

# UPI: A Primary Index for Uncertain Databases

Presented by **Hideaki Kimura**  
Brown University

With **Sam Madden**  
MIT CSAIL

**Stan Zdonik**  
Brown University

# Introduction: Uncertain Databases

## Author<sup>p</sup>

Name	Institution <sup>p</sup>	Exist?
Alice	Brown: 80%, MIT: 20%	90%
Bob	MIT: 95%, UCB: 5%	100%
Carol	Brown: 60%, U. Tokyo: 40%	80%

### Possible World 1

### Possible World 2

...

Name	Inst.
Alice	Brown
Bob	MIT

Name	Inst.
Bob	UCB

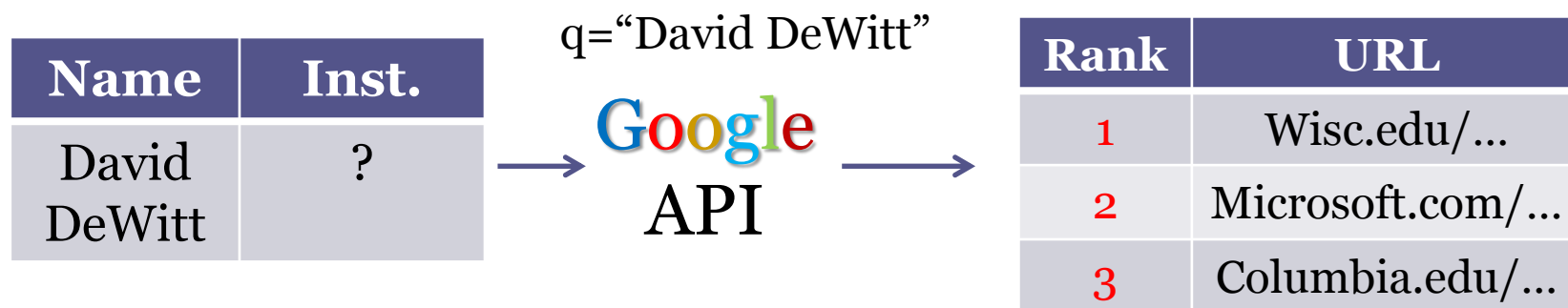
$$.9 * .8 * .95 * .2 = \underline{13\%}$$

$$.1 * .05 * .2 = \underline{0.1\%}$$

Querying  
over Possible Worlds

# Ex) DBLP with Uncertain Affiliation

- DBLP: 1.3M Papers and 0.7M Authors
- Complemented Author Affiliation



Zipfian Distribution

Name	Institution <sup>p</sup>	Country <sup>p</sup>
David DeWitt	Wisconsin: 40%, MS: 20%, Columbia: 13%, ...	US: 100%

# Indexes for Uncertain Database

- Special Secondary Indexes  
Ex) PII (Probabilistic Inverted Index)

Institution <sup>p</sup>	<u>Pointer</u>
Brown	[Alice], [Carol]
MIT	[Bob], [Alice]

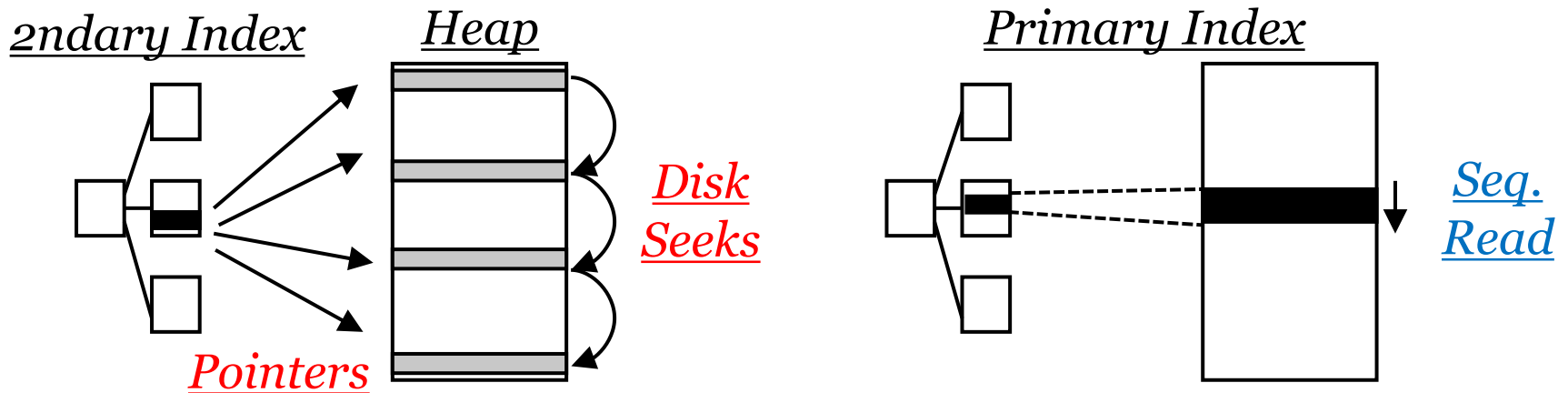
- But, None Has Explored Primary Index

Goal

Build Primary Indexes  
Over Uncertain Attributes

# Why Primary Index Matters?

- Much Faster than Secondary Index



- Benefits Correlated Secondary Indexes

- Faster Secondary Indexes
- Smaller Secondary Indexes

} For more details:  
**CORADD**  
Thursday 2p  
At Falcon

# Problem: Attribute Uncertainty

Name	Institution <sup>p</sup>	Exist?
Alice	Brown: 80%, MIT: 20%	90%
Bob	MIT: 95%, UCB: 5%	100%
Carol	Brown: 60%, U. Tokyo: 40%	80%

Cluster on most probable possible value?

Cluster	Tuples
Brown	<b>Alice, Carol</b>
MIT	<b>Bob</b>

Alice?



SELECT WHERE Inst.=MIT

Replicate tuples into inverted index?

Cluster	Tuples
Brown	<b>Alice, Carol, ...</b>
MIT	<b>Alice, Bob, ...</b>

**Too Large** for Long-tail distribution  
(e.g., 100 values with 0.1%)

# Overview

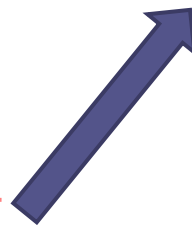
- UPI
  - Cutoff Index
  - Choosing Threshold
- Secondary Index on UPI
- Fractured UPI
- (Continuous UPI)
- Experiments

# UPI: Heap and Cutoff Index

Name	Institution <sup>p</sup>	Exist?
Alice	Brown: 80%, MIT: 20%	90%
Bob	MIT: 95%, UCB: 5%	100%
Carol	Brown: 60%, U. Tokyo: 40%	80%

Heap: Sorted by (Inst., Prob)

<u>Institution</u>	Tuple
Brown (72%)	<b>Alice</b>
Brown (48%)	<b>Carol</b>
MIT (95%)	<b>Bob</b>
MIT (18%)	<b>Alice</b>
<del>UCB (5%)</del>	<del><b>Bob</b></del>
U. Tokyo (32%)	<b>Carol</b>



Cutoff Index:

Sorted by (Inst., Prob)

<u>Institution</u>	#ID	Pointer
UCB (5%)	[Bob]	MIT

**Cutoff** Entries with  
Less than C probability  
(*Cutoff Threshold*)

# Answering Queries with UPI

- Probabilistic Threshold Query (PTQ)

```
SELECT * FROM Author WHERE Inst.=UCB  
With: Probability  $\geq QT$  (Query Threshold)
```

<u>Institution</u>	Tuple
MIT (95%)	<b>Bob</b>
...	
UCB (90%)	<b>Dan</b>
UCB (20%)	<b>Emily</b>

*Seek*

$C=10\%$

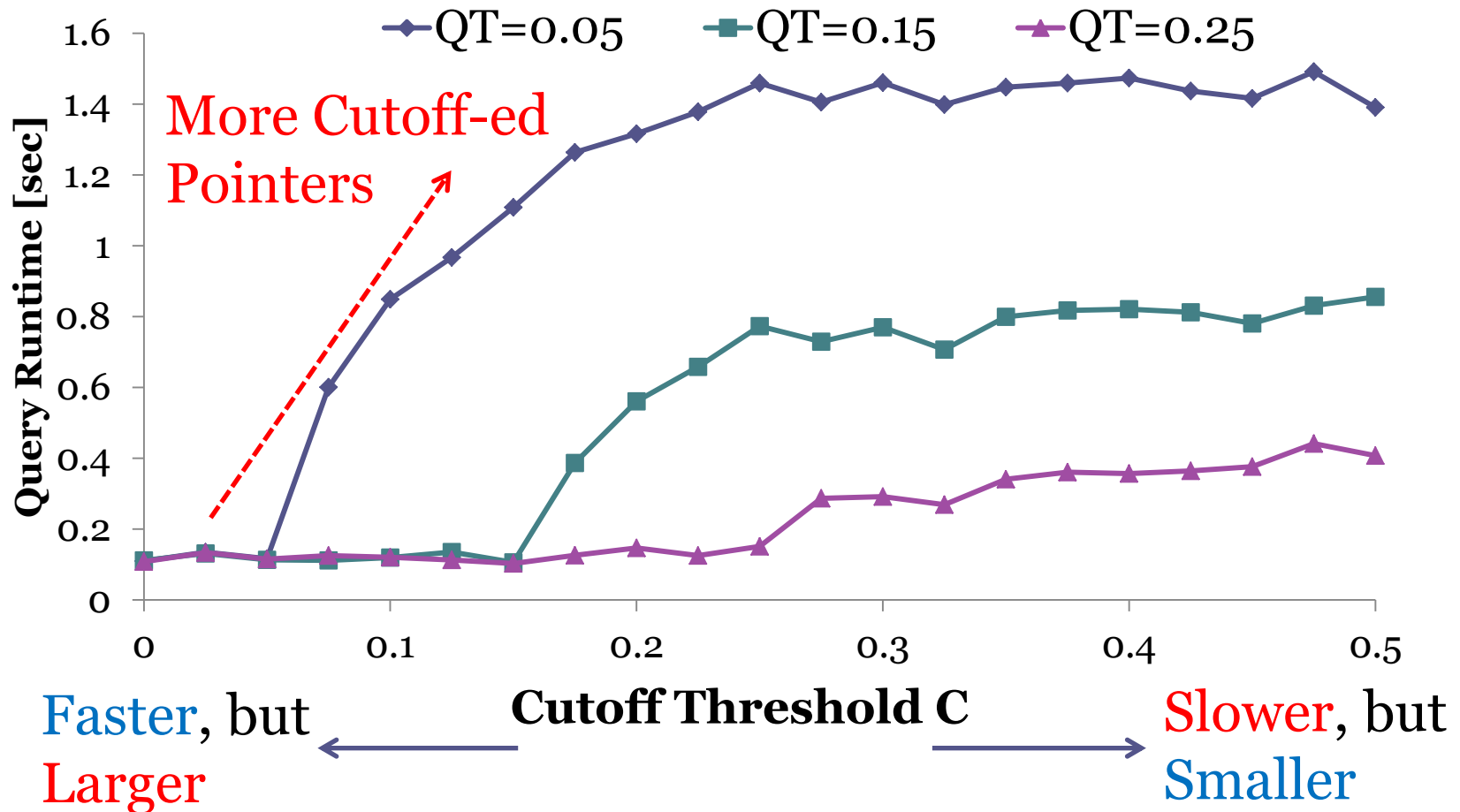
<u>Institution</u>	#ID	Pointer
UCB (5%)	[Bob]	<b>MIT</b>

*If  $QT < C$  (e.g.,  $QT=5\%$ ),  
follow Cutoff-ed pointers*

*If  $QT \geq C$  (e.g.,  $QT=20\%$ ),  
Sequentially Read*

# Choosing Cutoff Threshold

Selective Case (#Pointers=300)



# Analytic Cost Model

- Used to Determine 'C'
- Based on Value/Probability Histograms

Histograms (Inst.)

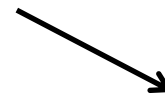
Value	#Keys	Prob.	#
Br-Bs	30,000	10%-15%	
Bs-Bt	31,000	15%-25%	
Bt-Bz	30,500	25%-40%	

*C*



UPI Size

*C, Query*



#Cutoff-ed  
Pointers

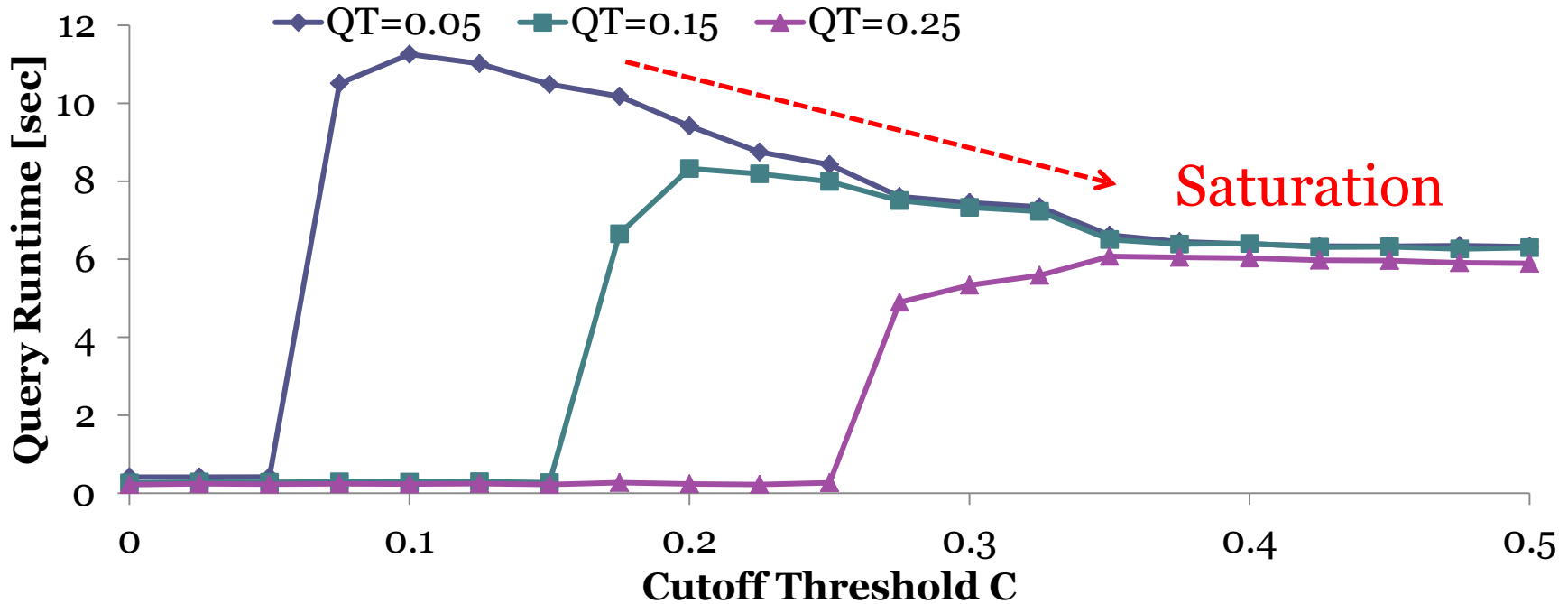


Query Runtime

= #Pointers \* SeekCost?

# #Pointers and Query Cost

Non-Selective Case (#Pointers=37000)



Cost Model

$$Cost_{cutoff} \equiv Cost_{fullscan} \cdot Selectivity + f(\#Pointers)$$

$$f(x) \equiv Cost_{fullscan} \left( \frac{1 - e^{-kx}}{1 + e^{-kx}} \right)$$

Logistic function

# Secondary Index on UPI

Name	Institution <sup>p</sup>	Country <sup>p</sup>	Exist?
Alice	Brown: 80%, MIT: 20%	US: 100%	90%
Bob	MIT: 95%, UCB: 5%	US: 100%	100%
Carol	Brown: 60%, U. Tokyo: 40%	US: 60%, Japan: 40%	80%

<u>UPI</u>	
Brown (72%)	<b>Alice</b>
...	
MIT (95%)	<b>Bob</b>
MIT (18%)	<b>Alice</b>

Tailored Index Access

SELECT ... WHERE Country=US

Secondary Index on (Country)

<u>Country</u>	Pointers
US (100%)	[MIT]
US (90%)	[Brown], [MIT]

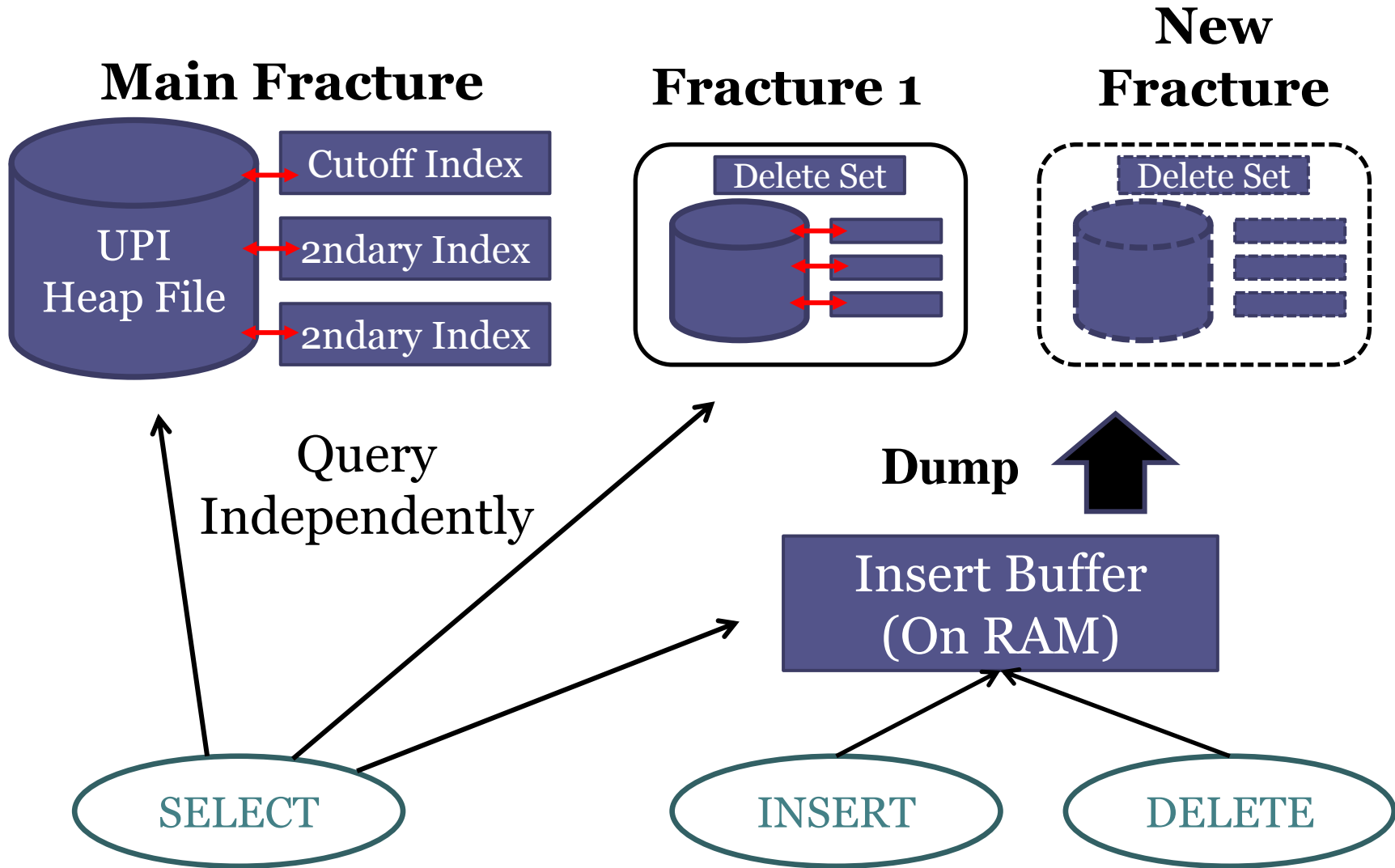
- Store Multiple Pointers
- Determine which to use at runtime

Alice is also in MIT...

# Maintaining Primary Index

- Random Read/Write
  - Disk Seeks for each INSERT
  - **Much Slower than Unclustered Heap**
- Fragmentation
  - B+Tree nodes Split/Merge
  - Internal/External Fragmentation
  - **Makes SELECTs Slower over time**

# Fractured UPI

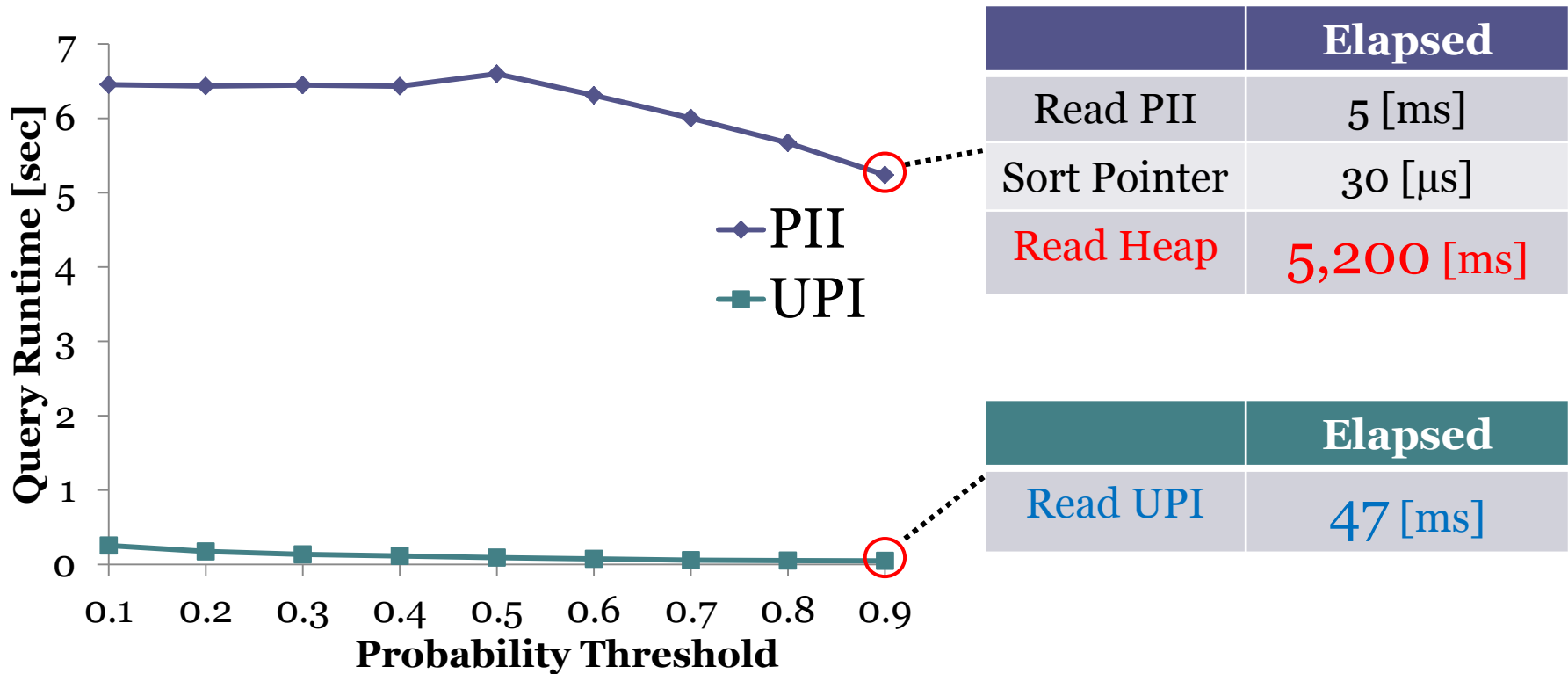


# Experiments

- **Environments**
  - C++ & BerkeleyDB 4.7 on Fedora Core 11
  - Quad-Core, 4GB RAM, 10k RPM SATA HDD
- **Dataset: DBLP w/ Uncertain Affiliation**
- **Compared With PII** (and U-Tree in paper)
- **Cold Start**

# Query Runtime: PII vs. UPI

Q1: SELECT \* FROM Author WHERE Institution=x

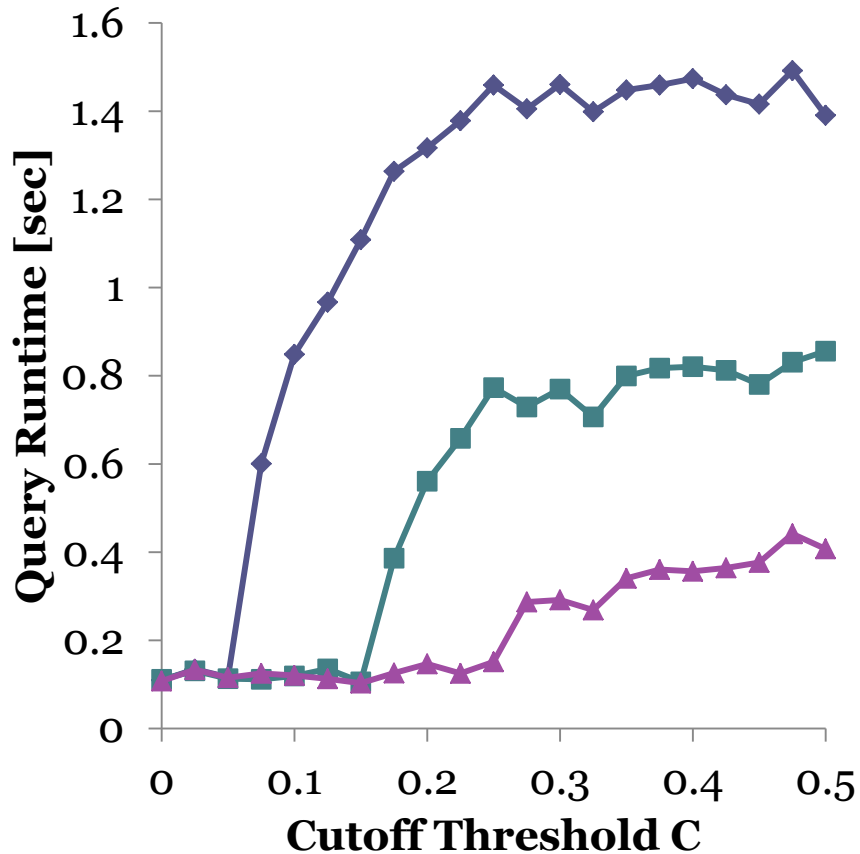


UPI Causes Much Fewer Disk Seeks

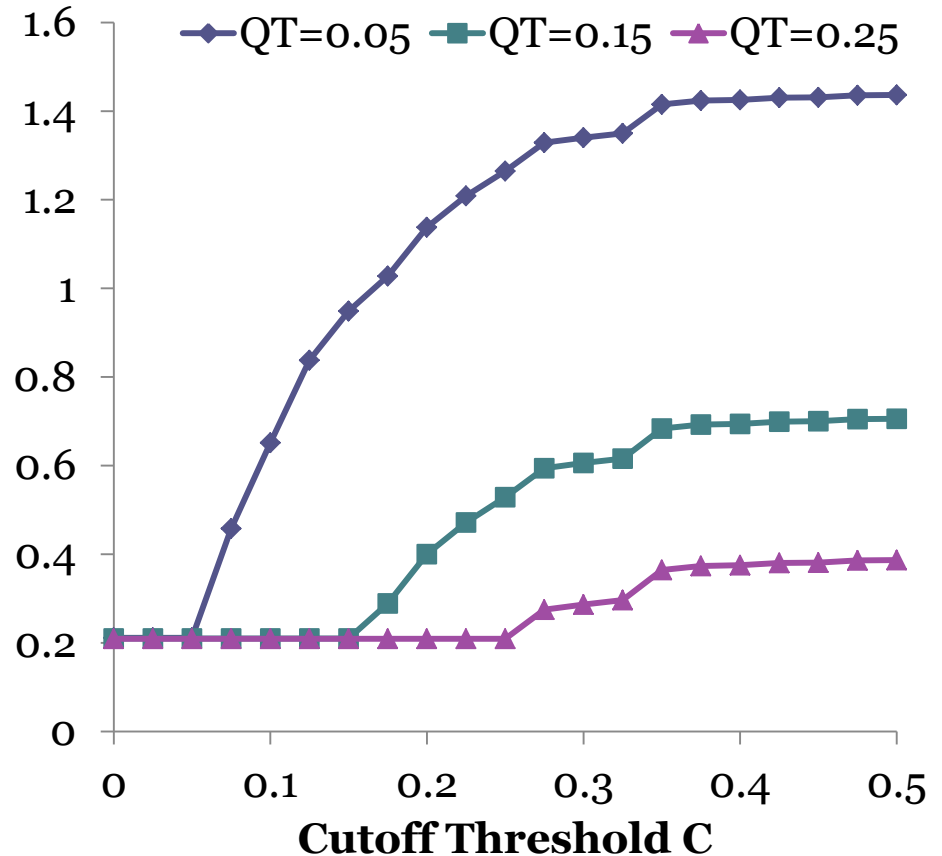
# Cutoff Index Cost Model (1)

Selective Case (Q1, #Pointers=300)

Real Runtime



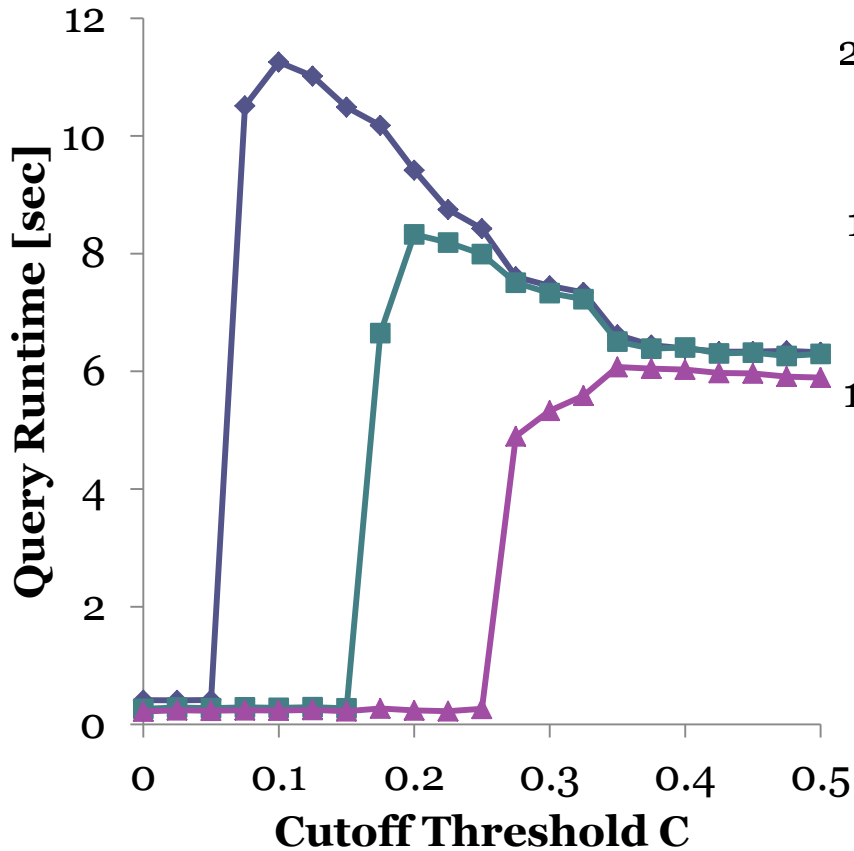
Estimated Runtime



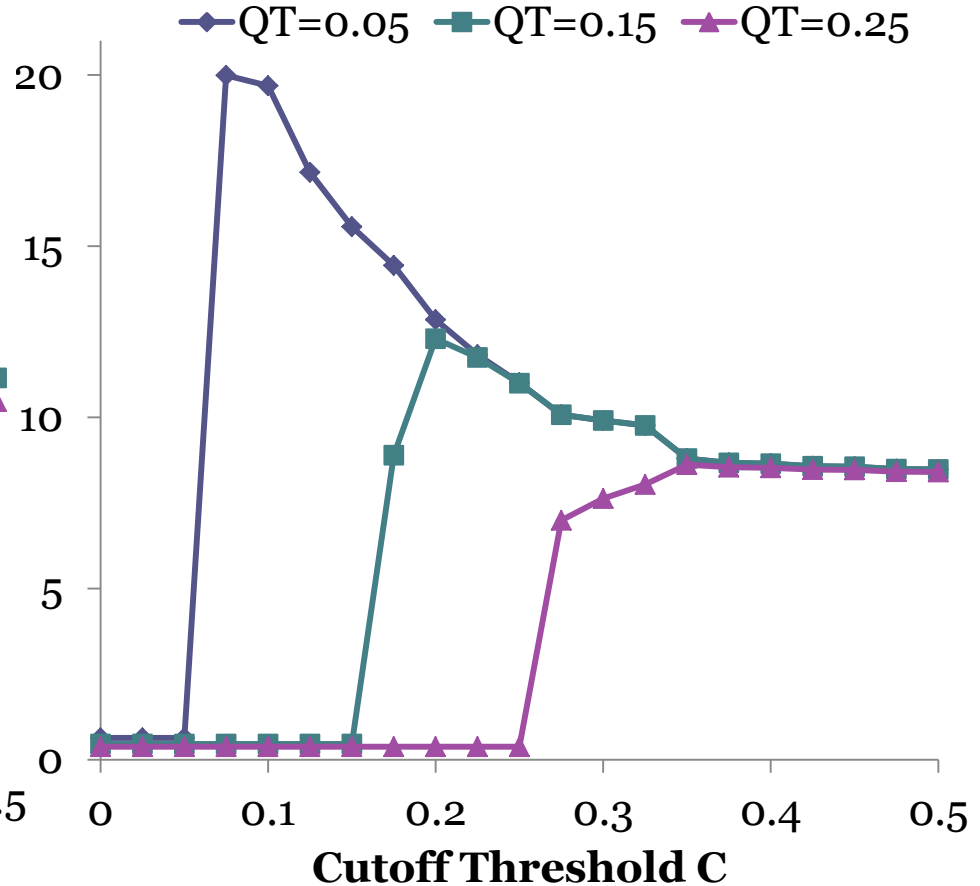
# Cutoff Index Cost Model (2)

Non-Selective Case (Q1, #Pointers=37000)

Real Runtime

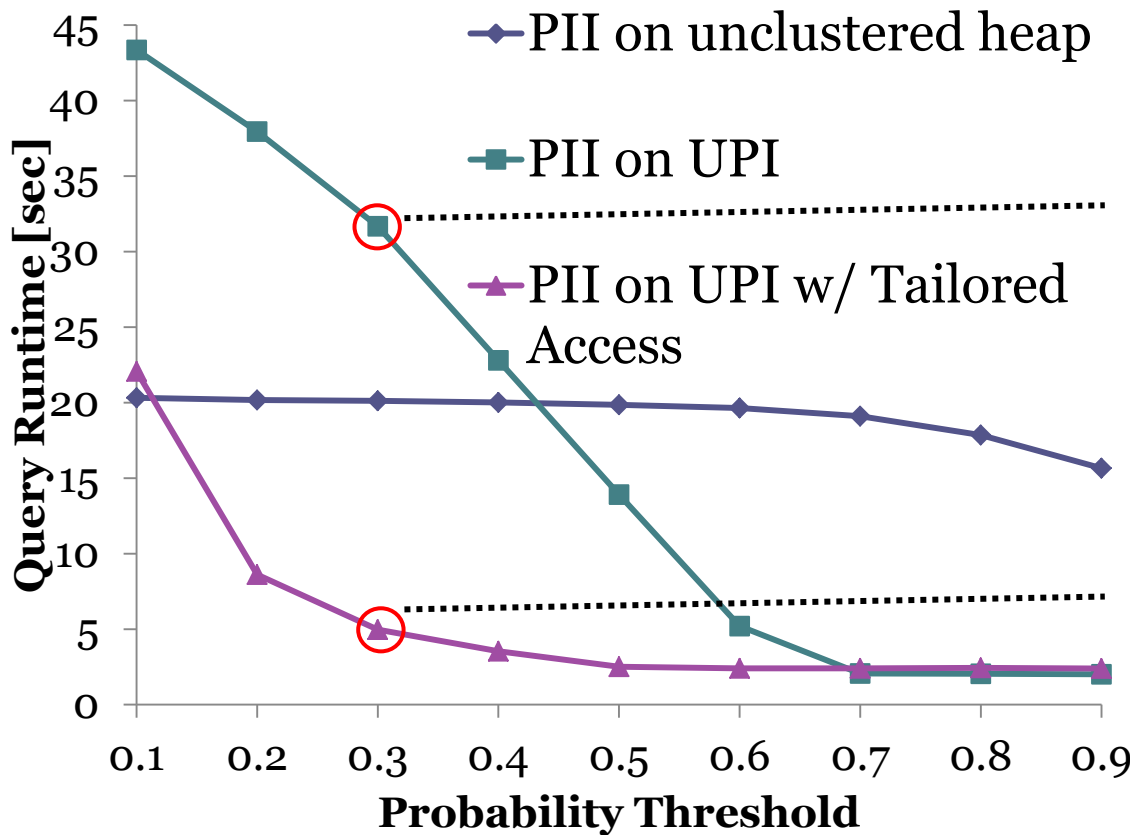


Estimated Runtime



# Tailored Secondary Index Access

Q2: SELECT Journal, COUNT (\*) FROM Publication WHERE Country=x GROUP BY Journal

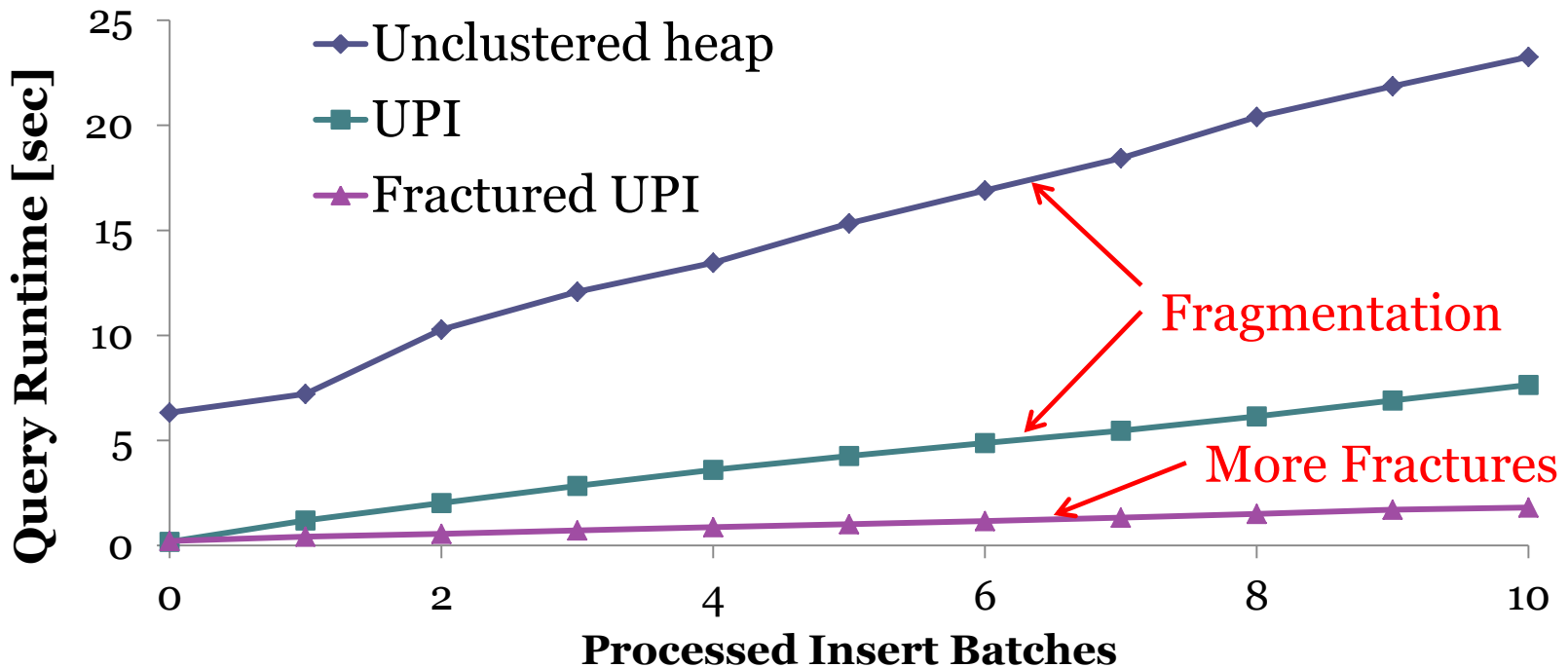


	Elapsed
Read PII	110 [ms]
Read UPI	<b>3,200 [ms]</b>

	Elapsed
Read PII	110 [ms]
Tailor	33 [ms]
Read UPI	<b>500 [ms]</b>

# Fractured UPI

	Insert 10%	Delete 1%
Unclustered Heap	8 sec	75 sec
UPI	650 sec	212 sec
Fractured UPI	4 sec	0.03 sec



# Conclusion

- **Primary Index for Uncertain DB**
  - Essential for Non-Selective Queries
  - Benefits Correlated Indexes
- **UPI**
  - Cutoff Index and Cutoff Threshold
  - Tailored Secondary Index Access
  - Continuous UPI
  - Fractured UPI

# Future Work

- Applying to other types of queries
  - Top-k Query: UPI as Tuple Access Layer
- Database Designer for Uncertain DB
  - Index/MV Recommendation
  - Correlation Analysis in Uncertain DB
  - Incremental Design