Chapter LXXV
Synopsis Data Structures for Representing, Querying, and Mining Data Streams

Alfredo Cuzzocrea
University of Calabria, Italy

INTRODUCTION

Data-stream query processing and mining is an emerging challenge for the database research community. This issue has recently gained the attention from the academic as well as the industrial world. Data streams are continuously flooding data produced by untraditional information sources. They are generated by a growing number of data entities, among all we recall performance measurements in network monitoring and traffic management systems, call details records in telecommunication systems, transactions in retail chains, ATM operations, log records generated by Web servers, sensor network data, RFID- (radio frequency identification) based readings, and so forth.

The most distinctive characteristics of data streams are (a) massive volumes of data (e.g., terabytes) and (b) tuples arriving rapidly; the latter make DBMS- (database management system) inspired computational models, which are typically memory bounded, inappropriate for efficiently processing such kinds of data. More specifically, as regards the model used to represent and perform computation over data streams, we can identify the following widely accepted properties characterizing data streams (Babcock, Babu, Datar, Motawani, & Widom, 2002). First, data items of the stream arrive online; typically a time stamp is associated with each, and, as a consequence, the reading of a stream can be modeled as a tuple of kind $\langle id, val, ts \rangle$ such that *id* is the identifier of the reading (e.g., the RFID tag), *val* is the value of the reading (e.g., the bar code of the product identified by the RFID tag), and *ts* is the time stamp of the data item (e.g., the time instant in which the value *val* was produced). Second, the stream processor has no control over the order in which data items arrive (and, thus, over the order in which data items can be processed). Next, the data stream is potentially unbounded. Finally, once an item of the data stream has been processed, it must be discarded so there is no possibility of it being reprocessing more times. For what regards queries, there are two distinct classes of queries that are of interest for extracting useful informa-

tion and knowledge from data streams (Babcock et al., 2002): (a) one-time queries, which are evaluated once over a point-in-time snapshot of the stream (e.g., an aggregate operator, such as SUM and COUNT, computed over the collection of items belonging to the target data stream and having a time stamp contained within a given time interval $[t_i, t_j]$, with $t_i < t_j$ and t_j smaller than the current time t), and (b) continuous queries, which produce a stream of answers over time, reflecting the evolution of the target data stream; in other words, a continuous query Q with frequency $\frac{1}{T_Q}$ produces, at each time t, an answer $A_t(Q)$ computed by considering the items of the stream whose time stamp is contained within the interval $[t\text{-}T_Q, t]$, with T_Q being an input parameter of Q. It should be noted that, while one-time queries are meaningful evolutions of conventional DBMS queries targeted at the particular application context (i.e., data-stream query processing), continuous queries represent a novel and very interesting class of queries that allow us to think of new scenarios of continuous, useful information and knowledge delivery beyond traditional client-server computational schemes, and also pose important challenges for the database and data warehouse research community. In order to efficiently support continuous query answering, new models and algorithms able to fit the novel requirements dictated by the computational model underlying such queries are needed.

Contrary to the above-described characteristics, data-stream-based applications and systems require processing queries and mining patterns over continuously evolving data in real time, thus involving the need for innovative models and algorithms for representing, querying, and mining data streams. As a consequence, there is a stringent need for designing and developing the so-called data-stream management system (DSMS), that is, a system that efficiently supports representation, indexing, query, and mining functionalities over data streams by overcoming the limitations of traditional DBMS.

Similar to querying data streams, mining data streams poses new challenges beyond the actual capabilities of data mining tools, which were designed and developed for mining massive, persistent amounts of data. Typical data mining tasks include association mining (e.g., discovering association rules), classification, and clustering; these techniques aim at finding interesting patterns, regularities, and anomalies in data. Data mining is a traditional discipline that has attracted the attention of a plethora of researchers so that in literature there exists an impressive number of data mining techniques and algorithms, also applied to breaking real-life scenarios such as fraud detection, disaster prevention, and so forth. Unfortunately, all these techniques cannot be exploited for data-stream mining issues directly; this is because they address disk-resident data sets and usually make several passes over data. Contrary to these assumptions, as said previously, streams cannot be stored persistently, and only single-pass algorithms can be applied over them. On the other hand, mining data streams in order to extract useful information and knowledge from them is an exciting line of research that will take over the scene during the coming years. Application-wise, an emerging context is that of RFID-based systems, which have a remarkable impact in leading application scenarios such as supply chain management systems, medical information systems, ambient intelligence systems, and so forth.

As a unifying view of the research challenges we address in this article, we can see the problem of efficiently querying data streams as a basic problem for the more general one concerning mining data streams. In fact, typically, stream mining techniques process data within a fixed time window over the stream (e.g., the last n items, with $n > 0$), and data and patterns to be processed are accessed via meaningful queries over such a bulk of data. In other words, we can think of the problem of querying data streams as the core layer of the general problem of mining data streams, which, indeed, is properly focused on the application layer of the target topic, that is, how to extract useful information and knowledge from streams.