Topics in Databases (and some nostalgia)

Jennifer Widom
Stanford University

November '01

History

- Cornell Ph.D. 1987
  - Advisor: David Gries
  - Topic: Trace-Based Network Proof Systems
- One year as "Visiting Assistant Professor" at Cornell (thanks!)
  - Taught intro. programming, compilers, databases
- Five years at IBM Almaden Research Center
  - Switched from programming languages to databases
  - And from theory to systems

History (cont'd)

- Eight years (so far) at Stanford
  - Stuck to databases
  - Still more systems than theory
- Many interesting contrasts
  - Theory vs. systems
  - Programming languages vs. databases
  - Research lab vs. academia
  - East "coast" vs. west coast

Theory versus Systems

- Prove a (hard) theorem
- Write papers
- Get nice citations
- People use the result

- Design and build a (hard) system
- Write papers
- Get nice citations
- People use the system
- Requires more students

⇒ More funding & logistics

⇒ More management

- More fun!

Programming Languages versus Databases

From whom do you want respect?

- Strong undergraduates
- Strong Ph.D. students
- C.S. "purists"
- Funding agencies

- Weaker undergraduates
- Strong Ph.D. students
- Industry
- Funding agencies

Research Lab versus Academia

- Research Lab
  - 90% of time doing research
  - But those were "golden years" at IBM Almaden
- University
  - 10% of time doing research
  - 40% of time managing research (i.e., students)
  - 20% of time managing other stuff
  - 30% of time teaching

Research Lab vs. Academia (cont'd)

- Who's your boss at a research lab?
  - "Big brother"
  - Product groups, customers
  - In golden years: yourself
- Who's your boss at a university?
  - Department chair, Dean
  - Funding agencies
  - Customers (students)
  - Mostly: yourself

East "Coast" versus West Coast

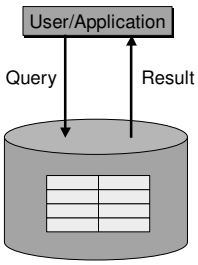
All the clichés are true

Database Research

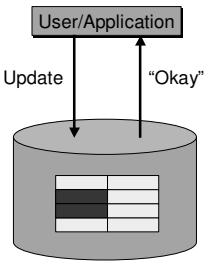
- Active Databases (1988-1993)
- Data Warehousing (1993-2001)
- Semistructured Data and XML (1994-2000)
- Querying Data Streams (2001-present)
- Miscellaneous
  - Approximate caching
  - Databases and the Web
  - Similarity
  - Constraints and Views
  - Temporal Databases
  - Heterogeneous and Distributed Databases

Active Databases

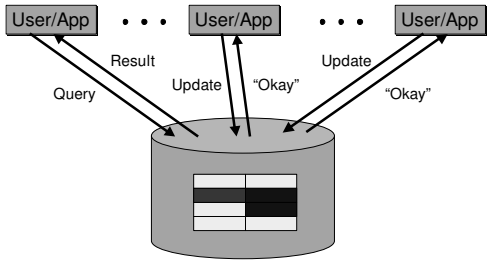
Traditional (Passive) Databases



Traditional (Passive) Databases



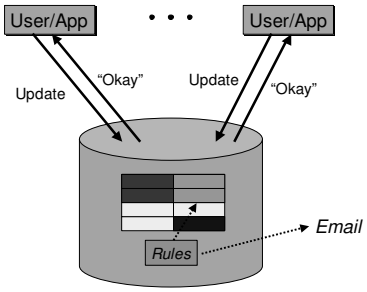
Traditional (Passive) Databases



Active Databases

- Can specify rules (triggers) activated by database updates
  - *If professor's rank is updated from 'associate' to 'full', raise salary by $20K*
  - *If average monthly sales fall below $50K, send email to field manager*
- Rules are part of database, activated regardless of user/application

Active Databases



Why Active Databases?

- Maintain or monitor constraints on data
  - *Sum of salary and bonus is < $100K*
  - *Total number of students in 300-level class is fewer than 50*
  - ⇒ Beyond what DBMS constraint system provides
  - ⇒ Programmable constraint enforcement
- Specify business logic within database
  - *When valued customer cancels order, offer discount*
  - *When GPA falls below 2.5, notify parents*

Why Active Databases? The Bottom Line

To move common functionality from applications into DBMS

- For performance
- For software reliability

What's Hard About Active Databases?

- Rule definition language
  - Syntax easy

    Event-condition-action rules based on SQL

    Event: insert, delete, update on database table

    Condition: predicate over modified data and rest of database

    Action: database modification or procedure call
  - Semantics very tricky!

Rule Language Semantics

- Many concurrent transactions
  - Each with many table-level update commands
  - … which generate many tuple-level updates
- Many rules triggered by each update
  - Rule actions may perform table-level update commands
  - … which generate many tuple-level updates
- Condition evaluation time, action execution time, rule ordering, sequential versus recursive activation, …

What Else is Hard?

- Implementing rule system in a real Database Management System (DBMS)
  - DBMSs are complex, monolithic pieces of software
  - Ex: What is authorization model for rules and rule execution?
  - Performance and scalability of utmost concern
  - Ex: transactions triggering no rules must incur no overhead due to rules

What Else?

- Programming rules is hard
  - But in many cases rules can be generated from higher-level (declarative) specifications
    - Constraints
    - Materialized views
    - Business logic
- Understanding rule behavior is hard
  - Are rules guaranteed to terminate?
  - Are they confluent?
  - ⇒ Rule analysis techniques

The (Sort of) Happy Ending

- Rules (triggers) standardized in SQL-99
  - Most products do not support all features
  - Many products do not adhere to execution model
- Syntactic restrictions on sets of rules
  - \+ Eliminates tricky interactions
  - – Limited expressiveness
- Performance and scalability
  - Verdict still out

The Happy Ending (cont'd)

- Automatic rule generation is promising
  - DBMS kernel
    - Referential integrity
    - Materialized views
  - DBMS services
    - Replication
    - "Extenders"
    - Audit trails
  - External applications
    - Workflow management

Semistructured Data and XML

Formula for A Database Research Project

- Choose an assumption underlying database systems
  - Simple and fundamental, but constraining
- Drop it
- Assumption

  Each database has a fixed schema
- Dropped

  "Semistructured" data (LORE project)

DB Research Project Formula (cont'd)

- Assumption

  Data is stored in finite sets (relations)
- Dropped

  Data streams (STREAM project)
- In both cases:
  - Need to reconsider nearly all aspects of data management and processing
  - Challenging and fun to build a prototype

The LORE Project

- Goal:

  Build a complete database management system for semistructured data

The LORE Project

- Goal:

  Build a complete database management system for XML

Semistructured Data

- Traditional DBMSs expect data to adhere to predefined, stable, structured schema
- In reality data often has irregular, dynamic, unpredictable structure
- LORE reconsidered all aspects of data management in presence of semistructured data
- Fortuitous emergence of XML during project

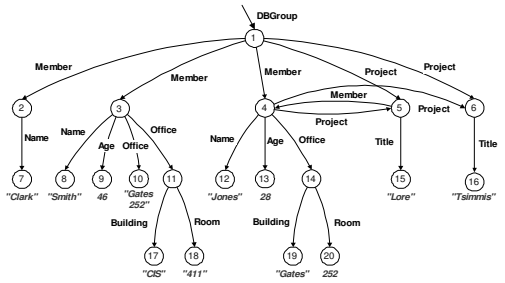
LORE Research Areas

- Data model for semistructured data
- Querying semistructured data effectively
- Indexing semistructured data
- Query processing and optimization
- *DataGuide* -- dynamic structural summary
- Keyword and proximity search
- Migration to XML

LORE Data Model

- Directed labeled graphs
- Essentially = XML with IDREF's

Example LORE Database



Portion of Example Database in XML

```
<Member ID="m1" Projects="p1 p2">
   <Name>Jones</Name>
   <Age>28</Age>
   <Office>
      <Building>Gates</Building>
      <Room>252</Room>
   </Office>
</Member>
<Project ID="p1" Member="m1">
   <Title>Lore</Title>
</Project>
...
```

Query Language

- Traditional relational
  - Tables and attributes
  - Regular, known structure
  - ➢ Relational algebra
  - ➢ SQL
- LORE (LOREL)
  - Graph-structured data
  - Irregular, incomplete
  - Structure may be evolving and partially unknown
  - ➢ Path expressions
  - ➢ Wildcards
  - ➢ No errors (warnings OK)
  - ➢ Extension of OQL

Indexes

- Traditional relational
  - ➢ Attribute indexes

  find all tuples with A < 5
- Object-oriented DBMS
  - ➢ Path indexes

  find all paths with A.B.C = 5
- LORE
  - ➢ Value indexes
  - ➢ Link indexes
  - ➢ Edge indexes
  - ➢ Path indexes
  - ➢ Text indexes
  - ➢ Distance indexes

Query Optimization

- Traditional relational
  - Explore space of possible query plans, pick estimated cheapest
  - ➢ Access methods
  - ➢ Join methods
  - ➢ Join ordering
- LORE
  - Explore space of possible query plans, pick estimated cheapest
  - ➢ Access methods
  - ➢ Graph traversal

Schema

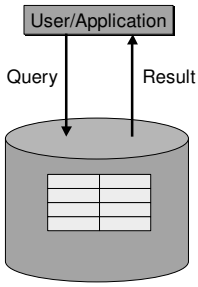
- Traditional relational
  - Schema drives data
  - Never changes
- LORE
  - Schema inferred from data (DataGuide)
  - Changes when data changes

The Happy Ending

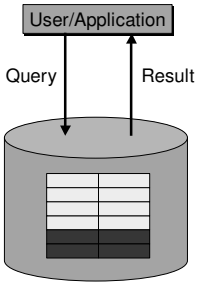
- Comprehensive prototype
  - 180,000 lines of C++ code
  - Semistructured and XML versions (were) publicly available
  - Reasonable number of users
- Commercial perspective
  - LORE was too early in XML wave
  - LORE clones now appearing in marketplace
  - Also XML support in traditional DBMS products

Querying Data Streams

Traditional (Stored Data) Databases



Traditional (Stored Data) Databases



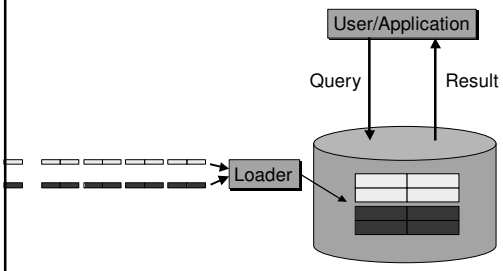
Data Streams

- In some applications data takes the form of continuous, rapid, time-varying data streams
  - Network monitoring and traffic engineering
  - Web logs and click-streams
  - Sensor networks
  - Manufacturing processes
  - Telecom call records
  - Financial monitoring
  - Etc.

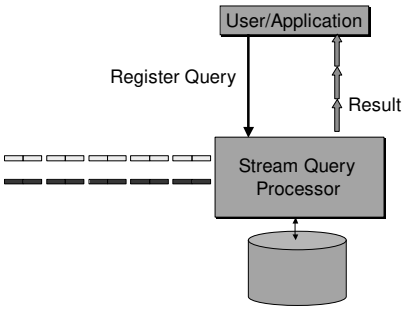
Data Stream Queries

- Continuous rather than one-time
- Complex, beyond element-at-a-time processing

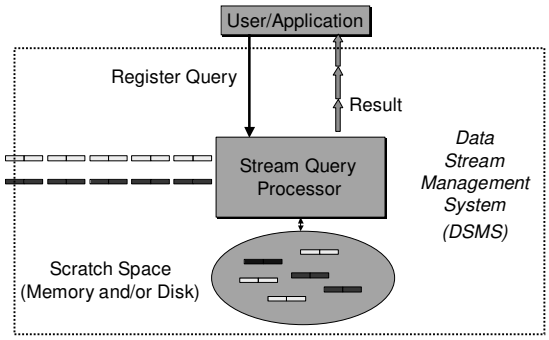
Using Traditional Database



New Approach for Data Streams



New Approach for Data Streams



DBMS versus DSMS

- Relations stored on disk
- Data access patterns controlled by query processor
- One-time queries
- Tuples read once and discarded
- Data must be processed as it arrives
- Continuous queries

Project Results So Far

- Some theory: Characterizing memory requirements for queries over continuous data streams
- Some design: Query processing strategies for continuous queries over data streams that minimize memory use
- Some implementation: Infrastructure for comprehensive prototype DSMS

Query Processing Approach

- Generate query plans, just like traditional DBMS's
- Except if streams are coming too fast, may need to sample or approximate in some other way
- Except if unbounded memory is required, may need to sample or build special-purpose synopses

Query Processing (cont'd)

- Continuous queries are long-running, cannot predict data statistics or system parameters
- Need adaptive query plans
- Our approach: dynamically adjustable tradeoff among
  - Efficiency
  - Accuracy
  - Storage

  at level of query plan operator and query

Other Interesting Aspects

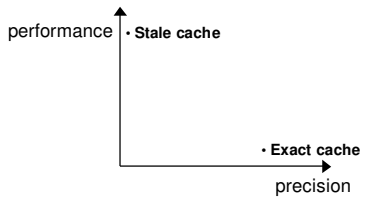
- Multiple continuous queries
  - Long-running
  - May be very large number of them
  - Shared resources
- Distributed streams
  - Many "physical" streams but one "logical" stream
  - Cannot process in one place
  - Ex: Find top 100 visited pages at Yahoo

Approximate Caching

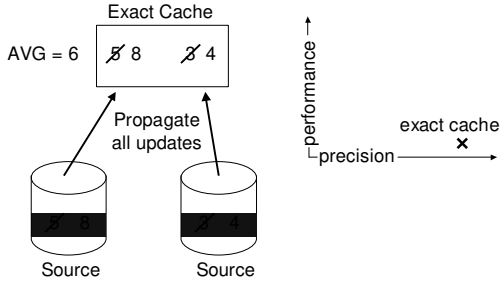
TRAPP

Tradeoff in ReplicAtion Precision and Performance

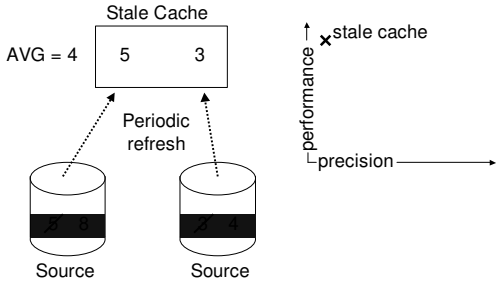
Data replication alternatives:



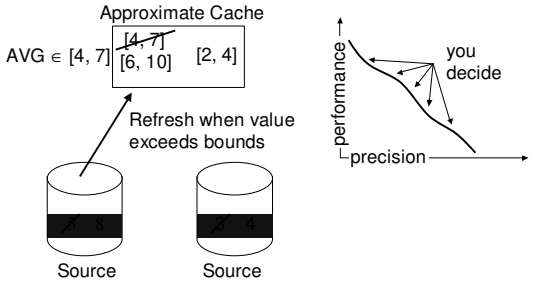
Replication Alternative 1: Exact Caching



Replication Alternative 2: Stale Caching



TRAPP Replication: Approximate Caching



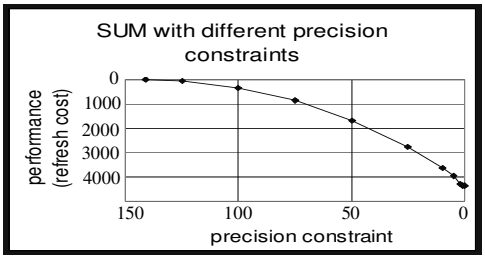
TRAPP Overview

- Caches store bounds that include exact source values
  - Source sends refresh when value exceeds bounds
- Queries over cached data include a precision constraint
- TRAPP answers queries by refreshing as few values as possible while guaranteeing to meet precision constraint

TRAPP Processing

- Query type determines algorithm for selecting minimum-cost refreshes
  - Some easy, some hard
- Setting approximations
  - Too "narrrow": many source-initiated refreshes
  - Too "wide": many query-initiated refreshes
  - Adaptive algorithm

Precision-Performance Tradeoff



TRAPP Directions

- Non-numeric data
- Continuous queries
- Related topic: "Best-effort" cache synchronization
  - TRAPP constrains precision, minimizes communication
  - "Best-effort" constrains communication, maximizes precision

Conclusion

The Happy Ending

- Planning to stick with …
  - Systems
  - Databases
  - Academia
  - West coast
- For more information on everything covered:

  http://www-db.stanford.edu/~widom