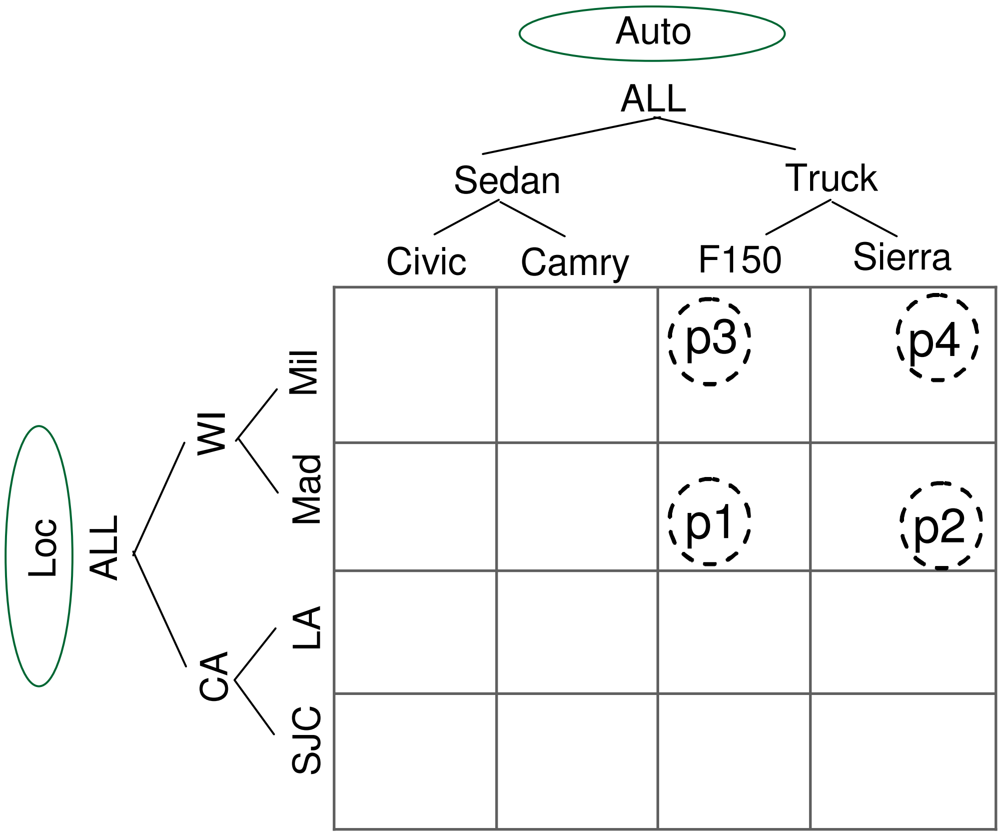
OLAP over Imprecise Data with Domain Constraints

Doug Burdick
University of Wisconsin – Madison

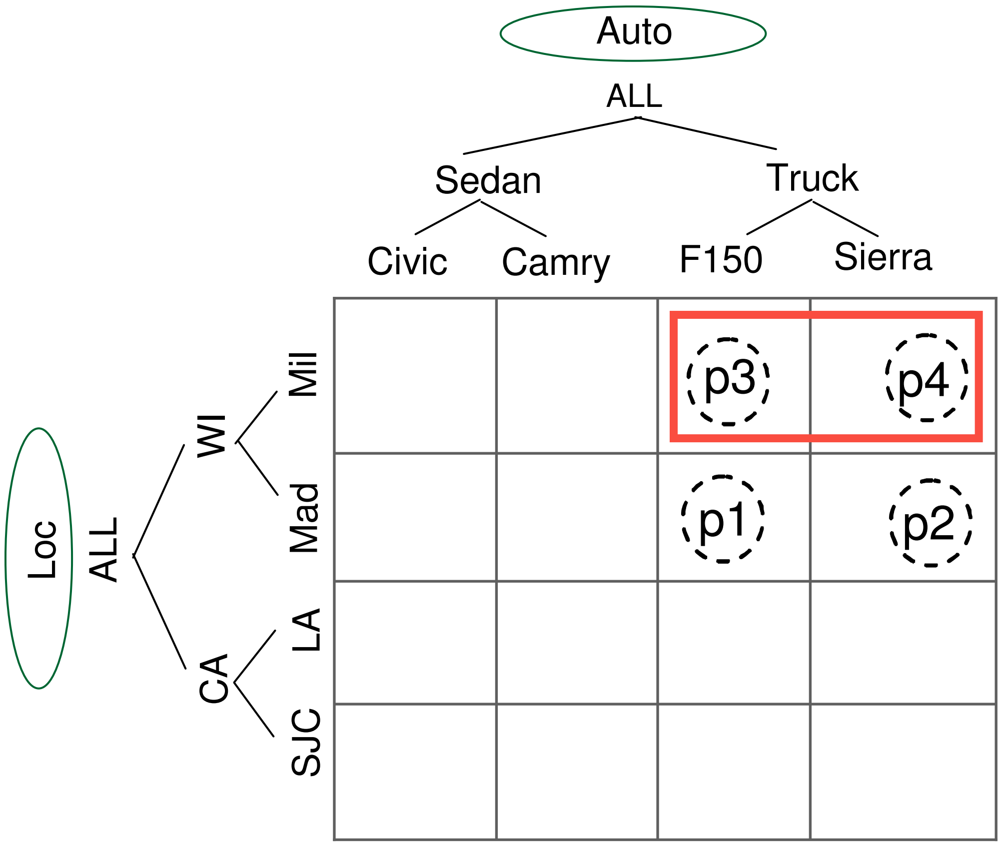
Joint work with AnHai Doan (UW-Madison), Raghu Ramakrishnan (Yahoo! Research), Shivakumar Vaithyanathan (IBM Research at Almaden)

Traditional OLAP: Data Model



FactID	Auto	Loc	Repair
p1	F150	Mad	100
p2	Sierra	Mad	500
p3	F150	Mil	100
p4	Sierra	Mil	200

Traditional OLAP: Queries



Auto = Truck
Loc = Mil
SUM(Repair) = ?

Answer: 300

FactID	Auto	Loc	Repair
p1	F150	Mad	100
p2	Sierra	Mad	500
p3	F150	Mil	100
p4	Sierra	Mil	200

Querying Information Extracted from Text

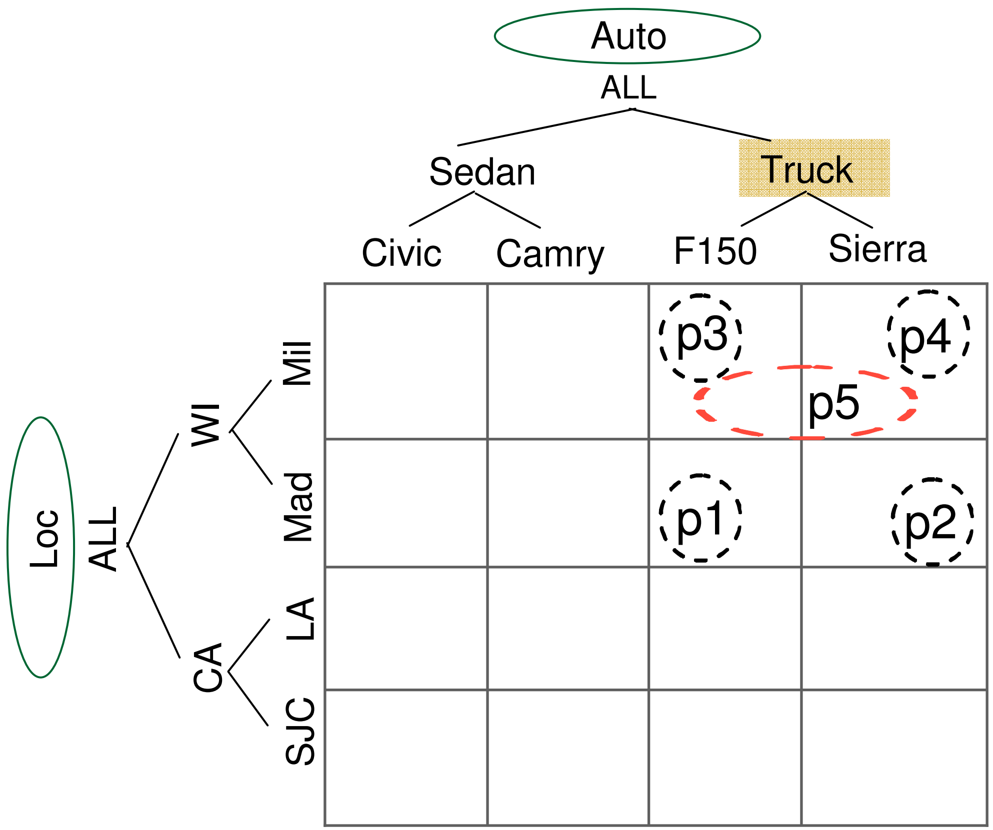
ID	Review Text
p1	I love the reliability of my F150 from Zimbrick Ford in Milwaukee. Much better than my Sierra. Paid $30000 for a 4WD.
p2	My 5-speed Subaru Outback handles well in Wisconsin winters. Great value at $25000
p3	After my old car was totaled in the Madison flood, I bought a BMW 330. It's at the mechanic's all the time.

For each location, what is the average price for different cars?

ID	Location	Model	Price
p1	Milwaukee	{F150, Sierra}	30000
p2	Wisconsin	Subaru Outback	25000
p3	Madison	BMW 330	330

In a dataset from a real-world application at IBM Almaden with 800,000 facts, 30% were imprecise

[VLDB 05] Proposed Solution: Allow Imprecise Facts

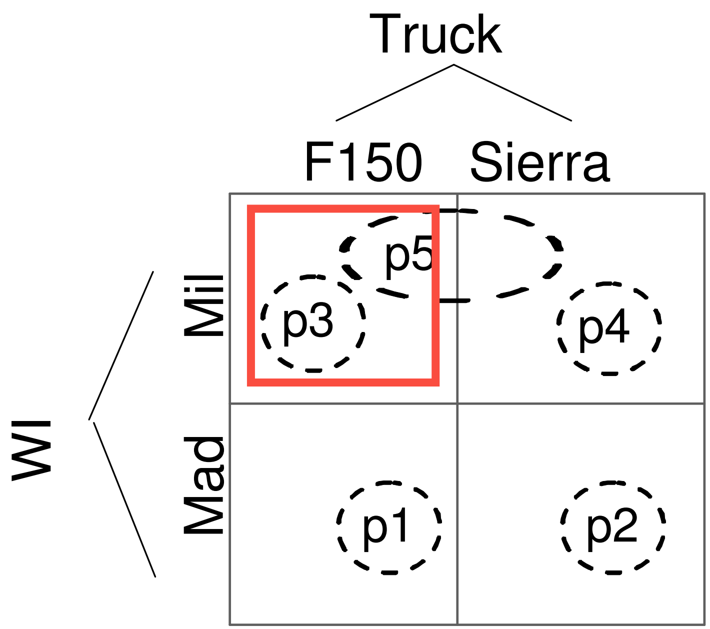


FactID	Auto	Loc	Repair
p1	F150	Mad	100
p2	Sierra	Mad	500
p3	F150	Mil	100
p4	Sierra	Mil	200
p5	Truck	Mil	100

[VLDB 05] Problem: How to Query Imprecise Facts

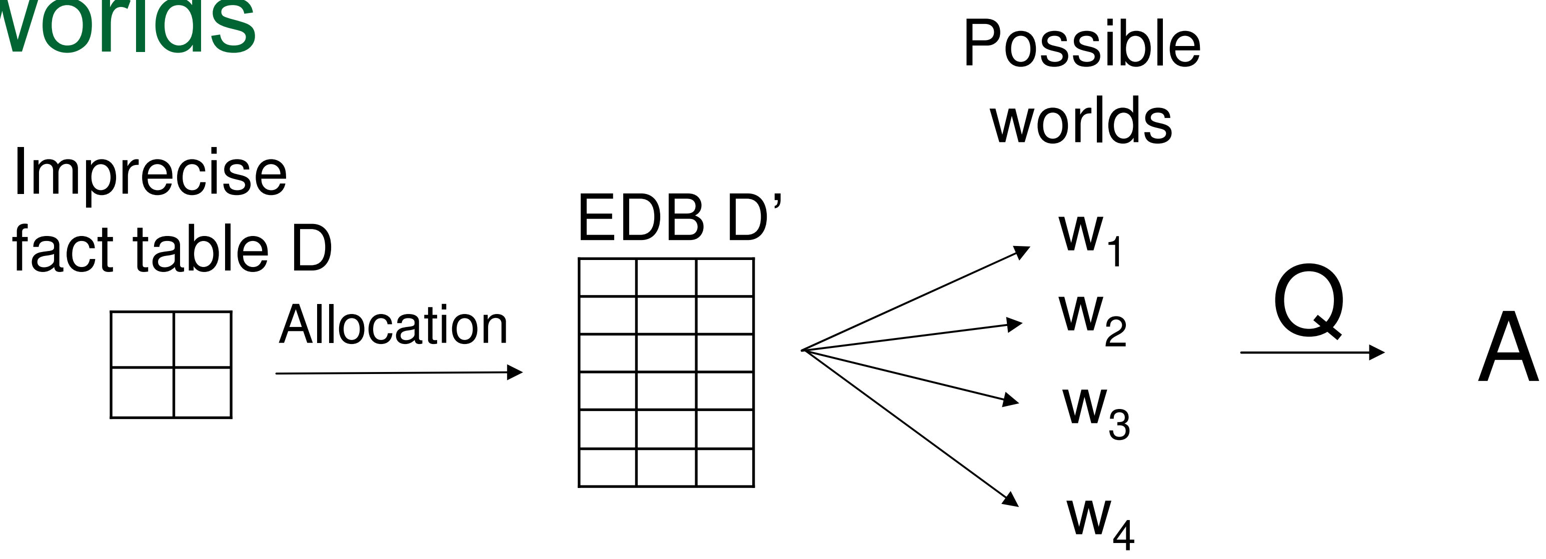
Auto = F150
Loc = Mil
SUM(Repair) = ?

Answer: ?



FactID	Auto	Loc	Repair
p1	F150	Mad	100
p2	Sierra	Mad	500
p3	F150	Mil	100
p4	Sierra	Mil	200
p5	Truck	Mil	100

[VLDB 05] Solution: Use possible worlds



Query answer is expected value over possible worlds

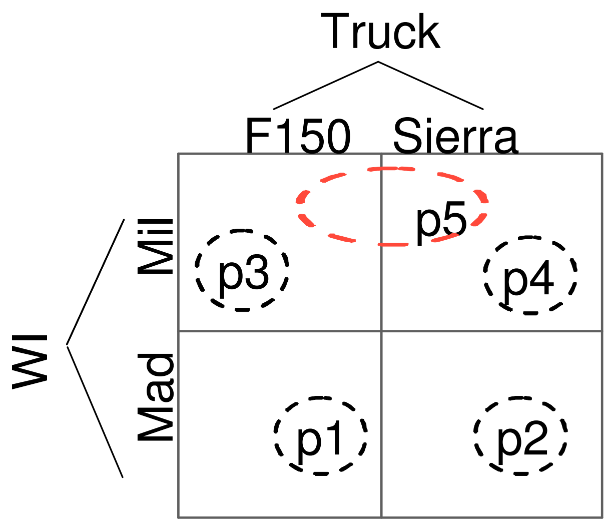
[VLDB 05] Example

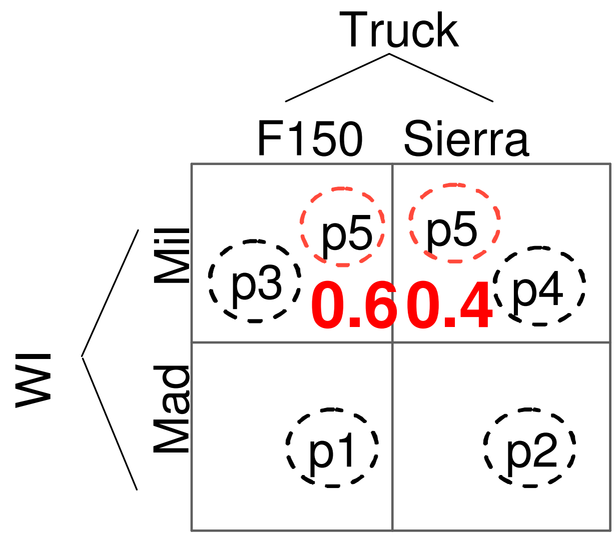
Imprecise Fact Table D

FactID	Auto	Loc	Repair
p1	F150	Mad	100
p2	Sierra	Mad	500
p3	F150	Mil	100
p4	Sierra	Mil	200
p5	Truck	Mil	100

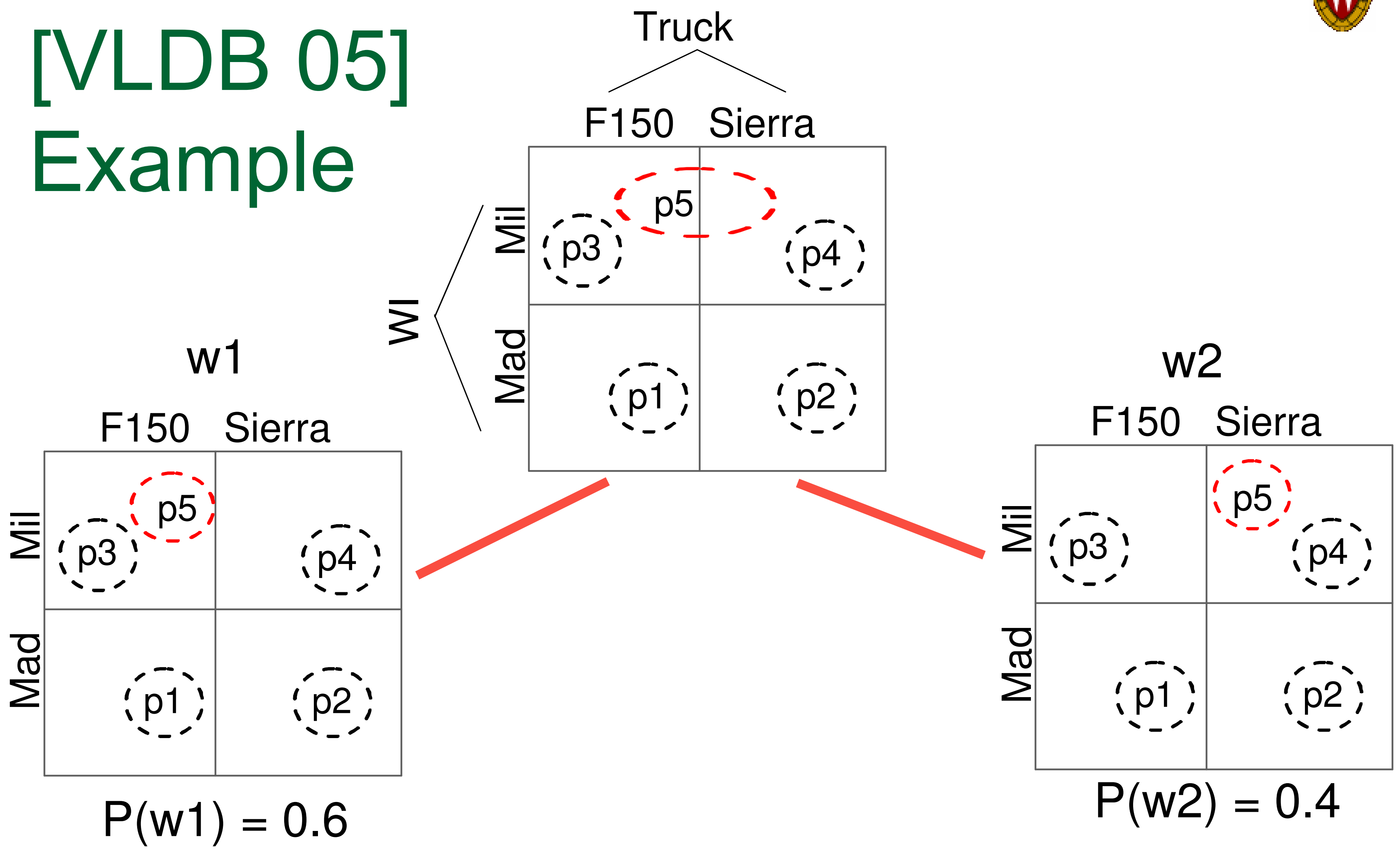
Extended Database D’

ID	FactID	Auto	Loc	Repair	Alloc
1	p1	F150	Mad	100	1.0
2	p2	Sierra	Mad	500	1.0
3	p3	F150	Mil	100	1.0
4	p4	Sierra	Mil	200	1.0
5	p5	F150	Mil	100	0.6
6	p5	Sierra	Mil	100	0.4





[VLDB 05] Example



Contributions [VLDB 05, VLDB 06]

- Formalize entire process

Assumes all imprecise facts are independent

- Demonstrate how to answer queries efficiently

Challenge: Incorporate Domain Constraints

ID	Repair Text
r1	F150, oil change, $100, WI, John Smith
r2	customer John Smith brought F150 to garage engine noise, WI, $250
r3	Madison, Honda, broken ex. pipe, Dells & I-90, towed 25 miles, $130

FactID	Loc	Auto	Name	Cost
p1	Wisconsin	F150	John Smith	100
p2	Wisconsin	F150	John Smith	250
p3	Madison	Honda	Dells	130
p4	Dells	Honda	Madison	130

"Two facts with same person name and model must have same city"

"Exactly one of facts p3 or p4 exists"

Summary of Contributions

- Present constraint language **L**
  - Define both syntax of **L** and semantics of answering queries with constraints defined in **L**

- Efficiently answer queries with constraints using a marginal database D*

- Present algorithms to efficiently construct marginal database D*

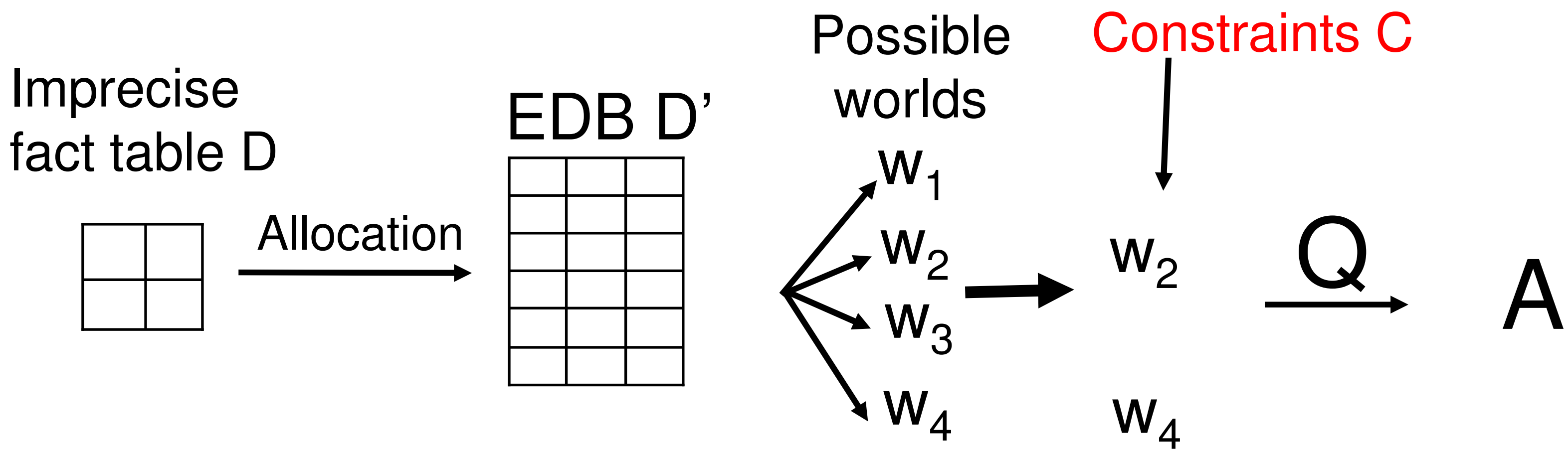
Constraint Language: Examples

- "Two facts with same person name and model must have same location"
  - (r.Name = r'.Name) ^ (r.Auto = r'.Auto) → (r.Loc = r'.Loc)
- "Exactly one of facts p3 or p4 exists"
  - exists(p3) → ¬exists(p4)
  - exists(p4) → ¬exists(p3)
- "If the location for p1 is Madison, then p3 must exist (and p4 cannot exist)"
  - (p1.Loc = "Madison") → exists(p3) ^ ¬exists(p4)

Constraint Language: Syntax

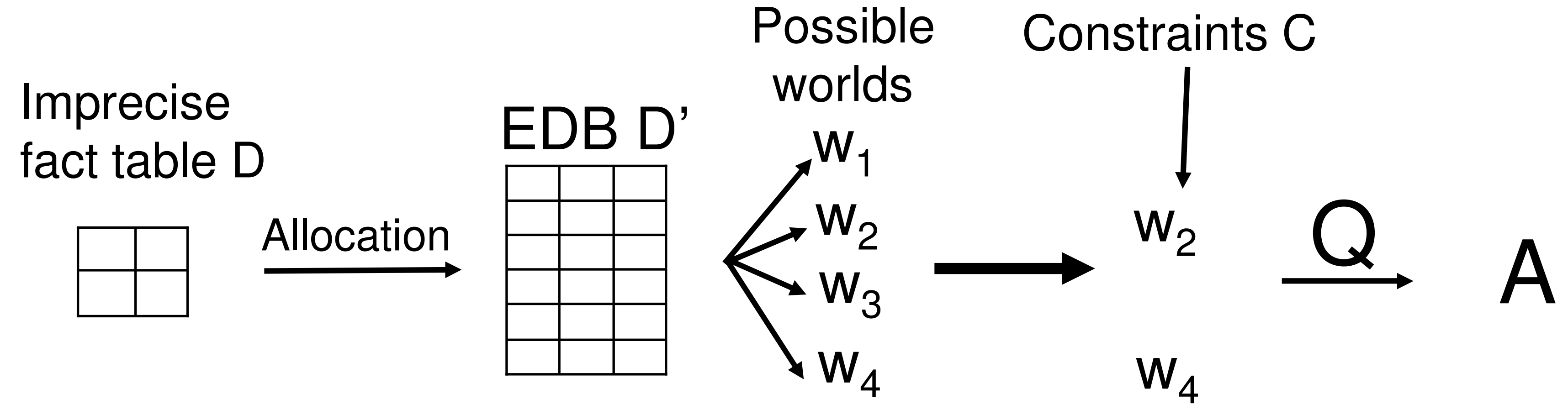
- A constraint has form A $\Rightarrow$ B where A,B are conjunctions of atoms
- Atoms have form [r.A Θ c] or [r.A Θ r'.A] or exists(r), $\neg$exists(r) where
  - r,r' are either
    - specific factIDs themselves
    - **variables** that bind to factIDs in D
  - r.A is the value of attribute A of fact r.
  - $\Theta \in \{=, \neq, \leq, <, \geq, >\}$ is a comparison operator over the appropriate domain
  - c is a constant from dom(A), and
  - exists(r) ($\neg$exists(r)) is a predicate that holds if fact r exists (cannot exist)

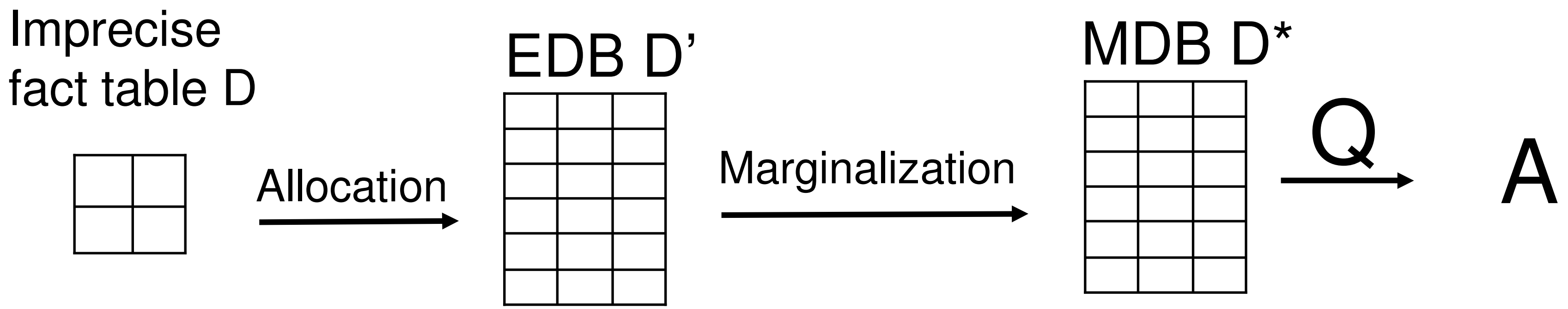
Constraint Language: Semantics



- A possible world satisfying all constraints is **valid**
- Query answer is expected value over **valid** possible worlds

Efficient Query Answering





- Can compute expected value over valid possible worlds in single scan of Marginal Database (MDB) D*

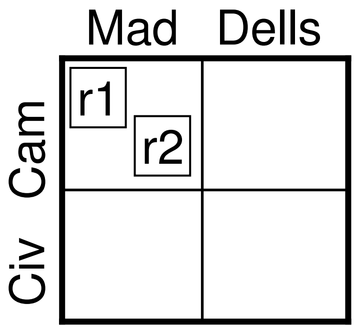
Constraint: (r.Model = r’.Model) → (r.Loc = r’.Loc)

EDB D’

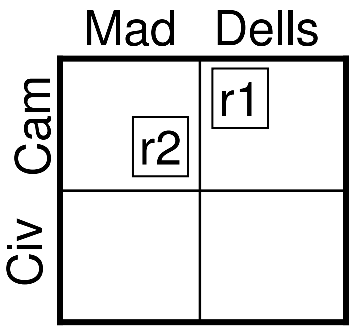
FactID	Model	Loc	Cost	Alloc
r1	Cam	Mad	100	0.7
r1	Cam	Dells	100	0.3
r2	Cam	Mad	400	0.8
r2	Cam	Dells	400	0.2

MDB D*

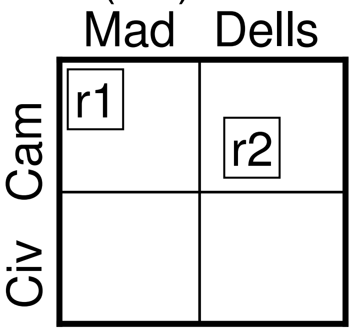
FactID	Model	Loc	Cost	Mar
r1	Cam	Mad	100	0.9
r1	Cam	Dells	100	0.1
r2	Cam	Mad	400	0.9
r2	Cam	Dells	400	0.1



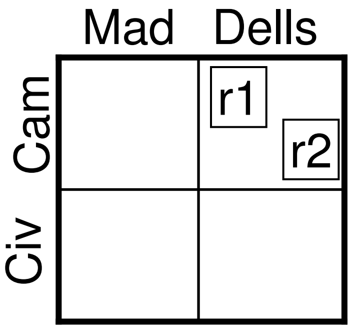
P(w1) = 0.56



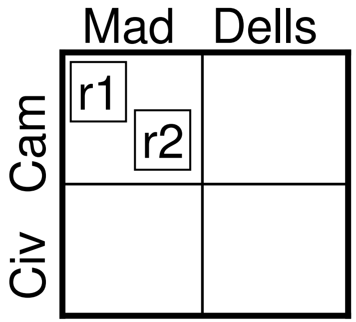
P(w2) = 0.24



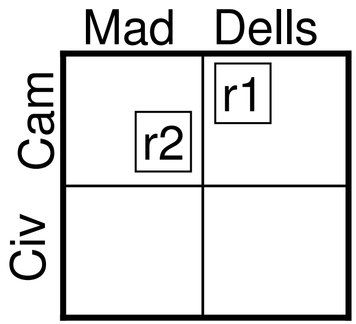
P(w3) = 0.56



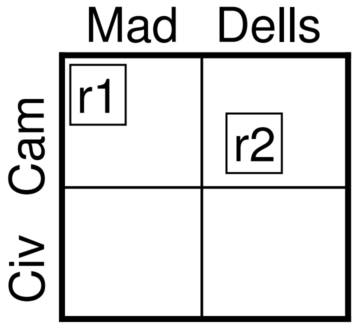
P(w4) = 0.06



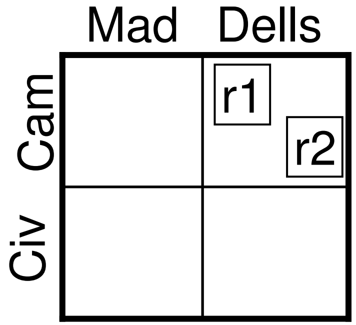
$P^N(w1) = 0.90$



$P^N(w2) = 0$



$P^N(w3) = 0$



$P^N(w4) = 0.10$

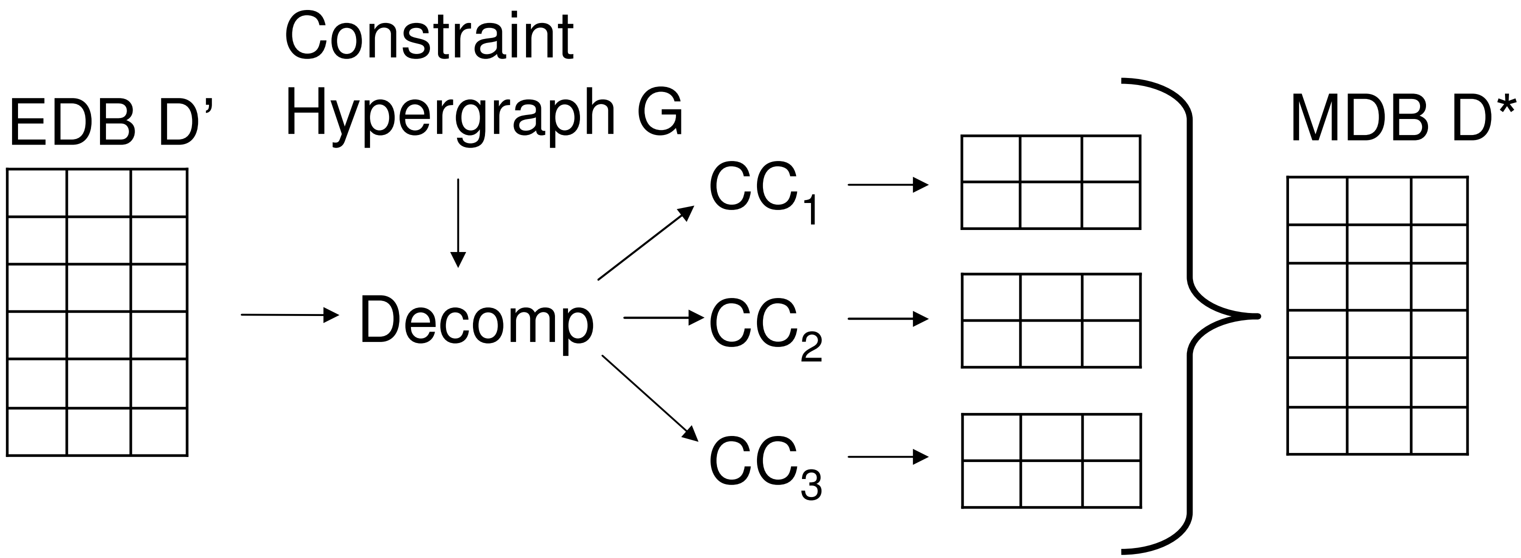
Marginal Database (MDB) D*

- Let D' be EDB obtained from imprecise fact table D
- Each claim in D' has tuple f_t with allocation weight w_t
- Let W be set of valid possible worlds satisfying a given set of constraints C
- Let m_t be the total probability of worlds in W where f_t is true.

- We refer to m_t as the marginal probability of f_t and (f_t, m_t) is a marginal tuple.

- Store all marginal tuples in marginal database (MDB) D*

Marginalization Algorithms

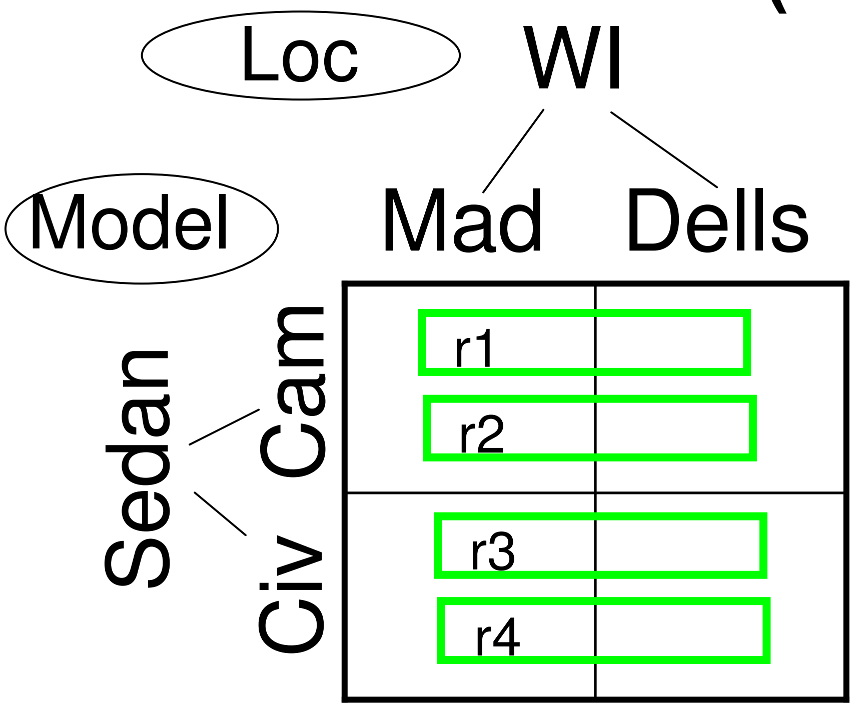


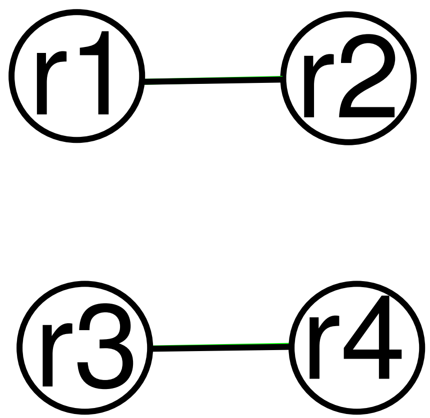
- Can process connected component in constraint hypergraph independently

Constraint Hypergraph: Example

Constraint:

(r.Model = r'.Model) → (r.Loc = r'.Loc)





Constraint Hypergraph: G=(V,H)

- **Nodes V:** For each fact r in given imprecise database D, introduce a node to V

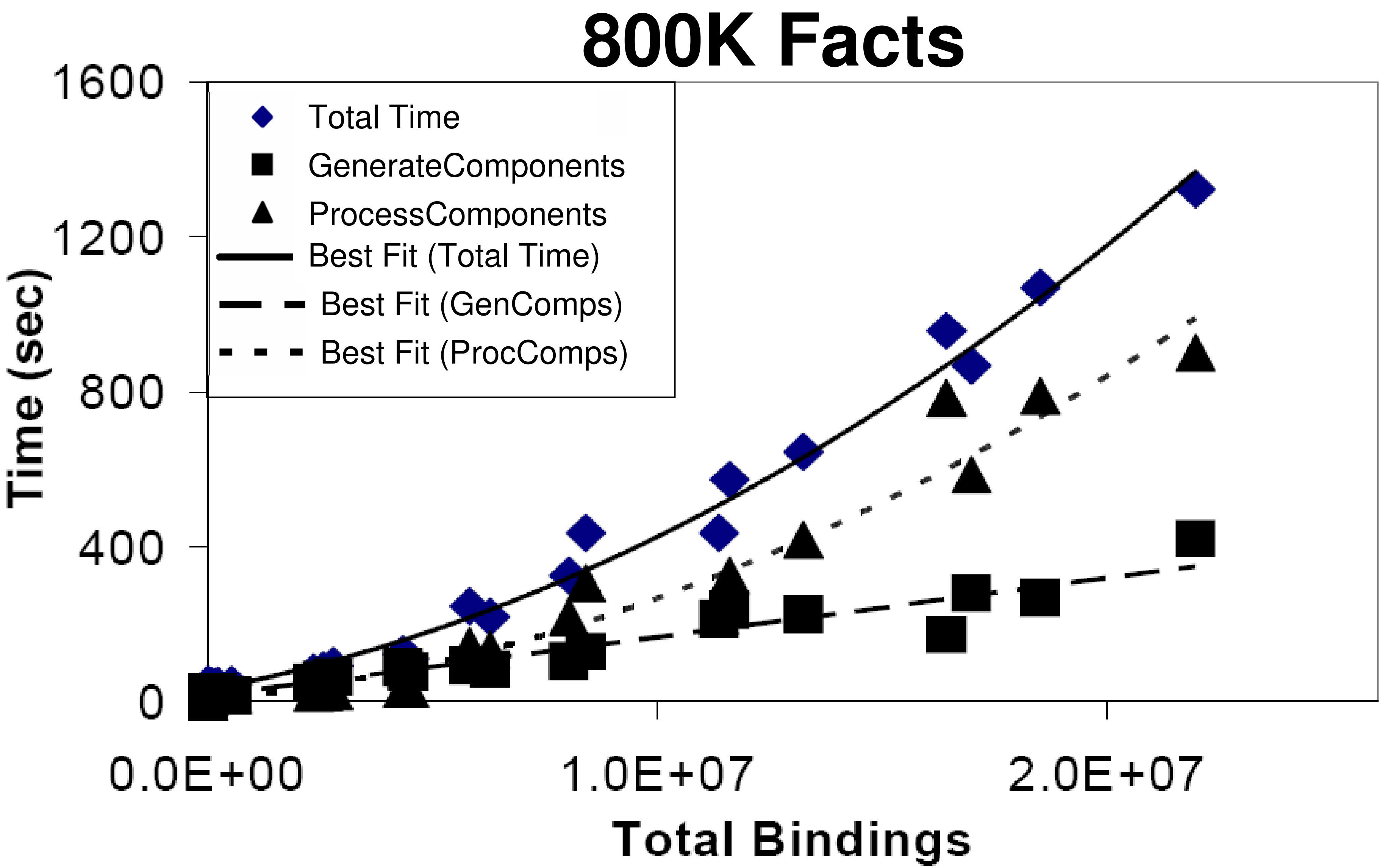
- **Hyperedges H:** For each minimal set of facts with a combination of completions violating a constraint, introduce a hyperedge to H

Experimental Setup

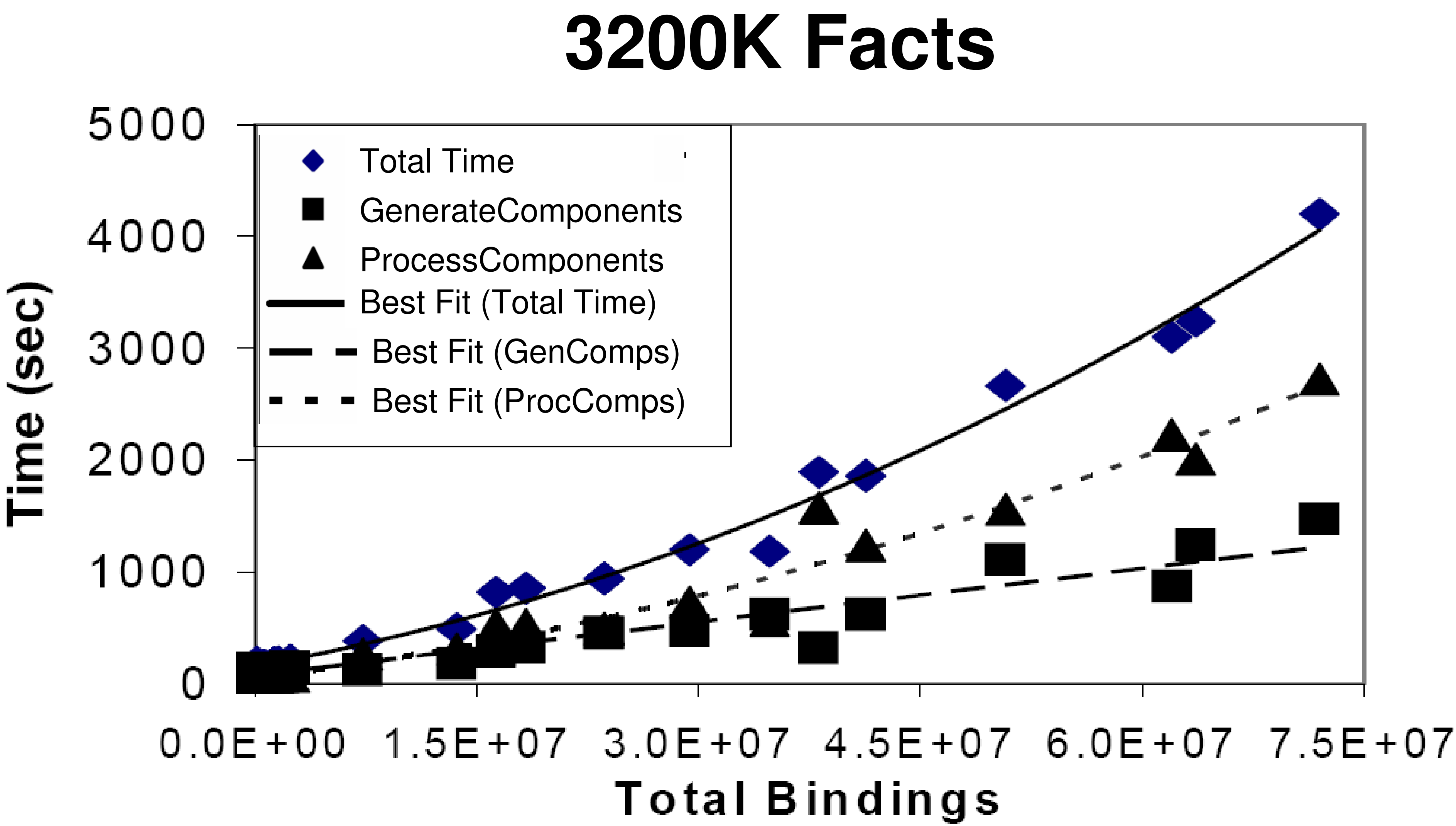
- Algorithms evaluated on several datasets
  - Real-world dataset: 798K facts , 4 dimensions
  - Used several synthetic datasets
    - Scalability (up to 3.2 million tuples)

- Constraint sets
  - Randomly generated several constraint sets of varying “complexity”
  - Develop suitable complexity metric

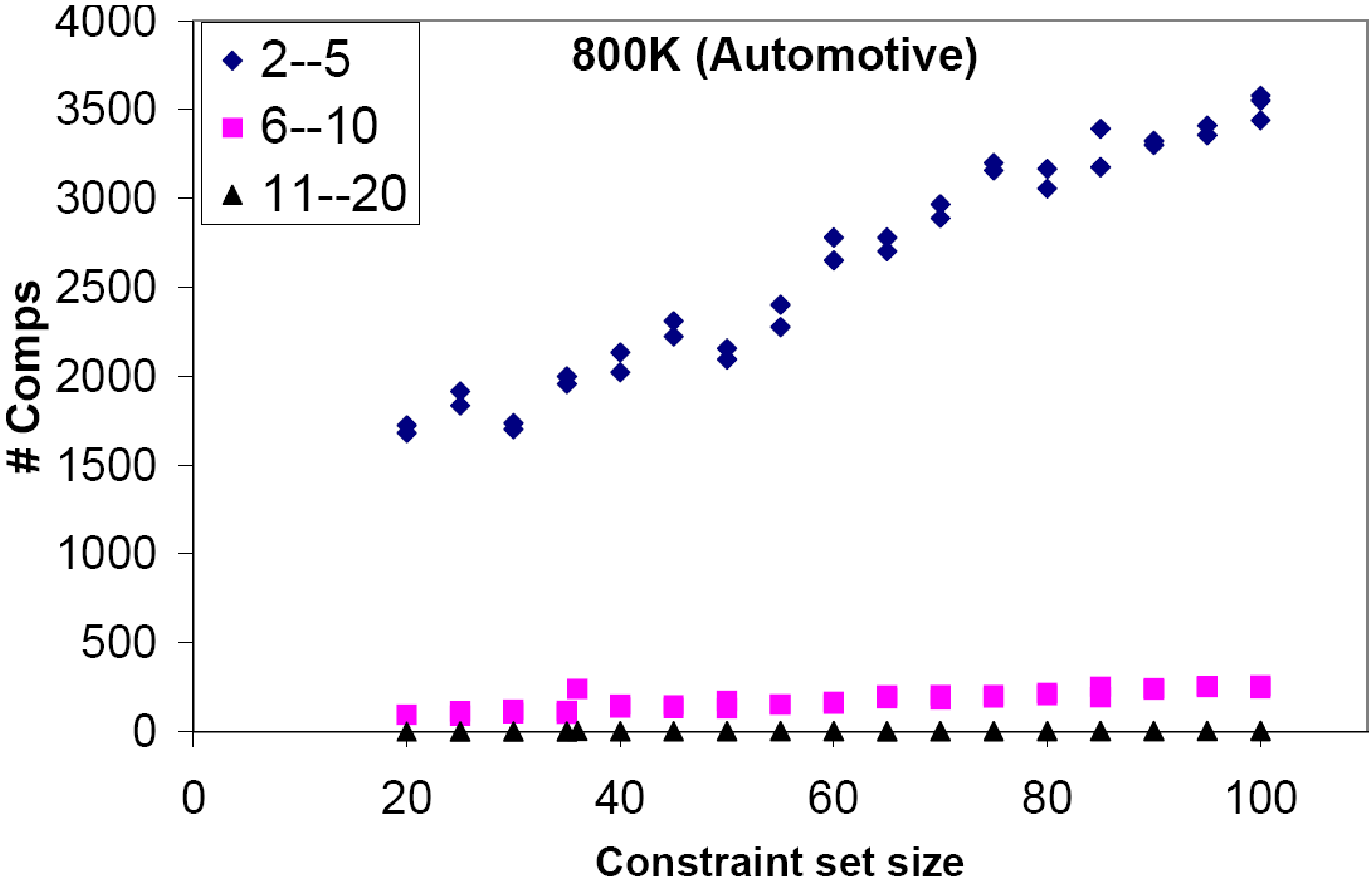
Performance



Performance



Component Sizes



Related work

- Imprecise data with constraints
  - MayBMS [Antova et al. 07]
  - Representing and Querying Correlated Tuples in Probabilistic Databases [Sen, Deshpande 07]
  - ConQuer [Fuxman et al 05]
- Probabilistic databases
  - Probabilistic Databases [Dalvi et al. 04]
  - TRIO system for uncertain data [Widom et al.05]
- OLAP
  - Constraints in OLAP [Hurtado et. al 02]
  - OLAP over Incomplete Data [Dyreson 96]

Summary

- We extend our framework for OLAP over imprecise data to support domain information.

- Eliminate the strong independence assumptions required earlier
  - Often violated in many applications (e.g., IE from text)

- First work we are aware of to consider OLAP aggregation queries over imprecise data **in the presence of constraints**

Discussion

- Pretty brute-force
- Fact Table => EDB, how?
- Other Queries: AVG, MIN, MAX
  - How to generate MDB?
- Expressiveness of Constraints
  - A => B (0.4) or C (0.6)
  - More complex distributional constraints on data