**1. Chapter 6**

 This chapter is one of the most critical in the book because here you will learn how the concepts you learned earlier all work together in database design to model a data-base that meets data integrity constraints as well as user reporting and performance requirements. These concepts include: • Identifying business rules • Identifying and defining business and data constraints • Defining functional dependencies • Identifying entities and relationships • Eliminating multivalued attribute

Unfortunately, the data structure depicted in Figure 6.1 does not conform to the rela-tional table requirements discussed in Chapter 3 (see Table 3.1) and therefore is not suitable to handle data updates well. Consider the following deficiencies: • The data structure invites data inconsistencies. For example, the JOB\_CLASS value “Elect. Engineer” might be entered as “Elect.Eng.” in some cases, “El. Eng.” in others, and “EE” in still others. The structure would allow John G. News and Alice K. Johnson in the Ever-green project to charge different rates even though they have the same job classification. • The data structure contains several multivalued attributes that make data management tasks very difficult. Because all of the employees working on a project are in a single cell, it is hard to identify each employee individually and for the database to answer questions such as “How many employees are working on the Starlight project?” • Employee data is redundant in the table because employees can work on multiple projects. Adding, updating, and deleting data are likely to be very cumbersome using this structure. For example, changing the job classification for Alice K. Johnson would require updating at least two rows.

The objective of normalization is to ensure that each table conforms to the concept of well-formed relations—in other words, tables that have the following characteristics: • Each relation (table) represents a single subject. For example, a COURSE table will contain only data that directly pertain to courses. Similarly, a STUDENT table will contain only student data. • Each row/column intersection contains only one value and not a group of values. • No data item will be unnecessarily stored in more than one table (tables have mini-mum controlled redundancy). The reason for this requirement is to ensure that the data is updated in only one place. • All nonprime attributes in a relation (table) are dependent on the primary key—the entire primary key and nothing but the primary key. The reason for this requirement is to ensure that the data is uniquely identifiable by a primary key value. • Each relation (table) has no insertion, update, or deletion anomalies, which ensures the integrity and consistency of the data.

Conversion to First Normal Form (1NF) : Step 1: Eliminate the Repeating Groups. Step 2: Identify the Primary Key . Step 3: Identify All Dependencies

As you examine Figure 6.3, note the following features of a dependency diagram: 1. The primary key attributes are bold, underlined, and in a different color. 2. The arrows above the attributes indicate all desirable dependencies—that is, dependencies based on the primary key. In this case, note that the entity’s attributes are dependent on the combination of PROJ\_NUM and EMP\_NUM. 3. The arrows below the dependency diagram indicate less desirable dependencies. Two types of such dependencies exist: a. Partial dependencies. You need to know only the PROJ\_NUM to determine the PROJ\_NAME; that is, the PROJ\_NAME is dependent on only part of the primary key. Also, you need to know only the EMP\_NUM to find the EMP\_NAME, the JOB\_CLASS, and the CHG\_HOUR. A dependency based on only a part of a com-posite primary key is a partial dependency. b. Transitive dependencies. Note that CHG\_HOUR is dependent on JOB\_CLASS. Because neither CHG\_HOUR nor JOB\_CLASS is a prime attribute—that is, nei-ther attribute is at least part of a key—the condition is indicative of a transitive dependency. In other words, a transitive dependency exists when a functional dependency exists only among nonprime attributes. Transitive dependencies yield data anomalies.

Our example still has the following anomalies: a. Update anomalies. Modifying the JOB\_CLASS for employee Annelise Jones requires updating many entries; otherwise, it will generate data inconsistencies. b. Insertion anomalies. Adding a new employee requires the employee to be assigned to a project and therefore to enter duplicate project information. If the employee is not yet assigned to a project, a phantom project must be created to complete the employee data entry. c. Deletion anomalies. Suppose that only one employee is associated with a given project. If that employee is deleted, the project information will also be deleted. The data redundancies occur because every row entry requires duplication of data. Such duplication of effort is very inefficient, and it helps create data anomalies; nothing prevents the user from typing slightly different versions of the employee name, position, or hourly pay. For instance, the employee name for EMP\_NUM = 102 might be entered as Dave Senior or D. Senior. The project name might also be entered correctly as Ever-green or misspelled as Evergeen. Such data anomalies violate the relational database’s integrity and consistency rules.

Conversion to Second Normal Form (2NF): Step 1: Make New Tables to Eliminate Partial Dependencies. Step 2: Reassign Corresponding Dependent Attributes.

Conversion to Third Normal Form (3NF): Step 1: Make New Tables to Eliminate Transitive Dependencies. Step 2: Reassign Corresponding Dependent Attributes.

Improving the design: evaluate pk assignments, evaluate naming conventions, refine attribute atomicity, identify new attributes, identify new relationships, refine primary keys as required for data granularity, maintain historical accuracy, evaluate using derived attributes

Table 6.5 reflects the following conditions: • Each CLASS\_CODE identifies a class uniquely. This condition illustrates the case in which a course might generate many classes. For example, a course labeled INFS 420 might be taught in two classes (sections), each identified by a unique code to facilitate registration. Thus, the CLASS\_CODE 32456 might identify INFS 420, class section 1, while the CLASS\_CODE 32457 might identify INFS 420, class section 2. Or, the CLASS\_CODE 28458 might identify QM 362, class section 5. • A student can take many classes. Note, for example, that student 125 has taken both 21334 and 32456, earning the grades A and C, respectively. • A staff member can teach many classes, but each class is taught by only one staff member. Note that staff member 20 teaches the classes identified as 32456 and 28458.

Specifically, the discussion of 4NF is largely academic if you make sure that your tables conform to the following two rules: 1. All attributes must be dependent on the primary key, but they must be independent of each other. 2. No row may contain two or more multivalued facts about an entity.

To understand the proper role of normalization in the design process, you should reexamine the operations of the contracting company whose tables were normalized in the preceding sections. Those operations can be summarized by using the following business rules: • The company manages many projects. • Each project requires the services of many employees. • An employee may be assigned to several different projects. • Some employees are not assigned to a project and perform duties not specifically related to a project. Some employees are part of a labor pool, to be shared by all proj-ect teams. For example, the company’s executive secretary would not be assigned to any one particular project. • Each employee has a single primary job classification, which determines the hourly billing rate. • Many employees can have the same job classification. For example, the company employs more than one electrical engineer. Given that simple description of the company’s operations, two entities and their attributes are initially defined: • PROJECT (PROJ\_NUM, PROJ\_NAME) • EMPLOYEE (EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, JOB\_ DESCRIPTION, JOB\_CHG\_HOUR) Those two entities constitute the initial ERD shown in Figure 6.13. After creating the initial ERD shown in Figure 6.13, the normal forms are defined: • PROJECT is in 3NF and needs no modification at this point. • EMPLOYEE requires additional scrutiny. The JOB\_DESCRIPTION attribute defines job classifications such as Systems Analyst, Database Designer, and Programmer. In turn, those classifications determine the billing rate, JOB\_CHG\_HOUR. Therefore, EMPLOYEE contains a transitive dependency. The removal of EMPLOYEE’s transitive dependency yields three entities: • PROJECT (PROJ\_NUM, PROJ\_NAME) • EMPLOYEE (EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, JOB\_CODE) • JOB (JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR) Because the normalization process yields an additional entity (JOB), the initial ERD is modified as shown in Figure 6.14.

As shown in the faculty evaluation report, the conflicts between design efficiency, information requirements, and performance are often resolved through compromises that may include denormalization. In this case, and assuming there is enough storage space, the designer’s choices could be narrowed down to: • Store the data in a permanent denormalized table. This is not the recommended solution because the denormalized table is subject to data anomalies (insert, update, and delete). This solution is viable only if performance is an issue. • Create a temporary denormalized table from the permanent normalized table(s). The denormalized table exists only as long as it takes to generate the report; it disap-pears after the report is produced. Therefore, there are no data anomaly problems. This solution is practical only if performance is not an issue and there are no other viable processing options. As shown, normalization purity is often difficult to sustain in the modern database environment.

You will discover that the data warehouse routinely uses 2NF structures in its complex, multilevel, multisource data environment. Although normalization is very important, especially in the so-called production database environment, 2NF is no lon-ger disregarded as it once was. Although 2NF tables cannot always be avoided, the problem of working with tables that contain partial and/or transitive dependencies in a production database environment should not be minimized. Aside from the possibility of troublesome data anomalies being created, unnormalized tables in a production database tend to suffer from these defects: • Data updates are less efficient because programs that read and update tables must deal with larger tables. • Indexing is more cumbersome. It is simply not practical to build all of the indexes required for the many attributes that might be located in a single unnormalized table. • Unnormalized tables yield no simple strategies for creating virtual tables known as views. You will learn how to create and use views in Chapter 8, Advanced SQL. Remember that good design cannot be created in the application programs that use a data-base.

TABlE 6.7 DATA-moDeliNG checKliST buSiNeSS RuleS • Properly document and verify all business rules with the end users. • Ensure that all business rules are written precisely, clearly, and simply. The business rules must help identify entities, attributes, relationships, and constraints. • Identify the source of all business rules, and ensure that each business rule is justified, dated, and signed off by an approving authority. DATA moDeliNG Naming conventions: All names should be limited in length (database-dependent size). • Entity names: • Should be nouns that are familiar to business and should be short and meaningful • Should document abbreviations, synonyms, and aliases for each entity • Should be unique within the model • For composite entities, may include a combination of abbreviated names of the entities linked through the composite entity • Attribute names: • Should be unique within the entity • Should use the entity abbreviation as a prefix • Should be descriptive of the characteristic • Should use suffixes such as \_ID, \_NUM, or \_CODE for the PK attribute • Should not be a reserved word • Should not contain spaces or special characters such as @, !, or & • Relationship names: • Should be active or passive verbs that clearly indicate the nature of the relationship entities: • Each entity should represent a single subject. • Each entity should represent a set of distinguishable entity instances. • All entities should be in 3NF or higher. Any entities below 3NF should be justified. • The granularity of the entity instance should be clearly defined. • The PK should be clearly defined and support the selected data granularity. Attributes: • Should be simple and single-valued (atomic data) • Should document default values, constraints, synonyms, and aliases • Derived attributes should be clearly identified and include source(s) • Should not be redundant unless this is required for transaction accuracy, performance, or maintaining a history • Nonkey attributes must be fully dependent on the PK attribute Relationships: • Should clearly identify relationship participants • Should clearly define participation, connectivity, and document cardinality eR model: • Should be validated against expected processes: inserts, updates, and deletions • Should evaluate where, when, and how to maintain a history • Should not contain redundant relationships except as required (see attributes) • Should minimize data redundancy to ensure single-place updates • Should conform to the minimal data rule: All that is needed is there, and all that is there is needed

Summary • Normalization is a technique used to design tables in which data redundancies are minimized. The first three normal forms (1NF, 2NF, and 3NF) are the most com-mon. From a structural point of view, higher normal forms are better than lower normal forms because higher normal forms yield relatively fewer data redundan-cies in the database. Almost all business designs use 3NF as the ideal normal form. A special, more restricted 3NF known as Boyce-Codd normal form, or BCNF, is also used. • A table is in 1NF when all key attributes are defined and all remaining attributes are dependent on the primary key. However, a table in 1NF can still contain both par-tial and transitive dependencies. A partial dependency is one in which an attribute is functionally dependent on only a part of a multiattribute primary key. A transi-tive dependency is one in which an attribute is functionally dependent on another nonkey attribute. A table with a single-attribute primary key cannot exhibit partial dependencies. • A table is in 2NF when it is in 1NF and contains no partial dependencies. Therefore, a 1NF table is automatically in 2NF when its primary key is based on only a single attribute. A table in 2NF may still contain transitive dependencies. • A table is in 3NF when it is in 2NF and contains no transitive dependencies. Given that definition, the Boyce-Codd normal form (BCNF) is merely a special 3NF case in which all determinant keys are candidate keys. When a table has only a single candi-date key, a 3NF table is automatically in BCNF. • A table that is not in 3NF may be split into new tables until all of the tables meet the 3NF requirements. • Normalization is an important part—but only a part—of the design process. As entities and attributes are defined during the ER modeling process, subject each entity (set) to normalization checks and form new entities (sets) as required. Incorporate the normalized entities into the ERD and continue the iterative ER process until all entities and their attributes are defined and all equivalent tables are in 3NF. • A table in 3NF might contain multivalued dependencies that produce either numer-ous null values or redundant data. Therefore, it might be necessary to convert a 3NF table to the fourth normal form (4NF) by splitting the table to remove the multival-ued dependencies. Thus, a table is in 4NF when it is in 3NF and contains no mul-tivalued dependencies. • The larger the number of tables, the more additional I/O operations and processing logic you need to join them. Therefore, tables are sometimes denormalized to yield less I/O in order to increase processing speed. Unfortunately, with larger tables, you pay for the increased processing speed by making the data updates less efficient, by making indexing more cumbersome, and by introducing data redundancies that are likely to yield data anomalies. In the design of production databases, use denormal-ization sparingly and cautiously. • The data-modeling checklist provides a way for the designer to check that the ERD meets a set of minimum requirements.

**Atomic attribute** - An attribute that cannot be further subdivided to produce meaningful components. For example, a person's last name attribute cannot be meaningfully subdivided.

An atomic attribute is one that cannot be further subdivided.

**Boyce-Codd Normal Form (BCNF)**  - A special type of third normal form which every determinant is a candidate key. A table in BCNF must be in 3NF.

A table is in boyce-codd normal form (bcNF) when every determinant in the table is a candidate key

**Denormalization -** A process by which a table is changed from a higher-level normal form to a lower-level normal form, usually to increase processing speed. Denormalization potentially yields data anomalies.

Denormalization produces a lower normal form; that is, a 3NF will be converted to a 2NF through denormalization. However, the price you pay for increased performance through denormalization is greater data redundancy

**Dependency diagram -** A representation of all data dependencies within a table.

Because such a diagram depicts all dependencies found within a given table structure, it is known as a dependency diagram. Depen-dency diagrams are very helpful in getting a bird’s-eye view of all the relationships among a table’s attributes, and their use makes it less likely that you will overlook an important dependency

**Determinant -** Any attribute in a specific row whose value directly determines other values in that row.

A determinant is any attribute whose value determines other values within a row.

**First Normal Form (1NF**) - The first stage in the normalization process. It describes a relation depicted in tabular format, with no repeating groups and a primary key identified. All nonkey attributes in the relation are dependent on the primary key.

The term first normal form (1NF) describes the tabular format that conforms to the definition of a relational table in which: • All of the key attributes are defined. • There are no repeating groups in the table. In other words, each row/column intersec-tion contains one and only one value, not a set of values. • All attributes are dependent on the primary key

**Fourth Normal Form (4NF)** - A table that is in 3NF and contains no multiple independent sets of multivalued dependencies.

table is in fourth normal form (4NF) when it is in 3NF and has no multivalued dependencies.

**Granularity -** The level of detail represented by the values stored in a table's row. Data stored at their lowest level of granularity are said to be atomic data.

granularity refers to the level of detail represented by the values stored in a table’s row. Data stored at its lowest level of granularity is said to be atomic data, as explained earlier

**Key attributes** - The attributes that form a primary key.

Also, in normalization terminology, any attribute that is at least part of a key is known as a prime attribute instead of the more common term key attribute,

**Nonprime attribute** - An attribute that is not part of a key.

non-prime attribute, or a nonkey attribute, is not part of any candidate key.

**Normalization -** A process that assigns attributes to entities so that data redundancies are reduced or eliminated.

Normalization is a process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies.

**Partial dependency** - In normalization, a condition in which an attribute is dependent on only a portion of the primary key.

A partial dependency exists when there is a functional dependence in which the determinant is only part of the primary key (remember the assumption, for this discussion, that there is only one candidate key)

**Prime attribute** - A key attribute; that is, an attribute that is part of a key or is the whole key.

any attribute that is at least part of a key is known as a prime attribute

**Repeating group** - In a relation, a characteristic describing a group of multiple entries of the same type for a single key attribute occurrence. For example, a car can have multiple colors for its top, interior, bottom, trim, and so on.

A repeating group derives its name from the fact that a group of multiple entries of the same or multiple types can exist for any single key attribute occurrence.

**Second Normal Form (2NF)** - The second stage in the normalization process, in which a relation is in 1NF and there are no partial dependencies.

A table is in second normal form (2NF) when: • It is in 1NF. and • It includes no partial dependencies; that is, no attribute is dependent on only a portion of the primary key.. It is still possible for a table in 2NF to exhibit transitive dependency. That is, the primary key may rely on one or more nonprime attributes to functionally determine other nonprime attributes, as indicated by a functional dependence among the nonprime attributes.

**Third Normal Form (3NF)** - A table is in 3NF when it is in 2NF and no nonkey attribute is functionally dependent on another nonkey attribute; that is, it cannot include transitive dependencies.

A table is in third normal form (3NF) when: • It is in 2NF. and • It contains no transitive dependencies.

**Transitive dependency -** A condition in which an attribute is dependent on another attribute that is not part of the primary key.

A transitive dependency exists when there are functional dependencies such that X S Y, Y S Z, and X is the primary key.

**2. Chapter 9**

 every application is composed of two parts: the data and the code (program instructions) by which the data is transformed into information. The data and code work together to represent real-world business functions and activities. At any given moment, physically stored data represents a snapshot of the business, but the picture is not complete without an understanding of the business activities represented by the code

The performance of an information system depends on three factors: • Database design and implementation • Application design and implementation • Administrative procedures

SDLC:1. Planning

An initial assessment of the information flow-and-extent requirements must be made during this discovery portion of the SDLC. Such an assessment should answer some important questions: • Should the existing system be continued? If the information generator does its job well, there is no point in modifying or replacing it. To quote an old saying, “If it ain’t broke, don’t fix it.” • Should the existing system be modified? If the initial assessment indicates deficiencies in the extent and flow of the information, minor (or even major) modifications might be needed. When considering modifications, the participants in the initial assessment must remember the distinction between wants and needs. • Should the existing system be replaced? The initial assessment might indicate that the current system’s flaws are beyond fixing. Given the effort required to create a new sys-tem, a careful distinction between wants and needs is perhaps even more important in this case than it is when modifying the system.

Participants in the SDLC’s initial assessment must begin to study and evaluate alter-native solutions. If a new system is necessary, the next question is whether it is feasible. The feasibility study must address the following: • The technical aspects of hardware and software requirements. The decisions might not yet be vendor-specific, but they must address the nature of the hardware requirements (desktop computer, mainframe, supercomputer, or mobile device) and the software requirements (single-user or multiuser operating systems, database type and soft-ware, programming languages to be used by the applications, and so on). • The system cost. The admittedly mundane question “Can we afford it?” is crucial. The answer might force a careful review of the initial assessment. A million-dollar solu-tion to a thousand-dollar problem is not defensible. At some point, the decision may be between building a system “in-house” or buying (and customizing) a third-party vendor system. In the long run, you need to find a cost-effective solution that best serves the needs (present and future) of the organization. • The operational cost. Does the company possess the human, technical, and finan-cial resources to keep the system operational? Should the feasibility study include the cost of management and end-user support needed to implement operational procedures to ensure the success of this system? What would be the impact of this new system in the company’s culture? People’s resistance to change should never be underestimated.3 Even if you choose to “buy” rather than to “build,” the system implementation must be carefully planned for it to be successful. Whatever the chosen option (build or buy), an analysis must be done to deploy the solution across the organization in ways that min-imize cost and culture changes, while maximizing value. The SDLC provides a frame-work for sound planning and implementation.

SDLC :2. Analysis

Problems defined during the planning phase are examined in greater detail during the analysis phase. A macro analysis must be made both of individual needs and organiza-tional needs, addressing questions such as: • What are the requirements of the current system’s end users? • Do those requirements fit into the overall information requirements? The analysis phase of the SDLC is, in effect, a thorough audit of user requirements. The existing hardware and software systems are also studied during the analysis phase. The result of the analysis should be a better understanding of the system’s functional areas, actual and potential problems, and opportunities.

When creating a logical design, the designer might use tools such as data flow dia-grams (DFDs), hierarchical input process output (HIPO) diagrams, entity relation-ship (ER) diagrams, and even some application prototypes.

SDLC:3. Detailed systems design

In the detailed systems design phase, the designer completes the design of the system’s processes.

SDLC:4. Implementation

During the initial stages of the implementation phase, the system enters into a cycle of coding, testing, and debugging until it is ready to be delivered. The actual database is created, and the system is customized by the creation of tables and views, user authorizations, and so on. The database contents might be loaded interactively or in batch mode, using a variety of methods and devices: • Customized user programs • Database interface programs • Conversion programs that import the data from a different file structure, using batch programs, a database utility, or both The system is subjected to exhaustive testing until it is ready for use. Traditionally, the implementation and testing of a new system took 50 to 60 percent of the total development time.

SDLC : 5. Maintenance

Almost as soon as the system is operational, end users begin to request changes in it. Those changes generate system maintenance activities, which can be grouped into three types: • Corrective maintenance in response to systems errors • Adaptive maintenance due to changes in the business environment • Perfective maintenance to enhance the system Because every request for structural change requires retracing the SDLC steps, the system is, in a sense, always at some stage of the SDLC. Each system has a predetermined operational life span, but its actual life span depends on its perceived utility. There are several reasons for reducing the operational life of cer-tain systems. Rapid technological change is one reason, especially for systems based on processing speed and expandability. Another common reason is the cost of maintaining a system.

**Bottom-up design** - A design philosophy that begins by identifying individual design components and then aggregates them into larger units. In database design, the process begins by defining attributes and then groups them into entities. Compare to top-down design.

Bottom-up design first identifies the data elements (items) and then groups them together in data sets. In other words, it first defines attributes, and then groups them to form entities.

**Boundaries -** The external limits to which any proposed system is subjected. These limits include budgets, personnel, and existing hardware and software.

The proposed system is also subject to limits known as boundaries, which are external to the system. Has any designer ever been told, “We have all the time in the world” or “Use an unlimited budget and as many people as needed to make the design come together”? Boundaries are also imposed by existing hardware and software. Ideally, the designer can choose the hardware and software that will best accomplish the system goals. In fact, software selection is an important aspect of the Systems Development Life Cycle. Unfortunately, in the real world, a system must often be designed around existing hardware. Thus, the scope and boundaries become the factors that force the design into a specific mold, and the designer’s job is to design the best system possible within those constraints. (Note that prob-lem definitions and the objectives must sometimes be reshaped to meet the system scope and boundaries.)

**Centralized design** - A process in which a single conceptual design is modeled to match an organization's database requirements. It is typically used when a data component consists of a relatively small number of objects and procedures. Compare to decentralized design.

Centralized design is productive when the data component has a relatively small number of objects and procedures. The design can be carried out and represented in a fairly simple database. Centralized design is typical of relatively simple, small databases and can be successfully done by a single database administrator or by a small, infor-mal design team. The company operations and the scope of the problem are sufficiently limited to allow even a single designer to define the problem(s), create the conceptual design, verify the conceptual design with the user views, define system processes and data constraints to ensure the efficacy of the design, and ensure that the design will com-ply with all the requirements. (Although centralized design is typical for small compa-nies, do not make the mistake of assuming that it is limited to small companies. Even large companies can operate within a relatively simple database environment.) Figure 9.15 summarizes the centralized design option. Note that a single conceptual design is completed and then validated in the centralized design approach.

**Clustered table -** A storage technique that stores related rows from two related tables in adjacent data blocks on disk.

The clustered tables storage technique stores related rows from two related tables in adjacent data blocks on disk. This ensures that the data is stored in sequentially adjacent locations, thereby reducing data access time and increasing system performance.

**Cohesivity -** The strength of the relationships between a module's components. Module cohesivity must be high.

The term cohesivity describes the strength of the relation-ships found among the module’s entities. A module must display high cohesivity—that is, the entities must be strongly related, and the module must be complete and self-sufficient

**Computer-Aided Systems Engineering (CASE)** - Tools used to automate part or all of the Systems Development Life Cycle.

If the system’s maintenance cost is high, its value becomes suspect. Computer-aided software engineering (CASE) tools, such as System Architect or Visio Professional, help produce better systems within a reasonable amount of time and at a reasonable cost. In addition, CASE-produced applications are more structured, better documented, and especially standardized, which tends to prolong the operational life of systems by making them easier and cheaper to update and maintain.

**Conceptual design -** A process that uses data-modeling techniques to create a model of a database structure that represents real-world objects as realistically as possible. The techniques are both software- and hardware-independent.

The goal at this stage is to design a database that is independent of database software and physical details. The output of this process is a conceptual data model that describes the main data entities, attributes, relationships, and constraints of a given problem domain. This design is descriptive and narrative in form. In other words, it is generally composed of a graphical representation as well as textual descriptions of the main data elements, relationships, and constraints.

In this stage, data modeling is used to create an abstract database structure that rep-resents real-world objects in the most realistic way possible. The conceptual model must embody a clear understanding of the business and its functional areas. At this level of abstraction, the type of hardware and database model to be used might not have been identified yet. Therefore, the design must be software-and hardware-independent so that the system can be set up within any platform chosen later.

**Database development** - The process of database design and implementation.

In a broad sense, the term database development describes the process of database design and implementation. The primary objective in database design is to create com-plete, normalized, nonredundant (to the greatest extent possible), and fully integrated conceptual, logical, and physical database models. The implementation phase includes creating the database storage structure, loading data into the database, and providing for data management. Consideration should be taken to design and implement a database that is flexible and scalable over time. Although most designs typically focus on solving current problems, it is important to create a design that is flexible enough to adapt to future changes (such as performance, size, or reporting requirements).

**Database fragment** - A subset of a distributed database. Although the fragments may be stored at different sites within a computer network, the set of all fragments is treated as a single database.

A database fragment is a subset of a database that is stored at a given location. The database fragment may be a subset of rows or columns from one or multiple tables.

**Database Life Cycle (DBLC)** - A cycle that traces the history of a database within an information system. The cycle is divided into six phases: initial study, design, implementation and loading, testing and evaluation, operation and maintenance, and evolution.

Within the larger information system, the database is subject to a life cycle as well. The Database Life Cycle (DBLC) contains six phases, as shown in Figure 9.3:

database initial study

The overall purpose of the database initial study is to:

• Analyze the company situation

The company situation describes the general condi-tions in which a company operates, its organizational structure, and its mission. To ana-lyze the company situation, the database designer must learn the company’s operational components, how they function, and how they interact. The following issues must be resolved: • What is the organization’s general operating environment, and what is its mission within that environment? The design must satisfy the operational demands created by the organization’s mission. For example, a mail-order business probably has operational requirements for its database that are quite different from those of a manufacturing business. • What is the organization’s structure? Knowing who controls what and who reports to whom is quite useful when you need to define required information flows, specific report and query formats, and so on.

• Define problems and constraints

• Define objectives

• What is the proposed system’s initial objective? • Will the system interface with other existing or future systems in the company? • Will the system share the data with other systems or users?

• Define scope and boundaries

The designer must recognize two sets of limits: scope and boundaries.

database design

As you examine the procedures required to complete the design phase in the DBLC, remember these points: • The process of database design is loosely related to the analysis and design of a larger system. The data component is only one element of a larger information system. • The systems analysts or systems programmers are in charge of designing the other system components. Their activities create the procedures that will help transform the data within the database into useful information. • The database design does not constitute a sequential process. Rather, it is an iterative process that provides continuous feedback designed to trace previous steps.

three stages:

conceptual design

Data analysis and requirements

Appropriate data element characteristics are those that can be transformed into appropriate information. Therefore, the designer’s efforts are focused on: • Information needs. What kind of information is needed? That is, what output (reports and queries) must be generated by the system, what information does the current system generate, and to what extent is that information adequate? • Information users. Who will use the information? How is the information to be used? What are the various end-user data views? • Information sources. Where is the information to be found? How is the information to be extracted once it is found? • Information constitution. What data elements are needed to produce the information? What are the data attributes? What relationships exist in the data? What is the data volume? How frequently is the data used? What data transformations will be used to generate the required information?

The designer obtains the answers to those questions from a variety of sources to compile the necessary information: • Developing and gathering end-user data views. The database designer and the end user(s) jointly develop a precise description of end-user data views, which in turn are used to help identify the database’s main data elements. • Directly observing the current system: existing and desired output. The end user usually has an existing system in place, whether it is manual or computer-based. The designer reviews the existing system to identify the data and its characteristics. The designer examines the input forms and files (tables) to discover the data type and volume. If the end user already has an automated system in place, the designer carefully examines the current and desired reports to describe the data required to support the reports. • Interfacing with the systems design group. As noted earlier in this chapter, the database design process is part of the SDLC. In some cases, the systems analyst in charge of designing the new system will also develop the conceptual database model. (This is usually true in a decentralized environment.) In other cases, the database design is considered part of the DBA’s job. The presence of a DBA usually implies the existence of a formal data-processing department. The DBA designs the database according to the specifications created by the systems analyst.

A business rule is a brief and precise description of a policy, procedure, or principle within a specific organization’s environment. For example: Examples of business rules are as follows: • A customer may make many payments on an account. • Each payment on an account is credited to only one customer. • A customer may generate many invoices. • Each invoice is generated by only one customer.

Business rules yield several important benefits in the design of new systems: • They help standardize the company’s view of data. • They constitute a communications tool between users and designers. • They allow the designer to understand the nature, role, and scope of the data. • They allow the designer to understand business processes. • They allow the designer to develop appropriate relationship participation rules and foreign key constraints. See Chapter 4, Entity Relationship (ER) Modeling

entity relationship modeling and normalization

DeveLopIng The ConCepTuaL moDeL uSIng er DIagramS StEp ACtIVIty 1Identify, analyze, and refine the business rules. 2. Identify the main entities, using the results of Step 1. 3. Define the relationships among the entities, using the results of Steps 1 and 2. 4. Define the attributes, primary keys, and foreign keys for each of the entities. 5. normalize the entities. (remember that entities are implemented as tables in an rDBmS.) 6. Complete the initial er diagram. 7. validate the er model against the end users’ information and processing requirements. 8. modify the er model, using the results of Step 7.

All objects (entities, attributes, relations, views, and so on) are defined in a data dictio-nary, which is used in tandem with the normalization process to help eliminate data anom-alies and redundancy problems. During this ER modeling process, the designer must: • Define entities, attributes, primary keys, and foreign keys. (The foreign keys serve as the basis for the relationships among the entities.) • Make decisions about adding new primary key attributes to satisfy end-user and processing requirements. • Make decisions about the treatment of composite and multivalued attributes. • Make decisions about adding derived attributes to satisfy processing requirements. • Make decisions about the placement of foreign keys in 1:1 relationships. • Avoid unnecessary ternary relationships. • Draw the corresponding ER diagram. • Normalize the entities. • Include all data element definitions in the data dictionary. • Make decisions about standard naming conventions.

Data model verification

Verification requires that the model be run through a series of tests against: • End-user data views • All required transactions: SELECT, INSERT, UPDATE, and DELETE operations • Access rights and security • Business-imposed data requirements and constraints

Distributed database design

logical design

physical design

plus the critical decision of DBMS selection

Although the factors that affect the purchasing decision vary from company to com-pany, some of the most common are: • Cost. This includes the original purchase price, along with maintenance, operational, license, installation, training, and conversion costs. • DBMS features and tools. Some database software includes a variety of tools that facilitate application development. For example, the availability of query by example (QBE), screen painters, report generators, application generators, and data dictionaries helps to create a more pleasant work environment for both the end user and the application programmer. Database administrator facili-ties, query facilities, ease of use, performance, security, concurrency control, transaction processing, and third-party support also influence DBMS software selection. • Underlying model. This can be hierarchical, network, relational, object/relational, or object-oriented. • Portability. A DBMS can be portable across platforms, systems, and languages. • DBMS hardware requirements. Items to consider include processor(s), RAM, disk space, and so on.

implementation and loading

Install the DBMS

Create the database(s)

For example, the implementation of the logical design in IBM’s DB2 would require the following: 1. The system administrator (SYSADM) would create the database storage group. This step is mandatory for such mainframe databases as DB2. Other DBMS software may create equivalent storage groups automatically when a database is created. (See Step 2.) Consult your DBMS documentation to see if you must create a storage group, and if so, what the command syntax must be. 2. The SYSADM creates the database within the storage group. 3. The SYSADM assigns the rights to use the database to a database administrator (DBA). 4. The DBA creates the table space(s) within the database. 5. The DBA creates the table(s) within the table space(s). 6. The DBA assigns access rights to the table spaces and to the tables within specified table spaces. Access rights may be limited to views rather than to whole tables. The creation of views is not required for database access in the relational environment, but views are desirable from a security standpoint. For example, using the following command, access rights to a table named PROFESSOR may be granted to the user Lynn Eilers, whose identification code is LEILERS: GRANT SELECT ON PROFESSOR TO USER LEILERS;

Load or convert the data

testing and evaluation

Programmers use database tools to prototype the applications during coding of the programs

Test the database

Data stored in the company database must be protected from access by unauthorized users. (It does not take much imagination to predict the likely results if students have access to a student database or if employees have access to payroll data!) Consequently, you must test for at least the following: • Physical security allows only authorized personnel physical access to specific areas. Depending on the type of database implementation, however, establishing physical security might not always be practical. For example, a university student research database is not a likely candidate for physical security. • Password security allows the assignment of access rights to specific authorized users. Password security is usually enforced at login time at the operating system level. • Access rights can be established through the use of database software. The assignment of access rights may restrict operations (CREATE, UPDATE, DELETE, and so on) on predetermined objects such as databases, tables, views, queries, and reports. • Audit trails are usually provided by the DBMS to check for access violations. Although the audit trail is an after-the-fact device, its mere existence can discourage unautho-rized use. • Data encryption can render data useless to unauthorized users who might have violated some of the database security layers. • Diskless workstations allow end users to access the database without being able to download the information from their workstations.

Fine tune the database

Evaluate the database and its application programs

Database backups can be full, differential or transaction log

Operation

maintenance and evolution.

Some of the required periodic maintenance activities include: • Preventive maintenance (backup) • Corrective maintenance (recovery) • Adaptive maintenance (enhancing performance, adding entities and attributes, and so on) • Assignment of access permissions and their maintenance for new and old users • Generation of database access statistics to improve the efficiency and usefulness of system audits and to monitor system performance • Periodic security audits based on the system-generated statistics • Monthly, quarterly, or yearly system usage summaries for internal billing or budgeting purposes

**Database role** - A set of database privileges that could be assigned as a unit to a user or group.

A database role is a set of database privileges that could be assigned as a unit to a user or group.

**Decentralized design** - A process in which conceptual design is used to model subsets of an organization's database requirements. After verification of the views, processes, and constraints, the subsets are then aggregated into a complete design. Such modular designs are typical of complex systems in which the data component has a relatively large number of objects and procedures. Compare to centralized design.

Decentralized design might be used when the system’s data component has a con-siderable number of entities and complex relations on which very complex operations are performed. Decentralized design is also often used when the problem itself is spread across several operational sites and each element is a subset of the entire data set. (See Figure 9.16.)

• Synonyms and homonyms. Various departments might know the same object by different names (synonyms), or they might use the same name to address different objects (homonyms). The object can be an entity, an attribute, or a relationship. • Entity and entity subtypes. An entity subtype might be viewed as a separate entity by one or more departments. The designer must integrate such subtypes into a higher-level entity. • Conflicting object definitions. Attributes can be recorded as different types (character, numeric), or different domains can be defined for the same attribute. Constraint defi-nitions can vary as well. The designer must remove such conflicts from the mode

**Description of operations -** A document that provides a precise, detailed, up-to-date, and thoroughly reviewed description of the activities that define an organization's operating environment.

Ideally, business rules are derived from a formal description of operations, which is a document that provides a precise, up-to-date, and thoroughly reviewed description of the activities that define an organization’s operating environment.

**Differential backup -** A level of database backup in which only the last modifications to the database are copied.

. • A differential backup of the database, in which only the objects that have been updated or modified since the last full backup are backed up.

**Full backup (database dump**) - A complete copy of an entire database saved and periodically updated in a separate memory location. A full backup ensures a full recovery of all data after a physical disaster or database integrity failure.

• A full backup, or dump, of the entire database. In this case, all database objects are backed up in their entirety

**Information system** - A system that provides for data collection, storage, and retrieval; facilitates the transformation of data into information; and manages both data and information. An information system is composed of hardware, the DBMS and other software, databases, people, and procedures.

he data-base is part of a larger whole known as an information system (IS), which provides for data collection, storage, transformation, and retrieval. The information system also helps transform data into information, and it allows for the management of both data and information. Thus, a complete information system is composed of people, hardware, software, the database(s), application programs, and procedures.

**Logical design** - A stage in the design phase that matches the conceptual design to the requirements of the selected DBMS and is therefore software-dependent. Logical design is used to translate the conceptual design into the internal model for a selected database management system, such as DB2, SQL Server, Oracle, IMS, Informix, Access, or Ingress.

The logical design goal is to design an enterprise-wide database that is based on a specific data model but independent of physical-level details. Logical design requires that all objects in the con-ceptual model be mapped to the specific constructs used by the selected database model. For example, the logical design for a relational DBMS includes the specifications for the relations (tables), relationships, and constraints (in other words, domain definitions, data validations, and security views). The logical design is generally performed in four steps, which are listed in Table 9.6. tABLe 9.6 LogICaL DeSIgn STepS StEp ACtIVIty

1 map the conceptual model to logical model components.

Mapping the conceptual model to the relational model

1. map strong entities.
2. map supertype/subtype relationships.
3. map weak entities.
4. map binary relationships.
5. map higher-degree relationships.

2. validate the logical model using normalization.

3. validate the logical model integrity constraints.

4. validate the logical model against user requirements.

Such steps, like most of the data-modeling process, are not necessarily performed sequentially, but in an iterative fashion. The following sections cover these steps in more detail.

**Minimal data rule** - Defined as…All that is needed is there, and all that is there is needed. In other words, all data elements required by database transactions must be defined in the model, and all data elements defined in the model must be used by at least one database transaction.

All that is needed is there, and all that is there is needed.

**Module -** 1 - A design segment that can be implemented as an autonomous unit, and is sometimes linked to produce a system. 2 - An information system component that handles a specific function, such as inventory, orders, or payroll.

Because real-world database design is generally done by teams, the database design is probably divided into major components known as modules. A module is an infor-mation system component that handles a specific business function, such as inventory, orders, or payroll. Under these conditions, each module is supported by an ER segment that is a subset or fragment of an enterprise ER model. Working with modules accom-plishes several important ends: • The modules (and even the segments within them) can be delegated to design groups within teams, greatly speeding up the development work. • The modules simplify the design work. The large number of entities within a complex design can be daunting. Each module contains a more manageable number of entities. • The modules can be prototyped quickly. Implementation and application program-ming trouble spots can be identified more readily. Quick prototyping is also a great confidence builder. • Even if the entire system cannot be brought online quickly, the implementation of one or more modules will demonstrate that progress is being made and that at least part of the system is ready to begin serving the end users.

As useful as modules are, they represent a loose collection of ER model fragments that could wreak havoc in the database if left unchecked. For example, the ER model fragments: • Might present overlapping, duplicated, or conflicting views of the same data • Might not be able to support all processes in the system’s modules

To avoid these problems, it is better if the modules’ ER fragments are merged into a single enterprise ER model. This process starts by selecting a central ER model segment and iteratively adding more ER model segments one at a time. At each stage, for each new entity added to the model, you need to validate that the new entity does not overlap or conflict with a previously identified entity in the enterprise ER model. Merging the ER model segments into an enterprise ER model triggers a careful reeval-uation of the entities, followed by a detailed examination of the attributes that describe those entities. This process serves several important purposes: • The emergence of the attribute details might lead to a revision of the entities them-selves. Perhaps some of the components first believed to be entities will instead turn out to be attributes within other entities. Or, a component that was originally con-sidered an attribute might turn out to contain a sufficient number of subcomponents to warrant the introduction of one or more new entities. • The focus on attribute details can provide clues about the nature of relationships as they are defined by the primary and foreign keys. Improperly defined relationships lead to implementation problems first and to application development problems later. • To satisfy processing and end-user requirements, it might be useful to create a new primary key to replace an existing primary key. For example, in the example illus-trated in Figure 9.9, a surrogate primary key (RENTAL\_ID) could be introduced to replace the original primary key composed of VIDEO\_ID and CUST\_NUM. • Unless the entity details (the attributes and their characteristics) are precisely defined, it is difficult to evaluate the extent of the design’s normalization. Knowledge of the normalization levels helps guard against undesirable redundancies. • A careful review of the rough database design blueprint is likely to lead to revi-sions. Those revisions will help ensure that the design is capable of meeting end-user requirements.

After finishing the merging process, the resulting enterprise ER model is verified against each of the module’s processes. The ER model verification process is detailed in Table 9.5.

The er moDeL verIfICaTIon proCeSS StEp ACtIVIty 1. Identify the er model’s central entity. 2. Identify each module and its components. 3. Identify each module’s transaction requirements: Internal: updates/inserts/deletes/queries/reports external: module interfaces 4. verify all processes against the module’s processing and reporting requirements. 5. make all necessary changes suggested in Step 4. 6. repeat Steps 2–5 for all modules.

The verification process starts with selecting the central (most important) entity, which is the focus for most of the system’s operations. To identify the central entity, the designer selects the entity involved in the greatest number of the model’s relationships. In the ER diagram, it is the entity with more lines connected to it than any other. The next step is to identify the module or subsystem to which the central entity belongs and to define that module’s boundaries and scope. The entity belongs to the module that uses it most frequently. Once each module is identified, the central entity is placed within the module’s framework to let you focus on the module’s details. Within the central entity/module framework, you must • Ensure the module’s cohesivity.. • Analyze each module’s relationships with other modules to address module coupling.

Processes may be classified according to their: • Frequency (daily, weekly, monthly, yearly, or exceptions) • Operational type (INSERT or ADD, UPDATE or CHANGE, DELETE, queries and reports, batches, maintenance, and backups)

**Module coupling -** The extent to which modules are independent of one another.

module coupling describes the extent to which modules are independent of one another. Modules must display low coupling, indicating that they are independent of other modules. Low coupling decreases unnecessary intermodule dependencies, thereby allowing the creation of a truly modular system and eliminating unnecessary relationships among entities.

**Physical design -** A stage of database design that maps the data storage and access characteristics of a database. Because these characteristics are a function of the types of devices supported by the hardware, the data access methods supported by the system physical design are both hardware- and software-dependent.

physical design is the process of determining the data storage organization and data access characteristics of the database to ensure its integrity, security, and performance. This is the last stage in the database design process.

Steps :

1. Define data storage organization.

Before you can define data storage organization, you must determine the volume of data to be managed and the data usage patterns. • Knowing the data volume will help you determine how much storage space to reserve for the database. To do this, the designer follows a process similar to the one used during ER model verification. For each table, identify all possible transactions, their frequency, and volume. For each transaction, you determine the amount of data to be added or deleted from the database. This information will help you determine the amount of data to be stored in the related table. • Conversely, knowing how frequently new data is inserted, updated, and retrieved will help the designer determine the data usage patterns. Usage patterns are critical, particularly in distributed database design. For example, are there any weekly batch uploads or monthly aggregation reports to be generated? How frequently is new data added to the system? Equipped with the two previous pieces of information, the designer must: • Determine the location and physical storage organization for each table. As you saw in Section 9-3c, tables are stored in table spaces, and a table space can hold data from multiple tables. In this step, the designer assigns which tables will use which table spaces, and assigns the location of the table spaces. For example, a useful technique available in most relational databases is the use of clustered tables. The clustered tables storage technique stores related rows from two related tables in adjacent data blocks on disk. This ensures that the data is stored in sequentially adjacent locations, thereby reducing data access time and increasing system performance. • Identify indexes and the type ofindexes to be used for each table. As you saw in previous chapters, indexes are useful for ensuring the uniqueness of data values in a column and to facilitate data lookups. You also know that the DBMS automatically creates a unique index for the primary key of each table. You will learn in Chapter 11 about the various types of index organization. In this step, you identify all required indexes and determine the best type of organization to use based on the data usage patterns and performance requirements. • Identify the views and the type of views to be used on each table. As you learned in Chapter 8, a view is useful to limit access to data based on user or transaction needs. Views can also be used to simplify processing and end-user data access. In this step the designer must ensure that all views can be implemented and that they provide only the required data. The designer must also become familiar with the types of views supported by the DBMS and how they could help meet system goals.

1. Define integrity and security measures.

Define user and security groups and roles. User management is more a function of database administration than database design. However, as a designer you must know the different types of users and groups of users to properly enforce database security. Most DBMS implementations support the use of database roles. A database role is a set of database privileges that could be assigned as a unit to a user or group. For exam-ple, you could define an Advisor role that has Read access to the vSCHEDULE view.

Assign security controls. The DBMS also allows administrators to assign specific access rights for database objects to a user or group of users. For example, you could assign the SELECT and UPDATE access rights to the user leilers on the CLASS table. An access right could also be revoked from a specific user or groups of users. This fea-ture could come in handy during database backups, scheduled maintenance events, or even during data breach incidents.

1. Determine performance measurements.

**Scope -**  The part of a system that defines the extent of the design, according to operational requirements.

The system’s scope defines the extent of the design according to operational requirements.

**Systems analysis -** The process that establishes the need for an information system and its extent.

Systems analysis is the process that establishes the need for an information system and its extent.

**Systems development** - The process of creating an information system.

The process of creating an information system is known as systems development.

**Systems Development Life Cycle (SDLC)** - The cycle that traces the history of an information system. The SDLC provides the big picture within which database design and application development can be mapped out and evaluated.

The systems development life cycle traces the history of an information system. the Systems Development Life Cycle is a general framework through which you can track and understand the activities required to develop and maintain information sys-tems. Within that framework, there are several ways to complete various tasks specified in the SDLC. For example, this book focuses on er modeling and on relational database design and implementation, and that focus is maintained in this chapter. However, there are alternative methodologies: • unified modeling Language (umL) provides object-oriented tools to support the tasks associated with the development of information systems. umL is covered in Appendix H, unified modeling Language (umL), at www.cengagebrain.com. • rapid Application Development (rAD)1 is an iterative software development method-ology that uses prototypes, CASe tools, and flexible management to develop appli-cation systems. rAD started as an alternative to traditional structured development, which suffered from long deliverable times and unfulfilled requirements. • Agile Software Development2 is a framework for developing software applications that divides the work into smaller subprojects to obtain valuable deliverables in shorter times and with better cohesion. this method emphasizes close communi-cation among all users and continuous evaluation with the purpose of increasing customer satisfaction. Although the development methodologies may change, the basic framework within which they are used does not change

**Top-down design** - A design philosophy that begins by defining the main structures of a system and then moves to define the smaller units within those structures. In database design, this process first identifies entities and then defines the attributes within the entities. Compare to bottom-up design.

top-down design starts by identifying the data sets and then defines the data ele-ments for each of those sets. This process involves the identification of different entity types and the definition of each entity’s attributes.

**Transaction log backup** - A backup of only the transaction log operations that are not reflected in a previous backup copy of the database.

• A transaction log backup, which backs up only the transaction log operations that are not reflected in a previous backup copy of the database. In this case, no other database objects are backed up. (For a complete explanation of the transaction log, see Chapter 10, Transaction Management and Concurrency Control.)

**Virtualization -** A technique that creates logical representations of computing resources that are independent of the underlying physical computing resources.

The DBMS may be installed on a new server or on existing servers. One current trend is called virtualization. Virtualization is a technique that creates logical representations of computing resources that are inde-pendent of the underlying physical computing resources.

Summary

• An information system is designed to help transform data into information and to manage both data and information. Thus, the database is a very important part of the information system. Systems analysis is the process that establishes the need for an information system and its extent. Systems development is the process of creating an information system. • The Systems Development Life Cycle (SDLC) traces the history of an application within the information system. The SDLC can be divided into five phases: planning, analysis, detailed systems design, implementation, and maintenance. The SDLC is an iterative process rather than a sequential process. • The Database Life Cycle (DBLC) describes the history of the database within the infor-mation system. The DBLC is composed of six phases: database initial study, database design, implementation and loading, testing and evaluation, operation, and main-tenance and evolution. Like the SDLC, the DBLC is iterative rather than sequential. • The conceptual portion of the design may be subject to several variations based on two basic design philosophies: bottom-up versus top-down and centralized versus decentralized.

**3. Chapter 13**

**Algorithm -** A process or set of operations in a calculation. 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.

**Attribute hierarchy -** A top-down data organization that is used for two main purposes: aggregation and drill-down/roll-up data analysis.

Attributes within dimensions can be ordered in a well-defined attribute hierarchy. The attribute hierarchy provides a top-down data organization that is used for two main purposes: aggregation and drill-down/roll-up data analysis. For example, Figure 13.8 shows how the location dimension attributes can be organized in a hierarchy by region, state, city, and store

**Business Intelligence (BI)** - A comprehensive, cohesive, and integrated set of tools and processes used to capture, collect, integrate, store, and analyze data with the purpose of generating and presenting information to support business decision making.

Business intelligence (BI)1 is a term that describes a comprehensive, cohesive, and integrated set of tools and processes used to capture, collect, integrate, store, and analyze data with the purpose of generating and presenting information to support business decision making.

BI provides a framework for: • Collecting and storing operational data • Aggregating the operational data into decision support data • Analyzing decision support data to generate information • Presenting such information to the end user to support business decisions • Making business decisions, which in turn generate more data that is collected, stored, and so on (restarting the process) • Monitoring results to evaluate outcomes of the business decisions, which again provides more data to be collected, stored, and so on • Predicting future behaviors and outcomes with a high degree of accuracy

BI architecture is composed of many interconnected parts: people, processes, data, and technology working together to facil-itate and enhance a business’s management and governance.

general BI framework depicted in Figure 13.1 has six basic components that encompass the functionality required on most current-generation BI systems. You will learn more about these components later in this and future chapters. The components are briefly described in Table 13.2. TABLE 13.2 BASIc BI ArchITecTurAl comPoNeNTS Component DesCrIptIon ETL tools Data extraction, transformation, and loading (etl) tools collect, filter, integrate, and aggregate internal and external data to be saved into a data store optimized for decision support. Data store The data store is optimized for decision support and is generally represented by a data warehouse or a data mart. The data is stored in structures that are optimized for data analysis and query speed. Query and reporting This component performs data selection and retrieval, and it is used by the data analyst to create queries that access the database and create the required reports. Data visualization This component presents data to the end user in a variety of meaningful and innovative ways. This tool helps the end user select the most appropriate presentation format, such as summary reports, maps, pie or bar graphs, mixed graphs, and static or interactive dashboards. Data monitoring and alerting This component allows real-time monitoring of business activities. The BI system will present concise information in a single integrated view. This integrated view could include specific metrics about the system performance or activities, such as number of orders placed in the last four hours, number of customer complaints by product by month, and total revenue by region. Alerts can be placed on a given metric; once the value of a metric goes below or above a certain baseline, the system will perform a given action, such as emailing shop floor managers, presenting visual alerts, or starting an application. Data analytics This component performs data analysis and data-mining tasks using the data in the data store. This tool advises the user as to which data analysis tool to select and how to build a reliable business data model. Business models are generated by special algorithms that identify and enhance the understanding of business situations and problems.

A modern BI system provides three distinctive reporting styles: • Advanced reporting. A BI system presents insightful information about the organiza-tion in a variety of presentation formats. Furthermore, the reports provide interactive features that allow the end user to study the data from multiple points of view—from highly summarized to very detailed data. The reports present key actionable informa-tion used to support decision making. • Monitoring and alerting. After a decision has been made, the BI system offers ways to monitor the decision’s outcome. The BI system provides the end user with ways to define metrics and other key performance indicators to evaluate different aspects of an organization. In addition, exceptions and alerts can be set to warn managers promptly about deviations or problem areas. • Advanced data analytics. A BI system provides tools to help the end user discover relationships, patterns, and trends hidden within the organization’s data. These tools are used to create two types of data analysis: explanatory and predictive. Explanatory analysis provides ways to discover relationships, trends, and patterns among data, while predictive analysis provides the end user with ways to create models that predict future outcomes.

Business intelligence benefits:

• Integrating architecture. Like any other IT project, BI has the potential of becoming the integrating umbrella for a disparate mix of IT systems within an organization. This architecture could support all types of company-generated data from operational to executive, as well as diverse hardware such as mainframes, servers, desktops, lap-tops, and mobile devices.

• Common user interface for data reporting and analysis. BI front ends can provide up-to-the -minute consolidated information using a common interface for all com-pany users. IT departments no longer have to provide multiple training options for diverse interfaces. End users benefit from similar or common interfaces in different devices that use multiple clever and insightful presentation formats. • Common data repository fosters single version of company data. In the past, multiple IT systems supported different aspects of an organization’s operations. Such systems collected and stored data in separate data stores. Keeping the data synchronized and up to date has always been difficult. BI provides a framework to integrate such data under a common environment and present a single version of the data. • Improved organizational performance. BI can provide competitive advantages in many different areas, from customer support to manufacturing processes. Such advantages can be reflected in added efficiency, reduced waste, increased sales, reduced employee and customer turnover, and most importantly, an increased bottom line for the business.

Business intelligence technology trends

• Data storage improvements. Newer data storage technologies, such as solid state drives (SSD) and Serial Advanced Technology Attachment (SATA) drives, offer increased performance and larger capacity that make data storage faster and more affordable. Currently you can buy single drives with a capacity approaching 10 terabytes. • Business intelligence appliances. Vendors now offer plug-and-play appliances opti-mized for data warehouse and BI applications. These new appliances offer improved price-performance ratios, simplified administration, rapid installation, scalability, and fast integration. Some of these vendors include IBM, Netezza, EMC Greenplum, and Teradata Aster. • Business intelligence as a service. Vendors now offer data warehouses and BI as a service. These cloud-based services allow any corporation to rapidly develop a data warehouse store without the need for hardware, software, or extra personnel. These prepackaged services offer “pay-as-you-go” models for specific industries and capaci-ties, and they provide an opportunity for organizations to pilot-test a BI project with-out incurring large time or cost commitments. For example, such services are offered by IBM, Oracle, Microsoft, Teradata, MicroStrategy, and SAP. • Big Data analytics. The Big Data phenomenon is creating a new market for data ana-lytics. Organizations are turning to social media as the new source for information and knowledge to gain competitive advantages. • Personal analytics. OLAP brought data analytics to the desktop of every end user in an organization. Mobile BI is extending business decision making outside the walls of the organization. BI can now be deployed to mobile users who are closer to cus-tomers. The main requirement is for the BI end user to have a key understanding of the business. Some personal analytics vendors include MicroStrategy, QlikView, and Tableau. There is a growing trend toward self-service, personalized data analytics. It is not so far-fetched to imagine that in a few years, end users will have smart data ana-lytics agents on their smartphones tailored to their personal interests. Such personal agents will provide users with up-to-the-minute “intelligent knowledge” about their personal interests.

The effectiveness of BI depends on the quality of data gathered at the operational level

From the data analyst’s point of view, decision support data differs from operational data in three main areas: time span, granularity, and dimensionality. • Time span. Operational data covers a short time frame. In contrast, decision support data tends to cover a longer time frame. Managers are seldom interested in a specific sales invoice to Customer X; rather, they tend to focus on sales generated during the last month, the last year, or the last five years. • Granularity (level of aggregation). Decision support data must be presented at different levels of aggregation, from highly summarized to nearly atomic. For exam-ple, if managers analyze regional sales, they must be able to access data showing the sales by region, by city within the region, by store within the city within the region, and so on. In that case, summarized data to compare the regions is required, along with data in a structure that enables a manager to drill down, or decompose, the data into more atomic components—that is, finer-grained data at lower levels of aggregation. In contrast, when you roll up the data, you are aggregating the data to a higher level. • Dimensionality. Operational data focuses on representing individual transactions rather than the effects of the transactions over time. In contrast, data analysts tend to include many data dimensions and are interested in how the data relates over those dimensions. For example, an analyst might want to know how Product X fared rela-tive to Product Z during the past six months by region, state, city, store, and customer. In that case, both place and time are part of the picture.

From the designer’s point of view, the differences between operational and decision support data are as follows: • Operational data represents transactions as they happen in real time. Decision sup-port data is a snapshot of the operational data at a given point in time. Therefore, decision support data is historic, representing a time slice of the operational data. • Operational and decision support data are different in terms of transaction type and transaction volume. Whereas operational data is characterized by update transac-tions, decision support data is mainly characterized by read-only transactions. Deci-sion support data also requires periodic updates to load new data that is summarized from the operational data. Finally, the concurrent transaction volume in operational data tends to be very high compared with the low to medium levels in decision support data. • Operational data is commonly stored in many tables, and the stored data rep-resents information about a given transaction only. Decision support data is gener-ally stored in a few tables derived from the operational data. The decision support data does not include the details of each operational transaction. Instead, decision support data represents transaction summaries; therefore, the decision support database stores data that is integrated, aggregated, and summarized for decision support purposes. • The degree to which decision support data is summarized is very high when con-trasted with operational data. Therefore, you will see a great deal of derived data in decision support databases. For example, rather than storing all 10,000 sales trans-actions for a given store on a given day, the decision support database might simply store the total number of units sold and the total sales dollars generated during that day. Decision support data might be collected to monitor such aggregates as total sales for each store or for each product. The purpose of the summaries is simple: they are used to establish and evaluate sales trends and product sales comparisons and to provide other data that serves decision needs. (How well are items selling? Should this product be discontinued? Has the advertising been effective as measured by increased sales?) • The data models that govern operational data and decision support data are differ-ent. The operational database’s frequent and rapid data updates make data anoma-lies a potentially devastating problem. Therefore, the data in a relational transaction (operational) system generally requires normalized structures that yield many tables, each of which contains the minimum number of attributes. In contrast, the decision support database is not subject to such transaction updates, and the focus is on query-ing capability. Therefore, decision support databases tend to be non-normalized and include few tables, each of which contains a large number of attributes. • The frequency and complexity of query activity in the operational database tends to be low to allow additional processing cycles for the more crucial update trans-actions. Therefore, queries against operational data typically are narrow in scope and low in complexity, and high speed is critical. In contrast, decision support data exists for the sole purpose of serving query requirements. Queries against decision support data typically are broad in scope and high in complexity, and less speed is needed. • Finally, decision support data is characterized by very large amounts of data. The large data volume is the result of two factors. First, data is stored in non-normalized struc-tures that are likely to display many data redundancies and duplications. Second, the same data can be categorized in many different ways to represent different snapshots. For example, sales data might be stored in relation to product, store, customer, region, and manager.

Decision support data requirements

1. Be aware of the schema (how normalized ect)
2. Data extraction and filtering
3. Database size

**Cube cache -** In multidimensional OLAP, the shared, reserved memory area where data cubes are held. Using the cube cache assists in speeding up data access.

To speed data access, data cubes are normally held in memory in the cube cache. (A data cube is only a window to a predefined subset of data in the database. A data cube and a database are not the same thing.)

**Dashboard -** In business intelligence, a Web-based system that presents key business performance indicators or information in a single, integrated view with clear and concise graphics.

Dashboards use web-based technologies to present key business performance indicators or information in a single integrated view, generally using graphics that are clear, concise, and easy to understand.

**Data analytics -** A subset of business intelligence functions that encompasses a wide range of mathematical, statistical, and modeling techniques with the purpose of extracting knowledge from data.

Data analytics is a subset ofbusiness intelligence (BI) functionality that encompasses a wide range of mathematical, statistical, and modeling techniques with the purpose of extracting knowledge from data. Data analytics is used at all levels within the BI framework, including queries and reporting, monitoring and alerting, and data visualization. Hence, data ana-lytics is a “shared” service that is crucial to what BI adds to an organization. Data analytics represents what business managers really want from BI: the ability to extract actionable business insight from current events and foresee future problems or opportunities. Data analytics discovers characteristics, relationships, dependencies, or trends in the organization’s data, and then explains the discoveries and predicts future events based on the discoveries. In practice, data analytics is better understood as a continuous spec-trum of knowledge acquisition that goes from discovery to explanation to prediction. The outcomes of data analytics then become part of the information framework on which decisions are built. Data analytics tools can be grouped into two separate (but closely related and often overlapping) areas: Explanatory and predictive

**Data cube -** The multidimensional data structure used to store and manipulate data in a multidimensional DBMS. The location of each data value in the data cube is based on its x-, y-, and z-axes. Data cubes are static, meaning they must be created before they are used, so they cannot be created by an ad hoc query.

Conceptually, MDBMS end users visualize the stored data as a three-dimensional cube known as a data cube. The location of each data value in the data cube is a func-tion of the x-, y-, and z-axes in a three-dimensional space. The three axes represent the dimensions of the data value. The data cubes can grow to n number of dimensions, thus becoming hypercubes. Data cubes are created by extracting data from the operational databases or from the data warehouse. One important characteristic of data cubes is that they are static; that is, they are not subject to change and must be created before they can be used. Data cubes cannot be created by ad hoc queries. Instead, you query precreated cubes with defined axes; for example, a cube for sales will have the product, location, and time dimensions, and you can query only those dimensions. Therefore, the data cube creation process is critical and requires in-depth front-end design work.

**Data extraction -** A process used to extract and validate data from an operational database and external data sources prior to their placement in a data warehouse.

**Data filtering -** A process used to extract and validate data from an operational database and external data sources prior to their placement in a data warehouse.

**Data mart -** A small, single-subject data warehouse subset that provides decision support to a small group of people.

A data mart is a small, single-subject data warehouse subset that provides decision sup-port to a small group of people. In addition, a data mart could be created from data extracted from a larger data warehouse for the specific purpose of supporting faster data access to a target group or function. That is, data marts and data warehouses can coexist within a business intelligence environment.

**Data mining -** A process that employs automated tools to analyze data in a data warehouse and other sources and to proactively identify possible relationships and anomalies.

Data mining refers to analyzing massive amounts of data to uncover hidden trends, pat-terns, and relationships; to form computer models to simulate and explain the findings; and then to use such models to support business decision making. In other words, data mining focuses on the discovery and explanation stages of knowledge acquisition.

To put data mining in perspective, look at the pyramid in Figure 13.18, which represents how knowledge is extracted from data. Data forms the pyramid base and represents what most organizations collect in their operational databases. The second level contains infor-mation that represents the purified and processed data. Information forms the basis for decision making and business understanding. Knowledge is found at the pyramid’s apex and represents highly distilled information that provides concise, actionable business insight

Despite the lack of precise standards, data mining consists of four general phases: • Data preparation • Data analysis and classification • Knowledge acquisition • Prognosis In the data preparation phase, the main data sets to be used by the data-mining oper-ation are identified and cleansed of any data impurities. Because the data in the data warehouse is already integrated and filtered, the data warehouse usually is the target set for data-mining operations. 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 algo-rithms to find: • Data groupings, classifications, clusters, or sequences • Data dependencies, links, or relationships • Data patterns, trends, and deviations The knowledge acquisition phase uses the results of the data analysis and classifica-tion phase. During the knowledge acquisition phase, the data-mining tool (with possible intervention by the end user) selects the appropriate modeling or knowledge acquisi-tion algorithms. The most common algorithms used in data mining are based on neural networks, decision trees, rules induction, genetic algorithms, classification and regres-sion trees, memory-based reasoning, and nearest neighbor. A data-mining tool may use 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 focus on the knowledge–discovery phase, others continue to the prognosis phase. In that phase, the data-mining findings are used to pre-dict future behavior and forecast business outcomes. Examples of data-mining findings can be: • Sixty-five percent of customers who did not use a particular credit card in the last six months are 88 percent likely to cancel that account. • Eighty-two percent of customers who bought a 42-inch or larger LCD TV are 90 per-cent likely to buy an entertainment center within the next four weeks. • If age < 30, income <= 25,000, credit rating < 3, and credit amount > 25,000, then the minimum loan term is 10 years. The complete set of findings can be represented in a decision tree, a neural network, a forecasting model, or a visual presentation interface that is used to project future events or results. For example, the prognosis phase might project the likely outcome of a new product rollout or a new marketing promotion. Figure 13.19 illustrates the different phases of the data-mining process. Because of the nature of the data-mining process, some findings might fall out-side the boundaries of what business managers expect. For example, a data-mining tool might find a close relationship between a customer’s favorite brand of soda and the brand of tires on the customer’s car. Clearly, that relationship might not be held in high regard among sales managers. (In regression analysis, those relationships are commonly described by the label “idiot correlation.”) Fortunately, data mining usu-ally yields more meaningful results. In fact, data mining has proven helpful in finding practical relationships among data that help define customer buying patterns, improve product development and acceptance, reduce health care fraud, analyze stock markets, and so on. Data mining can be run in two modes: • Guided. The end user guides the data-mining tool step by step to explore and explain known patterns or relationships. In this mode, the end user decides what techniques to apply to the data. • Automated. In this mode, the end user sets up the data-mining tool to run automat-ically and uncover hidden patterns, trends, and relationships. The data-mining tool applies multiple techniques to find significant relationships.

**Data store -** The component of the decision support system that acts as a database for storage of business data and business model data. The data in the data store have already been extracted and filtered from the external and operational data, and will be stored for access by the end-user query tool for the business data model.

**Data Visualization**

Data visualization is the process of abstracting data to provide a visual data representa-tion that enhances the user’s ability to comprehend the meaning of the data. The goal of data visualization is to allow the user to quickly and efficiently see the data’s big picture by identifying trends, patterns and relationships

explore and discover data insights by applying: • Pattern recognition: visually identifying trends, distribution, and relationships • Spatial awareness: Use of size and orientation to compare and relate data • Aesthetics: Use of shapes and colors to highlight and contrast data composition and relationships

Qualitative vs quantitative

**Data warehouse -** An integrated, subject-oriented, time-variant, nonvolatile collection of data that provides support for decision making, according to Bill Inmon, the acknowledged father of the data warehouse.

Bill Inmon, the acknowledged “father” of the data warehouse, defines the term as “an integrated, subject-oriented, time-variant, nonvolatile collection of data that provides support for decision making.”

• Integrated. The data warehouse is a centralized, consolidated database that inte-grates data derived from the entire organization and from multiple sources with diverse formats. Data integration implies that all business entities, data elements, data characteristics, and business metrics are described in the same way throughout the enterprise. Although this requirement sounds logical, you would be amazed to discover how many different measurements for “sales performance” can exist within an organization; the same scenario can be true for any other business element. For instance, the status of an order might be indicated with text labels such as “open,” “received,” “ canceled,” and “closed” in one department and as “1,” “2,” “3,” and “4” in another department. A student’s status might be defined as “freshman,” “sophomore,” “junior,” or “senior” in the accounting department and as “FR,” “SO,” “JR,” or “SR” in the computer information systems department. To avoid the potential format tangle, the data in the data warehouse must conform to a common format that is accept-able throughout the organization. This integration can be time-consuming, but once accomplished, it enhances decision making and helps managers better understand the company’s operations. This understanding can be translated into recognition of strategic business opportunities. • Subject-oriented. Data warehouse data is arranged and optimized to provide answers to questions from diverse functional areas within a company. Data warehouse data is organized and summarized by topic, such as sales, marketing, finance, distribution, and transportation. For each topic, the data warehouse con-tains specific subjects of interest—products, customers, departments, regions, promotions, and so on. This form of data organization is quite different from the more functional or process-oriented organization of typical transaction systems. For example, an invoicing system designer concentrates on designing normalized data structures to support the business process by storing invoice components in two tables: INVOICE and INVLINE. In contrast, the data warehouse has a subject orientation. Data warehouse designers focus specifically on the data rather than on the processes that modify the data. (After all, data warehouse data is not sub-ject to numerous real-time data updates!) Therefore, instead of storing an invoice, the data warehouse stores its “sales by product” and “sales by customer” compo-nents because decision support activities require the retrieval of sales summaries by product or customer. • Time-variant. In contrast to operational data, which focuses on current transactions, warehouse data represents the flow of data through time. The data warehouse can even contain projected data generated through statistical and other models. It is also time-variant in the sense that when data is periodically uploaded to the data ware-house, all time-dependent aggregations are recomputed. For example, when data for previous weekly sales is uploaded to the data warehouse, the weekly, monthly, yearly, and other time-dependent aggregates for products, customers, stores, and other vari-ables are also updated. Because data in a data warehouse constitutes a snapshot of the company history as measured by its variables, the time component is crucial. The data warehouse contains a time ID that is used to generate summaries and aggregations by week, month, quarter, year, and so on. Once the data enters the data warehouse, the time ID assigned to the data cannot be changed. • Nonvolatile. Once data enters the data warehouse, it is never removed. Because the data in the warehouse represents the company’s history, the operational data, which represents the near-term history, is always added to it. Because data is never deleted and new data is continually added, the data warehouse is always growing. Therefore, the DBMS must be able to support multiterabyte or greater databases operating on multiprocessor hardware.

12 rules for the data warehouse:

1. The data warehouse and operational environments are separated.
2. The data warehouse data is integrated.
3. The data warehouse contains historical data over a long time.
4. The data warehouse data is snapshot data captured at a given point in time.
5. The data warehouse data is subject oriented.
6. The data warehouse data is mainly read-only with periodic batch updates from operational data. No online updates are allowed.
7. The data warehouse development life cycle differs from classical systems development. Data warehouse development is data-driven; the classical approach is process-driven.
8. The data warehouse contains data with several levels of detail: current detail data, old detail data, lightly summarized data, and highly summarized data.
9. The data warehouse environment is characterized by read-only transactions to very large data sets. The operational environment is characterized by numerous update transactions to a few data entities at a time.
10. The data warehouse environment has a system that traces data sources, transformations, and storage.
11. The data warehouse’s metadata is a critical component of this environment. The metadata identifies and defines all data elements. The metadata provides the source, transformation, integration, storage, usage, relationships, and history of each data element.
12. The data warehouse contains a chargeback mechanism for resource usage that enforces optimal use of the data by end users.

**Decision Support System (DSS) -** An arrangement of computerized tools used to assist managerial decision making within a business.

decision support system (Dss) is an arrangement of computerized tools used to assist managerial decision making. A DSS typically has a much narrower focus and reach than a BI solution. At first, decision support systems were the realm of a few selected managers in an organization. Over time, and with the introduction of the desktop computer, decision support systems migrated to more agile platforms, such as midrange computers, high-end servers, commodity servers, appliances, and cloud-based offerings. This evolution effectively changed the reach of decision support systems; BI is no longer limited to a small group of top-level managers with training in statistical mod-eling. Instead, BI is now available to all users in an organization, from line managers to the shop floor to mobile agents in the field.

**Dimension tables -** In a data warehouse, tables used to search, filter, or classify facts within a star schema. The fact table is in a one-to-many relationship with dimension tables.

Such dimensions are normally stored in dimension tables. Figure 13.5 depicts a star schema for sales with product, location, and time dimensions.

**Dimensions -** In a star schema design, qualifying characteristics that provide additional perspectives to a given fact.

Dimensions are qualifying characteristics that provide additional perspectives to a given fact. Recall that dimensions are of interest because decision support data is almost always viewed in relation to other data.

Dimensions provide descriptive characteristics about the facts through their attributes.

**Drill down** - To decompose data into more atomic components. That is, data at lower levels of aggregation. This approach is used primarily in a decision support system to focus on specific geographic areas, business types, and so on.

Decompose the data into more atomic components

**End-user presentation tool -** A data analysis tool that organizes and presents selected data compiled by the end-user query tool.

**End-user query tool** - A data analysis tool used to create the queries that access desired information from the data store.

**Explanatory analytics -** Data analysis that provides ways to discover relationships, trends, and patterns among data.

• explanatory analytics focuses on discovering and explaining data characteristics and relationships based on existing data. Explanatory analytics uses statistical tools to formulate hypotheses, test them, and answer the how and why of such relationships—for example, how do past sales relate to previous customer promotions?

**Extraction, Transformation, and Loading (ETL) -** In a data warehousing environment, the integrated processes of getting data from original sources into the data warehouse. ETL includes retrieving data from original data sources, manipulating the data into an appropriate form, and storing the data in the data warehouse.

Data extraction, transformation, and loading (etl) tools collect, filter, integrate, and aggregate internal and external data to be saved into a data store optimized for decision support

**Fact table -** In a data warehouse, the star schema table that contains facts linked and classified through their common dimensions. A fact table is in a one-to-many relationship with each associated dimension table.

Facts are normally stored in a fact table that is the center of the star schema. The fact table contains facts that are linked through their dimensions, which are explained in the next section.

**Facts -** In a data warehouse, the measurements that represent a specific business aspect or activity. For example, sales figures are numeric measurements that represent product or service sales. Facts commonly used in business data analysis include units, costs, prices, and revenues.

facts are numeric measurements (values) that represent a specific business aspect or activity. For example, sales figures are numeric measurements that represent product and service sales. Facts commonly used in business data analysis are units, costs, prices, and revenues.

**Governance -** In business intelligence, the methods for controlling and monitoring business health and promoting consistent decision making.

Governance is a method or process of government

**Key Performance Indicators (KPIs)** - In business intelligence, quantifiable numeric or scale-based measurements that assess a company's effectiveness or success in reaching strategic and operational goals. Examples of KPI are product turnovers, sales by promotion, sales by employee, and earnings per share.

Key performance indicators (KpIs) are quantifiable numeric or scale-based measurements that assess the company’s effectiveness or success in reaching its strategic and operational goals. Many different KPIs are used by different industries. Some examples of KPIs are: • General. Year-to-year measurements of profit by line of business, same-store sales, product turnovers, product recalls, sales by promotion, and sales by employee • Finance. Earnings per share, profit margin, revenue per employee, percentage of sales to account receivables, and assets to sales • Human resources. Applicants to job openings, employee turnover, and employee longevity • Education. Graduation rates, number of incoming freshmen, student retention rates, publication rates, and teaching evaluation scores KPIs are determined after the main strategic, tactical, and operational goals are defined for a business. To tie the KPI to the strategic master plan of an organization, a KPI is compared to a desired goal within a specific time frame.

**Master Data Management (MDM) -** In business intelligence, a collection of concepts, techniques, and processes for the proper identification, definition, and management of data elements within an organization.

master data management (mDm) is a collection of concepts, techniques, and processes for the proper identification, definition, and management of data elements within an organization. MDM’s main goal is to provide a comprehensive and consistent definition of all data within an organization. MDM ensures that all company resources (people, procedures, and IT systems) that work with data have uniform and consistent views of the company’s data.

**Materialized view -** A dynamic table that not only contains the SQL query command to generate rows but stores the actual rows. The materialized view is created the first time the query is run and the summary rows are stored in the table. The materialized view rows are automatically updated when the base tables are updated.

A materialized view is a dynamic table that not only contains the SQL query command to generate the rows, it stores the actual rows. The materialized view is cre-ated the first time the query is run, and the summary rows are stored in the table. The materialized view rows are automatically updated when the base tables are updated. That way, the data warehouse administrator will create the view but will not have to worry about updating the view. The use of materialized views is totally transparent to the end user. The OLAP end user can create OLAP queries using the standard fact tables, and the DBMS query optimization feature will automatically use the materialized views if they provide better performance.

**Metrics** - In a data warehouse, numeric facts that measure a business characteristic of interest to the end user.

Facts can also be computed or derived at run time. Such computed or derived facts are sometimes called metrics to differentiate them from stored facts. The fact table is updated periodically with data from operational databases.

**Multidimensional database management system -** A database management system that uses proprietary techniques to store data in matrixlike arrays of n dimensions known as cubes.

An MDBMS uses proprietary techniques to store data in matrix-like n-dimensional arrays

**Multidimensional online analytical processing -** An extension of online analytical processing to multidimensional database management systems.

MOLAP’s premise is that multidimensional databases are best suited to manage, store, and analyze multidimensional data.

**Online Analytical Processing (OLAP) -** Decision support system tools that use multidimensional data analysis techniques. OLAP creates an advanced data analysis environment that supports decision making, business modeling, and operations research.

online analytical processing (olAp) is a BI style whose systems share three main characteristics: • Multidimensional data analysis techniques

• Advanced data presentation functions. These functions include 3D graphics, pivot tables, crosstabs, data rotation, and three-dimensional cubes. Such tools are compatible with desktop spreadsheets, statistical packages, and query and report packages.

• Advanced data aggregation, consolidation, and classification functions. These allow the data analyst to create multiple data aggregation levels, slice and dice data (see Section 13-5c), and drill down and roll up data across different dimensions and aggre-gation levels. For example, aggregating data by week, month, quarter, and year allows the data analyst to drill down and roll up across time dimensions. • Advanced computational functions. These include business-oriented variables such as market share, period comparisons, sales margins, product margins, and percentage changes; financial and accounting ratios, including profitability, overhead, cost allo-cations, and returns; and statistical and forecasting functions. These functions are provided automatically, so the end user does not need to redefine the components each time they are accessed. • Advanced data-modeling functions. These provide support for what-if scenarios, variable assessment, contributions to outcome, linear programming, and predictive

• Advanced database support

To deliver efficient decision support, OLAP tools must have the following advanced data access features: • Access to many different kinds of DBMSs, flat files, and internal and external data sources • Access to aggregated data warehouse data as well as to the detail data found in operational databases • Advanced data navigation features such as drill-down and roll-up • Rapid and consistent query response times • The ability to map end-user requests, expressed in either business or model terms, to the appropriate data source and then to the proper data access language (usually SQL). The query code must be optimized to match the data source, regardless of whether the source is operational or data warehouse data. • Support for very large databases. As explained earlier, the data warehouse could easily and quickly grow to multiple terabytes in size. To provide a seamless interface, OLAP tools map the data elements from the data warehouse and the operational database to their own data dictionaries. This metadata is used to translate end-user data analysis requests into the proper (optimized) query codes, which are then directed to the appropriate data sources.

• Easy-to-use end-user interfaces

OLAP architecture

The OLAP architecture is designed to meet ease-of-use requirements while keeping the system flexible. An OLAP system has three main architectural components: • Graphical user interface (GUI) • Analytical processing logic • Data-processing logic

**Partitioning -** The process of splitting a table into subsets of rows or columns.

partitioning splits a table into subsets of rows or columns and places the subsets close to the client computer to improve data access time.

**Periodicity -** Information about the time span of data stored in a table, usually expressed as current year only, previous years, or all years.

periodicity, which is usually expressed as current year only, previous years, or all years, provides information about the time span of the data stored in the table.

**Portal -** In terms of business intelligence, a unified, single point of entry for information distribution.

portals provide a unified, single point of entry for information distribution. Portals are a web-based technology that use a web browser to integrate data from multiple sources into a single webpage. Many different types of BI functionality can be accessed through a portal.

**Predictive analytics -** Data analytics that use advanced statistical and modeling techniques to predict future business outcomes with great accuracy.

• predictive analytics focuses on predicting future data outcomes with a high degree of accuracy. Predictive analytics uses sophisticated statistical tools to help the end user create advanced models that answer questions about future data occurrences—for example, what would next month’s sales be based on a given customer promotion?

**Relational Online Analytical Processing (ROLAP) -** Analytical processing functions that use relational databases and familiar relational query tools to store and analyze multidimensional data.

relational online analytical processing (rolAp) provides OLAP functionality by using relational databases and familiar relational query tools to store and analyze multidi-mensional data. This approach builds on existing relational technologies and represents a natural extension to companies that already use relational database management systems within their organizations. ROLAP adds the following extensions to traditional RDBMS technology: • Multidimensional data schema support within the RDBMS • Data access language and query performance optimized for multidimensional data • Support for very large databases (VLDBs)

**Replication -** The process of creating and managing duplicate versions of a database. Replication is used to place copies in different locations and to improve access time and fault tolerance.

replication makes a copy of a table or partition and places it in a different location, also to improve access time.

**Roll up** -  In SQL, an OLAP extension used with the GROUP BY clause to aggregate data by different dimensions. Rolling up the data is the exact opposite of drilling down the data.

Aggregating the data to a higher level

**Slice and dice -** The ability to cut slices off a data cube to perform a more detailed analysis.

Whatever the underlying database technology, one of the main features of multidi-mensional analysis is its ability to focus on specific “slices” of the cube. For example, the product manager may be interested in examining the sales of a product while the store manager is interested in examining the sales made by a particular store. In multidimen-sional terms, the ability to focus on slices of the cube to perform a more detailed anal-ysis is known as slice and dice. Figure 13.7 illustrates the slice-and-dice concept; note that each cut across the cube yields a slice. Intersecting slices produce small cubes that constitute the “dice” part of the slice-and-dice operation.

**Snowflake schema -** A type of star schema in which dimension tables can have their own dimension tables. The snowflake schema is usually the result of normalizing dimension tables.

a snowflake schema, which is a type of star schema in which the dimension tables can have their own dimension tables. The snowflake schema is usually the result of normal-izing dimension tables.

**Sparsity -** In multidimensional data analysis, a measurement of the data density held in the data cube.

sparsity measures the density of the data held in the data cube; it is computed by dividing the total number of actual values in the cube by its total number of cells.

**Star schema -**  A data modeling technique used to map multidimensional decision support data into a relational database. The star schema represents data using a central table known as a fact table in a 1:M relationship with one or more dimension tables.

The star schema is a data-modeling technique used to map multidimensional deci-sion support data into a relational database. In effect, the star schema creates the near equivalent of a multidimensional database schema from the existing relational database. Star schemas yield an easily implemented model for multidimensional data analysis while preserving the relational structures on which the operational database is built. The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies.

The following four techniques are often used to optimize data warehouse design: • Normalizing dimensional tables • Maintaining multiple fact tables to represent different aggregation levels • Denormalizing fact tables • Partitioning and replicating tables

**Very Large Databases (VLDBs) -** Databases that contain huge amounts of datagigabyte, terabyte, and petabyte ranges are not unusual.

 To support a VLDB adequately, the DBMS might be required to support advanced storage technologies, and even more importantly, to support multiple-processor technologies, such as a symmetric multiprocessor (SMP) or a massively parallel processor (MPP).

Summary • Business intelligence (BI) is a term for a comprehensive, cohesive, and integrated set of applications used to capture, collect, integrate, store, and analyze data with the purpose of generating and presenting information to support business decision making. • Decision support systems (DSSs) refer to an arrangement of computerized tools used to assist managerial decision making within a business. DSSs were the original precursor of current-generation BI systems. • Operational data is not well suited for decision support. From the end user’s point of view, decision support data differs from operational data in three main areas: time span, granularity, and dimensionality. • The data warehouse is an integrated, subject-oriented, time-variant, nonvolatile collection of data that provides support for decision making. The data warehouse is usually a read-only database optimized for data analysis and query processing. A data mart is a small, single-subject data warehouse subset that provides decision support to a small group of people. • The star schema is a data-modeling technique used to map multidimensional deci-sion support data into a relational database for advanced data analysis. The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies. Facts are numeric measurements or values that represent a specific business aspect or activity. Dimensions are general qualifying categories that provide additional per-spectives to facts. Conceptually, the multidimensional data model is best represented by a three-dimensional cube. Attributes can be ordered in well-defined hierarchies, which provide a top-down organization that is used for two main purposes: to permit aggregation and provide drill-down and roll-up data analysis. • Online analytical processing (OLAP) refers to an advanced data analysis environment that supports decision making, business modeling, and operations research. • Data analytics is a subset of BI functionality that provides advanced data analysis tools to extract knowledge from business data. Data analytics can be divided into explanatory and predictive analytics. Explanatory analytics focuses on discovering and explaining data characteristics and relationships. Predictive analytics focuses on creating models to predict future outcomes or events based on the existing data. • Data mining automates the analysis of operational data to find previously unknown data characteristics, relationships, dependencies, and trends. The data-mining pro-cess has four phases: data preparation, data analysis and classification, knowledge acquisition, and prognosis. • SQL has been enhanced with analytic functions that support OLAP-type processing and data generation. • Data visualization provides visual representations of data that enhance the user’s abil-ity to comprehend the meaning of the data.

**4. Chapter 16**

**Access plan -**  A set of instructions generated at application compilation time that is created and managed by a DBMS. The access plan predetermines how an application's query will access the database at run time.

**Active data dictionary -** A data dictionary that is automatically updated by the database management system every time the database is accessed, thereby keeping its information current.

**Audit log -** A security feature of a database management system that automatically records a brief description of the database operations performed by all users.

**Authorization management -** Procedures that protect and guarantee database security and integrity. Such procedures include user access management, view definition, DBMS access control, and DBMS usage monitoring.

**Availability -** In the context of data security, it refers to the accessibility of data whenever required by authorized users and for authorized purposes.

**Back-end CASE tools -** A computer-aided software tool that provides support for the coding and implementation phases of the SDLC. In comparison, front-end CASE tools provide support for the planning, analysis, and design phases.

**Compliance -** Activities that meet data privacy and security reporting guidelines or requirements.

**Computer-Aided Systems Engineering (CASE) -** Tools used to automate part or all of the Systems Development Life Cycle.

**Concurrent backup -** A backup that takes place while one or more users are working on a database.

**Confidentiality** - In the context of data security, ensuring that data are protected against unauthorized access, and if the data are accessed by an authorized user, that the data are used only for an authorized purpose.

**Data Administrator (DA) -** The person responsible for managing the entire data resource, whether it is computerized or not. The DA has broader authority and responsibility than the database administrator.

**Data profiling software -** Programs that analyze data and metadata to determine patterns that can help assess data quality.

**Data quality -** A comprehensive approach to ensuring the accuracy, validity, and timeliness of data.

**Data-profiling - S**oftware Programs that analyze data and metadata to determine patterns that can help assess data quality.

**Database Administrator (DBA)** - The person responsible for planning, organizing, controlling, and monitoring the centralized and shared corporate database. The DBA is the general manager of the database administration department.

**Database instance -** In an Oracle DBMS, the collection of processes and data structures used to manage a specific database.

**Database object** - Any object in a database, such as a table, view, index, stored procedure, or trigger.

**Database security -** The use of DBMS features and other related measures to comply with the security requirements of an organization.

**Database Security Officer** **(DSO)** - The person responsible for the security, integrity, backup, and recovery of the database.

**Dirty data -** Data that contain inaccuracies and/or inconsistencies.

**Disaster management -** The set of DBA activities dedicated to securing data availability following a physical disaster or a database integrity failure.

**Enterprise database** - The overall company data representation, which provides support for present and expected future needs.

**Front-end CASE tools** - A computer-aided software tool that provides support for the planning, analysis, and design phases of the SDLC. In comparison, back-end CASE tools provide support for the coding and implementation phases.

**Full backup (database dump)** - A complete copy of an entire database saved and periodically updated in a separate memory location. A full backup ensures a full recovery of all data after a physical disaster or database integrity failure.

**Incremental backup** - A process that only backs up data that has changed in the database since the last incremental or full backup.

**Information Engineering (IE) -** A methodology that translates a company's strategic goals into helpful data and applications. IE focuses on the description of corporate data instead of the processes.

**Information Systems Architecture (ISA) -** The output of the information engineering process that serves as the basis for planning, developing, and controlling future information systems.

**Information Systems (IS) Department** - An evolution of the data-processing department in which responsibilities are broadened to include service and production functions.

**Integrity -** In a data security framework, refers to keeping data consistent and free of errors or anomalies.

**Master Data Management (MDM)** - In business intelligence, a collection of concepts, techniques, and processes for the proper identification, definition, and management of data elements within an organization.

**Passive data dictionary** - A DBMS data dictionary that requires a command initiated by an end user to update its data access statistics.

**Policies -** General statements of direction that are used to manage company operations through the communication and support of the organization's objectives.

**Privacy -** The rights of individuals and organizations to determine access to data about themselves.

**Procedures -**  Series of steps to be followed during the performance of an activity or process.

**Profile -** In Oracle, a named collection of settings that controls how much of the database resource a given user can use.

**Role -** In Oracle, a named collection of database access privileges that authorize a user to connect to a database and use its system resources.

**Schema -** A logical grouping of database objects, such as tables, indexes, views, and queries, that are related to each other. Usually, a schema belongs to a single user or application.

**Security -** Activities and measures to ensure the confidentiality, integrity, and availability of an information system and its main asset, data.

**Security breach - A**n event in which a security threat is exploited to endanger the integrity, confidentiality, or availability of the system.

**Security policy -** A collection of standards, policies, and procedures created to guarantee the security of a system and ensure auditing and compliance.

**Security threat -** An imminent security violation that could occur due to unchecked security vulnerabilities.

**Security vulnerability -** A weakness in a system component that could be exploited to allow unauthorized access or cause service disruptions.

**Standards -** A detailed and specific set of instructions that describes the minimum requirements for a given activity. Standards are used to evaluate the quality of the output.

**Systems administrator** - The person responsible for coordinating an organization's data-processing activities.

**Tablespace -** In a DBMS, a logical storage space used to group related data. Also known as a file group.

**User -** In a system, a uniquely identifiable object that allows a given person or process to log on to the database.