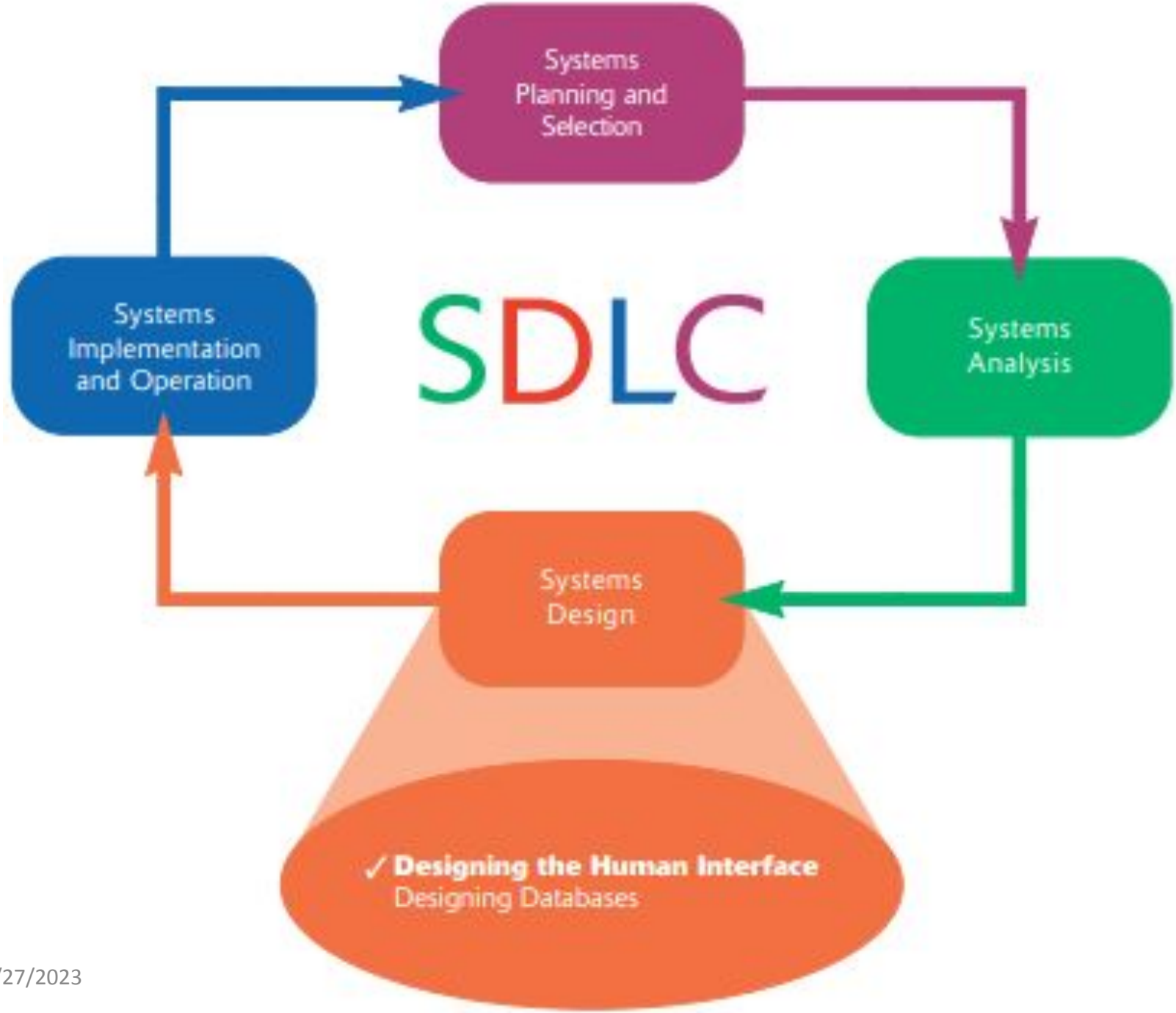
Systems Analysis And Design

Chapter Four: Systems Design


Systems Planning and Selection
Systems Analysis
Systems Design
Systems Implementation and Operation
SDLC
✓ Designing the Human Interface
Designing Databases

Designing Forms and Reports

- System inputs and outputs are produced at the end of the analysis phase
  - Precise appearance is not necessary
- Forms and reports are integrally related to **DFD and E-R diagrams**
  - every input form is associated with a data flow entering a process on a DFD
  - every output form or report is a data flow produced by a process on a DFD
  - data on all forms and reports must consist of data elements in data stores and on the E-R data model

Designing Forms and Reports
Key Concepts

- **Form**
  - A business document that contains some **predefined data and areas where additional data** are to be filled in
  - An instance of a form is typically based on **one database record**
- **Report**
  - A business document that contains only predefined data
  - A **passive document** for reading or viewing data
  - Typically **contains data organized/summarized** from many database records or transactions

Designing Forms and Reports (Cont.)

- Common Types of Reports:
  - *Scheduled:* produced at predefined time intervals (daily, weekly or monthly) for routine information needs of an organization.
  - *Key-indicator*: provide summary of critical information on regular basis.

Designing Forms and Reports (Cont.)

- *Exception*: highlights data outside of normal operating ranges.
- *Drill-down*: provide details behind summary of key-indicator or exception reports.
- *Ad-hoc*: respond to unplanned requests for non-routine information needs.

The Process of Designing Forms and Reports

- Is a User Focused Activity
- Follows a Prototyping Approach
- Requirements Determination:
  - Who will use the form or report?
  - What is the purpose of the form or report?
  - When is the report needed or used?(**periodic**)
  - Where does the form or report need to be delivered and used?
  - How many people need to use or view the form or report?

The Process of Designing Forms and Reports

- Prototyping
  - Initial prototype is designed from requirements
  - Users review prototype design and either accept the design or request changes
  - If changes are requested, the **construction-evaluation-request cycle** is repeated until the design is accepted

Deliverables and Outcome

- **Design specifications** are major deliverables and contain three sections
  1. **Narrative overview**
  2. **Sample design**
  3. **Testing and usability assessment**

(A) Narrative overview

Form: Customer Account Status
Users: Customer account representatives within corporate offices
Task: Assess customer account information: address, account balance, year-to-date purchases and payments, credit limit, discount percentage, and account status.
System: Microsoft Windows
Environment: Standard office environment

(B) Sample design

(C) Testing and usability assessment

User-Rated Perceptions (average 14 users):

consistency [1 = consistent to 7 = inconsistent]:	1.52
sufficiency [1 = sufficient to 7 = insufficient]:	1.43
accuracy [1 = accurate to 7 = inaccurate]:	1.67

Formatting Forms and Reports

General guidelines for the design of forms and reports

- *Meaningful titles*: clear, specific, version information, current date, valid date.
- *Meaningful information*: include only necessary information, with no need to modify.

Formatting Forms and Reports (Cont.)

- *Balanced layout*: adequate spacing, margins, and clear labels.
- *Easy navigation system*: show how to move forward and backward, and where you are currently.

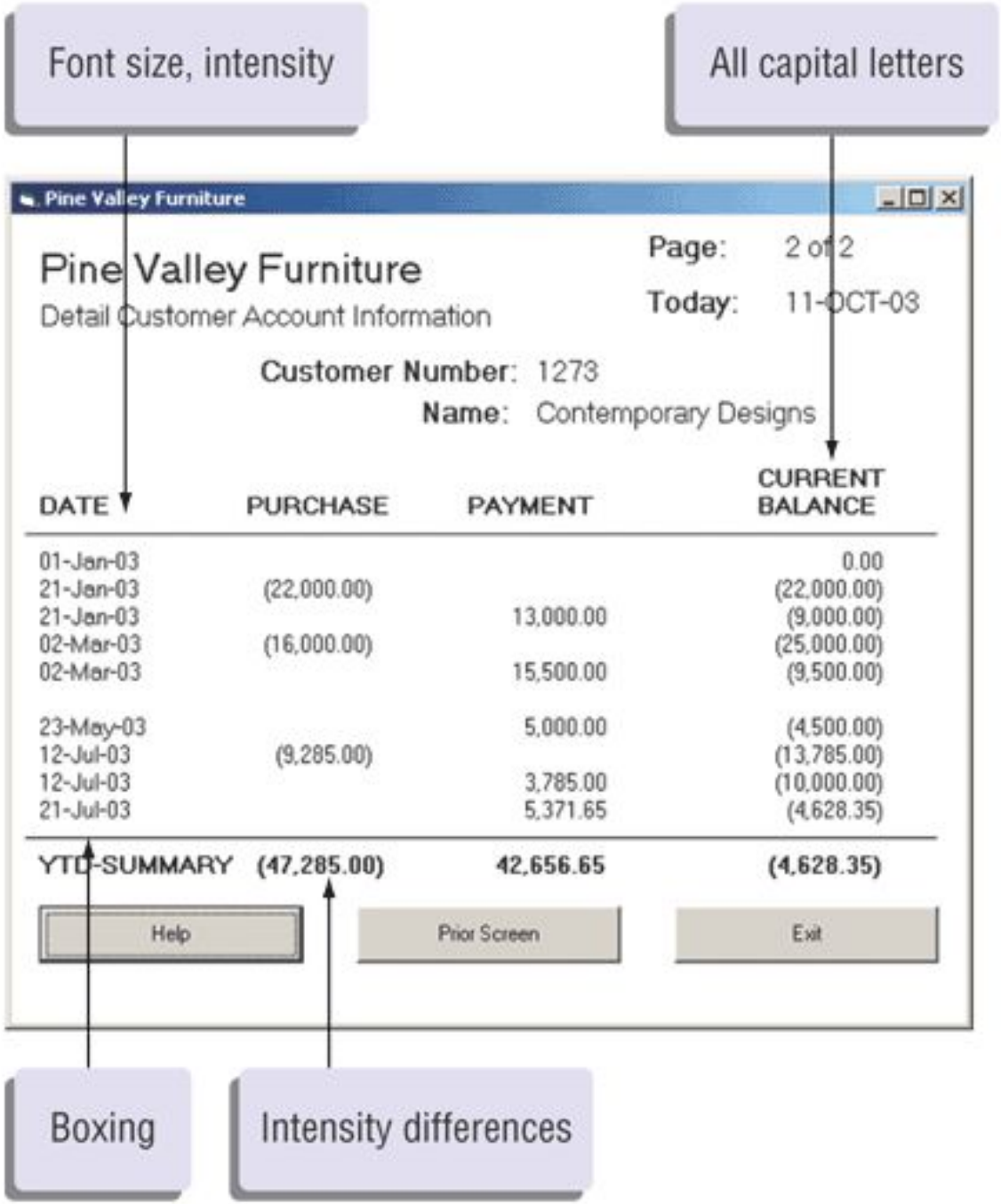
Page: 371 and 372

General Formatting Guidelines for Forms and Reports

- **Highlighting Information**
  - Use sparingly to draw user to or away from certain information
  - **Blinking and audible tones** should only be used to highlight critical information requiring user's immediate attention
  - Highlighting Methods should be consistently selected and used based upon level of importance of emphasized information

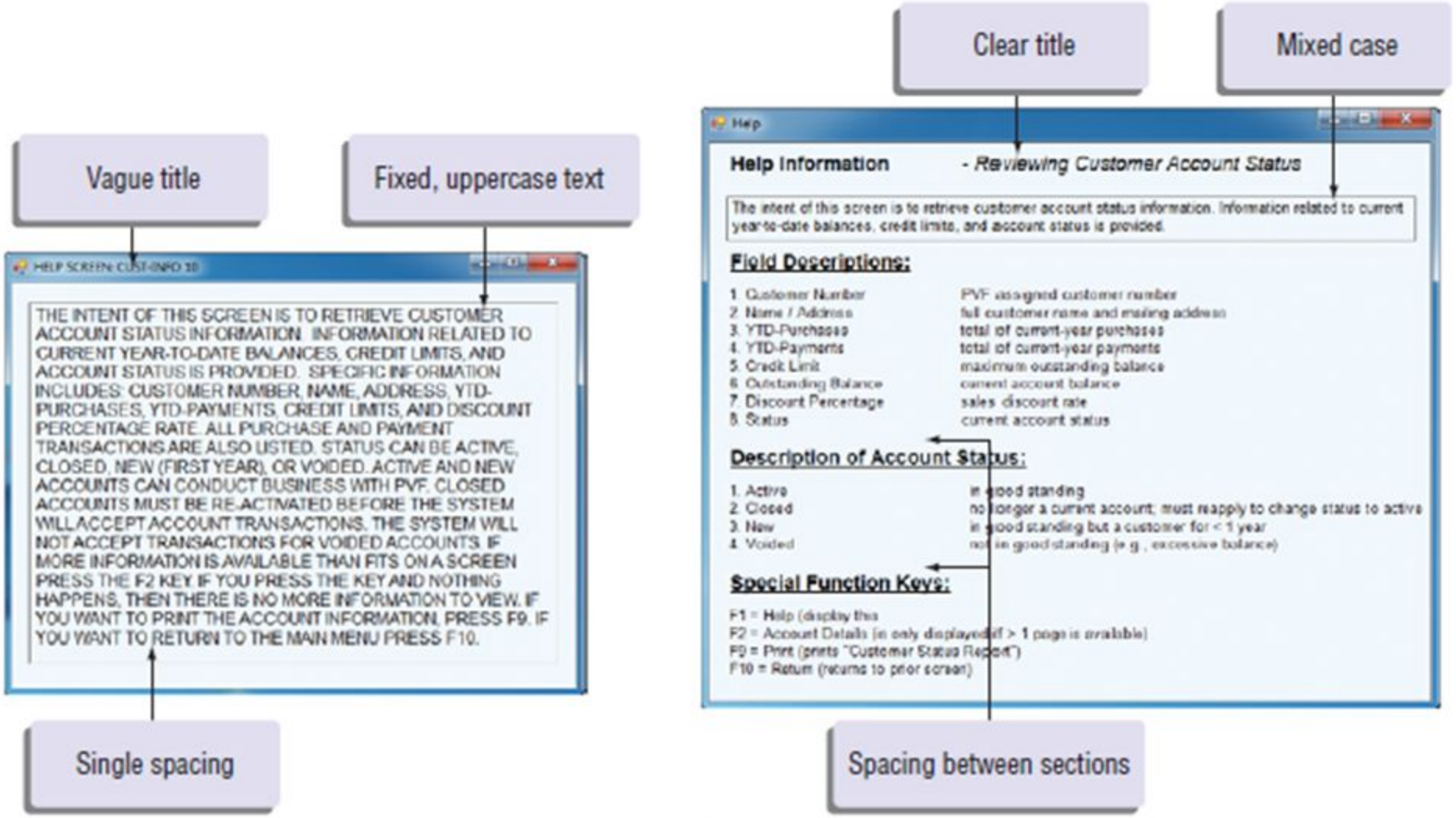
Methods of Highlighting

- Blinking and audible tones
- Color differences
- Intensity differences
- Size differences
- Font differences
- Boxing
- Underlining
- All capital letters


Font size, intensity
All capital letters
Pine Valley Furniture
Pine Valley Furniture
Detail Customer Account Information
Page: 2 of 2
Today: 11-OCT-03
Customer Number: 1273
Name: Contemporary Designs
DATE
PURCHASE
PAYMENT
CURRENT BALANCE
01-Jan-03 0.00
21-Jan-03 (22,000.00) (22,000.00)
21-Jan-03 13,000.00 (9,000.00)
02-Mar-03 (16,000.00) (25,000.00)
02-Mar-03 15,500.00 (9,500.00)
23-May-03 5,000.00 (4,500.00)
12-Jul-03 (9,285.00) (13,785.00)
12-Jul-03 3,785.00 (10,000.00)
21-Jul-03 5,371.65 (4,628.35)
YTD-SUMMARY (47,285.00) 42,656.65 (4,628.35)
Help
Prior Screen
Exit
Boxing
Intensity differences

General Formatting Guidelines for Forms and Reports

- **Displaying Text**
  - Display text in mixed upper and lower case and use conventional punctuation
  - Use double spacing if space permits. If not, place a blank line between paragraphs
  - Left-justify text and leave a ragged right margin
  - Do not hyphenate words between lines
  - Use abbreviations and acronyms only when they are widely understood by users and are **significantly shorter than the full text**



(A) A poorly designed help screen with many violations of the general guidelines for displaying text, (B) An improved design for a help screen.

General Formatting Guidelines for Forms and Reports

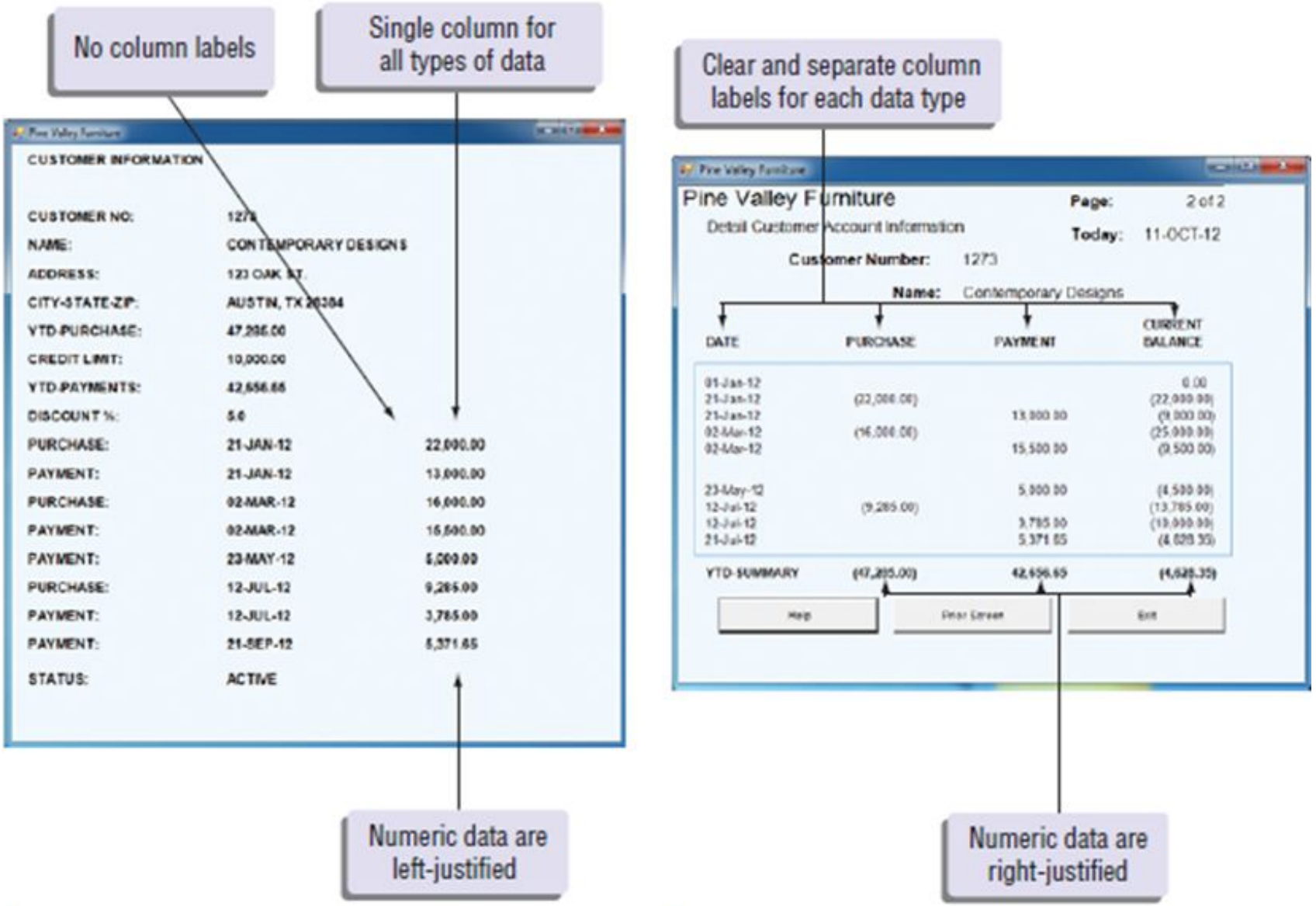
- **Displaying Tables and Lists**
  - **Labels**
    - All **columns and rows should have meaningful labels**
    - Labels should be separated from other information by using **highlighting**
    - Redisplay labels when the data extend beyond a single screen or page

General Formatting Guidelines for Forms and Reports

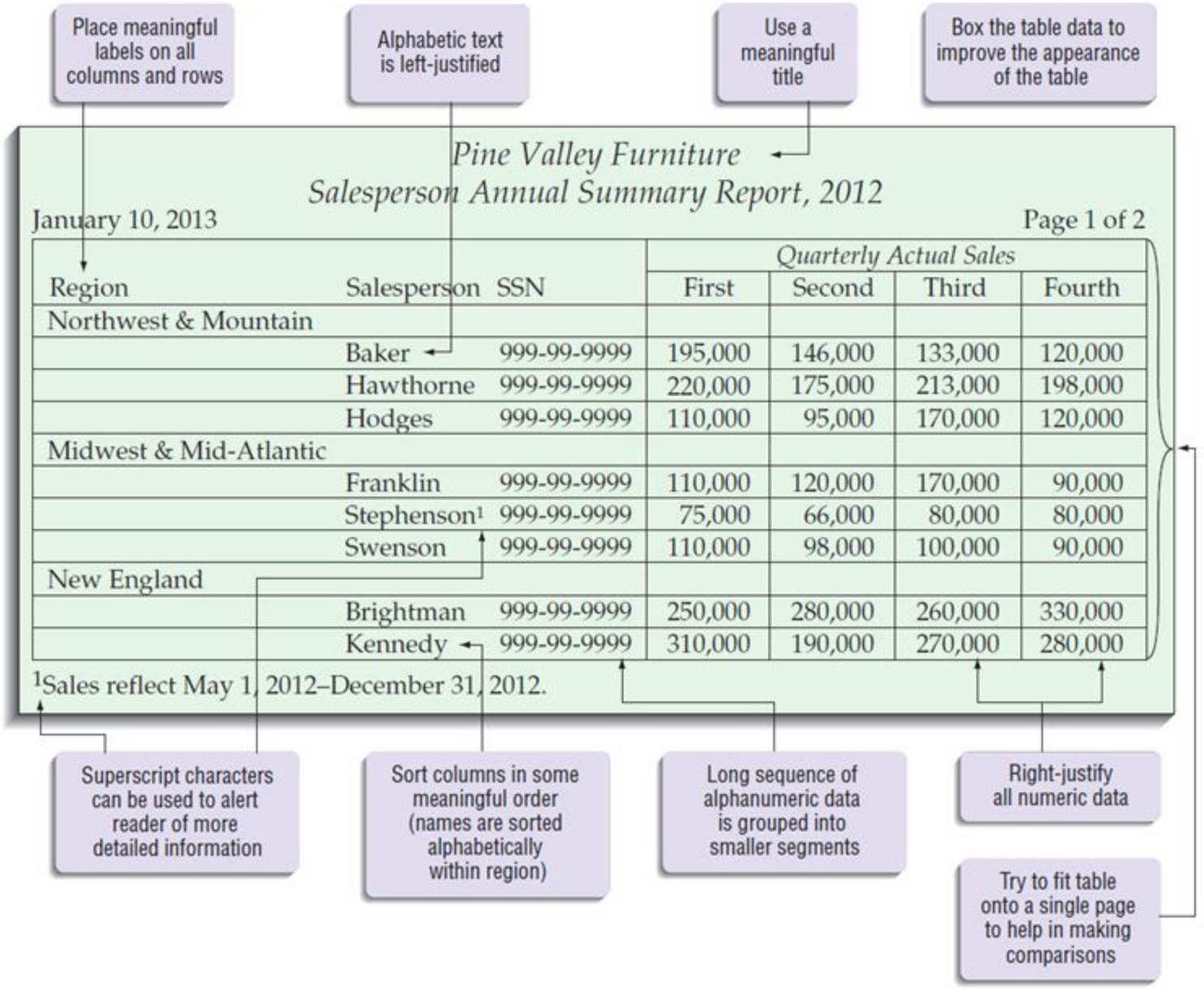
- **Displaying Tables and Lists (continued)**
  - **Formatting columns, rows, and text**
    - Sort in a meaningful order (intuitive design)
    - Place a blank line between every 5 rows in long columns
    - Columns should have at least two spaces between them
    - Allow white space on printed reports for user to write notes
    - Use a single typeface, except for emphasis
    - Use same family of typefaces within and across displays and reports
    - Avoid overly fancy fonts

General Formatting Guidelines for Forms and Reports (continued)

- **Displaying tables and lists (continued)**
  - **Formatting numeric, textual, and alphanumeric data**
    - **Right-justify *numeric* data** and align columns by decimal points or other delimiter
    - **Left-justify *textual* data**. Use short line length, usually 30 to 40 characters per line
    - Break long sequences of ***alphanumeric*** data into small **groups of three to four characters** each



(A) A poorly designed form, (B) An improved design form.



Tabular report illustrating good report design guidelines.

Assessing Usability

- Objective for designing forms, reports and all human-computer interactions is usability.
- There are three characteristics:
  - *Speed*. Can you complete a task efficiently?
  - *Accuracy*. Does the output provide what you expect?
  - *Satisfaction*. Do you like using the output?

Assessing Usability (Cont.)

- **Usability**: an overall *evaluation of how a system **performs** in supporting a particular user for a particular task*.

Usability Success Factors

- **Consistency**: of terminology, abbreviations, formatting, titles, navigation, response time.
- **Efficiency**: minimize required user actions.
- **Ease**: self-explanatory outputs and labels.
- **Format**: appropriate display of data and symbols.
- **Flexibility**: maximize user options for data input according to preference.

Usability Success Factors (Cont.)

- **Characteristics for consideration**:
  - **User**: experience, skills, motivation, education, personality.
  - **Task**: time pressure, cost of errors, work durations.
  - **System**: platform.
  - **Environment**: social and physical issues, e.g, lighting, sound, temperature, humidity, task interruptions.

Measures of Usability

- **Time to learn.**
- **Speed of performance.**
- **Rate of errors.**
- **Retention over time.**
- **Subjective satisfaction**.

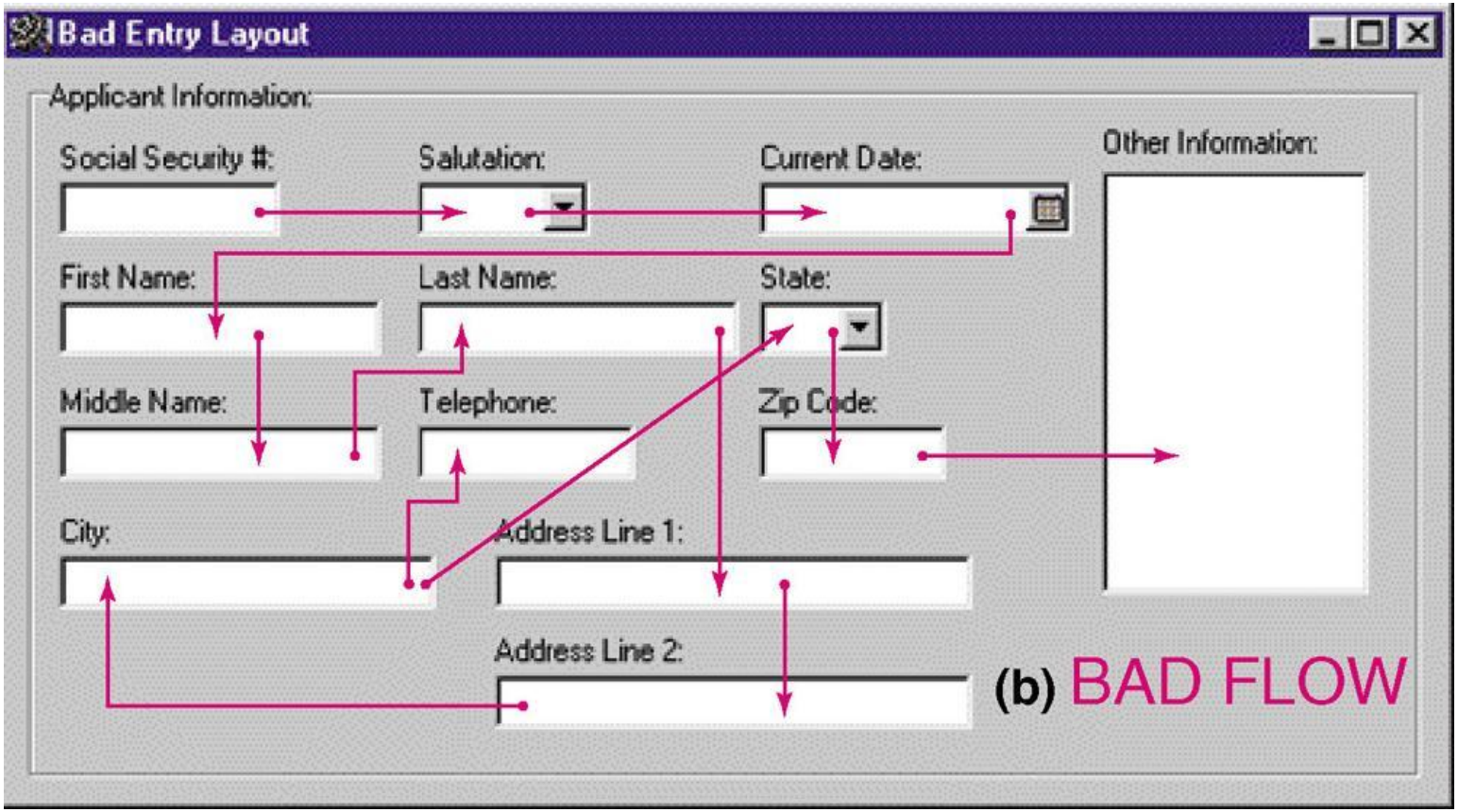
Designing Interfaces and Dialogues

- Focus on how information is provided to and captured from users
- Dialogues are analogous to a **conversation between two people**
- A good human-computer interface provides a unifying structure for **finding, viewing, and invoking the different components** of a system

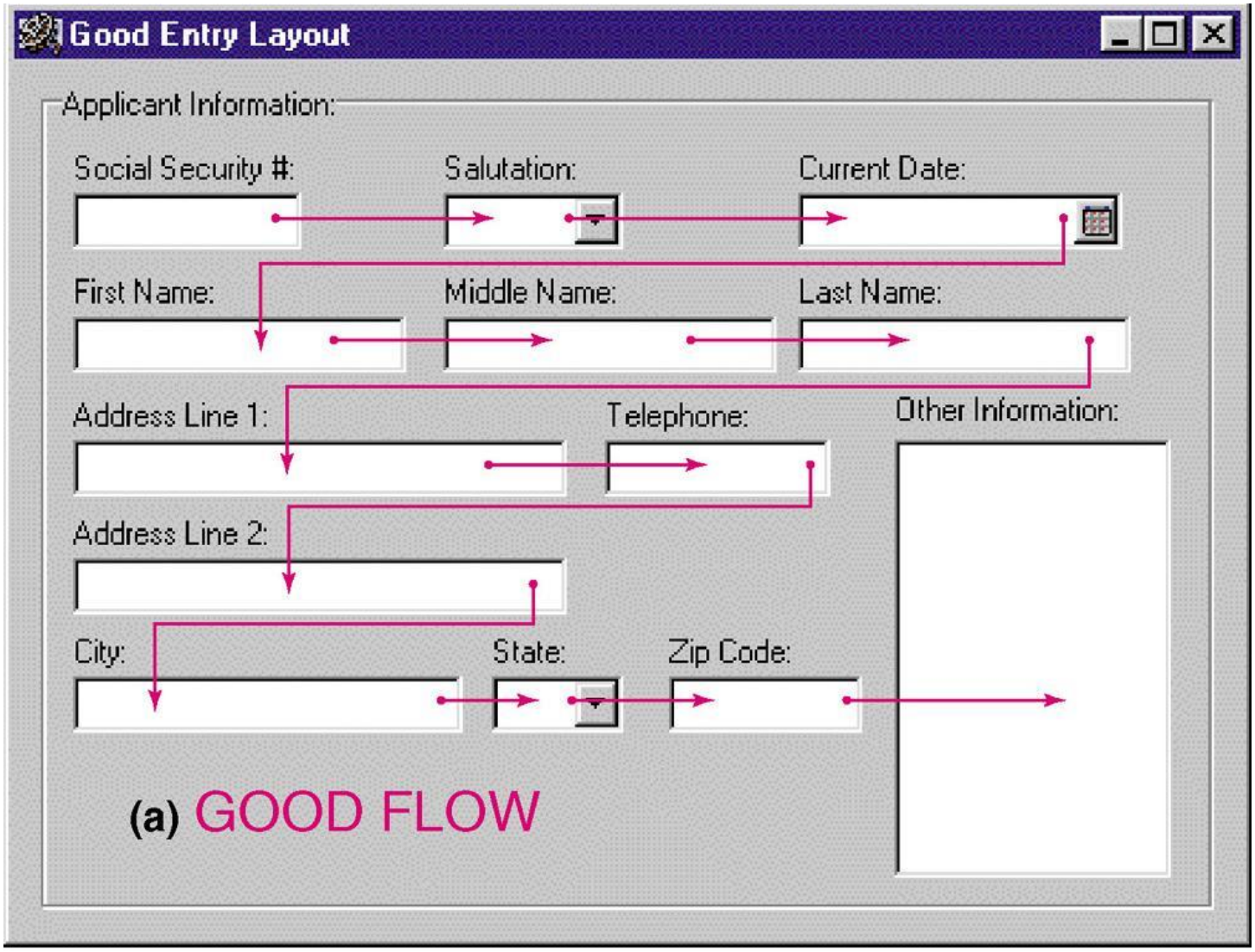
The Process of Designing Interfaces and Dialogues

- Is a user-focused activity
- Parallels **Form and Report Design Process**
- Employs Prototyping Methodology
  - Collect information
  - Construct prototype
  - Assess usability
  - Make refinements
- Deliverables - **Design Specifications**
  - **Narrative overview**
  - **Sample design**
  - **Testing and usability assessment**

Bad Screen Navigation



Good Screen Navigation



Designing Layouts

- Flexibility and consistency are primary design goals
  - Users should be able to move freely between fields
  - Data should not be permanently saved until the user explicitly requests to do so
  - Each key and command should be assigned to one function, and this function should be consistent throughout the entire system and across systems, if possible.

Structuring Data Entry

Guidelines for structuring data entry fields

Entry	Never require data that are already online or that can be computed
Defaults	Always provide default values when appropriate
Units	Make clear the type of data units requested for entry
Replacement	Use character replacement when appropriate
Captioning	Always place a caption adjacent to fields
Format	Provide formatting examples
Justify	Automatically(Left/Right) justify data entries
Help	Provide context-sensitive help when appropriate

Controlling Data Input

- One objective of interface design is to reduce data-entry errors
- As data is entered into an information system, steps must be taken to ensure that the input is valid
- Systems analysts are expected to anticipate the types of errors users may make and design features into the system's interfaces to avoid, detect, and correct data-entry mistakes
- See the guideline for detecting data errors in the next slide

p. 408 (validation tests and techniques to enhance the validity of data input)

Validation Test	Description
Class or composition	Test to ensure that data are of proper type (e.g., all numeric, all alphabetic, alphanumeric)
Combinations	Test to see that value combinations of two or more data fields are appropriate or make sense (e.g., does the quantity sold make sense given the type of product?)
Expected values	Test to see whether data are what is expected (e.g., match with existing customer names, payment amount, etc.)
Missing data	Test for existence of data items in all fields of a record (e.g., is there a quantity field on each line item of a customer order?)
Pictures/templates	Test to ensure that data conform to a standard format (e.g., are hyphens in the right places for a student ID number?)
Range	Test to ensure data are within a proper range of values (e.g., is a student's grade-point average between 0 and 4.0?)
Reasonableness	Test to ensure data are reasonable for situation (e.g., pay rate for a specific type of employee)
Self-checking digits	Technique by which extra digits, derived using a standard formula are added to a numeric field before transmission and checked after transmission
Size	Test for too few or too many characters (e.g., is social security number exactly nine digits?)
Values	Test to make sure values come from a set of standard values (e.g., two-letter state codes)

Providing Feedback

1. Status Information
   - Keeps users informed of system state
   - Displaying status information if the operation takes longer than a second or two
2. Prompting Cues
   - When prompting the user for information or action
   - Keep as specific as possible

   E.g. Enter the customer account number (123–456–7): _-_-_
3. **Error and Warning Messages**
   - Messages should be specific and free of error codes and jargon
   - User should be guided toward a result rather than scolded
   - Use terms familiar to user
   - Be consistent in format and placement of messages

   E.g. “No customer record found for that Customer ID. Please verify that digits were not transposed.”

Providing Help

- Put yourself in user's place when designing help
- Guidelines
  - **Simplicity** - Messages should be short and to the point
  - **Organization** – Use **lists** to break information into manageable pieces.
  - **Show examples** - Provide examples of proper use and the outcomes of such use.
- Context-Sensitive Help
  - Enables user to get field-specific help
- After leaving a help screen, users should always be returned to where they were when requesting help

Designing Dialogues

- The process of designing the overall sequences that users follow to interact with an information system is called **dialogue design**
- A dialogue is the sequence in which information is displayed to and obtained from a user

Dialogue Design Cont'd…

- The primary design guideline for designing dialogues is consistency; dialogues need to be consistent in sequence of actions, keystrokes, and terminology
- In other words, use the same labels for the same operations on all screens and the same location for the same information on all displays.

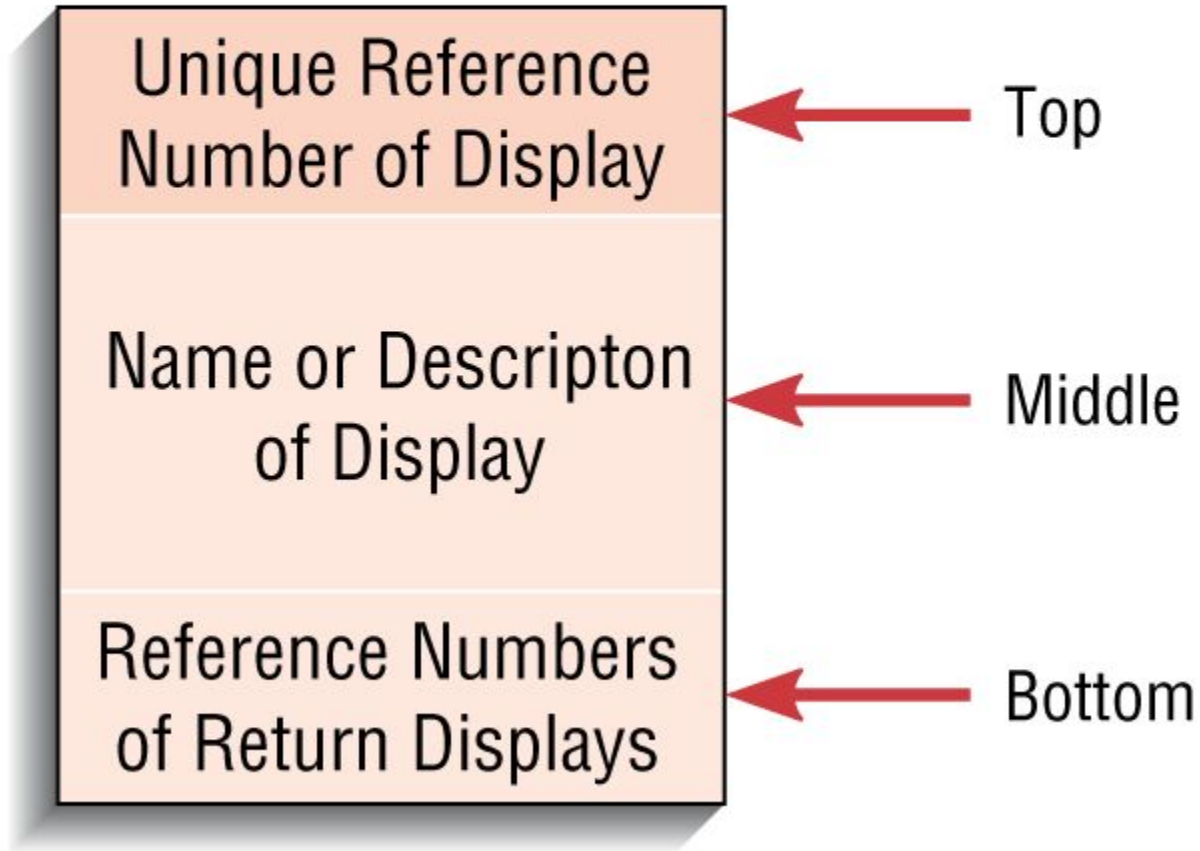
Designing the Dialogue Sequence

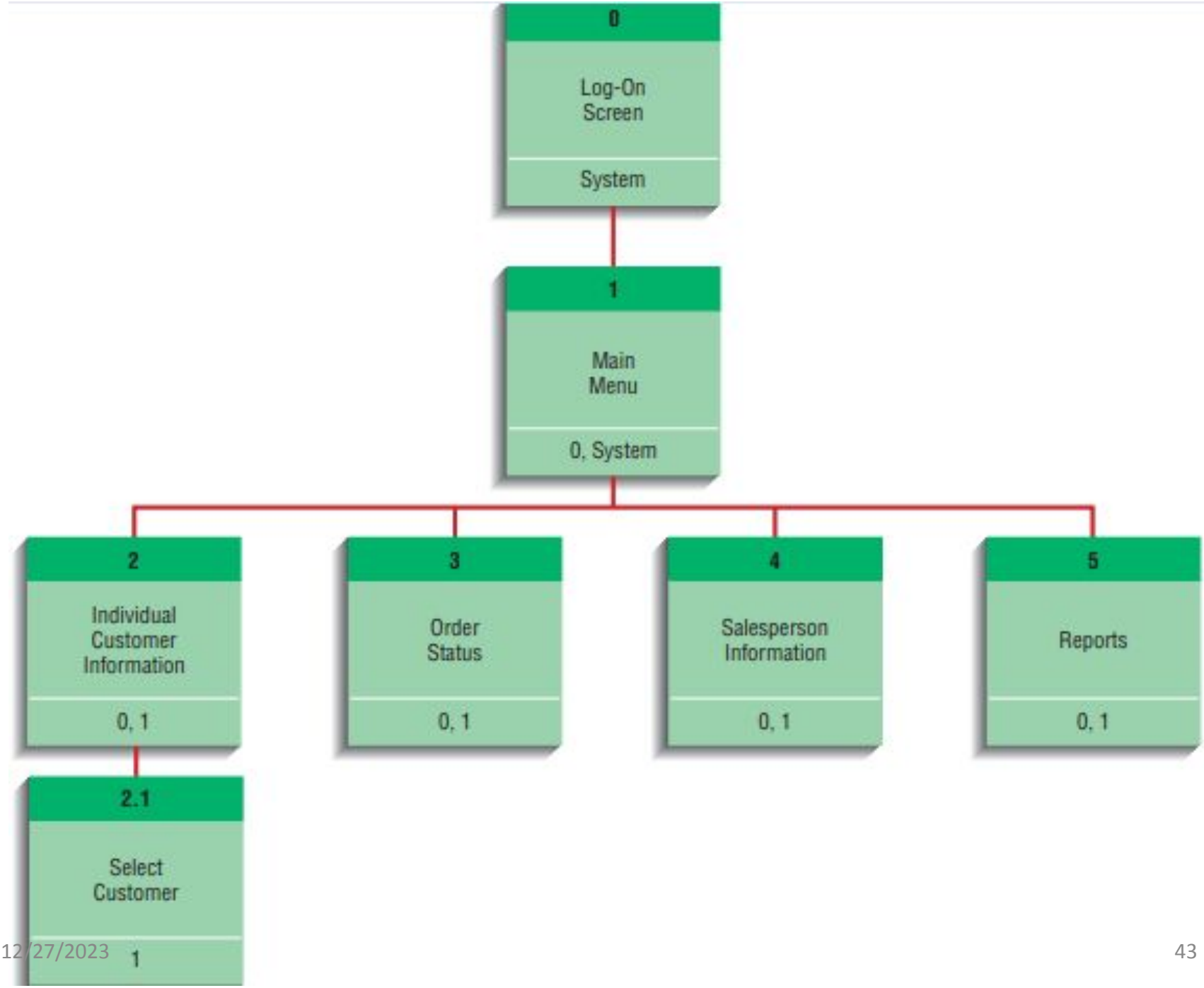
- Your first step in dialogue design is to define the **sequence**.
- A method for designing and representing dialogues is **dialogue diagramming**.
- **Dialogue diagrams** have only one symbol, a box with three sections; each box represents one display (which might be a full screen or a specific form or window) within a dialogue

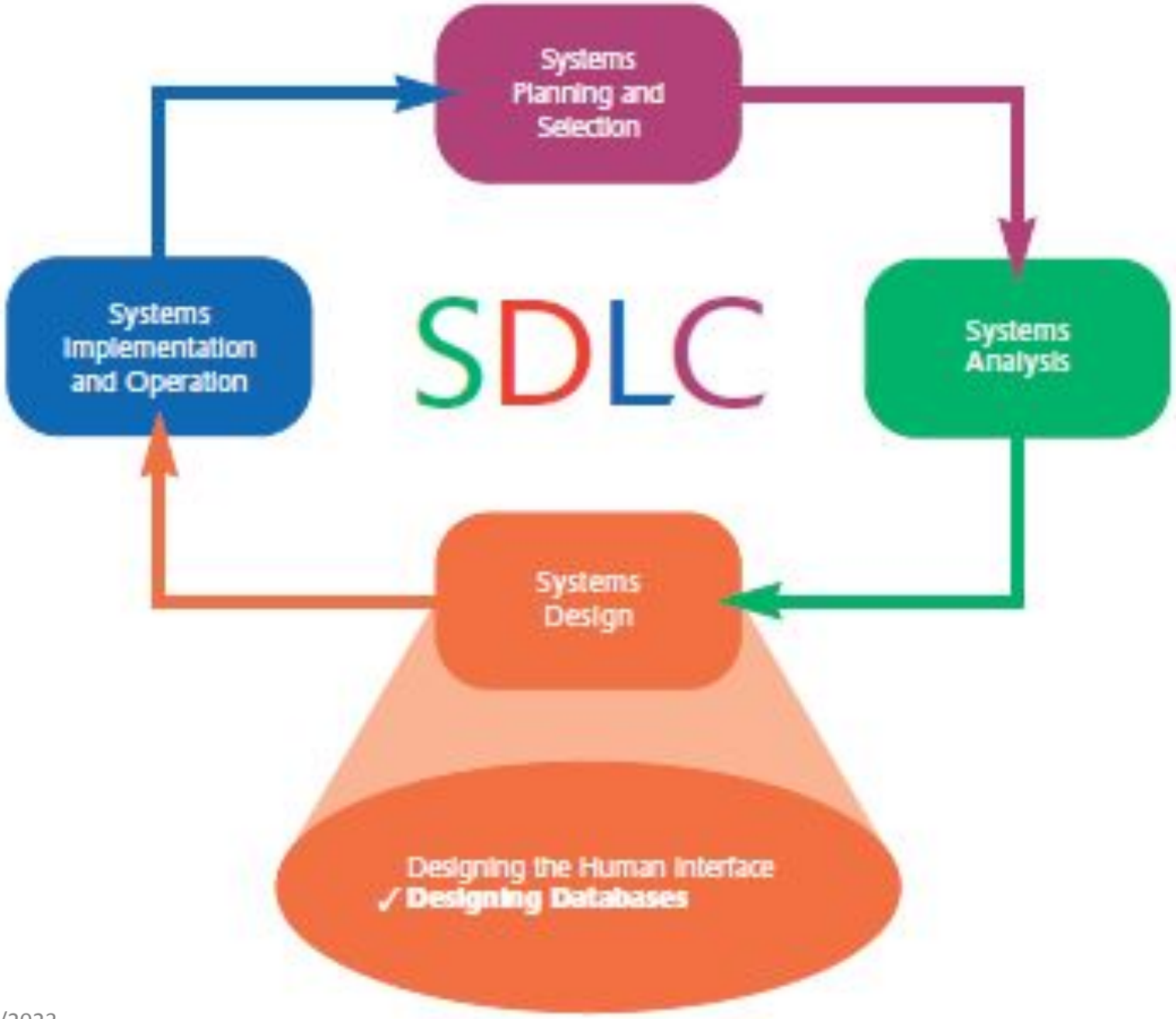
Dialogue Designs

- The three sections of the box are used as follows
  1. **Top**: Contains a unique display reference number that will be used by other displays for referencing it
  2. **Middle**: Contains the name or description of the display
  3. **Bottom**: Contains display reference numbers that can be accessed from the current display

Dialogue Design (Sample)




0
Log-On Screen
System
1
Main Menu
0, System
2
Individual Customer Information
0, 1
3
Order Status
0, 1
4
Salesperson Information
0, 1
5
Reports
0, 1
2.1
Select Customer
1


Systems Planning and Selection
Systems Analysis
Systems Design
Systems Implementation and Operation
SDLC
Designing the Human Interface
✓ Designing Databases

Deliverables and Outcomes

- **Logical database** is the representation of data using a model like relational database model, which represents data in simple tables with common columns to link related tables.
- **Physical database** is the representation of relations into files

Process of Database Design

- Logical Design
  - Four key steps:
    1. Develop a logical data model for each known user interface for the application using **normalization** principles
    2. Combine normalized data requirements from all user interfaces into one consolidated logical database model
    3. Translate the conceptual E-R data model for the application into normalized data requirements
    4. Compare the consolidated logical database design with the translated E-R model and produce one final logical database model for the application

Process of Database Design

- Physical Design
  - ***Based upon results of logical database design***
  - Key decisions:
    1. Choosing storage format for each attribute from the logical database model
    2. Grouping attributes from the logical database model into physical records
    3. Arranging related records in secondary memory (hard disks and magnetic tapes) so that records can be stored, retrieved, and updated rapidly
    4. Selecting media and structures for storing data to make access more efficient

Relational Database Model

- Data represented as a set of related tables or relations
- Relation
  - A named, two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows
  - Properties
    - Entries in cells are simple
    - Entries in columns are from the same set of values
    - Each row is unique
    - The sequence of columns can be interchanged without changing the meaning or use of the relation
    - The rows may be interchanged or stored in any sequence

Relational Database Model

- Well-Structured Relation
  - A relation that contains a minimum amount of redundancy and allows users to ***insert, modify, and delete the rows without errors or inconsistencies***

EMPLOYEE1

Emp_ID	Name	Dept	Salary
100	Margaret Simpson	Marketing	42,000
140	Allen Beeton	Accounting	39,000
110	Chris Lucero	Info Systems	41,500
190	Lorenzo Davis	Finance	38,000
150	Susan Martin	Marketing	38,500

Normalization

- The process of converting complex data structures into simple, stable data structures
- Eliminates redundancy

EMPLOYEE2

Emp_ID	Name	Dept	Salary	Course	Date_Completed
100	Margaret Simpson	Marketing	42,000	SPSS	6/19/2012
100	Margaret Simpson	Marketing	42,000	Surveys	10/7/2012
140	Alan Beeton	Accounting	39,000	Tax Acc	12/8/2012
110	Chris Lucero	Info Systems	41,500	SPSS	1/12/2012
110	Chris Lucero	Info Systems	41,500	C++	4/22/2012
190	Lorenzo Davis	Finance	38,000	Investments	5/7/2012
150	Susan Martin	Marketing	38,500	SPSS	6/19/2012
150	Susan Martin	Marketing	38,500	TQM	8/12/2012

Normalization

- Second Normal Form (2NF)
  - Each non-primary key attribute is identified by the whole key (called *full functional dependency*) i.e *No Partial dependency*
- Third Normal Form (3NF)
  - Non-primary key attributes do not depend on each other (called *transitive dependencies*) i.e *No transitive dependency*
- The result of normalization is that every non-primary key attribute depends upon the **whole primary key (No partial dependency)** and **nothing but the primary key (No transitive dependency)**.

Functional Dependencies and Primary Keys

- Functional Dependency
  - A particular relationship between two attributes. For a given relation, attribute B is functionally dependent on attribute A if, for every valid value of A, that value of A uniquely determines the value of B.
  - Instances (or sample data) in a relation do not prove the existence of a functional dependency
  - Knowledge of problem domain is the most reliable method for identifying functional dependency

Functional Dependencies and Primary Keys

- Second Normal Form (2NF)
  - A relation is in second normal form (2NF) if any of the following conditions apply:
    1. The primary key consists of only one attribute
    2. No non-primary key attribute exists in the relation
    3. Every non-primary key attribute is functionally dependent on the full set of primary key attributes

Functional Dependencies and Primary Keys

- Conversion to second normal form (2NF)
  - To convert a relation into 2NF, decompose the relation into new relations using the attributes, called determinates that determine other attributes
  - The determinants become the primary key of the new relation

Functional Dependencies and Primary Keys

- EMPLOYEE2 below is an example of a relation that is not in second normal form.
  - **EMPLOYEE2**(Emp_ID, Name, Dept, Salary, Course, Date_Completed)
- The functional dependencies in this relation are the following:
  - Emp_ID:Name, Dept, Salary
  - Emp_ID, Course:Date_Completed

Functional Dependencies and Primary Keys

- EMPLOYEE2 is decomposed into the following two relations:
  - **EMPLOYEE1**(Emp_ID, Name, Dept, Salary)
  - **EMP_COURSE**(Emp_ID, Course, Date_Completed)

Functional Dependencies and Primary Keys

- Third Normal Form (3NF)
  - A relation is in third normal form (3NF) if it is in second normal form (2NF) and there are no functional (transitive) dependencies between two (or more) non-primary key attributes
  - To convert a relation into 3NF, decompose the relation into two relations using the two determinants

Functional Dependencies and Primary Keys

- Consider the relation below:
  - SALES(Customer_ID, Customer_Name, Salesperson, Region)
- The following functional dependencies exist in the SALES relation where Customer_ID is the primary key :
  - Customer_ID:Customer_Name, Salesperson(non-key attribute), Region
  - Salesperson:Region (Each salesperson is assigned to a unique region.)

Functional Dependencies and Primary Keys

- SALES relation is decomposed into two relations to convert it into 3NF as follows:
  - SALES1(Customer_ID, Customer_Name, Salesperson)
  - SPERSON(Salesperson, Region)

Removing transitive dependencies: (A) Relation with transitive dependency, (B) Relations in 3NF.

SALES

Customer_ID	Customer_Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North

A

SALES1

Customer_ID	Customer_Name	Salesperson
8023	Anderson	Smith
9167	Bancroft	Hicks
7924	Hobbs	Smith
6837	Tucker	Hernandez
8596	Eckersley	Hicks
7018	Arnold	Faulb

SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hernandez	East
Faulb	North

B

Functional Dependencies and Primary Keys

- Foreign Key
  - An attribute that appears as a non-primary key attribute in one relation and as a primary key attribute (or part of a primary key) in another relation
- Referential Integrity
  - An integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation
  - Is implemented by using a foreign key

Transforming E-R Diagrams into Relations

- It is useful to transform the conceptual data model into a set of normalized relations
- Steps:
  - Represent entities
  - Represent relationships
  - Normalize the relations
  - Merge the relations

Transforming E-R Diagrams into Relations

1. Represent Entities
   - Each regular entity is transformed into a relation
   - The **identifier** of the entity type becomes the primary key of the corresponding relation
   - The primary key must satisfy the following two conditions
     a. The value of the key must uniquely identify every row in the relation
     b. The key should be **non-redundant** and **Non-Nullable**

A

Transforming an entity type to a relation: (A) E-R diagram, (B) Relation.

B

CUSTOMER

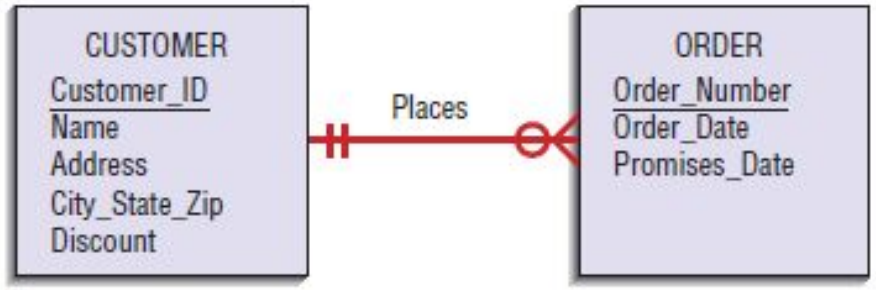
Customer_ID	Name	Address	City_State_Zip	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

Transforming E-R Diagrams into Relations

2. Represent Relationships

- ***Binary 1:*N *and 1:1 Relationships***
  - A binary one-to-many (1:N) relationship in an E-R diagram is represented by adding the primary key attribute (or attributes) of the entity **on the one side of the relationship as a foreign key in the relation that is on the many side** of the relationship.
  - For a binary or unary one-to-one (1:1) relationship between the two entities A and B (for a unary relationship, A and B would be the same entity type), the relationship can be represented by any of the following choices:
    1. Adding the primary key of A as a foreign key of B
    2. Adding the primary key of B as a foreign key of A
    3. Both of the above

Representing a 1:*N* relationship:
(A) E-R diagram, (B) Relations.



A

CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

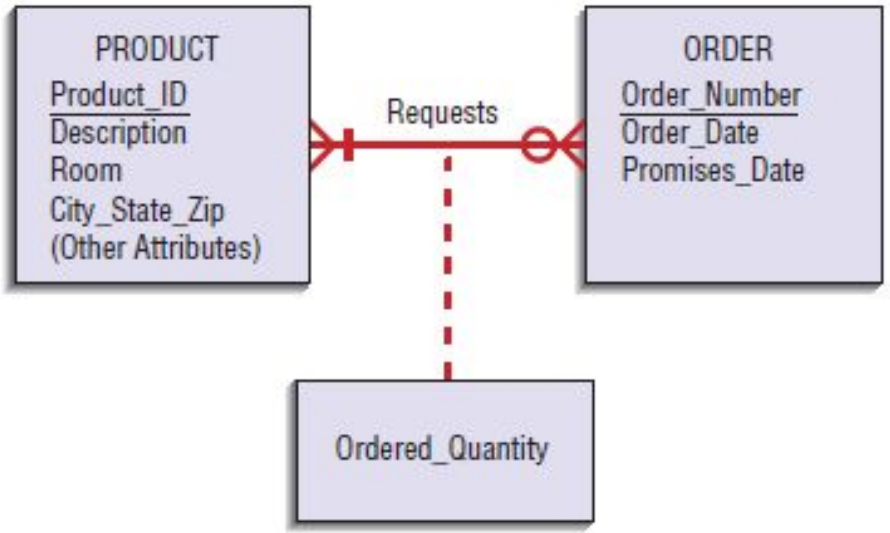
ORDER

Order_Number	Order_Date	Promised_Date	Customer_ID
57194	3/15/12	3/28/12	6390
63725	3/17/12	4/01/12	1273
80149	3/14/12	3/24/12	6390

B

Transforming E-R Diagrams into Relations

2. Represent Relationships (continued)
   - ***Binary and Higher-Degree* M:N *Relationships***
     - Suppose that a binary many-to-many (*M:N*) relationship (or **associative entity**) exists between two entity types A and B. **For such a relationship, we create a separate relation C**. The primary key of this relation is a composite key consisting of the primary key for each of the two entities in the relationship.



A

ORDER

Order_Number	Order_Date	Promised_Date
61384	2/17/2012	3/01/2012
62009	2/13/2012	2/27/2012
62807	2/15/2012	3/01/2012

ORDER LINE

Order_Number	Product_ID	Quantity_ Ordered
61384	M128	2
61384	A261	1

PRODUCT

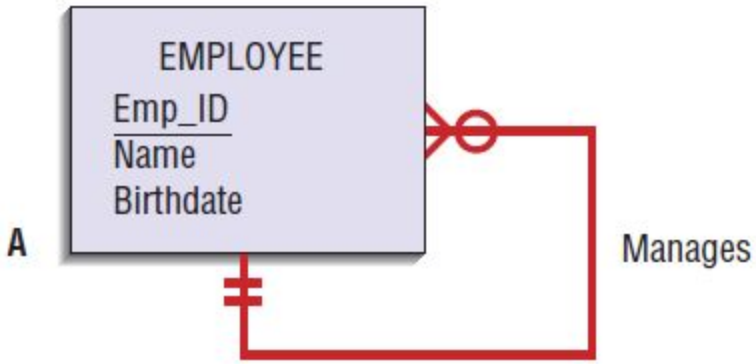
Product_ID	Description	(Other Attributes)
M128	Bookcase	—
A261	Wall unit	—
R149	Cabinet	—

B

Representing an *M:N* relationship: (A) E-R diagram, (B) Relations.

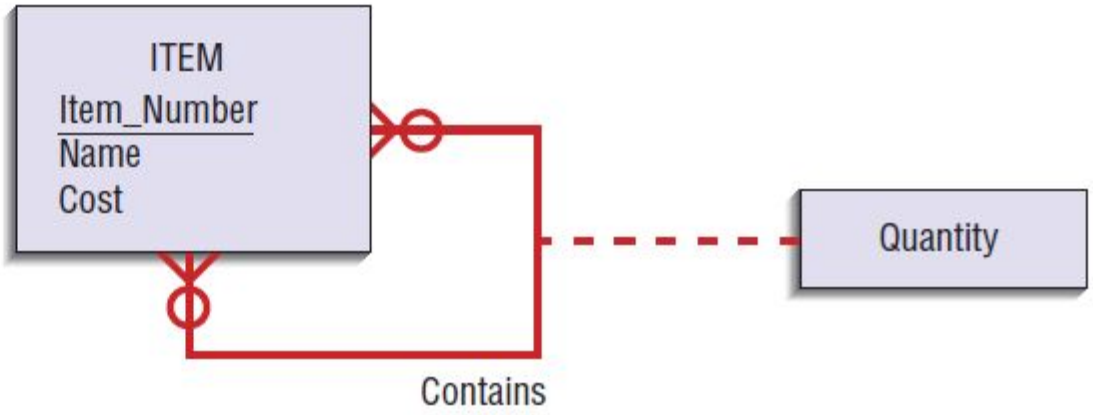
Transforming E-R Diagrams into Relations

- ***Unary 1:N Relationships***
  - Relationship between instances of a single entity type
  - Utilize a recursive foreign key
    - A foreign key in a relation that references the primary key values of that same relation
- ***Unary M:N Relationships***
  - Create a separate relation
  - Primary key of new relation is a composite of two attributes that both take their values from the same primary key(**but one of them renamed**)



A

EMPLOYEE(Emp_ID, Name, Birthdate, Manager_ID)



B

ITEM(Item_Number, Name, Cost)
ITEM-BILL(Item_Number, Component_Number, Quantity)

Table 9.1 E-R to Relational Transformation

E-R Structure	Relational Representation
Regular entity	Create a relation with primary key and nonkey attributes.
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which this weak entity depends) and nonkey attributes.
Binary or unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity or do this for both entities.
Binary 1:*N* relationship	Place the primary key of the entity on the one side of the relationship as a foreign key in the relation for the entity on the many side.
Binary or unary *M*:*N* relationship or associative entity	Create a relation with a composite primary key using the primary keys of the related entities, plus any non-key attributes of the relationship or associative entity.
Binary or unary *M*:*N* relationship or associative entity with additional key(s)	Create a relation with a composite primary key using the primary keys of the related entities and additional primary key attributes associated with the relationship or associative entity, plus any nonkey attributes of the relationship or associative entity.
Binary or unary *M*:*N* relationship or associative entity with its own key	Create a relation with the primary key associated with the relationship or associative entity, plus any nonkey attributes of the relationship or associative entity and the primary keys of the related entities (as nonkey attributes).

Transforming E-R Diagrams into Relations

3. Merging Relations (View Integration)
   - Purpose is ***to remove redundant relations***
   - View Integration Problems
     - Synonyms
       - Two different names used for the same attribute
       - When merging, get agreement from users on a single, standard name
     - Homonyms
       - A single attribute name that is used for two or more different attributes
       - Resolved by creating a new name
     - Dependencies between non-keys
       - Non-key Dependencies may be created as a result of view integration
       - In order to resolve, ***the new relation must be normalized***

Physical Database Design

- Physical Database Design is involved with how data are stored and related based on a particular DBMS.
- It involves two major activities:
  - Designing Fields
  - Designing Physical Tables

Designing Fields

- A **field** is the smallest unit of data recognized by database management systems
- In general, each attribute from each normalized relation is represented as one or more fields
- The basic decisions concerning a field are:
  - the type of data (or storage type) and
  - data integrity controls for the field.

Choosing Data Types

- A **data type** is a coding scheme for representing organizational data.
- The specific database management software to be used dictates which data type choices are available
- Selecting a data type balances four objectives that will vary in degree of importance for different applications:
  - Minimize **storage space**
  - Represent **all possible values** of the field
  - Improve **data integrity** for the field
  - Support **all data manipulations** desired on the field

Controlling Data Integrity

- The five popular data integrity control methods are:
  - **default value**: is the value a field will assume unless an explicit value is entered for the field
  - **input mask**: is a pattern of codes that restricts the width and possible values for each position within a field
  - **range control**: is a limited set of permissible values both numeric and alphabetic data may have
  - **referential integrity**: is constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation
  - **null value control**: is a constraint that insures that mandatory fields are given a value.

Designing Physical Tables

- A relational database is a set of related tables (tables are related by foreign keys referencing primary keys)
- a **physical table** is a **named set of rows and columns** that specifies the fields in each row of the table.
- A physical table may or may not correspond to one relation
- Whereas normalized relations possess properties of well-structured relations, the design of a physical table has two goals different from those of normalization:
  - efficient use of secondary storage and
  - data-processing speed.

Designing Physical Tables

- **De-normalization** is the process of **splitting or combining normalized relations into physical tables** based on affinity of use of rows and fields.
- By placing data used together closer to one another on disk, the number of disk I/O operations needed to retrieve all the data needed in a program is **minimized**.

Designing Physical Tables

Normalized Product Relation

PRODUCT(Product_ID,Description,Drawing_Number,Weight,Color,Unit_Cost,Burde

Denormalized Functional Area Product Relations for Tables

Engineering: E_PRODUCT(Product_ID,Description,Drawing_Number,Weight,Color)
Accounting: A_PRODUCT(Product_ID,Unit_Cost,Burden_Rate)
Marketing: M_PRODUCT(Product_ID,Description,Color,Price,Product_Manager)

A

Normalized relations:

VENDOR(Vendor_ID, Address, Contact_Name)

ITEM(Item_ID, Description)

PRICE QUOTE(Vendor_ID, Item_ID, Price)

Denormalized relations:

VENDOR(Vendor_ID, Address, Contact_Name)

ITEM-QUOTE(Vendor_ID, Item_ID, Description, Price)

B

File Organization

- **File Organization** is a technique for physically arranging the records of a file.
- The basic three families of file organizations used in most file management environments are:
  - **Sequential**
  - **indexed, and**
  - **hashed**

Sequential File Organization

- The rows in the file are stored in sequence according to a primary key value
- To locate a particular row, a program must normally scan the file from the beginning until the desired row is located
- Sequential files are fast if you want to process rows sequentially, but they are essentially impractical for random row retrievals
- Deleting rows can cause wasted space or the need to compress the file
- Adding or updating rows requires rewriting the file

Indexed File Organizations

- The rows are stored either sequentially or non-sequentially, and an index is created that allows software to locate individual rows.
- The main disadvantages of indexed file organizations are the extra space required to store the indexes and the extra time necessary to access and maintain indexes.

Hashed File Organizations

- In **hashed file organization,** the address of each row is determined using a hash function/algorithm that converts a primary key value into a row address
- rows are located non-sequentially as dictated by the hashing algorithm. Thus, sequential data processing is impractical.
- On the other hand, retrieval of random rows is fast

Comparative Features of Sequential, Indexed, and Hashed File Organizations

	File Organization		
Factor	**Sequential**	**Indexed**	**Hashed**
Storage space	No wasted space	No wasted space for data, but extra space for index	Extra space may be needed to allow for addition and deletion of records
Sequential retrieval on primary key	Very fast	Moderately fast	Impractical
Random retrieval on primary key	Impractical	Moderately fast	Very fast
Multiple key retrieval	Possible, but requires scanning whole file	Very fast with multiple indexes	Not possible
Deleting rows	Can create wasted space or require reorganizing	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy
Adding rows	Requires rewriting file	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy, except multiple keys with same address require extra work
Updating rows	Usually requires rewriting file	Easy, but requires maintenance of indexes	Very easy