Two Birds, One Stone: A Fast, yet Lightweight, Indexing Scheme for Modern Database Systems

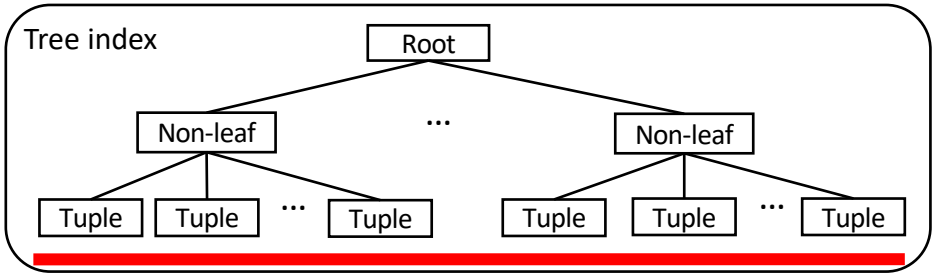
Jia Yu and Mohamed Sarwat

Arizona State University

Arizona State University

Motivation

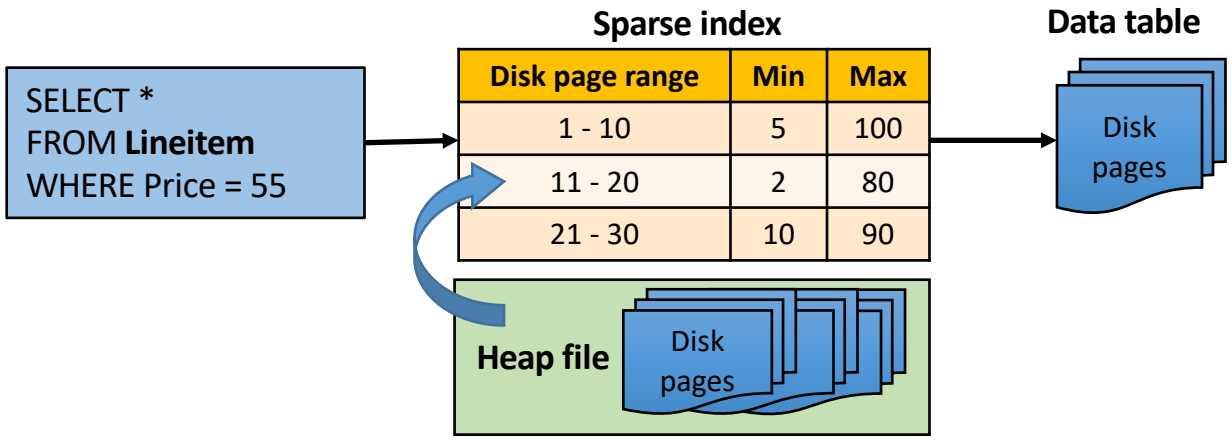
- **Tree index**
  - Fast index search
  - Large storage overhead: 5% - 15% additional overhead
  - Slow maintenance: Split, merge, redistribute nodes



B+ Tree on TPC-H	Index size	Initialization time	Insertion time (0.1%)
2 GB	0.25 GB	30 sec	10 sec
20 GB	2.51 GB	500 sec	1180 sec
200 GB	25 GB	8000 sec	42000 sec

Motivation (cont.)

- **Bitmap index**: Fast, small, low cardinality, read-only
- **Compressed index**: Small, slow search, slow maintenance
- **Approximate index**: Inaccurate result
- **Sparse index** (aggregated page info: min, max): Very small, not good at queries



Design goals

- **Low indexing overhead** (size and initialization time)
- **Fast index maintenance** (insertion and deletion)
- **Competitive query performance**
  - No need to be fast on all selectivity scenarios
  - Works for "**not too selective**" but still "**selective**" query

TPC-H Lineitem Table

ID	QUANTITY	PRICE	SHIPDATE	Comment
1	4	$50	1995-08-01	N/A
2	22	$90	1994-11-27	Refurbished
…	…	…	…	…

Highly selective, <0.001%

```
SELECT *
FROM Lineitem
WHERE Price > 0.00 AND Price < 0.01
```

Use B+ Tree

Still selective 0.01% - 1%

```
SELECT average(*)
FROM Lineitem
WHERE Price > 55 AND Price < 65
```

Use our index

Unselective > 1%

```
SELECT *
FROM Lineitem
WHERE Price > 10
```

Full table scan

Arizona State University

Hippo Index Structure

- Sparse index

Complete load balanced histogram

Bucket	Price
1	1 - 20
2	21 - 40
3	41 - 60
4	61 - 90
5	91 - 120

Manage

Hippo logical structure

Disk Page #	Partial histogram	Internal data
1 - 10	2,3,4	21,22,55,75,77
11 – 25	2,4,5	23,24,62,91,92
26 - 30	1,2,5	11,12,25,101,110

Grouped by partial histogram similarity

Bucket	Price
2	21 - 40
3	41 - 60
4	61 - 90

Partial histogram 1

Bucket	Price
2	21 - 40
4	61 - 90
5	91 - 120

Partial histogram 2

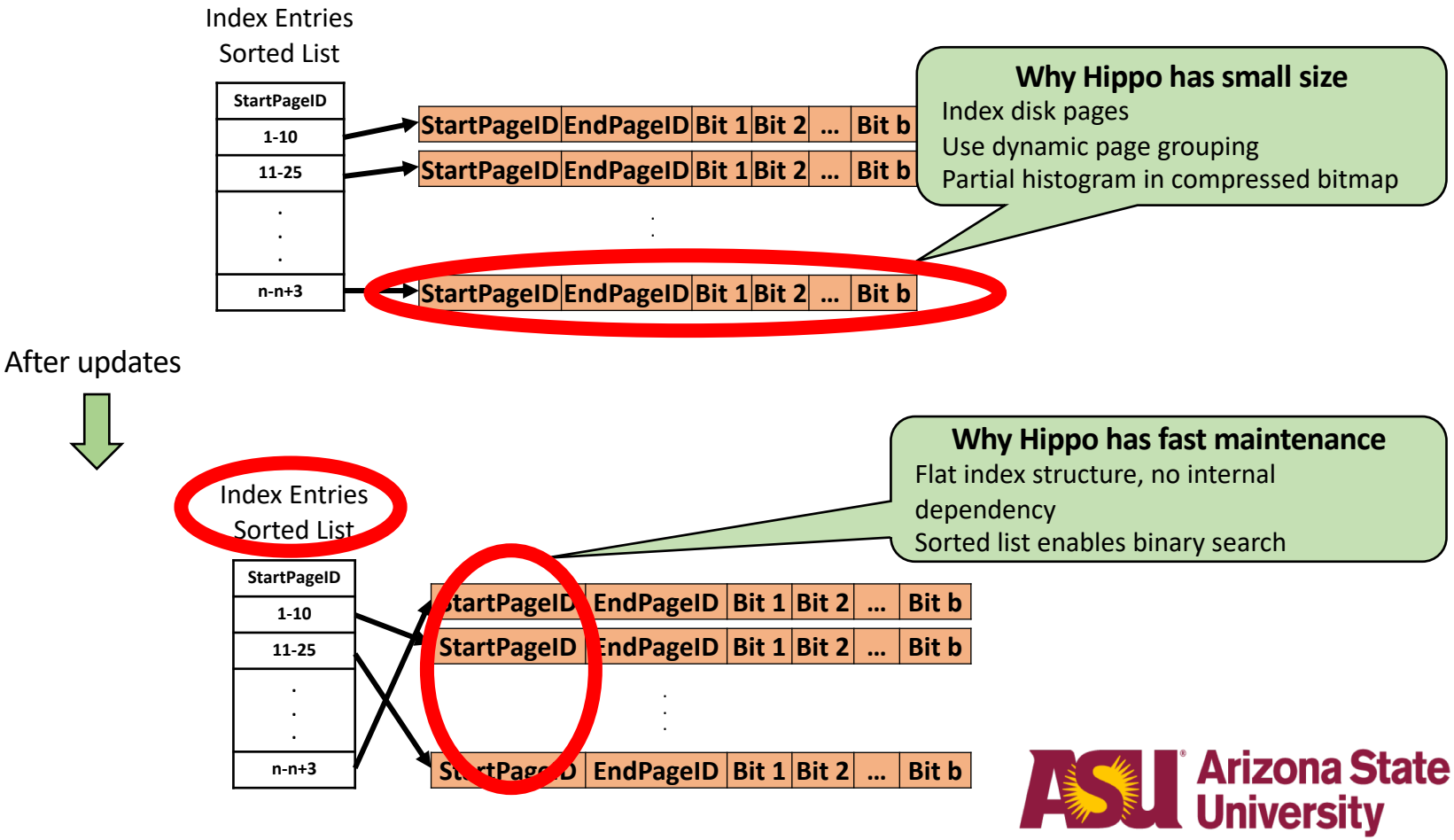
Bucket	Price
1	1 - 20
2	21 - 40
5	91 - 120

Partial histogram 3

- ✓ No initial. overhead
- ✓ Load-balanced, handle data skewness
- ✓ Global data distribution won't change frequently

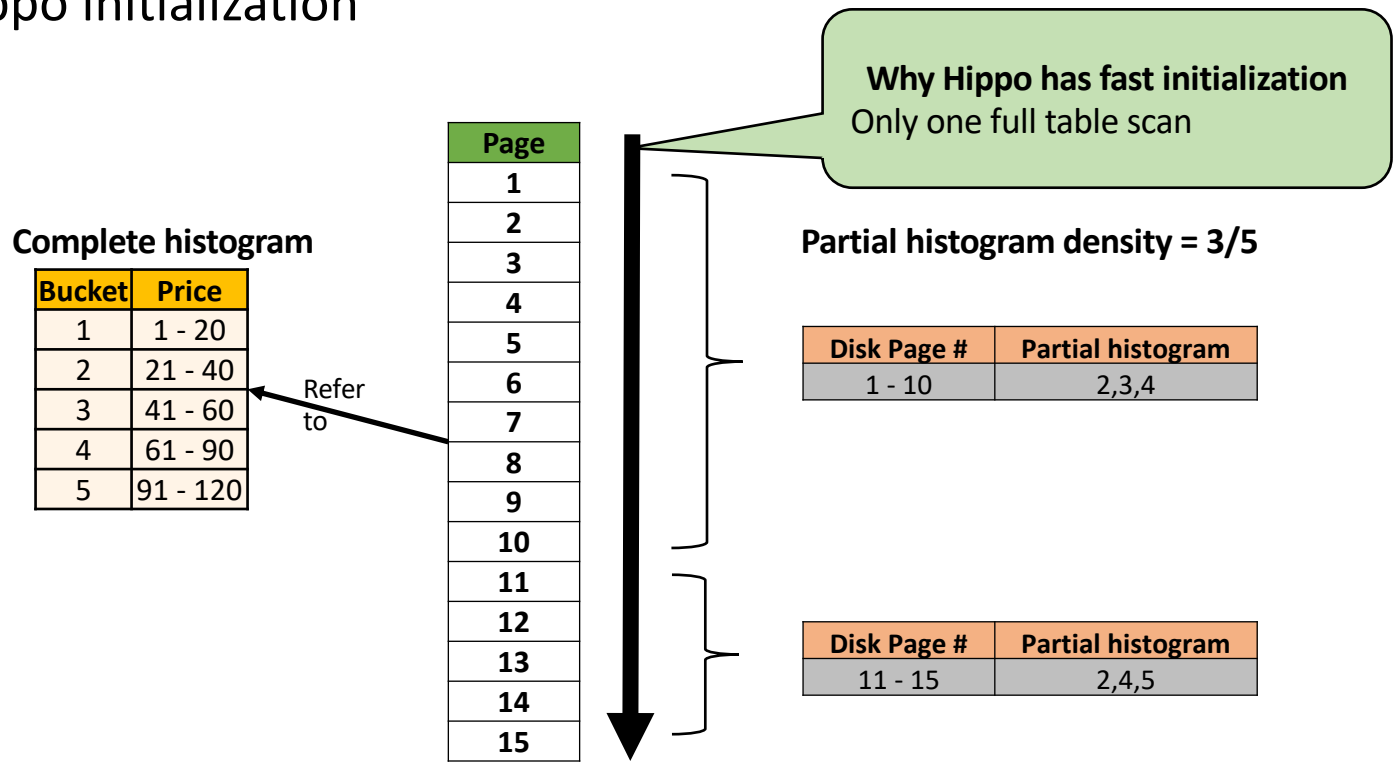
Hippo Index Structure (cont.)

- Hippo disk structure



Hippo Index Structure (cont.)

- Hippo initialization



Hippo Index Search

- Step 1: Scanning index entries

```
SELECT *
FROM Lineitem
WHERE Price = 55
```

Bucket	Price
1	1 - 20
2	21 - 40
3	41 - 60
4	61 - 90
5	91 - 120

```
SELECT *
FROM Lineitem
WHERE Price > 55
```

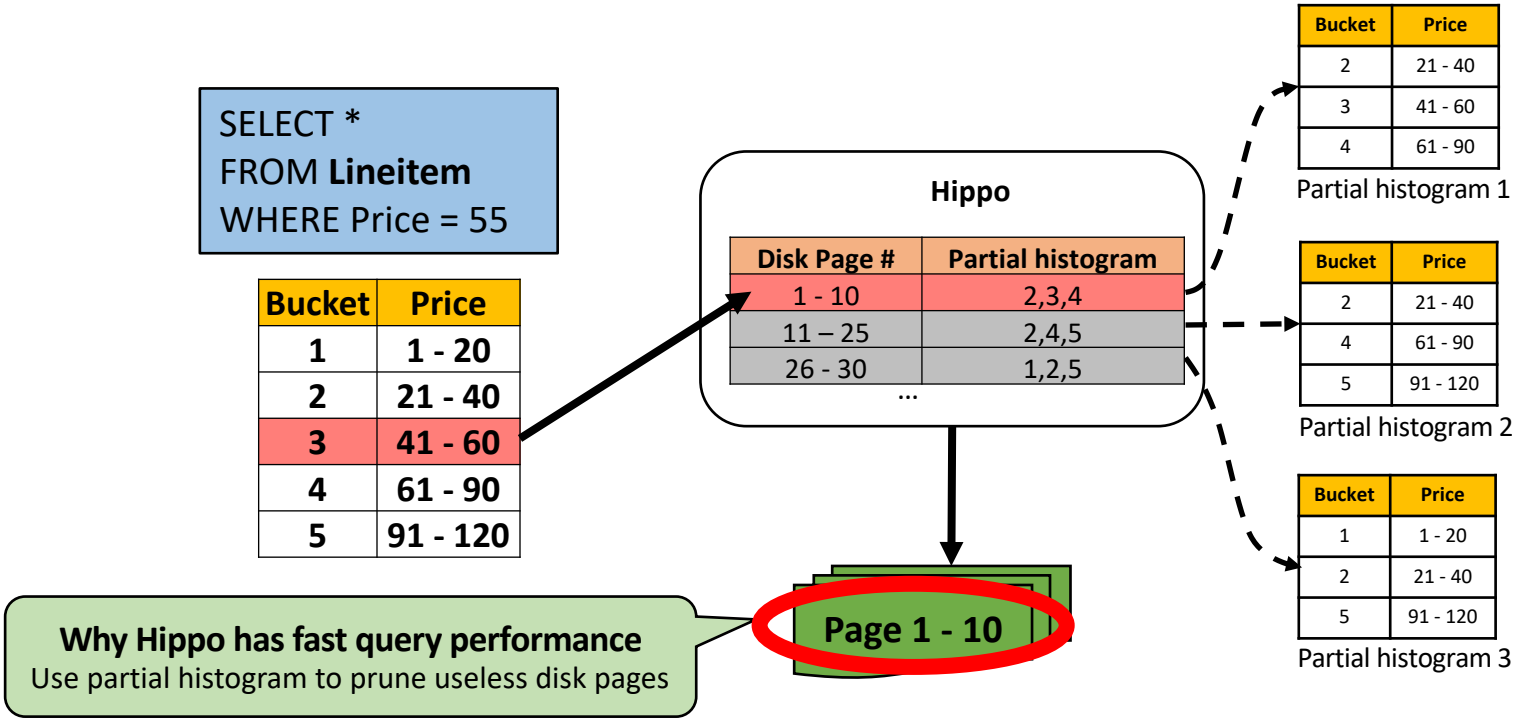
Bucket	Price
1	1 - 20
2	21 - 40
3	41 - 60
4	61 - 90
5	91 - 120

```
SELECT *
FROM Lineitem
WHERE Price > 55 AND Price < 65
```

Bucket	Price
1	1 - 20
2	21 - 40
3	41 - 60
4	61 - 90
5	91 - 120

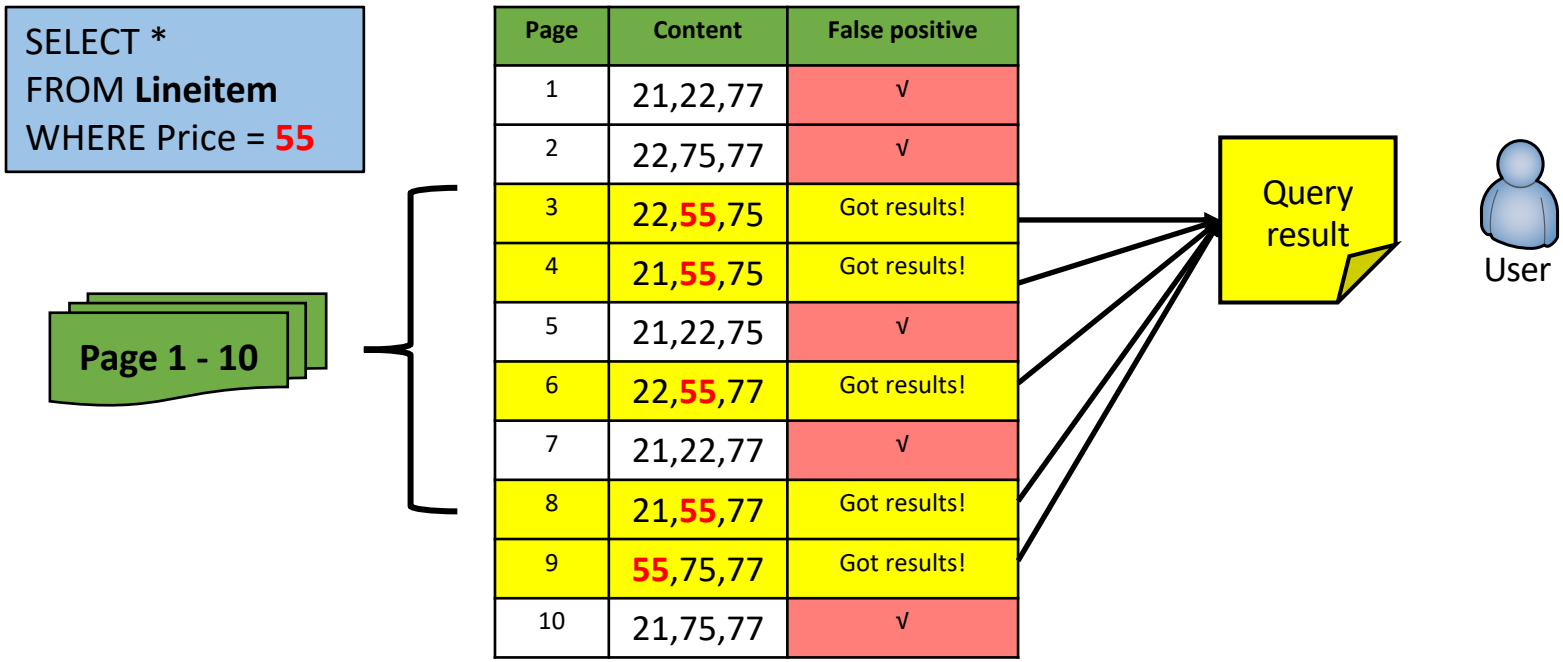
Hippo Index Search

- Step 1: Scanning index entries (continued)



Hippo Index Search

- Step 2: Filtering false positive pages



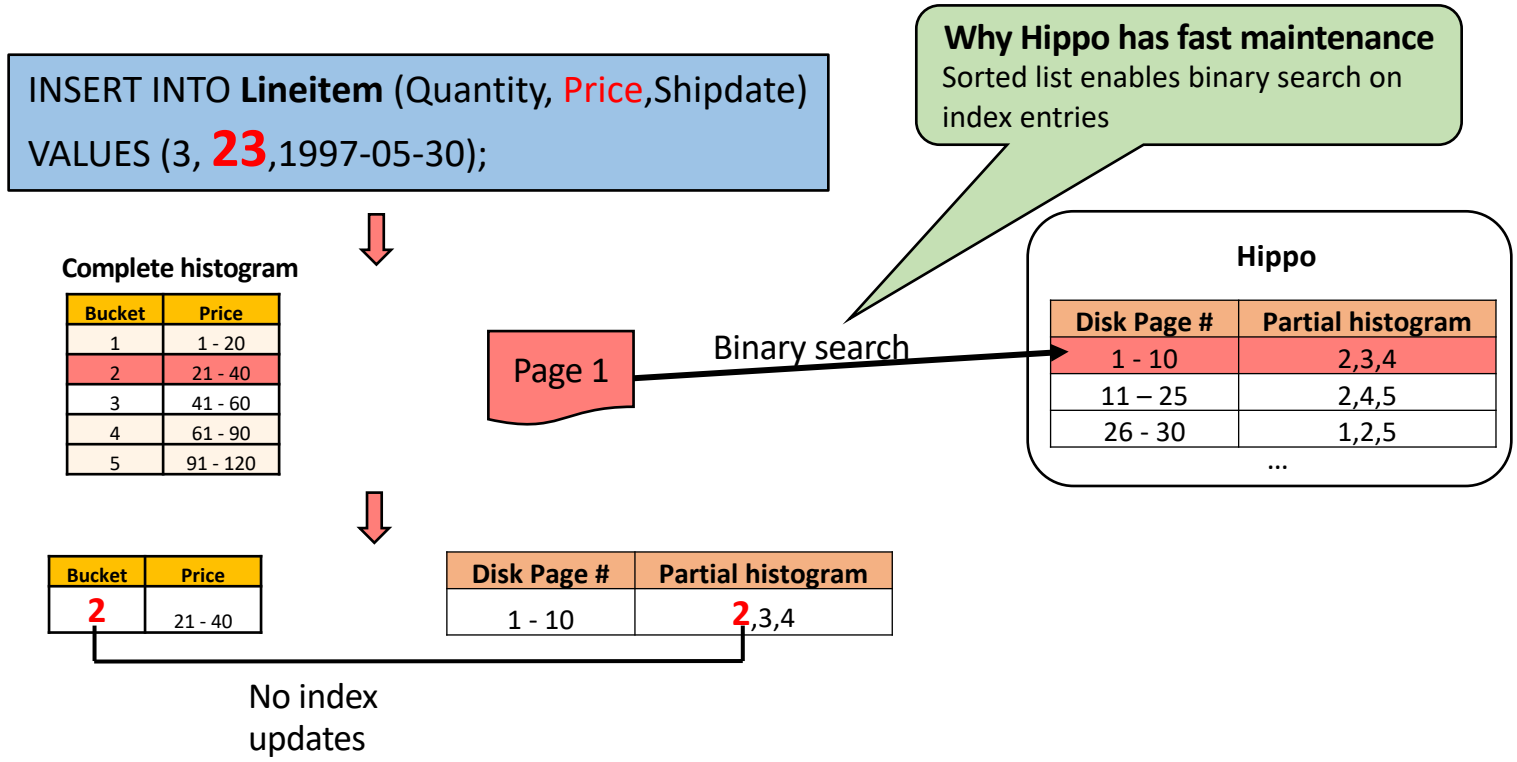
Page	Content	False positive
1	21,22,77	√
2	22,75,77	√
3	22,**55**,75	Got results!
4	21,**55**,75	Got results!
5	21,22,75	√
6	22,**55**,77	Got results!
7	21,22,77	√
8	21,**55**,77	Got results!
9	**55**,75,77	Got results!
10	21,75,77	√

Hippo Maintenance: Insertion - Eager

- **Check index entries right after each data insertion**
  - Check the complete histogram and each Hippo partial histogram
- **Update index entries right after each check**
  - S1. No need to update the corresponding index entry
  - S2. Need to update the corresponding index entry at once

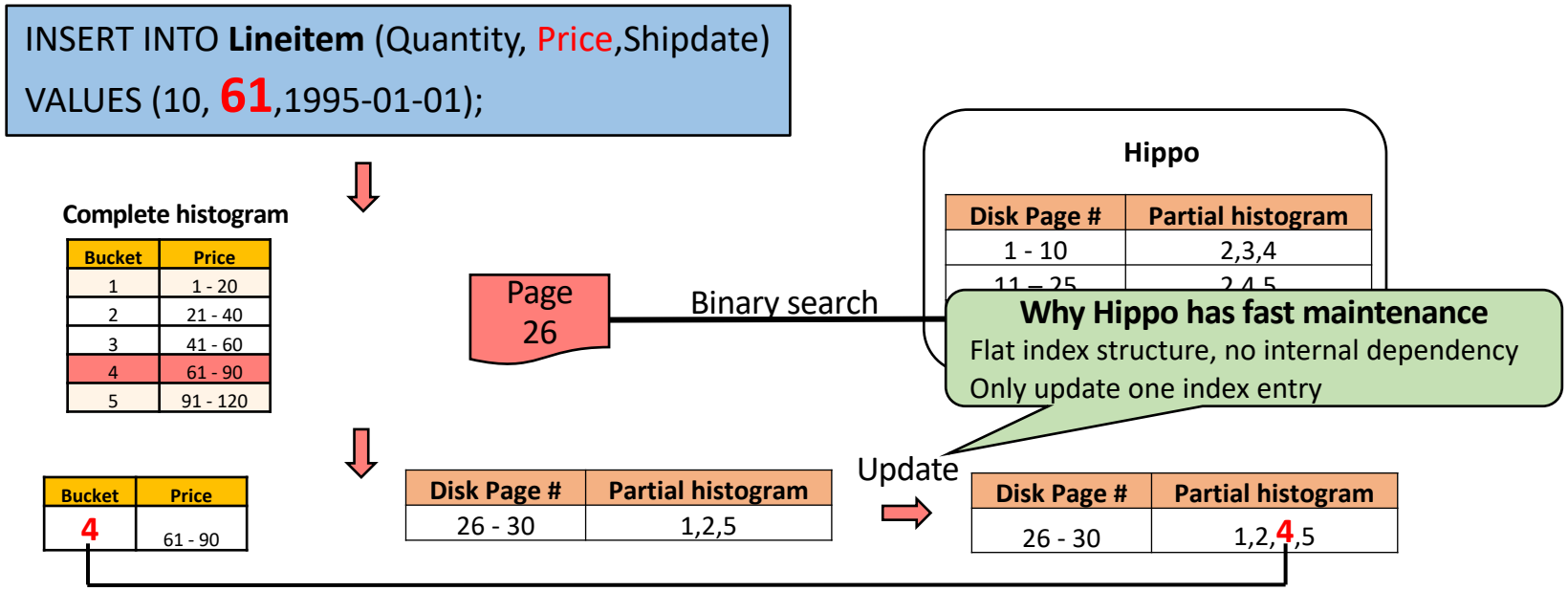
Hippo Maintenance: Insertion - Eager (cont.)

- Scenario 1: No need to update the entry



Hippo Maintenance: Insertion - Eager (cont.)

- Scenario 2: Need to update the entry

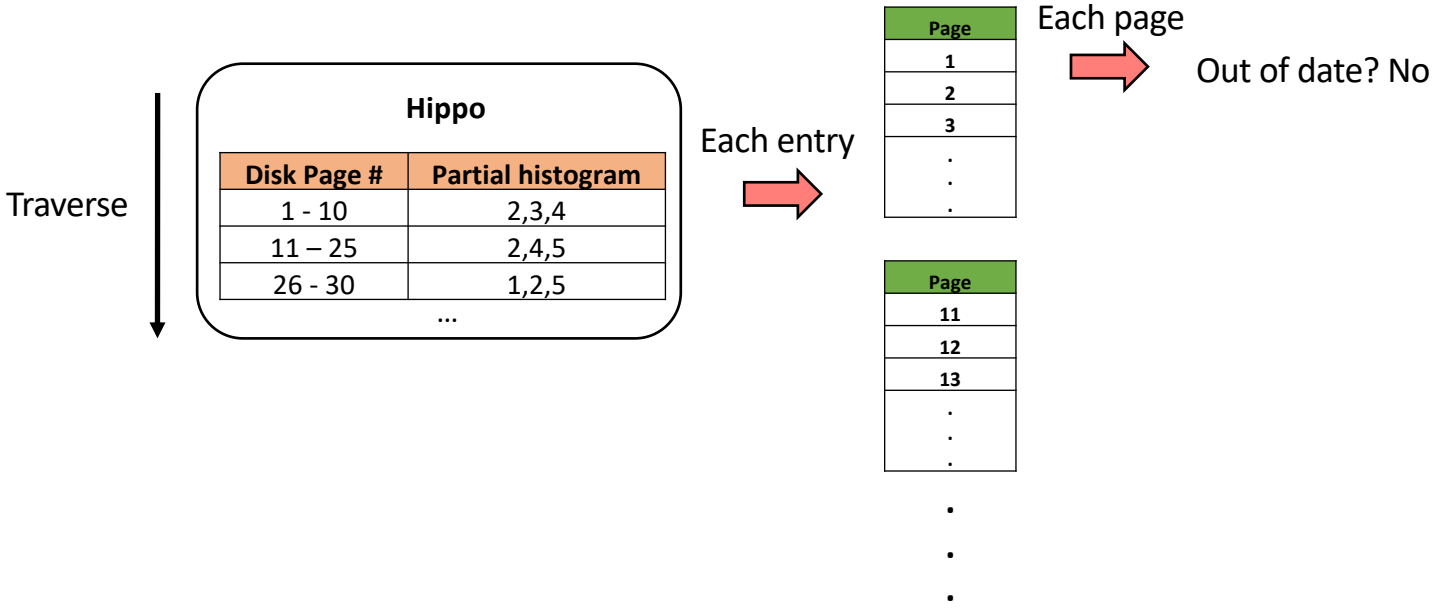


Hippo Maintenance: Deletion - Lazy

- **Don’t update index entries after each data deletion**
  - Deleted data tuples are marked as “deleted” and may be removed at any time
  - Still guarantee **accurate query results** because of “filtering false positive pages”
- **Update index periodically or invoked by the user**
  - Slow but can run at DBMS idle time
  - S1. No need to update the corresponding index entry
  - S2. Need to update the corresponding index entry

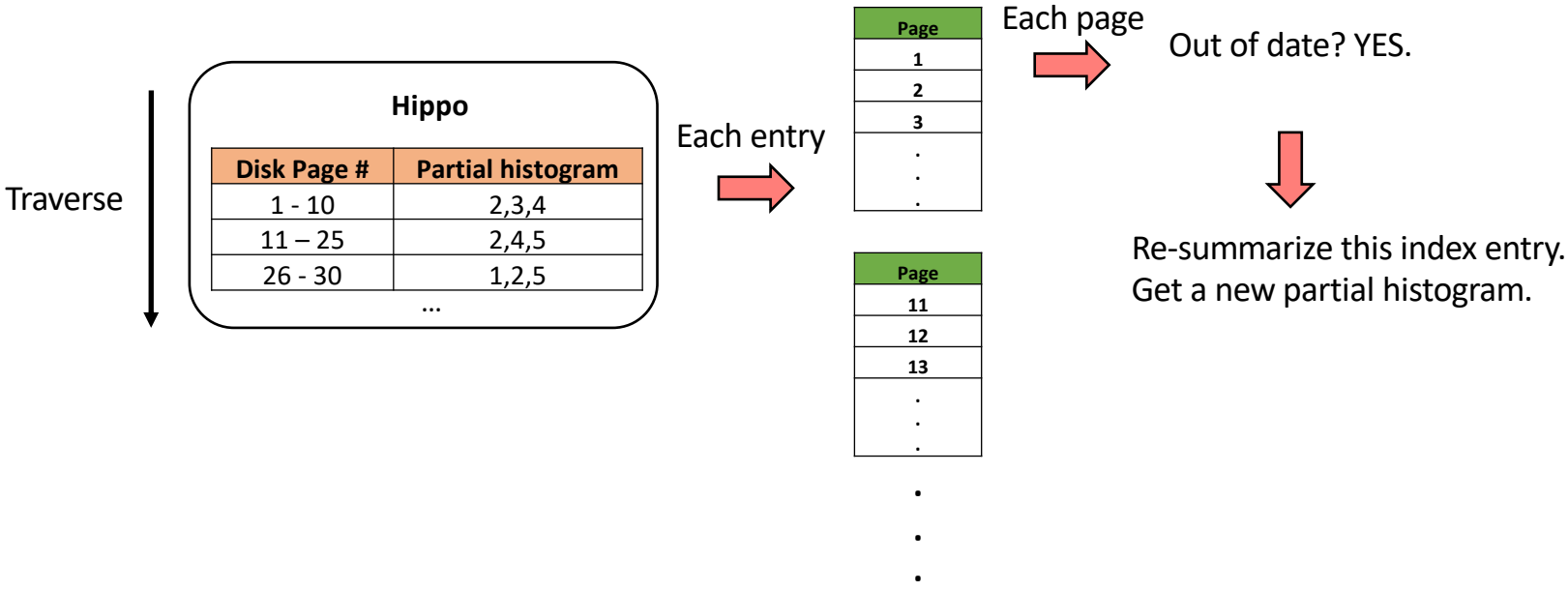
Hippo Maintenance: Deletion – Lazy (cont.)

- Scenario 1: No need to update the entry



Hippo Maintenance: Deletion – Lazy (cont.)

- Scenario 2: Need to update the entry



Experiments

- **Datasets (all around 200 GB):**
  1. TPC-H: Lineitem table
     1. PartKey: Uniform distribution. 40 million distinct values
     2. SuppKey: Uniform distribution. 2 million distinct values
     3. OrderKey: sorted
  2. Exponential distribution data: Skewed distribution
  3. Wikipedia page hit rates: Skewed distribution
  4. NYC Taxi trip records: pickup point, dimension reduction using Hilbert Curve
- **Compared indexes:**
  1. B+ Tree
  2. Hippo
  3. Block Range Index - BRIN (A sparse index)

Experiments (cont.)

- Indexing overhead
  - Size: Hippo 40x < B+ Tree
  - Initial. time: Hippo 2.5x < B+ Tree

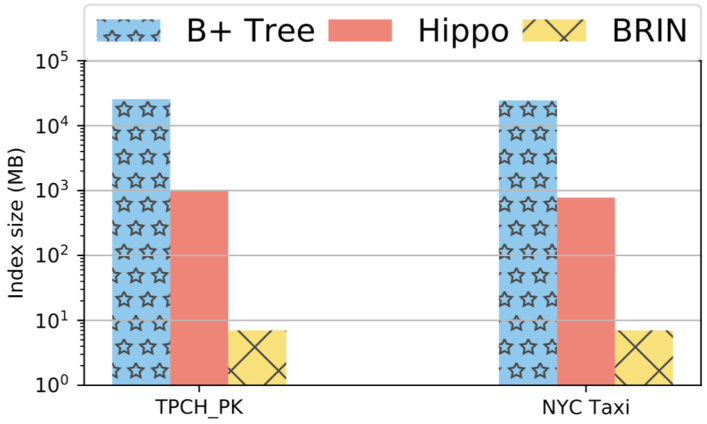


Figure: Index size on different datasets (logarithmic scale)

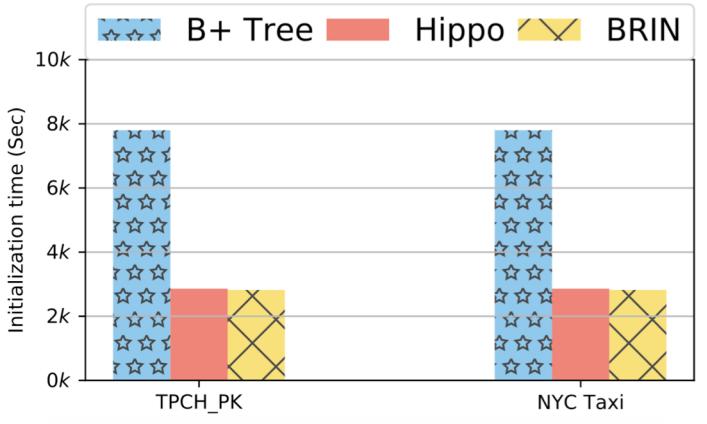


Figure: Initial. time on different datasets

Arizona State University

Experiments (cont.)

- Query time
  - Hippo ≈ B+ Tree at 0.1% and 1% selectivity
  - BRIN always scans lots of disk pages

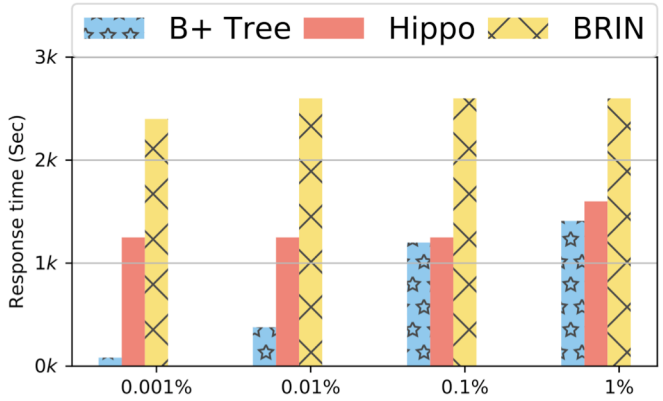


Figure: Query time on TPC-H

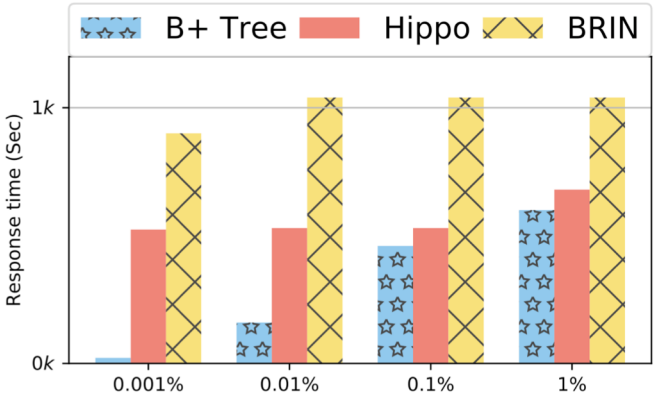


Figure: Query time on NYC Trips

Arizona State University

Experiments (cont.)

- Hybrid workload
  - Update-intensive workloads (10%-50%), Hippo is 100x > B+ Tree
  - Query-intensive workloads (70%-90%), Hippo ≈ B+ Tree

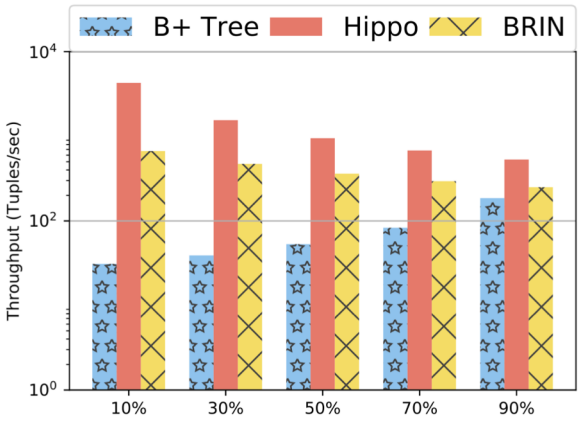


Figure: Throughput on TPC-H

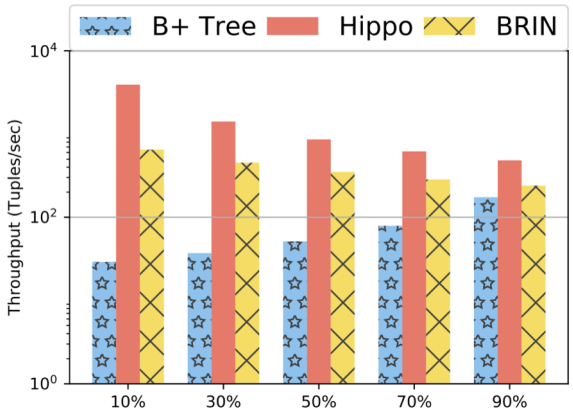


Figure: Throughput on NYC Trips

Arizona State University

Take-home Lesson

- Use Hippo if you have limited storage budget
- Use Hippo if your query selectivity is 0.1% - 1%
- Use Hippo if you have update-intensive workloads

Questions?

9.6.1 kernel

https://github.com/DataSystemsLab/hippo-postgresql

Build Hippo

CREATE INDEX hippo_idx ON hippo_tbl USING **hippo** (randomNumber) WITH (density = 20);

Arizona State University

Experiments (cont.)

- Maintenance time
  - Hippo 10x – 1000x < B+ Tree
  - Hippo 10x < BRIN. BRIN doesn't support deletion.

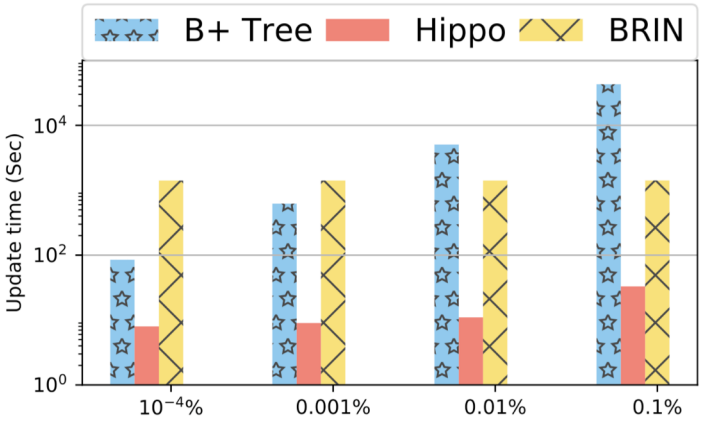


Figure: Update time on TPC-H (logarithmic scale)

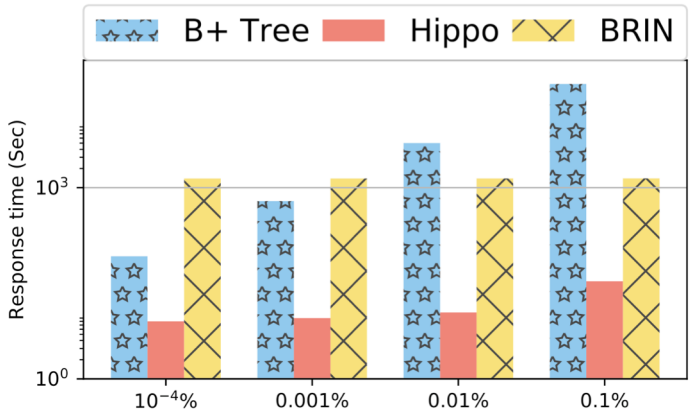


Figure: Update time on NYC Trips (logarithmic scale)

Arizona State University