The Case for Learned Index Structures

DS 5110/CS 5501: Big Data Systems

Spring 2024

Lecture 9

Rui Yang

Some material taken/derived from:

- Stanford EE380 Seminar talk by Alex Beutel

Learning Objectives

1. Understand the general purpose of index structures

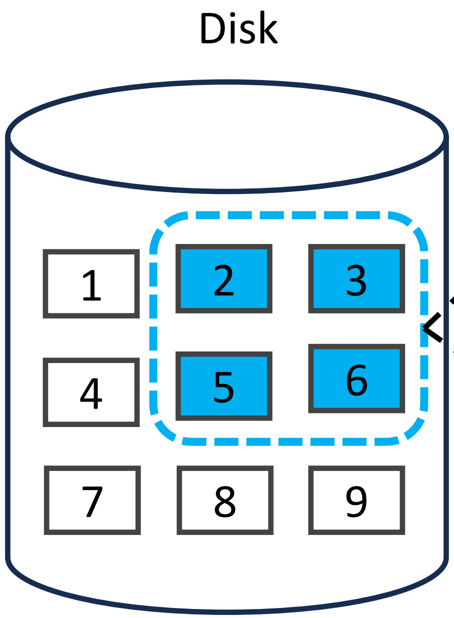
2. Understand the basic concepts of **learned index structures (LIS)**

3. Case study: Recursive Model Index (RMI)

Outline

- Background
- Learned Index Structures: Concept
- Learned Index Structures: Approaches
- Learned Index Structures: Discussion
- Learned Index Structures: Roadmap
- Demo

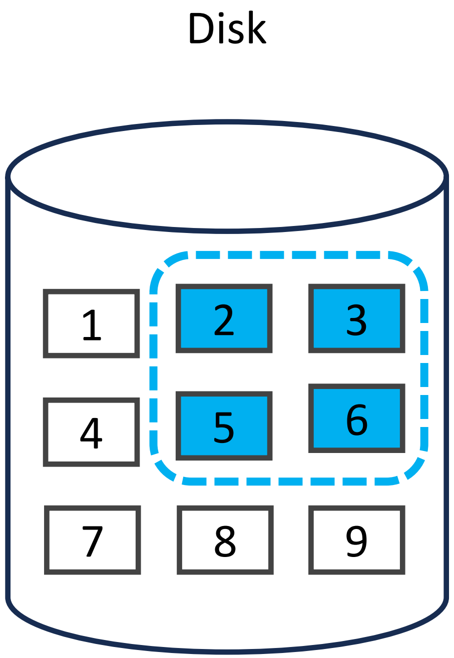
Background: A Table Layout in Disk



SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

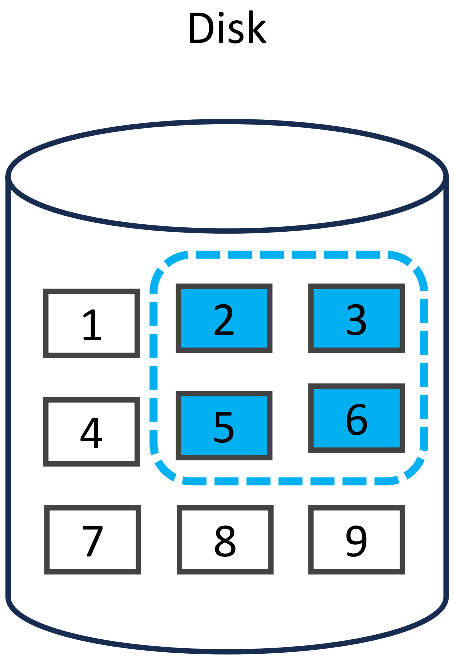
$\left.\right\}$ 100 students

Background: Data Retrieval (No Index)

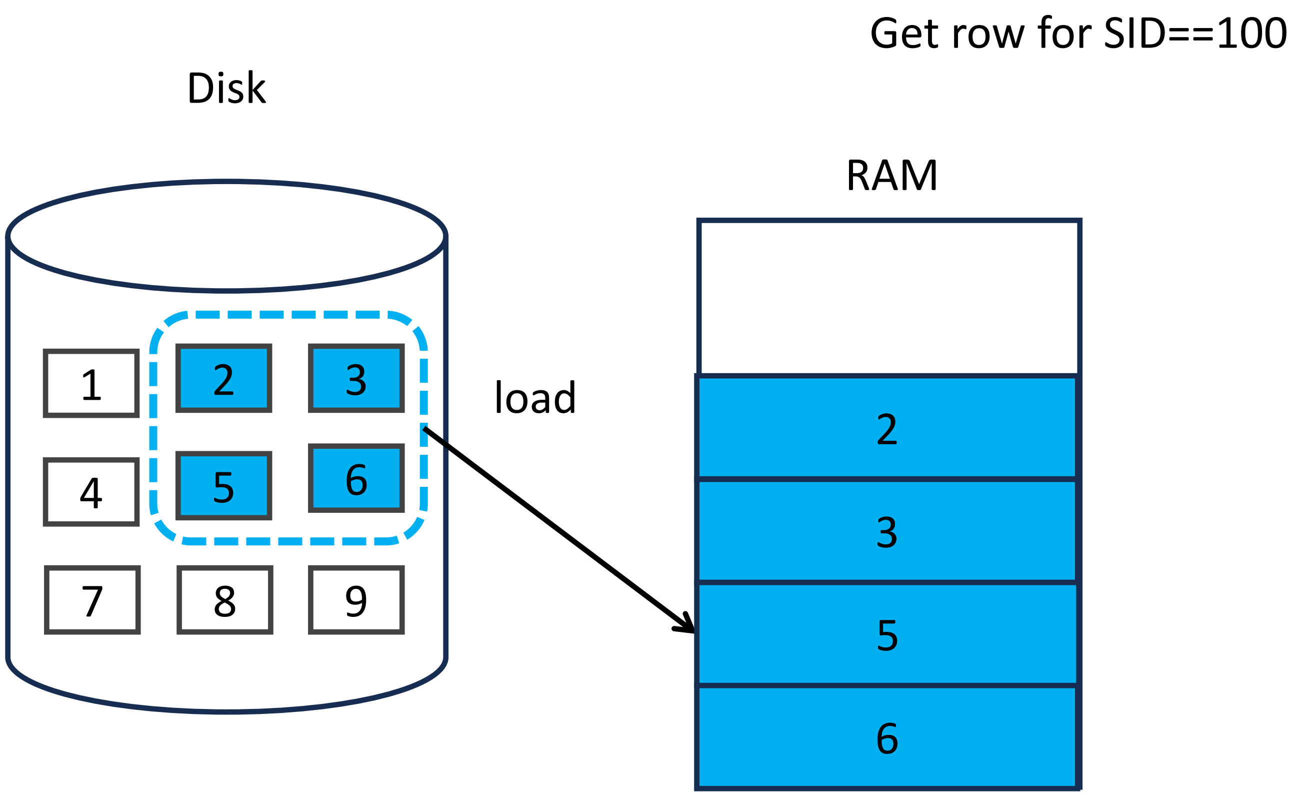


Background: Data Retrieval (No Index)

Get row for SID==100

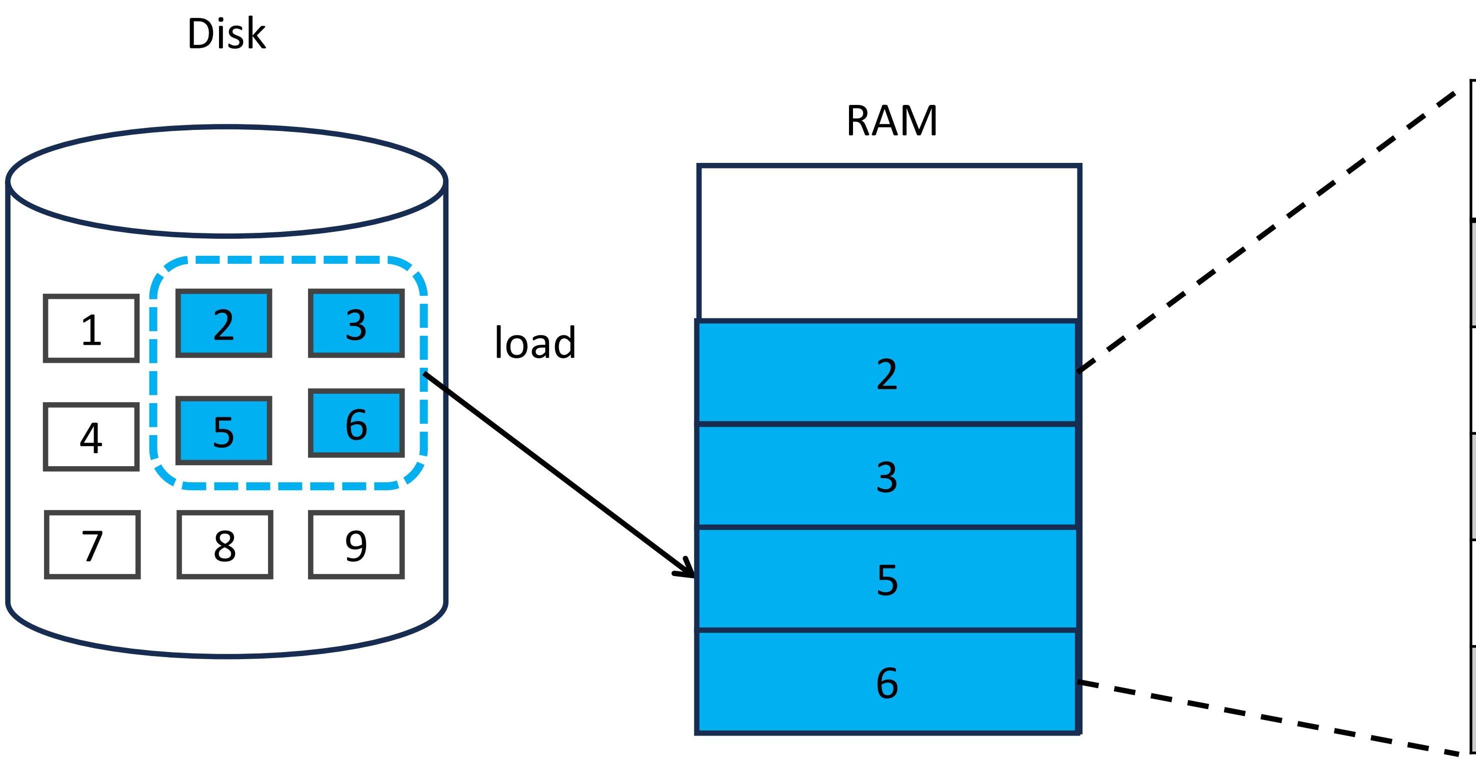


Background: Data Retrieval (No Index)



Background: Data Retrieval (No Index)

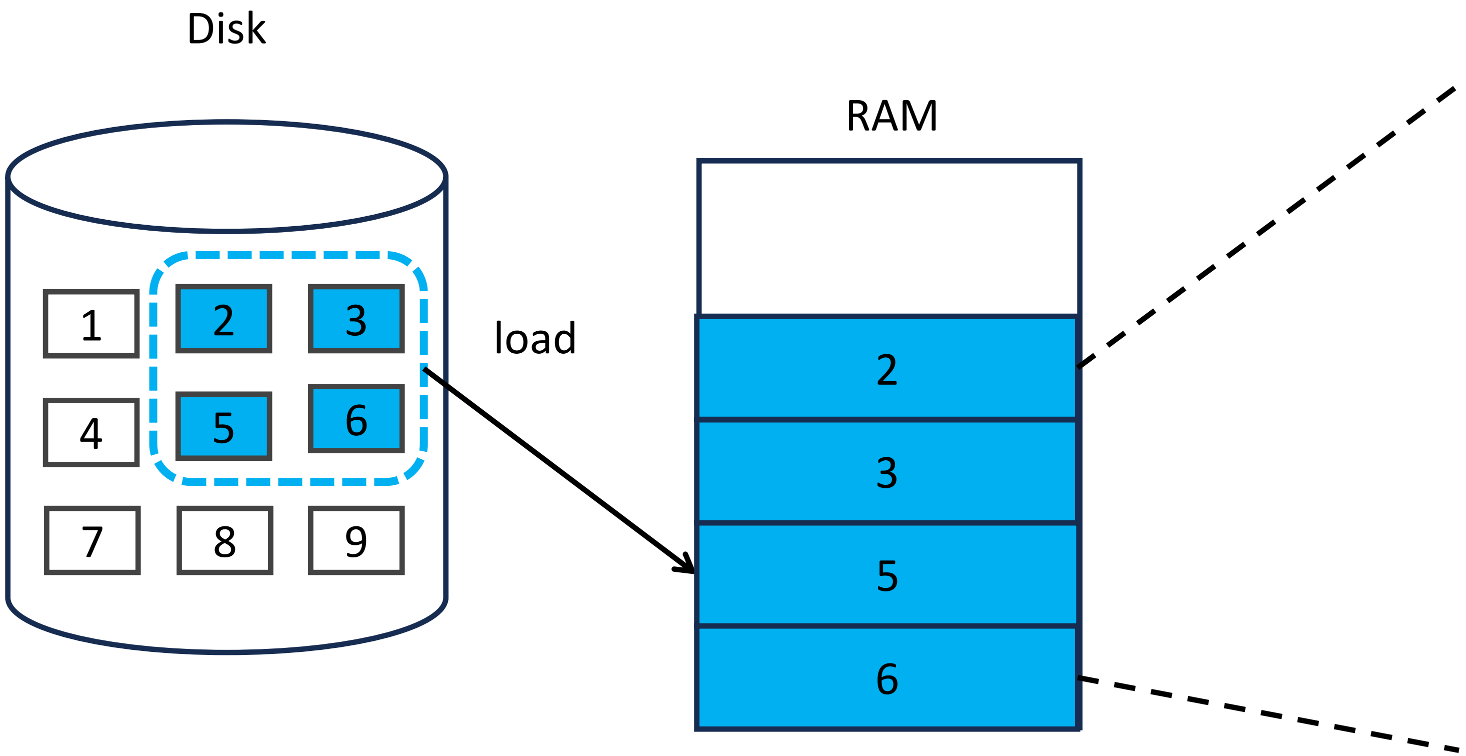
Get row for SID==100



SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

Background: Data Retrieval (No Index)

Get row for SID==100

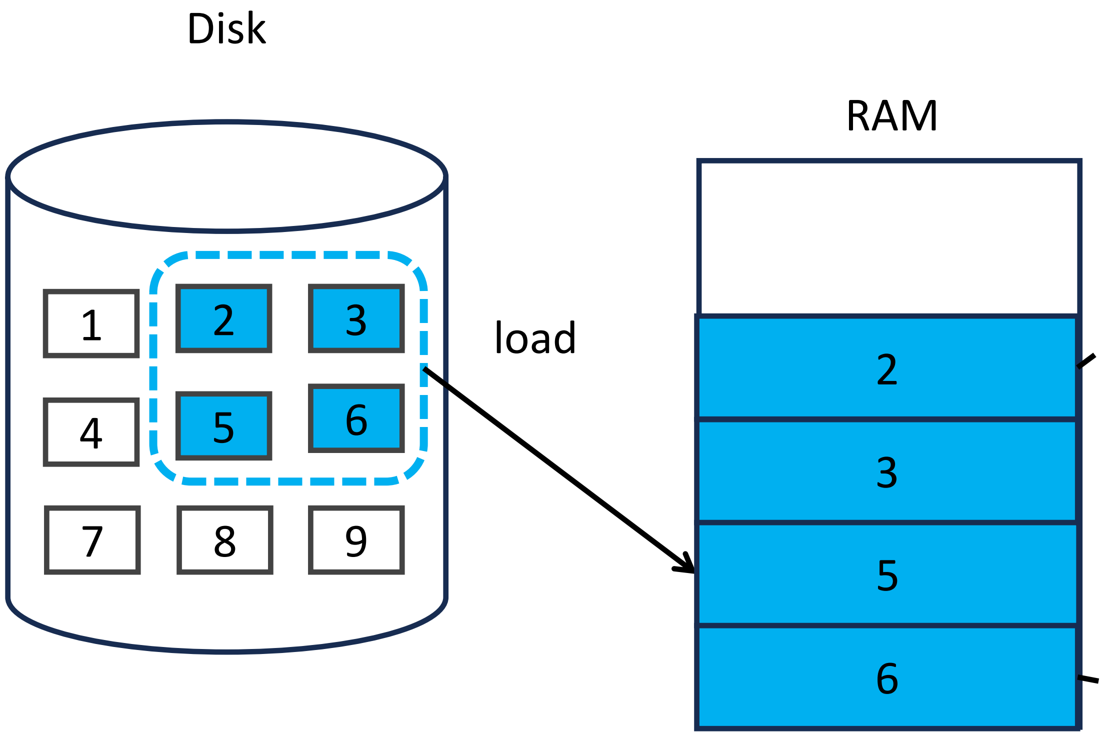


SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

Iterate each row to get SID==100

Background: Data Retrieval (No Index)

Get row for SID==100



SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

We need to load 4 pages into RAM. Too slow!
Can we do better? (Any thoughts)

Iterate each row to get SID==100

Background: Data Retrieval (B-Tree)

SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

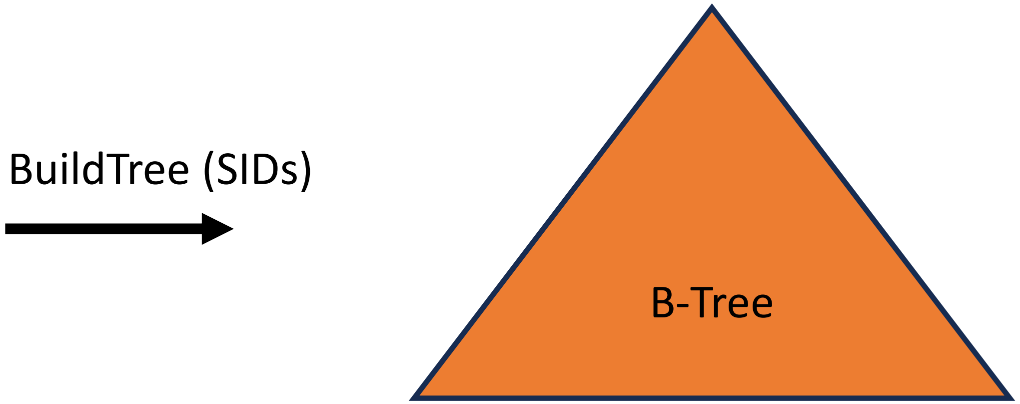
Background: Data Retrieval (B-Tree)

SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

BuildTree (SIDs) →

Background: Data Retrieval (B-Tree)

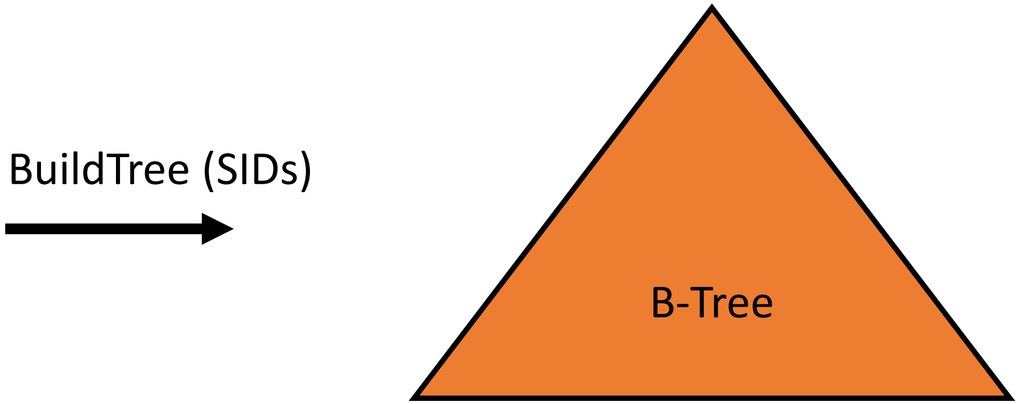
SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66



Background: Data Retrieval (B-Tree)

SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

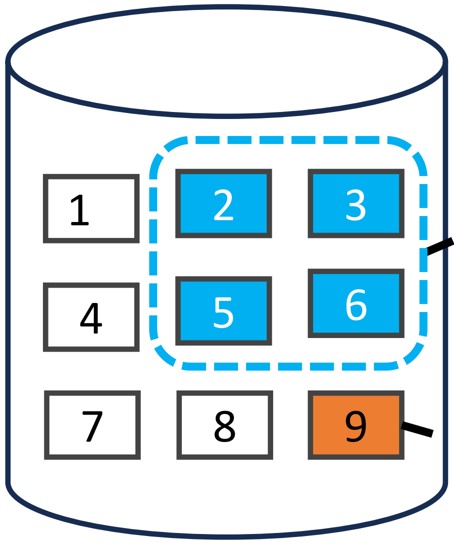
BuildTree (SIDs)



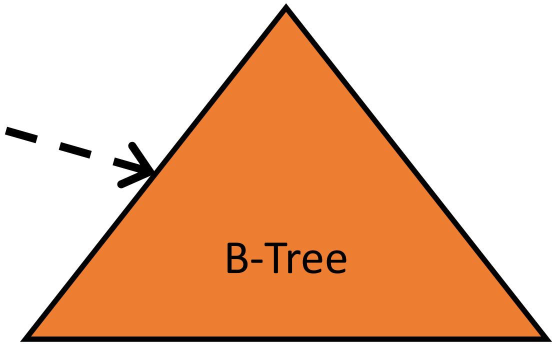
Sort the SIDs and record the row location of each SID
1 page

Background: A Table Layout in Disk (B-Tree)

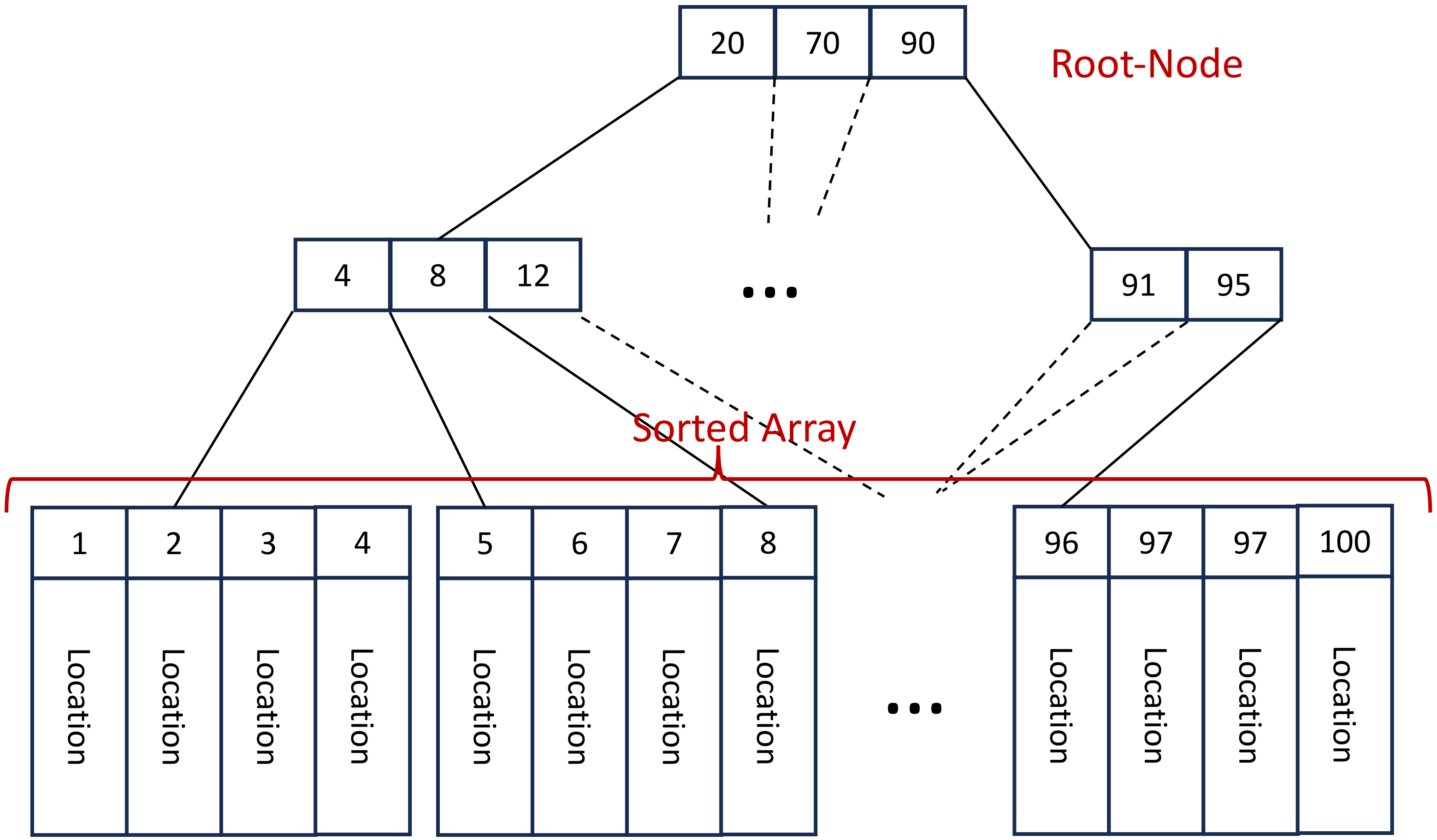
Disk



SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66



Background: B-Tree

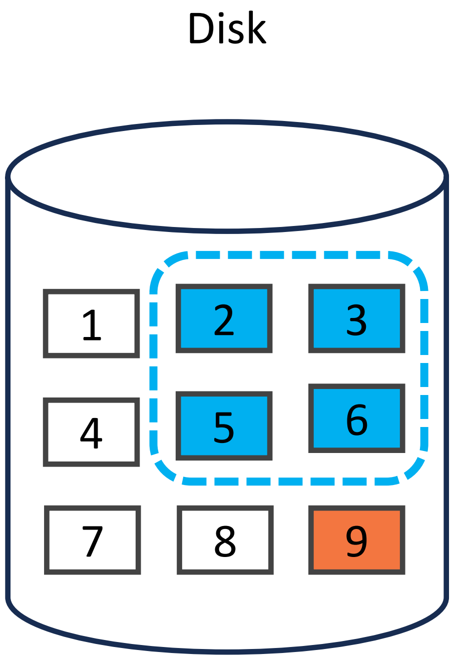


Background: Data Retrieval (B-Tree)

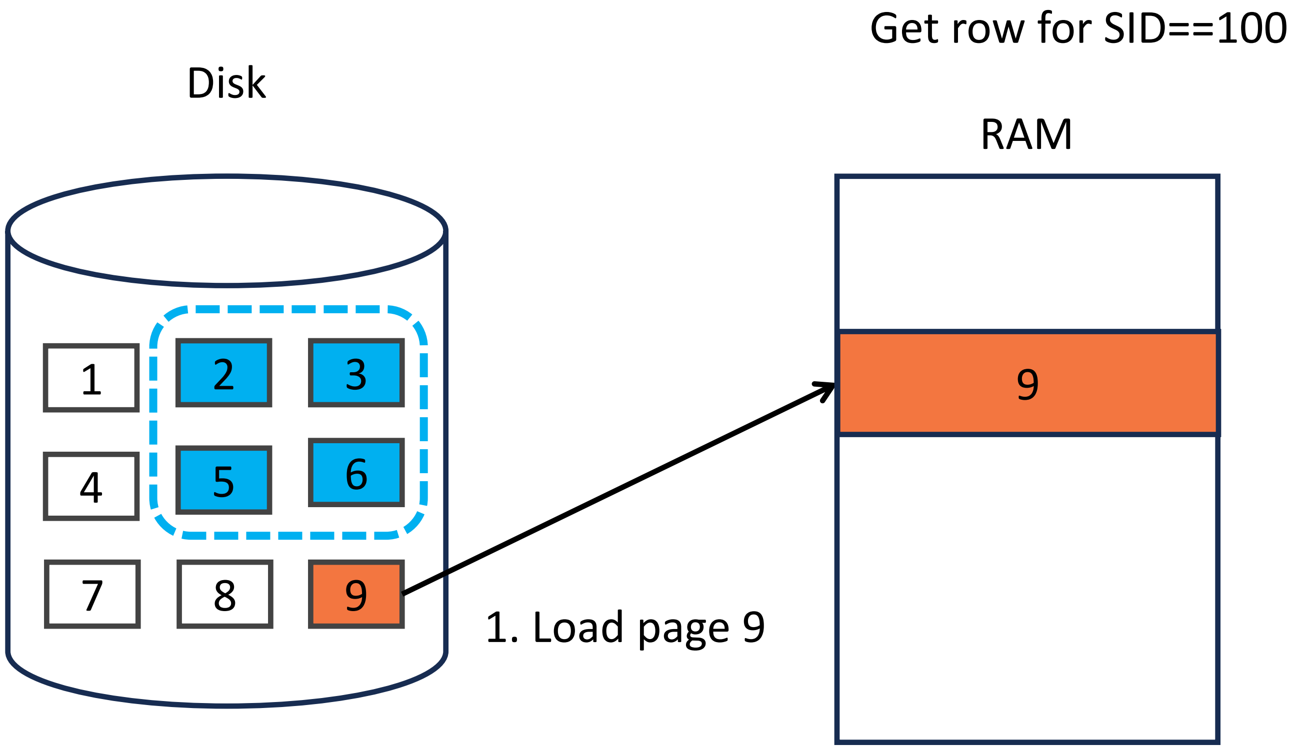
Get row for SID==100

Background: Data Retrieval (B-Tree)

Get row for SID==100

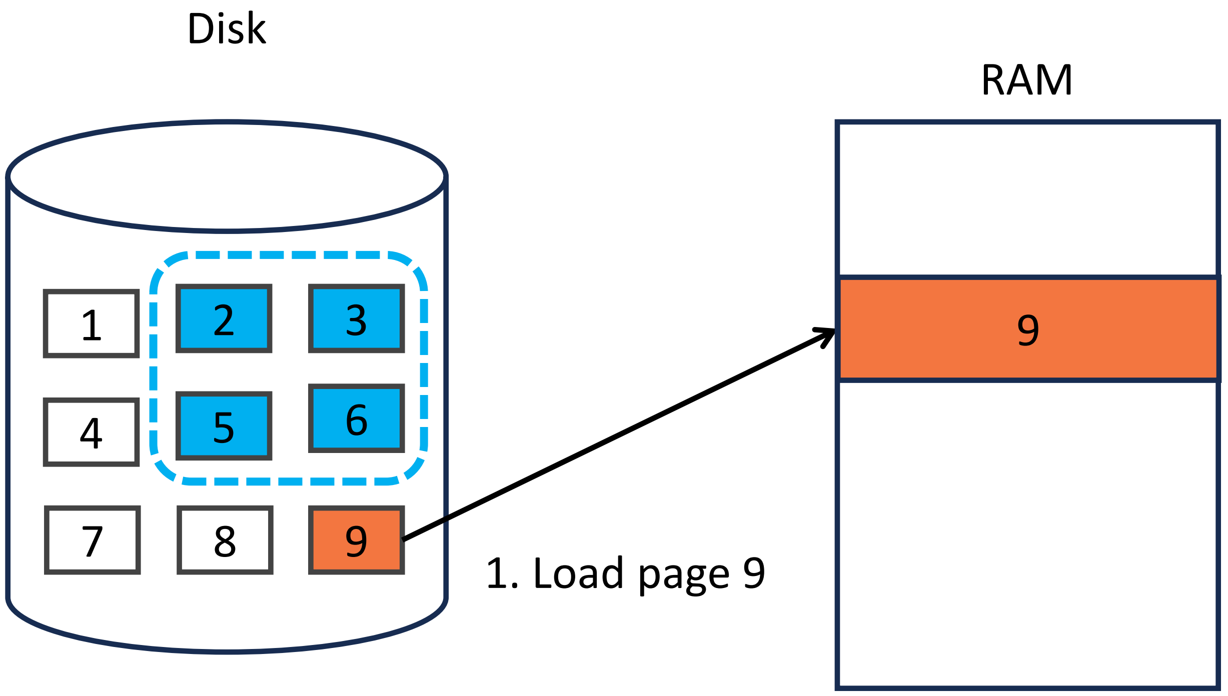


Background: Data Retrieval (B-Tree)



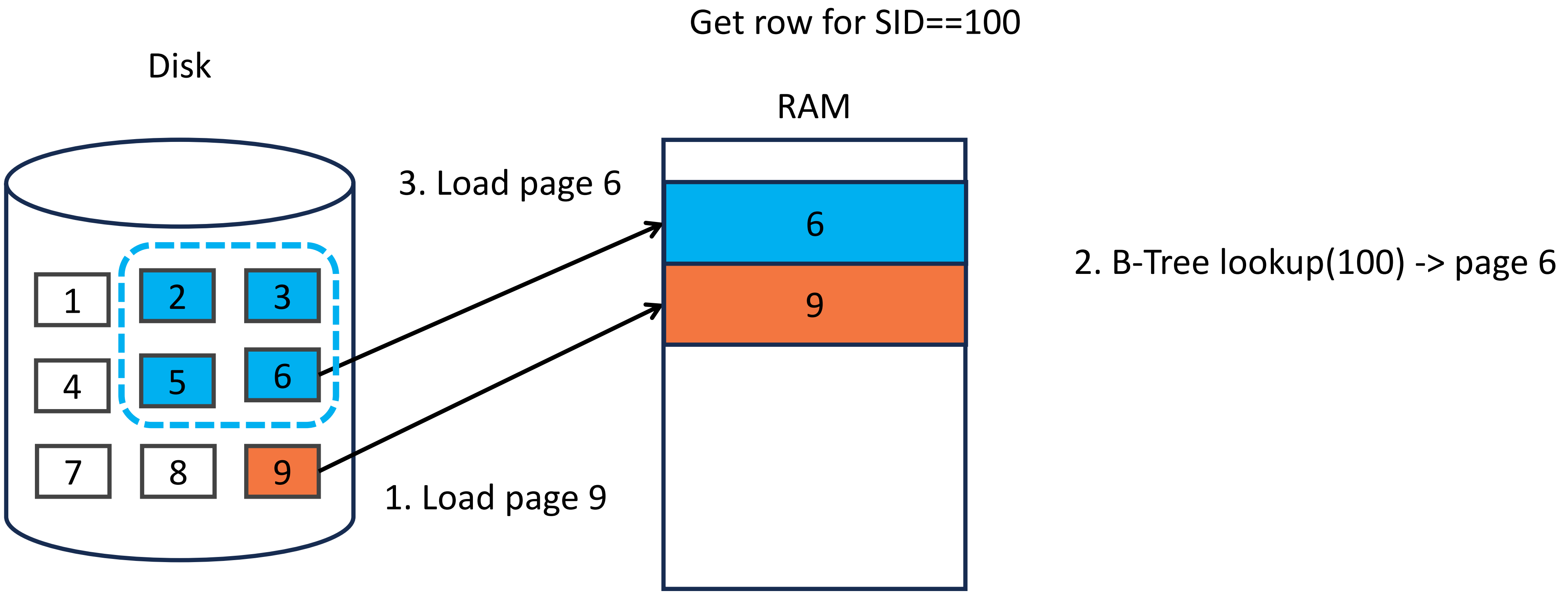
Background: Data Retrieval (B-Tree)

Get row for SID==100



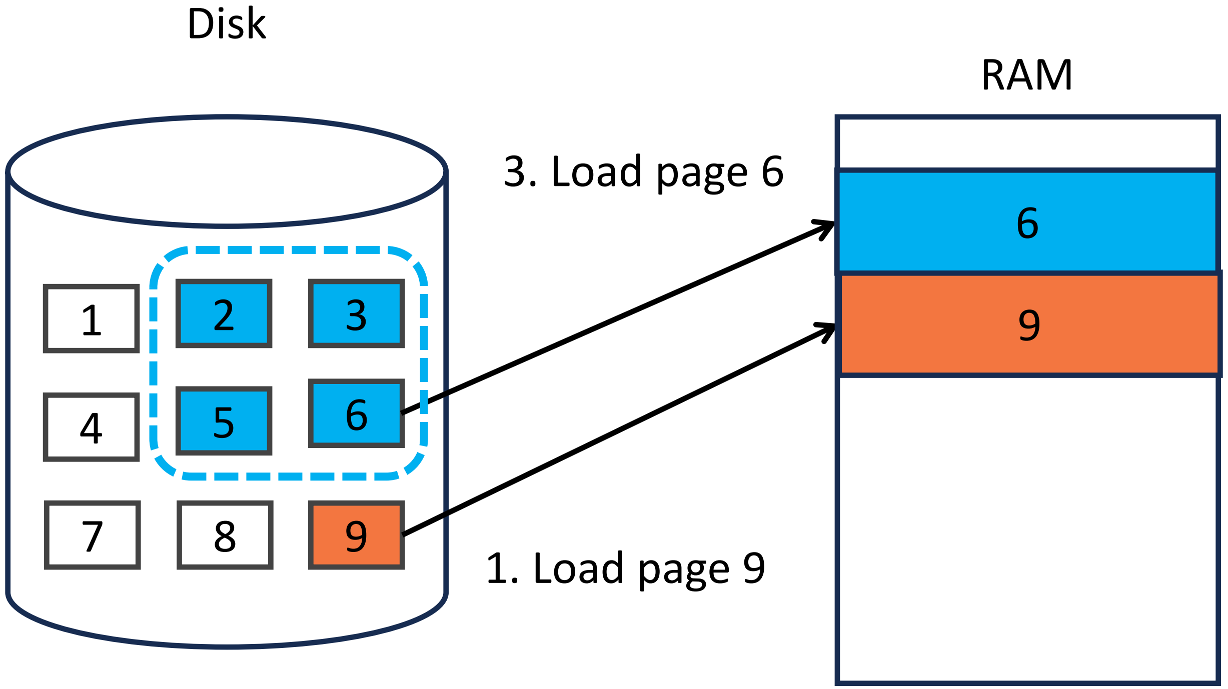
2. B-Tree Lookup(100) -> page 6

Background: Data Retrieval (B-Tree)



Background: Data Retrieval (B-Tree)

Get row for SID==100

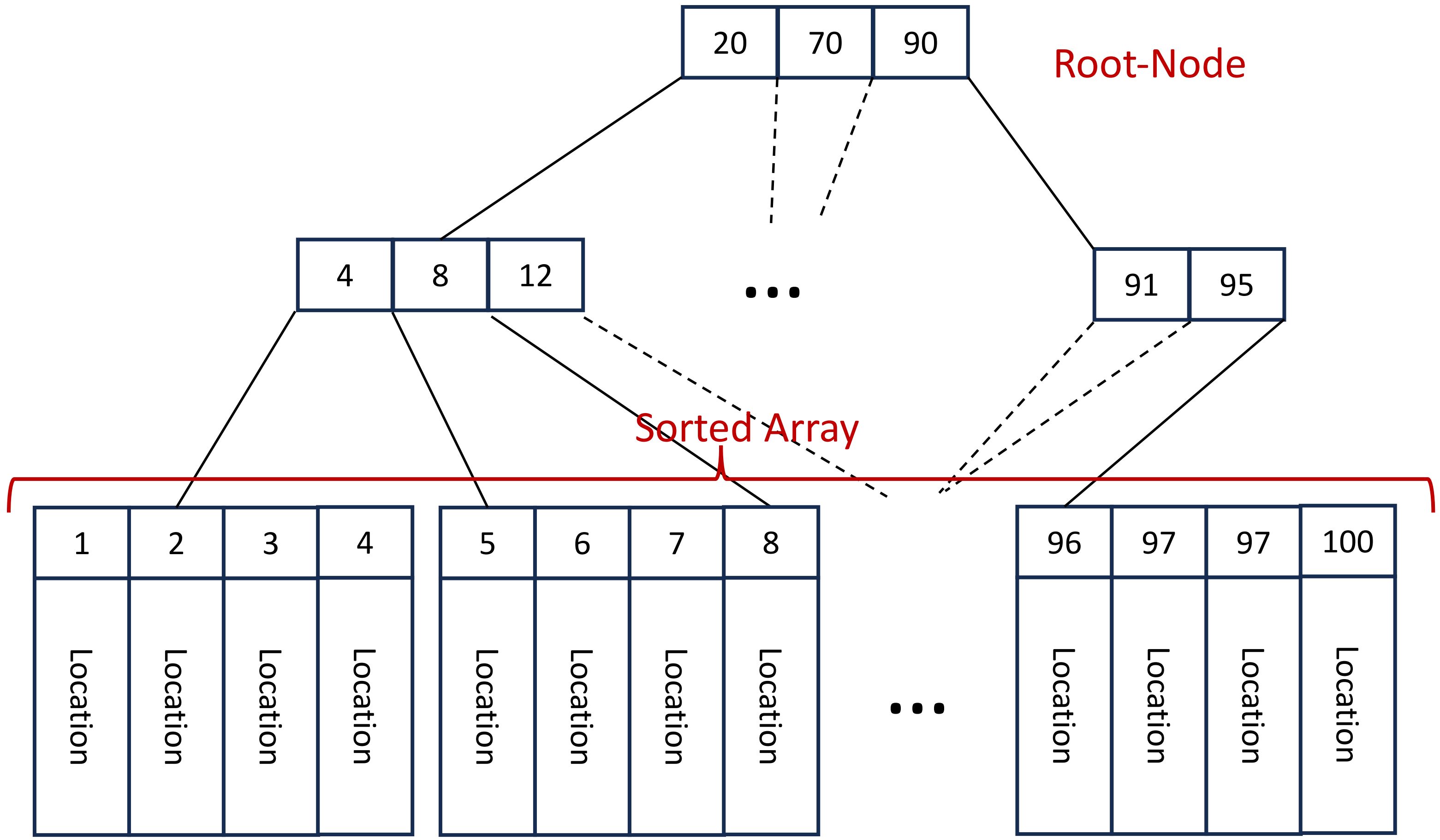


4. search SID==100 within page 6

2. B-Tree lookup(100) -> page 6

Disk I/O: 2 pages vs 4 pages
Time: O(log N) vs O(N)
Space cost: O(size of SID)

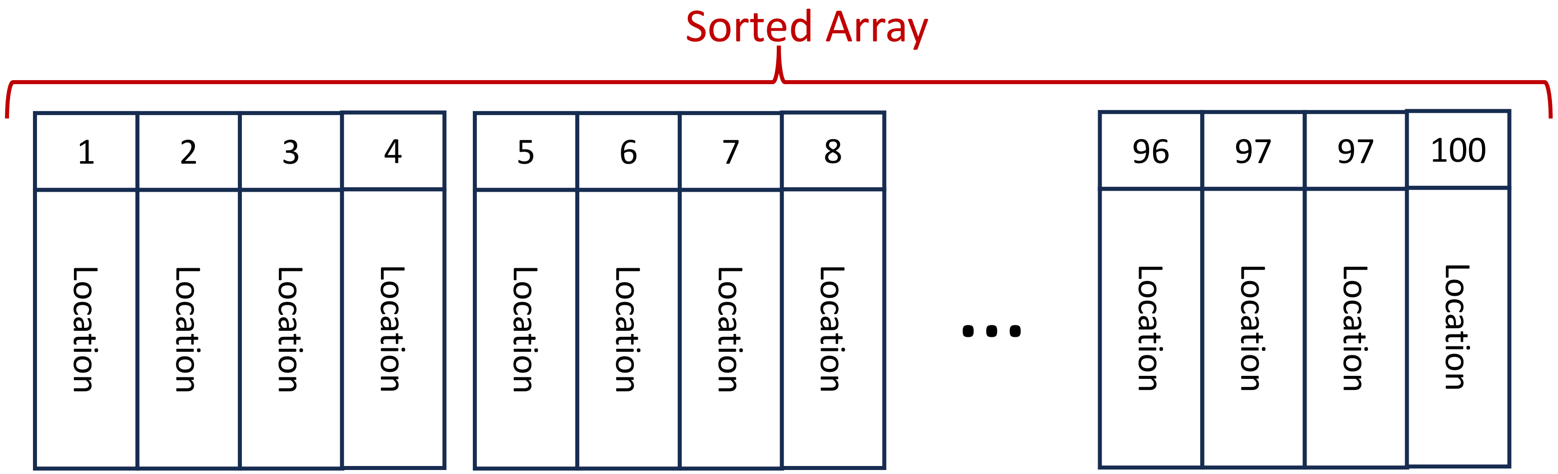
What if keys are uniformly distributed between 1 – 100?



What if keys are uniformly distributed between 1 – 100?

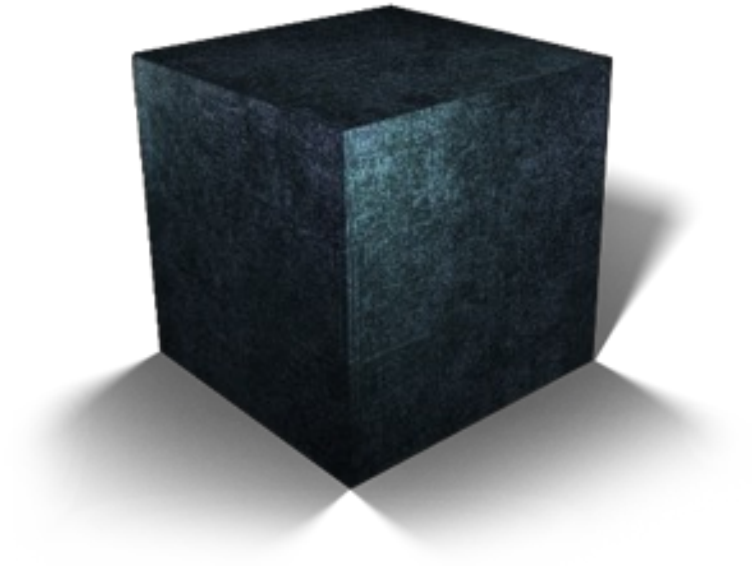
data_array[lookup_key]

B-Tree is unnecessary
O(1) Lookup
O(1) memory

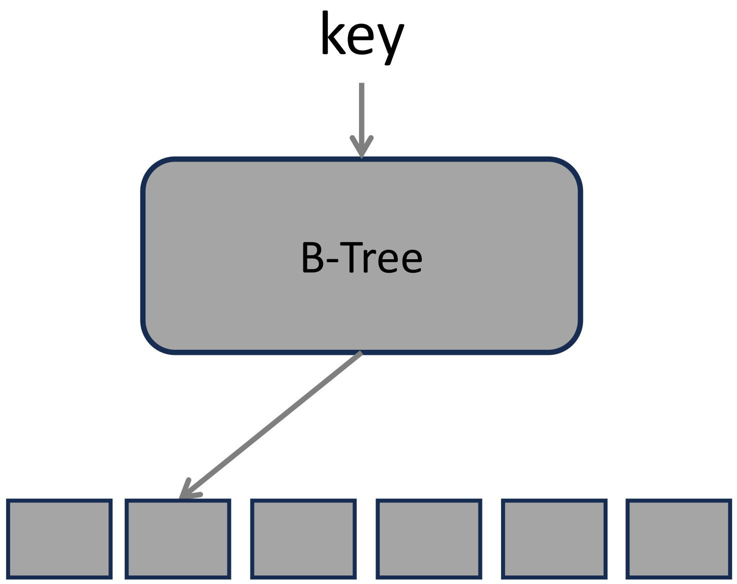


Learned Index Structures: Key Insights

- Traditional index structures make no assumptions about the data (black-box)
  - Scales with the data size
- Learning the data distribution allows for performance improvements
  - Overfitting
  - Scales with the complexity of data distribution, not size



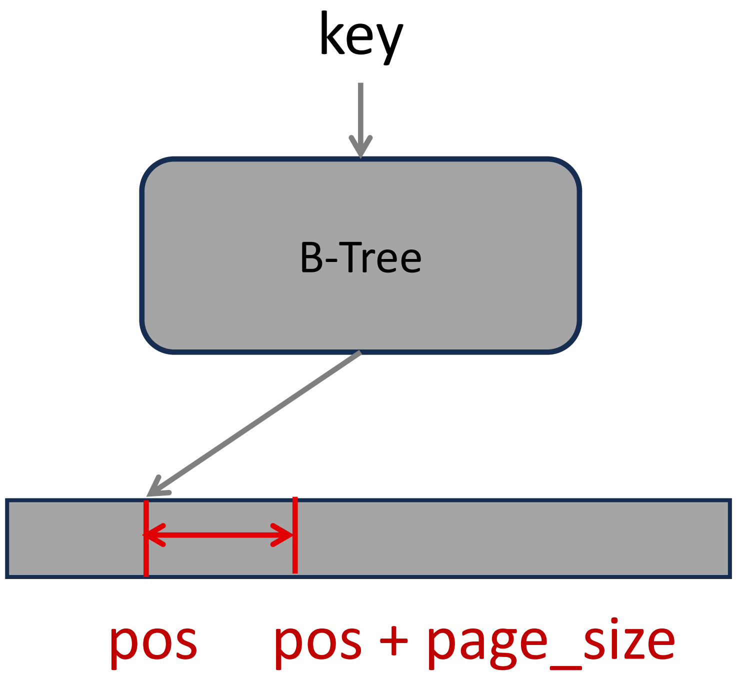
B-Trees



A B-Tree maps a key to a page

Then searches within the page

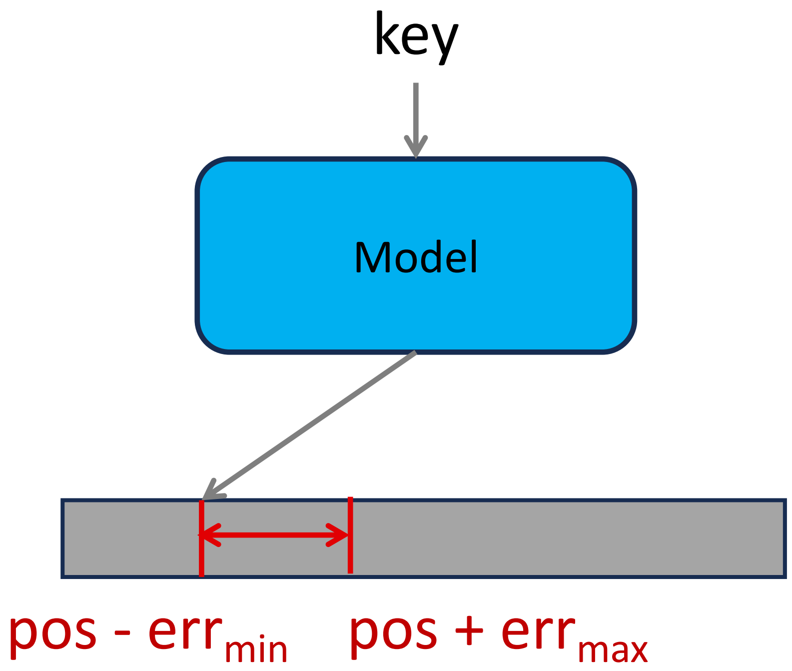
B-Trees



A B-Tree: key -> pos

Then searches from: [pos, pos + page_size]

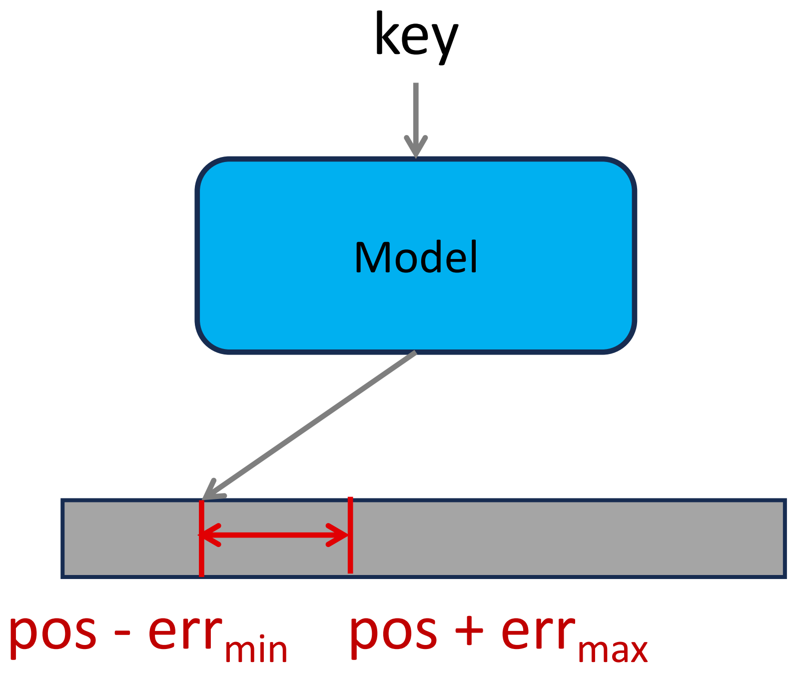
B-Trees are Models



Model: f(key) -> pos

Then searches from: [pos - err$_{mix}$, pos + err$_{max}$]

B-Trees are Models

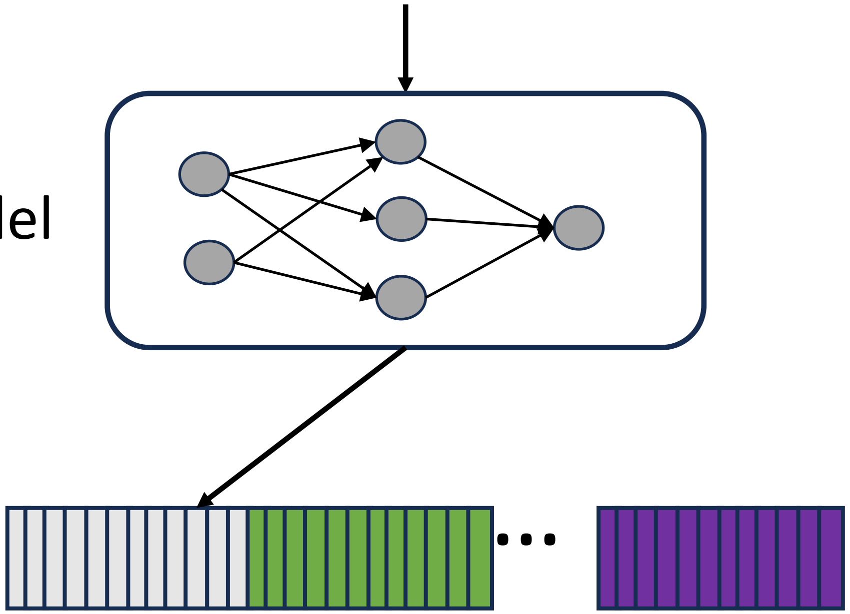


Model: f(key) -> pos

This is equivalent to modeling the CDF

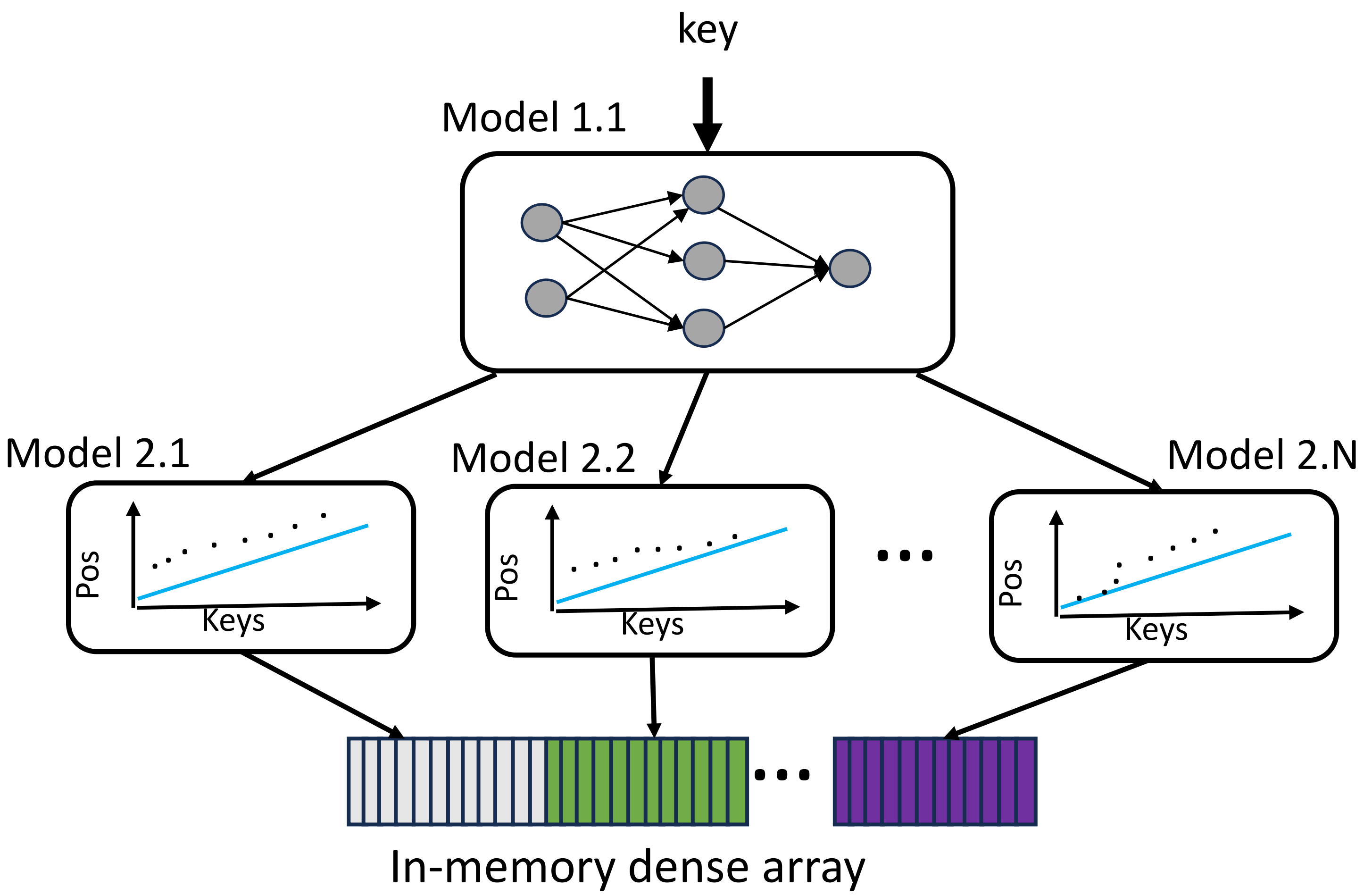
Pos = P(x ≤ key) * num_keys

Learned Index Structures: Naïve Approach

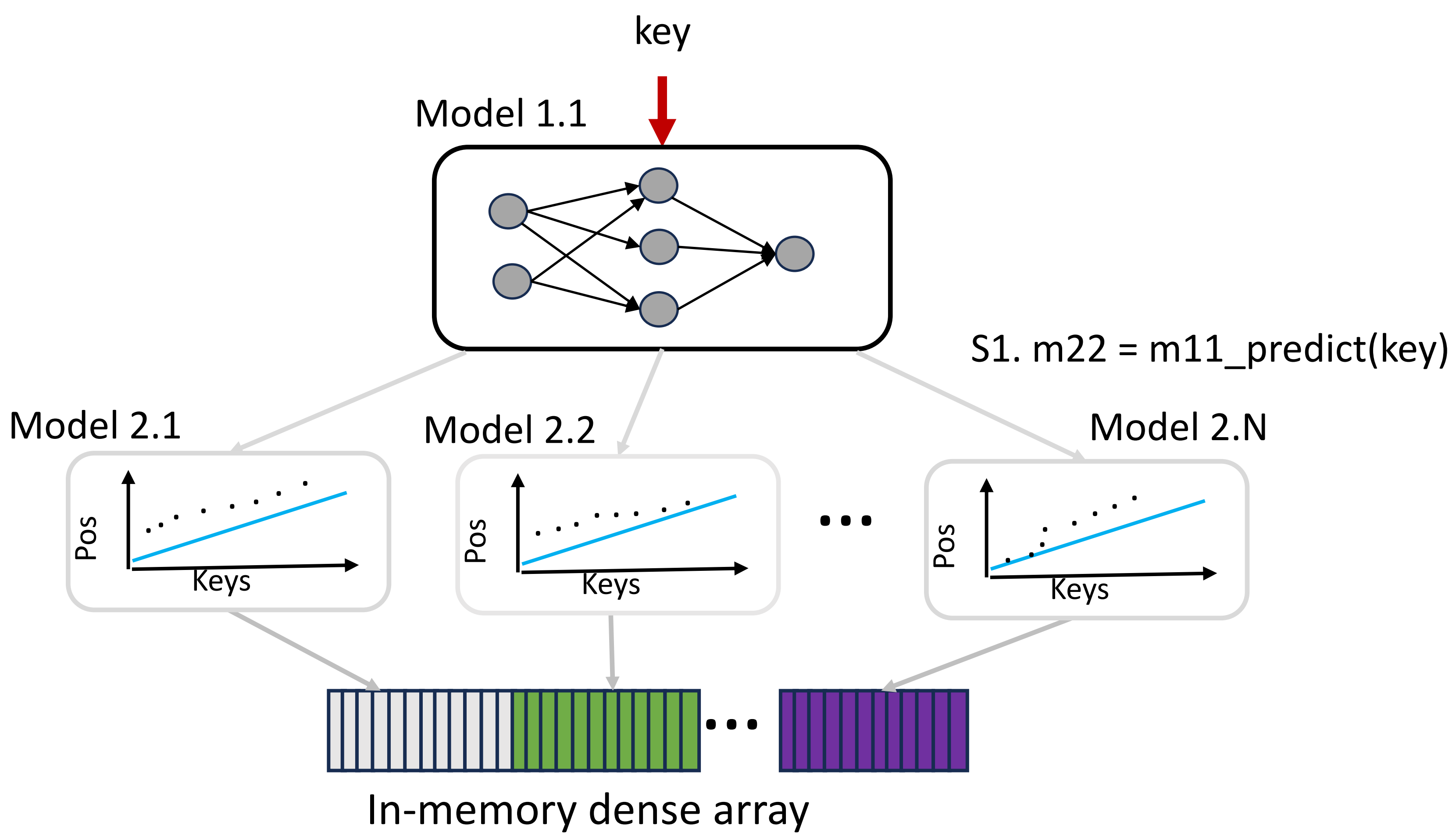


SID	Name	Major	GPA
1	Jay	DS	4.0
2	John	CS	3.9
…	…	…	…
99	Morgan	CS	3.95
100	Matt	CE	3.66

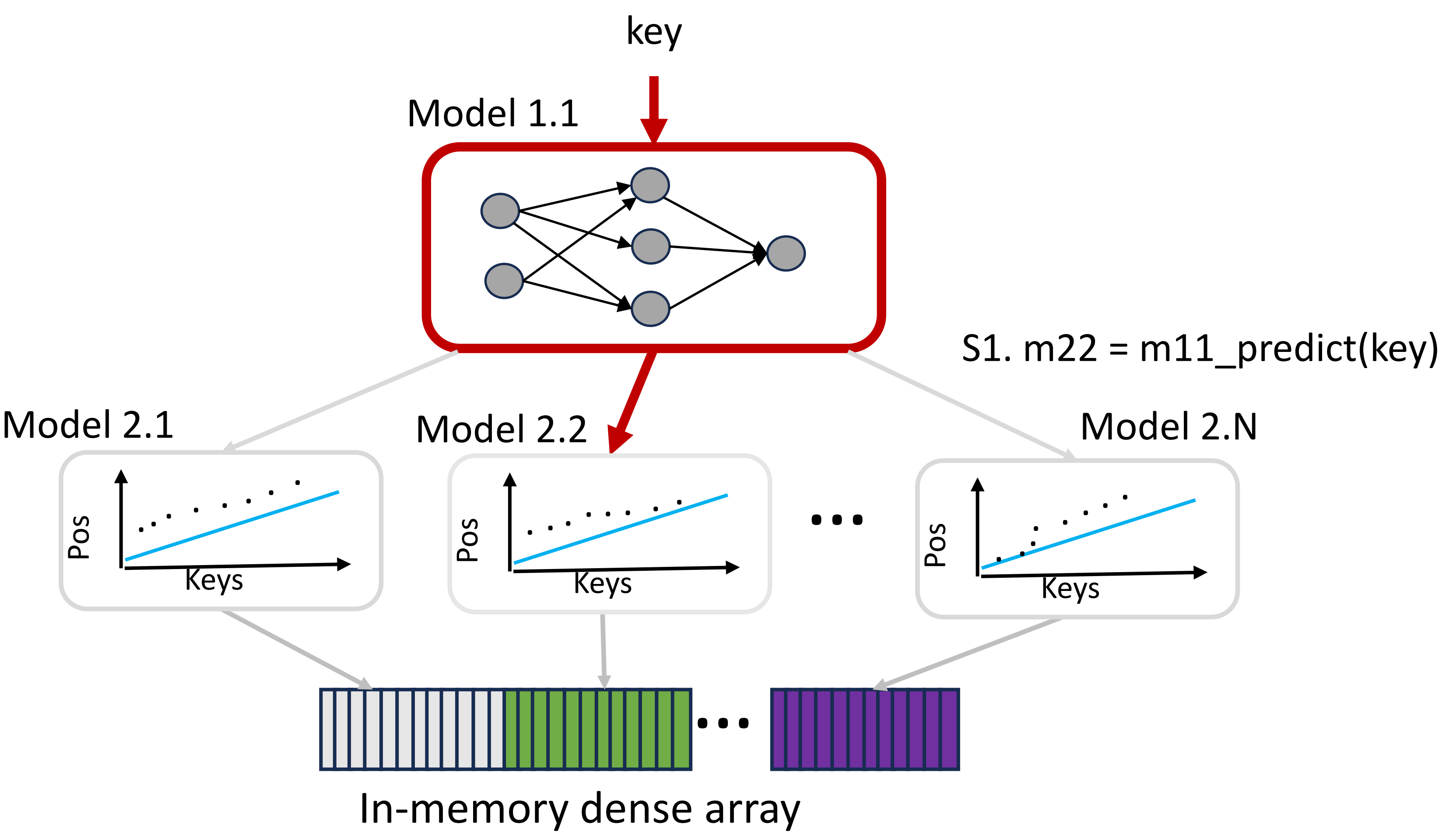
Learned Index Structures: RMI Overview



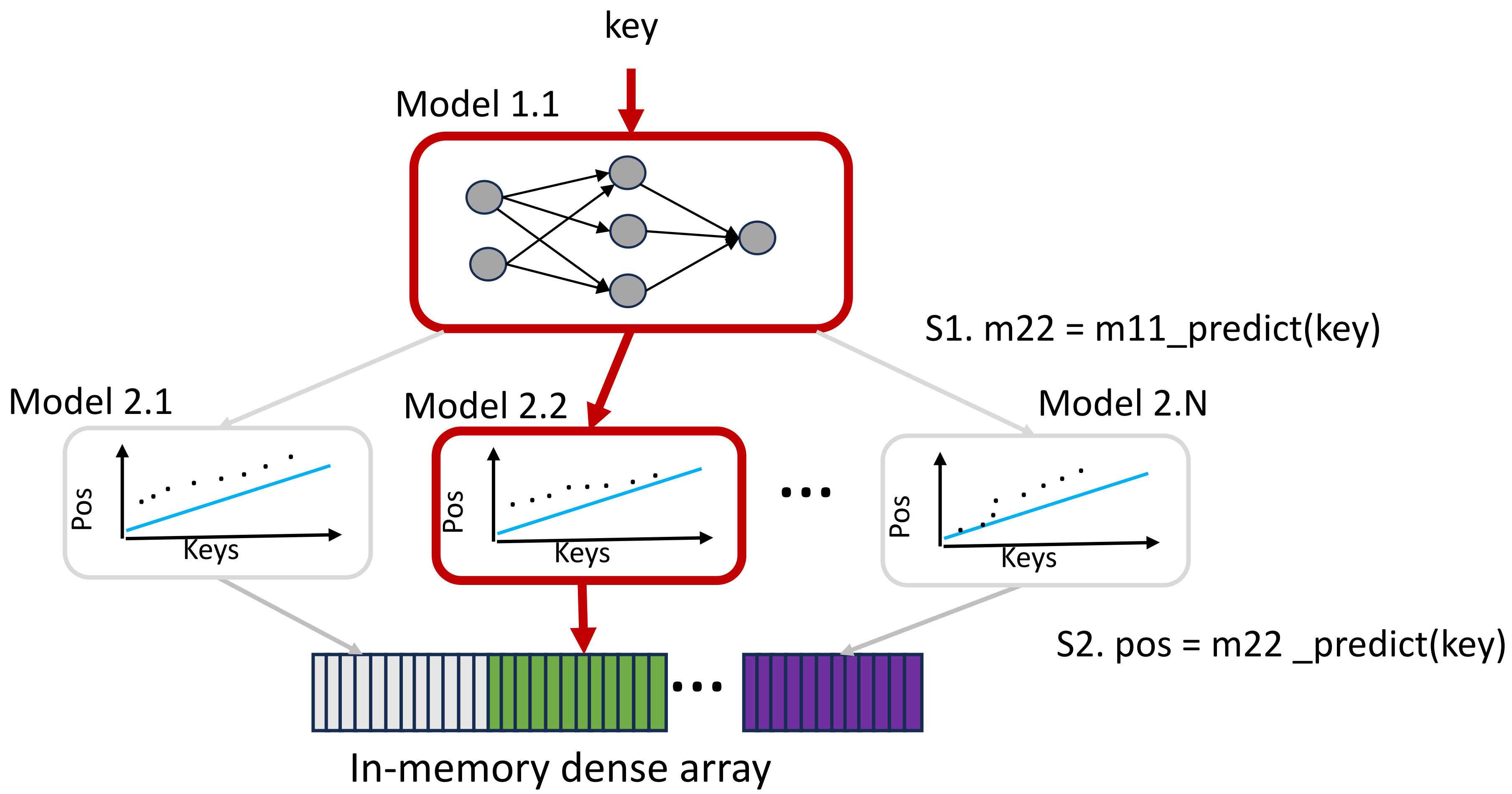
Learned Index Structures: RMI Lookup



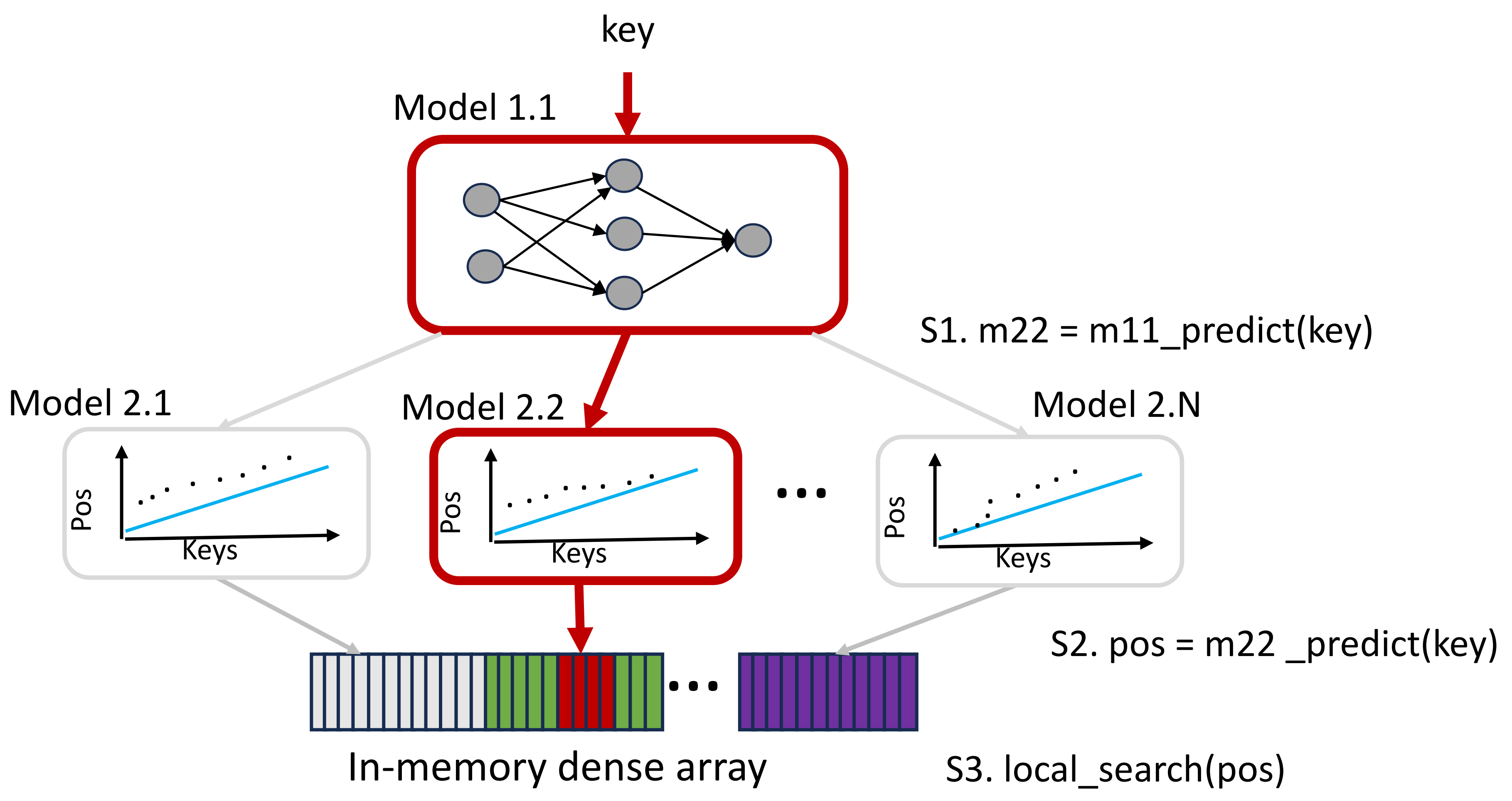
Learned Index Structures: RMI Lookup



Learned Index Structures: RMI Lookup

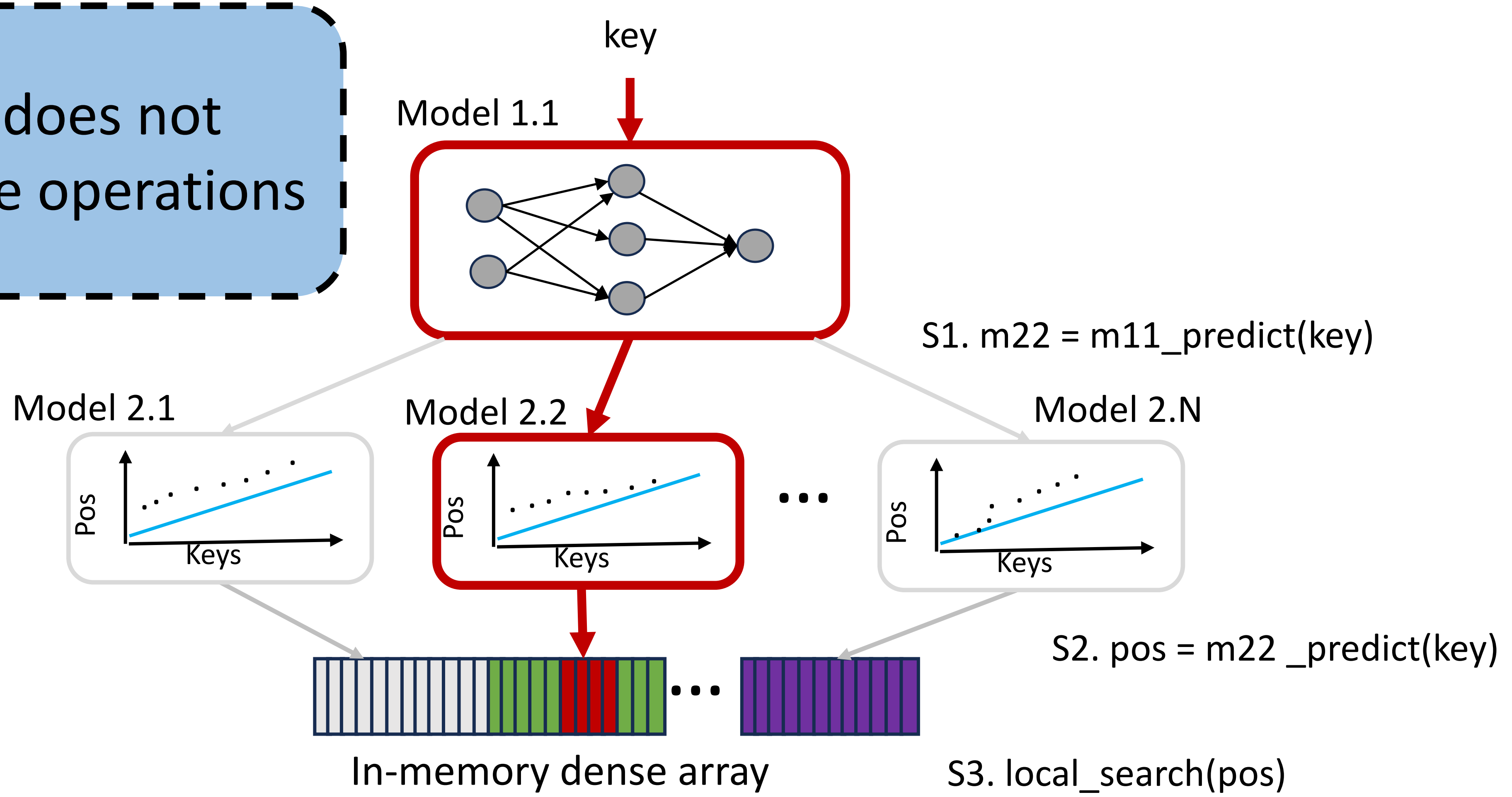


Learned Index Structures: RMI Lookup



Learned Index Structures: RMI Lookup

Limitation: does not support update operations



Discussion: B-Tree vs. Learned Index

		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 64	26.23 (2.00x)	277 (0.96x)	172 (62.0%)	25.97 (2.00x)	274 (0.95x)	171 (62.4%)	24.92 (2.00x)	274 (0.96x)	169 (61.7%)
	page size: 128	13.11 (1.00x)	265 (1.00x)	134 (50.8%)	12.98 (1.00x)	260 (1.00x)	132 (50.8%)	12.46 (1.00x)	263 (1.00x)	131 (50.0%)
	page size: 256	6.56 (0.50x)	267 (0.99x)	114 (42.7%)	6.49 (0.50x)	266 (0.98x)	114 (42.9%)	6.23 (0.50x)	271 (0.97x)	117 (43.2%)
	page size: 512	3.28 (0.25x)	286 (0.93x)	101 (35.3%)	3.25 (0.25x)	291 (0.89x)	100 (34.3%)	3.11 (0.25x)	293 (0.90x)	101 (34.5%)
Learned Index	2nd stage models: 10k	0.15 (0.01x)	98 (2.70x)	31 (31.6%)	0.15 (0.01x)	222 (1.17x)	29 (13.1%)	0.15 (0.01x)	178 (1.47x)	26 (14.6%)
	2nd stage models: 50k	0.76 (0.06x)	85 (3.11x)	39 (45.9%)	0.76 (0.06x)	162 (1.60x)	36 (22.2%)	0.76 (0.06x)	162 (1.62x)	35 (21.6%)
	2nd stage models: 100k	1.53 (0.12x)	82 (3.21x)	41 (50.2%)	1.53 (0.12x)	144 (1.81x)	39 (26.9%)	1.53 (0.12x)	152 (1.73x)	36 (23.7%)
	2nd stage models: 200k	3.05 (0.23x)	86 (3.08x)	50 (58.1%)	3.05 (0.24x)	126 (2.07x)	41 (32.5%)	3.05 (0.24x)	146 (1.79x)	40 (27.6%)

Figure 4: Learned Index vs B-Tree

Learned Index Structures: Roadmap

2018 — 2019 — 2020 — 2021 — 2022 — 2024

RMI [**SIGMOD’18**]

Learned Index Structures: Roadmap

2018 2019 2020 2021 2022 2024

RMI [**SIGMOD’18**]

FITing-Tree [**SIGMOD’19**]

Learned Index Structures: Roadmap

2018 2019 2020 2021 2022 2024

RMI [**SIGMOD'18**]

FITing-Tree [**SIGMOD'19**]

ALEX [**SIGMOD'20**]

PGM [**VLDB'20**]

Learned Index Structures: Roadmap

2018 2019 2020 2021 2022 2024

RMI [**SIGMOD’18**]

FITing-Tree [**SIGMOD’19**]

ALEX [**SIGMOD’20**]

PGM [**VLDB’20**]

LIPP [**VLDB’21**]

Learned Index Structures: Roadmap

2018 — RMI [**SIGMOD’18**]

2019 — FITing-Tree [**SIGMOD’19**]

2020 — ALEX [**SIGMOD’20**]; PGM [**VLDB’20**]

2021 — LIPP [**VLDB’21**]

2022 — APEX [**VLDB’22**]

2024

Learned Index Structures: Roadmap

2018 — RMI [**SIGMOD'18**]

2019 — FITing-Tree [**SIGMOD'19**]

2020 — ALEX [**SIGMOD'20**]

PGM [**VLDB'20**]

2021 — LIPP [**VLDB'21**]

2022 — APEX [**VLDB'22**]

2024 — Algorithmic Complexity Attacks on Dynamic Learned Indexes [**VLDB'24**]

Learned Index Structures: Roadmap

2018 2019 2020 2021 2022 2024

RMI [**SIGMOD'18**]

Today

FITing-Tree [**SIGMOD'19**]

ALEX [**SIGMOD'20**]

PGM [**VLDB'20**]

LIPP [**VLDB'21**]

APEX [**VLDB'22**]

Algorithmic Complexity Attacks on Dynamic Learned Indexes [**VLDB'24**]

Learned Index Structures: Roadmap

2018 — RMI [**SIGMOD’18**]

Today

2019 — FITing-Tree [**SIGMOD’19**]

2020 — ALEX [**SIGMOD’20**]

PGM [**VLDB’20**]

2021 — LIPP [**VLDB’21**]

2022 — APEX [**VLDB’22**]

2024 — Algorithmic Complexity Attacks on Dynamic Learned Indexes [**VLDB’24**]

Our work on investigating the robustness of dynamic learned indexes

Algorithmic Complexity Attacks on Dynamic Learned Indexes [**VLDB’24**]:
https://www.vldb.org/pvldb/vol17/p780-yang.pdf

Data Systems Group @ MIT

Home | ML for Systems | Projects | People | Blog

ML for Systems Papers

This list is incomplete. If we are missing a paper, please email mlsyspapers@lists.csail.mit.edu and we will include it. If you would like to be informed about new research papers, subscribe here.

Acknowledgement: Parts of this list were sourced from this repository.

Table of Contents

285 papers

RMI Demo and Quiz 8