Principled Research in Database Systems

Jennifer Widom
Stanford University
What Academics Give Talks About

- Grad Student: Other people’s papers
- Assistant Professor: Thesis and new results
- Associate Professor: Significant research projects
- Full Professor: The research field
- Other people’s papers
Recipe for Database Research Project

• Pick a simple but fundamental assumption underlying traditional database systems
  ❖ Drop it

• Must reconsider all aspects of data management and query processing
  ❖ Many Ph.D. theses
  ❖ Prototype from scratch
Recipe for Database Research Project

• Pick a simple but fundamental assumption underlying traditional database systems
  ❖ Drop it

• Example “simple but fundamental assumptions”
  – Structure of data (schema) declared in advance
    Drop: Semistructured data
  – Persistent (typically stable, disk-resident) data sets
    Drop: Data streams
  – Data elements contain known values
    Drop: Uncertain data
Recipe for Database Research Project

• Must reconsider all aspects of data management and query processing

• Reconsidering “all aspects”
  – Data model
  – Query language
  – Storage and indexing structures
  – Query processing and optimization
  – Concurrency control, recovery
  – Application and user interfaces
Principled Database Systems Research

To develop a new type of database system:

- Consider all of them
- In this order

⇒ Solid foundations first, then implementation
As research evolves, always revisit all three

Solid foundations first, then implementation
Disclaimer

This principled approach works for me
Other’s mileage may vary
Challenges of developing new data models and query languages that are ...

1) Well-defined
2) Understandable
3) Sufficiently expressive (and not more)
4) Similar to existing models & languages
5) Implementable
Challenges of developing new (1) data models and (2) query languages that are ...

1) Well-defined
2) Understandable
3) Sufficiently expressive (and not more)
4) Similar to existing models & languages
5) Implementable
Creating a New Data Model

Examples from my own experience

- Semistructured data  Hard
- Data streams  Harder
- Uncertain data  Hardest
Creating a New Data Model

Examples from my own experience

- Semistructured data
- Data streams
- Uncertain data
Data Model Challenge #2
Data Streams

• Relational model

• Except data arrives as continuous, unbounded streams

➢ Data may (or may not) be timestamped; order may (or may not) be relevant
Model for Data Streams

“A data stream is an unbounded sequence of \[\text{tuple timestamp}\] pairs”

Temperature Sensor 1:
\[
(72, 2:05) \quad (75, 2:20) \quad (74, 2:21) \quad (74, 2:24) \quad (81, 2:45) \ldots
\]

Temperature Sensor 2:
\[
(73, 2:03) \quad (76, 2:20) \quad (73, 2:22) \quad (75, 2:22) \quad (79, 2:40) \ldots
\]
“A data stream is an unbounded sequence of \textit{tuple timestamp} pairs”

Temperature Sensor 1:

Temperature Sensor 2:

★ Duplicate timestamps in streams?
★ If yes, is order relevant?
“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:
- [(72) 2:05]
- [(75) 2:20]
- [(74) 2:21]
- [(74) 2:24]
- [(81) 2:45] ... 

Temperature Sensor 2:
- [(73) 2:03]
- [(76) 2:20]
- [(73) 2:22]
- [(75) 2:22]
- [(79) 2:40] ... 

★ Are timestamps coordinated across streams?

Duplicates? Order relevant?
“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:

Temperature Sensor 2:

🌟 What about absolute duplicates?
A data stream is an unbounded sequence of \( [\text{tuple timestamp}] \) pairs

Temperature Sensor 1:
\[
[(72) \ 2:05] \quad [(75) \ 2:20] \quad [(74) \ 2:21] \quad [(74) \ 2:24] \quad [(81) \ 2:45] \quad \ldots
\]

Temperature Sensor 2:
\[
[(73) \ 2:03] \quad [(76) \ 2:20] \quad [(73) \ 2:22] \quad [(75) \ 2:22] \quad [(79) \ 2:40] \quad \ldots
\]

Sample Query (continuous)

“Average discrepancy between sensors”
Result depends heavily on details of model
Data Model Challenge #3
Uncertain Data

• Tuples may have alternative possible values
• Tuple presence may be uncertain
• Uncertainty may have associated confidence (probability) values
Uncertain Data: Semantics

- Tuples may have alternative possible values
- Tuple presence may be uncertain
- Uncertainty may have associated confidence (probability) values
- An uncertain database represents a set of possible certain databases (possible-instances, possible-worlds)
- With (optionally) a probability for each one
Uncertain Data: Simple Example

Jennifer attends a conference on Monday
Hector attends on Monday, Tuesday, or not at all

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
<tr>
<td>Hector</td>
<td>Monday</td>
</tr>
</tbody>
</table>

Instance 1

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
<tr>
<td>Hector</td>
<td>Tuesday</td>
</tr>
</tbody>
</table>

Instance 2

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
</tbody>
</table>

Instance 3

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
</tbody>
</table>

Representation:

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
<tr>
<td>Hector</td>
<td>Monday</td>
</tr>
</tbody>
</table>

?
Models for Uncertain Data

- A representation scheme (hereafter model) for uncertain data is well-defined if we know how to map any database in the model to its set of possible-instances.
- A model is complete if every set of possible-instances can be represented.
  - Alternative values + ?’s (+ confidences) model is well-defined but not complete.
Incompleteness Example

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
<tr>
<td>Hector</td>
<td>Tuesday</td>
</tr>
</tbody>
</table>

*Instance 1*

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
</tbody>
</table>

*Instance 2*

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hector</td>
<td>Tuesday</td>
</tr>
</tbody>
</table>

*Instance 3*

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
</table>

Generates 4th instance: empty relation
Completeness vs. Closure

An incomplete model may still be interesting if it’s expressive enough and closed under all useful operations.
More on Operations and Closure

Relational operation $Op$ on uncertain database $D$

- $D_1, D_2, \ldots, D_n$
- $Op(D_1), Op(D_2), \ldots, Op(D_n)$

Closure $D'$ always exists
Non-Closure Example

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday</td>
</tr>
<tr>
<td>Hector</td>
<td>Monday</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>day</th>
<th>weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>rain</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>person</th>
<th>weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>rain</td>
</tr>
<tr>
<td>Hector</td>
<td>rain</td>
</tr>
</tbody>
</table>

- Various solutions
- Most obvious & general: add *constraints* of some type

Four possible instances Should be two
Original Goal

Develop new data models [and query languages] that are ...

1) Well-defined
2) Understandable
3) Sufficiently expressive (and not more)
4) Similar to existing models & languages
5) Implementable
Original Goal

Develop new data models [and query languages] that are ...

1) Well-defined
2) Understandable
3) Sufficiently expressive (and not more)
4) Similar to existing models & languages
5) Implementable
Exploring Space of Models

- Only complete model
- Closure properties
- Relative expressiveness
- Only understandable models

What a mess!

Possible models
Fairy-Tale Ending

- Added lineage to simple uncertainty model

  We wanted lineage in our model & system anyway

- Representation became complete
Principled DB Systems Research

Data Model → Query Language → System
Query Language Design

- Notoriously difficult to publish
- But potential for huge long-term impact
- Semantics can be surprisingly tricky
Query Language Design

Warm-up

- SQL

Examples from my own experience

- Active databases
- Semistructured data
- Data streams
- Uncertain data
Query Language Design

Warm-up

• SQL

Examples from my own experience

• Active databases
• Semistructured data
• Data streams
• Uncertain data
SQL: A Dubious Beginning

Good: SQL is a declarative language
   - Semantics independent of execution model

Bad: Not everyone thinks of it that way
   - For some, only understanding is execution model
SQL: A Dubious Beginning

Good: SQL is a declarative language
  ❖ Semantics independent of execution model

Bad: Not everyone thinks of it that way
  ❖ For some, only understanding is execution model

Example: Add 5 to smallest values in table T

```
UPDATE T SET A = A+5
WHERE A <= ALL (SELECT A FROM T)
```
SQL: A Dubious Beginning

Good: SQL is a declarative language
- Semantics independent of execution model

Bad: Not everyone thinks of it that way
- For some, only understanding is execution model

Example: Group-By (non)restrictions

```sql
SELECT day, count(*)
FROM Attends
GROUP BY day
```
Good: SQL is a declarative language
- Semantics independent of execution model

Bad: Not everyone thinks of it that way
- For some, only understanding is execution model

Example: Group-By (non)restrictions

```
SELECT day, count(*), person
FROM Attends
GROUP BY day
```
SQL: A Dubious Beginning

Good: SQL is a declarative language
  - Semantics independent of execution model

Bad: Not everyone thinks of it that way
  - For some, *only* understanding is execution model
Install “active rules” (triggers) in the database

WHEN action occurs
IF condition holds
THEN perform additional actions

Insert/Delete/Update SQL
Tricky Semantics Example

Trigger 1: WHEN X makes sale > 500
THEN increase X’s salary by 1000

Trigger 2: WHEN average salary increases > 10%
THEN increase everyone’s salary by 500

Inserts: Sale(Mary,600) Sale(Mary,800) Sale(Mary,550)

• How many increases for Mary?
• If each causes average > 10%, how many global raises?
• What if global raise causes average > 10%?
The Lure of the System

Transition tables, Conflicts, Confluence, ...

“We finished our rule system ages ago”

“Write Code!”
Happy Ending

- Semantics got defined
  At least on our side of the bay
- System got built
- Even SQL triggers standard ended up reasonable
  Primarily through restrictions
Query Language Challenge #3
Data Streams

Temperature Sensor:

\[
\begin{array}{c}
(72) \ 2:00 \\
(74) \ 2:00 \\
(76) \ 2:00 \\
(60) \ 8:00 \\
(58) \ 8:00 \\
(56) \ 8:00
\end{array}
\]

Query (continuous):
Average of most recent three readings
Query Language Challenge #3
Data Streams

Temperature Sensor:

[(72) 2:00] [(74) 2:00] [(76) 2:00] [(60) 8:00] [(58) 8:00] [(56) 8:00]

Query (continuous):
Average of most recent three readings
System A: 74, 58
Query Language Challenge #3
Data Streams

Temperature Sensor:

[(72) 2:00]  [(74) 2:00]  [(76) 2:00]  [(60) 8:00]  [(58) 8:00]  [(56) 8:00]

Query (continuous):
Average of most recent three readings

System A: 74, 58
System B: 74, 70, 64.7, 58
Original Goal

Develop new [data models and] query languages that are ...

1) Well-defined
2) Understandable
3) Sufficiently expressive (and not more)
4) Similar to existing models & languages
5) Implementable
Original Goal

Develop new [data models and] query languages that are ...

1) Well-defined
2) Understandable
3) Sufficiently expressive (and)
4) Similar to existing models
5) Implementable

The “It’s Just SQL” Trap

No happy ending (yet)
Query Language Challenge #4
Uncertain Data

Tables: \textbf{Attends}(\text{person, day}) \textbf{Weather}(\text{day, temp})

Query: Hector chill factor

\begin{verbatim}
SELECT W.temp
FROM Attends A, Weather W
WHERE A.day = W.day AND A.person='Hector'
\end{verbatim}

\begin{tabular}{|c|c|}
\hline
\textbf{person} & \textbf{day} \\
\hline
Jennifer & \textit{Monday 0.8} || \textit{Tuesday 0.2} \\
 Hector & \textit{Monday 0.3} || \textit{Tuesday 0.7} \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline
\textbf{day} & \textbf{temp} \\
\hline
Monday & [60,65] \\
Tuesday & [62,70] \\
\hline
\end{tabular}
Query Language Challenge #4
Uncertain Data

- Two weighted temperature ranges?
- Single expected value across (weighted) days/ranges?
- What if Hector tuple has a ‘?’?

```
SELECT W.temp
FROM Attends A, Weather W
WHERE A.day = W.day AND A.person = 'Hector'
```

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jennifer</td>
<td>Monday 0.8</td>
<td></td>
</tr>
<tr>
<td>Hector</td>
<td>Monday 0.3</td>
<td></td>
</tr>
</tbody>
</table>

SQL vs. Weighted values

```
Monday (0.8, 0.3) || Tuesday (0.2, 0.7)
```

"Just SQL"
Original Goal

Defining SQL over a new data model can be tricky and complex (and fun)

- Semistructured data
- Data streams
- Uncertain data
- <Insert future model here>

Flip side: The relational model / SQL can really help
Exploit Relational Whenever Possible

Uncertain data – semantics of query $Q$

Possible instances

$D_1, D_2, \ldots, D_n$

Q on each instance

$Q(D_1), Q(D_2), \ldots, Q(D_n)$

30 years of refinement

Result

(implementation)

representation of instances
Exploit Relational Whenever Possible

Semantics of stream queries

Streams

Window

Relations

Implementation

Istream / Dstream

30 years of refinement
Principled DB Systems Research

Data Model → Query Language → System

Impact
Another “simple but fundamental assumption”

- Data is produced & stored by computers

Drop: Crowdsourced data
## Crowdsourced Data

<table>
<thead>
<tr>
<th>city</th>
<th>restaurant</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palo Alto</td>
<td>Evvia</td>
<td>4</td>
</tr>
<tr>
<td>Palo Alto</td>
<td>Tamarine</td>
<td>3</td>
</tr>
<tr>
<td>Menlo Park</td>
<td>Left Bank</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

...
### Crowdsourced Data

<table>
<thead>
<tr>
<th>city</th>
<th>restaurant</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palo Alto</td>
<td>Evvia</td>
<td>4</td>
</tr>
<tr>
<td>Palo Alto</td>
<td>Tamarine</td>
<td>3</td>
</tr>
<tr>
<td>Menlo Park</td>
<td>Left Bank</td>
<td>5 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
## Crowdsourced Data

<table>
<thead>
<tr>
<th>city</th>
<th>restaurant</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palo Alto</td>
<td>Evvia</td>
<td>4</td>
</tr>
<tr>
<td>Palo Alto</td>
<td>Tamarine</td>
<td>3</td>
</tr>
<tr>
<td>Menlo Park</td>
<td>Left Bank</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Crowdsourced Data

<table>
<thead>
<tr>
<th>city</th>
<th>restaurant</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palo Alto</td>
<td>Evvia</td>
<td>4</td>
</tr>
<tr>
<td>Palo Alto</td>
<td>Tamarine</td>
<td>3</td>
</tr>
<tr>
<td>Menlo Park</td>
<td>Left Bank</td>
<td>5</td>
</tr>
<tr>
<td>Berkeley</td>
<td>Chez Panisse</td>
<td>1</td>
</tr>
<tr>
<td>Berkeley</td>
<td>Emilia’s</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Jennifer Widom
Principled Database Systems Research

To develop a new type of database system:

- Consider all of them
- In this order

⇒ Solid foundations first, then implementation
• Start with relational

• Key differences:
  – Missing values
  – Uncertainty
    ▪ Alternatives, confidence values
    ▪ Hide or expose?
  – Missing tuples / unbounded # of tuples
  – “Rules” for crowdsourced values

Semantics: Map to relational

(city, restaurant) → rating
(city, rating) → restaurant
∅ → (city, restaurant)
Query Language

Ex: Highly rated restaurants in Palo Alto

```
SELECT restaurant
FROM RestRatings
WHERE city='Palo Alto'
AND rating >= 4
```

Multiple/uncertain ratings

rating → restaurant
or restaurant → rating?

How many?

The “It’s Just SQL” Trap

Semantics:
Map to SQL (to extent possible)
System

Check out Berkeley and MIT 😊
Back to Data Model and Query Language

“Rules” for crowdsourced values

Alarmingly close relationship between semantics and query processing

SELECT restaurant
FROM RestRatings
WHERE city='Palo Alto'
AND rating >= 4

Multiple/uncertain ratings

Alarmingly close relationship between semantics and query processing

Data and Query Semantics

Query Execution
Principled Database Systems Research

Data Model

Query Language

System

Applications

SLOW