

# Progetto di Applicazioni Software

**Antonella Poggi**

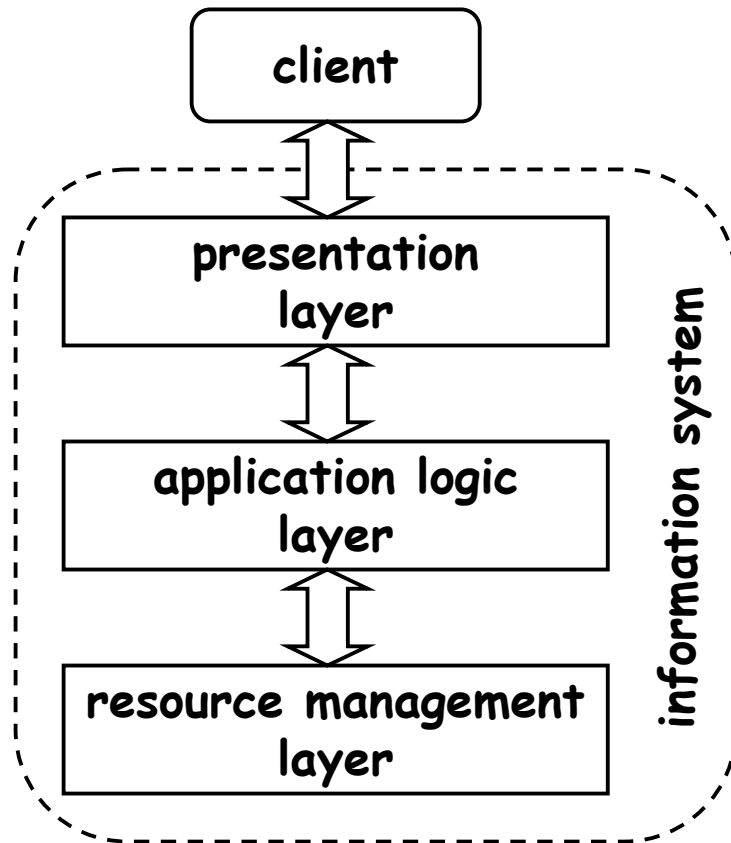
***Dipartimento di Informatica e Sistemistica “Antonio Ruberti”  
SAPIENZA Università di Roma***

Anno Accademico 2008/2009

*Questi lucidi sono stati prodotti sulla base del materiale preparato per il corso di Progetto di Basi di dati da D. Lembo.*

# **1. Architetture dei Sistemi Informativi**

# Layer di un Sistema Informativo



A livello **concettuale** i sistemi informativi sono progettati in termini di tre diversi componenti funzionali (livelli)

1. Presentazione
2. Logica dell'applicazione
3. Gestione delle risorse

# Presentation Layer

- E' il livello del sistema che gestisce la comunicazione con le entità esterne al sistema stesso (client)
- Comprende le **componenti che** si occupano di presentare l'informazione verso i client, e che **consentono ai client di interagire** con il sistema per sottomettere operazioni ed ottenere risultati
- I **client** possono essere **completamente esterni** ed indipendenti dal presentation layer: nei sistemi accessibili tramite web browser che visualizzano pagine HTML, il presentation layer è costituito dal web server e dai moduli che concorrono a creare i documenti HTML (ad es. Java Servlet), mentre il browser è il client
- In altri casi il **client ed il presentation layer possono essere fusi insieme**: spesso esiste un programma che assolve ad entrambi i compiti

# Application Logic Layer

- E' il livello del sistema che si occupa del **processamento dei dati** necessario per produrre i risultati da inoltrare al livello di presentazione
- Un programma che implementa le operazioni legate ad un prelievo su un conto corrente bancario, o la sequenza di passi da compiere per effettuare un acquisto on-line sono esempi di logica applicativa di un sistema
- Il livello della logica applicativa è anche chiamato **processo di business, insieme delle regole di business**, o semplicemente **server** (in questi casi il sottostante livello è rispettivamente chiamato persistence storage, business objects, o database)

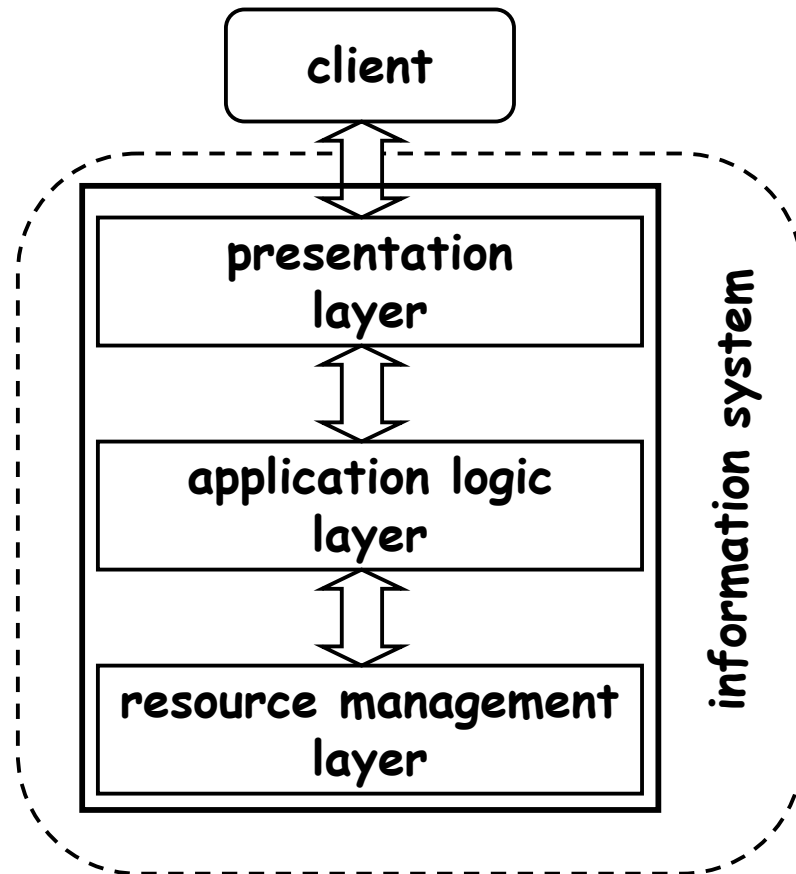
# Resource Management Layer

- E' il livello che **gestisce i dati** che sono necessari al funzionamento dell'intero sistema
- I dati possono risiedere su una base dati, un file system, o altri contenitori di informazioni
- In linea di principio, il livello di gestione delle risorse gestisce le differenti sorgenti di dati che fanno parte del sistema informativo, indipendentemente dalla loro natura
- Nel caso in cui esso è implementato tramite un **DBMS**, è detto semplicemente **data layer**
- Secondo un'accezione più generale, questo può includere qualsiasi sistema in grado di fornire informazione

# Architettura dei sistemi informativi

- Gli strati discussi finora sono costrutti concettuali che separano le funzionalità di un sistema informativo
- Nell'implementazione dei sistemi reali, questi strati possono essere combinati e distribuiti in diversi modi
- Quando consideriamo l'**implementazione** di un sistema informativo, facciamo riferimento ai livelli del sistema con il termine **tier**, piuttosto che conceptual layer
  - Tier = modulo software che implementa uno o più conceptual layer
- Due tier per essere distinti non devono necessariamente risiedere in due macchine distinte!

# Architettura One-tier



L'architettura One-tier combina tutti i livelli concettuali in un un unico tier

Rispecchia l'architettura hw basata su **mainframe** tipica dei primi calcolatori elettronici

Gli utenti accedevano tramite **terminali non intelligenti** (in genere schermo e tastiera)

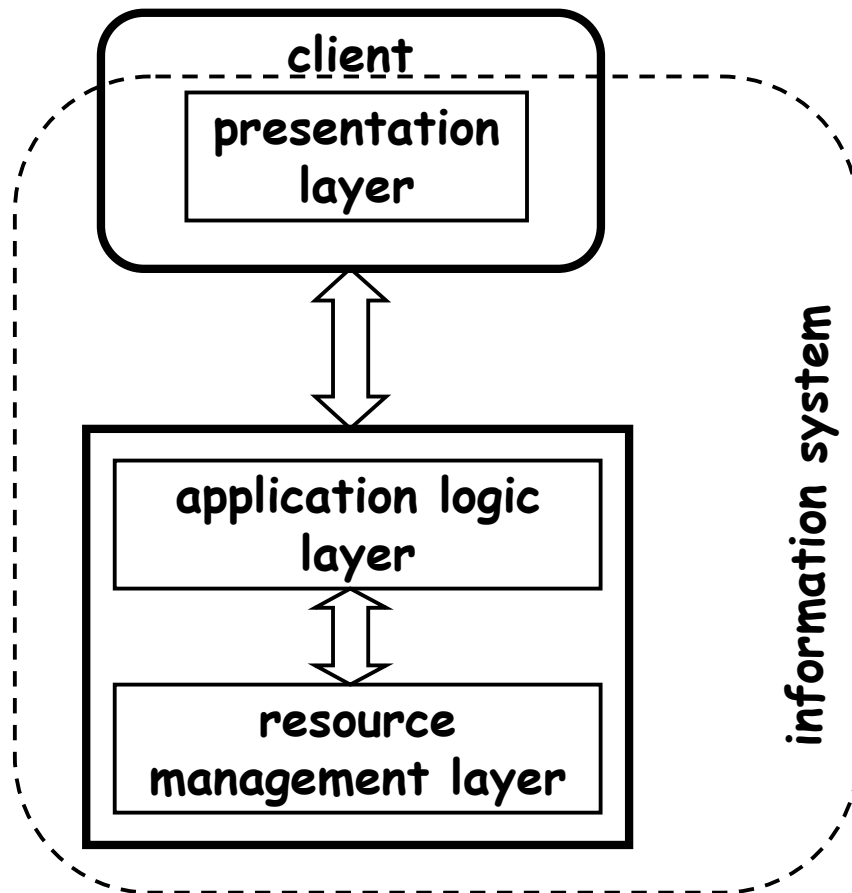
# Architettura One-tier – svantaggi

- I tre livelli sono gestiti da **un unico modulo**, che risiede quindi in un'unica macchina
  - L'intero livello di presentazione risiede sulla stessa macchina, che si prende carico di controllare **ogni aspetto dell'interazione** con il client
  - Il calcolo centralizzato delle **visualizzazioni grafiche** delle interfacce
    - richiede **molta potenza di calcolo**
    - supporta un **basso numero di utenti**
- Non vengono definite application program interface (API) che possano facilitare l'interazione con altri sistemi
  - **poca portabilità** del sistema
  - **difficile da mantenere**
  - **difficile da aggiornare**

# Architettura One-tier – aspetti positivi

- Tutto risiede in un'unica macchina
  - Non ci sono costi di comunicazione
- Client non intelligenti
  - Bassi costi di impiego
- Il progettista è libero di fondere i livelli concettuali
  - aumenta l'efficienza del sistema
- Nessuna interfaccia
  - nessuna complessità dovuta alla pubblicazione ed al mantenimento di interfacce, o a compatibilità fra diversi componenti

# Architettura two-tier



- Il sistema informativo è partizionato tra due tier: I client hanno la possibilità di **processare ulteriormente** l'informazione fornita dal server
- Si è affermata con il diffondersi di PCs e workstation e con lo sviluppo di nuove tecniche software per i sistemi distribuiti (ad es., RPC)
- Il tier nel client
  - gestisce il livello di presentazione
  - invoca le funzionalità dell'applicazione
- I client si dividono in Thin client (con soli compiti di presentazione) e Thick client (inglobano parte della logica dell'applicazione)

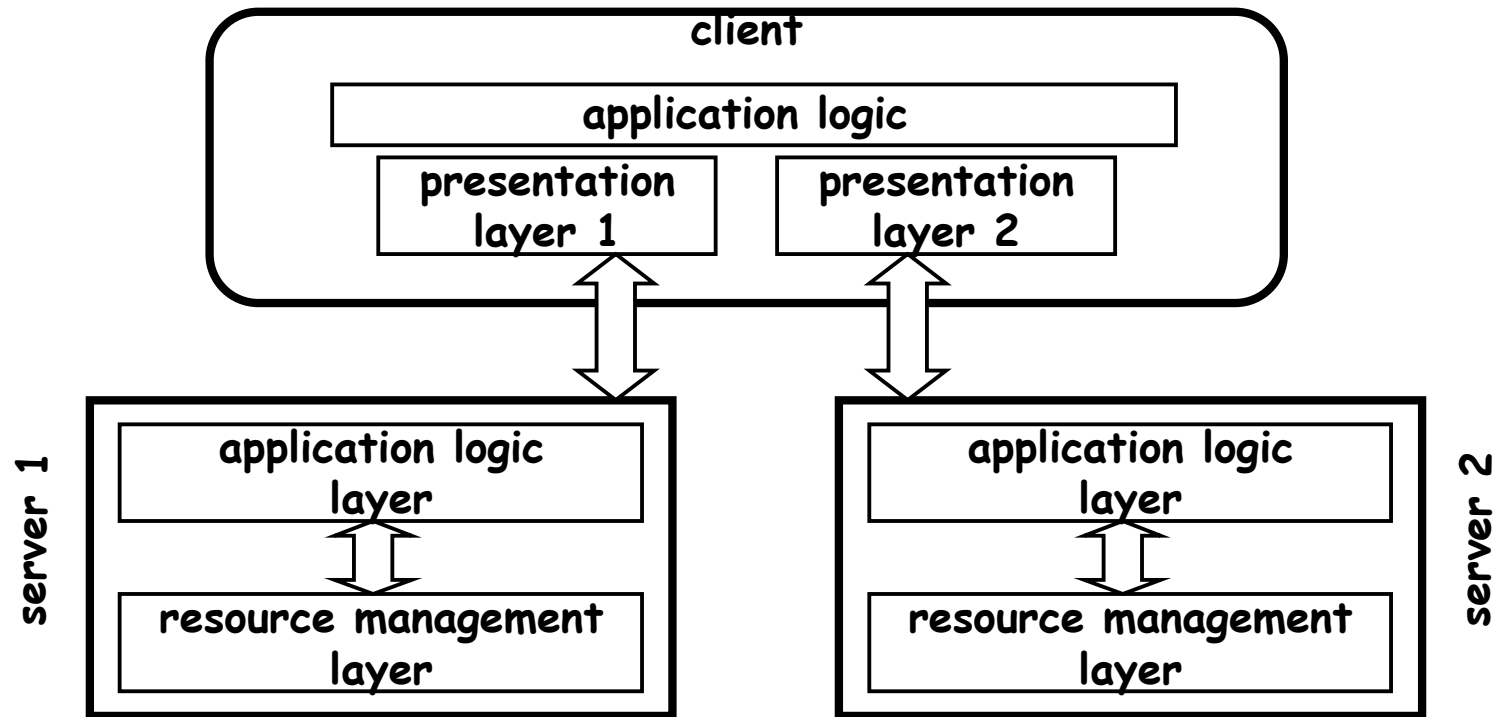
# Architettura two-tier – vantaggi

- Permette di realizzare un'architettura client/server
- Viene introdotto il concetto di **API** che facilita la comunicazione del client con l'applicazione
- Al tempo stesso, le architetture two-tier garantiscono **maggiore portabilità** rispetto all'architettura one-tier
- La capacità di calcolo del client consente di avere **interfacce grafiche sofisticate** ed allo stesso tempo risparmio di risorse sul server
- La possibilità di combinare sul server i livelli di logica dell'applicazione e di gestione delle risorse consente di mantenere una certa **efficienza**

## Architettura two-tier – svantaggi

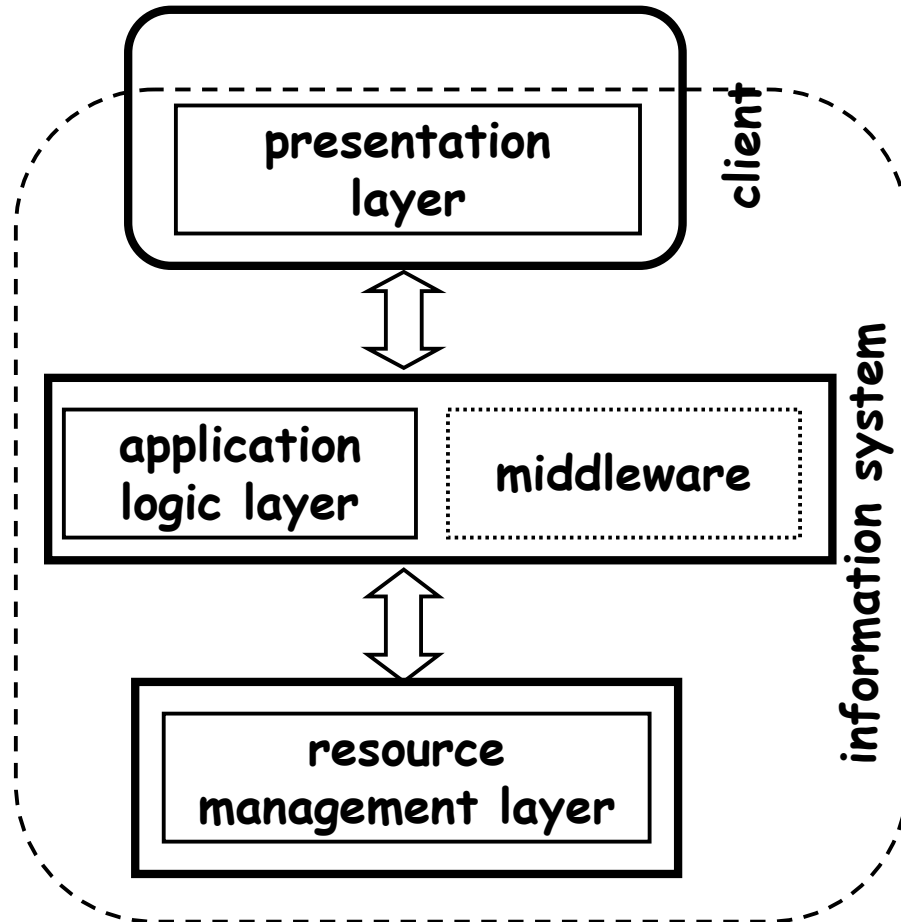
- La contemporanea presenza sul server della logica dell'applicazione e della gestione delle risorse richiede server dalle prestazioni abbastanza elevate
- Supportano un limitato numero di client, a causa della necessità di mantenere informazioni sulla connessione e sull'autenticazione dei client
- I **client** si sono evoluti indipendentemente dai server ed hanno presentato nel tempo funzionalità sempre più avanzate. In particolare, hanno cercato di **integrare più server**, ma con l'architettura sbagliata!

# Architettura two-tier ed Integrazione



L'interazione di uno stesso client con più livelli di gestione di risorse ha fatto nascere l'esigenza di avere client via via più potenti, oltre che uno **strato di logica applicativa per gestire l'integrazione delle risorse sul client** stesso  
→ Non scalabile!

# Architettura three-tier



- I livelli concettuali sono completamente **disaccoppiati**
- Risponde all'esigenza di scalabilità
  - integrare più server
  - gestire più utenti
- Si avvale della presenza di **stabili interfacce** sia per il livello della logica dell'applicazione che il livello della gestione delle risorse
  - Standard Application Program Interface (API) usate per l'interazione tra il livello della logica dell'applicazione e il livello di gestione delle risorse
- Si può avvalere della funzionalità offerte da un **middleware**
  - integrazione
  - replicazione

# Il middleware

- Il middleware è l'insieme delle **astrazioni di programmazione** e delle **infrastrutture** che supportano lo sviluppo della logica dell'applicazione
- Come astrazione di programmazione:
  - **Nasconde i dettagli** dell'hardware, della rete e della distribuzione della computazione
  - Presenta sempre più **potenti primitive** che non cambiano concetti di base di RPC ma **aumentano la flessibilità** nel loro uso
  - La sua evoluzione è influenzata dalle tendenze nei linguaggi di programmazione (RPC e linguaggio C, CORBA e linguaggio C++, RMI e linguaggio Java, SOAP-XML e Web services)
- Come infrastruttura:
  - Fornisce una **piattaforma per lo sviluppo e l'esecuzione** di applicazioni complesse
  - Presenta **interfacce** sempre più **standardizzate**
  - Si evolve verso **un'architettura orientata ai servizi**
  - L'obiettivo è l'**integrazione delle piattaforme** e la flessibilità nella configurazione

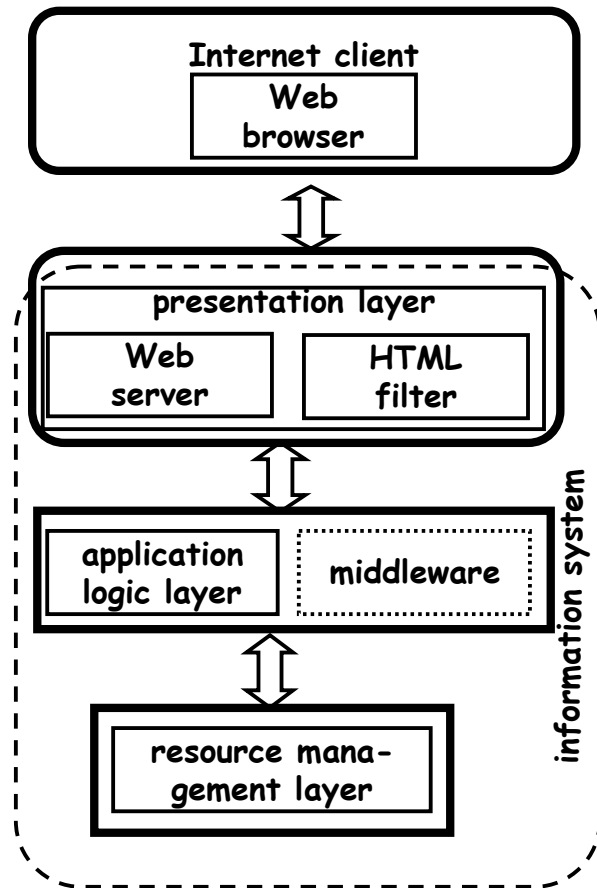
# Architettura three-tier - vantaggi

- I client sono indipendenti gli uni dagli altri: si possono avere **diversi livelli di presentazione** sulla base delle esigenze di diversi client
- Consente l'**integrazione di sistemi** differenti all'interno di una LAN
- Tutta la logica dell'applicazione risiede nello strato intermedio, garantendo per l'intero sistema:
  - **maggiore portabilità**
  - **manutenzione** più semplice
  - **aggiornamento** di uno qualsiasi dei due tier più semplice
  - maggiore **flessibilità**
  - maggiore **scalabilità**
- Permette di realizzare un'architettura client/server
- Il livello della **logica dell'applicazione** può essere distribuito **su diversi server**

# Architettura three-tier – svantaggi

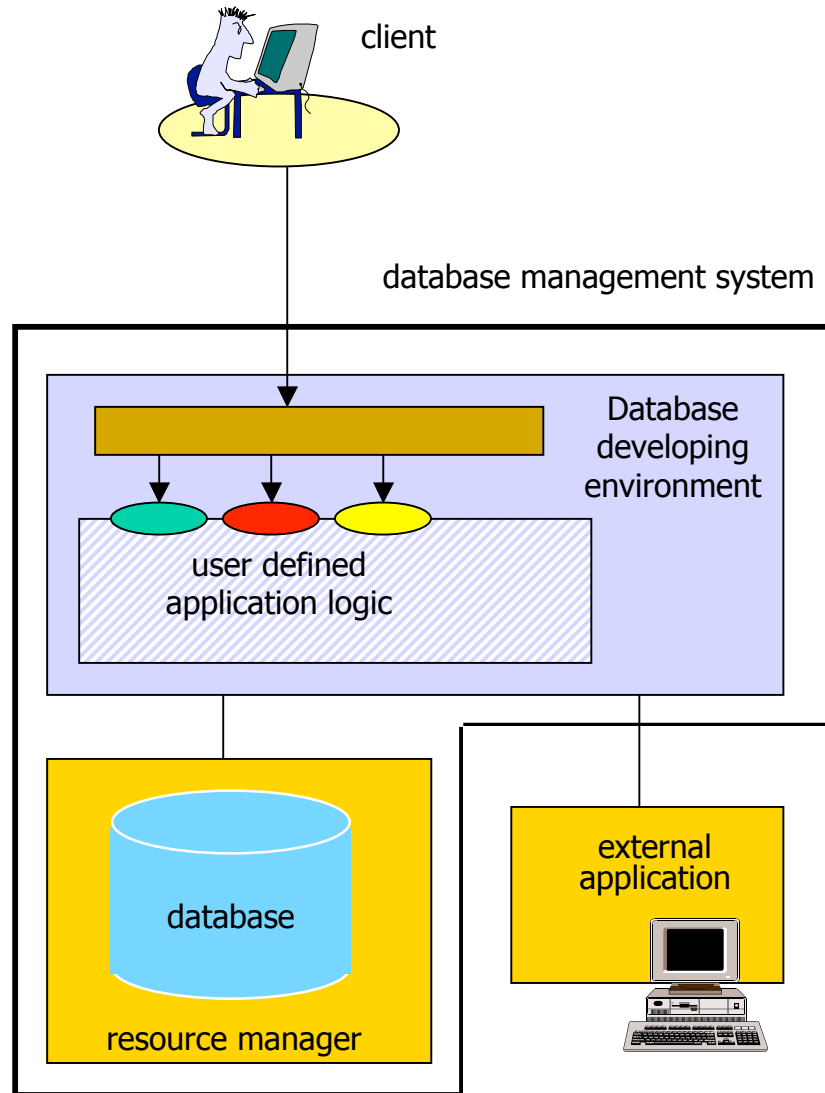
- Lo svantaggio principale è nel decadimento delle performance dovute ai **problemi di comunicazione** fra i diversi nodi del sistema
- Costo aggiuntivo per implementare le interfacce standard
- Per integrare sistemi su **internet** devono fare uso di un componente aggiuntivo: il **web server**

## Da three-tier a N-tier: il caso di Internet



Il livello di presentazione è più complesso: comprende un Web Server

# DBMS per la logica dell'applicazione



- DBMS sono usati tradizionalmente per gestire i dati
  - Forniscono molti strumenti per realizzare parte della logica dell'applicazione
    - triggers
    - replicazione
    - stored procedures
    - queuing systems
- funzioni estremamente efficienti
- problemi: scalabilità, modularità, manutenzione e aggiornamento

# In questo corso (1)

- Vogliamo realizzare semplici sistemi three-tier, in cui i livelli concettuali di **presentazione**, **logica dell'applicazione** e **gestione delle risorse** siano **disaccoppiati**
- Ci rifacciamo alle architetture **client/server** (anche quando il nostro sistema risiede su una singola macchina)
- Vogliamo che la **logica dell'applicazione non risieda completamente nel DBMS**
- Ci concentriamo sul livello della logica dell'applicazione, sul data layer e sulla **interazione** fra i due (ad es., JDBC)
- Il livello di presentazione deve essere separato, ma non richiediamo sia particolarmente sofisticato
- Non facciamo necessariamente uso di un middleware

## In questo corso (2)

