In this chapter we discuss two topics that are of growing importance in database management. The topics are data stream management (Section 18.1) and cloud data management (Section 18.2). Both of these topics have been topics of considerable interest in the community in recent years. They are still evolving, but there is a possibility that they may have considerable commercial impact. Our objective in this chapter is to give a snapshot of where the field is with respect to these systems at this point, and discuss potential research directions.

18.1 Data Stream Management

The database systems that we have discussed until now consist of a set of unordered objects that are relatively static, with insertions, updates and deletions occurring less frequently than queries. They are sometimes called snapshot databases since they show a snapshot of the values of data objects at a given point in time. Queries over these systems are executed when posed and the answer reflects the current state of the database. In these systems, typically, the data are persistent and queries are transient.

However, the past few years have witnessed an emergence of applications that do not fit this data model and querying paradigm. These applications include, among others, sensor networks, network traffic analysis, financial tickers, on-line auctions, and applications that analyze transaction logs (such as web usage logs and telephone call records). In these applications, data are generated in real time, taking the form of an unbounded sequence (stream) of values. These are referred to as the data stream applications. In this section, we discuss systems that support these applications; these systems are referred to as data stream management systems (DSMS).

A fundamental assumption of the data stream model is that new data are generated continually and in fixed order, although the arrival rates may vary across applications from millions of items per second (e.g., Internet traffic monitoring) down to several items per hour (e.g., temperature and humidity readings from a weather monitoring station). The ordering of streaming data may be implicit (by arrival time at the...
processing site) or explicit (by generation time, as indicated by a timestamp appended to each data item by the source). As a result of these assumptions, DSMSs face the following novel requirements.

1. Much of the computation performed by a DSMS is push-based, or data-driven. Newly arrived stream items are continually (or periodically) pushed into the system for processing. On the other hand, a DBMS employs a mostly pull-based, or query-driven computation model, where processing is initiated when a query is posed.

2. As a consequence of the above, DSMS queries are persistent (also referred to as continuous, long-running, or standing queries) in that they are issued once, but remain active in the system for a possibly long period of time. This means that a stream of updated results must be produced as time goes on. In contrast, a DBMS deals with one-time queries (issued once and then “forgotten”), whose results are computed over the current state of the database.

3. The system conditions may not be stable during the lifetime of a persistent query. For example, the stream arrival rates may fluctuate and the query workload may change.

4. A data stream is assumed to have unbounded, or at least unknown, length. From the system’s point of view, it is infeasible to store an entire stream in a DSMS. From the user’s point of view, recently arrived data are likely to be more accurate or useful.

5. New data models, query semantics and query languages are needed for DSMSs in order to reflect the facts that streams are ordered and queries are persistent.

The applications that generate streams of data also have similarities in the type of operations that they perform. We list below a set of fundamental continuous query operations over streaming data.

- **Selection**: All streaming applications require support for complex filtering.

- **Nested aggregation**: Complex aggregates, including nested aggregates (e.g., comparing a minimum with a running average) are needed to compute trends in the data.

- **Multiplexing and demultiplexing**: Physical streams may need to be decomposed into a series of logical streams and conversely, logical streams may need to be fused into one physical stream (similar to group-by and union, respectively).

- **Frequent item queries**: These are also known as top-k or threshold queries, depending on the cut-off condition.

- **Stream mining**: Operations such as pattern matching, similarity searching, and forecasting are needed for on-line mining of streaming data.

- **Joins**: Support should be included for multi-stream joins and joins of streams with static meta-data.