Database Support for Uncertainty

Jun Yang
CPS 296.1, Spring 2007
Sensor Data Processing
With most contents from D. Suciu & N. Dalvi

Probabilistic databases

- Related

Deterministic vs. probabilistic DB

- Databases today are deterministic
  - An item either is in the database or is not
  - A tuple either is in the query answer or is not
- What is a probabilistic database?
  - "An item belongs to the database" is a probabilistic event
  - "A tuple is an answer to the query" is a probabilistic event

So, why now?

- Application pull
  - Need to manage uncertainty in data
  - Many types: non-matching data values, imprecise queries, inconsistent data, misaligned schemas, noisy readings, prediction errors, etc.
- Technology push
  - Processing probabilistic data is fundamentally more complex than processing deterministic data
  - Some previous approaches sidestepped complexity
  - Active area of research

Possible worlds semantics

Attribute domains:

- `int`, `char(30)`, `varchar(55)`, `datetime`

- # values: $2^{32}$, $2^{140}$, $2^{244}$, $2^{64}$

Relational schema:

- `Employee(name:varchar(55), dob:datetime, salary:int)`

- # of tuples: $2^{140} \times 2^{64} \times 2^{244}$

- # of instances: $N$ (= BIG but finite)

Database schema:

- `Employee(. . .), Projects( . . . ), Groups( . . . ), WorksFor( . . . )`

- # of instances: $N$ (= BIG but finite)
The definition

The set of all possible database instances:

\[
\text{INST} = \{I_1, I_2, I_3, \ldots, I_N\}
\]

Definition A probabilistic database \( I^p \) is a probability distribution on \( \text{INST} \)

\[
\Pr : \text{INST} \rightarrow [0,1] \quad \text{s.t.} \quad \sum_{i=1}^N \Pr(I_i) = 1
\]

Definition A possible world is \( I \) s.t. \( \Pr(I) > 0 \)

Example

\[
\begin{array}{ccc}
\text{Customer} & \text{Address} & \text{Product} \\
\hline
\text{John} & \text{Seattle} & \text{Gizmo} \\
\text{John} & \text{Seattle} & \text{Camera} \\
\text{John} & \text{Denver} & \text{Gizmo} \\
\text{Sue} & \text{Seattle} & \text{Gizmo} \\
\end{array}
\]

\[\Pr(I_1) = 1/3\]

\[\Pr(I_2) = 1/12\]

\[\Pr(I_3) = 1/2\]

\[\Pr(I_4) = 1/12\]

Possible worlds = \( \{I_1, I_2, I_3, I_4\} \)

Tuples as events

One tuple \( t \Rightarrow \) event \( t \in I \)

\[
\Pr(t) = \sum_{I : t \in I} \Pr(I)
\]

Two tuples \( t_1, t_2 \Rightarrow \) event \( t_1 \in I \land t_2 \in I \)

\[
\Pr(t_1 \cdot t_2) = \sum_{I : t_1 \in I \land t_2 \in I} \Pr(I)
\]

Tuple correlation

Disjoint

\[\Pr(t_1 \cdot t_2) = 0\]

Negatively correlated

\[\Pr(t_1 \cdot t_2) < \Pr(t_1) \Pr(t_2)\]

Independent

\[\Pr(t_1 \cdot t_2) = \Pr(t_1) \Pr(t_2)\]

Positively correlated

\[\Pr(t_1 \cdot t_2) > \Pr(t_1) \Pr(t_2)\]

Identical

\[\Pr(t_1 \cdot t_2) = \Pr(t_1) = \Pr(t_2)\]

Query semantics

Given a query \( Q \) and a probabilistic database \( I^p \), what is the meaning of \( Q(I^p) \) ?
Query semantics

Semantics 1: Possible Answers
A probability distribution on sets of tuples

\[ \forall A. \Pr(Q = A) = \sum_{I \in \text{INST}} \Pr(I) \]

Semantics 2: Possible Tuples
A probability function on tuples

\[ \forall t. \Pr(t \in Q) = \sum_{I \in \text{INST}} t \in Q(I) \Pr(I) \]

Example: query semantics

<table>
<thead>
<tr>
<th>CameraDenverSue</th>
<th>GizmoDenverSue</th>
<th>CameraSeattleJohn</th>
<th>GizmoSeattleJohn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr(I1) = 1/3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pr(I2) = 1/12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pr(I3) = 1/2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pr(I4) = 1/12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
SELECT DISTINCT x.product
FROM Purchase x, Purchase y
WHERE x.name = 'John'
and x.product = y.product
and y.name = 'Sue'
```

Possible answers semantics:

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>7/12</td>
</tr>
<tr>
<td>Gizmo</td>
<td>5/12</td>
</tr>
</tbody>
</table>

Possible tuples semantics:

\[ \Pr(I1) + \Pr(I2) + \Pr(I3) + \Pr(I4) \]

Tuple prob. \Rightarrow possible worlds

\[ J = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \]

\[ \Pr = \begin{pmatrix} 0.8 & 0.6 & 0.9 \\ p1 & p2 & p3 \\ 1-(1-p1)(1-p2)(1-p3) & 1-(1-p1)p2(1-p3) & 1-(1-p1)(1-p2)p3 \end{pmatrix} \]

\[ I_1 = (1-p1)I_2 + p1(1-p2)I_3 + p1p2I_4 \]

\[ \sum = 1 \]

Summary thus far

Possible worlds semantics

- Very powerful model: any tuple correlations
- Needs separate representation formalism

Query semantics

- Very powerful: every SQL query has semantics
- Very intuitive: from standard semantics
- Two variations
  - Possible answers semantics: precise; can be used to compose queries, difficult user interface
  - Possible tuples semantics: less precise, but simple; sufficient for most apps; cannot be used to compose queries, simple user interface
Representation formalism
- How to represent possible worlds
  - What possible worlds can it represent?
  - What probability distributions on worlds?
  - Is it closed under query application?
- Next: intensional databases, a "complete" formalism

Intensional database
- Atomic event ids: $e_1, e_2, e_3, \ldots$
- Probabilities: $p_1, p_2, p_3, \ldots \in [0,1]$
- Event expressions: $\land, \lor, \neg$

Intensional probabilistic database $J$:
- each tuple $t$ has an event attribute $t.E$

Intensional DB $\Rightarrow$ possible worlds

Closure under operators

Pros and cons of intensional DB
- Expressive—event expression for each tuple
  - Any subset of possible worlds
  - Any probability distribution
- Size of E column can get out of hand
  - Easily on the same order as size of DB (how?)
- Still need to compute the probability of complex event expressions
  - Very expensive!

Probability of boolean expressions

NP = class of problems of the form "is there a witness?" SAT
$\#P$ = class of problems of the form "how many witnesses?" $\#SAT$
**Extensional query evaluation**

- Computing event expressions during query evaluation is expensive.
- Suppose we just need possible tuples semantics for the query anyway.

° Idea: just compute probabilities during query evaluation!

**Correctness**

SELECT DISTINCT x.City FROM Person p x, Purchase y WHERE x.Name = y.Cust and y.Product = 'Gadget'

<table>
<thead>
<tr>
<th>1 - (1 - p_1)(1 - p_2)…</th>
<th>( \Pi )</th>
<th>( \sigma )</th>
</tr>
</thead>
</table>

Data complexity: PTIME

**Safe plans**

- Idea: don’t use any plan; use a safe one that guarantees correct probability computation.
- Intuition: carefully preserve independence as you go.
- Assumptions:
  - Input DB has only independent atomic events (one per tuple).
  - Queries with \( \sigma, \pi, \times \) only, no self joins.
    - Can be extended, though some results not as pretty.

**Testing safety**

- \( \sigma(q) \) is okay.
- \( q \times q' \) is okay (assuming no self joins).
  - Atomic events involved in \( q \) are disjoint from those in \( q' \) ⇒ independence.
- \( \pi_{A_1, \ldots, A_k}(q) \) is the only problematic one.
  - Need to make sure the following functional dependency holds in \( q \) for every relation \( R \) involved in \( q \):
    - \( A_1, \ldots, A_k, R.E \rightarrow \) output attributes of \( q \).
  - That is, among all \( q \) tuples having the same \( A_1, \ldots, A_k \) (whose probabilities we are trying to combine), no atomic event should be associated with more than one such tuple.

**Finding safe plans**

- Example: \( q(D) \) :- \( S(A, B) \), \( T(C, D) \), \( B = C \).
- First, augment the output of \( q \) to keep as many attributes as possible.
  - Won’t work: \( q_1(A, D) \) :- \( S(A, B) \), \( T(C, D) \), \( B = C \).
  - \( q_1 \) is not safe.
  - Works: \( q_2(B, C, D) \) :- \( S(A, B) \), \( T(C, D) \), \( B = C \).
- Then, construct a constraint graph among relations, with edges connecting joining attributes that are NOT both output, and partition connected components.
  - (Typo in conference version)
  - \( q_{BC} = q_1 \circ q_2 \).
  - Continue recursively until we get a plan with no projections, or get stuck.
Complexity

- Sometimes there is no correct extensional plan!
- Data complexity is #P complete

Q_{bad} := R(x, \ldots), S(x, y, \ldots), T(y, \ldots)
where either x or y is not in final output

Theorem: The following are equivalent
- Q has PTIME data complexity
- Q admits an extensional plan (and one finds it in PTIME)
- Q does not have Q_{bad} as a subquery

Extensions

- For queries without safe extensional plans
  - Find an unsafe extensional plan that tries to minimize the error in probability calculation
  - Use an intensional plan and Monte-Carlo simulation to calculate probabilities for complex event expressions
- Additional operators, self joins

Discussion

- How does it compare with Trio/ULDB?
  - ULDB has a larger plan space
  - ULDB computes lineage (analogous to complex events) on demand \(\rightarrow\) avoids unnecessary computation
  - ULDB still faces the challenge of calculating probabilities for complex events
- Who comes up with the independent atomic events?
- Can probabilistic DB be extended to handle sensor data?
- Can we tolerate some uncertainty in the representation of uncertain data?