A Tutorial on Learned Multi-dimensional Indexes

SIGSPATIAL 2020

Presenters: Abdullah-Al-Mamun, Hao Wu, Walid G. Aref
Purdue University
West Lafayette, Indiana, USA

Website of Tutorial: https://www.cs.purdue.edu/homes/aref/learned-indexes-tutorial.html

Outline of the Tutorial

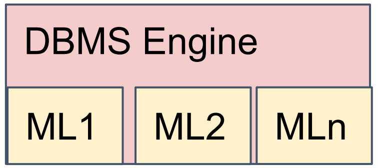
- Introduction and Taxonomy
- Indexing the Learned Models vs. Learning the Indexes
- Static vs. Dynamic Learned Indexes
- Learned One-Dimensional Indexes
- Learned Multidimensional Indexes
- Open Problems for Future Research

Outline of the Tutorial

- **Introduction and Taxonomy**
- Indexing the Learned Models vs. Learning the Indexes
- Static vs. Dynamic Learned Indexes
- Learned One-Dimensional Indexes
- Learned Multidimensional Indexes
- Open Problems for Future Research

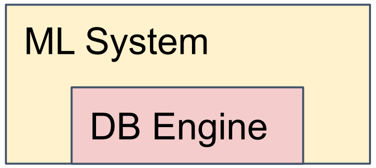
Introduction: ML and DB

- Machine Learning (ML) has been successful in many application domains
- Two recent trends of research in the area of Database Systems (DB):



ML for DB

- *Machine Learning for Database Systems (ML for DB)*
  - Replace core components of a Database System (e.g., query optimizer, Indexes, DB administration) with Machine Learning techniques
    - Achieve better performance
    - Less space requirement

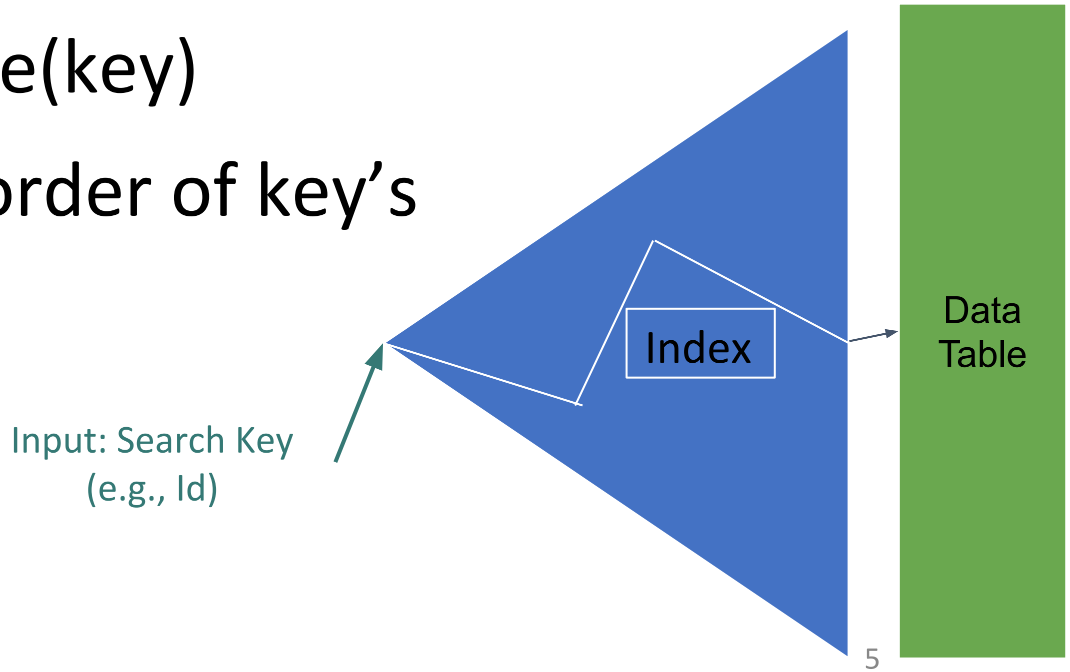


DB for ML

- *Database Systems for Machine Learning (DB for ML)*
  - Extend database system techniques to support efficient ML workloads

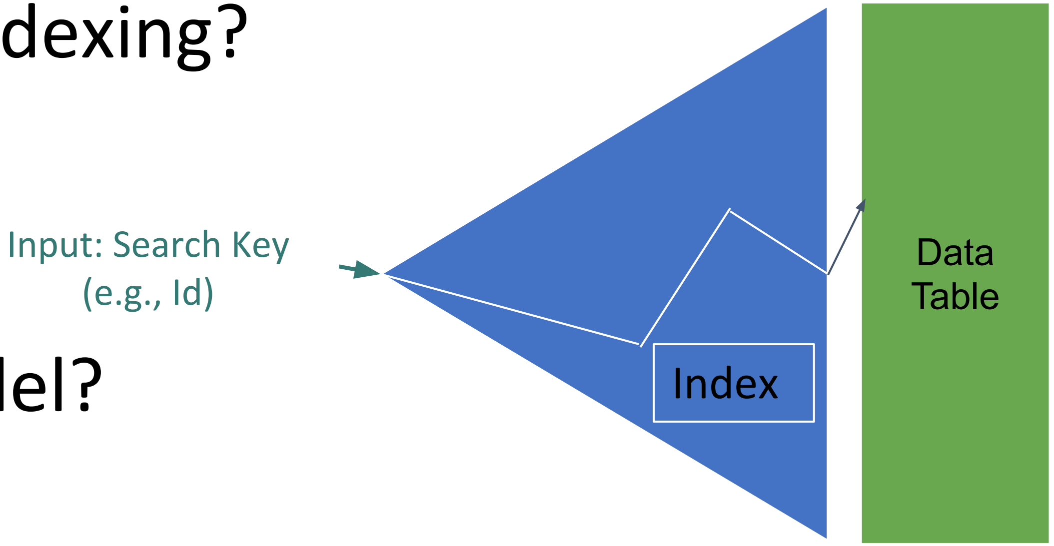
Introduction: Database Indexing

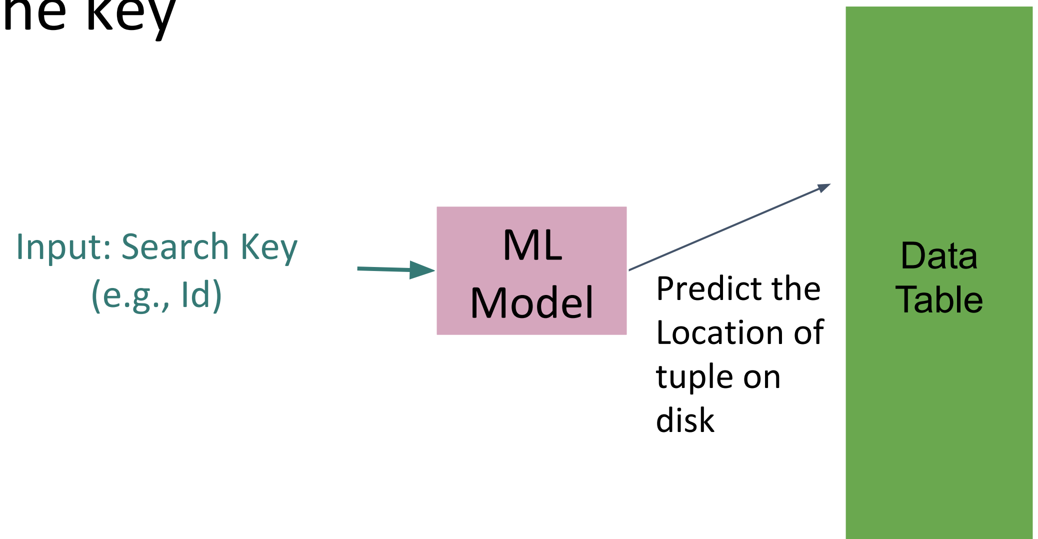
- Database Index: Provide efficient access to data
- Popular index structure is: B+-tree
- Given search key, B+-tree identifies the storage location of the tuple that contains the search key
- Can view the B+-tree as a function: B+-tree(key) that takes a key as input and returns the order of key's tuple inside the table



Introduction: Learned Index

- Can one use ML techniques to guide data indexing?
- Can we learn the function:
  - B+-tree(key) → Location of tuple in table?
- Can we replace the B+-tree with an ML model?
  - “Index as a model”
  - ML_Model (key) predicts the storage location of the key
  - Searching executes potentially in O(1) time
  - → “Learned Index”

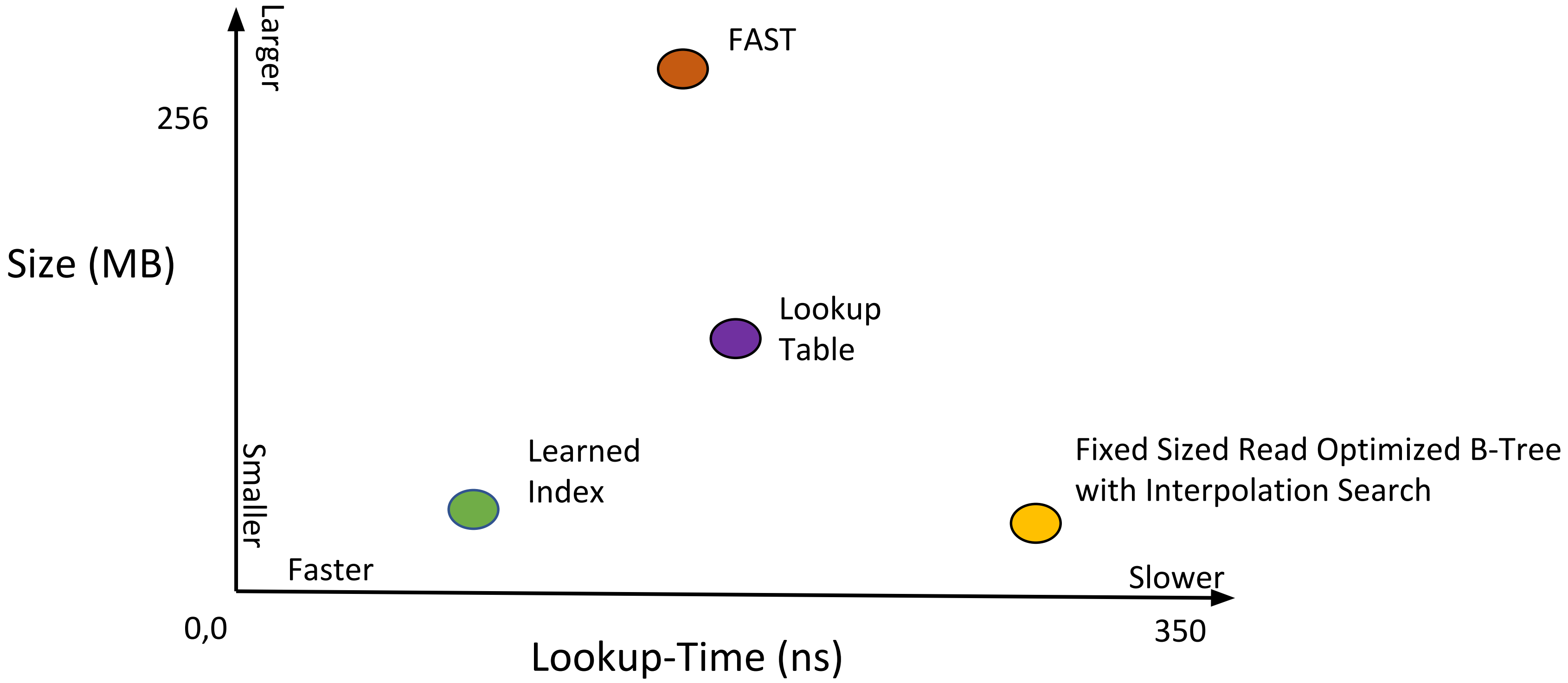




Introduction: Initial Results

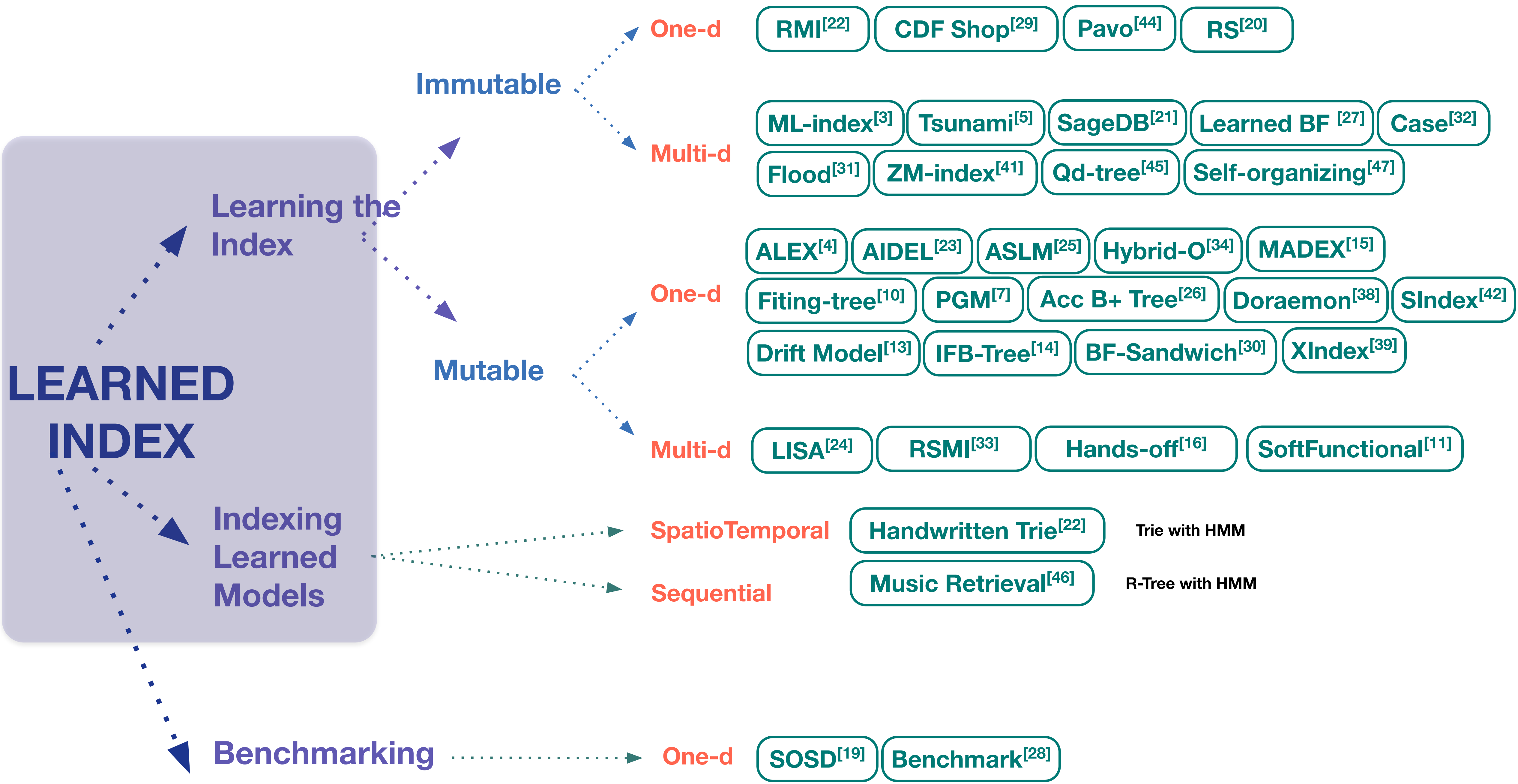
Initial performance results (approximate) of a learned index

Promising → Faster lookup time and smaller storage



[18]Stratos Idreos and Tim Kraska. 2019. From Auto-tuning One Size Fits All to Self-designed and Learned Data-intensive Systems (Tutorial). In Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference2019, Amsterdam, The Netherlands, June 30 - July 5, 2019. 2054–2059.

Taxonomy of Learned Indexes



Outline of the Tutorial

- Introduction and Taxonomy
- **Indexing the Learned Models vs. Learning the Indexes**
- Static vs. Dynamic Learned Indexes
- Learned One-Dimensional Indexes
- Learned Multidimensional Indexes
- Open Problems for Future Research

Taxonomy of Learned Indexes

Dimension 1: Indexing the Learned Models vs. Learning the Index

Machine Learning and data indexing interact in two possible ways:

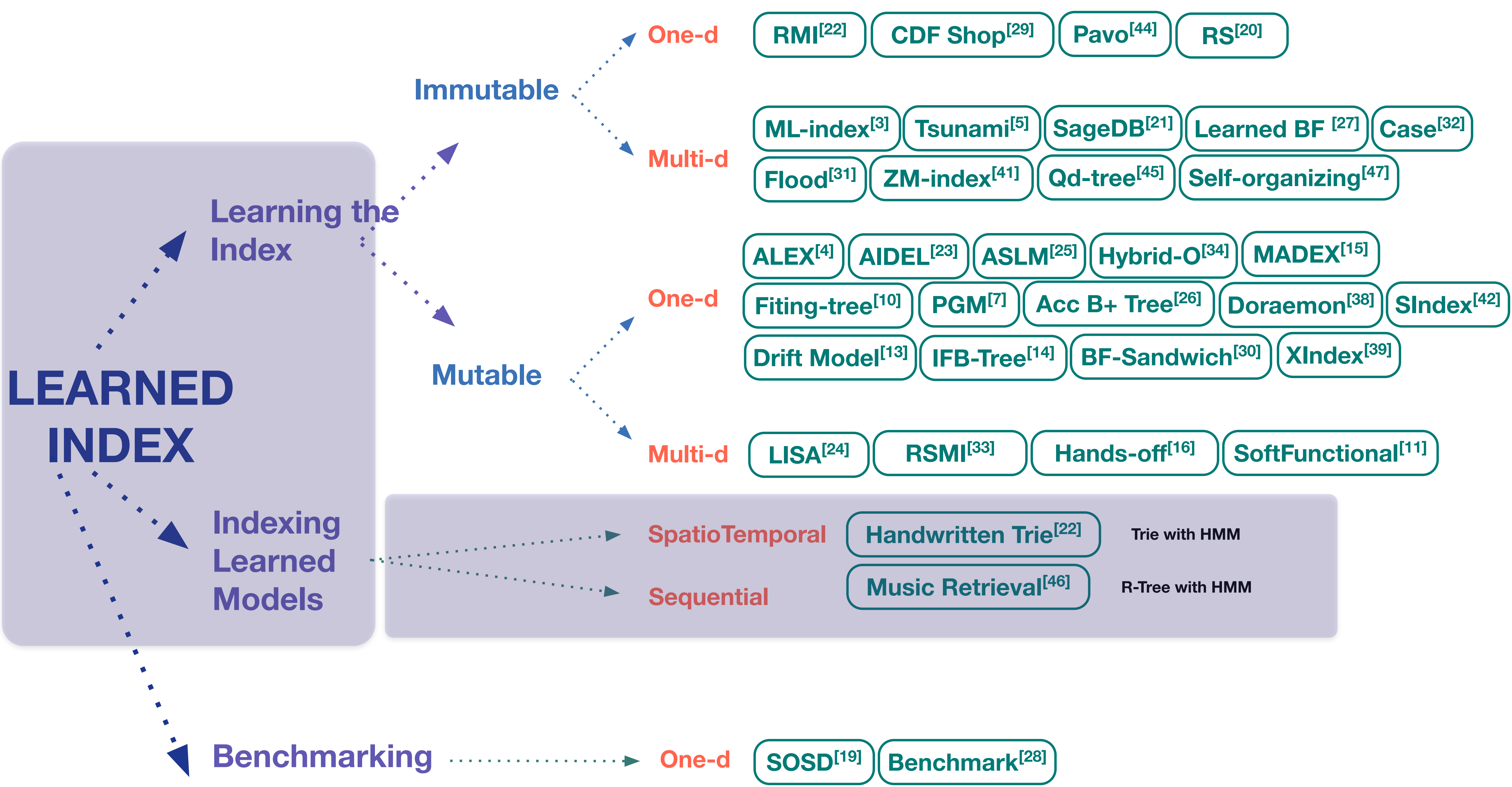
1 Indexing the Learned Models

- Given a collection of ML models, e.g., object recognition models (Cats, Dogs, Trains, etc.), and an input object, say o
- Identify the class of o (cat vs. dog, etc.)
- Instead of executing all models and identifying which has the highest matching score
- Can we index the learned models to speed up the matching process?

2 Learning the Index

- Given a key value, say k, and an ordered array of key values
- Build an ML-based model that helps predict the location of k in the ordered array

Taxonomy of Learned Indexes



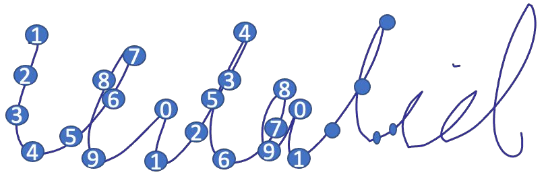
The Handwritten Trie: Indexing Electronic Ink [SIGMOD'95]

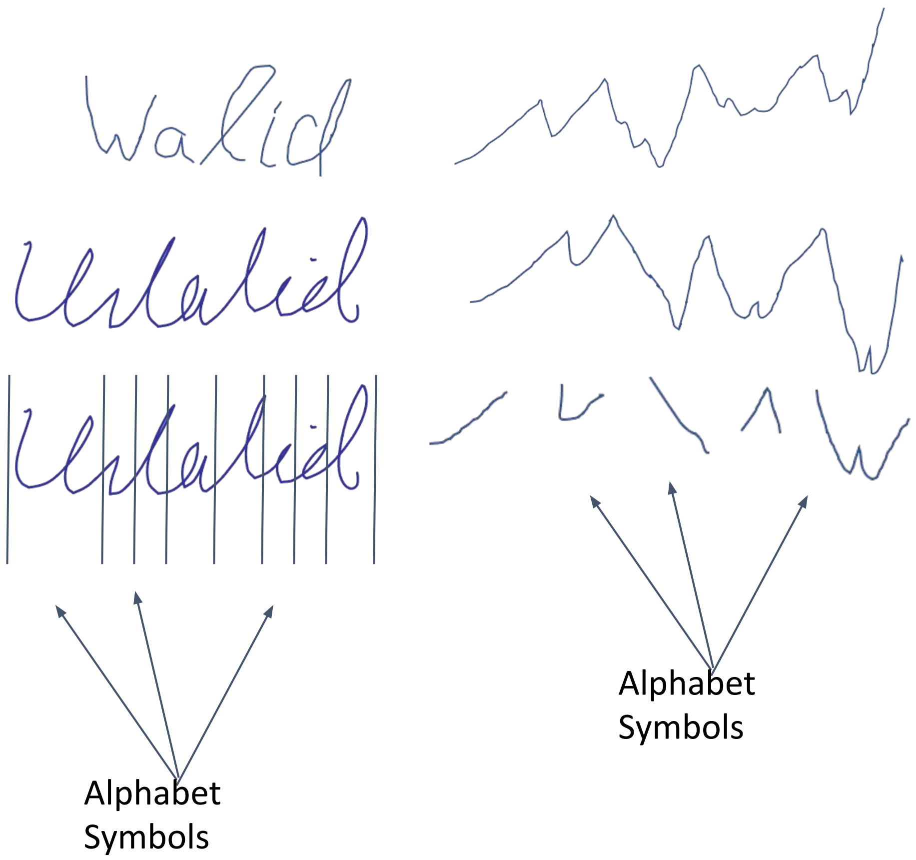
- Collection of spatiotemporal sequences, e.g., heart pulse rates, stock market trends over time, handwritten drawing on a tablet, object movement trajectory
    - Index shown in the context of handwritten text

- Divide each spatiotemporal sequence into basic alphabet symbols

- Because of the variability, there is a need for training to recognize similar, but not exactly the same, patterns

- Model each alphabet symbol in the spatiotemporal sequence using *local spatiotemporal features* along the trajectory of the sequence
    - Time
    - Velocity
    - Direction
    - Acceleration
    - Aspect ratio, . . .

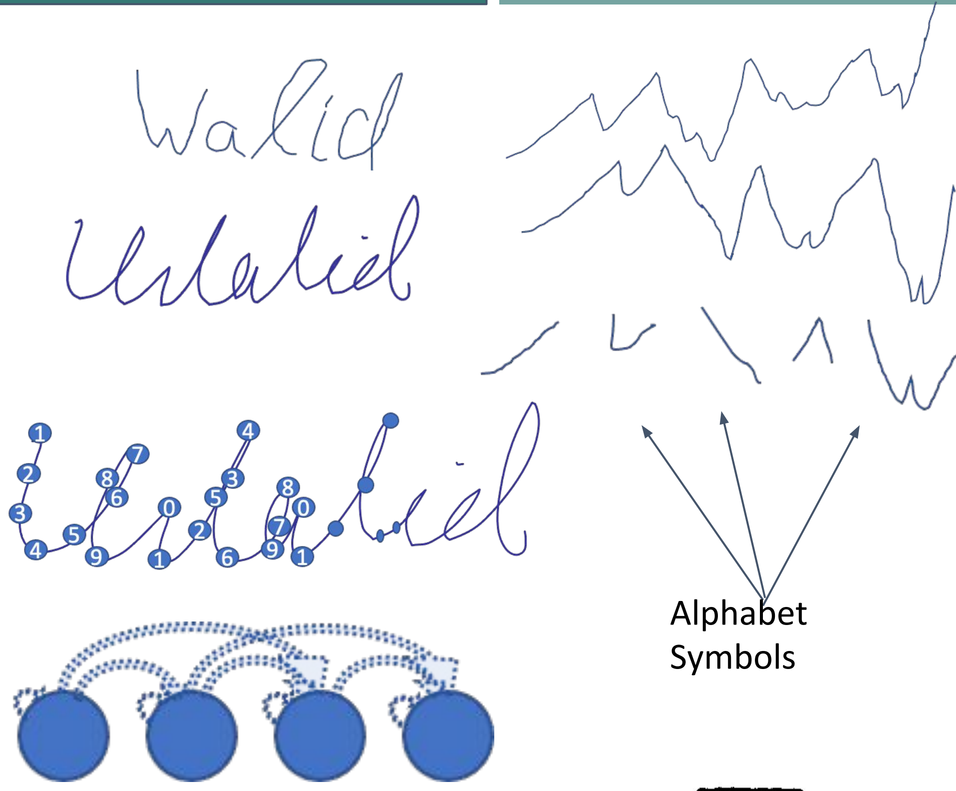




[1]Walid Aref, Daniel Barbará, and Padmavathi Vallabhaneni. 1995. The Handwritten Trie: Indexing Electronic Ink. SIGMOD Rec.24, 2 (May 1995), 151–162. https://doi.org/10.1145/568271.223811

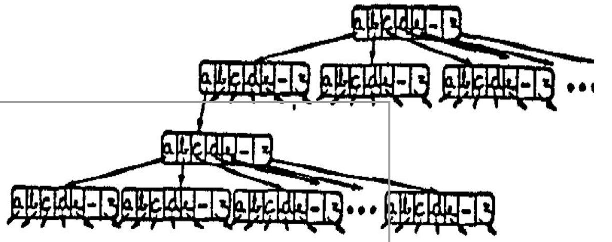
The Handwritten Trie: Indexing Electronic Ink [SIGMOD'95]

- Left-to-right Hidden Markov Models are suitable for representing spatio-temporal sequences
- Instead of building a Hidden Markov Model for each entire sequence, we build an HMM for each alphabet symbol in the spatiotemporal sequence
- Need to segment each spatio-temporal sequence into alphabet symbols
- Train the left-to-right Hidden Markov Model using multiple samples of the alphabet symbols
- Construct a trie structure over the learned alphabet symbols



Core Idea

- Indexing the learned Hidden Markov Models
- Trie Structure over learned alphabet symbols



[1]Walid Aref, Daniel Barbará, and Padmavathi Vallabhaneni. 1995. The Handwritten Trie: Indexing Electronic Ink. SIGMOD Rec.24, 2 (May 1995), 151–162. https://doi.org/10.1145/568271.223811

The Handwritten Trie: Indexing Electronic Ink

Hidden Markov Model

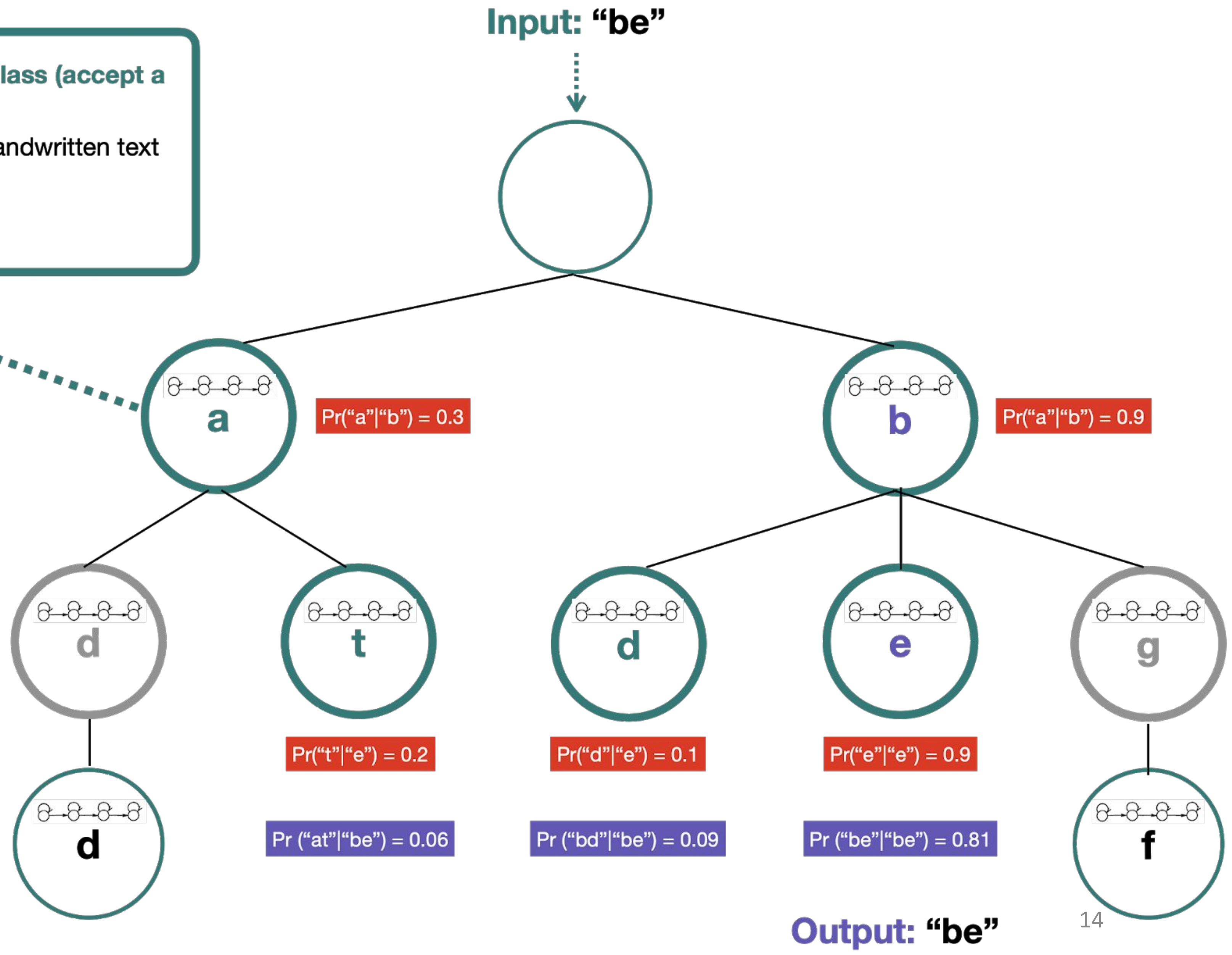
Each HMM is constructed to represent a pictogram class (accept a specific pictogram with high probability)

- **Left-to-right HMM:** Useful for modeling cursive handwritten text
- **Input:** Pictogram
- **Output:** Matching Probability

Traverse Nodes (HMM Models)

Choose Nodes with highest probability

choose combination of nodes with highest probability

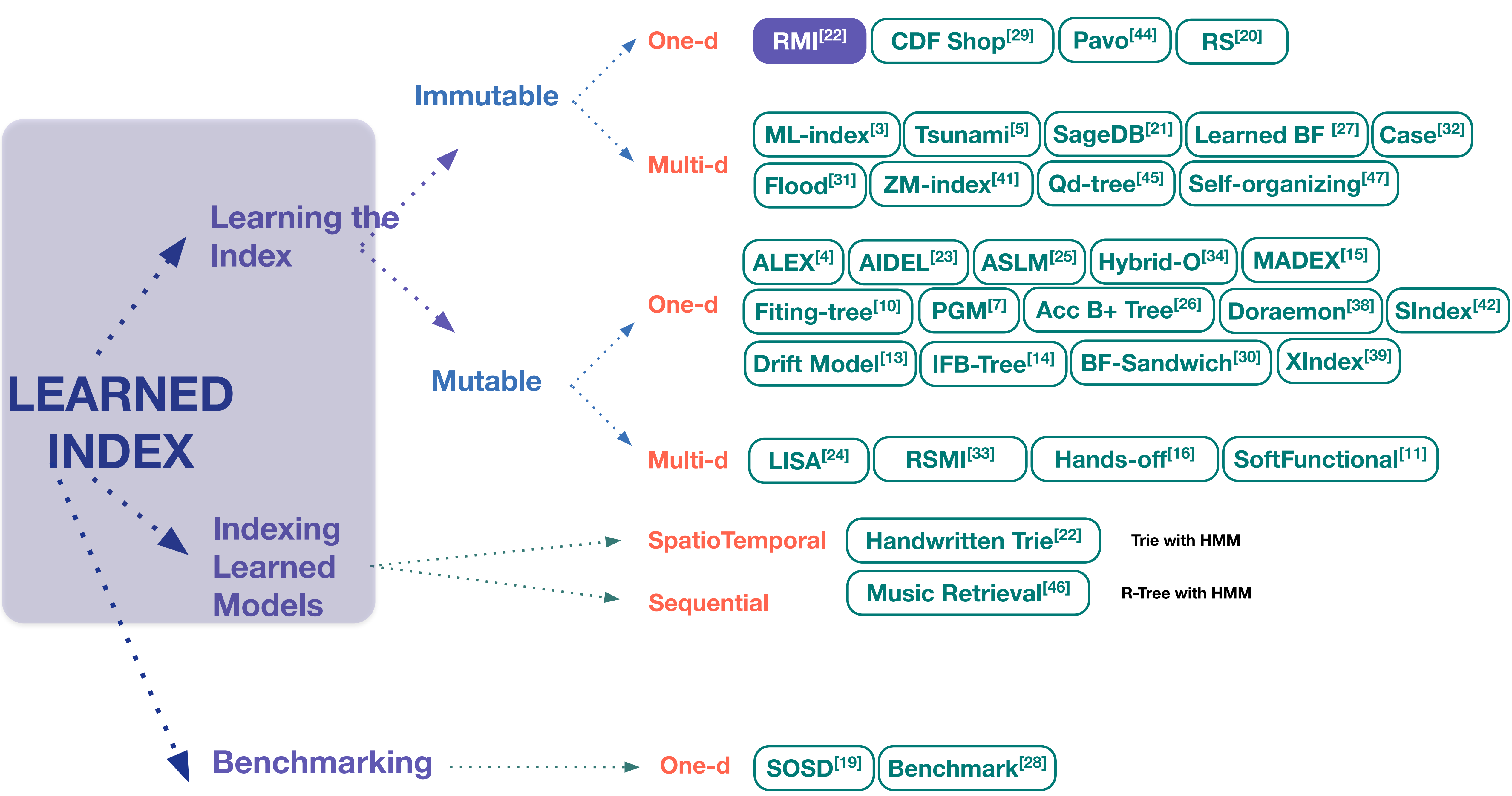


Indexing the Learned Models: Summary

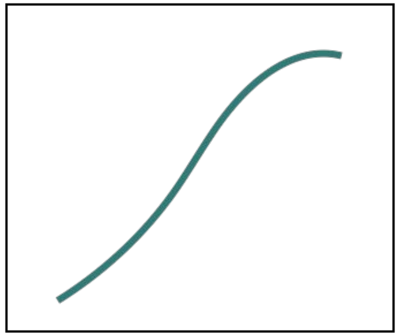
- The Handwritten Trie: Indexing Electronic Ink is one of the earlier works to index the models.
- Another early work about indexing the models using R-Tree-like structure in the context of indexing HMMs for music retrieval can be found in [ISMIR’02]

[2] Jin, Hui, and H. V. Jagadish. "Indexing Hidden Markov Models for Music Retrieval." In International Conference on Music Information Retrieval (ISMIR). 2002.

Taxonomy of Learned Indexes



The Case for Learned Index Structures [SIGMOD'18]



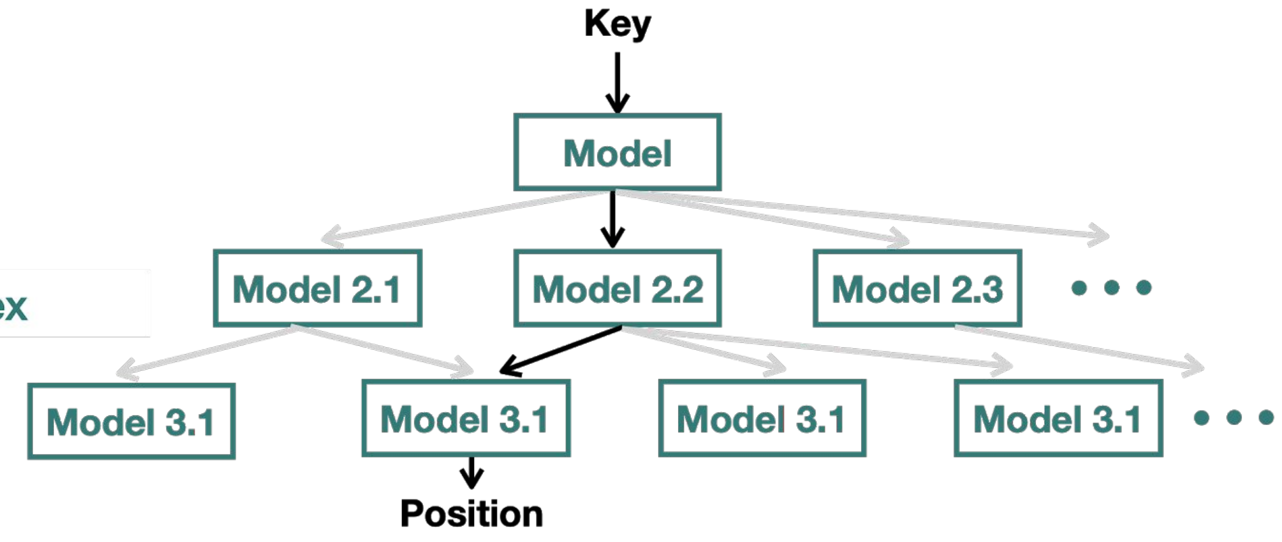
- Introduced the idea that “Indexes are models”
- Replace traditional database indexes by learned models
- Approximate the Cumulative Distribution Function (CDF) of the underlying (sorted) data
- Proposed Recursive Model Index (RMI), a multi-stage ML model
- Combine simpler ML models
  - The first stage model will make an initial prediction of the CDF for a specific key
  - The next stage models will be selected to refine this initial prediction
- Proposed Learned Index Structures: Range Index, Point Index, and Existence Index

[22]Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018.The case for learned index structures. In Proceedings of the 2018 International Conference on Management of Data. ACM, 489–504.

The Case for Learned Index Structures [SIGMOD'18]

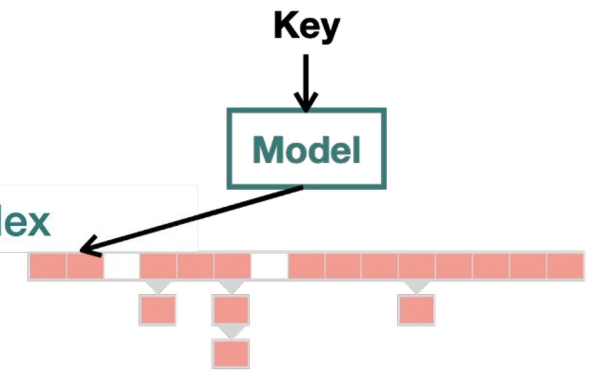
Mechanism

Range Index



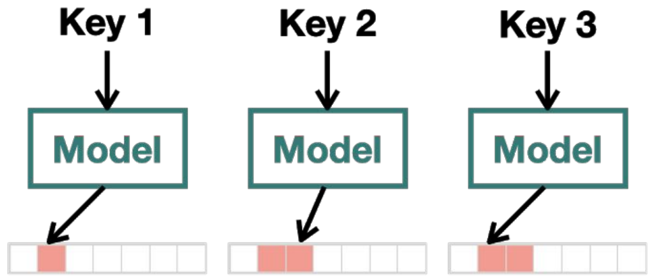
Recursive Model Index

Point Index

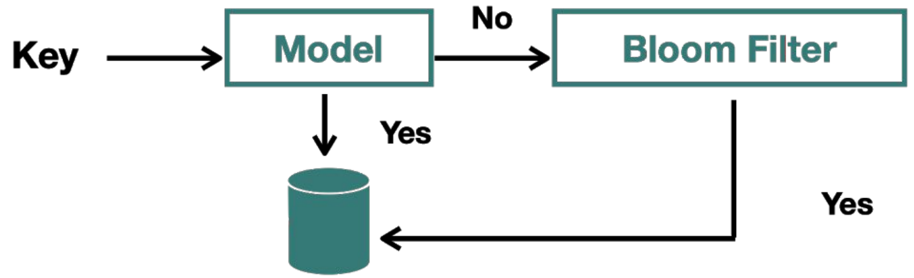


Learned Hash-map

Existence Index



Bloom Filter with Model-Hashes



Bloom Filter as Classification Problem

The Case for Learned Index Structures [SIGMOD'18]

- Limitations:
  - Focus on in-memory read-only workloads
  - The structure of RMI is static
  - Does not support updates (e.g., insertion, deletion)
- Many follow up works extend on this paper

Outline of the Tutorial

- Introduction and Taxonomy
- Indexing the Learned Models vs. Learning the Indexes
- **Static vs. Dynamic Learned Indexes**
- Learned One-Dimensional Indexes
- Learned Multidimensional Indexes
- Open Problems for Future Research

Taxonomy of Learned Indexes

Dimension 2: Immutable vs. Mutable Learned Indexes

Given a Learned Index, can we support updates?

Why are updates hard?

- The learned index takes significant time to train
- New data will require retraining because it changes the order of the data

Classify learned indexes based on the ability to support updates:

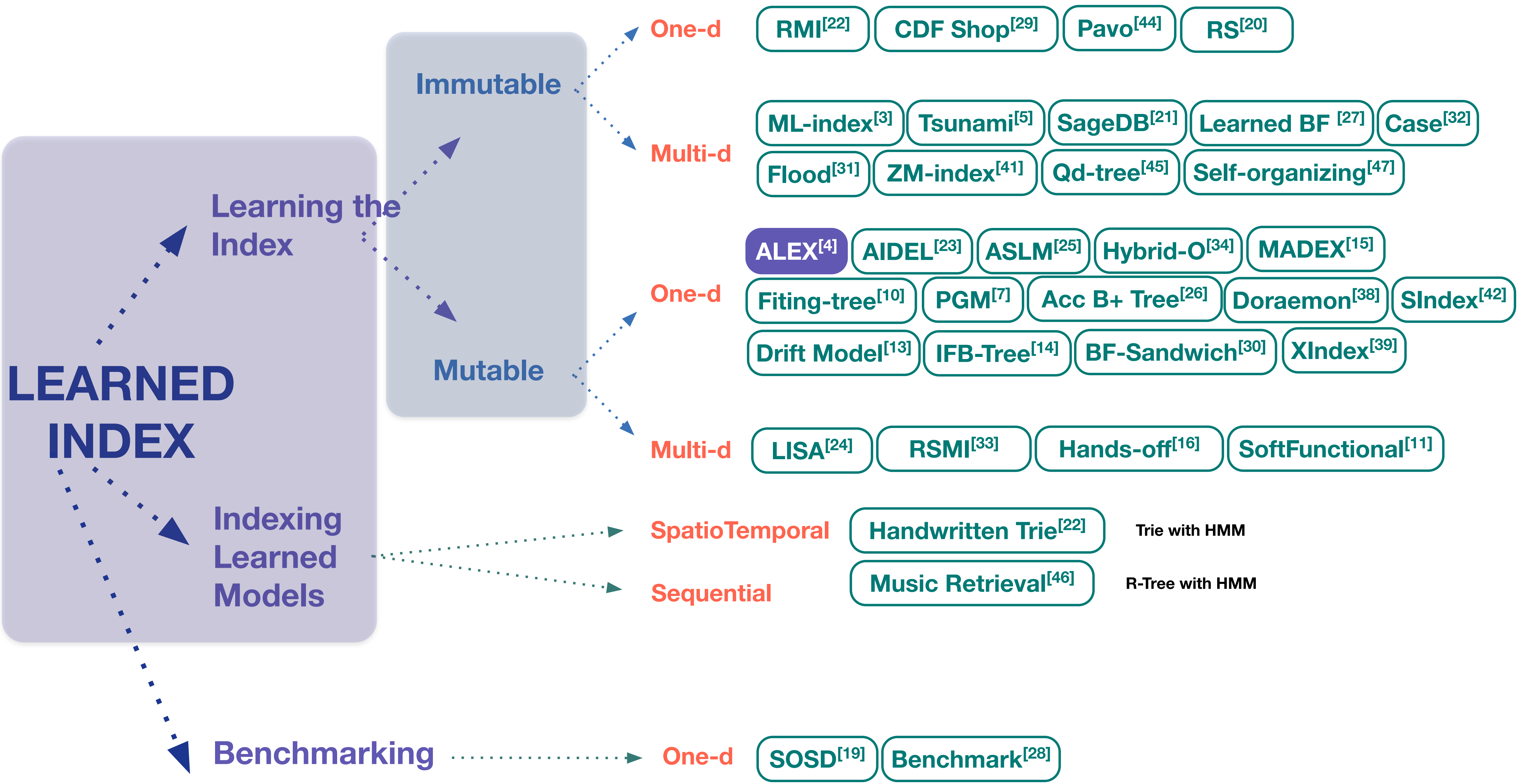
1 *Immutable Learned Index*:

- Does not support inserts, updates, or deletes

2 *Mutable Learned Index:*

- Supports inserts, updates or deletes

Taxonomy of Learned Indexes

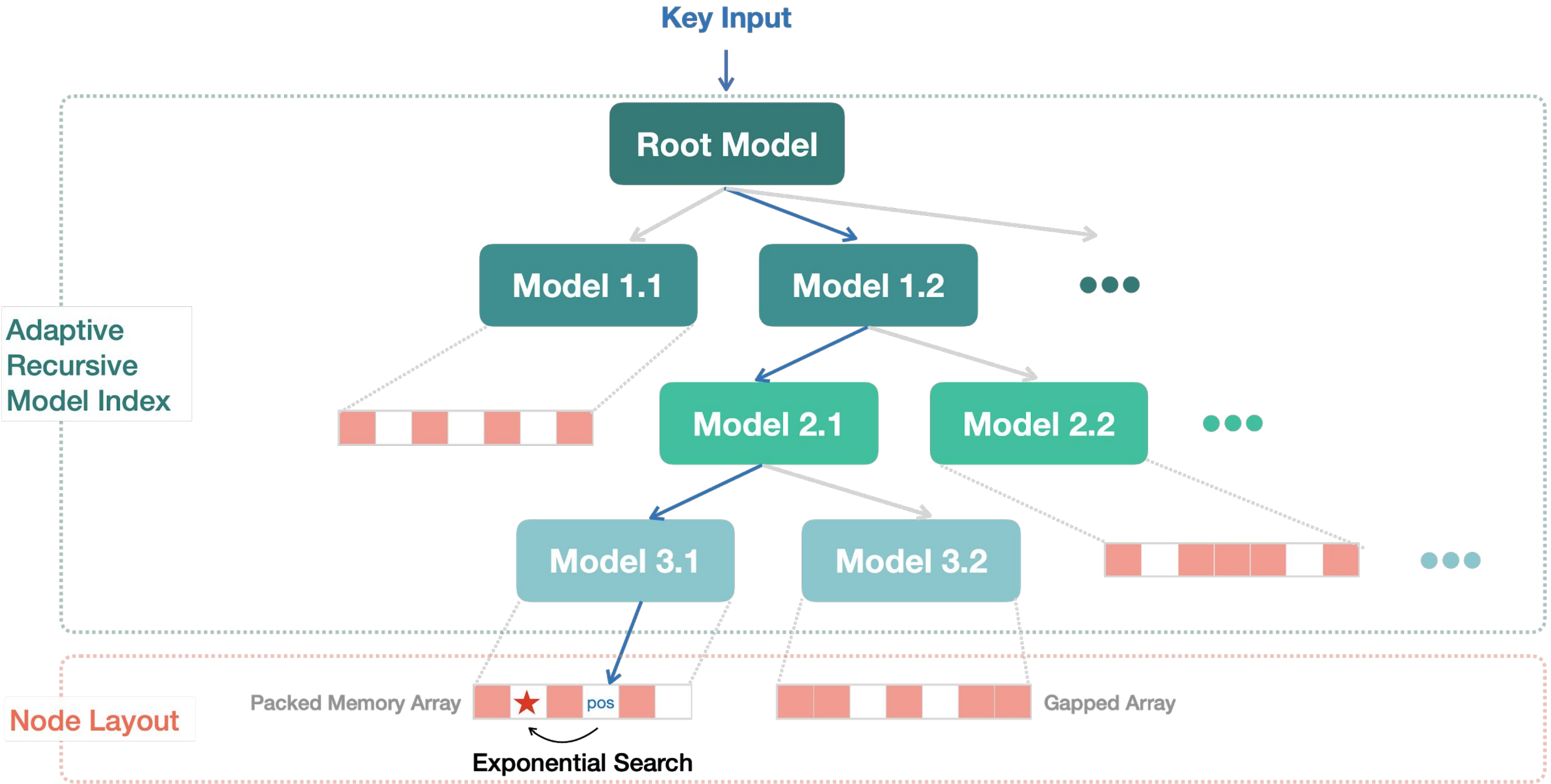


ALEX: An Updatable Adaptive Learned Index [SIGMOD'20]

Target	Learned Index (Stable & Dynamic Workload)
Objectives	Implement dynamic, updatable learned index to handle dynamic workload
Core Idea	• Adaptive RMI as Model Hierarchy • Linear Regression Model as Node • Gapped Array or Packed Memory Array as Node Layout

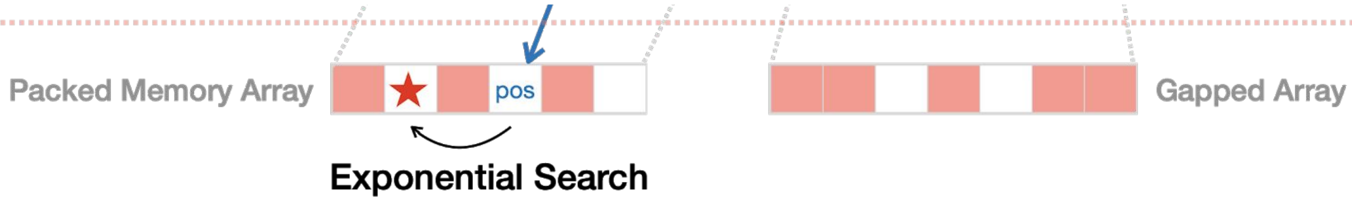
[4] Jialin Ding, Umar Farooq Minhas, Hantian Zhang, Yinan Li, Chi Wang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, and David Lomet. SIGMOD 2020. ALEX: An Updatable Adaptive Learned Index. arXiv preprint arXiv:1905.08898(2019).

ALEX: An Updatable Adaptive Learned Index [SIGMOD'20]



ALEX: An Updatable Adaptive Learned Index [SIGMOD'20]

Node Layout



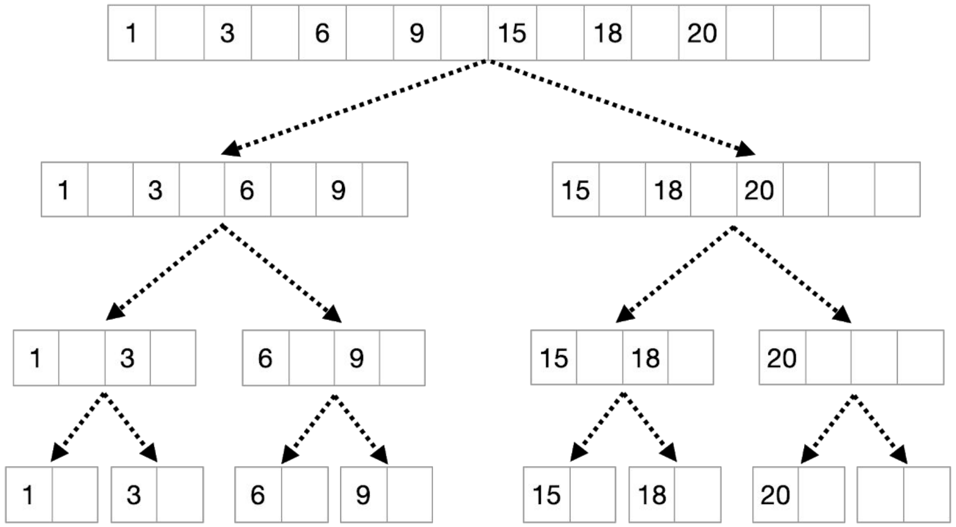
1 Gapped Array (GA)

	1	2	3	5		7	8		11

2 Packed Memory Array (PMA)

1		3		6		9		15		18		20			

Binary Tree Representation of PMA

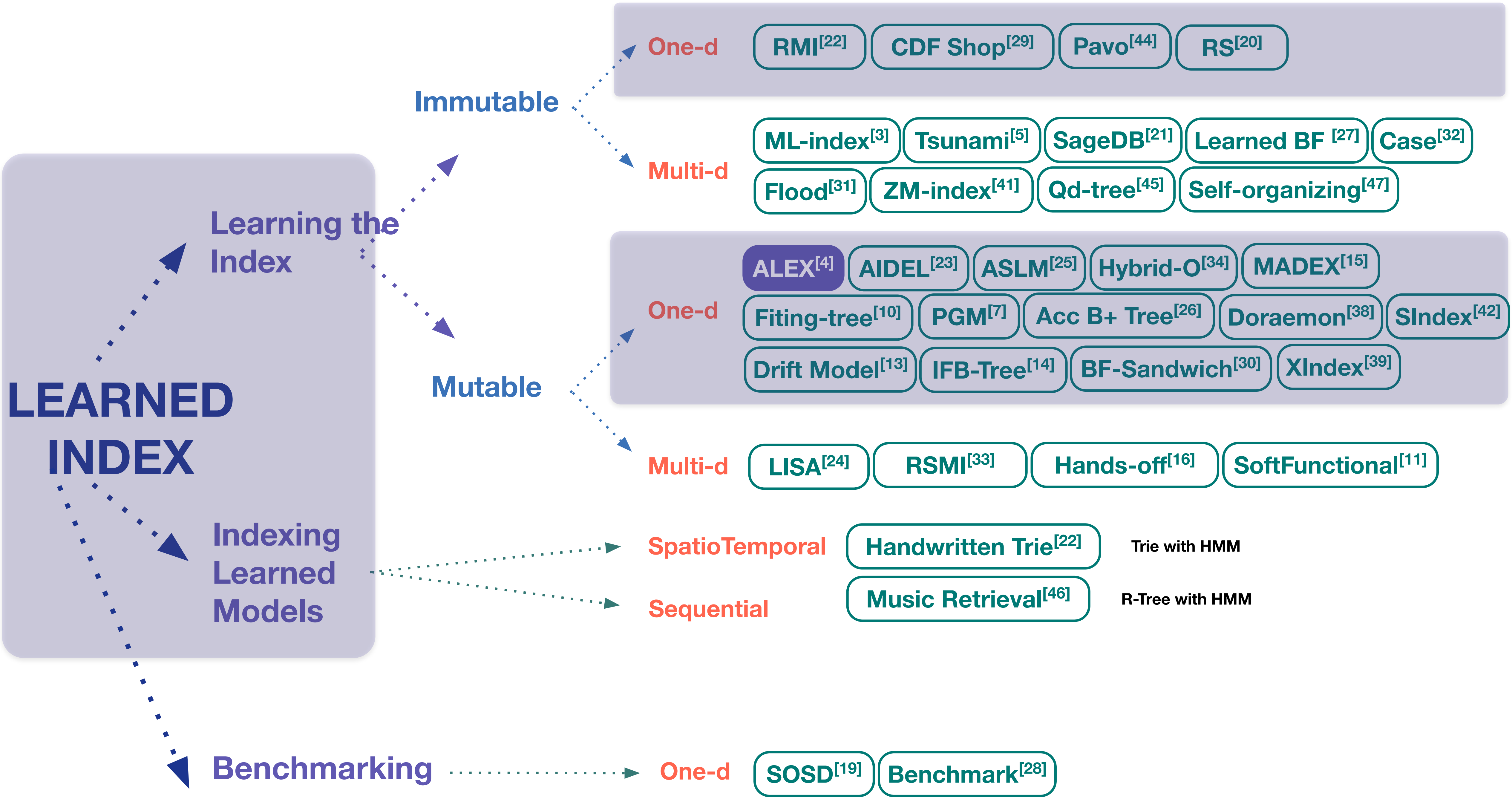


- Limitations:
  - Adversarial workload when data is skewed
  - ALEX does not support duplicate keys of secondary indexes
  - Requires concurrency control to handle updates with concurrent lookups
  - How to check sorted order during insertion in gapped array (linear search?)

Outline of the Tutorial

- Introduction and Taxonomy
- Indexing the Learned Models vs. Learning the Indexes
- Static vs. Dynamic Learned Indexes
- **Learned One-Dimensional Indexes**
- Learned Multidimensional Indexes
- Open Problems for Future Research

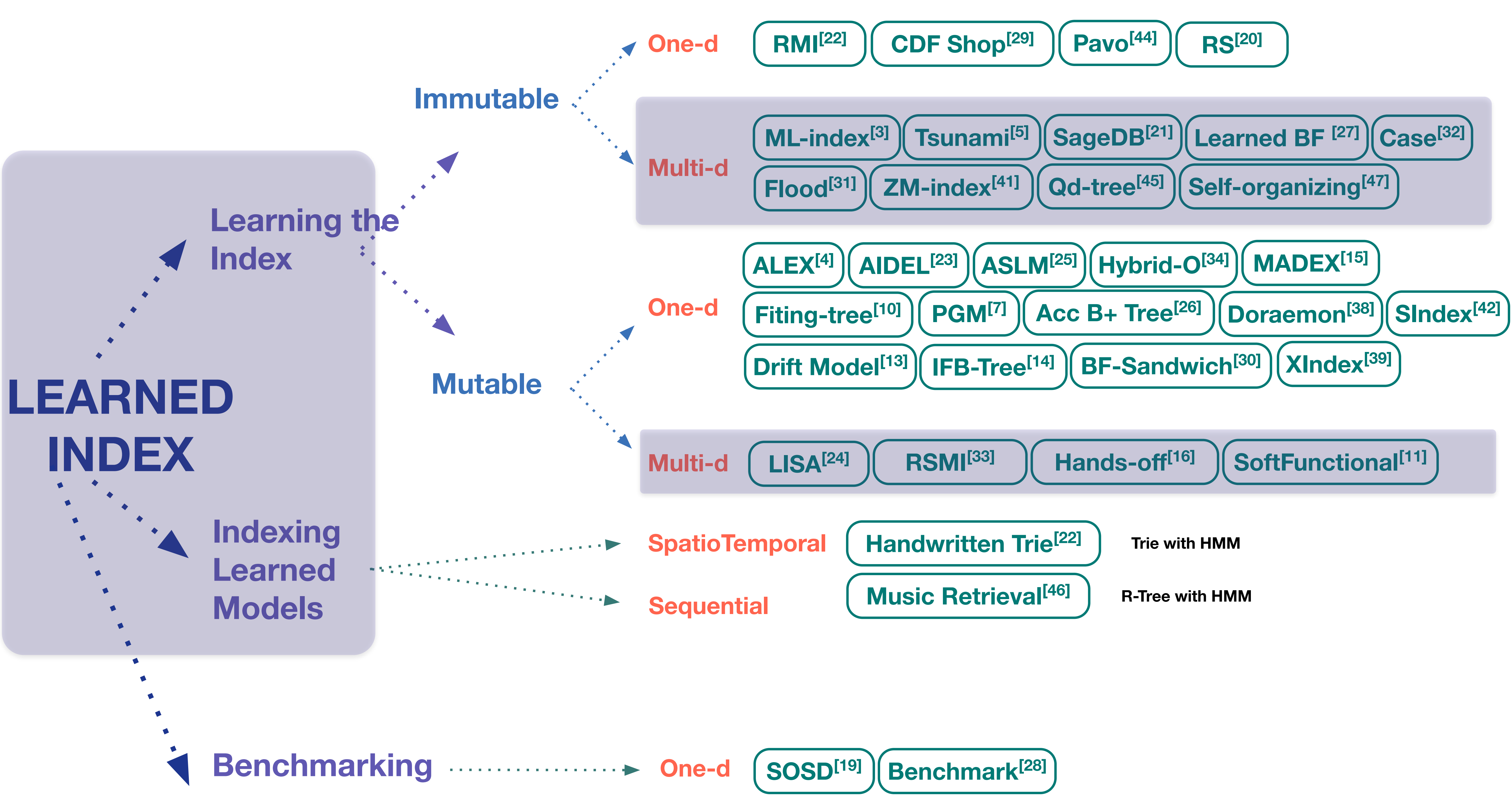
Taxonomy of Learned Indexes



Outline of the Tutorial

- Introduction and Taxonomy
- Indexing the Learned Models vs. Learning the Indexes
- Static vs. Dynamic Learned Indexes
- Learned One-Dimensional Indexes
- **Learned Multidimensional Indexes**
- Open Problems for Future Research

Taxonomy of Learned Indexes



Background

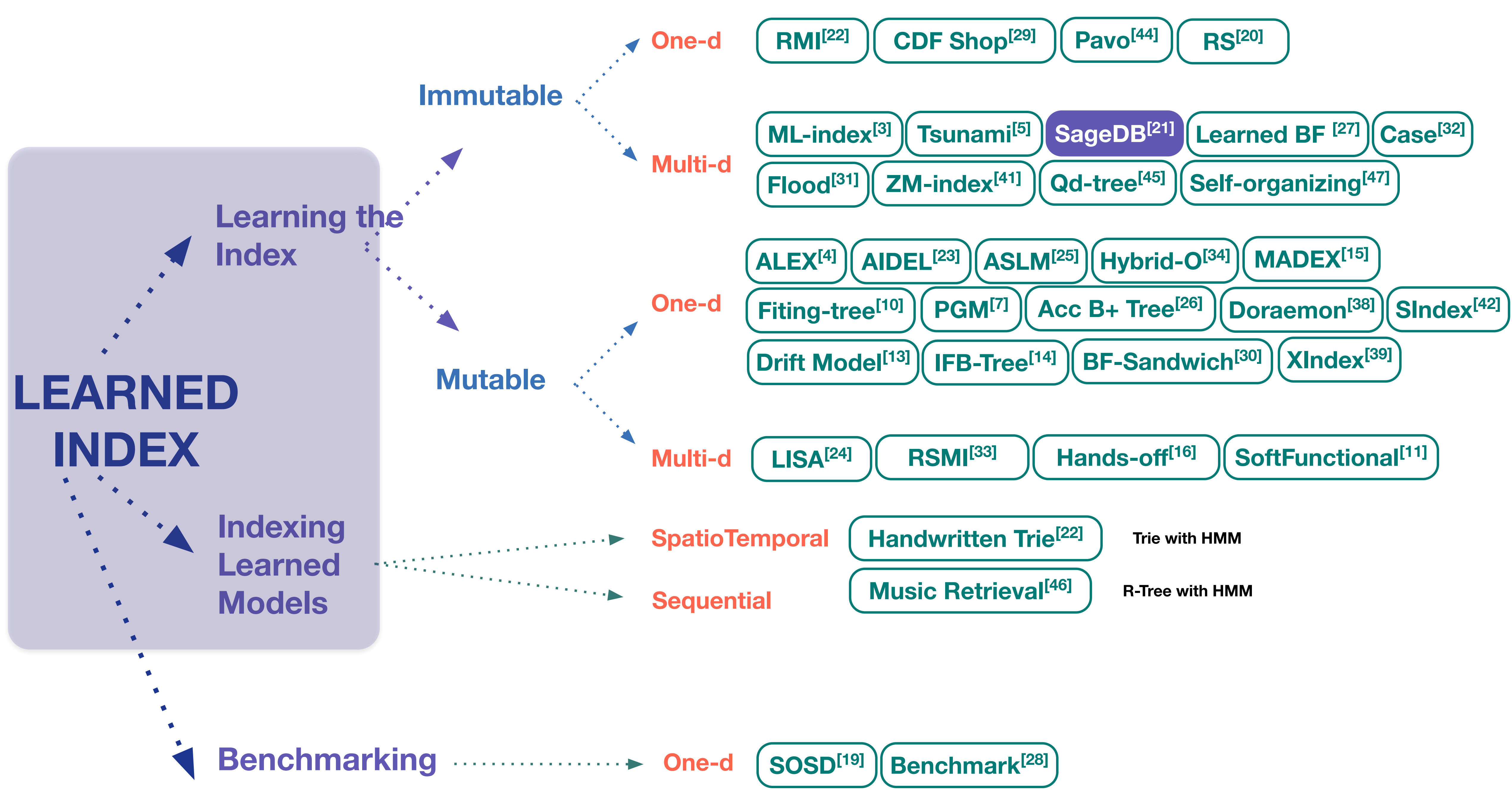
- One of the earliest distribution-aware spatial indexes can be found in:
  - [47] Babu, G. Phanendra. "Self-organizing neural networks for spatial data." Pattern Recognition Letters 18, no. 2 (1997): 133-142.

- Can ML models replace and act in place of a multi-dimensional index?
  - Yes, ML models can act in place of a multi-dimensional index, e.g., an R-Tree

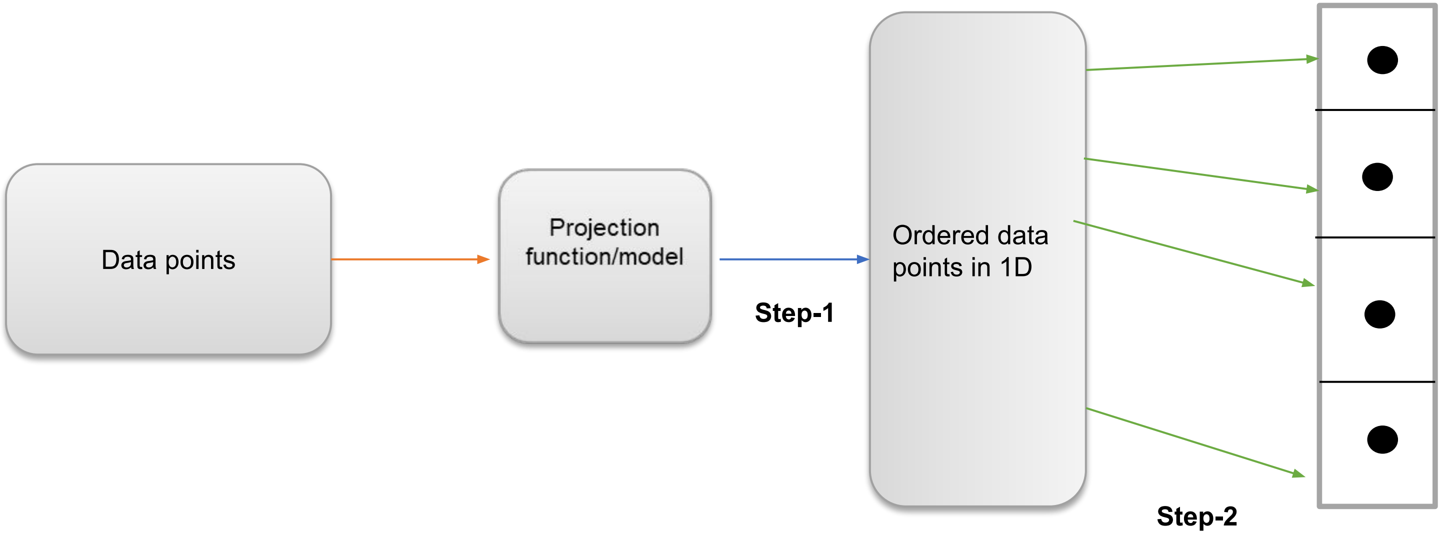
Challenges for Learned Multidimensional Indexes

- *Sorting/ordering of multi-dimensional data:*
  - No obvious sort order for multi-dimensional data
- *Error correction mechanism in case of misprediction:*
  - Difficult to define an error correction mechanism in case of mispredictions
- *Choice of the ML model:*
  - Which ML models to choose?
- *Layout of the data:*
  - How to store the data?
  - Affect range query time and model accuracy

Taxonomy of Learned Indexes



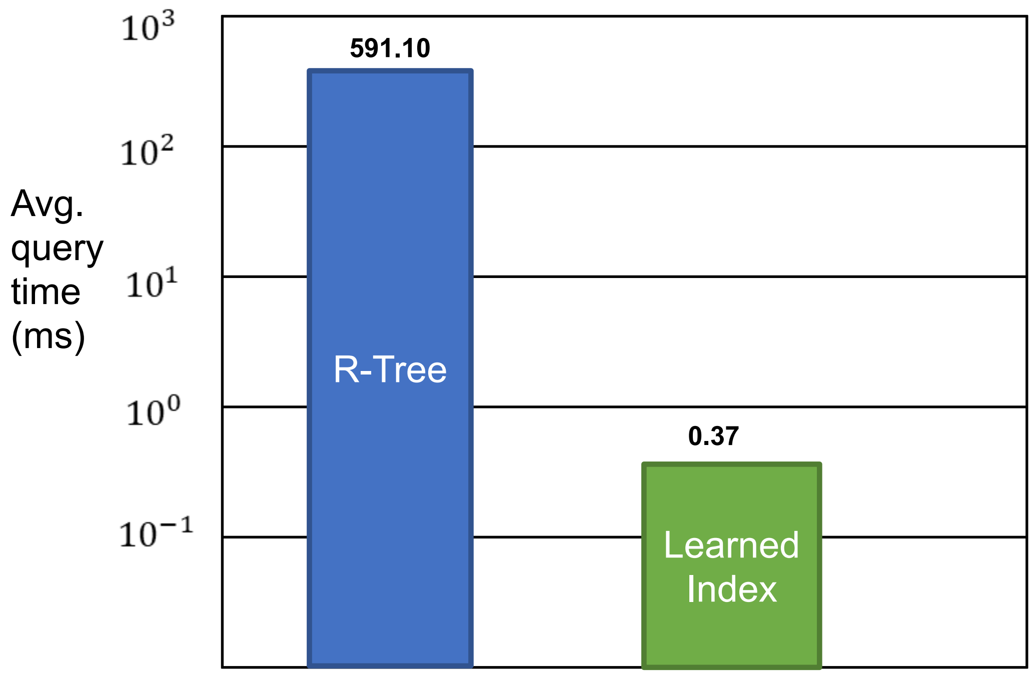
- Proposal:
  - *Step-1:* Project the multi-dimensional data points into one-dimensional space
    - Successively sorting and partitioning points along a sequence of dimensions into equally-sized cells
      - Produces a layout that is efficient to compute and learnable
        - Comparing with Z-order which is difficult to learn
  - *Step-2:* Uses a trained CDF model (e.g., RMI) to predict the physical location of the point



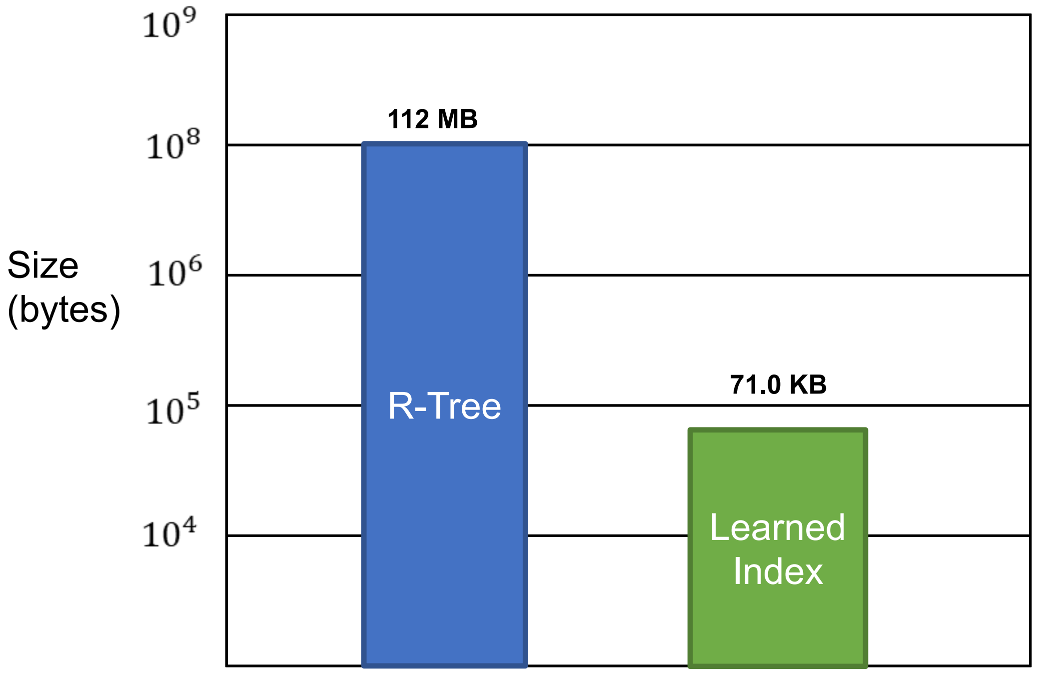
[21]Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H Chi, Jialin Ding, Ani Kristo,Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019.Sagedb: A learned database system. (2019).

Sagedb: A Learned Database System [CIDR'19]

- Initial Result:
  - R-Tree vs. Learned Multi-dimensional Index on TPC-H data



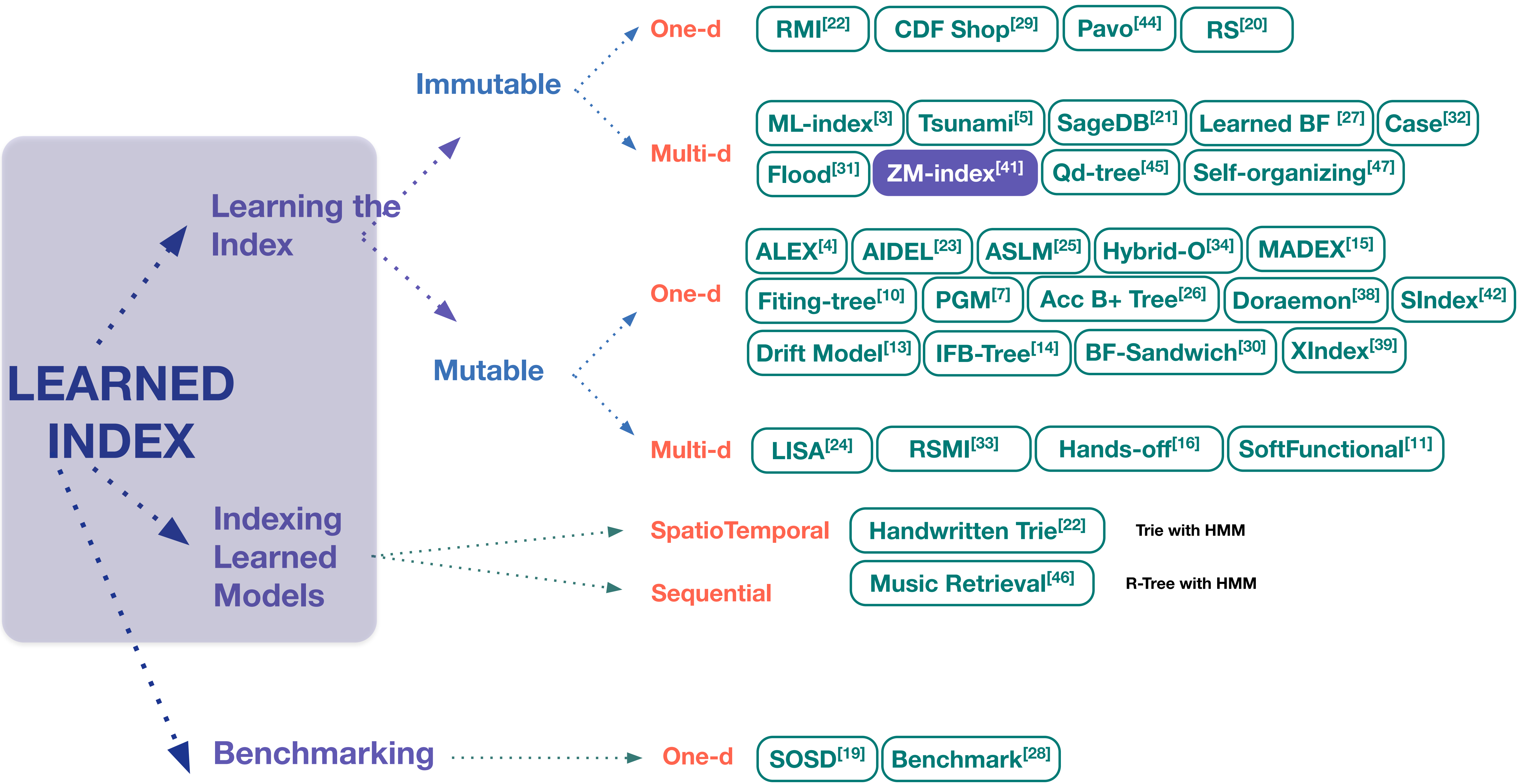
Result on average query time



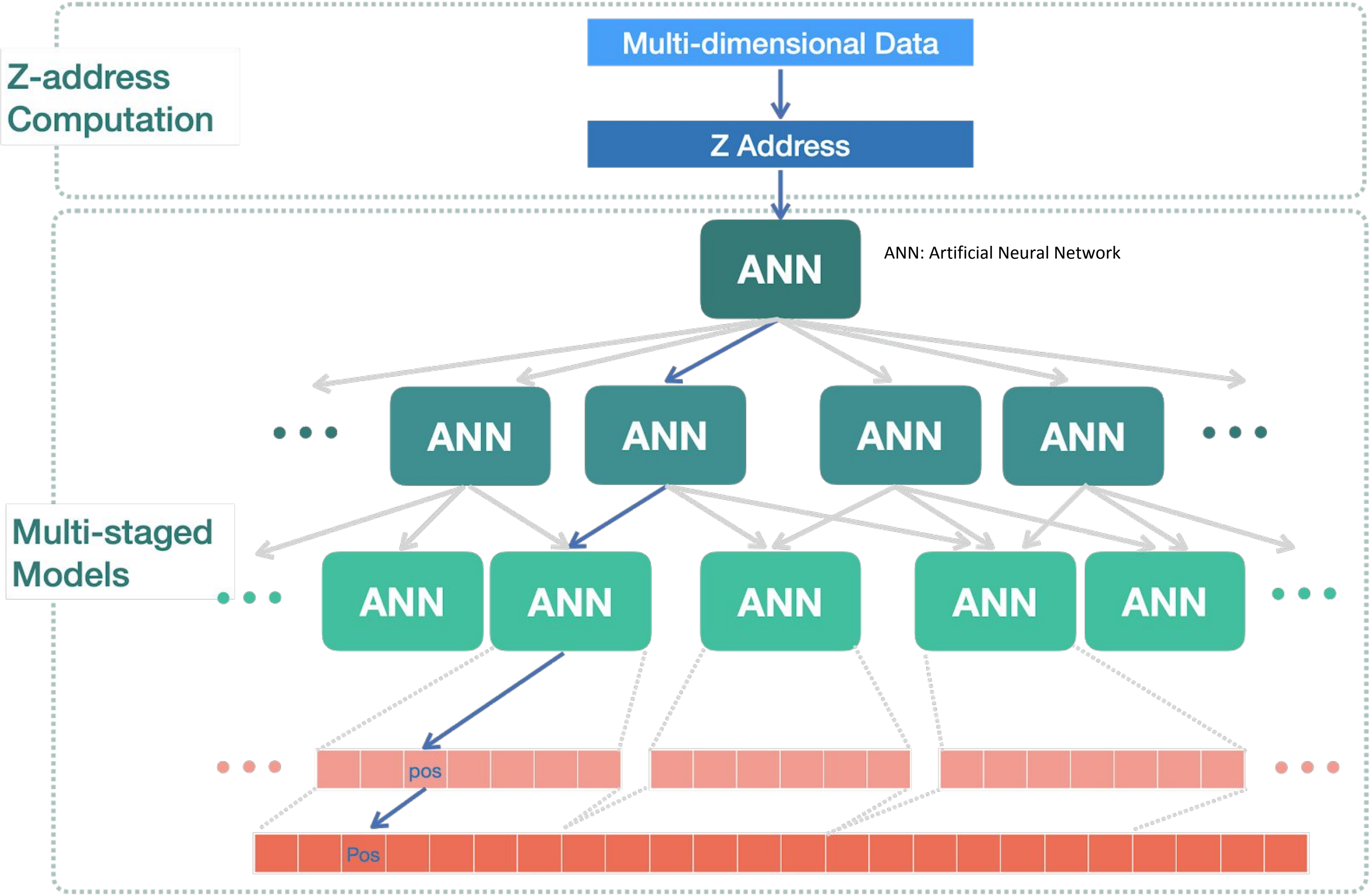
Result on index size

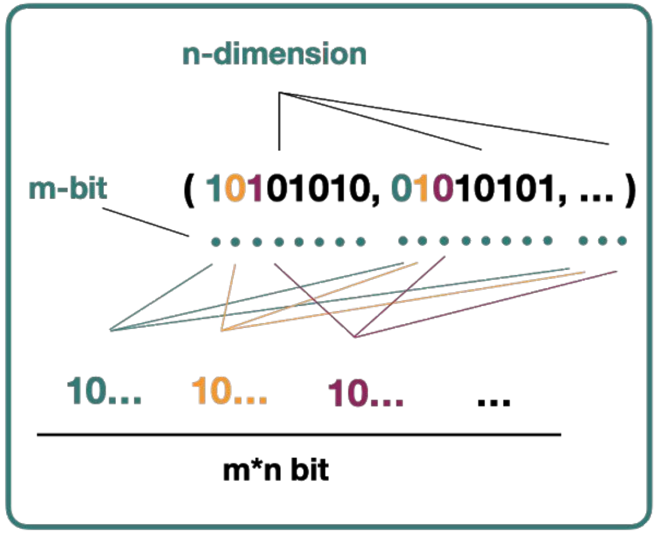
[21]Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H Chi, Jialin Ding, Ani Kristo,Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019.Sagedb: A learned database system. (2019).

Taxonomy of Learned Indexes



Learned Index for Spatial Queries [MDM'19]





Z-address computation of n-d vector

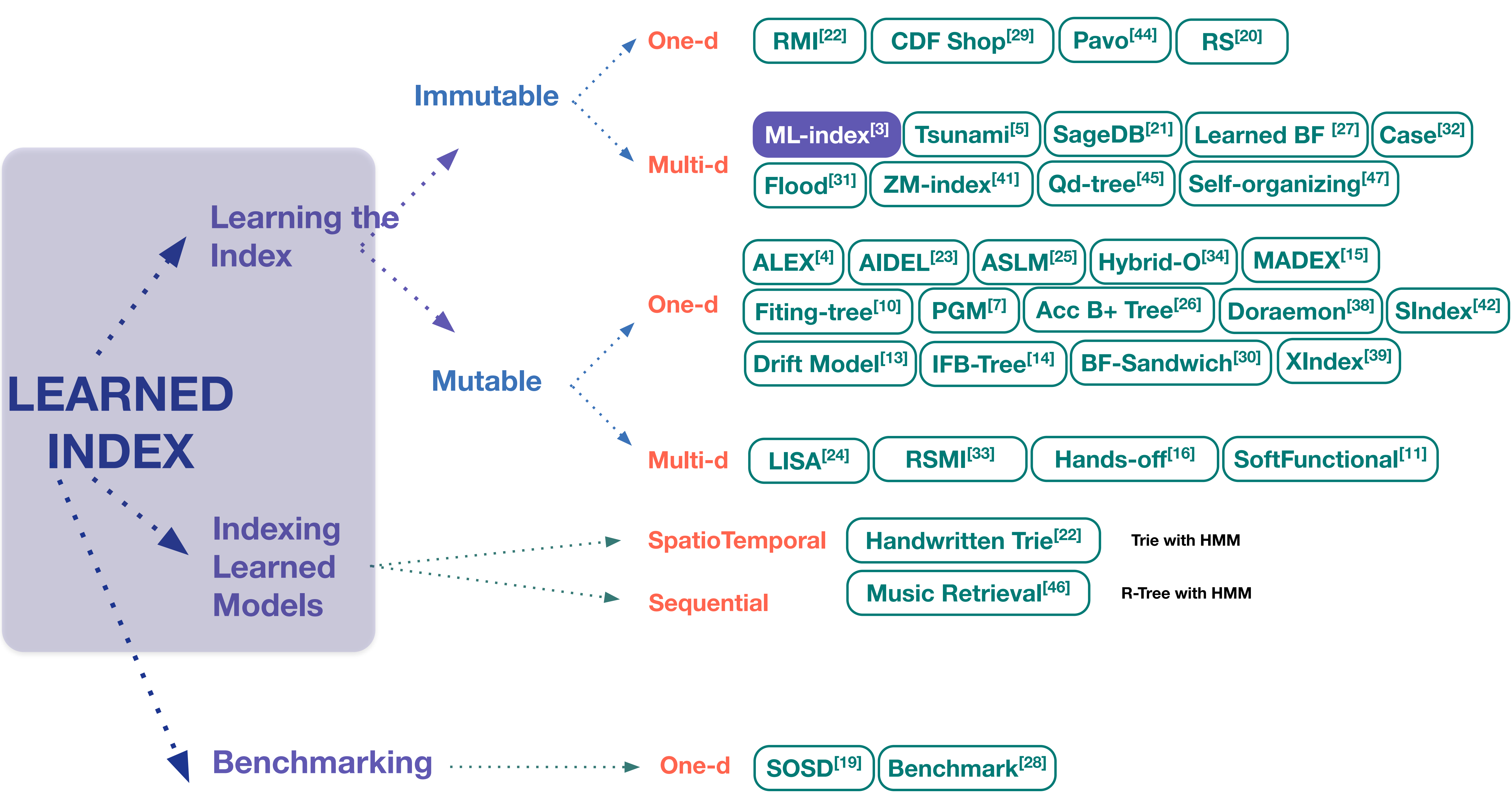
Core Idea

ZM-Index:

- Spatial Query Processing: Point and range queries
- Uses the Z-order to map the multi-dimensional values to the one-dimensional space
- Uses a multi-staged model (e.g., RMI) for learning

[41]Haixin Wang, Xiaoyi Fu, Jianliang Xu, and Hua Lu. 2019. Learned Index for Spatial Queries. In2019 20th IEEE International Conference on Mobile Data Management(MDM). IEEE, 569–574.

Taxonomy of Learned Indexes

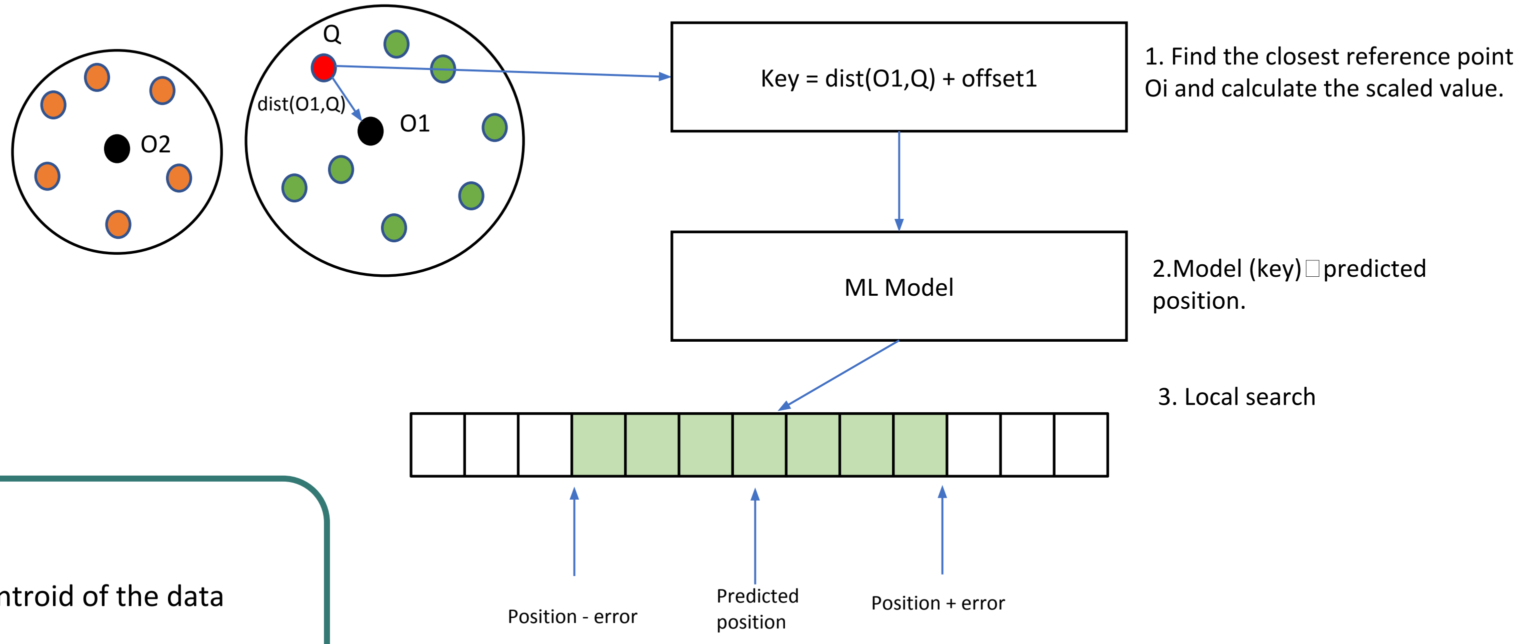


The ML-Index:A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries [EDBT'20]

Core Idea

ML-Index:

- Z/Morton order cannot be easily learned by ML models.
- Multi-dimensional data should be sorted in an order that can be easily learned.
- Partition and transform the data into one-dimensional values based on distribution-aware reference points.
- Combines the scaled ordering with ML models



Query Processing (Point)

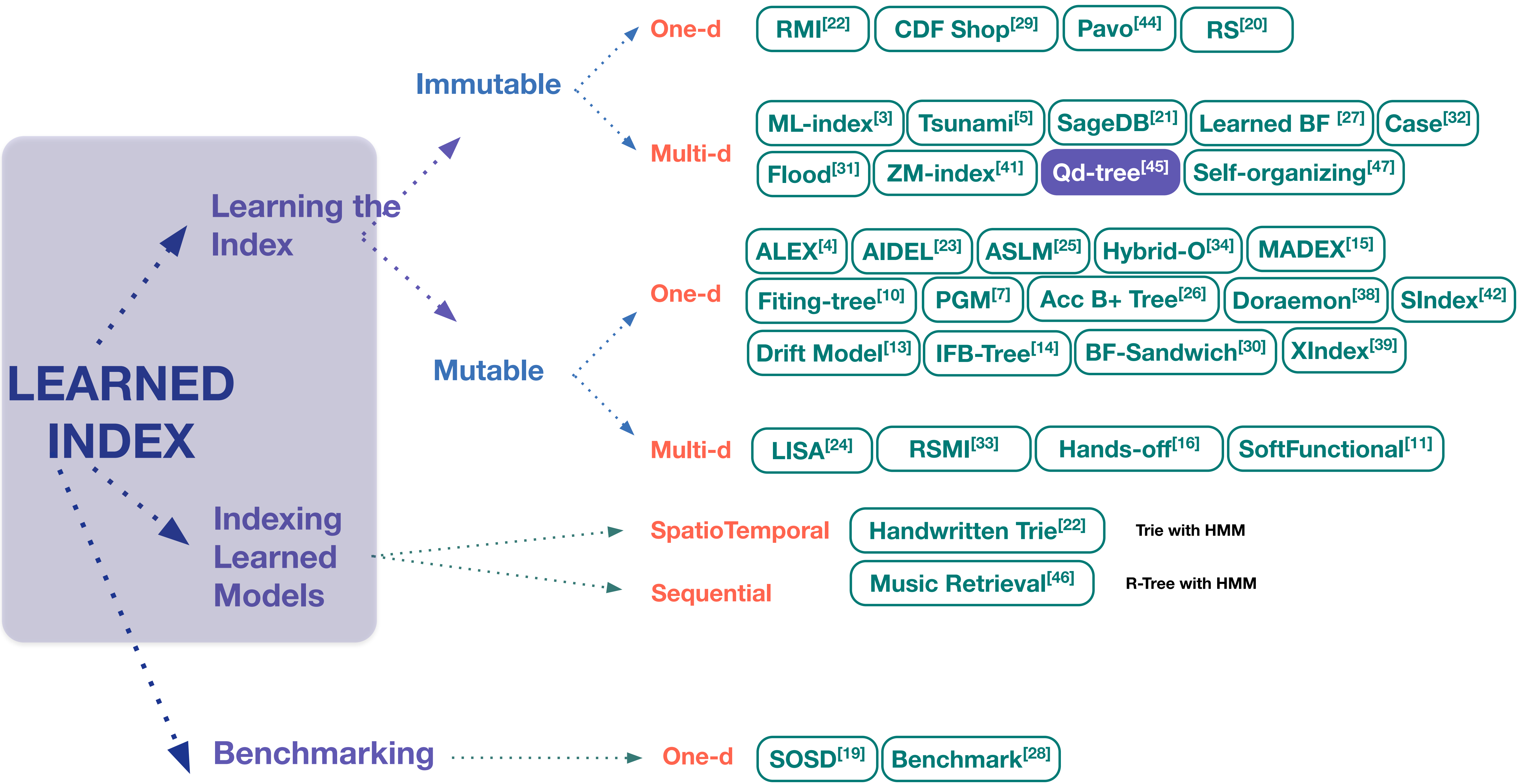
Efficient Scaling

Offset Method:

- *m* reference points *Oi* are chosen each can be thought as a centroid of the data partition *Pi*.
- The closest reference points of *Oi* are used to build the partition *Pi*.
- The minimal distance of a point to the reference points is *dl*
- Scaled value = *offset$_i$* + *dist(O$_i$, dl)*
- For reference points *O1, O2,…Om* and their partitions *P1,P2,…Pm* $offset_i = \sum_{j<i} r(j)$
- r: The maximal distance from *Oj* to the points in partition *Pj*

[3]Angjela Davitkova, Evica Milchevski, and Sebastian Michel. 2020. The ML-Index:A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries.. In EDBT. 407–410.

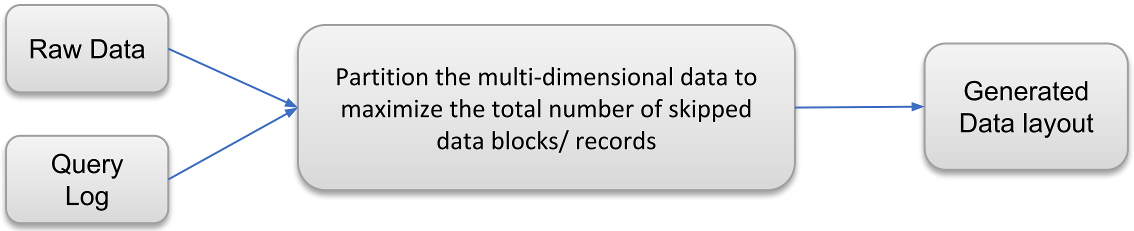
Taxonomy of Learned Indexes



Qd-tree: Learning Data Layouts for Big Data Analytics [SIGMOD'20]

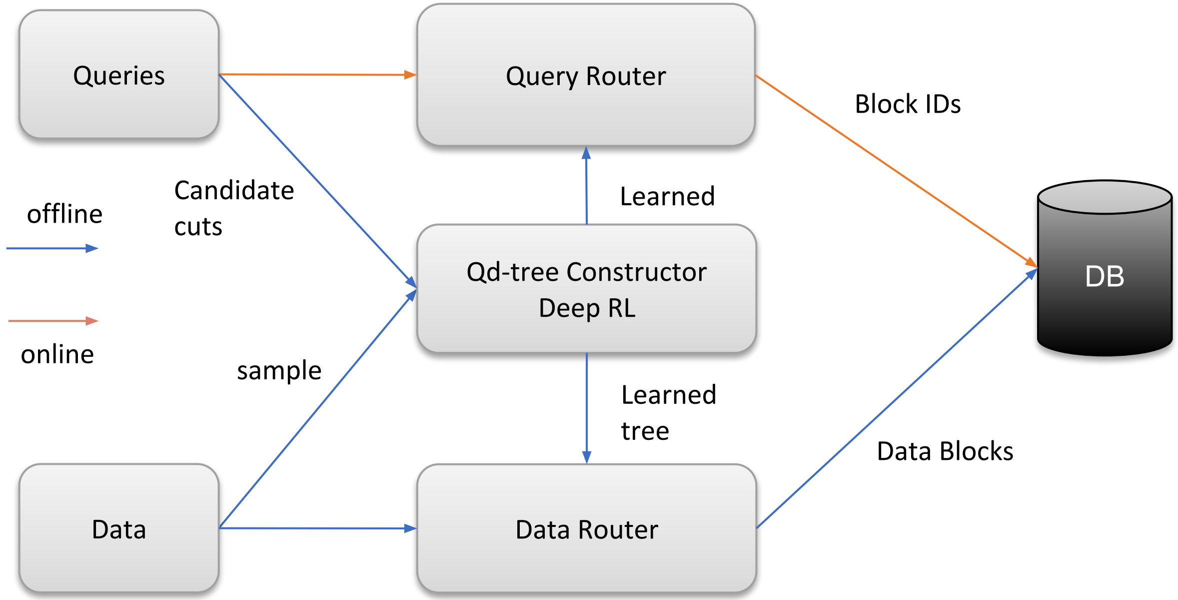
Motivation

- For disk-based systems, an important performance metric is:
  - The number of data blocks accessed by a query.
- Problem Statement:



Core Idea

- Minimize the number of blocks/records accessed by a workload
  - Generating block-level layouts with excellent I/O performance
- Query-data routing trees (Qd-trees) are neural network-generated decision trees that
  - Recursively partition the data space into smaller subspaces.
- Use Deep Reinforcement Learning to create Qd-trees
  - Proximal Policy Optimization (PPO)



System Architecture

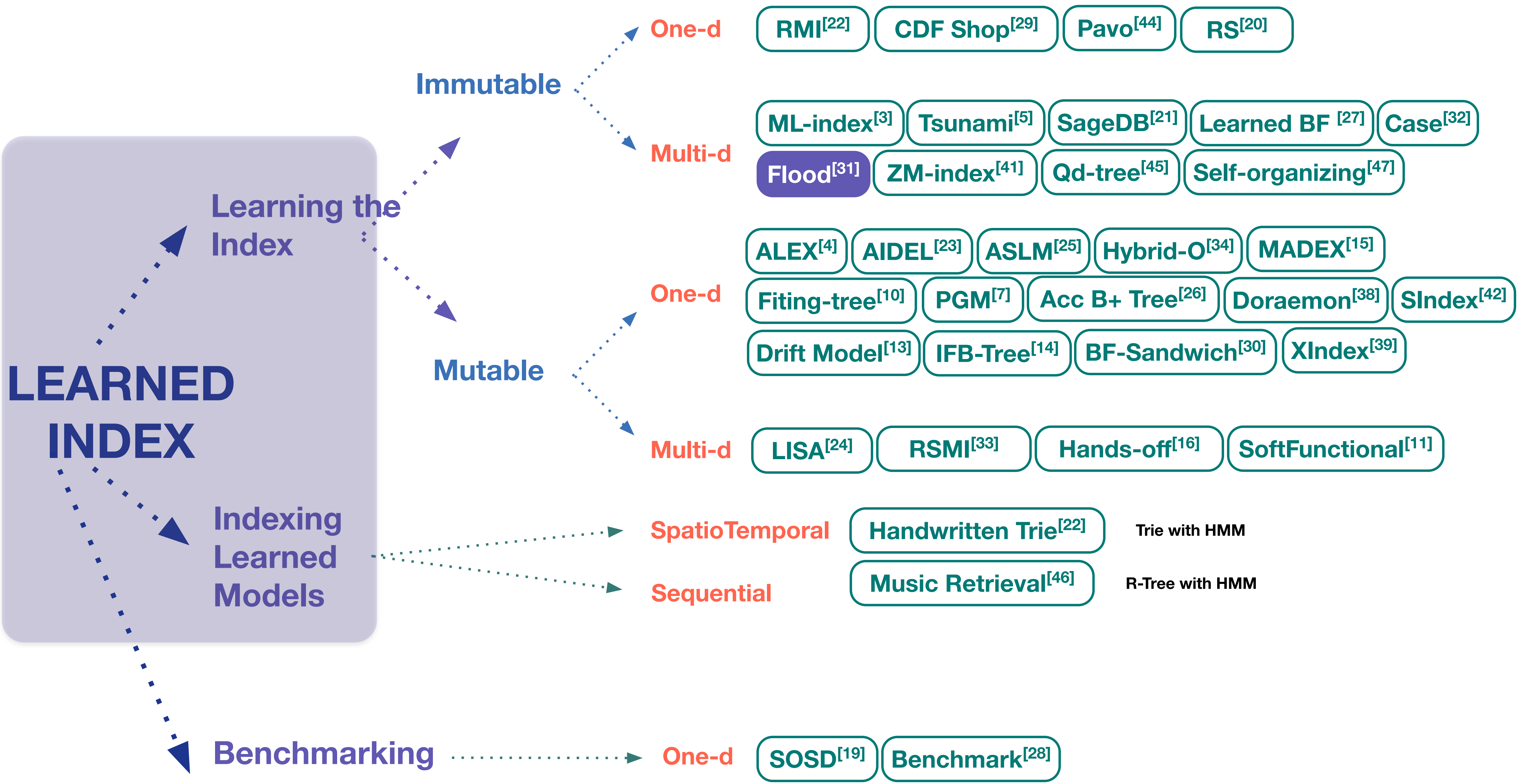
[45]Zongheng Yang, Badrish Chandramouli, Chi Wang, Johannes Gehrke, Yinan Li,Umar Farooq Minhas, Per-Åke Larson, Donald Kossmann, and Rajeev Acharya.2020. Qd-tree: Learning Data Layouts for Big Data Analytics. In Proceedings of the2020 ACM SIGMOD International Conference on Management of Data. 193–208.

Discussion

- Experiments over benchmark and real workloads
  - Compared to current blocking schemes:
    - Provides physical speedups of more than an order of magnitude
- For data skipping based on selectivity:
  - Performs within 2X of the lower bound

[45]Zongheng Yang, Badrish Chandramouli, Chi Wang, Johannes Gehrke, Yinan Li,Umar Farooq Minhas, Per-Åke Larson, Donald Kossmann, and Rajeev Acharya.2020. Qd-tree: Learning Data Layouts for Big Data Analytics. In Proceedings of the2020 ACM SIGMOD International Conference on Management of Data. 193–208.

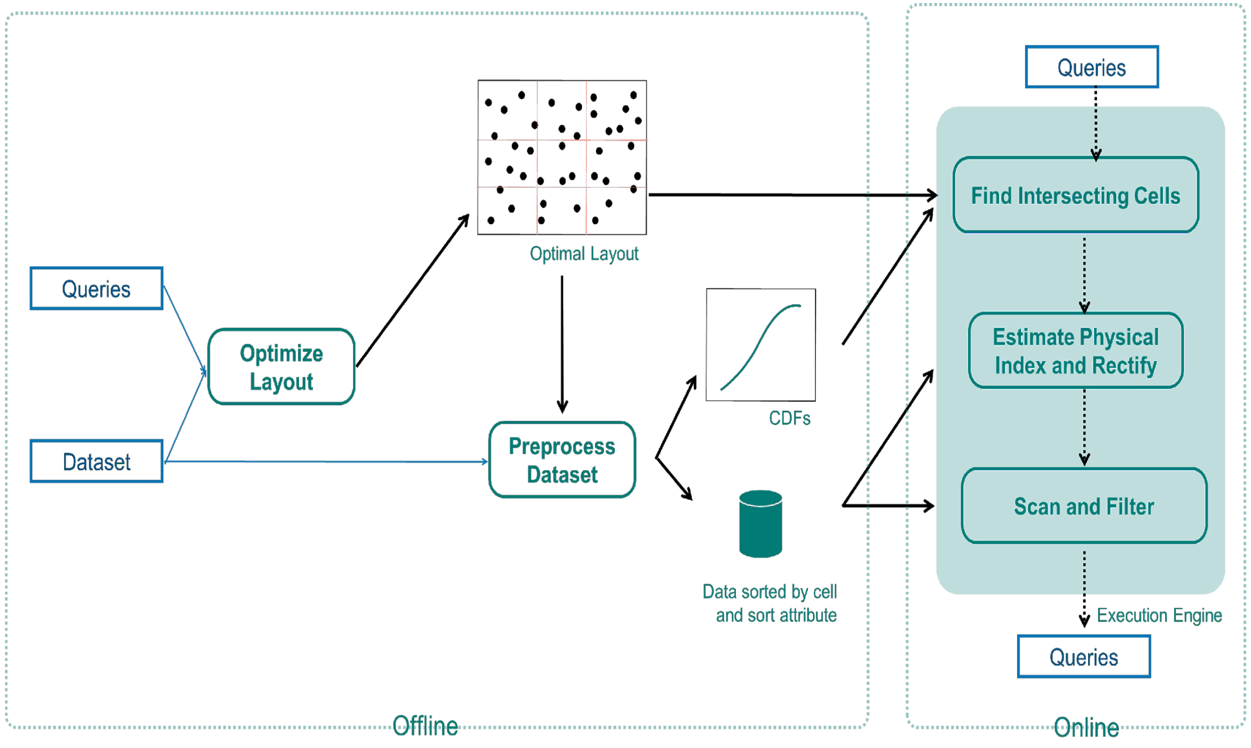
Taxonomy of Learned Indexes



Learning Multi-dimensional Indexes [SIGMOD'20]

Core Idea

- Proposed index structure: "Flood"
  - Read optimized grid-based index over the multi-dimensional space
  - Co-optimize the data layout and the index structure
    - For particular data and query distributions
  - Two components:
    - Offline (pre-processing)
      - Chooses an optimal layout
      - Creates an index based on that layout
    - Online
      - Query execution



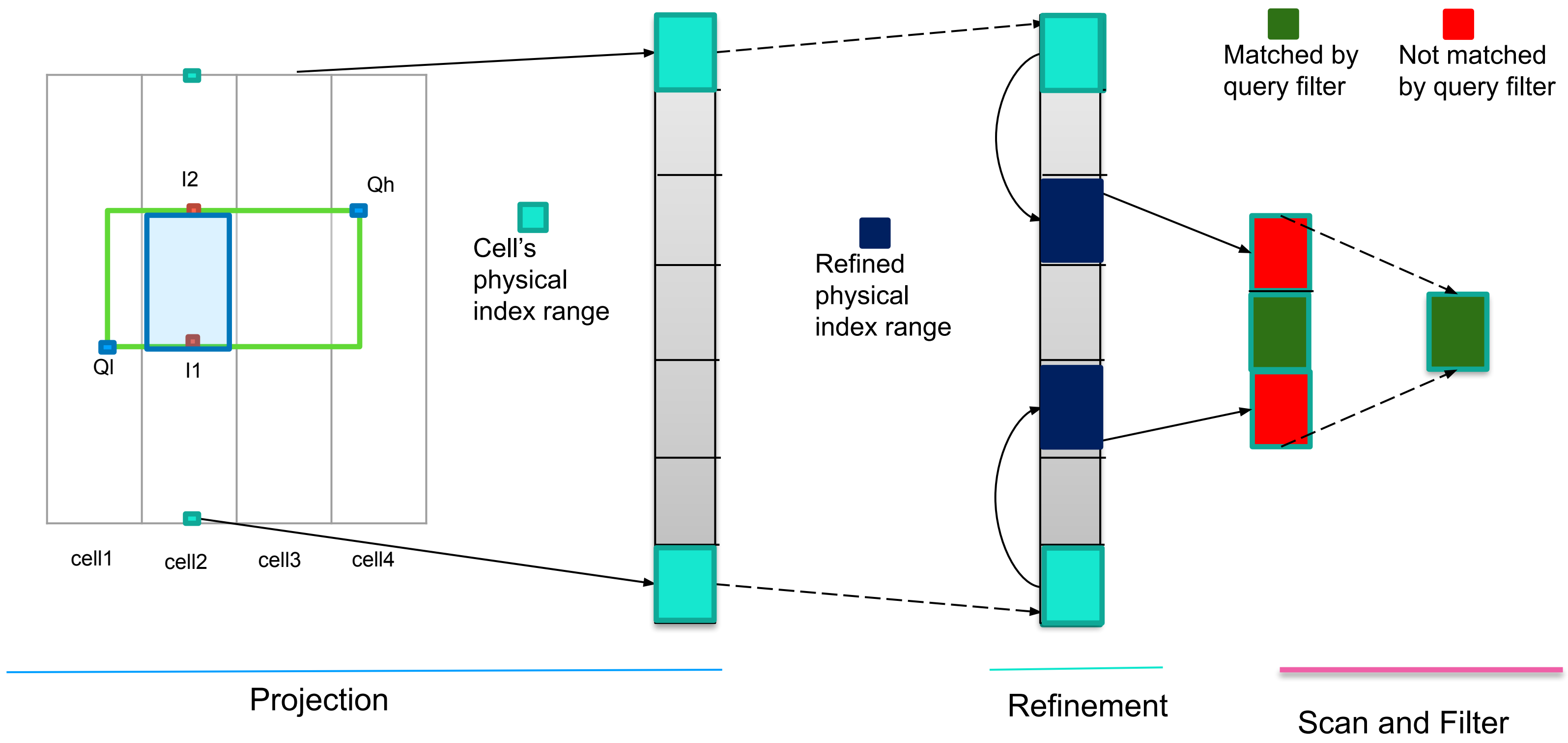
System Architecture

[31]Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-dimensional Indexes. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 985–1000

Learning Multi-dimensional Indexes [SIGMOD'20]

Flood's Workflow

- Projection:
  - Identifies the intersecting cells
  - Identifies the physical index range in each intersecting cell
- Refinement:
  - Utilizes the ordering of points within each cell to refine each physical index range
- Scan and Filter



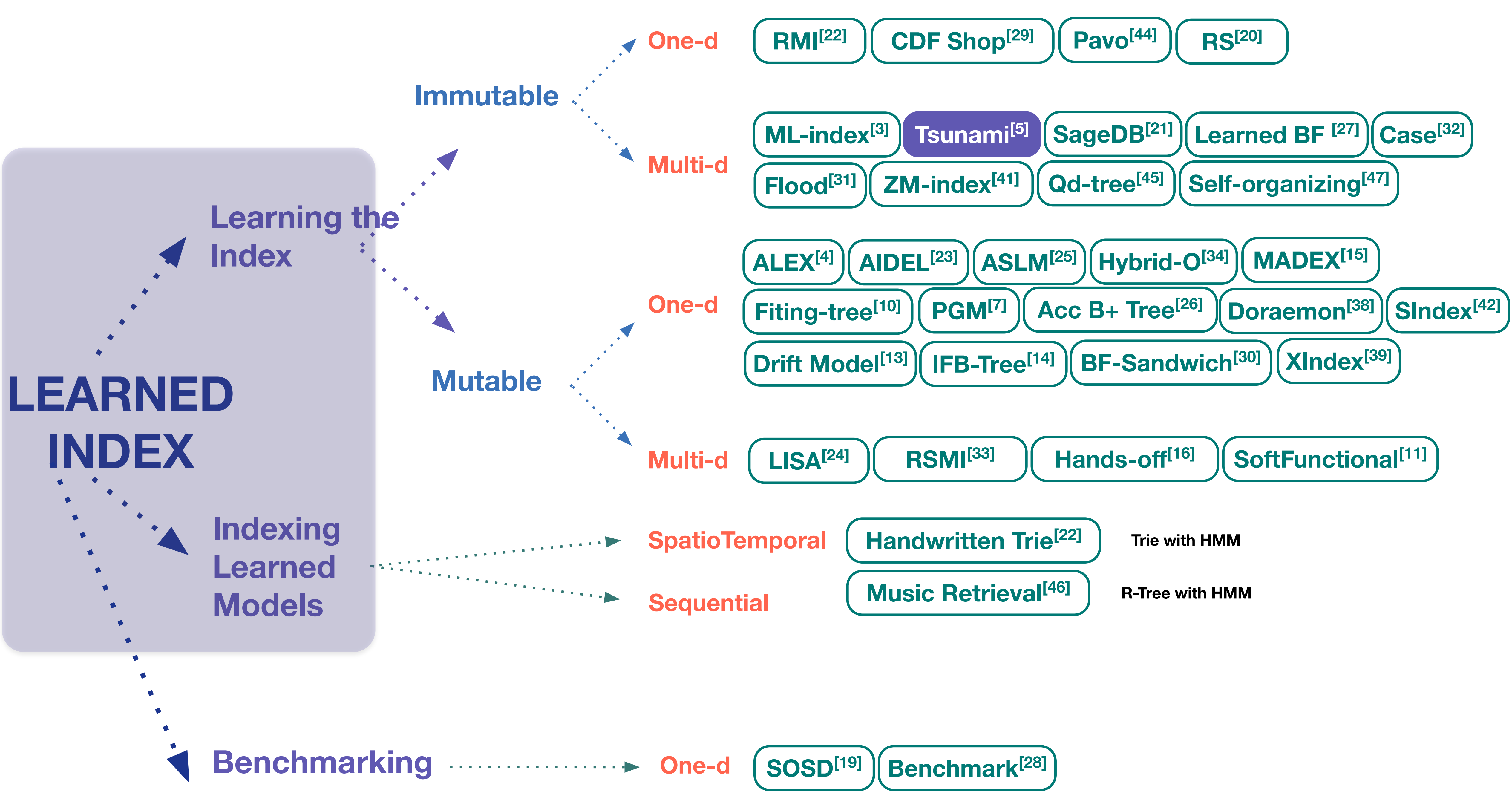
[31]Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-dimensional Indexes. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 985–1000

Learning Multi-dimensional Indexes [SIGMOD'20]

- Experimental Results:
  - Outperforms optimally tuned spatial indexes
  - Uses only a fraction of the space comparing with traditional indexes
- Limitations:
  - Cannot adapt to skewed query workload
  - If dimensions are correlated,
    - Performance and memory usage are affected

[31]Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-dimensional Indexes. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 985–1000

Taxonomy of Learned Indexes

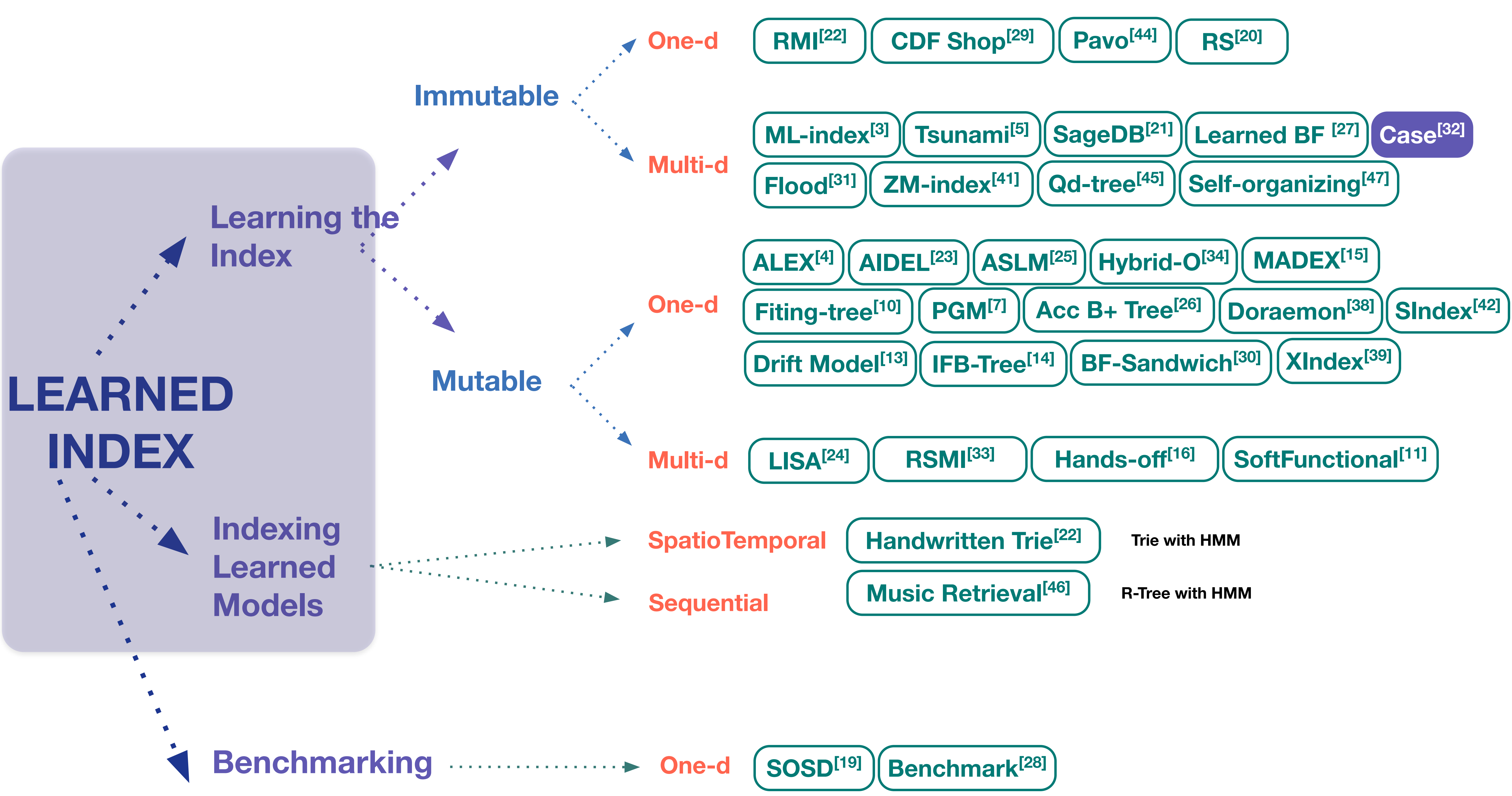


Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads

- Extend the idea of Flood to overcome its limitations
  - *Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. 2020. Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads. arXiv preprint arXiv:2006.13282(2020).*
    - Adaptable to changes in workload
    - Scales across data size, query selectivity, and dimensionality
    - Up to 6× faster

[31]Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-dimensional Indexes. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 985–1000

Taxonomy of Learned Indexes



Core Idea

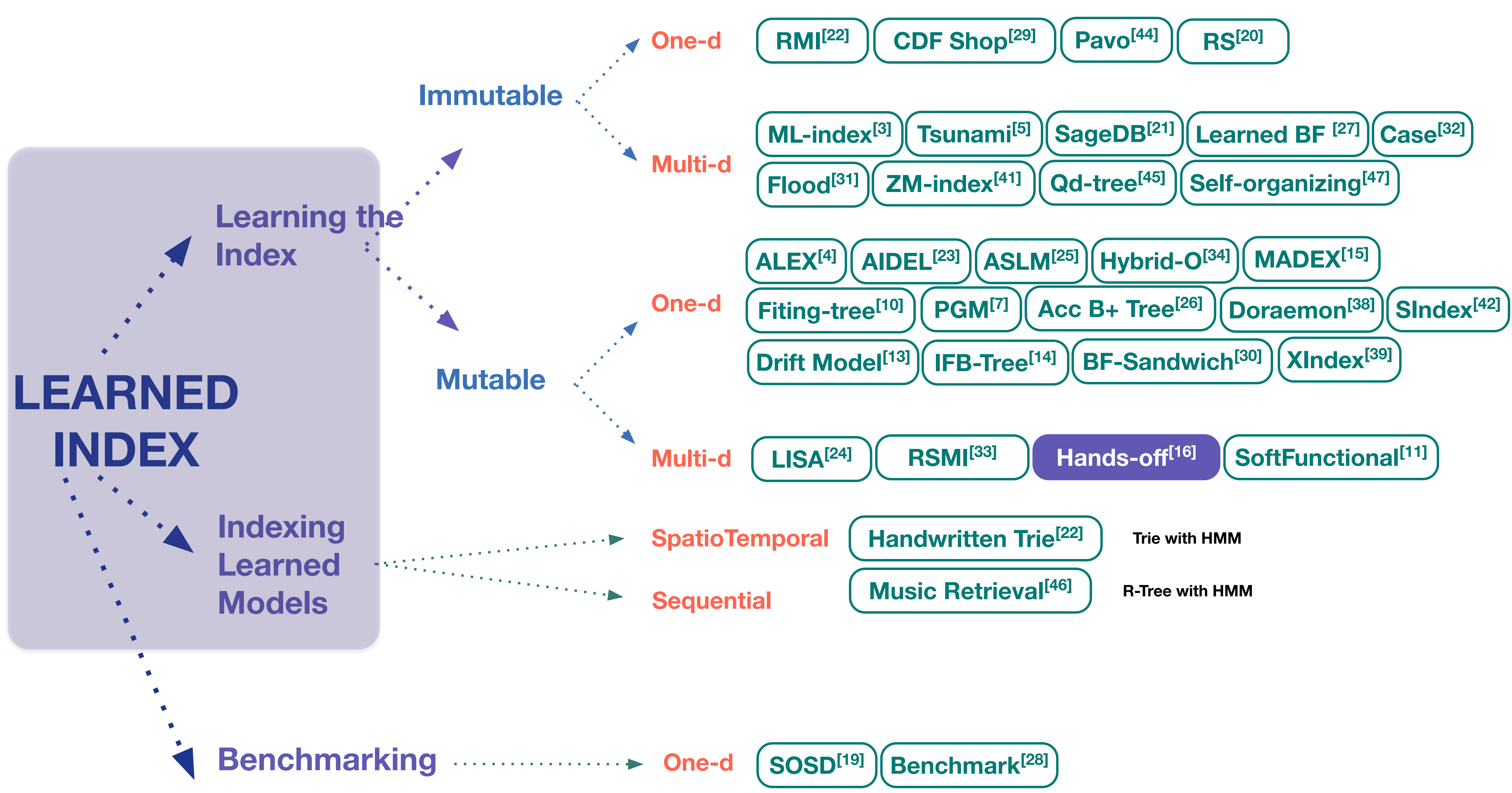
- Apply the techniques in Flood to five other multi-dimensional indexes to answer spatial range queries.
  - Fixed-grid, Adaptive-grid, Kd-tree, Quadtree and STR

Major Insights

- Replace binary search with a learned index within each partition
  - Improve query execution time by 11.79% to 39.51%
- Filter on 1D using traditional index then refine using learned indexes
  - 1.23x to 1.83x times faster than methods that filter on 2D
- Learned indexes are more effective on queries with low selectivity (e.g. 0.00001%) but less effective on queries with high selectivity (e.g. 0.1%).

[32]Varun Pandey, Alexander van Renen, Andreas Kipf, Ibrahim Sabek, Jialin Ding,and Alfons Kemper. 2020. The Case for Learned Spatial Indexes. arXiv preprintarXiv:2008.10349(2020).

Taxonomy of Learned Indexes



Hands-off Model Integration in Spatial Index Structures [AIDB@VLDB’20]

Motivation

- In-memory hierarchical trees require:
  - Excessive pointer-chasing
  - Time for chasing pointers impacts significantly the query execution time
- New approaches to design indexes are encouraged to utilize the modern hardware platforms

Core Idea

- Interpolation Friendly (IF) Indexes: IF-X
  - X is any multi-dimensional index
- Why Linear Interpolation?
  - Complex models have a higher capacity to fit the CDF
  - But complex models
    - Requires more parameters
    - Slower to compute
  - Linear interpolation is:
    - Simpler
    - Computationally inexpensive
  - Can eliminate expensive training process.

Leaf Node Layout

- IF-X indexes sort the records in each leaf node
  - Based on the best order using which the interpolation error is minimized.
- Store all required information in the header of the leaf node
  - No additional computation is needed
- The leaf node structure:
  - pDim: most predictable dimension which is used as the storage order

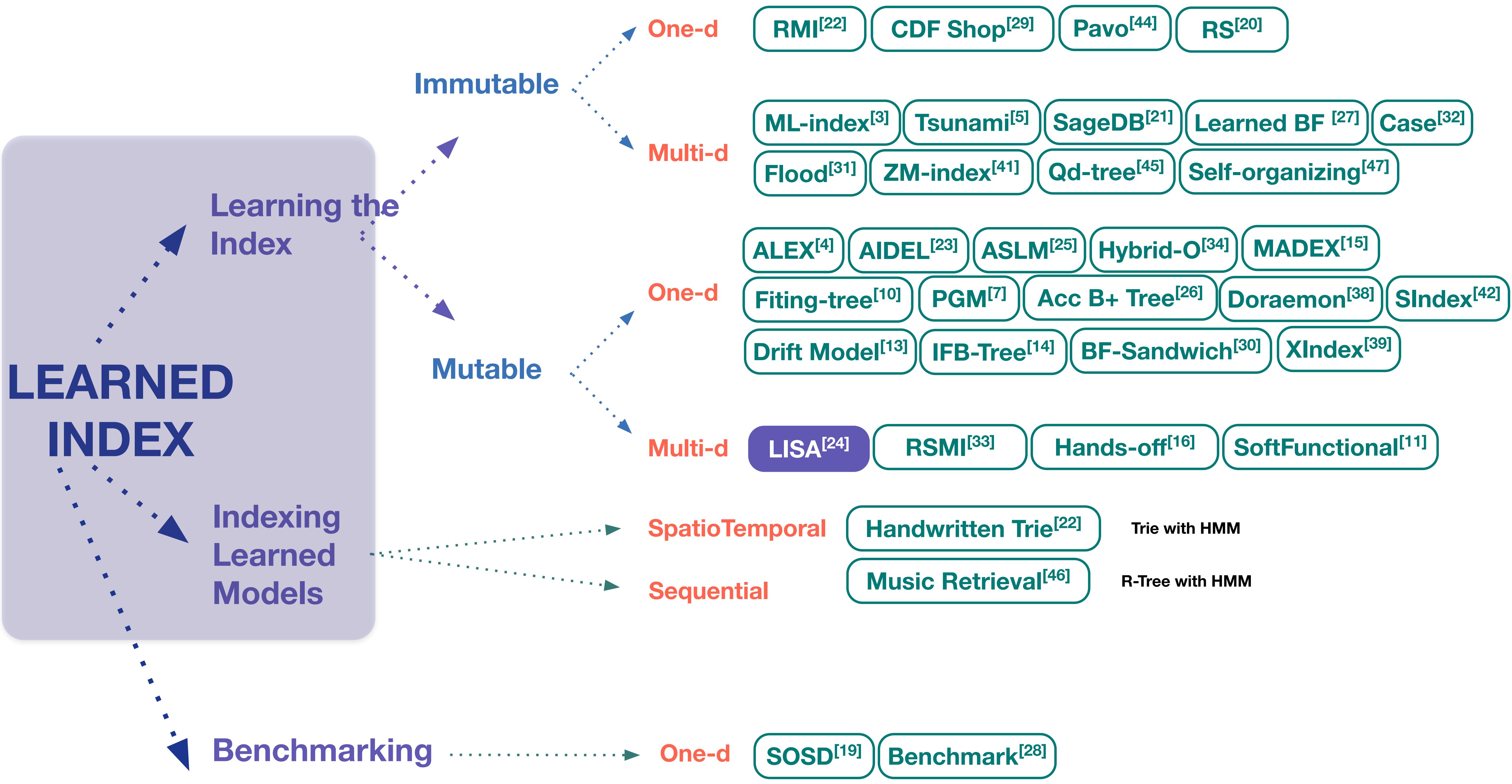
No of records (#)	Max error (delta)	Prediction axis (pDim)	Slope (A)	Base (C)	Records sorted by pDim

Performance

- Query execution time can be reduced by up to 60%
- Memory footprint can be reduced by over 90%

[3]Angjela Davitkova, Evica Milchevski, and Sebastian Michel. 2020. The ML-Index:A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries.. In EDBT. 407–410.

Taxonomy of Learned Indexes



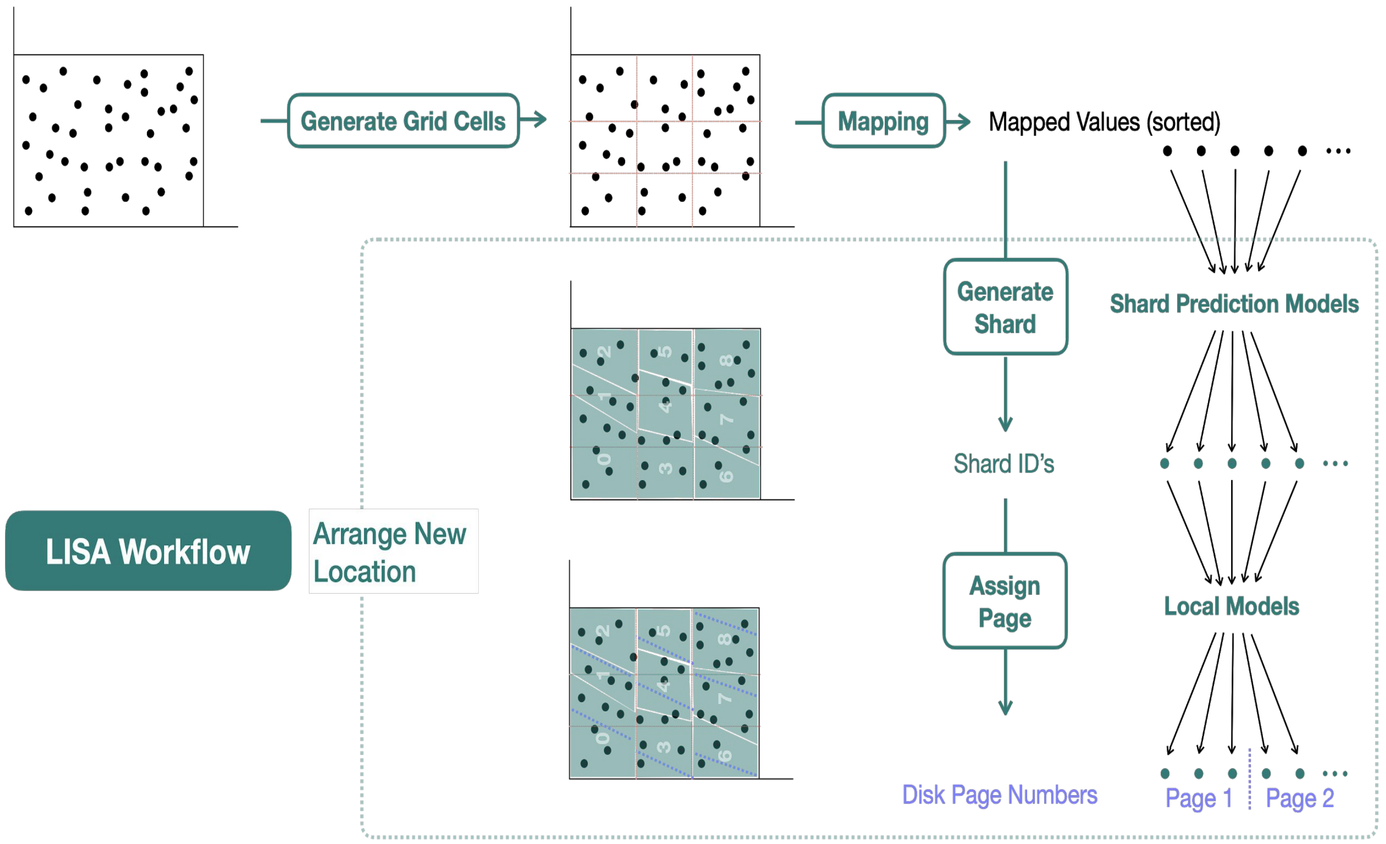
LISA: A Learned Index Structure for Spatial Data [SIGMOD'20]

Motivation

- Build a disk-based learned multi-dimensional index for spatial queries.
- Support updates

Core Idea

- Representation of grid cells
- Mapping function:
  - M(spatial keys) ⟹ 1D mapped values
- Learned Shard Prediction Function:
  - SP(mapped value) ⟹ Shard Id
    - Use ML models to generate searchable data layout in disk pages for arbitrary spatial dataset
- Local models:
  - Assign pages for all shards and perform intra-shard operations

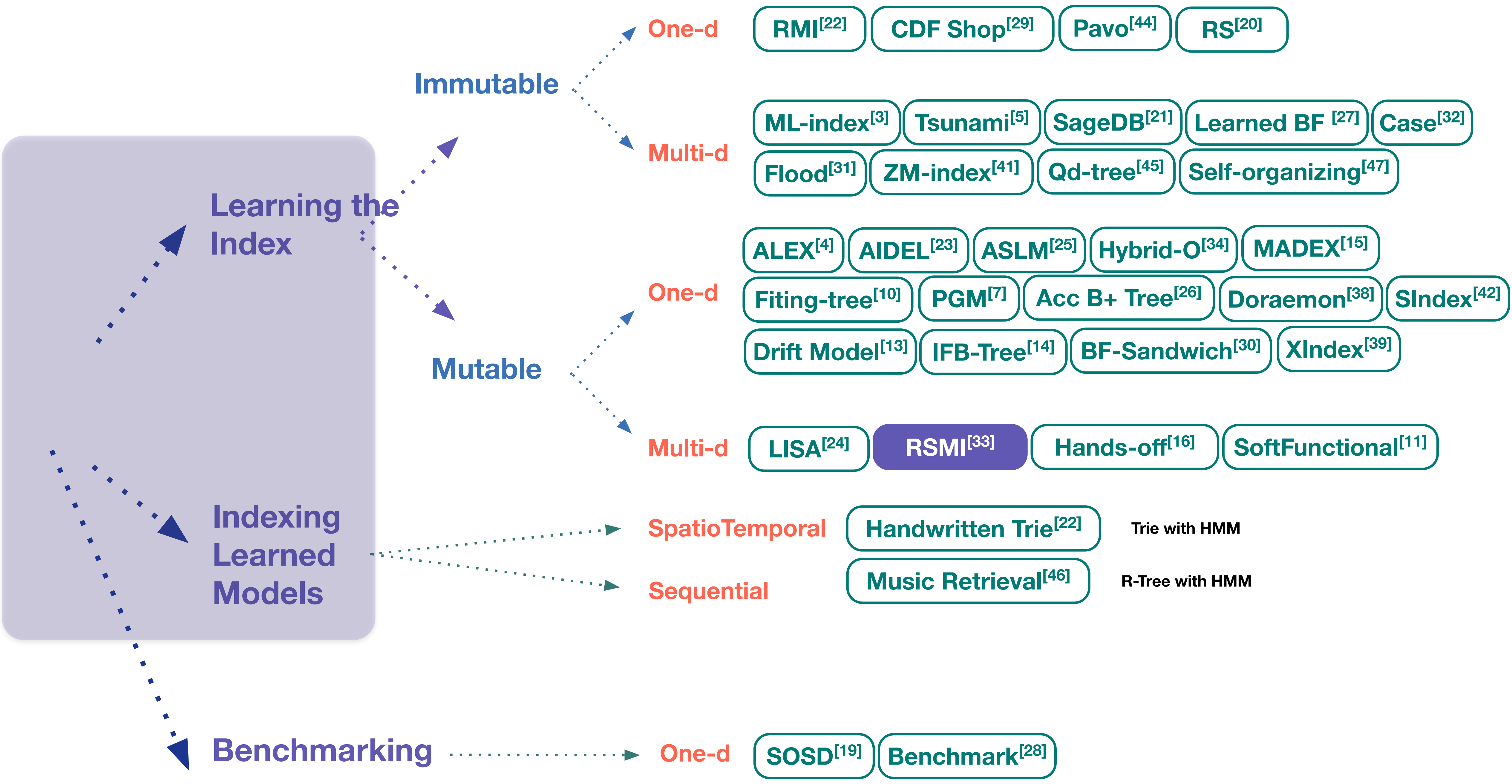


Performance

- Outperforms traditional spatial indexes for range and KNN queries :
  - Memory consumption
  - IO cost

[24]Pengfei Li, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. 2020. LISA: A Learned Index Structure for Spatial Data. SIGMOD(2020)

Taxonomy of Learned Indexes



Effectively Learning Spatial Indices [VLDB'20]

Motivation

- Selecting grid resolution for Z-order for learned multi-dimensional index (e.g. ZM-Index[41]) is difficult:
  - Large cells
    - More false positives due to many points per cell
  - Small cells
    - Hard to learn due to uneven gaps in Cumulative Distribution Function (CDF)

Core Idea

- Spatial index based on ordering the data points by a rank space-based transformation*
  - Simplify the indexing functions to be learned
  - M(search keys) $\Rightarrow$ disk block Ids (location)

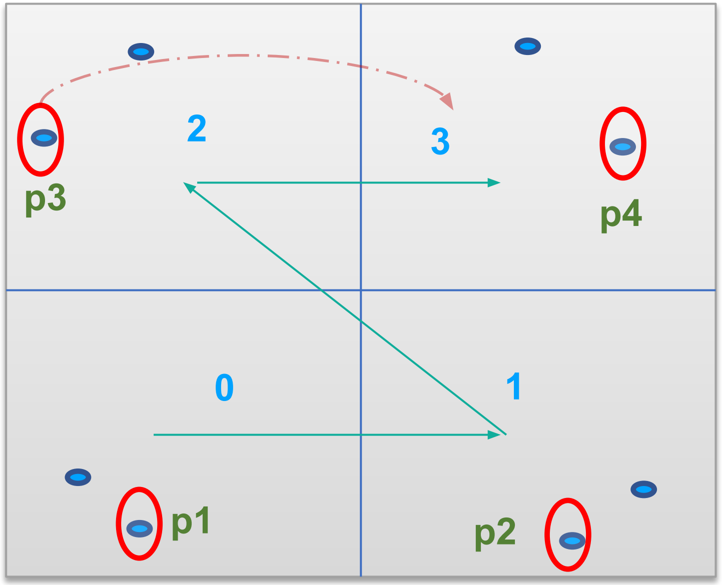
- For scaling to large datasets, proposes:
  - Introduce a Recursive Spatial Model Index (RSMI) (in lieu of RMI)
- Support point, window, and kNN queries
- Support updates

[33]Jianzhong Qi, Guanli Liu, Christian S Jensen, and Lars Kulik. 2020. Effectively learning spatial indices. Proceedings of the VLDB Endowment13, 12 (2020), 2341–2354.
*[48] J. Qi, Y. Tao, Y. Chang, and R. Zhang. Theoretically optimal and empirically efficient R-trees with strong parallelizability. PVLDB, 11(5):621–634, 2018.

Effectively Learning Spatial Indices [VLDB'20]

RSMI

- Recursive Spatial Model Index (RSMI):
  - Recursively partitions a dataset
  - Partitioning is learned over the distribution of data
- Steps:
  - Initially distribute the data into equal sized partitions
  - Use a Space Filling Curve (SFC) to assign Ids to partitions
  - Learn the partition Ids using a model $M_{0,0}$
  - Rearrange the data based on the prediction of $M_{0,0}$
  - Recursively repartition
    - Until each partition can be learned with a simple model



Point	p1	p2	p3	p4
Initial partition Id	0	1	2	3
Model predicted Id	0	1	3	3
Learned partition Id	0	1	3	3

Discussion

- Window and kNN query results are highly accurate but not exact.
  - i.e., over 87% across a variety of settings
  - Separate mechanism has been proposed for exact answer.
- Does not support query for spatial objects with non-zero extent

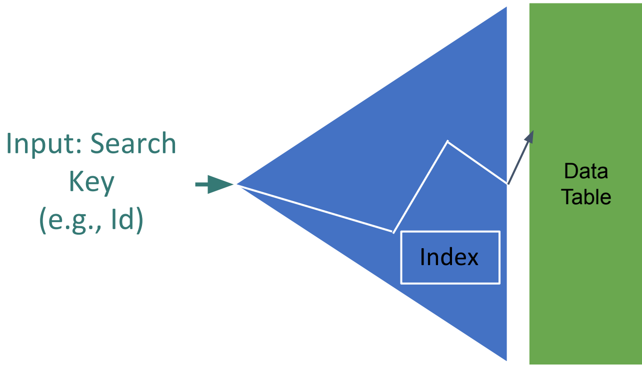
[33]Jianzhong Qi, Guanli Liu, Christian S Jensen, and Lars Kulik. 2020. Effectively learning spatial indices. Proceedings of the VLDB Endowment13, 12 (2020), 2341–2354.

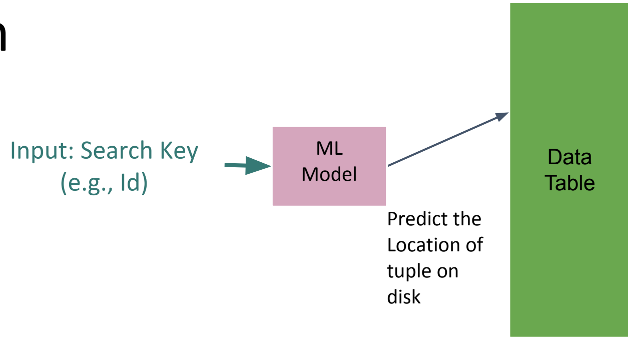
Outline of the Tutorial

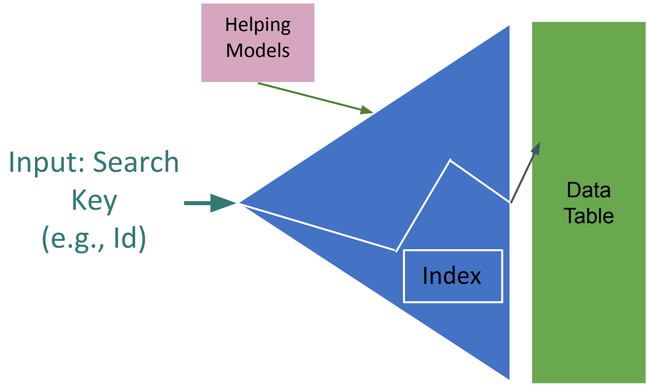
- Introduction and Taxonomy
- Indexing the Learned Models vs. Learning the Indexes
- Static vs. Dynamic Learned Indexes
- Learned One-Dimensional Indexes
- Learned Multidimensional Indexes
- **Open Problems for Future Research**

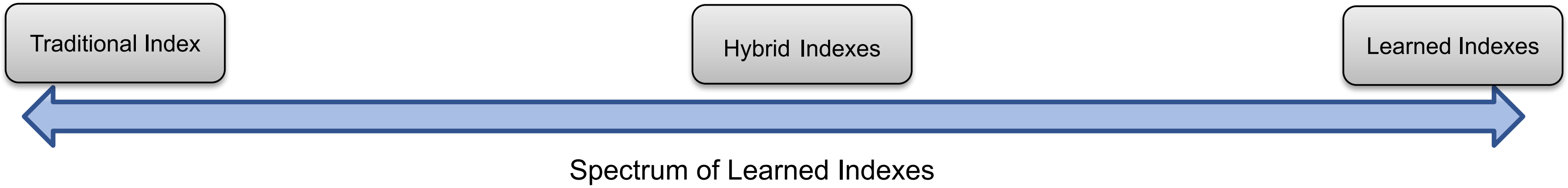
Spectrum of Learned Multi-dimensional Indexes

- *Traditional Indexes:*
  - Theoretical guarantee on performance
  - Well studied and successfully integrated in real systems
- *Learned Indexes:*
  - Learn search-key distribution with some error correction mechanism
  - Better performance with less space requirement
- *Hybrid Indexes:*
  - Optimizing traditional indexes with helping (e.g., ML) models







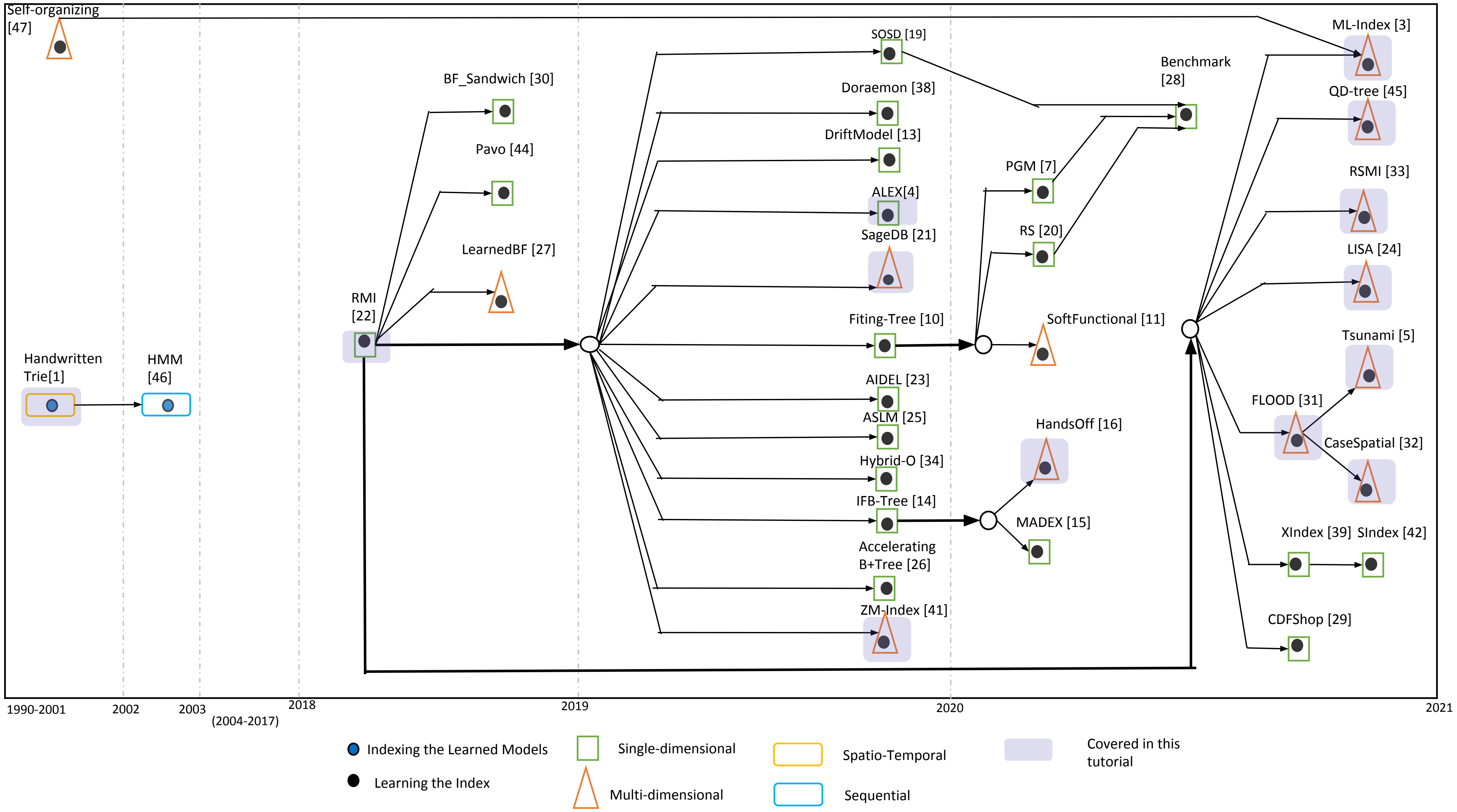


[14]Ali Hadian and Thomas Heinis. 2019. Interpolation-friendly B-trees: Bridging the gap between algorithmic and learned indexes. In 22nd International Conference on Extending Database Technology (EDBT 2019). https://doi.org/10.5441/002/edbt.2019.93

Some Open Problems

- Efficiently support Inserts/Updates
- Support for other spatial operations, e.g., KNN, spatial join, closest pairs
- What types of ML models to use?
- Integrate with real database engines
- Concurrency support
- Develop benchmark for Learned Multidimensional Indexes

Evolution of Learned Indexes



References

- [1]Walid Aref, Daniel Barbará, and Padmavathi Vallabhaneni. 1995. The Handwritten Trie: Indexing Electronic Ink. SIGMOD Rec.24, 2 (May 1995), 151–162. https://doi.org/10.1145/568271.223811
- [2]Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger.1990. 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. 322–331.
- [3]Angjela Davitkova, Evica Milchevski, and Sebastian Michel. 2020. The ML-Index: A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries.. In EDBT. 407–410.
- [4] Jialin Ding, Umar Farooq Minhas, Hantian Zhang, Yinan Li, Chi Wang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, and David Lomet. 2019. ALEX: An Updatable Adaptive Learned Index. arXiv preprint arXiv:1905.08898(2019).
- [5]Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. 2020. Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads. arXiv preprint arXiv:2006.13282(2020).
- [6]Paolo Ferragina and Giorgio Vinciguerra. 2020. Learned Data Structures. In Recent Trends in Learning From Data, Luca Oneto, Nicolò Navarin, AlessandroSperduti, and Davide Anguita (Eds.). Springer International Publishing, 5-41. https://doi.org/10.1007/978-3-030-43883-8_2
- [7]Paolo Ferragina and Giorgio Vinciguerra. 2020. The PGM-index. Proceedings of the VLDB Endowment13, 8 (Apr 2020), 1162–1175. https://doi.org/10.14778/3389133.3389135
- [8]Paolo Ferragina, Giorgio Vinciguerra, and Michele Miccinesi. 2019. Superseding traditional indexes by orchestrating learning and geometry. arXiv preprintarXiv:1903.00507(2019).

References

- [9]Raphael A. Finkel and Jon Louis Bentley. 1974. Quad trees a data structure for retrieval on composite keys. Acta informatica4, 1 (1974), 1–9.
- [10]Alex Galakatos, Michael Markovitch, Carsten Binnig, Rodrigo Fonseca, and Tim Kraska. 2019. Fiting-tree: A data-aware index structure. In Proceedings of the 2019International Conference on Management of Data. 1189–1206.
- [11]Behzad Ghaffari, Ali Hadian, and Thomas Heinis. 2020. Leveraging Soft Functional Dependencies for Indexing Multi-dimensional Data. arXiv preprintarXiv:2006.16393(2020).
- [12]Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. In Proceedings of the 1984 ACM SIGMOD international conference on Management of data. 47–57.
- [13]Ali Hadian and Thomas Heinis. 2019. Considerations for handling updates in learned index structures. In Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. ACM, 3.
- [14]Ali Hadian and Thomas Heinis. 2019. Interpolation-friendly B-trees: Bridging the gap between algorithmic and learned indexes. In22nd International Conference on Extending Database Technology (EDBT 2019). https://doi.org/10.5441/002/edbt.2019.93
- [15]Ali Hadian and Thomas Heinis. 2020. MADEX: Learning-augmented Algorithmic Index Structures. In Proceedings of the 2nd International Workshop on Applied AI for Database Systems and Applications.

References

- [16]Ali Hadian, Ankit Kumar, and Thomas Heinis. 2020. Hands-off Model Integration in Spatial Index Structures. In Proceedings of the 2nd International Workshop onApplied AI for Database Systems and Applications.
- [17]Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, Not from Queries!13, 7 (2020).
- [18]Stratos Idreos and Tim Kraska. 2019. From Auto-tuning One Size Fits All to Self-designed and Learned Data-intensive Systems (Tutorial). In Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference2019, Amsterdam, The Netherlands, June 30 - July 5, 2019. 2054–2059. http://people.csail.mit.edu/kraska/pub/sigmod19tutorialpart2.pdf
- [19]Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper,Tim Kraska, and Thomas Neumann. 2019. SOSD: A Benchmark for Learned Indexes. ArXivabs/1911.13014 (2019).
- [20]Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper,Tim Kraska, and Thomas Neumann. 2020. RadixSpline: A Single-Pass Learned Index.ArXivabs/2004.14541 (2020).
- [21]Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H Chi, Jialin Ding, Ani Kristo,Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019.Sagedb: A learned database system. (2019).
- [22]Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018.The case for learned index structures. In Proceedings of the 2018 International Conference on Management of Data. ACM, 489–504.
- [23]Pengfei Li, Yu Hua, Pengfei Zuo, and Jingnan Jia. 2019. A Scalable Learned Index Scheme in Storage Systems. CoRRabs/1905.06256 (2019). arXiv:1905.06256http://arxiv.org/abs/1905.06256
- [24]Pengfei Li, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. 2020. LISA: A Learned Index Structure for Spatial Data. SIGMOD(2020)

References

- [25]Xin Li, Jingdong Li, and Xiaoling Wang. 2019. ASLM: Adaptive single layer model for learned index. In International Conference on Database Systems for Advanced Applications. Springer, 80–95.
- [26]A Llavesh, Utku Sirin, R West, and A Ailamaki. 2019. Accelerating b+ tree search by using simple machine learning techniques. In Proceedings of the 1stInternational Workshop on Applied AI for Database Systems and Applications.
- [27]Stephen Macke, Alex Beutel, Tim Kraska, Maheswaran Sathiamoorthy,Derek Zhiyuan Cheng, and EH Chi. 2018. Lifting the curse of multidimensional data with learned existence indexes. InWorkshop on ML for Systems at NeurIPS.
- [28]Ryan Marcus, Andreas Kipf, Alexander van Renen, Mihail Stoian, Sanchit Misra,Alfons Kemper, Thomas Neumann, and Tim Kraska. 2020. Benchmarking Learned Indexes. arXiv preprint arXiv:2006.12804(2020).
- [29]Ryan Marcus, Emily Zhang, and Tim Kraska. 2020. CDFShop: Exploring and Optimizing Learned Index Structures. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2789–2792.
- [30] Michael Mitzenmacher. 2018. A model for learned bloom filters and optimizing by sandwiching. In Advances in Neural Information Processing Systems. 464–473.
- [31]Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-dimensional Indexes. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 985–1000.

References

- [32]Varun Pandey, Alexander van Renen, Andreas Kipf, Ibrahim Sabek, Jialin Ding,and Alfons Kemper. 2020. The Case for Learned Spatial Indexes. arXiv preprintarXiv:2008.10349(2020).
- [33]Jianzhong Qi, Guanli Liu, Christian S Jensen, and Lars Kulik. 2020. Effectively learning spatial indices. Proceedings of the VLDB Endowment13, 12 (2020), 2341–2354.
- [34]Wenwen Qu, Xiaoling Wang, Jingdong Li, and Xin Li. 2019. Hybrid indexes by exploring traditional B-tree and linear regression. In International Conference on Web Information Systems and Applications. Springer, 601–613.
- [35]Ibrahim Sabek and Mohamed F Mokbel. 2020. Machine learning meets big spatial data. In2020 IEEE 36th International Conference on Data Engineering (ICDE). IEEE,1782–1785.
- [36]Hanan Samet. 1984. The quadtree and related hierarchical data structures. ACM Computing Surveys (CSUR)16, 2 (1984), 187–260.
- [37]Hanan Samet. 2006.Foundations of multidimensional and metric data structures. Morgan Kaufmann.
- [38]Chuzhe Tang, Zhiyuan Dong, Minjie Wang, Zhaoguo Wang, and Haibo Chen.2019. Learned Indexes for Dynamic Workloads. arXiv preprint arXiv:1902.00655(2019).

References

- [39]Chuzhe Tang, Youyun Wang, Zhiyuan Dong, Gansen Hu, Zhaoguo Wang, MinjieWang, and Haibo Chen. 2020. XIndex: a scalable learned index for multicore data storage. Proceedings of the 25th ACM SIGPLAN Symposium on Principles andPractice of Parallel Programming(2020).
- [40]Peter Van Sandt, Yannis Chronis, and Jignesh M Patel. 2019. Efficiently Searching In-Memory Sorted Arrays: Revenge of the Interpolation Search?. In Proceedingsof the 2019 International Conference on Management of Data. ACM, 36–53.
- [41]Haixin Wang, Xiaoyi Fu, Jianliang Xu, and Hua Lu. 2019. Learned Index for Spatial Queries. In2019 20th IEEE International Conference on Mobile Data Management(MDM). IEEE, 569–574.
- [42]Youyun Wang, Chuzhe Tang, Zhaoguo Wang, and Haibo Chen. 2020. SIndex: a scalable learned index for string keys. In Proceedings of the 11th ACM SIGOPSAsia-Pacific Workshop on Systems. 17–24.
- [43]Yingjun Wu, Jia Yu, Yuanyuan Tian, Richard Sidle, and Ronald Barber. 2019.Designing Succinct Secondary Indexing Mechanism by Exploiting Column Correlations. arXiv preprint arXiv:1903.11203(2019).
- [44]Wenkun Xiang, Hao Zhang, Rui Cui, Xing Chu, Keqin Li, and Wei Zhou. 2018.Pavo: A RNN-Based Learned Inverted Index, Supervised or Unsupervised?IEEEAccess7 (2018), 293–303.
- [45]Zongheng Yang, Badrish Chandramouli, Chi Wang, Johannes Gehrke, Yinan Li,Umar Farooq Minhas, Per-Åke Larson, Donald Kossmann, and Rajeev Acharya.2020. Qd-tree: Learning Data Layouts for Big Data Analytics. In Proceedings of the2020 ACM SIGMOD International Conference on Management of Data. 193–208.
- [46] Jin, Hui, and H. V. Jagadish. "Indexing Hidden Markov Models for Music Retrieval." In ISMIR. 2002.
- [47] Babu, G. Phanendra. "Self-organizing neural networks for spatial data." Pattern Recognition Letters 18, no. 2 (1997): 133-142.
- [48] J. Qi, Y. Tao, Y. Chang, and R. Zhang. Theoretically optimal and empirically efficient R-trees with strong parallelizability. PVLDB, 11(5):621–634, 2018.

Q&A

Website of Tutorial: https://www.cs.purdue.edu/homes/aref/learned-indexes-tutorial.html