UNIT III MINING DATA STREAMS **9**

Introduction To Streams Concepts – Stream Data Model and Architecture - Stream Computing - Sampling Data in a Stream – Filtering Streams – Counting Distinct Elements in a Stream – Estimating Moments – Counting Oneness in a Window – Decaying Window - Real time Analytics Platform(RTAP) Applications - Case Studies - Real Time Sentiment Analysis, Stock Market Predictions

---

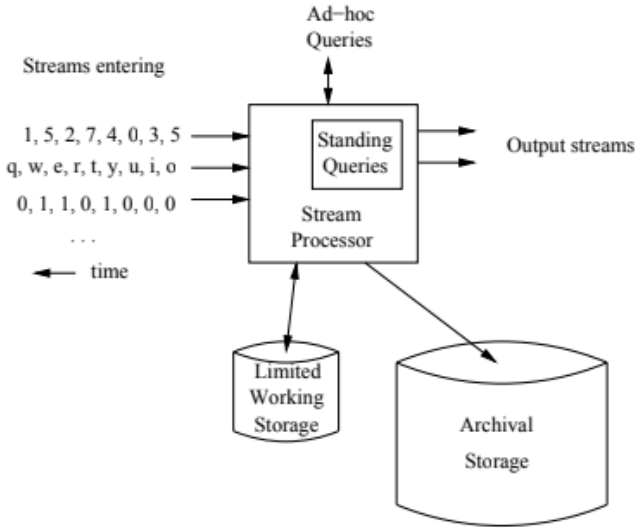
INTRODUCTION TO STREAMS CONCEPTS

Data arrives in a stream or streams, and if it is not processed immediately or stored, then it is lost forever. The data arrives so rapidly that it is not feasible to store it all in active storage (i.e., in a conventional database), and then interact with it at the time of our choosing. The algorithms for processing streams each involve summarization of the stream in some way.

The Stream Data Model

A Data-Stream-Management System

A stream processor is a kind of data-management system, with high-level organization.



Any number of streams can enter the system. Each stream can provide elements at its own schedule. They need not have the same data rates or data types, and the time between elements of one stream need not be uniform. The rate of arrival of stream elements that is not under the control of the system distinguishes stream processing from the processing of data within a database-management system. DBMS controls the rate at which data is read from the disk, and therefore never has to worry about data getting lost as it attempts to execute queries.

Streams may be archived in a large archival store, but we assume it is not possible to answer queries from the archival store. It could be examined only under special circumstances using time-consuming retrieval processes. There is also a working store, into which summaries or parts of streams may be placed, and which can be used for answering queries. The working store might be disk, or it might be main memory, depending on how fast we need to process queries. But either way, it is of sufficiently limited capacity that it cannot store all the data from all the streams.

Examples of Stream Sources

1. Sensor Data

Imagine a **temperature sensor** bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour. The data produced by this sensor is a stream of real numbers. It is not a very interesting stream, since the data rate is so low. It would not need modern technology, and the entire stream could be kept in main memory, essentially forever. If the sensor is a **GPS unit**, and report surface height instead of temperature. The surface height varies quite rapidly compared with temperature, so we might have the sensor send back a reading every tenth of a second. If it sends a 4-byte real number each time, then it produces 3.5 megabytes per day. It will still take some time to fill up the main memory. But one sensor might not be that interesting.

To learn something about ocean behavior, we might want to deploy a million sensors, each sending back a stream, at the rate of ten per second. A million sensors isn't very many; there would be one for every 150 square miles of ocean. Now we have 3.5 terabytes arriving every day, and we definitely need to think about what can be kept in working storage and what can only be archived.

2. Image Data

Satellites often send down to earth data streams consisting of many terabytes of images per day. **Surveillance cameras** produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second. London is said to have six million such cameras, each producing a stream.

3. Internet and Web Traffic

A **switching node** in the middle of the Internet receives streams of IP packets from many inputs and routes them to its outputs. Normally, the job of the switch is to transmit data and not to retain it or query it. But there is a tendency to put more capability into the switch, e.g., the ability to detect **denial-of-service attacks** or the ability to **reroute packets** based on information about congestion in the network.

Web sites receive streams of various types. For example, Google receives several hundred million search queries per day. Yahoo! accepts billions of "clicks" per day on its

various sites. Many interesting things can be learned from these streams. For example, an increase in queries like “sore throat” enables us to track the spread of viruses. A sudden increase in the click rate for a link could indicate some news connected to that page, or it could mean that the link is broken and needs to be repaired.

Stream Queries

There are two ways that queries get asked about streams. **Standing Queries** and **Ad Hoc Queries** where the standing queries are stored in a processor. These queries are, in a sense, permanently executing, and produce outputs at appropriate times.

Standing Queries Example: The stream produced by the ocean-surface-temperature sensor might have a standing query to output an alert whenever the temperature exceeds 25 degrees centigrade. This query is easily answered, since it depends only on the most recent stream element.

Alternatively, we might have a standing query that, each time a new reading arrives, produces the average of the 24 most recent readings. That query also can be answered easily, if we store the 24 most recent stream elements.

Another query we might ask is the maximum temperature ever recorded by that sensor. We can answer this query by retaining a simple summary: the maximum of all stream elements ever seen. It is not necessary to record the entire stream. When a new stream element arrives, we compare it with the stored maximum, and set the maximum to whichever is larger. We can then answer the query by producing the current value of the maximum.

Similarly, if we want the average temperature over all time, we have only to record two values: the number of readings ever sent in the stream and the sum of those readings. We can adjust these values easily each time a new reading arrives, and we can produce their quotient as the answer to the query.

Ad Hoc Queries:

The other form of query is ad-hoc, in which a question is asked once about the current state of a stream or streams. If we do not store all streams entirely, as normally we can not. If we want the facility to ask a wide variety of ad-hoc queries, a common approach is to store a **sliding window** of each stream in the working store. A sliding window can be the most recent n elements of a stream, for some n, or it can be all the elements that arrived within the last t time units, e.g., one day. Of course the stream-management system must keep the window fresh, deleting the oldest elements as new ones come in.

Ad Hoc Queries Examples: Web sites often like to report the number of unique users over the past month. If we think of each login as a stream element, we can maintain a window that is all logins in the most recent month. We must associate the arrival time

with each login, so we know when it no longer belongs to the window. If we think of the window as a relation Logins(name, time), then it is simple to get the number of unique users over the past month. The SQL query is:

```
SELECT COUNT(DISTINCT(name))
FROM Logins
WHERE time >= t;
```

Issues in Stream Processing

The constraints under which we work when dealing with streams are.

- First, streams often deliver elements very rapidly. We must process elements in real time, or we lose the opportunity to process them at all, without accessing the archival storage. Thus, it often is important that the stream-processing algorithm is executed in main memory, without access to secondary storage or with only rare accesses to secondary storage.
- When streams are “slow,” each stream by itself can be processed using a small amount of main memory, the requirements of all the streams together can easily exceed the amount of available main memory. Thus, many problems about streaming data would be easy to solve if we had enough memory, but become rather hard and require the invention of new techniques in order to execute them at a realistic rate on a machine of realistic size.
