Trio: A System for Data, Uncertainty, and Lineage

Jennifer Widom

Outline of Talk

1. The Motivation
2. The Discovery
3. The Vision
4. The Present
5. The Future

The Motivation

- Lots of applications have uncertain data (approximate, incomplete, imprecise, inaccurate, ...)
- Lots of the same applications need to track data lineage

Coincidence or Fate?

- Neither is supported by conventional Database Management Systems (DBMSs)

Applications

Deduplication

- Uncertainty: Match and merge
- Lineage: Source records

Information extraction

- Uncertainty: Extracted labels and values
- Lineage: Original context

Information integration

- Uncertainty: Inconsistent information
- Lineage: Original sources

Applications

Scientific experiments

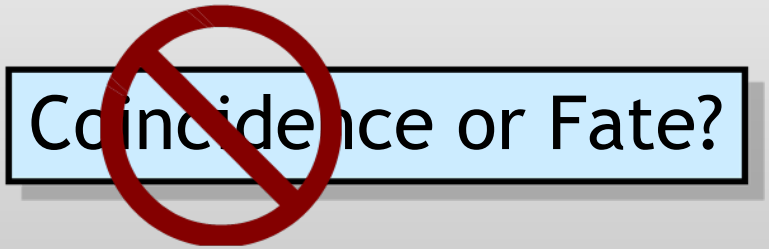
- Uncertainty: Captured (and derived) data
- Lineage: Layers of views

Sensor data

- Uncertainty: Sensor values, missing readings
- Lineage: Original readings, views

The Discovery

The connection between uncertainty and lineage goes deeper than just a shared need by several applications

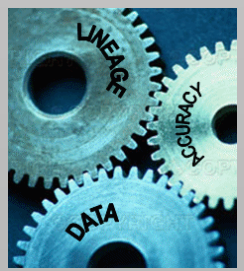


Lineage and Uncertainty

Lineage...

- Enables simple and consistent representation of uncertain data
- Correlates uncertainty in query results with uncertainty in the input data
- Can make computation over uncertain data more efficient

Applications use lineage to reduce or resolve uncertainty



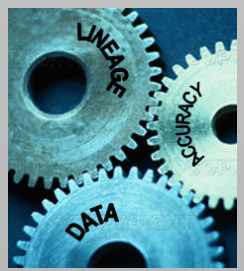
The Vision

A new kind of DBMS in which:

1. Data
2. Uncertainty
3. Lineage

Trio

are all first-class interrelated concepts



The Trio Trio

1. Data Model

   Simplest extension to relational model that's sufficiently expressive

2. Query Language

   Simple extension to SQL with well-defined semantics and intuitive behavior

3. System

   A complete open-source DBMS that people want to use

The Present

1. Data Model

   Uncertainty-Lineage Databases (ULDBs)

2. Query Language

   TriQL

3. System

   First prototype built on top of standard DBMS

Running Example: Crime-Solving

Saw(witness,car) // may be uncertain

Owns(owner,car) // may be uncertain

Suspects(person) = π_{owner}(Saw ⋈ Owns)

Data Model: Uncertainty

An uncertain database represents a set of possible instances

- *Amy saw either a Honda or a Toyota*
- *Jimmy owns a Toyota, a Mazda, or both*
- *Betty saw an Acura with confidence 0.5 or a Toyota with confidence 0.3*
- *Hank is a suspect with confidence 0.7*

Our Model for Uncertainty

1. Alternatives
2. ‘?’ (Maybe) Annotations
3. Confidences

Our Model for Uncertainty

1. **Alternatives:** uncertainty about value

2. '?' (Maybe) Annotations

3. Confidences

Saw (witness,car)
(Amy, Honda) // (Amy, Toyota) // (Amy, Mazda)

Three possible instances

=

witness	car
Amy	{ Honda, Toyota, Mazda }

Our Model for Uncertainty

1. Alternatives
2. **‘?’ (Maybe):** uncertainty about existence
3. Confidences

Saw (witness,car)
(Amy, Honda) // (Amy, Toyota) // (Amy, Mazda)
(Betty, Acura)

?

Six possible instances

Our Model for Uncertainty

1. Alternatives
2. '?' (Maybe) Annotations
3. **Confidences:** weighted uncertainty

Saw (witness,car)
(Amy, Honda): 0.5 // (Amy,Toyota): 0.3 // (Amy, Mazda): 0.2
(Betty, Acura): 0.6

?

Six possible instances,
each with a probability

Models for Uncertainty

- Our model (so far) is not especially new
- We spent some time exploring the space of models for uncertainty [two papers]
- Tension between understandability and expressiveness
  - Our model is understandable
  - But it is not complete, or even closed under common operations

Closure and Completeness

Completeness

Can represent all sets of possible instances

Closure

Can represent results of operations

Note: Completeness $\Rightarrow$ Closure

Our Model is Not Closed

Saw (witness,car)
(Cathy, Honda) // (Cathy, Mazda)

Owns (owner,car)
(Jimmy, Toyota) // (Jimmy, Mazda)
(Billy, Honda)
(Hank, Honda)

Suspects = Π_{owner}(Saw ⋈ Owns)

Suspects	
Jimmy	?
Billy	?
Hank	?

Cannot correctly capture possible instances in the result

Lineage to the Rescue

Lineage (provenance): "where data came from"

- Internal lineage
- External lineage

In Trio: A function **λ** from alternatives to other alternatives (or external sources)

Example with Lineage

ID	Saw (witness,car)
11	(Cathy, Honda) // (Cathy, Mazda)

ID	Owns (owner,car)
21	(Jimmy, Toyota) // (Jimmy, Mazda)
22	(Billy, Honda)
23	(Hank, Honda)

$$\text{Suspects} = \pi_{\text{owner}}(\text{Saw} \bowtie \text{Owns})$$

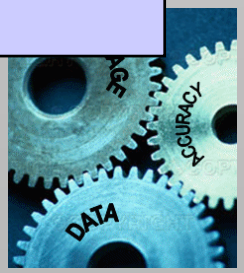
ID	Suspects
31	Jimmy
32	Billy
33	Hank

? $\lambda(31) = (11,2),(21,2)$

? $\lambda(32) = (11,1), 22$

? $\lambda(33) = (11,1), 23$

Correctly captures possible instances in the result



Trio Data Model

Uncertainty-Lineage Databases (ULDBs)

[recent paper]

1. Alternatives
2. '?' (Maybe) Annotations
3. Confidences
4. Lineage

ULDBs are closed and complete

ULDB Results

Conjunctive lineage sufficient for most operations

- Negative lineage for difference
- Disjunctive lineage for duplicate-elimination

Minimality of representations

- Data-minimal
- Lineage-minimal

Membership problems

Extraction of a relation from a ULDB

Querying ULDBs

TriQL

- Simple extension to SQL
- Formal semantics, intuitive meaning
- Ability to query confidences and lineage directly

TriQL Example

ID	Saw (witness,car)
11	(Cathy, Honda) // (Cathy, Mazda)

ID	Owns (owner,car)
21	(Jimmy, Toyota) // (Jimmy, Mazda)
22	(Billy, Honda)
23	(Hank, Honda)

```
SELECT Owns.person INTO Suspects
FROM Saw, Owns
WHERE Saw.car = Owns.car
```

ID	person
31	Jimmy
32	Billy
33	Hank

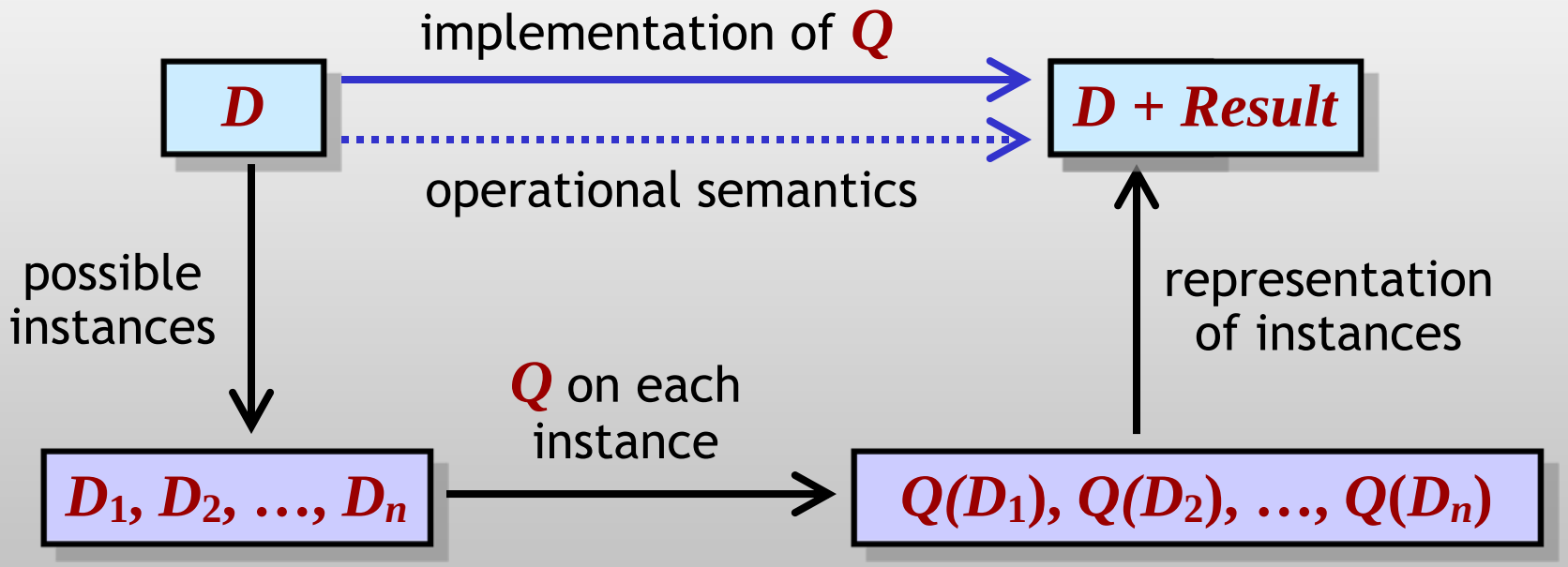
? λ(31) = (11,2),(21,2)

? λ(32) = (11,1), 22

? λ(33) = (11,1), 23

Formal Semantics

Query Q on ULDB D



TriQL: Querying Confidences

Built-in function: conf()

```
SELECT Owns.person INTO Suspects
FROM Saw, Owns
WHERE Saw.car = Owns.car
AND conf(Saw) > 0.5 AND conf(Owns) > 0.8
```

TriQL: Querying Lineage

Built-in join predicate: lineage()

```
SELECT Saw.witness INTO AccusesHank
FROM Suspects, Saw
WHERE lineage(Suspects,Saw)
AND Suspects.person = 'Hank'
```

Also lineage*()

Computing Confidences

Previous approach (probabilistic databases):

- Each operator computes confidences during query execution
- Only certain query plans allowed

Our approach

- Use any query plan
- Compute confidences afterwards based on lineage

The Trio System

Version 1

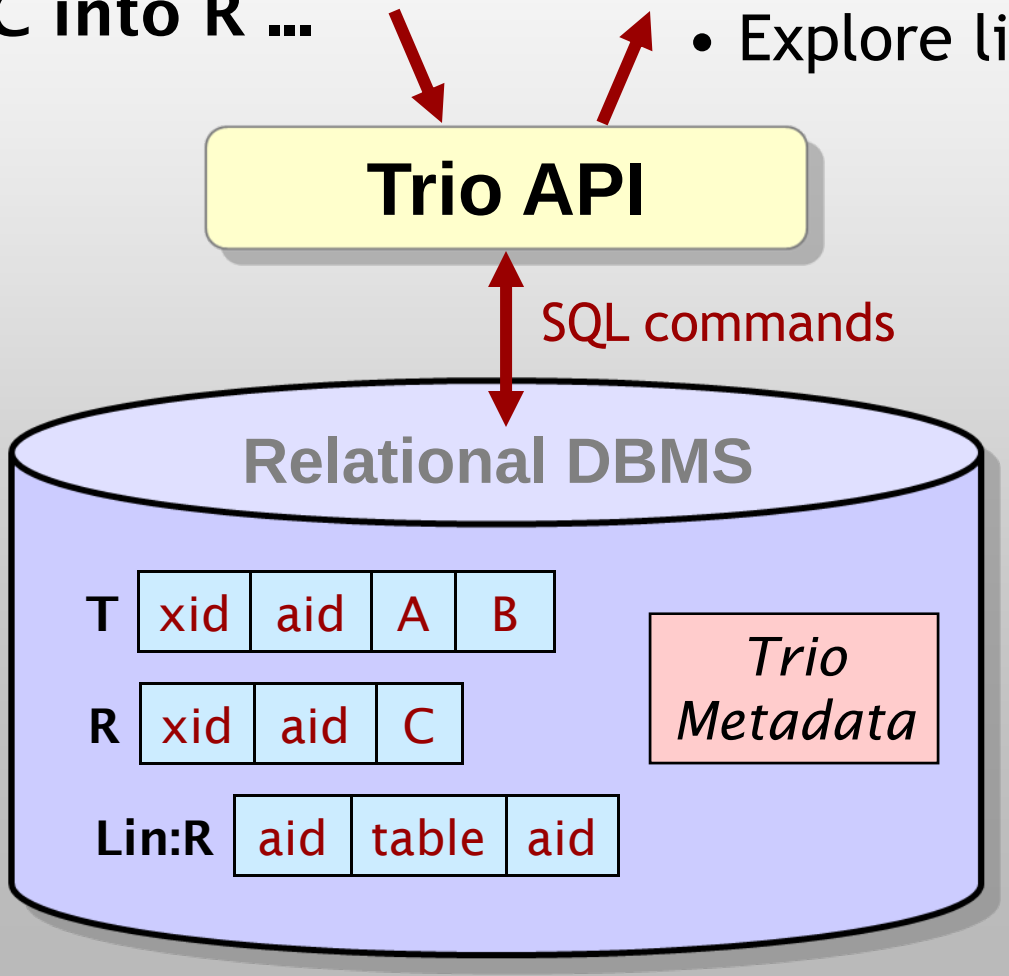
- Entirely on top of conventional DBMS
- Surprisingly easy and complete, reasonably efficient

The Trio System: Version 1

create trio table T(A,B)

select C into R …

- Result cursors
- Browse tables
- Explore lineage

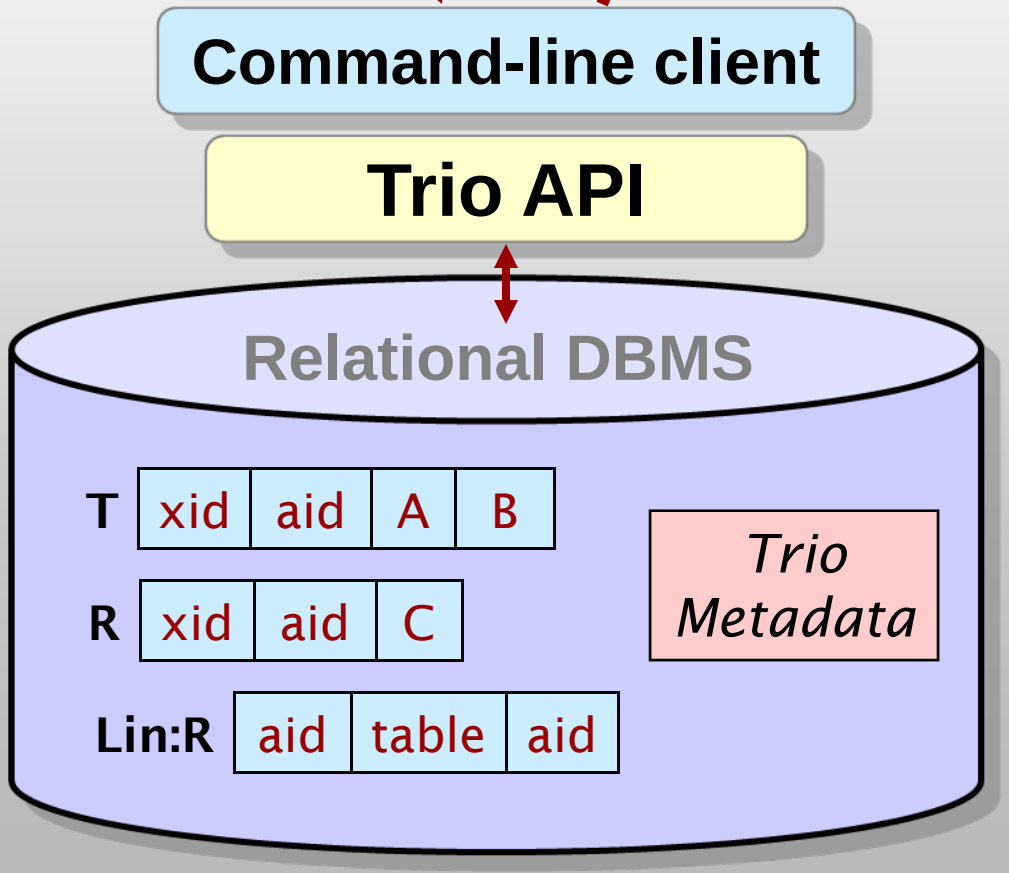


The Trio System: Version 1

create trio table T(A,B)

select C into R …

- Result cursors
- Browse tables
- Explore lineage

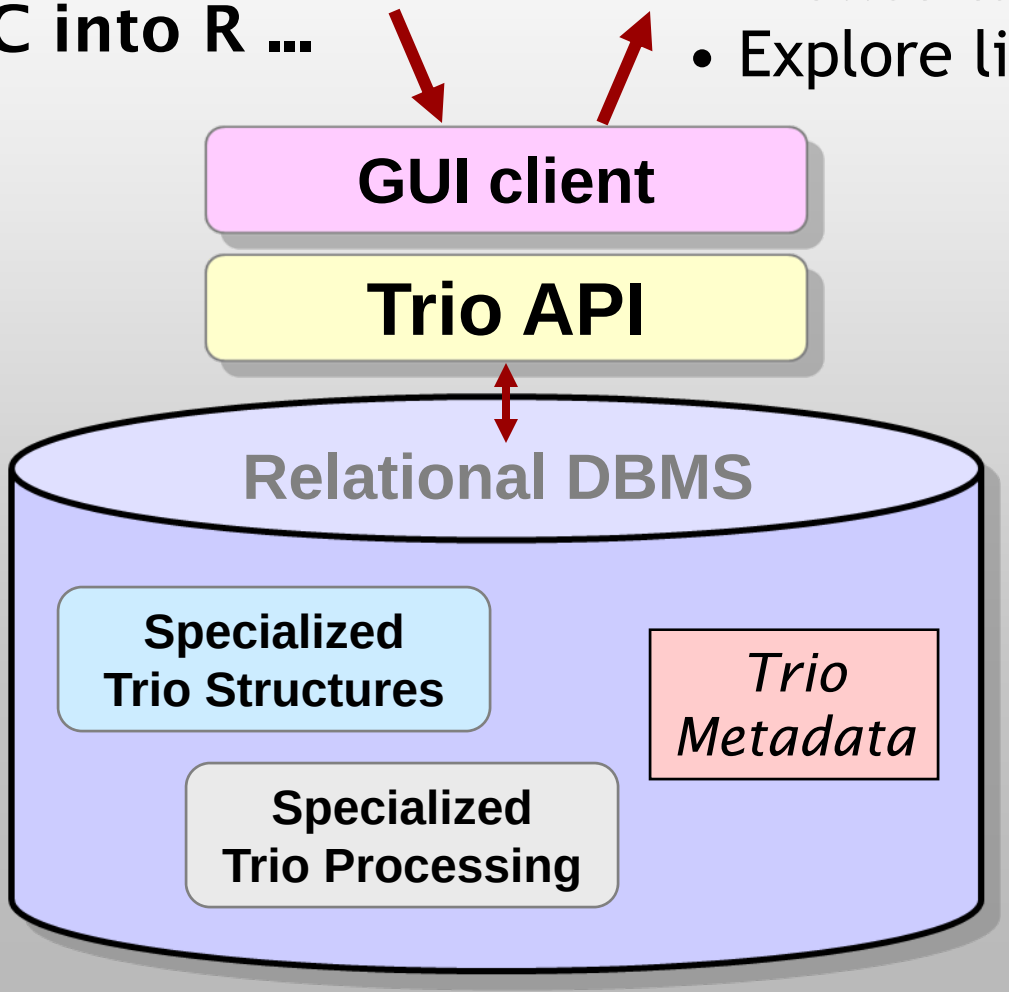


The Trio System: Version 2

create trio table T(A,B)

select C into R …

- Result cursors
- Browse tables
- Explore lineage



Current Topics

Confidence computation

- Minimize lineage traversal; memoization; batch computations

Updates

- Primitive operations; TriQL update statements

Additional query constructs

- “Horizontal” operators; top-*k* by confidence

System

- Keep up with research; GUI

Future Directions

Theory, Model, Algorithms

- Unlimited opportunities

System

- Storage, indexing, partitioning
- Statistics and query optimization

Long Range

- Continuous uncertainty; incomplete relations
- External lineage; versioning

but don't forget
the lineage…

Stanford Lab

Search "stanford trio"

[overview paper]

Trio group:

Parag Agrawal, Omar Benjelloun, Anish Das Sarma,
Chris Hayworth, Shubha Nabar, Jennifer Widom

Special thanks to:

Ashok Chandra, Alon Halevy, Jeff Ullman