Data Streams

João Gama
University of Porto, Portugal

Pedro Pereira Rodrigues
University of Porto, Portugal

INTRODUCTION

Nowadays, data bases are required to store massive amounts of data that are continuously inserted, and queried. Organizations use decision support systems to identify potential useful patterns in data. Data analysis is complex, interactive, and exploratory over very large volumes of historic data, eventually stored in distributed environments.

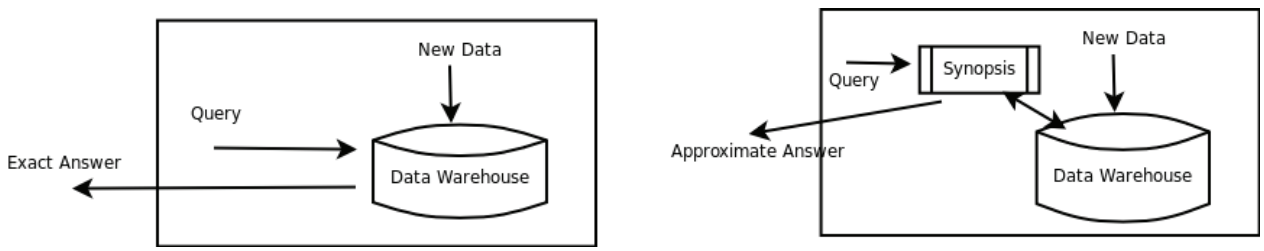
What distinguishes current data sources from earlier ones are the continuous flow of data and the automatic data feeds. We do not just have people who are entering information into a computer. Instead, we have computers entering data into each other (Muthukrishnan, 2005). Some illustrative examples of the magnitude of today data include: 3 billion telephone calls per day, 4 Giga Bytes of data from radio telescopes every night, 30 billion emails per day, 1 billion SMS, 5 Giga Bytes of Satellite Data per day, 70 billion IP Network Traffic per day. In these applications, data is modelled best not as persistent tables but rather as transient data streams. In some applications it is not feasible to load the arriving data into a traditional Data Base Management Systems (DBMS), and traditional DBMS are not designed to directly support the continuous queries required in these applications (Alon et al., 1996; Babcock et al. 2002; Cormode & Muthukrishnan, 2003). These sources of data are called *Data Streams*.

Computers play a much more active role in the current trends in decision support and data analysis. Data mining algorithms search for hypothesis, evaluate and suggest patterns. The pattern discovery process requires online ad-hoc queries, not previously defined, that are successively refined. Due to the exploratory nature of these queries, an exact answer may be not required: a user may prefer a fast but approximate answer to a exact but slow answer. Processing queries in streams require radically different type of algorithms. Range queries and selectivity estimation (the proportion of tuples that satisfy a query) are two illustrative examples where fast but approximate answers are more useful than slow and exact ones. Approximate answers are obtained from small data structures (synopsis) attached to data base that summarize information and can be updated incrementally. The general schema is presented in Figure 1.

An Illustrative Problem

A problem that clear illustrates the issues in streaming process (Datar, 2002), is the problem of finding the

Figure 1. Querying schemas: slow and exact answer vs fast but approximate answer using synopsis



maximum value (MAX) or the minimum value (MIN) in a sliding window over a sequence of numbers. When we can store in memory all the elements of the sliding window, the problem is trivial and we can find the exact solution. When the size of the sliding window is greater than the available memory, there is no exact solution. For example, suppose that the sequence is monotonically decreasing and the aggregation function is MAX. Whatever the window size, the first element in the window is always the maximum. As the sliding window moves, the exact answer requires maintaining all the elements in memory.

BACKGROUND

What makes Data Streams different from the conventional relational model? A key idea is that operating in the data stream model does not preclude the use of data in conventional stored relations. Some relevant differences (Babcock et al., 2002) include:

- The data elements in the stream arrive online.
- The system has no control over the order in which data elements arrive, either within a data stream or across data streams.
- Data streams are potentially unbound in size.
- Once an element from a data stream has been processed it is discarded or archived. It cannot be retrieved easily unless it is explicitly stored in memory, which is small relative to the size of the data streams.

In the streaming model (Muthukrishnan, 2005) the input elements $\mathbf{a_1}$, $\mathbf{a_2}$, ... arrive sequentially, item by item, and describe an underlying function **A**. Streaming models differ on how $\mathbf{a_i}$ describe **A**. Three different models are:

- *Time Series Model*: once an element $\mathbf{a_i}$ is seen, it can not be changed.
- *Turnstile Model:* each $\mathbf{a_i}$ is an update to **A(j)**.
- *Cash Register Model:* each $\mathbf{a_i}$ is an increment **to** $\mathbf{A(j) = A(j)+a_i}$.

Research Issues in Data Streams Management Systems

Data streams are unbounded in length. However, this is not the only problem. The domain of the possible values of an attribute is also very large. A typical example is the domain of all pairs of IP addresses on the Internet. It is so huge, that makes exact storage intractable, as it is impractical to store all data to execute queries that reference past data. *Iceberg queries* process relative large amount of data to find aggregated values above some specific threshold, producing results that are often very small in number (the tip of the iceberg). Some query operators are unable to produce the first tuple of the output before they have seen the entire input. They are referred as *blocking query operators* (Babcock et al., 2002). Examples of blocking query operators are aggregating operators, like SORT, SUM, COUNT, MAX, MIN, join between multiple streams, etc. In the stream setting, continuous queries using block operators are problematic. Their semantics in the stream context is an active research area.

There are two basic approaches: sliding windows (Datar & Motwani, 2007) and summaries (Aggarwal & Yu, 2007. In the sliding windows approach a *time stamp* is associated with each tuple. The time stamp defines when a specific tuple is valid (e.g. inside the sliding window) or not. Queries run over the tuples inside the window. In the case of join multiple streams the semantics of timestamps is much less clear. For example, what is the time stamp of an output tuple?

In the latter, queries are executed over a summary, a compact data-structure that captures the distribution of the data. This approach requires techniques for storing summaries or synopsis information about previously seen data. Large summaries provide more precise answers. There is a trade-off between the size of summaries and the overhead to update summaries and the ability to provide precise answers. Histograms and wavelets are the most common used summaries, and are discussed later in this chapter.

From the point of view of a Data Stream Management System several research issues emerge. Illustrative problems (Babcock et al., 2002; Datar et al., 2002) include:

- Approximate query processing techniques to evaluate queries that require unbounded amount of memory.