Data Stream Management Systems (DSMS)

– Introduction, Concepts and Issues –

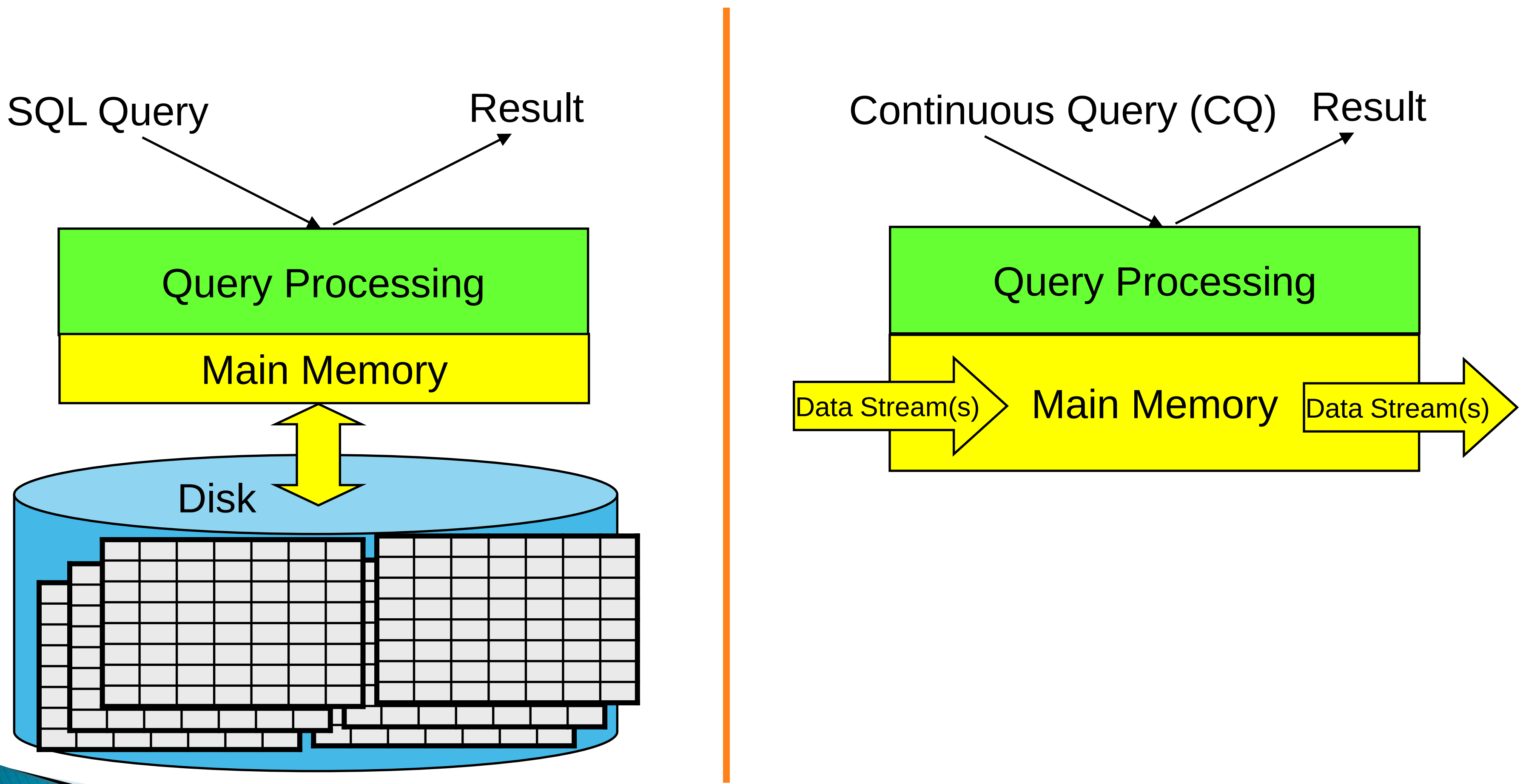
Today’s Agenda

- Introduction
  - Research field
  - DBMS vs. DSMS
  - Motivation
- Concepts and Issues
  - Requirements
  - Architecture
  - Data model
  - Queries
  - Data reduction

The DSMS Research Field

- New and active research field (~ 10 years) derived from the database community
  - Stream algorithms
  - Application and database perspective (we)
- Two syllabus articles:
  - Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, Jennifer Widom: "Models and issues in data stream systems"
  - Lukasz Golab, M. Tamer Ozsu: "Issues in data stream management”
- Future: Complex Event Processing (CEP)

DBMS vs. DSMS #1



DBMS vs. DSMS #2

- Traditional DBMS:
  - stored sets of relatively static records with no pre-defined notion of time
  - good for applications that require persistent data storage and complex querying

DSMS:

support on-line analysis of rapidly changing data streams

data stream: real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items, too large to store entirely, not ending

continuous queries

DBMS vs. DSMS #3

DBMS

- Persistent relations
  (relatively static, stored)
- One-time queries
- Random access
- "Unbounded" disk store
- Only current state matters
- No real-time services
- Relatively low update rate
- Data at any granularity
- Assume precise data
- Access plan determined by query processor, physical DB design

DSMS

Transient streams
(on-line analysis)
Continuous queries (CQs)
Sequential access
Bounded main memory
Historical data is important
Real-time requirements
Possibly multi-GB arrival rate
Data at fine granularity
Data stale/imprecise
Unpredictable/variable data arrival and characteristics

Adapted from [Motawani: PODS tutorial]

DSMS Applications

Pull-based

- Sensor Networks
  - E.g. TinyDB. See earlier lecture by Jarle Søberg
- Network Traffic Analysis
  - Real time analysis of Internet traffic. E.g., Traffic statistics and critical condition detection.
- Financial Tickers
  - On-line analysis of stock prices, discover correlations, identify trends.
- Transaction Log Analysis
  - E.g. Web click streams and telephone calls

Push-based

Data Streams – Terms

- A **data stream** is a (potentially unbounded) sequence of tuples
- Each tuple consist of a set of attributes, similar to a row in database table
- **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

Motivation #1

- Massive data sets:
  - Huge numbers of users, e.g.,
    - AT&T long-distance: ~ 300M calls/day
    - AT&T IP backbone: ~ 10B IP flows/day
  - Highly detailed measurements, e.g.,
    - NOAA: satellite-based measurements of earth geodetics
  - Huge number of measurement points, e.g.,
    - Sensor networks with huge number of sensors

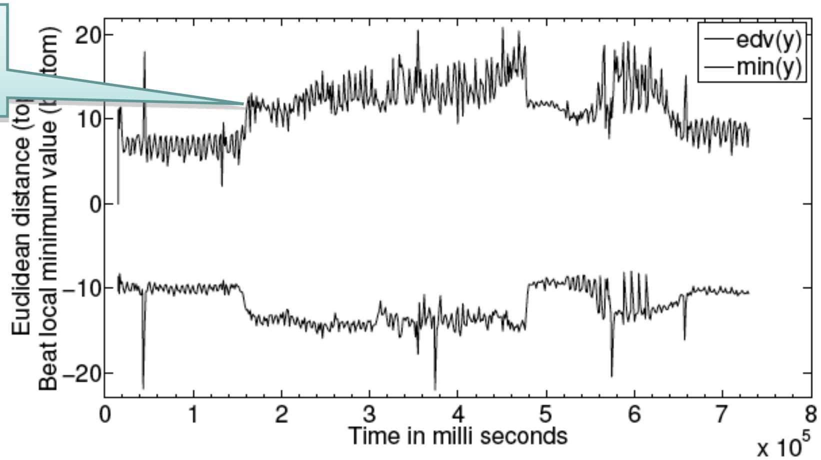
Motivation #2

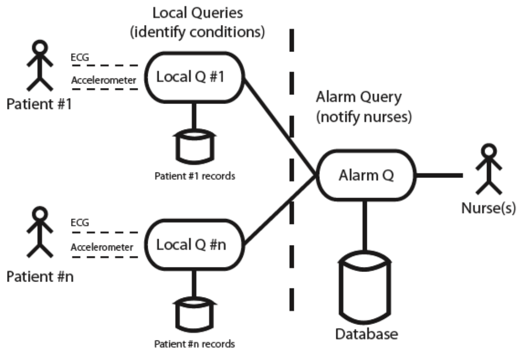
- Near real-time analysis
  - ISP: controlling service levels
  - NOAA: tornado detection using weather radar
  - Hospital: Patient monitoring
- Traditional data feeds
  - Simple queries (e.g., value lookup) needed in real-time
  - Complex queries (e.g., trend analyses) performed off-line

Motivation #3

- Stig Støa, Morten Lindeberg and Vera Goebel. *Online Analysis of Myocardial Ischemia From Medical Sensor Data Streams with Esper,* to appear 2008/2009
- Queries over sensor traces from surgical procedures on pigs performed at IVS, Rikshospitalet, running a open source java system called Esper
- Successful identification of occlusion to the heart (heart attack)

Heart attack!






```
SELECT y, timestamp
FROM Accelerometer.win:ext_timed(t, 5 s)
HAVING count(y) BETWEEN 2 AND 200
```

Motivation #4

2008
SSD seek time 0.1 ms, but capacity is small, e.g. 120 GB.

Performance of disks:

	1987	2004	Increase
CPU Performance	1 MIPS	2,000,000 MIPS	2,000,000 x
Memory Size	16 Kbytes	32 Gbytes	2,000,000 x
Memory Performance	100 usec	2 nsec	50,000 x
Disc Drive Capacity	20 Mbytes	300 Gbytes	15,000 x
Disc Drive Performance	**60 msec**	**5.3 msec**	**11 x**

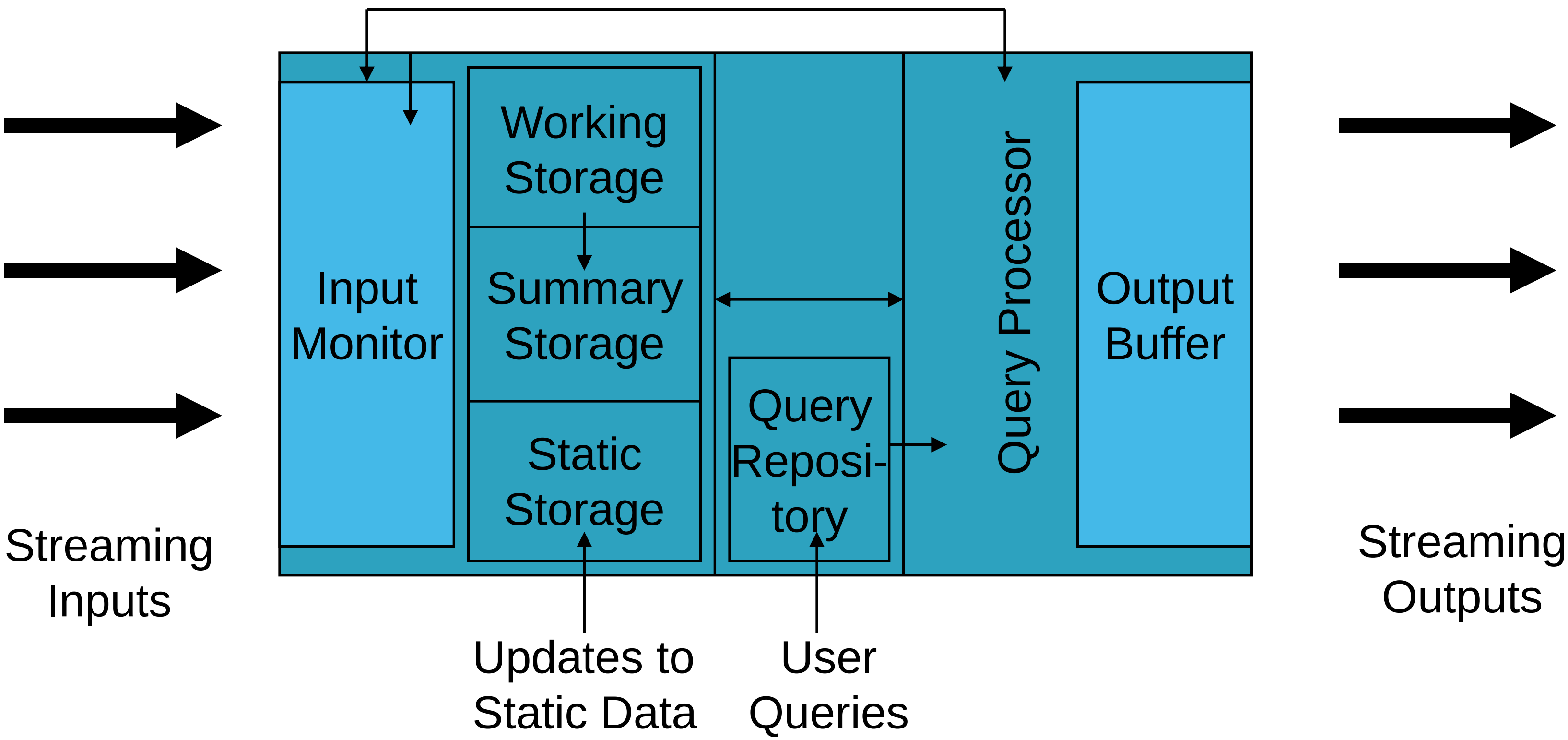
Memory I/O is much faster than disk I/O!

Source: Seagate Technology Paper: " Economies of Capacity and Speed: Choosing the most cost-effective disc drive size and RPM to meet IT requirements"

Requirements

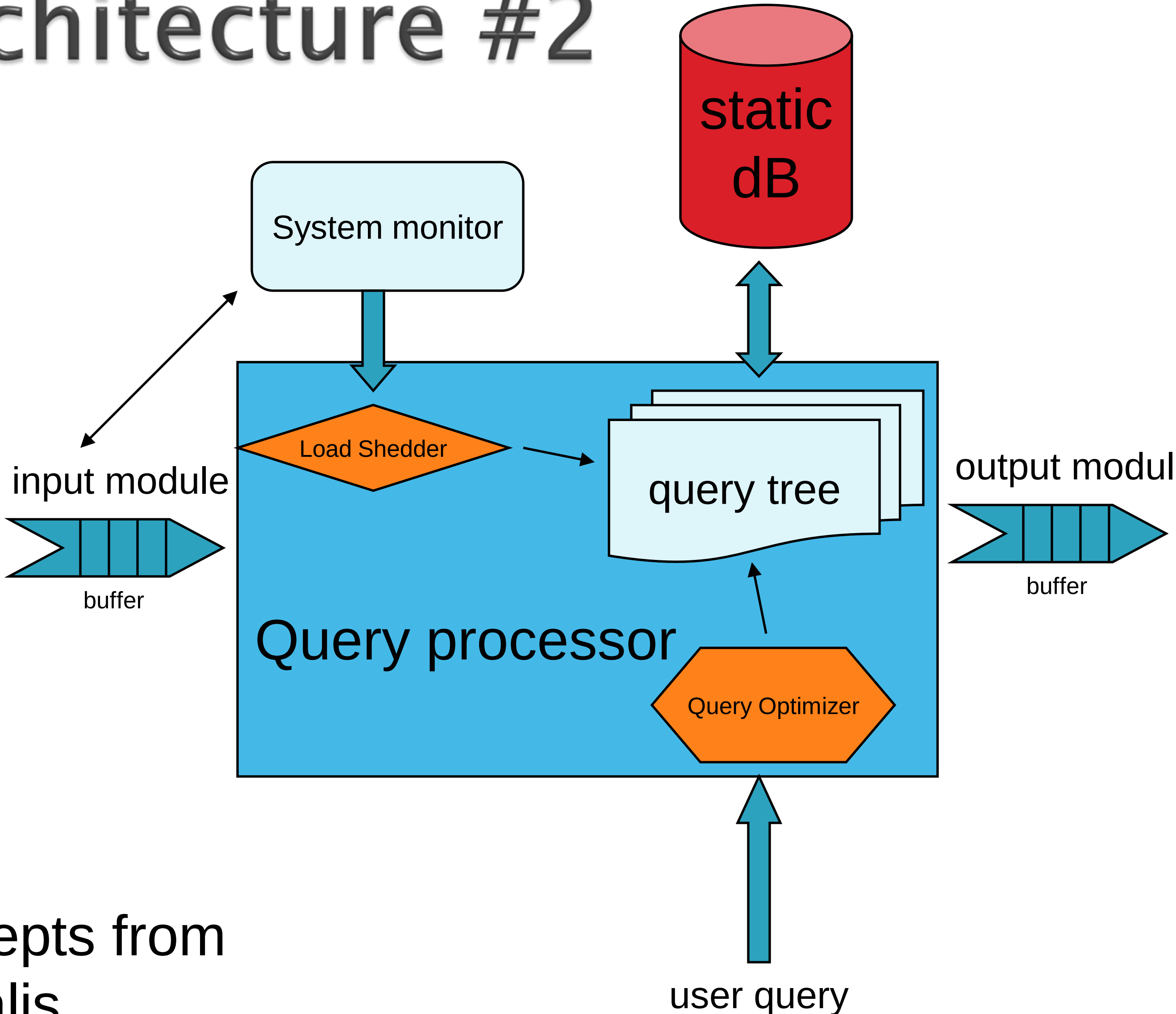
- Data model and query semantics: order- and time-based operations
  - Selection
  - Nested aggregation
  - Multiplexing and demultiplexing
  - Frequent item queries
  - Joins
  - Windowed queries
- Query processing:
  - Streaming query plans must use non-blocking operators
  - Only single-pass algorithms over data streams
- Data reduction: approximate summary structures
  - Synopses, digests => no exact answers
- Real-time reactions for monitoring applications => active mechanisms
- Long-running queries: variable system conditions
- Scalability: shared execution of many continuous queries, monitoring multiple streams

Generic DSMS Architecture



[Golab & Özsu 2003]

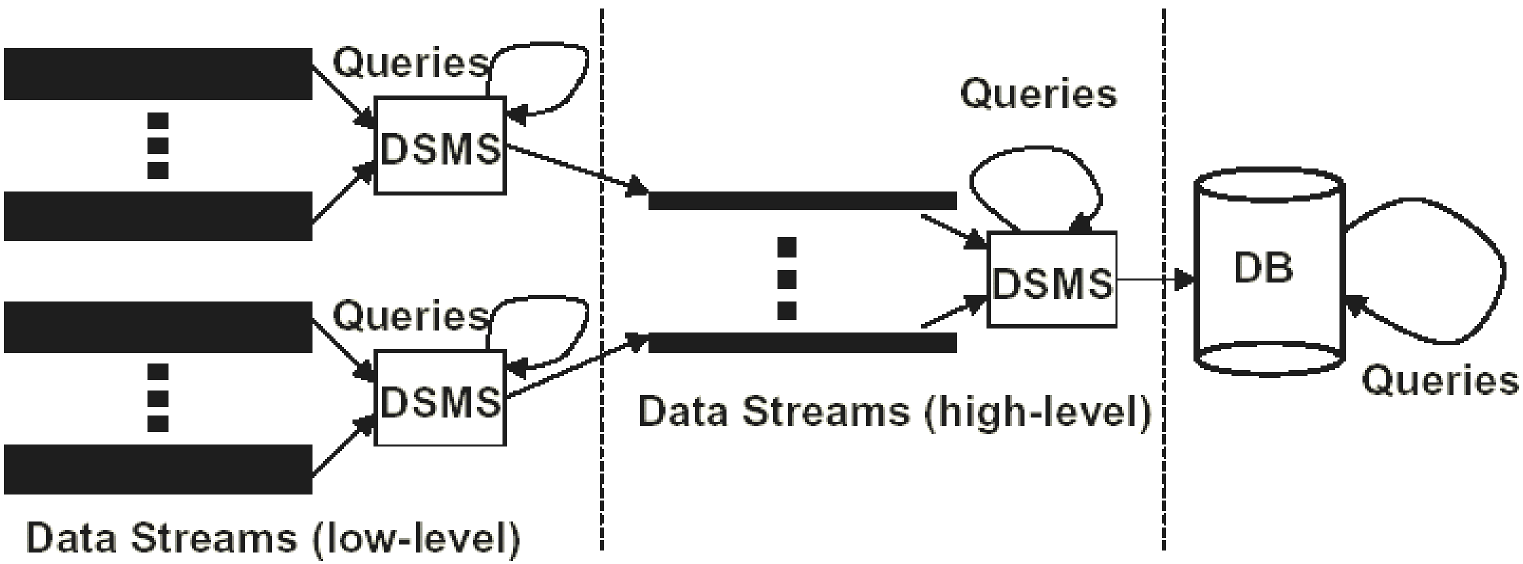
Architecture #2



Concepts from Borealis

3-Level Architecture

- Reduce tuples through several layered operations (several DSMSs)
- Store results in static DB for later analysis
- E.g., distributed DSMSs



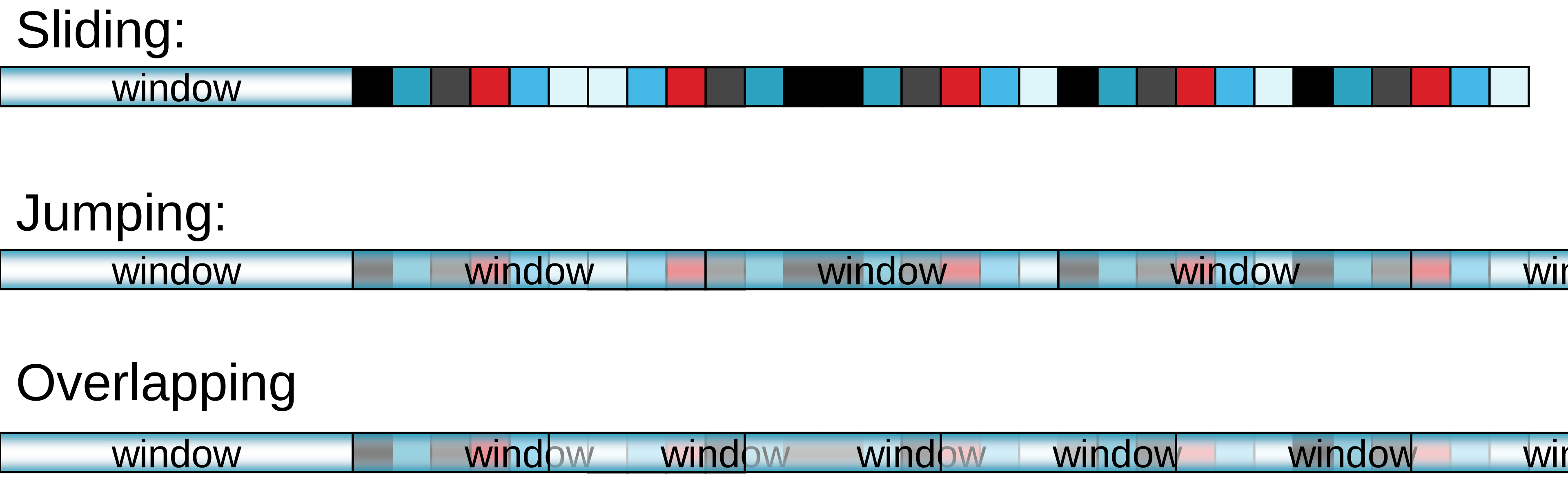
VLDB 2003 Tutorial [Koudas & Srivastava 2003]

Data Models

- Real-time data stream: sequence of items that arrive in some order and may only be seen once.
- Stream items: like relational tuples
  - Relation-based: e.g., STREAM, TelegraphCQ and Borealis
  - Object-based: e.g., COUGAR, Tribecca
- Window models
  - Direction of movements of the endpoints: fixed window, sliding window, landmark window
  - Time-based vs. Tuple-based
  - Update interval: eager (for each new arriving), lazy (batch processing), non-overlapping tumbling windows.

More on Windows

- Mechanism for extracting a finite relation from an infinite stream
- Solves blocking operator problem



(adapted from Jarle Søberg)

Timestamps

- Used for tuple ordering and by the DSMS for defining window sizes (time-based)
- Useful for the user to know when the the tuple originated
- Explicit: set by the source of data
- Implicit: set by DSMS, when it has arrived
- Ordering is an issue
- Distributed systems: no exact notion of time

Queries #1

- DBMS: one-time (transient) queries
- DSMS: continuous (persistent) queries
- Unbounded memory requirements
- Blocking operators: window techniques
- Queries referencing past data

Queries #2

- DBMS: (mostly) exact query answer
- DSMS: (mostly) approximate query answer
  - Approximate query answers have been studied:
    - sampling, synopses, sketches, wavelets, histograms, …
- Data reduction:
- Batch processing

One-pass Query Evaluation

- DBMS:
  - Arbitrary data access
  - One/few pass algorithms have been studied:
    - Limited memory selection/sorting: n-pass quantiles
    - Tertiary memory databases: reordering execution
    - Complex aggregates: bounding number of passes
- DSMS:
  - Per-element processing: single pass to reduce drops
  - Block processing: multiple passes to optimize I/O cost

Query Plan

- DBMS: fixed query plans optimized at beginning
- DSMS: adaptive query operators
  - Adaptive plans plans have been studied:
    - Query scrambling: wide-area data access
    - Eddies: volatile, unpredictable environments
    - Borealis: High Availability monitors and query distribution

Query Languages #1

- Stream query language issues (compositionality, windows)
- SQL-like proposals suitably extended for a stream environment:
  - Composable SQL operators
  - Queries reference relations or streams
  - Queries produce relations or streams
- Query operators (selection/projection, join, aggregation)
- Examples:
  - GSQL (Gigascope)
  - CQL (STREAM)
  - EPL (ESPER)

Query Languages #2

3 querying paradigms for streaming data:

1. Relation-based: SQL-like syntax and enhanced support for windows and ordering, e.g., CQL (STREAM), StreaQuel (TelegraphCQ), AQuery, GigaScope
2. Object-based: object-oriented stream modeling, classify stream elements according to type hierarchy, e.g., Tribeca, or model the sources as ADTs, e.g., COUGAR
3. Procedural: users specify the data flow, e.g., Borealis, users construct query plans via a graphical interface

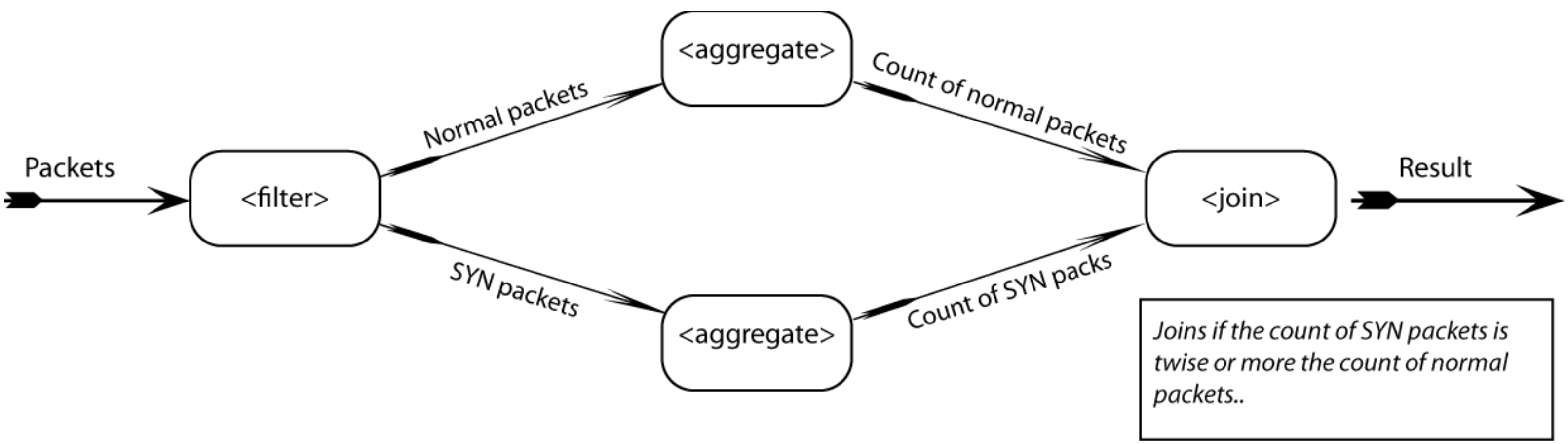
(1) and (2) are declarative query languages, currently, the relation-based paradigm is mostly used.

Sample Stream

```
Traffic  ( sourceIP -- source IP address
              sourcePort -- port number on source
           destIP -- destination IP address
           destPort -- port number on destination
           length -- length in bytes
           time -- time stamp
         );
```

Procedural Query (Borealis)

- Simple DoS (SYN Flooding) identification query



Selections and Projections

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

```
SELECT sourceIP, time
FROM Traffic
WHERE length > 512
```

Joins

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

Aggregations

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

Query Optimization

- DBMS: table based cardinalities used in query optimization
  => Problematic in a streaming environment
- Cost metrics and statistics: accuracy and reporting delay vs. memory usage, output rate, power usage
- Query optimization: query rewriting to minimize cost metric, adaptive query plans, due to changing processing time of operators, selectivity of predicates, and stream arrival rates
- Query optimization techniques
  - stream rate based
  - resource based
  - QoS based
- Continuously adaptive optimization
- Possibility that objectives cannot be met:
  - resource constraints
  - bursty arrivals under limited processing capability

Data Reduction Techniques

- Aggregation: approximations e.g., mean or median
- Load Shedding: drop random tuples
- Sampling: only consider samples from the stream (e.g., random selection). Used in sensor networks.
- Sketches: summaries of stream that occupy small amount of memory, e.g., randomized sketching
- Wavelets: hierchical decomposition
- Histograms: approximate frequency of element values in stream