Data Stream Query Processing

Nick Koudas and Divesh Srivastava
AT&T Labs-Research

Stream Map

- Part I: Motivation
  - Data streams: what, why now, applications, architecture
- Part II: Query processing, system issues

Data Streams: What and Where?

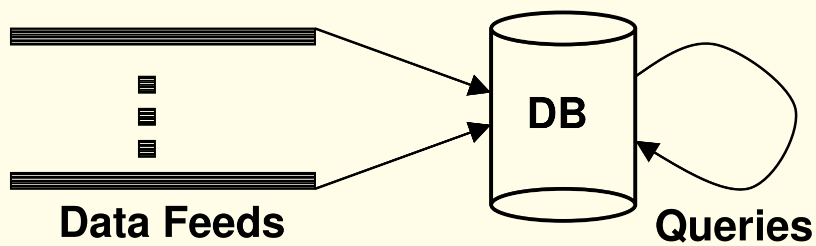
- A **data stream** is a (potentially unbounded) sequence of tuples

- **Transactional** data streams: log interactions between entities
  - Credit card: purchases by consumers from merchants
  - Telecommunications: phone calls by callers to dialed parties
  - Web: accesses by clients of resources at servers

- **Measurement** data streams: monitor evolution of entity states
  - Sensor networks: physical phenomena, road traffic
  - IP network: traffic at router interfaces
  - Earth climate: temperature, moisture at weather stations

Data Streams: Why Now?

- Haven't data feeds to databases always existed? Yes
  - Modify underlying databases, data warehouses
  - Complex queries are specified over stored data

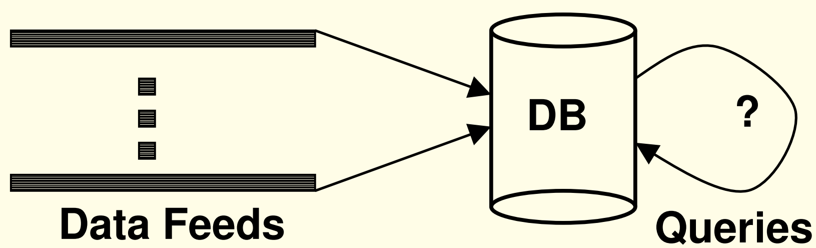


- Two recent developments: application- and technology-driven
  - Need for sophisticated near-real time queries/analyses
  - Massive data volumes of transactions and measurements

Data Streams: Real-Time Queries

- With traditional data feeds
  - Simple queries (e.g., value lookup) needed in real-time
  - Complex queries (e.g., trend analyses) performed offline

- Now need sophisticated near-real time queries/analyses
  - AT&T: fraud detection on call detail tuple streams
  - NOAA: tornado detection using weather radar data



Telecommunications Application: Fraud Detection

- Business Challenge: AT&T wanted to track calling pattern of each of ~100M callers, and raise real-time fraud alerts
- Previous Approach: Handwritten, optimized C code, computing evolving **signatures** for each customer, looking for variations
- Issues: Signature computation is I/O intensive, often modified

- Solution: Using Hancock domain-specific language
  - Abstract logical/physical streams and signatures
  - Express I/O and CPU efficient signature programs cleanly

- Lesson: Essential to consider I/O issues for data streams

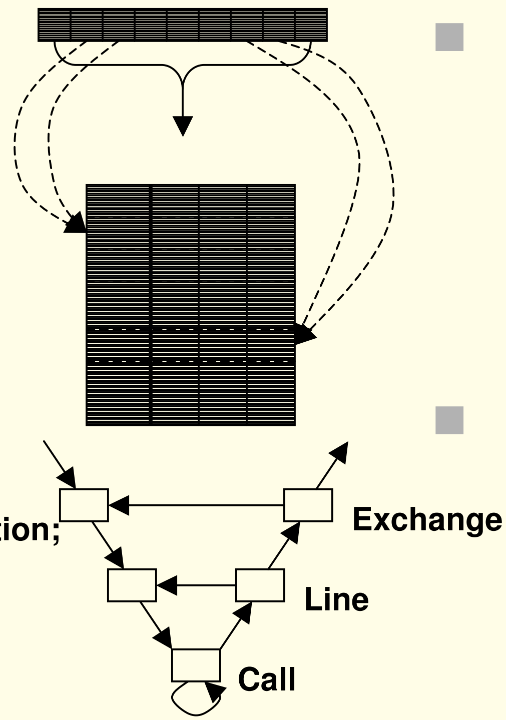
Hancock: Data Streams

```
typedef struct {
    line_t origin;
    line_t dialed;
    date_t connectTime;
    time_t duration;
    char isIncomplete;
    char isIntl;
    char isTollFree;
    …
} callRec_t;
```

- Physical data representation of tuples on disk
  - highly encoded structure
- Logical data representation
  - C struct
- Conversion functions
  - specified in Hancock

Hancock: Signature Programs

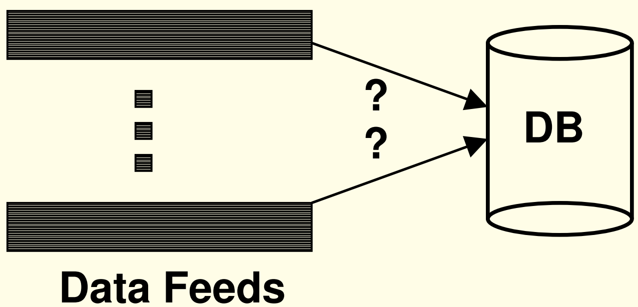
```
iterate (over calls
  sortedby origin
  filteredby noIncomplete
  withevents originDetect){

event line_begin(lpn_t pn){
  cumSec.outTF = 0;
}

event call(callRec_t c){
  if (c.isTollFreeCall)
    cumSec.outTF += c.duration;
}

event line_end(lpn_t pn){
  mySig us = data<:pn:>;
  us.outTF = blend(cumSec.outTF, us.outTF);
  data<:pn:> := us;
}}
```



- Hancock program paradigm:
  - stream-in, relation-out
  - block processing of data
  - multiple passes on block
- Hancock programs support:
  - iterating on sorted data
  - filtering
  - event clause hierarchy
- User-defined aggregation

Data Streams: Massive Volumes

- Now able to deploy transactional data observation points
  - AT&T long-distance: ~300M call tuples/day
  - AT&T IP backbone: >10B IP flows/day
- Now able to generate automated, highly detailed measurements
  - NOAA: satellite-based measurement of earth geodetics
  - Sensor networks: huge number of measurement points



IP Network Application: P2P Traffic Detection

- Business Challenge: AT&T IP customer wanted to accurately monitor P2P traffic evolution within its network
- Previous Approach: Netflow can be used to determine P2P traffic volumes using TCP port number found in Netflow data
- Issues: P2P traffic might not use known P2P port numbers

- Solution: Using Gigascope SQL-based packet monitor
  - Search for P2P related keywords within each TCP datagram
  - Identified 3 times more traffic as P2P than Netflow

- Lesson: Essential to query massive volume data streams

IP Network Application: Web Client Performance Monitoring

- Business Challenge: AT&T IP customer wanted to monitor latency observed by clients to find performance problems
- Previous Approach: Measure latency at “active clients” that establish network connections with servers
- Issues: Use of “active clients” is not very representative

- Solution: Using Gigascope SQL-based packet monitor
  - Track TCP synchronization and acknowledgement packets
  - Report round trip time statistics: latency

- Lesson: Essential to correlate multiple data streams

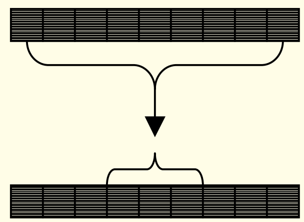
Gigascope: Data Streams

```
PROTOCOL IP (Layer2) {
    uint ipversion
}
PROTOCOL IPv4(IP) {
    uint hdr_length;
    uint service_type;
    uint total_length;
    uint id;
    bool do_not_fragment;
    bool more_fragments;
    uint offset;
    uint ttl;
    uint protocol;
}
```

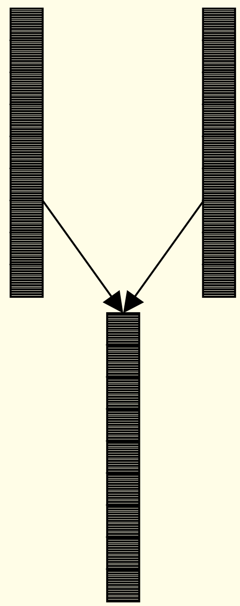
- GSQL queries get raw data from low level schemas
  - defined at packet level
  - inherits from lower layer
- Current schemas include
  - layer 2: ETH/HDLC
  - layer 3: IP/IPv4
  - layer 4: UDP/TCP/ICMP
  - layers 5-7: packet data

Gigascope: GSQL Queries

```
select tb, srcIP, sum(len)
from IPv4
where protocol = 6
group by time/60 as tb, srcIP
having count(*) > 5
```



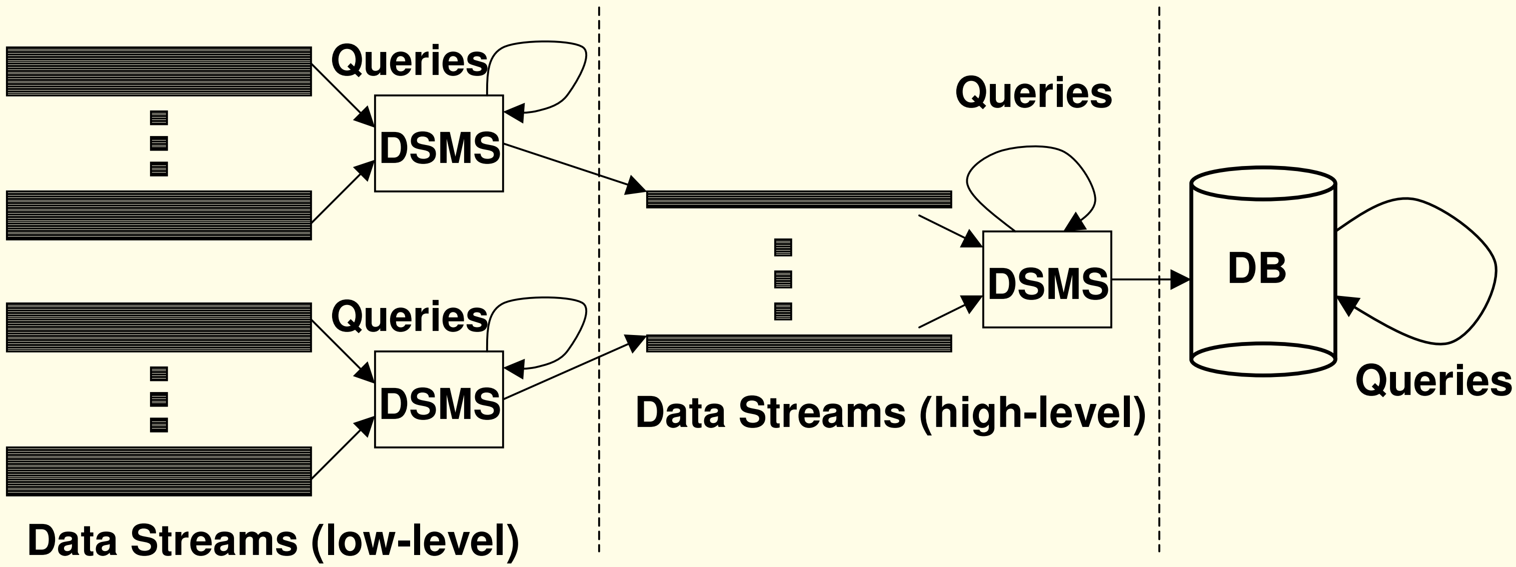
```
select S.tstmp,
       S.srcIP, S.destIP,
       S.srcPort, S.destPort
       (A.tstmp – S.tstmp) as rtt
from tcp_syn S, tcp_syn_ack A
where S.srcIP = A.destIP
       and S.destIP = A.srcIP
       and S.srcPort = A.destPort
       and S.destPort = A.srcPort
       and S.tb = A.tb
```



- GSQL queries support:
  - filtering, aggregation
  - merging data streams
  - joining data streams
  - hand-coded views
  - external functions
- GSQL query paradigm:
  - streams-in, stream-out
  - permits composability
  - permits data reduction

Data Streams: 3-Level Architecture

- Data stream management system at multiple observation points
  - (Voluminous) streams-in, (data reduced) streams-out
- Database management system
  - Data feeds to database can also be treated as data streams



Data Streams: 3-Level Architecture

Data Stream Systems

- Resource (memory, per-tuple computation) limited, especially at low-level
- Reasonably complex, near real time, query processing
- Useful to identify what data to populate in database

Database Systems

- Resource (memory, disk, per-tuple computation) rich
- Useful to audit query results of data stream system
- Supports sophisticated query processing, analyses

Databases vs Data Streams: Issues

Database Systems	Data Stream Systems
Model: persistent relations	Model: transient relations
Relation: tuple set/bag	Relation: tuple sequence
Data Update: modifications	Data Update: appends
Query: transient	Query: persistent
Query Answer: exact	Query Answer: approximate
Query Evaluation: arbitrary	Query Evaluation: one pass
Query Plan: fixed	Query Plan: adaptive

Really a continuum …

Relation: Tuple Set or Sequence?

- Traditional relation = set/bag of tuples
- Tuple sequences have been studied:
  - Temporal databases [TCG+93]: multiple time orderings
  - Sequence databases [SLR94]: integer “position” -> tuple
- Data stream systems:
  - Ordering domains: Gigascope [CJSS03], Hancock [CFP+00]
  - Position ordering: Aurora [CCC+02], STREAM [MWA+03]

Update: Modifications or Appends?

- Traditional relational updates: arbitrary data modifications

- Append-only relations have been studied:
  - Tapestry [TGNO92]: emails and news articles
  - Chronicle data model [JMS95]: transactional data

- Data stream systems:
  - Streams-in, stream-out: Aurora, Gigascope, STREAM
  - Stream-in, relation-out: Hancock

Query: Transient or Persistent?

- Traditional relational queries: one-time, transient
- Persistent/continuous queries have been studied:
  - Tapestry [TGNO92]: content-based email, news filtering
  - OpenCQ, NiagaraCQ [LPT99, CDTW00]: monitor web sites
  - Chronicle [JMS95]: incremental view maintenance
- Data stream systems:
  - Support persistent and transient queries

Query Answer: Exact or Approximate?

- Traditional relational queries: exact answer
- Approximate query answers have been studied [BDF+97]:
  - Synopsis construction: histograms, sampling, sketches
  - Approximating query answers: using synopsis structures
- Data stream systems:
  - Approximate joins: using windows to limit scope
  - Approximate aggregates: using synopsis structures

Query Evaluation: One Pass?

- Traditional relational query evaluation: arbitrary data access

- One/few pass algorithms have been studied:
  - Limited memory selection/sorting [MP80]: *n*-pass quantiles
  - Tertiary memory databases [SS96]: reordering execution
  - Complex aggregates [CR96]: bounding number of passes

- Data stream systems:
  - Per-element processing: single pass to reduce drops
  - Block processing: multiple passes to optimize I/O cost

Query Plan: Fixed or Adaptive?

- Traditional relational query plans: optimized at beginning

- Adaptive query plans have been studied:
  - Query scrambling [AFTU96]: wide-area data access
  - Eddies [AH00]: volatile, unpredictable environments

- Data stream systems:
  - Adaptive query operators
  - Adaptive plans

Data Stream Query Processing: Anything New?

Architecture

- Resource (memory, per-tuple computation) limited, especially at low-level
- Reasonably complex, near real time, query processing

Issues

- Model: transient relations
- Relation: tuple sequence
- Data Update: appends
- Query: persistent
- Query Answer: approximate
- Query Evaluation: one pass
- Query Plan: adaptive

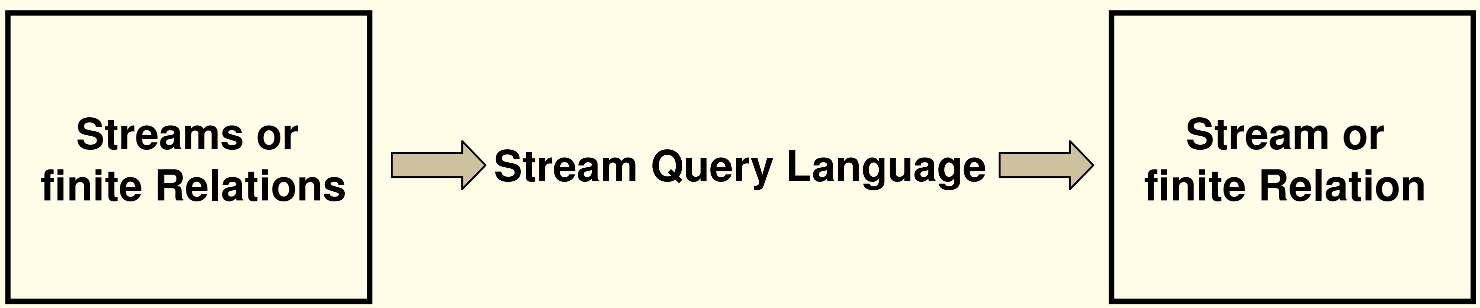
A lot of challenging problems ...

Stream Map

- Part I: Motivation
- Part II: Query processing, system issues
  - Stream query language issues
  - Query operators
  - Optimization objectives
  - Prototype systems
  - Open issues

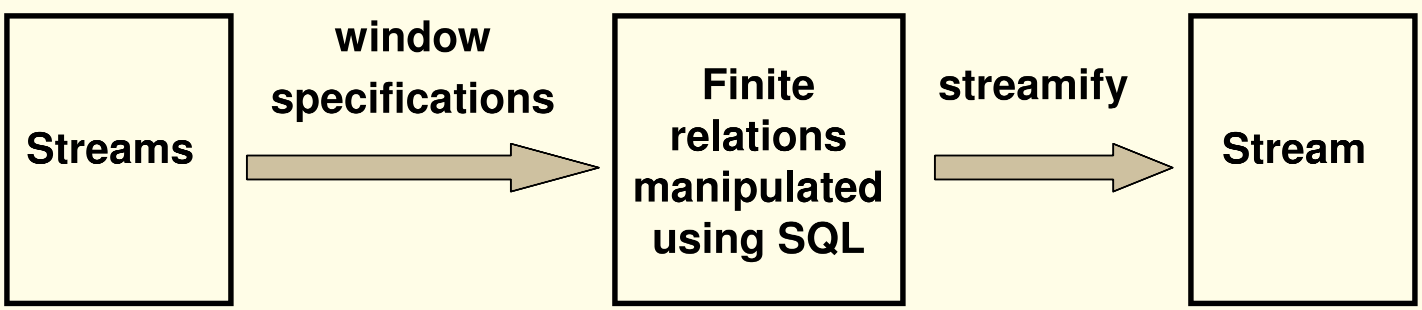
Stream Query Languages

- SQL-like proposals suitably extended for a stream environment:
  - Composable SQL operators
  - Queries reference relations or streams
  - Queries produce relations or streams
- GSQL [CJSS03]: SQL used by Gigascope
- CQL [ABW03]: SQL used by STREAM



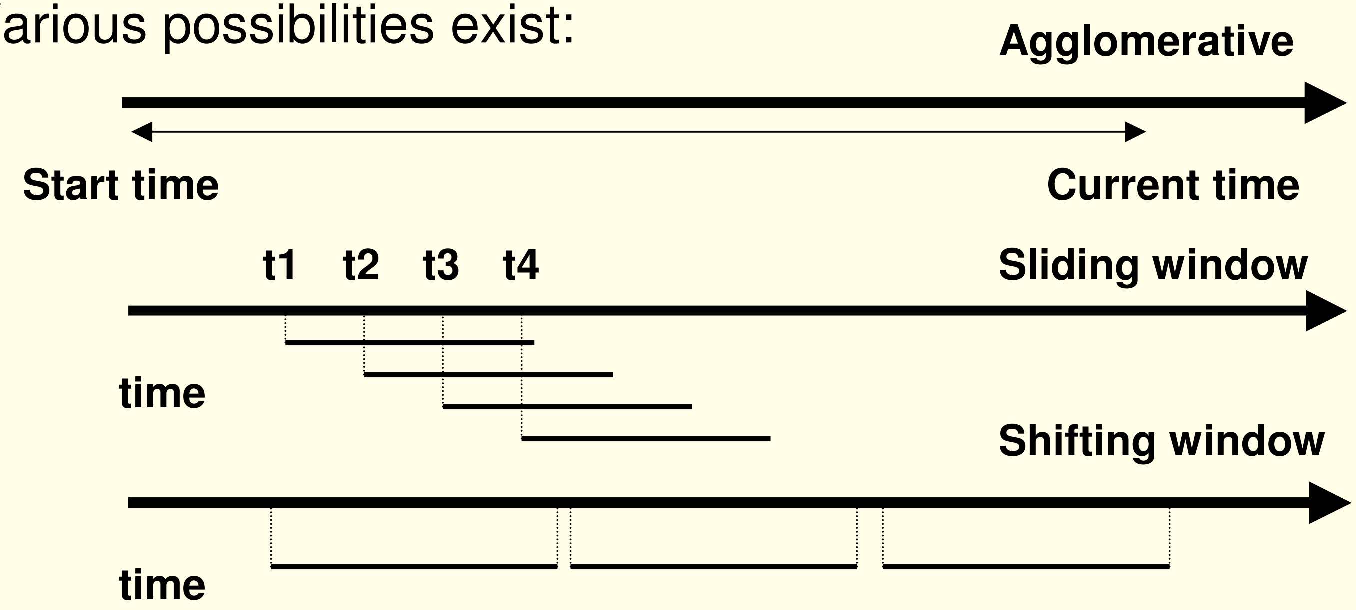
Windows

- Mechanism for extracting a finite relation from an infinite stream
- Various window proposals for restricting operator scope
  - Windows based on ordering attributes (e.g., time)
  - Windows based on tuple counts
  - Windows based on explicit markers (e.g., punctuations)
  - Variants (e.g., partitioning tuples in a window)



Ordering Attribute Based Windows

- Assumes the existence of an attribute that defines the order of stream elements/tuples (e.g., time)
- Let T be the window length (size) expressed in units of the ordering attribute (e.g., T may be a time window)
- Various possibilities exist:



Punctuation Based Windows [TMSF03]

- Application inserted “end-of-processing” markers
  - Each data item identifies “beginning-of-processing”

- Enables data item-dependent variable length windows
  - e.g., a stream of auctions
- Similar utility in query processing
  - Limit the scope of query operators relative to the stream

Selections, Projections

- Selections, (duplicate preserving) projections are straightforward
  - Local, per-element operators
  - Duplicate eliminating projection is like grouping
- Projection needs to include ordering attribute [JMS95]
  - No restriction for position ordered streams

```
Select sourceIP, time
from Traffic
where length > 512
```

Join Operators

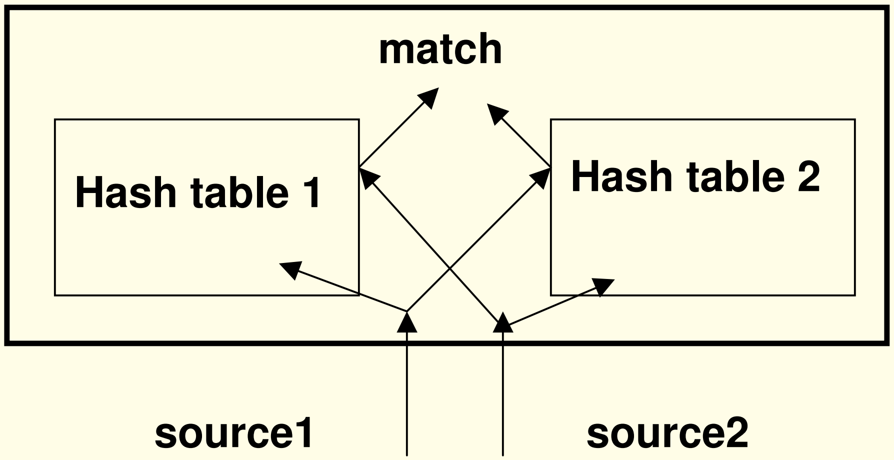
- General case of join operators problematic on streams
  - May need to join arbitrarily far apart stream tuples
  - Equijoin on stream ordering attributes is tractable [JMS95]

- Majority of work focuses on joins between streams with windows specified on each stream

Select A.sourceIP, B.sourceIP
from Traffic1 A [window T1], Traffic2 B [window T2]
where A.destIP = B.destIP

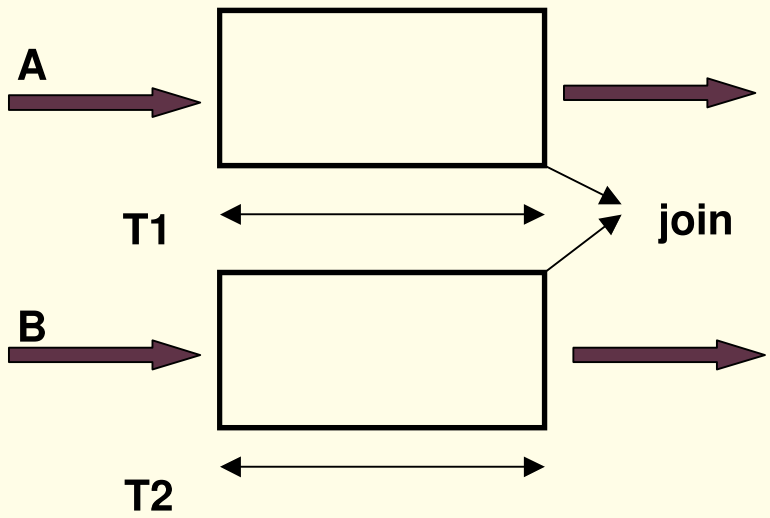
Join Operators: Background

- Symmetric Hash Joins [WA91]
  - Takes into account streaming nature of inputs



- XJoin [UF00]: extends Symmetric Hash Joins
  - overflowing inputs spilled to disk for later evaluation

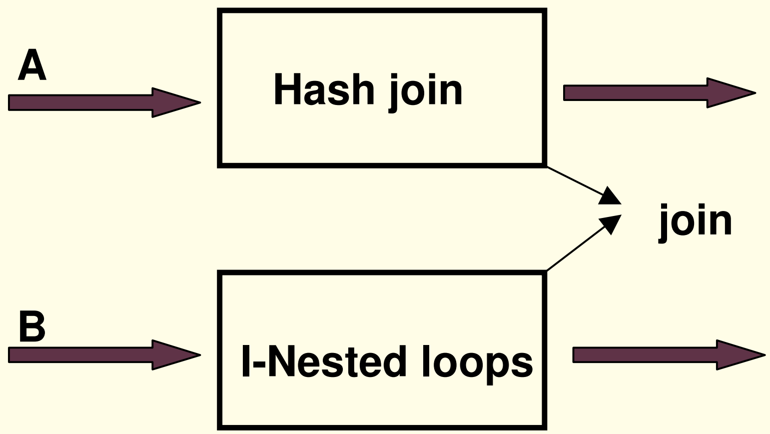
Binary Joins [KNV03]



New A tuple:

- Scan B's window for joining tuples and output result
- Insert tuple into A's window
- **Invalidate** all expired tuples in A's window

Binary Joins: Key Observations



- Asymmetric join processing has advantages if arrival rates differ
- Goal: maximize tuple output
  - limited computational capability but sufficient memory
  - limited memory but sufficient computational capability

Aggregation

- General form:
  - **select** G, F1 **from** S **where** P **group by** G **having** F2 op ϑ
  - G: grouping attributes, F1,F2: aggregate expressions
- Aggregate expressions:
  - distributive: sum, count, min, max
  - algebraic: avg
  - holistic: count-distinct, median

Aggregation in Theory

- An aggregate query result can be streamed if group by attributes include the ordering attribute [JMS95]
- A single stream aggregate query “select G,F from S where P group by G” can be executed in bounded memory if [ABB+02]:
  - every attribute in G is bounded
  - no aggregate expression in F, executed on an unbounded attribute, is holistic
- Arasu et al. [ABB+02] derive conditions for bounded memory execution of aggregate queries on multiple streams

Aggregation in Bounded Memory

- Aggregate query execution not in bounded memory:

```
select length
from Traffic [window T]
where length > 512
group by length
```

$\equiv$

```
select distinct length
from Traffic [window T]
where length > 512
```

- Aggregate query execution in bounded memory:

```
select length, count(*)
from Traffic [window T]
where length > 512 and length < 1024
group by length
```

Aggregation in Gigascope

- Grouping attributes contain window expressions restricting the scope of the group (e.g., temporally)
  - select peerid, tb, count(*) from Traffic group by time/60 as tb, f(destIP,'peerid.tbl') as peerid
  - time/60 is a minute-long shifting window
- Gigascope applies partial-aggregation on low-level data streams
  - bounded number of groups maintained at low level
  - unbounded number of groups maintainable at high level

Aggregation & Approximation

- When aggregates cannot be computed exactly in limited storage, approximation may be possible and acceptable
- Examples:
  - select G, median(A) from S group by G
  - select G, count(distinct A) from S group by G
  - select G, count(*) from S group by G having count(*) > φ |S|
- Use summary structures: samples, histograms, sketches …

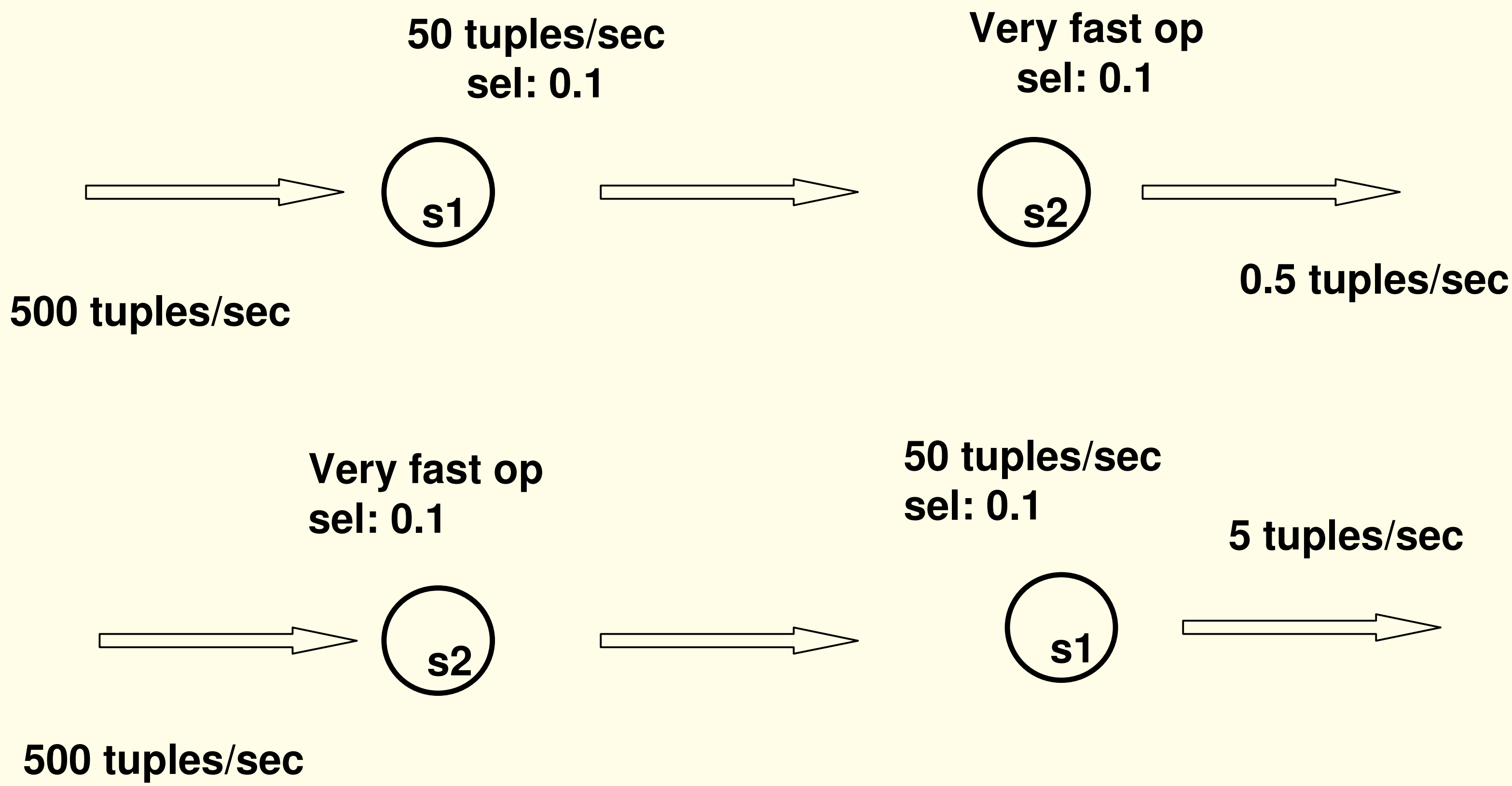
Optimization Objectives: Issues

- Traditionally table based cardinalities used in query optimization
- Problematic in a streaming environment
- Need for novel optimization objectives that are relevant when inputs consist of streaming information sources
  - Rate-based optimization
  - Memory-based optimization
  - QoS-based optimization

Rate-Based Optimization

- Rate-based optimization [VN02]:
  - Take into account the rates of the streams in the query evaluation tree during optimization
  - Rates can be known and/or estimated
- Overall objective is to maximize the tuple output rate for a query
  - Instead of seeking the least cost plan, seek the plan with the highest tuple output rate.

Rate-Based Optimization

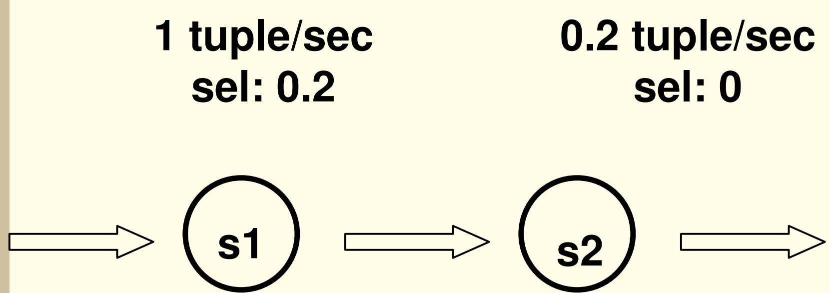


Memory-Based Optimization

- Optimize for resource (memory) consumption
- A query plan consists of interacting operators, with each tuple passing through a sequence of operators
- When streams are bursty, tuple backlog between operators may increase, affecting memory requirements
- Goal: scheduling policies that minimize resource consumption

Operator scheduling [BBDM03]

- When tuple arrival rate is uniform:
  - a simple FIFO scheduling policy suffices
  - let each tuple flow through the relevant operators



Time	Greedy	FIFO
0	1	1
1	1.2	1.2
2	1.4	2.0
3	1.6	2.2
4	1.8	3.0

Average arrival rate: 0.5 tuples/sec

FIFO: tuples processed in arrival order

Greedy: if tuple before s1 schedule it; otherwise process tuples before s2

Load Shedding

- When input stream rate exceeds system capacity a stream manager can shed load (tuples)
- Load shedding affects queries and their answers
- Introducing load shedding in a data stream manager is a challenging problem
- Random and semantic load shedding

Multi-query Processing on Streams

- In traditional multi-query optimization:
  - sharing (of expressions, results etc) among queries can lead to improved performance
- Similar issues arise when processing queries on streams:
  - sharing between select/project expressions
  - sharing between sliding window join expressions

Prototype systems

- Aurora (Brandeis, Brown, MIT) [CCC+02]
- Gigascope (AT&T) [CJSS03]
- Hancock (AT&T) [CFP+00]
- STREAM (Stanford) [MWA+03]
- Telegraph (Berkeley) [CCD+03]
- …

Aurora

- Geared towards monitoring applications (streams, triggers, imprecise data, real time requirements)
- Specified set of operators, connected in a data flow graph
- Optimization of the data flow graph
- Three query modes (continuous, ad-hoc, view)
- Aurora accepts QoS specifications and attempts to optimize QoS for the outputs produced
- Real time scheduling, introspection and load shedding

Gigascope

- Specialized stream database for network applications
- GSQL for declarative query specifications: pure stream query language (stream input/output)
- Uses ordering attributes in IP streams (timestamps and their properties) to turn blocking operators into non blocking ones
- GSQL processor is code generator.
- Query optimization uses a two level hierarchy

Hancock

- A C-based domain specific language which facilitates transactor signature extraction from transactional data streams
- Support for efficient and tunable representation of signature collections
- Support for custom scalable persistent data structures
- Elaborate statistics collection from streams

STREAM

- General purpose stream data manager
- CQL for declarative query specification
- Consider query plan generation
- Resource management:
  - operator scheduling
- Static and dynamic approximations

Telegraph

- Continuous query processing system
- Support for stream oriented operators
- Support for adaptivity in query processing
  - optimization
- Various aspects of optimized multi-query stream processing

Comparative Matrix

System	Data Stream Architecture	Data Model	Query Language	Query Answers	Query Plan
Aurora	low-level	RS-in RS-out	Operators	approximate	QoS-based, load shedding
Gigascope	two level (low, high)	S-in S-out	GSQL	exact	decomposition, avoid drops
Hancock	High-level	RS-in R-out	Procedural	exact, signatures	optimize for I/O, process blocks
STREAM	low-level	RS-in RS-out	CQL	approximate	optimize space, static analysis
Telegraph	high-level	RS-in RS-out	SQL-based	exact	adaptive plans, multi-query

Open Issues: Approximate Aggregates

- Large body of work on approximate aggregates over streams
- Issue: How can this work be used by data stream systems?
  - Engineering summary structures (sketches, samples) for low-level data stream processing
- Current status: Quantile computation is part of Gigascope, and engineered to reduce drops

Open Issues: Query Decomposition

- End-to-end three-level architecture:
  - Low-level and high-level data streams, DBMS
- Issue: How do we decompose a declarative (SQL) query?
  - Need to take resource limitations at each level into account
  - Which sub-queries are evaluated by which level?
- Current status: Gigascope does some automatic decomposition, and provides hooks for manual decomposition

Open Issues: Distributed Evaluation

- Low-level data stream processing may be highly distributed
- Issue: How do we correlate distributed data streams?
  - May not be feasible to bring all relevant data to a single site
  - Can one use techniques from distributed DBMSs?
- Current status: Some preliminary work by Gigascope, Aurora and STREAM people [BO03, CBB+03, OJW03]

Open Issues: I/O and Streaming

- High-level data stream processing can populate DBMS
- Issue: How do we process streams to minimize DBMS I/O?
  - Need to process streams in blocks, using multiple passes
  - How can multiple streams be correlated for this purpose?
- Current status: Hancock pays attention to I/O issues when computing signatures, other stream systems do not focus on I/O

Conclusions

- Data stream query processing has real applications
  - Need for sophisticated near-real time queries
  - Massive data volumes of transactions and measurements
- Wealth of challenging technical problems
  - Resource limitations exist, especially at low-level
  - Important to think of the end-to-end architecture

References

- [AF00] M. Altinel, M. J. Franklin: Efficient Filtering of XML Documents for Selective Dissemination of Information. VLDB 2000: 53-64
- [AFTU96] L. Amsaleg, M. J. Franklin, A. Tomasic, T. Urhan: Scrambling Query Plans to Cope With Unexpected Delays. PDIS 1996: 208-219
- [ABB+02] A. Arasu, B. Babcock, S. Babu, J. McAlister, J. Widom: Characterizing Memory Requirements for Queries over Continuous Data Streams. PODS 2002: 221-232
- [ABB+03] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, J. Widom: STREAM: The Stanford Stream Data Manager. IEEE Data Engineering Bulletin 26(1): 19-26 (2003)
- [ABW03] A. Arasu, S. Babu, J. Widom: An Abstract Semantics and Concrete Language for Continuous Queries Over Data Streams. DBPL 2003
- [AH00] R. Avnur, J. M. Hellerstein: Eddies: Continuously Adaptive Query Processing. SIGMOD Conference 2000: 261-272
- [BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom: Models and Issues in Data Stream Systems. PODS 2002: 1-16
- [BBDM03] B. Babcock, S. Babu, M. Datar, R. Motwani: Chain: Operator Scheduling for Memory Minimization in Data Stream Systems. SIGMOD Conference 2003: 253-264
- [BDM03] B. Babcock, M. Datar, R. Motwani: Load Shedding Techniques for Data Stream Systems. MPDS Workshop 2003
- [BO03] B. Babcock, C. Olston: Distributed Top-K Monitoring. SIGMOD Conference 2003: 28-39

References

- [BW01] S. Babu, J. Widom: Continuous Queries over Data Streams. SIGMOD Record 30(3): 109-120 (2001)
- [BR87] I. Balbin, K. Ramamohanarao: A Generalization of the Differential Approach to Recursive Query Evaluation. JLP 4(3): 259-262 (1987)
- [BDF+97] D. Barbara, W. DuMouchel, Christos Faloutsos, Peter J. Haas, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, T. Johnson, R. T. Ng, V. Poosala, K. A. Ross, K. C. Sevcik: The New Jersey Data Reduction Report. Data Engineering Bulletin 20(4): 3-45 (1997)
- [BCG+03] C. Barton, P. Charles, D. Goyal, M. Raghavachari, M. Fontoura, V. Josifovski: Streaming XPath Processing with Forward and Backward Axes. ICDE 2003
- [BFRS99] D. Bonachea, K. Fisher, A. Rogers, F. Smith: Hancock: a language for processing very large-scale data. DSL 1999: 163-176
- [BGKS03] N. Bruno, L. Gravano, N. Koudas, D. Srivastava: Navigation- vs. Index-Based XML Multi-Query Processing. ICDE 2003
- [CCC+02] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, S. B. Zdonik: Monitoring Streams - A New Class of Data Management Applications. VLDB 2002: 215-226
- [CCR+03] D. Carney, U. Cetintemel, A. Rasin, S. B. Zdonik, M. Cherniack, M. Stonebraker: Operator Scheduling in a Data Stream Manager. VLDB 2003
- [CFGR02] C. Y. Chan, P. Felber, M. N. Garofalakis, R. Rastogi: Efficient filtering of XML documents with XPath expressions. VLDB Journal 11(4): 354-379 (2002)
- [CF02] S. Chandrasekaran, M. J. Franklin: Streaming Queries over Streaming Data. VLDB 2002: 203-214

References

- [CCD+03] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, M. A. Shah: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. CIDR 2003
- [CR96] D. Chatziantoniou, K. A. Ross: Querying Multiple Features of Groups in Relational Databases. VLDB 1996: 295-306
- [CDTW00] J. Chen, D. J. DeWitt, F. Tian, Y. Wang: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. SIGMOD Conference 2000: 379-390
- [CBB+03] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, S. B. Zdonik: Scalable Distributed Stream Processing. CIDR 2003
- [CFP+00] C. Cortes, K. Fisher, D. Pregibon, A. Rogers, F. Smith: Hancock: a language for extracting signatures from data streams. KDD 2000: 9-17
- [CJSS03] C. D. Cranor, T. Johnson, O. Spatscheck, V. Shkapenyuk: Gigascope: A Stream Database for Network Applications. SIGMOD Conference 2003: 647-651
- [CJSS03a] C. D. Cranor, T. Johnson, O. Spatscheck, V. Shkapenyuk: The Gigascope Stream Database. IEEE Data Engineering Bulletin 26(1): 27-32 (2003)
- [DF03] Y. Diao, M. J. Franklin: High-Performance XML Filtering: An Overview of YFilter. IEEE Data Engineering Bulletin 26(1): 41-48 (2003)
- [DF03a] Yanlei Diao, Michael J. Franklin: Query Processing for High-Volume XML Message Brokering. VLDB 2003

References

- [FLBC02] L. Fegaras, D. Levine, S. Bose, V. Chaluvadi: Query Processing of Streamed XML Data. CIKM 2002
- [FHK+03] D. Florescu, C. Hillary, D. Kossmann, P. Lucas, F. Riccardi, T. Westmann, M. J. Carey, A. Sundararajan, G. Agrawal: A Complete and High-performance XQuery Engine for Streaming Data. VLDB 2003
- [GGR02] Minos N. Garofalakis, Johannes Gehrke, Rajeev Rastogi: Querying and Mining Data Streams: You Only Get One Look: A Tutorial. SIGMOD Conference 2002: 635
- [GO03] L. Golab, M. T. Ozsu: Processing Sliding Window Multi-Joins in Continuous Queries over Data Streams. VLDB 2003
- [GMOS03] T. J. Green, G. Miklau, M. Onizuka, D. Suciu: Processing XML Streams with Deterministic Automata. ICDT 2003
- [GS03] A. Gupta, D. Suciu: Stream Processing of XPath Queries with Predicates. SIGMOD Conference 2003
- [HFAE03] M. A. Hammad, M. J. Franklin, W. G. Aref, A. K. Elmagarmid: Scheduling for shared window joins over data streams. VLDB 2003
- [ILW00] Z. Ives, A. Y. Levy, D. Weld: Efficient evaluation of regular path expressions on streaming data. University of Washington Tech Report, 2000
- [JMS95] H. V. Jagadish, I. S. Mumick, A. Silberschatz: View Maintenance Issues for the Chronicle Data Model. PODS 1995: 113-124

References

- [KNV03] J. Kang, J. F. Naughton, S. Viglas: Evaluating window joins over unbounded streams. ICDE 2003: 37-48
- [LP02] L. V. S. Lakshmanan, S. Parthasarathy: On Efficient Matching of Streaming XML Documents and Queries. EDBT 2002: 142-160
- [LCHT02] M-L. Lee, B. C. Chua, W. Hsu, K-L. Tan: Efficient Evaluation of Multiple Queries on Streaming XML Data. CIKM 2002
- [LPT99] L. Liu, C. Pu, W. Tang: Continual Queries for Internet Scale Event-Driven Information Delivery. TKDE 11(4): 610-628 (1999)
- [MF02] S. Madden, M. J. Franklin: Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data. ICDE 2002: 555-566
- [MSHR02] S. Madden, M. A. Shah, J. M. Hellerstein, V. Raman: Continuously adaptive continuous queries over streams. SIGMOD Conference 2002: 49-60
- [MFHH03] S. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong: The Design of an Acquisitional Query Processor For Sensor Networks. SIGMOD Conference 2003: 491-502
- [MCC03] M. L. Massie, B. N. Chun, D. E. Culler: The Ganglia Distributed Monitoring System: Design, Implementation and Experience. Draft, 2003. See http://ganglia.sourceforge.net/
- [MWA+03] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. S. Manku, C. Olston, J. Rosenstein, R. Varma: Query Processing, Approximation, and Resource Management in a Data Stream Management System. CIDR 2003
- [MP80] J. I. Munro, M. Paterson: Selection and Sorting with Limited Storage. TCS 12: 315-323 (1980)

References

- [M03] S. Muthukrishnan: Data streams: algorithms and applications. SODA 2003: 413-413 http://athos.rutgers.edu/~muthu/stream-1-1.ps
- [MS03] S. Muthukrishnan, D. Srivastava: Workshop on Management and Processing of Data Streams (2003). http://www.research.att.com/conf/mpds2003/
- [OJW03] C. Olston, J. Jiang, J. Widom: Adaptive Filters for Continuous Queries over Distributed Data Streams. SIGMOD Conference 2003: 563-574
- [PC03] F. Peng, S. S. Chawathe: XPath Queries on Streaming Data. SIGMOD Conference 2003: 431-442
- [PFJ+01] J. Pereira, F. Fabret, H-A. Jacobsen, F. Llirbat, D. Shasha: WebFilter: A High-throughput XML-based Publish and Subscribe System. VLDB 2001: 723-724
- [RNC03] G. Russell, M. Neumuller, R. Connor: TypEx: A Type Based Approach to XML Stream Querying. WebDB 2003
- [SS96] S. Sarawagi, M. Stonebraker: Reordering Query Execution in Tertiary Memory Databases. VLDB 1996: 156-167
- [SV02] L. Segoufin, V. Vianu: Validating Streaming XML Documents. PODS 2002: 53-64
- [SLR94] P. Seshadri, M. Livny, R. Ramakrishnan: Sequence Query Processing. SIGMOD Conference 1994: 430-441
- [SLR95] P. Seshadri, M. Livny, R. Ramakrishnan: SEQ: A Model for Sequence Databases. ICDE 1995: 232-239
- [S96] M. Sullivan: Tribeca: A Stream Database Manager for Network Traffic Analysis. VLDB 1996: 594

References

- [TCG+93] A. U. Tansel, J. Clifford, S. K. Gadia, S. Jajodia, A. Segev, R. T. Snodgrass: Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings 1993
- [TCZ+03] N. Tatbul, U. Cetintemel, S. B. Zdonik, M. Cherniack, M. Stonebraker: Load Shedding in a Data Stream Manager. VLDB 2003
- [TGNO92] D. B. Terry, D. Goldberg, D. Nichols, B. M. Oki: Continuous Queries over Append-Only Databases. SIGMOD Conference 1992: 321-330
- [TMSF03] P. A. Tucker, D. Maier, T. Sheard, L. Fegaras: Exploiting Punctuation Semantics in Continuous Data Streams. TKDE 15(3): 555-568 (2003)
- [TMS03] P. A. Tucker, D. Maier, T. Sheard: Applying Punctuation Schemes to Queries Over Continuous Data Streams. IEEE Data Engineering Bulletin 26(1): 33-40
- [UF00] T. Urhan, M. J. Franklin: XJoin: A Reactively-Scheduled Pipelined Join Operator. IEEE Data Engineering Bulletin 23(2): 27-33 (2000)
- [VN02] S. Viglas, J. F. Naughton: Rate-based query optimization for streaming information sources. SIGMOD Conference 2002: 37-48
- [VNB03] S. Viglas, J. F. Naughton, J. Burger: Maximizing the Output Rate of Multi-Way Join Queries over Streaming Information Sources. VLDB 2003
- [WA91] A. N. Wilschut, P. M. G. Apers: Dataflow Query Execution in a Parallel Main-Memory Environment. PDIS 1991: 68-77
- [ZGTS02] D. Zhang, D. Gunopulos, V. J. Tsotras, B. Seeger: Temporal Aggregation over Data Streams Using Multiple Granularities. EDBT 2002: 646-663