Top-k Query Processing in Probabilistic Databases with Non-Materialized Views

Maximilian Dylla
Iris Miliaraki
Martin Theobald
Tuple-Independent Probabilistic Database

**Encounter**

<table>
<thead>
<tr>
<th>Kangaroo</th>
<th>Shark</th>
<th>Crocodile</th>
<th>Crocodile</th>
<th>Cane Toad</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>0.8</td>
<td>0.25</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.95</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fatality**

<table>
<thead>
<tr>
<th>Crocodile</th>
<th>Kangaroo</th>
<th>Shark</th>
<th>Cane Toad</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>0.7</td>
<td>0.05</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Queries

Query

\( \text{LuckyEncounter}(?A) \)

Deduction rule

\( \text{LuckyEncounter}(?A) \leftarrow \exists ?L \ \text{Encounter}(?A, ?L) \land \neg \text{Fatality}(?A) \)

Answer

0.08

LuckyEncounter(Crocodile)

\( \lor \)

\( \land \)

\( \neg \)

\( \land \)

\( \land \)

Encounter(Crocodile, Water) 0.2

Fatality(Crocodile) 0.7

Encounter(Crocodile, Land) 0.1
Top-k Answers

Query $\text{LuckyEncounter}(?A), k = 2$

- **LuckyEncounter(Cane Toad)**: $p = 0.94$
- **LuckyEncounter(Kangaroo)**: $p = 0.76$
- **LuckyEncounter(Shark)**: $p = 0.2$
- **LuckyEncounter(Crocodile)**: $p = 0.08$

![Diagram](image-url)
First-Order Lineage

\[ \text{LuckyEncounter(?A)} \leftarrow \exists ?L \ \text{Encounter(?A, ?L)} \land \neg \text{Fatality(?A)} \]
$\text{LuckyEncounter}(?A) \leftarrow \exists \ ?L \ \text{Encounter}(?A, ?L) \land \neg \text{Fatality}(?A)$
Scheduling

Impact on answer's bounds:

\[ \frac{d}{dg} = P \left( \phi[g \rightarrow true] \right) - P \left( \phi[g \rightarrow false] \right) \]

<table>
<thead>
<tr>
<th>Answers:</th>
<th>Bounds:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Crocodile</td>
<td>[0.03, 0.3]</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>Kangaroo</td>
<td>[0.0, 0.8]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?A ?A ≠ Crocodile ?A ≠ Kangaroo</td>
<td>[0.0, 1.0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encounter (Crocodile, ?L)</td>
<td>[0.0, 1.0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?L ≠ Land</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fatality (Crocodile)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encounter (Crocodile, Land)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encounter(?A, ?L)</td>
<td>[0.0, 1.0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?A ≠ Crocodile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>?A ≠ Kangaroo</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Impact: 0.27

1.0
Sorted Input Lists

Rule: \( \text{LuckyEncounter}(?A, ?B) \leftarrow \text{Encounter}(?A, ?B) \land \neg \text{Fatality}(?A) \)
Sorted Input Lists

Rule: \( \text{LuckyEncounter}(?A, ?B) \leftarrow \text{Encounter}(?A, ?B) \land \neg \text{Fatality}(?A) \)

Applicable if all variables are query variables!
Recursion

Rule: \( LocatedIn(?A, ?B) \leftarrow \exists ?C \ PartOf(?A, ?C) \land LocatedIn(?C, ?B) \)

Block cycles during grounding

Theorem:
Expanding a cycle more than once does not alter the validity of a lineage formula.
Experiments: Query Classes

Data: Imdb, 26M tuples, uniformly sampled confidences.
Queries: Each query pattern instantiated by 1000 constants.
Experiments: Performance Factors

\[ Q(A, B) \leftarrow \exists X \, R1(X, A) \land R2(X, B) \]

Answer

ms

- Top-10
- Top-20
- Top-50
- Postgres
- MayBMS
- Trio
Experiments: Performance Factors

\[ Q(A, B) \leftarrow \exists X R_1(X, A) \land R_2(X, B) \]

Answer

\[ \land \land \land \land \land \]

\[ \text{ms} \]

- Top-10
- Top-20
- Top-50
- Postgres
- MayBMS
- Trio
Experiments: Performance Factors

\[ Q(A, B) \leftarrow \exists X R1(X, A) \land R2(X, B) \]

Answer

\[ \land \]

R1, R2

Answer

\[ \lor \]

R1, R2, R3, R4, R5

ms

- Top-10
- Top-20
- Top-50
- Postgres
- MayBMS
- Trio
Experiments: Performance Factors

\[ Q(A, B) \leftarrow \exists X \ R1(X, A) \land R2(X, B) \]

Answer

\[ \land \]

R1 \quad R2

Answer

\[ \land \]

R1 \quad R2

\[ \land \]

ms

- Top-10
- Top-20
- Top-50
- Postgres
- MayBMS
- Trio
Summary

First-order lineage representing sets of answers + bounds on probabilities.

Integration of data and confidence computations.

Support for all select-project-join queries.