**Week 1: Readings of Chapter 1**

The difference between data and information?

Data are raw facts. The word "raw" indicates that the facts have not yet been processed to reveal their meaning. Processing data yields information on which business decisions can be based. Data management is a discipline that focuses on the proper generation, storage, and retrieval of data. Typically, data management requires the use of a computer database. By definition, a database is a shared, integrated computer structure that houses a collection of end user data, and metadata—data about the stored data. A well-designed database facilitates data management and becomes a valuable information generator. A poorly designed database is likely to produce redundant data, which can generate errors leading to bad decisions.

What a database is, about different types of databases, and why they are valuable assets for decision making?

A database is a shared, integrated computer structure that houses a collection of user data and metadata. Metadata is data about data such as descriptions of the relationships that exist among data. A database management system (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database. Databases are classified according to the number of users, the database locations, and the expected type and extent of use. The most popular way to classify databases is by the use and timeliness of the data. For example, a production database contains up-to-the-minute real-world information while a warehouse database stores data for making decisions.

Why database design is important?

A well-designed database facilitates data management and becomes a valuable information generator. A poorly designed database will likely become a breeding ground for redundant data. Redundant data are often the source of difficult-to-trace information errors. A database contains redundant data when the same data about the same entity are kept in different locations. The existence of redundant data can produce uncorrected data entries, and you probably will not know which value is the correct one. Reports might yield different results, depending on which version of the data was used. In short, uncontrolled data redundancies are typical of a poorly designed database. A poorly designed database tends to generate errors that are likely to lead to bad decisions, and bad decisions can lead to the failure of an organization.

How modern databases evolved from files and file systems?

File systems were widely used before the development of database software systems. Historically, the first computer applications focused on clerical tasks: order/entry processing, payroll, work scheduling, and other tasks. Such applications accessed data stored in files. Database models were developed to address the inherent weaknesses in these file systems. Rather than deposit data into different files, databases keep data within a single repository enabling tighter control to be maintained over data-related activities.

About flaws in file system data management?

In the past, file systems lacked the features that today's DBMS software provides. This made database management difficult and cumbersome. Each file typically required its own set of data management programs and was generally unique to the application that "owned" it. Many files would suffer from data redundancy leading to inconsistencies, anomalies, and lack of data integrity. Each file was used by many application programs. This meant that a mature file system might have generated hundreds of thousands of programs. Serious problems resulted from this data dependency. Access to any file was dependent on data characteristics and storage formats. Even a minor change to a data structure within a file would require changing all programs accessing that data.

How a database system differs from a file system, and how a DBMS functions within the database system?

A database management system (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database. The DBMS contains a query language, which can produce quick answers to ad hoc queries. It also helps to create an environment in which end users have better access to more and better-managed data. This promotes an integrated view of the organization's operations. The DBMS serves as an intermediary between the database and user. In effect, the DBMS receives user requests and translates these requests into the complex code required to fulfil the requests. In this way, the DBMS hides much of the database's internal complexities.

Why data models are important?

A model is an abstraction of a more complex real-world object or event. A data model is the relatively simple representation, usually graphical, of complex real-world data structures. The model's main function is to help us understand the complexities of the real-world environment. Within the database environment, a data model represents data structures and their characteristics, relations, constraints, and transformations. Good database design uses an appropriate data model as its foundation. A data model provides a blueprint of the data that is required for a functional system.

About the basic data-modeling building blocks?

The basic building blocks of all data models are entities, attributes, and relationships. An entity is anything, such as a person, place, thing, or event, about which data are to be collected and stored. Entities may be physical objects such as customers or products. But entities may also be abstractions such as flight routes or musical concerts. An attribute is a characteristic of an entity. For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone, customer address, and customer credit limit. The attributes are the equivalent of fields in file systems. A relationship describes an association among (two or more) entities. For example, a relationship between customers and agents might be described as “an agent can serve many customers and each customer might be served by one agent.” Data models use three types of relationships: one-to-many, many-to- many, and one-to-one. Database designers usually use the shorthand notations 1:M, M:N, and 1:1 for them, respectively.

What business rules are and how they affect database design?

A business rule is a brief, precise, and unambiguous description of a policy, procedure, or principle within a specific organization's environment. Business rules do not just apply to traditional business, but also to educational institutions, religious groups, government units, and others. Business rules, derived from a detailed description of an organization's operations, help to create and enforce actions within that organization's environment. Business rules must be rendered in writing and updated to reflect any change in the organization's operational environment. Properly written business rules are used to define entities, attributes, relationships, and constraints. Knowing the business rules promotes the creation of an accurate data model based on how the organization actually works and what role is played by the data within that organization's operations. Consequently, the database designer must identify the organization's business rules and analyze their impact on the nature, role, and scope of data.

How data models can be classified by level of abstraction?

When designing a database, the designer starts with an abstract view of the overall data environment and adds details as the design approaches implementation. The design of a database can be divided into four models with decreasing levels of abstraction. The conceptual model represents a global view of the database. It is the basis for the identification and description of the main data objects, avoiding details. The most widely used conceptual model is the entity relationship (ER) model. Once a specific DBMS has been selected, the internal model adapts the conceptual model to the DBMS. The internal model is the representation of the database as “seen” by the DBMS. The external model, based on the internal model, is the end user's view of the data environment. The physical model operates at the lowest level of abstraction, describing the way data are saved on storage media such as disks or tapes. The physical model requires the definition of both the physical storage devices and the (physical) access methods required to reach the data within those storage devices.

**Week 1: Readings of Chapter 2**

That the relational database model takes a logical view of data

The relational database model allows the designer to focus on the logical representation of data and its relationships, rather than on physical storage details. The practical significance of taking the logical view reminds the end user of the simple file concept of data. Unlike a file, however, the relational database provides the advantages of structural and data independence. Related records can be stored in independent tables that are related. This logical simplicity tends to yield simpler and more effective design methodologies.

That the relational database model's basic components are entities, attributes, and relationships among entities

The primary components of a relational database are entities, attributes, and resulting relationships. Entities are a person, place, or thing about which data are collected and stored. Attributes are characteristics that describe entities. Entities are often grouped according to common attributes. These groups are called entity sets and are stored in tables. Unique attributes or keys are used to identify specific occurrences of entities within a group. These key attributes are used to create controlled redundancies that link tables and form relationships. Relationships can be classified as one-to-one (1: 1), one-to-many (1: M), or many-to-many (M: N). This indicates the number of entities that can be related to another.

How entities and their attributes are organized into tables?

Entities are the basic building blocks of a relational database. Entities are represented with tables. A relational table is composed of intersecting rows and columns. Each row (or tuple) represents an occurrence of an entity. Each column represents a characteristic of the entity. These characteristics are called attributes. Each table row must have an attribute or attributes which uniquely define it. The unique identifier is called a primary key. Although tables are independent, they are linked through shared attributes. The primary key in one table appears again as a foreign key in another table. To maintain referential integrity, the foreign key must contain values only found in the other table, or null values to indicate that the rows are not linked.

About relational database operators, the data dictionary, and the system catalog?

Relational algebra operators are supported to various extents by different relational database management systems. Relational algebra defines the theoretical way of manipulating table contents by using eight primary operators: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE. The SELECT operator lists table rows according to a particular criterion. PROJECT yields a vertical subset for a table. JOIN combines information from two or more tables. INTERSECT yields rows that appear in both tables. UNION combines two tables. DIFFERENCE provides the rows in one table not found in the other. PRODUCT yields all possible pairs of rows from two tables. DIVIDE singles out particular characteristics shared by two tables. A minimally relational database supports at least the SELECT, PROJECT, and JOIN operators. The data dictionary provides a detailed accounting of all tables found within the database. It contains, at a minimum, all attribute names and characteristics for each table in the system. The system catalog contains more information than the data dictionary and is created by the system. The data dictionary can be derived from its contents.

How data redundancy is handled in the relational database model?

Database tables are independent but they can be linked through shared common attributes. These attributes are a form of controlled redundancy. Although, in many situations, data redundancy can lead to data anomalies and ruin a database application, proper use of foreign keys can minimize data redundancy and reduce the chance that destructive anomalies will develop. Database designers must often balance design elegance, processing speed, and information requirements when creating a database.

Why indexing is important?

An index provides a quick method for pointing to a particular location within a database. Indexes increase data retrieval speed and can be easily created with SQL commands. From a conceptual point of view, an index is composed of an index key and a set of pointers. The index key is the index's reference point. Without an index, each row in a database must be examined to find one meeting a particular criterion. An index speeds the search by pointing to the location of the data identified by the key.

**Week 2: Readings of Chapter 3**

That the relational database model takes a logical view of data

The relational data model enables designers to concentrate on the logical representation of data and its relationships, rather than on the physical storage details. The logical view of the relational database is facilitated by the creation of data relationships based on a logical construct known as a table. The pragmatic approach of taking the logical view is that it presents related records as being stored in independent tables. Thus, the relational database model is much easier to understand than its hierarchical and network database predecessors. Greater logical simplicity tends to yield simpler and more effective database design methodologies.

That the relational model's basic components are relations implemented through tables in a relational DBMS

A table is also called a relation because the relational model’s creator, E. F. Codd, used the term relation as a synonym for table. Independent tables in a relational DBMS are related to one another through the use of keys. A key consists of one or more attributes that determine other attributes. Keys are central to the use of relational tables. A key can be classified as a super key, a candidate key, a primary key, a secondary key, or a foreign key. A super key is an attribute (or combination of attributes) that uniquely identifies each row in a table. A candidate key is a minimal super key; that is, a super key that does not contain a subset of attributes that is itself a super key. A primary key is a candidate key selected to uniquely identify all other attribute values in any given row. Primary keys cannot contain null entries. A secondary key is an attribute (or combination of attributes) used strictly for data retrieval purposes. A foreign key is an attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

How relations are organized in tables composed of rows (tuples) and columns (attributes)

In a relational database, a table is perceived as a two-dimensional structure composed of rows and columns. Each table row (tuple) represents a single entity occurrence within the entity set. Each table column stands for an attribute, and each column has a distinct name. Each row/column intersection represents a single data value. All values in a column must conform to the same data format. Each column has a specific range of values known as the attribute domain. The order of the rows and columns is irrelevant to the DBMS. Each table must have an attribute or a combination of attributes that uniquely identifies each row.

About relational database operators, the data dictionary, and the system catalog

The relational database operators are SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE. UNION combines all rows from two tables, excluding duplicate rows. INTERSECT yields only the rows that appear in both tables. DIFFERENCE returns all rows in one table that are not found in the other table; that is, it subtracts one table from the other. PRODUCT yields all possible pairs of rows from two tables—also known as the Cartesian product. SELECT, also known as RESTRICT, returns values for all rows found in a table. SELECT can be used to list all of the row values, or it can yield only those row values that match a specified criterion. PROJECT yields all values for selected attributes. In other words, PROJECT yields a vertical subset of a table. JOIN allows information to be combined from two or more tables. The data dictionary provides a detailed record of all tables found within the user/designer-created database. Thus, the data dictionary contains at least all of the attribute names and characteristics for each table in the system. The purpose of the data dictionary is to ensure that all members of database design and implementation teams use the same table, attribute names and characteristics. Like the data dictionary, the system catalog houses metadata. The system catalog can be described as a detailed system data dictionary that describes all objects within the database, including data about table names, the table’s creator and creation date, the number of columns in each table, the data type corresponding to each column, index filenames, index creators, authorized users, and access privileges.

How data redundancy is handled in the relational database model

Data redundancy leads to data anomalies. These anomalies can destroy the effectiveness of a database. The relational database makes it possible to control data redundancies by using common attributes that are shared by tables, called foreign keys. The proper use of foreign keys is crucial to exercising data redundancy control. However, it is worth noting that, in the strictest sense, the use of foreign keys does not eliminate data redundancies because the foreign key values can be repeated many times.

Why indexing is important

An index is an orderly arrangement used to logically access rows in a table. From a conceptual point of view, an index is composed of an index key and a set of pointers. The index key is, in effect, the index’s reference point. More formally, an index is an ordered arrangement of keys and pointers. Each key point to the location of the data identified by the key. Indexes can also be used by a DBMS to retrieve data ordered by a specific attribute or attributes. When you define a table’s primary key, the DBMS automatically creates a unique index on the primary key column(s) you declared. Note that a table can have many indexes, but each index is associated with only one table. The index key can have multiple attributes (composite index). Creating an index is easy.

**Week 2: Readings of Chapter 4**

The main characteristics of entity relationship components

The Entity Relationship Model's (ERM) main components include entities, attributes and relationships. In both the Chen and Crow’s Foot models, an entity is represented by a rectangle containing the entity’s name. The entity name, a noun, is usually written in all capital letters. In the original Chen model, attributes are represented by ovals and are connected to the entity rectangle with a line. Each oval contains the name of the attribute it represents. In the Crow’s Foot model, the attributes are written in the attribute box below the entity rectangle. Attributes have a domain which is the attribute’s set of possible values. Attributes may share a domain. The ERM uses identifiers to uniquely identify each entity instance. In the relational model, such identifiers are mapped to primary keys in tables. Identifiers are underlined in the ERD.

How relationships between entities are defined and refined and how those relationships are incorporated into the database design process

A relationship is an association between entities. Relationships between entities always operate in both directions. Entity relationships may be classified as one-to-one (1:1), one-to-many (1:M), or many-to-many (M: N). Connectivity describes the relationship classification (1:1, 1:M, or M: N). Cardinality expresses the minimum and maximum number of entity occurrences associated with one occurrence of the related entity. In the Entity Relationship Diagram (ERD), cardinality is demonstrated by placing the appropriate numbers beside the entities, using the format (x, y). The first value signifies the minimum number of associated entities, while the second value represents the maximum number of associated entities. An ERD is capable of portraying relationship strength, optional or mandatory relationship participation, and the unary, binary and ternary degree of relationships.

How ERD components affect database design and implementation

The process of database design is an iterative one. An iterative process is, thus, one based on repetition of processes and procedures. In the initial phase, the initial ERD is developed after data about the main entities and relationships has been gathered. During the review process of building the ERD, it is likely that additional objects, attributes, and relationships will be uncovered. Therefore, the basic ERM will be altered to accommodate the newly discovered ER components. Subsequently, another round of reviews may yield additional components or clarification of the existing diagram. The process is repeated until the end users and designers agree that the ERD is a fair representation of the organization’s activities and functions.

That real-world database design often requires the reconciliation of conflicting goals

No matter how well database designers are able to produce designs that adhere to all applicable modeling conventions, they are often forced to make design compromises. These compromises are required when end users have vital transaction speed and/or information requirements that prevent the use of “perfect” modeling logic and adherence to all modeling conventions. Therefore, database designers must use their professional judgment to decide how and to what extent the modeling conventions are subject to modification. To ascertain that their professional judgments are sound, database designers must have detailed and in-depth knowledge of data-modeling conventions. Documenting the design process from start to finish is also essential. This helps keep the design process on track and allows for easy modifications down the road.

**Week 3: Readings of Chapter 5**

What normalization is and what role it plays in the database design process

Normalization is a process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the probability of data anomalies. The normalization process involves assigning attributes to tables based on the concept of determination. Normalization works through a series of stages called normal forms. The first three stages are defined as first normal form (1NF), second normal form (2NF), and third normal form (3NF). From a structural point of view, 2NF is better than 1NF and 3NF is better than 2NF. For most business database design purposes, 3NF is as high as you need to go in the normalization process. The role of normalization in database design is to identify and improve on any weaknesses that may exist in the table structures produced in entity relational (ER) modeling.

About the normal forms 1NF, 2NF, 3NF, BCNF, and 4NF

Partial dependencies should exist in the table but it may still have transitive dependencies. These transitive dependencies are eliminated in 3NF. The Boyce-Codd normal form (BCNF) is merely a special 3NF case in which all determinant keys are candidate keys. A table in 3NF may contain multivalued dependencies that produce either numerous null values or redundant data. Therefore, it may be necessary to alter a 3NF table to the fourth normal form (4NF) by splitting the table to remove the multivalued dependencies.

How normal forms can be transformed from lower normal forms to higher normal forms

Of 3NF, the Boyce-Codd normal form (BCNF) is merely a special 3NF case in which all determinants keys are candidate keys. 4NF ensures that the tables conform to two basic rules. One is that all attributes must be dependent on the primary key, but they must be independent of each other. The other rule is that no row may contain two or more multivalued facts about an entity.

That normalization and ER modeling are used concurrently to produce a good database design

To ensure good database design, normalization should be part of the design process in which tables are produced in the ER model and refined in the normalization process. First, an ERD is created through an iterative process. Relevant entities, their attributes, and their relationships are defined during this stage. Second, normalization focuses on the characteristics of specific entities; that is, normalization represents a micro view of the entities within the ERD. Therefore, separating the normalization process from the ER modeling process is difficult; the two practices are used in an iterative and incremental process.

That some situations require denormalization to generate information efficiently

The larger the number of tables, the more additional I/O operations become necessary to join them and the greater the amount of processing logic. Therefore, tables are sometimes denormalized to yield less I/O in order to increase processing speed. With larger tables, you pay for the increased processing speed by making the data updates less efficiently. Denormalization also sacrifices good indexing practices and introduces data redundancies that are likely to result in data anomalies. In the design of production databases, use denormalization sparingly and cautiously.

**Week 3: Readings of Chapter 6**

About the extended entity relationship (EER) model's main constructs

The extended entity relationship model (EERM) is the result of adding more semantic constructs to the original entity relationship (ER) model. The EER model constructs include entity supertypes, entity subtypes, and entity clustering. In modeling terms, an entity supertype is a generic entity type that is related to one or more entity subtypes, where the entity supertype contains the common characteristics and the entity subtypes contain the unique characteristics of each entity subtype. You can use different approaches to develop entity supertypes and subtypes. Specialization is the top-down process of identifying lower-level, more specific entity subtypes from a higher-level entity supertype. Generalization is the bottom-up process of identifying a higher-level, more generic entity supertype from lower-level entity subtypes. The property of inheritance allows an entity subtype to inherit the attributes and relationships of the supertype. A subtype discriminator is the attribute in the supertype entity that determines to which entity subtype each supertype occurrence is related. An entity cluster is a “virtual” entity type used to represent multiple entities and relationships in the ERD.

How entity clusters are used to represent multiple entities and relationships

Developing an ER diagram involves the discovery of probably hundreds of entity types and their respective relationships. As a design nears completion, the ERD will contain hundreds of entities and relationships that crowd the diagram to the point of making it unreadable and inefficient as a communication tool. In such cases, entity clusters are used to minimize the number of entities shown in the ERD. An entity cluster is formed by consolidating multiple interrelated entities into a single abstract entity object. An entity cluster is considered “virtual” or “abstract” in the sense that it is not actually an entity in the final ERD. It is a temporary entity used to depict multiple entities and relationships, with the purpose of simplifying the ERD and thus improving its readability.

The characteristics of good primary keys and how to select them

A primary key’s function is to guarantee entity integrity. In addition, primary keys and foreign keys work together to implement relationships in the relational model. Therefore, the importance of properly choosing the primary key has a direct bearing on the efficiency and effectiveness of database implementation. Natural keys are identifiers that exist in the real world. Primary keys must have unique values. They should also be nonintelligent. Primary keys should not change over time. Preferably, primary keys should be numeric and consist of a single attribute.

How to use flexible solutions for special data modeling cases

Data modeling and design require skills that are acquired through experience. This experience is acquired through practice which involves regular and frequent repetition, and applying the concepts learned to specific and different design problems. A data modeler will find that certain data modeling scenarios require flexible solutions. In a 1:1 relationship, the PK of the mandatory entity should be placed in the optional entity as a foreign key. It should be placed in an entity that causes the least number of nulls, or where the role is played. In tables that contain time-variant data that must be retained and maintained, create an entity containing the new values, the dates of change, and any other time-relevant data. This entity would maintain a 1:M relationship with the entity for which the history is to be maintained.

What issues to check for when developing data models based on EER diagrams

The data modeling checklist enables a designer to check that the ERD meets a set of minimum requirements. The checklist includes items associated with the project's business rules and the data modeling guidelines that specify the handling of the entity and attributes' naming conventions. The checklist also itemizes the required features of the entities, attributes, and relationships associated with the database design project. Lastly, the checklist includes the standards that the ER model should meet.

**Week 4: Readings of Chapter 9**

That successful database design must reflect the information system of which the database is a part

An information system facilitates the conversion of data into information and the management of both data and information. Data collection, storage, and retrieval capabilities make up an integral part of the information system. A complete information system is composed of people, hardware, software, the database(s), application programs, and procedures. One of the factors that an information system depends on is database development. Typically, the term database development describes the process of database design and implementation. The primary objective in database design is to build complete, normalized, nonredundant (to the extent possible), and fully integrated conceptual, logical, and physical database models. The implementation stage includes creating the database storage structure, loading data into the database, and providing for data management. Planning, analyzing, and designing any size or type of database is essential. However, the procedures that would be used to design a small database do not exactly scale up to the procedures that would be needed to design a database for a large corporation or even a segment of such a corporation.

That successful information systems are developed within a framework known as the Systems Development Life Cycle (SDLC)

The Systems Development Life Cycle (SDLC) outlines the history (life cycle) of an information system. The SDLC provides the environment within which the database design and application development can be planned and evaluated. The traditional SDLC is classified into five phases: planning, analysis, detailed systems design, implementation, and maintenance. It is an iterative rather than a sequential process. During the planning phase, an initial assessment of the information-flow-and-extent requirements must be conducted. If it is determined that the system is necessary, a feasibility study is done to evaluate the technical aspects of hardware and software requirements and the system cost. During the analysis phase, a thorough audit of user requirements and the existing systems occur. In addition, this phase includes the creation of a logical systems design. In the detailed systems design stage, the designer completes the design of the system processes. The design includes all necessary technical specifications for the screens, menus, reports, and other devices that might be used to help make the system a more resourceful information generator. During the implementation phase, the hardware, DBMS software, and application programs are installed and the database design is rolled out. Maintenance is the next phase and it can be grouped into three main activities: corrective, adaptive, and perfective.

That within the information system, the most successful databases are subject to frequent evaluation and revision within a framework known as the Database Life Cycle (DBLC)

The Database Life Cycle (DBLC) fits into and resembles the Systems Development Life Cycle (SDLC). Also, an iterative process, the DBLC consists of six phases: the database initial study, database design, implementation and loading, testing and evaluation, operation, and maintenance and evolution. The overall purpose of the database initial study is to analyze the company situation, define problems and constraints, define objectives, and define the scope and boundaries. During the database design phase, the conceptual design is created. DBMS software is selected and the logical and physical designs are created during the database design phase. In the implementation and loading phase, the DBMS is installed. Other activities that occur during this stage are the creation of the databases and the loading or conversion of data. In the testing and evaluation stage, the validity of the data is tested. The database is also fine-tuned during this phase, and the database and its application programs are evaluated. Once the database has passed the evaluation phase, it is said to be operational. At that juncture, the database, its management, its users, and its application programs make up a complete information system. System changes and enhancements occur during the maintenance and evolution phase.

How to conduct evaluation and revision within the SDLC and DBLC frameworks

Revision of the original database design starts with a careful reevaluation of the entities, followed by a detailed examination of the attributes that describe those entities. A careful review of the rough database design blueprint is likely to lead to revisions. Those revisions will help ensure that the design is capable of meeting end-user requirements. Because real-world database design is generally done by teams, it is advisable to organize the design's key components into modules. A module is an information system component that handles a specific function. The modules and their sub-sections can be delegated to design groups within teams, greatly accelerate the development work. Modules simplify the design work. Each module contains a more manageable number of entities. The modules can be prototyped quickly. Implementation and applications programming trouble spots can be identified more readily. Quick prototyping is also a great confidence builder.

About database design strategies: top-down vs. bottom-up design and centralized vs. decentralized design

Top-down vs. bottom-up design and centralized vs. decentralized design are basic design philosophies. Top-down design begins by identifying the data sets, then defines the data elements for each of those sets. It is a process that involves the discovery of different entity types and the definition of each entity' attributes. Bottom-up design first identifies the data elements (items), then groups them together in data sets. These two general approaches can be influenced by factors such as the scope and size of the system, the company' management style, and the company' structure (centralized or decentralized). Centralized design is useful when the data component is composed of a relatively small number of objects and procedures. Decentralized design is productive when the data component of the system has a considerable number of entities and complex relations on which very complex operations are performed. Decentralized design is also likely to be employed when the problem itself is spread across several operational sites and each element is a subset of the entire data set.

**Week 4: Readings of Chapter 16**

That data is a valuable business asset that require careful management and how databases play a critical role in an organization

Data are a valuable resource that can translate into information. If the information is accurate and timely, it is likely to facilitate actions that improve the business' competitive position and increase wealth. As a result, data management is an essential activity for any organization and must be regarded as a corporate asset. An organization’s managerial structure may be divided into three levels: top, middle, and operational. Top-level management makes strategic decisions while middle management makes tactical decisions. Operational management is responsible for making daily operational decisions. Operational decisions tend to be short term and affect only daily operations. Tactical decisions tend to cover a longer time period and affect larger-scale operations. Strategic decisions are those that affect the long-term well-being of the company or even its survival. The DBMS is the most widely used electronic tool for corporate data management. It facilitates strategic, tactical, and operational decision making at all levels of the organization.

That the introduction of a DBMS has important technological, managerial, and cultural organizational consequence

The introduction of a DBMS signifies a big change throughout an organization. It is likely to have a significant impact that may be positive or negative depending on how it is managed. The introduction of a DBMS into an organization has been described as a process that includes technological, managerial, and cultural aspects. The technological aspect includes selecting, installing, configuring, and monitoring the DBMS to ensure that it handles data storage, access, and security efficiently. Thus, the professional or team in charge of addressing the technological aspect of the DBMS installation must have the technical skills necessary to provide or secure sufficient support for the different users of the DBMS. Therefore, database administration staffing is a major technological factor in the DBMS introduction. The managerial aspect of the introduction of a database system is crucial. Careful planning is required to create an appropriate organizational structure to accommodate the person or people charged with the responsibility of managing the DBMS. The administrative personnel must have superior interpersonal and communications skills combined with broad organizational and business understanding. Senior management must be committed to the new system and must define and support the data administration functions, goals, and roles within the organization. The cultural impact of the introduction of a database system must be evaluated carefully. The DBMS' existence is likely to have an impact on people, functions, and interactions. The database administration department must accommodate end users. They must be ready to listen to their concerns, act on those concerns whenever possible, and train end users about the system' uses and advantages.

What the database administrator's managerial and technical roles are

At the very least, the managerial services of a DBA include: Supporting the end-user community. Defining and enforcing policies, procedures, and standards for the database function. Ensuring data security, privacy, and integrity. Offering data backup and recovery services. Monitoring the use and distribution of the data in the database. A DBA's technical role includes some of the following activities: Assessing, selecting, and installing the DBMS. Designing and implementing databases and software programs. Testing and evaluating databases and applications. Operating the DBMS, utilities, and applications. Educating and supporting users. Maintaining the DBMS, utilities, and software programs.

About several database administration tools and strategies

The development of the data administration strategy is closely tied to the mission and objectives of the business. Thus, the establishment of an organization's strategic plan corresponds to that of data administration, demanding a detailed analysis of company goals, situation, and business needs. To lead the development of this overall plan, an integrating methodology is necessary. The most commonly used integrating methodology is known as information engineering (IE). IE facilitates the translation of the company's strategic goals into the data and applications that will help the company achieve those goals. It focuses on the description of the corporate data instead of the processes. To assist in translating strategic plans into operational plans, the DBA has access to a wide assortment of database administration tools. These tools include the data dictionary and computer-aided software engineering (CASE) tools.

How various database administration technical tasks are performed with Oracle

To perform any administrative task, the Oracle DBA has to connect to the database, using a username with administrative (DBA) privileges. By default, Oracle automatically creates SYSTEM and SYS user IDs that have administrative privileges with every new database you create. The preferred credentials for each database can be defined by clicking on the Preferences link at the top of the page and then clicking on Preferred Credentials. Finally, the target username is chosen under Set Credentials. To perform database storage management tasks such as creating and administering tablespaces and data files, the DBA uses the Enterprise Manager, Administration, and Storage options. Another essential aspect of managing a database is monitoring the database objects that were created in the database. The Oracle Enterprise Manager gives the DBA a graphical user interface to create, edit, view, and delete database objects in the database. An Oracle schema is a logical section of the database that belongs to a given user, and that schema is identified by the username. Within the schema, the user can create his/her own tables and other objects. The Schema Manager provides an organized view of all of the objects in the database schema. With this program, a DBA is able to create, edit, view, and delete tables, indexes, views, functions, triggers, procedures, and other specialized objects. One of the most common database administration activities is creating and managing database users. The Security section of the Oracle Enterprise Manager's Administration page allows the DBA to create users, roles, and profiles.

Overview

This chapter shows you the basis for a successful database administration strategy. Such a strategy requires that data be treated and managed as a valuable corporate asset.

In this chapter, you will learn about important data management issues by looking at the managerial and technical roles of the database administrator (DBA). This chapter also explores database security issues, such as the confidentiality, integrity, and availability of data. In our information-based society, a key aspect of data management is ensuring that data is protected against intentional or unintentional access by unauthorized personnel. It is also essential to ensure that data is available as needed, even in the face of natural disaster or hardware failure, and to maintain the integrity of the data in the database.

The chapter includes a discussion of database administration tools and the corporate- wide data architectural framework. You will also learn how database administration management fits within classical organizational structures. Furthermore, you will learn about several considerations when evaluating cloud-based data services. Even though many new types of databases have emerged, recent studies [Footnote] show that relational databases still dominate the market share of the enterprise. Therefore, with the preponderance of relational databases in the market, it is important that you learn about some basic database administration tasks in Oracle RDBMS. Similar tasks can be performed in all major databases, such as Microsoft SQL Server, IBM DB2, Oracle, MySQL, and so on.

Summary

* Data management is a critical activity for any organization, so data must be treated as a corporate asset. The value of a data set is measured by the utility of the information derived from it. Good data management is likely to produce good information, which is the basis for better decision making
* Data quality is a comprehensive approach to ensure the accuracy, validity, and timeliness of data. Data quality focuses on correcting dirty data, preventing future inaccuracies in the data, and building user confidence in the data
* The DBMS is the most commonly used tool for corporate data management. The DBMS supports strategic, tactical, and operational decision making at all levels of the organization. The introduction of a DBMS into an organization is a delicate job; the impact of the DBMS on the organization’s managerial and cultural framework must be carefully examined
* The database administrator (DBA) is responsible for managing the corporate database. The internal organization of database administration varies from company to company. Although no standard exists, it is common practice to divide DBA operations according to phases of the Database Life Cycle. Some companies have created a position with a broader mandate to manage computerized data and other data; this activity is handled by the data administrator (DA)
* The DA and DBA functions tend to overlap. Generally speaking, the DA has more managerial tasks than the more technically oriented DBA. Compared to the DBA function, the DA function is DBMS-independent, with a broader and longer-term focus. However, when the organization does not include a DA position, the DBA executes all of the DA’s functions. In this combined role, the DBA must have a diverse mix of technical and managerial skills
* A DBA’s managerial services include supporting end users; defining and enforcing policies, procedures, and standards for the database; ensuring data security, privacy, and integrity; providing data backup and recovery services; and monitoring distribution and use of the data in the database
* The DBA’s technical role requires involvement in at least the following activities: evaluating, selecting, and installing the DBMS; designing and implementing databases and applications; testing and evaluating databases and applications; operating and maintaining the DBMS, utilities, and applications; and training and supporting users
* Security refers to activities and measures that ensure the confidentiality, integrity, and availability of an information system and its main asset, data. A security policy is a collection of standards, policies, and practices that guarantee the security of a system and ensure auditing and compliance
* A security vulnerability is a weakness in a system component that could be exploited to allow unauthorized access or service disruption. A security threat is an imminent security violation caused by an unchecked vulnerability. Security vulnerabilities exist in all components of an information system: people, hardware, software, network, procedures, and data. Therefore, it is critical to have robust database security. Database security refers to DBMS features and related measures that comply with the organization’s security requirements
* The development of a data administration strategy is closely related to the company’s mission and objectives. Therefore, the strategic plan requires a detailed analysis of company goals, its situation, and its business needs. To guide the development of this data administration plan, an integrating methodology is required. The most commonly used integrating methodology is known as information engineering (IE)
* To help translate strategic plans into operational plans, the DBA has access to an arsenal of database administration tools, including a data dictionary and computer-aided systems engineering (CASE) tools
* With the introduction of reliable cloud-based data services, the role of the DBA has expanded beyond corporate walls

**Week 5: Readings of Chapter 10**

What a database transaction is and what its properties are

A transaction is any action that reads from and/or writes to a database. It may consist of a simple SELECT statement to generate a list of table contents or a series of related UPDATE statements to change the values of attributes in various tables. A transaction may even consist of a series of INSERT statements to add rows to one or more tables; or it may consist of a combination of SELECT, UPDATE, and INSERT statements. All transactions must display atomicity, consistency, isolation, durability, and serializability properties. Atomicity requires that all operations (SQL requests) of a transaction be completed; if not, the transaction is aborted. Consistency indicates the permanence of the database’s consistent state. When a transaction is completed, the database reaches a consistent state. Isolation means that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed. Durability ensures that once transaction changes are done (committed), they cannot be undone or lost, even in the event of a system failure. Serializability ensures that the concurrent execution of several transactions yields consistent results.

What concurrency control is and what role it plays in maintaining the database's integrity

The coordination of the simultaneous execution of transactions in a multiuser database system is referred to as concurrency control. The aim of concurrency control is to ensure the serializability of transactions in a multiuser database environment. Concurrency control is essential because the simultaneous execution of transactions over a shared database can create several data integrity and consistency problems. The three primary problems are lost updates, uncommitted data, and inconsistent retrievals. The scheduler is a special DBMS program that establishes the order in which the operations within concurrent transactions are executed. The transaction execution order is vital and ensures database integrity in multiuser database systems. Locking, time stamping, and optimistic approaches are used by the scheduler to ensure the serializability of transactions.

What locking methods are and how they work

A lock guarantees exclusive use of a data item to a current transaction. A transaction acquires a lock prior to data access; the lock is released (unlocked) when the transaction is completed so that another transaction can lock the data item for its own use. Lock granularity depicts the level of lock use. Locking can take place at the database, table, page, row, or even field (attribute) levels. Two types of locks can be used in database systems, namely, binary locks and shared/exclusive locks. A binary lock can have only two states: 1 (locked) or 0 (unlocked). An exclusive lock is used when a transaction wants to update (write to) the database and no other locks (shared or exclusive) are held on the data. A shared lock is issued when a transaction wants to read data from a database and no other transaction is updating the same data. Serializability of schedules is guaranteed through the use of two-phase locking. The two-phase locking schema has a growing phase, in which the transaction acquires all of the locks that it needs without unlocking any data, and a shrinking phase, in which the transaction releases all of the locks without acquiring new locks. When two or more transactions wait indefinitely for each other to release a lock, they are said to be in a deadlock, or a deadly embrace. Prevention, detection, and avoidance are examples of deadlock control techniques.

How stamping methods are used for concurrency control

Concurrency control with time stamping techniques assigns a unique time stamp to each transaction and schedules the execution of conflicting transactions in time stamp order. There are two schemes that are used to decide which transaction is rolled back and which continues executing. These include the wait/die scheme and the wound/wait scheme. In the wait/die scheme, the older transaction waits and the younger is rolled back and rescheduled. In the wound/wait scheme, the older transaction rolls back the younger transaction and reschedules it. In both schemes, one of the transactions waits for the other transaction to finish and release the locks. In numerous situations, a transaction requests multiple lock. In cases whereby some transactions wait indefinitely, deadlocks may occur. To prevent these types of deadlocks, each lock request has an associated time-out value. If the lock is not granted before the time-out expires, the transaction is rolled back.

How optimistic methods are used for concurrency control

The optimistic approach to concurrency control is based on the premise that the majority of the database operations do not conflict. This method does not require locking or time stamping techniques. Instead, a transaction is executed without restrictions until it is committed. Using an optimistic approach, each transaction moves through two or three phases which are described as read, validation, and write. During the read phase, the transaction reads the database, implements the needed computations, and makes the updates to a private copy of the database values. All update operations of the transaction are recorded in a temporary update file, which is not accessed by the remaining transactions. During the validation phase, the transaction is authenticated to ensure that the changes made will not affect the integrity and consistency of the database. If the validation test is positive, the transaction goes to the write phase. If the validation test is negative, the transaction is restarted and the changes are abandoned. In the write phase, the changes are permanently saved to the database. The optimistic approach is acceptable for most read or query database systems that require few update transactions.

How database recovery management is used to maintain database integrity

Database recovery restores a database from a given state, typically inconsistent, to a previously consistent state. Recovery techniques are based on the atomic transaction property. Database backups are permanent copies of the database. Examples of database backups are full, differential, or transaction log backups. The backups are stored in a safe place and are to be used in case of a critical error in the master database. Transaction recovery procedures usually implement deferred-write and write-through techniques. When the recovery procedure uses deferred write or deferred update, the transaction operations do not immediately update the physical database. Instead, only the transaction log is updated. When the recovery procedure uses write-through or immediate update, the database is immediately updated by transaction operations during the transaction’s execution, even before the transaction reaches its commit point. If the transaction aborts before it reaches its commit point, a ROLLBACK or undo operation needs to be done to restore the database to a consistent state. In that scenario, the ROLLBACK operation will utilize the transaction log “before” values.

**Week 5: Readings of Chapter 13**

How operational data and decision support data differ

Operational data and decision support system (DSS) data perform different functions. From the end-user's perspective, DSS data differ from operational data in three main areas: timespan, granularity, and dimensionality. Operational data cover a short time frame. In contrast, DSS data tend to cover a longer time frame. DSS data is presented at different levels of aggregation, from highly summarized to almost atomic. Operational data concentrate on representing individual transactions rather than on the effects of the transactions over time. From the DSS designer' point of view, operational data represent transactions as they happen in real time, while DSS data are a snapshot of the operational data at a given point in time. Operational data are commonly stored in many tables, and the stored data represent the information about a given transaction only. DSS data are typically stored in a few tables that store data derived from the operational data. The data models that govern operational data and DSS data are different. Most operational data are stored in a relational database in which the structures (tables) tend to be highly normalized. DSS databases tend to be non-normalized and include few tables, each of which includes a large number of attributes. The operational database's frequent and rapid data updates make data anomalies a potentially devastating problem. Conversely, DSS databases are not subject to such transaction updates, and the focus is on querying capability. Query activity (frequency and complexity) in the operational database is inclined to be low to allow additional processing cycles for the more crucial update transactions. In contrast, DSS data exist for the sole purpose of serving query requirements. Finally, DSS data are characterized by very large amounts of data.

What a data warehouse is, how data for it are prepared, and how it is implemented

The data warehouse is an integrated, subject-oriented, time-variant, nonvolatile collection of data that facilitates decision making. A complete data warehouse architecture includes support for a DSS data store, a data extraction and integration filter, and a specialized presentation interface. To be productive, a data warehouse must conform to uniform structures and formats to avoid data conflicts and to facilitate decision making. Data integration implies that all business entities, data elements, data characteristics, and business metrics are described in the same way throughout the enterprise. Data warehouse data are arranged and optimized to offer answers to questions coming from diverse functional areas within a company. Because data in a data warehouse makes up a snapshot of the company history as measured by its variables, the time component is essential. The data warehouse contains a time ID that is used to generate summaries and aggregations by a particular time period. Once the data enter the data warehouse, the time ID assigned to the data cannot be changed. The implementation of a data warehouse involves several steps. These include the initial gathering of data, design and mapping, loading and testing, building and testing, and finally rollout and gathering feedback.

What star schemas are and how they are constructed

The star schema is a data modeling technique used to map multidimensional decision support data into a relational database. Essentially, the star schema creates the near equivalent of a multidimensional database schema from the existing relational database. Star schemas yield an easily implemented model for multidimensional data analysis while still preserving the relational structures on which the operational database is built. The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies. Facts and dimensions are normally represented by physical tables in the data warehouse database. The fact table is related to each dimension table in a many-to-one (M:1) relationship. Fact and dimension tables are related by foreign keys and are subject to the familiar primary key/foreign key constraints. In a typical star schema, each dimension record is related to thousands of fact records. Dimensional tables are normalized to achieve semantic simplicity and facilitate end-user navigation through the dimensions. By normalizing the dimension tables, data-filtering operations related to the dimensions are simplified.

What data mining is and what role it plays in decision support

Data mining automates the analysis of operational data with the aim of discovering previously unknown data characteristics, relationships, dependencies, and/or trends. The data mining process consists of four phases: data preparation, data analysis and classification, knowledge acquisition, and prognosis. In the data preparation phase, the main data sets to be used by the data mining operation are identified and any known data issues are removed. The data analysis and classification phase studies the data to identify common data characteristics or patterns. During this phase, the data mining tool applies specific algorithms to find the following: data groupings, classifications, clusters, or sequences; data dependencies, links, or relationships; and data patterns, trends, and deviations. The knowledge-acquisition phase uses the results of the data analysis and classification phase. During the knowledge-acquisition phase, the data mining tool (with possible intervention by the end user) selects the appropriate modeling or knowledge-acquisition algorithms. The most common algorithms used in data mining are based on neural networks, decision trees, rules induction, genetic algorithms, classification and regression trees, memory-based reasoning, and nearest neighbor and data visualization. A data mining tool may utilize many of these algorithms in any combination to generate a computer model that reflects the behavior of the target data set. Although many data mining tools stop at the knowledge-acquisition phase, others continue to the prognosis phase. In that phase, the data mining findings are used to predict future behavior and forecast business outcomes.

What online analytical processing (OLAP) is?

Online analytical processing (OLAP) describes an advanced data analysis environment that supports decision making, business modeling, and operations research. OLAP operational characteristics can be divided into three main modules. These include a graphical user interface (GUI), analytical processing logic, and data-processing logic. OLAP systems have four main characteristics: use of multidimensional data analysis techniques, advanced database support, easy-to-use end-user interfaces, and client/server architecture. In multidimensional analysis, data are processed and viewed as part of a multidimensional structure. This type of data analysis is particularly valuable to business decision makers because they tend to see business data as data that are related to other business data. Advanced OLAP features become more useful when access to them is kept simple. OLAP tool vendors have equipped their sophisticated data extraction and analysis tools with easy-to-use graphical interfaces. Many of the interface features are from previous generations of data analysis tools that are already familiar to end users. This familiarity makes OLAP easily accepted and readily used. Client/server architecture provides a framework within which new systems can be designed, developed, and implemented. The client/server environment enables an OLAP system to be divided into several components that define its architecture. Those components can then be placed on the same computer, or they can be distributed among several computers. Therefore, OLAP is designed to meet ease-of-use as well as system flexibility requirements.

How SQL extensions are used to support OLAP-type data manipulations

The increasing use of OLAP tools has brought about the development of SQL extensions to support multidimensional data analysis. Most SQL innovations are the result of vendor-related product enhancements, and many of the innovations have made their way into standard SQL. Note that a database is at the core of all data warehouses. Therefore, all SQL commands (such as CREATE, INSERT, UPDATE, DELETE, and SELECT) will work in the data warehouse as expected. However, most queries that are run in a data warehouse are inclined to include a lot of data groupings and aggregations over multiple columns. For example, the two extensions to the GROUP BY clause are particularly useful: ROLLUP and CUBE. The ROLLUP extension is used with the GROUP BY clause to generate aggregates by different dimensions. The ROLLUP extension is particularly useful when you want to get multiple nested subtotals for a dimension hierarchy. The CUBE extension will allow you to obtain a subtotal for each column listed in the expression, in addition to a grand total for the last column listed. The CUBE extension is particularly helpful in computing all possible subtotals within groupings based on multiple dimensions. Cross-tabulations are especially good candidates for application of the CUBE extension. To save query processing time, most database vendors have implemented additional “functionality” to manage aggregate summaries more efficiently. This new functionality resembles the standard SQL views for which the SQL code is predefined in the database. The difference in the added functionality is that the views also store the pre-aggregated rows, somewhat like a summary table.

**Week 6: Readings of Chapter 15**

About the different database connectivity technologies

Database connectivity refers to the means through which application programs connect and communicate with data repositories. Database connectivity software is also known as database middleware because it interfaces between the application program and the database. Examples of database connectivity technologies include the following: native SQL connectivity, Microsoft's Open Database Connectivity (ODBC), Data Access Objects (DAO) and Remote Data Objects (RDO), Microsoft's Object Linking and Embedding for Database (OLE-DB), and Microsoft's ActiveX Data Objects (ADO.NET). Native database connectivity applies to the connection interface that is provided by the database vendor and is unique to that vendor. ODBC is Microsoft's implementation of a superset of the SQL Access Group Call Level Interface (CLI) standard for database access. ODBC allows any Windows application to access relational data sources using standard SQL. DAO is an object-oriented API used to access MS Access, MS FoxPro, and dBase databases (using the Jet data engine) from Visual Basic programs. RDO is a higher-level object-oriented application interface used to access remote database servers. RDO utilizes the lower-level DAO and ODBC for direct access to databases. RDO was optimized to deal with server-based databases, such as MS SQL Server and Oracle.

How Web-to-database middleware is used to integrate databases with the Internet

Database access through the Web is accomplished through database-to-middleware software. Its main purpose is to provide Web access to other external services such as databases, e-mail systems, directory services, and FTP services. To support database queries, the Web server's capability must be extended so it can understand and process database requests. This job is done through a server-side extension. A server-side extension is a program that interacts directly with the Web server to handle specific types of requests. A database server-side extension program is also known as Web-to-database middleware. The interaction between the browser, the Web server, and the Web-to-database middleware consists of the following activities: The client browser sends a page request to the Web server. The Web server receives and validates the request. Generally, the requested page contains some type of scripting language to enable the database interaction. The Web-to-database middleware reads, validates, and executes the script. The database server executes the query and passes the result back to the Web-to-database middleware. The Web-to-database middleware compiles the result set, dynamically generates an HTML-formatted page that includes the data retrieved from the database, and sends it to the Web server. The Web server returns the just-created HTML page, which now includes the query result, to the client browser. The client browser displays the page on the local computer. The interaction between the Web server and the Web-to-database middleware is crucial to the development of a successful Internet database implementation. Therefore, the middleware must be well integrated with the other Internet services and the components that are involved in its use.

The first steps in building a Web-based database front end using Macromedia ColdFusion

ColdFusion is a fully developed Web application server that offers connections to databases, e-mail systems, directory systems, and search engines. To perform its functions, ColdFusion offers a server-side markup language (HTML extensions, or tags) referred to as the ColdFusion Markup Language (CFML), which is used to create ColdFusion application pages known as scripts. A script is a series of instructions executed in interpreter mode. ColdFusion scripts contain the code that is required to connect, query, and update a database from a Web front end. ColdFusion scripts contain a combination of HTML, ColdFusion tags, and when necessary, Java, JavaScript, or VBScript. ColdFusion tags start with (begin a query) and (end a query). ColdFusion scripts are saved in files with a .cfm extension. When a client browser requests a .cfm page from the Web server, the ColdFusion application server executes the .cfm script instructions and sends the resulting output (in HTML format) to the Web server. The Web server then sends the document to the client computer for display.

What Extensible Markup Language (XML) is and why it is important for Web database development

Most e-commerce transactions occur between businesses (business-to-business or B2B). Because B2B e-commerce integrates business processes among companies, the transfer of business information among different business entities is critical. However, the way in which businesses represent, identify, and use data tends to differ substantially from company to company. Extensible Markup Language (XML) enables the exchange of B2B and other data over the Internet. XML provides the semantics that facilitates the exchange, sharing, and manipulation of structured documents across organizational boundaries. With XML, organizations of all kinds can exchange data in XML-formatted documents. XML produces the description and the representation of data, thus facilitating data manipulation in ways that were not possible before XML. Just like HTML, XML document is a text file. It allows the definition of new tags to describe data elements. XML is case sensitive and XML tags must be well formed; that is, each opening tag has a corresponding closing tag. XML is not a new version or replacement for HTML. It is concerned with the description and representation of the data, rather than the way the data are displayed. Data display is still the function of HTML. Basically, XML and HTML perform complementary, rather than overlapping, functions.

Chapter 15 Overview

Databases are the central repository for critical data generated by business applications, including newer channels such as the web and mobile devices. For businesses to remain competitive, such data must be readily available, anywhere and anytime, to all business users and in all types of formats: a desktop spreadsheet, a Visual Basic application, a web front end, and using newer technologies such as smartphones and tablets. In this chapter, you will learn about various architectures used to connect applications to databases.

The Internet has changed how organizations of all types operate. Buying goods and services via the Internet has become commonplace. This chapter examines the fundamentals of web database technologies used to open databases to the Internet. In today’s environment, interconnectivity occurs not only between an application and the database but between applications exchanging messages and data. Extensible Markup Language (XML) provides a standard way of exchanging unstructured and structured data between applications.

Companies that want to integrate database and web technologies within their applications portfolio can now choose from a range of Internet-based services. Therefore, you will learn how organizations can benefit from cloud computing by leveraging the database-as-a-service model within their IT environments. These cloud-based services offer a quick and cost-efficient way to provide new business services.

Database connectivity refers to the mechanisms through which application programs connect and communicate with data repositories. Databases store data in persistent storage structures so it can be retrieved at a later time for processing. As you already learned, the database management system (DBMS) functions as an intermediary between the data (stored in the database) and the end-user’s applications. Before learning about the various data connectivity options, it is important to review some important fundamentals you have learned in this book:

* DBMSs provide means to interact with the data in their databases. This could be in the form of administrative tools and data manipulation tools. DBMSs also provide a proprietary way for external application programs to connect to the database by the means of an application programing interface. See Chapter 1, Database Systems
* Modern DBMSs have the option to store data locally or distributed in multiple locations. Locally stored data resides in the same processing host as the DBMS. A distributed database stores data in multiple geographically distributed nodes with data management capability. See Chapter 12, Distributed Database Management Systems
* The database connectivity software we discuss in this chapter supports Structured Query Language (SQL) as the standard data manipulation language. However, depending on the type of database model, some database connectivity interfaces may support other proprietary data manipulation languages
* Database connectivity software works in a client/server architecture, by which processing tasks are split among multiple software layers. In this model, the multiple layers exchange control messages and data. See Chapter 12 and Appendix F, Client/Server Systems, for more information on this topic

To better understand database connectivity software, we use client/server concepts in which an application is broken down in interconnected functional layers. In the case of database connectivity software, you could break down its basic functionality into three broad layers:

* A data layer where the data resides. You could think of this layer as the actual data repository interface. This layer resides closest to the database itself and normally is provided by the DBMS vendor
* A middle layer that manages multiple connectivity and data transformation issues. This layer is in charge of dealing with data logic issues, data transformations, ways to “talk” to the database below it, and so on. This would also include translating multiple data manipulation languages to the native language supported by the specific data repository
* A top layer that interfaces with the actual external application. This mostly comes in the form of an application programming interface that publishes specific protocols for the external programs to interact with the data

Chapter 15 Summary

* Database connectivity refers to the mechanisms through which application programs connect and communicate with data repositories. Database connectivity software is also known as database middleware
* Microsoft database connectivity interfaces are dominant players in the market and enjoy the support of most database vendors. ODBC, OLE-DB, and ADO.NET form the backbone of Microsoft’s Universal Data Access (UDA) architecture
* Native database connectivity refers to the connection interface that is provided by the database vendor and is unique to that vendor. ODBC is probably the most widely supported database connectivity interface. ODBC allows any Windows application to access relational data sources using standard SQL. Data Access Objects (DAO) is an older, object-oriented application interface. Remote Data Objects (RDO) is a higher-level, object-oriented application interface used to access remote database servers. RDO was optimized to deal with server-based databases such as MS SQL Server and Oracle
* Object Linking and Embedding for Database (OLE-DB) is database middleware developed with the goal of adding object-oriented functionality for access to relational and nonrelational data. ActiveX Data Objects (ADO) provides a high-level, application-oriented interface to interact with OLE-DB, DAO, and RDO. Based on ADO, ADO.NET is the data access component of Microsoft’s .NET application development framework. Java Database Connectivity (JDBC) is the standard way to interface Java applications with data sources
* Database access through the web is achieved through middleware. To improve the capabilities on the client side of the web browser, you must use plug-ins and other client-side extensions such as Java and JavaScript, or ActiveX and VBScript. On the server side, web application servers are middleware that expand the functionality of web servers by linking them to a wide range of services, such as databases, directory systems, and search engines
* Extensible Markup Language (XML) facilitates the exchange of B2B and other data over the Internet. XML provides the semantics that facilitate the exchange, sharing, and manipulation of structured documents across organizational boundaries. XML produces the description and the representation of data, thus setting the stage for data manipulation in ways that were not possible before. XML documents can be validated through the use of document type definition (DTD) documents and XML schema definition (XSD) documents
* Cloud computing is a computing model that provides ubiquitous, on-demand access to a shared pool of configurable resources that can be rapidly provisioned
* SQL data services (SDS) refers to a cloud computing-based data management service that provides relational data storage, ubiquitous access, and local management to companies of all sizes. This service enables rapid application development for businesses with limited information technology resources. SDS allows rapid deployment of business solutions using standard protocols and common programming interfaces