Where Is Database Research Headed?

Jeffrey D. Ullman

DASFAA

March 26, 2003

Outline

1. Core values for database research.
   - "Biggest" data.
   - Query optimization.
2. Some interesting directions.

New Directions

1. Information integration.
2. Stream processing.
3. Semistructured data and XML.
4. Peer-to-peer and grid databases.
5. Data mining.

Core Database Values

- Obvious: we deal with the largest amount of data possible.
- Less obvious: very-high-level languages.
  - Big data must be dealt with in uniform ways.
- Least obvious: query optimization essential for success.
  - Compare APL (failure) with SQL (success).

New Directions

1. **Information integration**.
2. Stream processing.
3. Semistructured data and XML.
4. Peer-to-peer and grid databases.
5. Data mining.

Information Integration

- Related sources of data need to be viewed as one whole.
- Applications: catalogs (seeing products from many suppliers), digital libraries, scientific databases, enterprise-wide information resources, etc., etc.

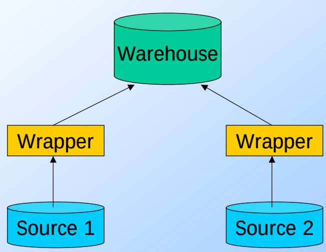
Local and Global Schemas

- Sources each have their own *local schema* = ways their data is stored, organized, and represented.
- Integration requires a *global schema* and mechanisms to translate between the global schema and each local schema.

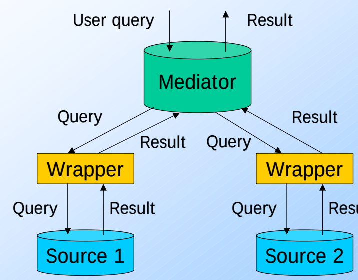
Two Approaches

1. *Warehousing* :
   - Collect data from sources into a “warehouse” periodically.
   - Do queries at the warehouse, while the sources execute transactions invisibly.
2. *Mediation* :
   - Virtual warehouse processes queries by translating between common schema and local schemas at sources.

Warehouse Diagram



A Mediator



Two Mediation Approaches

1. *Query-centric* : Mediator processes queries into steps executed at sources.
   - Enosys sells first example as BEA’s “liquid data.”
2. *View-centric* : Sources are defined in terms of global relations; mediator finds all ways to build query from views.

Very Simple Example

- Suppose Dell wants to buy a bus and a disk that share the same protocol.
- Global schema:

```
Buses(manf,model,protocol)
Disks(manf,model,protocol)
```

- Local schemas: each bus or disk manufacturer has a (model,protocol) relation --- manf is implied.

Example: Query-Centric

- Mediator might start by querying each bus manufacturer for model-protocol pairs.
  - The wrapper would turn them into triples by adding the manf component.
- Then, for each protocol returned, mediator queries disk manufacturers for disks with that protocol.
  - Again, wrapper adds manf component.

Example: View-Centric

- Sources' capabilities are defined in terms of the global predicates.
  - E.g., Hitachi's disk database could be defined by HitachiView(M,P) = Disks('Hitachi',M,P).
- Mediator discovers all combinations of a bus and disk "view," equijoined on the protocol components.
  - Theory: "answering queries using views" --- like fitting puzzle pieces together.

Comparison

- Query-centric is simpler to implement.
  - Lets you have control of what the mediator does.
- View centric is more extensible.
  - Same query engine works for any number of sources.
  - Add a new source simply by defining what it contributes as a view of the global schema.

Research Issues

- Optimization, optimization, optimization.
  - In query centric systems: how do we choose a plan?
    - E.g., is it better to ask about buses first, or disks?
  - In view-centric systems, how do we select a sufficient set of solutions to get most or all of the possible answers?

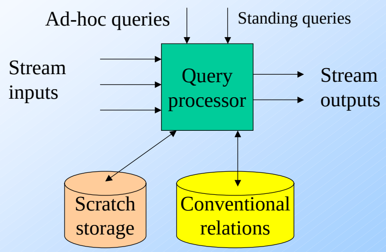
New Directions

1. Information integration.
2. **Stream processing.**
3. Semistructured data and XML.
4. Peer-to-peer and grid databases.
5. Data mining.

Stream Management Systems

- Adds to the relation a *stream* datatype = infinite sequence of tuples that arrive at a port one-at-a-time.
- Applications: Telecom billing, intrusion detection, monitoring Web hits, sensor networks, etc., etc.

Stream-DBMS Architecture



Stanford Approach (Widom, Motwani)

- Central idea is the *window*, a relation that is formed from a stream by some rule.
  - Examples: "last 10 tuples," "all tuples in the past 24 hours."
- Query language is SQL-like, with diction for converting a stream to a window to a relation.

Example:

```
SELECT …
FROM Stream1 [last 10] AS Window1,…
WHERE Window1.a = 5 AND …
```


MIT-Centered Approach (Stonebraker, others)

- Define and implement common operations on streams.
- Query language is algebraic: a sequence of operations to be applied to source streams and the results of other operations.

Research Challenges

- Again, optimization is central.
  - New language constructs and data types make old ideas less useful.
- Semantics is not 100% clear.
  - Example: when you join two windows created with different time limits, what does the result represent in terms of the original streams?
    - It matters if you want to apply algebraic laws to expressions.

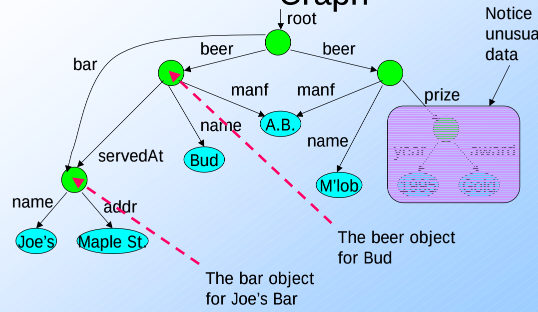
New Directions

1. Information integration.
2. Stream processing.
3. **Semistructured data and XML.**
4. Peer-to-peer and grid databases.
5. Data mining.

Semistructured Data

- This data model uses trees or graphs instead of relations.
- Key application: information integration, where global data is perceived as "flexible objects," with a variety of fields and structures.
- Evolved into XML, XSL, XPATH, XQUERY, etc.

Example: Semistructured Data Graph



XML and Semistructured Data

- XML (Extensible Markup Language) uses a semistructured data model to represent documents. Example:

```
<BARDOC><BAR><NAME>Joe's</NAME>
    <ADDR>Maple St.</ADDR></BAR>
<BAR> … </BAR> …
</BARDOC>
```


XML Applications

- Currently used as the document format for many systems that exchange information.
  - These documents rarely appear on the Web, so XML *appears* to be unused.
- XML documents may become the standard element in database systems.
  - Relation is a special case.

Querying XML Data

- XQUERY is new standard for querying XML documents.
  - Very-high-level, similar to SQL.
- Research just beginning on how to optimize queries about XML documents.
  - Successful techniques not like those applied successfully in SQL systems.

New Directions

1. Information integration.
2. Stream processing.
3. Semistructured data and XML.
4. **Peer-to-peer and grid databases**.
5. Data mining.

Peer-to-Peer and Grids

- Peer-to-peer systems are application-level attempts to share information and/or processes.
- Grid computing is an attempt to bring P2P support to the operating-system level.

P2P Applications

1. File sharing as in Napster, Kazaa, etc.
2. Specific scientific applications: Seti@home, Folding@home.
3. Distributed databases, e.g., digital libraries.
4. Replication within an intranet for high availability.

Additional Grid Goals

1. Scientific applications routinely solved using a network of workstations.
2. Reselling of unused cycles.
3. Global resources, e.g., buy your storage over the Internet rather than manage your own local disks.
4. Massive multiplayer games.

Grid Pro's and Con's

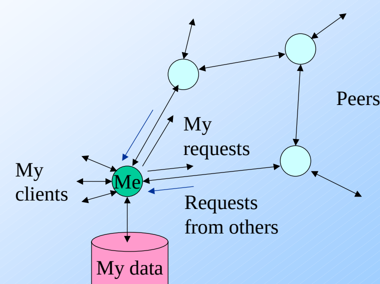
+ Possibly a good architecture for scientific computing.
+ Cross-platform support may lead to more P2P applications.

-- Businesses involving trade in resources among untrusted players is unlikely to win converts.

Peer-to-Peer Databases

- Data is distributed among independent sources.
- Similar to information-integration, but much looser constraints on cooperation.

P2P DBMS Architecture



P2P Research Issues

1. Strategies for trading storage.
   - How do I accept bids for someone to make a copy of my data? Will they keep it forever?
   - Storage auction strategies?
2. Query and search strategies.
   - How far to search?
   - How to manage competing requests?
   - Use of localized indexes?

Search Problem --- Continued

- Napster was a completely centralized index.
- Kazaa is a completely distributed index --- you can only find things by searching neighbors recursively.
- Optimum is undoubtedly some compromise, where nodes know about data at some, but not all, others.

New Directions

1. Information integration.
2. Stream processing.
3. Semistructured data and XML.
4. Peer-to-peer and grid databases.
5. **Data mining**.

Data Mining

- Means different things to different communities.
- Underlying theme: build models that represent data approximately.
- Examples: decision trees, clustering, hidden Markov models, Bayesian models, frequent itemsets (association rules) etc. etc.

The Database View

- Main research problem is how to implement very complicated queries on very large data efficiently.
  1. Invention of new algorithms or algorithms adapted to non-main-memory data.
  2. Can it be done in SQL? How do you optimize these queries?

Example --- Frequent Pairs

- Items={milk, coke, pepsi, beer, juice}.
- *Support* = pair appears in at least 3 "baskets."

B1 = {m, c, b}	B2 = {m, p, j}
B3 = {m, b}	B4 = {c, j}
B5 = {m, p, b}	B6 = {m, c, b, j}
B7 = {c, b, j}	B8 = {b, c}

- Frequent pairs: {m, b}, {c, b}, {j, c}.

Applications

1. Stores use frequent itemsets to plan layout of store, sale strategies.
   - Example, run sale on hamburger; raise the price of ketchup.
2. Looked at correctly ("item" = document, "basket" = sentence), frequent pairs = plagiarized documents.
3. Correlated pairs useful for on-line sellers to predict what you will buy.

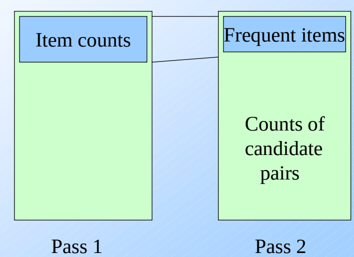
Frequent-Pair Algorithms

- Model: baskets in a file; "passes" stream the file, while main-memory is used to process in some way.
- Simplest idea: count all pairs in memory.
  - Limited by size-of-memory > O($items^2$).

A-Priori Algorithm

1. On first pass, count only the number of times each item appears.
2. Determine which items occur at least as many times as the support threshold *s*.
3. On second pass, count only pairs of items that both appear alone at least *s* times.

Picture of A-Priori



More Frequent-Pair Algorithms

- Hashed-based improvements take advantage of the fact that on the first pass, most of main memory is unused.
- Correlated-pair algorithms find rare, but correlated events, e.g., books bought by similar, small sets of customers.
  - "Min-hashing" is key idea.

Research Questions

- How fast can you compute frequent pairs with limited main memory?
- What is the best SQL query when data is stored as Baskets(bID, itemID), rather than as a list of basket contents?
- Similar questions in many other mining areas, e.g., clustering.

The End

- Thank you very much for listening.
- Now go out and solve some of these problems!