Database System Concepts and Architecture

406.426 Design & Analysis of Database Systems

Jonghun Park

jonghun@snu.ac.kr

Dept. of Industrial Engineering

Seoul National University

outline

- data models and their categories
- history of data models
- schemas, instances, and states
- three-schema architecture
- data independence
- DBMS languages and interfaces
- database system utilities and tools
- centralized and client-server architectures
- classification of DBMSs

data models

- DB approach provides some level of **data abstraction** by **hiding details of data storage** that are not needed by most DB users
- data model: a collection of concepts to describe the **structure** of a database, and certain **constraints** that the database should obey
- structure of a DB: data types, relationships, and constraints
- data model operations
  - **basic** operations: for specifying **retrievals** and **updates** on the DB
  - **user-defined** operations: for specifying the **behavior** of a database application (e.g., COMPUTE_GPA)

categories of data models

- **conceptual** (high-level, semantic) data models
  - provide concepts that are close to the way many users perceive data
  - uses concepts such as **entities**, **attributes**, and **relationships**
- **physical** (low-level, internal) data models
  - provide concepts that describe details of **how data is stored** in the computer
  - example: record formats, record orderings, access paths
- **representational (**implementation) data models
  - provide concepts that fall between the above two, balancing user views with some computer storage details
  - example: **relational**, **network**, and **hierarchical** models

schemas versus instances

- database schema (aka **intension**):
  - description of a database
  - includes descriptions of the database **structure** and the **constraints** that should hold on the database
  - not expected to change frequently
- schema diagram
  - a diagrammatic display of (some aspects of) a database schema
- schema construct
  - a **component** of the schema or an **object** within the schema
  - e.g., STUDENT, COURSE.
- database state (aka **extension** of a schema)
  - the actual data stored in a database at a particular moment in time.
  - also called the current set of occurrences or instances

example of a schema diagram

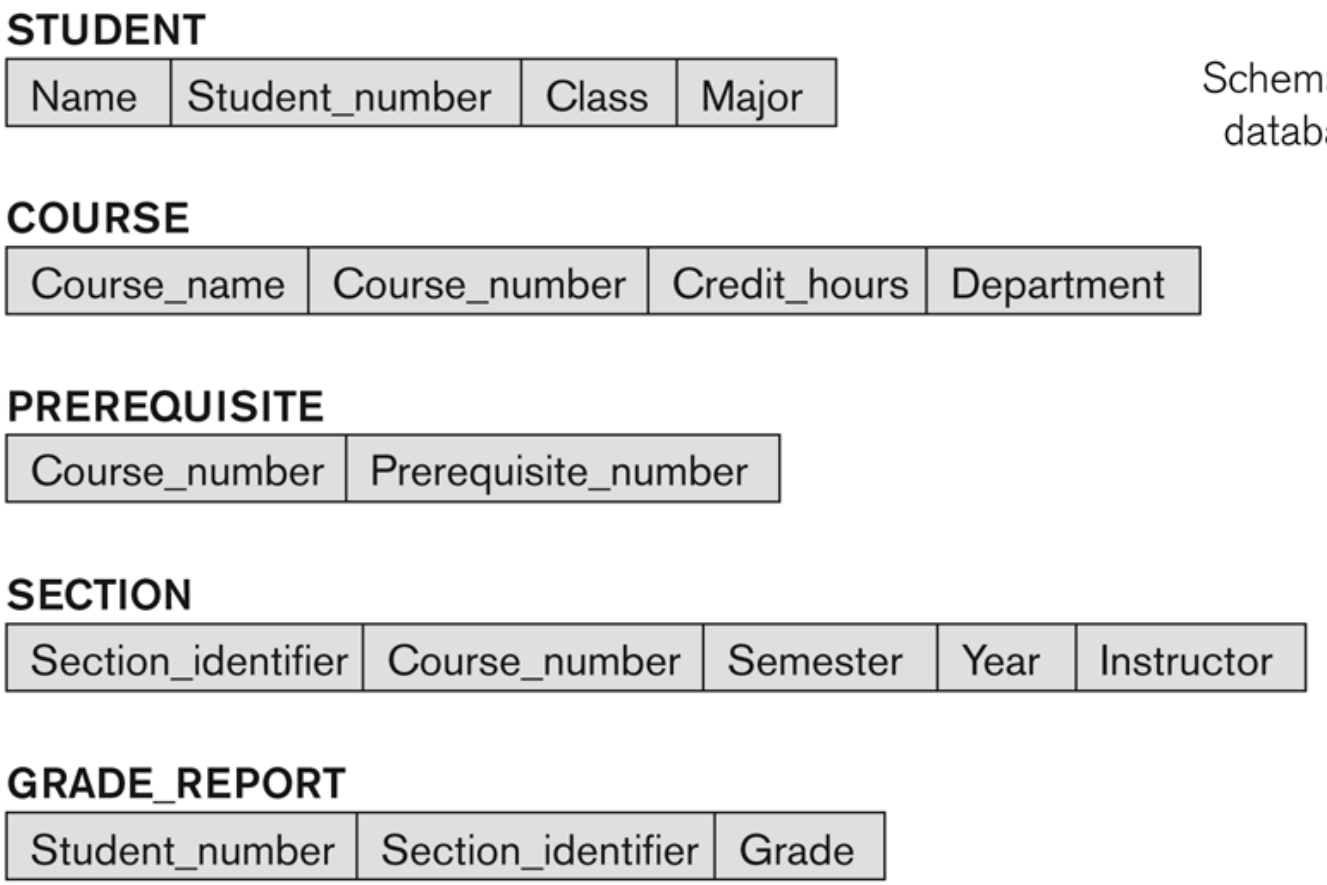


Figure 2.1
Schema diagram for the database in Figure 1.2.

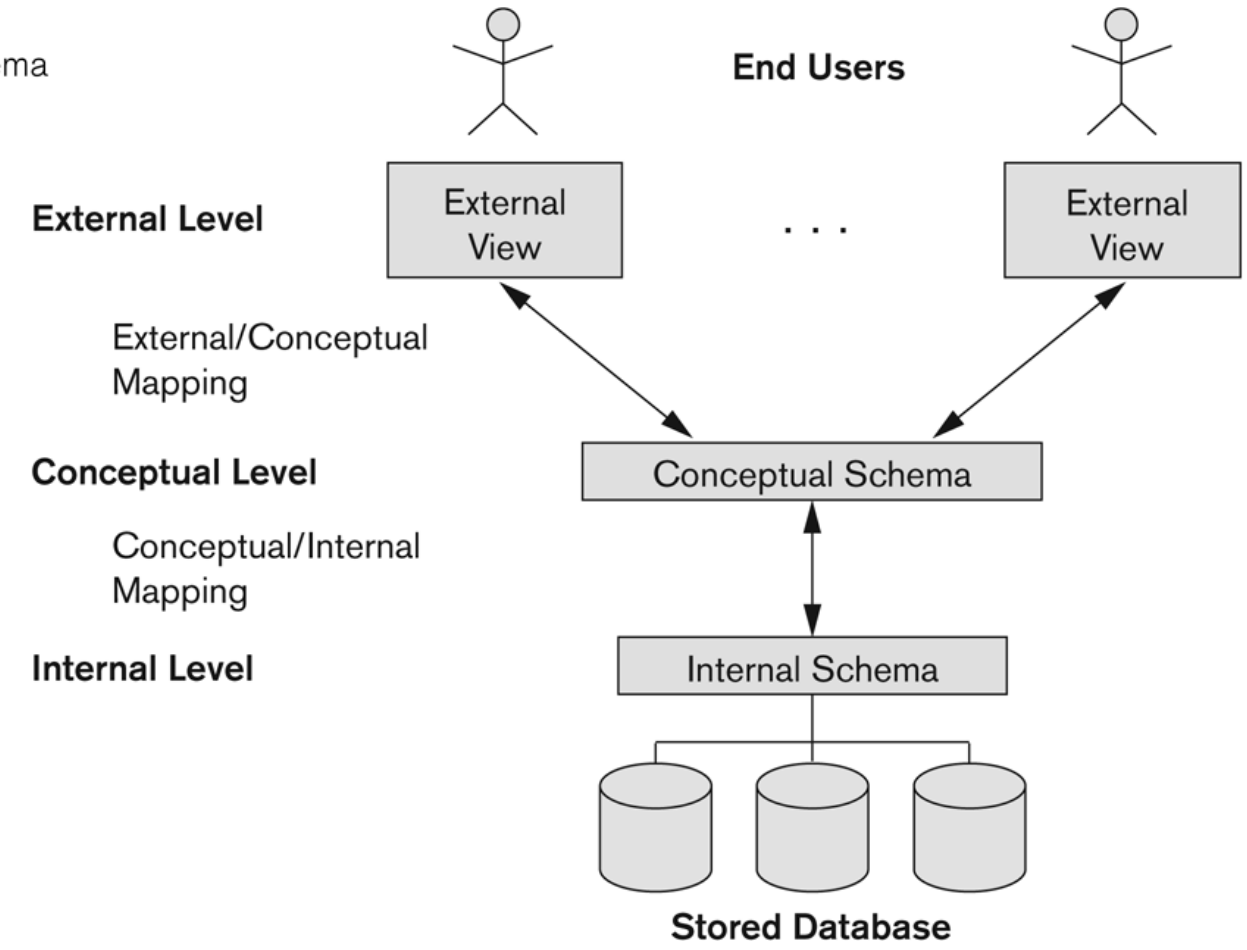
database schema vs. database state

- DB s**tate**: refers to the **content** of a database at a moment in time
- **initial** database state: refers to the database when it is loaded
- **valid** state: a state that satisfies the **structure** and **constraints** specified in the schema
- **meta-data**: the descriptions of the **schema constructs** and **constraints**
- schema evolution
  - most modern DBMS include some operations for schema evolution that can be applied while the database is operational

three-schema architecture

- goal: to separate the user applications and the physical database
- benefits of “layering”

Figure 2.2
The three-schema architecture.



three-schema architecture

- internal level: has an **internal schema** which describes the **physical storage structure** of the DB
- conceptual level
  - has a **conceptual schema**, which describes the **structure of the whole database** for a community of users
  - conceptual schema **hides the details of physical storage** structures and concentrates on describing **entities, data types, relationships, user operations**, and **constraints**
- external level (view level)
  - includes a number of external schemas or **user views**
  - each external schema describes **the part of the database** that a particular user group is interested in and **hides** the rest of the database from that user group
- **mappings** among schema levels are needed to transform requests and data -> overhead of DBMS approach!

data independence

- **data independence**: the capacity to **change the schema at one level** of a database system without having to change the schema at the next higher level
- logical data independence:
  - the capacity to **change the conceptual schema** without having to change the external schemas and their application programs
  - e.g., expand the DB, change constraints, reduce the DB
- physical data independence
  - the capacity to **change the internal schema** without having to change the conceptual schema
  - sometimes done to improve the performance

data independence

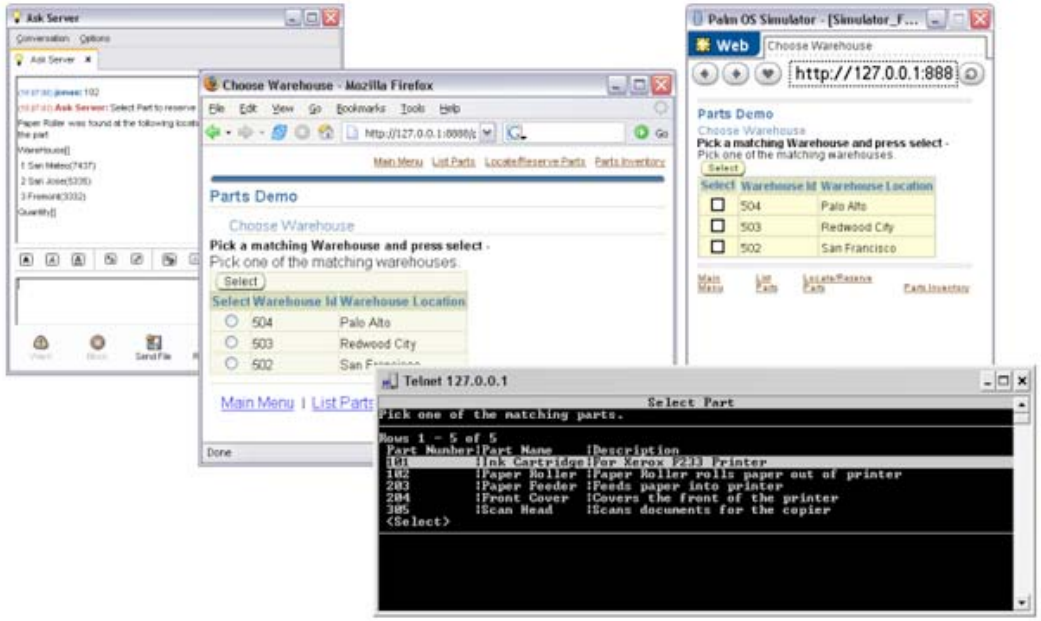
- when a schema at a lower level is changed, **only the mappings** between this schema and higher-level schemas **need to be changed** in a DBMS that fully supports data independence -> t**he higher-level schemas themselves are unchanged!**
- hence, the application programs need not be changed since they refer to the external schemas
- however, two levels of mappings create an **overhead** during compilation or execution of a query or program
  - few DBMSs have implemented the full three-schema architecture

DBMS languages

- data definition language (DDL)
  - used by the DBA and database designers to specify the **conceptual, internal, and external schemas** of a database
- storage definition language (SDL)
  - specifies the internal schema
- view definition language (VDL)
  - specifies user views and their mappings to the conceptual schema
- data manipulation language (DML)
  - used to specify database manipulations
  - high-level DML: to specify complex DB operations in a concise manner (used by casual end users)
  - low-level DML: must be embedded in a general-purpose PL (used by programmers)
- **SQL**: represents a combination of DDL, VDL, and DML, as well as statements for constraint specification, schema evolution, and other features

DBMS interfaces

- menu-based interfaces for web clients or browsing
- forms-based interfaces
- graphical user interfaces
- natural language interfaces
- speech IO
- interfaces for parametric users (special interfaces): e.g., function keys
- interfaces for the DBA



DBMS component modules

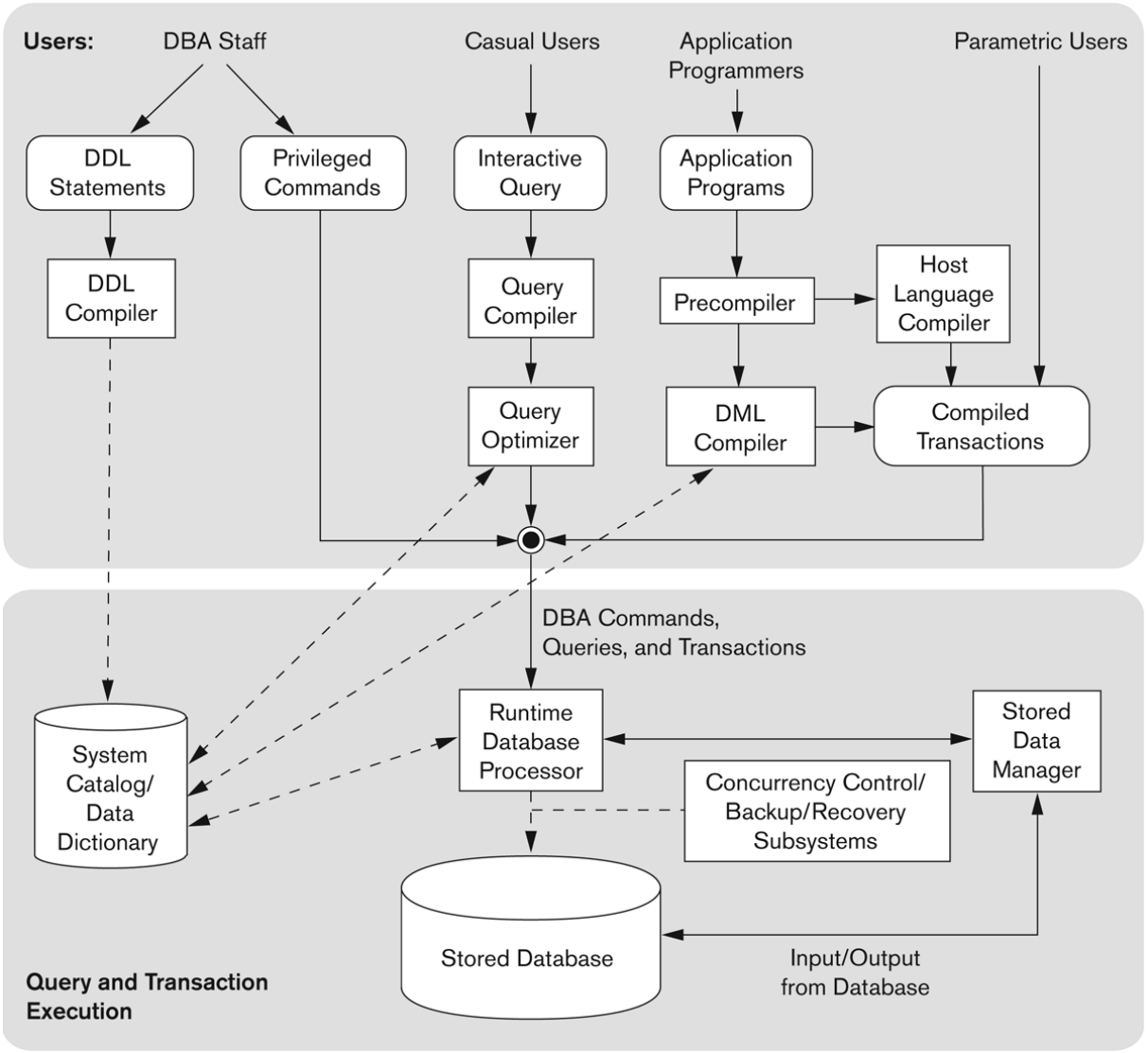


Figure 2.3
Component modules of a DBMS and their interactions.

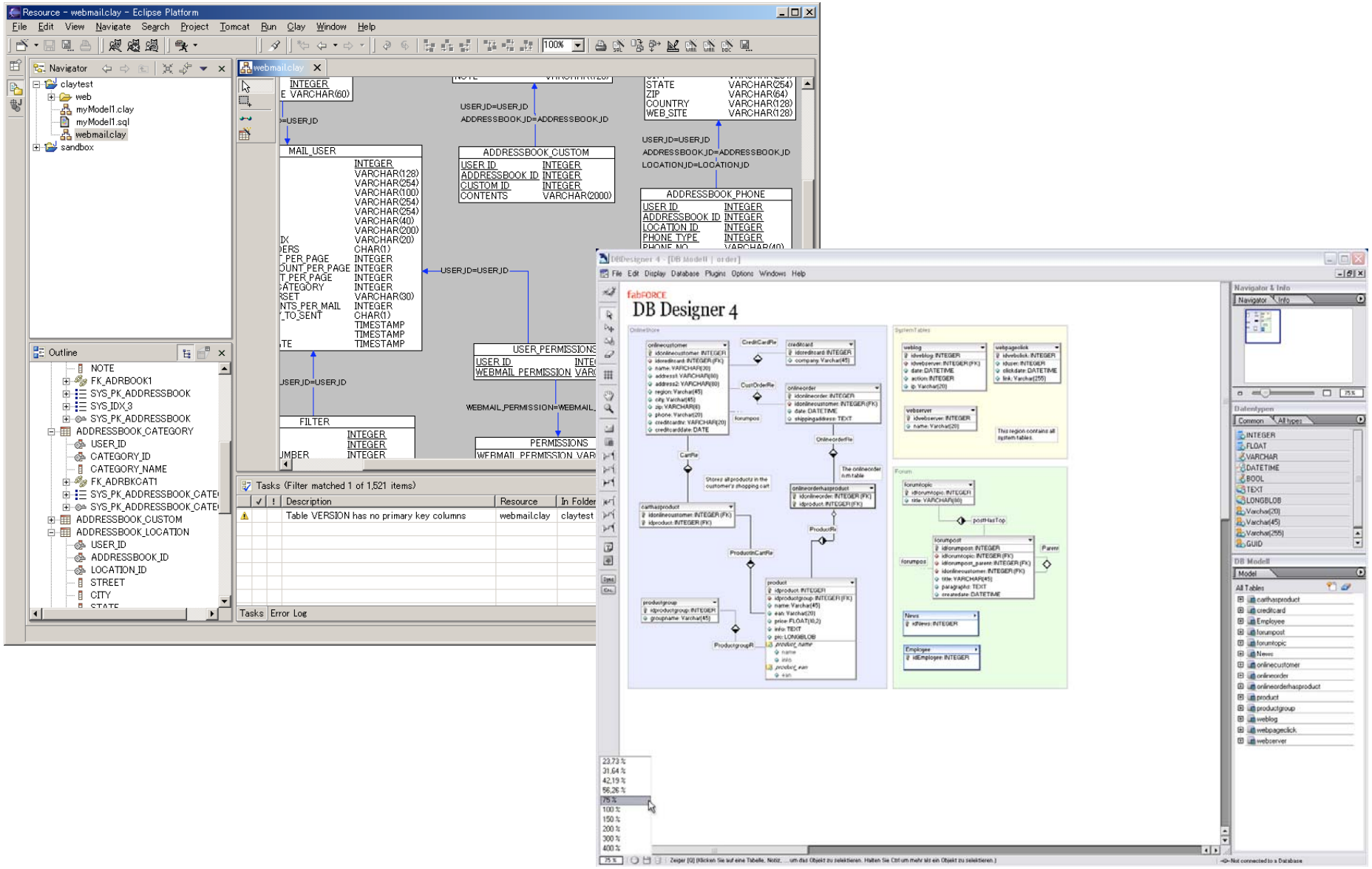
database system utilities

- help the DBA in **managing** the database system
- loading data stored in files into a database
  - includes data conversion tools
- backing up the database periodically on tape
- reorganizing database file structures
- report generation utilities
- performance monitoring utilities
- other functions, such as sorting, user monitoring, data compression, etc.

other tools

- data repository:
  - used to store other information such as design decisions, application program descriptions, user information, usage standards, etc., as well as **schema descriptions** and **constraints**
- application development environments and CASE (computer-aided software engineering) tools:
  - provide an environment for **developing database applications** and include facilities that help in many facets of database systems, including DB design, GUI development, querying and updating, and application program development
  - examples: PowerBuilder (Sybase), JBuilder (Borland)
- interface with communication software: APIs

examples


Resource – webmail.clay – Eclipse Platform
File Edit View Navigate Search Project Tomcat Run Clay Window Help
Navigator
claytest
web
myModel1.clay
myModel1.sql
webmail.clay
sandbox
Outline
NOTE
FK_ADRBOOK1
SYS_PK_ADDRESSBOOK
SYS_IDX_3
SYS_PK_ADDRESSBOOK
ADDRESSBOOK_CATEGORY
USER_ID
CATEGORY_ID
CATEGORY_NAME
FK_ADRBKCAT1
ADDRESSBOOK_CUSTOM
ADDRESSBOOK_LOCATION
USER_ID
ADDRESSBOOK_ID
LOCATION_ID
STREET
CITY
webmail.clay
MAIL_USER
ADDRESSBOOK_CUSTOM
USER ID INTEGER
ADDRESSBOOK ID INTEGER
CUSTOM ID INTEGER
CONTENTS VARCHAR(2000)
ADDRESSBOOK_PHONE
USER ID INTEGER
ADDRESSBOOK ID INTEGER
LOCATION ID INTEGER
PHONE TYPE INTEGER
USER_PERMISSIONS
USER ID
WEBMAIL PERMISSION
FILTER
PERMISSIONS
USERJD=USERJD
ADDRESSBOOK_JD=ADDRESSBOOK_JD
LOCATIONJD=LOCATIONJD
WEBMAIL_PERMISSION=WEBMAIL
Tasks (Filter matched 1 of 1,521 items)
Description
Resource
In Folder
Table VERSION has no primary key columns
webmail.clay
claytest
Tasks
Error Log
fabFORCE
DB Designer 4

centralized and client-server architectures

- centralized DBMS
  - combines all functions into **single system** including – user application programs and user interface programs, as well as the DBMS functionality
- client / server architecture
  - developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, web servers, and other equipment are connected via a network
  - idea: to define **specialized servers** with specific functionalities (e.g., file server, printer server, email server, web server)
  - the client machines provide the user with the **appropriate interface**s to utilize these servers, as well as with local processing power to run local applications
  - a server is a machine that can provide services to the client machines

a physical centralized architecture

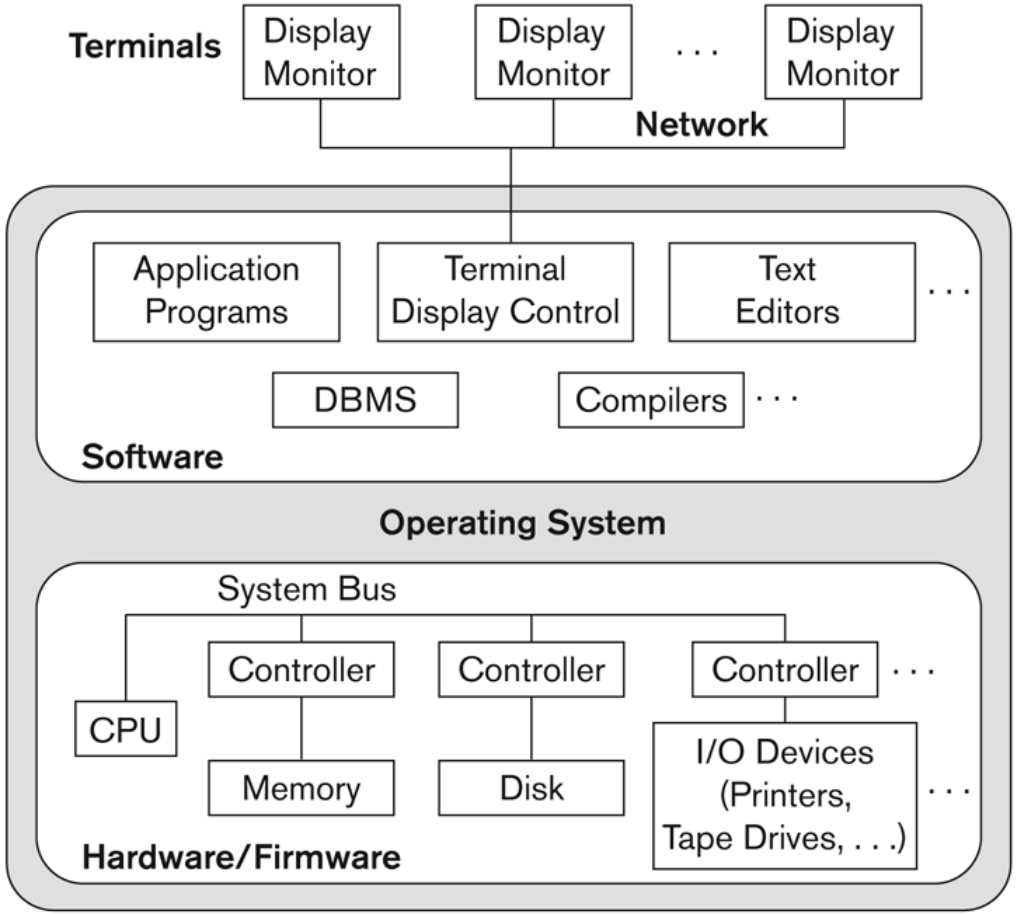
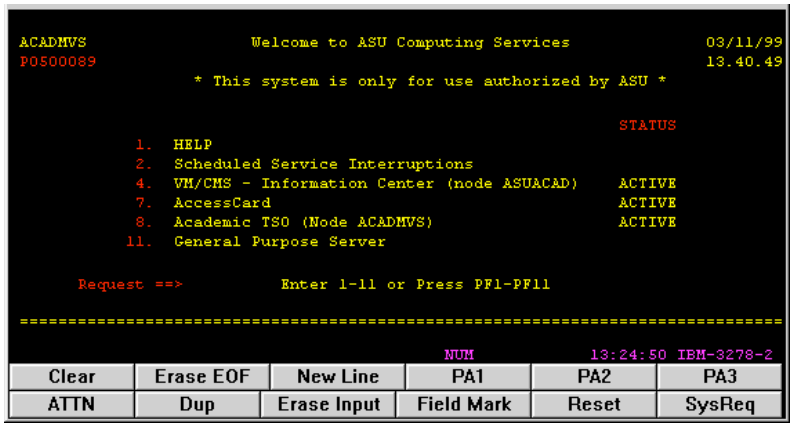
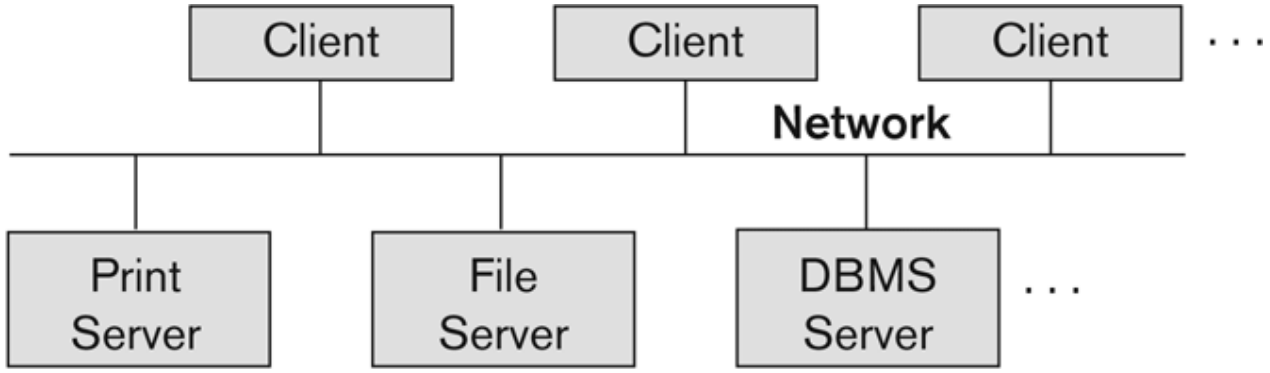


Figure 2.4
A physical centralized architecture.

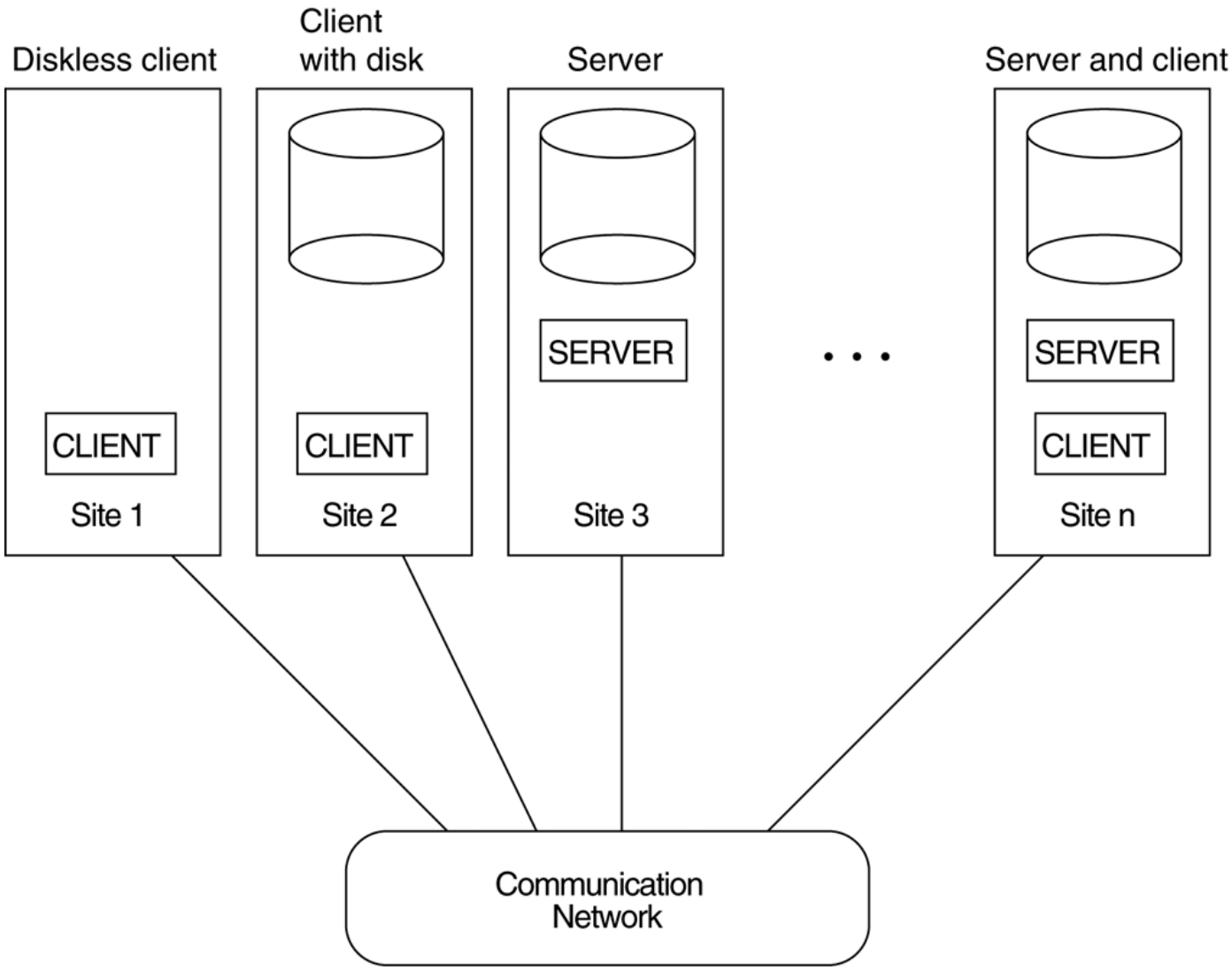


logical two-tier CS architecture

Figure 2.5
Logical two-tier client/server architecture.

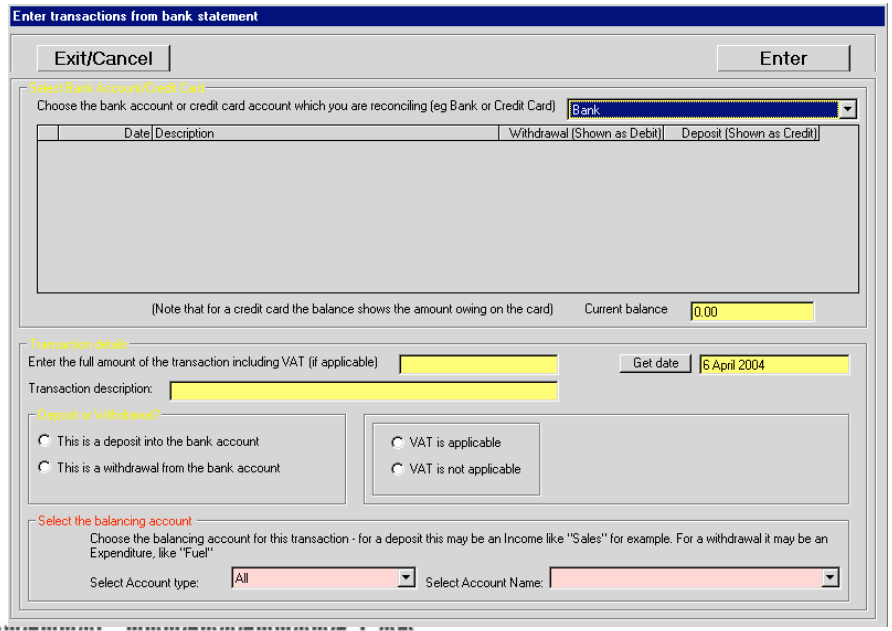


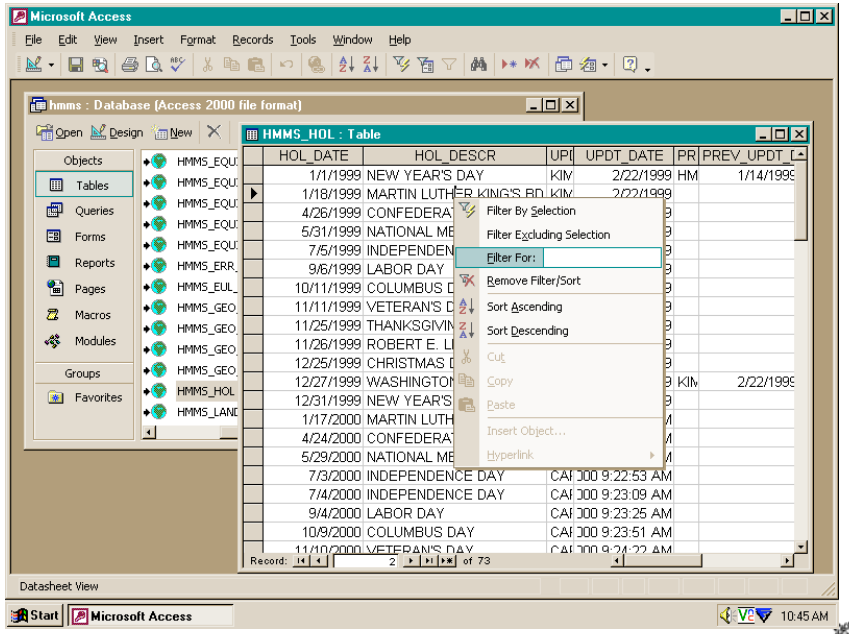
physical two-tier CS architecture



2-tier CS architectures for DBMSs

- client: **user interface** + **application programs**
- server: **query** and **transaction** functionality (hence the names such as query server, transaction server, and SQL server)
- ODBC
  - provides an **API** which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed
  - cf: JDBC
- advantages: simplicity and seamless compatibility with existing systems





3-tier CS architectures for web applications

- many **web applications** use 3-tier CS architecture
- adds an intermediate layer between the client and the DB server
- application server: plays an intermediary role by storing **business logic** that are used to access data from the DB server

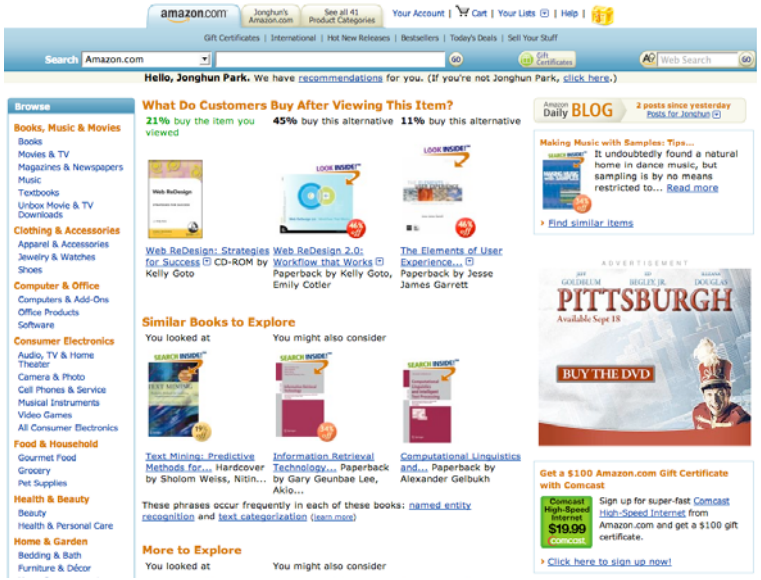
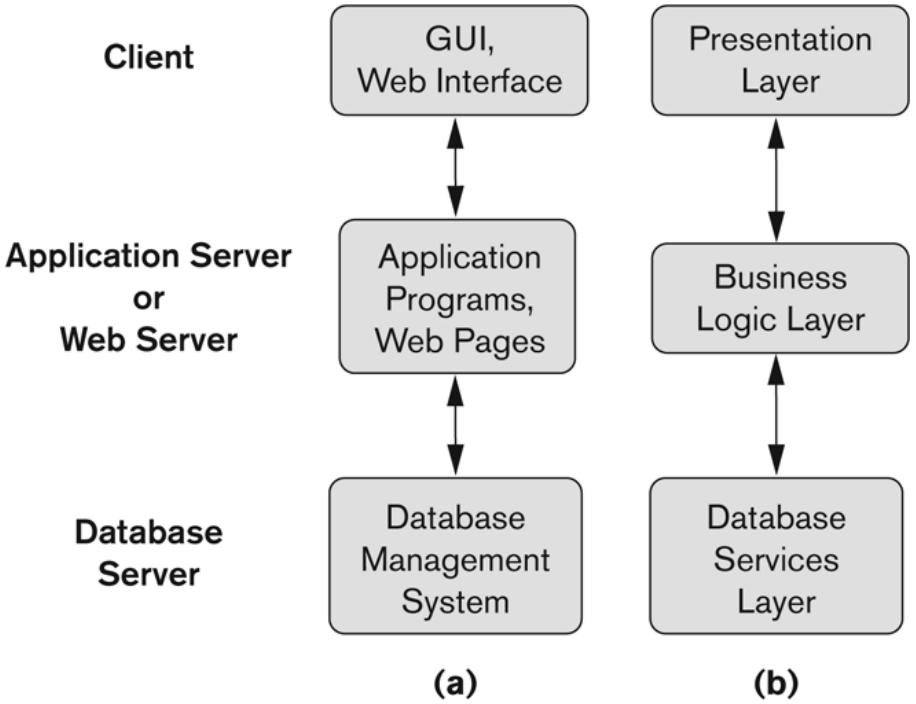
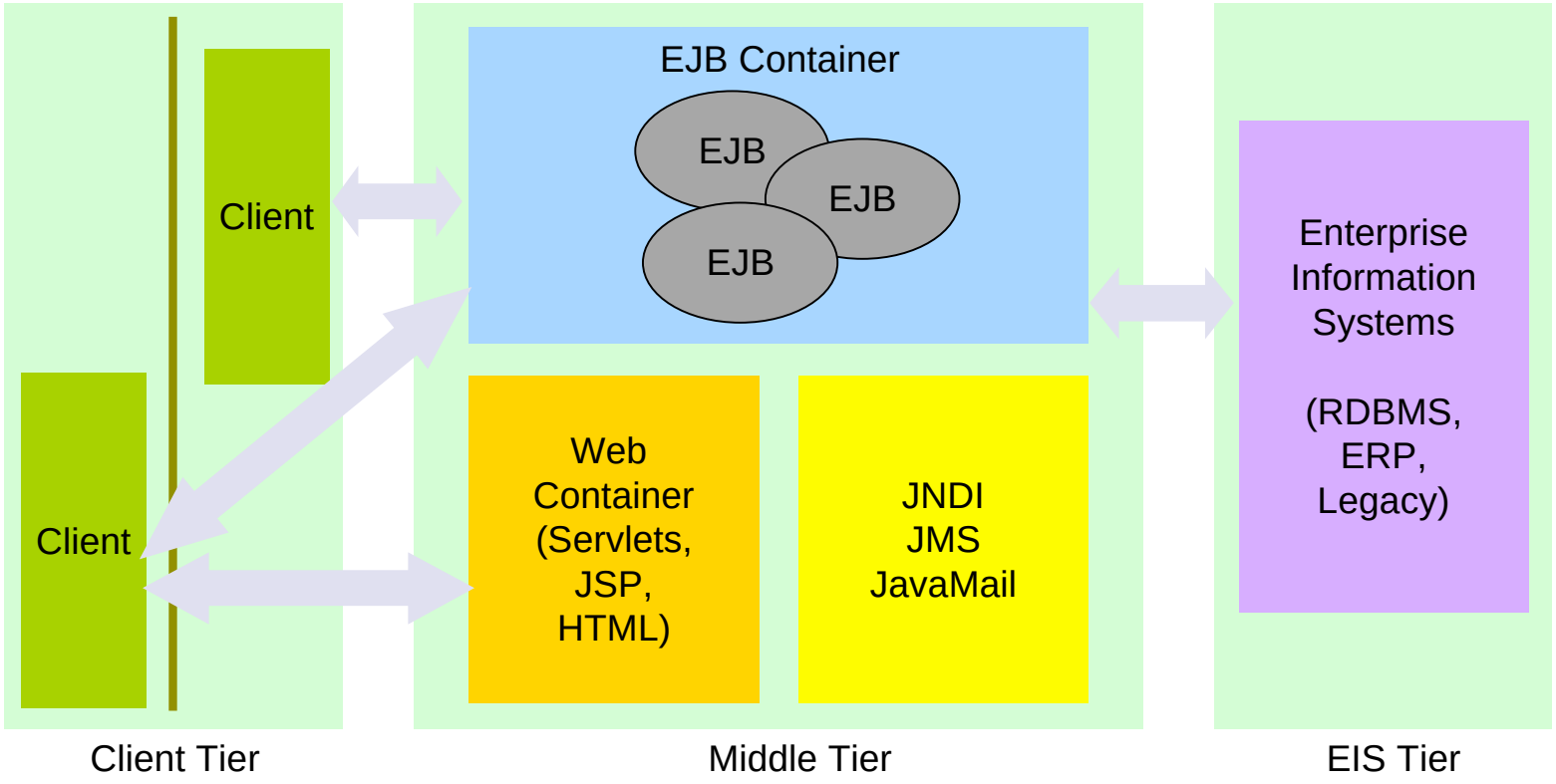


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



J2EE platform

- a Java-based reference platform for developing and integrating enterprise applications
- a collection of standards for enterprise services



MySQL (www.mysql.com)

- world's most popular open source database
  - over 100 million copies of its software downloaded or distributed
- MySQL AB
  - the company founded by the creators of the MySQL
  - set up in Sweden by two Swedes and a Finn

classification of DBMSs

- based on the data model used:
  - traditional: relational, network, hierarchical
  - emerging: object-oriented, object-relational, XML
- other classifications:
  - single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
  - centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)
  - homogeneous vs. heterogeneous (federated)
  - cost of DBMS
  - general purpose vs. special purpose (e.g., OLTP systems)
  - …