Trio: A System for Data, Uncertainty, and Lineage

Search “stanford trio”

Stanford Report: March ‘06

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.

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

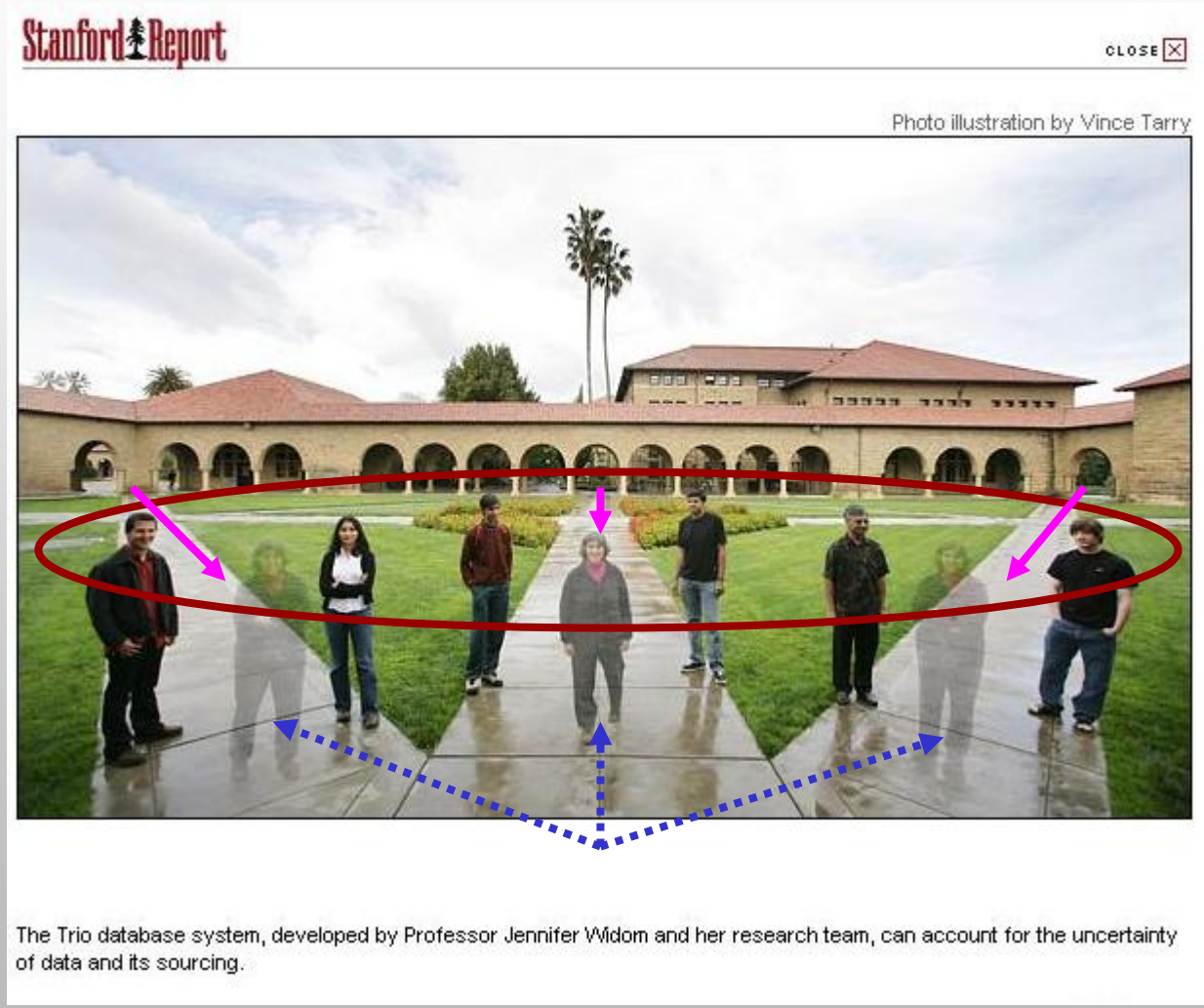
3. Lineage

   (456) ∈ HardWorker derived from:

   (456, CS) ∈ Major

   "CS is hard" ∈ some web page

The Picture



Data

Uncertainty

Lineage
("sourcing")

Why Uncertainty + Lineage?

Many applications seem to need both

- Information extraction systems
- Scientific and sensor data management
- Information integration
- Deduplication (data cleaning)
- Approximate query processing

Why Uncertainty + Lineage?

From a technical standpoint, it turns out that 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

Goal

A new kind of DBMS in which:

1. Data
2. Uncertainty
3. Lineage

} **Trio**

are all first-class interrelated concepts

With all the usual DBMS features

Scalable, reliable, efficient, ad-hoc declarative queries and updates, …

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 Trio Trio

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

Deficiency in Model

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

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

Suspects = π_{person}(Saw ⋈ 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

Uncertainty-Lineage Databases (ULDBs)

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

The ULDB model is "complete"

Querying ULDBs

TriQL

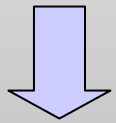
- Simple extension to SQL
- Formal semantics, intuitive meaning
- Query uncertainty, confidences, and lineage

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

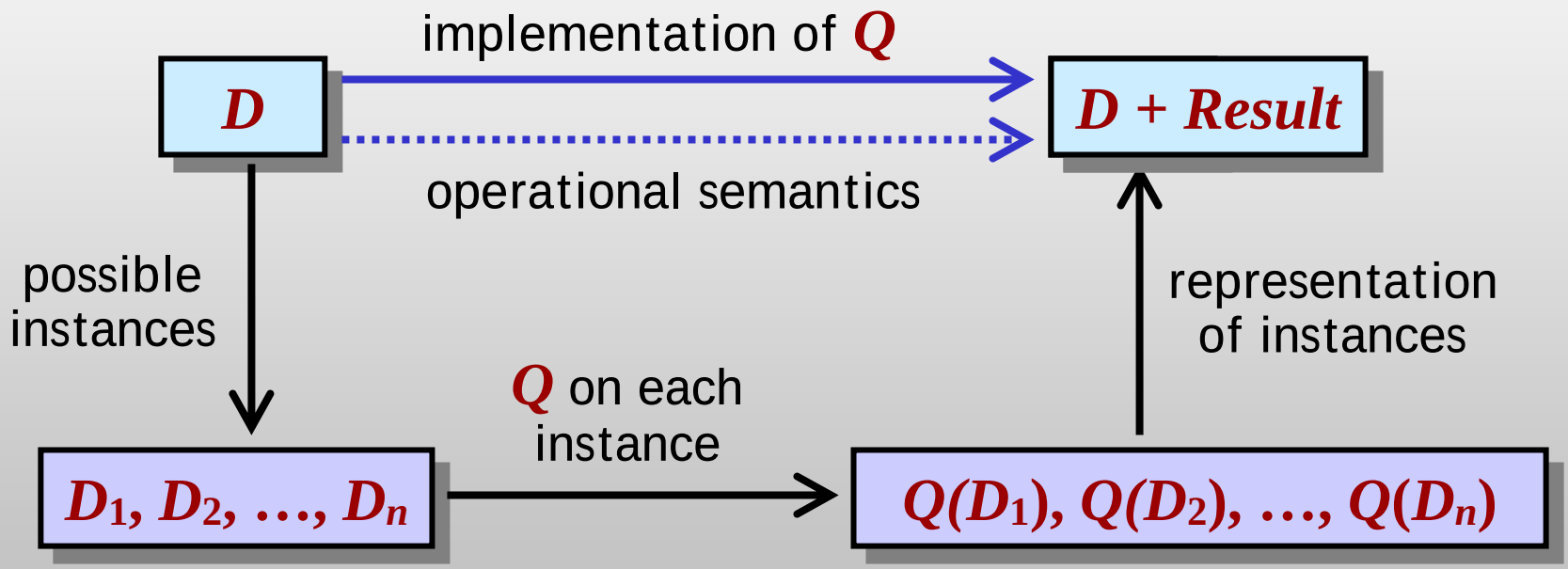
? $\lambda(31) = (11,2),(21,2)$

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

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

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'
```

Operational Semantics

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

Over conventional relational database:

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

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

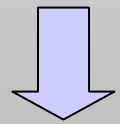
Final Example Query

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

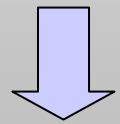
Final Example Query

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

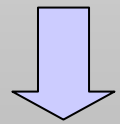
Final Example Query

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

Credibility

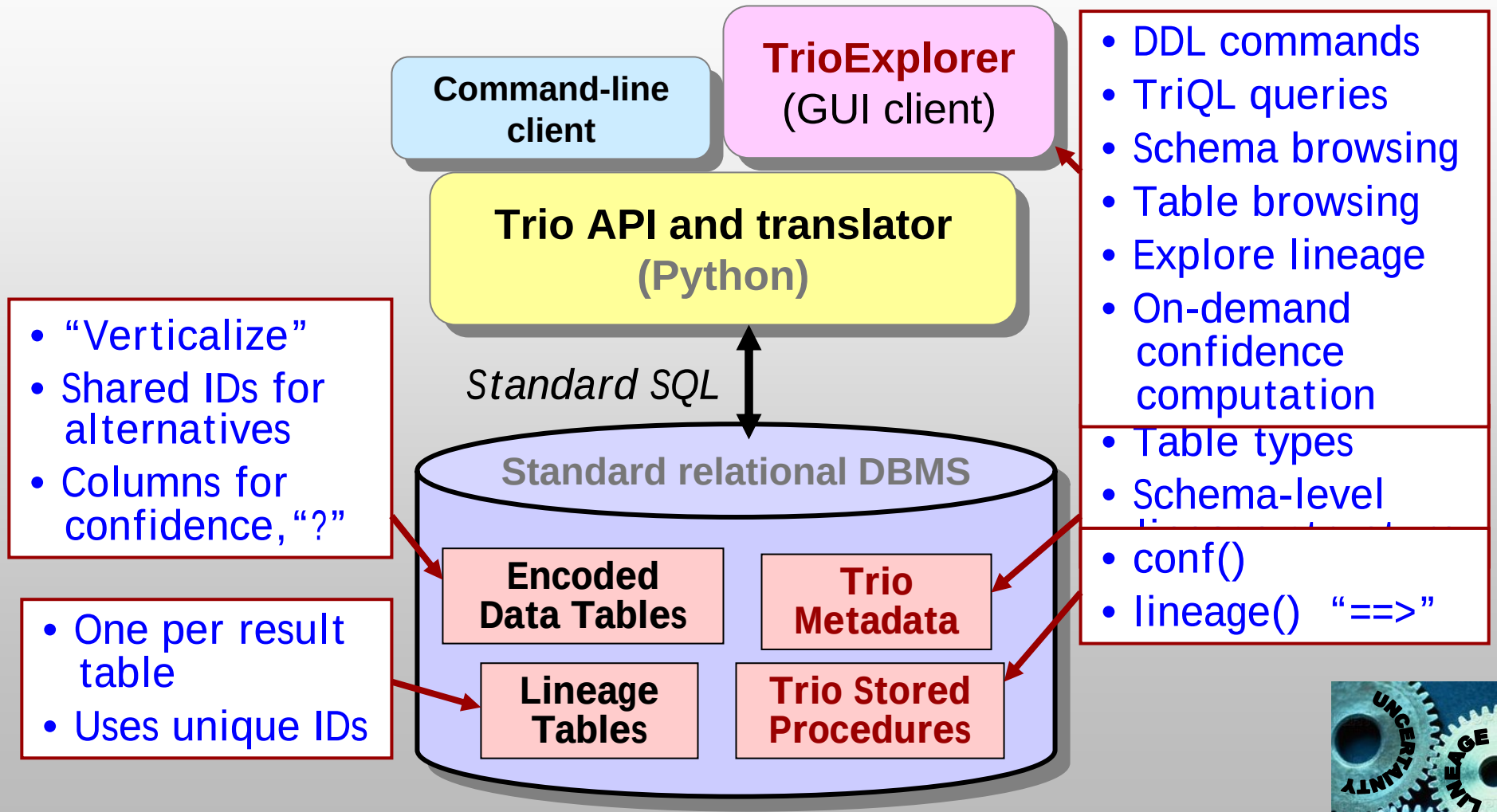
person	score
Amy	10
Betty	15
Cathy	5

```
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

Trio System: Version 1



Future Features (sample)

Uncertainty

- Incomplete relations
- Continuous uncertainty
- Correlated uncertainty

Lineage

- External lineage
- Update lineage

Query processing

- “Top-K” by confidence

but don’t forget the lineage…