chapter 2

Database Review

After reading this chapter and completing the exercises, you will be able to:

- Review the basic components of a database
- Define a database and identify the basic components of a database management system
- Identify the different database management system models and applications
- Identify and describe the architecture of Oracle, MySQL, and Microsoft SQL Server

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Security In Your World

Lady's Inc. is a music management company based out of Miami, Florida. The company, well known for its representation of some of the hottest talent in the state, has recently fallen victim to a major security breach. A list of phone numbers, addresses, and birth names of high-profile clients was stolen from the company's database and leaked to the press. Worried about the potential impact that this might have on the organization's reputation, and concerned for future prospects, the board of directors of Lady's Inc. decided to hire Amber E., one of the most sought-after and highly-respected security consultants in the state of Florida. With more than 15 years' experience securing network and database environments, Amber is well known for her exhaustive network environment audits and iron-clad security strategies.

After a few hours of extensive testing and exploration, Amber had identified the main source of the breach. Although great measures were put into place to secure the network and the computers where the database resided, there were significant deficiencies in the security measures that were applied to the logical data structure components. Many of the fundamental strategies that are necessary for securing a database system were missing, and critical database architectural components were left completely exposed.

Based on what she found, Amber was able to make accurate inferences about the previously hired consultant. In her report to the board, she stated that although it seemed that the previous security consultant appeared to be talented, based on the obvious neglect of the fundamental database components, it was clear that he did not have much knowledge or experience working with database systems. Had the prior consultant taken the time to research the basic architecture and conduct a general review of the components of a database, this breach could have been avoided.

As discussed in the previous chapter, there are three layers of security that must be addressed in order to maintain the integrity of a database system: network security, computer security, and database security. Prior knowledge of all three of these types of architectures is a necessity in order to provide security assurance within a database environment. This chapter provides an overview of database components and explores the architecture of an Oracle, MySQL, and SQL Server database system.

Database Defined

A **database** is a collection of data stored on a computer using an application called a database management system. A **database management system** (DBMS) is an application that allows others to search stored data in order to locate specific information. The goal of a DBMS is to provide users the means to manipulate, analyze, store, and retrieve information.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

A simple example of a database is your school library. Whether you take classes online or at a traditional school, your school library stores literature and materials that you can search through to find specific information that you need. In this example, the librarian (in an on-campus library) or the Web site (for an online campus library) acts as the database management system. The DBMS provides you with the means to conduct a search and access the information that is stored within the library.

Database Structure Components

The way information is stored within the database depends on the database type. For example, addresses in an address book are normally stored alphabetically, while books in a library are stored using a sophisticated numbering system like the Dewey Decimal system. Digital database management applications have common components and use common strategies for storing and organizing information. The common components that are found within these systems are introduced below.

Tables

A **table** is one of the most basic units of storage within a database, typically representing unique and specific data objects. For example, in Lady's Inc.'s database, a table can be created to hold information about an artist or an album. Tables are composed of vertical columns and horizontal rows. A **column**, also known as a **field**, is the component of a table that maintains a general category of information with similar datatypes. A **row**, often referred to as a **record**, or a **tuple**, holds distinct units of data, and each record or row within a table is identified using unique strings of numbers or characters.

For example, referring to Figure 2-1, you can see that the column *Name* is a general category of information that holds similar data, while the second row holds a distinct unit of data that is identified by the unique number two. When referring to columns of a group, the table name is included to indicate the table in which that particular column belongs. For example, Figure 2-1 displays a typical table within a database along with its naming convention. Tables are often used in conjunction with other tables to access information, so referencing rows and columns in a table correctly is critical to the effectiveness of a database.

We refer to a column similarly to the way we indicate the paths of a file in a Windows file system. For example, if a folder named *Budget* is saved on the main drive of a Windows PC (the C: drive), and a file called *Dec.docx* is added to this folder, the full reference to the **Dec.docx** file would be C:/Budget/Dec.docx. In Figure 2-1, the ID column is referred to as the Artists.ID and the Name column is referred to as the Artists.Name.

Keys

Keys are another concept within the structure of a database, and there are several different types. A **key** is a single field or group of fields used to identify an entry in a table. In a relational database, a key is used to access or manipulate records or rows within a table of a database.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

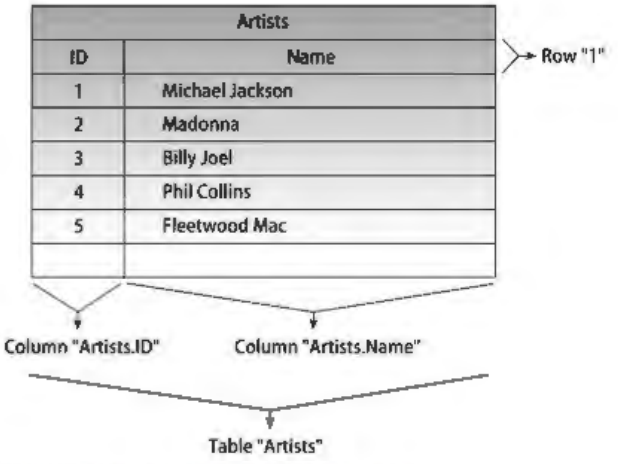
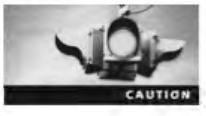


Figure 2-1 Common table found in a database
© Cengage Learning 2012

Keys are contained within the unique ID columns of a table and allow databases to be more resilient and flexible to changes. As you will discover later in this chapter, referring to a key rather than the specific table entry has a few advantages:

- Lowers the possibility of data entry errors and redundancy within a database
- Allows for less-taxing data entry changes
- Creates a more object-oriented relational environment in certain database models

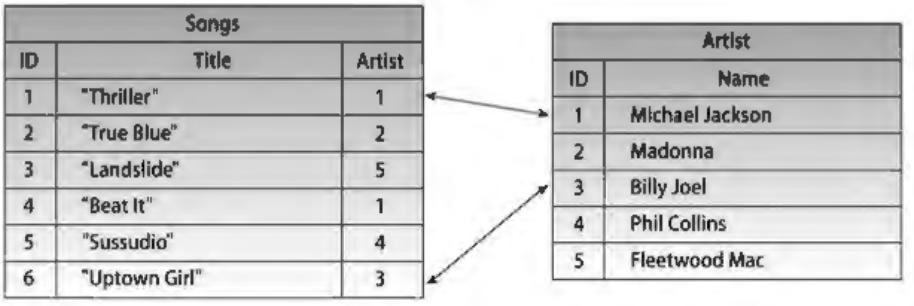
Primary Key A primary key is a field that contains a unique label by which we can identify a record or row in a table. Each table should have at least one primary key that must be unique to all other record keys within the same table, and that cannot change over time. To ensure the uniqueness of the primary key, often the key is a sequence or pattern of numbers that guarantees the absence of duplication. It is a best practice, but not necessary, to use keys that are meaningful to the data being stored. Examples of primary keys are employee ID numbers, student IDs, ISBNs, and Social Security numbers.



The use of Social Security numbers as a unique identifier is controversial. In some instances, it can be considered illegal due to inherent privacy concerns.

Foreign Key A foreign key is a field within a table that contains a label that is used to build a relationship between two tables. Often, a foreign key refers to a unique entry or primary key in a table different from the table where the foreign key resides. Figure 2-2 displays the use of a foreign key, where the numbers found within the Songs.Artist column represent a foreign key that refers to our primary key, Artists.ID from the Artists table. In

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.



Songs		
ID	Title	Artist
1	"Thriller"	1
2	"True Blue"	2
3	"Landslide"	5
4	"Beat It"	1
5	"Sussudio"	4
6	"Uptown Girl"	3

Artist	
ID	Name
1	Michael Jackson
2	Madonna
3	Billy Joel
4	Phil Collins
5	Fleetwood Mac

Figure 2-2 Use of foreign keys

© Cengage Learning 2012

most DBMSs, data can be added into a foreign key column only after the entries to where foreign keys point exist within another table. For example, we could not add a new song into Figure 2-2 and use the number six before we have identified an entry for six in our original Artists table.

Not all DBMSs allow foreign keys to be used. MySQL does not support the use of foreign keys. Although foreign keys can be included with the specifications for creating a table, it won't actually do anything. MySQL will not support foreign keys in this sense. This is a limitation in the way that MySQL implements the relational database management system (RDMS) model, yet the trade-off is that you get a significant performance boost.

Other Keys So far, two main types of keys have been discussed. Several other types of keys can be implemented, yet they do not exist as frequently. These keys are worth noting, and so it is important to understand this terminology. The usage of these keys varies, and depends on the administrator, the DBMS, and the database model within an environment:

- ***Secondary or alternative key***—A field with values that contains nonunique data and that can refer to several records at one time. For instance, Michael Jackson sings two songs from another table.
- ***Candidate key***—A field with values that meet the requirements for a primary key. This key meets the characteristics of a primary key.
- ***Composite key***—A group of two or more fields where their values can be combined to be used as a primary key.
- ***Sort or control key***—A field with values that are used to sequence data.
- ***Alternate key***—A field with values that are not chosen as a primary key, but can be used in cases where the primary key is not available. For instance, if a Social Security number is used as the primary key and an employee is hired from outside the United States, then an alternate key is used.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Queries

A **query** is a search initiated by users in an attempt to retrieve certain information from the database. A query consists of sets of variables or keywords that are formatted in a way that is defined by the programming language being used to query the DBMS in hopes of retrieving specific information. The query language to which we will refer throughout this book is known as Structured Query Language (SQL). A query written using SQL looks similar to this:

```
SELECT title FROM songs, artists, WHERE songs.artists = groups.ID
AND groups. Name= 'Madonna'
```

This query asks the database to pull all of the song titles for the songs written by Madonna. A query displays information within a **report**, which is a document that contains a formatted result of a user's query.

Database Models

As mentioned earlier, a database is a collection of data stored on a computer using a database management system. A **database model** is a representation of the way data is stored. The model for which a database is constructed also determines the way the data can be retrieved and manipulated. In this section, we will discuss four main database models: flat, relational, hierarchical, and network.

Flat Model

A **flat model** is a two-dimensional list of data entries, where all data within a field are understood to be similar, and all data within a record are understood to be related to one another. Essentially, a flat model system is similar to a sign-in sheet at a doctor's office. When a person arrives at the doctor's office, he normally writes his name in the first column, the time of arrival in the second, and the doctor he came to see in the third. All information provided in the first column is assumed to be similar (names), as well as with the second and third. All of the rows on this sheet are assumed to be related to individuals who sign in (name, time in, doctor, etc.), and each time a patient visits the doctor, he must add another row to the sheet, filling the information out again. This is certainly not a very sophisticated model, yet it is important to understand the advantages and disadvantages of this to be able to move toward the more complex relationship based on database storage models.

Consider this scenario: You work at a radio station and are asked to keep track of all of the songs that are played within an hour. You are given an Excel spreadsheet that looks like the one in Figure 2-3.

Here we have another example of a flat model database. The first column stores the same type of data (artist) and each row represents information that relates to each other row (artist's songs). The DJ plays the Billy Joel song "Uptown Girl" first, so you input this into the table. If Billy Joel is played again that hour, his name needs to be written in the table again.

Disadvantages to this system include multiple efforts and redundant data, as well as wasting time and storage space. This may not seem like much added effort, considering that only

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Artist	Song

Figure 2-3 Flat database model
© Cengage Learning 2012

around 15–20 songs are played on the radio in an hour. Imagine the wasted space of a much higher frequency and greater amount of records, for example, 20,000 entries an hour. At this level, it becomes much more of an issue.

Even though multiple efforts and redundant data are major problems, they are not the main issues with a flat database storage model. The main issues is the large margin for error that this system allows. For example, if one of Billy Joel's songs is played again later in the day and a new person is tracking the music, that person could create many mistakes. It could be something simple, such as not knowing the name of the individual who is singing the song, which could result in typing the incorrect name. Other instances include inconsistencies such as not using capital letters, placing an incorrect space in the title (e.g., "Up Town Girl," rather than "Uptown Girl"), or spelling Billy Joel incorrectly (e.g., Billie Joel). The possiblity for error is endless.

If inconsistency does exist and a query is run to find out how many Billy Joel songs were played that day, the results will not be accurate. Computers are literal when searching for identical character strings, so a query asking for B.I.L.L.Y. followed by J.O.E.L. would not return items that aren't an exact match, causing unreliable results. There are many issues with this system, and in some definitions of a DBMS, the flat model does not fit. It is important that we learn the disadvantages of this system to fully understand the advantages of the others.

Hierarchical Model

Popular in the late 1960s and well through the 1970s, hierarchical database structures are found mostly in legacy servers. The **hierarchical database structure** is a treelike storage schema that represents records and relationships through the use of tiers and parent-to-child relationships.

Similar to a family tree, relationships within a record in this schema are represented by layers. This one-to-many approach greatly minimizes redundancy (unlike the flat model) in the database and creates a more simplified view of the environment's data. The hierarchical structure restricts designers to one-to-many relationships between parent-and-child entities, whereas parents can have several children, yet children only have one parent. This model builds direct relationships only within one stem, creating one upward link to data. No direct relationships are made vertically *across* the tree, which would make searches through this storage model cumbersome and resource intensive. These restrictions have caused designers to conform their data to fit the structure, which means that data is less logically and conceptually stored than in relational database structures. Figure 2-4 illustrates the hierarchical database structure.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

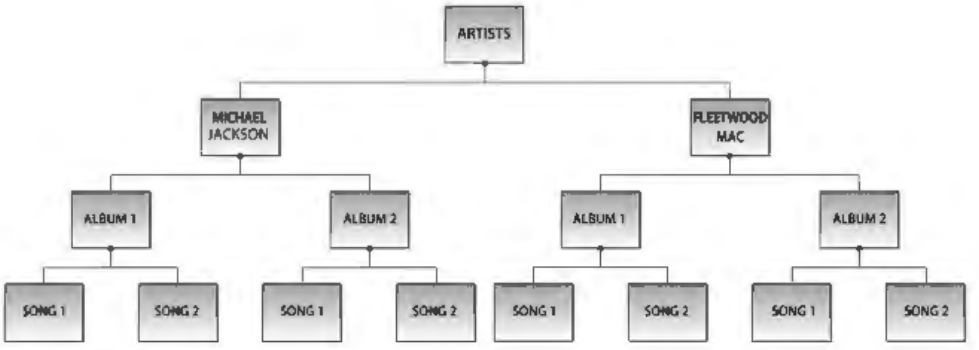


Figure 2-4 Hierarchical database model
© Cengage Learning 2012

Network Model

The network database model was developed as a solution to the one-to-many restrictive nature of the hierarchical database model. Popular during the same period as the hierarchical model, it provided an alternative for those environments that required many-to-many relationships. Much like the hierarchical data model, the **network database model** is a treelike structure that stores information in the form of a hierarchy, using tiers and parent-child-like entities to represent relationships.

Unlike the hierarchical model, the parent is referred to as a set of which the child entities are members. It is possible for child entities to be a member of more than one set (to hold more than one parent). This many-to-many relationship provided a more conceptual and logical design than the hierarchical model, and it was less resource intensive and easier to navigate. Figure 2-5 illustrates a network database model structure.

Relational Database

Developed in the 1970s, relational database storage became the flexible and conceptual solution of its predecessors. A **relational database** is a storage model in which common entities are stored within separate tables that use unique key identifiers to build relationships among these entities. There are three main concepts to understand when referring to a relational database: entities, keys, and relationships.

An **entity** is defined as a person, place, or thing stored within a table of a database for which attributes and relationships exist. An entity can be thought of as a noun. An **attribute** is a characteristic or variable that describes or further identifies an entity. An attribute can be thought of as an adjective. A **relationship** defines the association between two entities and binds them. A relationship can be thought of as a verb. Entities, attributes, and relationships together are the quintessential concepts on which relational databases are built.

Unlike a flat database model, which stores all information in one large table, a relational database is a logical and conceptual representation of information stored in several smaller

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

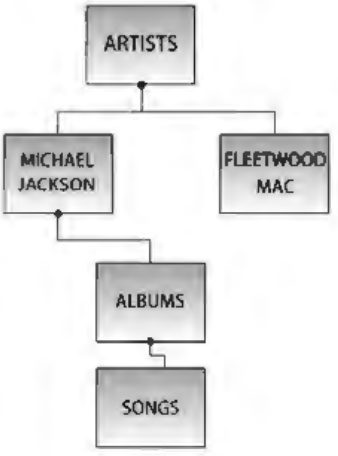


Figure 2-5 Network database model
© Cengage Learning 2012

tables. Each table is given a unique name and the database management system uses this name to retrieve and manipulate the data within the database. These unique identifiers (keys), are used to reference and build relationships with other tables or entities. A query in this model uses table names to retrieve information from the database. Figure 2-6 illustrates an example of information stored within a relational database model.

As you can see in this figure, the table Songs now includes the Album ID and the Artist ID, creating a relationship among these three tables. These relationships, as well as the object-oriented nature of relational databases, provide for a robust, flexible storage system. The issues inherent to the flat database model (e.g., redundancy, multiple efforts, and typographical errors) are greatly lowered because if designed well, data needs to be stored only once.

Object-Oriented Databases

With the advancement of Web technologies and programming languages, complex datatypes such as multimedia files have become more prominent and in demand, and so has the need to store them. Relational database management systems (RDBMSs) allow only simple datatypes such as strings and numbers to be stored within a table. Therefore, storing and retrieving a media file such as a 3-D graphic within a relational database is quite cumbersome. For this to happen, essentially all graphical information must be broken down into numbers and strings to be placed into storage, and retrieving the information requires the reconstruction of the graphic using a combination of SQL and programming code. For example, consider the process of storing a 3-D graphic of a house in an RDBMS. Each component (e.g., window, door, roof) would need to be stored separately in its own individual table and then broken down into integers that represent the position of that particular portion of the graphic as one part of the whole. To reconstruct the house, these tables must be retrieved and joined using SQL and then repositioned and drawn using different object-oriented programming code.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

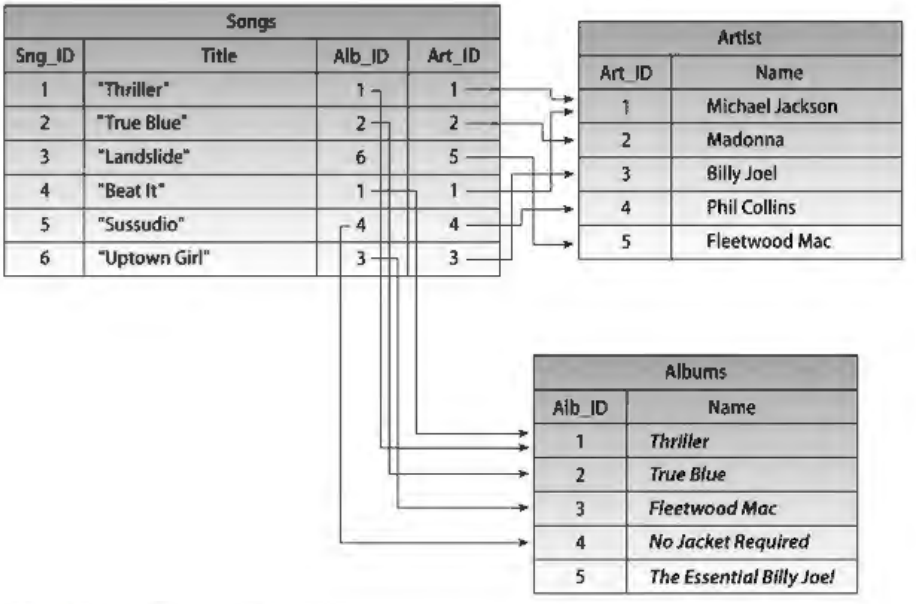


Figure 2-6 Relational database model
© Cengage Learning 2012

Object-oriented database management systems, or OODBMSs, were introduced as the means by which programmers can store objects and complex datatypes within a database. OODBMSs store objects rather than data in tables and all objects are stored as a whole, so there is no need for reconstruction. Objects are stored within the database as subclasses of a class from which subclasses inherit attributes. Object-oriented programming languages such as Java, C++, and .NET parallel those concepts and techniques of an OODBMS, as they both use the same type of objects and references to objects. Therefore, retrieval of objects from within the OODBMS is ideal for object-oriented programmers, as there is much less time wasted trying to reference application objects to their corresponding database objects.

Unfortunately, OODBMSs were introduced in 1985, well before the popularity of multimedia and object-oriented programming, so they did not catch the interest of database administrators right away. Also, OODBMS technology is quite a big leap from the relational database management systems that most administrators were used to, so the learning curve also affected the adoption of the system.

Still, OODBMSs are used in a number of specialized areas and offer great advantages for organizations that require the frequent storage and retrieval of CAD files, artificial intelligence objects, XML-compatible objects, and general multimedia (e.g., audio, video).

A special query language has been developed and standardized for OODBMSs, named object query language, or OQL. OQL is an alternative to object-oriented languages for retrieving and storing objects in the object-oriented database management system.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Object-Relational Database

Object-relational database management systems, or ORDBMSs introduced in the 1990s as a middle ground between relational and object-oriented database management systems. An ORDBMS is essentially a relational database management system with an expanded group of datatypes to accommodate object classes and inheritance. General programming languages and SQL can be used to retrieve information, and custom datatypes can be developed. This might seem like the best of both worlds for programmers and database administrators alike, yet at the time of this writing, relational database management systems are still the prominent choice in the market. Only the future can tell the potential for an ORDBMS.

Most database administrators and programmers use a relational database management system in combination with an object-relational application as a way to obtain the advantages of an ORDBMS without having to obtain one.

Relationships

Relationships define the association between entities and bind them together. The two entities in a relationship are often called parent and child entities. The types of relationships implemented between two entities are one-to-one, one-to-many, or many-to-many. Figure 2-7 displays these relationship types.

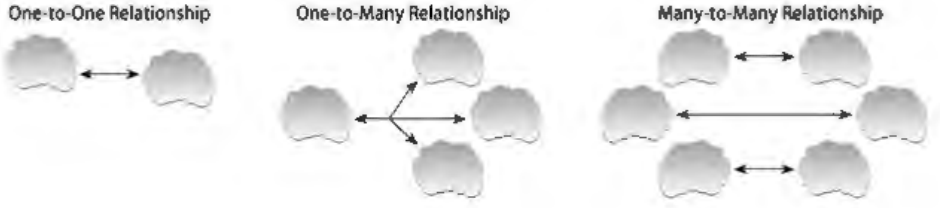


Figure 2-7 Entity relationship types
© Cengage Learning 2012

- *One-to-one relationship*—Often expressed as 1:1. This is the most simplistic relationship that two entities can have. This relationship has only one entity in both directions. These relationships rarely occur within a relational database. They are found only in unique situations. Too many one-to-one relationships found within a relational database schema usually indicate poor design choices. An example of a 1:1 relationship is an instructor who teaches a specific section of a course. Anthony Joseph teaches section two of the database security course, and the database security course contains only one instructor. Although these are rare, there are some security policies created in order to ensure that 1:1 relationships keep private information separate.
- *One-to-many relationship*—Often expressed as 1:N (N represents an indefinite number). This is the ideal type of relationship within a relational database schema. This relationship involves one entity who has a sole relationship with an entity that has a relationship

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

with several other entities as well. For instance, an instructor may belong to only one department within the university (e.g., Information Security), yet that department has many instructors that are members.

- *Many-to-many relationship*—Often expressed as M:N. This is a type of relationship that refers to an entity that has one or many partnerships with another entity that also has one or many other partnerships. For example, an instructor may be a member of several school committees, and school committees may have several members. This is used effectively within databases as long as the entities do not have many-to-many relationships with one another. A situation in which a many-to-many relationship causes issues is one where there are duplicate tables within a database to represent a many-to-many relationship. This type of redundancy is unnecessary and an inefficient way to store data. It can make querying the database difficult and confusing for both users and administrators.

Database Types

Databases can be used for many purposes and are capable of storing many types of data. The type of database that is used is determined by the data that will be housed in it.

OLTP

An **online transaction processing (OLTP) database** is a database that is created for real-time storage and manipulation of data within an organization. An OLTP was created to be used in an active environment and is optimized to serve thousands of users simultaneously. Typically, an OLTP stores data that results from large volumes of short transactions, usually from a point of sale or data entry application. A **point of sales (POS) system** is a system that is meant to handle cash register or sales transactions.

OLAP/DSS

An **online analytical processing (OLAP)**, or **decision support system (DSS)**, is a database that stores large volumes of historical data for report generating and analyzing. DSS and OLAP systems typically retrieve their data from an OLTP. The data stored is analyzed within a business environment as a way to improve productivity or meet a specific need. These systems are also referred to as data warehouses, or repositories, for an organization's digitally stored data. Designed to store large amounts of data and draw reports for analytical intentions, these types of systems do not cater to real-time, frequent, large-volume, or short transactions.

Database Management Systems

As defined earlier in this chapter, a database management system is an application that provides users with the means to manipulate, analyze, and query data. Almost all DBMSs in existence today are developed to be used with relational databases. These applications are known as Relational Database Management Systems (RDBMSs).

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

There are several relational database management systems in existence today. For this book, we will be focusing on the three most prominent and fastest-growing RDBMSs: Oracle, Microsoft SQL, and MySQL. The following section will explore the major components of their architectures.

Oracle

Oracle is an RDBMS developed by Oracle Corporation in the late 1970s. Holding 52% of the DBMS market in 2009, it remains one of the most popular database servers. Oracle is well known for being portable, its ability to run on almost any operating system, and its dominant role in providing solutions on which businesses rely to achieve their core competencies. Oracle databases offer a number of storage solutions, and can be found maintaining business-critical data such as human resources, billing, and financial records throughout the globe. The most current version and the version on which this book will focus is Oracle 11g.

MySQL

MySQL is an RDBMS developed by Sun Microsystems. Showing tremendous growth each year and well known for its speed, MySQL is the most popular open-source database server today. The term **open source** refers to software that has been written to be distributed for use and download free of charge. Besides cost, open-source applications provide the advantage of customization.

Anyone is able to view and study the code and can contribute to it by providing additional customization or features that fit several environments. A large contribution has already been made to MySQL, so several variations of tools have already been created that support almost any network configuration. MySQL is platform independent, and like Oracle, it is implemented to solve business needs, both large and small.

Microsoft SQL

Microsoft SQL is often referred to simply as SQL Server. SQL Server is an RDBMS developed by Microsoft to provide a fast, secure, and scalable data access platform. SQL Server's primary query languages are T-SQL and ANSI SQL. The scalability of the SQL server architecture is its most attractive feature, as it was developed to run equally in a wide variety of environments. From a database running on a personal computer servicing only one user, to a database stored on a cluster of servers servicing several thousands of users, SQL Server can meet the needs of any Windows environment.

Database Similarities

Two main areas of focus that exist in all database management systems are read consistency and query management. Obtaining an understanding of read consistency and query management is imperative in order to identify the different ways these applications address these items.

Read Consistency

Read consistency refers to the accuracy and reliability of data within a database and is dependent on a database's ability to process and commit transactions in a timely manner, as well as control concurrency effectively by applying the following locking mechanisms:

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

- *Transactions*—Initiated by a user via a request to change or obtain information from a database. A user initiates a request by querying the database using an API or a programming language such as SQL. A **transaction** is the group of statements or queries processed by the database to execute a user's request to update or change the database. In order to meet the standards of the database industry, transactions must operate as a single unit, maintain consistency of the database, work in isolation, and be durable enough to persist even after system failures.
- *Concurrency*—One of the greatest concerns within a multiuser environment is how to handle simultaneous access to data and resources in such a way that integrity and reliability of that data is maintained. For example, if two users are accessing data within the same table of a database at the same time and both are making changes to this table, problems will occur. One user's change will override the other's, making the database inaccurate for at least one of the two users. Concurrency is the term used to describe this concern. **Concurrency** is the simultaneous access of resources and data. Oracle, MySQL, and SQL Server handle concurrency in different ways, yet each application uses locks to maintain system integrity.
- *Locks*—A **lock** is a mechanism within a DBMS that controls concurrency by allowing users to take hold of the data until changes being made are completed or committed. If two users attempt to commit changes on the same data at the same time, a deadlock will occur. A **deadlock** is a situation when two transactions cannot proceed because each user has data that the other needs.
- *Commit*—A transaction is committed after it is processed by the database. To **commit** a change means to make it permanent and visible to other users.
- *Undo*—Transaction commits can be undone. To undo a transaction commit is to roll back changes made through users' API or SQL transactions. The undone transactions are reverted to the state they were in before the user committed the most recent transaction.

Query Management

Query management refers to the steps taken by a database management application to process a user query. Query management often involves retrieving, parsing, and optimizing user SQL requests. Query processing can be conducted on queries within a local table, within a partitioned table, or on a distributed server.

Each database application has a different process for parsing queries, yet in general terms, **parsing** involves the act of analyzing the construction of a query for correct syntax and semantics. During the parsing process, the database applications also identify the tables and data the user is requesting, checking that these items exist as well.

Optimization is the process of locating the quickest and most efficient way to retrieve the data requested by the user. Modern database applications often optimize using the variables of cost and speed to determine the best plan for executing a SQL statement. Cost is the amount of resources needed to retrieve data in a certain way.

Queries can be processed individually, where one process parses and optimizes one query at a time, in batches, where one process parses and optimizes several queries at one time; or in parallel (known as **parallel processing**), when more than one server process processes one query at the same time.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Oracle Architecture

Before you can install, use, or secure Oracle, you should have a basic understanding of the architecture of Oracle and how it conceptually works. This section will begin by identifying the two main components of the Oracle server—the instance and the database—and then will review the physical structure, memory structure, logical structure, and processes of the Oracle architecture. Oracle architecture is illustrated in Figure 2-8.

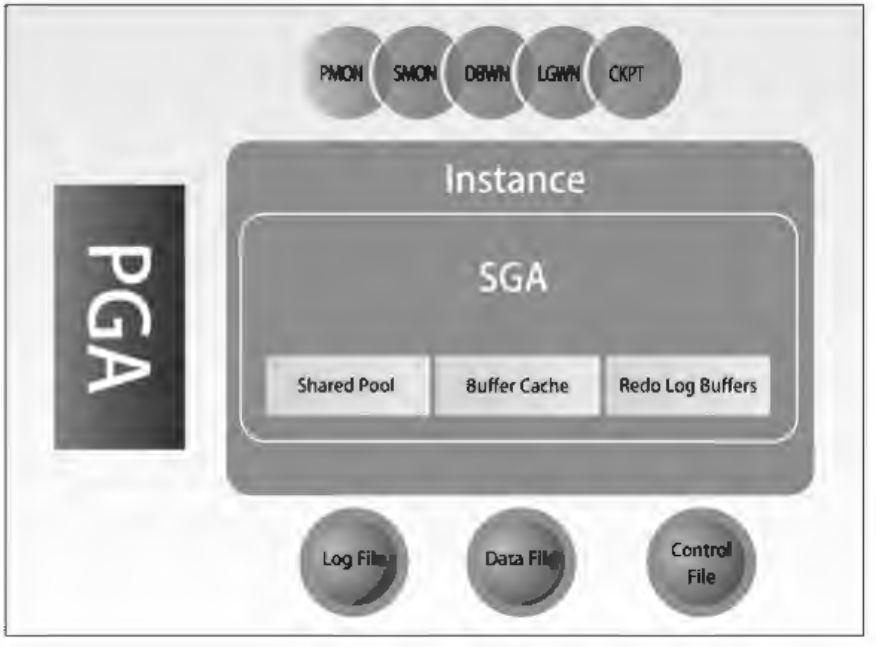


Figure 2-8 Oracle architecture

© Cengage Learning 2012

The Instance and the Database

An **instance** is a broad term that refers to the background processes and structured memory used during interaction with the database. To create an instance, a user must connect to the database and establish a session. A session begins when a user connects to the database and it ends when the user logs out of a database. The set of background processes in the instance include the performance monitor (PMON), the system monitor (SMON), the recovered (RECO), the log writer (LGWR), and the checkpoint (CKPT), and will be discussed in more detail later in this section. It is important to note that these resources play a vital role in Oracle's dependability and performance. Two critical memory structures included within an Oracle instance are the System Global Area (SGA) and the Program Global Area (PGA). These will be discussed in greater detail later in this section.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

The database portion of an Oracle server holds the database files that an environment needs to run the Oracle database. This set of files helps to configure the instance, make it possible for SQL executions to be processed, and ensure alert and recovery from hardware and software failure. Example files found in this area are the control, the redo log, and the data files. These files will be discussed in detail in the next section.

The Physical Structure

For the most part, the physical structure is dependent on the operating system on which Oracle is installed, yet, a few files are required with every Oracle install. The files contained within the physical structure interact with the operating system and are transparent to the user. These common files are the datafile, the control file, and the redo log.

- ***Datafile***—This file contains the actual data for the database and holds the information for all logical structures (tables, records, etc.) in the database.
- ***Control file***—This file contains the location and important credentialing information of other files, and is critical to the functionality of the database. Because it contains location information for other important files, if your control file fails, the database will not run. Examples of information held within the control file are: read/write status of files, database unique ID, point of failure if failure occurs, and the name and location of the datafile. Duplicate copies of the control file should be made for fail-safe reasons.
- ***Redo log***—This file contains information about all changes made to the data within the database. This is similar to an undo button for the database. The redo log can be used to restore data in the event that data is lost. All installations of Oracle include at least two copies of the redo log, and it is good practice to make a duplicate copy for fail-safe reasons.

The Memory Structure

The memory structure of Oracle holds the secret to why this RDBMS works so efficiently in a multiuser environment. As it is known in the information technology field, the main memory and cache is the quickest accessible storage area in any system. Therefore, running processes directly from memory will ensure that your systems are efficient and reliable.

In Oracle, practically everything happens from within the main memory. Even the users connect to the memory through the server process (defined later in this section). This is one reason that Oracle RDBMSs are capable of reliably servicing hundreds upon thousands of users concurrently.

One example of the way Oracle utilizes memory to speed up processes is in its query caching. **Caching** is the process of saving a duplicate of requested data to another area of a system in hopes of saving resources and speeding up the future requests for that same data. In an Oracle database, queries are cached into the buffer area to further utilize memory, and therefore increase the speed of future query returns. The memory structure of Oracle is divided into two main parts: the System Global Area (SGA) and the Process Global Area (PGA).

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

System Global Area The System Global Area (SGA) is the central area in which all shared data and processes are stored. The information contained in the SGA includes information shared by users and database processes. The SGA also holds the control data for one single instance in Oracle. Oracle 11g uses dynamic SGA, which means the data can be changed and an SGA is created for each new instance of Oracle. The SGA contains a number of required and optional memory structures. Tables 2-1 and 2-2 illustrate the required and optional memory structures found within the SGA.

Process Global Area The Process Global Area (PGA) is the central area where information is stored for background and server processes. Similar to the ways that the SGA allocates space for each new instance, the PGA allocates space for each individual background process. The Process Global Area content varies depending on the configuration of Oracle.

Structure	Description	Information and examples
Database buffer cache	Used to cache information read from the data files as well as recently used SQL and PL/SQL queries	A user executes a query on a client machine, and when the client connects to the dedicated server process. The server process first checks the cache to see if the file already exists. If it does, it passes the result to the users; if it is not in the cache, the server process takes data from the datafile and loads it into the buffer cache for future use
Shared pool	Stores the most recently executed SQL statements and data definitions Contains the library cache and data dictionary cache	When you execute a SELECT query, once executed, the statement is saved in the shared pool; it checks the rights of the user to ensure the user can view the requested information and then removes the requested information from the datafile, loading it into the buffer cache for the user to view
Library cache	Used to cache metadata information	An SQL statement is stored here parsed while the syntax is checked by the database; once validity is confirmed, it searches in the shared pool for a cached version of the statement
Data dictionary cache	Caches recently used data dictionary information	User account information, datafile names, etc.
Database buffer cache	Stores blocks of table and database data that has been retrieved in the past	A query is sent to the server process; the server process first looks in the database buffer to find the information needed to access the hard disk, thus speeding up performance
Redo log buffer	Stores all changes that have been made to the database	When the redo buffer gets filled, the excess moves to the redo file

Table 2-1 Required memory structures within the SGA

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Structure	Description	Information/example
Large pool	Stores large jobs to avoid filling the shared pool	RMAN backups
Java pool	Stores and caches Java commands	Necessary only when Java is installed
Streams pool	For advanced queuing	Only used in Oracle 10g and 11g

Table 2-2 Optional memory structures within the SGA

The Processes

A process is a set of instructions that is executed by the operation system and intended to complete a task. To complete an instance in Oracle, two tasks must take place. First, a user must run an application tool, such as SQL, to request a connection to the Oracle Server. This is referred to as a user process. Next, the server must handle the user's request and run a process to create the Oracle instance and complete the connection, referred to as a server process. Once the Oracle instance is created, background processes run to maintain relationships and resources among the Oracle structures.

Server processes can be shared or dedicated. In a dedicated server environment, each session owns a dedicated server process for executing SQL statements and handling user requests. Within a shared server environment, users must share server processes for handling requests and SQL statements. The configuration is dependent on an organization's size and needs. Larger organizations may opt for a shared server environment because it minimizes the number of processes running on the server, while smaller organizations may be less concerned with the server overhead.

There are several background processes that begin with every Oracle instance. Some are required, while others are options. Table 2-3 illustrates the different background processes found within an Oracle instance

Process name	Example process task
PMON (process monitor)	Cleans up after processes complete or processes fail
SMON (system monitor)	Recovers the database in case of failure by using the redo logs as well as database files
DBWN (database writer)	Writes changes from the database buffers to the database files
LGWR (log writer)	Writes modifications from the redo log buffer to the redo log files
CKPT (checkpoint)	Writes to the control file's established commit points where recovery begins, if necessary

Table 2-3 Background processes

MySQL Architecture

The MySQL architecture has been developed for multiplatform use. MySQL can be found on Solaris, Linux, or Windows. The architecture components defined in this section are the components that remain consistent on all three of these operation systems. This section will

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

explore the main structures of the MySQL server architecture: the database connection manager, query engine, transaction manager, and storage engine. The MySQL architecture is illustrated in Figure 2-9.

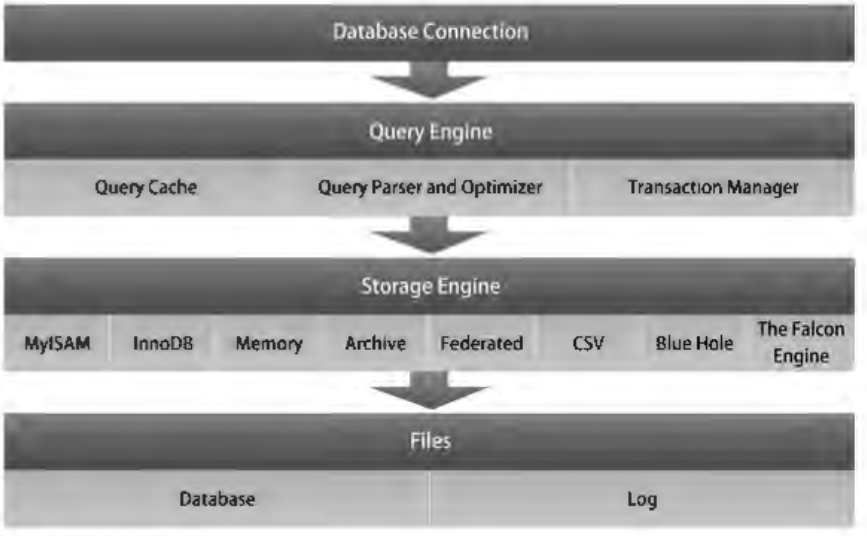


Figure 2-9 MySQL architecture

© Cengage Learning 2012

Database Connection Manager

The **database connection manager** does just what its name implies. It manages connections to the MySQL server. The connection management layer is very versatile, enabling virtually any client to connect to a MySQL server. An open-source application built to run on virtually any platform, MySQL has provided several ways for clients to connect via the connection management layer. Developers can create clients and application programming interfaces (APIs) in virtually any of today's modern programming languages (e.g., C, C++, Perl, and PHP), and clients using Open Database Connectivity (ODBC), Java, and .net are provided support through MySQL's interface.

Although there are several options for connecting to MySQL through the connection management layer, TCP/IP is the most common type of connection utilized. Not only is TCP/IP secure, but it can be used in virtually any OS environment (Windows, UNIX, OS/2 Linux and Solaris), making it a very attractive choice for users. It is safe to assume that most applications connecting to MySQL servers are running using TCP/IP.

Each server connection creates its own thread, to which MySQL dynamically assigns buffer space during query execution. A **thread** is an execution that runs independently from other processes and utilizes a portion of the CPU. These threads handle all requests from the client during a connection, including authentication and query processing. There are as many

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

threads as connected users at any given time. Although this dedicated thread approach provides advantages to the user, it can cause great overhead to the server, so systems are in place to ensure that threads are utilized efficiently.

Besides managing the actual connections to the database, the database connection manager allows clients to connect to local and remote databases using predefined connection profiles. These profiles allow users to define a specific role based on their intended use of the database, and then save that role along with its configuration parameters for later use when connecting to the network. The profile dialog box offers great convenience for users, eliminating the need to reconfigure the connection each time.

Query Engine

The **query engine** is the component of the architecture that optimizes and manages queries and SQL statements. The query engine has been built to use resources at the greatest efficiency, therefore, it can equally support data from both OLTPs and DSSs. This is partly because the queries in MySQL are written using a variation of the standard version of SQL that provides for much more powerful searching. In this section, we will review the process for receiving, parsing, and optimizing queries within the MySQL query engine.

Typically, when users connect to MySQL using their client application and the configuration manager, a query is initiated. Once the query request is received by the MySQL server, it is the responsibility of the data manipulation language (DML) to find relevant SQL statements and retrieve them from the client-side application. The data definition language (DDL) then provides access for the SQL to interact with the database.

Before the access takes place, the query parser creates a treelike structure based on the SQL statements extracted. The parsed SQL is then checked for validity within the syntax and the semantics of the tree. If the statement is not valid, an error message is sent back to the user. If the statement is indeed valid, then the integration manager determines whether the users have access to obtain the information that they are requesting.

The query optimizer runs testing speed variables. While most RDBMSs optimize for cost of resources, MySQL optimizes for speed, which is the reason for its success in this area. Once the fastest path to retrieving the information is found, the query is then executed by the query execution engine.

A memory component that plays a role in ensuring that query processing is successful is the **query cache**. The query cache maintains a set of recently requested queries within its storage space to save time and resources in the event that a similar query is requested again. If a similar query is requested, that would produce exact results of those in the query cache, resulting in shortened query processing, because there would be no need to parse the query. The query cache returns the previous results instead.

Transaction Manager

A MySQL transaction is a group of MySQL queries that are treated like one single process. As described earlier the transaction manager's task is to maintain concurrency throughout the database. This means that the transaction manager must ensure that simultaneous data handling will not cause the data to become corrupt or unreliable. The **transaction manager** is responsible for avoiding and resolving deadlocks and corrupted data by initiating the

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

COMMIT and ROLLBACK commands on the server. InnoDB and BDB are the storage engines that have been added to MySQL to manage transactions.

There are database environments that are transactional and some that are nontransactional; it depends on the needs of the organization. MySQL is a transaction-safe, ACID-compliant environment. In order to pass the ACID test, a transaction must maintain atomicity, consistency, isolation, and durability. As you will see in the next section, a MySQL administrator can choose whether or not to have a transactional database within the environment. MySQL's pluggable storage engines add this customization and are a feature that only MySQL provides. Table 2-4 illustrates MySQL's ACID compliance standards.

ACID characteristic	Description	MySQL sample compliance
Atomicity	SQL statements operate together as one entity group or alone as one entity; a group passes and/or fails as one process	MySQL statements begin and end, as well as pass and fail, together using the BEGIN and COMMIT, UNDO, and ROLLBACK statements
Consistency	Transactions do not affect the state of the database; consistency remains despite the success or failure of a transaction	MySQL uses logs (binary logs) to record all changes to the database as well as to help recover from a failure; the ROLLBACK statement is used if needed
Isolation	Transactions run separated from other transactions and are not viewable until committed	MyISAM permits locking, which avoids data corruption and visibility
Durability	Transactions persist despite system failures	Binary logs can be reverted

Table 2-4 ACID compliance and MySQL

Storage Management

As with Oracle, MySQL stores its data in files in secondary storage (/var/lib/mysql). Dynamically traversing these files is just not fast enough for our database needs. **Storage management** refers to the process of storing and retrieving data throughout the database. As with Oracle, most of the work takes place within the main memory (the fastest memory), and buffers take on the greatest load. Storage management in MySQL is a three-tiered process involving the resource manager, buffer manager, and storage manager. Figure 2-10 displays the efforts of storage management within MySQL.

The Storage Engine

Storage engines are components of a MySQL database architecture that read and write data to and from the database and offer services to enable customization of an environment. MySQL's pluggable storage engines differentiate MySQL from the rest of the RDBMS, because in MySQL, administrators can *pick and choose* which storage engines they want to use for certain tables and/or applications.

Having this option gives an administrator more control over the reliability and security of the data of an environment, yet the decision should be made early on in the design process to

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

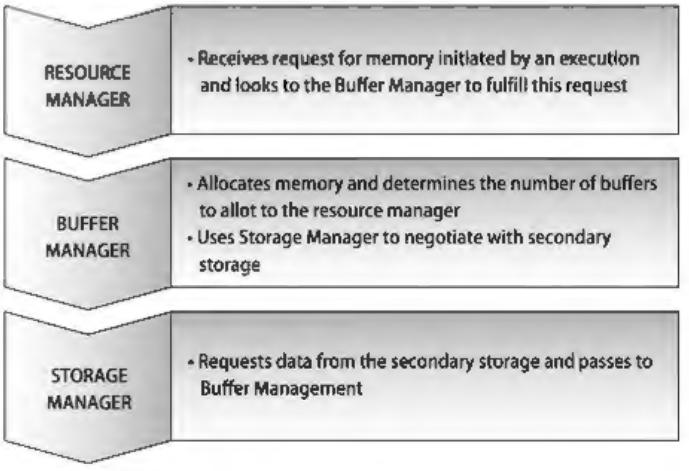


Figure 2-10 MySQL storage management
© Cengage Learning 2012

avoid issues later in the developmental cycle. This section will review each of the storage engines available in MySQL, as well as discuss the considerations that should be made before choosing one storage engine over another.

- *MyISAM*—As MySQL's default storage engine, providing variable row lengths and 256 TB of data, MyISAM is one of the most-used storage engines supported in all MySQL configurations. Fitting for both OLTP and DSS type servers, MyISAM offers fast searches and merge engine features. This engine can provide nontransactional environments as well as enable table locks.
- *InnoDB*—Another highly popular storage option, yet specifically designed for OLTP. InnoDB is a transaction-compliant engine that provides great performance for short-lived frequent transactions. InnoDB includes `COMMIT`, `ROLLBACK`, and row-level `LOCK`.
- *Memory*—Formally known as HEAP, this engine stores everything in RAM, so it is perfect for situations where you need fast access with data that never changes. This engine is great for DSS environments and aggregated data.
- *Archive*—This engine is great for storing and retrieving archive data. The archive engine only supports `INSERT` and `SELECT` queries where a full table scan is necessary for each `SELECT` statement.
- *Federated*—Enables separate MYSQL servers to be linked, referring to remote servers for all operations. This engine stores everything on the remote servers, so it is good for distributed environments where data does not need to be stored locally.
- *Comma-separated values (CSV)*—Stores data by comma delimitating within text files. Useful for certain kinds of logging, this engine provides easy exchange of data between applications.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

- *Black Hole Engine*—Accepts data, but has no storage utility, therefore it discards every `INSERT` and does not retrieve data. These are useful in distributed environments or in replication configurations.
- *The Falcon Engine*—Designed for 64-bit processors and larger memory banks, but flexible enough to work in smaller environments as well. Falcon is a transactional-safe engine capable of handling short and frequent transactions.

Microsoft SQL Server Architecture

Microsoft SQL is often referred to simply as *SQL Server*. SQL Server is an RDBMS developed by Microsoft to provide a fast, secure, and scalable data access platform. SQL Server's primary query languages are T-SQL and ANSI SQL. The scalability of the SQL server architecture is its most attractive feature, because it was developed to run equally in a wide variety of environments. From a database running on a personal computer servicing only one user, to a database stored on a cluster of servers servicing several thousand users, SQL Servers can meet the needs of any Windows environment. Figure 2-11 displays the architecture of a Microsoft SQL Server.

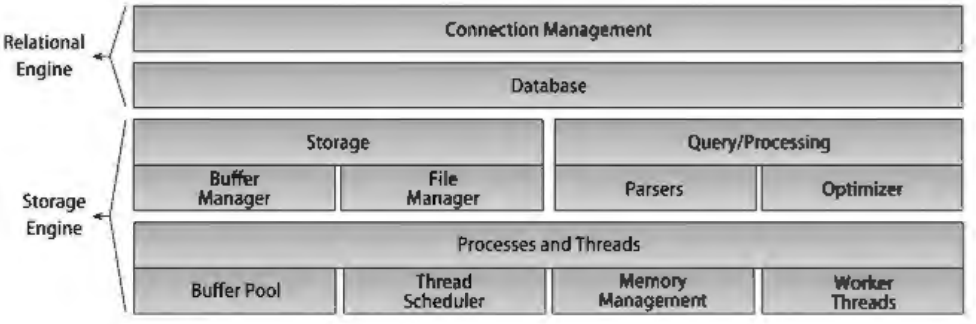


Figure 2-11 Microsoft SQL Server architecture

© Cengage Learning 2012

Architecture and Engines

The Microsoft SQL Server architecture has been developed for Windows platform use. This section will explore the main structures of MySQL. The topics covered in this section include connectivity, the relational engine, query processing, the storage engine, transaction processing, and the physical files.

Clients that connect to the server do so using the Tabular Data Stream, a Microsoft proprietary format. **Tabular Data Stream** (TDS) is a Microsoft-defined protocol that describes the specifications as to how the SQL Server and a client can communicate. TDS describes the specifications for which type of data, as well as the means by which data can travel. TDS is often encapsulated within another transport protocol such as TCP/IP, so clients

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

that want to transmit data to and from a SQL server can do so using TCP/IP, as well as other transport protocols in which TDS is encased. The SQL Server relational database management system architecture can be broken down into two main components: the relational engine and the storage engine.

- *The relational engine*—The relational engine's primary responsibilities lie in query processing and data retrieval. The tasks that the relational engine completes are initiated by user queries and include parsing the SQL, optimizing the execution plan, and retrieving the results for the user.
- *The storage engine*—The storage engine's primary responsibilities are to ensure the effective management of files, memory, recovery, logging, and transactions. The storage engine assures that memory is available and being allocated appropriately, files are being effectively utilized and stored, backup is being completed as often as necessary, and data or structures can be restored reliably if needed.

The Physical Structure

Every database management system needs a place to store data. Microsoft SQL Server is a database installed within an operating system, so its system is stored in a set of operating system files which are then broken down into pages (pages are described later in the chapter). SQL uses three different types of files to store data: primary data files, secondary data files, and log files. Each installation of SQL Server has at least one primary file and one log file.

- ***Primary data file***—The main data file for an SQL Server database, it references all other secondary data files and is the file of origin for the entire database. There must be a minimum of one primary file with every installed database. The file extension (although not enforced) of a primary file is .mdf.
- ***Secondary data file***—An optional data file found within a SQL Server database that is not a primary data file. Essentially, any data file that is not a primary file is a secondary file. The file extension (although not enforced) of a secondary file is .ndf.
- ***Log file***—A file that stores information about the transactions in the database to be used for recovery and backup. There must be a minimum of one log file in every database. The file extension (although not enforced) of a log file is .ldf.

In an SQL Server, data files can be collectively combined and placed into one group called a filegroup. Building filegroups enables administrators to create secondary files in the same group to better organize and place data. A **filegroup** is a collection of one or more physical data files within a SQL Server database. Only data files can reside in the same filegroup; log files must remain separate. There are two main types of filegroups:

- ***Primary filegroup***—The collection of files that contain all of the SQL Server system files, including the primary data files
- ***User-defined filegroup***—A collection of files created by a user

These filegroups can be used to organize a database into logical units of resources. The way filegroups are used will be dependent on the environment and the data that is stored within, yet ineffective organization of filegroups can have a negative impact on the backup and recovery of a database. A database has only one filegroup.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Memory Management

One of the most noteworthy characteristics of SQL Server's memory architecture is that it can dynamically allocate its own memory without a need for an administrator's intervention. By default, each instance of SQL Server dynamically acquires memory as needed without creating a shortage in the system. The goal of the database, as with MySQL and Oracle, is to limit the number of times that the secondary storage is accessed. The hard disks are the slowest and most resource-costly storage. By minimizing the reads and writes to them, the server performance will be increased.

As was explained earlier in the chapter, the goal is to maintain speed by maximizing the use of main memory and the system cache. Accomplishing this goal requires a great deal of monitoring and balancing. The main memory is used for many database processes within the server, so although increasing the size of the storage allotted to these processes will help maximize its use, using too much of it to store and retrieve data would starve the rest of the system of memory. Therefore, proper memory management is one of the keys to the success of SQL Server's dynamically allocated memory.

The virtual memory plays a large role in proper memory management. **Virtual memory** is a technique for extending the availability of memory where by units of storage located on different memory devices are used to store data from one entity in such a way that it appears the data has been stored in one continuous block of the same memory. The fixed units of storage that are transferred or *swapped* from one storage device to another are known as **pages**. The page is the primary unit of data storage in SQL Server. Pages are often swapped from the main memory (RAM) to a secondary form of memory (a hard disk drive) to handle overflow and/or ensure ideal utilization of storage. The dedicated swap space for the page is known as a **pagefile**. The **virtual address space** is the complete virtual memory area allotted to a program.

The goal is to maximize the use of the main memory for the database without exceeding the available memory on the server. In order to achieve this goal, SQL maintains a buffer cache in memory to hold pages that were previously read from the database. As stated earlier, the larger the size of the buffer pool, the fewer reads and writes to the hard disk are necessary. If the buffer cache uses up too much of the main memory, the result is that the operating system begins implementing virtual memory by swapping memory to and from the dedicated pagefile. This process ultimately uses fewer resources and increases performance.

There are two major components of the SQL server virtual memory architecture: the memory pool and the executable code. Figure 2-12 displays the memory architecture of an SQL Server machine.

The Executable Code The executable code is essentially the server engine within SQL Server. It includes the executable (exe) and dynamic link library (dll) files for the server itself. The executable code includes the following components:

- SQL Server code
- Server Net-Library DLLs
- Open services code
- Extended stored procedures

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

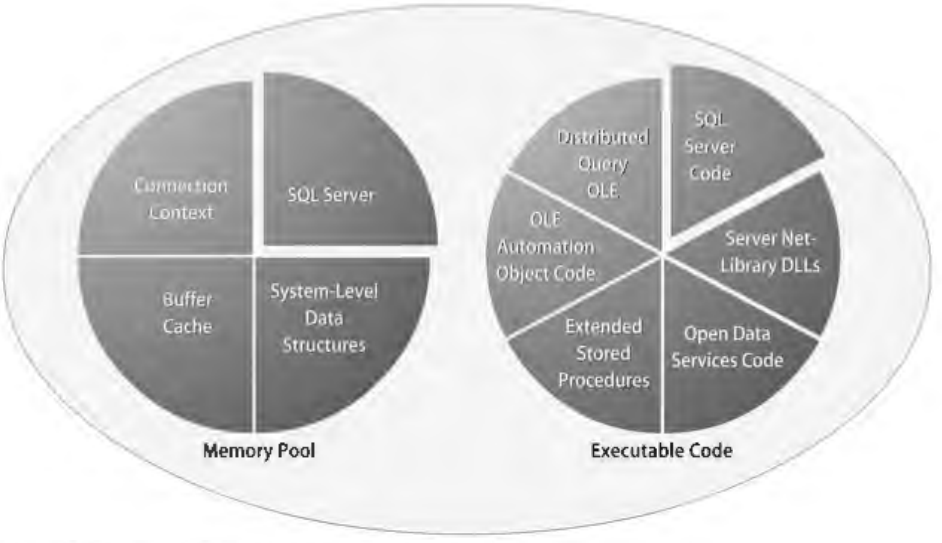


Figure 2-12 Microsoft SQL Server memory management
© Cengage Learning 2012

- OLE Automation
- Object code
- Distributed Query OLE DB provider DLLs

The Memory Pool The memory pool is the total amount of memory available for an instance of SQL Server. Virtually everything that uses memory in an instance of SQL Server uses the memory pool. The size of the memory pool depends on how many instances and applications are running. It is the most dynamically changing portion of a SQL Server instance.

The memory pool can fluctuate between an established minimum and maximum value as it attempts to fulfill the needs of the applications and instances and maintain a level of virtual memory allocation just below the true amount of physical memory (4–10 MB). The min server memory and the max server memory configuration settings are the established minimum and maximum amount of memory that can be used by the memory pool.

A server instance begins with the amount of memory that it needs (not the min server memory value) and then adds and frees memory as determined by the number of applications, instances, and user requests. The memory pool never exceeds the amount specified within the max server memory value, and if at any time the memory value reaches that established within the min server memory size variable, more memory is acquired. Five main objects within the memory pool are allocated memory:

- *System-level data structures*—Stores all data that is global to the instance.
- *Buffer cache*—Stores data pages from the database to avoid reads and writes to the hard drive

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

- *Procedure cache*—Stores execution plans for all currently and recently run SQL statements
- *Log cache*—Stores pages that are read from and written to logs
- *Connection context*—Stores a record of the current state of the connection

Buffer Management

Buffer management is vital to the performance of the SQL Server machine. As is apparent in the earlier section, buffers provide the location within physical memory where data is stored, minimizing the reads and writes to secondary storage devices. There are two important components within buffer management. The **buffer manager** is a portion of the SQL Server responsible for accessing data pages and updating the database. The **buffer pool**, or **buffer cache**, as described earlier, is the area where data pages from the database are stored to minimize the need to read and write from the database file located on the hard disk.

Once the SQL Server starts, it determines and reserves the buffer cache size by calculating the amount of remaining physical memory. The reserved space for the buffer cache is called the **memory target**. The buffer cache acquires from this only what is needed for a current instance.

When a request is made for data (and after query processing), the buffer manager accesses the database files stored on the hard disk and retrieves the requested data. The data is placed in the buffer cache pages, and from there the data is read. If changes are made to the data while in the buffer cache, it is considered dirty. A dirty page is one that has been modified within the buffer cache, but not yet written back to the database. A dirty page can go through several modifications before it is written back into the database.

With each modification, a transaction row is added to the log cache to maintain recovery integrity. Once the transaction has been logged, and the dirty page has stopped being referenced (by users or SQL statements), then the buffer cache frees the data to be written back into the database.

Threads and Processes

Microsoft SQL server uses thread and sometimes fibers to perform several simultaneous tasks within the operating system. As described in an earlier section, a thread is a process that contains tasks or executions that run independently from one another, yet utilize the same resources (portions of the CPU). The use of threads enables several different tasks to be performed simultaneously with minimal resource consumption.

When an instance occurs within the SQL Server database, an operating system process is created to handle that instance, which could potentially contain thousands of simultaneous user requests. To help manage these requests, the process is broken into different, independently working executions or threads that can run concurrently using very few resources.

A fiber is a subcomponent of a thread, and is handled by the server to accomplish a task. Fibers are not used in SQL by default; they must be configured. Where a thread is handled by the operating system and allocated one per CPU, a fiber is handled by the server and allocated one per user command. Therefore, there can be many fibers within one thread. Fibers are typically found in large enterprise environments where the processes generated are quite heavy and several CPUs exist on the server.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

SQL server keeps a pool of either threads or fibers for all user connections; these pools are called **worker processes.** The amount of threads or fibers available within one worker process depends on the size of the network. A value can be set to limit the number of threads within a worker process to lower resource consumption, yet this should be carefully considered. If the value of requests exceeds the value available, threads will be pooled, causing users to wait for a thread to be available to connect. If the value is set too high, too many resources will be consumed. Within Microsoft SQL Server, threads are prioritized on a scale from 1 to 31. If a thread is waiting to be executed, the operating system checks the priority and sends out the one with the highest priority. SQL Server threads have a priority of 7, giving them a normal priority level, which is enough to access resources without interfering with any other processes or applications. This value can be changed as well, yet again at a cost if not carefully considered.

Chapter Summary

- A database is a collection of data stored within a database management system that allows others to search, analyze, and manipulate stored data.
- Data is stored within a database using a table structure, whereas general categories can be found within columns; distinct units of data are contained in rows.
- Keys are used to identify distinct entries within a database and are used to access or manipulate rows of data. Using keys to reference data creates an object-oriented storage system that lowers the probability of redundancy and makes data entry and manipulation simpler.
- There are several types of keys used to reference data within a database: primary, foreign, secondary, candidate, composite, sort, and alternate.
- One primary key is required for each row of a table, as it provides a unique identifier for referencing that row.
- Foreign keys can be used to build relationships between tables and are not supported by all database management systems.
- Query languages act as the intermediary for users and DBMSs, providing users a way to retrieve and manipulate data from a database. There are a variety of query languages, but SQL is the most common query language in modern database systems.
- Different database models describe the way data is stored within a database: flat, hierarchical, network, and relational, to name a few models. Building relationships between tables, such as within a relational database model, is the most common and efficient way to store data in today's databases.
- The most common relationships formed between tables in a relational database are one-to-many relationships. These types of relationships are defined by which one entity relates with several other entities of that same database. Other examples of relationship types created within databases include one-to-one and many-to-many.
- Databases can be built to handle very active, high-volume query environments such as online transaction processing and POS environments, or stagnant, low-volume query environments such as a historical data warehouse or data support systems.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

- There are several different DBMSs available today, each having unique advantages and disadvantages. The most commonly used database management systems are Oracle, MySQL, and Microsoft SQL Server. It is important to understand a DBMS's architecture before choosing one for a specific environment.
- Two very important aspects of a DBMS are read consistency and query management. The method by which a DBMS accomplishes read consistency and query management can indicate the system's efficiency and reliability.
- Read consistency is dependent on a database's ability to manage multiple processes and refers to the accuracy and reliability of data. Read consistency is achieved through the use of efficient transaction processing, concurrency handling, and data-locking mechanisms.
- The general query processing steps of a database are parsing, optimization, and execution of a SQL statement.
- Oracle's architecture has two main components, the instance and the database. The instance begins with a user's connection and includes the background processes and memory managment of the application, while the database component involves the physical file structure.
- MySQL's main architectural components include the connection manager, query engine, and storage management engine. The connection manager handles connections to the database, the query manager processes SQL statements, and the storage engine manages memory and storage for the application.
- The main components of SQL Server include the relational engine and the storage engine. The relational engine involves query management, threads and processes, and the physical file structure, while the storage engine includes buffer and memory management.

Key Terms

alternate key A field with values that are not chosen as a primary key, but can be used in cases where the primary key is not available.

attribute A characteristic or variable that describes or further identifies an entity.

buffer manager A portion of the SQL Server responsible for accessing data pages and updating the database.

buffer pool (buffer cache) The area where data pages from a database are stored to minimize the need to read and write from the database file located on the hard disk.

caching The process of saving a duplicate of the requested data to another area of a system in hopes of saving resources and speeding up the future requests for that same data.

candidate key A field with values that meet the requirements for a primary key.

column (field) The component of a table that maintains a general category of information with similar datatypes.

commit To make a change within a DBMS that is permanent and visible to other users.

composite key A group of two or more fields where their values can be combined to be used as a primary key.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

concurrency The simultaneous access of resources and data.

control file A file within a database that contains the location and important credentialing information of other files.

database A collection of data stored on a computer using an application called a database management system.

database connection manager Manages connections to the MySQL server.

database management system (DBMS) An application that allows users to search stored data in order to locate specific information.

database model A representation of the way data is stored.

datafile A file that contains the actual data for the database and holds the information for all logical structures (tables, records, etc.) within the database.

deadlock A situation when two transactions cannot proceed because each user has data that the other needs.

entity A person, place, or thing stored within the table of a database and for which attributes and relationships exist.

fiber A subcomponent of a thread, which is handled by the server to accomplish a task.

filegroup A collection of one or more physical data files within a SQL Server database.

flat model A two-dimensional list of data entries, where all data within a field are understood to be similar, and all data within a record are understood to be related to one another.

foreign key A field within a table that contains a label used to build a relationship between two tables.

hierarchical database structure A treelike storage schema that represents records and relationships through the use of tiers and parent-child relationships.

instance A broad term that refers to the background processes and structured memory used during interaction with the database.

key A single field or group of fields used to identify an entry in a table.

lock A mechanism within a DBMS that controls concurrency by preventing users from taking hold of data until changes being made are completed or committed.

log file A file that stores information about the transactions in the database to be used for recovery and backup.

memory target The reserved space for the buffer cache.

network database model A treelike structure that stores information in the form of a hierarchy, using tiers and parent-child-like entities to represent relationships.

online analytical processing (OLAP), (decision support systems [DSS]) Databases that store large volumes of historical data for report generating and analyzing.

online transaction processing (OLTP) database A database that is created for real-time storage and manipulation of data within an organization.

open source A term that refers to software that has been written to be distributed for use and downloaded free of charge.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

optimization The process of locating the quickest and most efficient way to retrieve the data being requested by a user.

page A fixed unit of storage that is transferred or swapped from one storage device to another.

pagefile The dedicated swap space for a page.

parallel processing When more than one server processes one query at the same time.

parsing The act of analyzing a construction of a query for correct syntax and semantics.

point of sales (POS) system A system that is meant to handle cash register or sales transactions.

primary data file The main data file for an SQL Server database which is the file of origin for the entire database and references all other secondary data files.

primary filegroup The collection of files that contains all of the SQL Server system files, including the primary data files.

primary key A field that contains a unique label by which we can identify a record or row in a table.

process A set of instructions that is executed by the operating system intended to complete a task.

Process Global Area (PGA) The central area where information is stored for background and server processes. It allocates space for each individual background process.

query A search initiated by a user in an attempt to retrieve certain information from a database.

query cache A memory component that plays a role in ensuring that query processing is successful.

query engine A component of the architecture that optimizes and manages queries and SQL statements.

query management The steps taken by a database management application to process a user query.

read consistency A term that refers to the accuracy and reliability of data within a database.

redo log A file within a database that contains information regarding all changes made to the data within the database.

relational database A storage model in which common entities are stored within separate tables that use unique key identifiers to build relationships between these entities.

relationship Defines the association between two entities and binds them.

report A document that contains a formatted result of a user's query.

row (record, tuple) The component of a table that holds distinct units of data identified using unique strings of numbers or characters.

secondary (alternative) key A field with values that contains nonunique data and that can refer to several records at one time.

secondary data file An optional data file found within an SQL Server database that is not a primary data file.

sort (control) key A field in which values are used to sequence data.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

storage engine A component of the MySQL database architecture that reads and writes data to and from the database and offers services to enable customization of an environment.

storage management Refers to the process of storing and retrieving data throughout the database.

System Global Area (SGA) The central area where all shared data and processes are stored, including information shared by users and database processes.

table One of the most basic units of storage within a database, typically representing unique and specific data objects.

Tabular Data Stream (TDS) A Microsoft-defined protocol that describes the specifications as to how the SQL Server and a client can communicate.

thread A process that runs independently from other process. It utilizes a portion of the CPU and contains tasks or executions that share the same resources, yet run independently from one another.

transaction The group of statements or operations processed by a database to execute a user's request to update or change the database.

transaction manager A component of the MySQL database architecture that is responsible for avoiding and resolving deadlocks and corrupted data.

user-defined filegroup A collection of files created by a user.

virtual address space The complete virtual memory area allotted to a program.

virtual memory A technique for extending the availability of memory by which units of storage located on different memory devices are used to store data from one entity in such a way that it appears as though the data has been stored in one continuous block of the same memory.

worker process A pool of either threads or fibers that SQL Server keeps for all user connections.

Review Questions

1. What is a database?
2. What is the main goal of a DBMS?
3. What are the common components found within DBMSs?
4. List the common components of a table.
5. What are the advantages to using keys rather than specific table entries?
6. Identify and define the four main database models.
7. Explain the concept of relationships within a database.
8. Identify the three relationship types and give a brief explanation of each.
9. List the two types of databases, as well as their purposes.
10. What are the three most prominent and fastest-growing RDBMSs?
11. Identify two main areas of focus that exist in all DBMSs.

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

12. List the five locking mechanisms used in maintaining read consistency.
13. What are the main components of the Oracle architecture?
14. What are the main componenets of the MySQL architecture?
15. What are the most noteworthy characteristics of the SQL Server's memory architecture?

Case Projects

Case Project 2-1: Real-World Examples

Use the Internet or your current work environment and provide a practical, real-world use of an OLAP and a practical, real-world use of an OLTP. Include in your answer the name of the organization, the nature of the business, and the way in which the database is being manipulated.

Case Project 2-2: Database Management System Vendors

Oracle, MySQL, and Microsoft SQL Server are only three of the many database management systems. Use the Internet and find three more database management systems available in today's market. Provide a brief description of the systems that you found, include the year that they were developed, and the operating systems that can be installed, as well as the main features of the applications.

Case Project 2-3: Understanding ACID-Compliant Transactions

Choose one database management system from Case Project 2-2 and determine whether it is ACID-compliant. Create a table similar to Table 2-4, providing examples as to how the database does or does not meet ACID standards.

Case Project 2-4: Understanding Query Management

Choose one database management system from Case Project 2-2 and discuss its query management process.

Hands-On Projects

Hands-On Project 2-1: Improving Upon a System

A convenience store located within an apartment complex uses a log similar to the following one to keep track of customer purchases. The goal of the database is to keep track of the inventory sold in hopes of using the data to better meet customers' convenience store needs. From the information gathered, the store owner is able to obtain statistics that help her plan future inventory. Data such as the number of products sold, the highest and fewest products sold, the frequency and amount of customer purchases, and the total purchases made by a specific apartment aid the owner in determining how best to suit her customers' needs. Write a paper that discusses how you might

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

achieve the owner's goal in a more efficient manner using a database. Answer the following questions in your paper:

- What database model would you implement?
- What type of database fits this scenario best?
- What DBMS is best suited for this scenario, and why?
- What would be the cost of implementing your proposed system?
- How would training and maintenence of the system be handled?

Customer name	Apt #	Product name	Product price	# Purchased
Joseph Anthony	1125	Orange juice	4.59	1
Joseph Anthony	1125	Bread loaf	2.29	1
Yolanda Burns	3221	Milk	3.67	1
Yolanda Burns	3221	Candy bar	1.19	3
Frances Jordan	1138	Gum	.99	2
Steve Miller	2221	Gum	.99	1
Cho Lin	2239	Bread loaf	2.29	1

Hands-On Project 2-2: Building a Relational Database

You have been hired to design a relational database for a convenience store which is located within an apartment complex. The goal of the database is to keep track of the inventory sold in hopes of using the data to better meet the customers' convenience store needs. Up until your arrival, the store kept track of each customer's purchases using a flat database log, as shown in the following table. Using the information provided, build a relational database that will allow for querying things such as products sold, customer purchases, total apartment purchases, and total spent per apartment. Include any created tables and identify the keys and key types that are used. Identify all relationships, labeling them 1:1, 1:N, or M:N.

Customer name	Apt #	Product name	Product price	# Purchased
Joseph Anthony	1125	Orange juice	4.59	1
Joseph Anthony	1125	Bread loaf	2.29	1
Yolanda Burns	3221	Milk	3.67	1
Yolanda Burns	3221	Candy bar	1.19	3
Frances Jordan	1138	Gum	.99	2
Steve Miller	2221	Gum	.99	1
Cho Lin	2239	Bread loaf	2.29	1

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.