Trio: A System for Data, Uncertainty, and Lineage

Jennifer Widom

Stanford University

Stanford CS Faculty Lunch Series
Spring 2006

Two faculty independently proclaim uncertainty as the next major theme in Computer Science

- One old-timer, one youngster
- Proclamations not motivated by their own (or our) research

Uncertainty in Databases

- Not a new idea — proposed 20+ years ago
- Most initial (18+ years) work largely theoretical; not much systems-building until recently
- But applications weren't ready anyway
  - ➔ Are they now?

Depiction of Trio Project
Stanford News — Spring 2006

Stanford Report

CLOSE

Photo illustration by Vince Tarry

The Trio database system, developed by Professor Jennifer Widom and her research team, can account for the uncertainty of data and its sourcing.

The “Trio” in Trio

1. Data

   Student #123 is majoring in Econ: (123,Econ) ∈ Major

2. Uncertainty

   Student #123 is majoring in Econ or CS:

      (123, *Econ* || *CS*) ∈ Major

   With confidence 60% student #456 is a CS major:

      (456, *CS* 0.6) ∈ Major

3. Lineage

   456 ∈ HardWorker derived from:

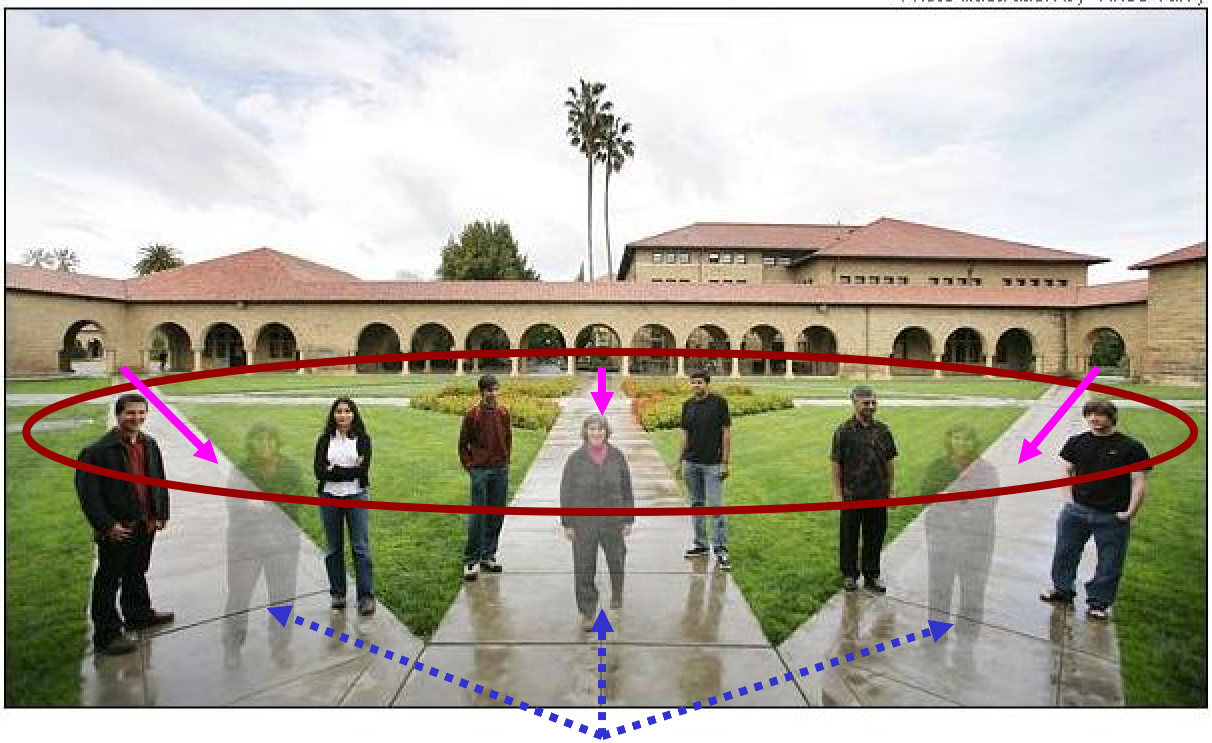
      (456, CS) ∈ Major
      “CS is hard” ∈ some web page

Depiction

Stanford Report

CLOSE

Photo illustration by Vince Tarry



The Trio database system, developed by Professor Jennifer Widom and her research team, can account for the uncertainty of data and its sourcing.

Data

Uncertainty

Lineage
("sourcing")

Original Motivation for the Project

New Application Domains

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

Coincidence or Fate?

Original Motivation for the Project

New Application Domains

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

Neither uncertainty nor lineage is supported in current database systems

Sample Applications

Information extraction

- Find & label entities in unstructured text
- Often probabilistic

Information integration

- Combine data from multiple sources
- Inconsistencies

Scientific experiments

- Inexact/incomplete data
- Many levels of "derived data products"

Sample Applications

Sensor data management

- Approximate readings
- Missing readings
- Levels of data aggregation

Deduplication ("data cleaning")

- Object linkage, entity resolution
- Often heuristic/probabilistic

Approximate query processing

- Fast but inexact answers

Our Claim

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

Coincidence or Fate?

Substantiation of Claim

[technical] Lineage:

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

[fluffy] Applications use lineage to reduce or resolve uncertainty

Our Goal

Develop a new kind of database management system (DBMS) in which:

1. Data
2. Uncertainty
3. Lineage

are all first-class interrelated concepts

➔ With all the "usual" DBMS features

Pop Quiz

Why should every slide be making you squirm in your seat?

Our Goal

Develop a new kind of database management system (DBMS) in which:

1. Data
2. Uncertainty
3. Lineage

are all first-class interrelated concepts

→ With all the "usual" DBMS features

The "Usual" DBMS Features

(From first lecture of my *Intro. to Databases* class)

1. Efficient,
2. Convenient,
3. Safe,
4. Multi-User storage of and access to
5. Massive amounts of
6. Persistent data

Standard Relational DBMSs

Persistent; Convenient

- All data stored in simple tables (“relations”)
- Queries and updates via simple but powerful *declarative* language (SQL)

Multi-User; Safe

- Transactions

Massive; Efficient

- Storage and indexing structures
- Query optimization

Trio: What Changes

Persistent; Convenient

- All **data stored** in simple tables (“relations”)
- Queries and updates via simple but powerful *declarative* **language** (SQL)

Multi-User; Safe

- Transactions

Massive; Efficient

- Storage and indexing structures
- Query optimization

Trio: What Changes

Persistent; Convenient

- All **data stored** in simple tables (“relations”)
- Queries and updates via simple but powerful *declarative* **language** (SQL)

Multi-User; Safe **- standard DBMS underneath**

- Transactions

Massive; Efficient **- standard DBMS underneath**

- Storage and indexing structures
- Query optimization

Another "Trio" in 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

Another “Trio” in Trio

1. Data Model

   Uncertainty-Lineage Databases (*ULDB*s)

2. Query Language

   TriQL

3. System

   Trio-One — built on top of standard DBMS

Remainder of Talk

1. Data Model

   Uncertainty-Lineage Databases (*ULDBs*)

2. Query Language

   TriQL

3. System

   Trio-One — built on top of standard DBMS

4. *Demo*

First a Disclaimer

We are not about machine learning or probabilistic reasoning!

We are about efficient and convenient storage, manipulation, and retrieval of large data sets (with uncertainty and lineage in them)

Running Example: Crime-Solving

Saw (witness, color, car) // may be uncertain

Drives (person, color, car) // may be uncertain

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

In Standard Relational DBMS

Saw (witness, color, car)		
Amy	red	Mazda
Betty	blue	Honda
Carol	green	Toyota

Drives (person, color, car)		
Jimmy	red	Toyota
Billy	blue	Honda
Frank	red	Mazda
Frank	green	Mazda

```
Create Table Suspects as
Select person
From Saw, Drives
Where Saw.color = Drives.color
And Saw.car = Drives.car
```

Suspects
Frank
Billy

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, *color*, *car*)	
Amy	*red, Honda* ‖ *red, Toyota* ‖ *orange, Mazda*

Three possible instances

Our Model for Uncertainty

1. Alternatives
2. **'?' (Maybe):** uncertainty about presence
3. Confidences

Saw (witness, *color, car*)		
Amy	***red, Honda* ‖ *red, Toyota* ‖ *orange, Mazda***	
Betty	***blue, Acura***	**?**

Six possible instances

Our Model for Uncertainty

1. Alternatives

2. **'?' (Maybe):** uncertainty about presence

3. Confidences

absent ≠ unknown

Saw (witness, *color*, *car*)		
Amy	*red, Honda ‖ red, Toyota ‖ orange, Mazda*	
Betty	*blue, Acura*	**?**

Betty	*blue, Acura ‖ NULL, NULL*

Our Model for Uncertainty

1. Alternatives

2. ‘?’ (Maybe) Annotations

3. **Confidences:** weighted uncertainty

Saw (witness, *color*, *car*)		
Amy	*red, Honda* 0.5 ‖ *red, Toyota* 0.3 ‖ *orange, Mazda* 0.2	
Betty	*blue, 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

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* ‖ *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 data elements to other data elements (or external sources)

Example with Lineage

ID	Saw (witness, *car*)	
11	Cathy	*Honda* ‖ *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*
32	*Billy* ‖ *Frank*
33	*Hank*

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

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

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

Correctly captures possible instances in the result

Example with Lineage

ID	Saw (witness, *car*)	
11	Cathy	*Honda* ‖ *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*	?
32	*Billy* ‖ *Frank*	?
33	*Hank*	?

Trio Data Model

Uncertainty-Lineage Databases (ULDBs)

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

General case after several queries:

- Boolean formula

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)

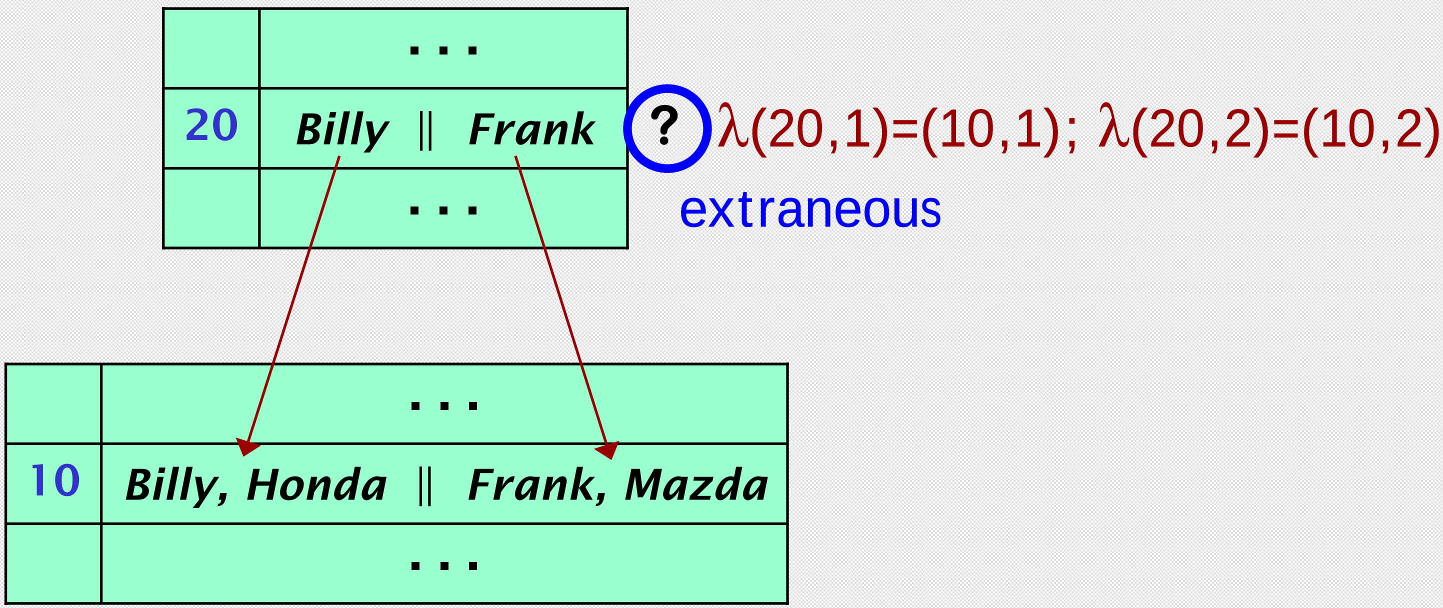
  Data-minimality

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

- Also Lineage-minimality

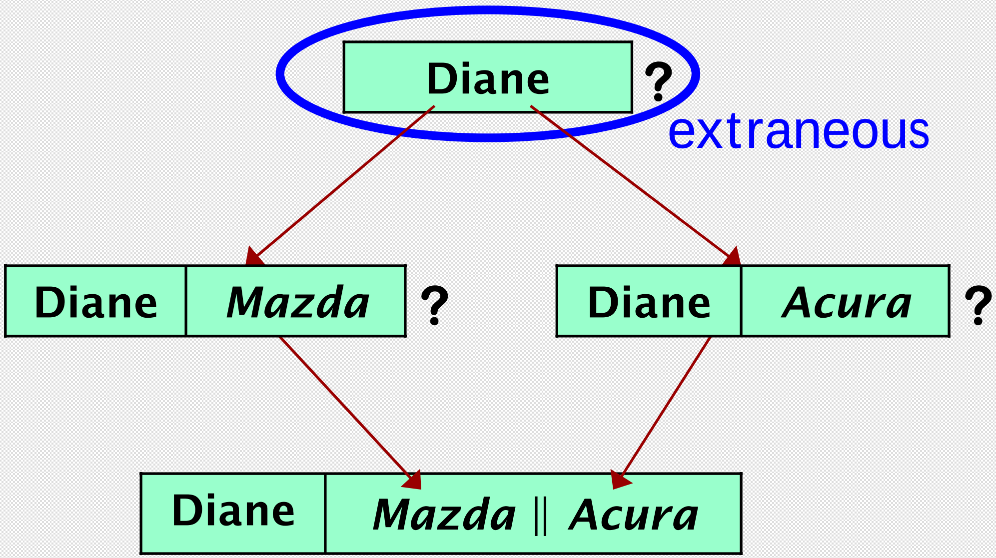
Data Minimality Examples

Extraneous ‘?’



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

Querying ULDBs TriQL

Simple extension to SQL

Formal semantics, intuitive meaning

Query uncertainty, confidences, and lineage

Simple TriQL Example

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

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

Create Table Suspects as
Select person
From Saw, Drives
Where Saw.car = Drives.car

ID	Suspects
31	*Jimmy*
32	*Billy* ‖ *Frank*
33	*Hank*

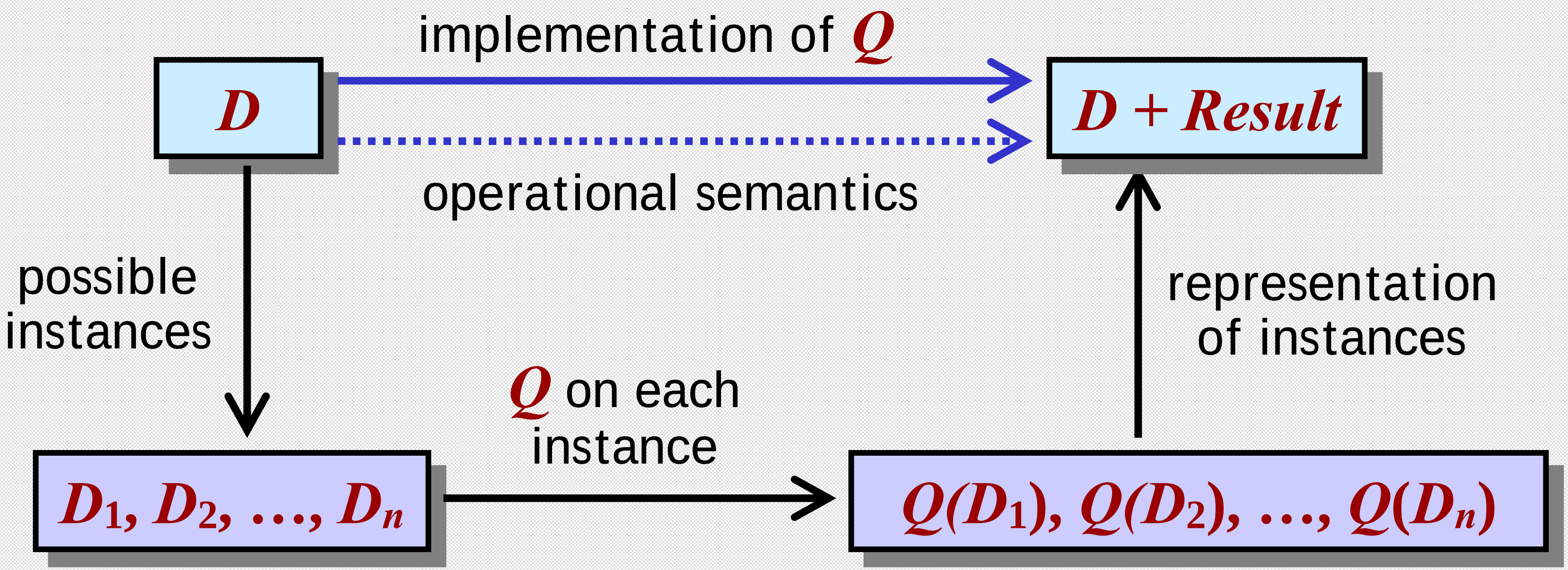
? $\lambda(31)=(11,2),(21,2)$

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

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

Formal Semantics

Relational (SQL) query *Q* on ULDB *D*



TriQL: Querying Confidences

Built-in function: Conf()

```
SELECT person
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 Suspects.person
FROM Suspects, Saw
WHERE Lineage(Suspects,Saw)
AND  Saw.witness = 'Amy'
```

X ==> Y shorthand for Lineage(X,Y)

Operational Semantics

SELECT attr-list
FROM $X_1, X_2, \ldots, X_n$
WHERE predicate

Over standard 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

Operational Semantics

```
SELECT attr-list
FROM X1, X2, …, Xn
WHERE predicate
```

Over ULDB:

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

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

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

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

Operational Semantics: Example

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

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

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

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

Operational Semantics: Example

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

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

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

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

Operational Semantics: Example

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

Saw (witness, *car*)	
Cathy	***Honda* || *Mazda***

Drives (*person*, car)	
Jim* || *Bill	**Mazda**
Hank	**Honda**

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

Operational Semantics: Example

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

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

Drives (*person*, car)	
Jim* ‖ *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 person
FROM Saw, Drives
WHERE Saw.car = Drives.car
```

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

Drives (*person*, car)	
Jim ‖ *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 person
FROM Saw, Drives
WHERE Saw.car = Drives.car

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

Drives (*person*, car)	
Jim* ‖ *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

CREATE TABLE Suspects AS
SELECT Drives.person
FROM Saw, Drives
WHERE Saw.car = Drives.car

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

Drives (*person*, car)	
Jim ‖ *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 ‖ *Mazda* 0.4

Drives (*person*, car)	
Jim 0.3 ‖ *Bill* 0.6	Mazda
Hank 1.0	Honda

Min

Suspects
Jim 0.3 ‖ *Bill* 0.4
Hank 0.6

Additional TriQL Constructs

- “Horizontal subqueries”
  - Refer to tuple alternatives as a relation
- Aggregations: *low*, *high*, *expected*
- *Unmerged* (horizontal duplicates)
- *Flatten*, *GroupAlts*
- *NoLineage*, *NoConf*, *NoMaybe*
- Query-computed confidences
- Data modification statements

Trio-Specific Additional Features

- Lineage tracing
- On-demand confidence computation
- Coexistence checks
- Extraneous data removal

➔ Interrelated algorithms

One More Example Query

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

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

One More Example Query

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

Credibility (person,score)	
Amy	10
Betty	15
Cathy	5

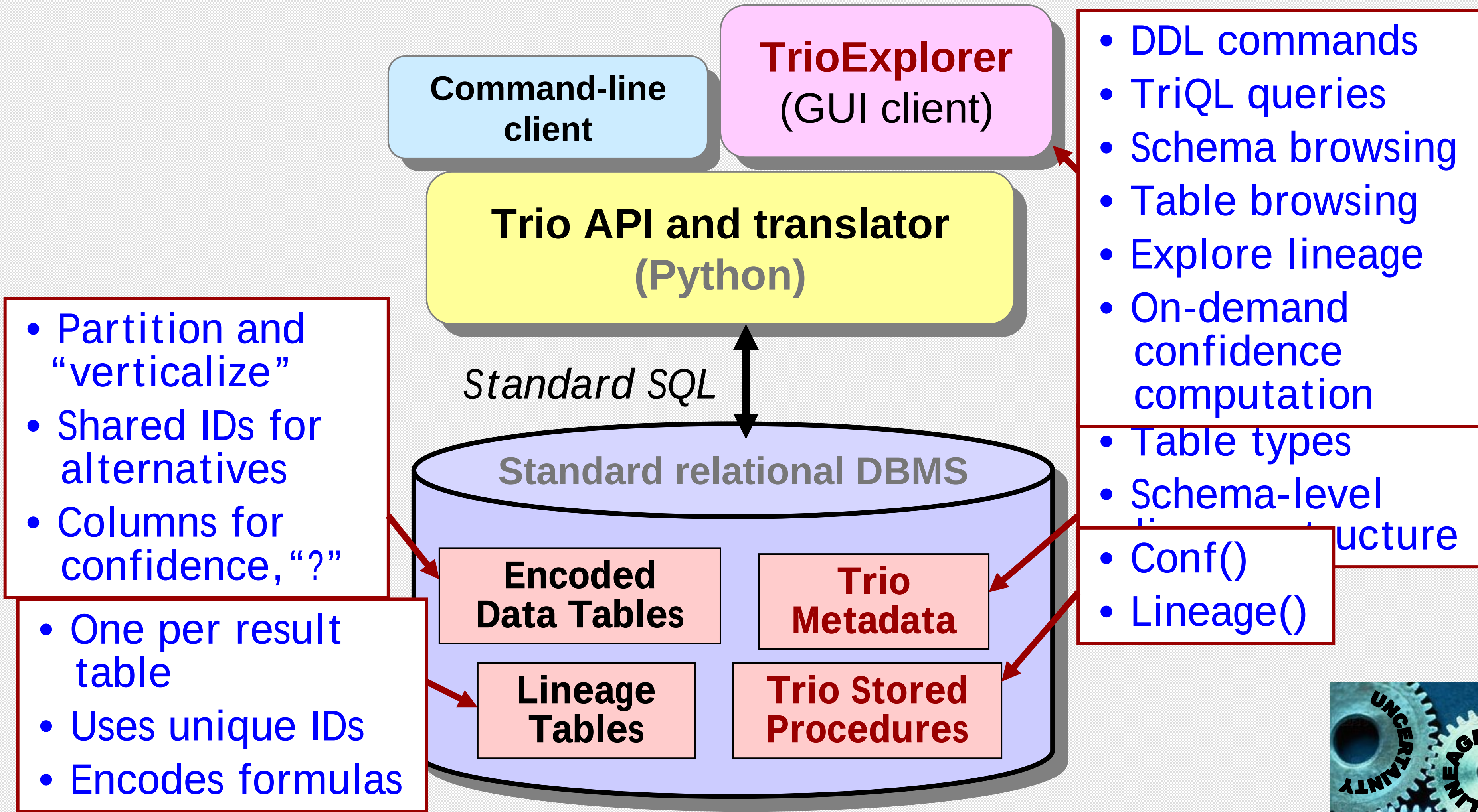
```
SELECT suspect, score/[sum(score)] as conf
FROM (SELECT suspect,
          (SELECT score FROM Credibility C
           WHERE C.person = P.accuser)
        FROM PrimeSuspect P)
```

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

The Trio System

- Version 1 (“Trio-One”)
  - On top of standard DBMS
  - Surprisingly easy and complete, reasonably efficient

Trio-One Overview



Example: Data Encoding

Saw (witness, *color*, *car*)	
Amy	*red, Honda* 0.3 ‖ *red, Toyota* 0.4 ‖ *orange, Mazda* 0.3
Betty	*blue, Acura* 0.8

View: Saw = Saw-C ⋈ Saw-U

Saw-C

xid	witness
1	Amy
2	Betty

Saw-U

xid	aid	conf	color	car
1	11	0.3	red	Honda
1	12	0.4	red	Toyota
1	13	0.3	orange	Mazda
2	14	0.8	blue	Acura

Example: Query Translation

Query *Q* into result table *R*

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

   Q' ≈ *Q* but adds ID's of source tuples, joins lineage tables when lineage() predicates, other tricks

2. Group *R-U* into alternatives, generate xid's
3. Move certain attrs. to *R-C*, lineage data to *R-Lin*
4. Compute confidences? (next slide)
5. Add metadata: view defn. for *R*, schemas, confidence info., lineage structure

6. Transient results: stop at 2, return cursor

Issue: Confidence Computation

Previous approach (probabilistic databases)

- Each operator computes confidences during query execution
- *Restricts allowable query execution strategies*

In Trio

Confidence of data element *d* is function of confidences in $\lambda^*(d)$

Confidence Computation (cont'd)

Our approach

- Use any query execution strategy
- Compute confidences on-demand based on lineage
- Some optimizations
  - "Independent lineage subtrees"
  - Memoization
  - Batch computation

Current Topics (sample)

More forms of uncertainty

- Continuous uncertainty (intervals, Gaussians)
- Correlated uncertainty
- Incomplete relations

More forms of lineage

- External lineage
- Update lineage

Confidence-based queries

- Threshold; “Top-K”

Current Topics (sample)

Design theory

- Dependencies, normal forms, decomposition
- New definitions, twists, and challenges

System

- Full query language
- Performance experiments
- Demo applications
- Version 2: *Go native*?
  - Storage and indexing structures
  - Statistics
  - Query optimization

Search "stanford trio"

Trio contributors, past and present

Parag Agrawal, Omar Benjelloun, Ashok Chandra, Anish Das Sarma, Alon Halevy, Chris Hayworth, Ander de Keijzer, Raghotham Murthy, Michi Mutsuzaki, Shubha Nabar, Tomoe Sugihara, Martin Theobald