LEARNED INDEXES FROM THE ONE-DIMENSIONAL TO THE MULTI-DIMENSIONAL SPACES: CHALLENGES, TECHNIQUES, AND OPPORTUNITIES

Presenters: Abdullah Al-Mamun, Jianguo Wang, Walid G. Aref

Purdue University

West Lafayette, Indiana, USA

Venue: 2025 ACM SIGMOD, Berlin, Germany

PURDUE UNIVERSITY

Outline

- Introduction and Historical Background
- Learned Indexes for the One-dimensional Space
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY®

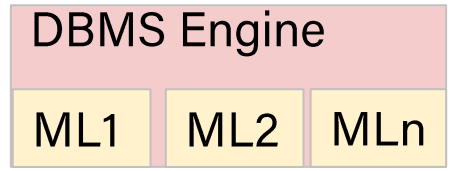
Outline

- Introduction and Historical Background
  - Indexing the Learned Models vs. Learning the Index
  - Evolution of Learned Indexes
- Learned Indexes for the One-dimensional Space
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

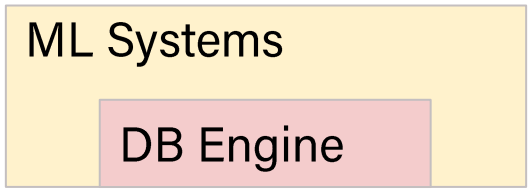
PURDUE UNIVERSITY®

Introduction: Database Systems + Machine Learning

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



ML for DB



DB for ML

PURDUE UNIVERSITY

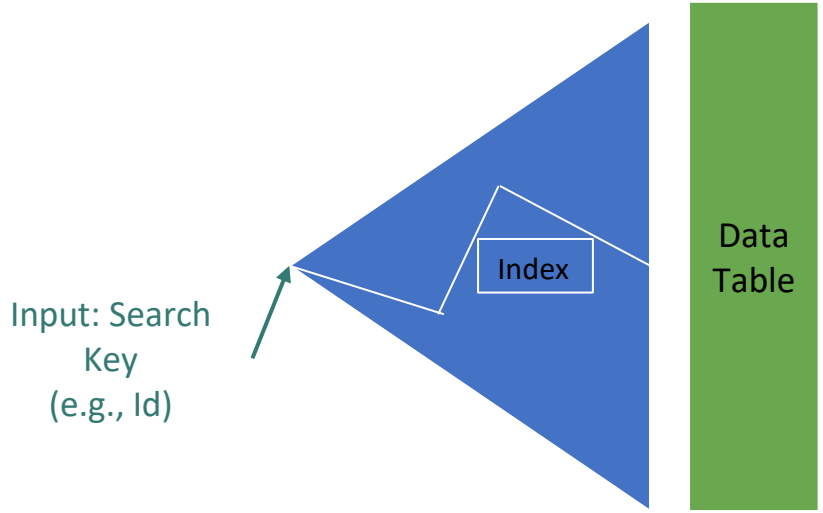
Introduction: Data Lookup

- Given a large collection of data, there are multiple ways for data lookup:
  - Linear Search
  - Binary Search if data is sorted
  - Build Index structures
- *What if we have a good estimation over the data distribution in advance?*
  - *Can we further speed up the process of data lookup?*

PURDUE UNIVERSITY

Introduction: Database Indexing

- Database Index: Provide efficient access to data
- Popular index structure: 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)→ position
  - Take key as input and return the order of key's tuple inside the table



PURDUE UNIVERSITY

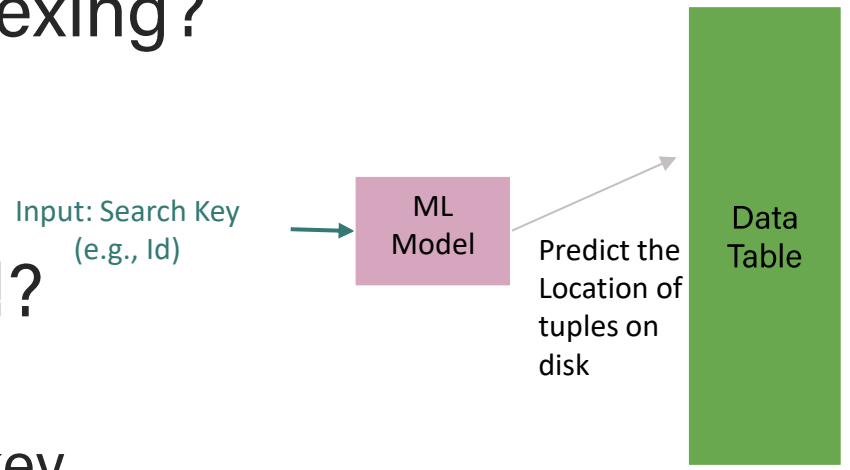
Introduction: One- vs. Multi-dimensional Index

- One-dimensional Index
  - Built on dataset with only a single dimension
  - Data can be sorted (i.e., totally ordered).

- Multi-dimensional Index
  - The dimension of the data is typically between 2 to 10.
  - No obvious total sort order for multi-dimensional data
  - In contrast, dimension of the data can be very high (e.g., 100) in the context of high-dimensional index.

Introduction: Learned Indexes

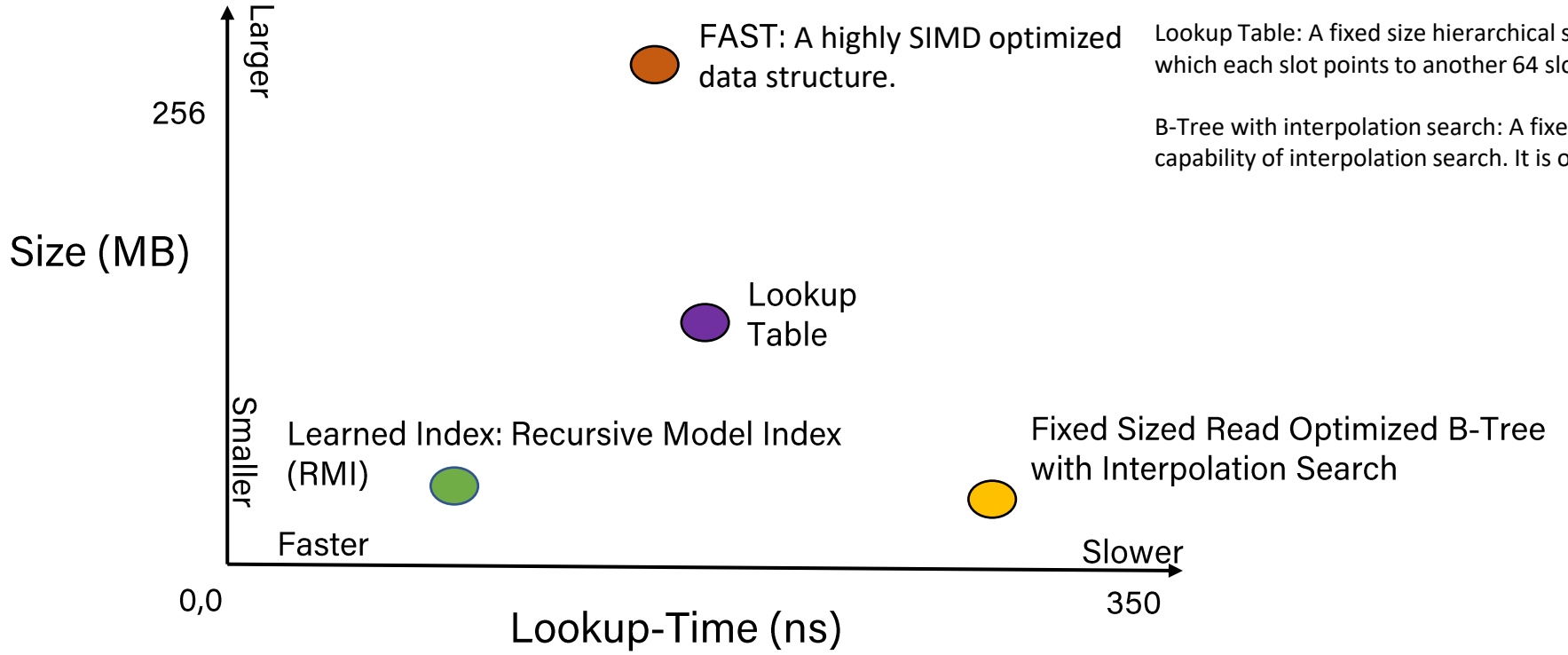
- Can one use ML techniques to guide data indexing?
- Can we learn the function:
  - B$^+$-tree(key) → Location of tuple in table?
- Can we replace the B$^+$-tree with an ML model?
  - Learned Index: viewing an index as a model
  - ML-Model(key) predicts the storage location of the key
  - Include an error correction mechanism (e.g., binary search) if the model mispredicts
    - Feasible approach due to the ***total order*** of the one-dimensional data
  - For example, with a simple linear model, we only need two floats: slope and intercept
    - Searching executes potentially in O(1) time
    - Space requirement is also potentially O(1)



PURDUE UNIVERSITY

Introduction: Initial Results

- Initial performance results (approximate) of a learned index
  - Promising → Faster lookup time and smaller storage



FAST: A highly SIMD optimized data structure.

Lookup Table: A fixed size hierarchical structure (e.g., 64 slots for which each slot points to another 64 slots, etc.).

B-Tree with interpolation search: A fixed-height B-Tree with the capability of interpolation search. It is optimized for read operations.

PURDUE UNIVERSITY®

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.

Outline

- ✓ Introduction and Historical Background
  - → Indexing the Learned Models vs. Learning the Index
  - Evolution of Learned Indexes
- Learned Indexes for the One-dimensional Space
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY

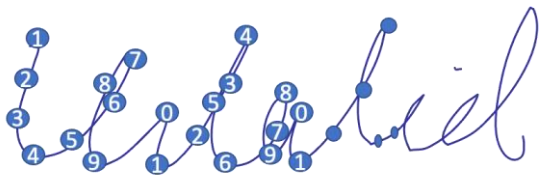
Introduction: Indexing the Learned Models vs. Learning the Index

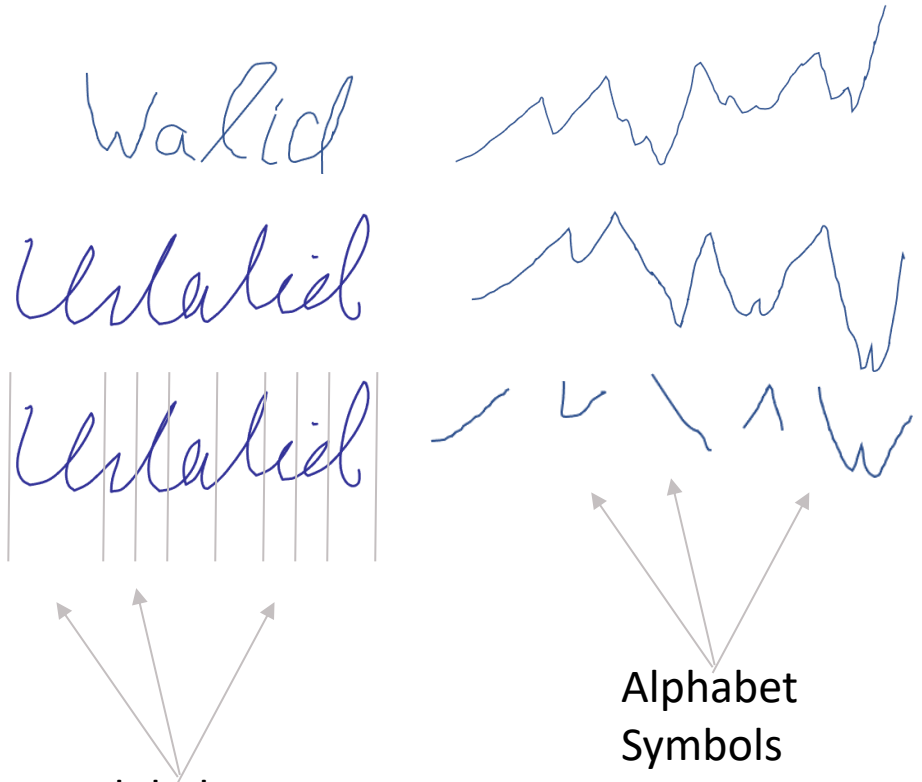
- Indexing the Learned Models
  - Given a collection of ML models, e.g., object recognition models (Cats, Dogs, Trains, etc.), and an input object, say o
    - Identify the class of o (cat vs. dog, etc.)
  - Instead of executing all models and identifying which has the highest matching score
    - Can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

PURDUE UNIVERSITY

Introduction: Indexing the Learned Models

- 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, . . .



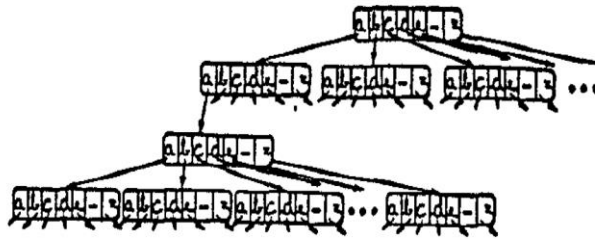


PURDUE UNIVERSITY®

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

Introduction: Indexing the Learned Models (Cont'd)

- 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
- Indexing the learned Hidden Markov Models
- Construct a trie structure over the learned alphabet symbols



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

Introduction: Indexing the Learned Models (Cont'd)

Hidden Markov Model

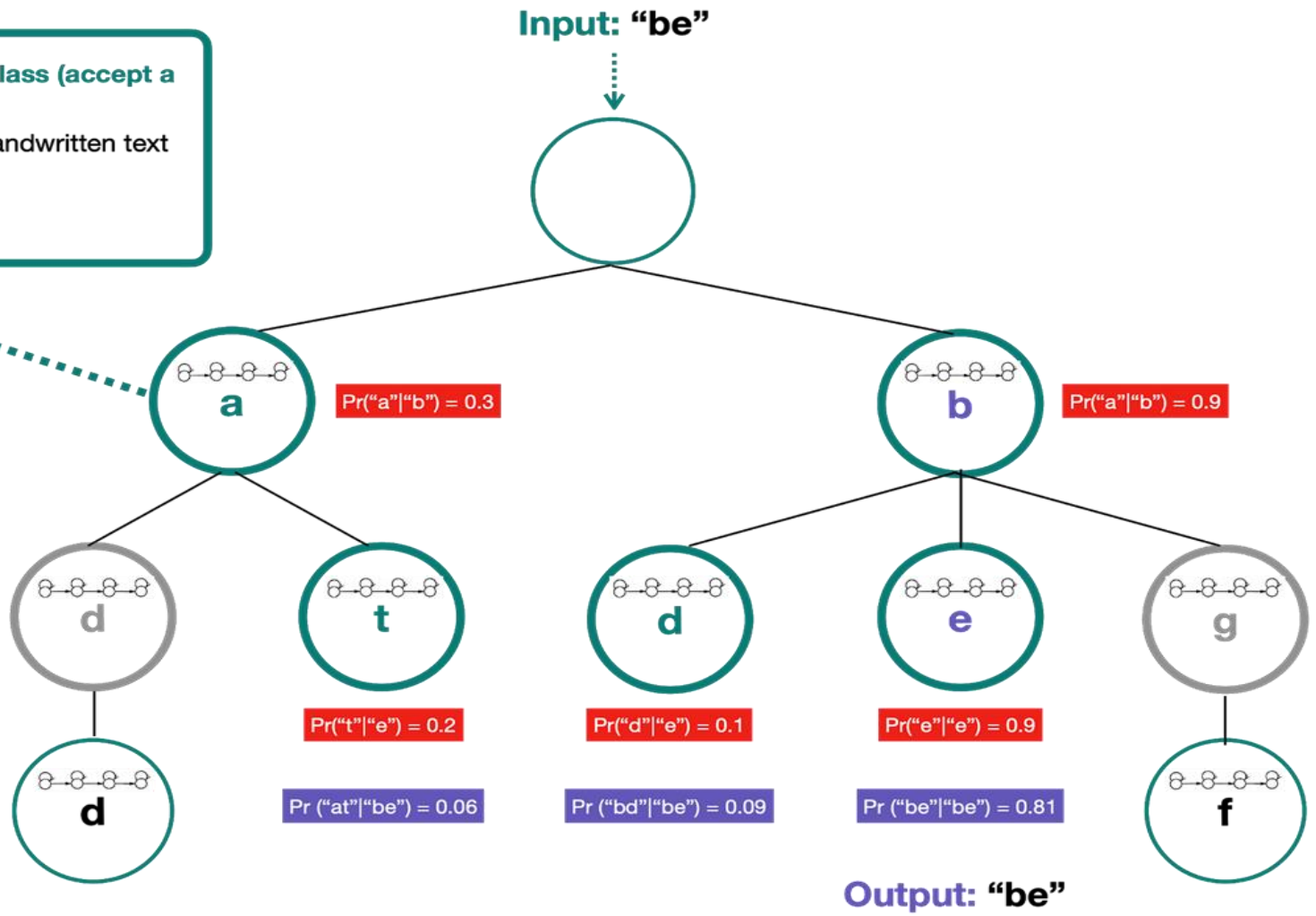
Each HMM is constructed to represent a pictogram class (accept a specific pictogram with high probability)

- **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



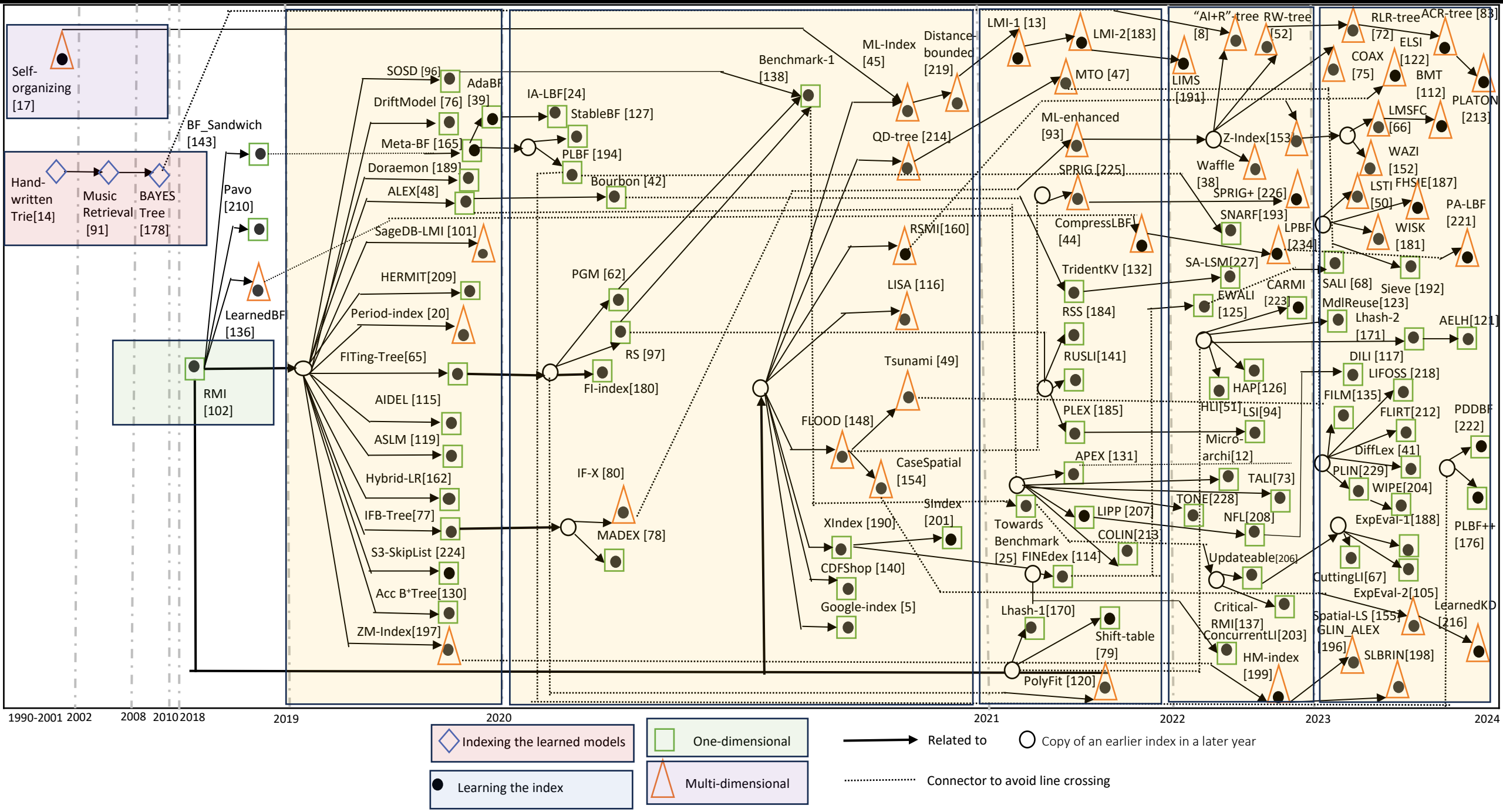
Walid Aref, Daniel Barbará, and Padmavathi Vallabhaneni. 1995. The Handwritten Trie: Indexing Electronic Ink. SIGMOD Rec.24, 2 (May 1995), 151–162. https://doi.org/10.1145/568271.223811

Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - → Evolution of Learned Indexes
- Learned Indexes for the One-dimensional Space
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY

Introduction: Evolution of Learned Indexes

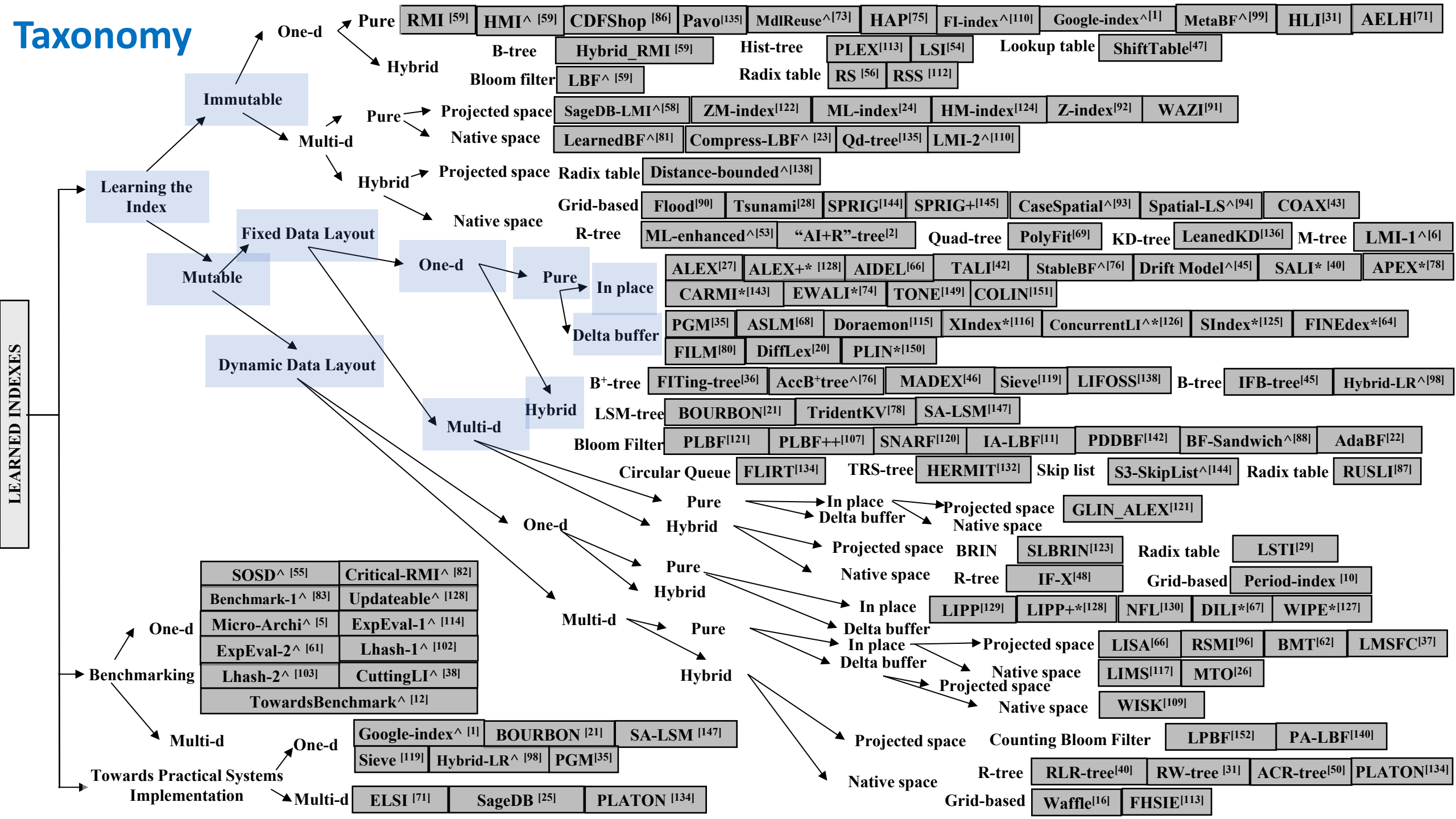


Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
  - → Learned Indexes for the One-dimensional Space
    - → Taxonomy of Learned Indexes
      - Immutable vs. Mutable Indexes
      - Fixed vs. Dynamic Data Layouts
      - Pure vs. Hybrid Indexes
      - In-place vs. Delta Buffer Insertion Strategies
  - Learned Indexes for the Multi-dimensional Space
  - Open Challenges for Future Research

PURDUE UNIVERSITY

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
Multi-d
Pure
Projected space
Native space
Hybrid
Projected space
Native space
Mutable
Fixed Data Layout
One-d
Pure
In place
Delta buffer
Hybrid
Dynamic Data Layout
Multi-d
Hybrid
One-d
Pure
In place
Delta buffer
Hybrid
Multi-d
Pure
In place
Delta buffer
Hybrid
Benchmarking
One-d
Multi-d
Towards Practical Systems Implementation
One-d
Multi-d
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Radix table
Distance-bounded^[138]
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Projected space
GLIN_ALEX[121]
BRIN
SLBRIN[123]
Radix table
LSTI[29]
R-tree
IF-X[48]
Grid-based
Period-index [10]
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
LIMS[117]
MTO[26]
WISK[109]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
ELSI [71]
SageDB [25]
PLATON [134]

Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
  - → Learned Indexes for the One-dimensional Space
    - → Taxonomy of Learned Indexes
      - → Immutable vs. Mutable Indexes
      - Fixed vs. Dynamic Data Layouts
      - Pure vs. Hybrid Indexes
      - In-place vs. Delta Buffer Insertion Strategies
  - Learned Indexes for the Multi-dimensional Space
  - Open Challenges for Future Research

PURDUE UNIVERSITY

Dimension 1: Immutable vs. Mutable Indexes

- Given a Learned Index, can it support updates?
- Why are updates challenging?
  - The learned index takes significant time to train.
  - New data will require retraining because it changes the old learned ordering of the data.
- Classify learned indexes based on the ability to support updates:
- Immutable Learned Index:
  - Does not support inserts, updates, or deletes
- Mutable Learned Index:
  - Supports inserts, updates, or deletes

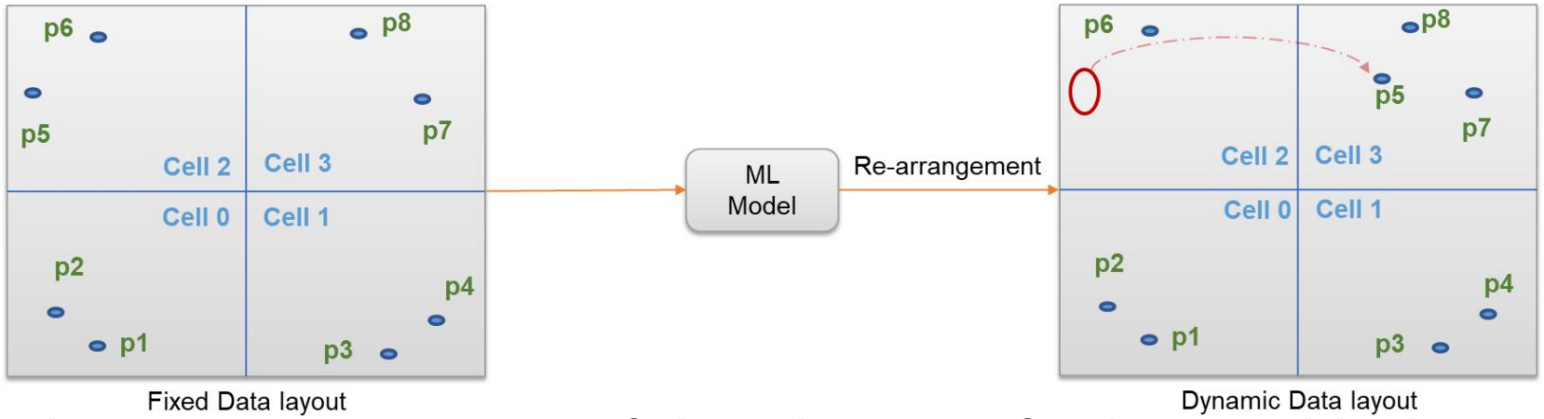
PURDUE UNIVERSITY

Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
- ✓ Learned Indexes for the One-dimensional Space
  - ✓ Taxonomy of Learned Indexes
    - ✓ Immutable vs. Mutable Indexes
    - → Fixed vs. Dynamic Data Layouts
    - Pure vs. Hybrid Indexes
    - In-place vs. Delta Buffer Insertion Strategies
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY

Dimension 2: Fixed vs. Dynamic Data Layouts



- Mainly, there are two types of data layouts for learned indexes:
  - Fixed Data Layout:
    - The layout of the data and the structure of the index are fixed before building the learned index.
  - Dynamic Data Layout:
    - The layout of the data is arranged and can be modified by the ML models while building and updating the learned index.

PURDUE UNIVERSITY

Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
- ✓ Learned Indexes for the One-dimensional Space
  - ✓ Taxonomy of Learned Indexes
    - ✓ Immutable vs. Mutable Indexes
    - ✓ Fixed vs. Dynamic Data Layouts
    - → Pure vs. Hybrid Indexes
    - In-place vs. Delta Buffer Insertion Strategies
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY

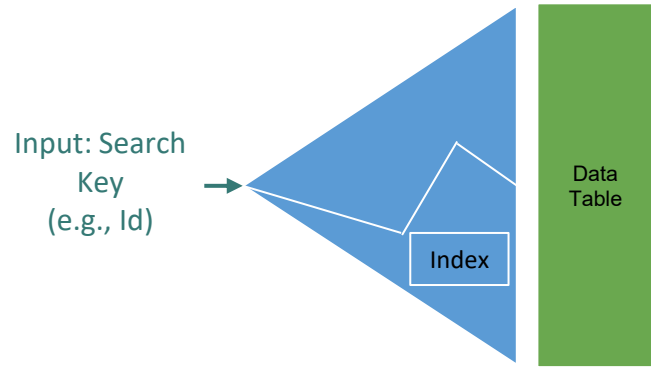
Dimension 3: Pure vs. Hybrid Indexes

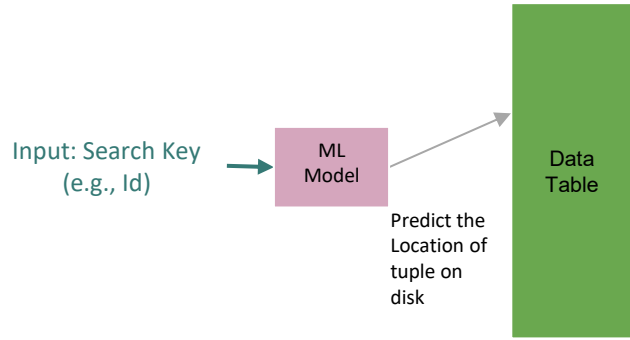
- Pure Learned Indexes:
  - Designed without leveraging a traditional index structure (e.g., a B-tree or an R-tree)
  - Replace a traditional index structure
- Hybrid Learned Indexes:
  - Combine a traditional index structure with ML models to build an ML-enhanced index structure

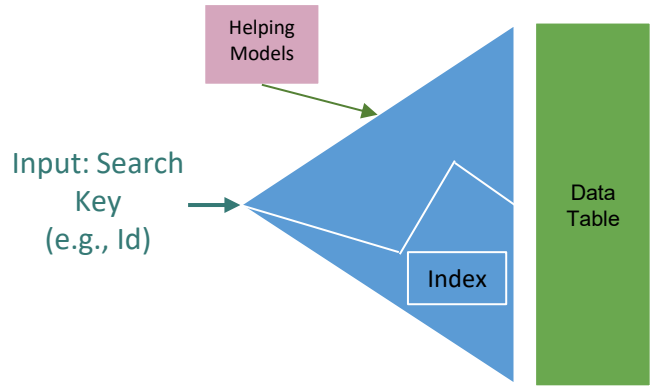
PURDUE UNIVERSITY

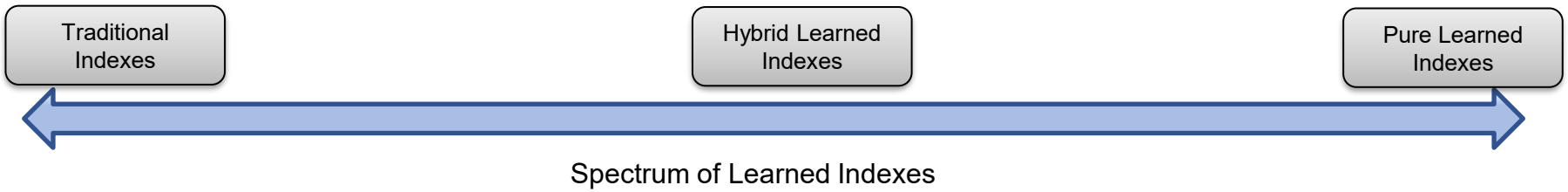
Dimension 3: Pure vs. Hybrid Indexes (Cont'd)

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









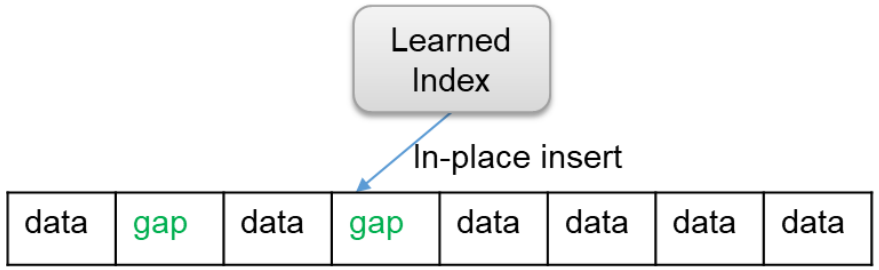
PURDUE UNIVERSITY

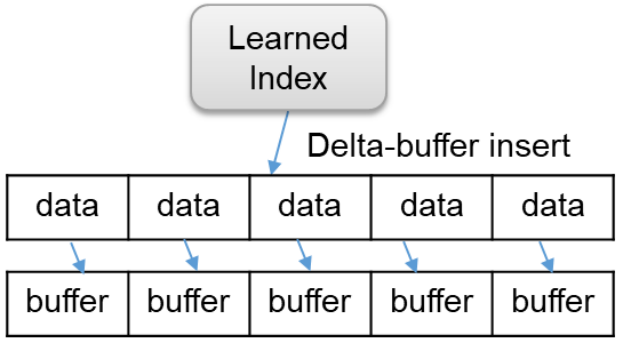
Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
- ✓ Learned Indexes for the One-dimensional Space
  - ✓ Taxonomy of Learned Indexes
    - ✓ Immutable vs. Mutable Indexes
    - ✓ Fixed vs. Dynamic Data Layouts
    - ✓ Pure vs. Hybrid Indexes
    - → In-place vs. Delta Buffer Insertion Strategies
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY

Dimension 4: In-place vs. Delta Buffer Insertion Strategies





- In-place:
  - Use ML models to find the position where the new data item should be inserted
  - Reserve some gaps in the index so that insert operations can be accommodated immediately
- Delta Buffer:
  - Employ fixed-size buffers to hold the new data items
  - Synchronize with the index structure periodically (e.g., during ML model re-training)

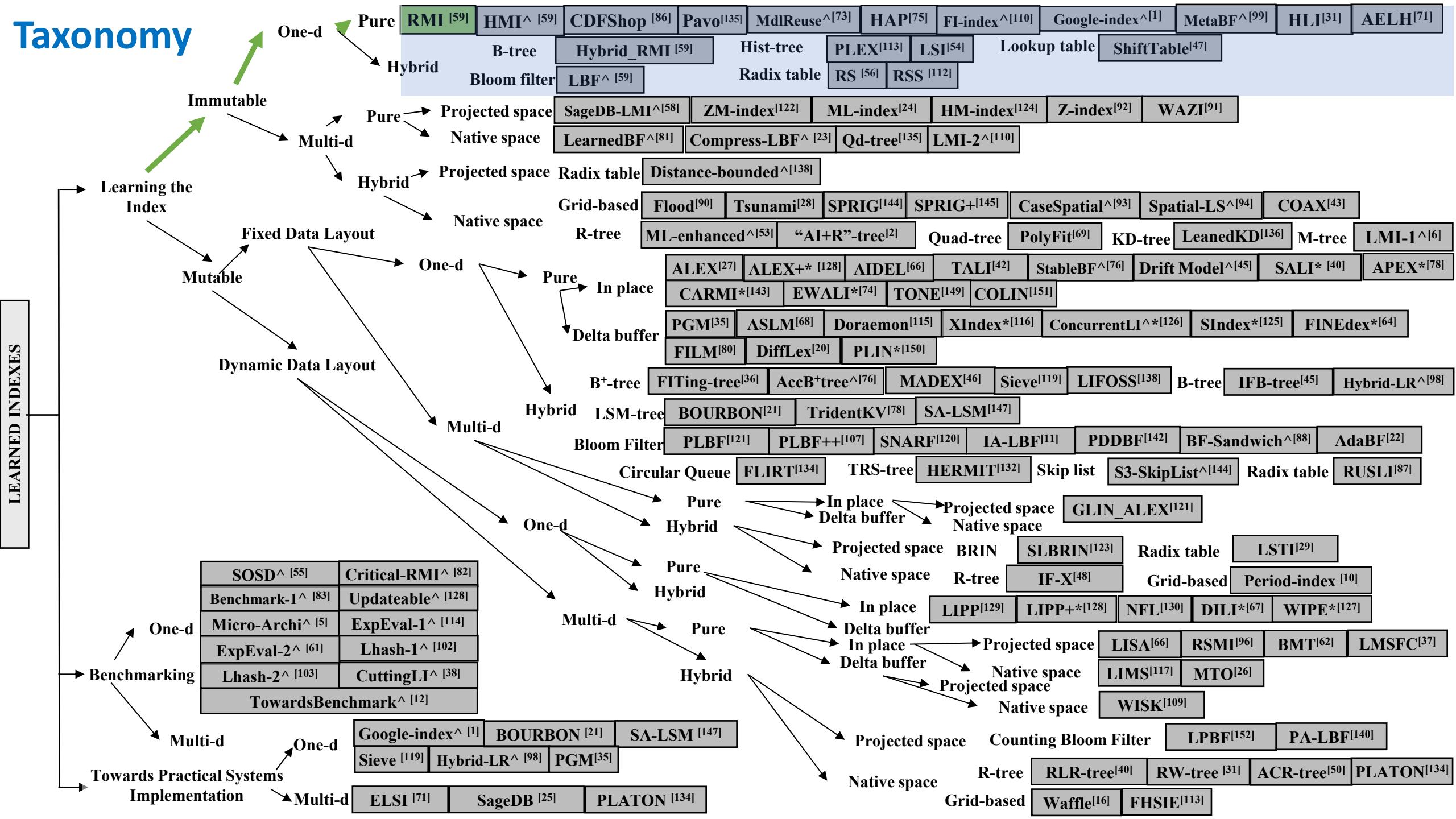
PURDUE UNIVERSITY

Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
- ✓ Learned Indexes for the One-dimensional Space
  - ✓ Taxonomy of Learned Indexes
    - ✓ Immutable vs. Mutable Indexes
    - ✓ Fixed vs. Dynamic Data Layouts
    - ✓ Pure vs. Hybrid Indexes
    - ✓ In-place vs. Delta Buffer Insertion Strategies
  - → Example One-dimensional Learned Indexes
- Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

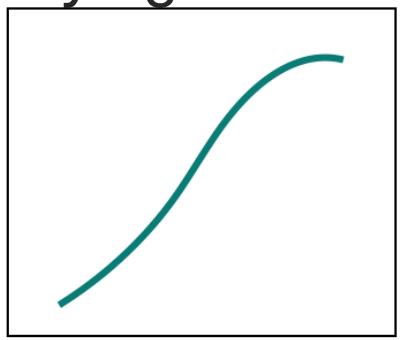
PURDUE UNIVERSITY

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]

Recursive Model Index (RMI) [SIGMOD'18]

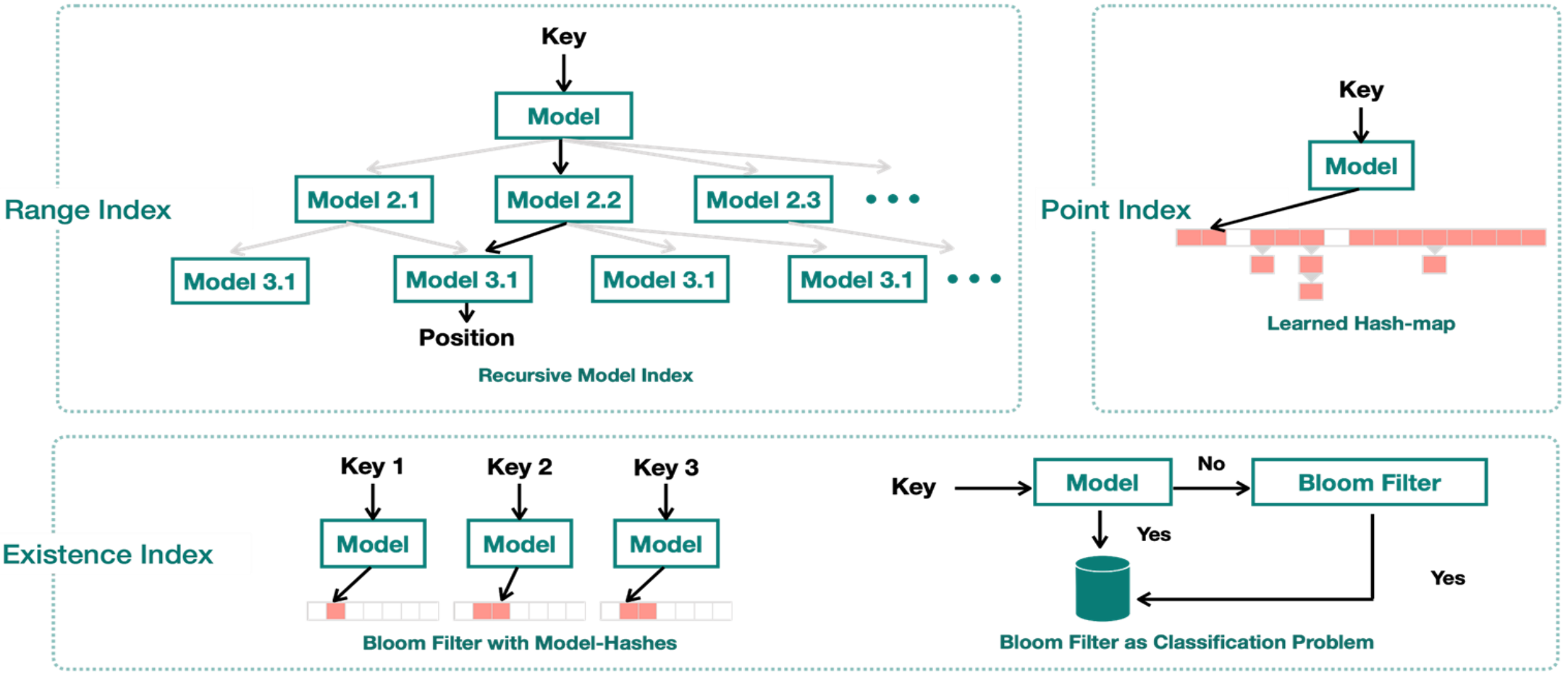
- An Immutable Pure Learned Index
- Approximate the Cumulative Distribution Function (CDF) of the underlying (sorted) data
- A multi-stage ML model
- Combine simpler ML models
  - The first stage model will make an initial prediction of the CDF for a specific key.
  - The next stage models will be selected to refine this initial prediction.
- Proposed Learned Index Structures: Range Index, Point Index, and Existence Index



PURDUE UNIVERSITY

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.

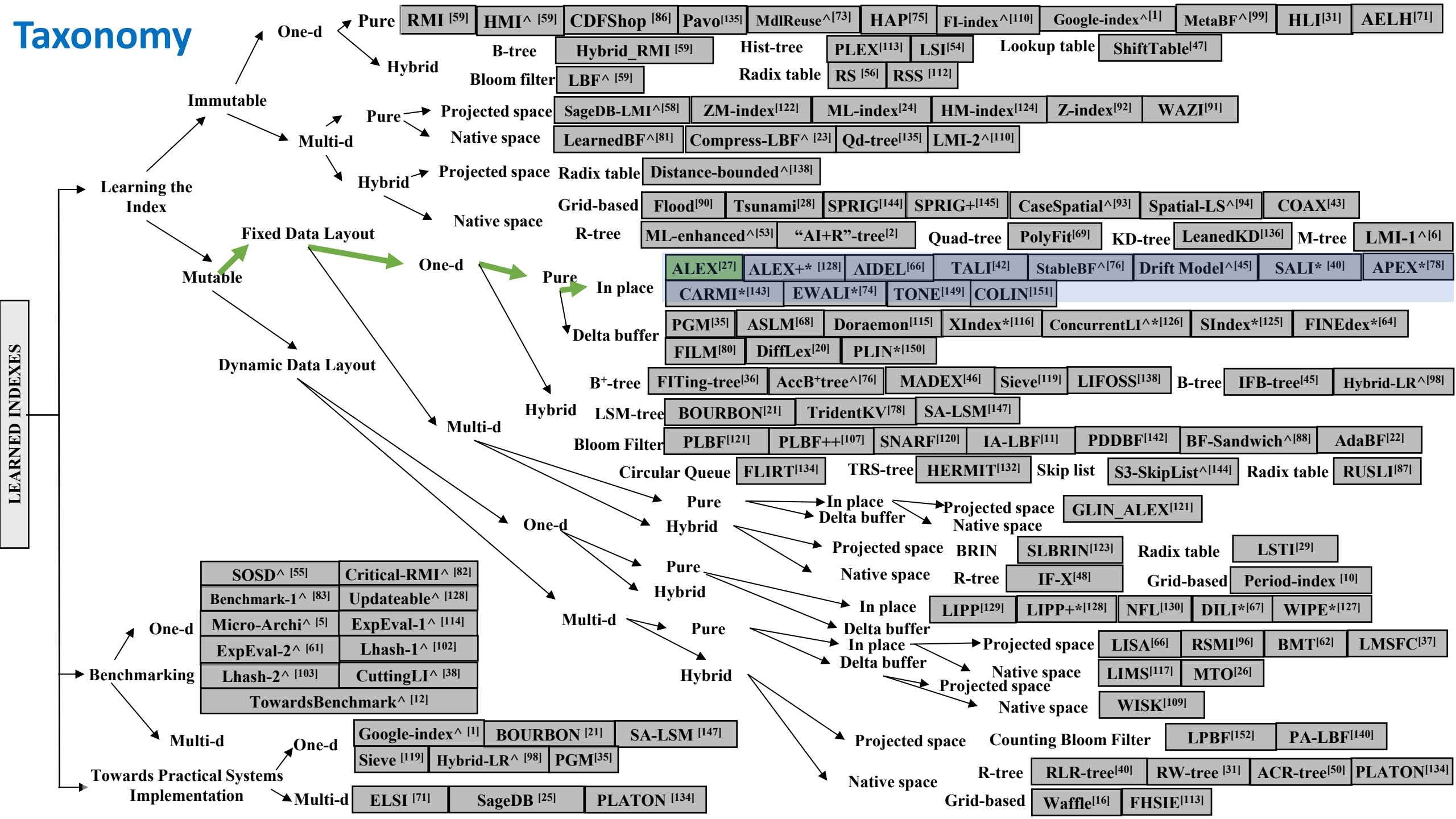
Recursive Model Index (RMI) [SIGMOD'18]



PURDUE UNIVERSITY

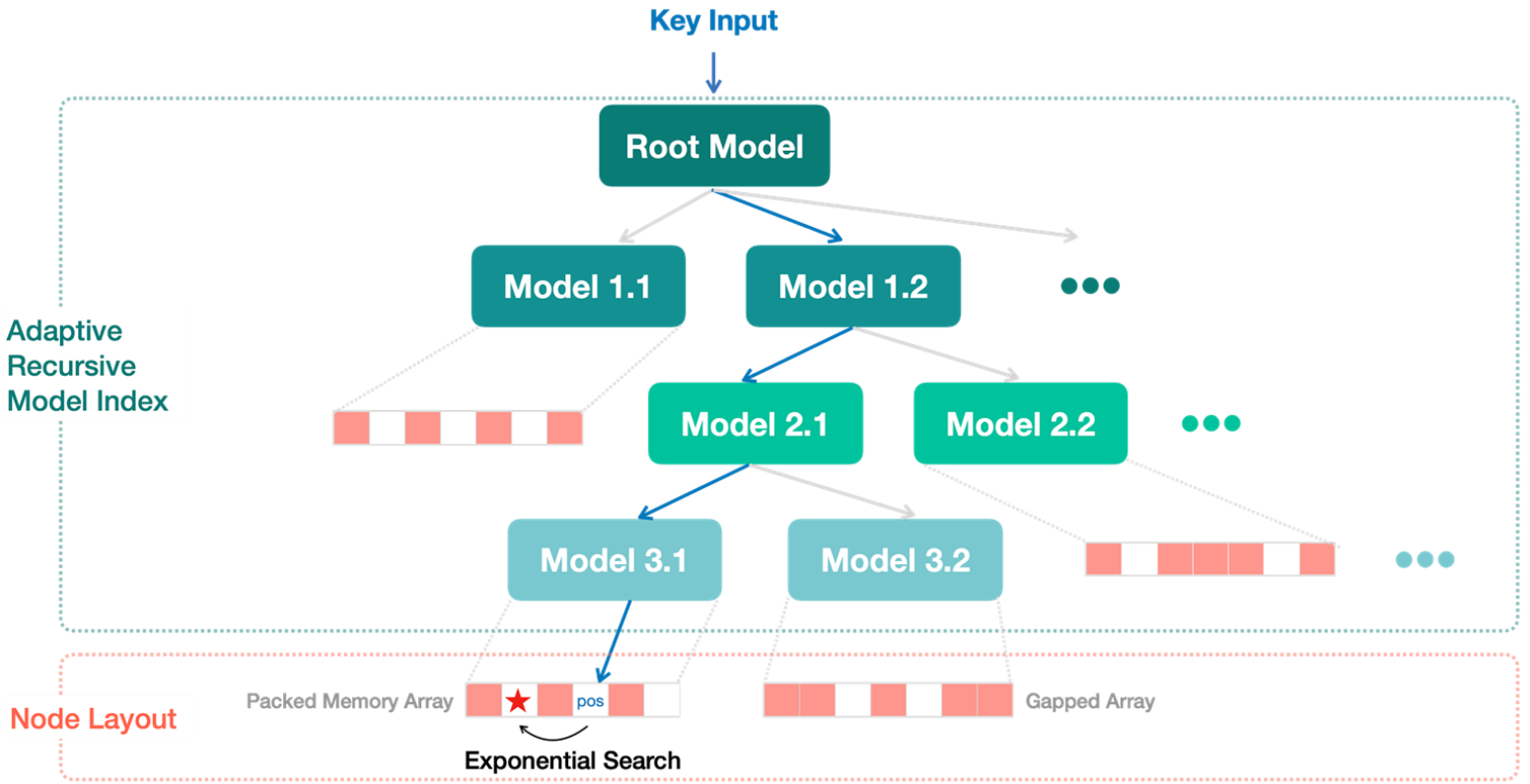
Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018.The case for learned index structures. In Proceedings of the 2018 International Conference on Management of Data. ACM, 489–504.

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hybrid
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

ALEX [SIGMOD'20]

- A Mutable Pure learned index with Fixed Data Layout
- Adopts an In-place insertion strategy
- Adaptive RMI as Model Hierarchy
- Linear Regression Model as Node
- Gapped Array or Packed Memory Array as Node Layout



PURDUE UNIVERSITY

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).

Outline

- ✓ Introduction and Historical Background
  - ✓ Indexing the Learned Models vs. Learning the Index
  - ✓ Evolution of Learned Indexes
- ✓ Learned Indexes for the One-dimensional Space
  - ✓ Taxonomy of Learned Indexes
    - ✓ Immutable vs. Mutable Indexes
    - ✓ Fixed vs. Dynamic Data Layouts
    - ✓ Pure vs. Hybrid Indexes
    - ✓ In-place vs. Delta Buffer Insertion Strategies
  - ✓ Example One-dimensional Learned Indexes
- → Learned Indexes for the Multi-dimensional Space
- Open Challenges for Future Research

PURDUE UNIVERSITY

Learned Multi-dimensional Indexes: Background

- One of the earliest distribution-aware spatial indexes can be found in:
  - 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., R-Tree

PURDUE UNIVERSITY

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
- → Motivation and Challenges
  - Projected vs. Native Space
  - Approaches
  - Immutable Indexes
  - Mutable Indexes with Fixed Data Layouts
  - Mutable Indexes with Dynamic Data Layouts
- Open Challenges for Future Research

PURDUE UNIVERSITY

Learned Multi-dimensional Indexes: Challenges

- 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 arrange the data?
  - Affect range query time and model accuracy

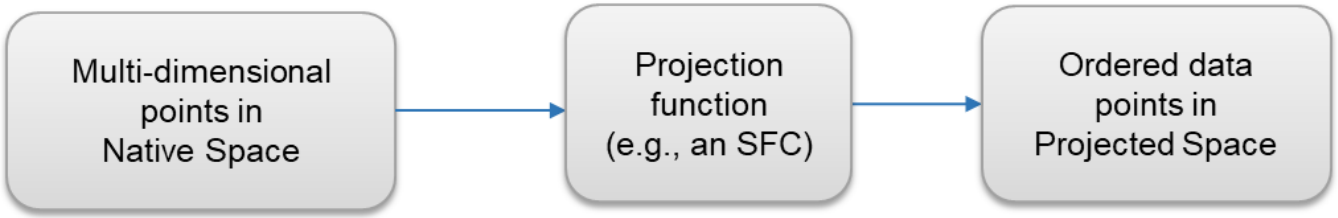
Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
  - ✓ Motivation and Challenges
  - → Projected vs. Native Space
  - Approaches
  - Immutable Indexes
  - Mutable Indexes with Fixed Data Layouts
  - Mutable Indexes with Dynamic Data Layouts
- Open Challenges for Future Research

PURDUE UNIVERSITY

Learned Multi-dimensional Indexes: Projected vs. Native Space

- Projected Space:
  - Multi-dimensional data is projected or linearized into a projected space using a projection function or a Space Filling Curve (e.g., Z-order, Hilbert-curve).



- Native Space:
  - ML models are trained on the original representation of the multi-dimensional data.

PURDUE UNIVERSITY

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
  - ✓ Motivation and Challenges
    - ✓ Projected vs. Native Space
    - → Approaches
    - Immutable Indexes
    - Mutable Indexes with Fixed Data Layouts
    - Mutable Indexes with Dynamic Data Layouts
  - Open Challenges for Future Research

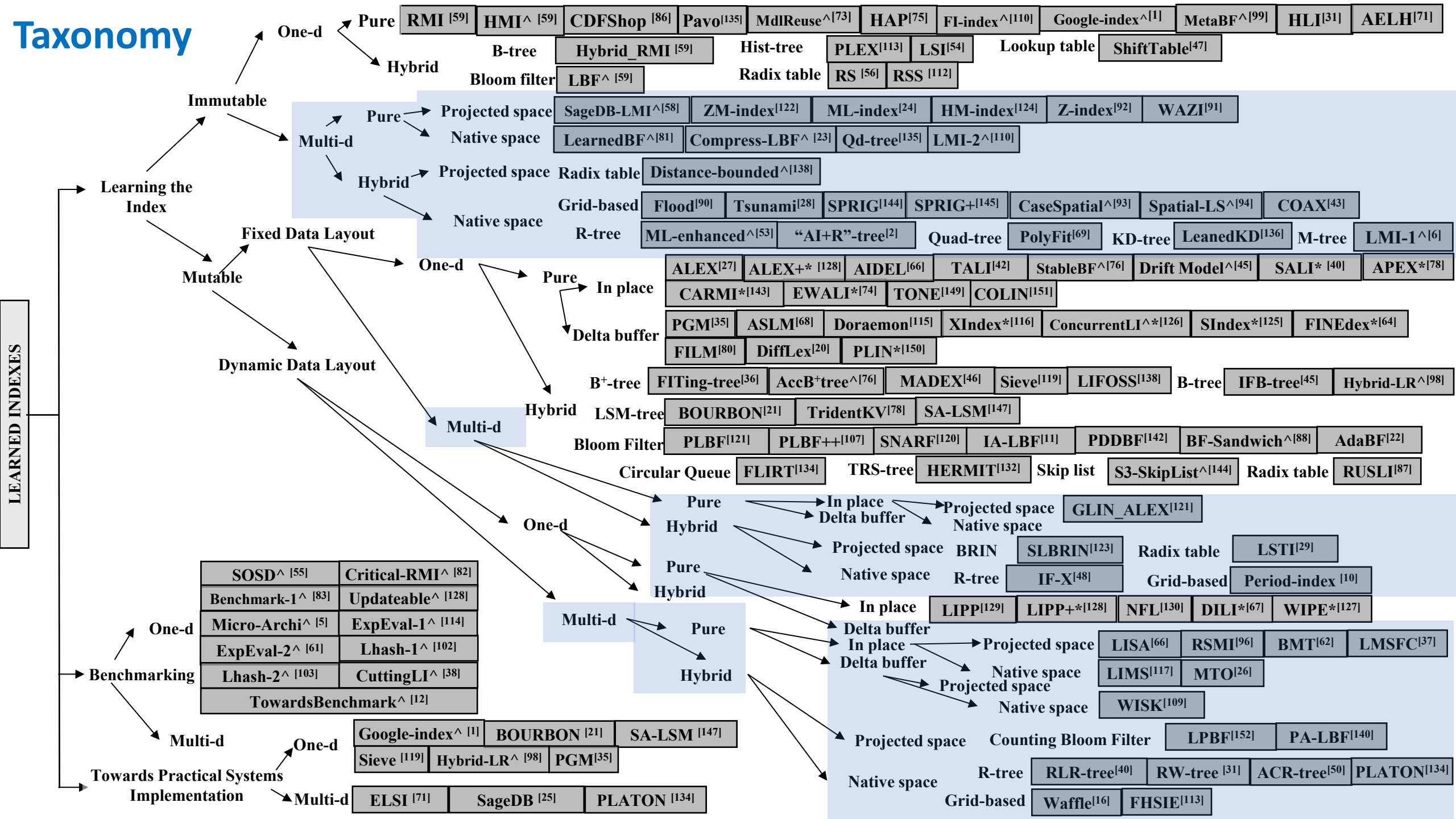
PURDUE UNIVERSITY®

Learned Multi-dimensional Indexes: Approaches

- The concept of learned one-dimensional indexes has been extended in multi-dimensional spaces mainly by four approaches:
  - Approach 1:
    - Augment traditional multi-dimensional index with ML models
    - Keep the traditional index
  - Approach 2:
    - Project multi-dimensional data into one-dimensional space and use a learned one-dimensional index
    - Replace the traditional index
  - Approach 3:
    - Apply a one-dimensional learned index in the native multi-dimensional space without using a projection function
    - Replace the traditional index
  - Approach 4:
    - Design learned multi-dimensional indexes from scratch
      - Create a data layout that is suitable for learning
    - Replace the traditional index

PURDUE UNIVERSITY®

Taxonomy

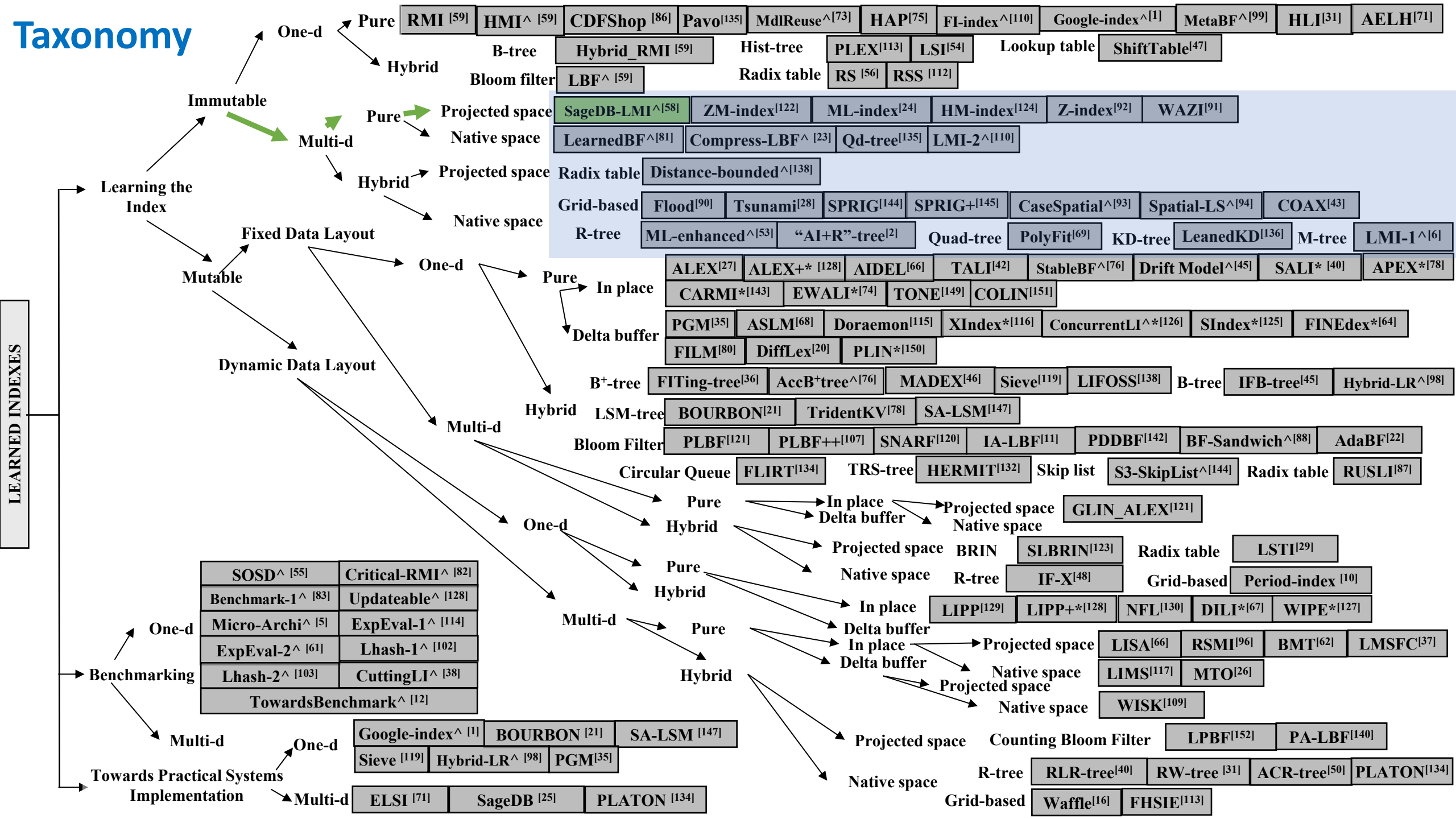

LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hybrid
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
One-d
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Multi-d
Pure
Delta buffer
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Hybrid
Delta buffer
Native space
LIMS[117]
MTO[26]
Projected space
Native space
WISK[109]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
  - ✓ Motivation and Challenges
    - ✓ Projected vs. Native Space
    - ✓ Approaches
  - → Immutable Indexes
    - Mutable Indexes with Fixed Data Layouts
    - Mutable Indexes with Dynamic Data Layouts
  - Open Challenges for Future Research

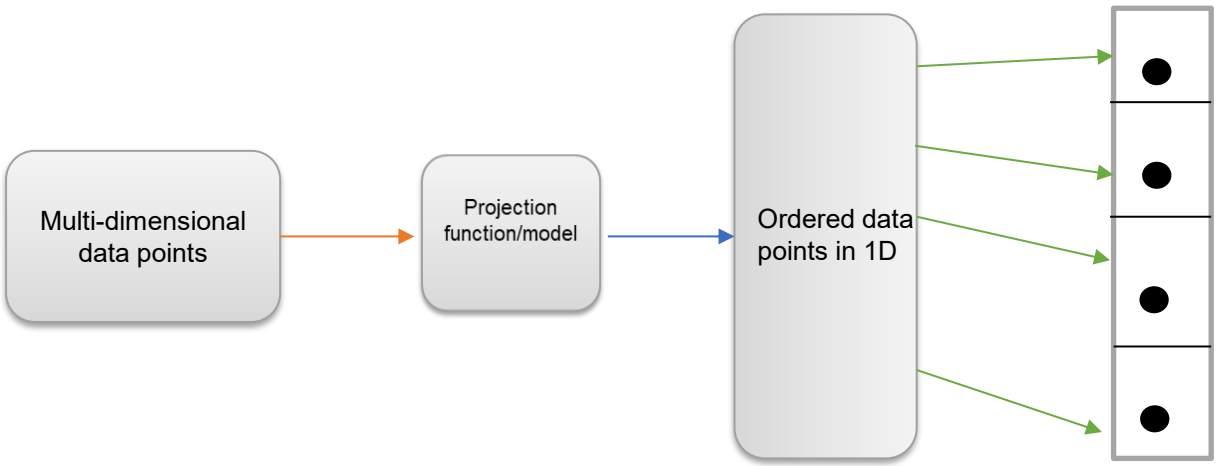
PURDUE UNIVERSITY

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hybrid
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]

Sagedb: A Learned Database System [CIDR'19]

- Propose a learned multi-dimensional index by applying RMI in a projected space
- Step-1: Project the multi-dimensional data points into one-dimensional space
  - Consecutively sorting and partitioning the points along a sequence of dimensions (e.g., first x-dimension then y-dimension) into uniformly-sized cells
  - Produce a layout that is efficient to compute and easily learnable
  - Comparing with Z-order which is difficult to learn
- Step-2: Use a trained CDF model (e.g., RMI) to predict the physical location of the point

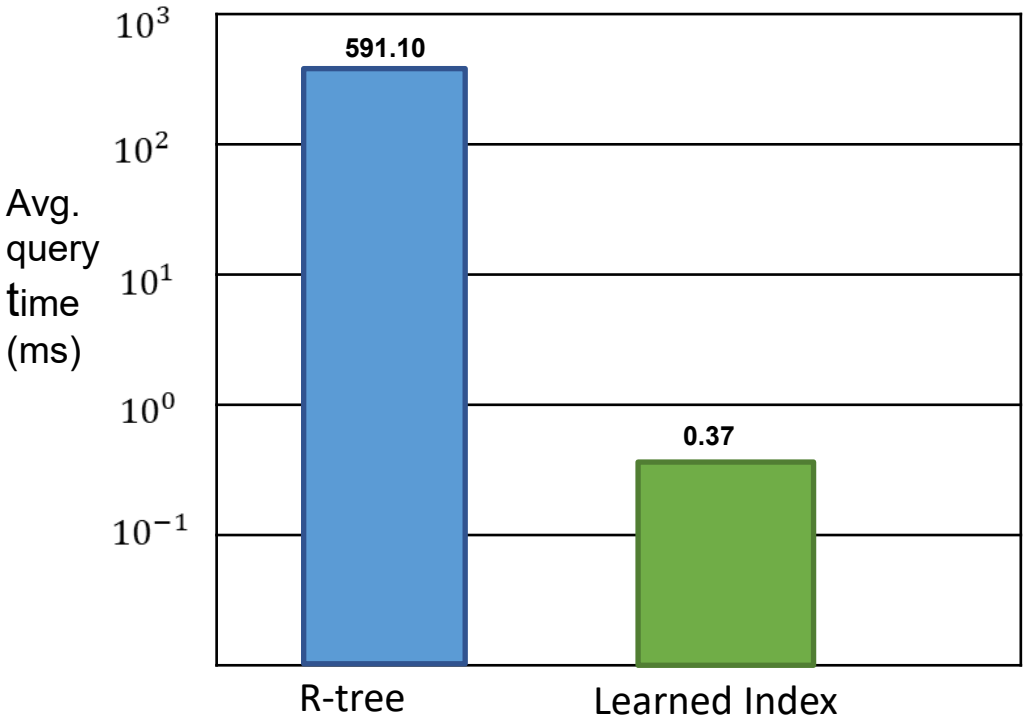


PURDUE UNIVERSITY

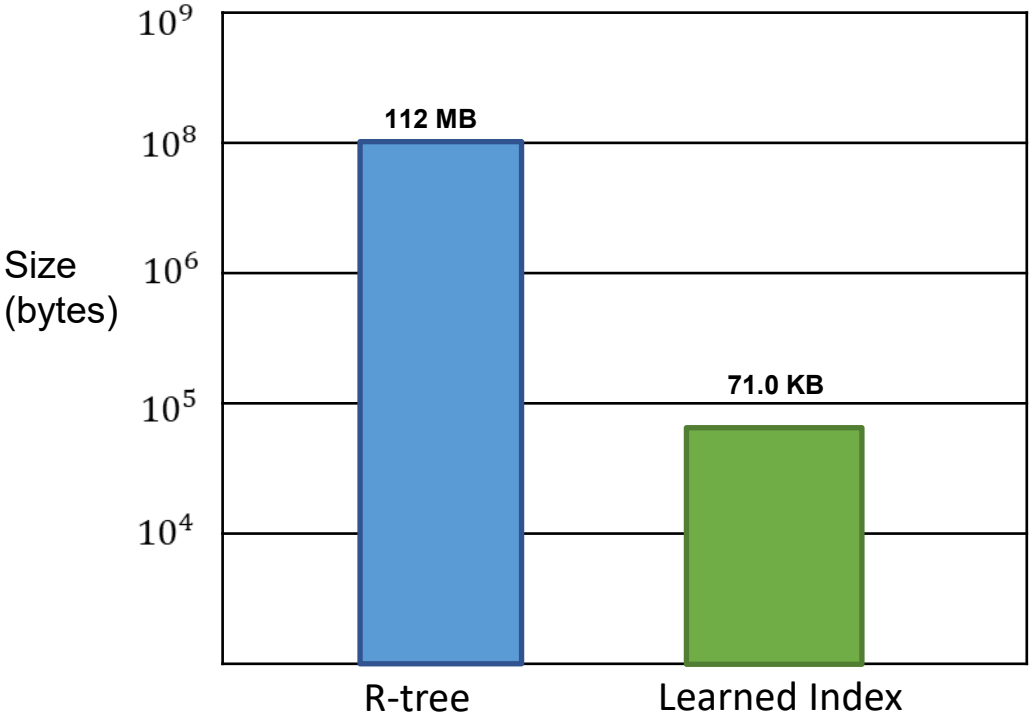
Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H Chi, Jialin Ding, Ani Kristo,Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019.Sagedb: A learned database system, CIDR, (2019).

Sagedb: A Learned Database System [CIDR'19] (Cont'd)

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



Result on average query time

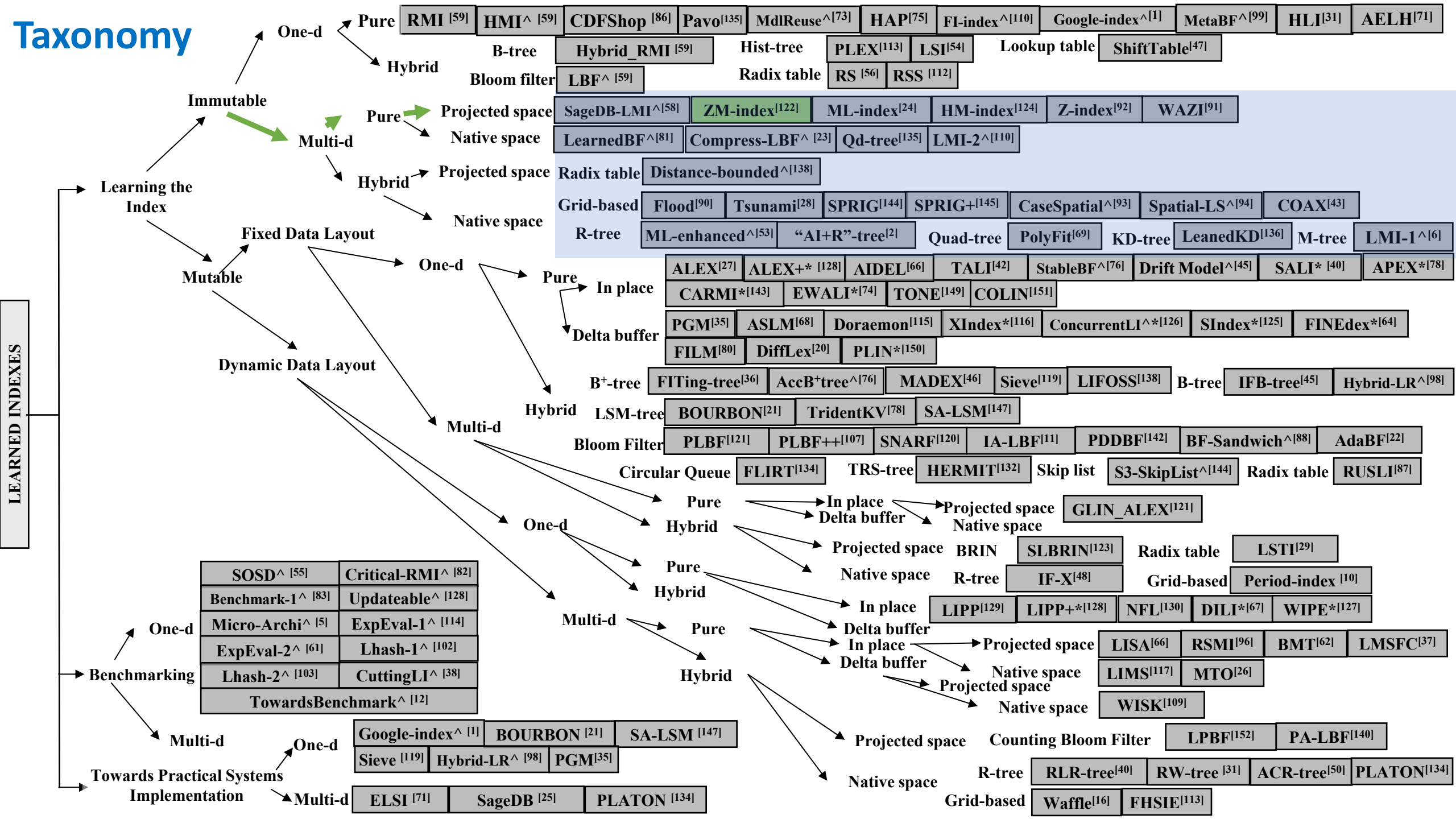


Result on index size

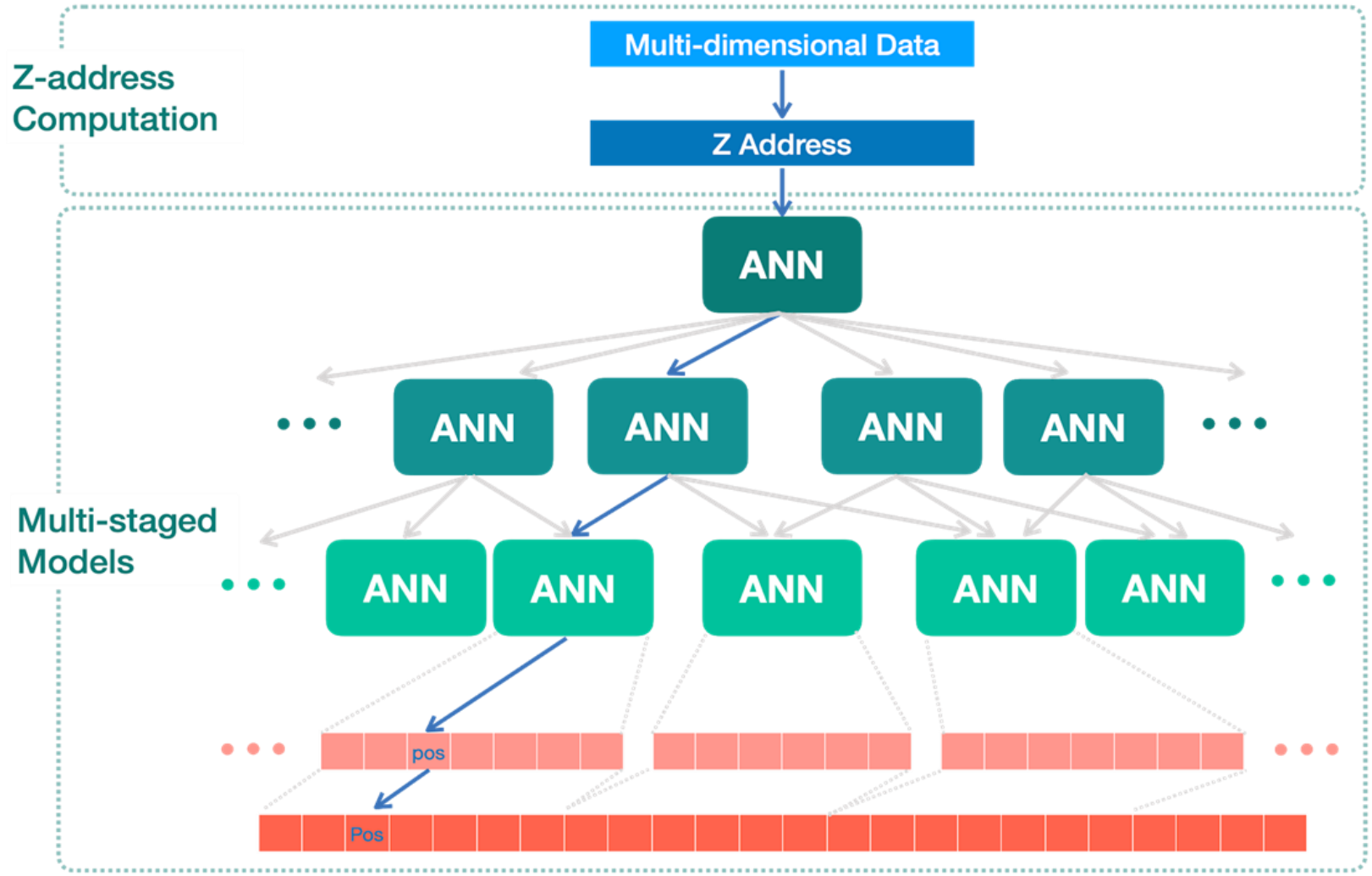
PURDUE UNIVERSITY®

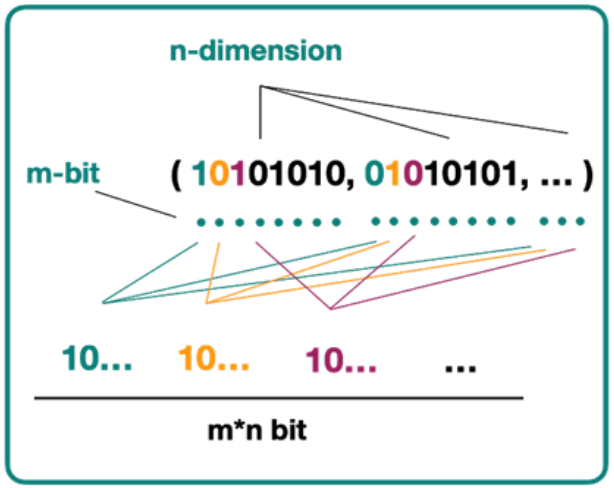
Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H Chi, Jialin Ding, Ani Kristo,Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019.Sagedb: A learned database system, CIDR, (2019).

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

Z-Order Model Index (ZM-Index) [MDM'19]





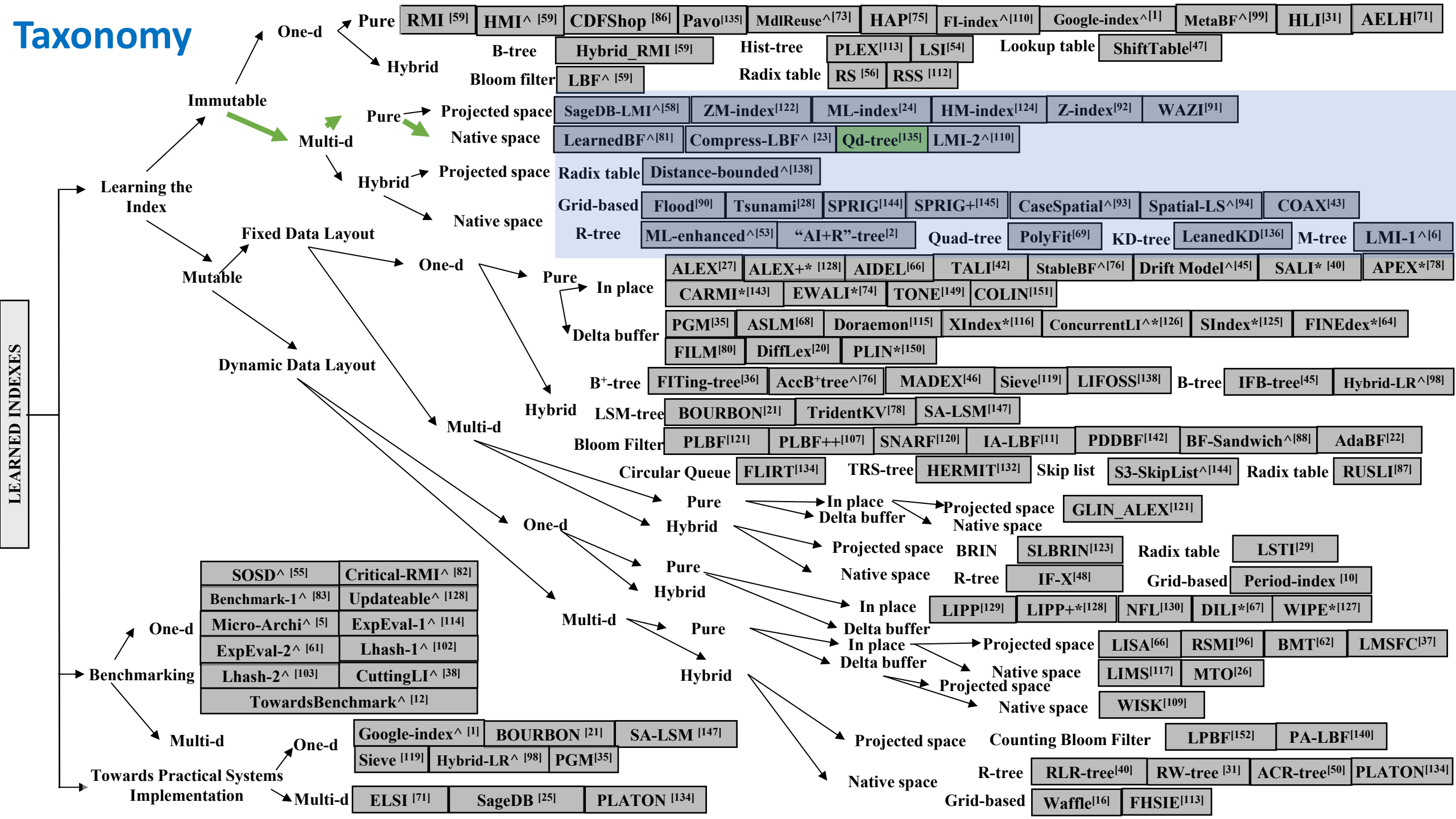
Z-address computation of n-d vector

- ZM-Index:
  - Use the Z-order to map the multi-dimensional values into the one-dimensional space
  - Use a multi-staged model (e.g., RMI) for learning
  - Support point and range queries

PURDUE UNIVERSITY

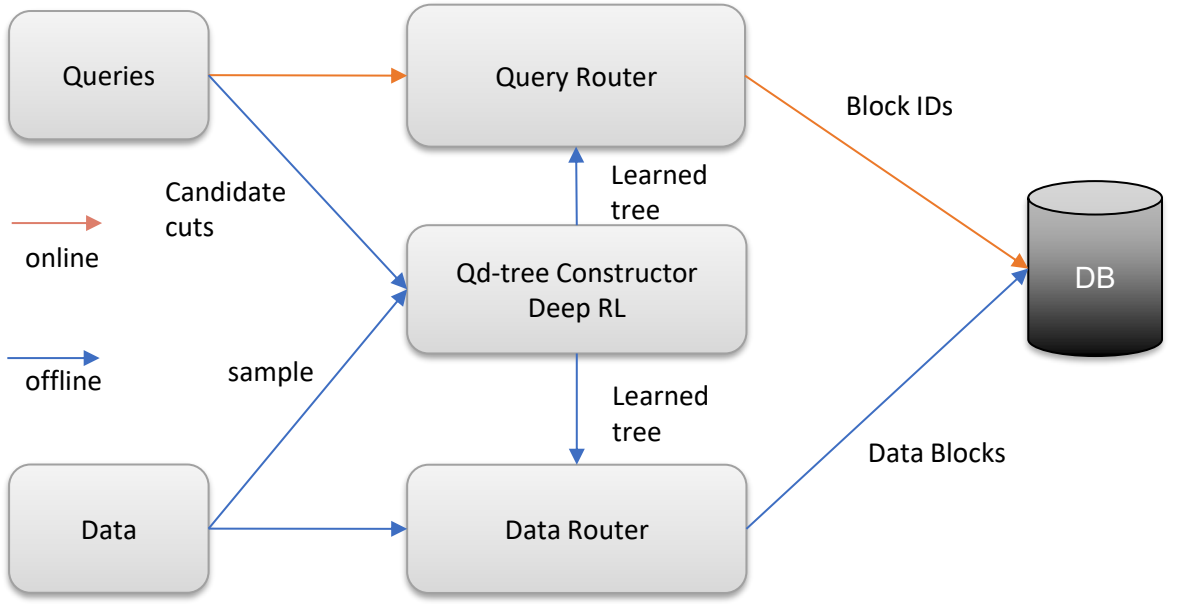
Haixin Wang, Xiaoyi Fu, Jianliang Xu, and Hua Lu. 2019. Learned Index for Spatial Queries. In2019 20th IEEE International Conference on Mobile Data Management(MDM). IEEE, 569–574.

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
B-tree
Hybrid_RMI [59]
Bloom filter
LBF^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

Query-data Routing Tree (Qd-tree) [SIGMOD'20]

- For disk-based systems, an important performance metric is:
  - The number of data blocks accessed by a query.
- Minimize the number of blocks/records accessed by a given workload
  - Generate block-level data layouts with high I/O performance
- QD-trees are neural network-generated decision trees that
  - Recursively partition the data space into smaller subspaces.
- Use deep Reinforcement Learning (RL) to create Qd-trees
  - Proximal Policy Optimization (PPO)

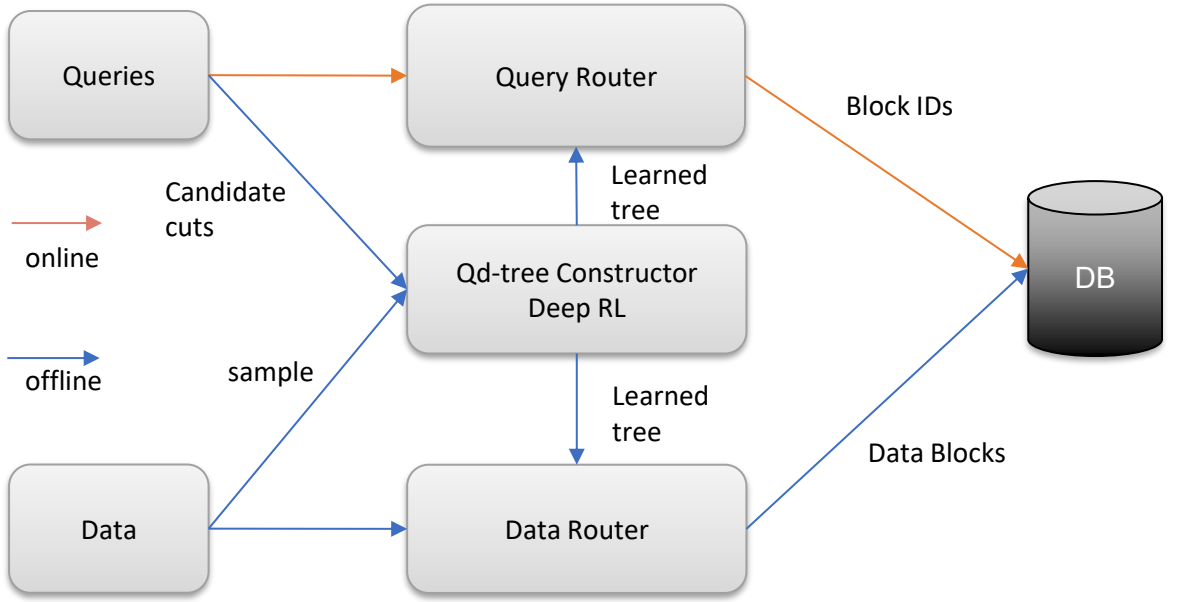


System Architecture

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 the 2020 ACM SIGMOD International Conference on Management of Data. 193–208.

Query-data Routing Tree (Qd-tree) [SIGMOD'20] (Cont'd)

- Experiments over benchmark and real workloads
  - Compared to existing 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

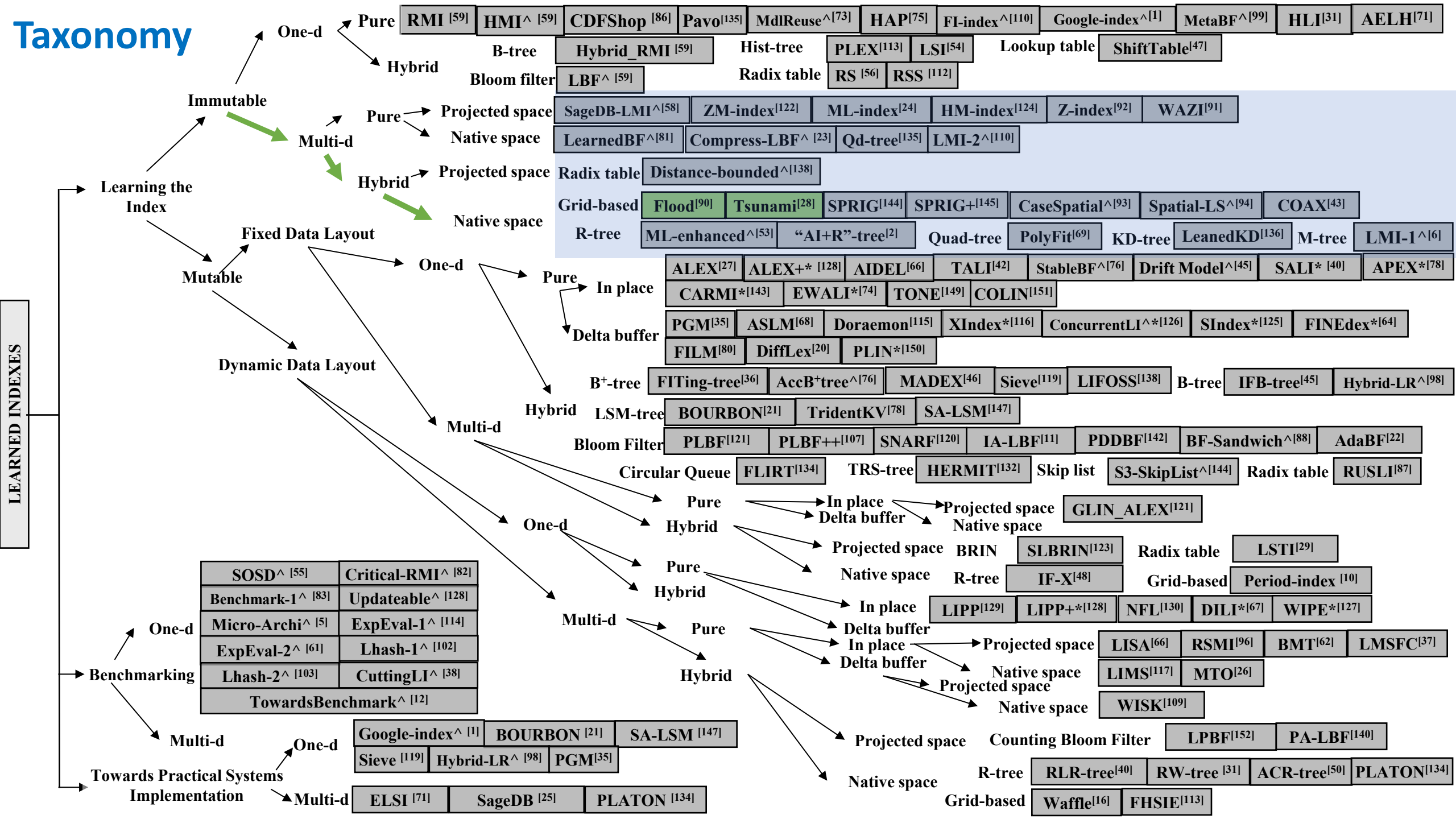


System Architecture

PURDUE UNIVERSITY

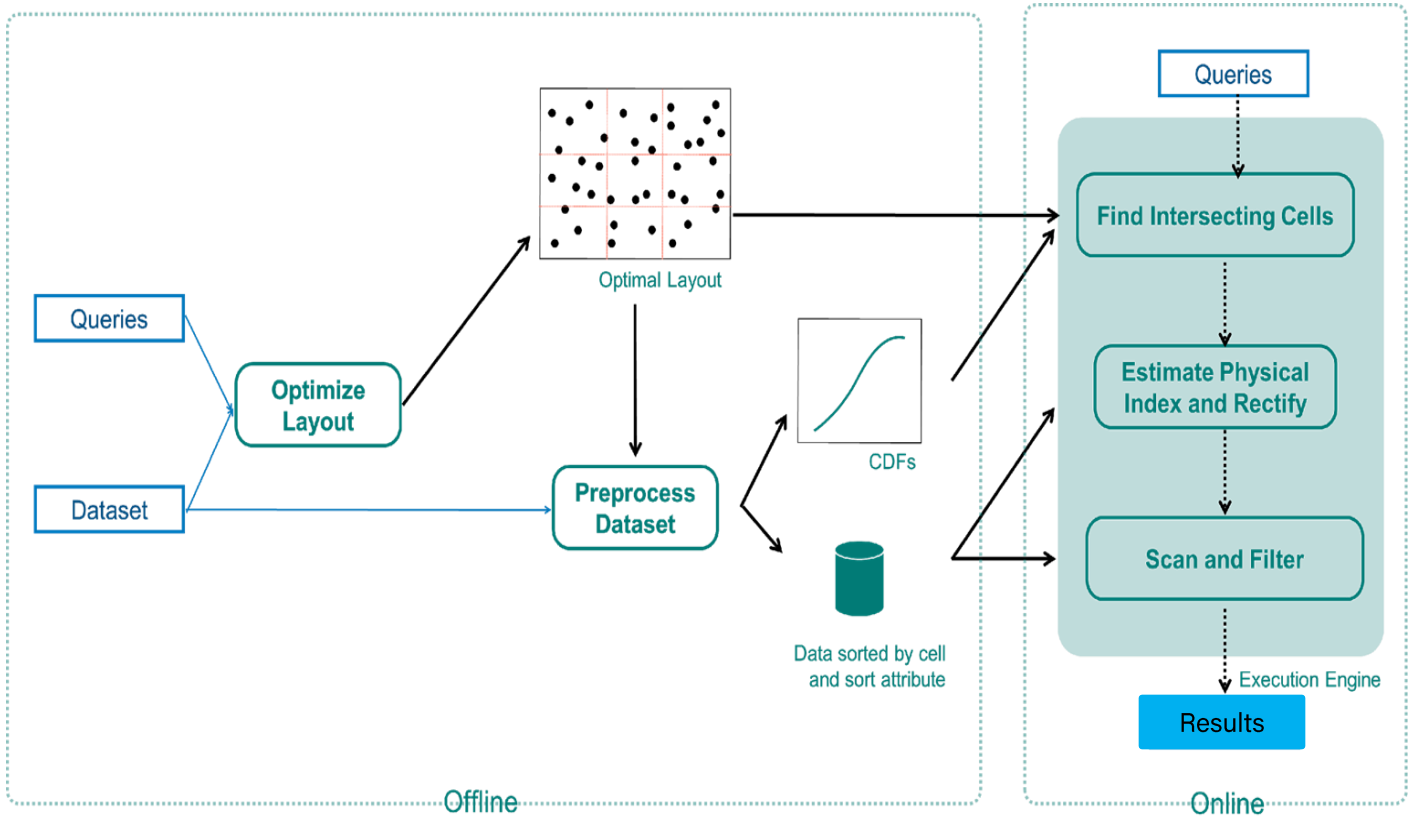
Zongheng Yang, Badrish Chandramouli, Chi Wang, Johannes Gehrke, Yinan Li,Umar Farooq Minhas, Per-Åke Larson, Donald Kossmann, and Rajeev Acharya.2020. Qd-tree: Learning Data Layouts for Big Data Analytics. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 193–208.

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
B-tree
Hybrid_RMI [59]
Bloom filter
LBF^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

Flood [SIGMOD'20]

- A read-optimized grid-based learned multi-dimensional index
- Co-optimize the data layout and the index structure
  - For a particular data and query distribution
- Two components:
  - Offline (pre-processing)
    - Pick an optimal layout
    - Create an index based on that layout
  - Online
    - Query execution

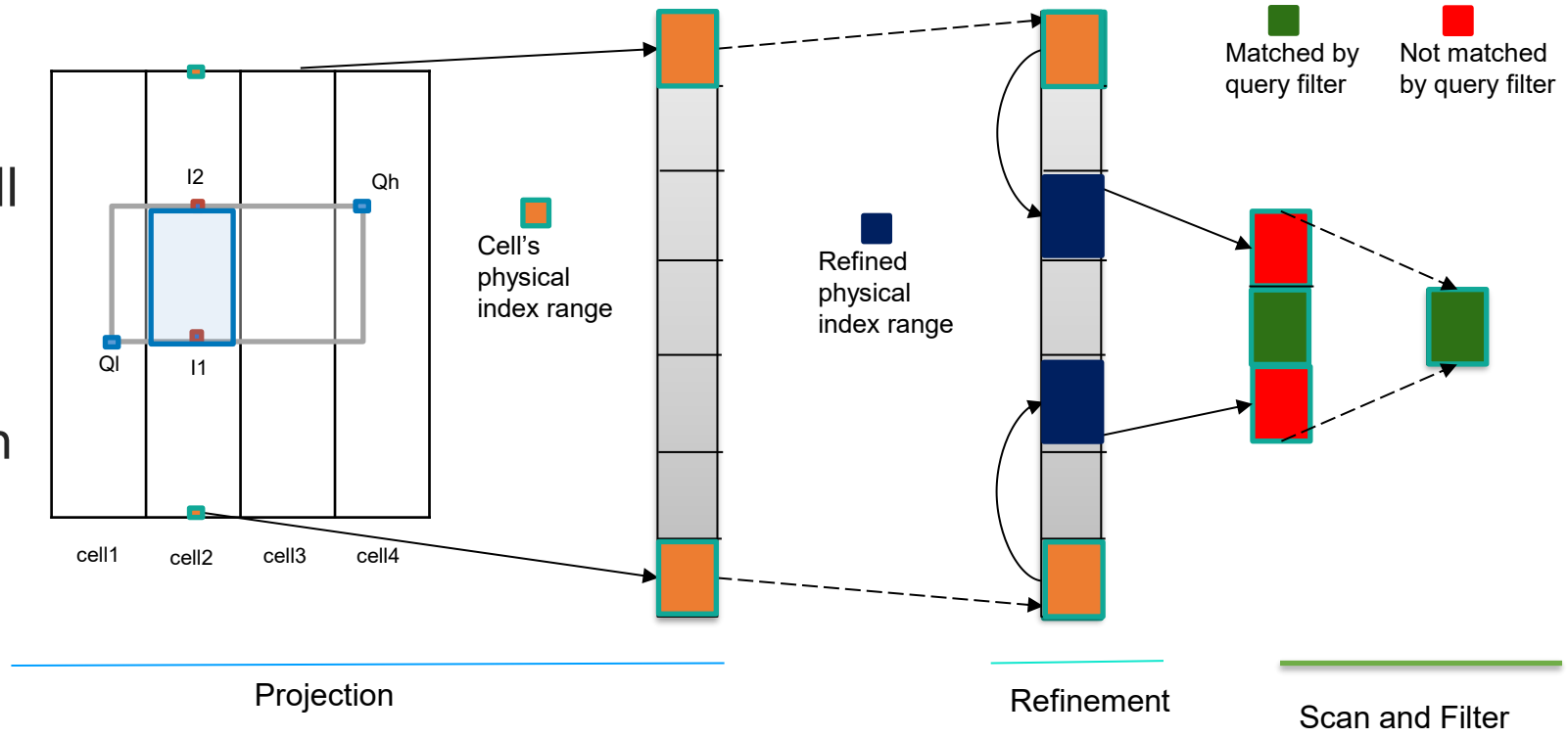


Flood's System Architecture

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

Flood [SIGMOD'20] (Cont'd)

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



Flood's Workflow

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

Flood [SIGMOD'20] (Cont'd)

- Experimental Results:
  - Outperform optimally tuned spatial indexes
  - Use 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

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

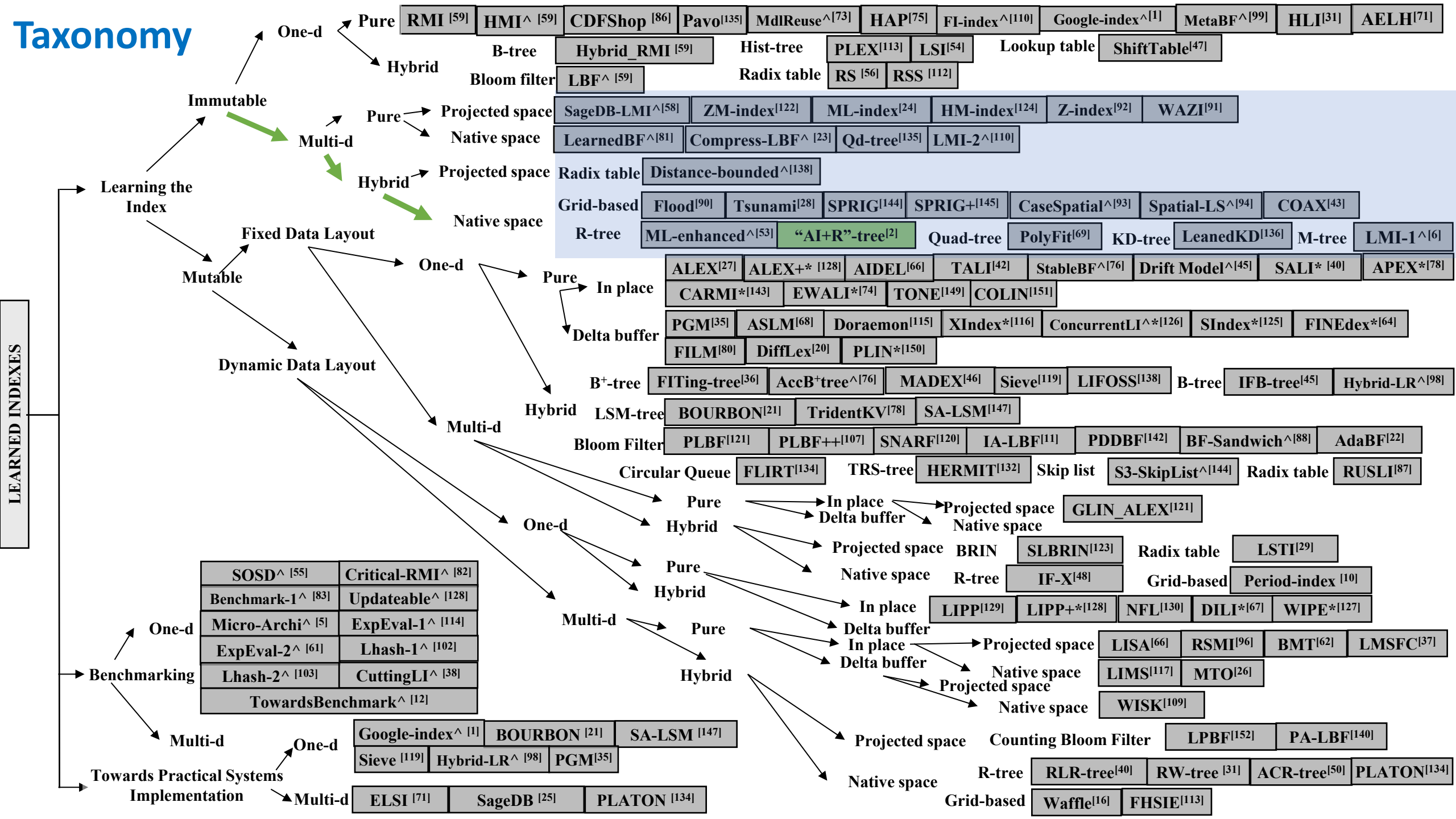
Tsunami [VLDB'20]

- Extend the idea of Flood to overcome its limitations over skewed query workload and correlated data
- Adaptable to changes in workload
- Scales across data size, query selectivity, and dimensionality
- Outperform Flood in scenarios with skewed query workloads and correlated data

PURDUE UNIVERSITY®

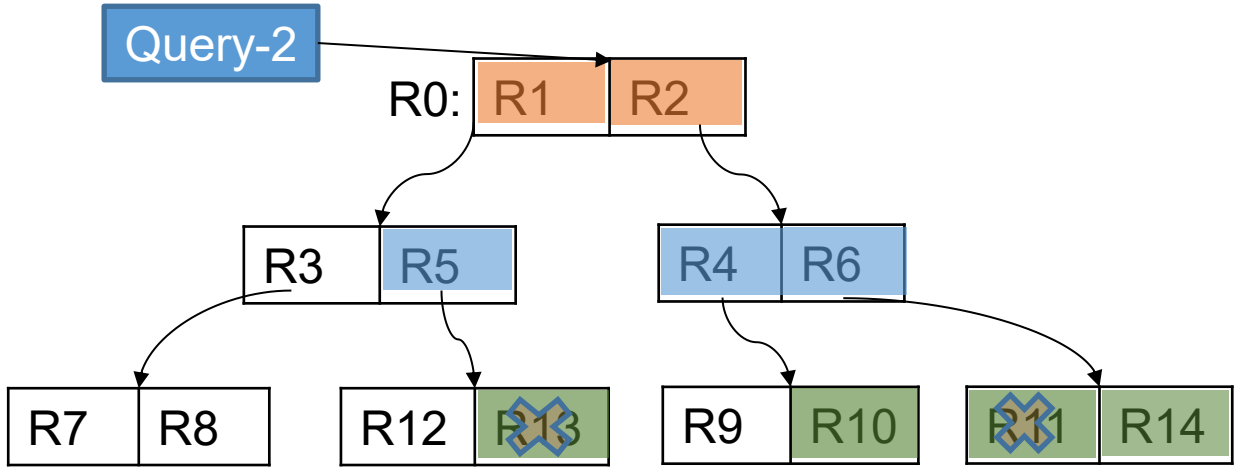
Ding, Jialin, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. "Tsunami: a learned multi-dimensional index for correlated data and skewed workloads." Proceedings of the VLDB Endowment 14, no. 2 (2020): 74-86.

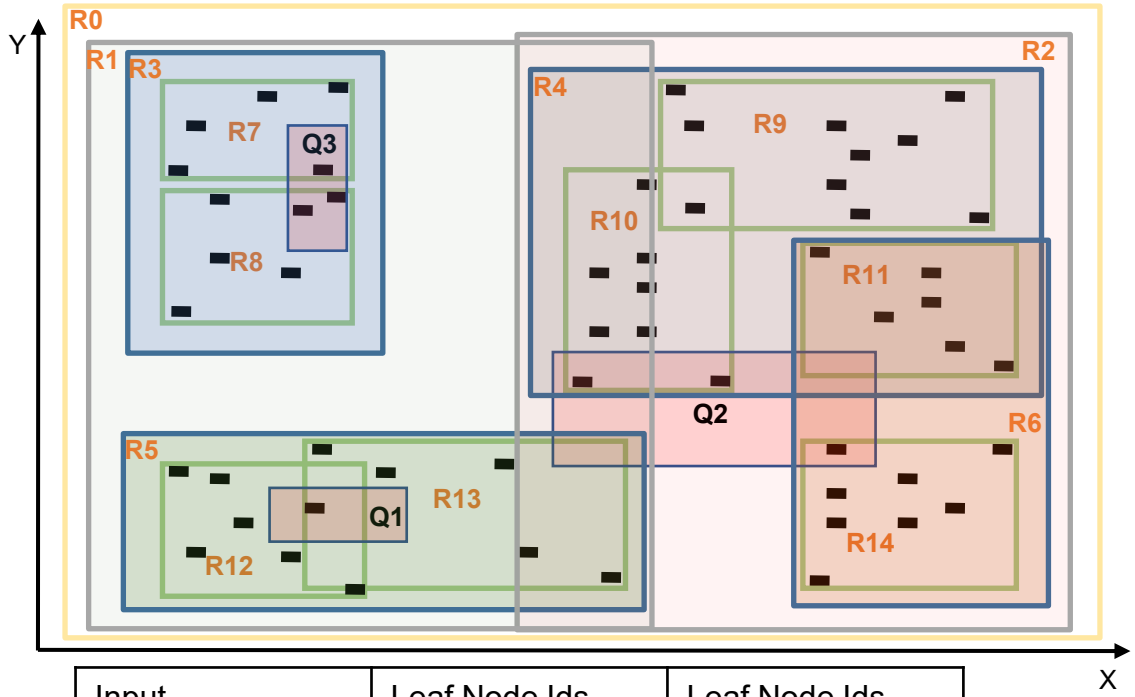
Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

The AI+R-tree [MDM'22]

- Problem Formulation:
  - Given a range query Q(X_min, Y_min, X_max, Y_max) as input, can we directly predict the true leafNodeIds in the R-Tree that contain the output data objects?



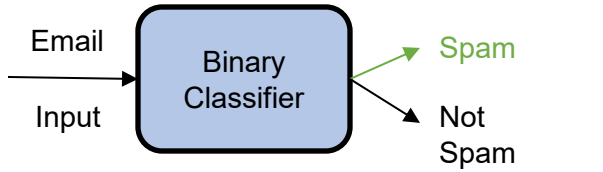


Input	Leaf Node Ids that will be searched by R-Tree	Leaf Node Ids that contains the actual data
Query-1	R12, R13	R13
Query-2	R10, R11, R13, R14	R10, R14

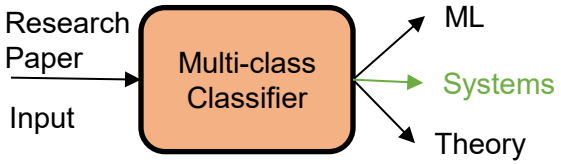
PURDUE UNIVERSITY

The AI+R-tree [MDM'22] (Cont'd)

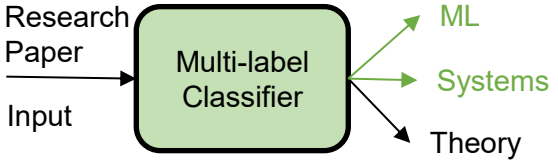
- Three different types of classification tasks:



Total number of classes = 2
For an input, the number of output classes = 1

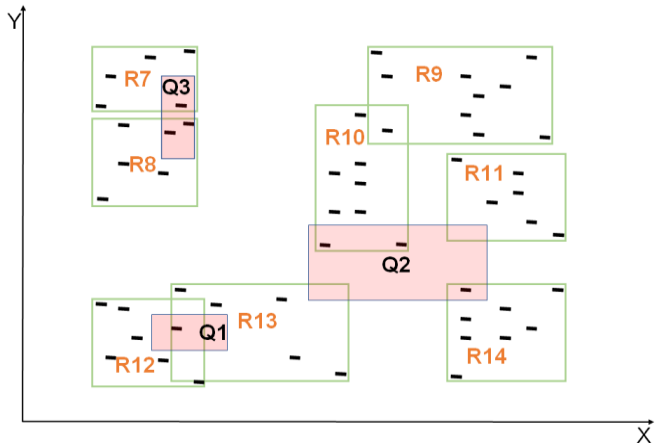


Total number of classes > 2
For an input, the number of output classes = 1

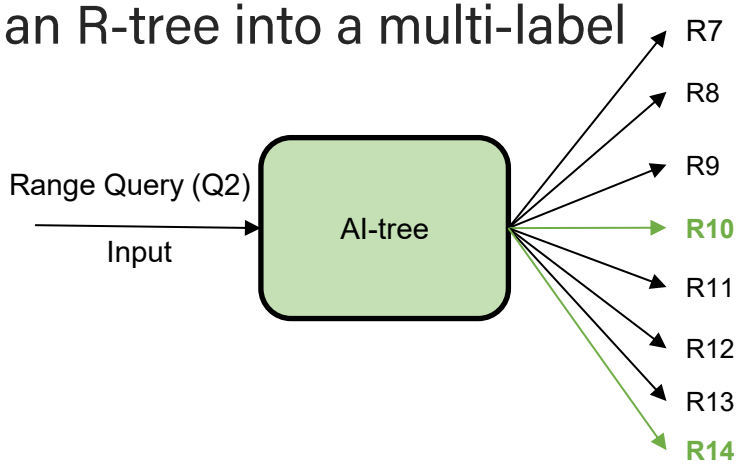


Total number of classes > 2
For an input, the number of output classes ≥ 0

- Proposal:
  - The AI-tree which transforms the search operation of an R-tree into a multi-label classification task.



An example of leaf nodes in an R-tree



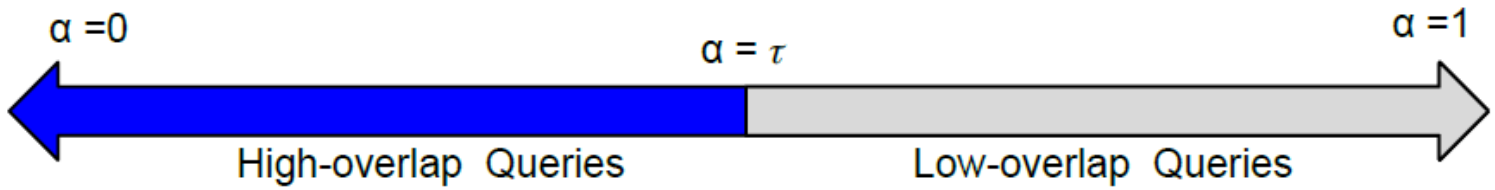
Total number of classes > 2
For an input, the number of output classes ≥ 0

PURDUE UNIVERSITY

Al-Mamun, Abdullah, Haider, Ch Md Rakin, Jianguo Wang, and Walid G. Aref. "The "AI+ R"-tree: An Instance-optimized R-tree." In 2022 23rd IEEE International Conference on Mobile Data Management (MDM), pp. 9-18. IEEE, 2022.

The AI+R-tree [MDM'22] (Cont'd)

- An overlap ratio α quantifes the degree of extraneous leaf node accesses required by a range query.
- For a range query:
  - $\alpha = \frac{Number\ of\ true\ leaf\ nodes\ that\ actually\ contains\ the\ output\ data\ objects}{Number\ of\ leaf\ nodes\ searched\ by\ the\ R-Tree}$
  - α is in the range [0,1]
- High-overlap Queries:
  - A range query is declared as a "high-overlap query" if the value of α is less than a pre-defined threshold τ.

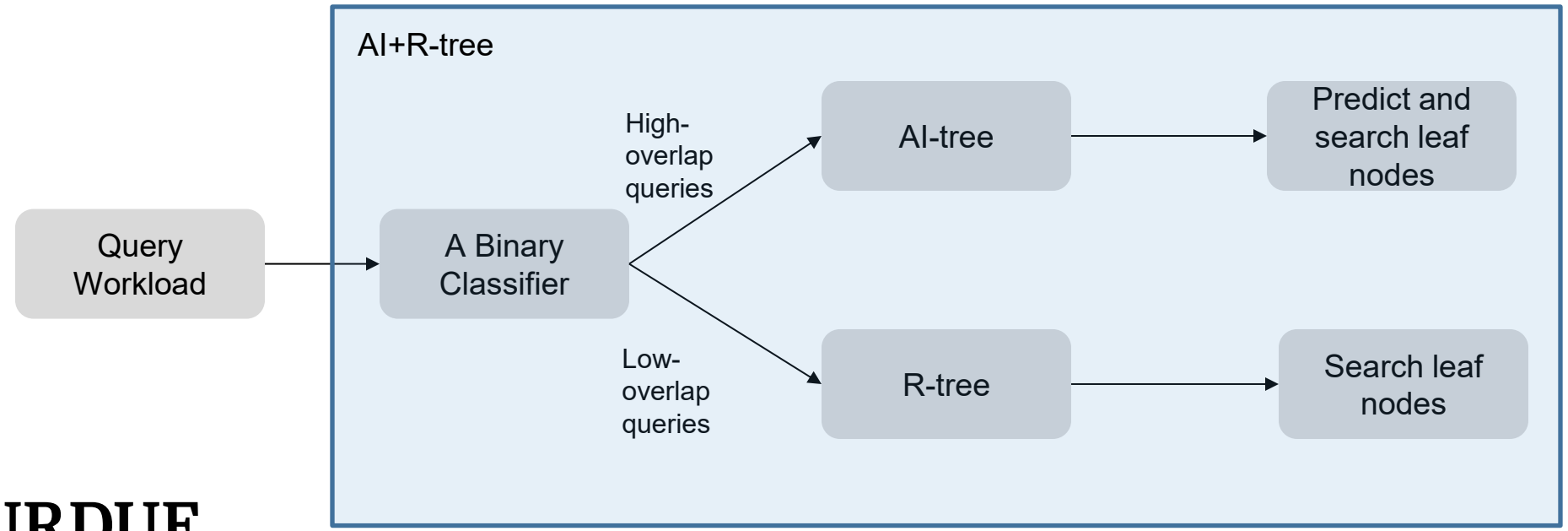


PURDUE UNIVERSITY

Al-Mamun, Abdullah, Haider, Ch Md Rakin, Jianguo Wang, and Walid G. Aref. "The "AI+ R"-tree: An Instance-optimized R-tree." In 2022 23rd IEEE International Conference on Mobile Data Management (MDM), pp. 9-18. IEEE, 2022.

The AI+R-tree [MDM'22] (Cont'd)

- Query Processing:
  - Leveraging a binary classifier to automatically differentiate between high- and low-overlap queries
  - Process the high-overlap queries using the AI-tree
  - Process the low-overlap queries using the R-tree
  - Adopting a hybrid approach, namely the AI+R-tree, to process both types of range queries efficiently



PURDUE UNIVERSITY

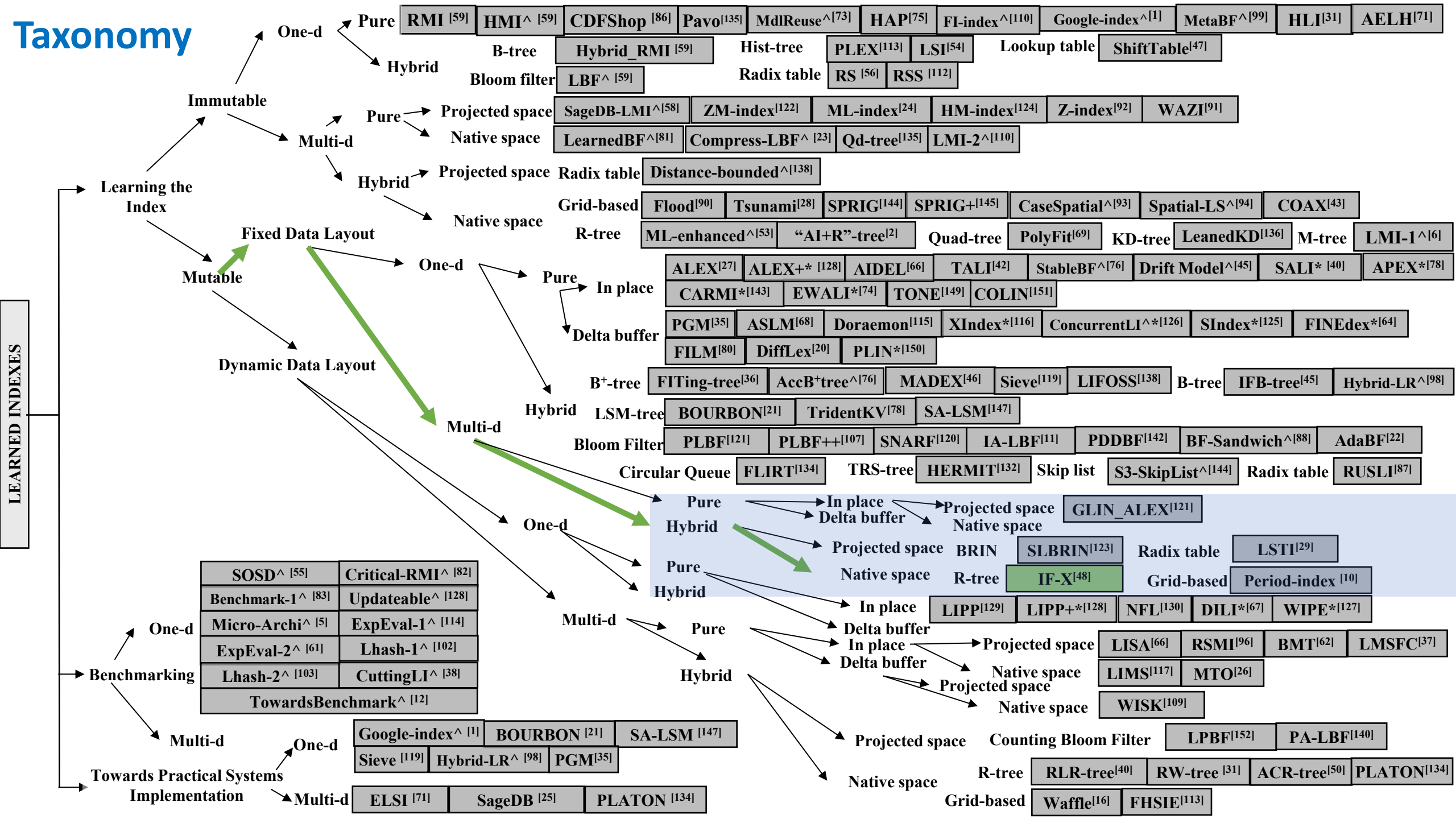
Al-Mamun, Abdullah, Haider, Ch Md Rakin, Jianguo Wang, and Walid G. Aref. "The "AI+ R"-tree: An Instance-optimized R-tree." In 2022 23rd IEEE International Conference on Mobile Data Management (MDM), pp. 9-18. IEEE, 2022.

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
  - ✓ Motivation and Challenges
    - ✓ Projected vs. Native Space
  - ✓ Approaches
  - ✓ Immutable Indexes
  - → Mutable Indexes with Fixed Data Layouts
  - Mutable Indexes with Dynamic Data Layouts
- Open Challenges for Future Research

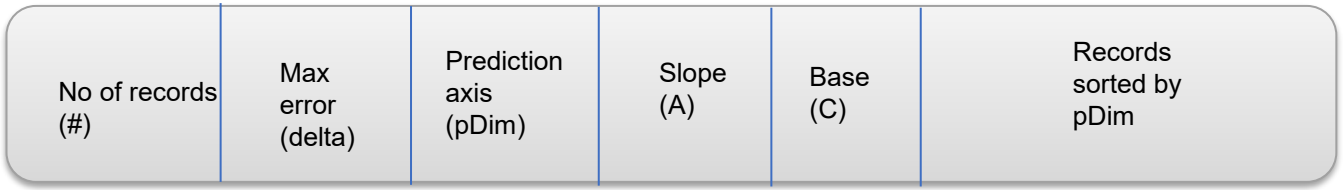
PURDUE UNIVERSITY

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
B-tree
Hybrid_RMI [59]
Bloom filter
LBF^ [59]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Multi-d
One-d
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Multi-d
Pure
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]

Interpolation Friendly (IF) Indexes: IF-X [EDBT'20]

- IF-X:
  - X can be any multi-dimensional index.
  - Why Linear Interpolation?
    - Complex models have a higher capacity to fit the CDF.
    - But complex models
      - Require more parameters
      - Slower to compute
  - Linear interpolation is:
    - Simpler
    - Computationally inexpensive
    - Can eliminate expensive training process
  - Query execution time can be reduced by up to 60%.
  - Memory footprint can be reduced by over 90%



- Leaf Node Layout:
  - IF-X indexes sort the records in each leaf node
    - Based on the best order, 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

PURDUE UNIVERSITY®

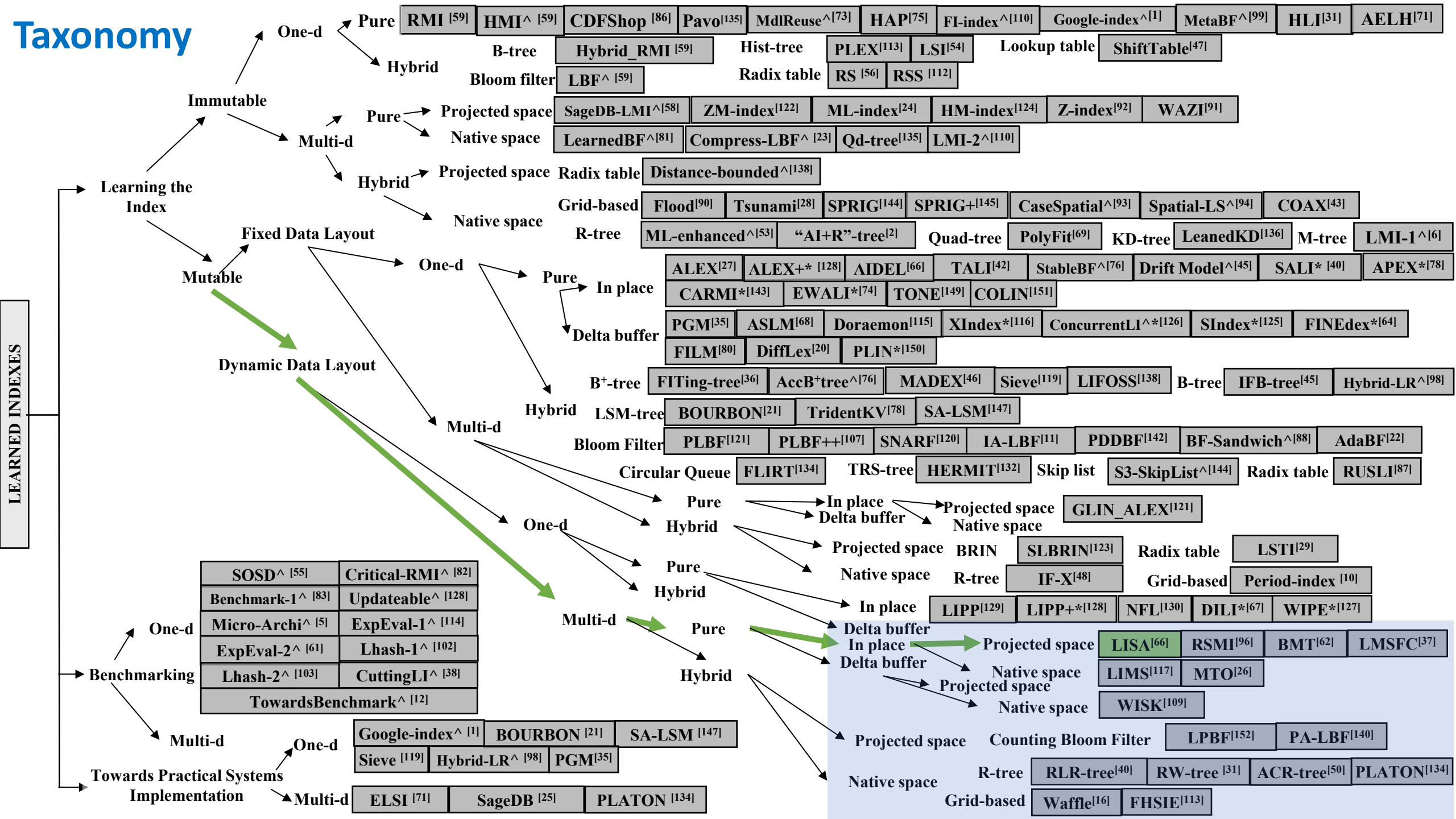
Angjela Davitkova, Evica Milchevski, and Sebastian Michel. 2020. The ML-Index:A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries.. In EDBT. 407–410.

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
  - ✓ Motivation and Challenges
    - ✓ Projected vs. Native Space
  - ✓ Approaches
  - ✓ Immutable Indexes
  - ✓ Mutable Indexes with Fixed Data Layouts
  - → Mutable Indexes with Dynamic Data Layouts
- Open Challenges for Future Research

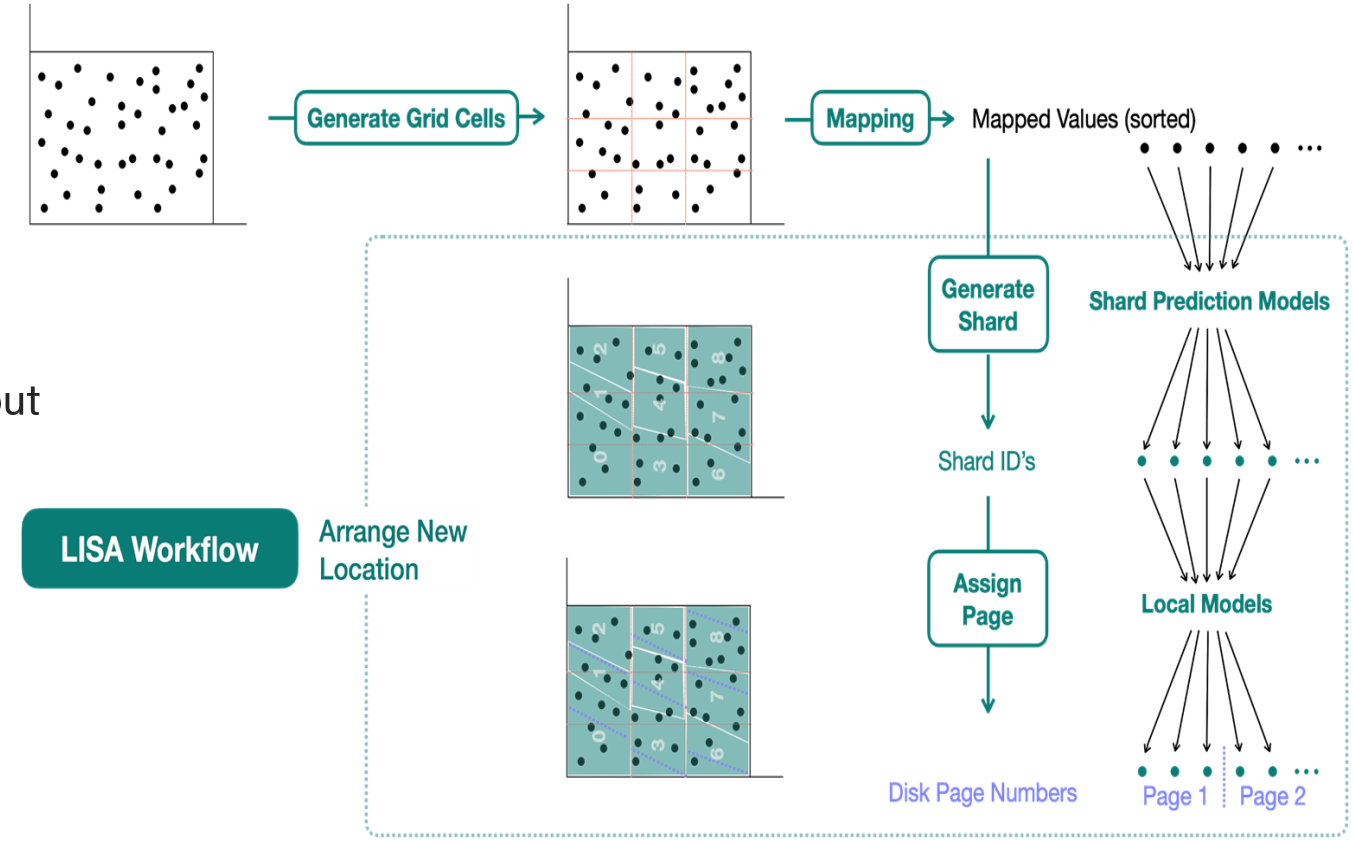
PURDUE UNIVERSITY

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
B-tree
Hybrid_RMI [59]
Bloom filter
LBF^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
One-d
Pure
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Hybrid
Multi-d
Pure
Delta buffer
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

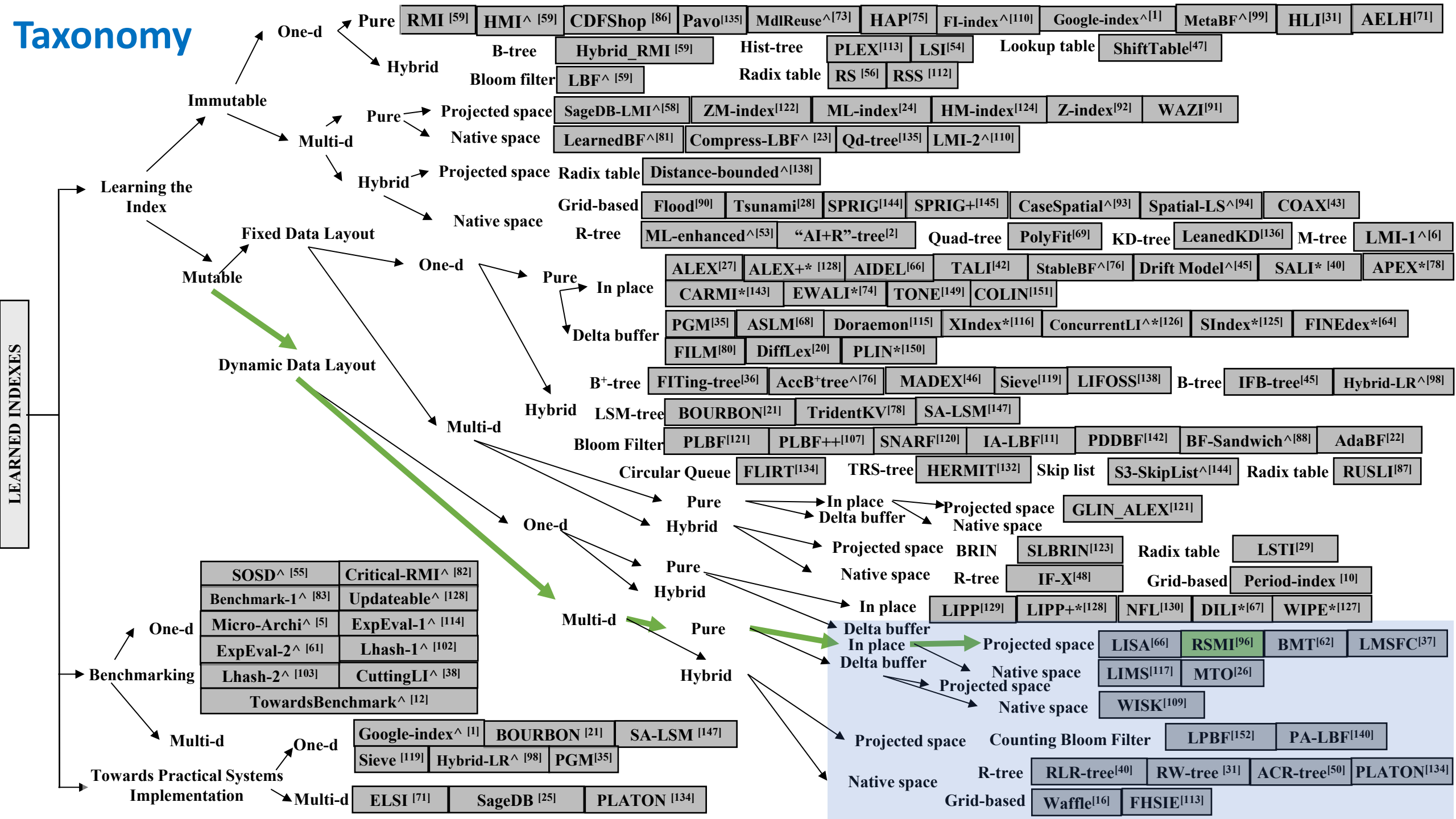
Learned Index for Spatial Data (LISA) [SIGMOD'20]

- A Mutable disk-based learned multi-dimensional index for spatial queries.
- 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
- Outperforms traditional spatial indexes for range and KNN queries:
  - Memory consumption
  - IO cost



Li, Pengfei, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. "LISA: A learned index structure for spatial data." In Proceedings of the 2020 ACM SIGMOD international conference on management of data, pp. 2119-2133. 2020.

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
Hybrid
HMI^ [59]
B-tree
Hybrid_RMI [59]
Bloom filter
LBF^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Multi-d
Pure
Delta buffer
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

Recursive Spatial Model Index (RSMI) [VLDB'20]

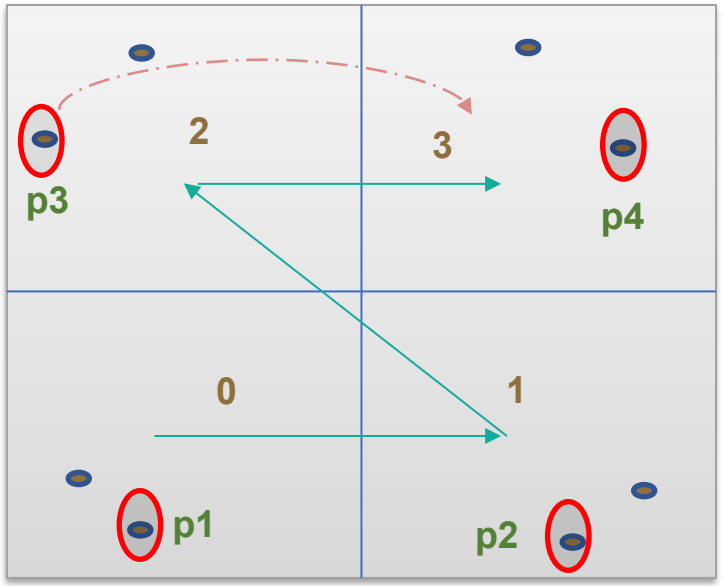
- Selecting grid resolution for Z-order for learned multi-dimensional index (e.g. ZM-Index) is challenging:
  - Large cells
    - More false positives due to many points per cell
  - Small cells
    - Hard to learn due to uneven gaps in Cumulative Distribution Function (CDF)
- Spatial index based on ordering the data points by a rank space-based transformation
  - Simplify the indexing functions to be learned
  - M(search keys)→disk block Ids (location)

PURDUE UNIVERSITY

Qi, Jianzhong, Guanli Liu, Christian S. Jensen, and Lars Kulik. "Effectively learning spatial indices." Proceedings of the VLDB Endowment 13, no. 12 (2020): 2341-2354.

Recursive Spatial Model Index (RSMI) [VLDB'20] (Cont'd)

- Recursively partition 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

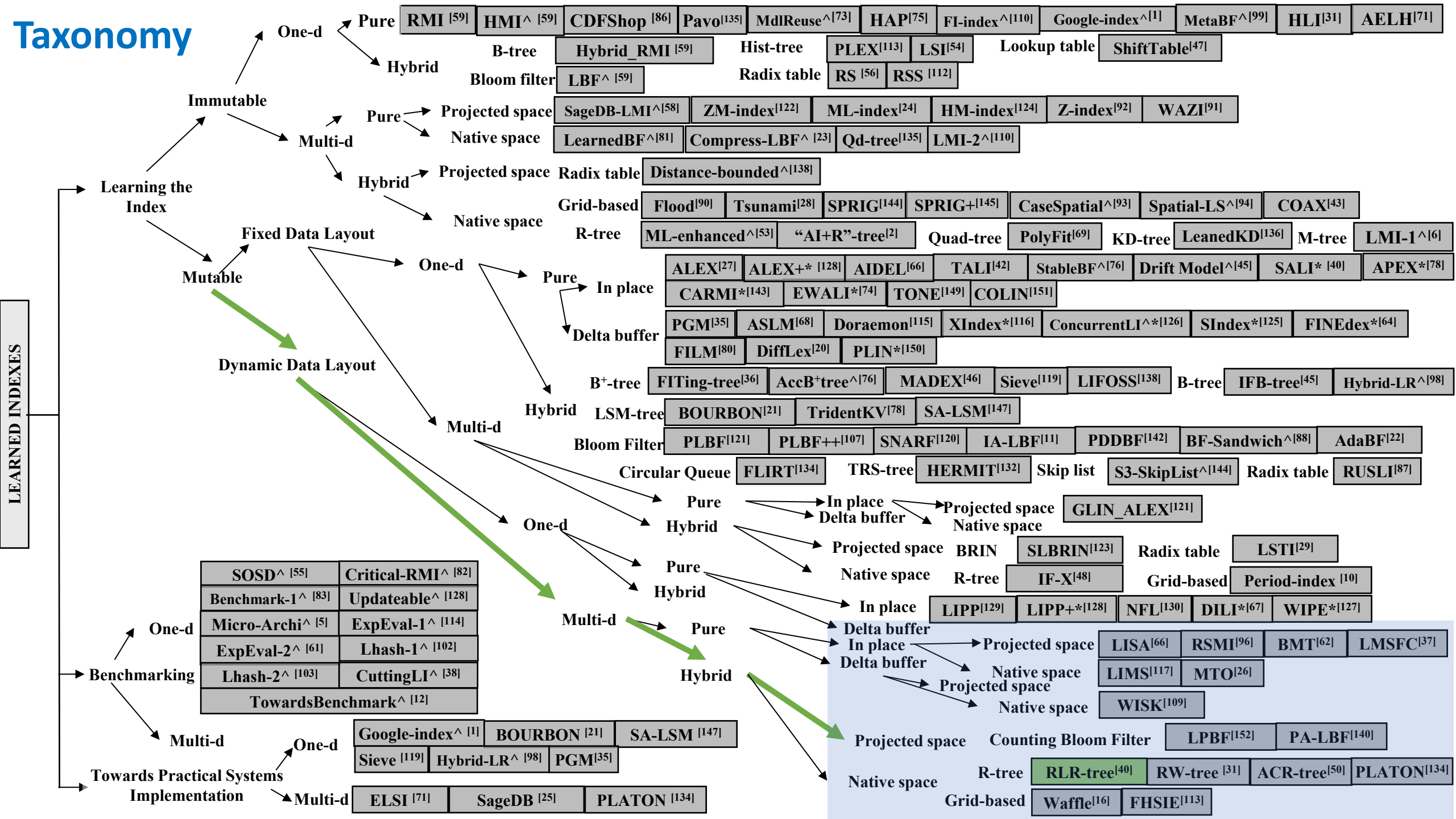


Point	p1	p2	p3	p4
Initial partition Id	0	1	2	3
Model predicted Id	0	1	3	3
Learned partition Id	0	1	3	3

PURDUE UNIVERSITY

Qi, Jianzhong, Guanli Liu, Christian S. Jensen, and Lars Kulik. "Effectively learning spatial indices." Proceedings of the VLDB Endowment 13, no. 12 (2020): 2341-2354.

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hybrid
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
One-d
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Pure
Hybrid
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Multi-d
Pure
Delta buffer
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Hybrid
Projected space
Native space
WISK[109]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]

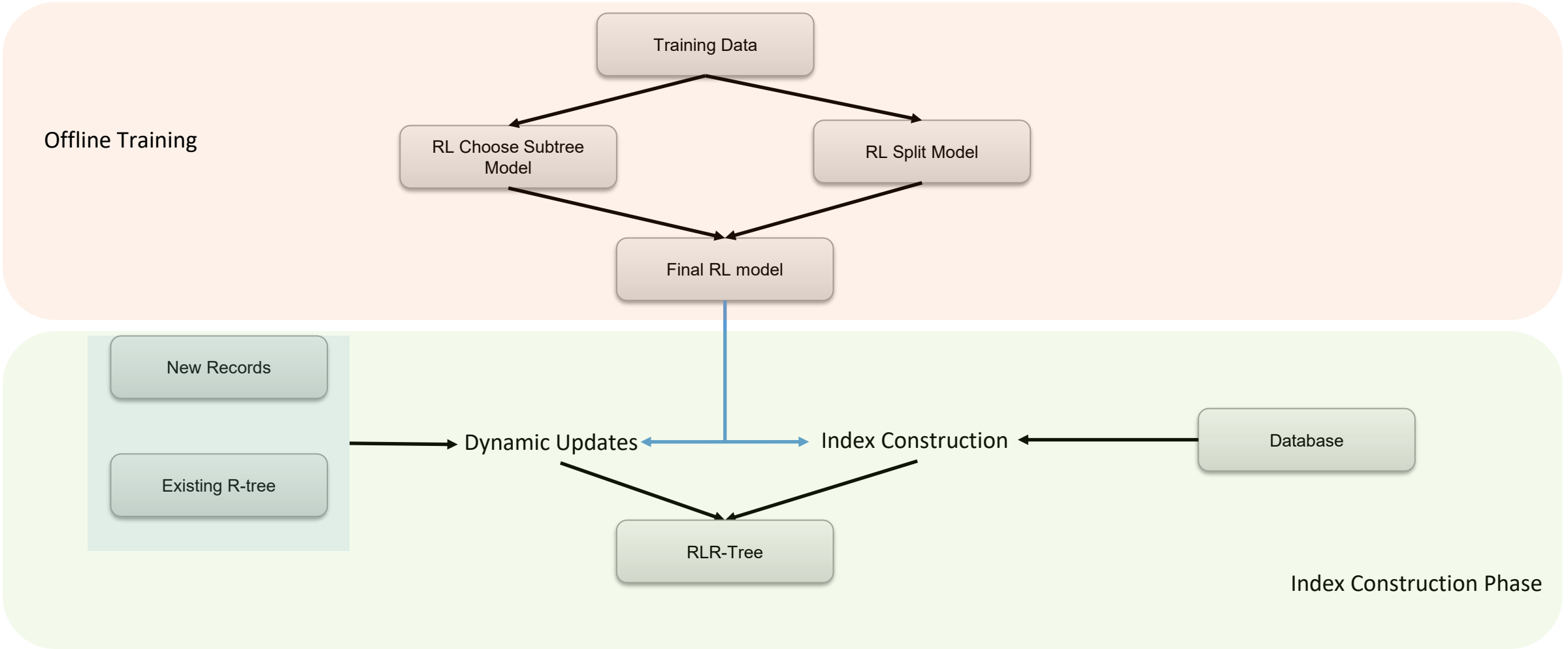
A Reinforcement Learning Based R-Tree (RLR-tree) [SIGMOD'23]

- Formulate the ChooseSubtree and Split operations of a traditional R-tree as a Markov Decision Process (MDP)
- Do not modify existing query processing algorithms
- State Space:
  - A tree node with the following features: area increase, perimeter increase, overlap increase, and occupancy rate.
- Action Space:
  - Given a current state, the RL agent picks a child node into which the new entry will be inserted.
- Transition Function:
  - The agent moves to a child node.
  - If the child node is a leaf node, the agent reaches a terminal state.
- Reward Function:
  - Use a reference tree with a pre-defined ChooseSubtree and Split strategy
  - Difference between the query processing cost of the RLR-tree and the reference R-tree
- The RLR-tree adopts a Deep Q-Network learning method for training the agent.

PURDUE UNIVERSITY®

Gu, Tu, Kaiyu Feng, Gao Cong, Cheng Long, Zheng Wang, and Sheng Wang. "The rlr-tree: A reinforcement learning based r-tree for spatial data." Proceedings of the ACM on Management of Data 1, no. 1 (2023): 1-26.

A Reinforcement Learning Based R-Tree (RLR-tree) [SIGMOD'23] (Cont'd)



Gu, Tu, Kaiyu Feng, Gao Cong, Cheng Long, Zheng Wang, and Sheng Wang. "The rlr-tree: A reinforcement learning based r-tree for spatial data." Proceedings of the ACM on Management of Data 1, no. 1 (2023): 1-26.

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
  - ✓ Motivation and Challenges
    - ✓ Projected vs. Native Space
  - ✓ Approaches
  - ✓ Immutable Indexes
  - ✓ Mutable Indexes with Fixed Data Layouts
  - ✓ Mutable Indexes with Dynamic Data Layouts
  - → Summary
- Open Challenges for Future Research

PURDUE UNIVERSITY

Learned Multi-dimensional Indexes: Summary

Properties of the Immutable Learned Multi-dimensional Indexes

Index	Pure Learned	Hybrid Learned	Data Space	Point Query	Range Query	kNN Query	Join Query
SageDB-LMI [101]	✓	×	Projected	Exact	Exact	×	×
ZM-index [197]	✓	×	Projected	Exact	Exact	×	×
ML-index [45]	✓	×	Projected	Exact	Exact	Exact	×
HM-index [199]	✓	×	Projected	Exact	Exact	×	×
Z-index [153]	✓	×	Projected	Exact	Exact	×	×
WAZI [152]	✓	×	Projected	Exact	Exact	×	×
LearnedBF [136]	✓	×	Native	–	–	–	–
CompressLBF [44]	✓	×	Native	–	–	–	–
Qd-tree [214]	✓	×	Native	Exact	Exact	×	×
LMI-2 [183]	✓	×	Native	Approx.	Approx.	Approx.	×
Distance-bounded [219]	×	Radix table	Projected	Approx.	Approx.	×	Approx.
Flood [148]	×	Grid	Native	Exact	Exact	×	×
Tsunami [49]	×	Grid	Native	Exact	Exact	×	×
SPRIG [225]	×	Grid	Native	Exact	Exact	Exact	×
SPRIG+ [226]	×	Grid	Native	Exact	Exact	Exact	×
ML-enhanced [93]	×	R-tree	Native	×	×	Approx.	×
"AI + R"-tree [8]	×	R-tree	Native	Exact	Exact	×	×
CaseSpatial [154]	×	Grid	Native	Exact	Exact	×	×
Spatial-LS [155]	×	Grid	Native	Exact	Exact	×	Exact
COAX [75]	×	Grid	Native	Exact	Exact	×	×
PolyFit [120]	×	Quad-tree	Native	Approx.	Approx.	×	×
LearnedKD [216]	×	KD-tree	Native	×	×	Approx.	×
LMI-1 [13]	×	M-tree	Native	Approx.	Approx.	Approx.	×

PURDUE UNIVERSITY

The citation numbers in the figure correspond to references from the survey paper: Al-Mamun, Abdullah, Hao Wu, Qiyang He, Jianguo Wang, and Walid G. Aref. "A survey of learned indexes for the multi-dimensional space." arXiv preprint arXiv:2403.06456 (2024).

Learned Multi-dimensional Indexes: Summary

Properties of the Mutable Learned Multi-dimensional Indexes with Fixed Data Layout

Index	Data Layout	Pure Learned	Hybrid Learned	Data Space	Point Query	Range Query	kNN Query	Join Query
GLIN-ALEX [196]	Fixed	In-place	×	Projected	Exact	Exact	×	×
SLBRIN [198]	Fixed	×	BRIN	Projected	Exact	Exact	Exact	×
LSTI [50]	Fixed	×	Radix Table	Projected	Exact	Exact	Exact	×
IF-X [80]	Fixed	×	R-tree	Native	Exact	Exact	×	×
Period Index [20]	Fixed	×	Grid	Native	Exact	Exact	×	×

PURDUE UNIVERSITY®

The citation numbers in the figure correspond to references from the survey paper: Al-Mamun, Abdullah, Hao Wu, Qiyang He, Jianguo Wang, and Walid G. Aref. "A survey of learned indexes for the multi-dimensional space." arXiv preprint arXiv:2403.06456 (2024).

Learned Multi-dimensional Indexes: Summary

Properties of the Mutable Learned Multi-dimensional Indexes with Dynamic Data Layout

Index	Data Layout	Pure Learned	Hybrid Learned	Data Space	Point Query	Range Query	kNN Query	Join Query
LISA [116]	Dynamic	In-place	×	Projected	Exact	Exact	Exact	×
RSMI [160]	Dynamic	In-place	×	Projected	Exact	Approx.	Approx.	×
BMT [112]	Dynamic	In-place	×	Projected	Exact	Exact	Exact	×
LMSFC [66]	Dynamic	In-place	×	Projected	Exact	Exact	×	×
LIMS [191]	Dynamic	In-place	×	Native	Exact	Exact	Exact	×
MTO [47]	Dynamic	In-place	×	Native	Exact	Exact	×	Exact
WISK [181]	Dynamic	Delta Buffer	×	Native	Exact	Exact	Exact	×
LPBF [234]	Dynamic	×	CBF	Projected	—	—	—	—
PA-LBF [221]	Dynamic	×	CBF	Projected	—	—	—	—
RLR-tree [72]	Dynamic	×	R-tree	Native	Exact	Exact	Exact	Exact
RW-tree [52]	Dynamic	×	R-tree	Native	Exact	Exact	Exact	×
ACR-tree [83]	Dynamic	×	R-tree	Native	Exact	Exact	Exact	×
PLATON [213]	Dynamic	×	R-tree	Native	Exact	Exact	Exact	×
Waffle [38]	Dynamic	×	Grid	Native	Exact	Exact	Exact	×
FHSIE [187]	Dynamic	×	Grid	Native	Exact	Exact	Exact	×

PURDUE UNIVERSITY

The citation numbers in the figure correspond to references from the survey paper: Al-Mamun, Abdullah, Hao Wu, Qiyang He, Jianguo Wang, and Walid G. Aref. "A survey of learned indexes for the multi-dimensional space." arXiv preprint arXiv:2403.06456 (2024).

Learned Multi-dimensional Indexes: Summary

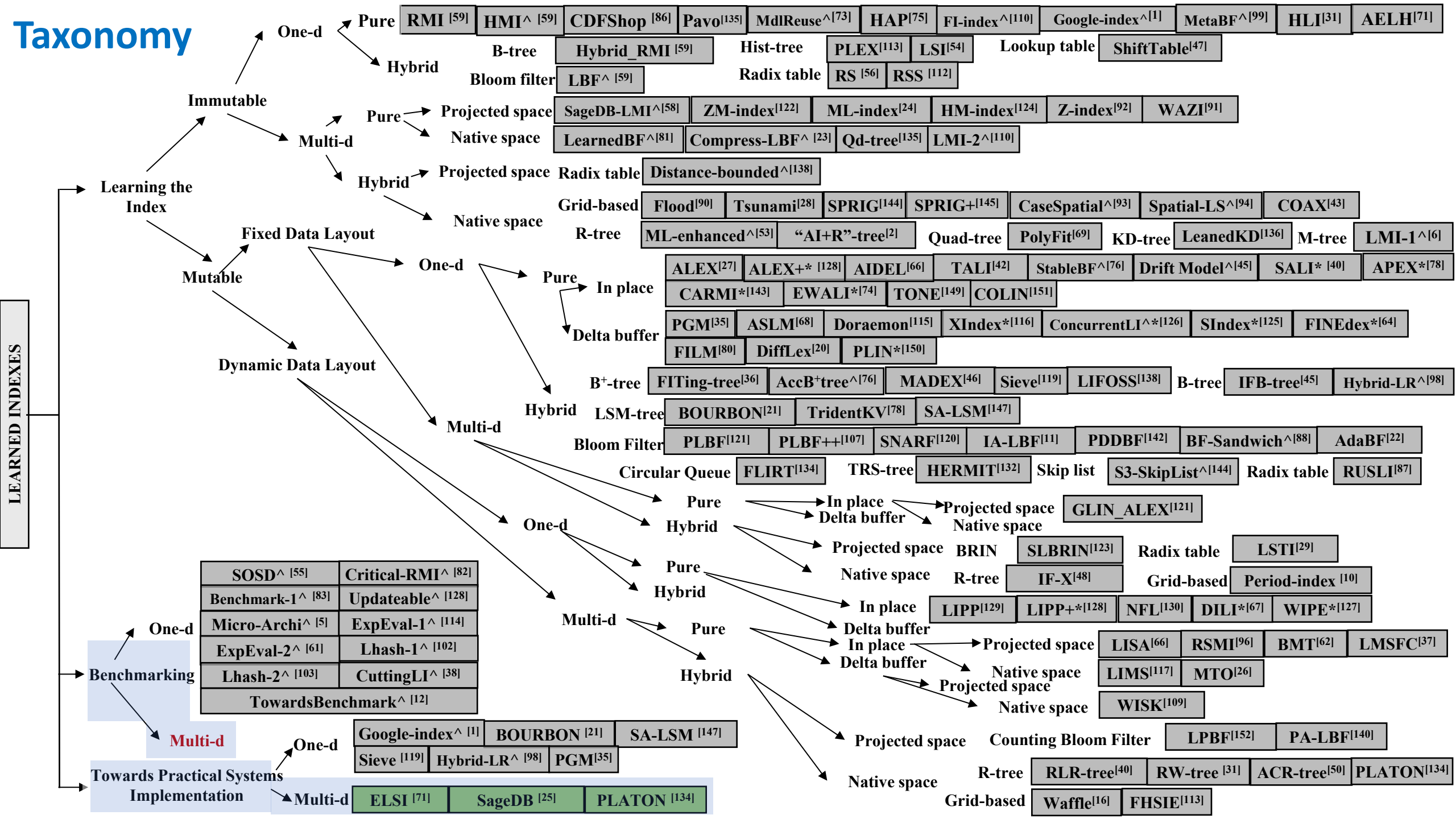
ML Techniques for Learned Multi-dimensional Indexes

ML Techniques	Index
RadixSpline, Random Forest Regression	LSTI [50]
Lattice Regression Model	LISA [116]
Ploynomial Regression	LIMS [191]
Linear Model	SageDB-LMI [101], Tsunami [49], COAX [75], GLIN_ALEX [196]
Density Model	Z-index [153]
Piecewise Linear Model, RMI	FLOOD [148]
Polynomial Function	PolyFit [120]
Linear Interpolation	IF-X [80]
Spatial Interpolation Function	SPRIG [225], SPRIG+ [226]
Cumulative Histogram	Period-index* [20]
K-means Clustering	LMI-2 [183], FHSIE [187]
Density based Clustering	RW-tree [52]
Random Forest Density Estimation Model	WAZI [152]
Random Forest, Decision Tree	LMSFC [66], "AI+R"-tree [8]
RadixSpline	CaseSpatial [154], Spatial-LS [155], Distance-bounded [219]
Gradient Boosting	LPBF [234], PA-LBF [221]
Neural Network, Linear Model	ZM-index [197]
Neural Network	LearnedKD [216], LMI-1 [13], ML-enhanced [93], CompressedBF [44], LearnedBF [136], ML-index [45], HM-index [199], RSMI [160], SLBRIN [198]
Reinforcement Learning	QD-tree [214], MTO [47], RLR-tree [72], Waffle [38], BMT [112], WISK [181], ACR-tree [83], PLATON [213]

PURDUE UNIVERSITY®

The citation numbers in the figure correspond to references from the survey paper: Al-Mamun, Abdullah, Hao Wu, Qiyang He, Jianguo Wang, and Walid G. Aref. "A survey of learned indexes for the multi-dimensional space." arXiv preprint arXiv:2403.06456 (2024).

Taxonomy


LEARNED INDEXES
Learning the Index
Immutable
One-d
Pure
RMI [59]
HMI^ [59]
CDFShop [86]
Pavo[135]
MdlReuse^[73]
HAP[75]
FI-index^[110]
Google-index^[1]
MetaBF^[99]
HLI[31]
AELH[71]
Hybrid
B-tree
Hybrid_RMI [59]
Hist-tree
PLEX[113]
LSI[54]
Lookup table
ShiftTable[47]
Bloom filter
LBF^ [59]
Radix table
RS [56]
RSS [112]
Multi-d
Pure
Projected space
SageDB-LMI^[58]
ZM-index[122]
ML-index[24]
HM-index[124]
Z-index[92]
WAZI[91]
Native space
LearnedBF^[81]
Compress-LBF^ [23]
Qd-tree[135]
LMI-2^[110]
Hybrid
Projected space
Radix table
Distance-bounded^[138]
Native space
Grid-based
Flood[90]
Tsunami[28]
SPRIG[144]
SPRIG+[145]
CaseSpatial^[93]
Spatial-LS^[94]
COAX[43]
R-tree
ML-enhanced^[53]
"AI+R"-tree[2]
Quad-tree
PolyFit[69]
KD-tree
LeanedKD[136]
M-tree
LMI-1^[6]
Mutable
Fixed Data Layout
One-d
Pure
In place
ALEX[27]
ALEX+* [128]
AIDEL[66]
TALI[42]
StableBF^[76]
Drift Model^[45]
SALI* [40]
APEX*[78]
CARMI*[143]
EWALI*[74]
TONE[149]
COLIN[151]
Delta buffer
PGM[35]
ASLM[68]
Doraemon[115]
XIndex*[116]
ConcurrentLI^*[126]
SIndex*[125]
FINEdex*[64]
FILM[80]
DiffLex[20]
PLIN*[150]
Hybrid
B+-tree
FITing-tree[36]
AccB+tree^[76]
MADEX[46]
Sieve[119]
LIFOSS[138]
B-tree
IFB-tree[45]
Hybrid-LR^[98]
LSM-tree
BOURBON[21]
TridentKV[78]
SA-LSM[147]
Multi-d
Bloom Filter
PLBF[121]
PLBF++[107]
SNARF[120]
IA-LBF[11]
PDDBF[142]
BF-Sandwich^[88]
AdaBF[22]
Circular Queue
FLIRT[134]
TRS-tree
HERMIT[132]
Skip list
S3-SkipList^[144]
Radix table
RUSLI[87]
Dynamic Data Layout
One-d
Pure
In place
Projected space
GLIN_ALEX[121]
Delta buffer
Native space
Hybrid
Projected space
BRIN
SLBRIN[123]
Radix table
LSTI[29]
Native space
R-tree
IF-X[48]
Grid-based
Period-index [10]
Multi-d
Pure
In place
LIPP[129]
LIPP+*[128]
NFL[130]
DILI*[67]
WIPE*[127]
Delta buffer
Hybrid
In place
Projected space
LISA[66]
RSMI[96]
BMT[62]
LMSFC[37]
Delta buffer
Native space
LIMS[117]
MTO[26]
Projected space
Native space
WISK[109]
Projected space
Counting Bloom Filter
LPBF[152]
PA-LBF[140]
Native space
R-tree
RLR-tree[40]
RW-tree [31]
ACR-tree[50]
PLATON[134]
Grid-based
Waffle[16]
FHSIE[113]
Benchmarking
One-d
SOSD^ [55]
Critical-RMI^ [82]
Benchmark-1^ [83]
Updateable^ [128]
Micro-Archi^ [5]
ExpEval-1^ [114]
ExpEval-2^ [61]
Lhash-1^ [102]
Lhash-2^ [103]
CuttingLI^ [38]
TowardsBenchmark^ [12]
Multi-d
Towards Practical Systems Implementation
One-d
Google-index^ [1]
BOURBON [21]
SA-LSM [147]
Sieve [119]
Hybrid-LR^ [98]
PGM[35]
Multi-d
ELSI [71]
SageDB [25]
PLATON [134]

Learned Multi-dimensional Indexes: Summary

- Benchmarking
  - Less studied compared to its one-dimensional counterpart
  - A recent experimental survey provides an initial benchmark.
    - Liu, Qiyu, Maocheng Li, Yuxiang Zeng, Yanyan Shen, and Lei Chen. "How good are multi-dimensional learned indexes? An experimental survey." The VLDB Journal 34, no. 2 (2025): 1-29.
- Towards Practical Systems Integration
  - Less explored compared to its one-dimensional counterpart

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
- ⟶ Open Challenges for Future Research

PURDUE UNIVERSITY

Future Research Directions

- The Lack of Total Ordering for the Multi-dimensional Case
  - Challenging to provide an error bound in case of ML model mispredictions
- Choice of ML models
  - Need to reduce the impact of ML model training time, storage overhead, and prediction latency
- ML Model Re-training
  - Significant changes in the underlying input data/query distribution should be detected as soon as possible.
  - Full vs. Partial model re-training

PURDUE UNIVERSITY

Future Research Directions

- Supporting Dynamic Inserts/Updates
  - Further research is needed to investigate the advantages and disadvantages of each approach.
- Supporting Concurrency
  - Need to consider the issue of concurrency as a first-class citizen while designing learned indexes
- Index Compression
  - Need to further explore the potential benefits of index compression using learned indexes

PURDUE UNIVERSITY

Future Research Directions

- Security
  - The issue of security in the context of learned indexes is little explored.
- Benchmarking for the Multi-dimensional Case
  - Extensive benchmarking studies is needed
- Theoretical Analysis for the Multi-dimensional Case
  - Analysis of the various components of learned multi-dimensional indexes is needed
- Leveraging Modern Hardware for Learned Indexes
  - Natively implementing learned indexes over modern hardware platforms

PURDUE UNIVERSITY

Outline

- ✓ Introduction and Historical Background
- ✓ Learned Indexes for the One-dimensional Space
- ✓ Learned Indexes for the Multi-dimensional Space
- ✓ Open Challenges for Future Research

PURDUE UNIVERSITY®

THANK YOU

PURDUE UNIVERSITY®