Querying Probabilistic Data

Dan Suciu
University of Washington
Probabilistic Databases

Traditional Databases: deterministic data
Accounting, inventory, …

Foundations: First Order Logic (FO) [Codd’71]

New Challenge: uncertain data
Business intelligence, information extraction, fuzzy object matching, moving objects, data integration (missing/conflicting data), data anonymization,

New Foundations: FO + Probabilities
The Landscape of ProbDBs

Long history

Wong ’82
Shoshani’82
Cavallo&Pittarelli’87
Barbara’92
Lakshmanan’97
Fuhr&Roellke’97
Zimanyi’97

Core challenge:
Computational complexity
of FO + probabilities

Today (incomplete list):

Stanford (Trio)
UW (MystiQ)
Cornell (MayBMS)
Oxford (MayBMS)
U.of Maryland
IBM Almaden (MCDB)
Rice (MCDB)
U. of Waterloo
UBC
U. of Florida
Purdue University
U. of Wisconsin
ProbDB at UW

Faculty: M. Balazinska, D. Suciu

Former Students
Gerome Miklau
Nilesh Dalvi
Christopher Re
Julie Letchner
Vibhor Rastogi

Current Students/Postdocs
Abhay Jha
Wolfgang Gatterbauer
Nodira Koussainova
Alexandra Meliou
Paris Koutris

Looking for a hot research area?
Top 10 ranked nationally in CS
Collegial and collaborative culture
Strong faculty
27 PYI/PFF/Career awards, 4 PECASE Awards,
14 Sloan Fellowships, 2 Packard Fellowships
MacArthur Award, TR-35 Awards, National Academy Members, CRA Habermann Award,….
Many nationally recognized programs
Graphics and vision, systems and networks, AI, HCI, databases, security, architecture, …
Record of producing great students
Dozens of students on top CS faculties nationally (including Penn, CMU, Berkeley, Princeton, ….)
Top PhD students in the country in several years
Strong connections to local industry
Microsoft, Intel, Google, Amazon
Successful history of tech transfer
Technologies (e.g., WebCrawler[AOL], MetaCrawler[Go2net], Hyper-threading[Intel, IBM], Photosynth [MSFT]
Companies (e.g., Impinj [RFID], Skytap [Cloud computing],
  * Farecast [Bing Travel], …)
Outline

Introduction

Problem Statement

Safe Query Plans

A General Algorithm
Sample Application: RFID Data

Noise makes RFID readings un-queryable

Deterministic data: noisy, brittle

<table>
<thead>
<tr>
<th>Tag</th>
<th>Loc</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Hall4</td>
<td>3</td>
</tr>
<tr>
<td>Joe</td>
<td>Office3</td>
<td>4</td>
</tr>
<tr>
<td>Joe</td>
<td>Hall4</td>
<td>7</td>
</tr>
<tr>
<td>Joe</td>
<td>Office3</td>
<td>7</td>
</tr>
<tr>
<td>Bob</td>
<td>....</td>
<td>...</td>
</tr>
</tbody>
</table>
Sample Application: RFID Data

Complex Query: “Who brought a laptop to the meeting?”
Many Other Applications

Information extraction
Data integration
Object and schema matching
Business analytics
Data anonymization

**Needed:** Scalable query processing on probabilistic data
Probabilistic Databases in this Talk

Data: Tuple-Independent Databases

Queries: Unions of Conjunctive Queries
Every tuple t in D = independent random variable

R(x) = “x was at the meeting”

S(x,y) = “x carried laptop y”

<table>
<thead>
<tr>
<th>R(A)</th>
<th>S(A,B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>a1</td>
<td>.4</td>
</tr>
<tr>
<td>a2</td>
<td>.2</td>
</tr>
<tr>
<td>a3</td>
<td>.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>.7</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>b1</td>
<td>.4</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>.2</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>B2</td>
<td>.5</td>
<td></td>
</tr>
</tbody>
</table>
Unions of Conjunctive Queries (UCQ)

\[ Q := R(x_1, \ldots, x_k) \mid \exists x. Q \mid Q_1 \land Q_2 \mid Q_1 \lor Q_2 \]

“Did anyone bring a laptop to the meeting?”

\[ \exists x. \exists y. R(x) \land S(x, y) \equiv R(x), S(x, y) \]

Traditional database: \( D \models Q \) (true/false)

Probabilistic database: \( P(Q) \) in \([0,1]\)
Query Q + Database D = Lineage*

FQ

Every tuple t in D = Boolean variable Xt

**Def** The *lineage* FQ is a Boolean Expression in the variables Xt, saying when Q is true on D

(Example on next slide)

*Called PosBool in [Green’07, Tannen’10]*
Example

\[ R(x) = \text{"x was at the meeting"} \]

\[ S(x,y) = \text{"x carried laptop y"} \]

\[ Q = \text{"Did anyone bring a laptop to the meeting?"} \]

\[ Q = R(x), S(x,y) \]

\[ FQ = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \lor X_2 Y_4 \lor X_2 Y_5 \]
SQL Engines are Great At…

…evaluating queries!
Hash joins, merge joins, younameit joins…
Indexes, materialized views
Query plan: joins, aggregates, selections…
Statistic-based query optimization
Parallel query processing, map/reduce
Problem. Extend the SQL engine to compute a query Q on a probabilistic database D: \( P(Q) \)

Same as computing \( P(FQ) \)
This is general probabilistic inference

What is special about databases?
Vardi, *The Complexity of Relational Query Languages*, STOC 1982

Query $Q$, database $D$

Data complexity:
fix $Q$, complexity $= f(D)$

Query complexity:
fix $D$, complexity $= f(Q)$

Combined complexity:
complexity $= f(D,Q)$

Moshe Vardi
2008 ACM SIGMOD Codd Innovation Award
Dichotomy Theorem.
Fix a UCQ \( Q \). Then:

- either \( P(Q) \) is in PTIME
- or \( P(Q) \) is hard for FP\#P

This also gives a dichotomy for computing \( P(FQ) \), based on its structure.
Outline

Introduction

Problem Statement

Safe Query Plans

A General Algorithm
Conjunctive Queries w/o Self-Joins

“Conjunctive Query” = without $\lor$

“W/o self-joins” = each relation occurs once

We start with these

\[ Q = R(x), S(x,y) \]

\[ Q = S(x,y), S(y,z) \]
\[ Q = \exists x. \exists y. R(x), S(x, y) \]

\[ P(Q) = 1 - \{1 - p1\} \times [1 - (1 - q1) \times (1 - q2)] \times \]
\[ \{1 - p2\} \times [1 - (1 - q3) \times (1 - q4) \times (1 - q5)] \]

\[ FQ = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \lor X_2 Y_4 \lor X_2 Y_5 \]
\[ = X_1 (Y_1 \lor Y_2) \lor X_2 (Y_3 \lor Y_4 \lor Y_5) \]

### R(A)

```
<table>
<thead>
<tr>
<th>A</th>
<th>P</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>p1</td>
<td>X1</td>
</tr>
<tr>
<td>a2</td>
<td>p2</td>
<td>X2</td>
</tr>
<tr>
<td>a3</td>
<td>p3</td>
<td>X3</td>
</tr>
</tbody>
</table>
```

### S(A,B)

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>q1</td>
<td>Y1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
<td>q2</td>
<td>Y2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>q3</td>
<td>Y3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
<td>q4</td>
<td>Y4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
<td>q5</td>
<td>Y5</td>
</tr>
</tbody>
</table>
```

"Read-once"
T1(A,B)

Join operator

R(A)

S(A,B)
\[ T_2(A) \]

\[ S(A, B) \]

\[ \Pi A \]

Projection w/ duplicate elimination
Q = \exists x. \exists y. R(x), S(x, y)

1 - (1 - p_1 q_1)(1 - p_1 q_2)(1 - p_2 q_3)(1 - p_2 q_4)(1 - p_2 q_5)

\[ 1 - \{1 - p_1 [1 - (1 - q_1)(1 - q_2)]\}^* \{1 - p_2 [1 - (1 - q_4)(1 - q_5)(1 - q_6)]\} \]

T_1(A, B)

T_2(A)

P_\Phi

\Pi A

\Pi \Phi

1 - (1 - q_1)(1 - q_2)

1 - (1 - q_4)(1 - q_5)(1 - q_6)
Summary

Some queries can be computed efficiently, and pushed in the database engine

Example: $Q = R(x), S(x, y)$
Performance $\approx$ native SQL

But other queries are hard

Example: $H_0 = R(x), S(x, y), T(y)$ is hard for $P\#P$
Performance $\approx 102 - 103$ worse
Root Variables, Hierarchical Queries

Def. \( x \) is *root variable* in \( \exists x. Q \), if it occurs in all atoms.

Def. \( Q \) is *hierarchical* if all variables are root variables.

\[
\exists x. R(x) \land (\exists y. S(x,y))
\]

\[
R(x), S(x,y), T(y)
\]

Hierarchical

\[
\exists x. R(x) \land (\exists y. S(x,y))
\]

Not hierarchical
A Simple Dichotomy Theorem

Dichotomy Theorem
If Q is a *conjunctive query without self-joins* then:
- Q is hierarchical \( \implies \) FQ is read-once, Q has safe plan
- Q is non-hierarchical \( \implies \) P(Q) is hard for FP#P

Dichotomy into PTIME/FP#P based on the syntax

\[
Q = R(x), S(x, y)
\]
Read-once; PTIME

\[
H_0 = R(x), S(x, y), T(y)
\]
Hard for FP#P
Outline

Introduction

Problem Statement

Safe Query Plans

A General Algorithm
CQ with Self-Joins

\[ QJ = q_1, q_2 = R(x_1), S(x_1, y_1), T(x_2), S(x_2, y_2) \]

\[ FJ = [X_1(Y_1 \lor Y_2) \lor X_2 (Y_3 \lor Y_4 \lor Y_5)] \land [Z_1(Y_1 \lor Y_2) \lor Z_2 (Y_3 \lor Y_4 \lor Y_5)] \]

\[ QU = q_1 \lor q_2 = R(x_1), S(x_1, y_1) \lor T(x_2), S(x_2, y_2) \]

\[ FU = (X_1 \lor Z_1)(Y_1 \lor Y_2) \lor (X_2 \lor Z_2)(Y_3 \lor Y_4 \lor Y_5) \]

\[ P(QJ) = P(q_1, q_2) = P(q_1) + P(q_2) - P(q_1 \lor q_2) \]
In order to handle self-joins, we needed $V$

$CQ = \text{not a natural class to study}$

$UCQ = \text{the natural class to study}$

Will give the algorithm next as four rules
Four Rules for Query Evaluation

Rule 1: Inclusion/Exclusion Formula
\[ P(Q_1 \land Q_2 \land Q_3) = P(Q_1) + P(Q_2) + P(Q_3) \]
\[ - P(Q_1 \lor Q_2) - P(Q_1 \lor Q_3) - P(Q_2 \lor Q_3) + P(Q_1 \lor Q_2 \lor Q_3) \]

Note: this is the dual of the more popular:
\[ P(Q_1 \lor Q_2 \lor Q_3) = \ldots \]
Precondition:
z needs to be a “separator variable”:
appears in all atoms and on same position

**Rule 2: Independent Project**

$$P(\exists z. Q) = 1 - (1 - P(Q[a1/z])) \times (1 - P(Q[a2/z])) \times \ldots$$

Where a1, a2, a3, … an, are all the constants in the database; the active domain.
Four Rules for Query Evaluation

Rule 3: Independent Join
\[ P(Q1 \land Q2) = P(Q1) \times P(Q2) \]

Rule 4: Independent Union
\[ P(Q1 \lor Q2) = 1 - (1 - P(Q1)) \times (1 - P(Q2)) \]

If Q1, Q2 are independent (no common symbols)
Summary

Four rules:
Every rule reduces $P(Q)$ to a simpler $P(Q')$
When $Q$ is ground tuple, then lookup $P(Q)$

If we succeed to compute $P(Q)$ using these rules, then $P(Q)$ is in PTIME

If we fail to apply the rules, is $P(Q)$ hard?
When No Rule Applies

Theorem. These queries are hard for FP#P

Proof: difficult reduction from positive-partitioned 2DNF, which was shown to be #P-hard by Ball&Provan
When the Rules Do Apply

DNF

QV = R(x1),S(x1,y1) ∨ S(x2,y2),T(y2) ∨ R(x3),T(y3)
When the Rules Do Apply

Disconnected query

\[
QV = R(x_1), S(x_1, y_1) \lor S(x_2, y_2), T(y_2) \lor R(x_3), T(y_3)
\]

DNF

= H1 (hard !)
When the Rules Do Apply

\[ QV = R(x_1), S(x_1, y_1) \lor S(x_2, y_2), T(y_2) \lor R(x_3), T(y_3) \]

Disconnected query

= H1
(hard !)

DNF

CNF

\[ QV = [S(x_2, y_2), T(y_2) \lor R(x_3)] \land [R(x_1), S(x_1, y_1) \lor T(y_3)] \]
When the Rules Do Apply

QV = R(x1), S(x1,y1) ∨ S(x2,y2), T(y2) ∨ R(x3), T(y3)

Disconnected query

DNF

QV = [S(x2,y2), T(y2) ∨ R(x3)] ∧ [R(x1), S(x1,y1) ∨ T(y3)]

Inclusion/exclusion:

P(QV) = P(q1 ∧ q2) = P(q1) + P(q2) - P(q1 ∨ q2)

= R(x3) V T(y3)
No Dichotomy Yet

An interesting example:

\[
QW = [R(x0),S1(x0,y0) \lor S2(x2,y2),S3(x2,y2)] \land /* Q1 */ \\
[R(x0),S1(x0,y0) \lor S3(x3,y3),T(y3)] \land /* Q2 */ \\
[S1(x1,y1),S2(x1,y1) \lor S3(x3,y3),T(y3)] /* Q3 */
\]
No Dichotomy Yet

An interesting example:

\[ QW = [R(x0), S1(x0, y0) \lor S2(x2, y2), S3(x2, y2)] \land /* Q1 */
[ R(x0), S1(x0, y0) \lor S3(x3, y3), T(y3) ] \land /* Q2 */
[ S1(x1, y1), S2(x1, y1) \lor S3(x3, y3), T(y3) ] /* Q3 */ \]

\[ P(QW) = P(Q1) + P(Q2) + P(Q3) + \\
- P(Q1 \lor Q2) - P(Q2 \lor Q3) - P(Q1 \lor Q3) \\
+ P(Q1 \lor Q2 \lor Q3) = H3 \]

Also = H3 (hard !)
No Dichotomy Yet

An interesting example:

\[ QW = [R(x_0), S_1(x_0, y_0) \lor S_2(x_2, y_2), S_3(x_2, y_2)] \land /* Q1 */ \\
    [R(x_0), S_1(x_0, y_0) \lor S_3(x_3, y_3), T(y_3)] \land /* Q2 */ \\
    [S_1(x_1, y_1), S_2(x_1, y_1) \lor S_3(x_3, y_3), T(y_3)] /* Q3 */ \]

\[ P(QW) = P(Q_1) + P(Q_2) + P(Q_3) + \\
    - P(Q_1 \lor Q_2) - P(Q_2 \lor Q_3) - P(Q_1 \lor Q_3) \\
    + P(Q_1 \lor Q_2 \lor Q_3) \]

Also = H3

PTIME !

\[ \text{H3} \]

(hard !)
August Ferdinand Möbius 1790-1868

Möbius strip
Möbius function $\mu$ in number theory
Generalized to lattices
[Stanley’97, Rota’09]

And now to queries!
The CNF Lattice

Def. The CNF lattice of $Q = Q_1 \land Q_2 \land \ldots$ is:

$Q^W = [R(x_0), S_1(x_0, y_0) \lor S_2(x_2, y_2), S_3(x_2, y_2)] \land /* Q_1 */$

$[R(x_0), S_1(x_0, y_0) \lor S_3(x_3, y_3), T(y_3)] \land /* Q_2 */$

$[S_1(x_1, y_1), S_2(x_1, y_1) \lor S_3(x_3, y_3), T(y_3)] /* Q_3 */$

$^1 = \text{max}(L)$

Some lattice nodes are in PTIME, others are #P hard.
Def. The Möbius function:
\[ \mu(1, 1) = 1 \]
\[ \mu(u, v) = - \sum_{u < v \leq 1} \mu(v, 1) \]

Möbius’ Inversion Formula:
\[ P(Q) = - \sum Qi \preceq 1 \mu(Qi, 1) P(Qi) \]

New Rule 1
Inclusion/Exclusion \[ \square \] Möbius’ Inversion Formula

[Stanley 97]
The Dichotomy

[Dalvi, Schnaitter, S.'2010]

**Dichotomy Theorem**  Fix a UCQ query $Q$.

1. Algorithm terminates, then $P(Q)$ is in PTIME.
2. Algorithm fails, then $P(Q)$ is hard for FP#P.

Dichotomy into PTIME/FP#P based on “syntax” where “syntax” includes the Mobius function!
**Representation Theorem**

**THEOREM** Every lattice $L$ is the CNF lattice of a query $Q$, s.t.
- The query at $^0 = \text{min}(L)$ is hard for $\text{FP}^\#P$
- All other queries are in $\text{PTIME}$

### Examples:

- $Q$ is in $\text{PTIME}$ iff $\mu(^0, ^1) = 0$!
Related Work

Computing $P(F)$ in PTIME:

Read-once RO Boolean Expressions
[Gurvitch’77, Golumbic et al.’06]

Ordered Binary Decision Diagrams OBDD
[Bryant’85]

Free Binary Decision Diagrams FBDD
[Gergov&Meinel’94, Sielig&Wegener’95]

d-DNNF [Darwiche’00]
Necessary and sufficient characterization

Sufficient Characterization (Necessary conjectured)

Möbius Über Alles

UCQ

UCQ(PTIME)

UCQ(d-DNNF)

UCQ(FBDD)

UCQ(OBDD)

UCQ(RO)

[Hja,S.'2011]

[Jha,S.'2011]
Outline

Introduction

Problem Statement

Safe Query Plans

A General Algorithm
Summary

Practical motivation: managing uncertain data

Fundamental problem: computational complexity of Logic + Probabilities

Dichotomy into PTIME/FP#P based on syntax

Algorithm: surprising, simple, complete
Implementations/Performance

Performance rule of thumb:
PTIME queries ≈ conventional SQL queries
P#P-hard queries = 102 - 103 times slower

Implementations of CQ w/o self-joins:
MystiQ: query plan = query hierarchy
MaybeMS/SPROUT: decouples the query plan from the hierarchy
Open Problems

How do we compute the “hard” queries?

Extensions to:

Full FO: \( \neg \forall \)

Independent AND disjoint tuples

Uniform probabilistic structures (MLNs)

Characterize UCQ(FBDD), UCQ(d-DNNF)
Thank You!