Introduction

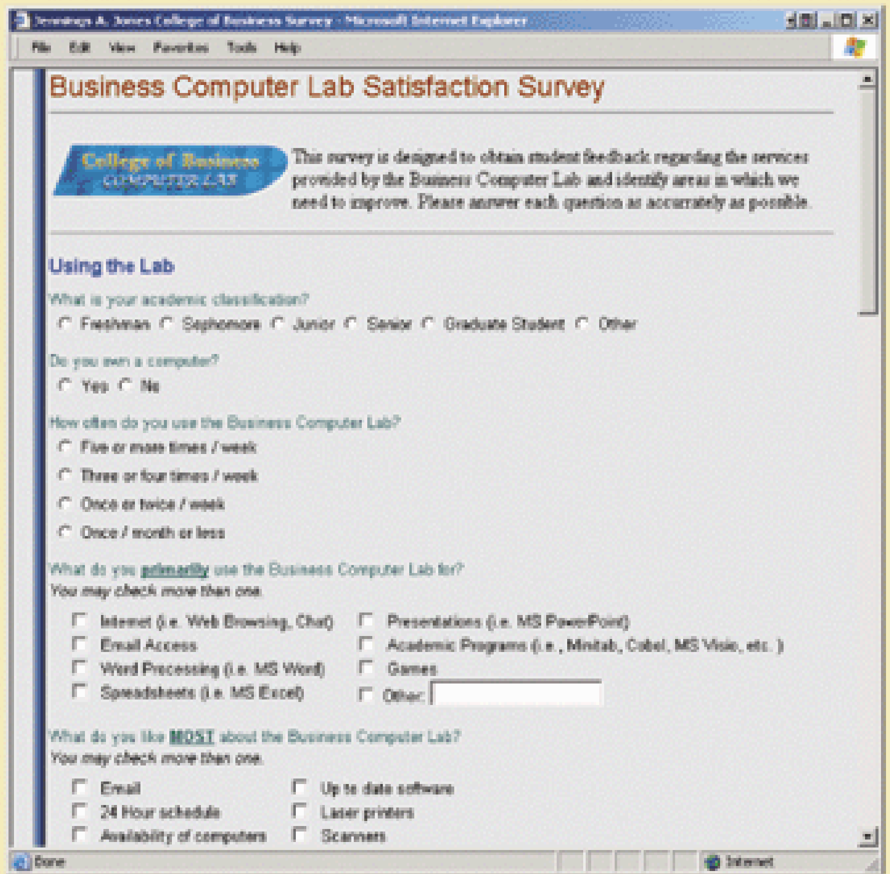
- Purpose of Database Systems
- View of Data
- Data Models
- Data Definition Language
- Data Manipulation Language

Data vs. Information

- Data:
  - Raw facts; building blocks of information
  - Unprocessed information
- Information:
  - Data processed to reveal meaning
- Accurate, relevant, and timely information is key for good decision making
- Good decision making is the key for survival in a global environment

Transforming Raw Data into Information

a) Initial Survey Screen



Transforming Raw Data into Information (continued)

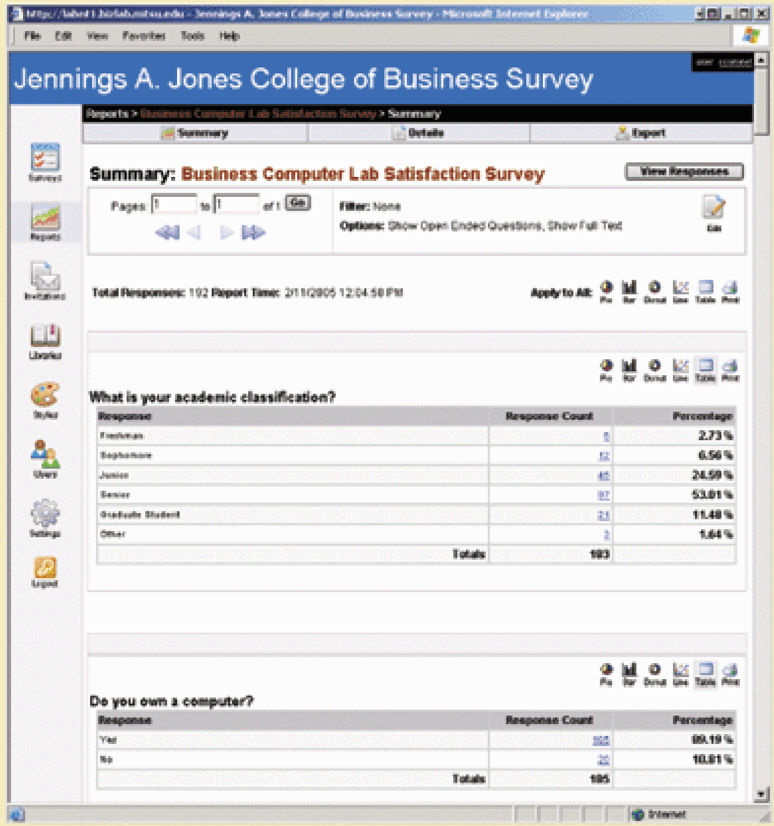
b) Raw Data

	A	B	C	D	E	F	G	H	I	J
1	AcadClas	OwnComputer	HowOftenUseLab	PrimarilyI	PrimarilyI	PrimarilyI	PrimarilyI	PrimarilyI	PrimarilyI	PrimarilyI
2	Gra	N	5W	0	1	1	1	1	1	1
3	Sen	Y	5W	1	0	0	0	1	1	1
4	Sen	Y	1W	0	0	0	1	0	0	0
5	Sen	Y	1W	1	0	0	0	1	0	0
6	Sen	Y	3W	0	0	0	0	1	0	0
7	Gra	N	5W	0	0	0	0	1	1	1
8	Sen	Y	1W	1	0	0	0	1	0	0
9	Sen	Y	3W	1	0	1	0	1	1	0
10	Sen	Y	1W	0	0	0	1	0	1	0
11	Sen	Y	5W	0	1	0	0	0	0	0
12	Jun	Y	1W	1	0	0	0	1	0	1
13	Sen	N	5W	1	0	0	1	1	1	1
14	Jun	Y	1W	0	0	0	1	0	0	0
15	Sen	Y	5W	0	0	0	1	0	0	0
16	Jun	Y	1M	0	1	0	0	0	0	1
17	Sen	Y	1W	0	0	1	0	0	0	0
18	Sen	Y	1W	0	0	1	1	0	0	0
19	Gra	N	5W	1	0	0	1	0	1	1
20	Gra	Y	1M	0	1	0	0	0	0	0
21	Gra	Y	5W	0	0	1	1	1	1	1
22	Sen	N	3W	0	0	1	1	1	1	0
23	Jun	Y	1W	1	0	0	0	0	0	0
24	Sen	Y	3W	1	1	0	1	1	1	0
25	Jun	Y	1W	1	0	0	0	1	0	0
26	Jun	Y	1W	1	0	0	0	0	1	1
27	Sen	Y	1M	1	0	0	0	0	1	0
28	Sen	Y	5W	0	0	0	0	1	0	1
29	Gra	Y	1M	0	0	0	1	0	0	1
30	Gra	Y	5W	1	1	0	0	1	1	1
31	Jun	N	5W	1	0	0	1	1	1	1
32	Jun	Y	3W	1	0	0	1	0	0	0
33	Gra	Y	5W	0	1	0	1	0	1	1

SurveyRawData

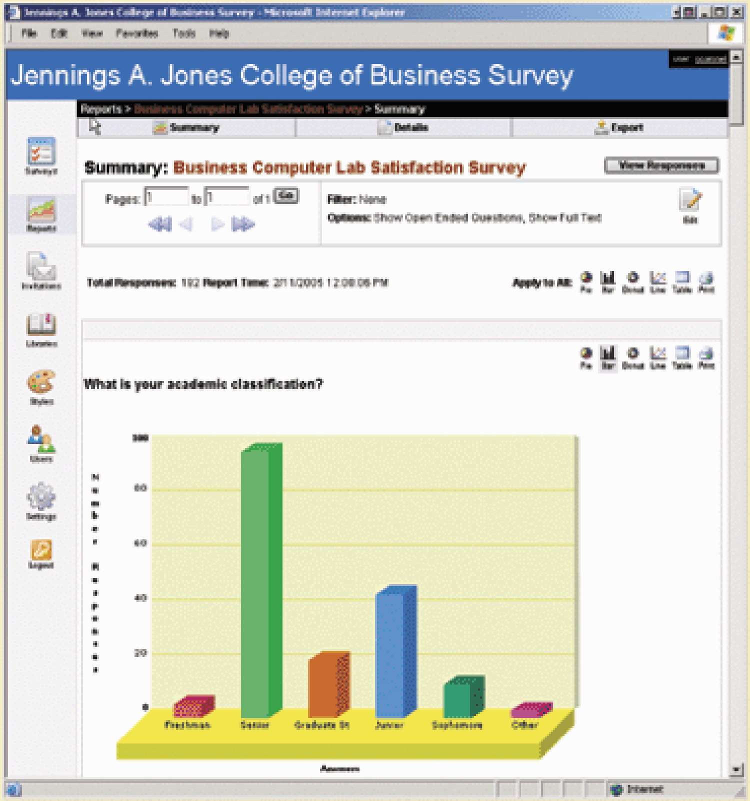
Transforming Raw Data into Information (continued)

c) Information in Summary Format



Transforming Raw Data into Information (continued)

d) Information in Graphic Format



Introducing the Database and the DBMS

- Database—shared, integrated computer structure that stores:
  - End user data (raw facts)
  - Metadata (data about data)

Introducing the Database and the DBMS (continued)

- DBMS (database management system):
  - Collection of programs that manages database structure and controls access to data
  - Possible to share data among multiple applications or users
  - Makes data management more efficient and effective

Database Management System (DBMS)

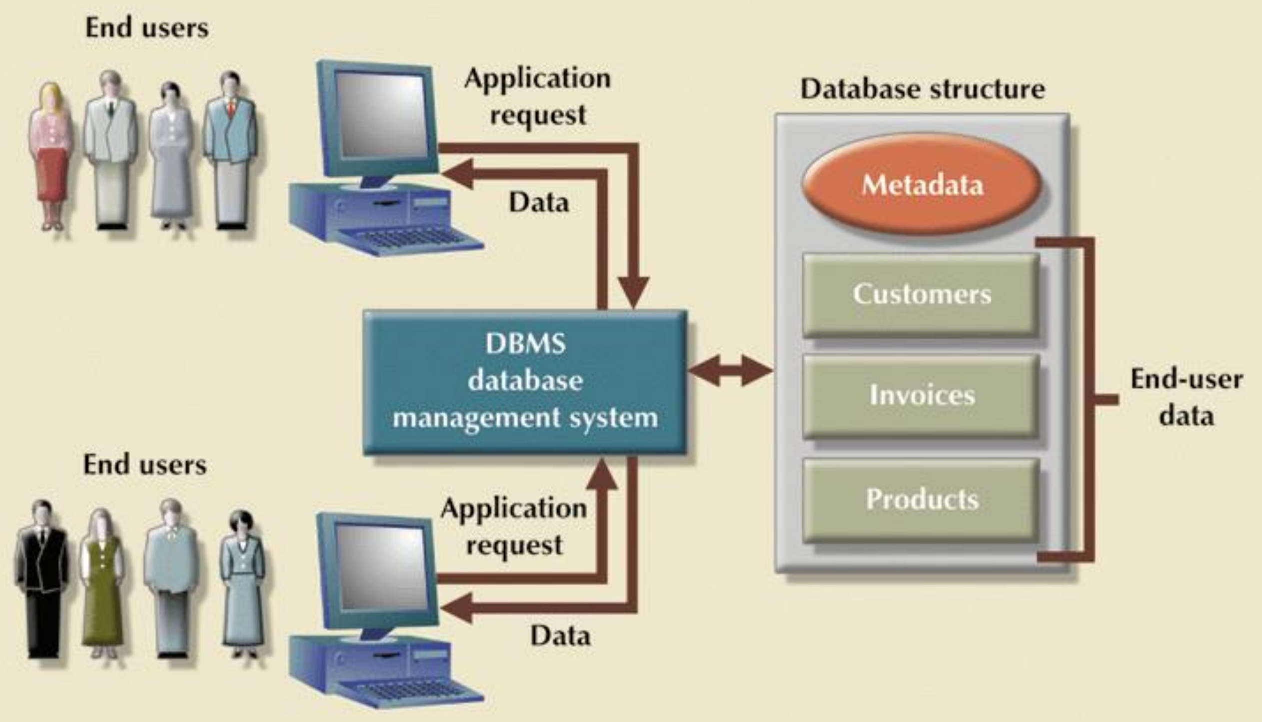
- Database Applications:
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives

Role and Advantages of the DBMS (continued)

- End users have better access to more and better-managed data
  - Promotes integrated view of organization's operations
  - Probability of data inconsistency is greatly reduced
  - Possible to produce quick answers to ad hoc queries

Role and Advantages of the DBMS (continued)

FIGURE 1.2 The DBMS manages the interaction between the end user and the database



Types of Databases

- Single-user:
  - Supports only one user at a time
- Desktop:
  - Single-user database running on a personal computer
- Multi-user:
  - Supports multiple users at the same time

Types of Databases (continued)

- Workgroup:
  - Multi-user database that supports a small group of users or a single department
- Enterprise:
  - Multi-user database that supports a large group of users or an entire organization

Types of Databases (continued)

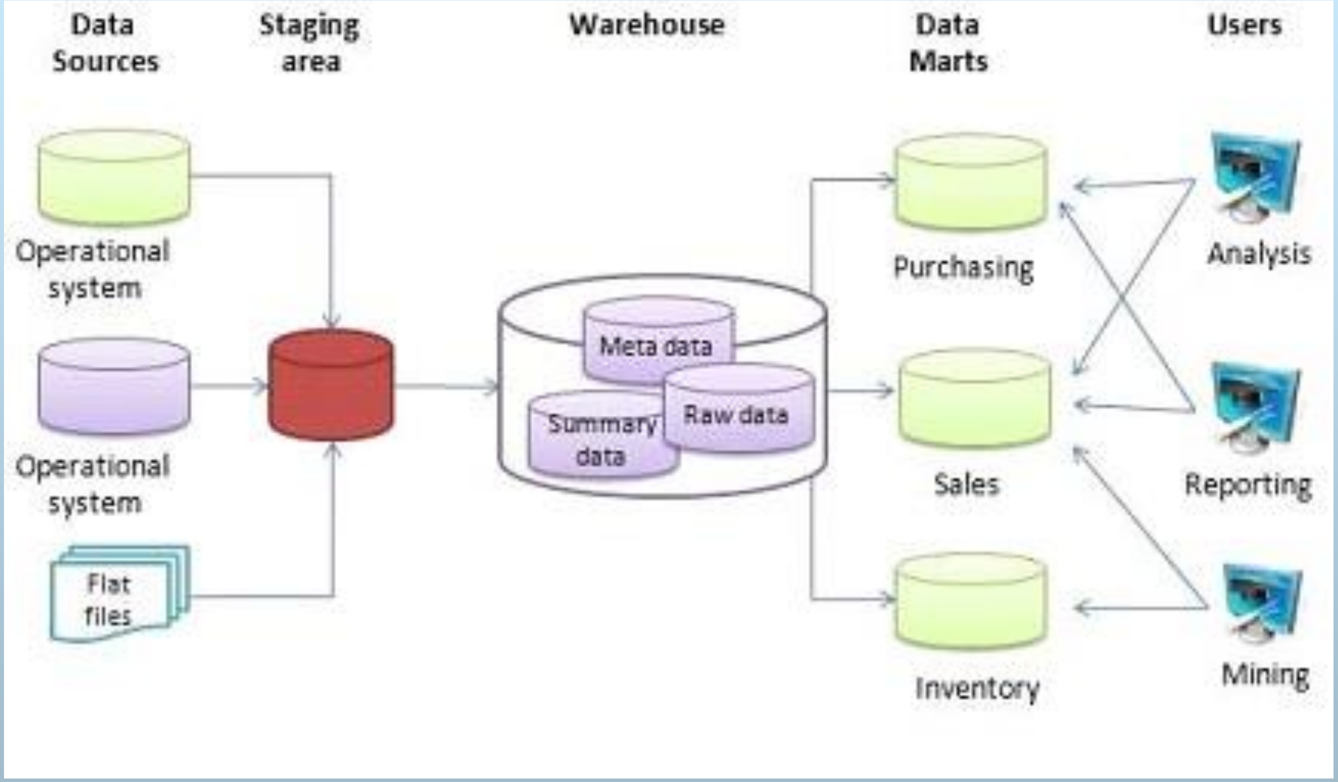
Can be classified by location:

- Centralized:
  - Supports data located at a single site
- Distributed:
  - Supports data distributed across several sites

Types of Databases (continued)

Can be classified by use:

- Transactional (or production):
  - Supports a company's day-to-day operations
- Data warehouse:
  - Stores data used to generate information required to make tactical or strategic decisions
  - Often used to store historical data
  - Structure is quite different



The basic architecture of a data warehouse

Extract, transform, load (ETL)
Extract, load, transform (E-LT)

Why Database Design is Important

- Defines the database’s expected use
- Different approach needed for different types of databases
- Avoid redundant data
- Poorly designed database generates errors → leads to bad decisions → can lead to failure of organization

Purpose of Database System

- In the early days, database applications were built on top of file systems
- Drawbacks of using file systems to store data:
  - Data redundancy and inconsistency
    - Multiple file formats, duplication of information in different files
  - Difficulty in accessing data
    - Need to write a new program to carry out each new task
  - Data isolation — multiple files and formats
  - Integrity problems
    - Integrity constraints (e.g. account balance > 0) become part of program code
    - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)
  - Atomicity of updates
    - Failures may leave database in an inconsistent state with partial updates carried out
    - E.g. transfer of funds from one account to another should either complete or not happen at all
  - Concurrent access by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - E.g. two people reading a balance and updating it at the same time
  - Security problems
- Database systems offer solutions to all the above problems

Historical Roots: Files and File Systems

- Managing data with file systems is obsolete
  - Understanding file system characteristics makes database design easier to understand
  - Awareness of problems with file systems helps prevent similar problems in DBMS
  - Knowledge of file systems is helpful if you plan to convert an obsolete file system to a DBMS

Historical Roots: Files and File Systems (continued)

Manual File systems:

- Collection of file folders kept in file cabinet
- Organization within folders based on data's expected use (ideally logically related)
- System adequate for small amounts of data with few reporting requirements
- Finding and using data in growing collections of file folders became time-consuming and cumbersome

Historical Roots: Files and File Systems (continued)

Conversion from manual to computer system:

- Could be technically complex, requiring hiring of data processing (DP) specialists
- Initially, computer files were similar in design to manual files (see Figure 1.3)

Historical Roots: Files and File Systems (continued)

FIGURE 1.3 Contents of the CUSTOMER file

C_NAME	C_PHONE	C_ADDRESS	C_ZIP	A_NAME	A_PHONE	TP	AMT	REN
Alfred A. Ramas	615-844-2573	218 Fork Rd., Babs, TN	36123	Leah F. Hahn	615-882-1244	T1	$100.00	05-Apr-2006
Leona K. Dunne	713-894-1238	Box 12A, Fox, KY	25246	Alex B. Alby	713-228-1249	T1	$250.00	16-Jun-2006
Kathy W. Smith	615-894-2285	125 Oak Ln, Babs, TN	36123	Leah F. Hahn	615-882-2144	S2	$150.00	29-Jan-2007
Paul F. Olowski	615-894-2180	217 Lee Ln., Babs, TN	36123	Leah F. Hahn	615-882-1244	S1	$300.00	14-Oct-2006
Myron Orlando	615-222-1672	Box 111, New, TN	36155	Alex B. Alby	713-228-1249	T1	$100.00	28-Dec-2006
Amy B. O'Brian	713-442-3381	387 Troll Dr., Fox, KY	25246	John T. Okon	615-123-5589	T2	$850.00	22-Sep-2006
James G. Brown	615-297-1228	21 Tye Rd., Nash, TN	37118	Leah F. Hahn	615-882-1244	S1	$120.00	25-Mar-2006
George Williams	615-290-2556	155 Maple, Nash, TN	37119	John T. Okon	615-123-5589	S1	$250.00	17-Jul-2006
Anne G. Farriss	713-382-7185	2119 Elm, Crew, KY	25432	Alex B. Alby	713-228-1249	T2	$100.00	03-Dec-2006
Olette K. Smith	615-297-3809	2782 Main, Nash, TN	37118	John T. Okon	615-123-5589	S2	$500.00	14-Mar-2006

C_NAME = Customer name
C_PHONE = Customer phone
C_ADDRESS = Customer address
C_ZIP = Customer zip code
A_NAME = Agent name
A_PHONE = Agent phone
TP = Insurance type
AMT = Insurance policy amount, in thousands of $
REN = Insurance renewal date

Historical Roots: Files and File Systems (continued)

TABLE 1.2 Basic File Terminology

TERM	DEFINITION
Data	"Raw" facts, such as a telephone number, a birth date, a customer name, and a year-to-date (YTD) sales value. Data have little meaning unless they have been organized in some logical manner. The smallest piece of data that can be "recognized" by the computer is a single character, such as the letter *A*, the number 5, or a symbol such as /. A single character requires 1 byte of computer storage.
Field	A character or group of characters (alphabetic or numeric) that has a specific meaning. A field is used to define and store data.
Record	A logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that constitute a record for a customer named J. D. Rudd might consist of J. D. Rudd's name, address, phone number, date of birth, credit limit, and unpaid balance.
File	A collection of related records. For example, a file might contain data about vendors of ROBCOR Company, or a file might contain the records for the students currently enrolled at Gigantic University.

Historical Roots: Files and File Systems (continued)

- DP specialist wrote programs for reports:
  - Monthly summaries of types and amounts of insurance sold by agents
  - Monthly reports about which customers should be contacted for renewal
  - Reports that analyzed ratios of insurance types sold by agent
  - Customer contact letters summarizing coverage

Historical Roots: Files and File Systems (continued)

- Other departments requested databases be written for them
  - SALES database created for sales department
  - AGENT database created for personnel department

Historical Roots: Files and File Systems (continued)

FIGURE 1.4 Contents of the AGENT file

A_NAME	A_PHONE	A_ADDRESS	ZIP	HIRED	YTD_PAY	YTD_FIT	YTD_FICA	YTD_SLS	DEP
Alex B. Alby	713-228-1249	123 Toll, Nash, TN	37119	01-Nov-1998	$26,566.24	$6,641.56	$2,125.30	$132,735.75	3
Leah F. Hahn	615-882-1244	334 Main, Fox, KY	25246	23-May-1984	$32,213.76	$8,053.44	$2,577.10	$138,967.35	0
John T. Okon	615-123-5589	452 Elm, New, TN	36155	15-Jun-2003	$23,198.29	$5,799.57	$1,855.86	$127,093.45	2

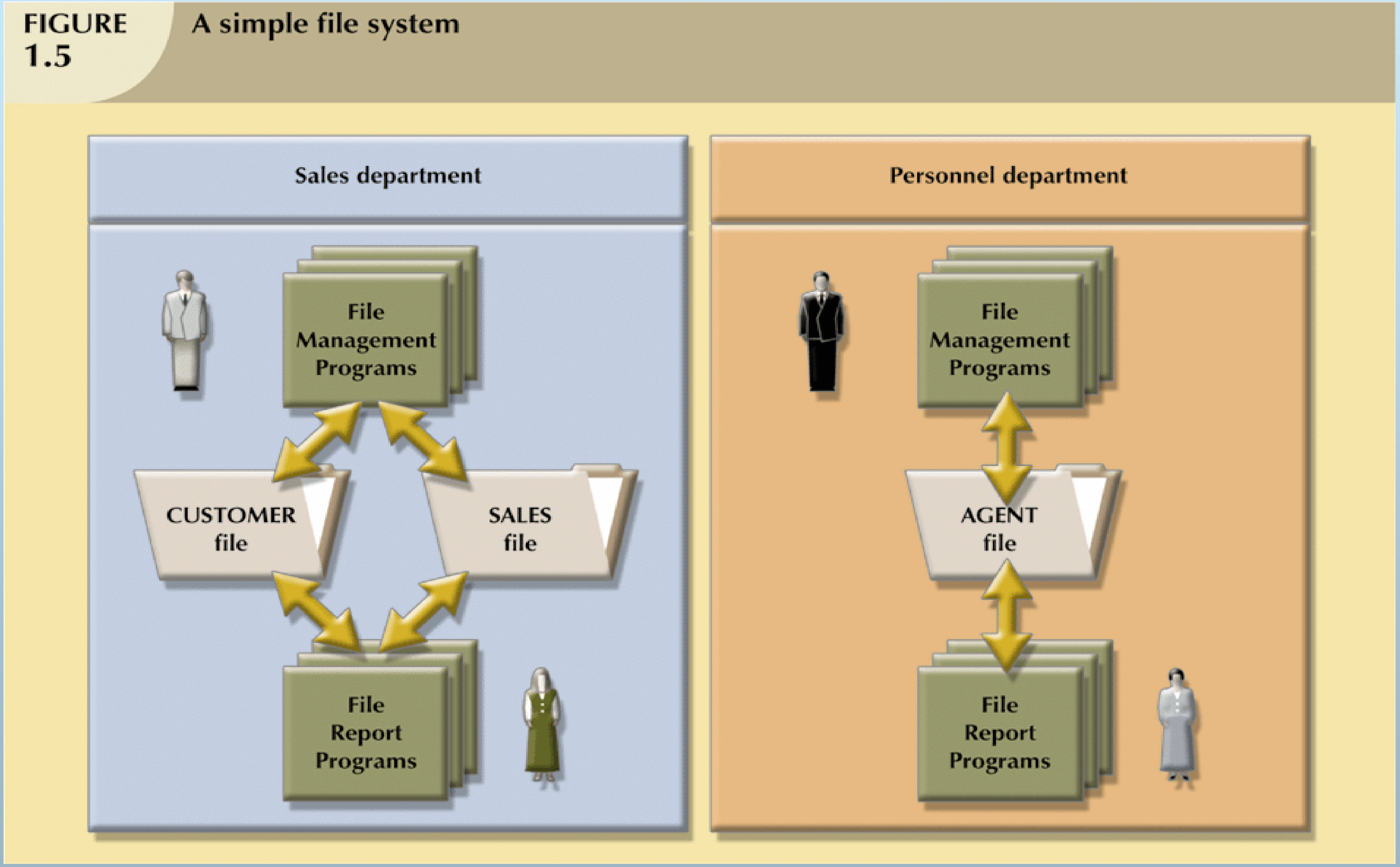
A_NAME = Agent name
A_PHONE = Agent phone
A_ADDRESS = Agent address
ZIP = Agent zip code
HIRED = Agent date of hire

YTD_PAY = Year-to-date pay
YTD_FIT = Year-to-date federal income tax paid
YTD_FICA = Year-to-date Social Security taxes paid
YTD_SLS = Year-to-date sales
DEP = Number of dependents

Historical Roots: Files and File Systems (continued)

- As number of databases increased, small file system evolved
- Each file used its own application programs
- Each file was owned by individual or department who commissioned its creation

Historical Roots: Files and File Systems (continued)



Example of Early Database Design (continued)

- As system grew, demand for DP's programming skills grew
- Additional programmers hired
- DP specialist evolved into DP manager, supervising a DP department
- Primary activity of department (and DP manager) remained programming

Problems with File System Data Management

- Every task requires extensive programming in a third-generation language (3GL)
  - Programmer must specify task and how it must be done
- Modern databases use fourth-generation languages (4GL)
  - Allow users to specify what must be done without specifying how it is to be done

Problems with File System Data Management

TABLE 1.3 3GL vs. 4GL Sample Code

3GL (GENERIC CODE)	4GL (SQL CODE)
DO WHILE NOT EOF() READ CUSTOMER IF CUSTOMER.C_ZIP = "36123" THEN PRINT C_NAME, C_PHONE, C_ZIP; ENDDO;	SELECT C_NAME, C_PHONE, C_ZIP FROM CUSTOMER WHERE CUSTOMER.C_ZIP = '36123';

Problems with File System Data Management

- Time-consuming, high-level activity
- As number of files expands, system administration becomes difficult
- Making changes in existing file structure is difficult
- File structure changes require modifications in all programs that use data in that file

Problems with File System Data Management

- Modifications are likely to produce errors, requiring additional time to “debug” the program
- Security features hard to program and therefore often omitted

Structural and Data Dependence

- Structural dependence
  - Access to a file depends on its structure

Structural and Data Dependence (continued)

- Data dependence
  - Changes in the data storage characteristics without affecting the application program’s ability to access the data
  - Logical data format
    - How the human being views the data
  - Physical data format
    - How the computer “sees” the data

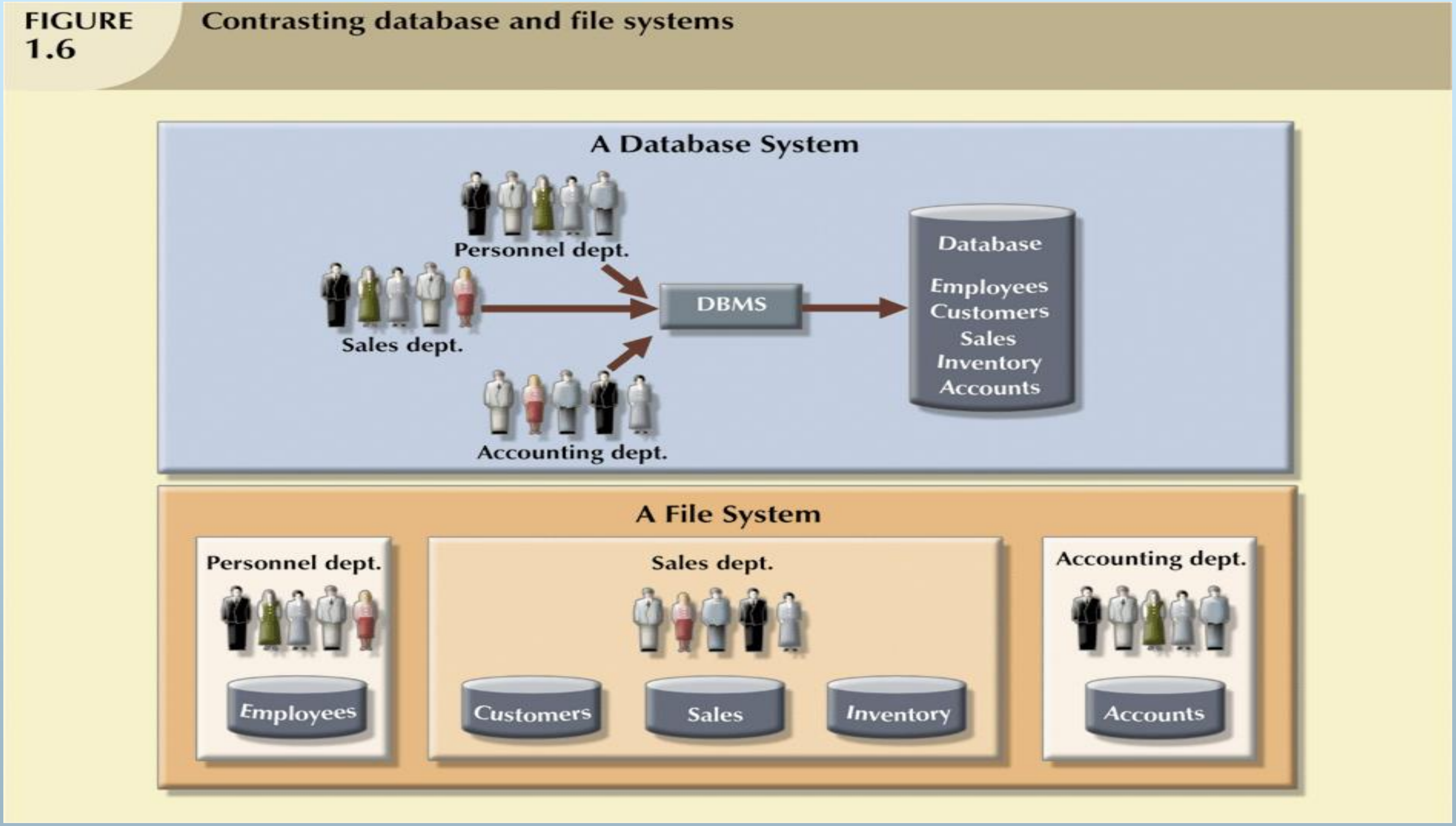
Field Definitions and Naming Conventions

- Flexible record definition anticipates reporting requirements by breaking up fields into their component parts

Database Systems

- Problems inherent in file systems make using a database system desirable
- File system
  - Many separate and unrelated files
- Database
  - Logically related data stored in a single logical data repository

Database Systems

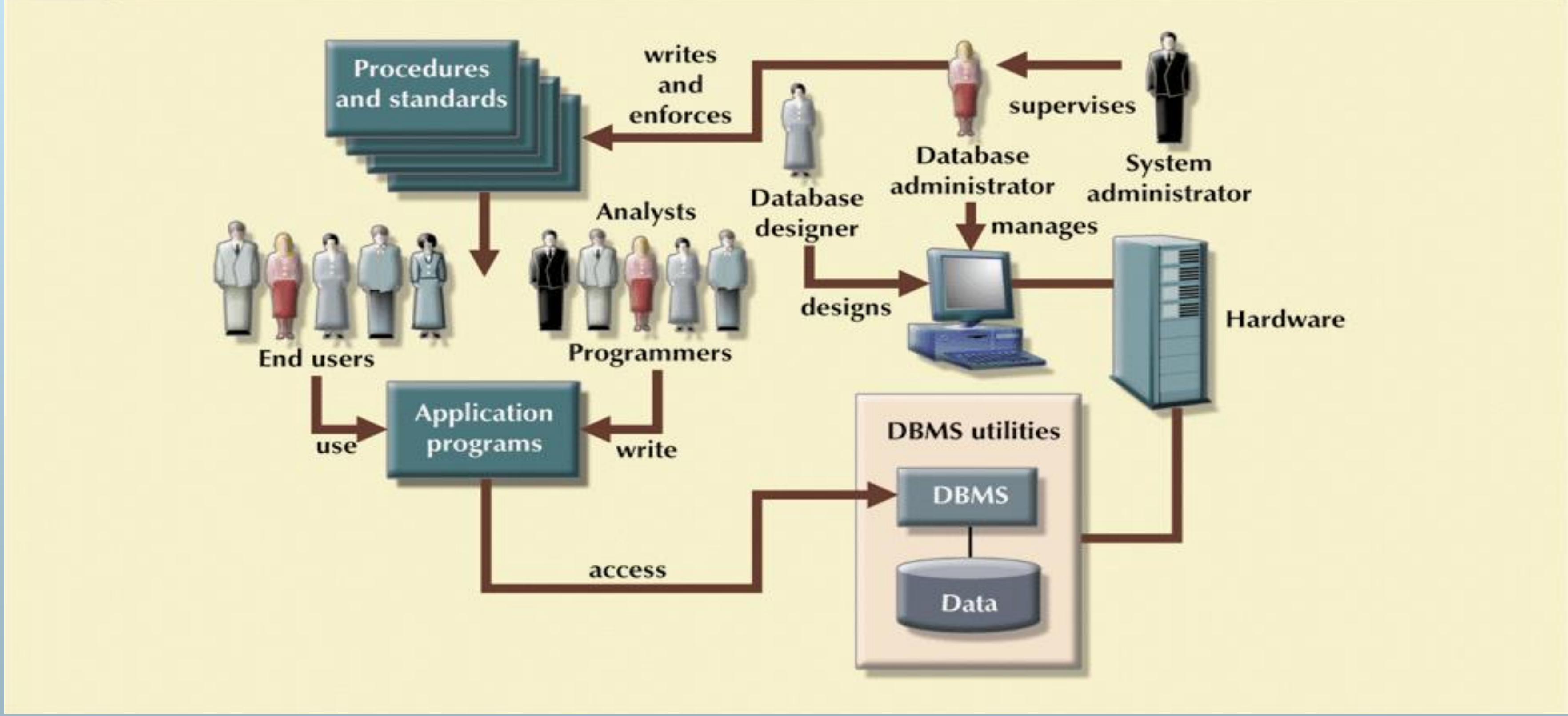


The Database System Environment

- Database system is composed of five main parts:
  - Hardware
  - Software
    - Operating system software
    - DBMS software
    - Application programs and utility software
  - People
  - Procedures
  - Data

The Database System Environment (continued)

FIGURE 1.7 The database system environment



DBMS Functions

- DBMS performs functions that guarantee integrity and consistency of data
  - Data dictionary management
    - defines data elements and their relationships
  - Data storage management
    - stores data and related data entry forms, report definitions, etc.

DBMS Functions (continued)

- Data transformation and presentation
  - translates logical requests into commands to physically locate and retrieve the requested data
- Security management
  - enforces user security and data privacy within database

DBMS Functions (continued)

- Multiuser access control
  - uses sophisticated algorithms to ensure multiple users can access the database concurrently without compromising the integrity of the database
- Backup and recovery management
  - provides backup and data recovery procedures
- Data integrity management
  - promotes and enforces integrity rules

DBMS Functions (continued)

- Database access languages and application programming interfaces
  - provide data access through a query language
- Database communication interfaces
  - allow database to accept end-user requests via multiple, different network environments

Data Abstraction

- Major aim of a DBMS is to provide users with an abstract view of data
- Hides certain details of how the data are stored & maintained
- DBMS must retrieve data efficiently
- Need for efficiency has led designers to use complex data structures to represent the data in the database
- Most DB users are not computer trained, developers hide complexity through several levels of abstraction to simplify user's interaction with the systems

Levels of Abstraction

- Physical level describes how a record (e.g., customer) is stored.
- Logical level: describes data stored in database, and the relationships among the data.

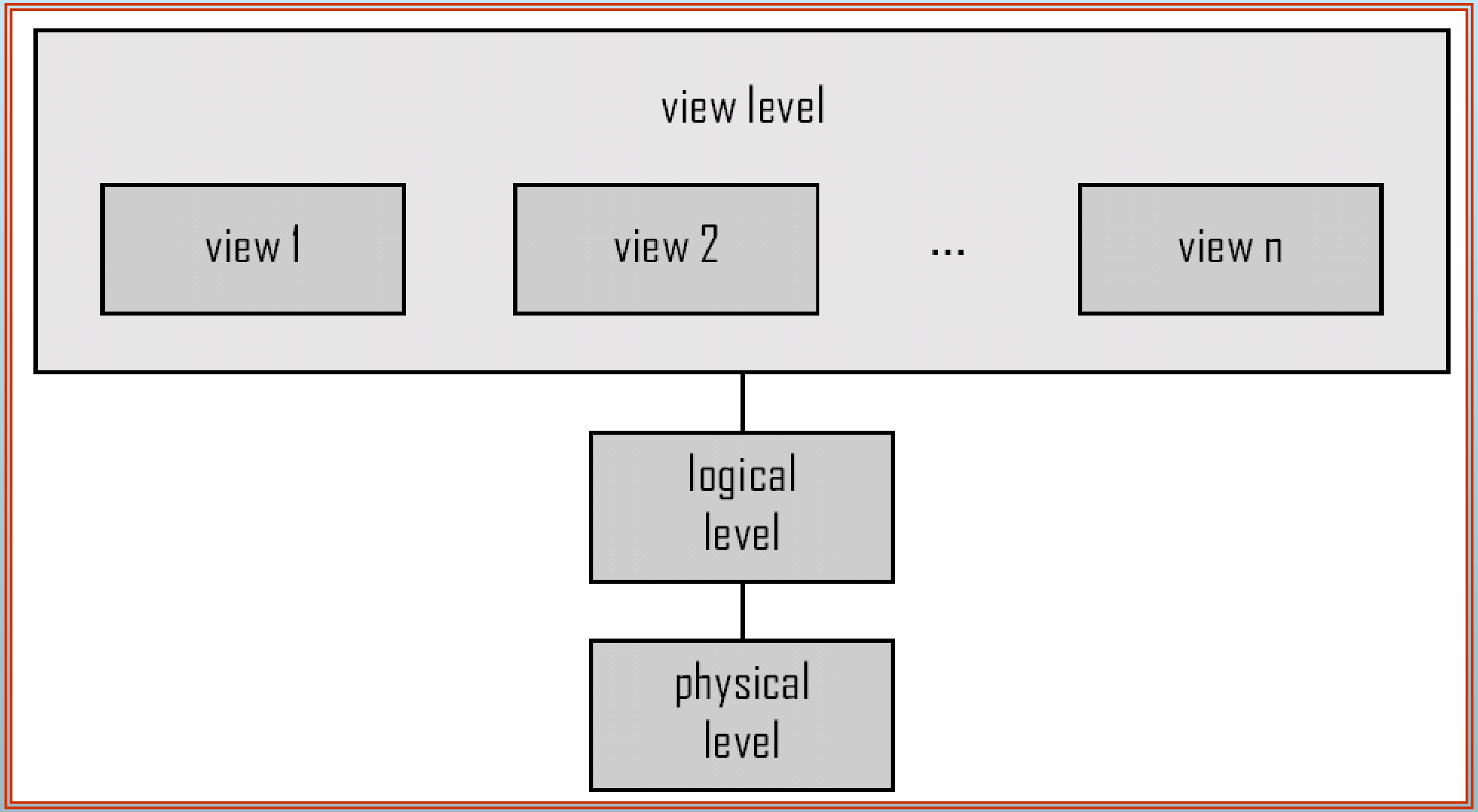
type customer = **record**
name : string;
street : string;
city : integer;
end;

- View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.

3 Levels of Abstraction

- **Logical or Conceptual Level**
  - Describes what data are stored in the DB & what relationships exist among those data
  - Describes the entire DB in terms of relatively simpler structures
  - Implementation of these simple structures at this level may involve complex physical-level structures
  - Users of the logical level need not be aware of this complexity
  - DBAs, who decide what information to keep in DB, use the logical level of abstraction

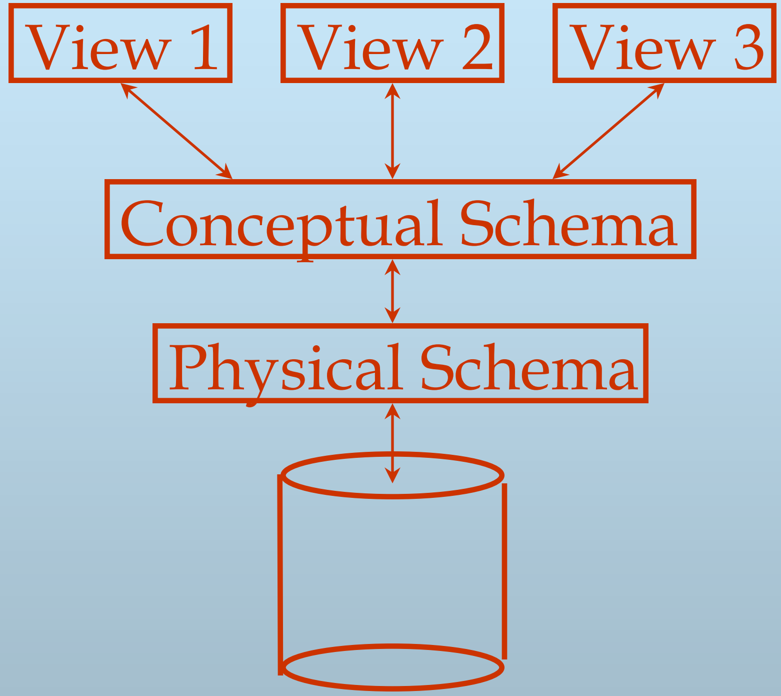
Levels of Abstraction



Levels of Abstraction

Many *views*, single *conceptual (logical) schema* and *physical schema*.

Views describe how users see the data.
Conceptual schema defines logical structure
Physical schema describes the files and indexes used.



☒ ***Schemas are defined using DDL; data is modified/queried*** *using DML.*

Instances & Schemas

Collection of information stored in the DB at a particular moment is called an INSTANCE

The overall design of the DB is called a SCHEMA

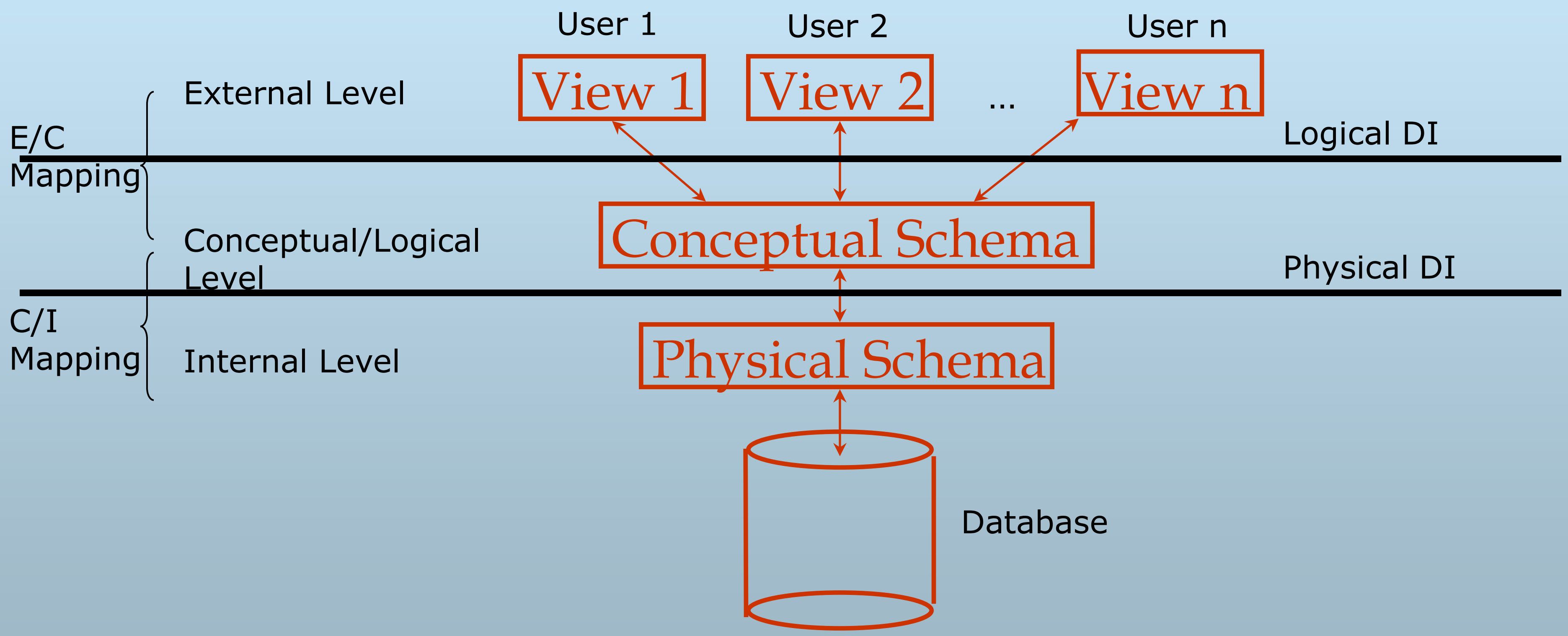
A DB has many schemas

- Physical
- Conceptual/Logical
- Sub-schemas

Example: University Database

- **Logical schema:**
  - ***Students(sid: string, name: string, login: string, age: integer, gpa:real)***
  - ***Faculty(fid:string, fname:string, sal:real)***
  - ***Courses(cid: string, cname:string, credits:integer)***
  - ***Enrolled(sid:string, cid:string, grade:string)***
- **Physical schema:**
  - **Relations stored as unordered files.**
  - **Index on first column of Students.**
- **External Schema (View):**
  - ***Course_info(cid:string,fname:string, enrollment:integer)***

ANSI/SPARC 3-Tier Architecture

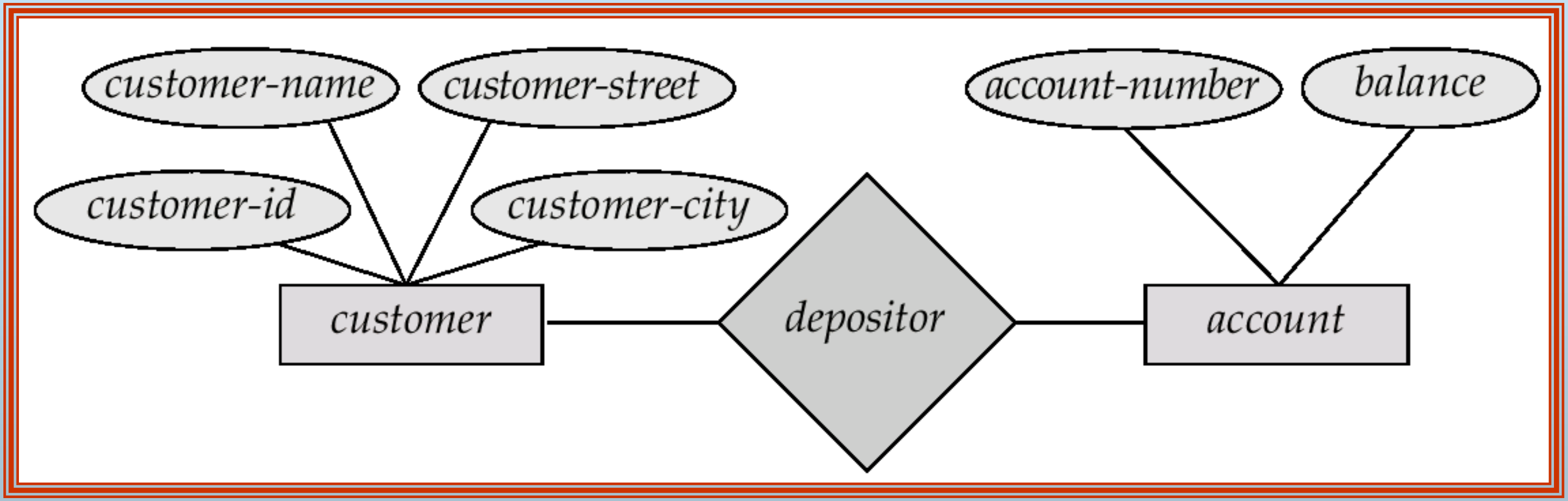


Data Models

- A collection of tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints
- Entity-Relationship model
- Relational model
- Other models:
  - object-oriented model
  - semi-structured data models
  - Older models: network model and hierarchical model

Entity-Relationship Model

Example of schema in the entity-relationship model



Entity Relationship Model (Cont.)

- E-R model of real world
  - Entities (objects)
    - E.g. customers, accounts, bank branch
  - Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set *depositor* associates customers with accounts
- Widely used for database design
  - Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing

Relational Model

Attributes

- Example of tabular data in the relational model

Customer-id	*customer-name*	*customer-street*	*customer-city*	*account-number*
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

A Sample Relational Database

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Data Definition Language (DDL)

- Specification notation for defining the database schema
  - E.g.

    create table *account* (
    account-number **char**(10),
    balance **integer**)
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - database schema
  - Data *storage and definition* language
    - language in which the storage structure and access methods used by the database system are specified
    - Usually an extension of the data definition language

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language

SQL

- SQL: widely used non-procedural language
  - E.g. find the name of the customer with customer-id 192-83-7465

    select *customer.customer-name*
    from *customer*
    where *customer.customer-id* = '192-83-7465'

  - E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465

    select *account.balance*
    from *depositor*, *account*
    where *depositor.customer-id* = '192-83-7465' **and**
    depositor.account-number = *account.account-number*

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database

Database Users

- Users are differentiated by the way they expect to interact with the system
- Application programmers – interact with system through DML calls
- Sophisticated users – form requests in a database query language
- Specialized users – write specialized database applications that do not fit into the traditional data processing framework
- Naïve users – invoke one of the permanent application programs that have been written previously
  - E.g. people accessing database over the web, bank tellers, clerical staff

Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

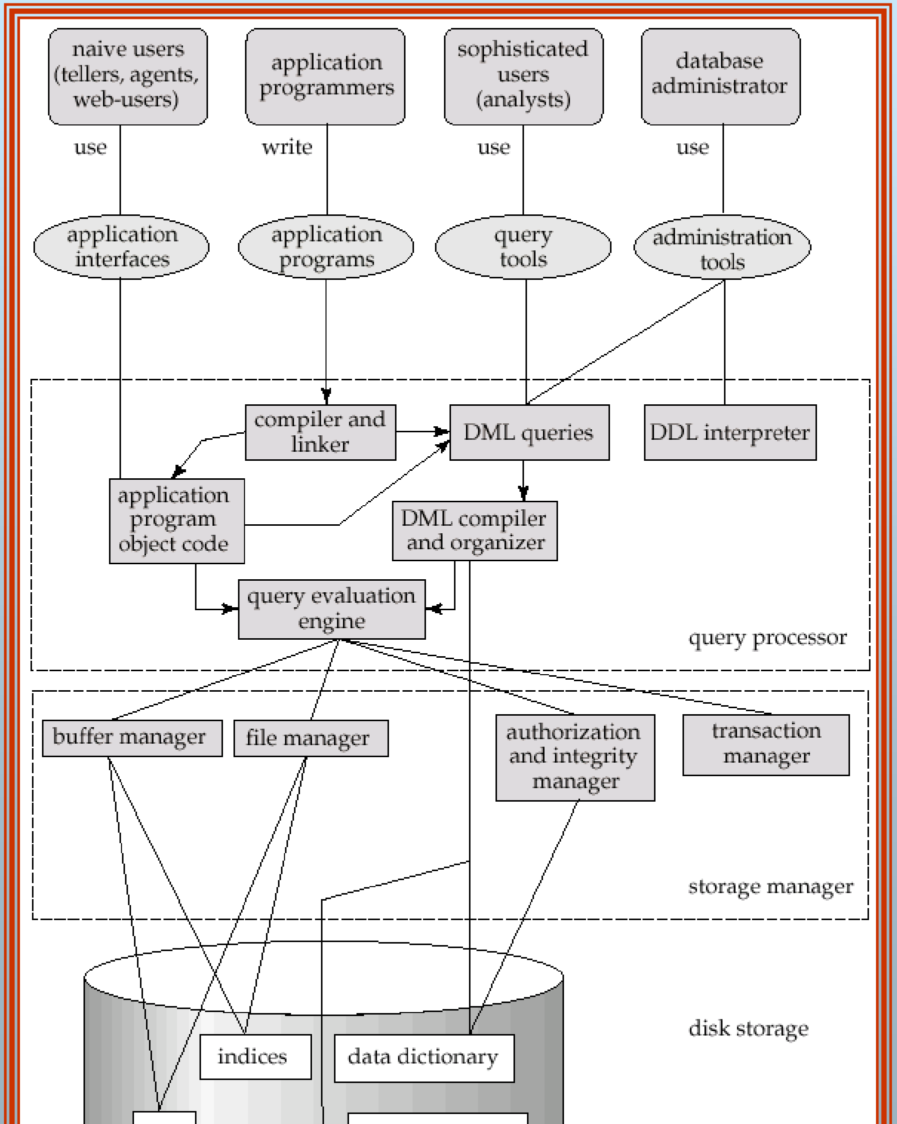
Transaction Management

- A *transaction* is a collection of operations that performs a single logical function in a database application
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

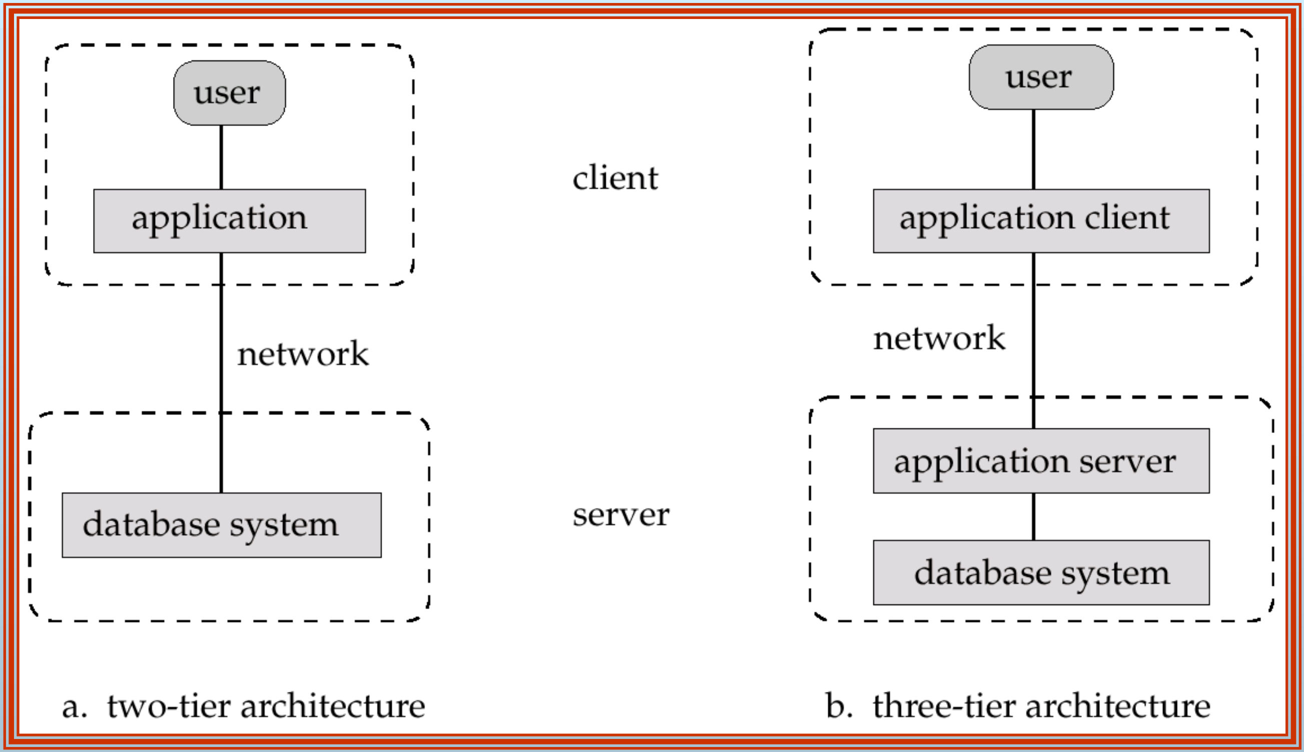
Storage Management

- Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data

Overall System Structure



Application Architectures



- **Two-tier architecture**: E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture**: E.g. web-based applications, and applications built using "middleware"