ULDBs: Databases with Uncertainty and Lineage

Omar Benjelloun
Anish Das Sarma
Alon Halevy
Jennifer Widom

Presented by: Sindhura Tokala
Results of IR queries are ranked and uncertain
Example:

```
SELECT * 
FROM Actor A 
WHERE A.name ≈ 'Kevin'
    and 1995 = 
SELECT MIN(F.year) 
FROM Film F, Casts C 
WHERE C.filmid = F.filmid 
    and C.actorid = A.actorid 
    and F.rating ≈ "high"
```
Example

```
SELECT * FROM Actor A
WHERE A.name ≈ 'Kevin'
and 1995 =
SELECT MIN(F.year)
FROM Film F, Casts C
WHERE C.filmid = F.filmid
and C.actorid = A.actorid
and F.rating ≈ "high"
```
Probabilistic Databases

- Each tuple has a probability of belonging to the database
Example

\[ S^p = \begin{array}{ccc}
\text{A} & \text{B} & 0.8 \\
's_1' & 1 & 0.5 \\
's_2' & 1 & 0.5 \\
\end{array} \]

\[ T^p = \begin{array}{ccc}
\text{C} & \text{D} & 0.6 \\
't_1' & 1 & 'p' \\
\end{array} \]
Possible Worlds

\[ S^p = \begin{array}{c|c} \text{A} & \text{B} \\ \hline s_1 & 'm' & 1 \\ s_2 & 'n' & 1 \end{array} \]

\[ T^p = \begin{array}{c|c} \text{C} & \text{D} \\ \hline t_1 & 1 & 'p' & 0.6 \end{array} \]

\[ p\text{wd}(D^p) = \begin{array}{|c|c|} \hline \text{database instance} & \text{probability} \\ \hline D_1 = \{s_1, s_2, t_1\} & 0.24 \\ D_2 = \{s_1, t_1\} & 0.24 \\ D_3 = \{s_2, t_1\} & 0.06 \\ D_4 = \{t_1\} & 0.06 \\ D_5 = \{s_1, s_2\} & 0.16 \\ D_6 = \{s_1\} & 0.16 \\ D_7 = \{s_2\} & 0.04 \\ D_8 = \emptyset & 0.04 \\ \hline \end{array} \]
Query evaluation on probabilistic Databases

Consider the query:

$$q(u): -S^p(x, y), T^p(z, u), y = z$$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>'m'</td>
<td>1</td>
</tr>
<tr>
<td>$s_2$</td>
<td>'n'</td>
<td>1</td>
</tr>
</tbody>
</table>

$$S^p = \begin{array}{cc}
0.8 \\
0.5 
\end{array}$$

$$T^p = \begin{array}{cc}
C & D \\
1 & 'p' 
\end{array}$$

$$q^{pwd}(D^p) = \begin{array}{|c|c|}
\hline
\text{answer} & \text{probability} \\
\hline
\{'p'\} & 0.54 \\
\emptyset & 0.46 \\
\hline
\end{array}$$
Problems with such data:

- Handling “Data Lineage”
- Handling “Uncertainty”
Data Lineage

- Metadata Management
- Functionality of determining:
  - Where the data came from
  - How it is transformed
  - Where it is going
Lineage => Resolve Uncertainty

- Example: Search Engines

- Uncertainty::Ranking

- Lineage::Text Snippet
Example:

- Two sets of base data: A and B
- Only one base set is correct
- Derived queries should not produce data that is mixed from A and B
- Lineage helps encode relationships that arise between base and derived data.
Agenda

- Define ULDBs: uncertain databases with lineage
- Combine Lineage and Uncertainty
- Querying ULDBs
- Extend ULDBs with confidence values.
Problem Setup:

- Database \( D \)
- Set of relations \( \bar{R} = R_1, R_2, \ldots, R_n \)
- Each \( R_i \) is a multi-set of tuples
- Each tuple has a unique identifier
- \( I(\bar{R}) \) denotes identifiers in relations \( R_1, R_2, \ldots, R_n \)
Databases with Lineage (LDB)

- Lineage of a tuple identifies the data from which it was derived
- External Lineage: derived from outside the LDB
- Internal Lineage: references to other tuples in the LDB
LDB Representation

- Triple \((\overline{R}, S, \lambda)\), where
- \(\overline{R}\) : set of relations
- \(S\) : set of symbols containing \(I(\overline{R})\)
- \(\lambda\) : lineage function from \(S\) to \(2^S\)
Example

- Drives(person, car)
- Saw(witness, car)
- Accuses(witness, person) : from the query $\pi_{\text{witness, person}}(\text{Saw} \Join \text{Drives})$
Uncertain Database Representation

- X-tuples
- X-relations
Definitions

- **x-tuple**: multiset of one or more tuples, called *alternatives* (mutually exclusive values for the tuple)
- **maybe x-tuple**: tuples that may be present or absent.

<table>
<thead>
<tr>
<th>ID</th>
<th>Saw(witness, car)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>(Amy, Mazda)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>(Betty, Honda)</td>
<td></td>
</tr>
</tbody>
</table>
Definitions

- x-relation: multiset of x-tuples.
- Construction:
  - Pick one alternative from each x-tuple that is not a maybe-x-tuple
  - Pick zero or one alternative from each x-tuple that is a maybe-x-tuple.

<table>
<thead>
<tr>
<th>ID</th>
<th>Saw(witness, car)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>(Amy, Mazda)</td>
</tr>
<tr>
<td>23</td>
<td>(Betty, Honda)</td>
</tr>
</tbody>
</table>
Combining Lineage and Uncertainty
ULDB

- Triple $(\bar{R}, S, \lambda)$
- $\bar{R}$: set of $x$-relations
- $S$: set of symbols containing $I(\bar{R})$
- $\lambda$: lineage function from $S$ to $2^S$
- $I(\bar{R})$: tuple alternatives $(i, j)$
  - $i$: $x$ tuple
  - $j$: index of one of its alternatives
### Accuses with Uncertainty and Lineage

<table>
<thead>
<tr>
<th>ID</th>
<th>Saw (witness, car)</th>
<th>Drives (person, car)</th>
<th>Accuses (witness, person)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>(Amy, Mazda)</td>
<td></td>
<td>(Amy, Toyota)</td>
</tr>
<tr>
<td>23</td>
<td>(Betty, Honda)</td>
<td>32</td>
<td>(Jimmy, Toyota)</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>33</td>
<td>(Billy, Mazda)</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>34</td>
<td>(Billy, Honda)</td>
</tr>
<tr>
<td>41</td>
<td>(Amy, Jimmy)</td>
<td>(\lambda(41,1)={(21,1), (31,1)})</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>(Amy, Jimmy)</td>
<td>(\lambda(42,1)={(21,2), (32,1)})</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>(Amy, Billy)</td>
<td>(\lambda(43,1)={(21,1), (33,1)})</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>(Betty, Billy)</td>
<td>(\lambda(44,1)={(23,1), (34,1)})</td>
<td></td>
</tr>
</tbody>
</table>
LDB of a ULBD $D$

- $D_k$ is a possible LDB of $D$, $S_k \subseteq S$
- $D_k$ is the triple $(R_k, S_k, \lambda_k)$
- $R_k$ includes exactly the alternatives of $x$-tuples in $\bar{R}$ such that $s_{(i,j)} \in S_k$ and $\lambda_k$ is the restriction of $\lambda$ to $S_k$
Conditions

- Let $s_{(i, j)} \in S_k$, then for every $j' \neq j$, $s(i, j') \notin S_k$
- $\forall s_{(i, j)} \in S_k$, $\lambda(s_{(i, j)}) \subseteq S_k$
- If for some $x$-tuple $t_i$, there does not exist a $s_{(i, j)} \in S_k$, then $t_i$ is a maybe $x$-tuple, and $\forall s_{(i, j)} \in t_i$, $\lambda(s_{(i, j)}) = \emptyset$ or $\lambda(s_{(i, j)}) \not\subseteq S_k$
Example:

- Let \( s_{(i, j)} \in S_k \), then for every \( j' \neq j \), \( s(i,j') \notin S_k \)
- \( \forall s_{(i, j)} \in S_k, \lambda(s_{(i, j)}) \subseteq S_k \)
- If for some \( x \)-tuple \( t_i \) there does not exist a \( s_{(i, j)} \in S_k \), then \( t_i \) is a maybe \( x \)-tuple, and \( \forall s_{(i, j)} \in t_i, \lambda(s_{(i, j)}) = \emptyset \) or \( \lambda(s_{(i, j)}) \not\in S_k \)
Completeness for ULDBs

- Given any set of possible LDBs $P = \{P_1, P_2, \ldots, P_m\}$ over relations $R = \{R_1, R_2, \ldots, R_n\}$, there exists a ULDB $D = (R, S, \lambda)$ whose possible LDBs are $P$. 
Well-Behaved Lineage

The lineage of an $x$-tuple $t_i$ is well-behaved if it satisfies the following three conditions:

1. **Acyclic**: $\forall s_{(i,j)}, s_{(i,j)} \not\in \lambda^*(s_{(i,j)})$

2. **Deterministic**: $\forall s_{(i,j)}, s_{(i,j')},$ if $j \neq j'$ then either $\lambda(s_{(i,j)}) \neq \lambda(s_{(i,j')})$ or $\lambda(s_{(i,j)}) = \emptyset$.

3. **Uniform**: $\forall s_{(i,j)}, s_{(i,j')}, B(s_{(i,j)}) = B(s(i, j'))$, where $B(s(i,j)) = \{t_k | \exists s_{(k,l)}, s_{(k,l)} \in \lambda(s(i, j))\}$
Conditions:

1. There are no cycles;
2. All alternatives of an x-tuple have distinct lineage; and
3. Their lineage points to alternatives of the exact same set of x-tuples.
Querying ULDBs
DL-monotonic query

Function $Q$ from LDBs to LDBs that satisfies the following conditions:

- Constrains the lineage of a result tuple to be a minimal subset of the database that produces exactly that tuple.
- Enforces monotonicity on both data and lineage.
DL Monotonic Example

### Saw

<table>
<thead>
<tr>
<th>ID</th>
<th>witness</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Amy</td>
<td>Mazda</td>
</tr>
<tr>
<td>22</td>
<td>Amy</td>
<td>Toyota</td>
</tr>
<tr>
<td>23</td>
<td>Betty</td>
<td>Honda</td>
</tr>
</tbody>
</table>

### Drives

<table>
<thead>
<tr>
<th>ID</th>
<th>person</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Jimmy</td>
<td>Mazda</td>
</tr>
<tr>
<td>32</td>
<td>Jimmy</td>
<td>Toyota</td>
</tr>
<tr>
<td>33</td>
<td>Billy</td>
<td>Mazda</td>
</tr>
<tr>
<td>34</td>
<td>Billy</td>
<td>Honda</td>
</tr>
</tbody>
</table>

### Accuses

<table>
<thead>
<tr>
<th>ID</th>
<th>witness</th>
<th>person</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>Amy</td>
<td>Jimmy</td>
</tr>
<tr>
<td>42</td>
<td>Amy</td>
<td>Jimmy</td>
</tr>
<tr>
<td>43</td>
<td>Amy</td>
<td>Billy</td>
</tr>
<tr>
<td>44</td>
<td>Betty</td>
<td>Billy</td>
</tr>
</tbody>
</table>

\[
\lambda(41) = \{21, 31\}
\lambda(42) = \{22, 32\}
\lambda(43) = \{21, 33\}
\lambda(44) = \{23, 34\}\]
ULDB Minimality
Data Minimality

- An alternative \((i, j)\) of an \(x\)-tuple \(t_i\) in a ULDB \(D\) is said to be extraneous if removing it from the \(x\)-relation does not change the possible instances of \(D\).
- A ‘?’ on an \(x\)-tuple in \(D\) is said to be extraneous if removing it does not change the possible instances of \(D\).
- A ULDB \(D\) is \(D\)-minimal if it does not include any extraneous alternatives or ‘?’s.
Example

<table>
<thead>
<tr>
<th>ID</th>
<th>Saw(witness, car)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Carol, Acura)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Car1(car)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Acura</td>
</tr>
<tr>
<td>3</td>
<td>Lexus</td>
</tr>
</tbody>
</table>

\[ \text{Saw1} = (\text{Car1} \bowtie \text{Saw}) \quad \text{Saw2} = (\text{Car2} \bowtie \text{Saw}) \]

\[ \lambda(4,1) = \{(1,1), (2,1)\} \quad \lambda(5,1) = \{(1,2), (3,1)\} \]

\[ (\text{Saw1} \bowtie \text{witness} \text{ Saw2}) \]

<table>
<thead>
<tr>
<th>ID</th>
<th>(witness, car1, car2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>(Carol, Acura, Lexus)</td>
</tr>
</tbody>
</table>

\[ \lambda(6,1) = \{(4,1), (5,1)\} \]

Extraneous
Lineage Minimality

- A ULDB $D = (\bar{R}, S, \lambda)$ is $L$-minimal if for any $D' = (\bar{R}, S', \lambda')$ over the same x-relations $\bar{R}$ such that:
  - $S' \subseteq S$, $\lambda'^* \subseteq \lambda^*$
  - $D$ and $D'$ have the same internal lineage
  - $D'$ has the same possible instances as $D$ only if $S' = S$ and $\lambda'^* = \lambda^*$. 
Membership Queries

- **Tuple Membership**: determine whether \( t \in R \), in some (resp. all) possible instance(s) of \( D \).

- **Instance Membership**: Given a ULDB \( D \) containing a relation \( R \), and a multiset \( T \) of tuples, determine whether \( R \) contains exactly the tuples of \( T \) in some (or all) possible instance(s) of \( D \).
Extraction: Extract a subset of the database

- Let D be a well-behaved ULDB with x-relations R and possible instances P, and let $\bar{X}$ be a subset of R.
- The problem of extracting $\bar{X}$ from R is to return a well-behaved ULDB $D'$ with $R' = \bar{X}$ and possible instances $P'$, such that the restriction of P to $\bar{X}$ equals $P'$ with respect to data and internal lineage
Confidences and Probabilistic Data
Confidences and Probabilistic Data

- Each base alternative \( a \) is associated with a confidence value \( c(a) \)
- The probability of a possible instance is the product of the confidences of the base alternatives and ‘?’ chosen in it
Example

- Instance where Amy saw an Acura, Betty saw a Mazda, and Hank does not drive an Acura has confidence $0.8 \times 0.6 \times (1 - 0.6) \approx 0.20$
Query Processing

- **Data Computation**: in which we compute the data and lineage in query results, just as in ULDBs without confidences
- **Confidence Computation**: in which we compute confidence values for query results based on their lineage (and confidence values on base data)
Related Work

- Query answering in probabilistic databases.
- Approximate query answering.
- Integrating lineage (provenance)
Conclusions

- ULDBs can represent any finite set of possible instances containing data and lineage.
- ULDBs can be extended naturally to represent and query probabilistic data.
- ULDB allows query evaluation to be decoupled from computation of confidences.
Future Work

- Algorithms and Optimizations when computing confidences
- On-demand confidence computation
- Incremental propagation of confidence updates
- Ordering queries based on confidences
- Efficient update algorithms
- Implementation, Theory.
- Incomplete relations, versioning of data, uncertainty and lineage
References