TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science, Institute for System Architecture, Database Technology Group

Linked Bernoulli Synopses

Sampling Along Foreign Keys

Rainer Gemulla, Philipp Rösch, Wolfgang Lehner
Technische Universität Dresden

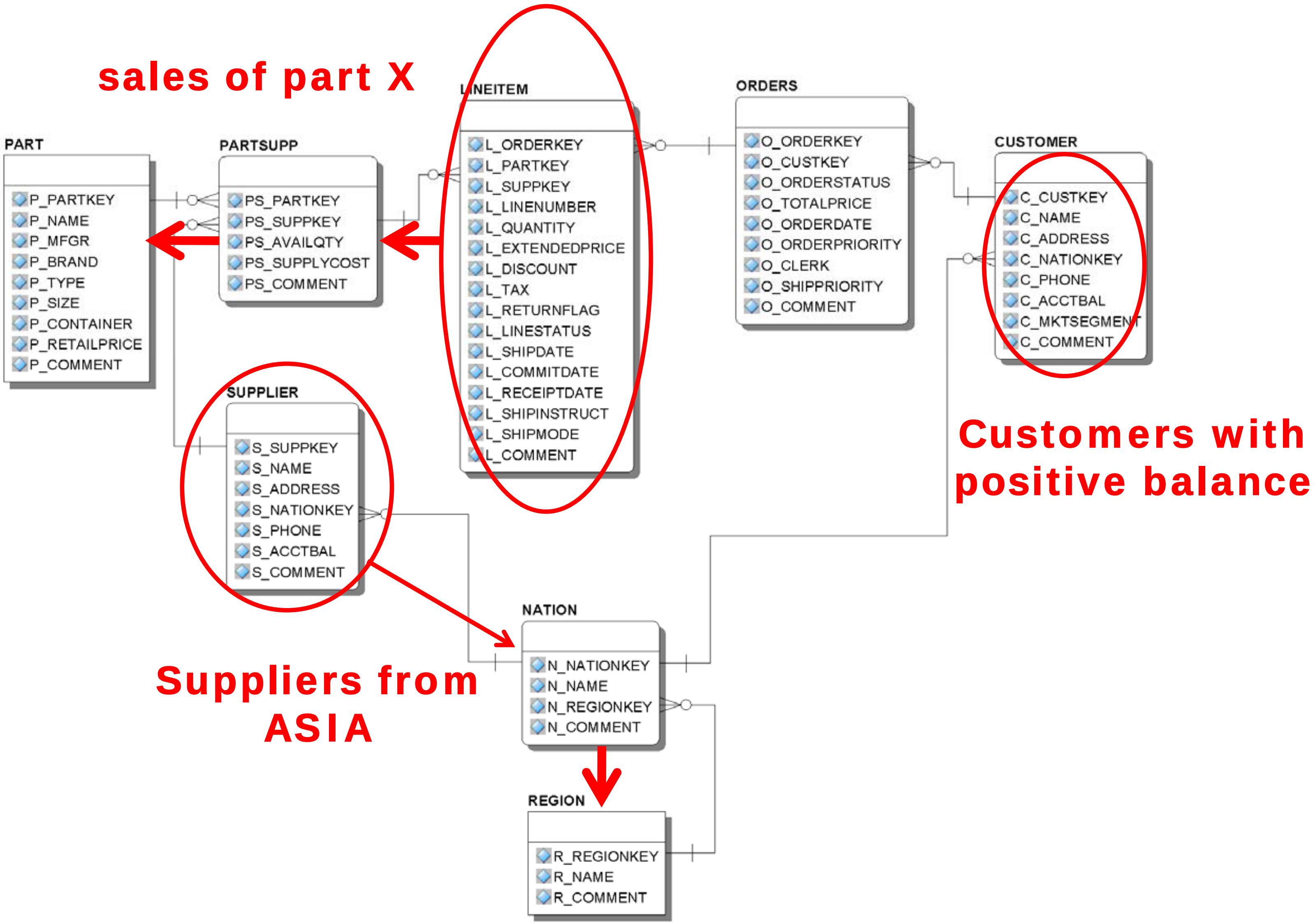
Outline

1. Introduction
2. Linked Bernoulli Synopses
3. Evaluation
4. Conclusion

Motivation

- **Scenario**
  - Schema with many foreign-key related tables
  - Multiple large tables
  - Example: galaxy schema
- **Goal**
  - Random samples of all the tables (schema-level synopsis)
  - Foreign-key integrity within schema-level synopsis
  - Minimal space overhead
- **Application**
  - Approximate query processing with arbitrary foreign-key joins
  - Debugging, tuning, administration tasks
  - Data mart to go (laptop) → offline data analysis
  - Join selectivity estimation

Example: TPC-H Schema

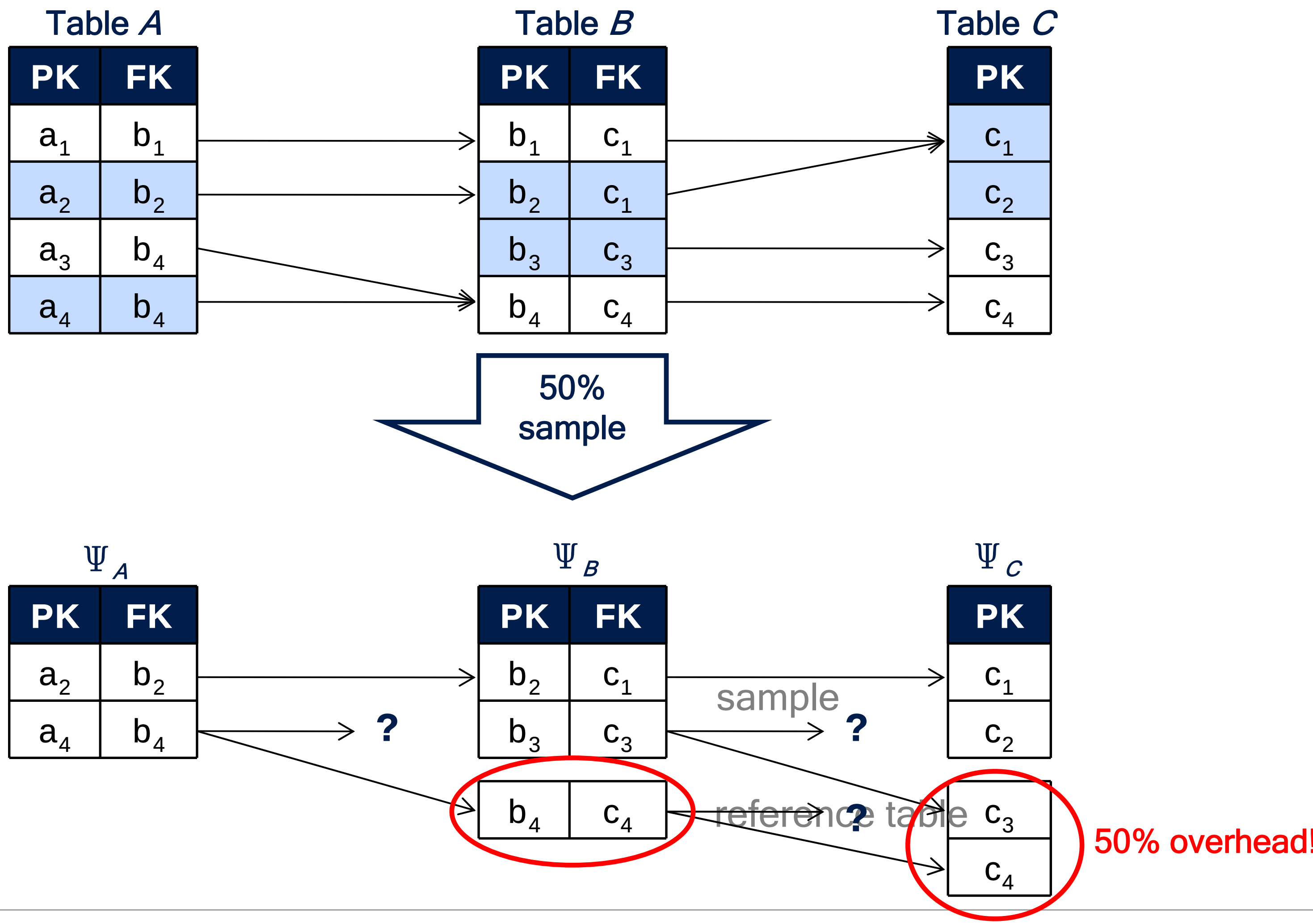


Known Approaches

- **Naïve solutions**
  - Join individual samples → skewed and very small results
  - Sample join result → no uniform samples of individual tables
- **Join Synopses [AGP+99]**
  - Sample each table independently
  - Restore foreign-key integrity using "reference tables"
  - Advantage
    - Supports arbitrary foreign-key joins
  - Disadvantage
    - Reference tables are overhead
    - Can be large

[AGP+99] S. Acharya, P.B. Gibbons, and S. Ramaswamy.
Join Synopses for Approximate Query Answering. In SIGMOD, 1999.

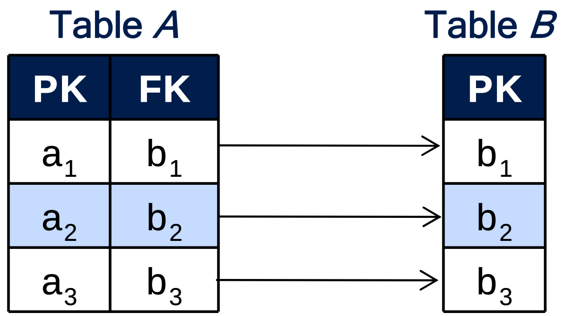
Join Synopses – Example



Outline

Linked Bernoulli Synopses

- **Observation**
  - Set of tuples in sample and reference tables is random
  - ➢ Set of tuples referenced from a predecessor is random



1:1 relationship

- **Key Idea**
  - Don’t sample each table independently
  - Correlate the sampling processes

- **Properties**
  - Uniform random samples of each table
  - Significantly smaller overhead (can be minimized)

Algorithm

- **Process the tables top-down**
  - Predecessors of the current table have been processed already
  - Compute sample and reference table

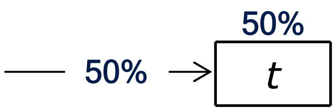
- **For each tuple *t***
  - Determine whether tuple *t* is referenced
  - Determine the probability *pRef(t)* that *t* is referenced
  - Decide whether to
    - Ignore tuple *t*
    - Add tuple *t* to the sample
    - Add tuple *t* to the reference table

"*t* is selected"

Algorithm (2)

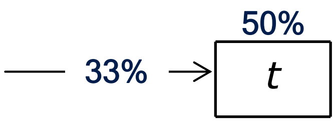
- **Decision: 3 cases**

1. *pRef(t) = q*
   - *t* is referenced: add *t* to sample
   - otherwise: ignore *t*



2. *pRef(t) < q*
   - *t* is referenced: add *t* to sample
   - otherwise: add *t* to sample with probability

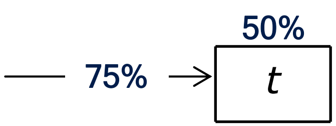
     (q – pRef(t)) / (1 – pRef(t)) *(= 25%)*



3. *pRef(t) > q*
   - *t* is referenced: add *t* to sample with probability

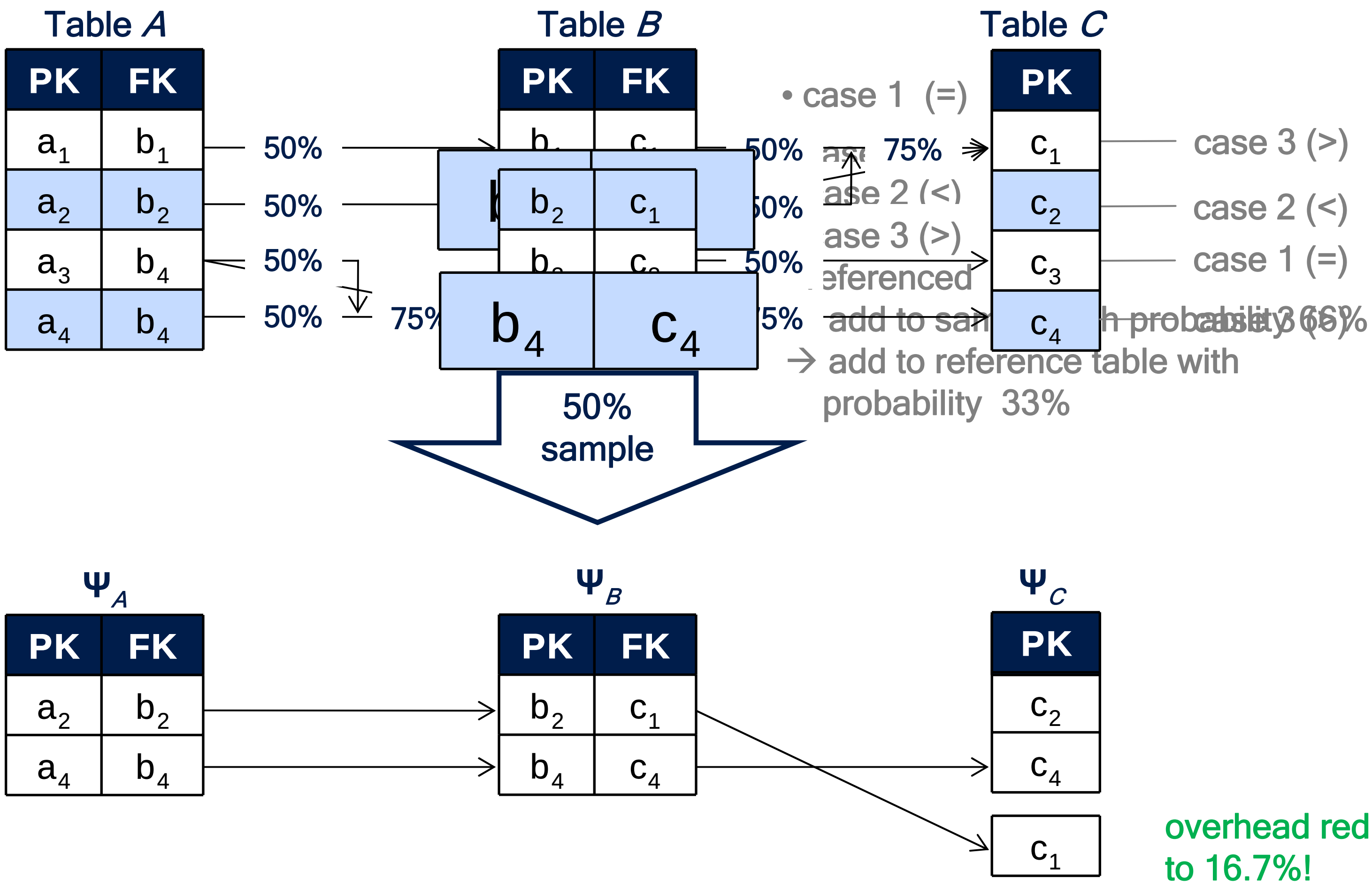
     q/pRef(t) (= 66%)

     or to the reference table otherwise
   - *t* is not referenced: ignore *t*



- **Note:** tuples added to reference table in case 3 only

Example

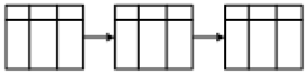


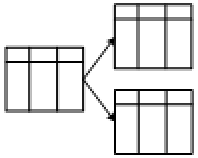
overhead reduced to 16.7%!

Computation of Reference Probabilities

- **General approach**
  - For each tuple, compute the probability that it is selected
  - For each foreign key, compute the probability of being selected
  - Can be done incrementally

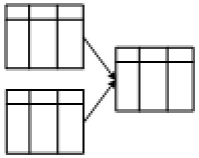
1. **Single predecessor** (previous examples)
   - References from a single table
   - Chain pattern or split pattern



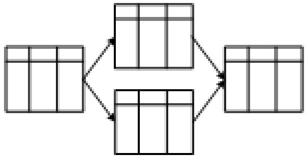


2. **Multiple predecessors**
   - references from multiple tables

   a) Independent references
      - merge pattern



   b) Dependent references
      - diamond pattern



Diamond Pattern

- **Diamond pattern in detail**
  - At least two predecessors of a table share a common predecessor
  - Dependencies between tuples of individual table synopses
  - Problems
    - Dependent reference probabilities
    - Joint inclusion probabilities

Diamond Pattern - Example

Table *A*

PK	FK_B	FK_C
a_1	b_1	c_1
a_2	b_2	c_3
a_3	b_3	c_2

Table *B*

PK	FK
b_1	d_1
b_2	d_2
b_3	d_3

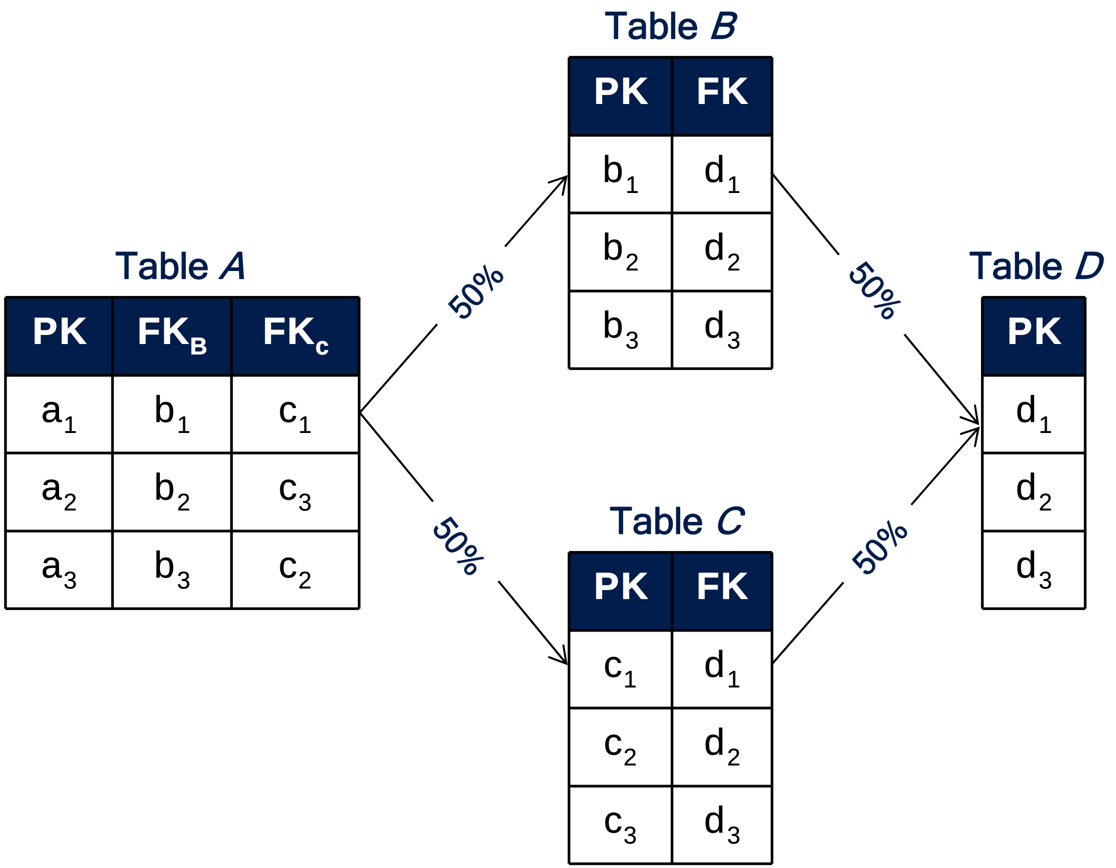
Table *C*

PK	FK
c_1	d_1
c_2	d_2
c_3	d_3

Table *D*

PK
d_1
d_2
d_3

Diamond Pattern – Example



Dep. reference probabilities

– tuple d_1 depends on b_1 and c_1

– Assuming independence: $pRef(d_1) = 75\%$

– b_1 and c_1 are dependent

➢ $pRef(d_1) = 50\%$

Diamond Pattern - Example

Table *B*

PK	FK
b_1	d_1
b_2	d_2
b_3	d_3

Table *A*

PK	FK_B	FK_C
a_1	b_1	c_1
a_2	b_2	c_3
a_3	b_3	c_2

Table *C*

PK	FK
c_1	d_1
c_2	d_2
c_3	d_3

Table *D*

PK
d_1
d_2
d_3

Joint inclusions

- Both references to d_2 are independent
- Both references to d_3 are independent
- But all 4 references are not independent
- ➢ d_2 and d_3 are always referenced jointly

Diamond Pattern

- **Diamond pattern in detail**
  - At least two predecessors of a table share a common predecessor
  - Dependencies between tuples of individual table synopses
  - Problems
    - Dependent reference probabilities
    - Joint inclusion probabilities

- **Solutions**

  a) Store tables with (possible) dependencies completely
    - For small tables (e.g., NATION of TPC-H)

  b) Switch back to Join Synopses
    - For tables with few/small successors

  c) Decide per tuple whether to use correlated sampling or not (see full paper)
    - For tables with many/large successors

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

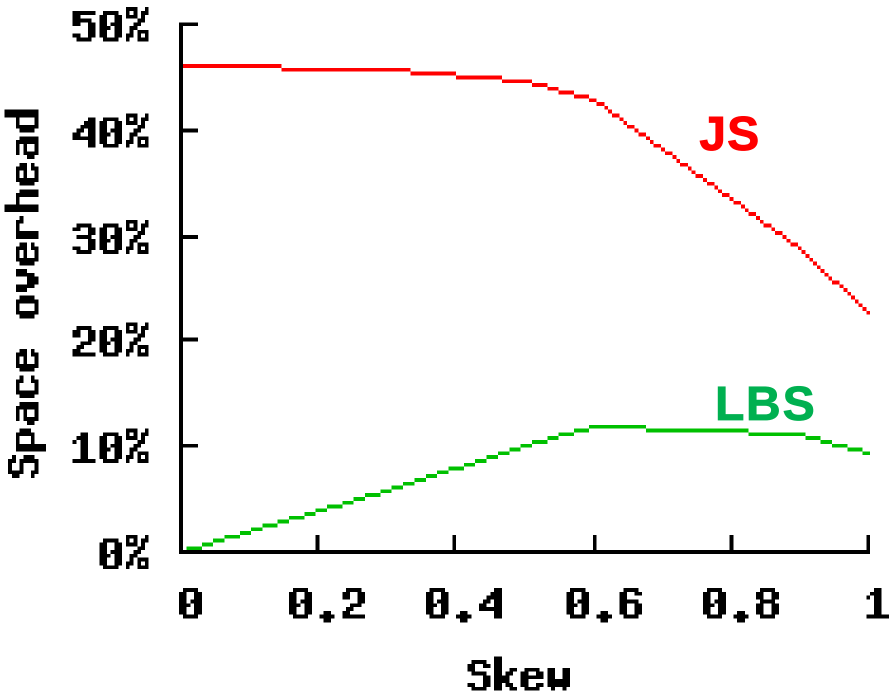
1. Introduction
2. Linked Bernoulli Synopses
3. Evaluation
4. Conclusion

Evaluation

- **Datasets**
  - TPC-H, 1GB
  - Zipfian distribution with z=0.5
    - For values and foreign keys
  - Mostly: equi-size allocation
  - Subsets of tables

Impact of skew

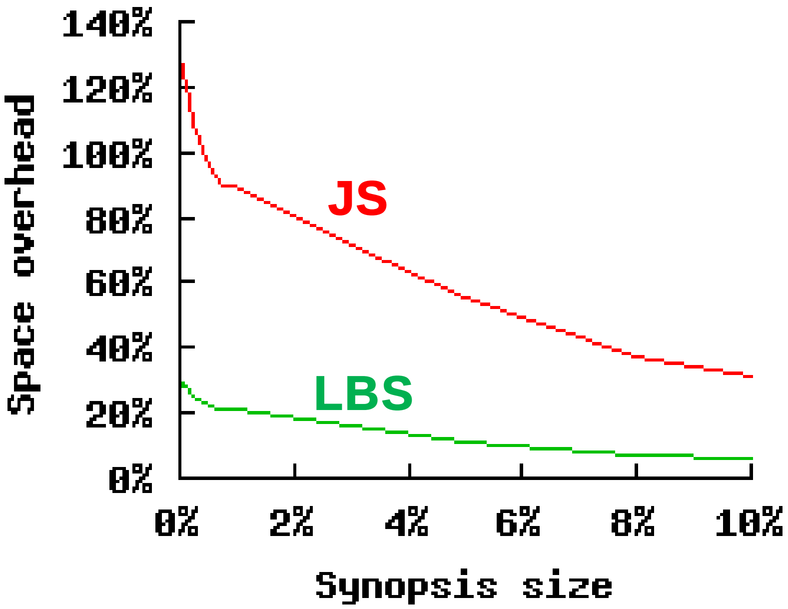
- **Tables: ORDERS and CUSTOMER**
  - varied skew of foreign key from 0 (uniform) to 1 (heavily skewed)



TPC-H, 1GB
Equi-size allocation

Impact of synopsis size

- **Tables: ORDERS and CUSTOMER**
  - varied size of sample part of the schema-level synopsis



Impact of number of tables

- **Tables**
  - started with LINEITEMS and ORDERS, subsequently added CUSTOMER, PARTSUPP, PART and SUPPLIER
  - shows effect of number transitive references

Outline

TECHNISCHE UNIVERSITÄT DRESDEN

Conclusion

- **Schema-level sample synopses**
  - A sample of each table + referential integrity
  - Queries with arbitrary foreign-key joins
- **Linked Bernoulli Synopses**
  - Correlate sampling processes
  - Reduces space overhead compared to Join Synopses
  - Samples are still uniform
- **In the paper**
  - Memory-bounded synopses
  - Exact and approximate solution

Thank you!

Questions?

TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science, Institute for System Architecture, Database Technology Group

Linked Bernoulli Synopses

Sampling Along Foreign Keys

Rainer Gemulla, Philipp Rösch, Wolfgang Lehner
Technische Universität Dresden

Backup:
Memory bounds

Memory-Bounded Synopses

- **Goal**
  - Derive a schema-level synopsis of given size M

- **Optimization problem**
  - Sampling fractions $q_1,\ldots,q_n$ of individual tables $R_1,\ldots,R_n$
  - Objective function $f(q_1,\ldots,q_n)$
    - Derived from workload information
    - Given by expertise
    - Mean of the sampling fractions
  - Constraint function $g(q_1,\ldots,q_n)$
    - Encodes space budget
    - $g(q_1,\ldots,q_n) \leq M$ (space budget)

Memory-Bounded Synopses

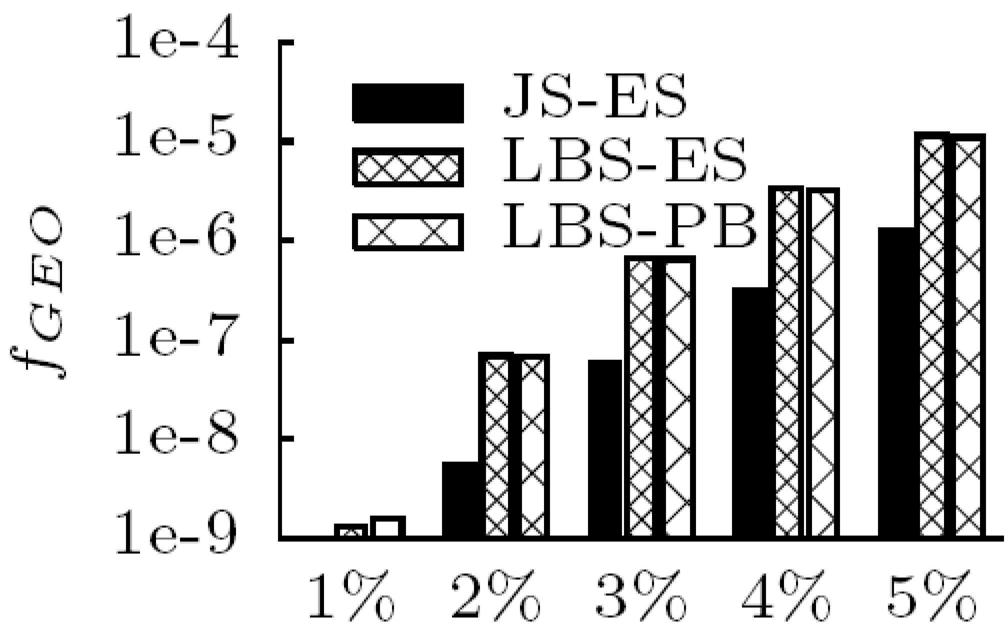
- **Exact solution**
  - *f* and *g* monotonically increasing
  - ➢ Monotonic optimization [TUY00]
  - But: evaluation of *g* expensive (table scan!)

- **Approximate solution**
  - Use an approximate, quick-to-compute constraint function
  - $g_l(q_1,\dots,q_n) = |R_1|\cdot q_1 + \dots + |R_n|\cdot q_n$
    - ignores size of reference tables
    - lower bound → oversized synopses
    - very quick
  - When objective function is mean of sampling fractions
    - $q_i \propto 1/|R_i|$
    - equi-size allocation

[TUY00] H. Tuy. Monotonic Optimization: Problems and Solution approaches. *SIAM J. on Optimization*, 11(2): 464-494, 2000.

Memory Bounds: Objective Function

- **Memory-bounded synopses**
  - All tables
  - computed f_{GEO} for both JS and LBS (1000 it.) with
    - equi-size approximation
    - exact computation



Memory Bounds: Queries

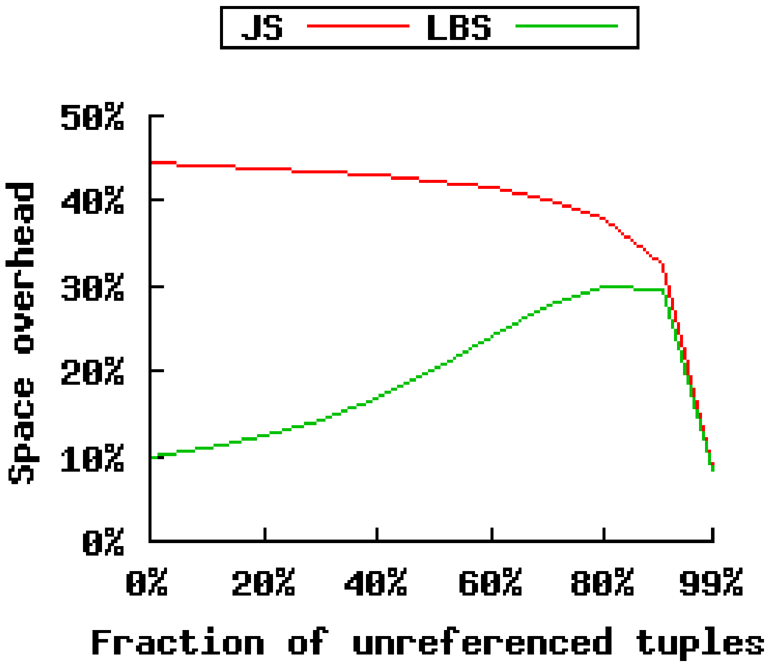
- **Example queries**
  - 1% memory bound
  - Q_1: average order value of customers from Germany
  - Q_2: average balance of these customers
  - Q_3: turnover generated by European suppliers
  - Q_4: average retail price of a part

	Q_1	Q_2	Q_3	Q_4
JS	3.51%	3.95%	3.28%	0.18%
LBS	2.69%	3.06%	2.43%	0.14%
	(−23.4%)	(−22.5%)	(−25.9%)	(−22.2%)

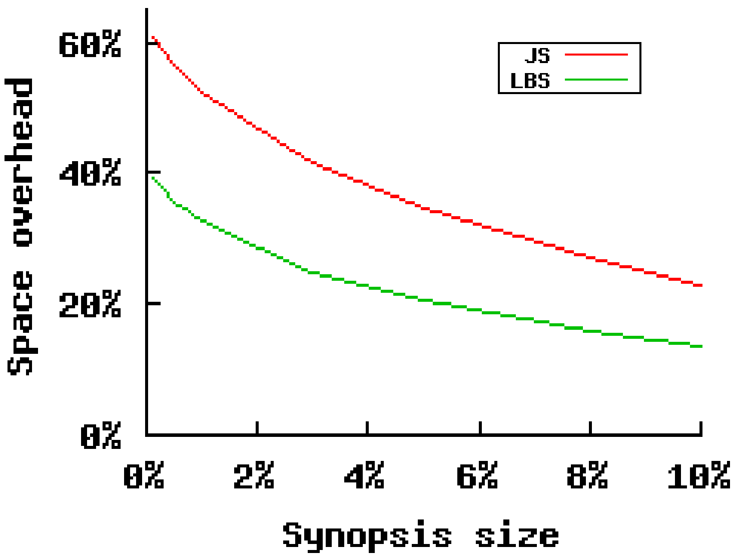
Backup:
Additional Experimental Results

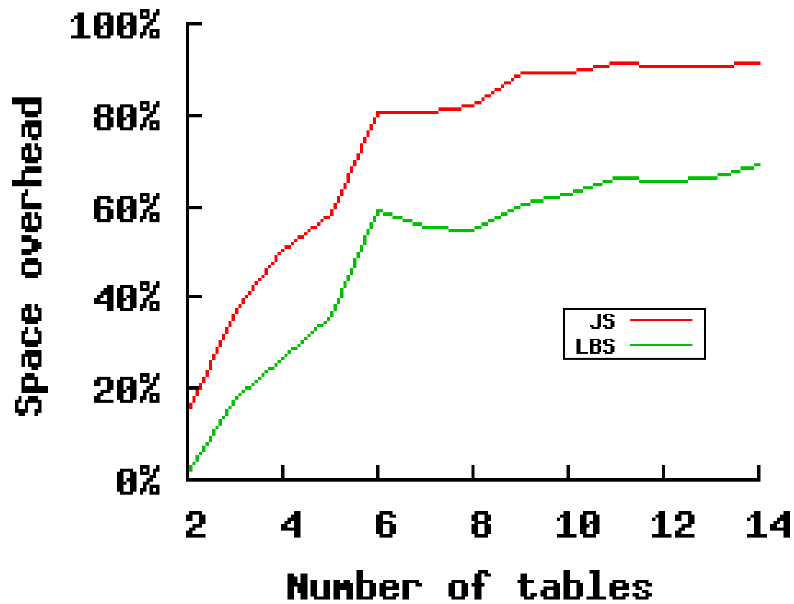
Impact of number of unreferenced tuples

- **Tables: ORDERS and CUSTOMER**
  - varied fraction of unreferenced customers from 0% (all customers placed orders) to 99% (all orders are from a small subset of customers)



CDBS Database





large number
of unreferenced
tuples (up to 90%)