http://www.cs.cornell.edu/big**red**data/maybms/
http://maybms.sourceforge.net/

**C. Koch*** D. Olteanu** L. Antova*
J. Huang*/** M. Goetz* O. Kennedy*

* Cornell University ** Oxford University

# The MayBMS Project

Both a research and a development project.

Research:

- Probabilistic DBMS are in their infancy.
- Foundations must be laid: query language, representation and storage, scalable query processing, updates, concurrency control, APIs, ...
- No usable systems yet, but a lot of excitement.

Goals of the development project:

- Build a first robust, industrial-strength probabilistic DBMS.
- Reuse as much database technology as possible.
- Establish a code base that researchers can build upon.
- See what users do with it: use cases, killer apps?

# DBMS for Uncertain/Probabilistic Data – Applications

- Social network analysis, protein-protein interactions, etc.
- Risk management: Decision support queries, hypothetical queries
- (Web) information extraction, data integration, data cleaning
- Forecasting/prediction
- Managing scientific data; sensor data
- Lean expert systems; diagnosis; causality
- Crime fighting, surveillance, plagiarism detection, predicting terrorist actions, ...

So far, probabilistic databases do **not** have a user base (unlike the graphical models work in AI) – but then, no systems are available.

# Probability of a triangle in a random graph of three nodes

| *T* | u | v | bit | p |
|---|---|---|---|---|
| | 1 | 2 | 1 | .5 |
| | 1 | 2 | 0 | .5 |
| | 1 | 3 | 1 | .5 |
| | 1 | 3 | 0 | .5 |
| | 2 | 3 | 1 | .5 |
| | 2 | 3 | 0 | .5 |

```
create table E0 as
select Q.u, Q.v
from  (repair key (u,v) in T weight by p) Q
where Q.bit = 1;
```

8 possible worlds, one has a triangle.

```
create table E as ((select * from E0) union (select v as u, u as v from E0));
```

```
select  conf() as triangle_prob
from    E e1, E e2, E e3
where   e1.v = e2.u and e2.v = e3.u and e3.v = e1.u
and     e1.u <> e2.u and e1.u <> e3.u and e2.u <> e3.u;
```

| triangle_prob |
|---|
| 0.125 |

# Hypothetical Queries: Skills Management

Suppose I buy a company and exactly one employee leaves.

Which skills do I gain for certain?

| CE | CID | EID |
|---|---|---|
| | Google | Bob |
| | Google | Joe |
| | Yahoo | Dan |
| | Yahoo | Bill |
| | Yahoo | Fred |

| ES | EID | Skill |
|---|---|---|
| | Bob | Web |
| | Joe | Web |
| | Dan | Java |
| | Dan | Web |
| | Bill | Search |
| | Fred | Java |

```
create table RemainingEmployees as
select CE.cid, CE.eid
from   CE,
       (repair key (dummy)
        in (select 1 as dummy, *
            from CE)) Choice
where CE.cid = Choice.cid
and    CE.eid <> Choice.eid;
```

```
create table SkillGained as
select Q1.cid, Q1.skill, p1, p2, p1/p2 as p
from (select R.cid, ES.skill, conf() as p1
      from RemainingEmployees R, ES
      where R.eid = ES.eid
      group by R.cid, ES.skill) Q1,
     (select cid, conf() as p2
      from RemainingEmployees
      group by cid) Q2
where Q1.cid = Q2.cid;

select cid, skill from SkillGained where p=1;
```

| CID | Skill | p1 | p2 | p |
|---|---|---|---|---|
| Google | Web | 2/5 | 2/5 | 1 |
| Yahoo | Java | 3/5 | 3/5 | 1 |
| Yahoo | Web | 2/5 | 3/5 | 2/3 |
| Yahoo | Search | 2/5 | 3/5 | 2/3 |

| CID | Skill |
|---|---|
| Google | Web |
| Yahoo | Java |

# Probabilistic c-tables

- Conditional tables [Imielinski, Lipski]
- Relational tables with variables (labeled nulls) in which each tuple has a local (Boolean) condition.
- Possible worlds semantics: world given by variable assignment; fill in variables, drop tuples that do not satisfy condition.

| CID | EID | $\phi$ |
|---|---|---|
| Google | Bob | x=1 |
| Google | Joe | x=2 |
| Yahoo | Dan | x=3 |
| Yahoo | Bill | x=4 |
| Yahoo | Fred | x=5 |

- Evaluation of relational algebra on such tables easy.

$$
\begin{aligned}
[\![R \times S]\!] &= \{\langle r, s, \phi \wedge \psi\rangle \mid \langle r, \phi\rangle \in R, \langle s, \psi\rangle \in S\} \\
[\![\sigma_\psi(R)]\!] &= \{\langle r, \phi \wedge \psi\rangle \mid \langle r, \phi\rangle \in R\} \\
[\![R - R']\!] &= \{\langle r, \phi \wedge \neg\psi\rangle \mid \langle r, \phi\rangle \in R, \langle r, \psi\rangle \in R'\} \\
\pi, \cup & \quad \ldots \quad \text{similar.}
\end{aligned}
$$

(Difference operation – here, simplifying assumption that tuples do not contain variables.)

- Probabilistic c-tables: variables are random variables with some joint distribution.
- Finite case: independent random variables no loss of generality!

# MayBMS Query Engine Architecture

Query

query evaluation

- computing probabilities, moments, and statistical tests
- query plans on c-tables

representation system

- (probabilistic) c-tables
- succinct representation of joint distribution of random variables (exchangeable)

# Model of Probabilistic Databases (discrete case)

## Definition

Given a relational schema $\Sigma$, a **probabilistic database** is a finite set of instances over $\Sigma$ (called possible worlds), where

- each world has a weight (called probability) between 0 and 1 and
- the weights of all worlds sum up to 1.

- A probabilistic database in our model is an uncertain relational database.
- Conceptually, one of the possible worlds is "true", but we do not know which one (subjectivist Bayesian interpretation).
- This is the conceptual model; the physical representation in the system is quite different!

# The Query Language: Core Algebra

- The operations of relational algebra.
  - Evaluated individually, in "parallel" in all possible worlds.
- An operation conf($R$) for computing tuple confidence values.
  - Computes, for each tuple that occurs in $R$ in at least one world, the sum of the probabilities of the worlds in which it occurs.
- An operation repair-key$_{\vec{A}[@P]}(R)$ for *introducing* uncertainty.
  - Conceptually, nondeterministically chooses a maximal repair of key $\vec{A}$.
  - Turns a possible world into the set of worlds consisting of all possible maximal repairs.
- Here I discuss only the core algebra: The query language implemented in MayBMS is strictly a generalization of SQL.
- Apart from repair-key and conf(), extensions include expectations of sums and counts (with group-by).

# Update Language

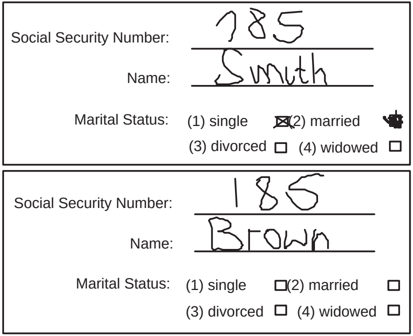
- Insert, update, and delete statements based on queries in the query language presented earlier.
  Example: insert into R (Q); where $Q$ is a query in the algebra.
- An operation assert$_\phi$ that conditions the database using a constraint $\phi$.
  - Removes those worlds that violate $\phi$.
- Note: Views that involve nondeterministic operations (repair-key) are conceptually materialized.
  - Repeated accesses return the same result.

# Desiderata for a representation system

1. Expressiveness.
   - Ability to represent query results.
2. Succinctness – Space-efficient storage.
   - Suppose that OCR results of census forms contain two possible readings for 0.1% of the answers.
   - For the US census, on the order of $2^{10,000,000}$ possible worlds, each one close to one Terabyte of data.
3. Efficient real-world query processing.
   - There is a tradeoff with succinctness.
   - We want to do well in practice.

# Representation systems: naive tables (SQL)

Census data scenario: Suppose we have to enter the information from forms like these into a database.




SQL table with nulls

| (TID) | S | N | M |
|---|---|---|---|
| $t_1$ | null | Smith | null |
| $t_2$ | null | Brown | null |

Much of the available information cannot be represented and is lost, e.g.

1. Smith's SSN is either 185 or 785.
2. Brown's SSN is either 185 or 186.
3. Data cleaning: No two distinct persons can have the same SSN.

# U-Relational Databases

| $U_{R[SSN]}$ | V $\mapsto$ D | TID | SSN |
|---|---|---|---|
| | $x \mapsto 1$ | $t_1$ | 185 |
| | $x \mapsto 2$ | $t_1$ | 785 |
| | $y \mapsto 1$ | $t_2$ | 185 |
| | $y \mapsto 2$ | $t_2$ | 186 |

| $U_{R[M]}$ | V $\mapsto$ D | TID | M |
|---|---|---|---|
| | $v \mapsto 1$ | $t_1$ | 1 |
| | $v \mapsto 2$ | $t_1$ | 2 |
| | $w \mapsto 1$ | $t_2$ | 1 |
| | $w \mapsto 2$ | $t_2$ | 2 |
| | $w \mapsto 3$ | $t_2$ | 3 |
| | $w \mapsto 4$ | $t_2$ | 4 |

| $U_{R[N]}$ | TID | N |
|---|---|---|
| | $t_1$ | Smith |
| | $t_2$ | Brown |

| $W$ | V $\mapsto$ D | P |
|---|---|---|
| | $x \mapsto 1$ | .4 |
| | $x \mapsto 2$ | .6 |
| | $y \mapsto 1$ | .7 |
| | $y \mapsto 2$ | .3 |
| | $v \mapsto 1$ | .8 |
| | $v \mapsto 2$ | .2 |
| | $w \mapsto 1$ | .25 |
| | $w \mapsto 2$ | .25 |
| | $w \mapsto 3$ | .25 |
| | $w \mapsto 4$ | .25 |

- Discrete independent (random) variables $(x, y, v, w)$.
- Representation: U-relations + table $W$ representing distributions.
- The schema of each U-relation consists of
  - a tuple id column,
  - a *set* of column pairs $(V_i, D_i)$ representing variable assignments, and
  - a set of value columns.

# Semantics of U-Relational Databases

- Each possible world is identified by a valuation $\theta$ that assigns one of the possible values to each variable.
- The probability of the possible world is the product of weights of the values of the variables.
- The value-component of a tuple of a U-relation is in a given possible world if its variable assignments are consistent with $\theta$.
- Attribute-level uncertainty through vertical decompositioning.
- Theorem [Antova, Jansen, K., Olteanu, ICDE 2008]: U-relations are a complete representation system for finite probabilistic databases.
  - Graphical models cannot express dependencies that U-relations cannot express.

# Semantics of U-Relational Databases

| $U_{R[SSN]}$ | V $\mapsto$ D | TID | SSN |
|---|---|---|---|
| | $x \mapsto 1$ | $t_1$ | 185 |
| | $x \mapsto 2$ | $t_1$ | 785 |
| | $y \mapsto 1$ | $t_2$ | 185 |
| | $y \mapsto 2$ | $t_2$ | 186 |

| $U_{R[M]}$ | V $\mapsto$ D | TID | M |
|---|---|---|---|
| | $v \mapsto 1$ | $t_1$ | 1 |
| | $v \mapsto 2$ | $t_1$ | 2 |
| | $w \mapsto 1$ | $t_2$ | 1 |
| | $w \mapsto 2$ | $t_2$ | 2 |
| | $w \mapsto 3$ | $t_2$ | 3 |
| | $w \mapsto 4$ | $t_2$ | 4 |

| $U_{R[N]}$ | TID | N |
|---|---|---|
| | $t_1$ | Smith |
| | $t_2$ | Brown |

| $W$ | V $\mapsto$ D | P |
|---|---|---|
| $\rightarrow$ | $x \mapsto 1$ | .4 |
| | $x \mapsto 2$ | .6 |
| | $y \mapsto 1$ | .7 |
| $\rightarrow$ | $y \mapsto 2$ | .3 |
| $\rightarrow$ | $v \mapsto 1$ | .8 |
| | $v \mapsto 2$ | .2 |
| $\rightarrow$ | $w \mapsto 1$ | .25 |
| | $w \mapsto 2$ | .25 |
| | $w \mapsto 3$ | .25 |
| | $w \mapsto 4$ | .25 |

- We choose possible world $\{x \mapsto 1, y \mapsto 2, v \mapsto 1, w \mapsto 1\}$.

# Semantics of U-Relational Databases

| $U_{R[SSN]}$ | V $\mapsto$ D | TID | SSN |
|---|---|---|---|
| | $x \mapsto 1$ | $t_1$ | 185 |
| | $y \mapsto 2$ | $t_2$ | 186 |

| $U_{R[M]}$ | V $\mapsto$ D | TID | M |
|---|---|---|---|
| | $v \mapsto 1$ | $t_1$ | 1 |
| | $w \mapsto 1$ | $t_2$ | 1 |

| $U_{R[N]}$ | TID | N |
|---|---|---|
| | $t_1$ | Smith |
| | $t_2$ | Brown |

| $W$ | V $\mapsto$ D | P |
|---|---|---|
| $\rightarrow$ | $x \mapsto 1$ | .4 |
| | $x \mapsto 2$ | .6 |
| | $y \mapsto 1$ | .7 |
| $\rightarrow$ | $y \mapsto 2$ | .3 |
| $\rightarrow$ | $v \mapsto 1$ | .8 |
| | $v \mapsto 2$ | .2 |
| $\rightarrow$ | $w \mapsto 1$ | .25 |
| | $w \mapsto 2$ | .25 |
| | $w \mapsto 3$ | .25 |
| | $w \mapsto 4$ | .25 |

- We choose possible world $\{x \mapsto 1, y \mapsto 2, v \mapsto 1, w \mapsto 1\}$.
- Probability weight of this world: $.4 * .3 * .8 * .25 = .024$.
- Vertically decomposed version of the chosen possible world.

# Efficient Query Evaluation: Positive relational algebra

Query evaluation under *possible worlds semantics*:

$$\begin{tikzcd} T \arrow[r, "\overline{q}"] \arrow[d, "rep"'] & \overline{q}(T) \arrow[d, "rep"] \\ \{\mathcal{A}_1, \ldots, \mathcal{A}_n\} \arrow[r, "q"] & \{q(\mathcal{A}_1), \ldots, q(\mathcal{A}_n)\} \end{tikzcd}$$

For any positive relational algebra query $q$ over any U-relational database $T$, there exists a positive relational algebra query $\overline{q}$ of polynomial size such that

$$rep(\overline{q}(T)) = \{q(\mathcal{A}_i) \mid \mathcal{A}_i \in rep(T)\}.$$

# Efficient Query Evaluation

The following operations can be mapped to relational algebra over U-relational representations; no new joins are introduced.

$$
\begin{aligned}
[\![R \times S]\!] &:= \pi_{U_R.\overline{VD} \cup U_S.\overline{VD} \to \overline{VD}, sch(R), sch(S)}( \\
&\qquad U_R \bowtie_{U_R.\overline{VD} \text{ consistent with } U_S.\overline{VD}} U_S) \\
[\![\sigma_\phi R]\!] &:= \sigma_\phi(U_R) \\
[\![\pi_{\vec{B}} R]\!] &:= \pi_{\overline{VD}, \vec{B}}(R) \\
[\![R \cup S]\!] &:= U_R \cup U_S \\
[\![\text{poss}(R)]\!] &:= \pi_{sch(R)}(U_R).
\end{aligned}
$$

$S :=$ repair-key$_{\vec{A}@B} R$ for complete relation $R$ is translated as

$$U_S := \pi_{(\vec{A}) \to V, ((sch(R) - \vec{A}) - \{B\}) \to D, sch(R)} U_R$$

with

$$W := W \cup \pi_{(\vec{A}) \to Var, (sch(R) - \vec{A}) - \{B\} \to Dom, B \to P} U_R$$

[Antova, Jansen, K., Olteanu ICDE 2008]

# Operation repair-key

Repair-key starting from a complete relation is just a projection/copying of columns (even though we may create an exponential number of possible worlds)!

Example: Tossing a biased coin twice.

| $R$ | Toss | Face | FProb |
|---|---|---|---|
| | 1 | H | .4 |
| | 1 | T | .6 |
| | 2 | H | .4 |
| | 2 | T | .6 |

$S := \text{repair-key}_{\mathrm{Toss@FProb}}(R)$

| $U_R$ | V | D | Toss | Face | FProb |
|---|---|---|---|---|---|
| | 1 | H | 1 | H | .4 |
| | 1 | T | 1 | T | .6 |
| | 2 | H | 2 | H | .4 |
| | 2 | T | 2 | T | .6 |

| $W$ | V | D | P |
|---|---|---|---|
| | 1 | H | .4 |
| | 1 | T | .6 |
| | 2 | H | .4 |
| | 2 | T | .6 |

# Query Evaluation: Example

Names of possibly married (M=2) persons: $possible(\pi_{Name}(\sigma_{M=2}(S)))$

| $U_{S[Name]}$ | $V \mapsto D$ | TID | Name |
|---|---|---|---|
| | $x_3 \mapsto 1$ | $t_1$ | Smith |
| | $x_5 \mapsto 1$ | $t_2$ | Brown |

| $U_{S[M]}$ | $V \mapsto D$ | TID | M |
|---|---|---|---|
| | $x_3 \mapsto 1$ | $t_1$ | 1 |
| | $x_3 \mapsto 2$ | $t_1$ | 2 |
| | $x_6 \mapsto 1$ | $t_2$ | 1 |
| | $x_6 \mapsto 2$ | $t_2$ | 2 |

Evaluation steps:

1. merge U-relations storing the necessary columns and rewrite:

$$Q' := \pi_{Name}(\sigma_{M=2}(U_{S[Name]} \bowtie_{\psi \wedge \phi} U_{S[M]}))$$

$$\psi := (U_{S[Name]}.V = U_{S[M]}.V \Rightarrow U_{S[Name]}.D = U_{S[M]}.D) \quad \text{... consistency}$$

$$\phi := (U_{S[Name]}.TID = U_{S[M]}.TID) \quad \text{... reverse vertical partitioning}$$

2. feed query to *any* relational query optimizer

| | $V_1 \mapsto D_1$ | $V_2 \mapsto D_2$ | TID | Name | M |
|---|---|---|---|---|---|
| | $x_5 \mapsto 1$ | $x_6 \mapsto 2$ | $t_2$ | Brown | 2 |

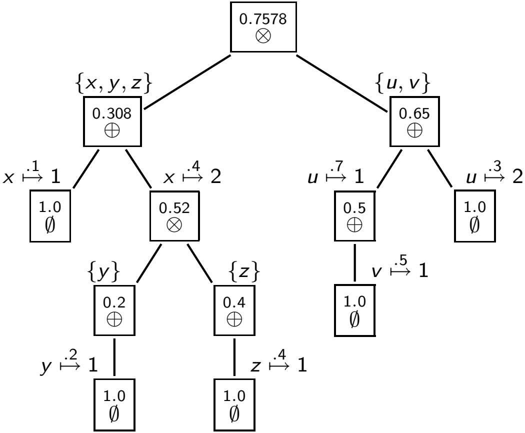
# Exact Confidence Computation

Exact confidence computation is #P-hard. Two techniques implemented:

1. AI heuristic search technique [K. and Olteanu, VLDB 2008].

| $U$ | $V_1$ | $D_1$ | $V_2$ | $D_2$ |
|---|---|---|---|---|
| | $x$ | 1 | $x$ | 1 |
| | $x$ | 2 | $y$ | 1 |
| | $x$ | 2 | $z$ | 1 |
| | $u$ | 1 | $v$ | 1 |
| | $u$ | 2 | $u$ | 2 |

| $W$ | V | D | P |
|---|---|---|---|
| | $x$ | 1 | .1 |
| | $x$ | 2 | .4 |
| | $x$ | 3 | .5 |
| | $y$ | 1 | .2 |
| | $y$ | 2 | .8 |
| | $z$ | 1 | .4 |
| | $z$ | 2 | .6 |
| | $u$ | 1 | .7 |
| | $u$ | 2 | .3 |
| | $v$ | 1 | .5 |
| | $v$ | 2 | .5 |



# Exact Confidence Computation

Exact confidence computation is #P-hard. Two techniques implemented:

1. AI heuristic search technique [K. and Olteanu, VLDB 2008].
   - Also: best-first search for approximate solution.
   - Algorithm optimized for secondary storage.
2. For hierarchical queries, special PTIME techniques.
   - Th. Hierchical queries = maximal PTIME class: dichotomy theorem [Dalvi and Suciu 2004].
   - Special secondary storage operator [Huang, Olteanu, K. ICDE 2009].
   - Generalizations to obtain larger PTIME query fragment via integrity constraints (functional dependencies).

# Approximate Confidence Computation

Approximation algorithm based on MC simulation algorithm for DNF counting [Karp, Luby, Madras].

- FPRAS: gives approximation in linearly many iterations in the size of the database!
  - Importance sampling: relative error bound in terms of size of probability value: essential for conditional confidences, MAP, MLE!
- Provably optimal number of iterations via stopping rule technique/sequential analysis [Dagum, Karp, Luby, Ross].
- Improvement based on [Vazirani]: fractional estimates, lower variance. Basic estimator:
  1. Sample a clause.
  2. Sample a possible world for the clause.
  3. Return $\frac{1}{\#\text{clauses that are true in that world}}$.
- Secondary-storage implementation: doing $n$ MC iterations in bulk using joins etc.
- Generalization to continuous case.

# Efficient Query Evaluation, ctd.

Properties of relational-algebra reduction for positive relational algebra:

- PTIME (even AC0) data complexity
- parsimonious reduction: query plans are hardly more complicated than the input queries $\Rightarrow$ off-the-shelf query optimizers do well.
- preserves the provenance of answer tuples

Remaining operations: Difference, conf, and assert.

- conf can be efficiently approximated by Monte Carlo simulation.
- Difference: In (conditional) confidence computations, universal constraints can often be made existential (see next slides).
- assert is an update operation. In queries, assert can be replaced by conf: computation of conditional probabilities.

# Conditional Confidences; Rewriting Universal Queries

Census example: Find, for each TID $x$ and SSN $y$, the probability

$$\Pr\Big[\underbrace{\exists t \in R\ t.TID = x \wedge t.SSN = y}_{\phi(x,y)} \mid \underbrace{\text{fd: } SSN \rightarrow\ TID}_{\psi}\Big],$$

i.e., find the probability that individual $x$ has SSN $y$ assuming that social security numbers uniquely identify individuals.
Compute the conditional probability as

$$\Pr[\phi \mid \psi] = \frac{\Pr[\phi \wedge \psi]}{\Pr[\psi]} = \frac{\Pr[\phi] - \Pr[\phi \wedge \neg\psi]}{1 - \Pr[\neg\psi]}$$

- $\neg\psi = \exists t, t'\ t.SSN = t'.SSN \wedge t.TID \neq t'.TID$ is existential.
- $\phi(x, y)$, $\phi(x, y) \wedge \neg\psi$, and $\neg\psi$ expressible in *positive* relational algebra.

# A MayBMS Run: Census Example (1)

```
$ create table Census_SSN_0 (tid integer, ssn integer, p float);
$ insert into  Census_SSN_0 values (1, 185, .4);
$ insert into  Census_SSN_0 values (1, 785, .6);
$ insert into  Census_SSN_0 values (2, 185, .7);
$ insert into  Census_SSN_0 values (2, 186, .3);

$ create table Census_SSN as
  repair key (tid) in Census_SSN_0 weight by p;

$ select * from Census_SSN;
 tid | ssn |  p  | _v0 | _d0 | _p0
-----+-----+-----+-----+-----+-----
   1 | 185 | 0.4 |  s1 | 185 | 0.4
   1 | 785 | 0.6 |  s1 | 785 | 0.6
   2 | 185 | 0.7 |  s2 | 185 | 0.7
   2 | 186 | 0.3 |  s2 | 186 | 0.3
```

# A MayBMS Run: Census Example (2)

```
$ create table FD_Violations as
  select S1.ssn
  from   Census_SSN S1, Census_SSN S2
  where  S1.tid < S2.tid and S1.ssn = S2.ssn;
  /* violations of fd ssn->tid */

$ select * from FD_Violations;
 ssn | _v0 | _d0 | _p0 | _v1 | _d1 | _p1
-----+-----+-----+-----+-----+-----+-----
 185 |  s1 | 185 | 0.4 |  s2 | 185 | 0.7
```

# A MayBMS Run: Census Example (3)

```
$ create table TidSsnPosterior as
  select Q1.ssn, p1, p2, p3,
         cast((p1-p2)/(1-p3) as real) as posterior
  from (select tid, ssn, conf() as p1
        from Census_SSN group by tid, ssn) Q1,
       ((select ssn, conf() as p2 from FD_Violations group by ssn)
        union
        ((select ssn, 0 as p2 from Census_SSN_0)
         except
         (select possible ssn, 0 as p2 from FD_Violations))) Q2,
       (select conf() as p3 from FD_Violations) Q3
  where Q1.ssn = Q2.ssn;

$ select * from TidSsnPosterior;
 tid | ssn | p1  |  p2  |  p3  | posterior
-----+-----+-----+------+------+-----------
   1 | 185 | 0.4 | 0.28 | 0.28 |  0.166667
   1 | 785 | 0.6 |    0 | 0.28 |  0.833333
   2 | 185 | 0.7 | 0.28 | 0.28 |  0.583333
   2 | 186 | 0.3 |    0 | 0.28 |  0.416667
```

# A MayBMS Run: Census Example (4)

```
$ select * from TidSsnPosterior;
 tid | ssn | p1  |  p2  |  p3  | posterior
-----+-----+-----+------+------+-----------
   1 | 185 | 0.4 | 0.28 | 0.28 |  0.166667
   1 | 785 | 0.6 |    0 | 0.28 |  0.833333
   2 | 185 | 0.7 | 0.28 | 0.28 |  0.583333
   2 | 186 | 0.3 |    0 | 0.28 |  0.416667

$ select tid, argmax(ssn, posterior) as map
  from   TidSsnPosterior
  group by tid;
 tid | map
-----+-----
   1 | 785
   2 | 185
```

# Complexity Summary: Query Evaluation

| Language Fragment | Complexity | Reference |
|---|---|---|
| *On nonsuccinct representations:* | | |
| RA + conf + assert + possible + choice-of | **in PTIME** (SQL) | [PODS 2008] |
| RA + possible + repair-key | **NP-&coNP-hard** <br> **in P**$^{NP}$ | [SIGMOD 2007] <br> [ICDT 2009] |
| RA + possible$_Q$ + repair-key | **=PHIER** | [ICDT 2009] |
| *On U-relations:* | | |
| Pos.RA + repair-key + possible | **in AC0** | [ICDE 2008] |
| RA + possible | **co-NP-hard** | – |
| Conjunctive queries + conf | **#P-hard** | [Dalvi & Suciu] |
| All operations | **in P**$^{\#P}$ | [PODS 2008] |
| Pos.RA + repair-key + possible + approx.conf + egds | **in PTIME** | [PODS 2008] |

RA = relational algebra

All operations = RA + repair-key + conf + assert + possible

# The MayBMS System

- A modification of the Postgres server backend.
  - Compiles and runs on the same platforms as Postgres.
  - Postgres APIs and middleware can be (readily!) used, e.g. ODBC, JDBC, PLSQL, PHP, ...
  - Full SQL support. Same performance as Postgres on complete data.
- Full support for updates, transactions and recovery.
- Secondary-storage versions of *all* techniques.
- Open source: http://maybms.sourceforge.net
  - Source code of alpha version available for download now (from CVS, not packaged yet).
  - Upcoming release is for discrete finite distributions only; prototype for continuous distributions exists, to be released in Spring.

# Foundations: Expressive Power of Queries

- Reminder: Relational completeness: expressive power of relational algebra.
  - Relational algebra = (domain-independent) first-order logic [Codd].
- World-set algebra: The algebra of this talk minus "conf", plus "possible", difference, and grouping worlds.
  - Th. World-set algebra = second-order logic [K., ICDT 2009].
  - Closed under composition (nontrivial).
  - An expressiveness yardstick for queries on uncertain databases?
- Open: expressiveness of probabilistic world-set algebra.
  - Expresses at least all of #P.
  - Probabilistic dynamic logic?

# Desiderata for a Query Language for Uncertain Data

- genericity – clean language design independent from representation details.
- ability to transform data.
- ability to introduce additional uncertainty (!!!)
  - Need for a data manipulation language (construct the probabilistic database); compositionality.
  - Decision support queries/hypothetical queries.
  - Probabilistic databases: extending the hypothesis space to use evidence.
  - Queries that map from prior to posterior probabilities.
- right degree of expressive power
  - Not too strong and not too weak.
- efficient query evaluation.

Arguably, the MayBMS query language satisfies these desiderata.

# Conclusions

- MayBMS is on the way to becoming a mature probabilistic DBMS.
  - Relevant to real users.
  - Service to the research community: open-source and extensible.
  - First release of the system by late fall, hopefully.
- Many interesting research problems left; this is currently one of the hottest areas in data management!
- For more information, see the overview paper

  C.Koch, "MayBMS: A System for Managing Large Uncertain and Probabilistic Databases", to appear as Chapter 6 of C. Aggarwal, ed., *Managing and Mining Uncertain Data*, Springer, 2008.

  http://www.cs.cornell.edu/bigreddata/maybms/maybms.pdf

# Selected MayBMS2 Publications


- C. Koch, MayBMS: A System for Managing Large Uncertain and Probabilistic Databases, to appear as Chapter 6 of C. Aggarwal, ed., *Managing and Mining Uncertain Data*, Springer, 2008.
- L. Antova, C. Koch, and D. Olteanu. From Complete to Incomplete Information and Back. *SIGMOD 2007*.
- ——— & T. Jansen. Fast and Simple Relational Processing of Uncertain Data. *ICDE 2008*.
- C. Koch. Approximating Predicates and Expressive Queries on Probabilistic Databases. *PODS 2008*.
- C. Koch and D. Olteanu. Conditioning Probabilistic Databases. *VLDB 2008*.
- L. Antova and C. Koch. On APIs for Probabilistic Databases. *MUD 2008*.
- D. Olteanu, J. Huang, and C. Koch. Lazy versus Eager Query Plans for Tuple-Independent Probabilistic Databases. To appear in *ICDE, 2009*.
- C. Koch. A Compositional Query Algebra for Second-Order Logic and Uncertain Databases. To appear in *ICDT, 2009*.


- http://www.cs.cornell.edu/bigreddata/maybms/

# Appendix: Query language syntax and semantics

- The operations are presented, where meaningful, in a probabilistic and a nonprobabilistic version.
- The former can express all queries of the latter, but the latter may be easier to understand at first.
- Probabilistic case: a database represents a finite set $\mathbf{W}$ of possible worlds (relational databases) and their probabilities

$$\mathbf{W} = \{(\mathcal{A}_1, p_1), \ldots, (\mathcal{A}_n, p_n)\}$$

  s.t. $p_1 + \cdots + p_n = 1$.
- Nonprobabilistic case: a database represents a finite set of possible worlds.
- Semantically, each query operation extends the schema and thus each possible world by a new relation.
- Relational algebra operations, e.g. $\sigma_\phi(R)$, prob. case:

$$[\![\sigma_\phi(R)]\!](\mathbf{W}) := \{(\mathcal{A}, \sigma_\phi(R^{\mathcal{A}}), p) \mid (\mathcal{A}, p) \in \mathbf{W}\}$$

# Operation choice-of

| $R^1$ | A | B | C |
|---|---|---|---|
| | a | 1 | c |
| | a | 1 | d |
| | b | 3 | e |

$Pr = .5$ ... (further worlds)

$S :=$ choice-of$_{A@B}(R)$

| $S^{1.1}$ | A | B | C |
|---|---|---|---|
| | a | 1 | c |
| | a | 1 | d |

Pr $= .5 * 1/4 = 1/8$

| $S^{1.2}$ | A | B | C |
|---|---|---|---|
| | b | 3 | e |

Pr $= .5 * 3/4 = 3/8$

... (further worlds)

There must be a functional dependency $R : A \rightarrow B$.

choice-of is expressible using repair-key:

$$\text{choice-of}_{\vec{A}@P}(R) := R \bowtie \text{repair-key}_{\emptyset @P}(\pi_{\vec{A},P}(R)).$$

# Operation choice-of: Example

| $R^{\mathcal{A}}$ | A | B | C |
|---|---|---|---|
| | $a$ | 2 | $c$ |
| | $a$ | 2 | $c'$ |
| | $a'$ | 3 | $c''$ |
| | $a'$ | 3 | $c'''$ |

$\mathrm{Pr} = .1$

| $R^{\mathcal{B}}$ | A | B | C |
|---|---|---|---|
| | $a''$ | 0 | $c^{iv}$ |

$\mathrm{Pr} = .9$

choice-of$_{A@B}(R)$ results in the world-set

| $R^{\mathcal{A}_1}$ | A | B | C |
|---|---|---|---|
| | $a$ | 2 | $c$ |
| | $a$ | 2 | $c'$ |

$\mathrm{Pr} = \frac{2}{2+3} \cdot \frac{.1}{.1} = .4$

| $R^{\mathcal{A}_2}$ | A | B | C |
|---|---|---|---|
| | $a'$ | 3 | $c''$ |
| | $a'$ | 3 | $c'''$ |

$\mathrm{Pr} = \frac{3}{2+3} \cdot \frac{.1}{.1} = .6$

# Operation choice-of

Nonprobabilistic case:

$$[\![\text{choice-of}_{\vec{A}}(R)]\!](\mathbf{W}) := \{\langle \mathcal{A}, \sigma_{\vec{A}=\vec{a}}(R^{\mathcal{A}})\rangle \mid \mathcal{A} \in \mathbf{W},\ \vec{a} \in \pi_{\vec{A}}(R^{\mathcal{A}})\}$$

Probabilistic case:

- Syntax: $\text{choice-of}_{\vec{A}@B}(R)$
- $R$ must satisfy the functional dependency $R : \vec{A} \rightarrow B$ and the $B$ values must be reals $\geq 0$.
- Semantics ($\mathbf{W}$ is a world-set with probabilities):

$$[\![\text{choice-of}_{\vec{A}@B}(R)]\!](\mathbf{W}) := \\ \{(\langle \mathcal{A}, \sigma_{\vec{A}=\vec{a}}(R^{\mathcal{A}})\rangle, p \cdot b/N) \mid (\mathcal{A}, p) \in \mathbf{W},\ (\vec{a}, b) \in \pi_{\vec{A},B}(R^{\mathcal{A}}), \\ N = \sum_B (\pi_{\vec{A},B}(R^{\mathcal{A}})) \neq 0,\ b \neq 0\}$$

- Note: Worlds in which the $B$ column sums up to 0 are dropped (chosen with probability 0).
- The probabilities do not necessarily sum up to 1 anymore: renormalize.

# Operation repair-key

Nonprobabilistic case:

$$[\![\text{repair-key}_{\vec{A}}(R)]\!](\mathbf{W}) := \{\langle \mathcal{A}, \mathrm{Img}(f)\rangle \mid \mathcal{A} \in \mathbf{W},$$
$$\text{function } f : \pi_{\vec{A}}(R^{\mathcal{A}}) \to R^{\mathcal{A}} \text{ such that } f(\vec{a}).\vec{A} = \vec{a}\}$$

$\Rightarrow$ If $\vec{A}$ is a key for $R$, then $[\![\text{repair-key}_{\vec{A}}(R)]\!](\mathbf{W}) = \mathbf{W}$.

Probabilistic case:

- $B \notin \vec{A}$, fd $R : (\mathrm{sch}(R) \backslash B) \to B$
- Semantics:

$$[\![\text{repair-key}_{\vec{A}}(R)]\!](\mathbf{W}) := \Big\{(\langle \mathcal{A}, \mathrm{Img}(f)\rangle, p'/n) \mid (\mathcal{A}, p) \in \mathbf{W},$$
$$\text{function } f : \pi_{\vec{A}}(R^{\mathcal{A}}) \to R^{\mathcal{A}} \text{ such that } f(\vec{a}).\vec{A} = \vec{a},$$
$$p' = p \cdot \prod_{\vec{a} \in \pi_{\vec{A}}(R^{\mathcal{A}})} \frac{f(\vec{a}).B}{\sum_B(\sigma_{\vec{A}=\vec{a}}(R^{\mathcal{A}}))} \neq 0\Big\}$$

s.t.

$$n = \sum_{(\mathcal{A},p) \in [\![\text{repair-key}_{\vec{A}}(R)]\!](\mathbf{W})} p.$$

# Power of repair-key

- Given relation $R$, repair-key($R$) computes as alternative worlds all minimal repairs of a functional dependency.
- Power-world-set operation (w.l.o.g., $A \notin sch(R)$)

  $$pws(R) := \pi_{sch(R)}(\sigma_{A=1}(\text{repair-key}_{sch(R)@P}(R \times \rho_A(\{0,1\}) \times \rho_P(\{1\}))))$$

  Each world is a subset of $R$, and the set of worlds created is the powerset of $R$.
- All repairs of an arbitrary FO constraint $\phi$:

  $$\text{assert}[\phi](pws(R)).$$

# Operation repair-key

Example: Tossing a biased coin twice.

| $R$ | Toss | Face | FProb |
|---|---|---|---|
| | 1 | H | .4 |
| | 1 | T | .6 |
| | 2 | H | .4 |
| | 2 | T | .6 |

$\Pr = 1$

$S := \text{repair-key}_{\mathrm{Toss@FProb}}(R)$ results in four worlds:

| $S^1$ | Toss | Face | FProb |
|---|---|---|---|
| | 1 | H | .4 |
| | 2 | H | .4 |

| $S^2$ | Toss | Face | FProb |
|---|---|---|---|
| | 1 | H | .4 |
| | 2 | T | .6 |

| $S^3$ | Toss | Face | FProb |
|---|---|---|---|
| | 1 | T | .6 |
| | 2 | H | .4 |

| $S^4$ | Toss | Face | FProb |
|---|---|---|---|
| | 1 | T | .6 |
| | 2 | T | .6 |

$$p_1 = 1 \cdot \frac{.4}{.4 + .6} \cdot \frac{.4}{.4 + .6} = .16, \quad p_2 = p_3 = .24, \quad p_4 = .36$$

# Operation conf

- Returns all the tuples in the world-set and their confidences.
- Syntax: $\text{conf}(R)$
- $\text{sch}(\text{conf}(R)) = \text{sch}(R) \cup \{\text{Conf}\}$
- Semantics:

$$[\![\text{conf}(R)]\!](\mathbf{W}) := \{(\langle \mathcal{A}, R_{\text{conf}(R)} \rangle, p) \mid (\mathcal{A}, p) \in \mathbf{W}\}$$

where

$$R_{\text{conf}(R)} = \{\langle t, p \rangle \mid t \in R^*_{\mathbf{W}}, p = P_{\mathbf{W}}(t \in R) > 0\}$$

and

$$R^*_{\mathbf{W}} = \bigcup_{(\mathcal{A},p) \in \mathbf{W}} R^{\mathcal{A}}, \qquad P_{\mathbf{W}}(t \in R) = \sum_{(\mathcal{B},p) \in \mathbf{W}, t \in R^{\mathcal{B}}} p.$$

- Shortcuts: the possible/certain tuples

$$\begin{aligned} \text{possible}(R) &:= \pi_{\text{sch}(R)}(\sigma_{\text{Conf}>0}(\text{conf}(R))) \\ \text{certain}(R) &:= \pi_{\text{sch}(R)}(\sigma_{\text{Conf}=1}(\text{conf}(R))) \end{aligned}$$

# Operation conf: Example

| $R^{\mathcal{A}}$ | A | B |
|---|---|---|
| | a | b |
| | b | c |

.3

| $R^{\mathcal{B}}$ | A | B |
|---|---|---|
| | a | b |
| | c | d |

.2

| $R^{\mathcal{C}}$ | A | B |
|---|---|---|
| | a | c |
| | c | d |

.5

conf: Compute, for each possible tuple, the sum of the weights of the possible worlds in which it occurs.

| $conf(R)$ | $A$ | $B$ | P |
|---|---|---|---|
| | a | b | .5 |
| | a | c | .5 |
| | b | c | .3 |
| | c | d | .7 |

# Operation assert

- Syntax: $\text{assert}_\phi(R)$
- Selects those worlds that satisfy condition $\phi$.
- Semantics (nonprobabilistic case):

$$[\![\text{assert}_\phi(R)]\!](\mathbf{W}) := \{\mathcal{A} \mid \mathcal{A} \in \mathbf{W}, \mathcal{A} \vDash \phi\}$$

- Semantics (probabilistic case):

$$[\![\text{assert}_\phi(R)]\!](\mathbf{W}) := \{(\mathcal{A}, p/p_0) \mid (\mathcal{A}, p) \in \mathbf{W}, \mathcal{A} \vDash \phi\}$$

  where

$$p_0 = \sum_{(\mathcal{A},p)\in\mathbf{W},\mathcal{A}\vDash\phi} p.$$

- $R$ is the name of the relation passed on to the direct superexpression, if there is one.

# Example Query: Conditioning using assert

| $R^1$ | TID | SSN |
|---|---|---|
| | $t_1$ | 185 |
| | $t_2$ | 185 |

| $R^2$ | TID | SSN |
|---|---|---|
| | $t_1$ | 185 |
| | $t_2$ | 186 |

| $R^3$ | TID | SSN |
|---|---|---|
| | $t_1$ | 785 |
| | $t_2$ | 185 |

| $R^4$ | TID | SSN |
|---|---|---|
| | $t_1$ | 785 |
| | $t_2$ | 186 |

$$assert_{fd\ R:SSN \rightarrow TID}$$

This deletes the first of the four worlds and renormalizes the probabilities to sum up to one.

# Coin Example, #1: Prior Probabilities

We pick a coin from a bucket of one double-headed and two fair coins.

| Coins | Type | Count |
|---|---|---|
| | fair | 2 |
| | 2headed | 1 |

| Faces | Type | Face | FProb |
|---|---|---|---|
| | fair | H | .5 |
| | fair | T | .5 |
| | 2headed | H | 1 |

$$\begin{aligned} R &:= \pi_{\mathrm{Type}}(\text{repair-key}_{\emptyset @ \mathrm{Count}}(\mathrm{Coins})) \\ &= \pi_{\mathrm{Type}}(\text{choice-of}_{\mathrm{Type} @ \mathrm{Count}}(\mathrm{Coins})) \end{aligned}$$

The resulting probabilistic database has two possible worlds:

| $R^f$ | Type |
|---|---|
| | fair |

$\Pr = 2/3$

| $R^{dh}$ | Type |
|---|---|
| | 2headed |

$\Pr = 1/3$

# Coin Example, #2: Modeling Coin Faces and Tosses

| Coins | Type | Count |
|---|---|---|
| | fair | 2 |
| | 2headed | 1 |

| Faces | Type | Face | FProb |
|---|---|---|---|
| | fair | H | .5 |
| | fair | T | .5 |
| | 2headed | H | 1 |

| $R^f$ | Type |
|---|---|
| | fair |

Pr $= 2/3$

| $R^{dh}$ | Type |
|---|---|
| | 2headed |

Pr $= 1/3$

$$S := (R \bowtie \mathrm{Faces}) \times \rho_{\mathrm{Toss}}(\{1, 2\})$$

| $S^f$ | Type | Face | FProb | Toss |
|---|---|---|---|---|
| | fair | H | .5 | 1 |
| | fair | T | .5 | 1 |
| | fair | H | .5 | 2 |
| | fair | T | .5 | 2 |

| $S^{dh}$ | Type | Face | FProb | Toss |
|---|---|---|---|---|
| | 2headed | H | 1 | 1 |
| | 2headed | H | 1 | 2 |

# Coin Example, #3: Extending the Hypothesis Space

| $S^f$ | Type | Face | FProb | Toss |
|---|---|---|---|---|
| | fair | H | .5 | 1 |
| | fair | T | .5 | 1 |
| | fair | H | .5 | 2 |
| | fair | T | .5 | 2 |

| $S^{dh}$ | Type | Face | FProb | Toss |
|---|---|---|---|---|
| | 2headed | H | 1 | 1 |
| | 2headed | H | 1 | 2 |

$$T := \pi_{\text{Toss,Face}}(\text{repair-key}_{\text{Toss@FProb}}(S))$$

| $T^{f.HH}$ | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | H |

Pr=1/6

| $T^{f.HT}$ | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | T |

Pr=1/6

| $T^{f.TH}$ | Toss | Face |
|---|---|---|
| | 1 | T |
| | 2 | H |

Pr=1/6

| $T^{f.TT}$ | Toss | Face |
|---|---|---|
| | 1 | T |
| | 2 | T |

Pr=1/6

| $T^{dh}$ | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | H |

Pr=1/3

# Coin Example, #4: Using Evidence

| $T^{f.HH}$ | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | H |

Pr=1/6

| $T^{f.HT}$ | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | T |

Pr=1/6

| $T^{dh}$ | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | H |

Pr=1/3

| $T^{f.TH}$ | Toss | Face |
|---|---|---|
| | 1 | T |
| | 2 | H |

Pr=1/6

| $T^{f.TT}$ | Toss | Face |
|---|---|---|
| | 1 | T |
| | 2 | T |

Pr=1/6

| Ev | Toss | Face |
|---|---|---|
| | 1 | H |
| | 2 | H |

What are the posterior probabilities that a coin of type $x$ was picked, given the evidence $Ev$ ?

$$\Pr[x \in R \mid T = Ev] = \Pr[x \in R \wedge T = Ev] / \Pr[T = Ev]$$

$C_1 := \text{conf}(R \times \text{Algebra}(T = Ev))$; $C_2 := \text{conf}(\text{Algebra}(T = Ev))$;
$Q := \pi_{\text{Type}, C_1.P/C_2.P \to P}(C_1 \times C_2)$

| $C_1$ | Type | P |
|---|---|---|
| | fair | 1/6 |
| | 2headed | 1/3 |

| $C_2$ | P |
|---|---|
| | $1/6+1/3 = 1/2$ |

| $Q$ | Type | P |
|---|---|---|
| | fair | $\frac{1/6}{1/2} = 1/3$ |
| | 2headed | $\frac{1/3}{1/2} = 2/3$ |

# Motivation: Decision Support Queries

| Company_Emp | CID | EID |
|---|---|---|
| | Google | e1 |
| | Google | e2 |
| | Yahoo | e3 |
| | Yahoo | e4 |
| | Yahoo | e5 |

| Emp_Skills | EID | Skill |
|---|---|---|
| | e1 | Web |
| | e2 | Web |
| | e3 | Java |
| | e3 | Web |
| | e4 | Solve problems |
| | e5 | Java |

1. Suppose I choose to buy exactly one company.
2. Assume that one (key) employee leaves that company.
3. If I acquire that company, which skills can I obtain for certain?
4. Now list the possible acquisition targets if I want to guarantee to gain the skill "Web" by the acquisition.

| Result | CID |
|---|---|
| | Google |

# Motivation: Decision Support Queries

| Company_Emp | CID | EID |
|---|---|---|
| | Google | e1 |
| | Google | e2 |
| | Yahoo | e3 |
| | Yahoo | e4 |
| | Yahoo | e5 |

| Emp_Skills | EID | Skill |
|---|---|---|
| | e1 | Web |
| | e2 | Web |
| | e3 | Java |
| | e3 | Web |
| | e4 | Solve problems |
| | e5 | Java |

- Suppose I choose to buy exactly one company.

$$U := \text{choice\_of}_{CID}(\text{Company\_Emp});$$

| U | CID | EID |
|---|---|---|
| | Google | e1 |
| | Google | e2 |

| U | CID | EID |
|---|---|---|
| | Yahoo | e3 |
| | Yahoo | e4 |
| | Yahoo | e5 |

# Motivation: Decision Support Queries

| Company_Emp | CID | EID |
|---|---|---|
| | Google | e1 |
| | Google | e2 |
| | Yahoo | e3 |
| | Yahoo | e4 |
| | Yahoo | e5 |

| Emp_Skills | EID | Skill |
|---|---|---|
| | e1 | Web |
| | e2 | Web |
| | e3 | Java |
| | e3 | Web |
| | e4 | Solve problems |
| | e5 | Java |

- Assume that one (key) employee leaves that company.

$$V := \pi_{1.CID,2.EID}(\text{choice\_of}_{EID}(U) \bowtie_{1.CID=2.CID \wedge 1.EID \neq 2.EID} \text{Company\_Emp})$$

| V | CID | EID |
|---|---|---|
| | Google | e1 |

| V | CID | EID |
|---|---|---|
| | Google | e2 |

| V | CID | EID |
|---|---|---|
| | Yahoo | e3 |
| | Yahoo | e4 |

| V | CID | EID |
|---|---|---|
| | Yahoo | e3 |
| | Yahoo | e5 |

| V | CID | EID |
|---|---|---|
| | Yahoo | e4 |
| | Yahoo | e5 |

# Motivation: Decision Support Queries

| Company_Emp | CID | EID |
|---|---|---|
| | Google | e1 |
| | Google | e2 |
| | Yahoo | e3 |
| | Yahoo | e4 |
| | Yahoo | e5 |

| Emp_Skills | EID | Skill |
|---|---|---|
| | e1 | Web |
| | e2 | Web |
| | e3 | Java |
| | e3 | Web |
| | e4 | Solve problems |
| | e5 | Java |

- If I acquire that company, which skills can I obtain for certain ?

$$W := \text{certain}_{\pi_{CID}}(\pi_{CID,Skill}(V \bowtie Emp\_Skills))$$

| W | CID | Skill |
|---|---|---|
| | Google | Web |

| W | CID | Skill |
|---|---|---|
| | Yahoo | Java |

# Motivation: Decision Support Queries

| Company_Emp | CID | EID |
|---|---|---|
| | Google | e1 |
| | Google | e2 |
| | Yahoo | e3 |
| | Yahoo | e4 |
| | Yahoo | e5 |

| Emp_Skills | EID | Skill |
|---|---|---|
| | e1 | Web |
| | e2 | Web |
| | e3 | Java |
| | e3 | Web |
| | e4 | Solve problems |
| | e5 | Java |

- Now list the possible acquisition targets if I want to guarantee to gain the skill "Web" by the acquisition.

$$\text{possible}(\pi_{CID}(\sigma_{Skill='Web'}(W)))$$

| Result | CID |
|---|---|
| | Google |

# Exact Confidence Computation and Conditioning

Given a tuple $t$ with a set of valuations $S$, compute conf($t$) by partitioning $S$

(a) into independent subsets (*exploit contextual independence*)

(b) by removing variables (*modified Davis-Putnam*)

(c) by removing valuations (*compute equiv. set of pairwise mutex valuations*)

[VLDB 2008] combines (a)-(c) using cost estimation heuristics.

# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$

# Confidence computation example

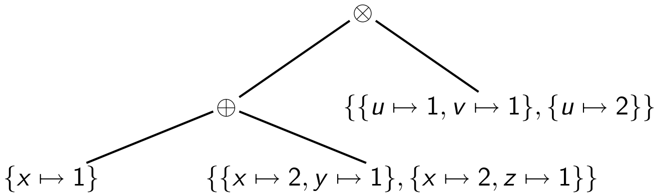
$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$

$$\otimes$$

$$\{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}\} \qquad \{\{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2$$

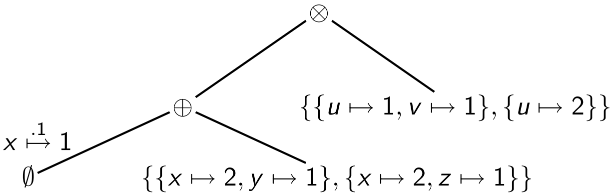
# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$



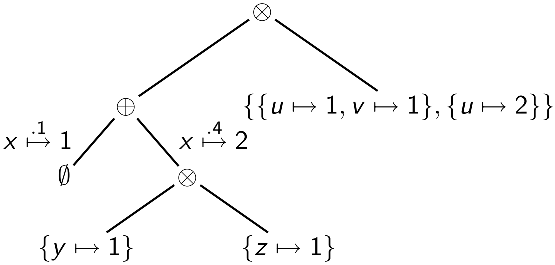
# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$



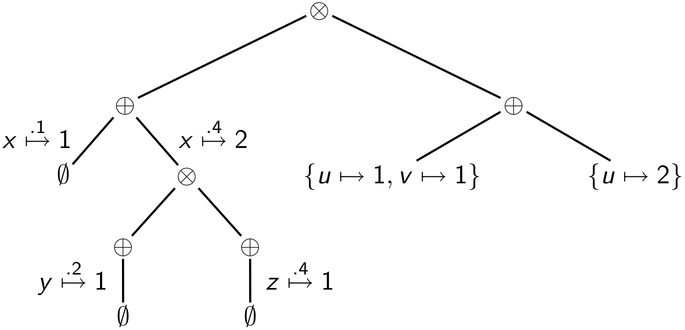
# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$



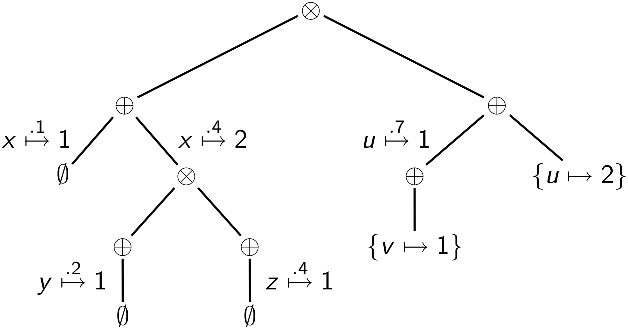
# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$



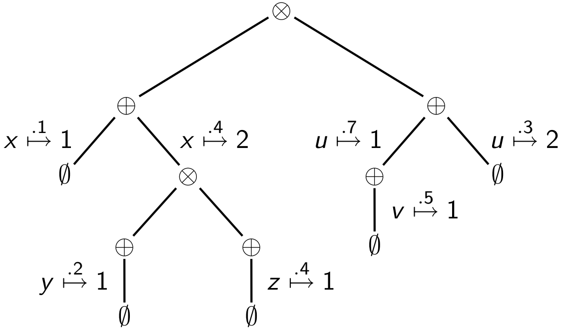
# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$



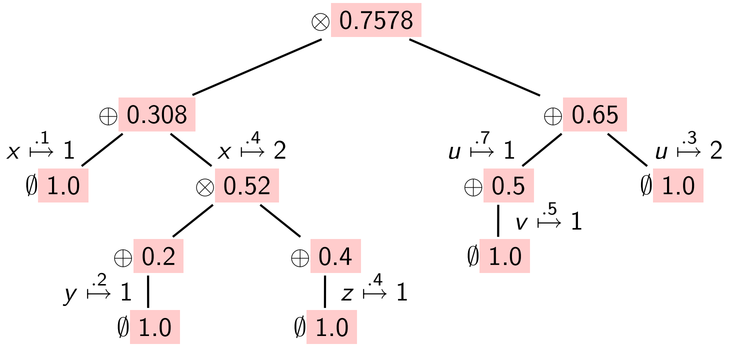
# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$



# Confidence computation example

$$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$$




$$P(S) = 0.7578.$$

# Interaction of Approximation and Selection $\sigma$

Consider the previous query in the census data scenario. What if we only want to select those tuples for which this confidence value is at least .5?

Assuming that conf is computed by approximation, we have a very powerful query language whose results can be very efficiently approximated. But there is a problem.

- The query language is compositional: we may select tuples based on conditions that access approximated (confidence) values.
- A slightly erroneous approximation result may lead to a completely incorrect decision to keep or remove a tuple in a selection.
- How do errors propagate? What is the relationship between approximation and query unreliability?

# Approximating Tuple Confidence : Karp-Luby FPRAS

$F$: set of clauses; $M = \sum_{f \in F} p_f$; $\omega(f)$: set of possible worlds consistent with clause $f$.

## Definition (Karp-Luby Estimator)

*Consider the following definition of random variable $X_i$:*

1. *Choose an $f$ from $F$ with probability $p_f / M$.*
2. *Choose a complete function $f^* \in \omega(f)$ with probability $p_{f*}/p_f$. That is, on each variable $C$ on which $f$ is undefined, chose alternative $x$ with probability $Pr[X = x]$ according to $W$.*
3. *If $f$ is, among the members of $F$ that are consistent with $f^*$, the one of the smallest index, return 1, otherwise return 0.* □

- An unbiased estimator for $\frac{p}{M}$.
- Approximate $p$ by summing up $m$ runs of the estimator, and multiply by $M/m$.
- $(\epsilon, \delta)$-approximation, i.e.

$$Pr\left[|p - \hat{p}| \geq \epsilon \cdot p \mid \hat{p}\right] \leq \delta \qquad \text{if } m \geq \frac{3 \cdot |F| \cdot \log \frac{2}{\delta}}{\epsilon^2}.$$

# Approximating Predicates

Let $B_i(\epsilon)$ be an upper bound on the error $\delta$ of computing approximate value $\hat{p}_i$.

For instance, for the Karp-Luby algorithm, $B_i(\epsilon) = 2 \cdot e^{-\frac{m_i \cdot \epsilon^2}{3 \cdot |F_i|}}$ is such a bound.

## Lemma

*Let $\phi$ be a predicate over unreliable attributes modeled as random variables $p_1, \ldots, p_n$.*
*Assume that the values obtained for these are $\hat{p}_1, \ldots, \hat{p}_k$.*
*If $\epsilon$ is chosen such that the member points of the axis-parallel orthotope defined by the product of open intervals*

$$\left]\frac{\hat{p}_1}{1+\epsilon}, \frac{\hat{p}_1}{1-\epsilon}\right[ \times \cdots \times \left]\frac{\hat{p}_k}{1+\epsilon}, \frac{\hat{p}_k}{1-\epsilon}\right[$$

*all agree on $\phi(\cdot)$, then*

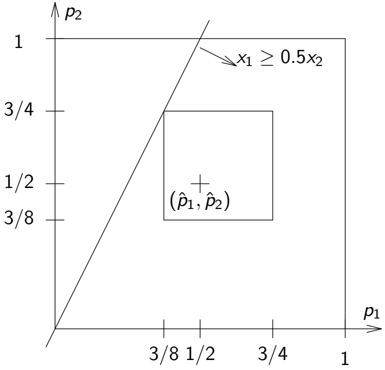
$$Pr[\phi(p_1, \ldots, p_k) \neq \phi(\hat{p}_1, \ldots, \hat{p}_k)] \leq \sum_{i=1}^{k} B_i(\epsilon).$$

# Approximating Predicates

Suppose $\phi(x_1, x_2) = (x_1/x_2 \geq c)$ and $\phi(\hat{p}_1, \hat{p}_2)$ is true. The error probability is $Pr[p_1 < c \cdot p_2 \mid \hat{p}_1 \geq c \cdot \hat{p}_2] \leq 1 - (1 - B(\epsilon))^2$ where

$$\epsilon_\phi(\hat{p}_1, \hat{p}_2) = \frac{\hat{p}_1 - c \cdot \hat{p}_2}{\hat{p}_1 + c \cdot \hat{p}_2}.$$

If $\hat{p}_1 = \hat{p}_2 = c = 1/2$, then $\epsilon = 1/3$ and the maximal orthotope is $[3/8; 3/4]^2$.



# Approximating Predicates: Linear Inequalities

## Theorem (PODS 2008)

*Given predicate*

$$\phi(x_1, \ldots, x_k) = \Big( \sum_{1 \leq i \leq k} a_i \cdot x_i \geq b \Big).$$

*Let*

$$\alpha = \sum_{1 \leq i \leq k} a_i \cdot \hat{p}_i \qquad \beta = \sum_{1 \leq i \leq k} |a_i \cdot \hat{p}_i|.$$

*Then,*

$$\epsilon = \begin{cases} \alpha/\beta & \ldots \quad b = 0 \\ \frac{\beta}{2 \cdot b} + \sqrt{\frac{\beta^2}{4 \cdot b^2} - \frac{\alpha}{b} + 1} & \ldots \quad \textit{otherwise} \end{cases}$$

*minimizes the error bound of the previous Lemma.*

# Approximating Predicates: Variables do not occur twice

**Theorem (PODS 2008)**

*Given a constant* $\epsilon > 0$ *and a predicate*

$$\phi(x_1, \ldots, x_k) = (f(x_1, \ldots, x_k) \geq 0)$$

*where* $f$ *is an algebraic expression built from constants, exactly one occurrence of each of the variables* $x_1, \ldots, x_k$*, and the operations* $+, -, \cdot,$ *and* $/$*. Then, if each of the corner points of the orthotope*

$$\left[\frac{\hat{p}_1}{1+\epsilon}, \frac{\hat{p}_1}{1-\epsilon}\right] \times \cdots \times \left[\frac{\hat{p}_k}{1+\epsilon}, \frac{\hat{p}_k}{1-\epsilon}\right]$$

*agrees with point* $(\hat{p}_1, \ldots, \hat{p}_k)$ *on* $\phi$*, then so do all points in the orthotope.*

We can simply maximize $\epsilon$ by binary search.

# Approximating Predicates: Singularities

## Definition ($\epsilon_0$-singularity)

*A point* $(p_1, \ldots, p_k)$ *is called an* $\epsilon_0$*-singularity if there is a point* $(x_1, \ldots, x_k)$ *such that* $\bigwedge_i |p_i - x_i| \leq \epsilon_0 \cdot p_i$ *and* $\phi(p_1, \ldots, p_k) \neq \phi(x_1, \ldots, x_k)$.

## Example

$\epsilon_0 = 0.05$; $\phi(y) = (y < 0.5)$. If $p = 0.49$ and $x = 0.5$, then $\phi(p), \neg\phi(x)$, and

$$|p - x| = 0.01 \leq \epsilon_0 \cdot p = 0.0245.$$

Therefore, $p$ is an $\epsilon_0$-singularity.

By "zooming" into the area surrounding $p$ not closer than $\epsilon_0$, the orthotope will contain points with disagreeing truth values for $\phi$.

# Approximating Predicates

Algorithm:

**foreach** $i$ **do** $\{\ X_i := 0; \quad m_i := 0;\ \}$

**do** $\{$

  **foreach** $i$ **do** $\{$

    **repeat** $|F_i|$ times **do** $X_i := X_i +$ Karp-Luby-estimator$(F_i)$;

    $m_i := m_i + |F_i|; \quad \hat{p}_i := X_i \cdot M_i / m_i;$

  $\}$

  **if** $\phi(\hat{p}_1, \ldots, \hat{p}_k)$ is true **then** $\epsilon := \max(\epsilon_0, \epsilon_\phi(\hat{p}_1, \ldots, \hat{p}_k))$;

  **else** $\epsilon := \max(\epsilon_0, \epsilon_{\neg\phi}(\hat{p}_1, \ldots, \hat{p}_k))$;

$\}$

**until** $\sum_i B_i(\epsilon) \leq \delta$;

**output** $\phi(\hat{p}_1, \ldots, \hat{p}_k)$ with error $\leq \min(0.5, \sum_i B_i(\epsilon))$

### Theorem (PODS 2008)

*On input of $F_1, \ldots, F_k$, $\epsilon_0$, and $\delta$, if point $(p_1, \ldots, p_k)$ is not an $\epsilon_0$-singularity, then this algorithm computes $\phi(p_1, \ldots, p_k)$ with error probability $\leq \delta$.*

# Example: Approximation and Selections, $\Pr[\phi \mid \psi] \geq 0.5$ ?

$$T = \pi_{TID,S}(\sigma_{P_{\phi\wedge\psi}/P_\psi \geq 0.5}(S))$$

| $S$ | TID | S | $P_{\phi\wedge\psi}$ | $P_\psi$ |
|---|---|---|---|---|
| $s_1$ | $t_1$ | 185 | .12 | .72 |
| $s_2$ | $t_1$ | 785 | .6 | .72 |
| $s_3$ | $t_2$ | 185 | .42 | .72 |
| $s_4$ | $t_2$ | 186 | .30 | .72 |

| $U_T$ | V | D | TID | TID' | S |
|---|---|---|---|---|---|
| | $s_1$ | 1 | $s_1$ | $t_1$ | 185 |
| | $s_2$ | 1 | $s_2$ | $t_1$ | 785 |
| | $s_3$ | 1 | $s_3$ | $t_2$ | 185 |
| | $s_4$ | 1 | $s_4$ | $t_2$ | 186 |

| $W$ | V | D | P |
|---|---|---|---|
| | $s_1$ | 1 | $\leq \delta$ |
| | $s_1$ | 0 | $\geq 1-\delta$ |
| | $s_2$ | 1 | $\geq 1-\delta$ |
| | $s_2$ | 0 | $\leq \delta$ |
| | $s_3$ | 1 | $\geq 1-\delta$ |
| | $s_3$ | 0 | $\leq \delta$ |
| | $s_4$ | 1 | $\leq \delta$ |
| | $s_4$ | 0 | $\geq 1-\delta$ |

- Selection on approximate relations yields an unreliable database.
- Differently from the model of Grädel, Gurevich, Hirsch, the probabilities of the tuples are only upper- or lower-bounded.

# Main Theorem

## Theorem

*Fix $\epsilon_0$ and a query of positive RA[conf, repair-key]. There is a PTIME algorithm that, given $\delta$, computes, for all tuples that do not have an $\epsilon_0$-singularity in their provenance, their membership in the result with error $\leq \delta$.*

Difficulties: Operator tree contains several conf and selection operation on a path:

- Computed $\epsilon$ of a higher selection depends on approximate confidence values below.
- We could use $\epsilon_0$ everywhere, but that would not make use of the fact that larger $\epsilon$ values can be derived from approximation results an predicates.
- Iterative algorithm that moves up and down in the operator tree to refine the approximations until the output tuples have overall reliability at least $1 - \delta$.
- Only confidence computations have to be refined and results are written into the $W$-table; the other operations do not have to be recomputed.

# Uncertain data generator

- Extend TPC-H population generator 2.6 to generate U-relational databases.
- Any generated world has the sizes of relations and join selectivities of the original TPC-H one-world case.
- Parameters: scale (s), uncertainty ratio (x), correlation ratio (z), max alternatives per field (8), drop after correlation (0.25)
- Correlations follow a pattern obtained by chasing egds on uncertain data [ICDE'07].

# Uncertainty and storage

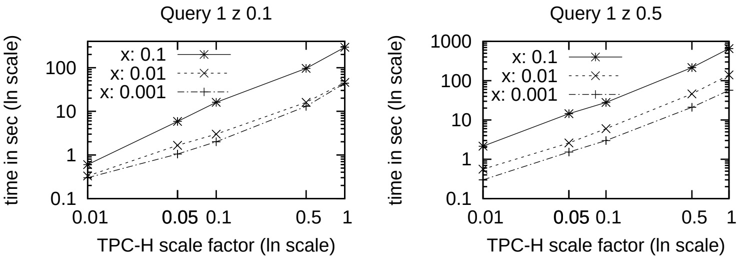
Total number of worlds, max. number of domain values for a variable (Rng), and size in MB of the U-relational database for each of our settings.

| s | z | TPC-H dbsize | #worlds | Rng | dbsize | #worlds | Rng | dbsize | #worlds | Rng | dbsize |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.01 | 0.1 | 17 | $10^{857.076}$ | 21 | 82 | $10^{7955.30}$ | 57 | 85 | $10^{79354.1}$ | 57 | 114 |
| 0.01 | 0.5 | 17 | $10^{523.031}$ | 71 | 82 | $10^{4724.56}$ | 901 | 88 | $10^{46675.6}$ | 662 | 139 |
| 0.05 | 0.1 | 85 | $10^{4287.23}$ | 22 | 389 | $10^{39913.8}$ | 33 | 403 | $10^{396137}$ | 65 | 547 |
| 0.05 | 0.5 | 85 | $10^{2549.14}$ | 178 | 390 | $10^{23515.5}$ | 449 | 416 | $10^{232650}$ | 1155 | 672 |
| 0.10 | 0.1 | 170 | $10^{8606.77}$ | 27 | 773 | $10^{79889.9}$ | 49 | 802 | $10^{793611}$ | 53 | 1090 |
| 0.10 | 0.5 | 170 | $10^{5044.65}$ | 181 | 776 | $10^{46901.8}$ | 773 | 826 | $10^{466038}$ | 924 | 1339 |
| 0.50 | 0.1 | 853 | $10^{43368.0}$ | 49 | 3843 | $10^{400185}$ | 71 | 3987 | $10^{3.97e+06}$ | 85 | 5427 |
| 0.50 | 0.5 | 853 | $10^{25528.9}$ | 214 | 3856 | $10^{234840}$ | 1832 | 4012 | $10^{2.33e+06}$ | 2586 | 6682 |
| 1.00 | 0.1 | 1706 | $10^{87203.0}$ | 57 | 7683 | $10^{800997}$ | 99 | 7971 | $10^{7.94e+06}$ | 113 | 11264 |
| 1.00 | 0.5 | 1706 | $10^{51290.9}$ | 993 | 7712 | $10^{470401}$ | 1675 | 8228 | $10^{4.66e+06}$ | 3392 | 13312 |
| | | x = 0.0 | x = 0.001 | | | x = 0.01 | | | x = 0.1 | | |

- exponentially more succinct than representing worlds individually
- $10^{8 \cdot 10^6}$ worlds need 13 GBs $\approx$ 8 times the size of one world (1.4 GBs)
- case $x = 0$ is the DB generated by the original TPC-H (without uncertainty)

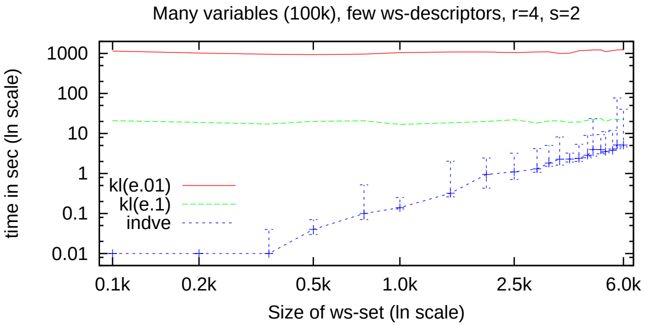
# Evaluation of positive relational algebra queries

$Q_1$: **possible** (**select** o.orderkey, o.orderdate, o.shippriority **from** customer c, orders o, lineitem l **where** c.mktsegment = 'BUILDING'
**and** c.custkey = o.custkey and o.orderkey = l.orderkey
**and** o.orderdate > '1995-03-15' **and** l.shipdate < '1995-03-17')

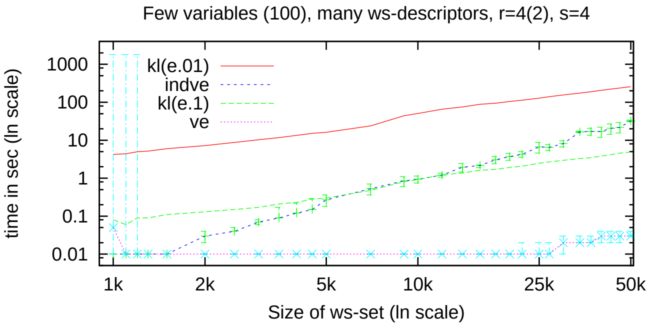



- uncertainty varies from 0.001 to 0.1 → evaluation time up to 6 times slower
- correlation varies from 0.1 to 0.5 → evaluation time up to 3 times slower
- scale varies from 0.01 to 1 → evaluation time up to 400 times slower
  scale=1: the answer size ranges from tens of thousands to tens of millions.

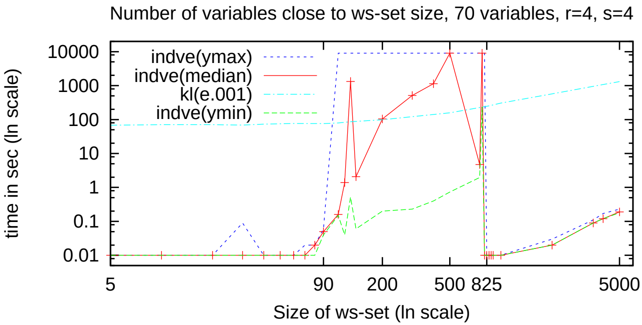
# Confidence Experiments: Many Variables



# Confidence Experiments: Few Variables



# Confidence Experiments: Easy-hard-easy Pattern



# Confidence Experiments: Heuristics