Database Support for Uncertainty

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

Announcements (Apr. 3)

❖ Project milestone 2
  ❖ I want to know your progress by April 20!
❖ Reading for next Tuesday
  ❖ Posted on Web; no review due

Probabilistic databases

❖ Nilesh Dalvi and Dan Suciu. “Efficient Query Evaluation on Probabilistic Databases.” International Conference on Very Large Data Bases, 2004
❖ Related
  ❖ Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. “ULDBs: Databases with Uncertainty and Lineage.” International Conference on Very Large Data Bases, 2006
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: $2^{52}$, $2^{240}$, $2^{440}$, $2^{64}$

Relational schema:

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

# of tuples: $2^{240} \times 2^{440} \times 2^{64}$
# of instances: $2^{240} \times 2^{440} \times 2^{64}$

Database schema:

Employee(. . .), Projects(. . .), Group(. . .), 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 \(Ip\)

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

\[ I^p = \]

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gadget</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

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

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 \(t_1 \in I \land t_2 \in I\)

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

- **Disjoint** \( \Pr(t_1 \land t_2) = 0 \)
- **Negatively correlated** \( \Pr(t_1 \land t_2) < \Pr(t_1) \Pr(t_2) \)
- **Independent** \( \Pr(t_1 \land t_2) = \Pr(t_1) \Pr(t_2) \)
- **Positively correlated** \( \Pr(t_1 \land t_2) > \Pr(t_1) \Pr(t_2) \)
- **Identical** \( \Pr(t_1 \land t_2) = \Pr(t_1) = \Pr(t_2) \)

Example

\[ \mathcal{P} = \]

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

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

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

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

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

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

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

Query semantics

Given a query \( Q \) and a probabilistic database \( \mathcal{P} \), what is the meaning of \( Q(\mathcal{P}) \)?
Query semantics

Semantics 1: Possible Answers
A probability distributions on sets of tuples
\[ \forall A. \Pr(Q = A) = \sum_{I \in \text{INST}. Q(I) = A} \Pr(I) \]

Semantics 2: Possible Tuples
A probability function on tuples
\[ \forall t. \Pr(t \in Q) = \sum_{I \in \text{INST}. t^2 \in Q(I)} \Pr(I) \]

Example: query semantics

```
<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Denver</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Pr(I_1) = 1/3
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>John</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>John</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>John</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>John</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>John</td>
</tr>
</tbody>
</table>

Pr(I_2) = 1/12

Pr(I_3) = 1/2

Pr(I_4) = 1/12

```

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

Possible answers semantics:

- Possible answers semantics:
  - Answer set | Probability
  - Gizmo, Camera | 1/6
  - Camer
  - Gaza

Possible tuples semantics:

- Possible tuples semantics:
  - Tuple | Probability
  - Camera | 1/12
  - Camera | 7/12
```

Special case

Tuple independent probabilistic database

\[ \text{TUP} = \{t_1, t_2, \ldots, t_M\} = \text{all tuples} \]

\[ \text{pr : TUP} \rightarrow [0,1] \quad \text{No restrictions} \]

\[ \Pr(I) = \prod_{t \in I} \text{pr}(t) \times \prod_{t \notin I} (1 - \text{pr}(t)) \]
Tuple prob. ⇒ possible worlds

\[ p^* = \{ \text{John, Seattle; Sue, Boston; Fred, Boston} \} \]

\[ J = \begin{array}{cc}
\text{Name} & \text{City} \\
\text{John} & \text{Seattle} \\
\text{Sue} & \text{Boston} \\
\text{Fred} & \text{Boston} \\
\end{array} \]

\[ p_1 = 0.8, \quad p_2 = 0.6, \quad p_3 = 0.9, \quad \text{El size}(P^*) J = 2.5 \text{ tuples} \]

\[ \sum = 1 \]

Tuple prob. ⇒ query evaluation

\[ \text{SELECT DISTINCT x.city} \]
\[ \text{FROM Person x, Purchase y} \]
\[ \text{WHERE x.Name = y.Customer} \]
\[ \text{and y.Product = 'Gadget'} \]

```
Customer | Product | Date | pr
--- | --- | --- | ---
John | Gizmo | . . . | q_1
John | Gadget | . . . | q_2
John | Gadget | . . . | q_2
Sue | Camera | . . . | q_3
Sue | Gadget | . . . | q_4
Sue | Gadget | . . . | q_4
Fred | Gadget | . . . | q_5
```

```
<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>((1-p_1)(1-p_2)(1-q_2))</td>
</tr>
<tr>
<td>Boston</td>
<td>((1-p_1)(1-p_2)(1-q_2)) \times (1-p_3, q_5)</td>
</tr>
</tbody>
</table>
```

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

$J = \begin{array}{|c|c|c|} \hline Name & Address & E \\ \hline John & Seattle & e_1 \land (e_2 \lor e_3) \\ Sue & Denver & (e_1 \land e_2) \lor (e_2 \land e_3) \\ \hline \end{array}$

$e_1, e_2, e_3 = 000, 001, 010, 011, 100, 101, 110, 111$

$p_1 (1-p_1) p_3 (1-p_1) p_2 (1-p_1) p_3$

$+ p_1 (1-p_1) (1-p_3) p_2 (1-p_1) p_3$

$+ p_1 (1-p_1) p_3 (1-p_1) p_2 (1-p_1) p_3$

$+ p_1 (1-p_1) p_3 (1-p_1) p_2 (1-p_1) p_3$

$+ p_1 (1-p_1) p_3 (1-p_1) p_2 (1-p_1) p_3$

$+ p_1 (1-p_1) p_3 (1-p_1) p_2 (1-p_1) p_3$
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

Randomly make each variable true with the following probabilities
\[ \Pr(X_i) = p_1, \ Pr(X_2) = p_2, \ldots, \Pr(X_6) = p_6 \]

What is \( \Pr(E) \)??

Theorem [Valiant:1979]
For a boolean expression \( E \), computing \( \Pr(E) \) is \#P-complete

\( \text{NP} = \text{class of problems of the form “is there a witness?”} \text{ SAT} \)
\( \#P = \text{class of problems of the form “how many witnesses?”} \text{ \#SAT} \)
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!

Data complexity: PTIME

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

Wrong!

Correct

Depends on plan!!!
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: $q_1(A, D) :- S(A, B), T(C, D), B = C$
    - $\pi_B(q_1)$ isn’t safe
  - Works: $q_{BC}(B, C, D) :- S(A, B), T(C, D), B = C$
- 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)
  - $q_{BC} = q_1 \pi_B q_2$, where $q_1(B) :- S(A, B)$, and $q_2(C, D) :- T(C, D)$
- Continue recursively until we get a plan with no projections, or get stuck
Complexity

- Sometimes there is no correct extensional plan!

\[ Q_{bad} ::= R(x, \ldots), S(x, y, \ldots), T(y, \ldots) \]
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

Data complexity is \( \#P \) complete

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?