Chapter 8

Database Design

Database Systems:
In this chapter, you will learn:

• That successful database design must reflect the information system of which the database is a part
• That successful information systems are developed within a framework known as the Systems Development Life Cycle (SDLC)
• That, within the information system, the most successful databases are subject to frequent evaluation and revision within a framework known as the Database Life Cycle (DBLC)
• How to conduct evaluation and revision within the SDLC and DBLC frameworks
• What database design strategies exist:
  – Top-down vs. bottom-up design
  – Centralized vs. decentralized design
Changing Data into Information

• Proper collection, storage, & retrieval of data are essential to successful operation of any organization
• Design of effective databases is crucial
• Data
  – Raw facts stored in databases
  – Need additional processing to become useful
  – Examples:
    • Age, gender, income, …
• Information
  – Required by decision maker
  – Data processed and presented in a meaningful form
  – Can be as simple as tabulating the data
    • Makes certain data patterns more obvious
  – Example:
    • How many employees have over $50,000 income
• Transformation
  – Changing data into information
  – Can be
    • Very simple (tables)
    • Very complex (statistics)
Data vs. Information

Figure 8.1 Generating Information for Decision Making
Transforming Data into Information

• A Simple Cross-Classification Table Example:
  – Age and gender distribution for 9,040 clients
  – Most frequently used form
  – Usually used in conjunction with other organizational structures

<table>
<thead>
<tr>
<th></th>
<th>UNDER 25</th>
<th>25 – 45</th>
<th>46 – 60</th>
<th>61 AND OVER</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>119</td>
<td>1,892</td>
<td>2,641</td>
<td>876</td>
<td>5,528</td>
</tr>
<tr>
<td>Female</td>
<td>48</td>
<td>1,117</td>
<td>1,805</td>
<td>542</td>
<td>3,512</td>
</tr>
<tr>
<td>Total</td>
<td>167</td>
<td>3,009</td>
<td>4,446</td>
<td>1,418</td>
<td>9,040</td>
</tr>
</tbody>
</table>
Information System

• Database
  – Carefully designed and constructed repository of facts
  – Part of an information system
  – Provides
    • Data collection
    • Data storage
    • Data retrieval

• Information System
  – Wikipedia
  – Provides for data collection, storage, and retrieval
  – Facilitates
    • Data transformation
    • Data management
    • Information management
  – 6 Components:
    • People
    • Hardware
    • Software
    • Database(s)
    • Application programs
    • Procedures
System Analysis / Development

- **Systems analysis**
  - Process that establishes need for and extent of an information system
  - Wikipedia
  - eLook.org

- **Systems development**
  - Process of creating an information system

- **System Development Life Cycle (SDLC)**
  - Traces history (life cycle) of an information system
  - Provides “big picture” within which database design and application development can be mapped out and evaluated
  - Dictionary definitions:
    - Wikipedia
    - FOLDOC
Applications

• Transform data into information that forms the basis for decision making
• Usually produce
  – Formal report
  – Tabulations
  – Graphic displays
• Composed of two parts
  – Data
  – Code by which the data are transformed into information
Database Development

1. Performance depends on 3 factors:
   - Database design and implementation
   - Application design and implementation
   - Administrative procedures

2. Database development
   - Process of database design and implementation

3. Goal:
   - Creation of database models
     • Create complete, normalized, non-redundant, & fully functional conceptual, logical, & physical DB model
   - Database implementation
     • Creating storage structure
     • Loading data into database
     • Providing for data management

4. Regardless of the size of the database, you must plan, analyze, & design.
5. Procedures used might differ in scale
6. Example:
   - Designing a house vs. designing a space shuttle
Systems Development Life Cycle (SDLC)

- Traditionally 5 phases
  1. Planning
  2. Analysis
  3. Detailed systems design
  4. Implementation
  5. Maintenance
- Iterative rather than sequential process
- Sometime called the “Waterfall model”
  - [www.albany.edu/acc/courses/fall97/acc681/ch7.html](http://www.albany.edu/acc/courses/fall97/acc681/ch7.html)
## Systems Development Life Cycle (SDLC)

**Figure 8.2 The Systems Development Life Cycle (SDLC)**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Action(s)</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Initial assessment, Feasibility study</td>
<td>8.3.1</td>
</tr>
<tr>
<td>Analysis</td>
<td>User requirements, Existing system evaluation, Logical system design</td>
<td>8.3.2</td>
</tr>
<tr>
<td>Detailed systems design</td>
<td>Detailed system specification</td>
<td>8.3.3</td>
</tr>
<tr>
<td>Implementation</td>
<td>Coding, testing, and debugging, Installation, fine-tuning</td>
<td>8.3.4</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Evaluation, Maintenance, Enhancement</td>
<td>8.3.5</td>
</tr>
</tbody>
</table>
Planning

• Yields general overview of the company and its objectives
• Initial assessment made of information-flow- and-extent requirements
  – Should existing system be continued, modified, or replaced?
• Participants must begin to study and evaluate alternate solutions
  – Technical aspects of hardware and software requirements
    • Hardware and software requirements
  – System cost
Analysis

• Problems defined during the planning phase are examined in greater detail during analysis.

• A thorough audit of user requirements
  – Make sure that those requirements fit into overall information requirements
  – End user & system designer work together

• Existing hardware and software systems are studied.

• Goal:
  – Better understanding of system’s functional areas, actual and potential problems, and opportunities.
Analysis

• Tools used:
  – Data Flow Diagrams
    • FOLDOC (DFD)
    • Google search
    • http://www.smartdraw.com/tutorials/software-dfd/dfd.htm
    • Data Flow Analysis
  – Hierarchical input process output (HIPO) charts
    • Flowcharts, pseudocode and HIPO charts
    • http://www.yourdon.com/books/msa2e/CH15/CH15.html
    • Google search
  – Entity-relationship diagrams (ERD)
Logical Systems Design

- Must specify appropriate conceptual data model, inputs, processes, and expected output requirements
- Might use tools such as data flow diagrams (DFD), (HIPO) diagrams, or entity relationship (ER) diagrams
- Yields functional descriptions of system’s components (modules) for each process within database environment
Detailed Systems Design

• System’s processes design is completed
  – Technical specifications for:
    • Screens
    • Menus
    • Reports
    • Other devices, if any
  – Steps to convert from old to new system
  – Training principles & methodology
  – Management approval
Implementation

• **Install** hardware, DBMS, & application programs

• Involve **coding, testing, & debugging**

• **Create actual database** (tables, views, user authorization, …)
  – Database may be loaded in interactive or batch modes
  – Debugging tools decrease testing & debugging time

• **Train** end users

• Continuous **evaluation & fine-tuning**
Maintenance

• Three types:
  – *Corrective maintenance* in response to systems errors
  – *Adaptive maintenance* due to changes in the business environment
  – *Perfective maintenance* to enhance the system

• Every request requires re-tracing the SDLC

• Every system has a *life span* depending on
  – Rapid change in technology
  – Maintenance cost

• Computer-assisted systems engineering (CASE) tools (e.g. Visio) make it possible to produce better systems within reasonable amount of time and at a reasonable cost
The Database Life Cycle (DBLC)

**Figure 8.3 The Database Life Cycle (DBLC)**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Action(s)</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database initial study</td>
<td>Analyze the company situation</td>
<td>8.4.1</td>
</tr>
<tr>
<td></td>
<td>Define problems and constraints</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define objectives</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define scope and boundaries</td>
<td></td>
</tr>
<tr>
<td>Database design</td>
<td>Create the conceptual design</td>
<td>8.4.2</td>
</tr>
<tr>
<td></td>
<td>DBMS software selection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create the logical design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create the physical design</td>
<td></td>
</tr>
<tr>
<td>Implementation and loading</td>
<td>Install the DBMS</td>
<td>8.4.3</td>
</tr>
<tr>
<td></td>
<td>Create the database(s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Load or convert the data</td>
<td></td>
</tr>
<tr>
<td>Testing and evaluation</td>
<td>Test the database</td>
<td>8.4.4</td>
</tr>
<tr>
<td></td>
<td>Fine-tune the database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evaluate the database and its application programs</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Produce the required information flow</td>
<td>8.4.5</td>
</tr>
<tr>
<td>Maintenance and evaluation</td>
<td>Introduce changes</td>
<td>8.4.6</td>
</tr>
<tr>
<td></td>
<td>Make enhancements</td>
<td></td>
</tr>
</tbody>
</table>
DBLC: 1. Database Initial Study

• Overall purpose:
  – Analyze the company situation
    • Operating environment
    • Organizational structure
  – Define problems and constraints
  – Define objectives
  – Define scope and boundaries

• Interactive and iterative processes required to complete the first phase of the DBLC successfully
Summary of Activities in the Database Initial Study

Figure 8.4 Summary of Activities in the Database Initial Study

- Analysis of the company situation
  - Company objectives
  - Company operations
  - Company structure

- Definition of problems and constraints

- Database system specifications
  - Objectives
  - Scope
  - Boundaries
Database Initial Study

• Initiated when designers are called in case the existing system failed
  – Must determine causes of failure
  – Needs excellent communication & interpersonal skills
Analyze the Company Situation

• Analysis
  – Wikipedia
  “To break up any whole into its parts so as to find out their
  nature, function, and so on”

• Company situation
  – General conditions in which a company operates, its
    organizational structure, and its mission

• Analyze the company situation
  – Discover what the company’s operational components are,
    how they function, and how they interact
Define Problems and Constraints

- Managerial view of company’s operation is often different from that of end users
  - Please read page 404!!
    - See the difference in view between designer, marketing manager, production manager, & machine operator
- Designer must continue to carefully probe to generate additional information that will help define problems within larger framework of company operations
- Finding precise answers is important
- Defining problems does not always lead to the perfect solution
Define Problems and Constraints

• Formal or informal sources of information
  – How does existing system function?
  – What input is required by the system?
  – What documents are generated?
  – How is system output used? By whom?
  – How does the marketing work?
  – How do different departments interact?
  – What are the constraints?
    • Budget, time, personnel, …
Define Objectives

• Designer must ensure that database system objectives correspond to those envisioned by end user(s)

• Designer must begin to address the following questions:
  – 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 data with other systems or users?

• Caution!!
  – Don’t try to solve the symptoms, but the cause of the problem
Define Scope and Boundaries

• Scope
  – Defines extent of design according to operational requirements
  – Helps define requirements
    • Required data structures
    • Type
    • Number of entities
    • Physical size of the database
  – Check what will the DB design encompass
    • Entire organization
    • One or more department
    • One or more function within a department
Define Scope and Boundaries

• Boundaries
  – Wikipedia
  – Limits external to the system
  – Often imposed by existing hardware and software
    • Limited budget
    • Have to use existing hardware / software
DBLC: 2. Database Design

• Most Critical DBLC phase
• Makes sure final product meets requirements
• Focus on data requirements
• Not a sequential process
• Continuous feedback necessary
• Necessary to concentrate on the data
• Characteristics required to build database model
• Two views of data within system:
  – Business view of data as information source
  – Designer’s view of data structure, its access, and the activities required to transform the data into information
Business Manager vs. Designer Views

**Figure 8.5** Two Views of Data: Business Manager and Designer

**Manager’s view**
- *What are the problems?*
- *What are the solutions?*
- *What information is needed to implement the solutions?*
- *What data are required to generate the desired information?*

**Designer’s view**
- *How must the data be structured?*
- *How will the data be accessed?*
- *How are the data transformed into information?*
Database Design

• Sub-phases
  – Create conceptual design
  – DBMS software selection
  – Create logical design
  – Create physical design

• We will discuss them after we learn about the other phases
Database Design

• Loosely related to analysis and design of larger system
• Systems analysts or systems programmers are in charge of designing other system components
  – Their activities create procedures that will help transform data within the database into useful information
• Iterative process that provides continuous feedback
Procedure Flow in Database Design

**FIGURE 8.6 PROCEDURE FLOW IN THE DATABASE DESIGN**

1. **Conceptual Design**
   - Database analysis and requirements
   - Entity relationship modeling and normalization
   - Data model verification
   - Distributed database design*

   - Determine end-user views, outputs, and transaction-processing requirements.
   - Identify main processes; insert, update, and delete rules. Validate reports, queries, views, integrity, sharing, and security.
   - Define the location of tables, access requirements, and fragmentation strategy.

2. **DBMS software selection**

3. **Logical design**

4. **Physical design**

* See Chapter 10, “Distributed Database Management Systems.”

DBMS-independent

DBMS-dependent

Hardware-dependent
DBLC: 3. Implementation and Loading

- Implementation of the logical design
- Requires creation of special storage-related constructs to house end-user tables.
- Includes
  - Create storage group
  - Create the database within the storage group
  - Assign rights (privileges) to use the database
  - Create table space(s) within the database
    - A table space may contain more than one table
  - Create table(s) within the table space
  - Assign access rights to the table spaces, tables, or views within a specific table space
  - Data loaded into tables
Implementation and Loading

• Other implementation activities for more sophisticated DBMS
  – Performance tuning
  – Security
  – Backup and recovery
  – Integrity
  – Company standards
  – Concurrency controls
Implementation and Loading

Example:
Performance

- Wikipedia
- One of the most important factors in certain database implementations
- Not all DBMSs have performance-monitoring and fine-tuning tools embedded in their software
- System & configuration parameters affect performance
  - Access path definition, indexes, buffer size, …
- There is no standard measurement for database performance
- Not only (nor even main) factor
Security

• Data must be protected from access by unauthorized users
• Must provide for the following:
  – Physical security
  – Password security
  – Access rights
  – Audit trails
  – Data encryption
  – Diskless workstations
Backup and Recovery

• **Backup**
• **Recovery**

• Database can be subject to data loss through unintended data deletion and power outages

• Data backup and recovery procedures
  – Create a safety valve
  – Allow database administrator to ensure availability of consistent data
Integrity

- Integrity
- Enforced through proper use of primary and foreign key rules
Company Standards

- Company requirements
- May partially define database standards
- Database administrator must implement and enforce such standards
Concurrency Control

- Feature that allows simultaneous access to a database while preserving data integrity
- Failure to maintain can quickly destroy a database’s effectiveness

<table>
<thead>
<tr>
<th>TABLE 8.6 THE NEED FOR CONCURRENCY CONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESS A (MEMORY)</td>
</tr>
<tr>
<td>Read 500</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Stock = 500 – 150</td>
</tr>
<tr>
<td>Stock = 500 – 300—write</td>
</tr>
</tbody>
</table>

Value stored in database: 200
DBLC: 4. Testing and Evaluation

• Occurs in parallel with applications programming
• Database tools used to prototype applications
• If implementation fails to meet some of the system’s evaluation criteria
  – Fine-tune specific system and DBMS configuration parameters
  – Modify the physical design
  – Modify the logical design
  – Upgrade or change the DBMS software and/or the hardware platform
DBLC: 5. Operation

• Once the database has passed the evaluation stage, it is considered operational
• Beginning of the operational phase starts the process of system evolution
• Unforeseen problems may surface
  – Some are minor problems, & some may need some patch work
• Demand for change is constant
DBLC: 6. Maintenance and Evolution

• Required periodic maintenance:
  – Preventive maintenance: Backup
  – Corrective maintenance: Recovery
  – Adaptive maintenance: Enhancing performance, adding entities & attributes, ...

• Assignment of access permissions and their maintenance for new and old users

• Generation of database access statistics

• Periodic security audits

• Periodic system-usage summaries
Parallel Activities DBLC and SDLC

- Many of the activities described in DBLC are similar too those in the SDLC
Conceptual Design
Conceptual Design

- High level of abstraction
  - Data modeling creates **abstract data structure** to represent real-world items
- Must embody a clear understanding of the **business** and its **functional areas**
- Software/Hardware-independent
- Focus on immediate & future data needs
- Minimal data rule:
  "All that is needed is there & all that is there is needed"
Conceptual Design

• Four steps

1. Conceptual Design
   - Database analysis and requirements
     - Determine end-user views, outputs, and transaction-processing requirements.
   - Entity relationship modeling and normalization
   - Data model verification
     - Identify main processes; insert, update, and delete rules. Validate reports, queries, views, integrity, sharing, and security.
   - Distributed database design*
     - Define the location of tables, access requirements, and fragmentation strategy.
I.1 Data Analysis and Requirements

• First step in conceptual design
• Discover **data element characteristics**
  – Obtains characteristics from different sources
• Must take into account business rules
  – Derived from description of operations
    • Document that provides precise, detailed, up-to-date, and thoroughly reviewed description of activities that define an organization’s operating environment
Data Analysis and Requirements

• Focus on:
  – Information needs
    • What input and output, & to what extent
  – Information users
    • Who will use it
    • How will it be used
    • What are the different end users’ views
  – Information sources
    • Where is information found
    • How is it extracted
  – Information constitution
    • Which data elements needed
    • What attributes
    • What relationships
    • What volume
    • What frequency
    • What transformations
Data Analysis and Requirements

• Data sources
  – Developing and gathering end-user data views
    • End user & designer interact to develop descriptions for end used data views
  – Direct observation of current system
    • Review existing system
    • Examine input forms & files
    • Examine current & desired reports
  – Interfacing with systems design group
    • If the designer works in a team
Data Analysis and Requirements

• Business rules
  – Helps to create/enforce actions
  – Brief, precise narrative description of organization environment’s policy, procedure, or a principle
  – Main characteristics of data as viewed by the company
  – If written properly, they define entities, attributes, relationships, connectivities, cardinalities, & constraints
  – Main source:
    • Company managers, policy makers, department manager, written company documentation, …
  – Other sources:
    • Interviews, questionnaires, …
Data Analysis and Requirements

• Business rules benefits
  – Standardize company’s view of data
  – Constitute a communication tool between users & designers
  – Allow designers to understand the nature, role, & scope and data
  – Allow designers to understand business processes
  – Allows designer to develop appropriate relationship participation rules & foreign key constraints

• You will find more about business rules in chapter 4
I.2 ER Modeling and Normalization

• Standardization
  – Designer must communicate and enforce appropriate standards to be used in the documentation of design
    • Use of diagrams and symbols
    • Documentation writing style
    • Layout
    • Other conventions to be followed during documentation
  – Failure to standardize means failure to communicate later
  – Designer must incorporate business rules into the conceptual model
Developing Conceptual Model Using ERD

- **Steps**
  - Some steps can work concurrently and some steps might demand more attributes or entities

**TABLE 8.2 DEVELOPING THE CONCEPTUAL MODEL USING ER DIAGRAMS**

<table>
<thead>
<tr>
<th>STEP</th>
<th>ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify, analyze, and refine the business rules.</td>
</tr>
<tr>
<td>2</td>
<td>Identify the main entities, using the results of Step 1.</td>
</tr>
<tr>
<td>3</td>
<td>Define the relationships among the entities, using the results of Steps 1 and 2.</td>
</tr>
<tr>
<td>4</td>
<td>Define the attributes, primary keys, and foreign keys for each of the entities.</td>
</tr>
<tr>
<td>5</td>
<td>Normalize the entities.</td>
</tr>
<tr>
<td>6</td>
<td>Complete the initial ER diagram.</td>
</tr>
<tr>
<td>7</td>
<td>Have the main end users verify the model in Step 6 against the data, information, and processing requirements.</td>
</tr>
<tr>
<td>8</td>
<td>Modify the ER diagram, using the results of Step 7.</td>
</tr>
</tbody>
</table>
ER Modeling Is an Iterative Process Based on Many Activities

**FIGURE 8.8** ER MODELING IS AN ITERATIVE PROCESS BASED ON MANY ACTIVITIES

Diagram showing the iterative process of ER modeling with steps including:
- Database initial study
- Data analysis (User views and Business rules)
- Initial ER model
- Verification
- Normalization
- Attributes
- Final ER model
Designer Responsibilities During ER Modeling Process

- **Define** entities, attributes, PK’s, FK’s
- **Make decisions** about
  - Adding new PK’s to satisfy end-user and/or processing requirements
  - Multi-valued attributes
  - Adding derived attributes
  - Placement of FK in 1:1 relationships
- **Avoid unnecessary ternary relationships**
- **Draw the ER diagram**
- **Normalize the data model**
- **Include all data element definitions in the data dictionary**
- **Designers use tools to make the job more effective & faster**
Conceptual Design Tools and Information Sources

**FIGURE 8.9 Conceptual Design Tools and Information Sources**

- **Information sources**
  - Business rules and data constraints
  - Data flow diagrams (DFD*)
  - Process functional descriptions (FD)* (user views)

- **Design tools**
  - ER diagram
  - Normalization
  - Data dictionary

- **Conceptual model**
  - ERD

* Output generated by the systems analysis and design activities
Data Dictionary

• Defines all objects (entities, attributes, relations, views, and so on)
• Used in tandem with the normalization process to help eliminate data anomalies and redundancy problems
I.3 Data Model Verification

- Model must be verified against proposed system processes to confirm that intended processes can be supported by database model
- Revision of original design starts with a careful re-evaluation of entities, followed by a detailed examination of attributes that describe these entities
- Define design’s major components as *modules*, each handling a specific function
Data Model Verification

• A series of tests should be made regarding
  – End user views and required transactions
    • Select, insert, delete, & update operations
    • Queries & reports
  – Access paths, security, concurrency control
    • Allowing simultaneous access to the database, while preserving integrity
  – Business-imposed data requirements and constraints
ER Model Verification

- **Processes** may be classified according to their **frequency** or operational type.
- Process verification may reveal **additional entity or attribute details**.
- Process verification is **repeated to all modules** of the ER model.

**TABLE 8.4 The ER Model Verification Process**

<table>
<thead>
<tr>
<th>STEP</th>
<th>ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify the ER model's central entity.</td>
</tr>
<tr>
<td>2</td>
<td>Identify each module and its components.</td>
</tr>
</tbody>
</table>
| 3    | Identify each module's transaction requirements:  
     Internal: Updates/Inserts/Deletes/Queries/Reports  
     External: Module interfaces |
| 4    | Verify all processes against the ER model. |
| 5    | Make all necessary changes suggested in Step 4. |
| 6    | Repeat Steps 2 through 5 for all modules. |
Iterative ER Model Verification

**Figure 8.10 Iterative ER Model Verification Process**

1. **Identify processes**
2. **Define transaction steps**
3. **Verify results**

- Change process
- Change ER model

ER model
Verification Process

• Select the **central entity** (most important)
  – Defined in terms of its participation in most of the model’s relationships
• Identify the module or **subsystem** to which the central entity belongs and define **boundaries** and **scope**
• Place central entity within the module’s framework
DBM Software Selection
DBMS Software Selection

• Critical to the information system’s smooth operation
• Advantages and disadvantages should be carefully studied
• End user must be made aware of limitations of both DBMS & DB to avoid false expectations
Factors Affecting DMBS Choice

• **Cost**
  – Purchase, maintenance, operational, license, installation, training, conversion, …

• **DBMS features and tools**
  – Query by example (QBE), screen painters, report generators, application generators, data dictionaries, …

• **Underlying model**
  – Hierarchical, network, relational, object, …

• **Portability**
  – Across platform, system, language, …

• **DBMS hardware requirements**
  – Processor, RAM, disk space, …
Logical Design
Logical Design

• Used to translate conceptual design into internal model for a selected database management system
• Is software-dependent
• Requires that all objects in the model be mapped to specific constructs used by selected database software
Logical Design

• Design components
  – Tables
  – Indexes
  – Views
  – Transactions
  – Access authorities
  – Others
Logical Design

- Logical design’s tables must correspond to the entities of the conceptual design
- Example: A Simple Conceptual Model

**FIGURE 8.11 A SIMPLE CONCEPTUAL MODEL**
Logical Design

- Example: Definition of Attributes & Domains of CLASS Entity
  - CLASS_CODE
    - Valid class code
    - Type: numeric
    - Range: minimum = 1000, maximum = 9999
    - Display format: 9999
    - Length: 4
  - CLASS_DAYS
    - Valid day code
    - Type: character
    - Display format: XXX
    - Length: 3
  - CLASS_TIME
    - Valid time
    - Type: character
    - Display format: 99:99 (24 hour clock)
    - Length: 5
Logical Design

• Example: Initial table layout for the COURSE table

<table>
<thead>
<tr>
<th>CRS_CODE</th>
<th>CRS_TITLE</th>
<th>CRS_DESCRIPIT</th>
<th>CRS_CREDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIS-4367</td>
<td>Database Systems Design</td>
<td>Design and implementation of database systems. Includes conceptual design, logical design, implementation, and management. Prerequisites: CIS 2040, CIS-2345, and CIS 3680 and upper division standing.</td>
<td>4</td>
</tr>
<tr>
<td>QM-3456</td>
<td>Statistics II</td>
<td>Statistical applications. Course requires use of statistical software (MINITAB and SAS) to interpret data. Prerequisites: MATH-2345 and QM-2233.</td>
<td>3</td>
</tr>
</tbody>
</table>
IV

Physical Design
Physical Design

• The process of selecting data storage & access characteristics of the DB
  – Very technical
  – Depends on
    • Types of devices supported by the hardware
    • Type of data access methods supported by the system
    • The DBMS
  – More important in older hierarchical and network models

• Affects system performance
  – Storage media characteristics
  – Use of index
  – Type of data, …

• Becomes more complex for distributed systems
• Designers favor software that hides physical details
  – Modern RDBMS took that burden from the DBA’s
DB Design Strategies

• Two approaches and two philosophies:
  – Approaches
    • Top-down design
    • Bottom-up design
  – Philosophies
    • Centralized
    • Decentralized
DB Design Strategies

• Approaches
  – Top-down design
    • Identifies data sets
    • Defines data elements for each of those sets
    • More easily managed
  – Bottom-up design
    • Identifies data elements (items)
    • Groups them together in data sets
    • More productive for small businesses
Top-Down vs. Bottom-Up Design

**Figure 8.14** Top-Down vs. Bottom-Up Design Sequencing
Centralized vs. Decentralized Design

• Database design may be based on two very different design philosophies:
  – Centralized design
    • Typical of simple databases
    • Conducted by single person or small team
    • Productive when the data component is composed of a relatively small number of objects and procedures
  – Decentralized design
    • Larger numbers of entities and complex relations
    • Spread across multiple sites
    • Developed by teams
    • Used when the data component of system has considerable number of entities and complex relations on which very complex operations are performed
Centralized Design

**Figure 8.15 Centralized Design**

- Conceptual model
- Conceptual model verification
  - User views
  - System processes
  - Data constraints
- Data dictionary
Decentralized Design

**Figure 8.16 Decentralized Design**

Diagram showing the relationship between conceptual models, views, processes, constraints, aggregation, and the final conceptual model leading to data dictionary.
Aggregation Process

• Requires designer to create a single model in which various aggregation problems must be addressed:
  – Synonyms and homonyms
    • **Synonym**: Same object by different names in different departments
    • **Homonym**: Same name for different objects
  – Entity and entity subtypes
    • Entity subtype might be viewed as a separate entity by one or more department.
    • The designer must integrate the subtypes into the higher-level entity
  – Conflicting object definitions
    • Attributes recorded as different types (character, numeric) or different domains defined for the same attribute
    • Constraint definitions can vary
    • Designer must remove such conflicts from the model
Summary of Aggregation Problems

**Synonyms:** two departments use the different names for the same entity.

**Homonyms:** two different entities are addressed by the same label. (Department B uses the label X to describe both entity X and entity Y.)

**Entity and entity subclass:** The entities X1 and X2 are subsets of entity X.

**Conflicting object definitions:** attributes for the entity PROFESSOR

- **Primary key:** PROF_SSN
- **Phone attribute:** 898-2853
- **Payroll Dept:** Systems Dept.
- **License:** 2853

**Example:**

- **Name**
- **Address**
- **Phone**

- **Common attributes**
- **Distinguishing attributes**
Summary

• Transformation from data to information is produced when programming code operates on the data, thus producing applications
• Information system is designed to facilitate transformation of data into information and to manage both data and information
• SDLC traces the history (life cycle) of an application within the information system
Summary (continued)

- DBLC describes the history of the database within the information system.
- Database design and implementation process moves through a series of well-defined stages.
- Conceptual portion of the design may be subject to several variations, based on two design philosophies.