Web Database Application

CIT414 (2 Units)

Dr. Bilkisu Muhammad-Bello

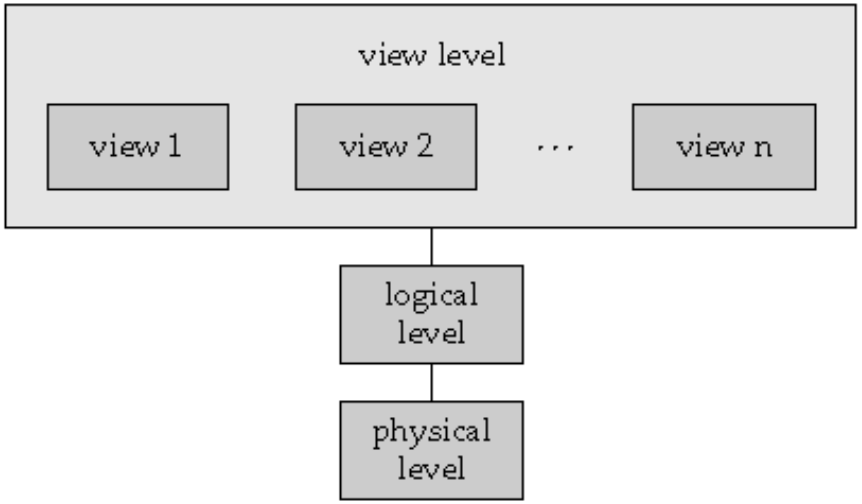
Objectives

At the end of this unit, you should be able to:

- Describe Data Independence in Relational Databases
- Distinguish between Logical and Physical Data Independence.
- Identify the components of N-tier architectures
- Describe 1-Tier Architecture
- Describe 2-Tier Architecture
- Describe a typical 3-Tier Architecture
- Describe the 3-Tier Architecture for Web Apps and the advantages of the 3-Tier Architecture

Overview

- Data Independence in Relational Databases

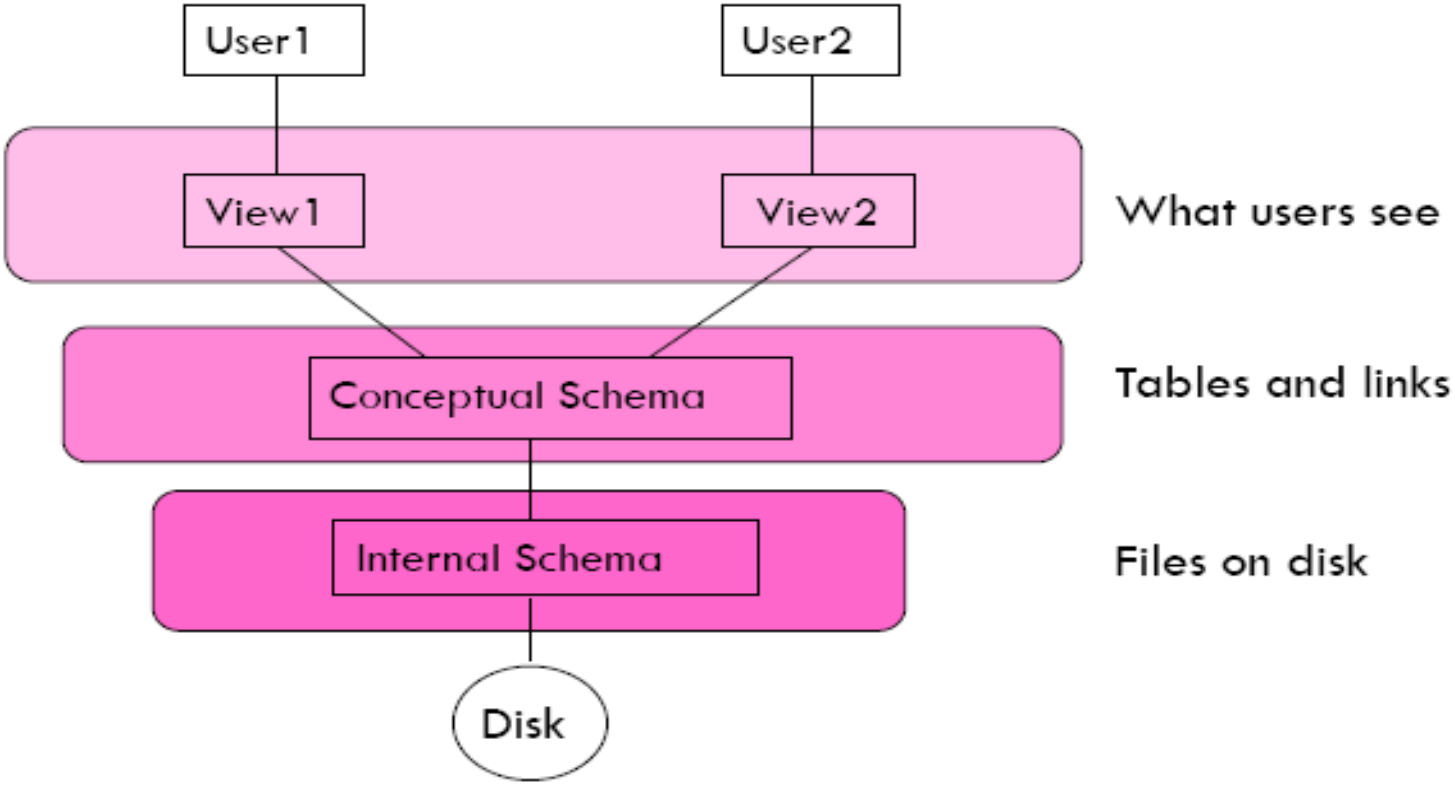


- N-tier Architectures

Web Three-Tier Architectures

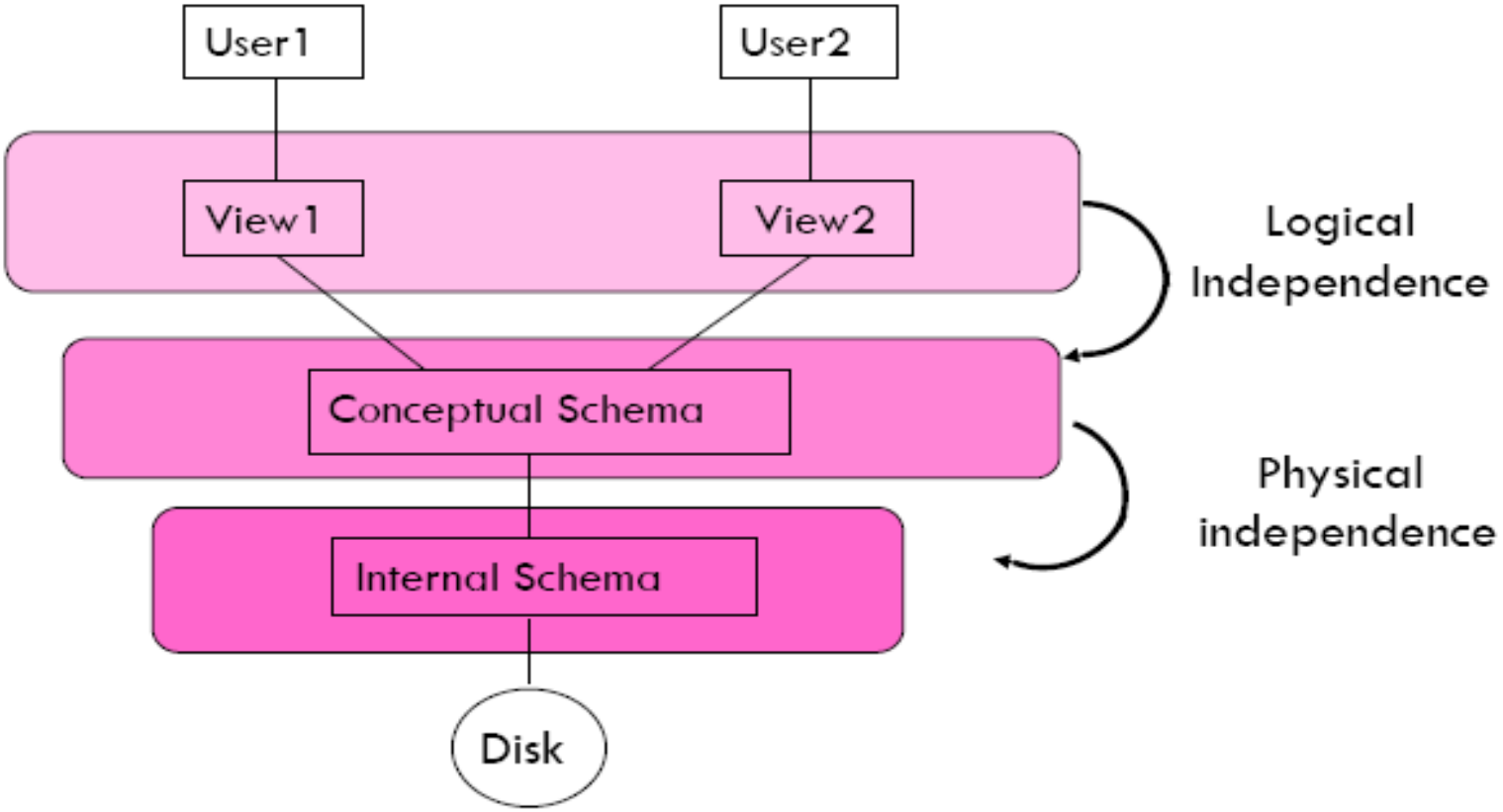
- The Client Tier
- The Middle Tier
- The Database Tier

Database Architecture With Views



Each level is independent of the levels below

Logical and Physical Independence



Each level is independent of the levels below

Data Independence

- **Logical Independence**: The ability to change the logical schema without changing the external schema or application programs
  - Can add new fields, new tables without changing views
  - Can change structure of tables without changing view

- **Physical Independence**: The ability to change the physical schema without changing the logical schema
  - Storage space can change
  - Type of some data can change for reasons of optimization

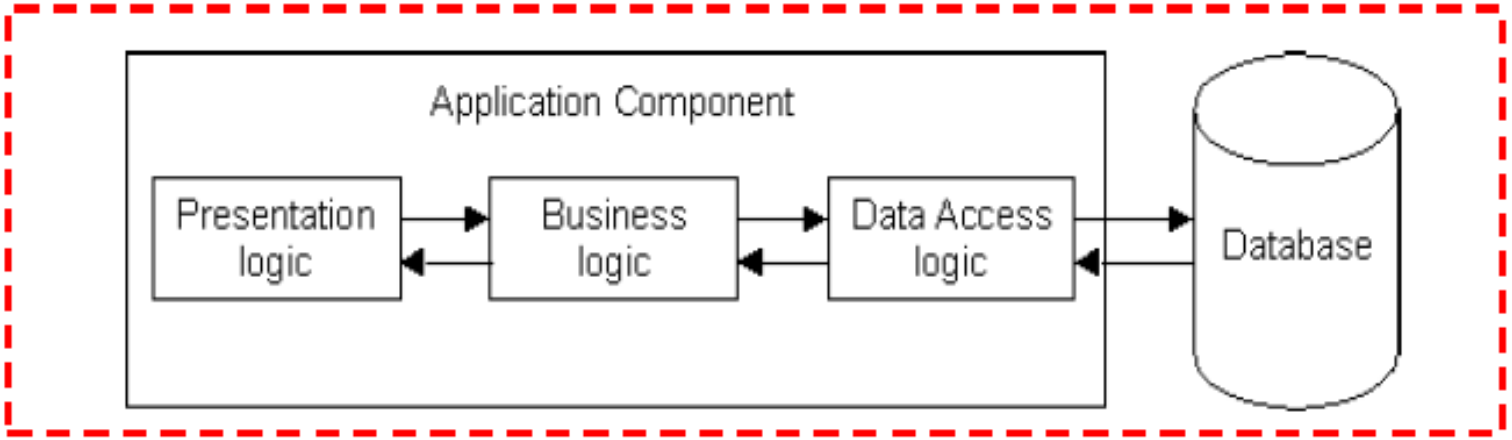
LESSON: Keep the VIEW (what the user sees) independent of the MODEL (domain knowledge)

Significance of "Tiers"

- N-tier architectures have the same components
  - Presentation
  - Business/Logic
  - Data

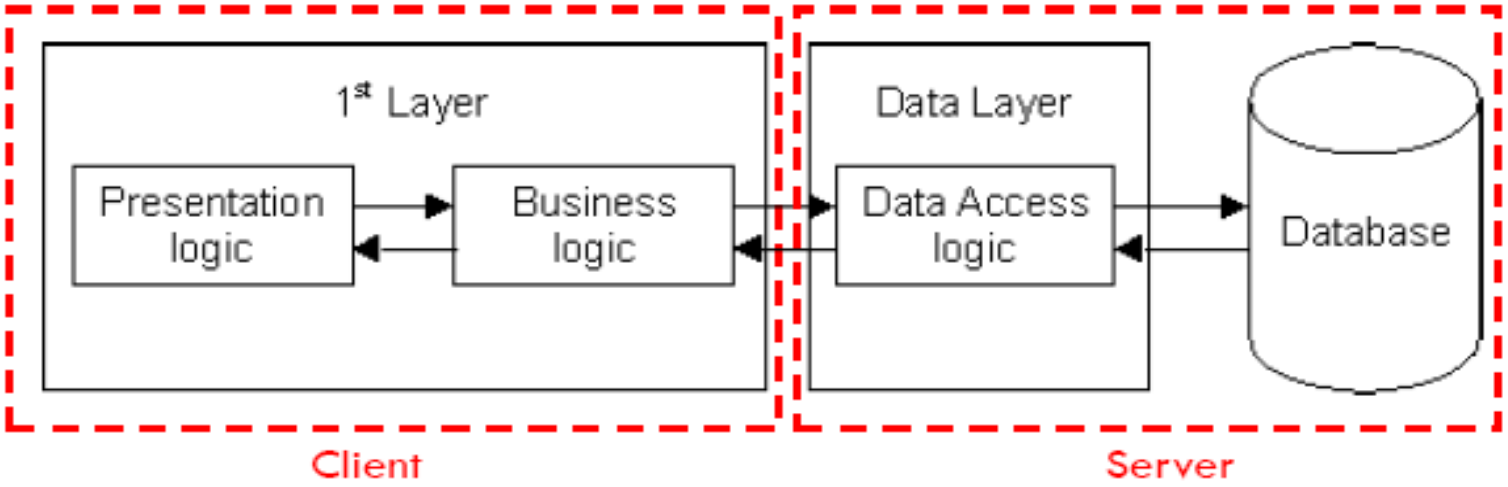
- N-tier architectures try to separate the components into different tiers/layers
  - Tier: physical separation
  - Layer: logical separation

1-Tier Architecture



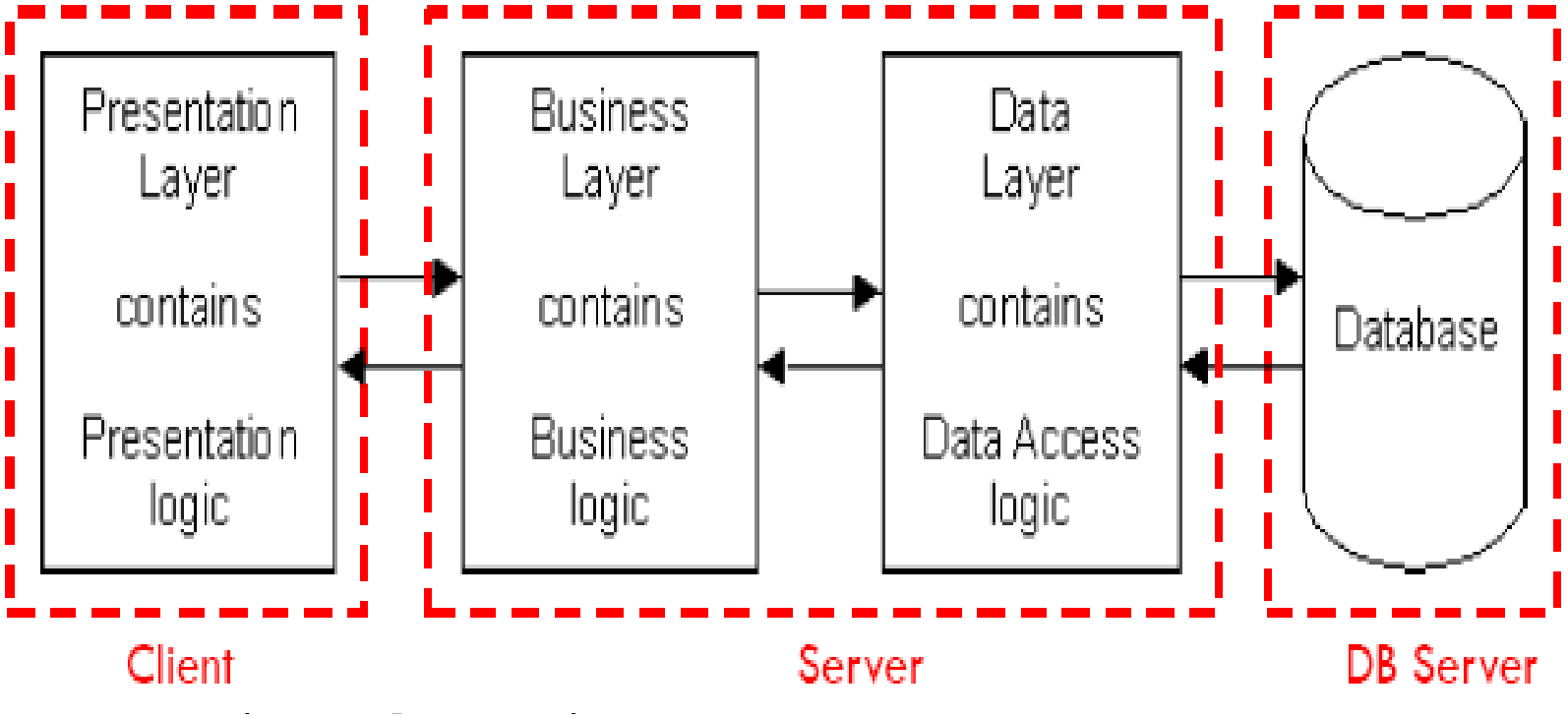
- All 3 layers are on the same machine
  - All code and processing kept on a single machine
- Presentation, Logic, Data layers are tightly connected
  - Scalability: Single processor means hard to increase volume of processing
  - Portability: Moving to a new machine may mean rewriting everything
  - Maintenance: Changing one layer requires changing other layers

2-Tier Architecture



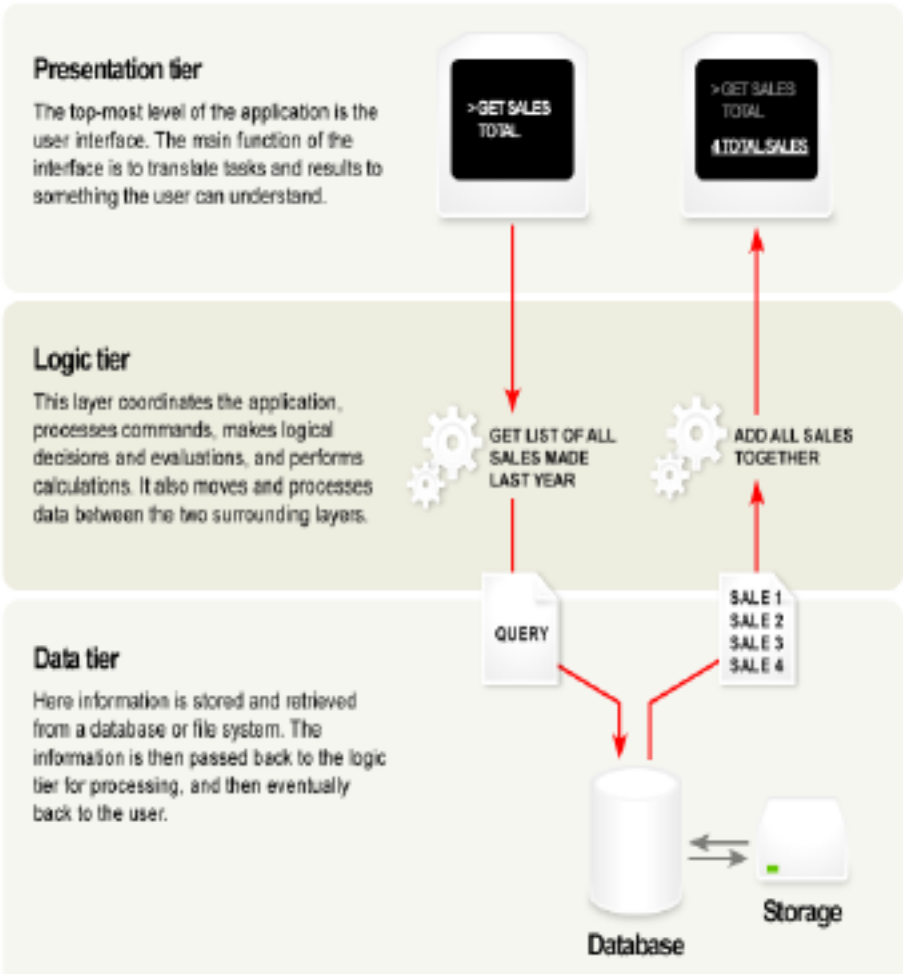
- Database runs on Server
  - Separated from client
  - Easy to switch to a different database
- Presentation and logic layers still tightly connected
  - Heavy load on server
  - congestion on network
  - Presentation still tied to business logic

3-Tier Architecture



- Presentation, logic, data layers disconnected

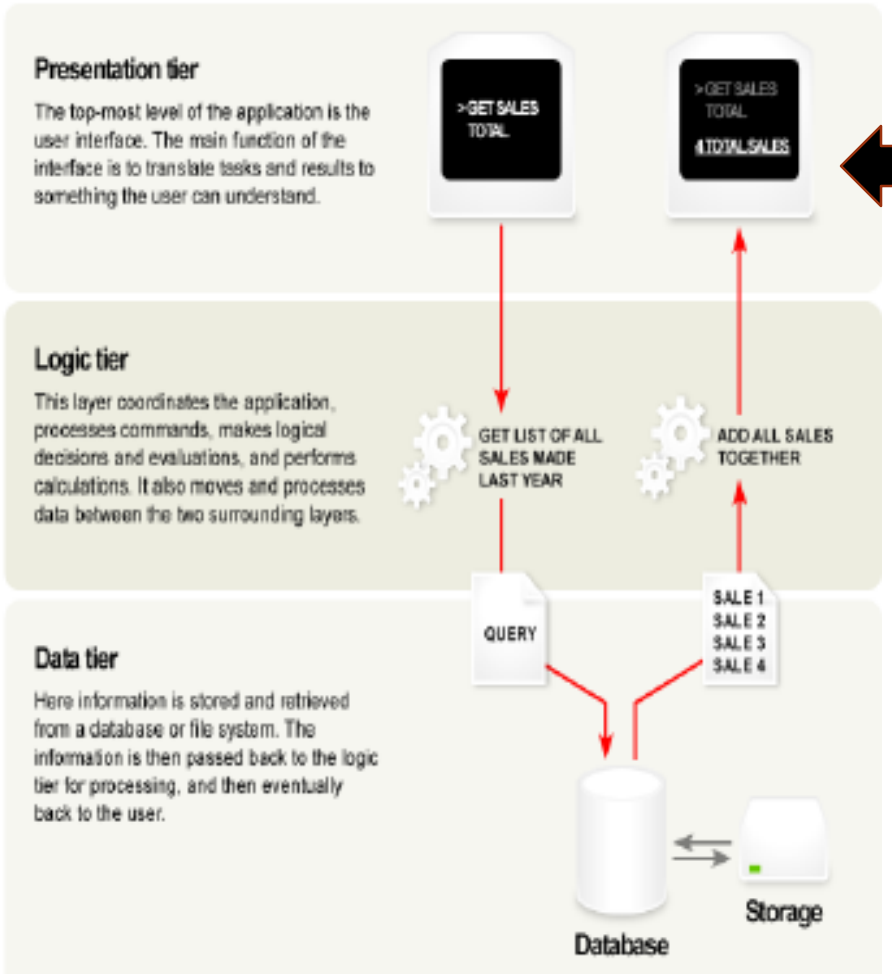
A Typical 3-tier Architecture



Architecture Principles

- Client-server architecture
- Each tier (Presentation, Logic, Data) should be independent and should not expose dependencies related to the implementation
- Unconnected tiers should not communicate
- Change in platform affects only the layer running on that particular platform

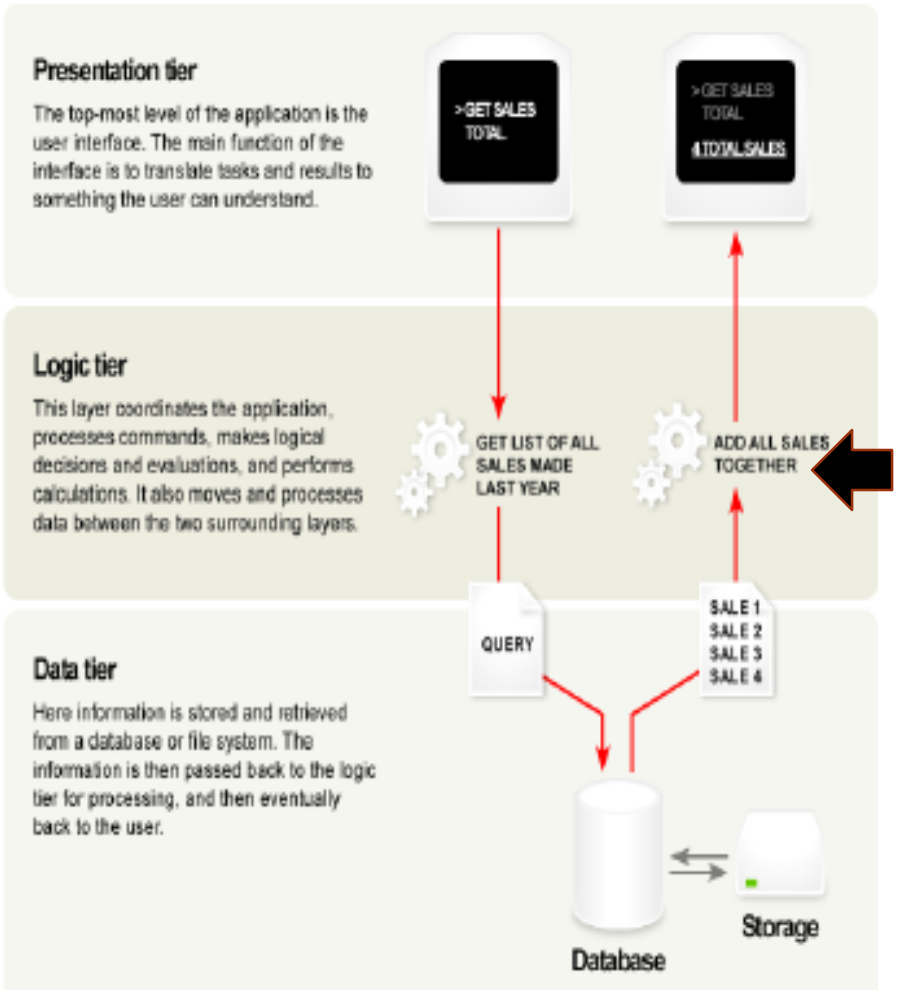
A Typical 3-tier Architecture



Presentation Layer

- Provides user interface
- Handles the interaction with the user
- Sometimes called the GUI or client view or front-end
- Should not contain business logic or data access code

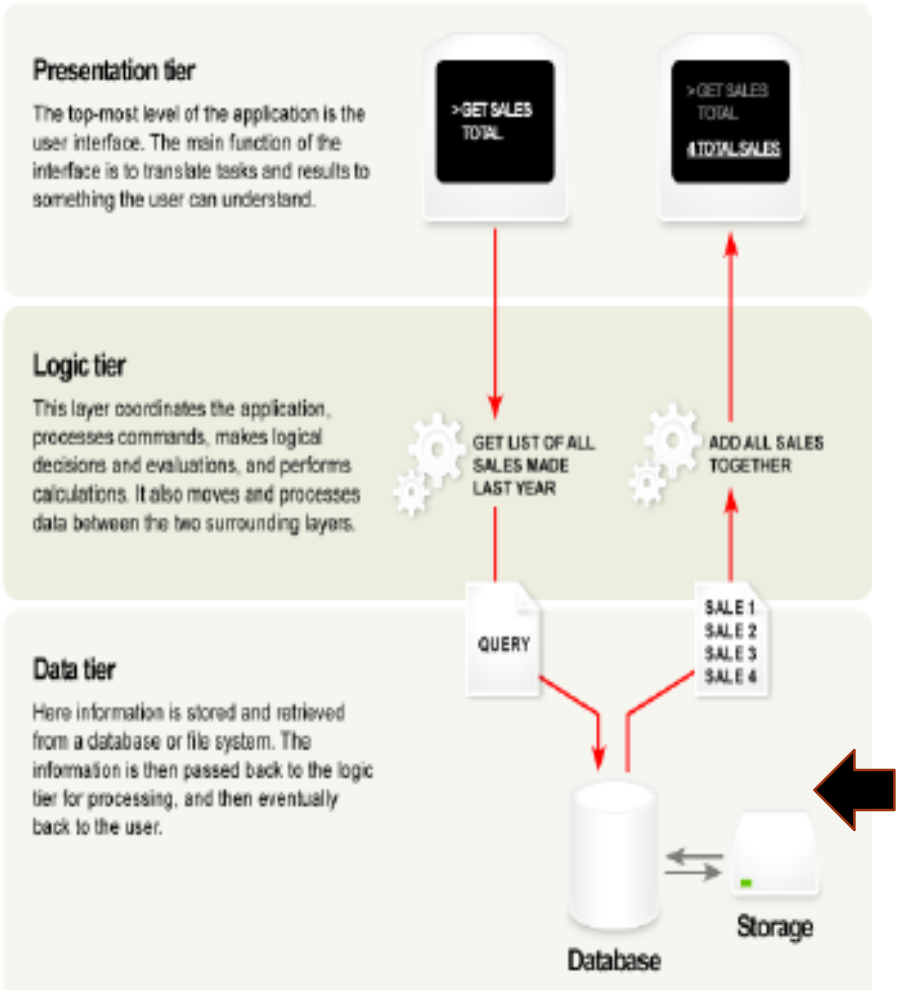
A Typical 3-tier Architecture



Logic Layer

- The set of rules for processing information
- Can accommodate many users
- Sometimes called middleware/ back-end
- Should not contain presentation or data access code

A Typical 3-tier Architecture



Data Layer

- The physical storage layer for data persistence
- Manages access to DB or file system
- Sometimes called back-end
- Should not contain presentation or business logic code

The 3-Tier Architecture for Web Apps

- **Presentation Layer**

  Static or dynamically generated content rendered by the browser (front-end)

- **Logic Layer**

  A dynamic content processing and generation level application server, e.g., Java EE, ASP.NET, PHP, ColdFusion platform (middleware)

- **Data Layer**

  A database, comprising both data sets and the database management system or RDBMS software that manages and provides access to the data (back-end)

3-Tier Architecture - Advantages

- Independence of Layers
  - Easier to maintain
  - Components are reusable
  - Faster development (division of work)
    - Web designer does presentation
    - Software engineer does logic
    - DB admin does data model

Design Problems & Decisions

- Construction and testing
  - how do we build a web application?
  - what technology should we choose?
- Re-use
  - can we use standard components?
- Scalability
  - how will our web application cope with large numbers of requests?
- Security
  - how do we protect against attack, viruses, malicious data access, denial of service?
- Different data views
  - user types, individual accounts, data protection

What You Should Already Know

- HTML
- CSS
- Javascript
- SQL
- Error debugging
- Understanding of generic program building blocks
- Conditional/Selection statements
- Looping/Iteration
- Variable declaration
- Array

Required Tools

- **Web browser(s)**
- **Text Editor/PHP script Editor e.g Notepad, Notepad ++, Atom text editor, Zend Studio etc**
- **Web application server (Apache, Microsoft IIS)**
- **PHP toolkit. Download php from php.net**
- **Database server (MySQL, Microsoft SQL server etc)**

- **WAMPServer**
  - WampServer is a Web development platform on Windows that allows you to create dynamic Web applications with Apache2, PHP, MySQL and MariaDB.
  - WampServer automatically installs everything you need to intuitively develop Web applications. You will be able to tune your server without even touching its setting files.

Recommended Texts

1. MySQL/PHP Database Applications

   By Brad Bulfer, Jay Greenspan & David Wall

2. PHP & MySQL Web Development

   By Luke Welling & Laura Thompson

3. Sams Teach Yourself PHP, MySQL® and Apache All in One *By Julie C. Meloni*

4. PHP5 and MySQL® Bible

   By Tim Converse and Joyce Park with Clark Morgan

- **Online Tutorial:**
  - **http://www.w3schools.com**