More Modern B-Tree Techniques

Other titles in Foundations and Trends® in Databases

Algorithmic Aspects of Parallel Data Processing
Paraschos Koutris, Semih Salihoglu and Dan Suciu
ISBN: 978-1-68083-406-2

Data Infrastructure for Medical Research
Thomas Heinis and Anastasia Ailamaki
ISBN: 978-1-68083-348-5

Main Memory Database Systems
Franz Faerber, Alfons Kemper, Per-Ake Larson, Justin Levandoski, Thomas Neumann and Andrew Pavlo
ISBN: 978-1-68083-324-9

Query Processing on Probabilistic Data: A Survey
Guy Van den Broeck and Dan Suciu
ISBN: 978-1-68083-314-0

Big Graph Analytics Platforms
Da Yan, Yingyi Bu, Yuanyuan Tian and Amol Deshpande
978-1-68083-242-6

More Modern B-Tree Techniques

Goetz Graefe
Google Inc.
GoetzG@google.com

now
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

G. Graefe. *More Modern B-Tree Techniques*. Foundations and Trends® in Databases, vol. 13, no. 3, pp. 169–249, 2024.

ISBN: 978-1-63828-373-7

Foundations and Trends® in Databases
Volume 13, Issue 3, 2024

Editorial Board

Editorial Scope

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, 2024, Volume 13, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Contents

More Modern B-Tree Techniques

Goetz Graefe

Google Inc., USA; GoetzG@google.com

ABSTRACT

An earlier survey of modern b-tree techniques is now over a decade old. Obviously, it lacks descriptions of techniques invented and published during this time. Just as importantly, it lacks descriptions of insertion-optimized b-trees in the forms of log-structured merge-trees and stepped-merge forests, which seem to have become almost as ubiquitous as b-trees themselves. This monograph complements the earlier survey in order to bring the combined contents up-to-date.

Goetz Graefe (2024), "More Modern B-Tree Techniques", Foundations and Trends® in Databases: Vol. 13, No. 3, pp 169–249. DOI: 10.1561/1900000070.

1

Introduction

B-tree indexes have been ubiquitous in databases for decades [11]. Their contribution to efficient database query processing can hardly be overstated. An earlier survey of modern b-tree techniques [29] has been widely used for education and reference. Unfortunately, it omits some of the techniques known then and of course those invented since. This monograph is intended to complement this earlier survey and to bring the combined contents more up-to-date.

Section 2 summarizes some of the highlights from this earlier survey [29]. It is intended as a motivation for reading the earlier survey, not as a substitute. Section 3 adds new techniques for tree structures. After a short discussion on in-page formats, a common foundation for further techniques is a reversal from B^{link}-trees and from multiple pointers to each b-tree node. A single pointer to each b-tree node enables write-optimized b-tree, pointer swizzling in a database buffer pool, foster b-trees, self-repairing b-trees, and more. This section also discusses deferred updates and deferred index optimization, i.e., it separates delayed maintenance of index contents and index structure.

Section 4 reviews log-structured merge-forests and stepped-merge forests, topics omitted from the earlier survey [29]. For those, it proposes new storage structures, even b-trees without branch nodes, which probably seems like a contradiction, and new algorithms, including a rather unconventional schedule for merging runs in an external merge sort or for compacting deltas in a log-structured merge-forest.

Section 5 adds a few new techniques in query processing. This includes better use of b-trees on storage and in memory. During index intersection and join, the secondary sort keys enable very efficient merge algorithms without explicit sorting.

Section 6 focuses on concurrency control. The topics include new granularities of locking for fewer invocations of the lock manager and for fewer false conflicts for transactions in serializable transaction isolation – the new technique is called orthogonal key-value locking. They further include shorter enforcement periods of locks, both during execution of the transaction's application logic and during commit processing – the new techniques are called deferred lock enforcement and controlled lock violation. They are not specific to b-trees but the combination of orthogonal key-value locking, deferred lock enforcement, controlled lock violation, and multi-version storage promises to eliminate practically all false conflicts in database concurrency control.

Section 7 covers in-memory b-trees, from thread-private via shared-but-transient to persistent. Among the core considerations are the requirements for concurrency control, logging, and recovery. For in-memory databases and their b-trees, instant reboot combines the techniques of instant restart and instant restore, in effect applying the logic for a double failure in a traditional database with external storage. The final Section 8 sums up and concludes.

References

[1] S. Agarwal, J. A. Blakeley, T. Casey, K. Delaney, C. A. Galindo-Legaria, G. Graefe, M. Rys, and M. J. Zwilling, *Microsoft SQL server (Chapter 27),* in *Database System Concepts*, 4th edition. 2001, pp. 969–1006.

[2] A. Ailamaki, D. J. DeWitt, M. D. Hill, and M. Skounakis, "Weaving relations for cache performance," *VLDB Conference*, 2001, pp. 169–180.

[3] G. Antoshenkov, "Dictionary-based order-preserving string compression," *The VLDB Journal*, vol. 6, no. 1, 1997, pp. 26–39.

[4] D. F. Bacon *et al.*, "Spanner: Becoming a SQL system," *ACM SIGMOD Conference*, 2017, pp. 331–343.

[5] R. Bayer and K. Unterauer, "Prefix b-trees," *ACM TODS*, vol. 2, no. 1, 1977, pp. 11–26.

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

[7] D. Bitton and D. J. DeWitt, "Duplicate record elimination in large data files," *ACM TODS*, vol. 8, no. 2, 1983, pp. 255–265.

[8] M. J. Carey, D. J. DeWitt, M. J. Franklin, N. E. Hall, M. L. McAuliffe, J. F. Naughton, D. T. Schuh, M. H. Solomon, C. K. Tan, O. G. Tsatalos, S. J. White, and M. J. Zwilling, "Shoring up persistent applications," *ACM SIGMOD Conference*, 1994, pp. 383–394.

[9] A. Chan, S. Fox, W.-T. K. Lin, A. Nori, and D. R. Ries, "The implementation of an integrated concurrency control and recovery scheme," *ACM SIGMOD Conference*, 1982, pp. 184–191.

[10] S. Chen, P. B. Gibbons, T. C. Mowry, and G. Valentin, "Fractal prefetching b-trees: Optimizing both cache and disk performance," *ACM SIGMOD Conference*, 2002, pp. 157–168.

[11] D. Comer, "The ubiquitous b-tree," *ACM Computing Surveys*, vol. 11, no. 2, 1979, pp. 121–137.

[12] W. M. Conner, "Offset value coding," *IBM Technical Disclosure Bulletin*, vol. 20, no. 7, 1977, pp. 2832–2837.

[13] J. C. Corbett *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems*, vol. 31, no. 3, 2013, 8:1–8:22.

[14] N. Dayan, M. Athanassoulis, and S. Idreos, "Monkey: Optimal navigable key-value store," *ACM SIGMOD Conference*, 2017, pp. 79–94.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *OSDI*, 2004, pp. 137–150.

[16] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," *CACM*, vol. 53, no. 1, 2010, pp. 72–77.

[17] D. J. DeWitt, R. H. Katz, F. Olken, L. D. Shapiro, M. Stonebraker, and D. A. Wood, "Implementation techniques for main memory database systems," *ACM SIGMOD*, 1984, pp. 1–8.

[18] T. Do and G. Graefe, "Robust and efficient sorting with offset-value coding," *ACM TODS*, vol. 48, no. 1, 2023, 2:1–2:23.

[19] T. Do, G. Graefe, and J. Naughton, "Efficient sorting, duplicate removal, grouping, and aggregation," *ACM TODS*, vol. 47, no. 4, 2022, 16:1–16:35.

[20] Enterprise system architecture/370, principles of operation. IBM publication SA22-7200-0, August 1988. CFC "compare and form codeword" and UPT "update tree" instructions.

[21] E. Fredkin, "Trie memory," *Communications of the ACM*, vol. 3, no. 9, 1960, pp. 490–499.

[22] D. Gawlick and D. Kinkade, "Varieties of concurrency control in IMS/VS fast path," *IEEE Data Engineering Bulletin*, vol. 8, no. 2, 1985, pp. 3–10.

[23] N. Glombiewski, B. Seeger, and G. Graefe, "Waves of misery after index creation," *BTW Conference*, 2019, pp. 77–96.

[24] N. Glombiewski, B. Seeger, and G. Graefe, "Continuous merging: Avoiding waves of misery in tiered log-structured merge trees, unpublished," 2021.

[25] G. Graefe, *Sorting and Indexing with Partitioned B-Trees*. CIDR, 2003.

[26] G. Graefe, "Write-optimized b-trees," *VLDB Conference*, 2004, pp. 672–683.

[27] G. Graefe, "B-tree indexes, interpolation search, and skew," *DaMoN*, 2006, p. 5.

[28] G. Graefe, "Efficient columnar storage in b-trees," *SIGMOD Record*, vol. 36, no. 1, 2007, pp. 3–6.

[29] G. Graefe, "Modern b-tree techniques," *Foundations and Trends in Databases*, vol. 3, no. 4, 2011, pp. 203–402.

[30] G. Graefe, "Avoiding index-navigation deadlocks," in *Synthesis Lectures on Data Management*, Morgan & Claypool Publishers, 2019.

[31] G. Graefe, "Deferred lock enforcement," in *Synthesis Lectures on Data Management*, Morgan & Claypool Publishers, 2019.

[32] G. Graefe, "Orthogonal key-value locking," in *Synthesis Lectures on Data Management,* Morgan & Claypool Publishers, Extended from Goetz Graefe, Hideaki Kimura: Orthogonal Key-Value Locking, *BTW Conference*, pp. 237–256.

[33] G. Graefe and T. Do, "Offset-value coding in database query processing," *EDBT Conference*, 2023, pp. 464–470.

[34] G. Graefe, W. Guy, and C. Sauer, "Instant recovery with write-ahead logging: Page repair, system restart, media restore, and system failover," in *Synthesis Lectures on Data Management*, 2nd edition, Morgan & Claypool Publishers, 2016, pp. 1–113.

[35] G. Graefe, H. Kimura, and H. A. Kuno, "Foster b-trees," *ACM ToDS*, vol. 37, no. 3, 2012, 17:1–17:29.

[36] G. Graefe and H. A. Kuno, "Fast loads and fast queries," *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, vol. 2, 2010, pp. 31–72.

[37] G. Graefe and H. A. Kuno, "Self-selecting, self-tuning, incrementally optimized indexes," *EDBT Conference*, 2010, pp. 371–381.

[38] G. Graefe and H. A. Kuno, "Definition, detection, and recovery of single-page failures, a fourth class of database failures," *PVLDB*, vol. 5, no. 7, 2012, pp. 646–655.

[39] G. Graefe, H. A. Kuno, and B. Seeger, "Self-diagnosing and self-healing indexes," *DBTest*, 2012, p. 8.

[40] G. Graefe, M. Lillibridge, H. A. Kuno, J. Tucek, and A. C. Veitch, "Controlled lock violation," *ACM SIGMOD Conference*, 2013, 85–96. Also in *Synthesis Lectures on Data Management*, Morgan & Claypool Publishers.

[41] G. Graefe, I. Petrov, T. Ivanov, and V. Marinov, "A hybrid page layout integrating PAX and NSM," *IDEAS*, 2013, pp. 86–95.

[42] G. Graefe, H. Volos, H. Kimura, H. A. Kuno, J. Tucek, M. Lillibridge, and A. C. Veitch, "In-memory performance for big data," *PVLDB*, vol. 8, no. 1, 2014, pp. 37–48.

[43] J. Gray and A. Reuter, *Transaction Processing Concepts and Techniques*. Morgan Kaufmann, 1993.

[44] T. Härder, "Observations on optimistic concurrency control schemes," *Information Systems*, vol. 9, no. 2, 1984, pp. 111–120.

[45] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker, "OLTP through the looking glass, and what we found there," *ACM SIGMOD Conference*, 2008, pp. 981–992.

[46] B. R. Iyer, "Hardware-assisted sorting in IBM's DB2 DBMS," in *COMAD Conference*, Hyderabad, 2005.

[47] H. V. Jagadish, P. P. S. Narayan, S. Seshadri, S. Sudarshan, and R. Kanneganti, "Incremental organization for data recording and warehousing," *VLDB Conference*, 1997, pp. 16–25.

[48] J. R. Jordan, J. Banerjee, and R. B. Batman, "Precision locks," *ACM SIGMOD Conference*, 1981, pp. 143–147.

[49] A. Kipf, R. Marcus, A. van Renen, M. Stoian, A. Kemper, T. Kraska, and T. Neumann, "RadixSpline: A single-pass learned index," *aiDM@SIGMOD*, 2020, 5:1–5:5.

[50] T. Kraska, A. Beutel, E. H. Chi, and J. Dean, "The case for learned index structures," in *ACM SIGMOD Conference*, 2018, pp. 489–504.

[51] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM ToDS*, vol. 6, no. 2, 1981, pp. 221–226.

[52] P. L. Lehman and S. B. Yao, "Efficient locking for concurrent operations on b-trees," *ACM ToDS*, vol. 6, no. 4, 1981, pp. 650–670.

[53] H. Leslie, R. Jain, D. Birdsall, and H. Yaghmai, "Efficient search of multi-dimensional b-trees," *VLDB Conference*, 1995, pp. 710–719.

[54] D. B. Lomet, "Key range locking strategies for improved concurrency," *VLDB Conference*, 1993, pp. 655–664.

[55] D. B. Lomet, "The evolution of effective b-tree: Page organization and techniques: A personal account," *ACM SIGMOD Record*, vol. 30, no. 3, 2001, pp. 64–69.

[56] C. Luo and M. J. Carey, "LSM-based storage techniques: A survey," *The VLDB Journal*, vol. 29, no. 1, 2020, pp. 393–418.

[57] Y. Mao, E. Kohler, and R. T. Morris, "Cache craftiness for fast multicore key-value storage," *EuroSys*, 2012, pp. 183–196.

[58] C. Mohan, "ARIES/KVL: A key-value locking method for concurrency control of multiaction transactions operating on b-tree indexes," *VLDB Conference*, 1990, pp. 392–405.

[59] C. Mohan and F. E. Levine, "ARIES/IM: An efficient and high concurrency index management method using write-ahead logging," *ACM SIGMOD Conf.*, 1992, pp. 371–380.

[60] T. Neumann, T. Mühlbauer, and A. Kemper, "Fast serializable multi-version concurrency control for main-memory database systems," *ACM SIGMOD Conference*, 2015, pp. 677–689.

[61] C. Nyberg, T. Barclay, Z. Cvetanovic, J. Gray, and D. B. Lomet, "AlphaSort: A cache-sensitive parallel external sort," *The VLDB Journal*, vol. 4, no. 4, 1995, pp. 603–627.

[62] P. E. O'Neil, "The SB-tree: An index-sequential structure for high-performance sequential access," *Acta Informatica*, vol. 29, no. 3, 1992, pp. 241–265.

[63] P. E. O'Neil, E. Cheng, D. Gawlick, and E. J. O'Neil, "The log-structured merge-tree (LSM-tree)," *Acta Informatica*, vol. 33, no. 4, 1996, pp. 351–385.

[64] D. A. Patterson, G. A. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *ACM SIGMOD Conference*, 1988, pp. 109–116.

[65] P. V. Sandt, Y. Chronis, and J. M. Patel, "Efficiently searching in-memory sorted arrays: Revenge of the interpolation search?" *ACM SIGMOD Conference*, 2019, pp. 36–53.

[66] C. Sauer, G. Graefe, and T. Härder, "Single-pass restore after a media failure," *BTW*, 2015, pp. 217–236.

[67] R. Sears and R. Ramakrishnan, "bLSM: A general purpose log-structured merge tree," *ACM SIGMOD Conference*, 2012, pp. 217–228.

[68] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," *ACM SIGMOD Conf.*, 1979, pp. 23–34.

[69] D. G. Severance and G. M. Lohman, "Differential files: Their application to the maintenance of large databases," *ACM Transactions on Database Systems*, vol. 1, no. 3, 1976, pp. 256–267.

[70] A. Verbitski, A. Gupta, D. Saha, M. Brahmadesam, K. Gupta, R. Mittal, S. Krishnamurthy, S. Maurice, T. Kharatishvili, and X. Bao, "Amazon Aurora: Design considerations for high throughput cloud-native relational databases," *ACM SIGMOD Conference*, 2017, pp. 1041–1052.

[71] G. Weikum and G. Vossen, *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann, 2002.

[72] Y. Wu, J. Yu, Y. Tian, R. Sidle, and R. Barber, "Designing a succinct secondary indexing mechanism by exploiting column correlations," *ACM SIGMOD Conference*, 2019, pp. 1223–1240.

[73] H. Zhang, X. Liu, D. G. Andersen, M. Kaminsky, K. Keeton, and A. Pavlo, "Order-preserving key compression for in-memory search trees," *ACM SIGMOD Conference*, 2020, pp. 1601–1615.