Probabilistic/Uncertain Data Management


Slides based on the Suciu/Dalvi SIGMOD’05 tutorial
Databases Today are Deterministic

• An item either is in the database or is not
  – Database represents a “complete world”

• A tuple either is in the query answer or is not

• This applies to all variety of data models:
  – Relational, E/R, NF2, hierarchical, XML, …
What is a Probabilistic Database?

• “An item belongs to the database” is a probabilistic event
  – Tuple-existence uncertainty
  – Attribute-value uncertainty

• “A tuple is an answer to the query” is a probabilistic event

• Can be extended to all data models; we discuss only probabilistic relational data
Two Types of Probabilistic Data

• Database is deterministic
  Query answers are probabilistic
    – E.g., IR-style/”fuzzy-match” queries
    – Approximate query answers

• Database is probabilistic
  Query answers are probabilistic
Long History

Probabilistic relational databases have been studied from the late 80’s until today:

- Cavallo & Pitarelli: 1987
- Barbara, Garcia-Molina, Porter: 1992
- Lakshmanan, Leone, Ross & Subrahmanian: 1997
- Fuhr & Roellke: 1997
- Dalvi & Suciu: 2004
- Widom: 2005
So, Why Now?

Application pull:
• The need to manage imprecisions in complex data and query-processing tasks

Technology push:
• Advances in query-processing tools/techniques
Application Pull

Need to manage imprecisions in data

- Many types: non-matching data values, imprecise queries, inconsistent data, misaligned schemas, etc.

The quest to manage imprecisions = major driving force in the database community

- Ultimate driver for many research areas: data mining, semistructured data, schema matching, NN queries

Thesis: A large class of data imprecisions can be effectively modeled with probabilities
Technology Push

Processing probabilistic data is \textit{fundamentally more complex} than other data models

- Some previous approaches sidestepped complexity

There exists a rich collection of powerful, non-trivial techniques and results, some old, some very recent, that could lead to practical management techniques for probabilistic databases
Managing Imprecisions: Applications

1. *Ranking query answers*
2. *Record linkage*
3. Quality in data integration
4. Inconsistent data / Data cleaning
5. Information disclosure
1. Ranking Query Answers

Database is deterministic

The query returns a \textit{ranked list of tuples}

\begin{itemize}
  \item Based on some application-specific \textit{ranking function}
  \item User interested in top-k answers
\end{itemize}
The Empty Answers Problem

Query is overspecified: no answers
Example: try to buy a house in SF…

```
SELECT *
FROM Houses
WHERE bedrooms = 3
  AND style = 'craftsman'
  AND district = 'Noe Valley'
  AND price < 400000
```

… good luck!
Ranking:
Compute a similarity score between a tuple and the query

\[
Q = \text{SELECT } * \\
\text{FROM } R \\
\text{WHERE } A_1 \sim v_1 \text{ AND } \ldots \text{ AND } A_m \sim v_m
\]

Query is a vector: \( Q = (v_1, \ldots, v_m) \)

Tuple is a vector: \( T = (u_1, \ldots, u_m) \)

Rank tuples by their TF/IDF similarity to the query \( Q \)

“Expanded” query answer – Includes partial matches

[Agrawal, Chaudhuri, Das, Gionis 2003]
Similarity Predicates in SQL

Beyond a single table:
“Find the good deals in a neighborhood !”

```sql
SELECT *
FROM Houses x
WHERE x.bedrooms ~ 3 AND x.style ~ 'craftsman' AND x.price ~ 600k
    AND NOT EXISTS
    (SELECT *
     FROM Houses y
     WHERE x.district = y.district AND x.ID != y.ID
     AND y.bedrooms ~ 3 AND y.style ~ 'craftsman' AND y.price ~ 600k

Users specify similarity predicates with ~
System combines atomic similarities using probabilities
```
Types of Similarity Predicates

• String edit distances:
  – Levenstein distance, Q-gram distances

• TF/IDF scores

• Ontology distance / semantic similarity:
  – Wordnet

• Phonetic similarity:
  – SOUNDEX

Keyword Searches in Databases

Goal:
• Users want to search via keywords
• Do not know the schema

Techniques:
• Matching objects may be scattered across physical tables due to normalization; need on the fly joins
• Score of a tuple = number of joins, plus “prestige” based on in-degree
Summary on Ranking Query Answers

Types of imprecision addressed:
Data is precise, query answers are imprecise:
• User has limited understanding of the data
• User has limited understanding of the schema
• User has personal preferences

Probabilistic approach would…
• Principled semantics for complex queries
• Integrate well with other types of imprecision
2. Record Linkage

Determine if two data records describe same object

Scenarios:

- Join/merge two relations
- Remove duplicates from a single relation
- Validate incoming tuples against a “reference set”
Application: Data Cleaning, ETL

- Merge/purge for *large* databases, by sorting and clustering
  
  [Hernandez, Stolfo: 1995]

- Use of dimensional hierarchies in data warehouses and exploit co-occurrences
  
  [Ananthakrishna, Chaudhuri, Ganti: 2002]

- Novel similarity functions that are amenable to indexing
  
  [Chaudhuri, Ganjam, Ganti, Motwani: 2002]

- Declarative language to combine cleaning tasks
  
  [Galhardas et al.: 2001]
Application: Data Integration

WHIRL

• All attributes in in all tables are of type *text*
• Datalog queries with two kinds of predicates:
  – Relational predicates
  – Similarity predicates $X \sim Y$

[matches two sets on the fly, but not really a “record linkage” application.]
Example 1:

\[ Q_1(*) \leftarrow P(\text{Company}_1, \text{Industry}_1), \]
\[ Q(\text{Company}_2, \text{Website}), \]
\[ R(\text{Industry}_2, \text{Analysis}), \]
\[ \text{Company}_1 \sim \text{Company}_2, \]
\[ \text{Industry}_1 \sim \text{Industry}_2 \]

Score of an answer tuple = product of similarities
WHIRL

Example 2 (with projection):

\[ Q_2(\text{Website}) \ :- \ P(\text{Company}_1, \text{Industry}_1), \]
\[ Q(\text{Company}_2, \text{Website}), \]
\[ R(\text{Industry}_2, \text{Analysis}), \]
\[ \text{Company}_1 \sim \text{Company}_2, \]
\[ \text{Industry}_1 \sim \text{Industry}_2 \]

Support(t) = set of tuples supporting the answer t

score(t) = 1 - \( \prod_{s \in \text{Support}(t)} (1 - \text{score}(s)) \)
Summary on Record Linkage

Types of imprecision addressed:
Same entity represented in different ways
- Misspellings, lack of canonical representation, etc.

A probability model would…
- Allow system to use the match probabilities: cheaper, on-the-fly
- But need to model complex probabilistic correlations: is one set a reference set ("high-quality" items)? how many duplicates are expected?
Other Applications

• Data lineage + accuracy: Trio [Widom:2005]
• Sensor data [Deshpande, Guestrin, Madden:2004]
• Personal information management
  Semex [Dong & Halevy:2005, Dong, Halevy, Madhavan:2005]
  Heystack [Karger et al. 2003], Magnet [Sinha & Karger:2005]
• Using statistics to answer queries [Dalvi & Suciu:2005]
Applications: Summary

Common in these applications:

• Data in database and/or in query answer is uncertain, ranked; sometimes probabilistic

Need for common probabilistic model

• Main benefit: uniform, principled approach to imprecision

• Other benefits:
  – Handle complex queries (instead of single table TF/IDF)
  – Cheaper/better solutions through improved probabilistic techniques
Probabilistic Data Semantics

- The possible worlds model
- Query semantics
Possible Worlds Semantics

Attribute domains:

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

# values: $2^{32}$, $2^{120}$, $2^{440}$, $2^{64}$

Relational schema:

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

# of tuples: $2^{440} \times 2^{64} \times 2^{23}$

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* \( \text{IP} \) is a probability distribution on \( \text{INST} \)

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

**Definition** A *possible world* is \( I \) s.t. \( \text{Pr}(I) > 0 \)
$I^p =$

**Example**

<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>Gizmo</td>
</tr>
<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\}
Tuples as Events

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

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

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

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

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

Semantics 2: Possible Tuples
A probability function on *tuples*

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

**Possible answers** semantics:

<table>
<thead>
<tr>
<th>Answer set</th>
<th>Probability</th>
<th>Pr(I₁)</th>
<th>Pr(I₂)</th>
<th>P(I₃) + P(I₄)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo, Camera</td>
<td>1/3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gizmo</td>
<td>1/12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>7/12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Possible tuples** semantics:

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
<th>Pr(I₁)+P(I₃) + P(I₄)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>11/12</td>
<td>Pr(I₁)+P(I₃) + P(I₄)</td>
</tr>
<tr>
<td>Gizmo</td>
<td>5/12</td>
<td>Pr(I₁)+Pr(I₂)</td>
</tr>
</tbody>
</table>

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

Pr(I₁) = 1/3
Pr(I₂) = 1/12
Pr(I₃) = 1/2
Pr(I₄) = 1/12

<table>
<thead>
<tr>
<th>Name</th>
<th>City</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>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Camera</td>
</tr>
<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>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Gadget</td>
</tr>
<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>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>John</td>
<td>Boston</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Possible-Worlds Semantics: Summary

Very powerful model

– **Complete**: Can capture *any* instance distribution, *any* tuple correlations

Intuitive, clean formal semantics for *any* SQL query

– Translates to queries over deterministic instances
Possible Worlds Semantics: Summary (contd.)

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
Possible Worlds Semantics: Summary (contd.)

*Not* very useful as a representation or implementation tool

- HUGE number of possible worlds!

Need more effective representation formalisms

- Something that users can understand/explore
- Allow more efficient query execution
  - Avoid “possible worlds explosion”
- *Perhaps* giving up completeness