Chapter 13
New Data Warehouse Technologies

Big data refers to large collections of data that may be unstructured or may grow so large and at such a high pace that it is difficult to manage them with standard database systems or analysis tools. Examples of big data include web logs, radio-frequency identification tags, sensor networks, and social networks, among other ones. It has been reported as of the time of writing this book that 7 and 10 terabytes of data are added and processed, respectively, by Twitter and Facebook every day. Approximately 80% of these data are unstructured, and 90% of them have been created in the last 2 years. Management and analysis of these massive amounts of data demand new solutions that go beyond the traditional processes or software tools. All of these have great implications on the way data warehousing practice is going to be performed in the future. For instance, big data analytics requires in many cases the data latency (the time elapsed between the moment some data are collected and the action based on such data is taken) to be dramatically reduced. Thus, near-real-time data management techniques must be developed. Also, external data sources like the semantic web may need to be queried.

Technology has started to give answers to the challenges introduced by big data: massive parallel processing, column-store database systems, and in-memory database systems (IMDBSs) are some of these answers that we will discuss in this chapter. In Sect. 13.1, we present the MapReduce framework and its most popular implementation, Apache Hadoop. In Sect. 13.2, we study Hive and Pig Latin, two high-level languages that make it easier to write the MapReduce code. We then study two architectures increasingly used in data warehousing: column-store database systems (Sect. 13.3) and IMDBSs (Sect. 13.4). To give a complete picture, in Sect. 13.5 we briefly describe several database systems that exploit the architectures above. We conclude the chapter with a study of real-time data warehousing (Sect. 13.6) and the extraction, loading, and transformation paradigm (ELT), which is challenging the traditional ETL process (Sect. 13.7). These new data
warehousing paradigms are built on the technologies that we study in the first part of the chapter.

13.1 MapReduce and Hadoop

MapReduce is a processing framework originally developed by Google to perform web search on a very large number of commodity machines. MapReduce can be implemented in many languages over many data formats. It works on the concept of divide and conquer, breaking a task into smaller chunks and processing them in parallel over a collection of identical machines (a cluster). Data in each processor are typically stored in the file system, although data in database management systems (DBMSs) are supported by several extensions, like HadoopDB. A MapReduce program consists of two phases, namely, Map and Reduce, which run in parallel in clustered commodity servers as we will see below.

Among the many MapReduce implementations, the most popular one is Hadoop, an open-source framework written in Java. It has the capability to handle structured, unstructured, or semistructured data using commodity hardware, dividing a task into parallel chunks of jobs and data. Hadoop runs on its distributed file system (HDFS) but can also read and write other file systems. Hadoop uses blocks (typically of 128MB) to store files on the file system. One block of Hadoop may consist of many blocks of the underlying operating system. Moreover, blocks can be replicated in several different nodes. For example, block1 can be stored in node1 and node3, block2 in node2 and node4, and so on. There are two main pieces of software that handle MapReduce jobs:

- The job tracker receives all the jobs from clients, schedules the Map and Reduce tasks to appropriate task trackers, monitors failing tasks, and reschedules them to different task tracker nodes. One job tracker exists in each Hadoop cluster.
- The task trackers are the modules that execute the job. There are many task trackers in a Hadoop cluster to manage parallelism in Map and Reduce tasks. They continuously send messages to the job tracker to let the latter know that they are alive and asking for a task.

The process and elements involved in a MapReduce job can be succinctly described as follows:

- The MapReduce program tells a job client to run a MapReduce job. The job client sends a message to a job tracker and gets an ID for the job.
- The job client copies the job resources (e.g., a .jar file) to the shared file system, usually HDFS.
- The job client sends a request to the job tracker to start the job. The job tracker computes the ways of splitting the data so that it can send each chunk of job to a different mapper process to maximize throughput.