Benchmarking of multi-dimensional in-memory index structures

— Exposé für eine Masterarbeit —

Lukas Schramm
Betreuer: Prof. Dr. Ulf Leser

---

1 Motivation

Die moderne technologische Welt besteht aus immer mehr Anwendungen, deren Daten in großen Datenbanken organisiert werden müssen. Ein Hauptproblem solcher Anwendungen ist es, Daten schnell verfügbar zu machen, also die Reaktionszeit für Abfragen durch bessere Indexstrukturen zu verkleinern. Insbesondere Bereichsabfragen (eng. *Range Queries*) sind hierbei problematisch, da das Ergebnis gegebenenfalls eine größere Menge von Daten umfasst, die potentiell über eine ganze Datenstruktur fragmentiert sein können[Ho+97].

Dabei spielt die Dimensionalität der Daten eine entscheidende Rolle. Geographische Koordinaten (engl. *spatial data*) mit zwei Dimensionen sind potentiell anders zu behandeln als Daten mit sehr vielen Dimensionen. Bereits im vorherigen Jahrhundert entstanden daher spezielle Datenstrukturen zur Organisation von mehrdimensionalen Daten in Datenbanken wie zum Beispiel R-Trees[Gut84], R*-Trees[Bec+90], KD-Trees[Ben75], VA-files[WSB98] oder UB-Trees[Bay97]. Der Fokus dieser Masterarbeit ist die systematische Analyse moderner Ansätze des vergangenen Jahrzehnts. Die modernen Datenstrukturen können grob in drei verschiedene Kategorien unterteilt werden:

- Traditionelle Indexstrukturen: Die gesamte Indexstruktur über alle Daten ist vor der ersten Abfrage bereits vorhanden, Aktualisierung nur bei Datenänderungen
- Adaptive Indexstrukturen: Die Indexstruktur ist zu Beginn leer und wird schrittweise mit jeder neuen Abfrage aufgebaut, sie passt sich also an die Anfragen an
- Maschinelles Lernen: Indexe werden als mathematische Modelle verstanden, in denen man Positionen von Schlüsseln vorhersagen kann[Din+20]

Die hierfür gewählten Indexstrukturen sind zwei traditionelle, drei adaptive sowie eine auf maschinellem Lernen basierende Indexstruktur. Dies sind der BB-Tree[SSL19; Spr19] und Elf[Bro19], Cracking KD[Hol+18], QUASII[Pav+18] und Adaptive Adaptive Indexing[SDL18] sowie ALEX[Din+20]. Andere Forschende haben den Programmcode ihrer Indexstrukturen weder öffentlich noch auf Nachfrage zur Verfügung gestellt.[1]

2 Zielstellung

Das Ziel der Arbeit ist es, eine gemeinsame Grundlage für eine Evaluation der Indexstrukturen zu definieren und diese durchzuführen. Als vielversprechend gelten hierbei fünf Datensätze, die teilweise für einzelne spezifische Indexstrukturen entwickelt wurden. Diese sind im Folgenden:

[1] ML-Index[DMM20], ParISSS[EDE21], Flood[Nat+20] und Hashed B+Tree[CZA21]

- Genomic Multidimensional Range Query Benchmark: genomische Daten aus der realen Welt aus dem *1000 Genomes Project*[2] aus [SSL18] (verwendet unter anderem für [SSL19; Spr19])
- Skyserver, Power, Genomics: aus [Hol+18][3]
- Synthetische Daten, geordnet nach acht verschiedenen Workloads: *Uniform, Clustered, Skewed, Sequential, Periodic, Zoom, Sequential Zoom, Zoom Alternating* (genutzt in [Hol+18])
- TPC-H[4], genutzt in [Köp+15]
- TPC-DS[5], eine Weiterentwicklung des TPC-H-Standards

Hieraus ergeben sich vier spezifische Fragen, die es zu beantworten gilt:

1. Wie gut funktionieren die Indexstrukturen verglichen zueinander?
2. Gibt es Unterschiede in Abhängigkeit zu bestimmten Workloads?
3. Gibt es erkennbare Unterschiede in Abhängigkeit vom Typ der Indexstruktur (traditionell, adaptiv, maschinelles Lernen) und wie hängt diese von anderen Faktoren wie der Größe des Datensatzes ab?
4. Wie wird Effizienz definiert und wie vergleicht man Indexstrukturen ideal?

Wichtig zu beachten ist insbesondere, dass alle Kosten beachtet werden und auch Preprocessingvorgänge mit einbezogen werden[6]. Durch eine verschieden große Skalierung der Größen von Workloads wird experimentell ermittelt, bei welcher Menge von Daten welche Indexstrukturen bessere Antwortzeiten erreichen. Es ist zum Beispiel zu erwarten, dass adaptive Indexstrukturen, die ihren Index während der Anfragen zusammensetzen bei weniger Abfragen schneller sind, da die Gesamtzeit des vollständigen Indexaufbaus der klassischen Indexstrukturen dessen Reaktionszeit übersteigt. Auf eine unendliche Datenmenge extrapolieren sieht dies aber potentiell anders herum aus.

3 Forschungsstand

Im vorhergehenden Kapitel 1 wurden Indexstrukturen vorgestellt, die in dieser Masterarbeit verwendet werden. Bereits zur Jahrtausendwende gab es Vergleichsberichte zu den oben genannten älteren Indexstrukturen[GG98; BBK01]. Des weiteren gab es viele wissenschaftliche Abhandlungen zu spezifischen Teilgebieten, zum Beispiel zeigte Oracle in [Das+15], dass Full Scans häufig schneller sind als komplexe ältere Varianten. Auch Vergleiche, die sich auf spezifische Typen Datenstrukturen beziehen, wurden zahlreich durchgeführt, wie zum Beispiel für Hashing-Verfahren[Wan+15], R-Tree-Varianten[SK18] sowie Radixbäume und Hashtabellen[Alv+15].

[2] https://www.1000genomes.org/
[3] https://github.com/pdet/MultidimensionalAdaptiveIndexing
[4] https://www.tpc.org/tpch/
[5] https://www.tpc.org/tpcds/
[6] Das vorherige Preprocessing von vorhandenen Daten wird zum Beispiel in [Köp+15] vollständig ignoriert.

Auch die Paper, welche die sechs von uns behandelten Indexstrukturen präsentierten, vergleichen sich teilweise direkt mit anderen Datenstrukturen. So gibt zwei Paper, in denen der Cracking KD Tree[Hol+18] direkt mit QUASII[Pav+18] verglichen wird. Ein neueres Projekt des Hasso-Plattner-Instituts versucht sich auch an der Erschaffung eines Vergleichsframeworks anhand des TPC-H-Benchmarks, dieses enthält jedoch von den oben genannten Indexstrukturen nur den BB-Tree[Wei21; Dre+19]. In [Blo+20] wird Elf anhand der TPC-H und GMRQB-Benchmarks getestet und [Jen+21] ist ein weiteres Tool, das Cracking KD und QUASII miteinander vergleicht. Letzteres wirft leider bei der Ausführung Fehler aus, es ist daher fraglich, in wie weit man das reparieren und brauchbar nutzen kann.

Als theoretische Grundlagen für den Vergleich von Indexstrukturen gelten Metriken, welche in [Hel+02; HKP97] sowie [Ner+21] definiert sind.

4 Vorgehen

Die Implementationen der oben genannten Indexstrukturen BB-Tree[SSL19; Spr19], Elf[Bro19], Cracking KD[Hol+18] und QUASII[Pav+18] (sowie potentiell weitere maschinelle Lernindexstrukturen) werden kompiliert und auf einem Referenzrechner auf die verschiedenen Benchmarks getestet. Der Fokus hierbei liegt auf Leseoperationen, insbesondere Bereichsabfragen, da andere Operationen wie Lesen- und Schreiben nicht von allen Indexstrukturen unterstützt werden und viele Benchmarks nur Select-Abfragen enthalten. Ferner sind viele Indexstrukturen explizit für OLAP-Systeme entwickelt worden, deren Fokus auf Bulkloads mit anschließenden Leseoperationen liegt[Ho+97; Hol+18]. Alle Indexstrukturen wurden in C++ implementiert und werden mit dem gleichen Compiler auf dem gleichen Betriebssystem getestet. Die Experimente werden mehrfach durchgeführt und Durchschnittswerte sowie Varianzen angegeben.

5 Bibliography

Literatur

[Alv+15] V. Alvarez, S. Richter, X. Chen und J. Dittrich. „A comparison of adaptive radix trees and hash tables“. In: *2015 IEEE 31st International Conference on Data Engineering*. Apr. 2015, S. 1227–1238.

[Bay97] R. Bayer. „The universal B-tree for multidimensional indexing: General concepts“. In: *International Conference on Worldwide Computing and Its Applications*. Springer. 1997, S. 198–209.

[BBK01] C. Böhm, S. Berchtold und D. A. Keim. „Searching in High-Dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases“. In: *ACM Comput. Surv.* 33.3 (Sep. 2001), S. 322–373.

[Bec+90] N. Beckmann, H.-P. Kriegel, R. Schneider und B. Seeger. „The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles“. In: *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*. SIGMOD '90. Atlantic City, New Jersey, USA: Association for Computing Machinery, 1990, S. 322–331.

[Ben75] J. L. Bentley. „Multidimensional Binary Search Trees Used for Associative Searching“. In: *Commun. ACM* 18.9 (Sep. 1975), S. 509–517.

[Blo+20] P. Blockhaus, D. Broneske, M. Schäler, V. Köppen und G. Saake. „Combining Two Worlds: MonetDB with Multi-Dimensional Index Structure Support to Efficiently Query Scientific Data“. In: *32nd International Conference on Scientific and Statistical Database Management*. SSDBM 2020. Vienna, Austria: Association for Computing Machinery, 2020.

[Bro19] D. Broneske. „Accelerating mono and multi-column selection predicates in modern main-memory database systems“. Diss. Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, 2019.

[CZA21] S. Chen, G. Zhou und X. An. „An efficient indexing structure for multi-dimensional range query“. In: *Frontiers of Computer Science* 15.4, 154612 (2021), S. 154612.

[Das+15] D. Das, J. Yan, M. Zait, S. R. Valluri, N. Vyas, R. Krishnamachari, P. Gaharwar, J. Kamp und N. Mukherjee. „Query Optimization in Oracle 12c Database In-Memory“. In: *Proc. VLDB Endow.* 8.12 (Aug. 2015), S. 1770–1781.

[Din+20] J. Ding, U. F. Minhas, J. Yu, C. Wang, J. Do, Y. Li, H. Zhang, B. Chandramouli, J. Gehrke, D. Kossmann u. a. „ALEX: An Updatable Adaptive Learned Index“. In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. Portland, OR, USA: Association for Computing Machinery, 2020, S. 969–984.

[DMM20] A. Davitkova, E. Milchevski und S. Michel. „The ML-Index: A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries“. In: *EDBT*. 2020.

[Dre+19] M. Dreseler, J. Kossmann, M. Boissier, S. Klauck, M. Uflacker und H. Plattner. „Hyrise Re-engineered: An Extensible Database System for Research in Relational In-Memory Data Management“. In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. Hrsg. von M. Herschel, H. Galhardas, B. Reinwald, I. Fundulaki, C. Binnig und Z. Kaoudi. OpenProceedings.org, 2019, S. 313–324.

[EDE21] M. Elmeiligy, A. Desouky und S. Elghamrawy. „An efficient parallel indexing structure for multi-dimensional big data using spark“. In: *The Journal of Supercomputing* 77 (Okt. 2021).

[GG98] V. Gaede und O. Günther. „Multidimensional Access Methods“. In: *ACM Comput. Surv.* 30.2 (Juni 1998), S. 170–231.

[Gut84] A. Guttman. „R-Trees: A Dynamic Index Structure for Spatial Searching“. In: *SIGMOD Rec.* 14.2 (Juni 1984), S. 47–57.

[Hel+02] J. M. Hellerstein, E. Koutsoupias, D. Miranker und V. Samoladas. „On a model of indexability and its bounds for range queries“. In: *Journal of the ACM (JACM)* 49 (Jan. 2002), S. 35–55.

[HKP97] J. M. Hellerstein, E. Koutsoupias und C. H. Papadimitriou. „On the Analysis of Indexing Schemes“. In: *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS '97. Tucson, Arizona, USA: Association for Computing Machinery, 1997, S. 249–256.

[Ho+97] C.-T. Ho, R. Agrawal, N. Megiddo und R. Srikant. „Range Queries in OLAP Data Cubes“. In: *SIGMOD Rec.* 26.2 (Juni 1997), S. 73–88.

[Hol+18] P. Holanda, M. Nerone, E. Almeida und S. Manegold. „Cracking KD-Tree: The First Multidimensional Adaptive Indexing (Position Paper)“. In: Jan. 2018, S. 393–399.

[Jen+21] A. Jensen, F. Lauridsen, F. Zardbani, S. Idreos und P. Karras. „Revisiting Multidimensional Adaptive Indexing [Experiment & Analysis]“. In: *EDBT*. 2021.

[Köp+15] V. Köppen, D. Broneske, G. Saake und M. Schäler. „Elf: A main-memory structure for efficient multi-dimensional range and partial match queries“. In: *Otto-von-Guericke-University Magdeburg, Tech. Rep* (2015), S. 002–2015.

[Nat+20] V. Nathan, J. Ding, M. Alizadeh und T. Kraska. „Learning Multi-Dimensional Indexes“. In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM, Mai 2020.

[Ner+21] M. A. Nerone, P. Holanda, E. C. de Almeida und S. Manegold. „Multidimensional Adaptive & Progressive Indexes“. In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. Apr. 2021, S. 624–635.

[Pav+18] M. Pavlovic, D. Sidlauskas, T. Heinis und A. Ailamaki. „QUASII: QUery-Aware Spatial Incremental Index“. In: *EDBT*. 2018.

[SDL18] F. M. Schuhknecht, J. Dittrich und L. Linden. „Adaptive Adaptive Indexing“. In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. Apr. 2018, S. 665–676.

[SK18] O. Sharma und K. Kumar. „Review for Best Multidimensional Index Structure“. In: *IJCRT* 6 (März 2018).

[Spr19] S. Sprenger. „Efficient Processing of Range Queries in Main Memory“. Diss. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät, 2019.

[SSL18] S. Sprenger, P. Schäfer und U. Leser. „Multidimensional range queries on modern hardware“. In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. 2018, S. 1–12.

[SSL19] S. Sprenger, P. Schäfer und U. Leser. „BB-Tree: A Main-Memory Index Structure for Multidimensional Range Queries“. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, S. 1566–1569.

[Wan+15] J. Wang, W. Liu, S. Kumar und S.-F. Chang. „Learning to hash for indexing big data—A survey“. In: *Proceedings of the IEEE* 104.1 (2015), S. 34–57.

[Wei21] M. Weisgut. „Experimental Index Evaluation for Partial Indexes in Horizontally Partitioned In-Memory Databases.“ In: *GvDB*. 2021.

[WSB98] R. Weber, H.-J. Schek und S. Blott. „A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces“. In: *Proceedings of the 24rd International Conference on Very Large Data Bases*. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, S. 194–205.