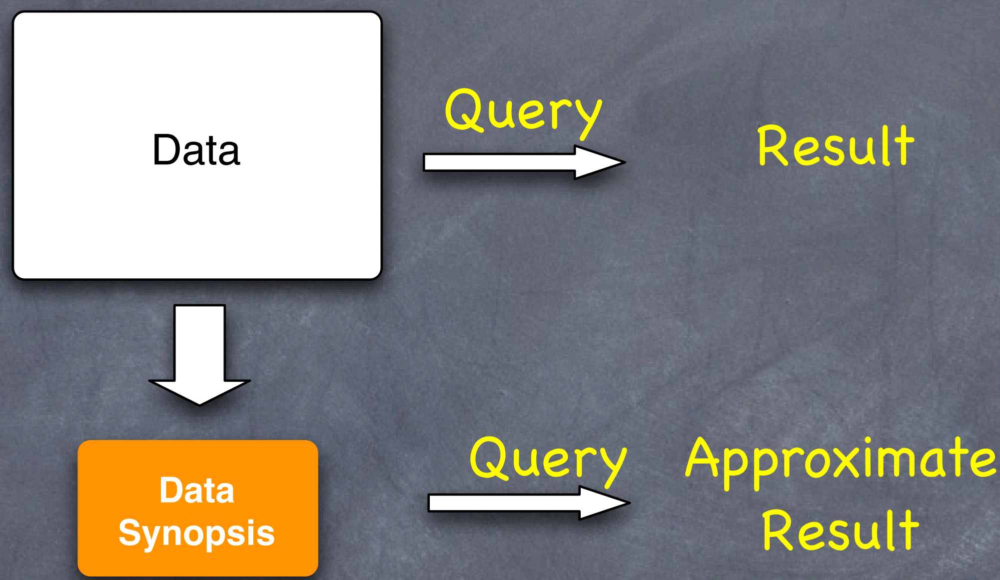
Graph-Based Synopses for Relational Data

Alkis Polyzotis (UC Santa Cruz)

Data Synopses



- Problem: exact answer may be too costly to compute
  - Examples: massive data set exploration, selectivity estimation
- Solution: run query on a synopsis and return an approximate answer
  - Synopsis: lossy summary of data instance

Data Synopses and Query Optimization

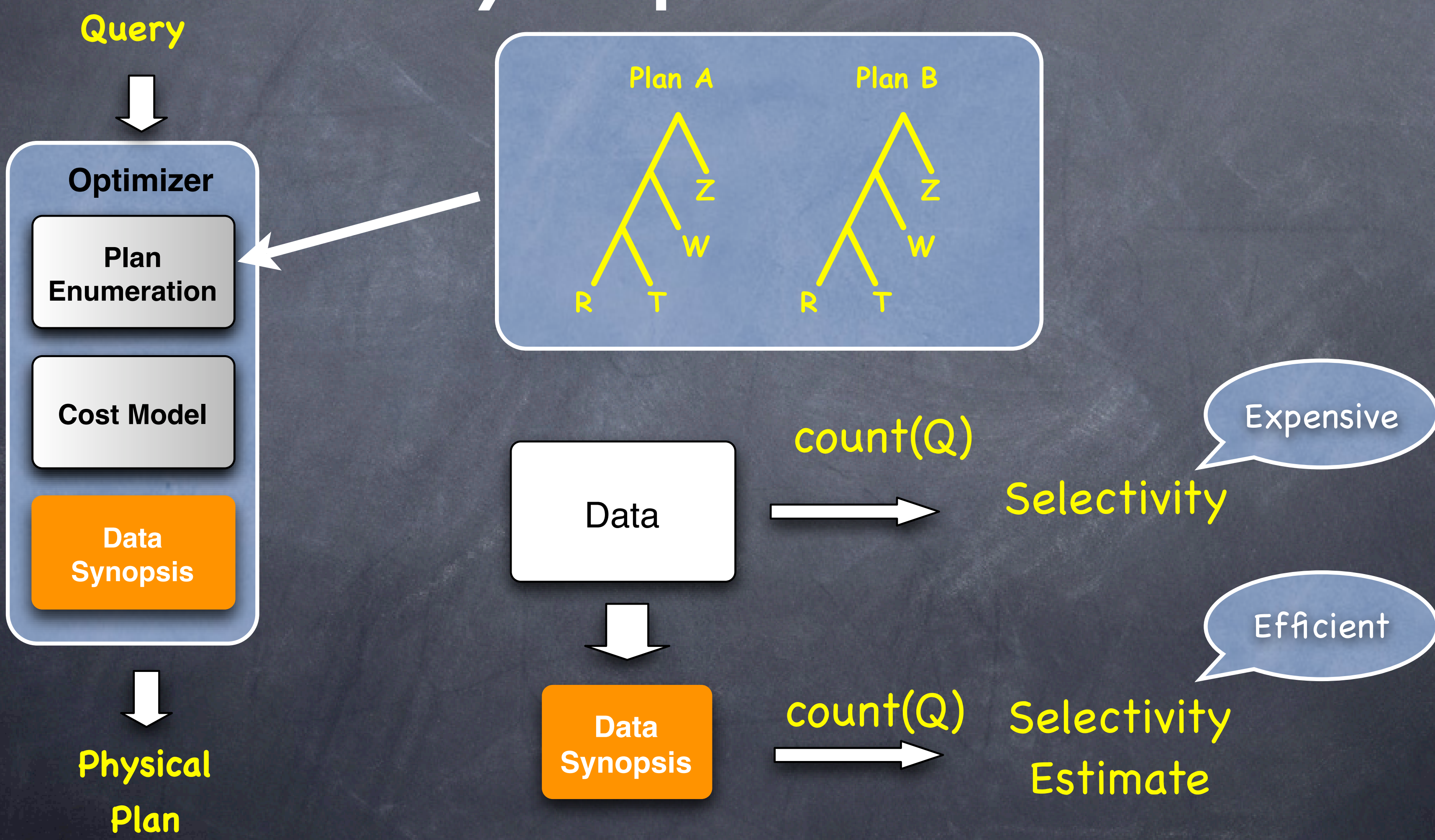
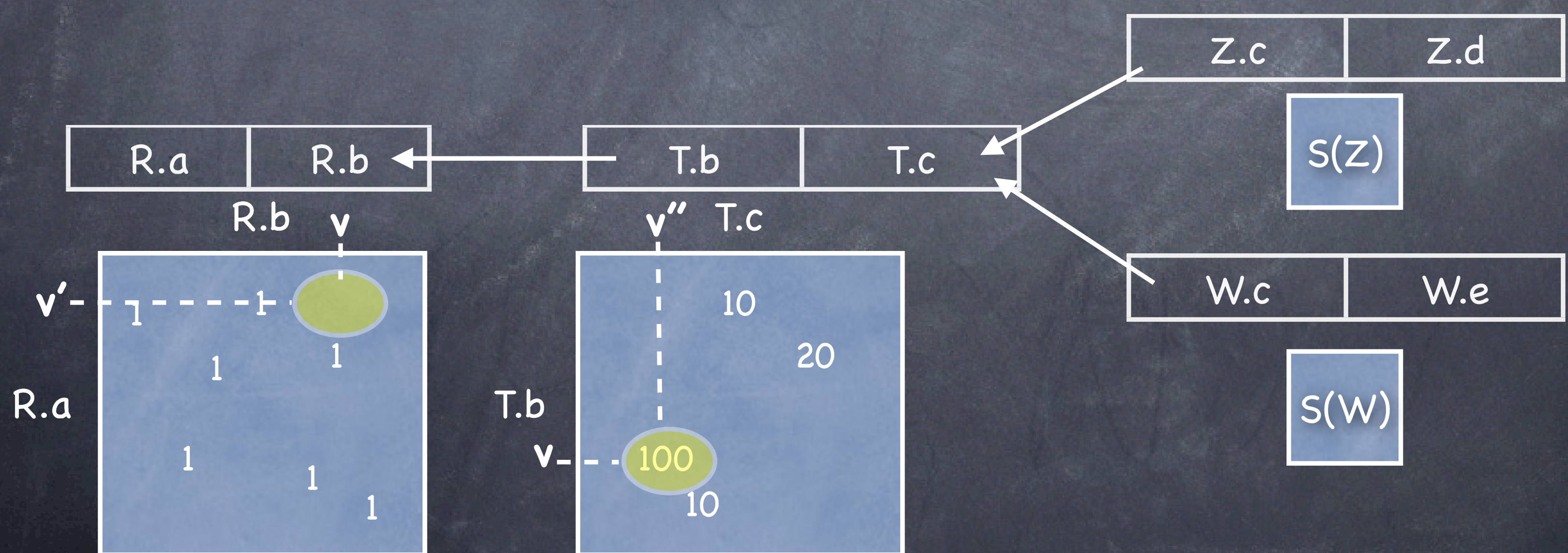


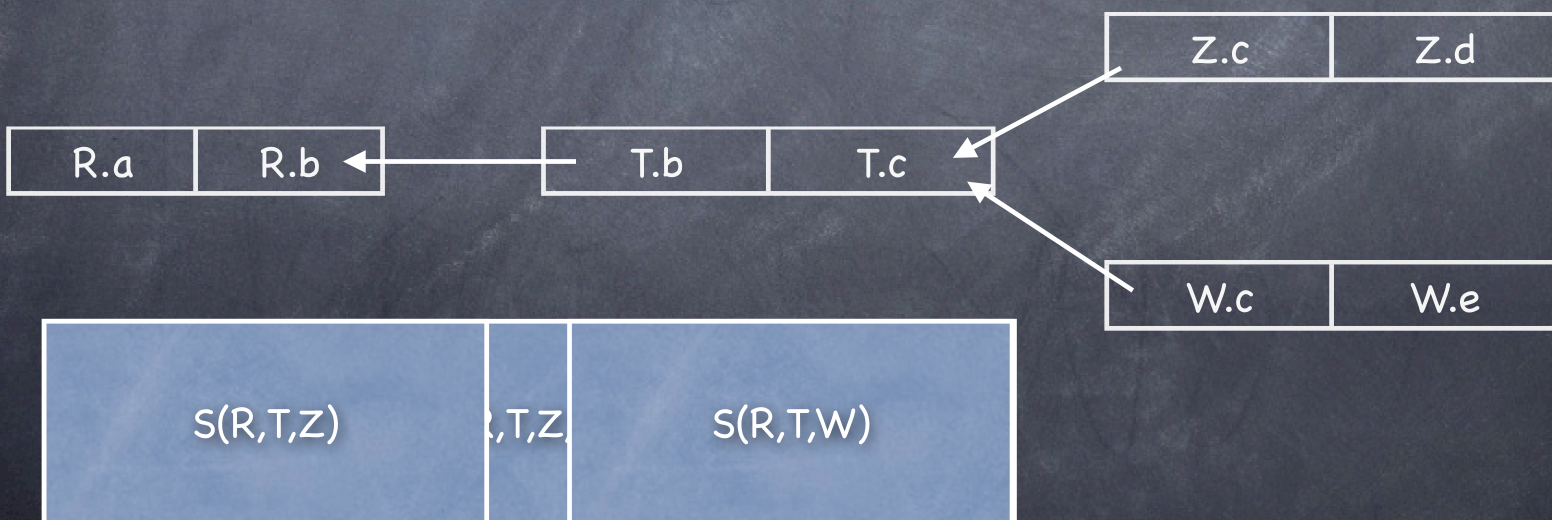
Table-Level Synopses

- Examples: histograms, wavelets, table samples, sketches
- One synopsis per table
  - The synopsis summarizes the frequency matrix
- Problem: ineffective for key/foreign-key joins



Schema-Level Synopses

- Examples: Join Synopses, Prob. Rel. Models
- One synopsis for the whole schema
- Problem: restricted to specific schemata
  - Many-to-many joins cannot be handled



Desiderata

- Schema-level synopsis
- Applicable to general schemata and queries
  - Many-to-many joins
  - Join graphs with cycles
- Affordable to construct

Intuition #1

- Relational database <=> Semi-structured data graph

Movie

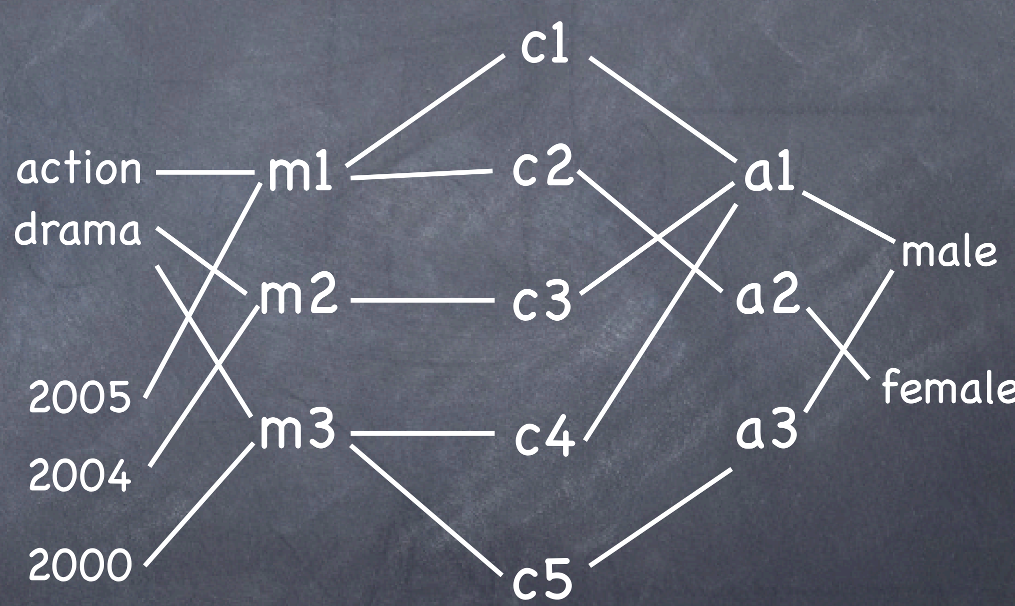
mid	year	genre
1	2005	"action"
2	2004	"drama"
3	2000	"drama"

Actors

aid	sex
1	male
2	female
3	male

Cast

aid	mid
1	1
2	1
1	2
1	3
3	3



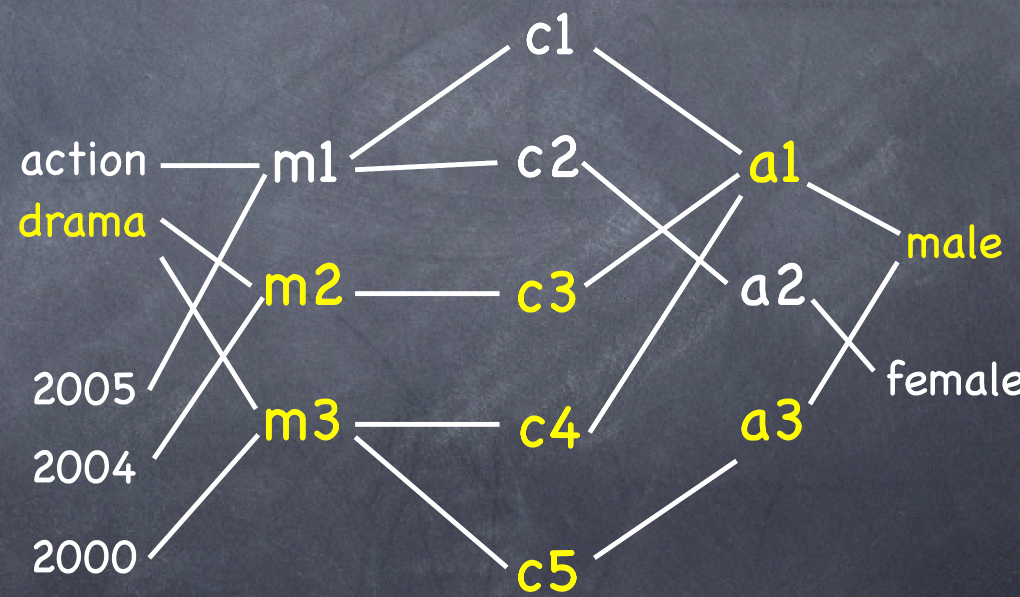
Intuition #2

- Join query <=> Sub-graph matching
- Selectivity <=> Count of matching sub-graphs

```
SELECT *
FROM M, C, A
WHERE M.mid=C.mid
AND C.aid=A.aid
AND a.sex=male AND
a.genre=drama
```

M — C — A

drama male



Tuple Graph Synopses (TuGs)

- Graph-based summaries for relational data
  - Key idea: summarize structure of data graph
  - Schema-level synopses
  - Support for a large class of schemata
- Joint work with Josh Spiegel (UCSC)
- Sponsors: NSF (CAREER Award), IBM (Faculty Development Award)

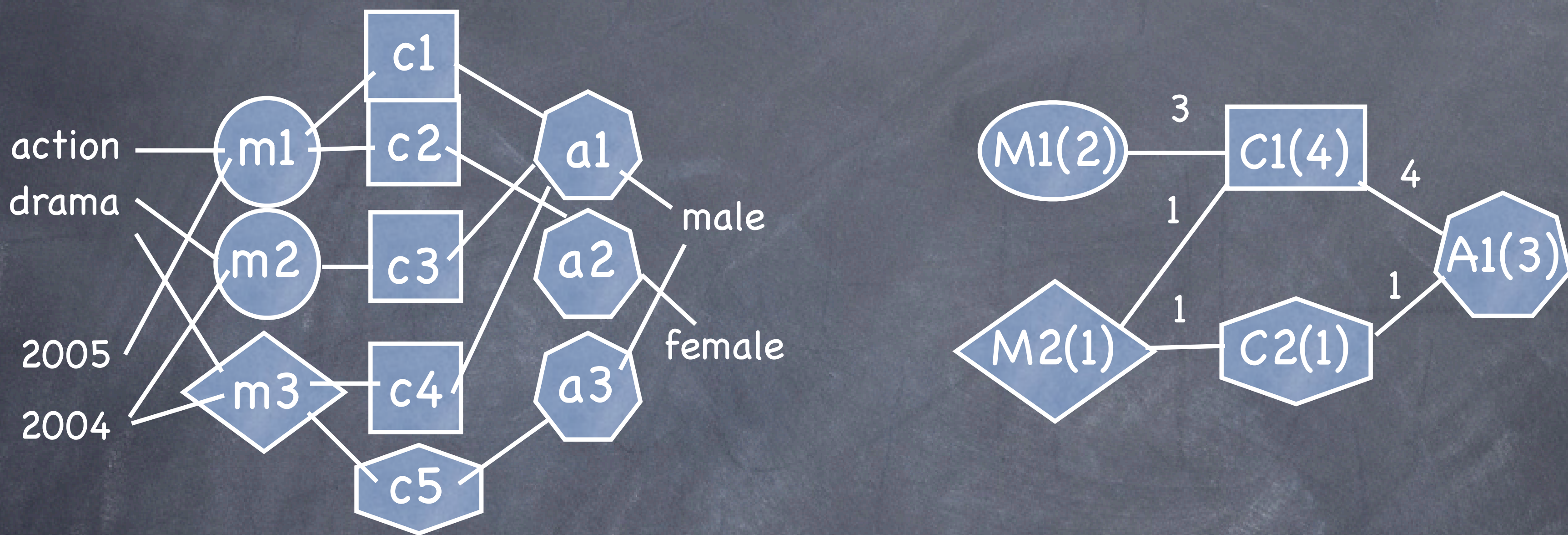
TuGs and XML

- Why not use an existing XML technique?
  - Relational data graph resembles XML data
  - Relational queries resemble twig queries
- The summarization problem is inherently different
  - Relational data graph vs. XML tree
  - Relational queries are fully specified (no // or *)
  - Relational queries are undirected
- Opportunities for an alternative approach!

Outline

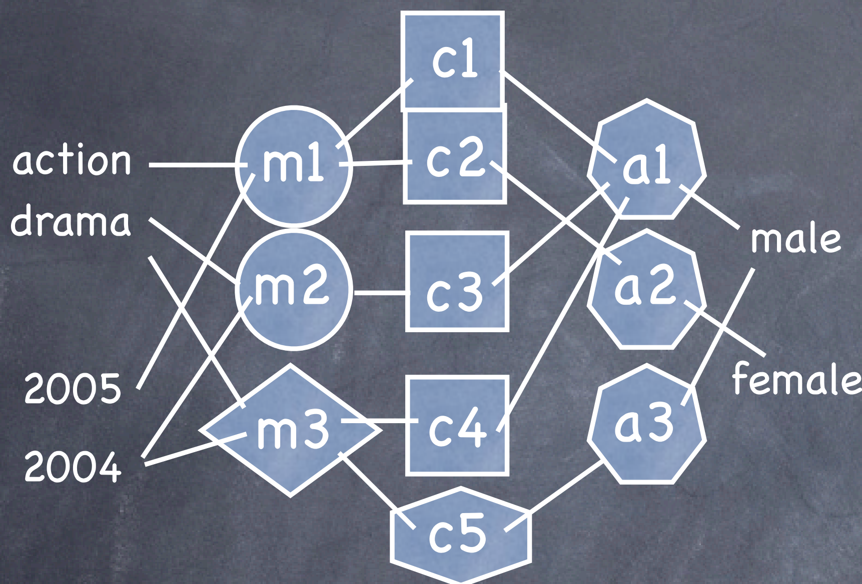
- TuG Synopses
  - Synopsis Model
  - Estimation Framework
- TuG Construction
- Experimental Study
- Conclusions

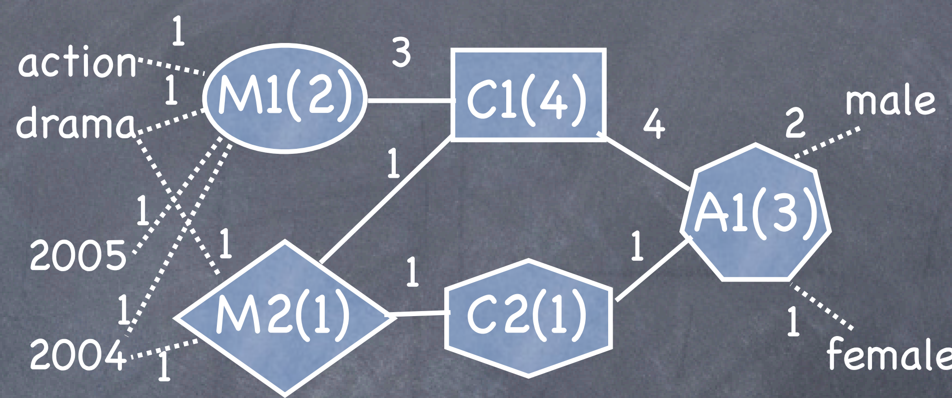
TuG Synopsis: Joins



- Node: Set of tuples from same relation
- Edge: Join between tuple-sets

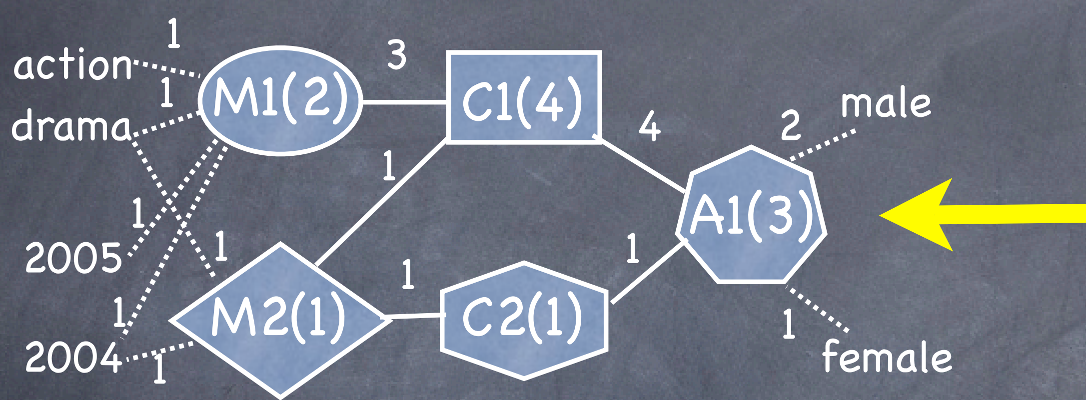
TuG Synopsis: Values





- Values are represented as nodes + edges

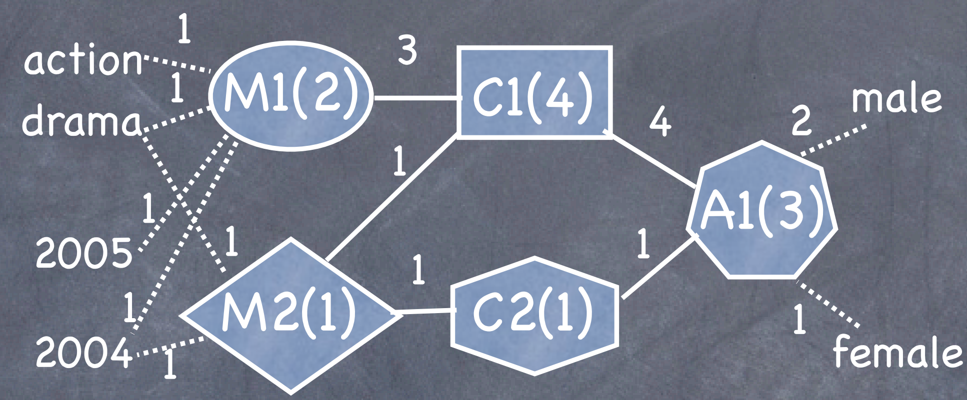
TuG Synopsis Model



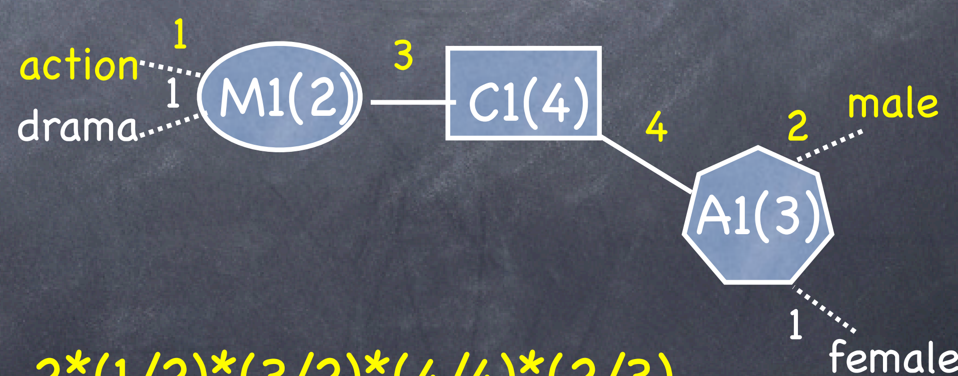
Each actor has:
–4/3 joining tuples in C1
–1/3 joining tuples in C2
–Prob[sex=male]=2/3
–Prob[sex=female]=1/3

- A node aggregates information about its tuples
- Basic assumptions: independence and uniformity
- Correspondence to clustering
  - Each node has a representative "centroid" of ratios
  - Tight clusters <=> Validity of independence

Example TuG Estimation



SELECT *
FROM M, C, A
M —— C —— A
WHERE M.mid=C.mid
AND C.aid=A.aid
action   male
AND A.sex=male
AND M.genre=action



est=2*(1/2)*(3/2)*(4/4)*(2/3)

TuG Estimation Model

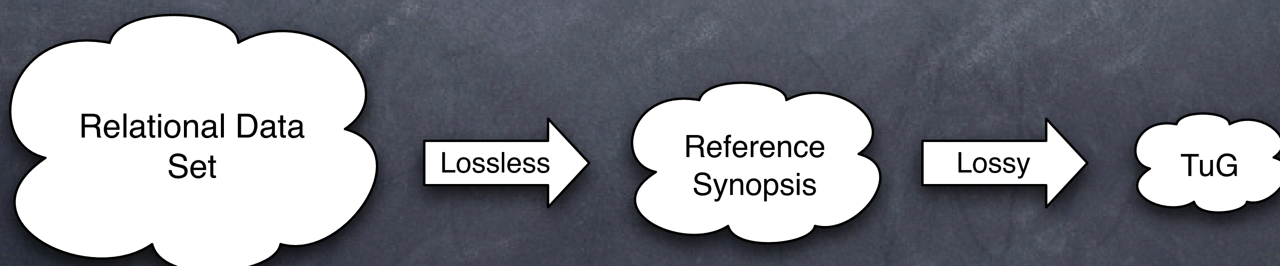
- Two step process:
  1. Identify query embeddings
  2. Estimate selectivity of each embedding
- Estimates are computed based on ratios
  - Closed expression for embedding estimates
  - Methodology extends to queries with cycles
- Estimation uses independence => Accuracy depends on validity of independence
  - Intuition: centroid must be a good representative

Outline

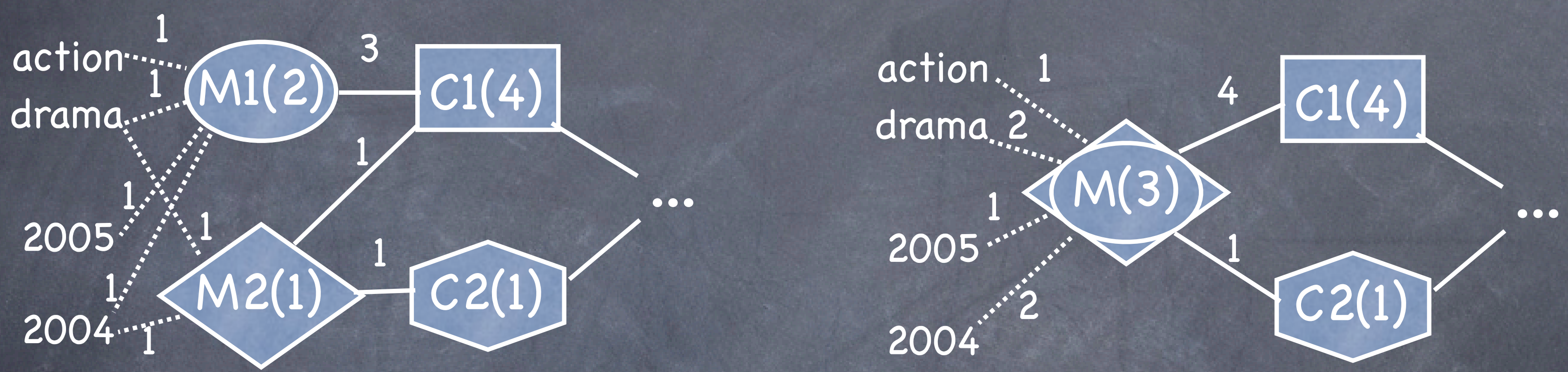
- TuG Synopses
  - Synopsis Model
  - Estimation Framework
- TuG Construction
- Experimental Study
- Conclusions

TuG Construction: Outline

- Problem: Construct an accurate TuG for a specific storage budget
- Outline of construction algorithm:
  - Basic compression operation: node-merge
  - Stage 1: Apply lossless node-merge operations
  - Stage 2: Apply lossy node-merge operations

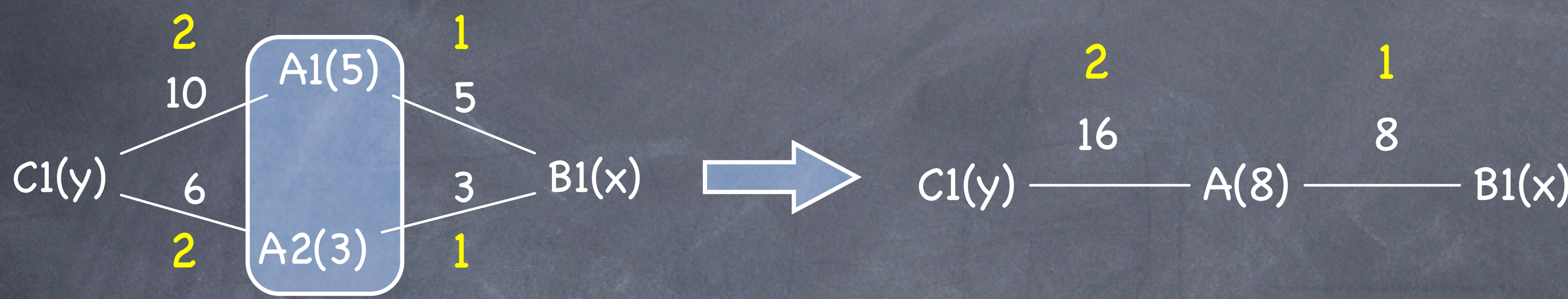


Node-Merge Operation



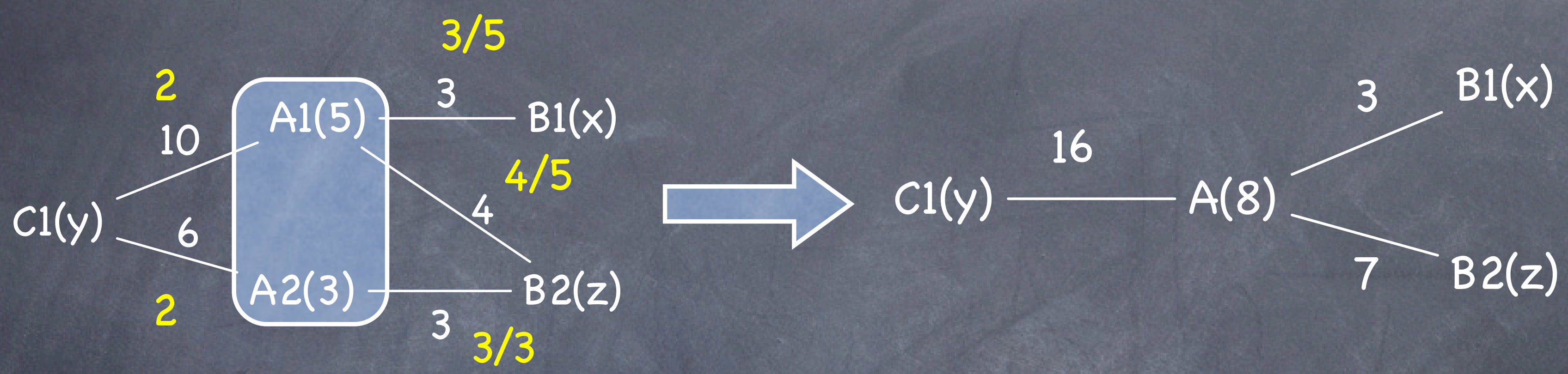
- Collapse a set of nodes to one new node
  - New node acquires aggregated characteristics
  - Similar to merging clusters

Lossless Node-Merge



- Lossless merge => estimates remain unchanged
- Observation: A merge is lossless if the merged centroids are equal
  - Definition used in XML summarization
- TuGs enable a relaxed condition => Opportunity for higher compression

All-but-1 Similarity



- Nodes u and v are ab1-similar <=> Equal join ratios to all schema neighbors except one
  - Fully similar <=> Equal join ratios to all neighbors
- Theorem: if u and v are ab1-similar then their merge is lossless

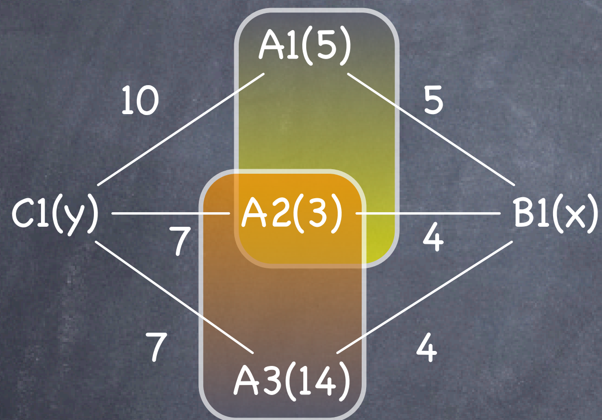
All-but-1 vs Full Similarity

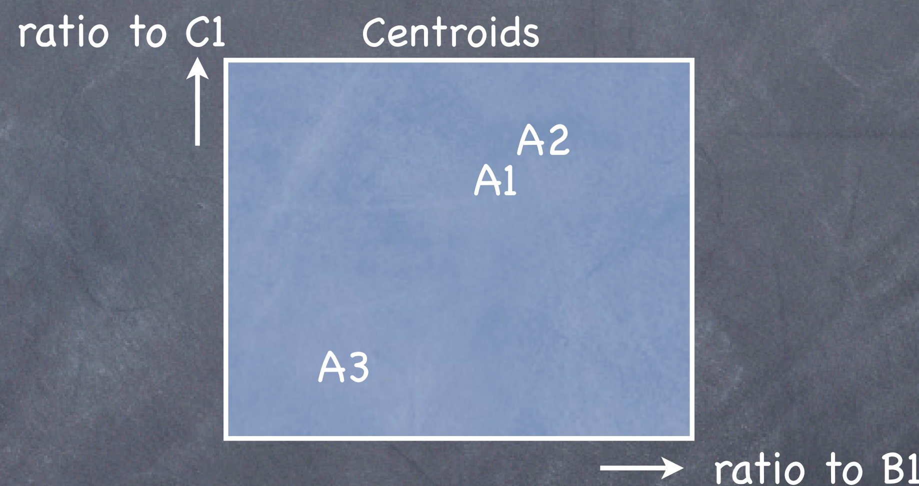
Number of nodes in different synopses

Data Set	Data Graph	Full-Similarity Summary	Ab1-Similarity Summary
TPC-H	8 million	4.4 million	33K
IMDB	4.7 million	4.5 million	65K

Lossy Merges

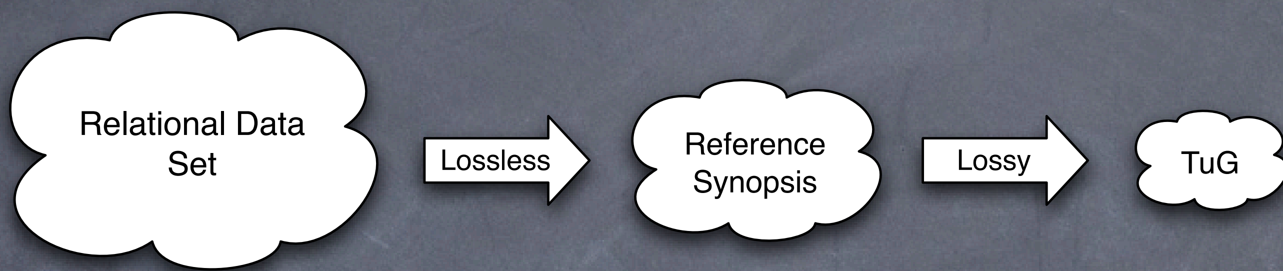
- Question: when is a lossy merge good?





- Intuition: Good merge <=> Similar centroids
- Measure quality through error of clustering
  - Radius, Diameter, Manhattan distance, ...

Construction Algorithm



- Stage 1: Apply lossless node-merge ops on data graph to derive a smaller reference summary
- Stage 2: Compress reference summary with lossy node-merge ops
- Stage 3: Compress value distributions

Construction: Stage 1

- Algorithm sketch:

```
do until no change
    for each (R:table, N: all-but-one neighbors)
        apply lossless node-merge
```

- Order of iteration is based on “clusterability”
  - Intuition: select (R,N) with the most lossless node-merge operations

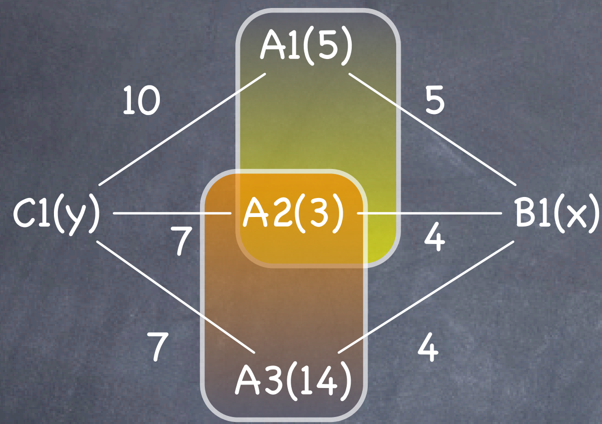
Construction: Stage 2

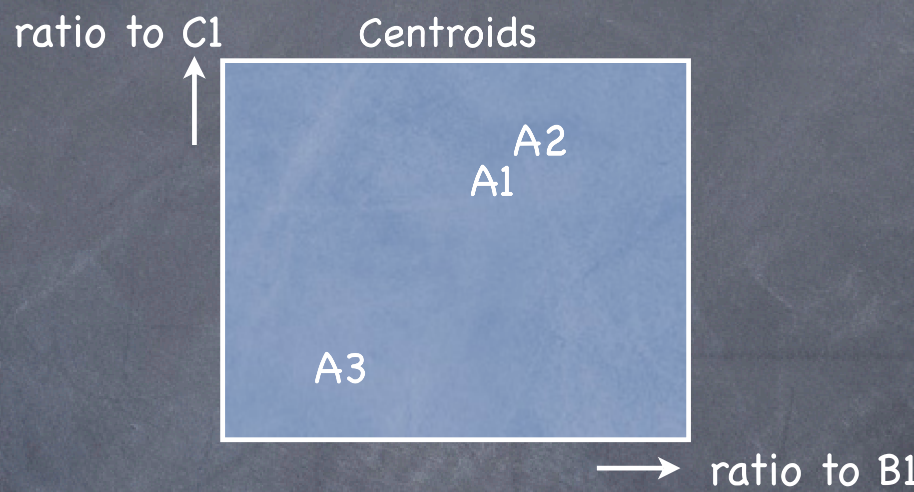
- Algorithm sketch:

```
r := low
while synopsis size > budget
      select R
      apply lossy node-merge on R of radius <= r
      if no such R exists then increase r
```

- r: Threshold of quality
  - Start with good mergers, deteriorate as needed
- Order of processing based on "clusterability"
  - R has high priority if it can be clustered well

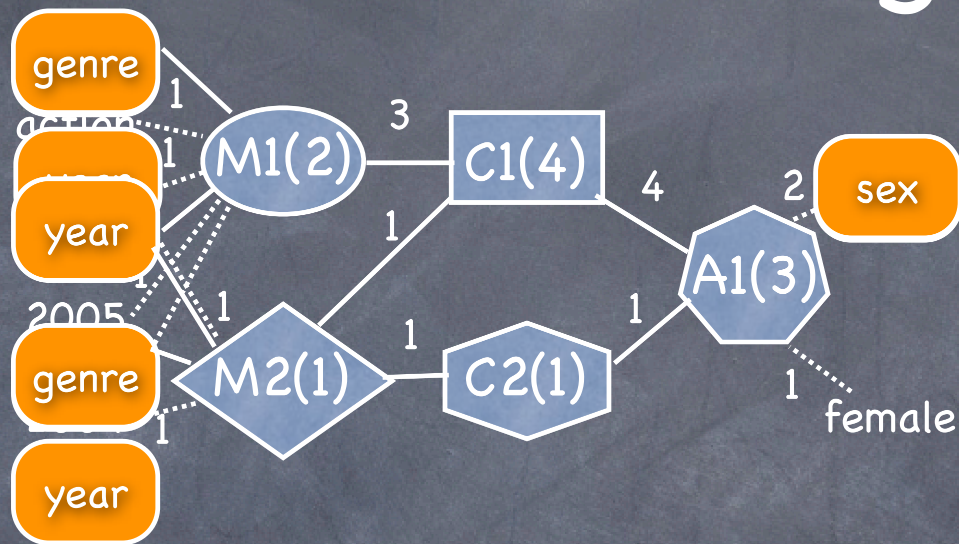
Identifying Merge Operations





- Discover node-mergers through clustering
  - Variable r controls the radius of clusters
- Clustering is computed with variant of BIRCH
  - Use of randomized sketches to approximate distances
- Typically single-pass processing
- Controllable memory overhead

Construction: Stage 3



- Goal: substitute detailed value distributions with compressed value distributions
- Key idea: use a single compressed distribution for multiple nodes

Construction Efficiency

- Processing based on disk-based structures
- Scalable clustering algorithm as the core module
- Result: increased efficiency for large data sets
  => affordable construction times

Outline

- TuG Synopses
  - Synopsis Model
  - Estimation Framework
- TuG Construction
- Experimental Study
- Conclusions

Techniques

- Baseline: 1-d histograms and indexes
  - Existing implementation in commercial system X
  - Size of histograms used as storage budget
- Multi-dimensional wavelets [Chakrabarti+00]
- Join Synopses [Acharya+99]
- TuGs

Data Sets

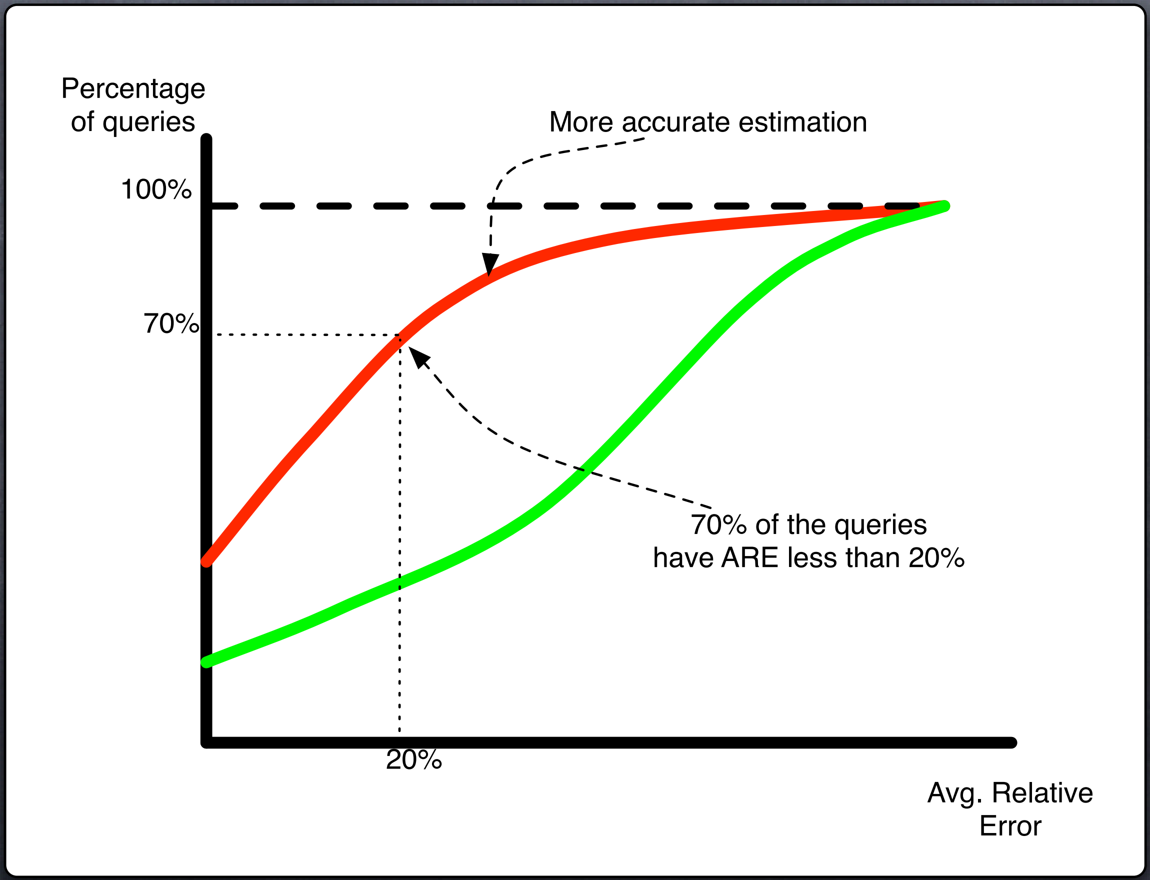
	TPC-H	IMDB
Number of Relations	8	8
#Tuples in largest relation	6 million	2.7 million
#Tuples in smallest relation	5	68K
Size of text files	1 GB	139 MB

Workloads

	TPC-H	IMDB
Avg. result size of positive queries	600K	50K
Number of join predicates	4-8	4-6
Number of selection predicates	1-7	1-5

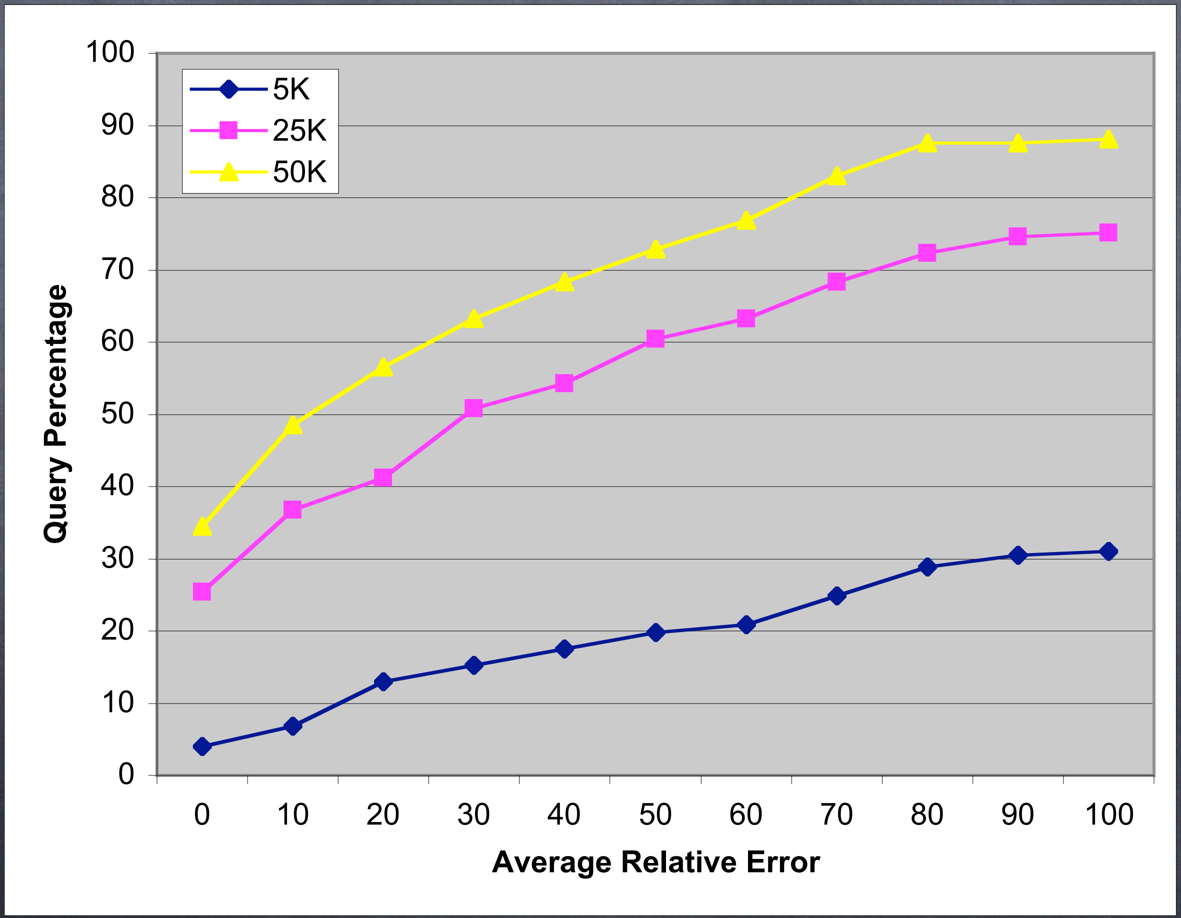
Evaluation Metric

CFD of Average Relative Error



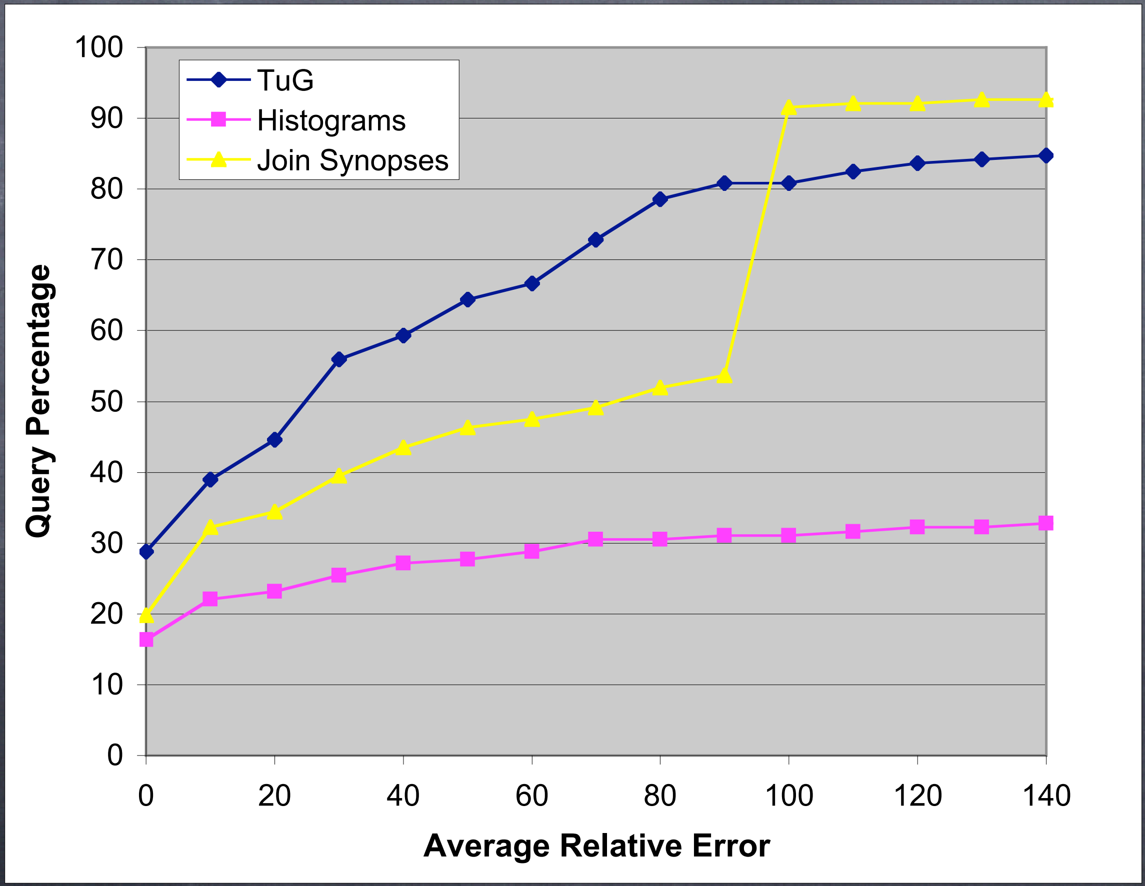
TuG Accuracy vs. Space

TPC-H



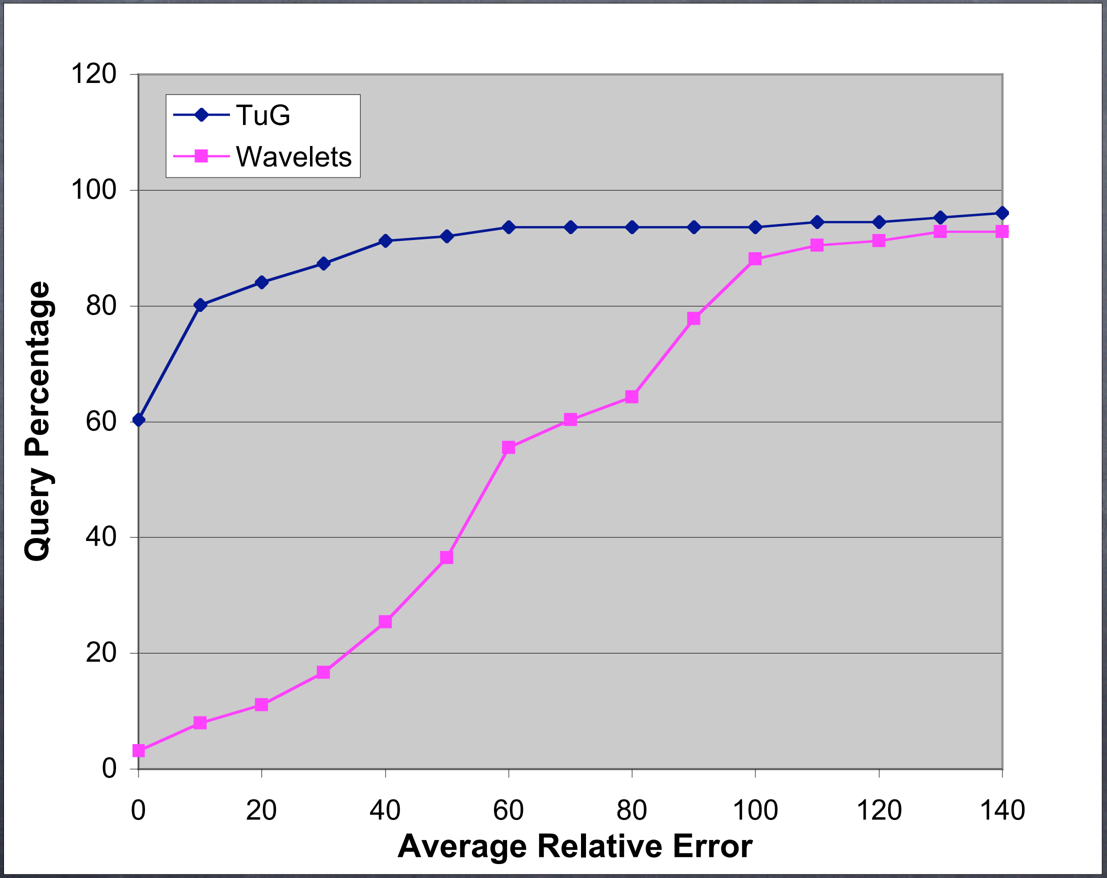
TuG vs. Join Synopses

TPC-H



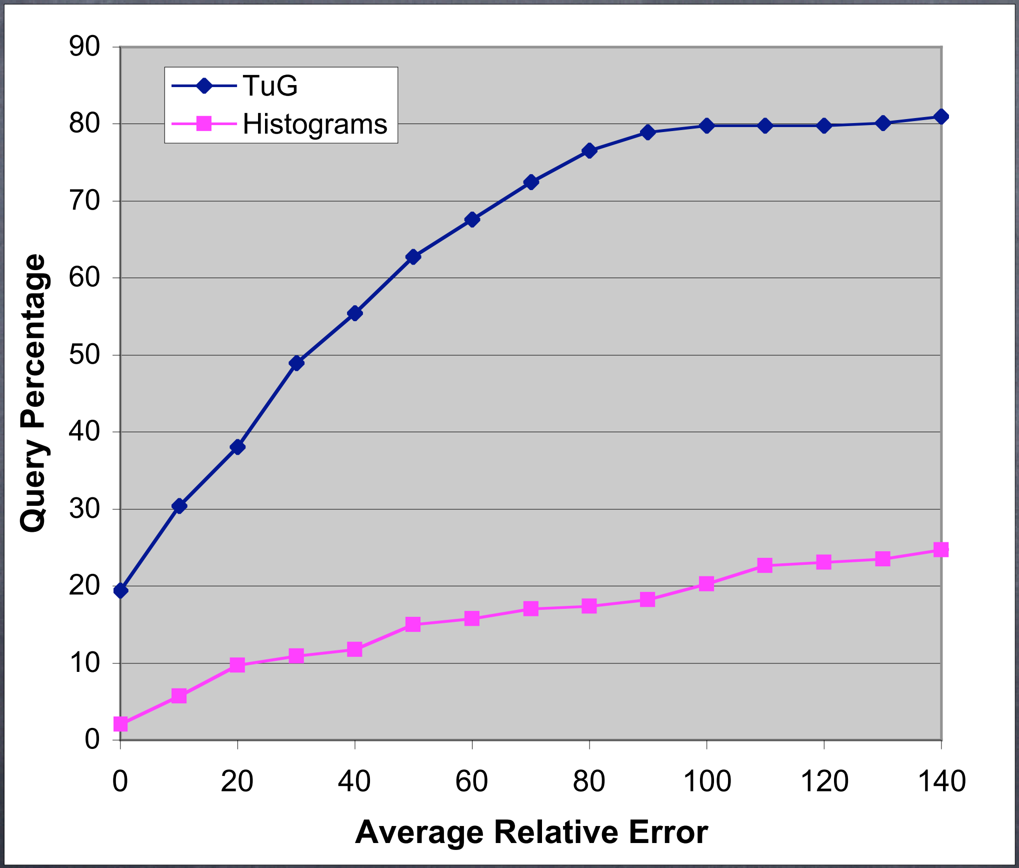
TuG vs. Wavelets

IMDB



TuGs vs. Histograms

IMDB



Conclusions

- Key idea: relational data is semi-structured
- TuG Synopses
  - Schema-level relational summaries
  - Selectivity estimates for complex join queries
  - Support for general schemata
- Experimental results:
  - Accurate selectivity estimates
  - Affordable construction
  - Benefits over existing techniques

Future Work

- Incremental synopsis maintenance
- Guarantees on estimation accuracy
- Transfer to XML domain

Links

- Google: alkis santa cruz
- DB Research at UCSC: http://db.cs.ucsc.edu