Query Processing over Data Streams

Joint project with Prof. Rajeev Motwani and a group of graduate students

Formula for a Database Research Project

- Pick a simple but fundamental assumption underlying traditional database systems
  - Drop it
- Reconsider all aspects of data management and query processing
  - Many Ph.D. theses
  - Prototype from scratch

Following the Formula

- We followed this formula once before
  - The LORE project
    - Dropped assumption: Data has a fixed schema declared in advance
    - XML
- The STREAM Project
  - Dropped assumption:
    - First load data, then index it, then run queries
    - Continuous data streams (+ continuous queries)
Data Streams

- Continuous, unbounded, rapid, time-varying streams of data elements
- Occur in a variety of modern applications
  - Network monitoring and traffic engineering
  - Sensor networks
  - Telecom call records
  - Financial applications
  - Web logs and click-streams
  - Manufacturing processes
- DSMS = Data Stream Management System

The STREAM System

- Data streams and stored relations
- Declarative language for registering continuous queries
- Designed to cope with high data rates and query workloads
  - Graceful approximation when needed
  - Careful resource allocation and usage

The STREAM System

- Designed to cope with bursty streams and changing workloads
  - Continuous monitoring of data and system behavior
  - Continuous reoptimization
- Relational, centralized (for now)
Contributions to Date

- Language for continuous queries
- Query plans
- Query optimization and dynamic reoptimization
- Exploiting stream constraints
- Operator scheduling
- Approximation techniques
- Resource allocation to maximize precision
- Initial running prototype
Language for Continuous Queries

GOALS

• Continuous queries over multiple streams and relations
• Exploit relational semantics to the extent possible
• Easy queries should be easy to write
• Queries should do what you expect

Example Query 1
Two streams, contrived for ease of examples:
Orders (orderID, customer, cost)
Fulfillments (orderID, clerk)
Total cost of orders fulfilled over the last day by clerk “Sue” for customer “Joe”
Select Sum(O.cost) From Orders O, Fulfillments F [Range 1 Day]
Where O.orderID = F.orderID And F.clerk = “Sue” And O.customer = “Joe”

Example Query 2
Using a 10% sample of the Fulfillments stream, take the 5 most recent fulfillments for each clerk and return the maximum cost
Select F.clerk, Max(O.cost) From Orders O, Fulfillments F [Partition By clerk Rows 5] 10% Sample
Where O.orderID = F.orderID Group By F.clerk
Abstract Semantics and Concrete Language

- Abstract: window specification language and relational query language as “black boxes”
- Concrete: SQL-based instantiation for our system: includes syntactic shortcuts, defaults, equivalences

Relations and Streams

- Assume global, discrete, ordered time domain
- Relation
  - Maps time $T$ to set-of-tuples $R$
- Stream
  - Set of (tuple, timestamp) elements

Conversions

Window specification

Streams

Special operators: $istream, Dstream, Rstream$

Relations

Any relational query language
Conversion Definitions

• Stream-to-relation
  – $S[W]$ is a relation — at time $T$ it contains all tuples in window $W$ applied to stream $S$ up to $T$
  – When $W = \infty$, contains all tuples in stream $S$ up to $T$
• Relation-to-stream
  – $I_{stream}(R)$ contains all $(r, T)$ where $r \in R$ at time $T$ but $r \not\in R$ at time $T-1$
  – $D_{stream}(R)$ contains all $(r, T)$ where $r \in R$ at time $T-1$ but $r \not\in R$ at time $T$
  – $R_{stream}(R)$ contains all $(r, T)$ where $r \in R$ at time $T$

Abstract Semantics

• Take any relational query language
• Can reference streams in place of relations
  – But must convert to relations using any window specification language
    (default window = $[\infty]$)
• Can convert relations to streams
  – For streamed results
  – For windows over relations
    (note: converts back to relation)

Query Result at Time $T$

• Use all relations at time $T$
• Use all streams up to $T$, converted to relations
• Compute relational result
• Convert result to streams if desired
Abstract Semantics – Example 1

Select F.clerk, Max(O.cost) From O [\infty], F [Rows 1000]
Where O.orderID = F.orderID
Group By F.clerk

• Maximum-cost order fulfilled by each clerk in last 1000 fulfillments

Abstract Semantics – Example 1

Select F.clerk, Max(O.cost) From O [\infty], F [Rows 1000]
Where O.orderID = F.orderID
Group By F.clerk

• At time $T$: entire stream $O$ and last 1000 tuples of $F$ as relations
• Evaluate query, update result relation at $T$

Abstract Semantics – Example 1

Select IsStream(F.clerk, Max(O.cost)) From O [\infty], F [Rows 1000]
Where O.orderID = F.orderID
Group By F.clerk

• At time $T$: entire stream $O$ and last 1000 tuples of $F$ as relations
• Evaluate query, update result relation at $T$
• Streamed result: New element $\langle$clerk,max$, T\rangle$ whenever $\langle$clerk,max$\rangle$ changes from $T^{-1}$

Abstract Semantics – Example 2

Relation CurPrice(stock, price)
Select stock, Avg(price)
From Istream(CurPrice) [Range 1 Day]
Group By stock
• Average price over last day for each stock

Abstract Semantics – Example 2

Relation CurPrice(stock, price)
Select stock, Avg(price)
From Istream(CurPrice) [Range 1 Day]
Group By stock
• Istream provides history of CurPrice
• Window on history, back to relation, group and aggregate

Concrete Language – CQL
• Relational query language: SQL
• Window spec. language derived from SQL-99
  – Tuple-based, time-based, partitioned
• Syntactic shortcuts and defaults
  – So easy queries are easy to write and simple queries do what you expect
• Equivalences
  – Basis for query-rewrite optimizations
  – Includes all relational equivalences, plus new stream-based ones
Two Extremely Simple CQL Examples

Select * From Strm
• Had better return Strm (It does)
  – Default ∞ window for Strm
  – Default Istream for result

Select * From Strm, Rel Where Strm.A = Rel.B
• Often want "NOW" window for Strm
• But may not want as default

Stream Systems
• (At least) three general-purpose DSMS prototypes underway
  – STREAM (Stanford)
  – Aurora (MIT, Brown, Brandeis)
  – TelegraphCQ (Berkeley)
• All will be demo’d at conference next week
• Cooperating to develop stream system benchmark
  ➢ Goal: demonstrate that conventional systems are far inferior for data stream applications

Contributions to Date
• Language for continuous queries
• Query plans
• Query optimization and dynamic reoptimization
• Exploiting stream constraints
• Operator scheduling
• Approximation techniques
• Resource allocation to maximize precision
• Initial running prototype
http://www-db.stanford.edu/stream