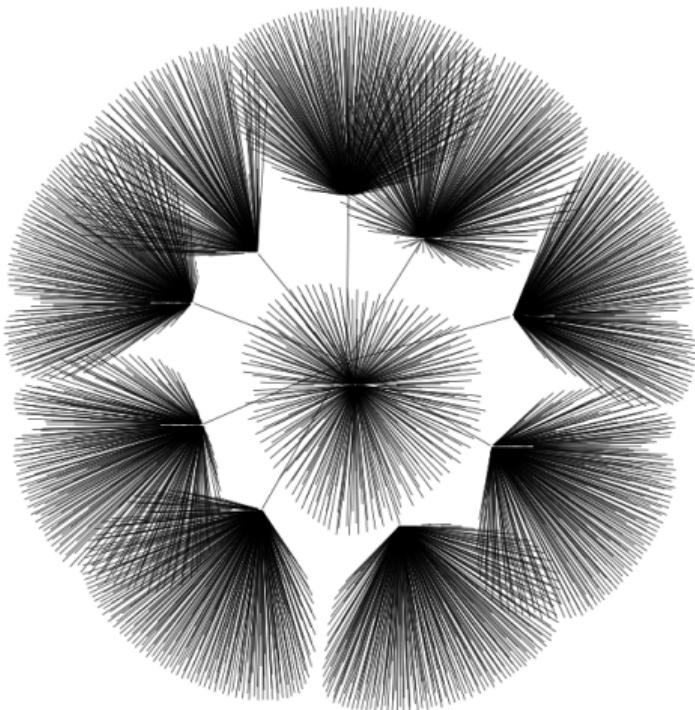


Modern BTree techniques

DMITRII DOLGOV

24-09-2022



→ Comer D. (1979). The Ubiquitous B-Tree.
Computing Surveys, 11 (2).

- Comer D. (1979). The Ubiquitous B-Tree.
Computing Surveys, 11 (2).
- Graefe G. (2011). Modern B-Tree Techniques.
Foundations and Trends in Databases.

→ RUM conjecture

→ B-Tree internals

Key normalization

Prefix/Suffix truncation

→ B-Tree as a component

Hybrid index

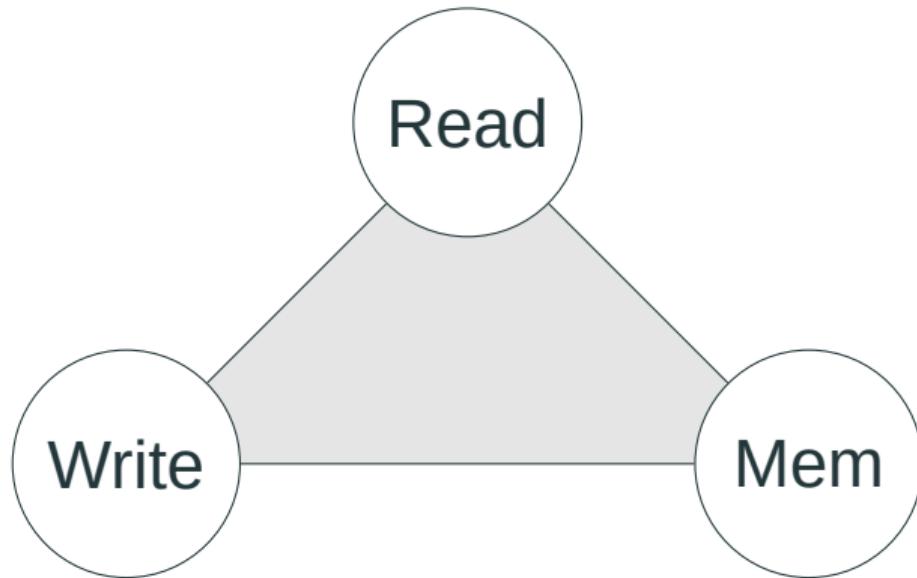
Bw-Tree

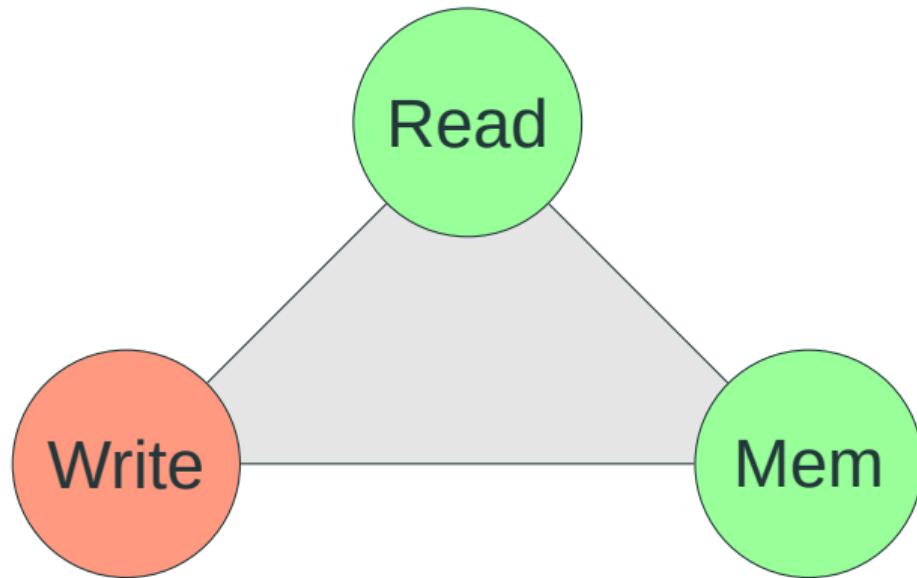
Partitioned B-Tree

→ Learned Indexes

→ Everything else

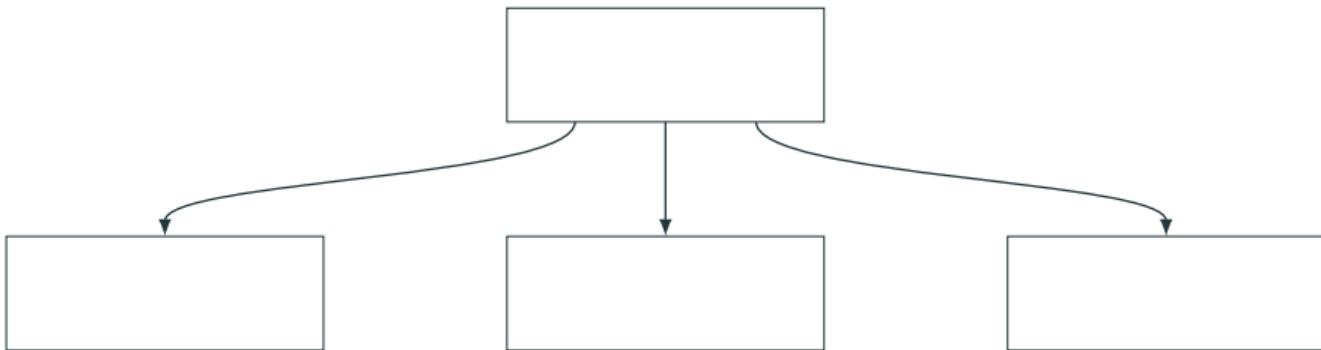
RUM conjecture





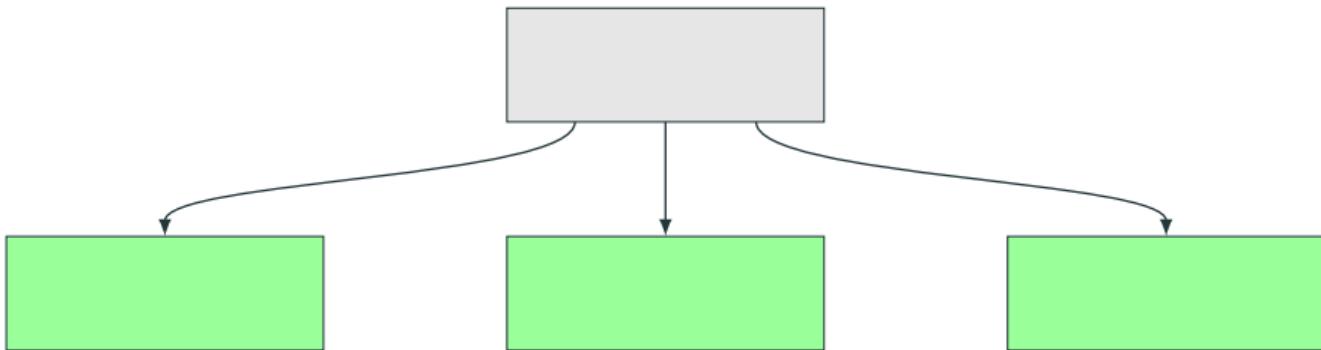
B-Tree internals

B-Tree → B⁺Tree



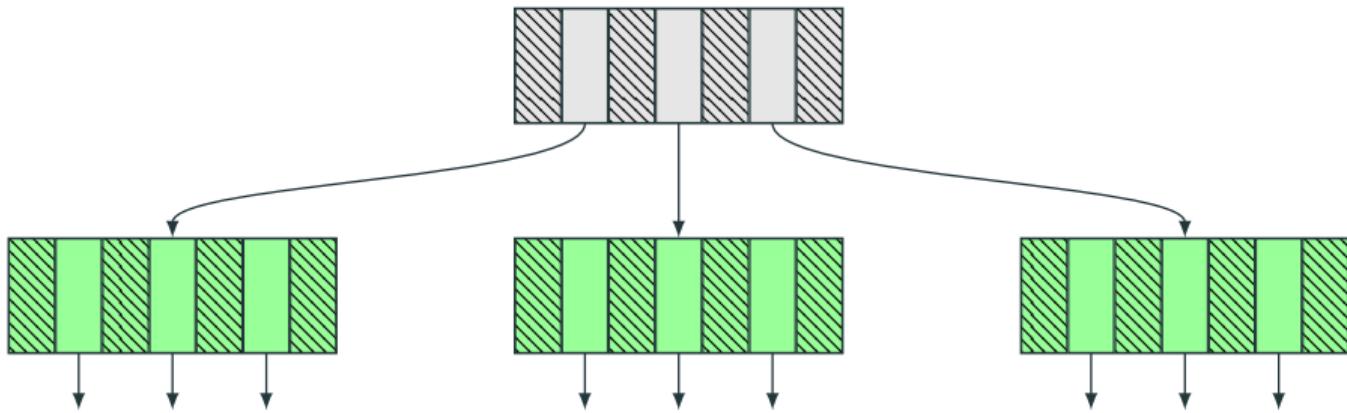
Lehman P., Yao S. (1981). Efficient Locking for Concurrent Operations on B-Trees.
ACM Transactions on Database Systems.

B-Tree → B⁺Tree



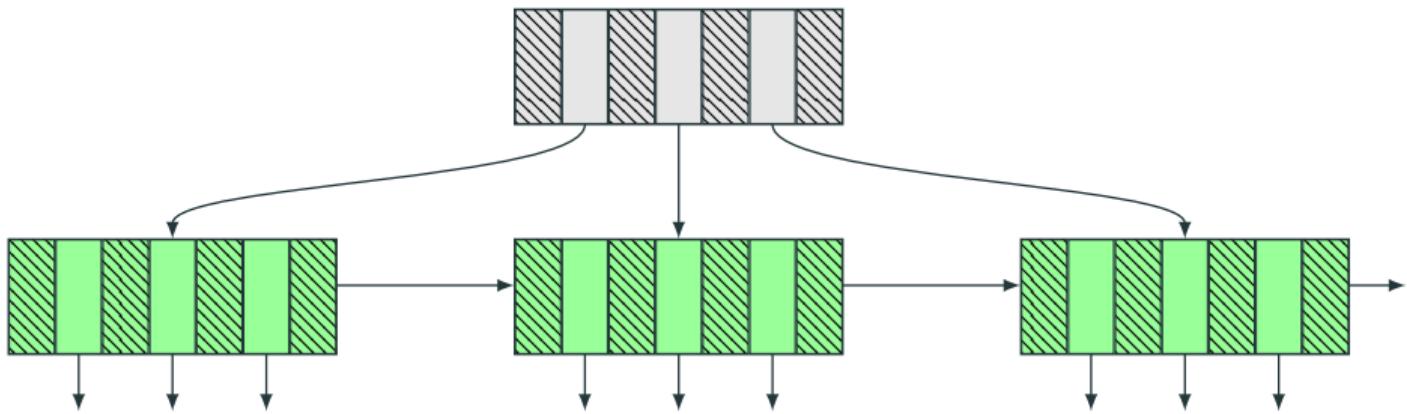
Lehman P., Yao S. (1981). Efficient Locking for Concurrent Operations on B-Trees.
ACM Transactions on Database Systems.

B-Tree → B⁺Tree



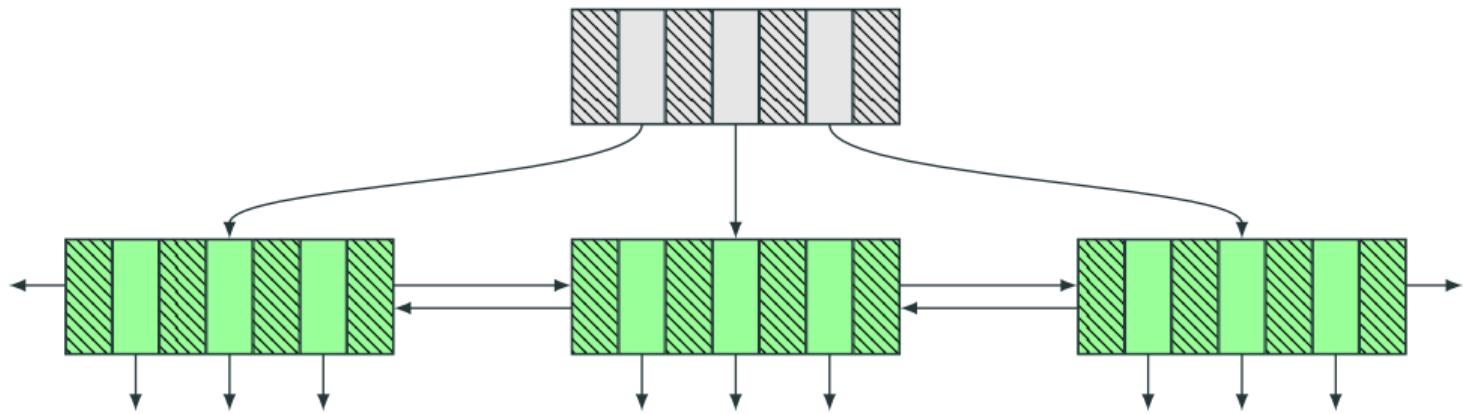
Lehman P., Yao S. (1981). Efficient Locking for Concurrent Operations on B-Trees.
ACM Transactions on Database Systems.

B-Tree → B⁺Tree



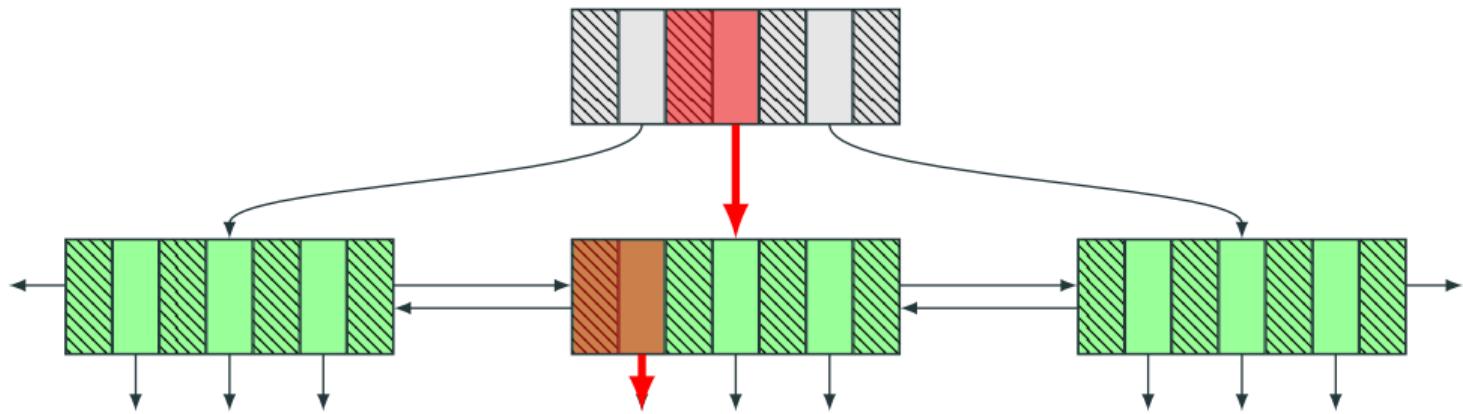
Lehman P., Yao S. (1981). Efficient Locking for Concurrent Operations on B-Trees.
ACM Transactions on Database Systems.

B-Tree → B⁺Tree



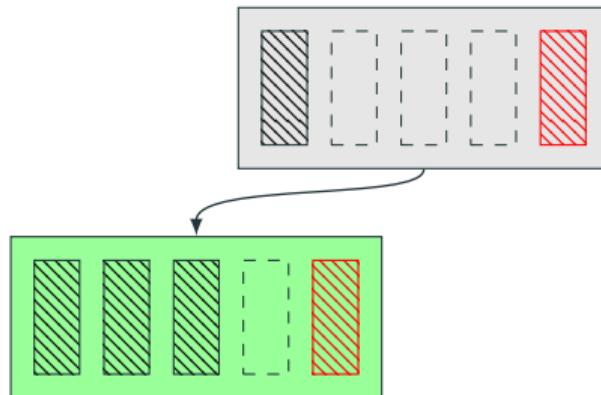
Lehman P., Yao S. (1981). Efficient Locking for Concurrent Operations on B-Trees.
ACM Transactions on Database Systems.

B-Tree → B⁺Tree

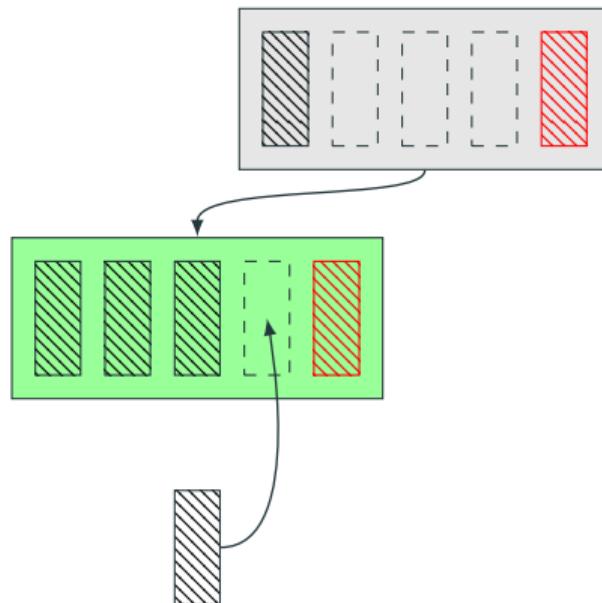


Lehman P., Yao S. (1981). Efficient Locking for Concurrent Operations on B-Trees.
ACM Transactions on Database Systems.

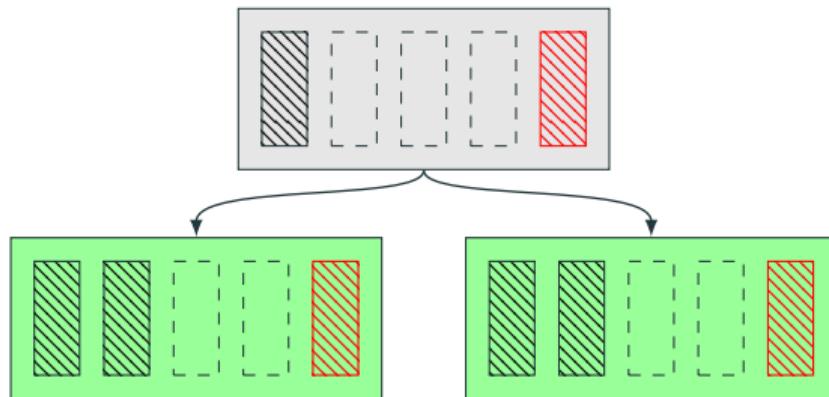
Page split



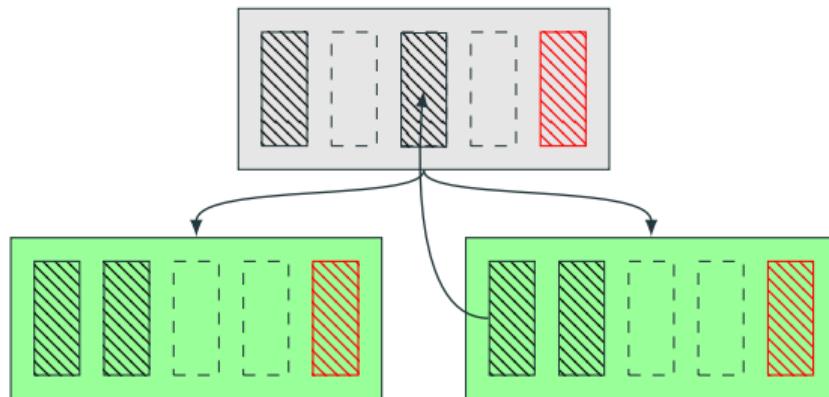
Page split



Page split



Page split



Page merge

"By adding periodic rebuilding of the tree, we obtain a data structure that is theoreticaly superior to standard B-trees in many ways. Our results suggest that rebalancing on deletion no only unnecessary but may be harmful."

Sen S., Tarjan R. E. (2009). Deletion without rebalancing in multiway search trees ACM Transactions on Database Systems

Key normalization

1	"Dirk"	"Bart"	1	0..01	1	11..00	∅	1	10..10	∅
2	"Todd"	Null	1	0..10	1	01..00	∅	0		
Null	""	Null	0	1	∅	0				

Singleton, R. C. (1969). An efficient algorithm for sorting with minimal storage.
Communications of the ACM.

Prefix truncation

Smith Jack	01-02-2019
Smith Jane	02-03-2019
Smith James	03-04-2019

Prefix truncation

Smith Jack	01-02-2019
Smith Jane	02-03-2019
Smith James	03-04-2019

Prefix truncation

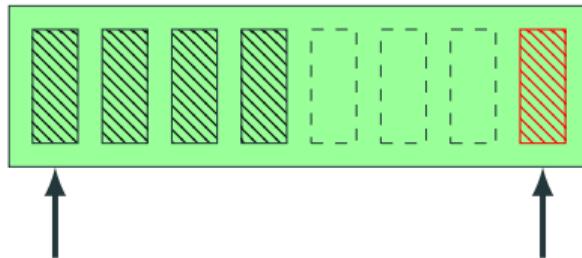
Smith Ja

ck 01-02-2019

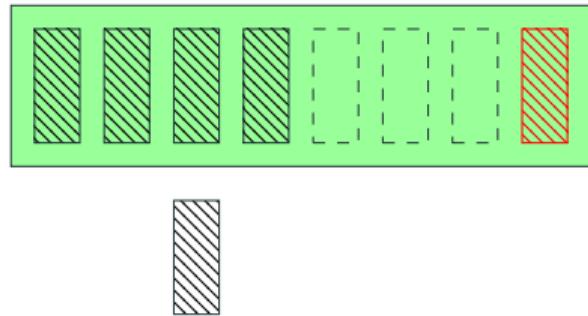
ne 02-03-2019

mes 03-04-2019

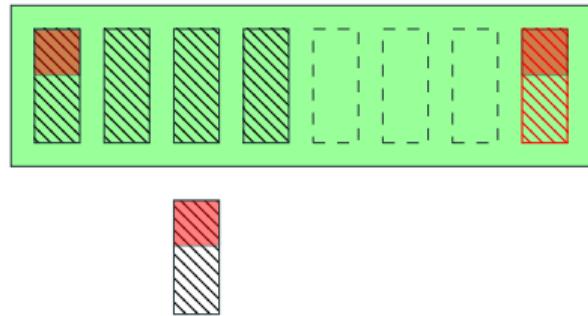
Prefix truncation



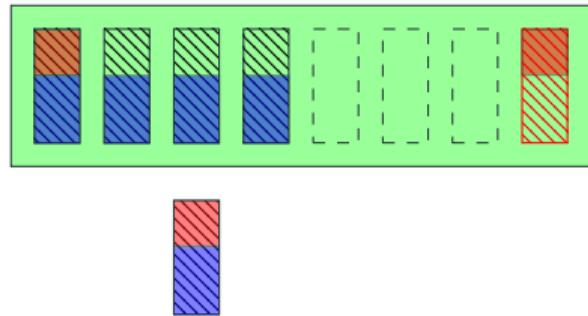
Dynamic prefix truncation



Dynamic prefix truncation



Dynamic prefix truncation



Suffix truncation

Cohen Richard	02-02-2019
Johnson David	04-02-2019
Miller Kevin	01-02-2019
Miller Mary	02-03-2019
Miller Steven	03-04-2019
Smith Susan	01-02-2019

Bayer, R., Unterauer, K. (1977). Prefix B-trees.
ACM Transactions on Database Systems.

Suffix truncation

Cohen Richard 02-02-2019

Johnson David 04-02-2019

Miller Kevin 01-02-2019

Miller Mary 02-03-2019

Miller Steven 03-04-2019

Smith Susan 01-02-2019

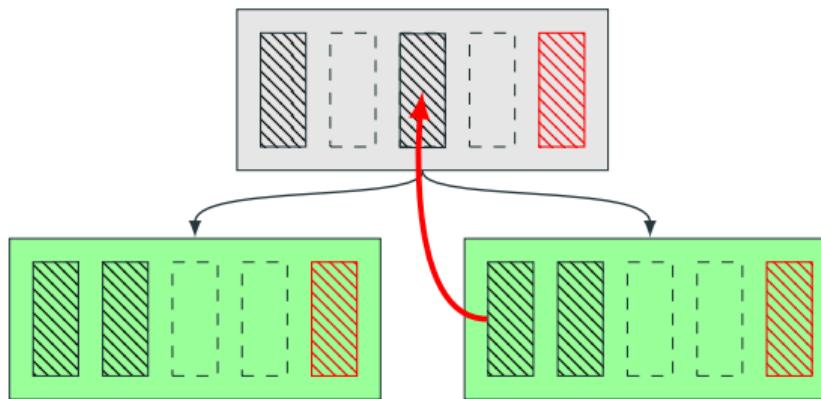
Bayer, R., Unterauer, K. (1977). Prefix B-trees.
ACM Transactions on Database Systems.

Suffix truncation

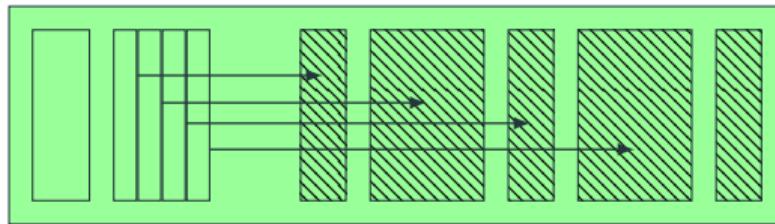
Cohen Richard	02-02-2019
Johnson David	04-02-2019
Miller Kevin	01-02-2019
Miller Mary	02-03-2019
Miller Steven	03-04-2019
Smith Susan	01-02-2019

Bayer, R., Unterauer, K. (1977). Prefix B-trees.
ACM Transactions on Database Systems.

Page split

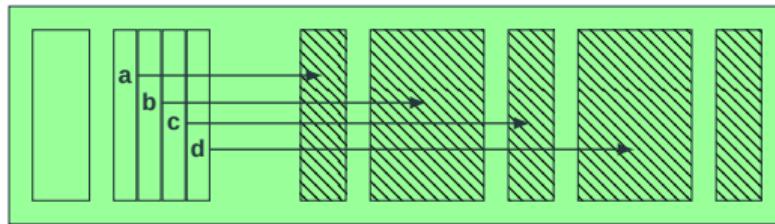


Indirection vector



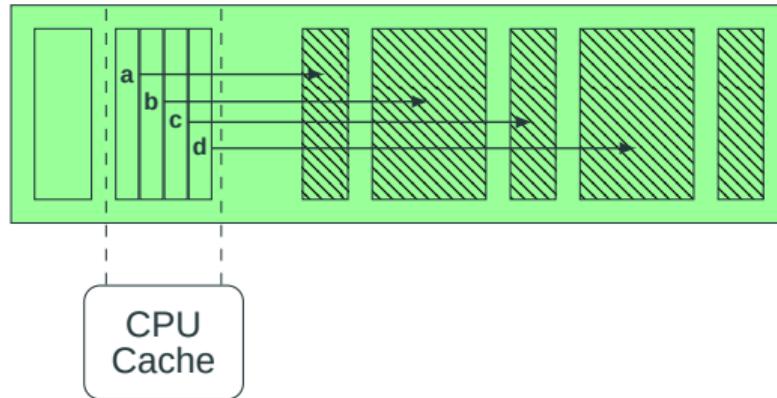
Graefe G., Larson P. (2001). B-tree indexes and CPU caches.
International Conference on Data Engineering.

Indirection vector

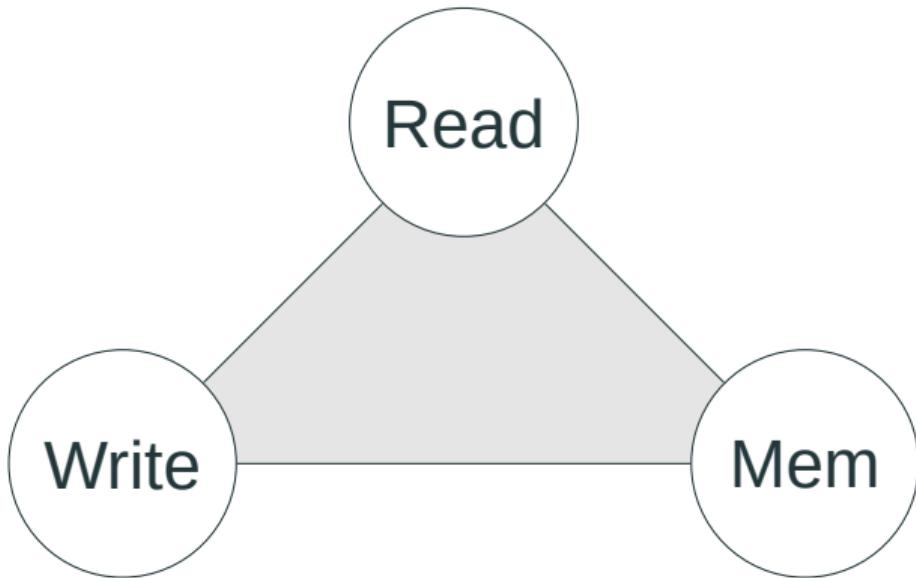


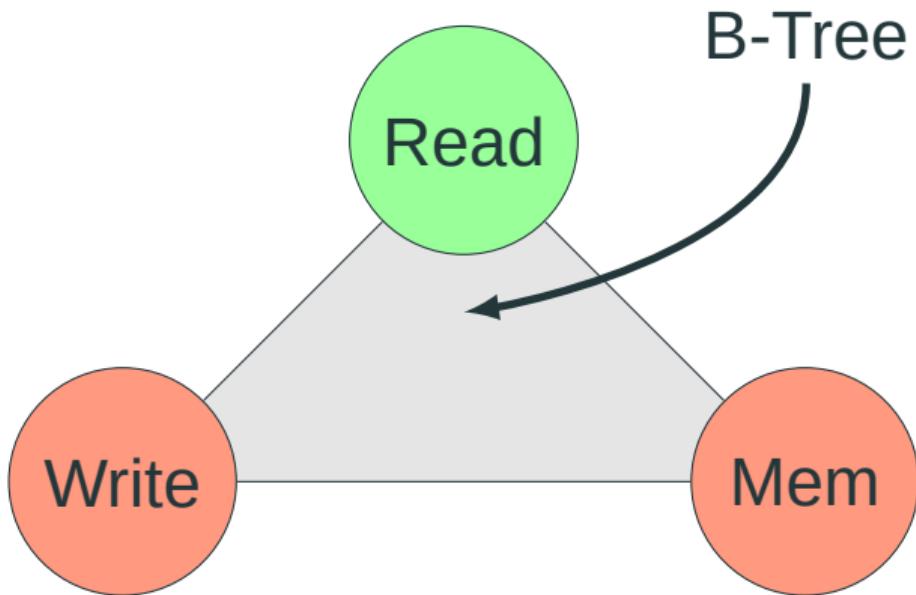
Graefe G., Larson P. (2001). B-tree indexes and CPU caches.
International Conference on Data Engineering.

Indirection vector



Graefe G., Larson P. (2001). B-tree indexes and CPU caches.
International Conference on Data Engineering.





Waves of misery after index creation

Nikolaus Glombiewski¹, Bernhard Seeger², Goetz Graefe³

Abstract: After creation of a new b-tree, the ordinary course of database updates and index maintenance causes waves of node splits. Thus, a new index may at first speed up database query processing but then the first “wave of misery” requires effort for frequent node splits and imposes spikes of buffer pool contention and of I/O. Waves of misery continue over multiple instances although eventually the waves widen, flatten, and spread further apart. Free space in each node left during index creation fails to prevent the problem; it merely delays the onset of the first wave. We have found a theoretically sound way to avoiding these waves of misery as well as some simple and practical means to reduce their amplitude to negligible levels. Experiments demonstrate that these techniques are also effective. Waves of misery occur in databases and in key-value stores, in primary and in secondary b-tree indexes, after load operations, and after b-tree reorganization or rebuild. The same remedies apply with equal effect.

Keywords: Indexing, Bulk Loading, B-tree

1 Introduction

The purpose of adding an index to a database table is to improve the performance of query processing. Unfortunately, after an initial “honey moon” of using a new index and of enjoying fast queries, the index may grow and require many node splits – thus creating a spike in buffer pool contention and I/O activity. In other words, the index may actually reduce query performance or at least fail to achieve the expected performance. Consider the following example. First, a new secondary b-tree index enables fast look-ups and fast ordered scans. In order to absorb updates without node splits, each node might start with 10% free space – often a parameter of index creation. However, once the table and the

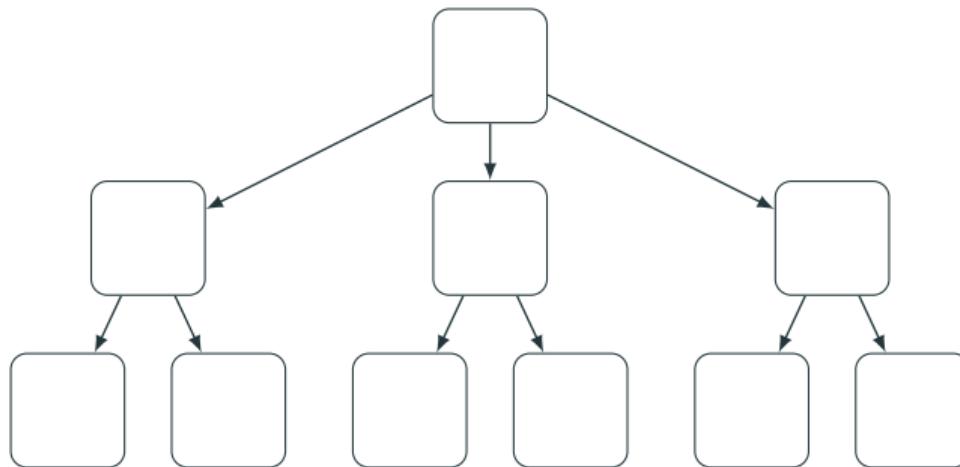
B-Tree as a component

B-Tree	B ⁺ -Tree	B _{link} -Tree	DPTree
wB ⁺ -Tree	NV-Tree	FPTree	FASTFAIR
HiKV	Masstree	Skip List	ART
WORT	CDDS-Tree	Bw-Tree	HOT
KISS-Tree	VAST-Tree	FAST	HV-Tree
UB-Tree	LHAM	PBT	Hybrid B ⁺ -Tree

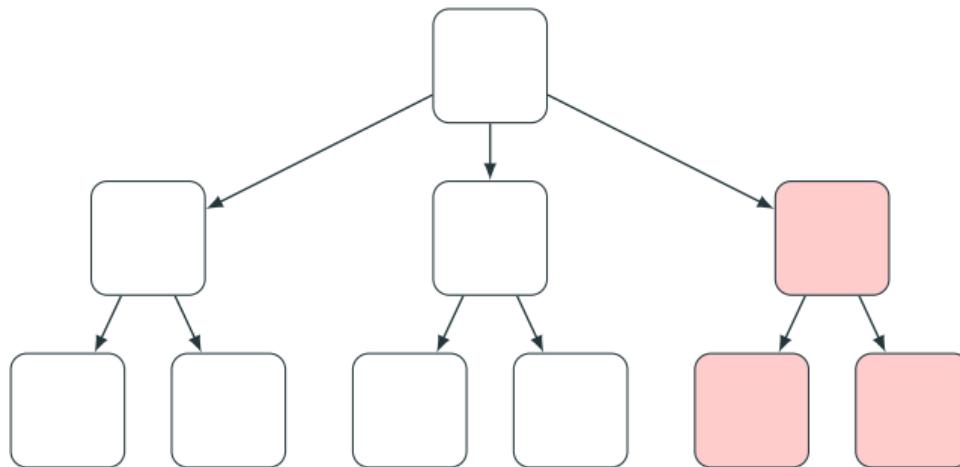
B-Tree	B ⁺ -Tree	B _{link} -Tree	DPTree
wB ⁺ -Tree	NV-Tree	FPTree	FASTFAIR
HiKV	Masstree	Skip List	ART
WORT	CDDS-Tree	Bw-Tree	HOT
KISS-Tree	VAST-Tree	FAST	HV-Tree
UB-Tree	LHAM	PBT	Hybrid B ⁺ -Tree

B-Tree	B ⁺ -Tree	B _{link} -Tree	DPTree
wB ⁺ -Tree	NV-Tree	FPTree	FASTFAIR
HiKV	Masstree	Skip List	ART
WORT	CDDS-Tree	Bw-Tree	HOT
KISS-Tree	VAST-Tree	FAST	HV-Tree
UB-Tree	LHAM	PBT	Hybrid B ⁺ -Tree

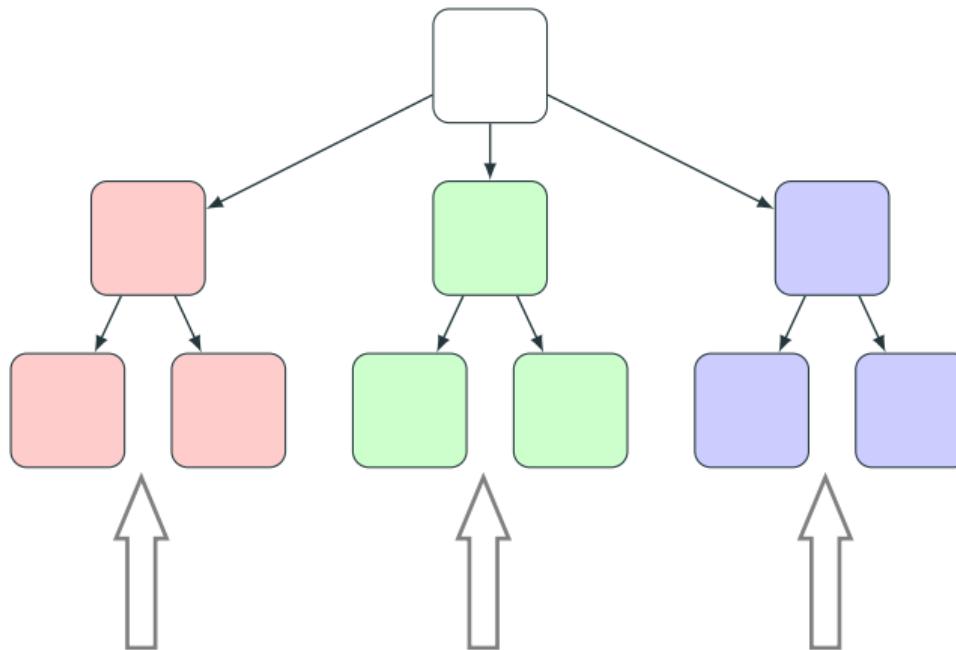
Partitioned B-tree



Partitioned B-tree

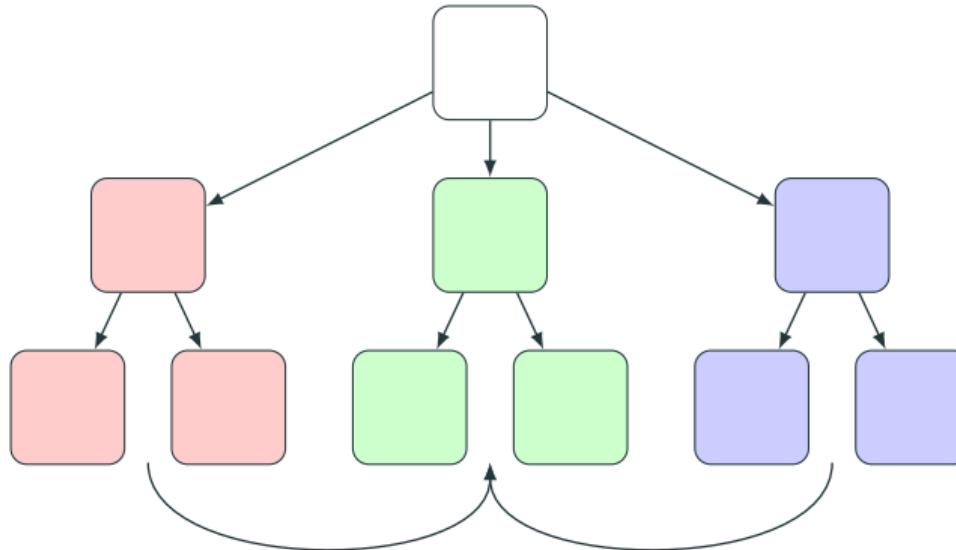


Partitioned B-tree



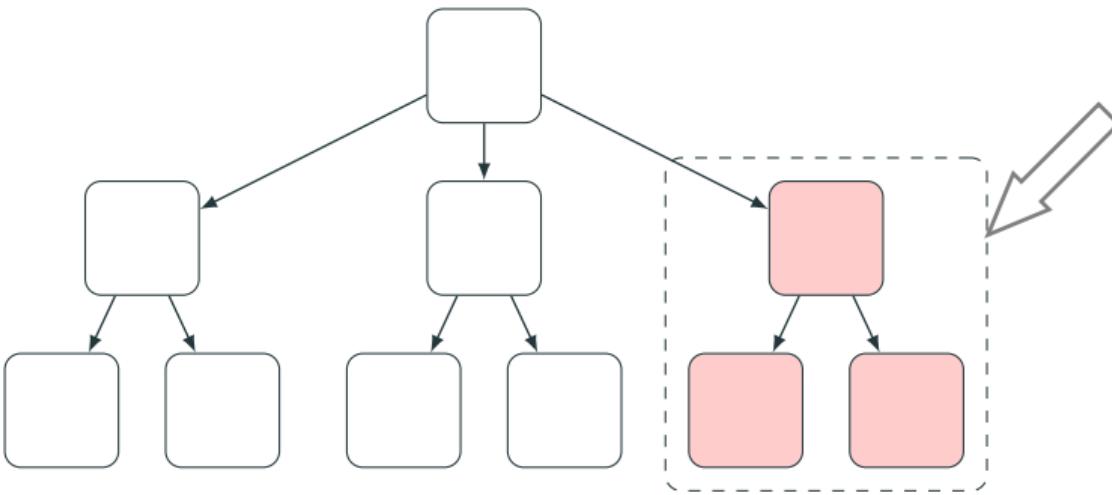
Graefe G. (2003). Sorting and indexing with partitioned B-Trees.
Classless Inter Domain Routing

Partitioned B-tree



Graefe G. (2003). Sorting and indexing with partitioned B-Trees.
Classless Inter Domain Routing

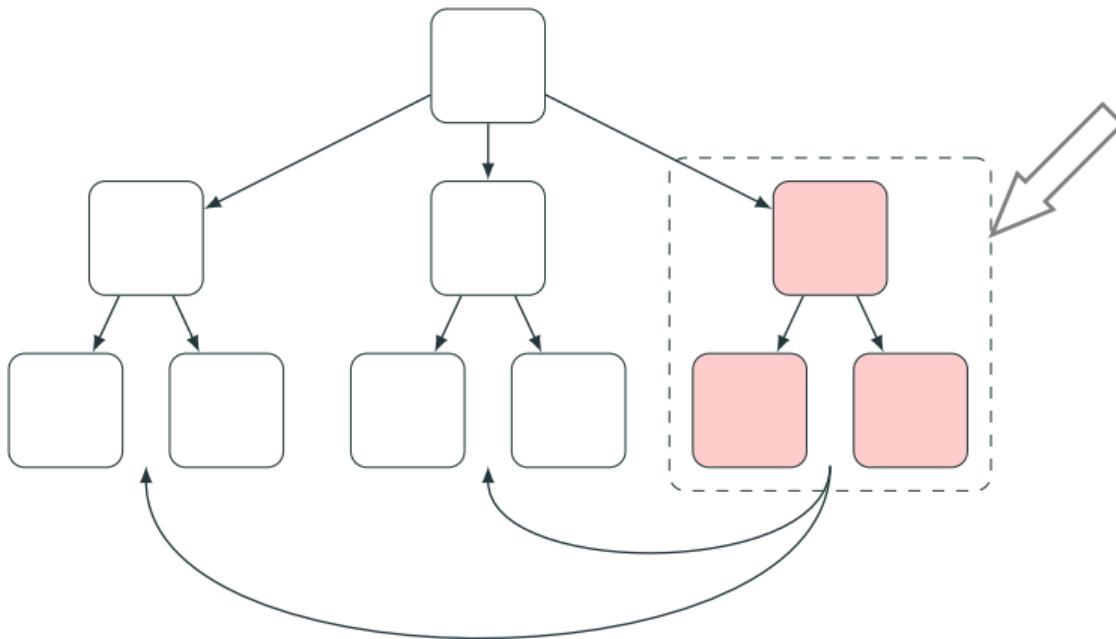
Partitioned B-tree



Riegger C., Vincon T., Petrov I. (2017). Write-optimized indexing with partitioned b-trees.
Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services

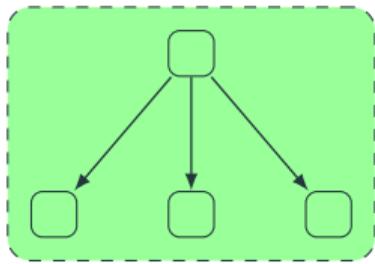


Partitioned B-tree



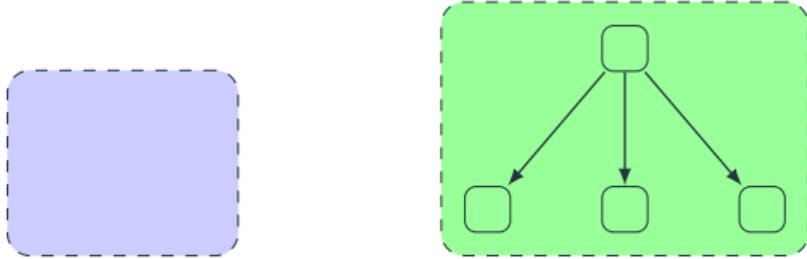
Riegger C., Vincon T., Petrov I. (2017). Write-optimized indexing with partitioned b-trees.
Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services

Hybrid indexes



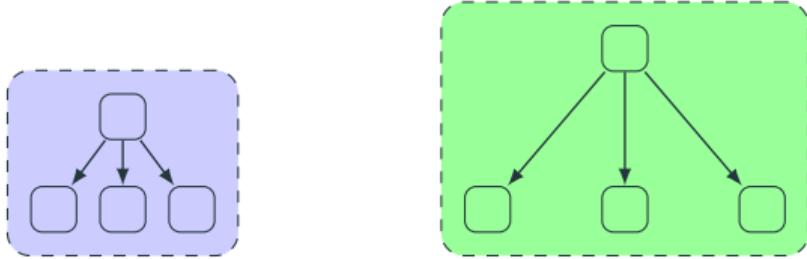
Zhang H., Andersen D. G., Pavlo A., Kaminsky M., Ma L., Shen R. (2016).
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.
Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)

Hybrid indexes



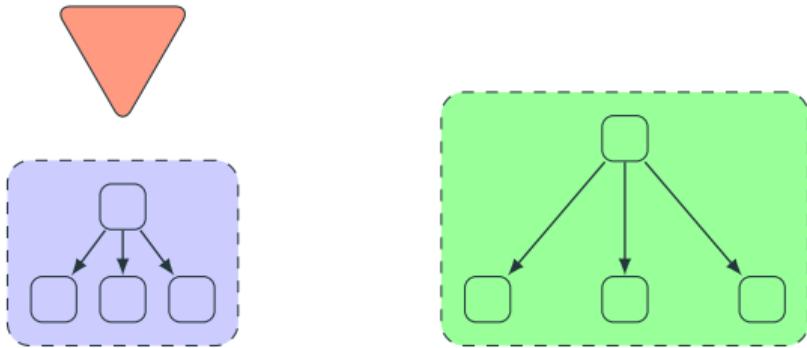
Zhang H., Andersen D. G., Pavlo A., Kaminsky M., Ma L., Shen R. (2016).
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.
Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)

Hybrid indexes



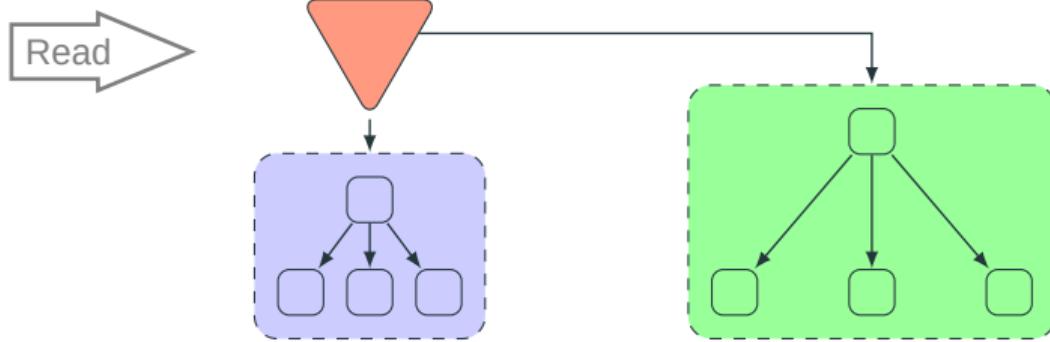
Zhang H., Andersen D. G., Pavlo A., Kaminsky M., Ma L., Shen R. (2016).
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.
Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)

Hybrid indexes



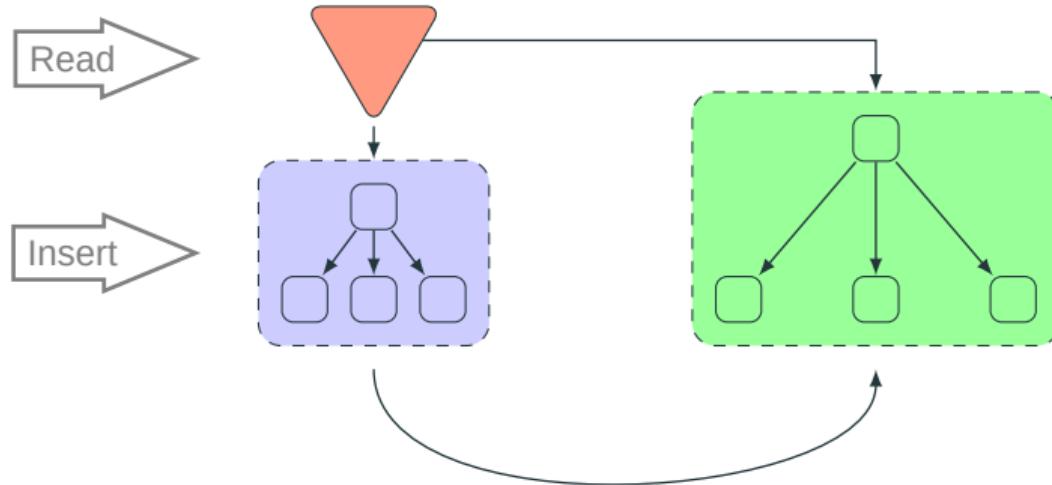
Zhang H., Andersen D. G., Pavlo A., Kaminsky M., Ma L., Shen R. (2016).
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.
Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)

Hybrid indexes



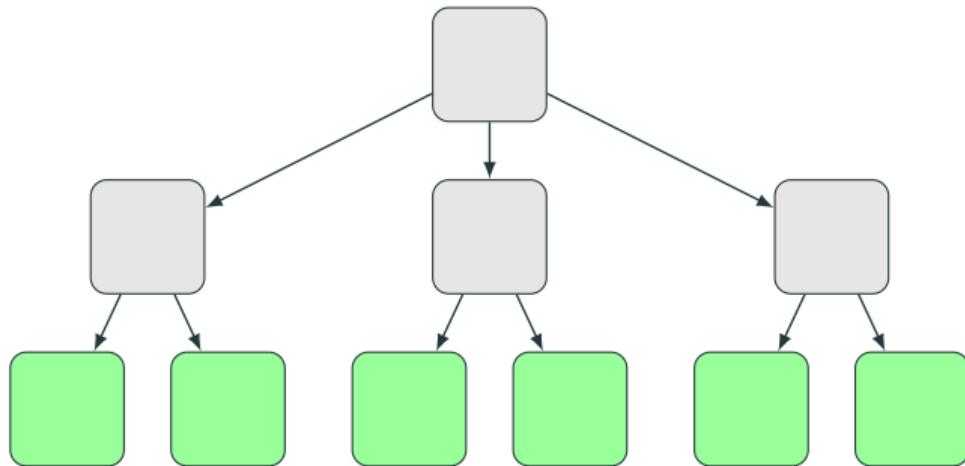
Zhang H., Andersen D. G., Pavlo A., Kaminsky M., Ma L., Shen R. (2016).
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.
Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)

Hybrid indexes



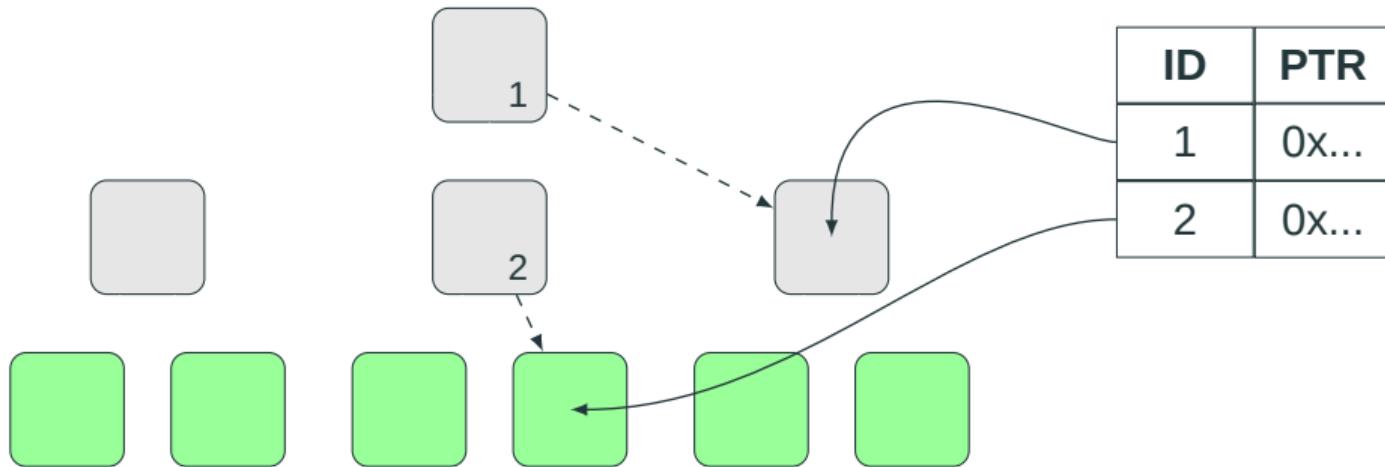
Zhang H., Andersen D. G., Pavlo A., Kaminsky M., Ma L., Shen R. (2016).
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.
Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)

Bw-Tree



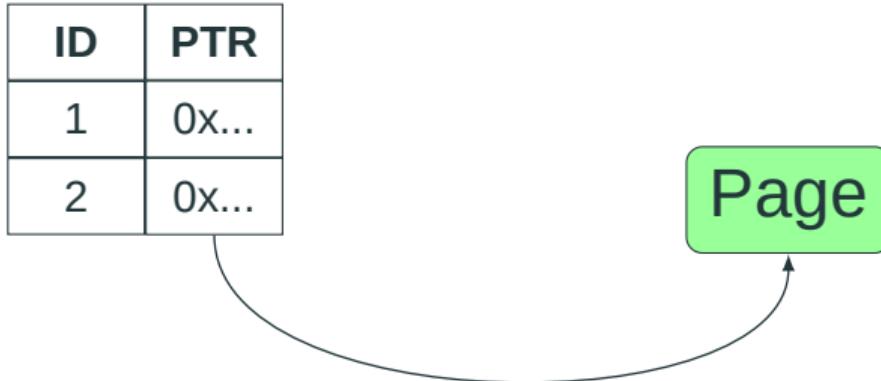
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



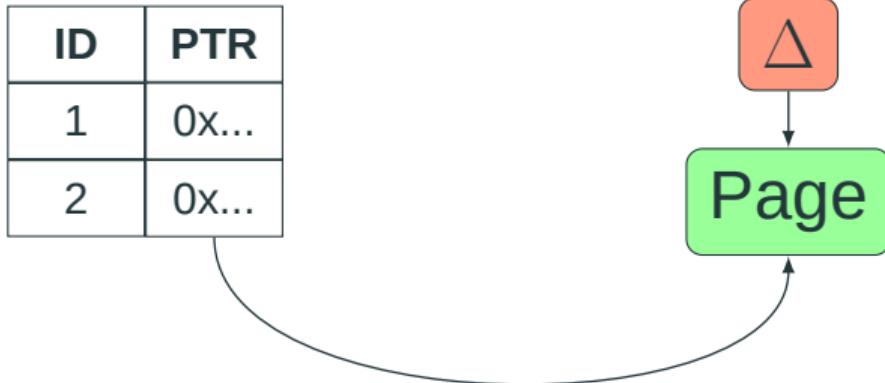
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



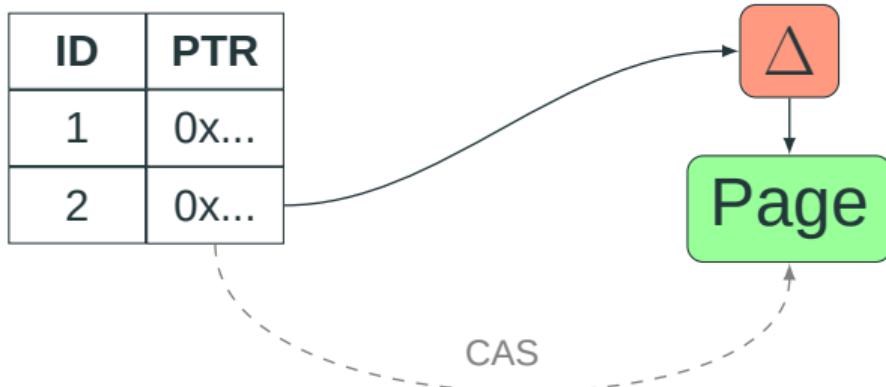
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



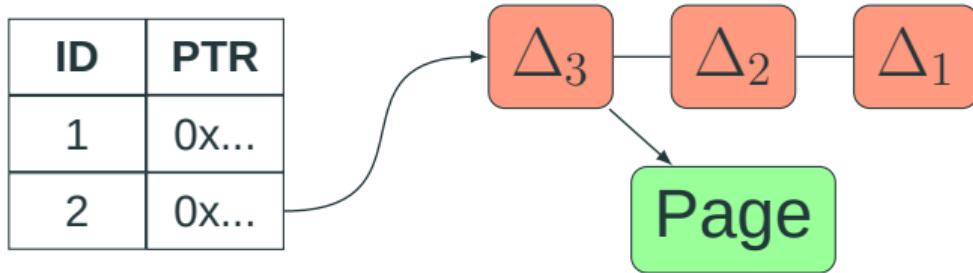
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



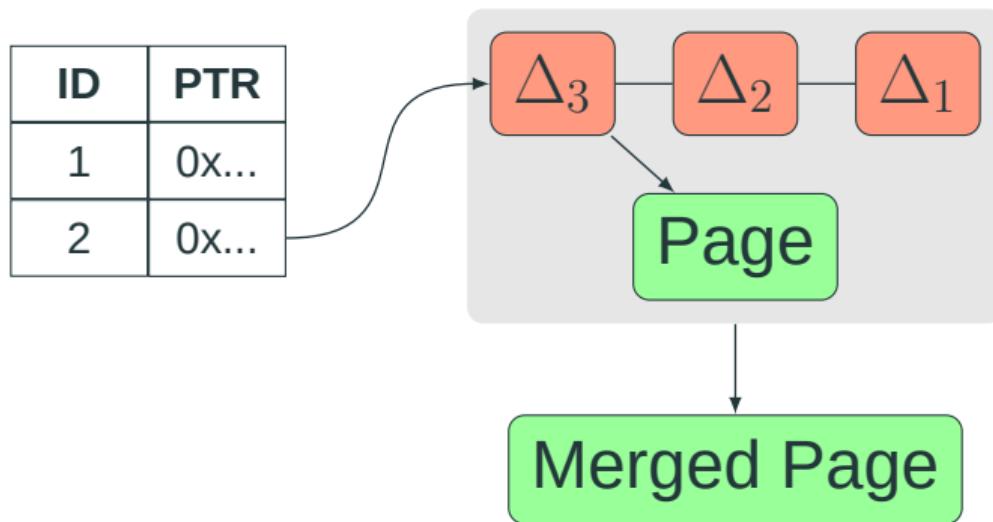
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



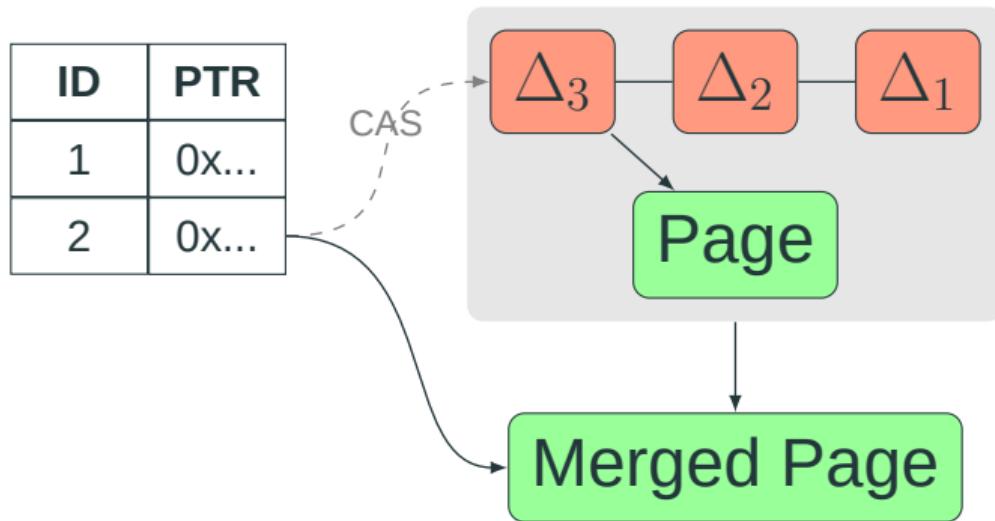
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



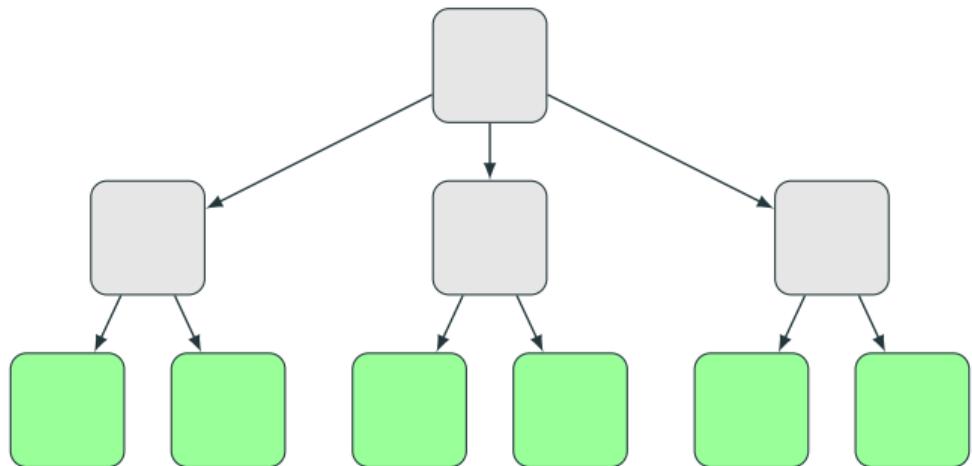
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms.
International Conference on Data Engineering.

Bw-Tree



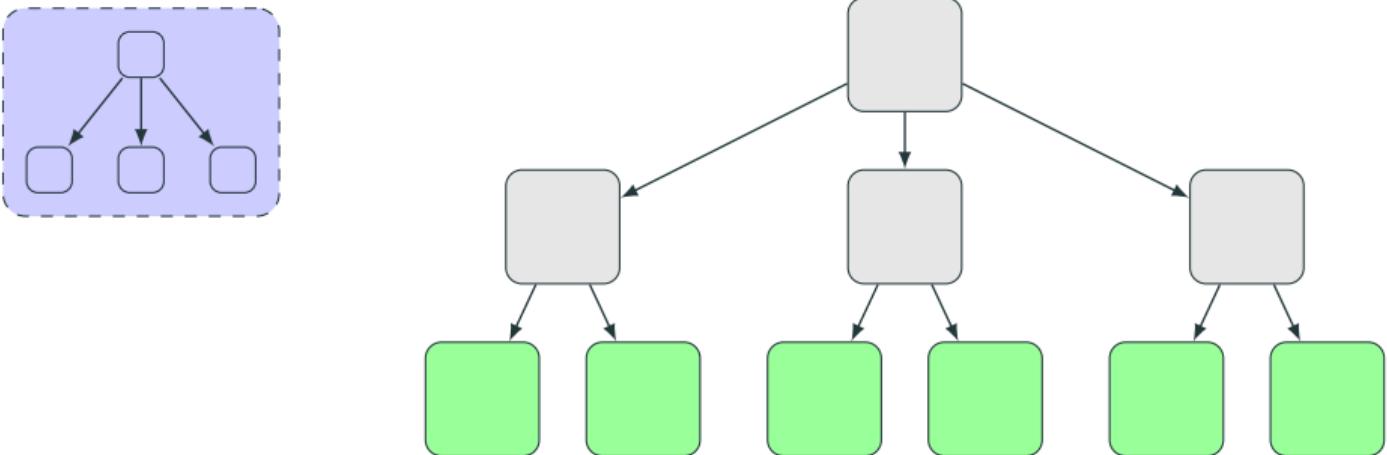
Levandoski J., Lomet D., Sengupta S. (2012). The Bw-Tree: A B-tree for New Hardware Platforms. International Conference on Data Engineering.

DPTree



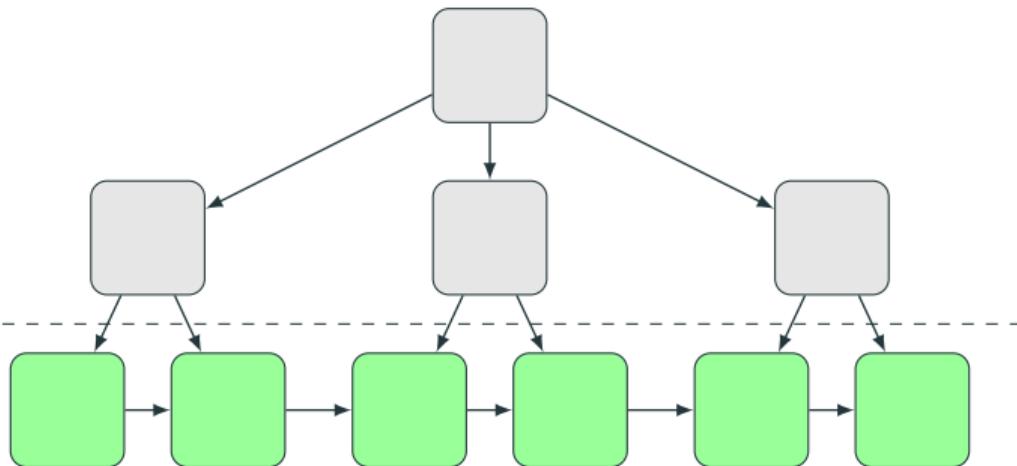
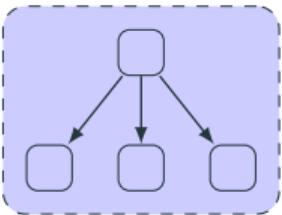
Zhou Xinjing, Shou Lidan, Chen Ke, Hu Wei, Chen Gang. (2019). DPTree: differential indexing for persistent memory.
Proceedings of the VLDB Endowment.

DPTree



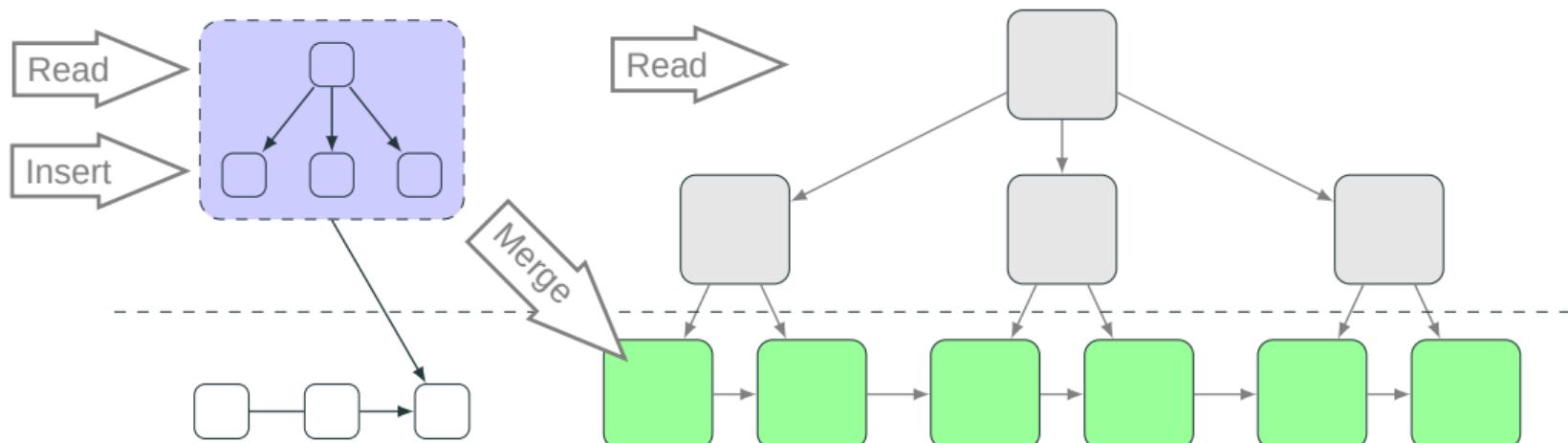
Zhou Xinjing, Shou Lidan, Chen Ke, Hu Wei, Chen Gang. (2019). DPTree: differential indexing for persistent memory.
Proceedings of the VLDB Endowment.

DPTree



Zhou Xinjing, Shou Lidan, Chen Ke, Hu Wei, Chen Gang. (2019). DPTree: differential indexing for persistent memory.
Proceedings of the VLDB Endowment.

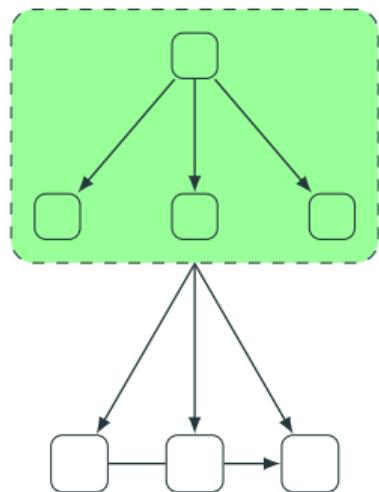
DPTree



Zhou Xinjing, Shou Lidan, Chen Ke, Hu Wei, Chen Gang. (2019). DPTree: differential indexing for persistent memory. Proceedings of the VLDB Endowment.

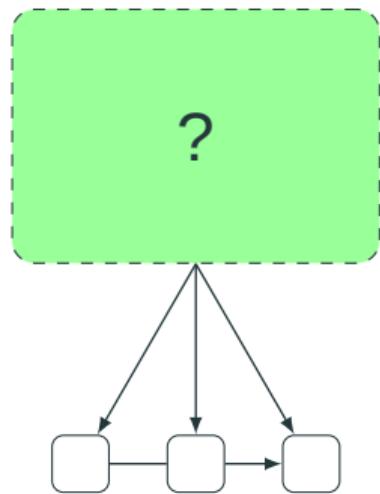
Learned Indexes

Learned Indexes



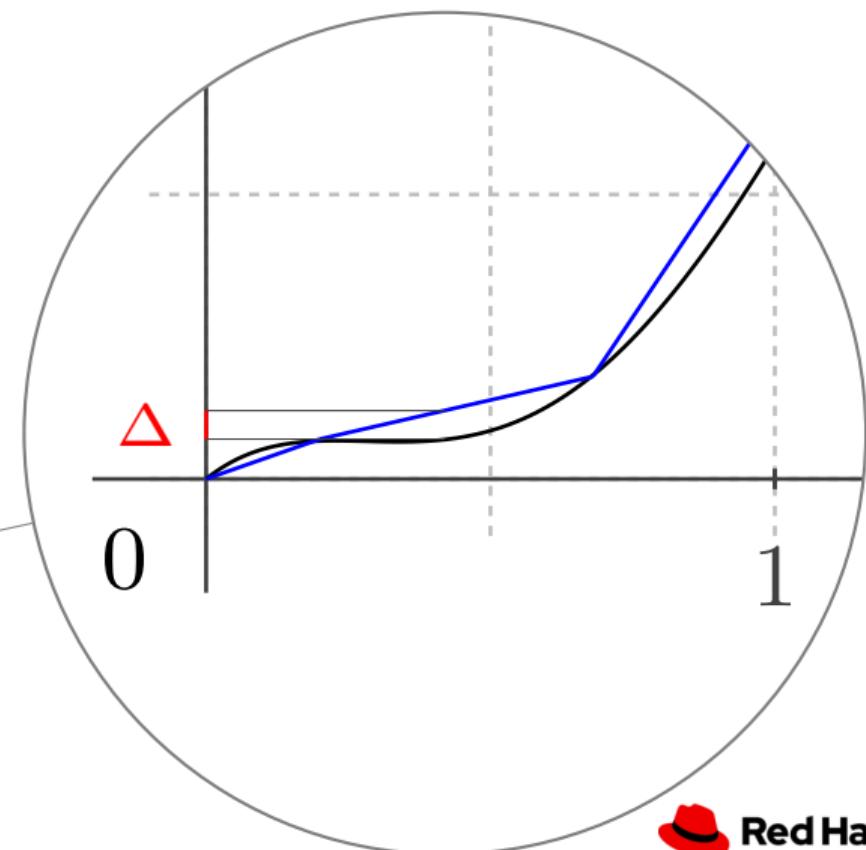
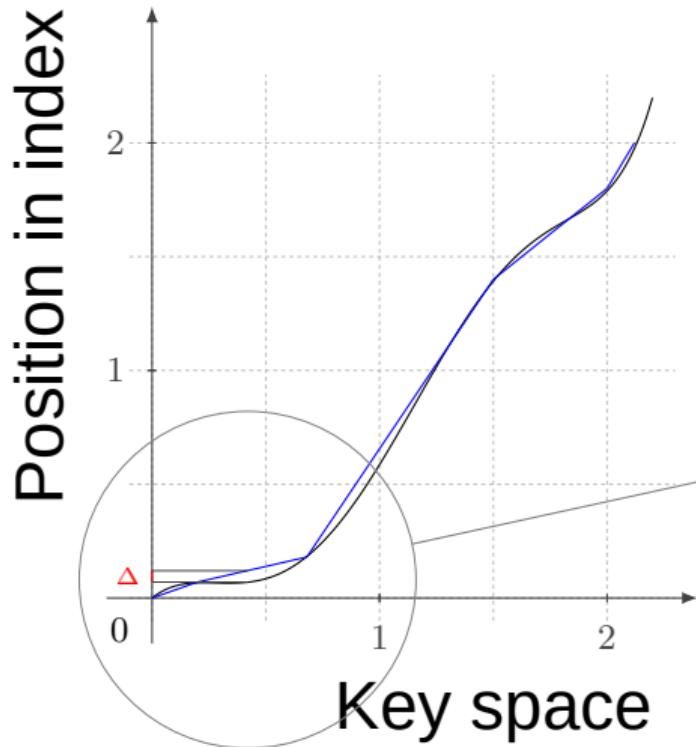
Kraska T., Beutel A., Chi E. H., Dean J., Polyzotis N. (2018). The Case for Learned Index Structures.
2018 International Conference on Management of Data (SIGMOD '18).

Learned Indexes



Kraska T., Beutel A., Chi E. H., Dean J., Polyzotis N. (2018). The Case for Learned Index Structures.
2018 International Conference on Management of Data (SIGMOD '18).

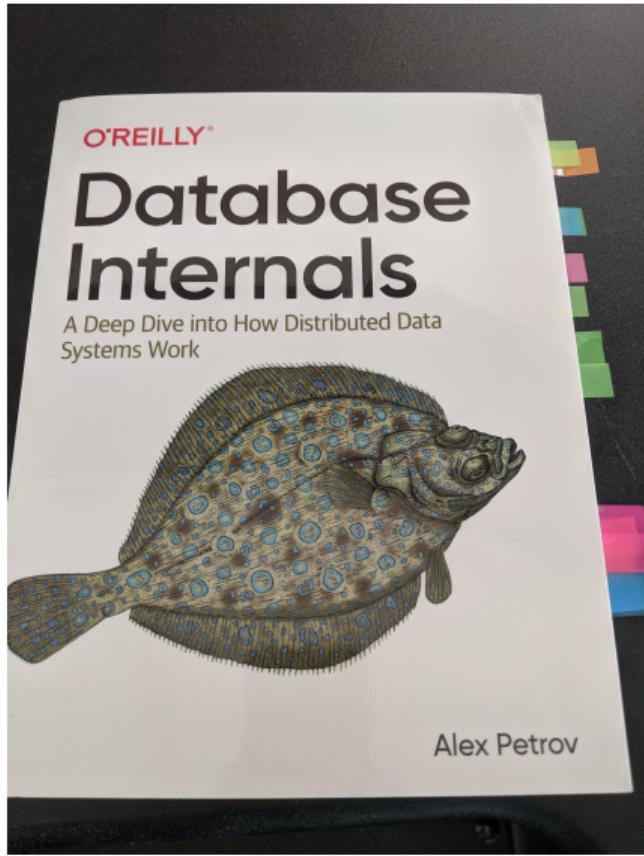
Data approximation



Is that all?



- Buffered (Lazy) B-Tree
- Cache conscious B-Tree
- Distributed B-Tree
- SB-Tree
- MDAM
- LSM Tree
- Trie
- ...



Questions?

 @erthalion

 ddolgov at redhat dot com