The interaction of theory and practice in database research

Ron Fagin
IBM Research—Almaden
Purpose of This Talk

• Encourage collaboration between theoreticians and system builders
  – via two case studies
    • One initiated by the system builders, and one by the theoreticians

• For theoreticians:
  – How to apply theory to practice
  – Why applying theory to practice can lead to better theory

• For system builders:
  – The value of theory
  – The value of involving theoreticians
System Builders / Practitioners / Engineers

• Are strong software architects & programmers
• Understand system thinking
• Have a large vocabulary for software issues
• Work in teams to produce complex systems
• Have good insights into their problem domains
Theoreticians

• Are strong at precise, mathematical thinking
• Can generalize well and find insightful abstractions
• Can find good formal systems
• Know many algorithms and can come up with more
• Can prove properties of systems and algorithms
First Case Study: Garlic (1996)
Mr. Database Theoretician, we’ve got a problem with Garlic, our multimedia database system!

What was the problem?

Example databases:

- The answers to queries in DB/2 are sets
- The answers to queries in QBIC are sorted lists
- How do you combine the results?
Example

• Searching a CD database for Artist = “Beatles” yields a set, via, say DB/2

Musicbrainz has 12 million recordings in its DB
Example

- AlbumColor = “Red” yields a sorted list, via, say QBIC

![Images of albums with red covers]

<table>
<thead>
<tr>
<th>Album</th>
<th>Redness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weezer</td>
<td>.697</td>
</tr>
<tr>
<td>Lena</td>
<td>.683</td>
</tr>
<tr>
<td>Shuttertaste</td>
<td>.670</td>
</tr>
<tr>
<td>The Red Album</td>
<td>.659</td>
</tr>
<tr>
<td>Game</td>
<td>.629</td>
</tr>
</tbody>
</table>

Redness
• How do we make sense of 
  \((\text{Artist} = \text{‘Beatles’}) \land (\text{AlbumColor} = \text{‘Red’})\) ?
  – Here it is probably a list of albums by the Beatles, sorted by how red they are

• What about 
  \((\text{Artist} = \text{‘Beatles’}) \lor (\text{AlbumColor} = \text{‘Red’})\) ?

• And what about 
  \((\text{Color} = \text{‘Red’}) \land (\text{Shape} = \text{‘Round’})\) ?
What Was My Solution?

- These weren’t just sorted lists: they were **scored lists**
- Can view sets as scored lists (scores 0 or 1)
- This reminded me of fuzzy logic
- In fuzzy logic, conjunction (\(\land\)) is min, and disjunction (\(\lor\)) is max
Laura Haas

Use fuzzy logic

I like your solution. But we also need an efficient algorithm that can find the top k results while minimizing database accesses.

Ron Fagin

I have an algorithm that finds the top k with only $\sqrt{n}$ database accesses.

Good, that beats linear! But we database people are spoiled, and are used to only log n accesses. Be smarter and get me a log n algorithm.

I proved that you can’t do better than $\sqrt{n}$
Time for the Accesses

• Say \( n = 12,000,000 \) CDs
• Assume 1000 accesses per second
• \( n \) accesses (naïve algorithm) would take 3 hours
• \( \sqrt{n} \) accesses would take 3 seconds
Generalizing the Algorithm

• The algorithm works for arbitrary monotone scoring functions
  – increasing the scores of arguments cannot decrease the overall score
Influence

Algorithm implemented in Garlic
Influenced other IBM products, including
• Watson Bundled Search system
• InfoSphere Federation Server
• WebSphere Commerce

Paper introducing my algorithm (now called “Fagin’s Algorithm”) has over 800 citations (Google Scholar)
The Threshold Algorithm

- In 2001, we found the Threshold Algorithm

Amnon Lotem  Moni Naor  Ron Fagin
The Problem

• There are $m$ attributes
• Each object in a database has a score $x_i$ for attribute $i$
• The objects are given in $m$ sorted lists, one list per attribute
• Goal: Find the top $k$ objects according to a monotone scoring function, while minimizing access to the lists

• Can think of the attributes as voters, and the objects as candidates, where each voter assigns a score to each candidate
## Multimedia Example

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
<tr>
<td>139: 0.991</td>
<td>666: 0.996</td>
</tr>
<tr>
<td>702: 0.982</td>
<td>820: 0.992</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>235: 0.325</td>
<td>177: 0.406</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
</tbody>
</table>
Scoring Functions

• Let $f$ be the scoring function

• Popular choices for $f$:
  – $\min$ (used in fuzzy logic)
  – average

• Let $x_1, \ldots, x_m$ be the scores of object $R$ under the $m$ attributes

• Then $f(x_1, \ldots, x_m)$ is the overall score of object $R$
  – Sometimes write $f(R)$ to mean $f(x_1, \ldots, x_m)$

• A scoring function $f$ is monotone if whenever $x_i \leq y_i$ for every $i$, then $f(x_1, \ldots, x_m) \leq f(y_1, \ldots, y_m)$
Modes of Access

- **Sorted (or sequential) access**
  - Can obtain the next object and its score for attribute $i$
- **Random access**
  - Can obtain the score of object $R$ for attribute $i$
- **Wish to minimize total number of accesses**
Algorithms

- Want an algorithm for finding the top $k$ objects
- Naïve algorithm retrieves every score of every object
  - Too expensive
Threshold Algorithm

• Do sorted access in parallel to each of the \( m \) scored lists.

• As each object \( R \) is seen under sorted access:
  – Do random access to retrieve all of its scores \( x_1, \ldots, x_m \)
  – Compute its overall score \( f(x_1, \ldots, x_m) \)
  – If this is one of the top \( k \) answers so far, remember it

• For each list \( i \), let \( t_i \) be the score of the last object seen under sorted access

• Define the threshold value \( T \) to be \( f(t_1, \ldots, t_m) \). When \( k \) objects have been seen whose overall score is at least \( T \), stop

• Return the top \( k \) answers
Threshold Algorithm: Example (using min)

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
</tbody>
</table>

Scoring function is min
### Threshold Algorithm: Example (using min)

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>177: 0.406</td>
<td>. . .</td>
</tr>
</tbody>
</table>
Threshold Algorithm: Example (using min)

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>235: 0.325</td>
<td>177: 0.406</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Threshold Algorithm: Example (using min)**

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>235: 0.325</td>
<td>177: 0.406</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
</tbody>
</table>

Overall score for 177: \( \min(0.993, 0.406) = 0.406 \)

Overall score for 235: \( \min(0.325, 0.999) = 0.325 \)
Threshold Algorithm: Example (using min)

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>177: <strong>0.993</strong></th>
<th>235: 0.325</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROUNDNESS</th>
<th>235: <strong>0.999</strong></th>
<th>177: 0.406</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Overall score for 177: \(\min(0.993, 0.406) = 0.406\)
Overall score for 235: \(\min(0.325, 0.999) = 0.325\)
Threshold value: \(\min(0.993, 0.999) = 0.993\)
## Threshold Algorithm: Example (using min)

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
<tr>
<td>139: 0.991</td>
<td>666: 0.996</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>235: 0.325</td>
<td>177: 0.406</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Threshold Algorithm: Example (using min)

Threshold value: \( \min(0.991, 0.996) = 0.991 \)
## Threshold Algorithm: Example (using min)

<table>
<thead>
<tr>
<th>REDNESS</th>
<th>ROUNDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>177: 0.993</td>
<td>235: 0.999</td>
</tr>
<tr>
<td>139: 0.991</td>
<td>666: 0.996</td>
</tr>
<tr>
<td>702: 0.982</td>
<td>820: 0.992</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>235: 0.325</td>
<td>177: 0.406</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Threshold Algorithm: Example (using min)

Threshold value: \( \min(0.982, 0.992) = 0.982 \)
Correctness of the Halting Rule

Suppose the current top $k$ objects have scores at least $T$ (the current threshold).

Assume (by way of contradiction):

- $R$ unseen;
- $S$ in current top $k$; \[ f(R) > f(S) \]

$R$ has scores $x_1, \ldots, x_m$

$\Rightarrow x_i \leq t_i$ for every $i$ (as $R$ has not been seen)

$\Rightarrow f(R) = f(x_1, \ldots, x_m) \leq f(t_1, \ldots, t_m) = T \leq f(S)$

$\Rightarrow$ contradiction!
Instance Optimality

\[ A = \text{class of algorithms}, \]
\[ D = \text{class of legal inputs}. \]

For \( A \in A \) and \( D \in D \) have \( \text{cost}(A,D) \geq 0 \).

- An algorithm \( A \in A \) is \textbf{instance optimal} over \( A \) and \( D \) if there are constants \( c_1 \) and \( c_2 \) s.t. for every \( A' \in A \) and \( D \in D \)

\[
\text{cost}(A,D) \leq c_1 \cdot \text{cost}(A',D) + c_2.
\]

\( c_1 \) is called the \textbf{optimality ratio}
Instance Optimality of TA

Intuition about why TA is instance optimal:

Cannot stop any sooner, since the next object to be explored might have the threshold value.

But, life is a bit more delicate...
Wild guesses: random access for a field $i$ of object $R$ that has not been sequentially accessed before

- Neither FA nor TA use wild guesses
- Subsystem might not allow wild guesses
Theorem: For each monotone $f$ let

- $A$ be the class of algorithms that
  - correctly find top $k$ answers, with scoring function $f$, for every database.
  - Do not make wild guesses.

- $D$ be the class of all databases.

Then $TA$ is instance optimal over $A$ and $D$.

Optimality ratio is $m + m(m-1) \cdot c_R/c_S$ - best possible!
Our “threshold algorithm” is an even better algorithm (optimal in a stronger sense)

But Ron, you told me that your algorithm is optimal!?

Well, Laura, there is optimal, and then there is optimal
PODS

– PODS is the premier database theory conference
– Considered by the database community and a much broader community as a top-tier conference
– Strangely, the China Computer Federation rates PODS as a B-level conference
– A search of conference rankings on the web shows that every ranking except CCF’s gives PODS the highest ranking
Influence

• We submitted the paper to PODS ’01
• I was worried that the Threshold Algorithm was so simple that the paper would be rejected
  – So I called it a “remarkably simple algorithm”
  – The paper won the PODS Best Paper Award!

• The paper was very influential
  – Over 1800 citations (Google Scholar)
  – PODS Test of Time Award in 2011
  – IEEE Technical Achievement Award in 2011
  – Gödel Prize in 2014
  – Gems of PODS 2016
Applications of TA

- relational databases
- multimedia databases
- music databases
- semistructured databases
- text databases
- uncertain databases
- probabilistic databases
- graph databases
- spatial databases
- spatio-temporal databases
- web-accessible databases
- XML data
- web text data
- semantic web
- high-dimensional datasets

- information retrieval
- fuzzy data sets
- data streams
- search auctions
- wireless sensor networks
- distributed sensor networks
- distributed networks
- social-tagging networks
- document tagging systems
- peer-to-peer systems
- recommender systems
- personal information management systems
- group recommendation systems
- document annotation
Morals

• How did theory help?
  – Resolving Laura Haas’s dilemma
  – Knowledge of the literature (fuzzy logic)
  – Abstraction (using scoring functions)
  – Devising optimal algorithms and proving optimality

• Figure out the real problem
  – For example, there are scores, not just sorted lists

• Don’t stop at original problem
  – Example: doing a weighted version (with Ed Wimmers)
  – Led to a successful and influential body of work
Measures of Success

• Making our products better
  – An ultimate measure of success for practitioners

• Creating a new subfield
  – An ultimate measure of success for theoreticians

• A paper that arose by resolving a practical problem won the Gödel Prize!
Second Case Study: Clio (2003)
• Clio deals with “data exchange,” where we convert data from one format to another
• When Laura Haas started Clio, I followed her
• I attended Clio meetings for a year

Let’s start from scratch and lay the foundations for data exchange!
Data Exchange

Translate data from source format to target format

Source Schema $S$ to Target Schema $T$ through $\Sigma$:
- $I$ to $J$
Data Exchange

Data exchange is an old, but recurrent, database problem

– Phil Bernstein—2003
  “Data exchange is the oldest database problem”

– EXPRESS: IBM San Jose Research Lab—1977
  • Transforms data between hierarchical databases

– Data exchange underlies:
  • Data warehousing, ETL (Extract-Transform-Load), …
### Example

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP</td>
<td>EMP</td>
</tr>
<tr>
<td>MGR</td>
<td>DEPT</td>
</tr>
<tr>
<td></td>
<td>DEPT</td>
</tr>
<tr>
<td></td>
<td>MGR</td>
</tr>
</tbody>
</table>

- Relationship between source and target (the “schema mapping”) specified by tuple-generating dependencies ($tgds$)
  - Originally used to help specify “normal forms” for relational databases

\[
\text{EM}(e,m) \rightarrow \exists d \ (\text{ED}(e,d) \land \text{DM}(d,m))
\]
# Example

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EMP</strong></td>
<td><strong>EMP</strong></td>
</tr>
<tr>
<td>Fagin</td>
<td></td>
</tr>
<tr>
<td>Clarkson</td>
<td></td>
</tr>
<tr>
<td>Haas</td>
<td>Haas</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMP</th>
<th>MGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haas</td>
<td>Welser</td>
</tr>
</tbody>
</table>
# Example – 3 Possible Solutions

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP</td>
<td>MGR</td>
</tr>
<tr>
<td>Fagin</td>
<td>Haas</td>
</tr>
<tr>
<td>Clarkson</td>
<td>Haas</td>
</tr>
<tr>
<td>Haas</td>
<td>Welser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMP</th>
<th>DEPT</th>
<th>MGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fagin</td>
<td>Haas</td>
<td>Haas</td>
</tr>
<tr>
<td>Clarkson</td>
<td>Haas</td>
<td>Welser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMP</th>
<th>DEPT</th>
<th>MGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fagin</td>
<td>$d_1$</td>
<td>Haas</td>
</tr>
<tr>
<td>Clarkson</td>
<td>$d_1$</td>
<td>Welser</td>
</tr>
<tr>
<td>Haas</td>
<td>$d_2$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMP</th>
<th>DEPT</th>
<th>MGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fagin</td>
<td>$d_1$</td>
<td>Haas</td>
</tr>
<tr>
<td>Clarkson</td>
<td>$d_2$</td>
<td>Haas</td>
</tr>
<tr>
<td>Haas</td>
<td>$d_3$</td>
<td>Welser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMP</th>
<th>DEPT</th>
<th>MGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>Haas</td>
<td></td>
</tr>
<tr>
<td>$d_2$</td>
<td>Haas</td>
<td></td>
</tr>
<tr>
<td>$d_3$</td>
<td>Welser</td>
<td></td>
</tr>
</tbody>
</table>
Which Solution Should We Produce?

- We define a “universal” solution to be one as general as possible
- Third solution is universal
Target Constraints

• Might have target constraints specified by equality-generating dependencies (egds), like
  \[ DM(d,m) \land DM(d',m) \rightarrow (d = d') \]

• If this egd is a target constraint, then second solution is universal
How Do We Obtain a Universal Solution?

- There is a well-known mechanical procedure called the “chase”, originally used as a tool in database design.
- We use the chase to generate the target from the source efficiently.
- Example: \( EM(e, m) \rightarrow \exists d \ (ED(e, d) \land DM(d, m)) \)
  - From \( EM(Fagin, Haas) \), create \( ED(Fagin, d) \) and \( DM(d, Haas) \), where \( d \) is a newly introduced “labeled null”.
- The egds tell when to equate labeled nulls.
With Phokion Kolaitis, Lucian Popa, and Wang-Chiew Tan, we studied composition of schema mappings:
- Composition can take us out of first-order logic!
- We found the right language for composition ("second-order tgds")
- We gave an algorithm for composition
Measures of Success

• Used in DB2 Control Center, Rational Data Architect, and Content Manager
  – Using universal solutions
  – Using our algorithm to produce a universal solution, and our algorithm to compose schema mappings

• Our initial paper won the ICDT Test of Time Award in 2013.
  – With over 1100 citations, our paper was the 2nd most highly cited paper of the decade in the journal TCS

• Our follow-up paper on composition won the PODS Test of Time Award in 2014.

• This work created a new subfield
  – Special sessions on data exchange in every major database conference
Morals

• How did theory help?
  – Established principles rather than *ad hoc* approaches
  – Yielded algorithms for converting data, and for composing schema mappings

• Theorists need a partner to keep us honest

• Never too late to lay the foundations for an area, even for existing systems
  – Can cause essential changes and improvement

• Again, don’t stop at original problem
Conclusions
Conclusions (for System Builders)

• Consult with theoreticians
  – Explaining the problem is useful by itself
  – Principled approaches can improve your product
  – Better or new algorithms can differentiate your product
  – Algorithm analysis can provide performance expectations and provide product guarantees
  – Abstractions can expand the function of your product
Conclusions (for Theoreticians)

• Involvement with system builders can help your theory!
  – Novel questions will be asked
  – New models and new, interesting areas of study will arise
  – Implementation can reveal weaknesses in the theory
  – Theory will be relevant
  – **Practical impact!**