CompSci 516
Data Intensive Computing Systems

Lecture 24
View Maintenance
Provenance
Probabilistic Databases
Crowd Sourcing

Instructor: Sudeepa Roy
Announcements

• Last lecture tomorrow 04/19 (Tuesday), D106, 3:05 pm

• HW5 due on 04/20 (Wednesday), 11:55 pm

• If you are/were unable to attend today’s (04/18) lecture, and have questions on the slides, send the instructor an email
Today

Overview of some research areas in databases

- View Materialization and Maintenance
- Data Provenance
- Uncertain Data and Probabilistic Databases
- Crowd Sourcing
- (very briefly) Data Integration

• Understand the high-level ideas and basic techniques
  - The lecture slides will be sufficient for the exams, no additional reading material is needed
View Materialization and Maintenance

[RG] Chapters 25.8-25.10
(slides adapted from the instructor material by the authors)
Views

• Motivation (example)
  – Different groups of analysts within an organization are typically concerned with different aspects of a business
  – It is convenient to define “views” that give each group insight into the relevant business details
  – Other views can be defined or queries can be written using these views
  – Convenient and Efficient
**View Example**

View
(sales of products by category and state)

CREATE VIEW RegionalSales(category, sales, state)
AS SELECT P.category, S.sales, L.state
FROM Products P, Sales S, Locations L
WHERE P.pid=S.pid AND S.locid=L.locid

Query
(total sales for each category by state)

SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales AS R
GROUP BY R.category, R.state

Query Modification
(SQL does not specify how to evaluate queries on views, but can consider it as a replacement)

SELECT R.category, R.state, SUM(R.sales)
FROM (SELECT P.category, S.sales, L.state
      FROM Products P, Sales S, Locations L
      WHERE P.pid=S.pid AND S.locid=L.locid) AS R
GROUP BY R.category, R.state
Views and OLAP/Warehousing

• OLAP queries are typically aggregate queries
  – Precomputation is essential for interactive response times
  – The CUBE is in fact a collection of aggregate queries, and precomputation is especially important
  – lots of work on what is best to precompute given a limited amount of space to store precomputed results.

• Warehouses can be thought of as a collection of asynchronously replicated tables and periodically maintained views
  – Factors: size, number of tables involved, many are from external independent databases
  – Has renewed interest in (asynchronous) view maintenance (more later)
View Materialization

- **Query Modification may not be efficient**
  - when the underlying view is complex
  - even with sophisticated optimization and evaluation
  - esp. when the underlying tables are in a remote database (connectivity and availability)

- **Alternative: View Materialization**
  - Precompute the view definition and store the result
  - Materialized views can be used as regular relations
  - Provides fast access, like a (very high-level) cache
  - Can create index on views too for further speedup
  - Drawback: to maintain the consistency of the materialized view when the underlying table(s) are updated (View Maintenance)
  - Ideally, we want **Incremental View Maintenance** algorithms (Lecture 21)
Index on Materialized Views: Examples

CREATE VIEW RegionalSales(category, sales, state) AS
SELECT P.category, S.sales, L.state
FROM Products P, Sales S, Locations L
WHERE P.pid=S.pid AND S.locid=L.locid

SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales AS R
GROUP BY R.category, R.state

• Suppose we precompute RegionalSales and store it with a clustered B+ tree index on [category, state, sales].
  − Then, the query can be answered by an index-only scan.

  SELECT R.state, SUM(R.sales)
  FROM RegionalSales R
  WHERE R.category="Laptop"
  GROUP BY R.state

  SELECT R.category, SUM(R.sales)
  FROM RegionalSales R
  WHERE R.state="Wisconsin"
  GROUP BY R.category

  Index on precomputed view is great!

 Index is less useful (must scan entire leaf level)
(Research) Issues in View Materialization

1. What views should we materialize, and what indexes should we build on the precomputed results?

2. Given a query and a set of materialized views, can we use the materialized views to answer the query?
   – related to the first question (workload dependent)
   – Try to materialize a small, carefully chosen set of views that can be utilized to quickly answer most of the important queries

3. How frequently should we refresh materialized views to make them consistent with the underlying tables?
   – And how can we do this incrementally?
View Maintenance

• Two steps:
  – Propagate: Compute changes to view when data changes
  – Refresh: Apply changes to the materialized view table

• Maintenance policy: Controls when we do refresh
  – Immediate: As part of the transaction that modifies the underlying data tables
    • + Materialized view is always consistent
    • - updates are slowed
  – Deferred: Some time later, in a separate transaction
    • - View becomes inconsistent
    • + can scale to maintain many views without slowing updates
Types of Deferred Maintenance

Three flavors:

• Lazy:
  – Delay refresh until next query on view; then refresh before answering the query (slows down queries than updates)

• Periodic (Snapshot):
  – Refresh periodically (e.g. once in a day). Queries possibly answered using outdated version of view tuples. Widely used, especially for asynchronous replication in distributed databases, and for warehouse applications

• Event-based or Forced:
  – E.g., Refresh after a fixed number of updates to underlying data tables

• e.g. Snapshot in Oracle 7
  – periodically refreshed by entirely recomputing the view
  – Incremental ”fast refresh” or “simple snapshots” for simpler views (no aggregate, group by, join, distinct etc.)
Provenance

Selected/adapted slides from the keynote by Prof. Val Tannen, EDBT 2010
(optional material: full slide deck is available on Val’s webpage)
Data Provenance

provenance, n.
The fact of coming from some particular source or quarter; origin, derivation [Oxford English Dictionary]

• Data provenance [BunemanKhannaTan 01]: aims to explain how a particular result (in an experiment, simulation, query, workflow, etc.) was derived.

• Most science today is data-intensive. Scientists, eg., biologists, astronomers, worry about data provenance all the time.
Propagating annotations through database operations

The annotation $p \cdot r$ means joint use of data annotated by $p$ and data annotated by $r$.

Slide by Val Tannen, EDBT 2010
Another way to propagate annotations

The annotation \( p + r \) means alternative use of data
Another use of +

\[ R \]

\[
\begin{array}{cccc}
A & B & C \\
\vdots & a & b & c_1 \\
\vdots & a & b & c_2 \\
\vdots & a & b & c_3 \\
\end{array}
\]

\[ \pi_{AB} R \]

\[
\begin{array}{cc}
A & B \\
\vdots & a & b \\
\end{array}
\]

\[ p + r + s \]

+ means alternative use of data

03/24/10

EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010
An example in positive relational algebra (SPJU)

\[ Q = \sigma_{C=e} \pi_{AC}( \pi_{AC}R \bowtimes \pi_{BC}R \cup \pi_{AB}R \bowtimes \pi_{BC}R ) \]

For selection we multiply with two special annotations, 0 and 1
Summary so far

A space of annotations, $K$

$K$-relations: every tuple annotated with some element from $K$.

Binary operations on $K$: $\cdot$ corresponds to joint use (join), and $+$ corresponds to alternative use (union and projection).

We assume $K$ contains special annotations 0 and 1.

“Absent” tuples are annotated with 0!

1 is a “neutral” annotation (no restrictions).

Algebra of annotations? What are the laws of $(K, +, \cdot, 0, 1)$?
Annotated relational algebra

• DBMS query optimizers assume certain equivalences:
  – union is associative, commutative
  – join is associative, commutative, distributes over union
  – projections and selections commute with each other and with union and join (when applicable)
  – Etc., but no $R \square R = R \cup R = R$ (i.e., no idempotence, to allow for bag semantics)

• Equivalent queries should produce same annotations!

Proposition. Above identities hold for queries on $K$-relations iff $(K, +, \cdot, 0, 1)$ is a commutative semiring
What is a commutative semiring?

Different meanings (examples later): $+ = \text{plus}_K$, $\cdot = \text{mult}_K$, $0 = 0_K$, $1 = 1_K$

An algebraic structure $(K, +, \cdot, 0, 1)$ where:
- $K$ is the domain
- $+$ is associative, commutative, with $0$ identity
- $\cdot$ is associative, with $1$ identity
- $\cdot$ distributes over $+$
- $a \cdot 0 = 0 \cdot a = 0$
- $\cdot$ is also commutative

Unlike ring, no requirement for inverses to $+$
Back to the example

R

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
</tr>
</tbody>
</table>

Q

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
</tr>
</tbody>
</table>

\[
(p \cdot p + p \cdot p) \cdot 0
\]

\[
p \cdot r \cdot 1
\]

\[
r \cdot p \cdot 0
\]

\[
(r \cdot r + r \cdot s + r \cdot r) \cdot 1
\]

\[
(s \cdot s + s \cdot r + s \cdot s) \cdot 1
\]
Using the laws: polynomials

Polynomials with coefficients in \( \mathbb{N} \) and annotation tokens as indeterminates \( p, r, s \) capture a very general form of provenance.
Provenance reading of the polynomials

- three different ways to derive $\text{de}$
- two of the ways use only $r$
- but they use it twice
- the third way uses $r$ once and $s$ once

Slide by Val Tannen, EDBT 2010
Provenance Semiring is the Most General Semiring and Has Several Useful “Specialization”

- **Set semantic**
  - Given input tuples exist or not, whether output tuples exist
  - \( K = \{ T, F \} \), \( \text{mult}_K = \land \), \( \text{plus}_K = \lor \), \( 1_K = T \), \( 0_K = F \)
  - e.g. \( p = r = T \), \( s = F \). Then
  - annotation of \((p, r)\): \( T \land T = T \)
  - annotation of \((d, e)\): \( r \lor (r \land s) = T \)
  - annotation of \((f, e)\): \( (r \land s) \lor s = F \)

\( \text{adapted from} \) slide by Val Tannen, EDBT 2010

\( \text{No need to recompute that complex query} \)
Provenance Semiring is the Most General Semiring and Has Several Useful “Specialization”

- **Bag semantic**
  - Given multiplicity of input tuples, compute multiplicities of output tuples
  - $K = \mathbb{N}$ (natural nums), $\text{mult}_K = \ast$, $\text{plus}_K = +$, $1_K = 1$, $0_K = 0$
  - e.g. $p = 2$, $r = 1$, $s = 3$. Then
  - annotation of $(p, r)$: $2 \ast 1 = 2$
  - annotation of $(d, e)$: $2 \ast 1^2 + 1 \ast 3 = 5$
  - annotation of $(f, e)$: $1 \ast 3 + 2 \ast 3^2 = 21$

(adopted from) slide by Val Tannen, EDBT 2010
• **Positive Boolean Expression (PosBool) or Lineage**
  
  – Given variables for input tuples, find Boolean expressions for the output tuples (condition of existence)
  
  – $K = \text{BoolExp}(X)$ (set of input variables is $X$), $\text{mult}_K = \wedge$, $\text{plus}_K = \vee$, $1_K = T$, $0_K = F$
  
  – e.g. given $p, r, s$. Then
  
  – annotation of $(p, r)$: $pr$
  
  – annotation of $(d, e)$: $(r \land r) \lor (r \land s) = r$
  
  – annotation of $(f, e)$: $(r \land s) \lor (s \land s) = s$

Applications in probabilistic databases (see later)

No need to recompute that complex query for $(d, e)$ to exist in the output, it suffices as long as $(d, b, e)$ exist in the output

(adapted from) slide by Val Tannen, EDBT 2010 other examples in the tutorial
Low-hanging fruit: deletion propagation

Delete \( \text{d be e} \) from \( R \)?
Set \( r = 0 \)!

No need to recompute that complex query

Slide by Val Tannen, EDBT 2010
Probabilistic Databases

Selected/adapted slides on the Probabilistic Database book by Prof. Dan Suciu, 2014
(optional material: full slide decks are available on Dan’s webpage)
Probabilistic Databases

- **Data**: standard relational data, plus *probabilities* that measure the degree of uncertainty

- **Queries**: standard SQL queries, whose answers are annotated with *output probabilities*
A Little History of Probabilistic DBs

Early days

- Wong’82
- Shoshani’82
- Cavallo&Pittarelli’87
- Barbara’92
- Lakshmanan’97,’01
- Fuhr&Roellke’97
- Zimanyi’97

Recent work

- 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

Main challenge: Query Evaluation (=Probabilistic Inference)

Unfortunately, no “practical/usable” prob. db. systems
Why?

Many applications need to manage uncertain data

- Information extraction
- Knowledge representation
- Fuzzy matching
- Business intelligence
- Data integration
- Scientific data management
- Data anonymization
What?

• Probabilistic Databases extend Relational Databases with probabilities

• Combine Formal Logic with Probabilistic Inference

• Requires a new thinking for both databases and probabilistic inference
Example 1: Information Extraction

52-A Goregaon West Mumbai 400 076

<table>
<thead>
<tr>
<th>Id</th>
<th>House_no</th>
<th>Area</th>
<th>City</th>
<th>Pincode</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52</td>
<td>Goregaon West</td>
<td>Mumbai</td>
<td>400 062</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>52-A</td>
<td>Goregaon</td>
<td>West Mumbai</td>
<td>400 062</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>52-A</td>
<td>Goregaon West</td>
<td>Mumbai</td>
<td>400 062</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>52</td>
<td>Goregaon</td>
<td>West Mumbai</td>
<td>400 062</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Standard DB: keep the most likely extraction

Probabilistic DB: keep most/all extractions to increase recall

[Gupta’2006]

Slide by Dan Suciu
Example 2: Modeling Missing Data

<table>
<thead>
<tr>
<th>id</th>
<th>age</th>
<th>edu</th>
<th>inc</th>
<th>nw</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>20</td>
<td>HS</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>t2</td>
<td>20</td>
<td>BS</td>
<td>50K</td>
<td>100K</td>
</tr>
<tr>
<td>t3</td>
<td>20</td>
<td>?</td>
<td>50K</td>
<td>?</td>
</tr>
<tr>
<td>t4</td>
<td>20</td>
<td>100K</td>
<td>500K</td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>20</td>
<td>HS</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>t6</td>
<td>20</td>
<td>HS</td>
<td>50K</td>
<td>100K</td>
</tr>
<tr>
<td>t7</td>
<td>20</td>
<td>HS</td>
<td>50K</td>
<td>500K</td>
</tr>
<tr>
<td>t8</td>
<td>?</td>
<td>HS</td>
<td>50K</td>
<td>500K</td>
</tr>
<tr>
<td>t9</td>
<td>30</td>
<td>BS</td>
<td>?</td>
<td>100K</td>
</tr>
<tr>
<td>t10</td>
<td>30</td>
<td>?</td>
<td>100K</td>
<td>?</td>
</tr>
<tr>
<td>t11</td>
<td>30</td>
<td>HS</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>t12</td>
<td>30</td>
<td>MS</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>t13</td>
<td>40</td>
<td>BS</td>
<td>100K</td>
<td>100K</td>
</tr>
<tr>
<td>t14</td>
<td>40</td>
<td>HS</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>t15</td>
<td>40</td>
<td>BS</td>
<td>50K</td>
<td>500K</td>
</tr>
<tr>
<td>t16</td>
<td>40</td>
<td>HS</td>
<td>?</td>
<td>500K</td>
</tr>
<tr>
<td>t17</td>
<td>40</td>
<td>HS</td>
<td>100K</td>
<td>500K</td>
</tr>
</tbody>
</table>

**Standard DB:** NULL

**Probabilistic DB:** distribution on possible values

```
<table>
<thead>
<tr>
<th>id</th>
<th>age</th>
<th>edu</th>
<th>inc</th>
<th>nw</th>
<th>prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>t12</td>
<td>30</td>
<td>MS</td>
<td>50K</td>
<td>100K</td>
<td>0.30</td>
</tr>
<tr>
<td>t12</td>
<td>30</td>
<td>MS</td>
<td>50K</td>
<td>500K</td>
<td>0.45</td>
</tr>
<tr>
<td>t12</td>
<td>30</td>
<td>MS</td>
<td>100K</td>
<td>100K</td>
<td>0.10</td>
</tr>
<tr>
<td>t12</td>
<td>30</td>
<td>MS</td>
<td>100K</td>
<td>500K</td>
<td>0.15</td>
</tr>
</tbody>
</table>
```
Example 3: Data Cleaning

This slide should give you some idea about the goals in data cleaning! (e.g. maintain key constraints or FDs)

Standard DB cleaning data means choosing one possible repair

Probabilistic DB keep many/all possible repairs

[Beskales’2009]
Example 4: OCR

They use OCRopus from Google Books: output is a stochastic automaton
Traditionally: retain only the Maximum Apriori Estimate (MAP)
With a probabilistic database: may retain several alternative recognitions: increase recall

SELECT DocId, Loss
FROM Claims
WHERE Year = 2010
AND DocData LIKE '%Ford%';
Summary of Applications

- Structured, but uncertain data
- Modeled as probabilistic data
- Answers to SQL queries annotated with probabilities

Probabilistic database:
- Combine data management with probabilistic inference
Review: Complexity of Query Evaluation

Query $Q$, database $D$

- **Data complexity:**
  
  $\text{fix } Q$, complexity $= f(D)$

- **Query complexity:**
  
  $\text{fix } D$, complexity $= f(Q)$

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

Moshe Vardi

Data complexity is unique to database research

All query languages that exist today in db systems have Poly-time data complexity (SQL, Datalog, Datalog+negation, Xquery for XML)
Incomplete Database

**Definition** An **Incomplete Database** is a finite set of database instances \( W = (W_1, W_2, \ldots, W_n) \)

Each \( W_i \) is called a possible world
# Incomplete Database

**Definition** An Incomplete Database is a finite set of database instances $W = (W_1, W_2, \ldots, W_n)$

Each $W_i$ is called a possible world.

<table>
<thead>
<tr>
<th>Owner</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W₁</th>
<th>W₂</th>
<th>W₃</th>
<th>W₄</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="table.png" alt="Table" /></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Definition An **Incomplete Database** is a finite set of database instances \( W = (W_1, W_2, \ldots, W_n) \)

Each \( W_i \) is called a possible world

<table>
<thead>
<tr>
<th>( W_1 )</th>
<th>( W_2 )</th>
<th>( W_3 )</th>
<th>( W_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Owner</strong></td>
<td><strong>Location</strong></td>
<td><strong>Object</strong></td>
<td><strong>Time</strong></td>
</tr>
<tr>
<td>Name</td>
<td>Object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Object | Time | Loc
--- | --- | ---
Laptop77 | 5:07 | Hall
Laptop77 | 9:05 | Office
Book302 | 8:18 | Office
**Incomplete Database**

**Definition** An *Incomplete Database* is a finite set of database instances $W = (W_1, W_2, \ldots, W_n)$

Each $W_i$ is called a possible world

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$W_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td><strong>Object</strong></td>
<td><strong>Location</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Time</strong></td>
<td><strong>Loc</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Laptop77</td>
<td>5:07</td>
<td>Hall</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:05</td>
<td>Office</td>
<td>Laptop77</td>
</tr>
</tbody>
</table>
Incomplete Database: Query Semantics

**Definition** Given query $Q$, incomplete database $W$:
- An answer $t$ is **certain** if $\forall W_i, \ t \in Q(W_i)$
- An answer $t$ is **possible** if $\exists W_i, \ t \in Q(W_i)$
Incomplete Database: Query Semantics

**Definition** Given query $Q$, incomplete database $W$:
- An answer $t$ is **certain** if $\forall W_i, t \in Q(W_i)$
- An answer $t$ is **possible** if $\exists W_i, t \in Q(W_i)$

$$Q(z) = \text{Owner}(z, x), \text{Location}(x, t, 'Office')$$
**Incomplete Database: Query Semantics**

**Definition** Given query $Q$, incomplete database $W$:

- An answer $t$ is **certain**, if $\forall W_i, t \in Q(W_i)$
- An answer $t$ is **possible** if $\exists W_i, t \in Q(W_i)$

$$Q(z) = \text{Owner}(z,x), \text{Location}(x,t,’Office’)$$

---

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$W_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td><strong>Object</strong></td>
<td><strong>Time</strong></td>
<td><strong>Loc</strong></td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Time</strong></td>
<td><strong>Loc</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Laptop77</td>
<td>5:07</td>
<td>Hall</td>
<td>Book302</td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:05</td>
<td>Office</td>
<td>Book302</td>
</tr>
<tr>
<td>Book302</td>
<td>8:18</td>
<td>Office</td>
<td></td>
</tr>
</tbody>
</table>
Incomplete Database: Query Semantics

**Definition** Given query $Q$, incomplete database $W$:
- An answer $t$ is **certain**, if $\forall W_i, \ t \in Q(W_i)$
- An answer $t$ is **possible** if $\exists W_i, \ t \in Q(W_i)$

$Q(z) = Owner(z,x), \ Location(x,t,’Office’)$

**Certain answers to $Q$:** Joe  
**Possible answers to $Q$:** Joe, Jim

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$W_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td><strong>Object</strong></td>
<td><strong>Time</strong></td>
<td><strong>Loc</strong></td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Time</strong></td>
<td><strong>Loc</strong></td>
<td></td>
</tr>
<tr>
<td>Laptop77</td>
<td>5:07</td>
<td>Hall</td>
<td></td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:05</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>Book302</td>
<td>8:18</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$Q = \text{Joe, Jim}$

**Query Examples**

$Q=$ Duke CS, Spring 2016  
CompSci 516: Data Intensive Computing Systems  
Slide by Dan Suciu
**Definition** A **Probabilistic Database** is \((W, P)\), where \(W\) is an incomplete database, and \(P: W \rightarrow [0,1]\) a probability distribution: \(\sum_{i=1}^{n} P(W_i) = 1\)
**Definition**  
A **Probabilistic Database** is \((W, P)\), where \(W\) is an incomplete database, and \(P: W \rightarrow [0,1]\) a probability distribution: \(\Sigma_{i=1,n} P(W_i) = 1\)

<table>
<thead>
<tr>
<th>(W_1)</th>
<th>(W_2)</th>
<th>(W_3)</th>
<th>(W_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
</tr>
<tr>
<td>Joe Book302</td>
<td>Joe Book302</td>
<td>Jim Laptop77</td>
<td>Joe Book302</td>
</tr>
<tr>
<td>Joe Laptop77</td>
<td>Jim Laptop77</td>
<td>Jim Laptop77</td>
<td>Jim Laptop77</td>
</tr>
<tr>
<td>Jim Laptop77</td>
<td>Fred GgleGlass</td>
<td>Fred GgleGlass</td>
<td>Fred GgleGlass</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td><strong>Location</strong></td>
<td><strong>Location</strong></td>
<td><strong>Location</strong></td>
</tr>
<tr>
<td>Laptop77 5:07 Hall</td>
<td>Laptop77 5:07 Hall</td>
<td>Laptop77 5:07 Hall</td>
<td>Laptop77 5:07 Hall</td>
</tr>
<tr>
<td>Laptop77 9:05 Office</td>
<td>Laptop77 9:05 Office</td>
<td>Laptop77 9:05 Office</td>
<td>Laptop77 9:05 Office</td>
</tr>
</tbody>
</table>
Probabilistic Database: Query Semantics

**Definition**  
Given query $Q$, probabilistic database $(W, P)$:

- The marginal probability of an answer $t$ is:
  \[ P(t) = \sum \{ P(W_i) \mid W_i \in W, \ t \in Q(W_i) \} \]
**Probabilistic Database: Query Semantics**

**Definition** Given query $Q$, probabilistic database $(W,P)$:
- The marginal probability of an answer $t$ is:
  $$P(t) = \sum \{ P(W_i) \mid W_i \in W, \ t \in Q(W_i) \}$$

$$Q(z) = \text{Owner}(z,x), \ 	ext{Location}(x,t,'Office')$$

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$W_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
<td><strong>Owner</strong></td>
</tr>
<tr>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Object</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Location</strong></th>
<th><strong>Object</strong></th>
<th><strong>Time</strong></th>
<th><strong>Loc</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop77</td>
<td>5:07</td>
<td>Hall</td>
<td></td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:05</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>Book302</td>
<td>8:18</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>Laptop77</td>
<td>5:07</td>
<td>Hall</td>
<td></td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:05</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>Book302</td>
<td>8:18</td>
<td>Office</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Object</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Location</strong></th>
<th><strong>Object</strong></th>
<th><strong>Time</strong></th>
<th><strong>Loc</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop77</td>
<td>5:07</td>
<td>Hall</td>
<td></td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:05</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>Book302</td>
<td>8:18</td>
<td>Office</td>
<td></td>
</tr>
</tbody>
</table>
**Probabilistic Database: Query Semantics**

**Definition** Given query $Q$, probabilistic database $(W, P)$:
- The marginal probability of an answer $t$ is:
  $$P(t) = \sum \{ P(W_i) \mid W_i \in W, \ t \in Q(W_i) \}$$

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$W_2$</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$W_3$</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Laptop77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$W_4$</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Joe</td>
<td>Book302</td>
</tr>
<tr>
<td>Jim</td>
<td>Laptop77</td>
</tr>
<tr>
<td>Fred</td>
<td>GgleGlass</td>
</tr>
</tbody>
</table>
**Definition** Given query \( Q \), probabilistic database \( \langle W, P \rangle \):

- The marginal probability of an answer \( t \) is:
  \[
P(t) = \sum \{ P(W_i) \mid W_i \in W, \ t \in Q(W_i) \}
  \]

\( Q(z) = \text{Owner}(z,x), \text{Location}(x,t,'Office') \)

\[
P(\text{Joe}) = 1.0 \quad P(\text{Jim}) = 0.4
\]
Discussion

• **Intuition**: a probabilistic database says that the database can be in one of possible states, each with a probability

• **Possible query answers**: a set of answers annotated with probabilities:

\[(t_1, p_1), (t_2, p_2), (t_3, p_3), \ldots\]

Usually: \(p_1 \geq p_2 \geq p_3 \geq \ldots\)

• **Problem**: the number of possible world in a probabilistic database is astronomically large. To represent it, we impose some restrictions
  • independence and/or disjointness of tuples
Independent, Disjoint Tuples

**Definition** Given a probabilistic database \((W, P)\). Two tuples \(t_1, t_2\) are called:
- **Independent**, if: \(P(t_1 t_2) = P(t_1) P(t_2)\)
- **Disjoint** (or exclusive), if: \(P(t_1 t_2) = 0\)
Independent, Disjoint Tuples

Definition Given a probabilistic database \((W, P)\). Two tuples \(t_1, t_2\) are called:
- **Independent**, if: \(P(t_1 \land t_2) = P(t_1) P(t_2)\)
- **Disjoint** (or exclusive), if: \(P(t_1 \land t_2) = 0\)

Definition A probabilistic database is called **Block-Independent-Disjoint** (BID), if its tuples are grouped into blocks such that:
- Tuples from the same block are **disjoint**
- Tuples from different blocks are **independent**
Example: BID Table

<table>
<thead>
<tr>
<th>Object</th>
<th>Time</th>
<th>Loc</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop77</td>
<td>9:07</td>
<td>Rm444</td>
<td>p₁</td>
</tr>
<tr>
<td>Laptop77</td>
<td>9:07</td>
<td>Hall</td>
<td>p₂</td>
</tr>
<tr>
<td>Book302</td>
<td>9:18</td>
<td>Office</td>
<td>p₃</td>
</tr>
<tr>
<td>Book302</td>
<td>9:18</td>
<td>Rm444</td>
<td>p₄</td>
</tr>
<tr>
<td>Book302</td>
<td>9:18</td>
<td>Lift</td>
<td>p₅</td>
</tr>
</tbody>
</table>

- At most two tuples in a world
- At most one tuple from the first block
- At most one tuple from the second block
The Query Evaluation Problem

Given: a probabilistic database $D$, a query $Q$, and output tuple $t$

Compute: $P(t)$

Note: $D$ has, say, 1000000 tuples, while the number of possible worlds is $2^{1000000}$

Challenge: compute $P(t)$ efficiently, in the size of $D$

Data complexity: the complexity of $P$ depends dramatically on $Q$
Two approaches to query evaluation on tuple-independent probabilistic databases

1. Intensional query evaluation

2. Extensional query evaluation
Approach 1: Intensional Query Evaluation

Query $Q$ + database $D \rightarrow$ lineage (provenance) expression $F_Q$

Compute $P(F)$ using a general model counting system

In general, computationally hard (weighted model counting)

But poly-time for some $Q$ or $Q, D$

Revisit the third (last) example of provenance semiring on slide 27!
Example: Intensional Query Evaluation

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

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

```
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>Y1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
<td>Y2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>Y3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
<td>Y4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
<td>Y5</td>
</tr>
</tbody>
</table>
```

\[ F_Q = X1 \ Y1 \ V \ X1 \ Y2 \ V \ X2 \ Y3 \ V \ X2 \ Y4 \ V \ X2 \ Y5 \]

Now compute \( \Pr[F_Q] \), given \( \Pr[X1] \), \( \Pr[X2] \), ..., and assuming the variables are independent.
For some provenance formulas, probability can be computed in poly-time

• Example 1: If the formula is “read once”
  – see next slide

• Example 2: If poly-size knowledge compilation forms (OBDD, FBDD) exist
  – similar idea like read-once
  – not covered in this class
Read-Once Boolean Formulas

A Boolean formula $F$ is called **read-once** if it can be written such that every Boolean variable occurs only once.

- $P(F)$ can be computed in linear time (independence):

$$P(F_1 \land F_2) = P(F_1) \times P(F_2)$$
$$P(F_1 \lor F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$$

---

Bottom-up computation in the read-once tree:

$$P_{X} = 0.1, \ P_{Z} = 0.5, \ P_{Y} = 0.9, \ P_{U} = 0.2$$

$$(X \lor Z) \land (Y \lor U)$$

$$P_{X} = 0.1, \ P_{Z} = 0.5, \ P_{Y} = 0.9, \ P_{U} = 0.2$$

$$(X \lor Z) \land (Y \lor U)$$

- $P(F_1 \land F_2) = P(F_1) \times P(F_2)$
- $P(F_1 \lor F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$
Read-Once Example

SELECT DISTINCT ‘true’
FROM R, S
WHERE R.x = S.x

Q = R(x), S(x,y)

R

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
</tr>
</tbody>
</table>

S

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
</tr>
</tbody>
</table>

F_Q = X1 Y1 V X1 Y2 V X2 Y3 V X2 Y4 V X2 Y5

= X1 (Y1 V Y2) V X2 (Y3 V Y4 V Y5)

Read-once
Approach 2: Extensional Query Evaluation

Main idea:
- Modify each operator to compute output probabilities
- Correct plans are “safe plans” (work for all databases)
- Not always exist
An Example

SELECT DISTINCT ‘true’
FROM R, S
WHERE R.x = S.x

\[ P(Q) = \]

Boolean query

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

Slide by Dan Suciu
An Example

Boolean query

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

**Query**

\[
P(Q) = 1 - (1-q1)(1-q2)
\]

**Database R**

<table>
<thead>
<tr>
<th>x</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>p1</td>
</tr>
<tr>
<td>a2</td>
<td>p2</td>
</tr>
<tr>
<td>a3</td>
<td>p3</td>
</tr>
</tbody>
</table>

**Database S**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>q1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
<td>q2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>q3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
<td>q4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
<td>q5</td>
</tr>
</tbody>
</table>
An Example

Boolean query

$$\Pr(Q) = \text{p1} \cdot [1-(1-\text{q1}) \cdot (1-\text{q2})]$$

<table>
<thead>
<tr>
<th>R</th>
<th></th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>P</td>
<td>x</td>
</tr>
<tr>
<td>a1</td>
<td>p1</td>
<td>a1</td>
</tr>
<tr>
<td>a2</td>
<td>p2</td>
<td>a1</td>
</tr>
<tr>
<td>a3</td>
<td>p3</td>
<td>a2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a2</td>
</tr>
</tbody>
</table>
### An Example

Boolean query

$$Q() = R(x), S(x,y)$$

$$P(Q) = \text{p1 \ast [1-} (1-q1)(1-q2) \text{]}$$

$$1-\text{[}1-q3)(1-q4)(1-q5)$$

---

**R**

<table>
<thead>
<tr>
<th>x</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>p1</td>
</tr>
<tr>
<td>a2</td>
<td>p2</td>
</tr>
<tr>
<td>a3</td>
<td>p3</td>
</tr>
</tbody>
</table>

**S**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>q1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
<td>q2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>q3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
<td>q4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
<td>q5</td>
</tr>
</tbody>
</table>
An Example

Boolean query

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

\[ P(Q) = p_1 \left[ 1 - (1 - q_1)(1 - q_2) \right] \]

\[ p_2 \left[ 1 - (1 - q_3)(1 - q_4)(1 - q_5) \right] \]

---

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>q1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
<td>q2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>q3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
<td>q4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
<td>q5</td>
</tr>
</tbody>
</table>
An Example

SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x

\[ P(Q) = 1 - \{1 - p_1[1 - (1-q_1)*(1-q_2)]\} \times \{1 - p_2[1 - (1-q_3)*(1-q_4)*(1-q_5)]\} \]

Condition for join on a1 or on a2

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a1</td>
<td>b1</td>
<td>q1</td>
</tr>
<tr>
<td>R</td>
<td>a1</td>
<td>b2</td>
<td>q2</td>
</tr>
<tr>
<td></td>
<td>a2</td>
<td>b3</td>
<td>q3</td>
</tr>
<tr>
<td></td>
<td>a2</td>
<td>b4</td>
<td>q4</td>
</tr>
<tr>
<td></td>
<td>a2</td>
<td>b5</td>
<td>q5</td>
</tr>
</tbody>
</table>

Boolean query \( Q() = R(x), S(x,y) \)
An Example

Boolean query

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

\[
\begin{align*}
P(Q) &= 1 - \{1 - p_1 \left[ 1 - (1-q_1)(1-q_2) \right]\} \times \\
&\quad \left\{1 - p_2 \left[ 1 - (1-q_3)(1-q_4)(1-q_5) \right]\right\}
\end{align*}
\]

One can compute \( P(Q) \) in PTIME in the size of the database \( D \)

\[
\begin{array}{|c|c|c|}
\hline
x & y & P \\
\hline
a1 & b1 & q1 \\
a1 & b2 & q2 \\
a2 & b3 & q3 \\
a2 & b4 & q4 \\
a2 & b5 & q5 \\
\hline
\end{array}
\]
Extensional Operators

Independent join

\[ S(A, B) \]

\[
\begin{array}{|c|c|c|}
\hline
A & B & P \\
\hline
a1 & b1 & p1*q1 \\
a1 & b2 & p1*q2 \\
a2 & b3 & p2*q3 \\
a2 & b4 & p2*q4 \\
a2 & b5 & p2*q5 \\
\hline
\end{array}
\]

\[ R(A) \]

\[
\begin{array}{|c|c|}
\hline
A & P \\
\hline
a1 & p1 \\
a2 & p2 \\
a3 & p3 \\
\hline
\end{array}
\]
Extensional Operators

Independent join

\[
\begin{array}{ccc}
A & B & P \\
a1 & b1 & p1 \cdot q1 \\
a1 & b2 & p1 \cdot q2 \\
a2 & b3 & p2 \cdot q3 \\
a2 & b4 & p2 \cdot q4 \\
a2 & b5 & p2 \cdot q5 \\
\end{array}
\]

\[
S(A, B)
\]

Independent project

\[
\begin{array}{c}
A \\
a1 \\
a2 \\
\end{array}
\begin{array}{c}
P \\
1 - (1 - q1) \cdot (1 - q2) \\
1 - (1 - q3) \cdot (1 - q4) \cdot (1 - q5)
\end{array}
\]

\[
R(A)
\]

\[
\prod_{A}^i
\]

\[
S(A, B)
\]

\[
\begin{array}{ccc}
A & B & P \\
a1 & b1 & q1 \\
a1 & b2 & q2 \\
a2 & b3 & q3 \\
a2 & b4 & q4 \\
a2 & b5 & q5 \\
\end{array}
\]
Extensional Operators

Independent Join

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>p1*q1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
<td>p1*q2</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>p2*q3</td>
</tr>
<tr>
<td>a2</td>
<td>b4</td>
<td>p2*q4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
<td>p2*q5</td>
</tr>
</tbody>
</table>

Independent Project

<table>
<thead>
<tr>
<th>A</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1 - (1-q1)*(1-q2)</td>
</tr>
<tr>
<td>a2</td>
<td>1 - (1-q3)<em>(1-q4)</em>(1-q5)</td>
</tr>
</tbody>
</table>

Selection

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2</td>
<td>b2</td>
<td>q3</td>
</tr>
<tr>
<td>a2</td>
<td>b3</td>
<td>q4</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>q5</td>
</tr>
</tbody>
</table>

R(A) \ ⋈ \ S(A,B)

\prod_A S(A,B)

\sigma_{A=a2} S(A,B)
Old Example

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

\[
P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))] 
* [1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))]
\]

\[
\begin{array}{|c|c|c|}
\hline
x & y & P \\
\hline
a_1 & b_1 & q_1 \\
\hline
a_1 & b_2 & q_2 \\
\hline
a_2 & b_3 & q_3 \\
\hline
a_2 & b_4 & q_4 \\
\hline
a_2 & b_5 & q_5 \\
\hline
\end{array}
\]

R

S

Duke CS, Spring 2016
CompSci 516: Data Intensive Computing Systems
### Probabilistic Databases - Dan Suciu

**SELECT DISTINCT** ‘true’

**FROM** R, S

**WHERE** R.x = S.x

**Q() = R(x), S(x,y)**

\[ P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))] \]
\[ *[1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))] \]

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

### Slide by Dan Suciu

Wrong

### No longer independent

### Independent

** optional slide (check yourself!)**
# Two approaches to query evaluation on tuple-independent probabilistic databases

<table>
<thead>
<tr>
<th></th>
<th>Intentional Approach</th>
<th>Extensional Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Idea</strong></td>
<td>Find the “provenance/lineage expression” as a Boolean formula (PosBool semiring). Compute the probability of this Boolean formula assuming the variables are independent</td>
<td>Find a “safe query plan” if possible</td>
</tr>
<tr>
<td><strong>Specific to</strong></td>
<td>A database and a query</td>
<td>A query (works for all input databases)</td>
</tr>
<tr>
<td><strong>Existence</strong></td>
<td>Always exist for a RA query</td>
<td>May or may not exist</td>
</tr>
<tr>
<td><strong>Computation</strong></td>
<td>Computation of the formula is in poly-time (data complexity), but computation of the probability may be computationally hard (#P-hard)</td>
<td>If a safe plan exists, computation is in poly-time (data complexity)</td>
</tr>
</tbody>
</table>
Challenges

• No safe plan even for simple queries like Q() :- R(x), S(x, y), T(y)
  – No safe plan for SPJU (RA with union) => query is computationally hard (seminal result by Dalvi-Suciu)

• Study models and query evaluation (exact and approximate inference) that work well in practice

• Uncertain rules/queries vs. uncertain data
  – e.g. Markov Logic Network (Domingos et al.) or PSL (Getoor et al.)
Crowd Sourcing

Selected/adapted slides from the tutorial by Profs. Daniel Deutch and Tova Milo, SIGMOD 2011
(optional material: full slide deck is available on Tova’s webpage)
CrowdSourcing

• Main idea: Harness the crowd to a “task”
  – Task: solve bugs
  – Task: find an appropriate treatment to an illness
  – Task: construct a database of facts
...

• Why now?
  – Internet and smart phones ...
    We are all connected, all of the time!!!
The classical example

WIKIPEDIA

English
The Free Encyclopedia
3 907 000+ articles

Español
La enciclopedia libre
879 000+ artículos

Polski
Wolna encyklopedia
887 000+ hasel

Deutsch
Die freie Enzyklopädie
1 383 000+ Artikel

Italiano
L’enciclopedia libera
905 000+ voci

Français
L’encyclopédie libre
1 230 000+ articles

Português
A enciclopédia livre
718 000+ artigos

中文
自由的百科全书
429 000+ 修目

日本語
フリー百科事典
799 000+ 記事
Galaxy Zoo

...just like MOON ZOO

GALAXY ZOO HUBBLE

Contact Us
Collaborative Testing

Gain Confidence in Your Software Product.
Crowdsourced Software Testing by Passionate Testers.

Create a test requirement
which is simply a clear outline of what you need tested. To begin, post this requirement to 99tests and set your amount.

Testers submit Bugs
to compete for your prize. Be sure to provide continual feedback to help the testers verify the functionality that you have developed.

Client Signup

Top testers get prize
and you’ll receive full access to all the bugs.

Tester Signup
CrowdSourcing: Unifying Principles

• Main goal
  – “Outsourcing” a task to a crowd of users

• Kinds of tasks
  – Tasks that can be performed by a computer, but inefficiently
  – Tasks that can’t be performed by a computer

• Challenges
  – How to motivate the crowd?
  – Get data, minimize errors, estimate quality
  – Direct users to contribute where is most needed \ they are experts
Motivating the Crowd

Altruism

Fun

Money

HITs - Human Intelligence Tasks - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:
- Can work from home
- Choose your own work hours
- Get paid for doing good work

Find an
interesting task
Work
Earn
money

comp: Mech Turk
Crowd Data Sourcing

• The case where the task is collection of data

• Two main aspects [DFKK’12]:
  – Using the crowd to create better databases
  – Using database technologies to create better crowd datasourcing applications

Data-related Tasks (that can be) Performed by Crowds

- Data cleaning
  - E.g. repairing key violations by settling contradictions
- Data Integration
  - E.g. identify mappings
- Data Mining
  - E.g. entity resolution
- Information Extraction

[Internet-Scale Collection of Human-Reviewed Data, Q. Su, D. Pavlov, J. Chow, W.C. Baker, WWW '07]
[Matching Schemas in Online Communities: A Web 2.0 Approach, R. McCann, W. Shen, A. Doan, ICDE '08]
[Amplifying Community Content Creation with Mixed Initiative Information Extraction, R. Hoffman, S. Amershi, K. Patel, F. Wu., J. Fogarty, D. Weld, CHI '09]
Main Tasks in Crowd Data Sourcing

- What questions to ask?
- How to define correctness of answers?
- How to clean the data?
- Who to ask? how many people?
- How to best use resources?

<table>
<thead>
<tr>
<th>Declarative Framework!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic Data!</td>
</tr>
<tr>
<td>Data Cleaning!</td>
</tr>
<tr>
<td>Optimizations and Incremental Computation</td>
</tr>
</tbody>
</table>
Platforms for Crowdsourcing

Quirk (MIT)

CrowdDB (Berkeley and ETH Zurich)

CrowdForge (CMU)

Deco (Stanford and UCSC)

MoDaS (Tel Aviv University)

...
Conclusions

• All classical issues:
  Data models, query languages, query processing, optimization, HCI

• Database techniques are very useful
  – “Classical” as well as new

• BUT
  • (Very) interactive computation
  • (Very) large scale data
  • (Very) little control on quality/reliability
Many (Research) Challenges

• Not only in databases, but in several other communities: ML, KD, Web, ...

• Latency, quality, cost
  – Ask small #questions in small #rounds
  – Ask the right questions

• Efficiency
  – distributed processing
  – incremental processing

• Semantic
  – text/image processing
  – data mining with crowd (model how people think)
Data Integration

(Overview Only: [RG] Chapter 29.2)
Motivation

- As databases grow, users want to access data from more than one source
  - e.g., compare travel packages from multiple agents/sites
  - e.g. large organizations have several databases created/maintained by different divisions – may have common info – need to determine the relationships between these databases
  - different forms of data – prices in USD/item, USD/dozen-of-items etc.
  - XML data – may not follow the same DTD – legacy databases – semantic mismatches
Approaches to Data Integration

• Semantic mismatches can be resolved and hidden from users by defining views over the two databases
  – Semantic aggregation
  – Challenges due to poor documentation – difficult to understand the meaning and define unifying views

• If the underlying databases are managed using different DBMSs,..
  – some kind of “middleware” must be used to evaluate queries over the integrated views to give the databases a uniform interface (ODBC)
  – alternatively, the integrating views can be materialized and stored in a data warehouse -- queries can be executed over the warehouse data without accessing the source DBMSs at run-time