree: A B-tree for New Hardware Platforms". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 302–313. DOI: 10.1109/ICDE.2013.6544834.

Li, Y., B. He, J. Yang, Q. Luo, K. Yi, and R. J. Yang. (2010). "Tree Indexing on Solid State Drives". *Proceedings of the VLDB Endowment*. 3(1-2): 1195–1206. URL: http://dl.acm.org/citation.cfm?id=1920841.1920990.

Lim, H., B. Fan, D. G. Andersen, and M. Kaminsky. (2011). "SILT: A Memory-Efficient, High-Performance Key-Value Store". In: *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*. 1–13. URL: http://dl.acm.org/citation.cfm?id=2043556.2043558.

Lin, K.-I., H. V. Jagadish, and C. Faloutsos. (1994). "The TV-Tree: An Index Structure for High-Dimensional Data". *The VLDB Journal*. 3(4): 517–542. URL: http://www.vldb.org/journal/VLDBJ3/P517.pdf.

Litwin, W. (1980). "Linear Hashing: A New Tool for File and Table Addressing". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 212–223.

Litwin, W. and D. B. Lomet. (1986). "The Bounded Disorder Access Method". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 38–48. URL: http://dl.acm.org/citation.cfm?id=645471.655390.

Litwin, W., M.-A. Neimat, and D. A. Schneider. (1994). "RP*: A Family of Order Preserving Scalable Distributed Data Structures". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 342–353. URL: http://www.vldb.org/conf/1994/P342.PDF.

Litwin, W., M.-A. Neimat, and D. A. Schneider. (1996). "LH* - A Scalable, Distributed Data Structure". *ACM Transactions on Database Systems (TODS)*. 21(4): 480–525. DOI: 10.1145/236711.236713.

Lively, T., L. Schroeder, and C. Mendizábal. (2018). "Splaying Log-Structured Merge-Trees". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1839–1841. DOI: 10.1145/3183713.3183723.

Lomet, D. B. and B. Salzberg. (1990). "The hB-Tree: A Multiattribute Indexing Method with Good Guaranteed Performance". *ACM Transactions on Database Systems (TODS)*. 15(4): 625–658. DOI: 10.1145/99935.99949.

Lomet, D. B. and B. Salzberg. (1992). "Access Method Concurrency with Recovery". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 351–360. DOI: 10.1145/130283.130336.

Lu, H. and B. C. Ooi. (1993). “Spatial Indexing: Past and Future”. *IEEE Data Engineering Bulletin*. 16(3): 16–21. URL: http://sites.computer.org/debull/93SEP-CD.pdf.

Lu, L., T. S. Pillai, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. (2016). “WiscKey: Separating Keys from Values in SSD-conscious Storage”. In: *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*. 133–148. URL: https://www.usenix.org/conference/fast16/technical-sessions/presentation/lu.

Luo, C. and M. J. Carey. (2022). “DynaHash: Efficient Data Rebalancing in Apache AsterixDB”. In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 485–497. DOI: 10.1109/ICDE53745.2022.00041.

Luo, C. and M. J. Carey. (2020). “LSM-based Storage Techniques: A Survey”. *The VLDB Journal*. 29(1): 393–418. DOI: 10.1007/s00778-019-00555-y.

Ma, S., T. Ma, K. Chen, and Y. Wu. (2022). “A Survey of Storage Systems in the RDMA Era”. *IEEE Transactions on Parallel and Distributed Systems*. 33(12): 4395–4409. DOI: 10.1109/TPDS.2022.3188656.

MacNicol, R. and B. French. (2004). “Sybase IQ Multiplex - Designed For Analytics”. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 1227–1230. URL: http://www.vldb.org/conf/2004/IND8P3.PDF.

Al-Mamun, A., H. Wu, and W. G. Aref. (2020). “A Tutorial on Learned Multi-dimensional Indexes”. In: *Proceedings of the International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. 1–4. DOI: 10.1145/3397536.3426358.

Manegold, S. (2009). “Memory Hierarchy”. In: *Encyclopedia of Database Systems*. Ed. by L. Liu and T. Özsu. Boston, MA: Springer US. 1707–1713. DOI: 10.1007/978-0-387-39940-9{_}657.

Manegold, S., P. A. Boncz, and M. L. Kersten. (2002a). “Generic Database Cost Models for Hierarchical Memory Systems”. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 191–202. URL: http://dl.acm.org/citation.cfm?id=1287369.1287387.

Manegold, S., P. A. Boncz, and M. L. Kersten. (2002b). "Optimizing Main-Memory Join on Modern Hardware". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 14(4): 709–730. DOI: 10.1109/TKDE.2002.1019210.

Manegold, S., P. A. Boncz, and N. Nes. (2004). "Cache-Conscious Radix-Decluster Projections". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 684–695. URL: http://www.vldb.org/conf/2004/RS18P3.PDF.

Manolopoulos, Y., A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis. (2006). *R-Trees: Theory and Applications. Advanced Information and Knowledge Processing*. Springer. DOI: 10.1007/978-1-84628-293-5.

Mao, Y., E. Kohler, and R. T. Morris. (2012). "Cache Craftiness for Fast Multicore Key-value Storage". In: *Proceedings of the ACM European Conference on Computer Systems (EuroSys)*. 183–196. DOI: 10.1145/2168836.2168855.

Margaritov, A., D. Ustiugov, E. Bugnion, and B. Grot. (2018). "Virtual Address Translation via Learned Page Table Indexes". In: *Proceedings of the Workshop on ML for Systems at NeurIPS*. URL: http://mlforsystems.org/assets/papers/neurips2018/virtual_margaritov_2018.pdf.

Mediano, M. R., M. A. Casanova, and M. Dreux. (1994). "V-Trees - A Storage Method for Long Vector Data". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 321–330. URL: http://www.vldb.org/conf/1994/P321.PDF.

Megiddo, N. and D. S. Modha. (2004). "Outperforming LRU with an Adaptive Replacement Cache Algorithm". *Computer*. 37(4): 58–65. DOI: 10.1109/MC.2004.1297303.

Mehta, M., V. Soloviev, and D. J. DeWitt. (1993). "Batch Scheduling in Parallel Database Systems". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 400–410. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=344041.

Menon, J., D. A. Pease, R. M. Rees, L. Duyanovich, and B. L. Hillsberg. (2003). "IBM Storage Tank - A heterogeneous scalable SAN file system". *IBM Systems Journal*. 42(2): 250–267. DOI: 10.1147/sj.422.0250.

Milojicic, D. S. and T. Roscoe. (2016). "Outlook on Operating Systems". *IEEE Computer*. 49(1): 43–51. DOI: 10.1109/MC.2016.19.

Moerkotte, G. (1998). "Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 476–487. URL: http://dl.acm.org/citation.cfm?id=645924.671173.

Mohan, C. (2014). "Tutorial: An In-Depth Look at Modern Database Systems". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 674. DOI: 10.5441/002/edbt.2014.72.

Morrison, D. R. (1968). "PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric". *Journal of the ACM*. 15(4): 514–534. DOI: 10.1145/321479.321481.

Muth, P., P. E. O'Neil, A. Pick, and G. Weikum. (2000). "The LHAM Log-structured History Data Access Method". *The VLDB Journal*. 8(3-4): 199–221.

Na, G.-J., B. Moon, and S.-W. Lee. (2011). "IPLB+-tree for Flash Memory Database Systems". *Journal of Information Science and Engineering (JISE)*. 27(1): 111–127. URL: http://www.iis.sinica.edu.tw/page/jise/2011/201101_08.html.

Narayanan, D., A. Phanishayee, K. Shi, X. Chen, and M. Zaharia. (2021). "Memory-Efficient Pipeline-Parallel DNN Training". In: *Proceedings of the International Conference on Machine Learning (ICML)*. Vol. 139. *Proceedings of Machine Learning Research*. 7937–7947. URL: http://proceedings.mlr.press/v139/narayanan21a.html.

Nath, S. and A. Kansal. (2007). "FlashDB: dynamic self-tuning database for NAND flash". *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*.

O'Neil, E. J., P. E. O'Neil, and G. Weikum. (1993). "The LRU-K page replacement algorithm for database disk buffering". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 297–306. DOI: 10.1145/170035.170081.

O'Neil, P. E., E. Cheng, D. Gawlick, and E. J. O'Neil. (1996). "The log-structured merge-tree (LSM-tree)". *Acta Informatica.* 33(4): 351–385. URL: http://dl.acm.org/citation.cfm?id=230823.230826.

Olma, M., M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, and A. Ailamaki. (2020). "Adaptive partitioning and indexing for in situ query processing". *The VLDB Journal.* 29(1): 569–591. URL: https://doi.org/10.1007/s00778-019-00580-x.

Ooi, B. C., R. Sacks-Davis, and J. Han. (1993). "Indexing in Spatial Databases". *Tech. rep.*

Overmars, M. H. and J. van Leeuwen. (1981). "Worst-Case Optimal Insertion and Deletion Methods for Decomposable Searching Problems". *Information Processing Letters.* 12(4): 168–173. DOI: 10.1016/0020-0190(81)90093-4.

Pandey, P., A. Conway, J. Durie, M. A. Bender, M. Farach-Colton, and R. Johnson. (2021). "Vector Quotient Filters: Overcoming the Time/Space Trade-Off in Filter Design". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 1386–1399. DOI: 10.1145/3448016.3452841.

Papon, T. I. and M. Athanassoulis. (2021a). "A Parametric I/O Model for Modern Storage Devices". In: *Proceedings of the International Workshop on Data Management on New Hardware (DAMON).*

Papon, T. I. and M. Athanassoulis. (2021b). "The Need for a New I/O Model". In: *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR).*

Papon, T. I. and M. Athanassoulis. (2023). "ACEing the Bufferpool Management Paradigm for Modern Storage Devices". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE).*

Paul, J., S. Lu, and B. He. (2021). "Database Systems on GPUs". *Foundations and Trends in Databases.* 11(1): 1–108. DOI: 10.1561/1900000076.

Perl, Y., A. Itai, and H. Avni. (1978). "Interpolation Search—A log logN Search". *Communications of the ACM.* 21(7): 550–553. URL: http://dl.acm.org/citation.cfm?id=359545.359557.

Petraki, E., S. Idreos, and S. Manegold. (2015). "Holistic Indexing in Main-memory Column-stores". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data.*

Pohl, C., K.-U. Sattler, and G. Graefe. (2020). "Joins on high-bandwidth memory: a new level in the memory hierarchy". *The VLDB Journal*. 29(2-3): 797–817. DOI: 10.1007/s00778-019-00546-z.

Polychroniou, O., A. Raghavan, and K. A. Ross. (2015). "Rethinking SIMD Vectorization for In-Memory Databases". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1493–1508. DOI: 10.1145/2723372.2747645.

Porobic, D., E. Liarou, P. Tözün, and A. Ailamaki. (2014). "ATraPos: Adaptive transaction processing on hardware Islands". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 688–699. DOI: 10.1109/ICDE.2014.6816692.

Psaroudakis, I., M. Athanassoulis, and A. Ailamaki. (2013). "Sharing Data and Work Across Concurrent Analytical Queries". *Proceedings of the VLDB Endowment*. 6(9): 637–648. URL: http://dl.acm.org/citation.cfm?id=2536360.2536364.

Pugh, W. (1990). "Skip Lists: A Probabilistic Alternative to Balanced Trees". *Communications of the ACM*. 33(6): 668–676. URL: http://dl.acm.org/citation.cfm?id=78977.

Qiao, L., V. Raman, F. Reiss, P. J. Haas, and G. M. Lohman. (2008). "Main-memory Scan Sharing for Multi-core CPUs". *Proceedings of the VLDB Endowment*. 1(1): 610–621. URL: http://dl.acm.org/citation.cfm?id=1453856.1453924.

Qin, W. and S. Idreos. (2016). "Adaptive Data Skipping in Main-Memory Systems". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2255–2256. DOI: 10.1145/2882903.2914836.

Ramakrishnan, R. and J. Gehrke. (2002). *Database Management Systems*. McGraw-Hill Higher Education, 3rd edition.

Ramamurthy, R., D. J. DeWitt, and Q. Su. (2002). "A Case for Fractured Mirrors". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 430–441. DOI: 10.1007/s00778-003-0093-1.

Ramamurthy, R., D. J. DeWitt, and Q. Su. (2003). "A Case for Fractured Mirrors". *The VLDB Journal*. 12(2): 89–101. DOI: 10.1007/s00778-003-0093-1.

Raman, A., S. Sarkar, M. Olma, and M. Athanassoulis. (2023). "Indexing for Near-Sorted Data". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*.

Raman, V., G. M. Lohman, T. Malkemus, R. Mueller, I. Pandis, B. Schiefer, D. Sharpe, R. Sidle, A. Storm, L. Zhang, G. K. Attaluri, R. Barber, N. Chainani, D. Kalmuk, V. KulandaiSamy, J. Leenstra, S. Lightstone, and S. Liu. (2013). "DB2 with BLU Acceleration: So Much More Than Just a Column Store". *Proceedings of the VLDB Endowment*. 6(11): 1080–1091. DOI: 10.14778/2536222.2536233.

Raman, V., G. Swart, L. Qiao, F. Reiss, V. Dialani, D. Kossmann, I. Narang, and R. Sidle. (2008). "Constant-Time Query Processing". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 60–69. DOI: 10.1109/ICDE.2008.4497414.

Rao, J. and K. A. Ross. (2000). "Making B+-trees Cache Conscious in Main Memory". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 475–486. DOI: 10.1145/342009.335449.

Riegger, C., T. Vinçon, R. Gottstein, and I. Petrov. (2020). "MV-PBT: Multi-Version Indexing for Large Datasets and HTAP Workloads". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 217–228. DOI: 10.5441/002/edbt.2020.20.

Riegger, C., T. Vinçon, and I. Petrov. (2017). "Multi-version indexing and modern hardware technologies: a survey of present indexing approaches". In: *Proceedings of the International Conference on Information Integration and Web-based Applications & Services (iiWAS)*. 266–275. DOI: 10.1145/3151759.3151779.

Riegger, C., T. Vinçon, and I. Petrov. (2019). "Indexing large updatable datasets in multi-version database management systems". In: *Proceedings of the Symposium on International Database Engineering & Applications (IDEAS)*. 36:1–36:5. DOI: 10.1145/3331076.3331118.

Robinson, J. T. (1981). "The K-D-B-Tree: A Search Structure For Large Multidimensional Dynamic Indexes". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 10–18. DOI: 10.1145/582318.582321.

Robinson, J. T. (1986). "Order Preserving Linear Hashing Using Dynamic Key Statistics". In: *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 24-26, 1986, Cambridge, Massachusetts, USA*. 91–99. DOI: 10.1145/6012.6014.

Rodeh, O. (2008). "B-trees, shadowing, and clones". *ACM Transactions on Storage*. 3(4): 2:1–2:27. DOI: 10.1145/1326542.1326544.

Roh, H., S. Park, S. Kim, M. Shin, and S.-W. Lee. (2011). "B+-Tree Index Optimization by Exploiting Internal Parallelism of Flash-based Solid State Drives". *Proceedings of the VLDB Endowment*. 5(4): 286–297. URL: http://dl.acm.org/citation.cfm?id=2095686.2095688.

Ross, K. A. (2021). "Utilizing (and Designing) Modern Hardware for Data-Intensive Computations: The Role of Abstraction". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1. DOI: 10.1145/3448016.3460535.

Ross, K. A. (2004). "Selection Conditions in Main Memory". *ACM Transactions on Database Systems (TODS)*. 29: 132–161. DOI: 10.1145/974750.974755.

Sabek, I., K. Vaidya, D. Horn, A. Kipf, M. Mitzenmacher, and T. Kraska. (2022). "Can Learned Models Replace Hash Functions?" *Proceedings of the VLDB Endowment*. 16(3): 532–545. URL: https://www.vldb.org/pvldb/vol16/p532-sabek.pdf.

Sacco, G. M. (1987). "Index Access with a Finite Buffer". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 301–309. URL: http://www.vldb.org/conf/1987/P301.PDF.

Sagiv, Y. (1986). "Concurrent Operations on B*-Trees with Overtaking". *Journal of Computer and System Sciences (JCSS)*. 33(2): 275–296. DOI: 10.1016/0022-0000(86)90021-8.

Sarkar, S. and M. Athanassoulis. (2022). "Dissecting, Designing, and Optimizing LSM-based Data Stores". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2489–2497. DOI: 10.1145/3514221.3522563.

Sarkar, S., K. Chen, Z. Zhu, and M. Athanassoulis. (2022). "Compactionary: A Dictionary for LSM Compactions". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2429–2432. DOI: 10.1145/3514221.3520169.

Sarkar, S., N. Dayan, and M. Athanassoulis. (2023). “The LSM Design Space and its Read Optimizations”. In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*.

Sarkar, S., T. I. Papon, D. Staratzis, and M. Athanassoulis. (2020). “Lethe: A Tunable Delete-Aware LSM Engine”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 893–908. DOI: 10.1145/3318464.3389757.

Sarkar, S., D. Staratzis, Z. Zhu, and M. Athanassoulis. (2021). “Constructing and Analyzing the LSM Compaction Design Space”. *Proceedings of the VLDB Endowment*. 14(11): 2216–2229. URL: http://vldb.org/pvldb/vol14/p2216-sarkar.pdf.

Schlegel, B., R. Gemulla, and W. Lehner. (2009). “k-ary search on modern processors”. In: *Proceedings of the International Workshop on Data Management on New Hardware (DAMON)*. 52–60. DOI: 10.1145/1565694.1565705.

Scholten, H. W. and M. H. Overmars. (1989). “General Methods for Adding Range Restrictions to Decomposable Searching Problems”. *Journal of Symbolic Computation*. 7(1): 1–10. DOI: 10.1016/S0747-7171(89)80002-1.

Schuhknecht, F. M., J. Dittrich, and L. Linden. (2018). “Adaptive Adaptive Indexing”. In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 665–676. DOI: 10.1109/ICDE.2018.00066.

Sears, R. and R. Ramakrishnan. (2012). “bLSM: A General Purpose Log Structured Merge Tree”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 217–228. DOI: 10.1145/2213836.2213862.

Sedgewick, R. (1983). “Balanced Trees”. In: *Algorithms*. Addison-Wesley.

Selinger, P. G., M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. (1979). “Access Path Selection in a Relational Database Management System”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 23–34. DOI: 10.1145/582095.582099.

Severance, D. G. and G. M. Lohman. (1976). "Differential files: their application to the maintenance of large databases". *ACM Transactions on Database Systems (TODS)*. 1(3): 256–267. URL: http://dl.acm.org/citation.cfm?id=320473.320484.

Shasha, D. E. and N. Goodman. (1988). "Concurrent Search Structure Algorithms". *ACM Transactions on Database Systems (TODS)*. 13(1): 53–90. DOI: 10.1145/42201.42204.

Sheehy, J. and D. Smith. (2010). "Bitcask: A Log-Structured Hash Table for Fast Key/Value Data". *Basho White Paper*.

Shehabi, A., S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner. (2016). "United States Data Center Energy Usage Report". *Ernest Orlando Lawrence Berkeley National Laboratory*. LBNL-10057. URL: https://eta.lbl.gov/publications/united-states-data-center-energy.

Sidirourgos, L. and M. L. Kersten. (2013). "Column Imprints: A Secondary Index Structure". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 893–904. URL: http://dl.acm.org/citation.cfm?id=2463676.2465306.

Sleator, D. D. and R. E. Tarjan. (1985). "Self-Adjusting Binary Search Trees". *Journal of the ACM*. 32(3): 652–686. DOI: 10.1145/3828.3835.

Smith, A. J. (1978). "Sequentiality and Prefetching in Database Systems". *ACM Trans. Database Syst.* 3(3): 223–247. DOI: 10.1145/320263.320276.

Spectra. (2017). "Digital Data Storage Outlook". *Spectra White Paper*.

Stonebraker, M. (1981). "Operating System Support for Database Management". *Communications of the ACM*. 24(7): 412–418. DOI: 10.1145/358699.358703.

Stonebraker, M., D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. R. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. Zdonik. (2005). "C-Store: A Column-oriented DBMS". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 553–564. URL: http://dl.acm.org/citation.cfm?id=1083592.1083658.

Sweeney, A., D. Doucette, W. Hu, C. Anderson, M. Nishimoto, and G. Peck. (1996). "Scalability in the XFS File System". In: *Proceedings of the USENIX Annual Technical Conference (ATC)*. 1–14. URL: http://www.usenix.org/publications/library/proceedings/sd96/sweeney.html.

Tarjan, R. E. (1978). "Complexity of Combinatorial Algorithms". *SIAM Review*. 20(3): 457–491. DOI: 10.1137/1020067.

Teixeira, E. M., P. R. P. Amora, and J. C. Machado. (2018). "MetisIDX - From Adaptive to Predictive Data Indexing". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 485–488. DOI: 10.5441/002/edbt.2018.53.

Thonangi, R., S. Babu, and J. Yang. (2012). "A Practical Concurrent Index for Solid-State Drives". In: *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*. 1332–1341. DOI: 10.1145/2396761.2398437.

Unterbrunner, P., G. Giannikis, G. Alonso, D. Fauser, and D. Kossmann. (2009). "Predictable Performance for Unpredictable Workloads". *Proceedings of the VLDB Endowment*. 2(1): 706–717. URL: http://dl.acm.org/citation.cfm?id=1687627.1687707.

Van Sandt, P., Y. Chronis, and J. M. Patel. (2019). "Efficiently Searching In-Memory Sorted Arrays: Revenge of the Interpolation Search?" In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 36–53. DOI: 10.1145/3299869.3300075.

Varman, P. J. and R. M. Verma. (1997). "An Efficient Multiversion Access STructure". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 9(3): 391–409. DOI: 10.1109/69.599929.

Viglas, S. D. (2015). "Data Management in Non-Volatile Memory". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1707–1711. DOI: 10.1145/2723372.2731082.

Vitter, J. S. (2001). "External Memory Algorithms and Data Structures". *ACM Computing Surveys*. 33(2): 209–271. DOI: 10.1145/384192.384193.

Wasay, A., S. Chatterjee, and S. Idreos. (2021). "Deep Learning: Systems and Responsibility". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2867–2875. DOI: 10.1145/3448016.3457541.

Wasay, A., B. Hentschel, Y. Liao, S. Chen, and S. Idreos. (2020). "MotherNets: Rapid Deep Ensemble Learning". In: *Proceedings of Machine Learning and Systems (MLSys)*. URL: https://proceedings.mlsys.org/book/301.pdf.

Wasay, A., X. Wei, N. Dayan, and S. Idreos. (2017). "Data Canopy: Accelerating Exploratory Statistical Analysis". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 557–572. DOI: 10.1145/3035918.3064051.

Wei, Z., K. Yi, and Q. Zhang. (2009). "Dynamic external hashing: the limit of buffering". In: *Proceedings of the Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 253–259. DOI: 10.1145/1583991.1584055.

Weiner, P. (1973). "Linear Pattern Matching Algorithms". In: *Proceedings of the Annual Symposium on Switching and Automata Theory (SWAT)*. 1–11. DOI: 10.1109/SWAT.1973.13.

Willhalm, T., N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. (2009). "SIMD-Scan: Ultra Fast in-Memory Table Scan using on-Chip Vector Processing Units". *Proceedings of the VLDB Endowment*. 2(1): 385–394. URL: http://www.vldb.org/pvldb/2/vldb09-327.pdf.

Wilson, P. R. (1991). "Pointer swizzling at page fault time: efficiently supporting huge address spaces on standard hardware". *SIGARCH Computer Architecture News*. 19(4): 6–13. DOI: 10.1145/122576.122577.

Wong, C. K. and M. C. Easton. (1980). "An Efficient Method for Weighted Sampling Without Replacement". *SIAM Journal on Computing (SICOMP)*. 9(1): 111–113. DOI: 10.1137/0209009.

Wu, K., E. J. Otoo, and A. Shoshani. (2006). "Optimizing Bitmap Indices with Efficient Compression". *ACM Transactions on Database Systems (TODS)*. 31(1): 1–38. DOI: 10.1145/1132863.1132864.

Wu, X., Y. Xu, Z. Shao, and S. Jiang. (2015). "LSM-trie: An LSM-tree-based Ultra-Large Key-Value Store for Small Data Items". In: *Proceedings of the USENIX Annual Technical Conference (ATC)*. 71–82. URL: https://www.usenix.org/conference/atc15/technical-session/presentation/wu.

Wulf, W. A. and S. A. McKee. (1995). "Hitting the Memory Wall: Implications of the Obvious". *ACM SIGARCH Computer Architecture News*. 23(1): 20–24. DOI: 10.1145/216585.216588.

Yao, A. C.-C. and F. F. Yao. (1976). "The Complexity of Searching an Ordered Random Table (Extended Abstract)". In: *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 173–177. DOI: 10.1109/SFCS.1976.32.

Yi, K. (2009). "Dynamic indexability and lower bounds for dynamic one-dimensional range query indexes". In: *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*. 187–196. DOI: 10.1145/1559795.1559825.

Yi, K. (2012). "Dynamic Indexability and the Optimality of B-Trees". *Journal of the ACM*. 59(4): 21:1–21:19. DOI: 10.1145/2339123.2339129.

Yuan, Z., Y. Sun, and D. Shasha. (2023). "Forgetful Forests: Data Structures for Machine Learning on Streaming Data under Concept Drift". *Algorithms*. 16(6). DOI: 10.3390/a16060278.

Zhang, H., G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang. (2015). "In-Memory Big Data Management and Processing: A Survey". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 27(7): 1920–1948. DOI: 10.1109/TKDE.2015.2427795.

Zhang, H., D. G. Andersen, A. Pavlo, M. Kaminsky, L. Ma, and R. Shen. (2016). "Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1567–1581. DOI: 10.1145/2882903.2915222.

Zhou, J. and K. A. Ross. (2004). "Buffering Database Operations for Enhanced Instruction Cache Performance". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 191–202. DOI: 10.1145/1007568.1007592.

Zoumpatianos, K. and T. Palpanas. (2018). "Data Series Management: Fulfilling the Need for Big Sequence Analytics". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 1677–1678. DOI: 10.1109/ICDE.2018.00211.

Zukowski, M. and P. A. Boncz. (2012). "Vectorwise: Beyond Column Stores". *IEEE Data Engineering Bulletin*. 35(1): 21–27. URL: http://sites.computer.org/debull/A12mar/vectorwise.pdf.

Zukowski, M., S. Héman, N. J. Nes, and P. A. Boncz. (2007). "Cooperative Scans: Dynamic Bandwidth Sharing in a DBMS". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 723–734. URL: http://dl.acm.org/citation.cfm?id=1325851.1325934.