Principled Research in Database Systems

Jennifer Widom
Stanford University
What Academics Give Talks About

- Other people's papers
- Thesis and new results
- Significant research projects
- The research field
- BIG VISION
- Other people's papers

Grad Student
Assistant Professor
Associate Professor
Full Professor
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

Jennifer Widom
Bulk of Talk

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
- Data streams
- Uncertain data

Jennifer Widom
Data Model Challenge #1
Semistructured Data

- No schema fixed in advance
- Data is *self-describing*
- Irregularity is OK

Motivated by Data Integration, Web
The “Object Exchange Model” (OEM)
≈ XML

Element tags

- Member
  - Name
    - "Clark"
  - Age
    - 46
  - Office
    - Building
      - "CIS"
  - Room
    - "411"

- Member
  - Name
    - "Smith"
  - Age
    - 46
  - Office
    - Building
      - "CIS"
  - Room
    - "411"

- Member
  - Name
    - "Jones"
  - Age
    - 28
  - Office
    - Building
      - "CIS"
  - Room
    - "411"

- Member
  - Name
    - "Smith"
  - Age
    - 46
  - Office
    - Building
      - "CIS"
  - Room
    - "411"

Root element

- DBGroup
  - Member
    - Name
      - "Jones"
    - Age
      - 28
    - Office
      - Building
        - "CIS"
      - Room
        - "411"
  - Project
    - Building
      - "Gates"
    - Room
      - "411"

Subelements

- Project
  - Title
    - "Lore"
  - Building
    - "CIS"
  - Room
    - "411"

- Project
  - Title
    - "Tsimmis"
  - Building
    - "Gates"
  - Room
    - "411"

Attributes

- "Clark"
- "Smith"
- "Jones"
- "Lore"
- "Tsimmis"
Data Model Challenge #2
Data Streams

- Back to fixed-schema relational
- But data arrives as *continuous, unbounded* streams
  - Data may (or may not) be timestamped; order may (or may not) be relevant
“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:


Temperature Sensor 2:

“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:

Temperature Sensor 2:

★ Duplicate timestamps in streams?
★ If yes, is order relevant?
Model for Data Streams

“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:

Temperature Sensor 2:

★ Are timestamps coordinated across streams?
Duplicates? Order relevant?
Model for Data Streams

“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:


Temperature Sensor 2:


★ What about absolute duplicates?
Model for Data Streams

“A data stream is an unbounded sequence of [tuple timestamp] pairs”

Temperature Sensor 1:


Temperature Sensor 2:


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
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>

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

<table>
<thead>
<tr>
<th>person</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hector</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 \textit{model}) for uncertain data is \textbf{well-defined} if we know how to map any database in the model to its set of possible-instances.

- A model is \textbf{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*

Generates 4\textsuperscript{th} 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 $D$

- $D$
  - possible instances
  - implementation of $Op$
  - $D_1, D_2, ..., D_n$
  - $Op(D_1), Op(D_2), ..., Op(D_n)$

- $D'$
  - representation of instances
  - 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>

Four possible instances should be two

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

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 Challenge

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
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

\[
\text{UPDATE T SET A = A+5 WHERE A <= ALL (SELECT A FROM T)}
\]
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

```
SELECT day, count(*)
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

Example: Group-By (non)restrictions

```sql
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
Query Language Challenge #1
Active Databases

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!”

“We finished our rule system ages ago”

Jennifer Widom
Happy Ending

- Semantics got defined
  At least on our side of the SF Bay
- System got built
- Even SQL triggers standard ended up reasonable
  Primarily through restrictions
Query Language Challenge #2
Semistructured Data

Query: SELECT Student WHERE Advisor=‘Widom’

```
<Student>
  <ID> 123 </ID>
  <Name> Susan </Name>
  <Major> CS </Major>
</Student>
<Student>
  <ID> 567 </ID>
  <Name> Jennifer Widom </Name>
  <Major> CS </Major>
</Student>
```

• Error?
• Empty result?
• Warning?
Semistructured Data

Query: SELECT Student WHERE Advisor='Widom'

```
<Student>
  <ID> 123 </ID>
  <Name> Susan </Name>
  <Major> CS </Major>
</Student>
<Student>
  ... 
</Student>
```

Lore

- Empty result
- Warning
Semistructured Data

Query: SELECT Student WHERE Advisor='Widom'

XQuery

- Empty result

```
<Student>
  <ID> 123 </ID>
  <Name> Susan </Name>
  <Major> CS </Major>
</Student>
<Student>
  ...
</Student>
```
Semistructured Data

Query: SELECT Student WHERE Advisor='Widom'

<html>
  <Student>
    <ID>123</ID>
    <Advisor>Garcia</Advisor>
    <Advisor>Widom</Advisor>
  </Student>
  <Student>
    <Advisor>Widom</Advisor>
  </Student>
</html>
Semistructured Data

Query: SELECT Student WHERE Advisor=‘Widom’

XQuery

- Implicit ∃
- Variety of =

```xml
<Student>
  <ID> 123 </ID>
  <Advisor> Garcia </Advisor>
  <Advisor> Widom </Advisor>
</Student>

...<Student>
</Student>
```
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
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
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
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 Challenge

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

No happy ending (yet)
The “It’s Just SQL” Trap

Tables: **Attends(person,day)**  **Weather(day,temp)**

Query: Hector chill factor

```
SELECT W.temp
FROM Attends A, Weather W
WHERE A.day = W.day AND A.person='Hector'
```
The “It’s Just SQL” Trap

- 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>
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 Challenge

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

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

- Semistructured data
- Data streams
- Uncertain data
- <Insert future model here>
Stream — **Orders**(*cust*,*item*)
Table — **Items**(*item*,*store*,*price*)

```
SELECT * FROM Orders O, Items I
WHERE O.item = I.item
```

- Is result a stream, relation, or something else?
- Is output ordering relevant for duplicate items?
- What happens if table is updated?
Original Challenge

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
Exploit Relational Whenever Possible

Uncertain data — semantics of query $Q$

- $D$:
  - possible instances
- $D_1, D_2, \ldots, D_n$:
  - $Q$ on each instance
  - $Q(D_1), Q(D_2), \ldots, Q(D_n)$
- Result:
  - representation of instances

(implementation)
Exploit Relational Whenever Possible

Semantics of stream queries

Streams

Window

Relations

Relational

Istream / Dstream

30 years of refinement

Jennifer Widom
Exploit Relational Whenever Possible

- **Active databases:** “transition tables”
- **Lore:** semantics based on OQL

3 years of refinement
Current Research

TRIO

Data

Uncertainty

Lineage
Current Research

TRIO

Data
Uncertainty
Lineage

TRIO
Data
Uncertainty
Lineage

TRIO
Data
Uncertainty
Lineage

Jennifer Widom
Current Research

TRIO
Data
Uncertainty
Lineage

DUO
Data
Uncertainty
Lineage

QUATRO
Data
Uncertainty
Lineage
Current Research

TRIO
- Data
- Uncertainty
- Lineage

DUO
- Data
- Uncertainty
- Lineage

QUATRO
- Data
- Uncertainty
- Lineage
- Entity Resolution
Lineage: Where data came from and how it evolved over time

- Critical for scientific data applications
  - But useful for many others too

- Everyone wants it, nobody knows quite what it is
DUO – Features

• Lineage on query results
• Lineage to external data sources
• Processing-based lineage
• Rich constructs for querying lineage
• User-defined lineage creation and modification
• Lineage-assisted error detection and correction
• Efficient lineage management — compaction, approximation
• **Entity-Resolution**
  – Match & merge multiple data records representing same real-world entities
  – *Deduplication, record linkage, reference reconciliation, ...*

• **Premises**

  \[
  \text{Uncertainty} + \text{Lineage} + \text{DBMS} \Rightarrow \\
  \text{Ideal platform for Entity-Resolution}
  \]

  \[
  \text{Data} + \text{Uncertainty} + \text{Lineage} + \text{E-R} \Rightarrow \\
  \text{(Ideal?) platform for Data Integration}
  \]

  1. Trio-ER
  2. Quatro
Loop until fixed-point:

```
SELECT Combine *
FROM Hotel h1 ERJ Hotel h2
ON EXISTS [h1.zip = h2.zip
    AND (StrComp(h1.URL,h2.URL) > 0.95
        OR StrComp(h1.name,h2.name) > 0.95)]
```
Principled Database Systems Research

Data Model → Query Language → System

Impact

Jennifer Widom