Trio: A System for Data, Uncertainty, and Lineage

Jennifer Widom

Stanford University

Enjoy uncertainty.

Motivation

- Many applications involve data that is uncertain (approximate, probabilistic, inexact, incomplete, imprecise, fuzzy, inaccurate, …)
- Many of the same applications need to track the lineage of their data

Coincidence or Fate?

Neither uncertainty nor lineage are supported by conventional Database Management Systems (DBMSs)

Sample 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

Sample Applications

Scientific experiments

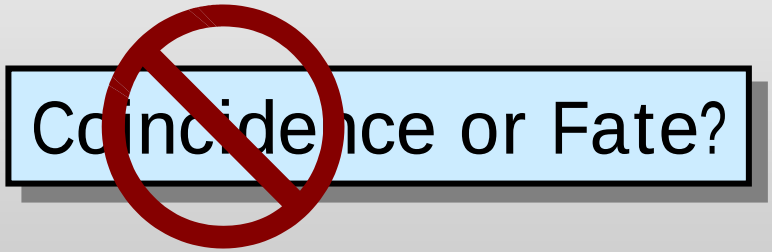
- Uncertainty: Captured (and derived) data
- Lineage: Layers of views

Sensor data

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

Claim

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

Goal

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

Drives(person,car) // may be uncertain

Suspects(person) = π_{person}(Saw $\bowtie$ Drives)

Data Model: Uncertainty

An uncertain database represents a set of possible instances

- *Amy saw either a Honda or a Toyota*
- *Jimmy drives 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 presence

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)

Drives (person,car)
(Jimmy, Toyota) ‖ (Jimmy, Mazda)
(Billy, Honda) ‖ (Frank, Honda)
(Hank, Honda)

$$\text{Suspects} = \pi_{\text{person}}(\text{Saw} \bowtie \text{Drives})$$

Suspects	
Jimmy	?
Billy ‖ Frank	?
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	Drives (person,car)
21	(Jimmy, Toyota) ‖ (Jimmy, Mazda)
22	(Billy, Honda) ‖ (Frank, Honda)
23	(Hank, Honda)

$$\text{Suspects} = \pi_{\text{person}}(\text{Saw} \bowtie \text{Drives})$$

ID	Suspects		
31	Jimmy	?	$\lambda(31) = (11,2),(21,2)$
32	Billy ‖ Frank	?	$\lambda(32,1) = (11,1),(22,1)$; $\lambda(32,2) = (11,1),(22,2)$
33	Hank	?	$\lambda(33) = (11,1), 23$

Correctly captures possible instances in the result

Trio Data Model

Uncertainty-Lineage Databases (ULDBs)

[paper]

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

<u>ULDBs are closed and complete</u>

ULDBs: Lineage

Conjunctive lineage sufficient for most operations

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

ULDBs: Minimality

A ULDB relation R represents a set of possible instances

- ✶ Does every tuple in R appear in some possible instance? (no extraneous tuples)

  Data-minimality

- ✶ Does every *maybe*-tuple in R *not appear* in some possible instance? (no extraneous '?'s)

Also Lineage-minimality

Data Minimality Examples

Extraneous '?'

	. . .
20	Billy ‖ Frank
	. . .

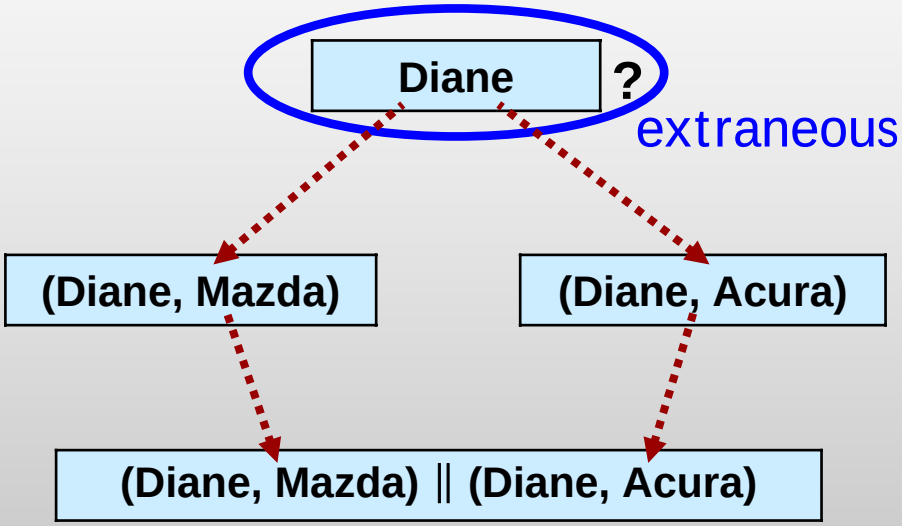
? $\lambda(20,1)=(10,1)$; $\lambda(20,2)=(10,2)$

extraneous

	. . .
10	(Billy, Honda) ‖ (Frank, Honda)
	. . .

Data Minimality Examples

Extraneous tuple



ULDBs: Membership Questions

- Does a given tuple t appear in some (all) possible instance(s) of R ?

  Polynomial algorithms based on data-minimization

- Is a given table T one of (all of) the possible instances of R ?

  NP-Hard

Non-Theorists…

Wake Back Up!

Querying ULDBs

TriQL

- Simple extension to SQL
- Formal semantics, intuitive meaning
- Query uncertainty, confidences, and lineage

Initial TriQL Example

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

ID	Drives (person,car)
21	(Jimmy, Toyota) ‖ (Jimmy, Mazda)
22	(Billy, Honda) ‖ (Frank, Honda)
23	(Hank, Honda)

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

ID	Suspects
31	Jimmy
32	Billy ‖ Frank
33	Hank

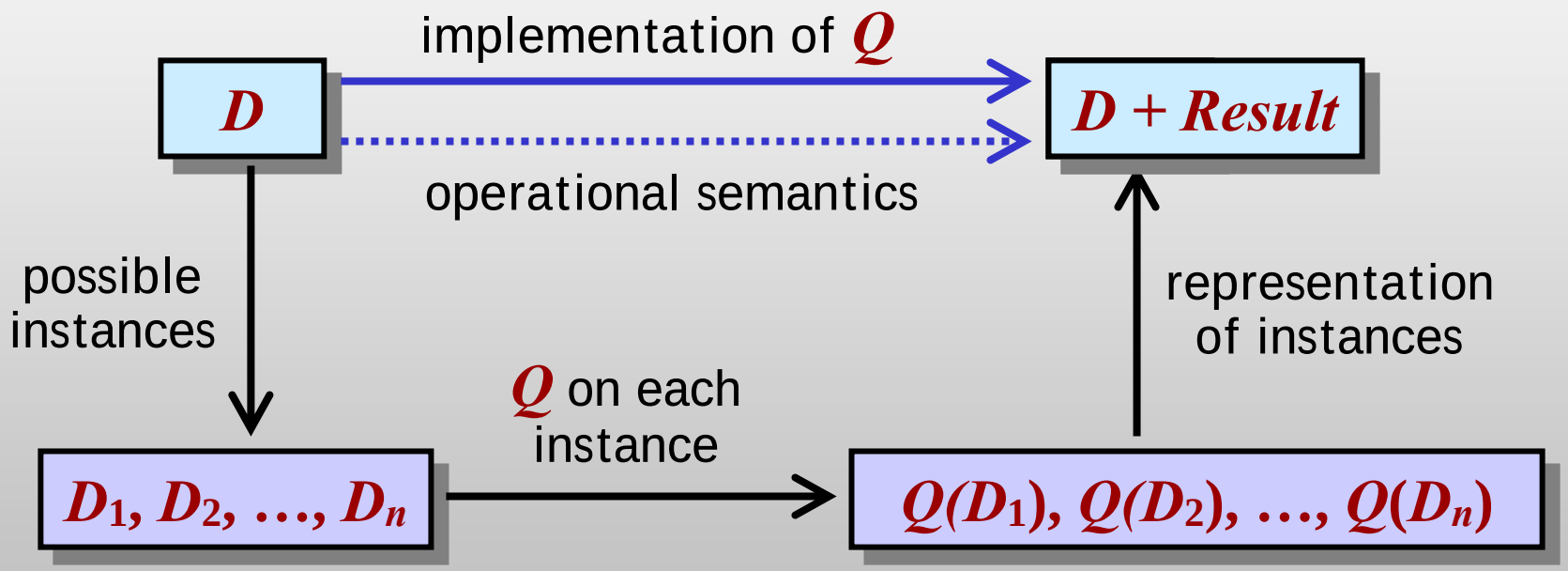
? $\lambda(31) = (11,2),(21,2)$

? $\lambda(32,1)=(11,1),(22,1)$; $\lambda(32,2)=(11,1),(22,2)$

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

Formal Semantics

Query Q on ULDB D



TriQL: Querying Confidences

Built-in function: Conf()

```
SELECT Drives.person INTO Suspects
FROM Saw, Drives
WHERE Saw.car = Drives.car
AND Conf(Saw) > 0.5 AND Conf(Drives) > 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*()

Operational Semantics

```
SELECT attr-list [ INTO table ]
FROM X1, X2, …, Xn
WHERE predicate
```

Over conventional relational database:

For each tuple in cross-product of $X_1, X_2, \ldots, X_n$

1. Evaluate the predicate
2. If true, project attr-list to create result tuple
3. If INTO clause, insert into table

Operational Semantics

SELECT attr-list [INTO table]
FROM X_1, X_2, …, X_n
WHERE predicate

Over ULDB:

For each tuple in cross-product of X_1, X_2, …, X_n

1. Create "super tuple" T from all combinations of alternatives
2. Evaluate predicate on each alternative in T; keep only the true ones
3. Project attr-list on each alternative to create result tuple
4. Details: '?', lineage, confidences

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

(Cathy,Honda,Jim,Mazda)‖(Cathy,Honda,Bill,Mazda)‖(Cathy,Mazda,Jim,Mazda)‖(Cathy,Mazda,Bill,Mazda)

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

(Cathy,Honda,Jim,Mazda)‖(Cathy,Honda,Bill,Mazda)‖(Cathy,Mazda,Jim,Mazda)‖(Cathy,Mazda,Bill,Mazda)

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

(Cathy,Honda,Jim,Mazda)‖(Cathy,Honda,Bill,Mazda)‖(Cathy,Mazda,Jim,Mazda)‖(Cathy,Mazda,Bill,Mazda)

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

(Cathy,Honda,Jim,Mazda)‖(Cathy,Honda,Bill,Mazda)‖(Cathy,Mazda,Jim,Mazda)‖(Cathy,Mazda,Bill,Mazda)
(Cathy,Honda,Hank,Honda) ‖ (Cathy,Mazda,Hank,Honda)

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

(Cathy,Honda,Jim,Mazda)‖(Cathy,Honda,Bill,Mazda)‖(Cathy,Mazda,Jim,Mazda)‖(Cathy,Mazda,Bill,Mazda)
(Cathy,Honda,Hank,Honda) ‖ (Cathy,Mazda,Hank,Honda)

Operational Semantics: Example

```
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

(Cathy,Honda,Jim,Mazda)‖(Cathy,Honda,Bill,Mazda)‖(Cathy,Mazda,Jim,Mazda)‖(Cathy,Mazda,Bill,Mazda)
(Cathy,Honda,Hank,Honda) ‖ (Cathy,Mazda,Hank,Honda)

Operational Semantics: Example

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

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

Suspects	
Jim ‖ Bill	**?** $\lambda(\ ) = \ldots$
Hank	**?** $\lambda(\ ) = \ldots$

Confidences

Confidences supplied with base data

Trio computes confidences on query results

- Default probabilistic interpretation
- Can choose to plug in different arithmetic

Saw (witness,car)
(Cathy, Honda): 0.6 ‖ (Cathy, Mazda): 0.4

Drives (person,car)	
(Jim, Mazda): 0.3 ‖ (Bill, Mazda): 0.6	?
(Hank, Honda)	

Min

Suspects	
Jim 0.3 ‖ Bill: 0.4	?
Hank: 0.6	?

Confidence Computation

Previous approach (probabilistic databases)

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

In ULDBs

Confidence of alternative A is function of confidences in $\lambda^*(A)$

Our approach

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

Additional Query Constructs

- "Horizontal subqueries"
  Refer to tuple alternatives as a relation
- *Unmerged* (horizontal duplicates)
- *Flatten*, *GroupAlts*
- *NoLineage*, *NoConf*, *NoMaybe*
- Query-computed confidences
- Data modification statements

Final Example Query

PrimeSuspect (crime#, accuser, suspect)
(1, Amy, Jimmy) ‖ (1, Betty, Billy) ‖ (1, Cathy, Hank)
(2, Cathy, Frank) ‖ (2, Betty, Freddy)

Credibility

person	score
Amy	**10**
Betty	**15**
Cathy	**5**

List suspects with conf values based on accuser credibility

Suspects
Jimmy: 0.33 ‖ Billy: 0.5 ‖ Hank: 0.166
Frank: 0.25 ‖ Freddy: 0.75

Final Example Query

PrimeSuspect (crime#, accuser, suspect)
(1, Amy, Jimmy) ‖ (1, Betty, Billy) ‖ (1, Cathy, Hank)
(2, Cathy, Frank) ‖ (2, Betty, Freddy)

Credibility

person	score
Amy	10
Betty	15
Cathy	5

```
SELECT suspect, score/[sum(score)] as conf
FROM (SELECT suspect,
                (SELECT score FROM Credibility C
                 WHERE C.person = P.accuser)
        FROM PrimeSuspect P)
```

Suspects
Jimmy: 0.33 ‖ Billy: 0.5 ‖ Hank: 0.166
Frank: 0.25 ‖ Freddy: 0.75

Final Example Query

PrimeSuspect (crime#, accuser, suspect)
(1, Amy, Jimmy) ‖ (1, Betty, Billy) ‖ (1, Cathy, Hank)
(2, Cathy, Frank) ‖ (2, Betty, Freddy)

Credibility

person	score
Amy	**10**
Betty	**15**
Cathy	**5**

```
SELECT suspect, score/[sum(score)] as conf
FROM (SELECT suspect,
                (SELECT score FROM Credibility C
                 WHERE C.person = P.accuser)
        FROM PrimeSuspect P)
```

Suspects
Jimmy: 0.33 ‖ Billy: 0.5 ‖ Hank: 0.166
Frank: 0.25 ‖ Freddy: 0.75

Final Example Query

PrimeSuspect (crime#, accuser, suspect)
(1, Amy, Jimmy) ‖ (1, Betty, Billy) ‖ (1, Cathy, Hank)
(2, Cathy, Frank) ‖ (2, Betty, Freddy)

Credibility

person	score
Amy	10
Betty	15
Cathy	5

```
SELECT suspect, score/[sum(score)] as conf
FROM (SELECT suspect,
                (SELECT score FROM Credibility C
                 WHERE C.person = P.accuser)
        FROM PrimeSuspect P)
```

Suspects
Jimmy: 0.33 ‖ Billy: 0.5 ‖ Hank: 0.166
Frank: 0.25 ‖ Freddy: 0.75

The Trio System

Version 1

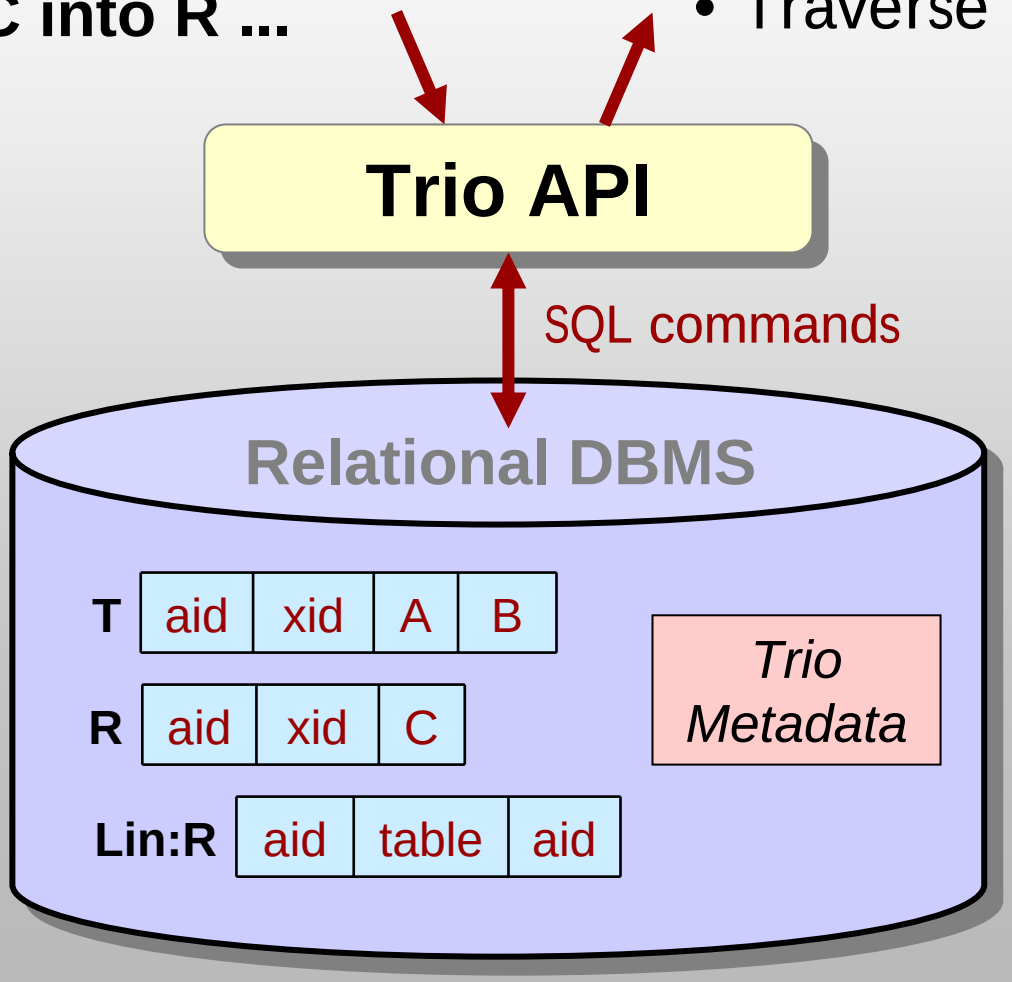
- Entirely on top of conventional DBMS
- Surprisingly easy and complete, reasonably efficient

The Trio System: Version 1

create trio table T(A,B)

select C into R ...

- Result cursors
- Traverse lineage



Relation Encoding

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

Drives (person,car)
(Jim, Mazda) ‖ (Bill, Mazda)
(Hank, Honda)

Suspects	
Jim ‖ Bill	?
Hank	?

Saw

aid	xid	conf	witness	car
11	1	–2	Cathy	Honda
12	1	–2	Cathy	Mazda

Drives

aid	xid	conf	person	car
21	1	–2	Jim	Mazda
22	1	–2	Bill	Mazda
23	2	–2	Hank	Honda

Suspects

aid	xid	conf	person
31	1	–1	Jim
32	1	–1	Bill
33	2	–1	Hank

Lin:Suspects

aidFr	table	aidTo
31	Saw	12
31	Drives	21
32	Saw	12
32	Drives	22
33	Saw	11
33	Drives	23

Relation Encoding with Confidences

Saw (witness,car)
(Cathy, Honda): 0.6 ‖ (Cathy, Mazda): 0.4

Drives (person,car)
(Jim, Mazda): 0.3 ‖ (Bill, Mazda): 0.6
(Hank, Honda)

Saw

aid	xid	conf	witness	car
11	1	0.6	Cathy	Honda
12	1	0.4	Cathy	Mazda

Drives

aid	xid	conf	person	car
21	1	0.3	Jim	Mazda
22	1	0.6	Bill	Mazda
23	2	1.0	Hank	Honda

Suspects

aid	xid	conf	person
31	1	0.12	Jim
32	1	0.24	Bill
33	2	0.6	Hank

Query Translation

Query Q into result table R

1. Run query Q' to produce "super-result" R

   $Q' \approx Q$ but adds aid/xid's of source tuples, joins lineage tables when lineage() predicates

2. Group R into alternatives, generate xid's
3. Move lineage data to $lin{:}R$
4. Compute confidences? (optional)
5. Add metadata: schemas, confidence info, lineage structure

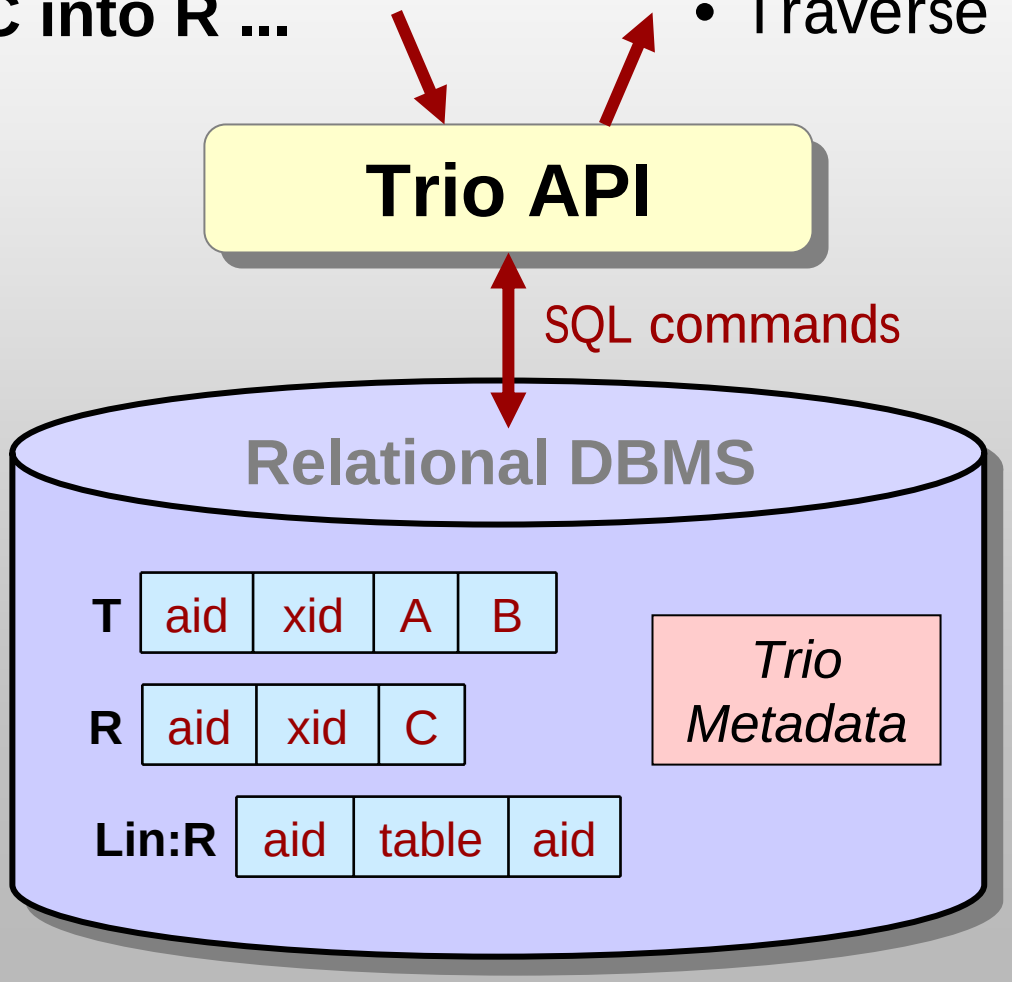
Transient results: stop at 2, return cursor

The Trio System: Version 1

create trio table T(A,B)

select C into R ...

- Result cursors
- Traverse lineage

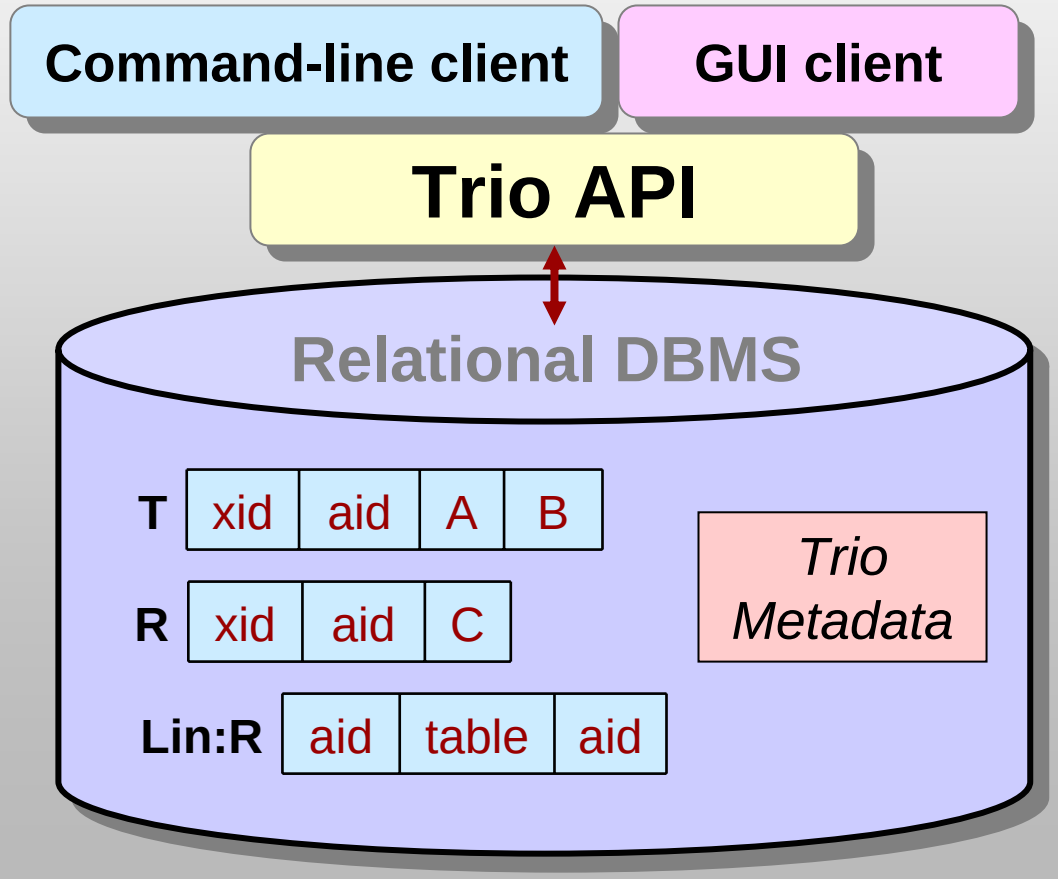


The Trio System: Version 1

create trio table T(A,B)

select C into R ...

- Schema and table browsing
- Result browsing
- Explore lineage

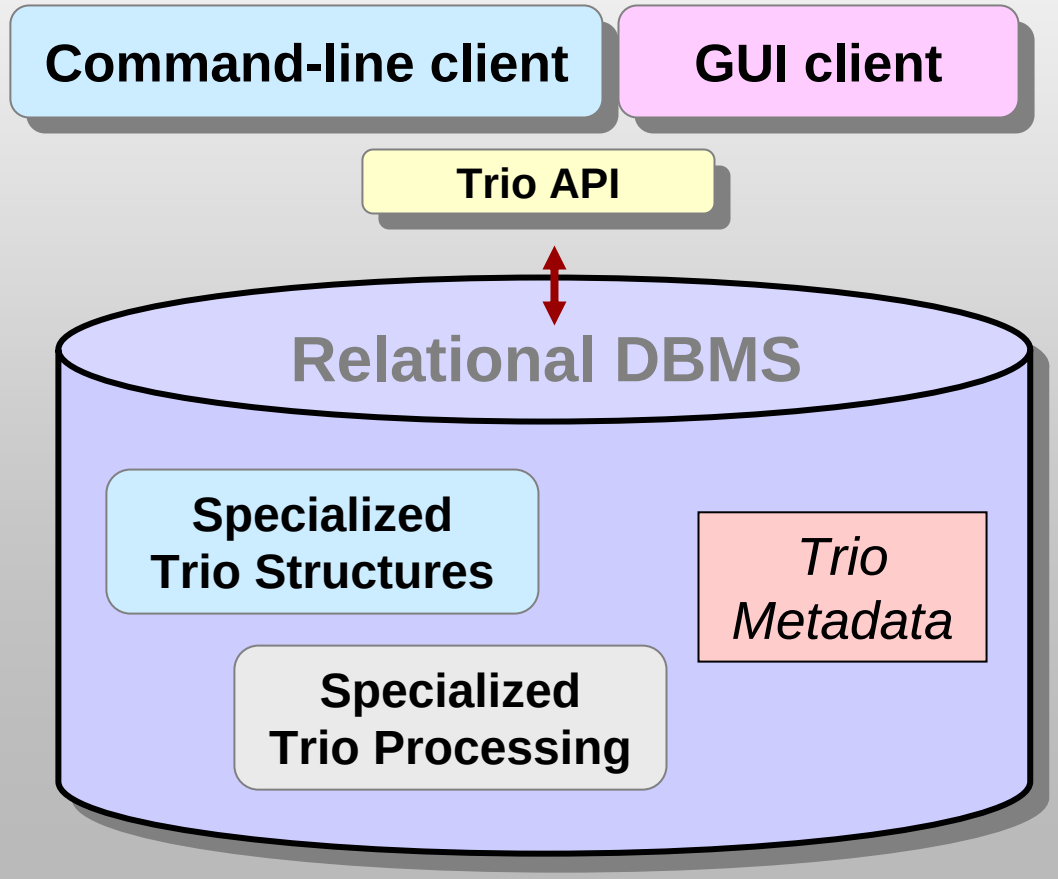


The Trio System: Version 2

create trio table T(A,B)

select C into R ...

- Schema and table browsing
- Result browsing
- Explore lineage



Current Topics

System

- Full query language
- Nice interface
- Performance experiments
- Demo applications

Algorithms: confidence computation, extraneous data, membership questions

- Minimize lineage traversal
- Memoization
- Batch computations

Future Directions

Theory, Model, Algorithms

- Unlimited opportunities

System

- Storage, indexing, partitioning
- Statistics and query optimization

More features

- Top-*K* by confidence
- Incomplete relations; continuous uncertainty; correlated uncertainty
- External lineage; update lineage; versioning

but don't forget
the lineage…

Search "stanford trio"

[overview paper]

Trio group:

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

Special thanks to:

Ashok Chandra, Alon Halevy, Jeff Ullman