Module 6

This is a single, concatenated file, suitable for printing or saving as a PDF for offline viewing. Please note that some animations or images may not work.

Module 6 Study Guide and Deliverables

Background Concepts Readings:	• *12th edition:* Coronel & Morris, chapters 12 through 14 and chapter 15 (only section 15.1) • *13th edition:* Coronel & Morris, chapter 12, chapter 13 (only sections 13.1 through 13.8), chapter 14 (only sections 14.1 through 14.5), chapter 15 (only section 15.1)
Optional SQL Readings:	• There are no SQL readings this week.
Assignments:	• Term Project Iteration 6 due **Tuesday, February 28 at 6:00 AM ET**
Course Evaluation:	Course Evaluation opens on Tuesday, February 21, at 10:00 AM ET and closes on Tuesday, February 28, at 11:59 PM ET. Please complete the course evaluation. Your feedback is important to MET, as it helps us make improvements to the program and the course for future students.
Live Classroom:	• **Tuesday, February 21 from 8:00-9:30 PM ET** • **Wednesday, February 22 from 8:00-9:30 PM ET**

Which First?

Read the book chapters before reading the online lectures.

In Section 12.8 (Distribution Transparency), the textbook indicates generically that for Local Mapping Transparency, either the "location" or "node" where the fragment is located must be specified. The

term "location" implies a physical location, and the term "node" implies a physical node, and both implications are incorrect. With Local Mapping Transparency, the end-user or programmer must *uniquely identify in full the database instance on which the fragment resides.* While each database instance in a distributed database typically resides in a different geographic location on a different physical node, specifying a physical location or even a node is not enough, as there can be multiple database instances in a single location and on a single node. Specifying a geographic location is generally not supported by any DBMS, and identifying a database instance simply by node indication is not generally supported by any DBMS.

Lecture 16 - Preview of Advanced Topics

Introduction

In this lecture, we preview a variety of interesting and advanced topics – business intelligence and data warehouses, database connectivity and web technologies, database administration and security, embedded SQL, and big data. The term “preview” is used with these topics because you are not expected to apply the knowledge you learn here on assignments or labs (though this content may appear in quizzes and the final exam), i.e. you are not required to gain applied knowledge of the material. Rather, we want to teach you the fundamentals and theory of these important topics, with a view toward having you learn more about topics of interest via other means. Boston University offers specialized classes that cover some of these topics. Business Intelligence is covered in more detail in MET CS 699 *Data Mining and Business Intelligence*, Data Warehouses are covered in more detail in MET CS 689 *Designing and Implementing a Data Warehouse*, and Database Security is covered in more detail in MET CS 674 *Database Security*. Many of these topics are also covered in more detail in MET CS 779 *Advanced Database Management*.

OLTP and OLAP Databases

Two significant types of modern databases are online transaction processing databases (OLTP) and online analytical processing databases (OLAP). Each type serves its own purpose. OLTP databases are designed to support day-to-day business operations, and OLAP databases are designed to support long-term analysis and decision making. The following table contrasts the properties of OLTP databases and OLAP databases.

OLTP Databases	OLAP Databases
All details needed for the day-to-day transactions are maintained in the database.	The level of detail of the data is only as fine-grained as needed; granularity is specifically chosen for each data item in order to support decision making and analysis.
Data redundancy is reduced or eliminated to help avoid data anomalies, especially because data may be written and read from a variety of systems in real-time.	Data redundancy is not a concern because data is loaded exclusively through a well-controlled process. Duplicating data oftentimes makes querying the data more straightforward.
The database design is intended to scale well for the reading and writing necessary for concurrent day-to-day transactions.	The performance is tuned for reading and analyzing aggregated data as efficiently as possible.
Data is considered up-to-date because data is put into the database as soon as it is available.	Data is considered time-variant, i.e., present up to a specific point in time, because data is imported into the database at specifically chosen intervals.
The structure of the data varies according to the need of each subschema in the database.	The structure of the data is homogenous throughout the database.

Test Yourself 6.1

Select all true statements regarding OLTP and OLAP databases. (Check all that apply.)

OLAP databases are designed to support day-to-day business operations.

This is false. OLTP databases are designed to support day-to-day business operations, while OLAP databases are designed to support long-term analysis and decision making.

Data is up-to-date in OLTP databases because the data is put into the database when it is available.

This is true. Data is usually up-to-date in OLTP databases, because the user often enters it as soon as it is available. There is no lag between when the data is available in the database and when the data can be used.

OLTP databases tend to use the same type of structure for all data and relationships.

This is false. Although some common design patterns are used in many OLTP databases, OLTP databases tend to use a wide variety of structures to support data and relationships.

Normalization is usually used in OLTP databases to help avoid data anomalies.

This is true. We do often use normalization in operational databases so that we can reduce or eliminate data anomalies.

OLAP databases are designed to support long-term decision making and analysis.

This is true. Long-term decision making and analysis are the reasons why OLAP database are used.

Decision Support Systems

A decision support system (DSS) is an arrangement of computerized tools used to assist managerial decision-making within an enterprise. A DSS usually requires extensive data massaging to produce information. DSS data differ greatly from operational data because they serve different purposes. Most operational data are stored in a relational database in which table structures tend to be highly normalized. Operational data storage is optimized to support daily transactions. DSS data, on the other hand, are used for bringing strategic and tactical meaning to operational data. DSS data and operational data differ in: time span, granularity, and dimensionality. Operational data cover a short time span and represent atomic transactions. DSS data cover much longer time frames. In terms of granularity, operational data represent specific transactions while DSSs represent data at different levels of aggregation. Dimensionality, however, is the most distinguishing characteristic of DSS data. In a DSS, data are related along many dimensions. This is not true in operational data.

Test Yourself 6.2

Which of the following are true regarding decision support systems? (Please check all of the following that are true.)

A decision support system database may contain information about individual transactions and also data aggregated from those transactions, such as daily sales by product.

This is true. Decision support databases typically represent data at different levels of aggregation along different dimensions. For example, a decision support database may store sales data aggregated by store by day and by product by day.

A decision support database generally stores data for the same time span as the operational database, to make it easier to transfer data from the operational database to the decision support database.

This is false. Decision support databases typically store data for decades, and often for the full history of the organization, while operational databases typically store data only for a few months.

A decision support database is typically organized according to subjects (dimensions) such as products, dates, and locations, to support analysis of data in relation to those dimensions.

This is true. This organization by dimensions is characteristic of a decision support database. In contrast an operational database is organized to support transactions and enforce integrity constraints.

Decision support databases are usually normalized to at least third normal form.

This is false. Decision support databases are usually highly denormalized, with redundant storage of aggregated data to efficiently support data analysis at different scales by different dimensions.

Definition and Preparation of Data Warehouse

The data warehouse is an integrated, subject-oriented, time-variant, nonvolatile database that provides support for decision-making. Because the data warehouse is integrated, it consolidates data from multiple and diverse sources with many formats. It helps managers better understand the company's activities across all areas. The data warehouse has the ability to answer questions arising in various functional areas within an organization. The data are focused on an organization-specific history which can be used to project expected future outcomes. In addition, the data are nonvolatile, meaning once data enter the data warehouse, the data never leaves. The data warehouse is generally optimized for data analysis and query processing. Business analysts extract the data using front-end tools.

Test Yourself 6.3

Which of the following are correct regarding data warehouses? (Please check all of the following that are correct.)

The data contained in a data warehouse may be obtained or derived from data contained in multiple other databases.

This is true. A data warehouse may obtain or derive data from multiple databases within an organization and even from outside of an organization. The data in a data warehouse is organized by subject and integrated.

To avoid unmanageable size, data from beyond a certain time period, such as five years back from present, is typically deleted from data warehouses.

This is false. A characteristic of a data warehouse is that data is never deleted. This is referred to as being *nonvolatile*.

Star Schemas and their Construction

The star schema is a data modeling technique used for mapping multidimensional decision support data into a relational database. In effect, the star schema creates the near equivalent of a multidimensional database schema from the existing relational database. The star schema was developed because existing relational modeling techniques, ER, and normalization did not yield a database structure that served advanced data analysis requirements well. Star schemas yield an easily implemented model for multidimensional data analysis, while still preserving the relational structures on which the operational database is built. The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies. Facts are numeric measurements (values) that represent a specific business aspect or activity. Dimensions are qualifying characteristics that provide additional perspectives to a given fact. Each dimension table contains attributes. Attributes are often used to search, filter, or classify facts. Attributes within dimensions can be ordered in a well-defined attribute hierarchy. The attribute hierarchy provides a top-down data organization that is used for two main purposes: aggregation and drill-down/roll-up data analysis.

We now illustrate a dimensional design for a data warehouse for a chain of grocery stores. This design is illustrated in the following animation. The design is a *snowflake* schema. A snowflake schema has the components of a star schema, and some of the dimensions have associated tables.

metcs669_W06_dimensional_object_forflv_1 video cannot be displayed here

Test Yourself 6.4

Which of the following are true regarding star schemas? (Please check all of the following that are true.)

The number of foreign keys in a fact table is equal to the number of dimensions in a star schema.

This is true. Each foreign key in the fact table corresponds to the primary key of a dimension table.

The fact table in a star schema usually has the largest number of records.

This is true. Generally, the fact table will have many times the number of records of a dimension table.

Looking at sales figures for a nation during a given time period then a state for the same time period then a city for the same time period is an example of drilling down.

This is true. Drilling down goes from a higher level of aggregation to a less aggregated data. Going to a higher level of aggregation is referred to as *rolling up*.

Fact tables have attribute hierarchies.

This is false. Dimension tables have attribute hierarchies. The attribute hierarchies provide the aggregation levels for drilling down and rolling up.

Converting a star schema design to a snowflake schema design reduces the number of tables.

This is false. Converting a star schema to a snowflake schema involves normalizing dimensions and adding tables related to the main dimension tables. This increases the number of tables.

Steps for Successful Implementation of a Data Warehouse

No single formula can describe perfect data warehouse development. Every organization is subject to a variety of constraints based on available funding, management's view of IS development, and the extent and depth of the information requirements. Remember that a data warehouse is not a static database. Instead, it is a dynamic framework for decision support that is, almost by definition, always a work in progress. Although it is easy to focus on the data warehouse database as the DSS central data repository, you must remember that the decision support infrastructure includes hardware, software, people, and procedures, as well as data. Building the perfect data warehouse is not just a matter of knowing how to create a star schema; it requires managerial skills to deal with conflict resolution, mediation, and arbitration. The technical aspects of the data warehouse must be addressed as well. The foremost technical concern in implementing a data warehouse is to provide end user decision support with advanced data analysis capabilities—at the right moment, in the right format, using the right data, and at the right cost. Developing a data warehouse is a company-wide effort that requires many resources: human, financial, and technical. Providing company-wide decision support requires a sound architecture based on a mix of people skills, technology, and managerial procedures that is often difficult to find and implement.

Test Yourself 6.5

Which of the following are true regarding data warehouses? (Please check all of the following that are true.)

Decision support IT infrastructure is a complex interplay of people, software, computers, and data, under the guidance of policies and procedures.

This is true. These components are the foundation for successful decision support systems.

People skills are required to implement organization-wide decision support.

This is true. High levels of people management skill are required to orchestrate an enterprise decision support system, as well as specialized decision support system technology skills.

Data Mining

Data mining tools automatically search the data for anomalies and possible relationships, thereby identifying problems and opportunities that have not yet been identified by the end user. In other words, data-mining tools analyze the data, uncover problems or opportunities hidden in the data relationships, form computer models based on their findings, and then use the models to predict business behavior requiring minimal end-user intervention. The current generation of data-mining tools contains many design and application variations to fit data-mining requirements. In spite of the lack of precise standards, we can conclude that data mining is subject to four phases: data preparation, data analysis and classification, knowledge acquisition, and prognosis. Because data-mining technology is still in its infancy, some of the data-mining findings might fall outside the boundaries of what business managers expect. Data mining has proved to be very helpful in finding practical relationships among data that help define customer buying patterns, improve product development and acceptance, reduce healthcare fraud, analyze stock markets, and many other areas. Most larger enterprises have one or more organizational units that specialize in data mining.

Database Connectivity and Web Technologies

Applications and users can communicate with databases using native protocols such as Oracle's Oracle Call Interface, through language-specific but portable database interfaces such as Java Database Connectivity (JDBC), or through the de facto standard portable SQL database Open Database Connectivity (ODBC). Each interface approach has advantages and disadvantages. The native interfaces can be much faster, and they often provide nonstandard features to improve performance and provide access to non-standard DBMS features. Most modern web applications use an N-tier architecture in which the database is connected to an application server which is in turn connected to the web servers which communicate with the web clients. In these tiered

architectures the application server may use a pool of database connections to support many end-user clients. In N-tier applications the business logic that is not fundamental to the database resides primarily in the application server, with operations that are fundamental to the data resident in the database as stored procedures, constraints, and triggers. In N-tier applications the web server has primary responsibility for supporting the web communication with the clients. In some applications basic data validation and other user-centered operations may be migrated all the way to the client workstation browser to improve the user interface and reduce latency. This is implemented using browser extensions such as JavaScript and Java.

The eXtensible Markup Language (XML) is increasingly important as the standard language for data interchange, with XML-based data interchange standards for most industries. Modern DBMS can store and query XML. XML is a self describing language, meaning that a user or computer application can determine how an XML document is organized by inspecting it. XML Document Type Definition (DTD) and XML Schema Definition Documents (XSD) can be used to validate that an XML document conforms to the format and constraints defined in the corresponding DTD or XSD document.

Test Yourself 6.6

Which of the following are true? (Please check all of the following that are true.)

XML (extensible markup language) is a self-describing language.

This is true. A user or computer application can determine how an XML document is organized by inspecting it.

Modern DBMS can store and query XML.

This is true. XML is increasingly important as the standard language for data interchange, with XML-based data interchange standards for most industries.

Database vendors provide an interface for connecting to their database. This is generally the only method available to communicate with the database.

This is false. Generally, there are multiple interfaces that can be used to connect to a database. The de facto standard is Microsoft's Open Database Connectivity (ODBC).

The end users of most modern web applications communicate directly with application servers.

This is false. With most modern web applications, an application server is connected to a web server which communicates with web clients.

Database Administration and Security

Data is the second most valuable asset in most enterprises, after the people in the enterprise, and it must be managed and protected carefully. Database management systems are the dominant tool for managing the data in an enterprise, and the DBMS roles include operational transactions and decision making at operational, tactical, and strategic levels. The database administrator's roles and the DBA organization vary between enterprises and within larger enterprises. Broadly, there are two DBA skill sets—those of development DBAs and those of production DBAs. Development DBAs are responsible for defining databases as parts of larger information systems, and they routinely deal with uncertainty and changing requirements. Production DBAs are responsible for the data assets of the enterprise, and they are focused on enforcing the security and other policies that involve the data and databases. Those policies may be defined by the DBA, or, in organizations that have a Data Administrator (DA), by the DA. The DBA's managerial roles frequently include defining and enforcing policies procedures, and standards; and ensuring data integrity, security, and privacy. A DBA's technical roles frequently include selecting, evaluating, and tuning DBMS, designing and implementing databases; testing databases; tuning databases; backing up and restoring database schemas; operating the DBMS; executing database utilities; and training and supporting the many users of databases, including application developers. The Data Administrator's (DAs) roles include developing the enterprise's strategic data plans and defining the policies for protecting data.

Test Yourself 6.7

Which of the following are true? (Please check all of the following that are true.)

A DBA (database administrator) at one organization may perform a function that is performed by a DA (data administrator) at another organization.

This is true. Not every organization has a DA. Also, DBA roles can vary between organizations.

A DA’s roles include testing databases, tuning databases, and executing database utilities.

This is false. These are frequent technical roles of a DBA. A DA’s roles include developing the enterprise's strategic data plans and defining the policies for protecting data.

A Production DBA is responsible for enforcing security and other policies that involve the data and database.

This is true. Production DBAs are responsible for the data assets of the enterprise.

Development DBAs are responsible for defining databases as parts of larger information systems.

This is true. They routinely deal with uncertainty and changing requirements.

Embedded SQL

Embedded SQL is a term used to refer to SQL statements that are contained within an application programming language such as COBOL, C++, ASP, Java, or ColdFusion. No matter what language you use, if it contains embedded SQL statements it is called the *host language*. Embedded SQL is still the most common approach to maintaining procedural capabilities in DBMS-based applications. Embedded SQL defines a standard syntax to identify embedded SQL code within the host language (e.g., EXEC SQL/END-EXEC). It also includes a standard syntax to identify host variables. Host variables are variables in the host language that will receive data from the database (through the embedded SQL code) and process the data in the host language. All host variables are preceded by a colon (":") when they are referenced in the embedded SQL. The final part of the standard is a communication area used to exchange status and error information between SQL and the host language. This communications area contains two variables—SQLCODE and SQLSTATE.

Embedded SQL can be static or dynamic. Static means that the executed code will always perform exactly the same operation. Dynamic embedded SQL uses variables in the host language to control the SQL statements that are executed - and different operations can be executed each time the program is run. Because static SQL is the same each time it is run, the DBMS can cache the execution plan for the SQL and not need to re-optimize the query each time. Dynamic SQL is more flexible. The applications can adjust dynamic SQL to the particular needs of each query, which can speed the execution of complex expensive queries, relative to using one static general query even when all of the power of the query is not needed. Because business intelligence applications execute a wide range of ad hoc queries based on interactive end-user business analysis requests, business intelligence applications invariably use dynamic SQL. You can study business intelligence and data mining in CS699 Data Mining and Business Intelligence.

Test Yourself 6.8

Which of the following is true regarding SQL? (Please select all of the following that are true.)

It is possible to access a database from an application written in Java by placing SQL statements within the Java code.

This is true. This is referred to as embedded SQL. The language that the SQL is embedded in is referred to as the host language. There are many possible host languages. (Does not have to be Java.)

There is a standard method to label SQL statements contained in a host language.

This is true. Embedded SQL defines a standard syntax to identify embedded SQL code within the host language (e.g., EXEC SQL/END-EXEC).

SQL statements contained in a host language can be used in assigning some value contained in the database to a variable in the host language.

This is true. There is a standard syntax to identify host variables in embedded SQL. Host variables are variables in the host language that will receive data from the database (through the embedded SQL code) and process the data in the host language.

SQL statements embedded in a host language always perform exactly the same operations.

This is false. Embedded SQL can be static or dynamic. Static embedded SQL always performs the same operation. Dynamic embedded SQL can perform different operations. The operations performed by dynamic embedded SQL are controlled through host language variables.

Big Data

You may have previously heard the popular phrase “Big Data”, and in this section we preview what Big Data is and is not. New technologies and processes have been developed to practically support Big Data, and in effect we have a different way of thinking about data. You are not only learning how we process large amounts of data, but you are learning how we treat data differently. For example, what if a user expects to search 100 petabytes of information to find what they are looking for in a few seconds? What if an organization needs to analyze thousands of newsfeeds in real-time to discover the most important news topics? What if millions of email messages need be categorized based upon their contents in a short time? Big data is capable of handling these types of problems, but traditional data systems are not.

Traditional data management involves defining manageable data structures and implementing them in a database capable of handling these structures. Commonly the ER model and the relational database are used for operational systems, and the dimensional model and a data warehouse using a relational OLAP, multidimensional OLAP, or hybrid OLAP implementation are used for decision-support systems. Even if a less common but useful database is used to solve a particular problem, for example an object-oriented database, the structure of the data, and its relationships, are well-defined and supported.

This structure strongly guides, and limits, how data is read, written, and processed in traditional systems. In particular, the strong structure lends itself to the use of centralized, logical schemas, such as relational database schemas. These schemas work well for processing granular data such as a person’s first and last name, or a person’s address, but do not work as well for processing unstructured data such as email messages, tweets, or newsfeeds. Additionally, to increase the speed of processing large amounts of data, we are able to parallelize access to these logical schemas to an extent through a variety of techniques, including processor overloading, clustering, and data segregation; however, the centralized structure limits the practical degree of parallelism that can be achieved on the system as a whole. Lastly, the fact that the structure must first be well-defined before implementation places a practical limit on how quickly a system is able to adapt to handling new sources or types of data. Thus, the strong structure provides benefits, but limits the capability of processing unstructured data, large amounts of data at extremely highs speeds, and the speed at which a system can adapt to new sources or types of data.

Big Data is often defined as handing three V’s—volume, velocity, and variety—with the assumption that one or more of these V’s exceeds a threshold that pushes the system beyond the capabilities of traditional data systems.

We can understand these three V's quite simply. Is the sheer amount of the data in storage too large to process in reasonable time with a traditional data system? Then the *volume* has pushed the system into the Big Data space. The precise size varies by organization and system, but today the threshold is often in the hundreds of terabytes, or in petabytes. Is the speed at which new data arrives too fast for a traditional data system to process? Then the *velocity* has pushed the system into the Big Data space. For example, analyzing tweets or newsfeeds in real-time may push a system into the Big Data space. Are there so many different kinds of data, structured and unstructured, that a traditional data system cannot be developed to handle all of the kinds in a reasonable time? Then the *variety* has pushed the system into the Big Data space. For example, processing a lot of unstructured text-based content, such as tweets, email messages, or scanned documents, often pushes a system into the Big Data space. The three V's give us an easy way to summarize the requirements that drive a system into the Big Data space.

It is important to note that Big Data systems are not a replacement for traditional data systems. We are not likely to phase out traditional data systems as Big Data technologies and methodologies mature. We will likely continue to use, well into the future, traditional data systems to solve the same problems we use them to solve today. We use Big Data systems to solve problems that traditional systems cannot practically solve.

Test Yourself 6.9

Select all true statements about Big Data. (Check all that apply.)

The three V's commonly associated with Big Data are volume, velocity, and variety.

This is true. Volume, velocity, and variety are indeed the three V's, and if thresholds of one or more of these are exceeded, we use a Big Data system.

Big Data systems are considered the "next generation" of data management and as such will likely replace traditional data systems soon.

This is false. Although Big Data systems solve some problems that traditional cannot solve in a reasonable amount of time, traditional data systems still serve an important purpose.

In traditional data systems, developers define and use logical data structures, commonly using the ER model and a relational database.

This is true. One of the precepts of traditional data systems is the use of well-defined structures.

The well-defined structure needed by traditional data systems naturally limits how much data these systems can process in a short period of time.

This is true. The requirement for a well-defined structure naturally limits how quickly traditional data systems can be adapted to new types of data, and how quickly large volumes of data can be processed.

MapReduce

One big player in the Big Data space is MapReduce, a model developed by Google that separates the specific logic needed to process each particular dataset from the logic needed to parallelize the processing. The parallel data processing logic is abstracted entirely away from the database developer, so that the developer need only be concerned about writing the logic to handle each problem or request. In terms of parallel processing, the model is extremely scalable and dynamic. The processing that takes place to solve a single request can be spread across hundreds or even thousands of computers, and computers can be added and taken away quickly and easily from the processing. Errors are handled gracefully in that if a particular computer becomes unresponsive or encounters an error, the work the computer was doing is restarted and given to another computer. The end result is that certain types of processing that traditionally take a long time to execute, for example, searching all known Internet content worldwide, can be executed very quickly using this new model. The separation of the parallelization logic from the data processing logic offers many useful benefits.

The mechanics of MapReduce are straightforward and are described by its name. The map function is given unprocessed data from input files and generates a list of key/value pairs. The results of the map function are fed into the reduce function, which uses those key/value pairs to generate a list of new values. Neither the map function nor the reduce function modifies the input data. Both functions only generate new lists of results. This mechanism ensures that the original data remains unchanged, to ensure that work can be restarted and rescheduled on different computers without compromising the correctness of the results.

Jeffrey Dean and Sanjay Ghemawat defined the MapReduce framework in a publication titled “MapReduce: Simplified Data Processing on Large Clusters” in their work for Google. In that publication, a simple example introduced that of a word count mechanism using MapReduce (Dean, p. 2). Imagine that the map function is given the name and contents of a text-based document, and that the output is a list of words and the number “1” to indicate that the word has appeared once. The pseudo-code for this is as follows:

For each word in the document, emit the key/value pair “word/1”.

For example, if the document contents are “A fox quickly outruns another fox before it outruns a turkey”, then this map function would produce the following list of key/value pairs:

A/1, fox/1, quickly/1, outruns/1, another/1, fox/1, before/1, it/1, outruns/1, a/1, turkey/1

The map function emits the word and the number 1 for each word, and the MapReduce framework collects all emissions from the function and puts them into a list. The MapReduce framework does not give the entire list to the reduce function all at once, but rather groups the list by key, and gives the reduce function one particular key and all values associated with that key.

For example, imagine that the reduce function coupled with this map function sums up the number of occurrences for each word to give a total count. The pseudo-code for such a reduce function is as follows:

For the given word, add up each value associated to that word.

If the MapReduce framework gives the word “fox” to the reduce function, it would also give it the list of values associated with that word, which is just **"1, 1**", because the map function emitted two entries of **fox/1.**The reduce function would then sum up 1+1 and produce the result of 2, and it would be known that the word “fox” appears 2 times in the document.

In summary, the map function emits the number 1 for each instance of “fox”, and the reduce function sums up all of the numbers associated with the word “fox”, resulting in a total word count from the document.

Let us review how the MapReduce framework is able to parallelize the operations. Imagine that the word count is needed for 1 billion documents rather than just a single document. The MapReduce framework can distribute the map function processing across hundreds or thousands of computers so that each computer processes a single document at a time. Similarly, the MapReduce framework can distribute the reduce processing across hundreds or thousands of computers so that each computer sums up the results for a single word at a time. The processing thus becomes massively parallelized without requiring the developer to write any parallelization logic.

The overall flow of the MapReduce processing begins as follows (Dean, pp. 3–8). A computer program asks the framework to process a specific set of input files. Before map and reduce processing occurs, the input files involved are partitioned into chunks, the intent being that different computers can process each chunk in parallel. One of the computers in the MapReduce system is designated as the master computer, and the rest are designated as worker computers. The master computer identifies idle worker computers and assigns them each a chunk of the input files, then those worker computers perform the map function on their assigned chunk to extract the list of key/value pairs. The key/value pairs generated by each worker computer are written to temporary files.

The overall flow of MapReduce processing continues as follows. The master computer is given the location of the temporary files produced by the map function from each worker computer. The master computer then identifies idle worker computers and assigns them a chunk of those temporary files. Each worker computer first groups all of the key/value pairs by key, thereby producing one key/value list for each key. For example, if two key value pairs are **fox/1 fox/1**, grouping it by key would generate the result **fox/1,1.** The reduce function is then invoked by each worker to process the key/value list pairs, and the results of the reduce function (one result value per key) are saved in output files. After all of the map and reduce processing has occurred for all input files, the output files containing the results for each key are given to the computer program that had requested the processing.

It is worth mentioning a few other details of MapReduce. The MapReduce framework is coupled with a distributed file system so that the computer involved in the distributed processing can efficiently access the data files. Because of the nature of the processing, MapReduce does not support the ACIDS properties of transactions—atomicity, consistency, isolation, durability, serializabiliy—and is not a transaction-based framework.

Hadoop MapReduce is a widely-used, open-source implementation of the MapReduce framework. The underlying language for Hadoop is Java and developers may implement the map and reduce functions in Java. The distributed file system is named the Hadoop Distributed File System, often denoted with the acronym HDFS. Hadoop runs well on commodity class servers on a variety of operating systems, so it is possible for organizations to purchase many servers inexpensively in order to support a high degree of parallelism.

Works Cited

Dean, J. (2004). MapReduce: Simplified Data Processing on Large Clusters. Retrieved October 8, 2014.

Test Yourself 6.10

Check all that are true about MapReduce.

When using the MapReduce framework, application developers must write some parallelization logic with each program.

This is false. A key principle of the MapReduce framework is that the parallization logic is abstracted away from the developer, who is free to focus only on the map and reduce logic.

The map function generates a list of key/value pairs from input files.

This is true. Indeed, the map function operates on the raw input file to produce a list of key/value pairs, intended to be consumed by the reduce function.

The reduce function uses the list of key/value pairs output from the map function, and then generates one value per key.

This is true. This is the way the reduce function works, so that the intermediate work performed by the map function can be brought to completion and the intended result can be achieved.

Hadoop MapReduce usually requires the use of sophisticated, high-end computers in order to achieve good performance.

This is false. One of the useful features of Hadoop MapReduce and other MapReduce implementations is that relatively inexpensive, commodity class servers can be used to achieve good performance. This feature works because each worker computer only processes one chunk at a time.