Probabilistic/Uncertain Data Management

1. *Dalvi, Suciu. "Efficient query evaluation on probabilistic databases", VLDB Jrnl, 2004*
2. *Das Sarma et al. "Working models for uncertain data", ICDE'2006.*

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 *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 *ranked list of tuples*

- Based on some application-specific *ranking function*
- User interested in top-k answers

[Agrawal,Chaudhuri,Das,Gionis 2003]

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 !

[Agrawal,Chaudhuri,Das,Gionis 2003]

Ranking:

Compute a similarity score between a tuple and the query

Q = SELECT *
FROM R
WHERE $A_1 \sim v_1$ AND … 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

[Motro:1988,Dalvi&Suciu:2004]

Similarity Predicates in SQL

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

```
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 <u>probabilities</u>

Types of Similarity Predicates

- String edit distances:
  - Levenstein distance, Q-gram distances
- TF/IDF scores
- Ontology distance / semantic similarity:
  - Wordnet
- Phonetic similarity:
  - SOUNDEX

[Theobald&Weikum:2002, Hung,Deng&Subrahmanian:2004]

[Hristidis&Papakonstantinou’2002,Bhalotia et al.2002]

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

[Cohen: Tutorial]

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]

[Cohen:1998]

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 ~ Y

Matches two sets on the fly, but
not really a “record linkage” application.

[Cohen:1998]

WHIRL

datalog

Example 1:

$$\begin{aligned} Q_1(*) :- \; & P(Company_1, Industry_1), \\ & Q(Company_2, Website), \\ & R(Industry_2, Analysis), \\ & \quad Company_1 \sim Company_2, \\ & \quad Industry_1 \sim Industry_2 \end{aligned}$$

Score of an answer tuple = product of similarities

[Cohen:1998]

WHIRL

Example 2 (with projection):

$$
\begin{aligned}
Q_2(\text{Website}) \text{ :- } & P(\text{Company}_1, \text{Industry}_1), \\
& Q(\text{Company}_2, \text{Website}), \\
& R(\text{Industry}_2, \text{Analysis}), \\
& \quad \text{Company}_1 \sim \text{Company}_2, \\
& \quad \text{Industry}_1 \sim \text{Industry}_2
\end{aligned}
$$

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

Depends on query plan !!

$$\text{score}(t) = 1 - \prod_{s \in \text{Support}(t)} (1-\text{score}(s))$$

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: $2^{440} \times 2^{64} \times 2^{23}$

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

Database schema:

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

# of instances: N (= BIG but finite)

The Definition

The set of all possible database instances:

$$\boxed{\mathrm{INST} = \{I_1, I_2, I_3, \ldots, I_N\}}$$

will use Pr or I^p interchangeably

Definition A *probabilistic database* I^p is a probability distribution on INST

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

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

Example

$I^p =$

Customer	Address	Product
John	Seattle	Gizmo
John	Seattle	Camera
Sue	Denver	Gizmo

$Pr(I_1) = 1/3$

Customer	Address	Product
John	Boston	Gadget
Sue	Denver	Gizmo

$Pr(I_2) = 1/12$

Customer	Address	Product
John	Seattle	Gizmo
John	Seattle	Camera
Sue	Seattle	Camera

$Pr(I_3) = 1/2$

Customer	Address	Product
John	Boston	Gadget
Sue	Seattle	Camera

$Pr(I_4) = 1/12$

Possible worlds = $\{I_1, I_2, I_3, I_4\}$

Tuples as Events

One tuple t $\Rightarrow$ event $t \in I$

$$\boxed{Pr(t) = \sum_{I:\, t \in I} Pr(I)}$$

Two tuples t_1, t_2 $\Rightarrow$ event $t_1 \in I \wedge t_2 \in I$

$$\boxed{Pr(t_1\ t_2) = \sum_{I:\, t_1 \in I \wedge t_2 \in I} Pr(I)}$$

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*

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

Semantics 2: Possible Tuples

A probability function on *tuples*

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

Example: Query Semantics

$Purchase^p$

Name	City	Product
John	Seattle	Gizmo
John	Seattle	Camera
Sue	Denver	Gizmo
Sue	Denver	Camera

$Pr(I_1) = 1/3$

Name	City	Product
John	Boston	Gizmo
Sue	Denver	Gizmo
Sue	Seattle	Gadget

$Pr(I_2) = 1/12$

Name	City	Product
John	Seattle	Gizmo
John	Seattle	Camera
Sue	Seattle	Camera

$Pr(I_3) = 1/2$

Name	City	Product
John	Boston	Camera
Sue	Seattle	Camera

$Pr(I_4) = 1/12$

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

Possible answers semantics:

Answer set	Probability	
Gizmo, Camera	1/3	$Pr(I_1)$
Gizmo	1/12	$Pr(I_2)$
Camera	7/12	$P(I_3) + P(I_4)$

Possible tuples semantics:

Tuple	Probability	
Camera	11/12	$Pr(I_1)+P(I_3) + P(I_4)$
Gizmo	5/12	$Pr(I_1)+Pr(I_2)$

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*