



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction

[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

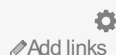
Tools

[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)

Print/export

[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Languages



[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#)

[More](#) ▾

Log-structured merge-tree

From Wikipedia, the free encyclopedia



This article includes a [list of references](#), but **its sources remain unclear** because it has **insufficient inline citations**. Please help to [improve](#) this article by [introducing](#) more precise citations. (*April 2013*)

In [computer science](#), the **log-structured merge-tree** (or LSM tree) is a [data structure](#) with performance characteristics that make it attractive for providing [indexed](#) access to files with high insert volume, such as [transactional log data](#). LSM trees, like other search trees, maintain key-value pairs. LSM trees maintain data in two or more separate structures, each of which is optimized for its respective underlying storage medium; data is synchronized between the two structures efficiently, in batches.

One simple version of the LSM tree is a two-level LSM tree.^[1] As described by O'Neil, a two-level LSM tree comprises two [tree-like](#) structures, called *C*₀ and *C*₁. *C*₀ is smaller and entirely resident in memory, whereas *C*₁ is resident on disk. New records are inserted into the memory-resident *C*₀ component. If the insertion causes the *C*₀ component to exceed a certain size threshold, a contiguous segment of entries is removed from *C*₀ and merged into *C*₁ on disk. The performance characteristics of LSM trees stem from the fact that each component is tuned to the characteristics of its underlying storage medium, and that data is efficiently migrated across media in rolling batches, using an algorithm reminiscent of [merge sort](#).

Most LSM trees used in practice employ multiple levels. Level 0 is kept in main memory, and might be represented using a tree. The on-disk data is organized into sorted runs of data. Each run contains data sorted by the index key. A run can be represented on disk as a single file, or alternatively as a collection of files with non-overlapping key ranges. To perform a query on a particular key to get its associated value, one must search in the Level 0 tree, as well as each run.

A particular key may appear in several runs, and what happens depends on the application. Some applications simply want the newest key-value pair with a given key. Some applications must combine the values in some way to get the proper aggregate value to return. For example, in [Apache Cassandra](#), each value represents a row in a database, and different versions of the row may have different sets of columns.^[2]

In order to keep down the cost of queries, the system must avoid a situation where there are too many runs.

Extensions to the 'levelled' method to incorporate B+ structures have been suggested, for example bLSM^[3] and Diff-Index.^[4]

LSM trees are used in database management systems such as [BigTable](#), [HBase](#), [LevelDB](#), [MongoDB](#), [SQLite4](#), [RocksDB](#), [WiredTiger](#)^[5] and [Apache Cassandra](#).

References [\[edit\]](#)

- ↑ O'Neil 1996, p. 4
- ↑ [Leveled Compaction in Apache Cassandra](#)
- ↑ <http://www.eecs.harvard.edu/~margo/cs165/papers/gp-lsm.pdf>
- ↑ <http://researcher.ibm.com/researcher/files/us-wtan/DiffIndex-EDBT14-CR.pdf>
- ↑ <https://github.com/wiredtiger/wiredtiger/wiki/LSMTrees>

General

- O'Neil, Patrick E.; Cheng, Edward; Gawlick, Dieter; O'Neil, Elizabeth (June 1996). "The log-structured merge-tree (LSM-tree)" . *Acta Informatica* **33** (4): 351–385. doi:10.1007/s002360050048 . Retrieved 2014-08-03.
- Li, Yinan; He, Bingsheng; Luo, Qiong; Yi, Ke (2009). "Tree Indexing on Flash Disks" . *2009 IEEE 25th International Conference on Data Engineering*. pp. 1303–6. doi:10.1109/ICDE.2009.226 . ISBN 978-1-

Log-structured merge-tree

Type Hybrid (two tree-like components)

Invented 1996

Invented by Patrick O'Neil, Edward Cheng, Dieter Gawlick, Elizabeth O'Neil

Time complexity in big O notation

Average Worst case

Space

Search

Insert

Delete

External links [\[edit\]](#)

- [An Overview of Log Structured Merge Trees](#) [↗](#)

v · t · e	Tree data structures [hide]
Search trees (dynamic sets/associative arrays)	2–3 · 2–3–4 · AA · (a,b) · AVL · B · B+ · B* · B ^X · (Optimal) Binary search · Dancing · HTree · Interval · Order statistic · (Left-leaning) Red-black · Scapegoat · Splay · T · Treap · UB · Weight-balanced
Heaps	Binary · Binomial · Fibonacci · Leftist · Pairing · Skew · Van Emde Boas
Tries	Hash · Radix · Suffix · Ternary search · X-fast · Y-fast
Spatial data partitioning trees	BK · BSP · Cartesian · Hilbert R · <i>k</i> -d (implicit <i>k</i> -d) · M · Metric · MIP · Octree · Priority R · Quad · R · R+ · R* · Segment · VP · X
Other trees	Cover · Exponential · Fenwick · Finger · Fusion · Hash calendar · iDistance · K-ary · Left-child right-sibling · Link/cut · Log-structured merge · Merkle · PQ · Range · SPQR · Top

Categories: [Trees \(data structures\)](#) | [Database index techniques](#)