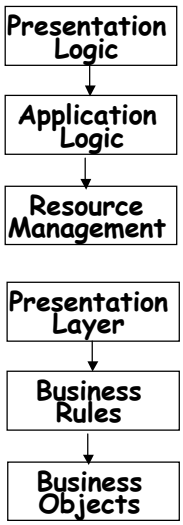
# Distributed Information Systems: Architecture

# Logical Components of Information System




- Logical Components provide vertically layered architecture
  - ☆ Client needs a presentation layer through which it can submit operations and obtain a result
  - ☆ The application logic establishes what operations can be performed over the system and how they take place.
    - enforces the business rules
    - establishes the business processes
  - ☆ The resource management deals with the organization (storage, indexing, and retrieval) of the data necessary to support the application logic.
    - Provides persistence and querying capability

# Presentation and Application

- Presentation Logic
  - Web-Browser
    - Static web-pages
    - Dynamic web-pages
  - Touch screen
  - Specialized Software for banks/travel agencies/CAD
- Application Logic
  - BookFlight / ReserveCar / PayBill
  - Java Classes / Java Beans

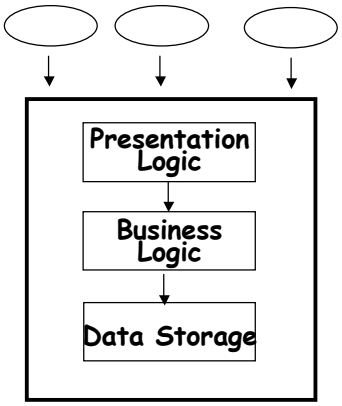


# Resource Manager / Services

- Resource Management
  - Persistent Object Store
  - Relational Database Management System
  - XML data
  - File System
  - Set/get data
- System Services
  - Security: Authentication / Access Control / Encryption
  - Transactions: All-or-Nothing / Isolation / Durability
  - Query optimization and execution
  - Service/Object Location
  - Communication: RMI/multicast/persistent queues
  - ...

# Mainframe

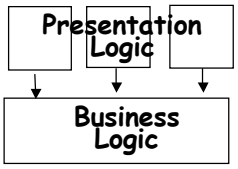
1-tier architecture

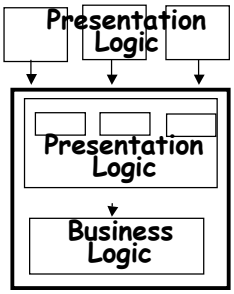



- Clients access the system through display terminals
  - What is displayed and how it appears is controlled by server ("dumb" terminals)
  - Often limited GUI
- Typical architecture of main-frame applications.
- Advantages:
  - Highly optimized
  - Easy to keep data consistent
- Disadvantages
  - Often no conceptual separation of components
    - Little modularity
    - Experts must know all
  - Highly-available mainframe server extremely expensive
- Examples
  - Bank application (terminals with green text-only screens)



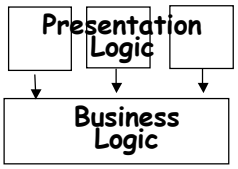
# Separation of presentation logic from other layers

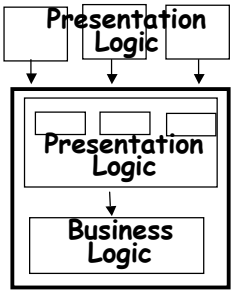



- Move (part of) presentation layer to client
- Modern Bank Software:
  - Presentation Logic at PC/client (implemented in Swing, Applets, etc.)
  - API (application programming interface) allows the presentation logic module to call application logic methods
    - retrieveClientaccountinfobyname(&clientname)
    - retrieveClientaccountinfobyid(&clientid)
- Web-Pages
  - Powerful Web Browser at client
  - Web-page creation or webpage storage at server
  - Presentation Logic split between client and server

# Separation of presentation logic from other layers

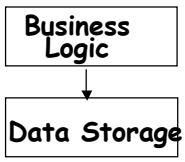


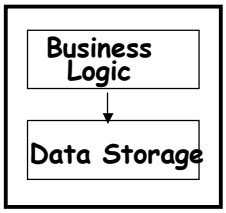




- Advantages:
  - Individual presentation layers for different clients (web-browser, PDA browser, telephone,…)
  - Use computing power at client for sophisticated presentation layer
  - Introduces concept of API (application program interface)
    - Specify an interface (set of objects/methods or functions) that can be called from the outside
  - Provides conceptual separation of presentation and business logic
- Compare with integrated solutions
  - Cgi scripts and servlets creating html AND containing SQL to call dbs



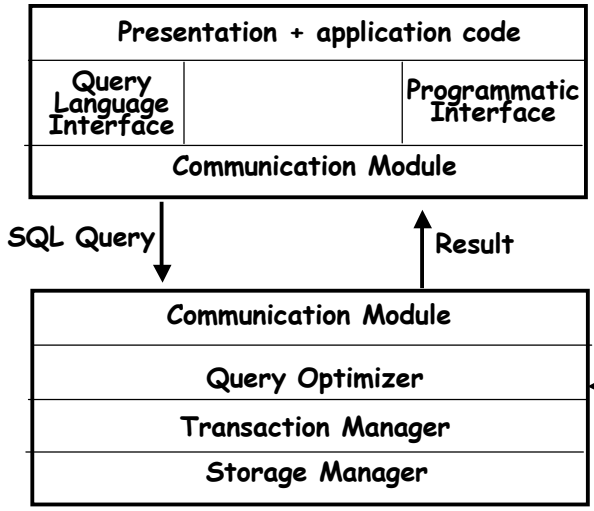
# Separation of application logic from storage management




- Standard
  - Application Logic is implemented in application programs. Access DB through JDBC, ODBC, etc.
  - Application Logic client of DBS
- Performance optimization
  - Stored procedures: move application logic to DBS
- Data Integrity
  - Use triggers

# Location of System Services - Traditional Relational

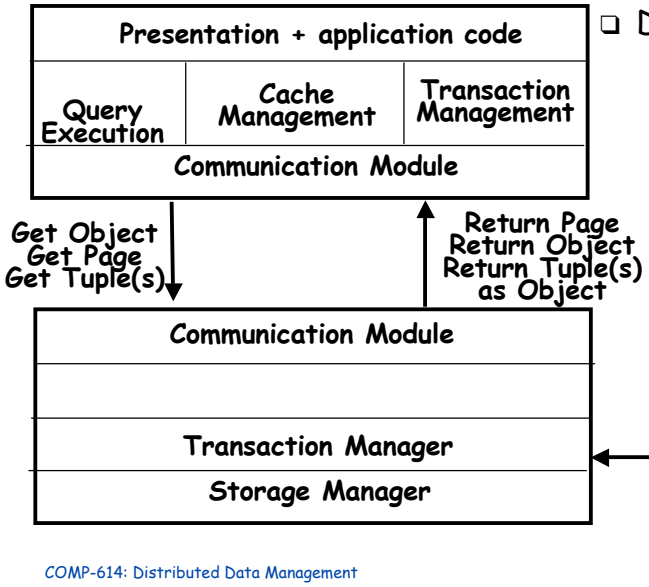



- Function Shipping
  - JDBC
  - C with embedded SQL
  - Phyton
  - 4th Generation Languages (Dbase etc.)



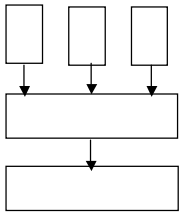


# Migration of System Services -- OO

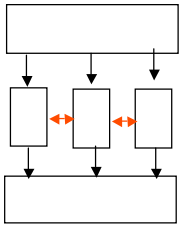



- Data Shipping
  - OO DBS
  - CAD Systems
  - ***Application Servers*** (TP-Monitors)
    - Object layer on top of relational DBS
  - Object relational DBS

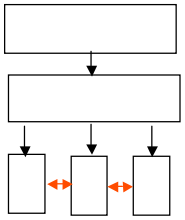
# Horizontal Distribution at different Layers



Supporting multiple client types / Client interfaces



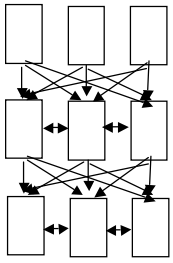
Separating application logic



Data Distribution and replication



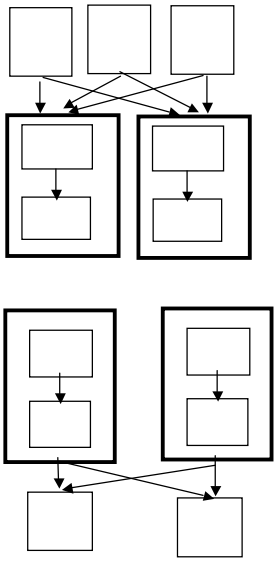
# System Design



- Each box is a part of the system
- The more boxes,
  - the more modular the system
  - better distribution and parallelism
  - Better encapsulation, component based design, reuse
  - Fault-tolerance?
- The more boxes, the more arrows
  - More session and connection maintenance
  - More coordination
  - More complex to monitor and manage
- The more boxes,
  - the more context switches
  - More intermediate steps for each task
  - Performance problems
- Need to balance the advantages and disadvantages of the different architectures

·There is no problem in system design that cannot be solved by adding a level of indirection.
·There is no performance problem that cannot be solved by removing a level of indirection

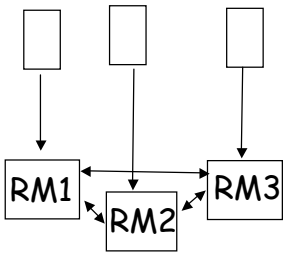
# Accessing more than one resource



- Problems:
  - The resource managers don't know about each other and a potential common business logic
    - Each client has to implement this
  - The resource managers are probably different
    - Each client has to deal with heterogeneous environment
  - Very inefficient and complex

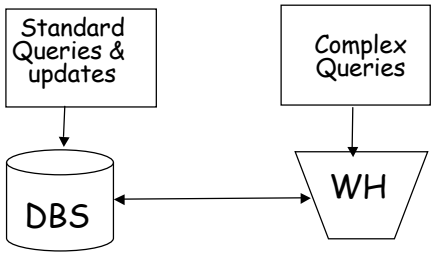


# 1-layer coordination




- ***Parallel database systems / cluster database systems /distributed databases*: e.g., all DBS are part of a single system.**
  - **Client connects to any DBS node**
  - **Transparent forward of remote requests**
  - **Transparent load balancing, distributed query execution**
  - **Oracle**
    - **transparent replication**
    - **Supports distribution**
    - **Provides communication network between different database servers**
    - **Provides parallel database system**

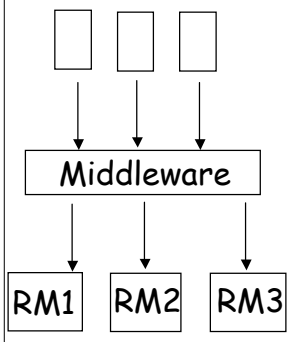
# Data Warehouse




- Data Warehouse collects and **copies** data from data sources; further processing of data for advanced analysis
- Execution and data flow:
  - Updates and standard queries: to local DBS
  - Complex queries: to data warehouse
  - Data warehouse does not forward queries but accesses own copy
  - Changes in data sources via push/pull integrated into data warehouse

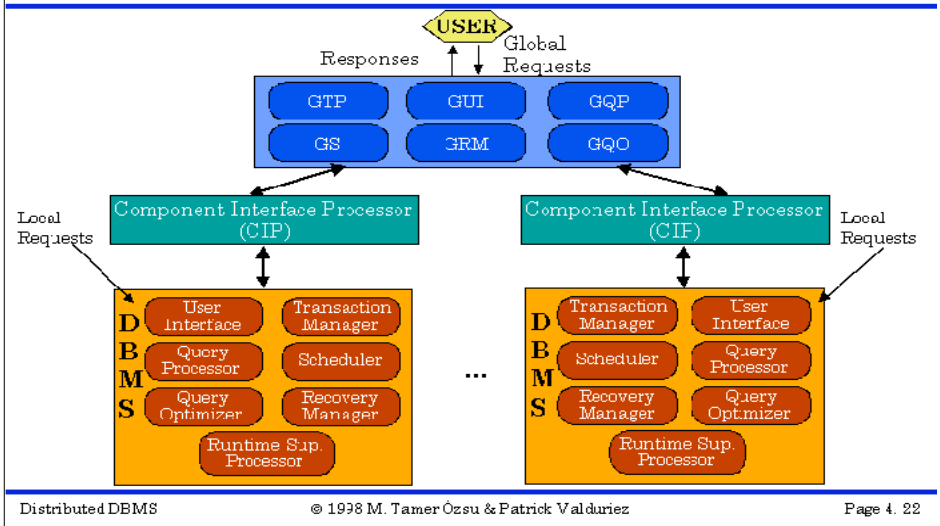


# Middleware approach




- **Middleware is just a level of indirection separating client from several servers**
- **Advantages**
  - **Simplify design of clients by reducing interfaces (only sees middleware)**
  - **Transparent access to underlying systems**
  - **Centralizes control**
  - **Functionality available to all clients**
  - **Is able to handle heterogeneity**
  - **Advanced System Services**
    - **Takes care of locating resources, accessing them, gathering results**
- **Disadvantages**
  - **Another indirection**
  - **Single Point of Failure**

## Components of a Multi-DBMS

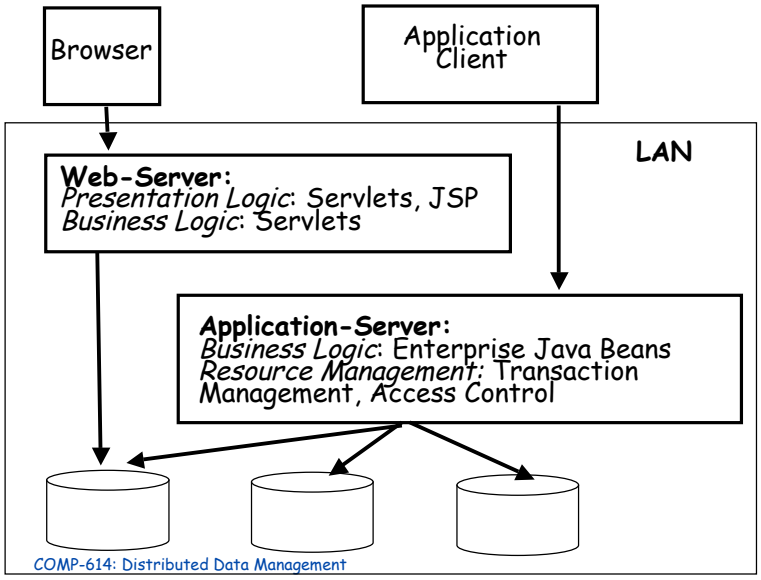





# DBS Middleware

- Tasks
  - ☆ Divides user query into sub-queries to underlying (hetereogeneous) data sources
  - ☆ Collects and results and performs post-processing (additional query processing: necessary for join over tables from different sources)
  - ☆ Distributed transaction management for isolation and atomicity
  - ☆ Contains meta information in order to perform tasks automatically
- Existing systems
  - ☆ Many research prototypes
  - ☆ Commercial systems for heterogeneous query processing OR for distributed transaction processing
  - ☆ J2EE Application Server
    - Restricted support for transaction isolation in case of caching (relies on underlying DBS)
    - Programmer must explicitly state for each DB call which DBS to access
    - No post-processing of queries from different data sources (programmer must do this)

# J2EE

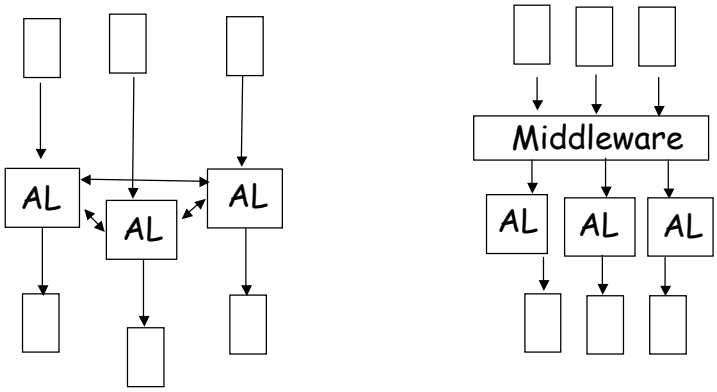





# Application Server Examples (not only J2EE)

- BEA Tuxedo (TP-Monitor)
- BEA Weblogic (J2EE)
- Bluestone Sapphire/Web
- ColdFusion
- Compaq (Tandem) Pathway
- Compaq (DEC) ACMS
- IBM CICS (TP-Monitor)
- IBM IMS/DC (TP-Monitor)
- IBM Websphere (J2EE)
- Iona iPortal App Server
- iPlanet (Sun/Netscape) (J2EE)
- Microsoft COM+ (formerly MS Transaction Server, or MTS)
- Oracle Application Server
- SilverStream
- WebObjects
- And many others. See serverwatch.internet.com

# Distributed Application Logic




- Distribution of system services similar to DBS case