STREAM: The Stanford Stream Data Manager (Demonstration Description)*

Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar,
Keith Ito, Itaru Nishizawa, Justin Rosenstein, and Jennifer Widom
Stanford University
http://www-db.stanford.edu/stream

STREAM is a general-purpose relational *Data Stream Management System* (*DSMS*). STREAM supports a declarative query language and flexible query execution plans. It is designed to cope with high data rates and large numbers of continuous queries through careful resource allocation and use, and by degrading gracefully to approximate answers as necessary. A description of language design, algorithms, system design, and implementation as of late 2002 can be found in [3]. The demonstration focuses on two aspects:

- Registering and observing multiple continuous queries over multiple continuous data streams and updatable relations. Queries are registered using our declarative query language *CQL* [1]. In addition, query plans can be entered directly through a graphical interface or using XML.
- An interactive interface for online visualization and management of query execution and resource utilization, and for easy experimentation with DSMS query processing techniques.

A simplified view of a centralized DSMS such as STREAM is shown in Figure 1. On the left are the incoming *Data Streams*, which produce data indefinitely and drive query processing. Although we are concerned primarily with the online processing of continuous queries, stream data also may be copied to an *Archive*, for preservation and offline processing of expensive analysis or mining queries. Finally, processing of continuous queries typically requires intermediate state, which we denote as *Scratch Store* in the figure. This state can be stored and accessed in memory or on disk.

Across the top of the figure we see that users or applications register *Continuous Queries*, which remain active in the system until they are deregistered. Results of continuous queries are generally transmitted as output data streams, but they can also be relational results that are updated over time (similar to materialized views).

When a continuous query is registered using our declarative query language CQL it is compiled into a *query plan*. A separate plan is generated for each query, then the system merges plans with matching or similar subplans. Alternatively, for experimentation with different optimization strategies, query plans can be entered directly and plans can be merged manually.

*This work was supported by NSF Grants IIS-0118173 and IIS-9817799, by Stanford Graduate Fellowships from 3Com and Rambus, and by a Siebel Scholarship. Nishizawa is visiting Stanford from Hitachi, Ltd.

SIGMOD 2003, June 9-12, 2003, San Diego, CA.

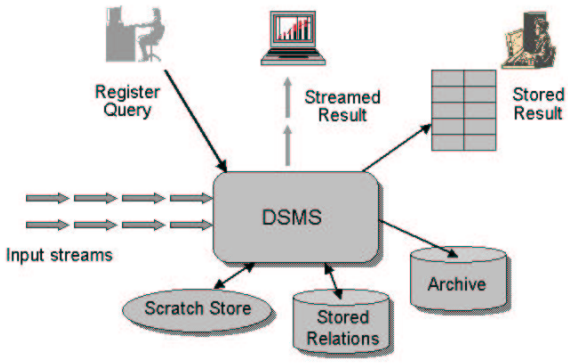

Figure 1: Simplified View of Centralized DSMS.

A query plan runs continuously and is composed of: (1) query *operators*, similar to those in a traditional DBMS; (2) inter-operator *queues*, also similar to the approach taken by some traditional DBMS's; and (3) *synopses*, used to maintain state associated with operators. Operators include the standard relational operators over streams, general-purpose *window* and *sample* operators, and efficient windowed versions of some operators (e.g., aggregation, join, and duplicate elimination).

Note that the queues and synopses in active query plans comprise the *Scratch Store* depicted in Figure 1. Multiple queries and operators share synopses and queues whenever possible. In addition, operators execute incrementally whenever possible, and a number of query rewrites are performed: applicable relational rewrites as well as new rewrites based on windows and other stream-specific features. Execution of query operators is controlled by a global *scheduler*, using a variant of the novel algorithm described in [2].

1. REFERENCES

[1] A. Arasu, S. Babu, and J. Widom. An abstract semantics and concrete language for continuous queries over streams and relations. Technical report, Stanford University Database Group, 2002. `http://dbpubs.stanford.edu:8090/pub/2002-57`.

[2] B. Babcock, S. Babu, M. Datar, and R. Motwani. Chain: Operator scheduling for memory minimization in data stream systems. In *Proc. ACM SIGMOD International Conference on Management of Data*, San Diego, California, June 2003.

[3] R. Motwani, J. Widom, et al. Query processing, approximation, and resource management in a data stream management system. In *Proc. First Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 245–256, Monterey, California, January 2003.