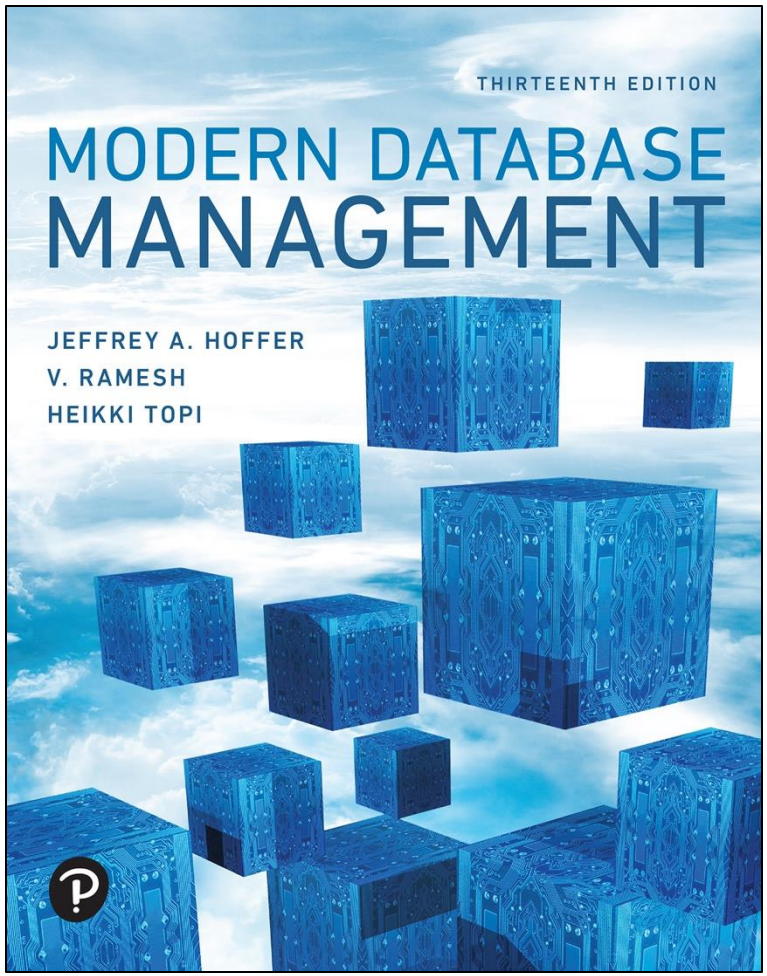
Modern Database Management

Thirteenth Edition



Chapter 7

Databases in Applications

Learning Objectives

7.1 Define terms

7.2 Explain three components of client/server systems: presentation, processing, and storage

7.3 Distinguish between two-tier and three-tier architectures

7.4 Describe key components and information flow in Web applications

7.5 Describe how to connect to databases in 3-tier applications using Java (JSP) and Python

7.6 Understand the notion of transaction integrity

7.7 Compare optimistic and pessimistic concurrency control

7.8 Understand the basics of application security

Client/Server Architectures

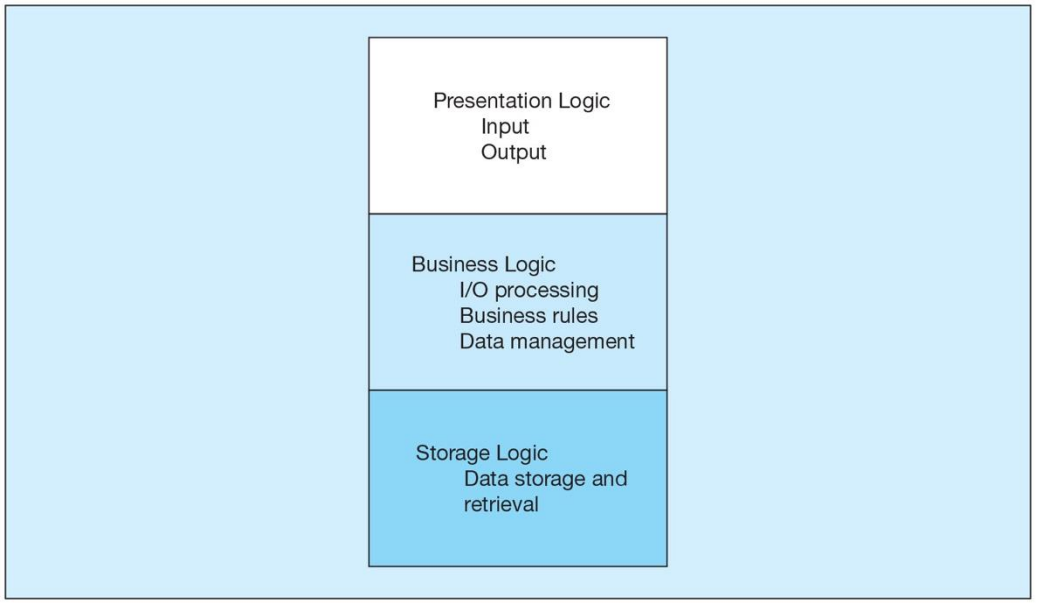
- Networked computing model
- Processes distributed between clients and servers
- Client – Workstation (PC, smartphone, tablet) that requests and uses a service
- Server – Powerful computer (PC/mini/mainframe) that provides a service
- For DBMS, server is a database server
- For the Internet, server is a Web server

Figure 7-1 Application Logic Components

GUI interface (e.g. through a browser)

Procedures, functions, programs

DBMS activities



Application Partitioning

- Placing portions of the application code in different locations (client vs server) after it is written
- Advantages
  - Improved performance
  - Improved interoperability
  - Balanced workloads

Fat vs Thin Clients

- Fat client – a client PC that is responsible for processing presentation logic, extensive application and business rules logic, and many DBMS functions
- Thin client – an application where the client (PC) accessing the application primarily provides the user interfaces and some application processing, usually with no or limited local data storage

Figure 7-2 Common Logic Distributions (1 of 2)

(a) Two-tier client-server environments

Processing logic could be at client (fat client), server (thin client), or both (distributed environment).

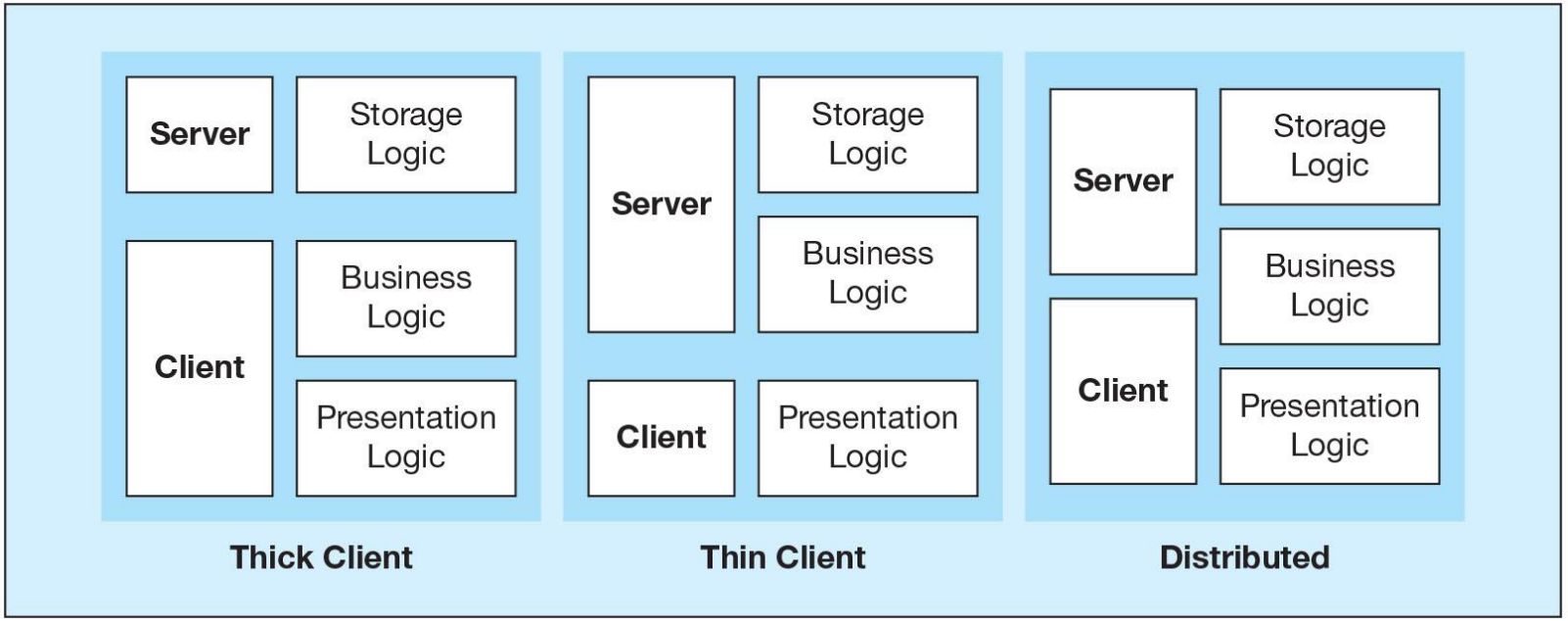
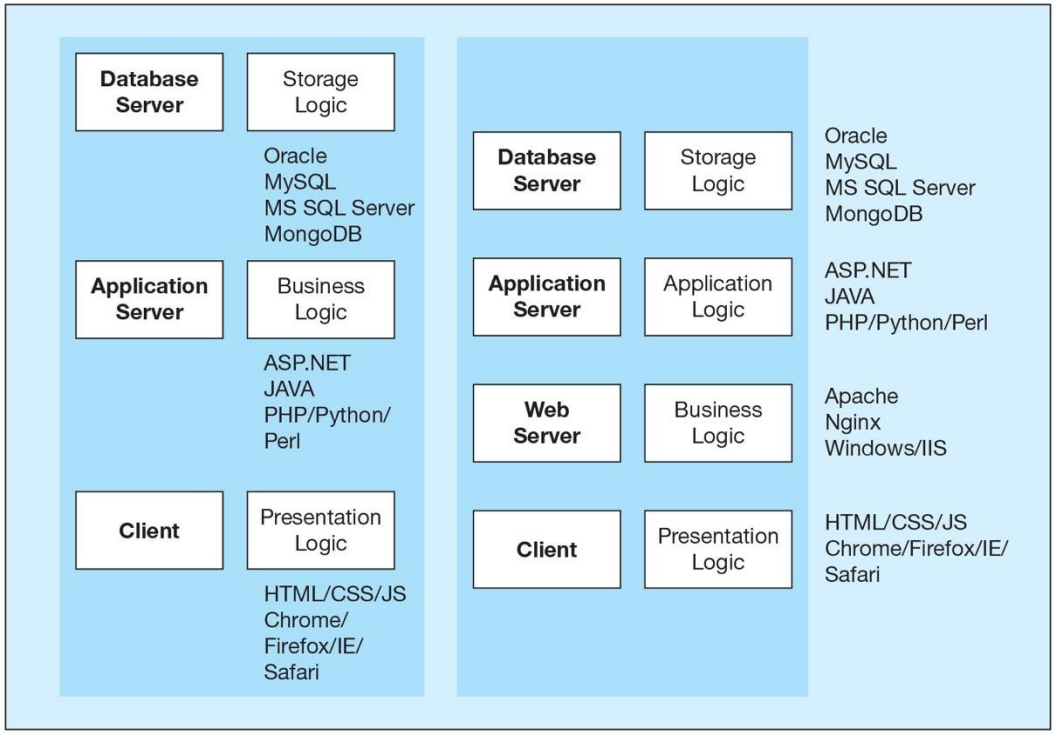


Figure 7-2 Common Logic Distributions (2 of 2)

(b) *n*-tier client/server environments

Processing logic will be at application server or Web server.



Web Application Components

- Database server – hosts the DBMS
  - e.g., Oracle, SQL Server, Informix, MS Access, MySql
- Web server – receives and responds to browser requests using HTTP protocol
  - e.g., Apache, Internet Information Services (IIS)
- Application server – software building blocks for creating dynamic Web sites
  - e.g., MS ASP.NET framework, Java EE, PHP
- Web browser – client program that sends Web requests and receives Web pages
  - e.g., Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome

Figure 7-3 A Database-Enabled Intranet/Internet Environment

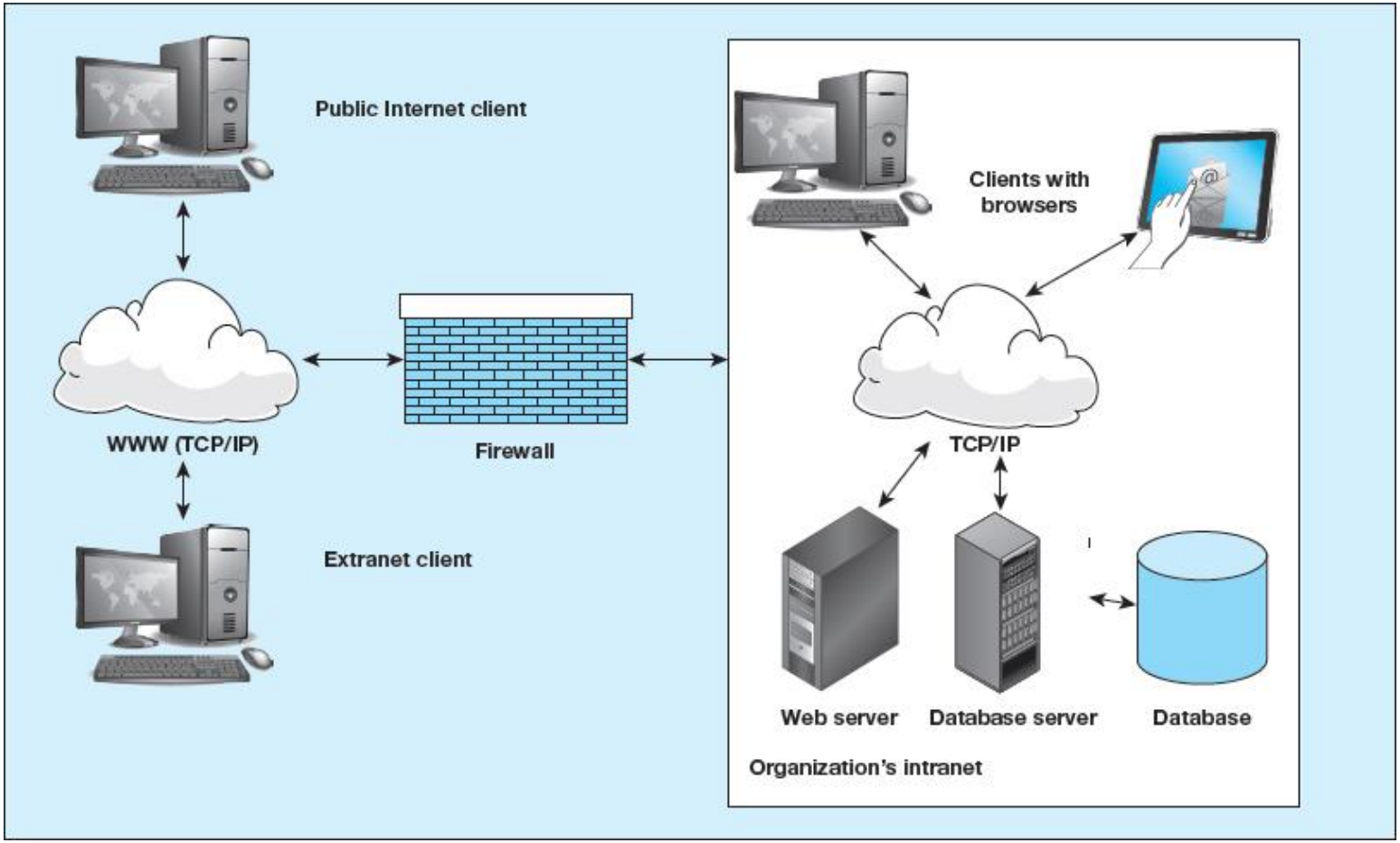
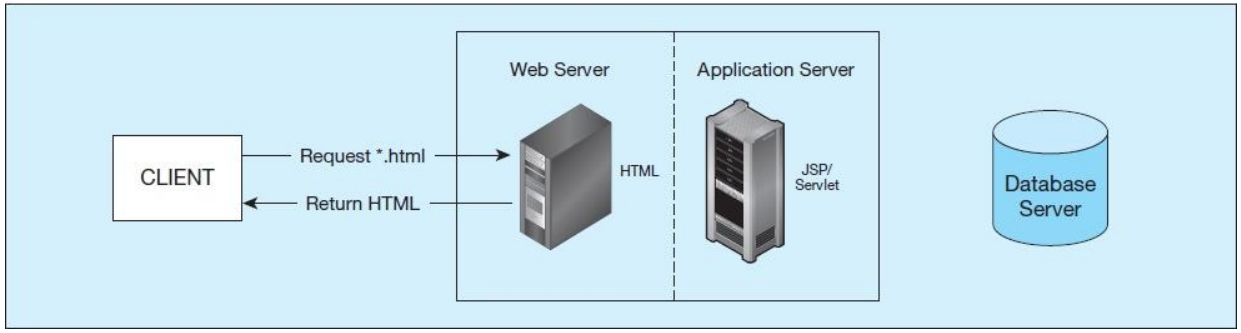
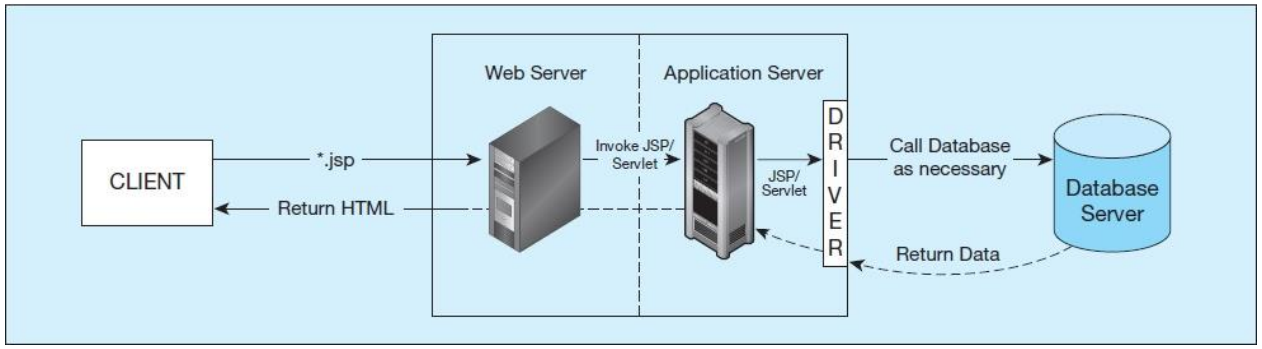


Figure 7-5 Information Flow in a Three-Tier Architecture

(a) Static page request



(b) Dynamic page request



Middleware and APIs

- Middleware – software that allows an application to interoperate with other software without requiring user to understand and code low-level operations
- Application Program Interface (API) – routines that an application uses to direct the performance of procedures by the computer's operating system
- Common database APIs – ODBC, ADO .NET, JDBC

Steps for Using Databases via Middleware APIs

1. Identify and register a database driver.
2. Open a connection to a database.
3. Execute a query against the database.
4. Process the results of the query.
5. Repeat steps 3–4 as necessary.
6. Close the connection to the database.

Figure 7-7 Database Access from a Java Program

Java implementation of database access to an Oracle database using JDBC

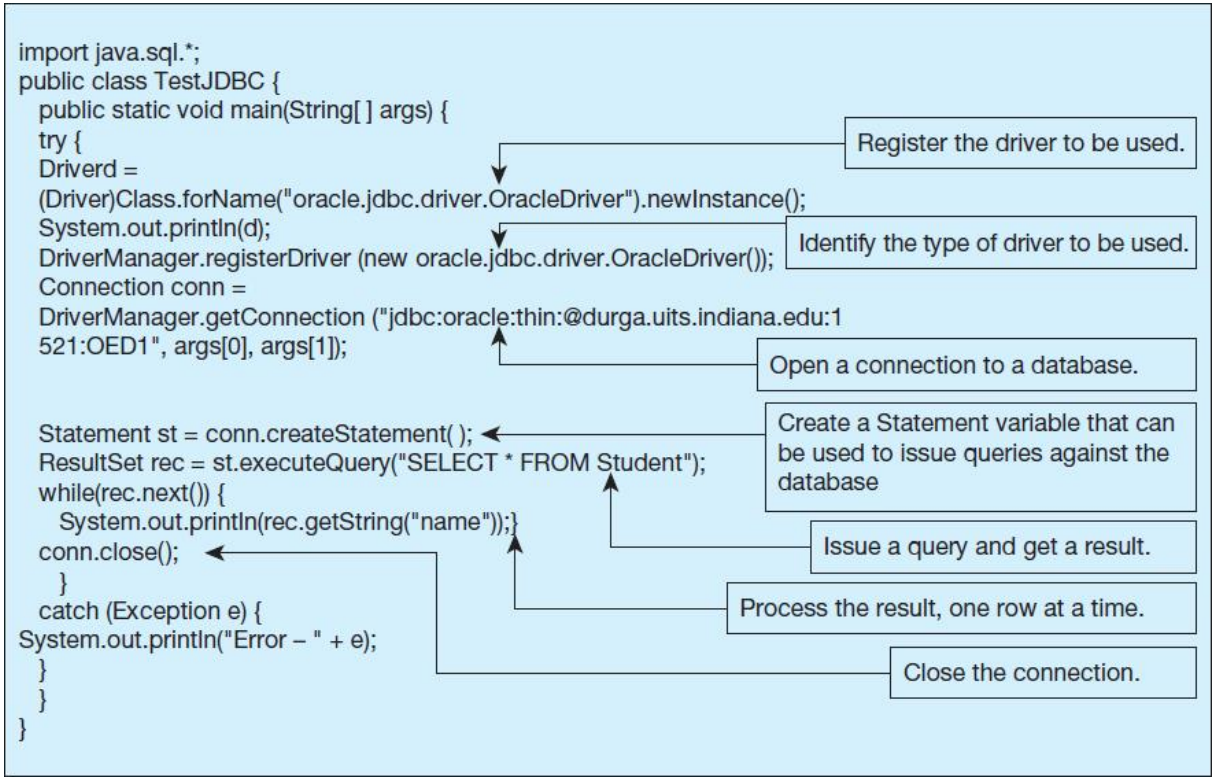


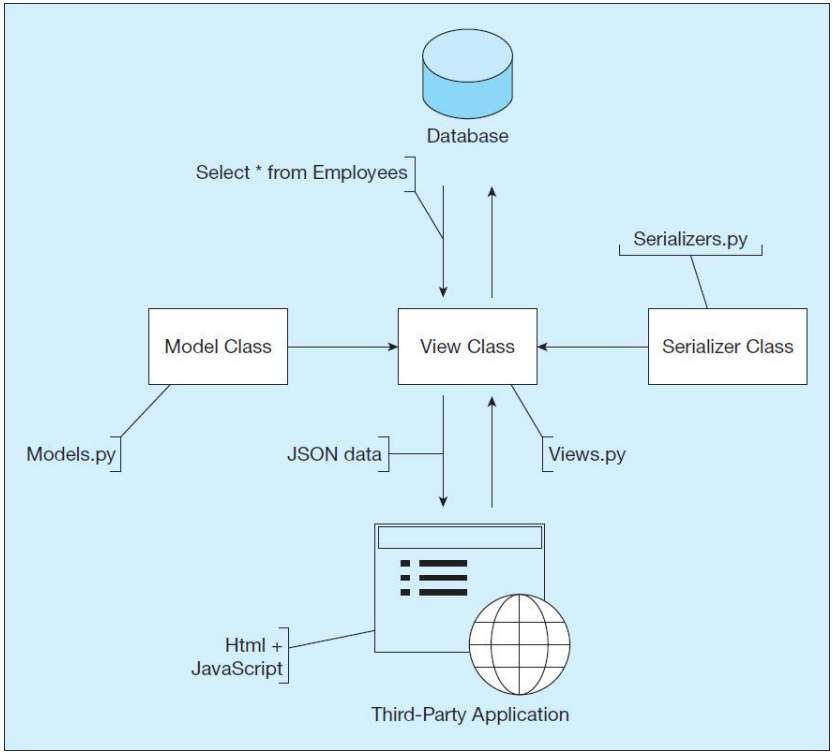
Figure 7-8 System Architecture of Sample Application

Python application for database processing.

Model class creates instances of data from table queries.

Serializer class transforms database data into a form that can be used by the client (JSON).

View classes display in browsers for view or edit.



Considerations in 3-Tier Applications

- Stored procedures
  - Code logic embedded in DBMS
  - Improve performance, but proprietary
- Transactions
  - Involve many database updates
  - Either all must succeed, or none should occur
- Database connections
  - Maintaining an open connection is resource-intensive
  - Use of connection pooling

Advantages and Disadvantages of Stored Procedures

- Advantages
  - Performance improves for compiled SQL statements
  - Reduced network traffic
  - Improved security
  - Improved data integrity
  - Thinner clients
- Disadvantages
  - Programming takes more time
  - Proprietary, so algorithms are not portable

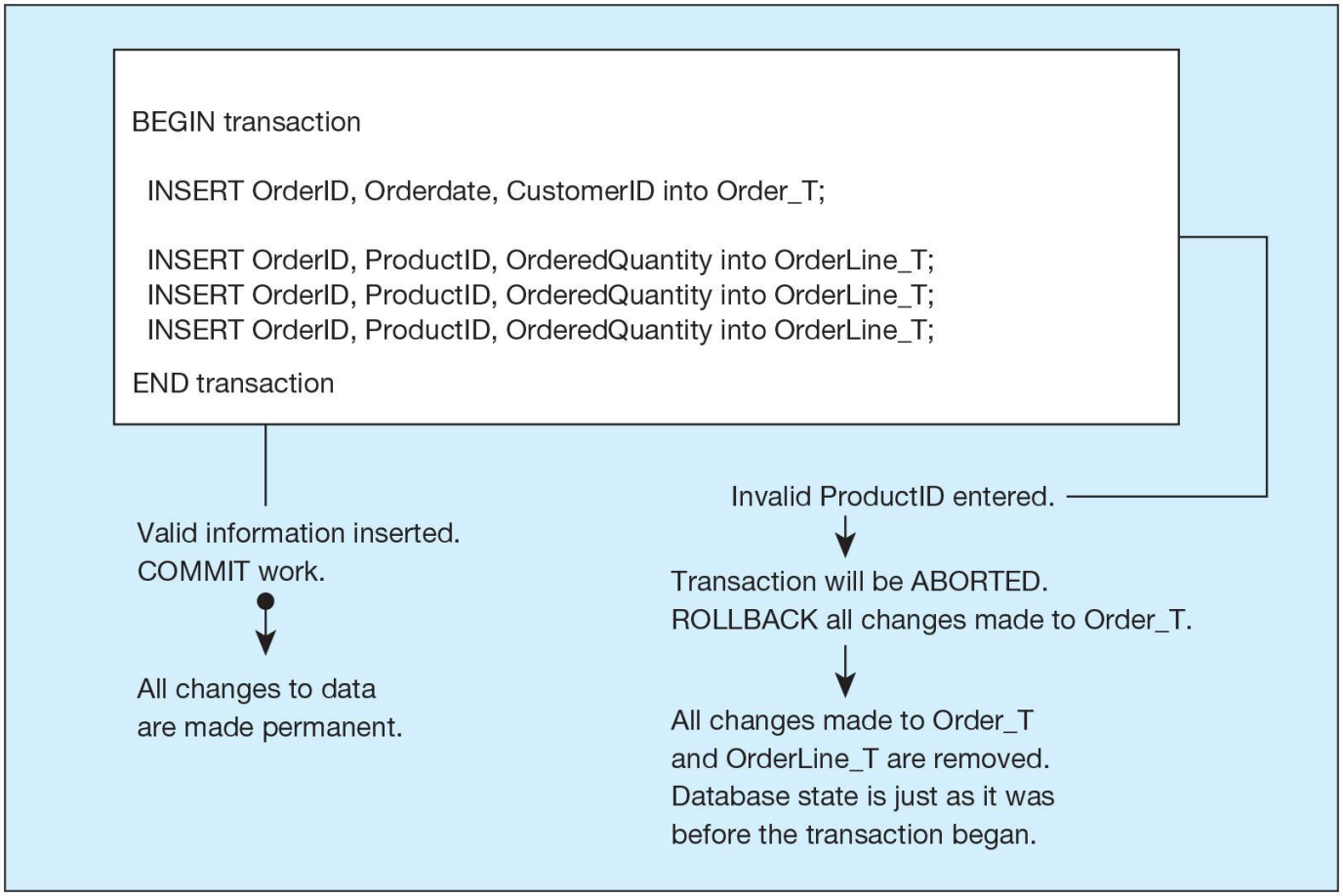
Benefits of Three-Tier Architectures

- Scalability
- Technological flexibility
- Long-term cost reduction
- Better match of systems to business needs
- Improved customer service
- Competitive advantage
- Reduced risk

Transaction Integrity: ACID Rules

- Atomic
  - Transaction cannot be subdivided
- Consistent
  - Constraints don’t change from before transaction to after transaction
- Isolated
  - Database changes not revealed to users until after transaction has completed
- Durable
  - Database changes are permanent

Figure 7-19 An SQL Transaction Sequence (Pseudocode)

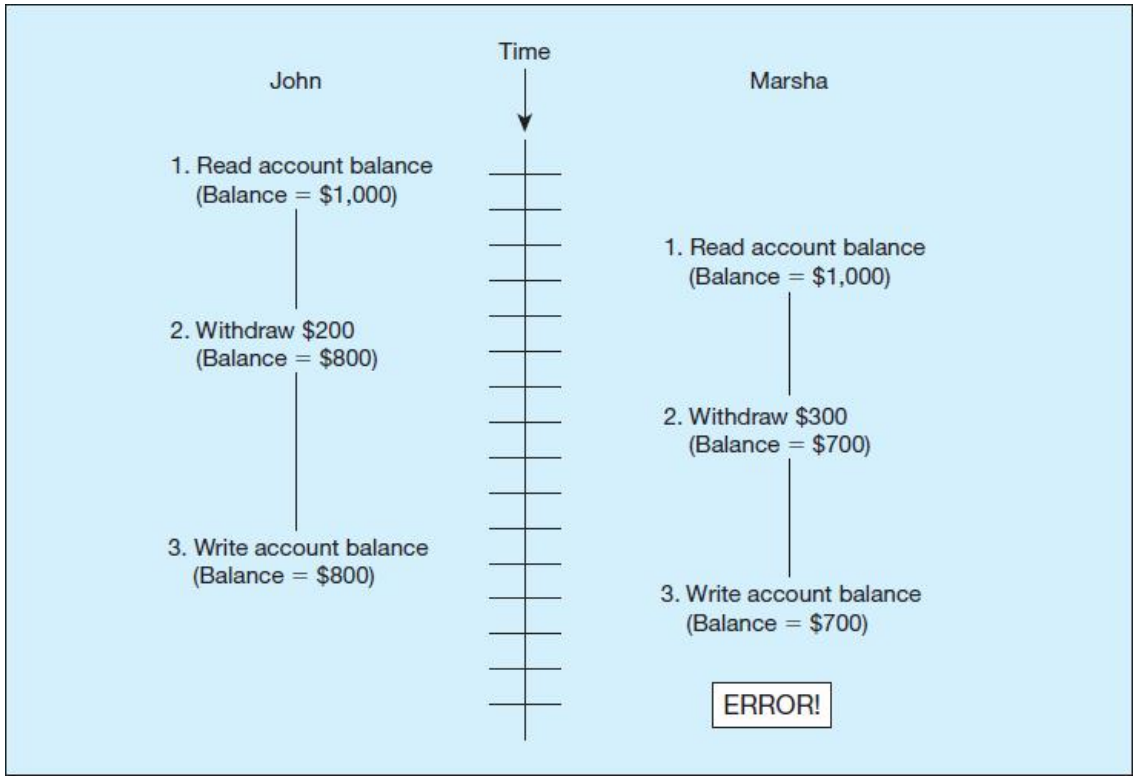


Controlling Concurrent Access

- Problem
  - In a multi-user environment, simultaneous access to data can result in interference and data loss (**lost update** problem)
- Solution – Concurrency Control
  - Managing simultaneous operations against a database so that data integrity is maintained and the operations do not interfere with each other in a multi-user environment

Figure 7-20 Lost Update (No Concurrency Control in Effect)

Simultaneous access causes updates to cancel each other. A similar problem is the **inconsistent read** problem.

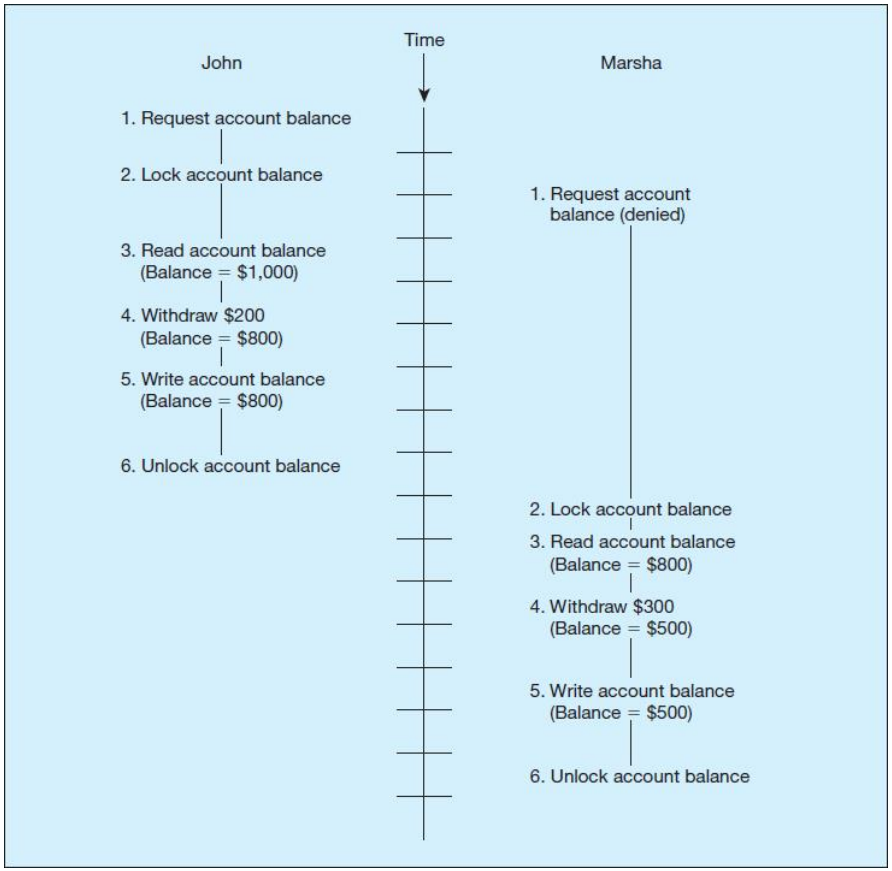


Concurrency Control Techniques

- Serializability
  - Finish one transaction before starting another
- Locking Mechanisms
  - The most common way of achieving serialization
  - Data that is retrieved for the purpose of updating is locked for the updater
  - No other user can perform update until unlocked

Figure 7-21 Updates with Locking (Concurrency Control)

Locking solves the lost update problem

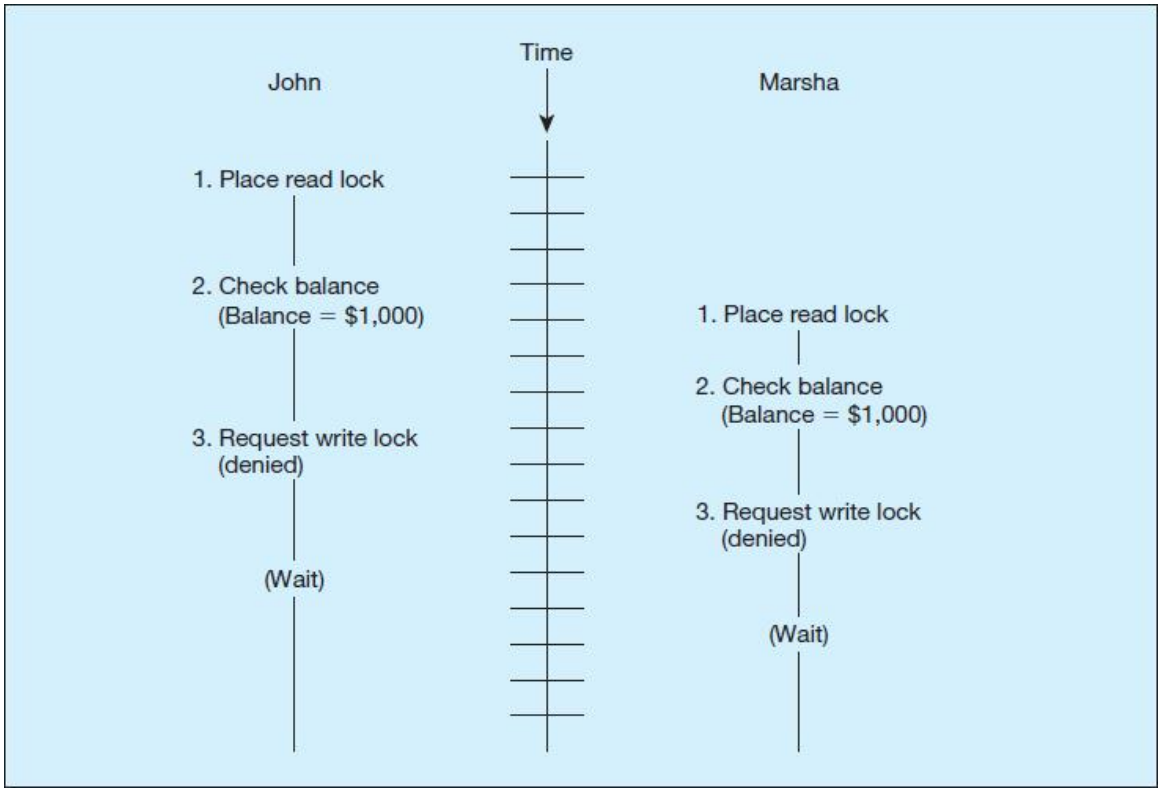


Locking Mechanisms

- Locking level:
  - Database – used during database updates
  - Table – used for bulk updates
  - Block or page – very commonly used
  - Record – only requested row; fairly commonly used
  - Field – requires significant overhead; impractical
- Types of locks:
  - Shared lock – Read but no update permitted. Used when just reading to prevent another user from placing an exclusive lock on the record
  - Exclusive lock – No access permitted. Used when preparing to update

Deadlock

- An impasse that results when two or more transactions have locked common resources, and each waits for the other to unlock their resources
- Figure 7-22 shows the problem of deadlock
- John and Marsha will wait forever for each other to release their locked resources!



Managing Deadlock

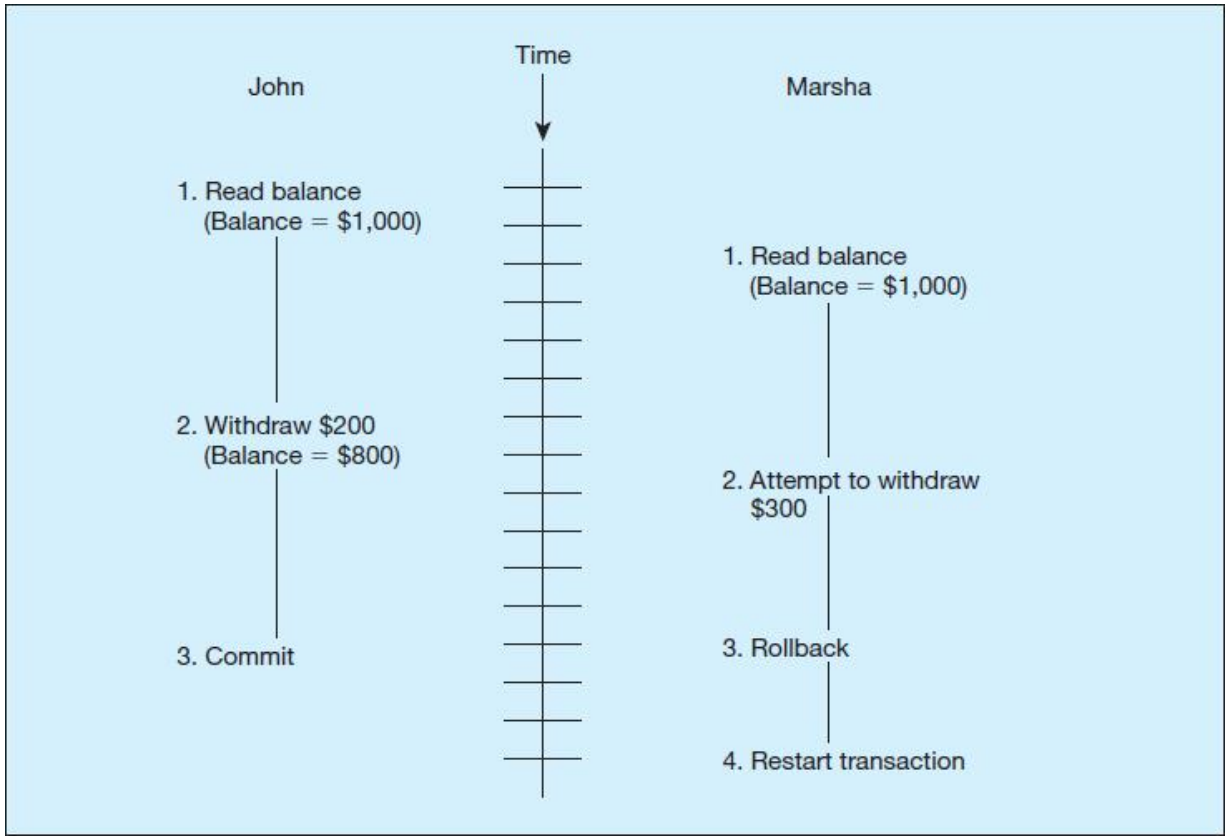
- Deadlock prevention:
  - Lock all records required at the beginning of a transaction
  - Two-phase locking protocol
    - Growing phase
    - Shrinking phase
  - May be difficult to determine all needed resources in advance
- Deadlock Resolution:
  - Allow deadlocks to occur
  - Mechanisms for detecting and breaking deadlock
    - Resource usage matrix
    - Back out one deadlock at a time
    - Rerun transaction

Versioning

- Optimistic approach to concurrency control
- Instead of locking
- Assumption is that simultaneous updates will be infrequent
- Each transaction can attempt an update as it wishes
- The system will create a new version of a record instead of replacing the old one
- When a conflict occurs, accept one user's update and inform the other user that its update needs to be tried again.
- Use of rollback and commit for this

Figure 7-24 The Use of Versioning

Better performance than locking



Data Security

- **Database Security:** Protection of the data against accidental or intentional loss, destruction, or misuse
- Increased difficulty due to Internet access and client/server technologies

Threats to Data Security

- Accidental losses attributable to:
  - Human error
  - Software failure
  - Hardware failure
- Theft and fraud
- Loss of privacy or confidentiality
  - Loss of privacy (personal data)
  - Loss of confidentiality (corporate data)
- Loss of data integrity
- Loss of availability (e.g., through sabotage)

Figure 7-25 Possible Locations of Data Security Threats

Threats come from many sources and vulnerabilities exist in multiple places within an information system.

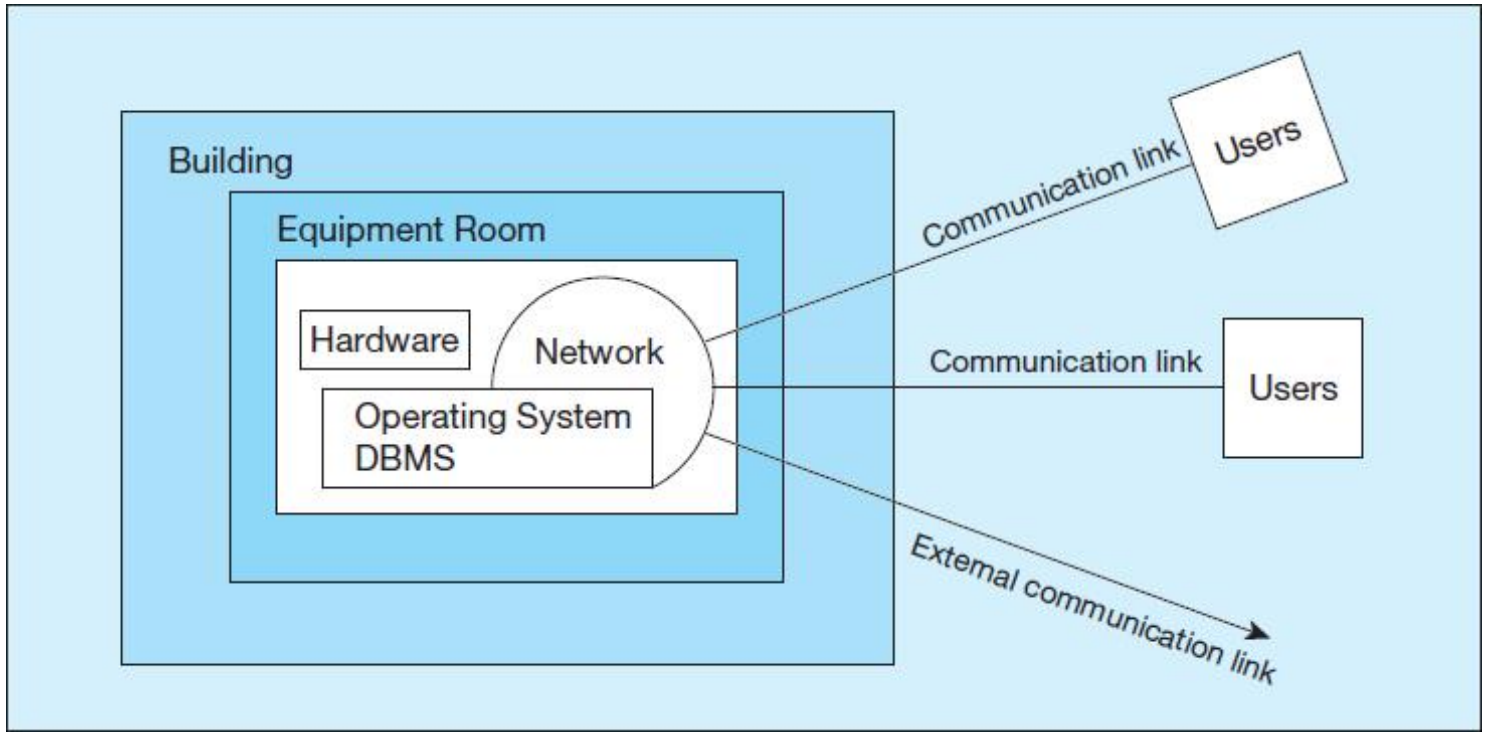
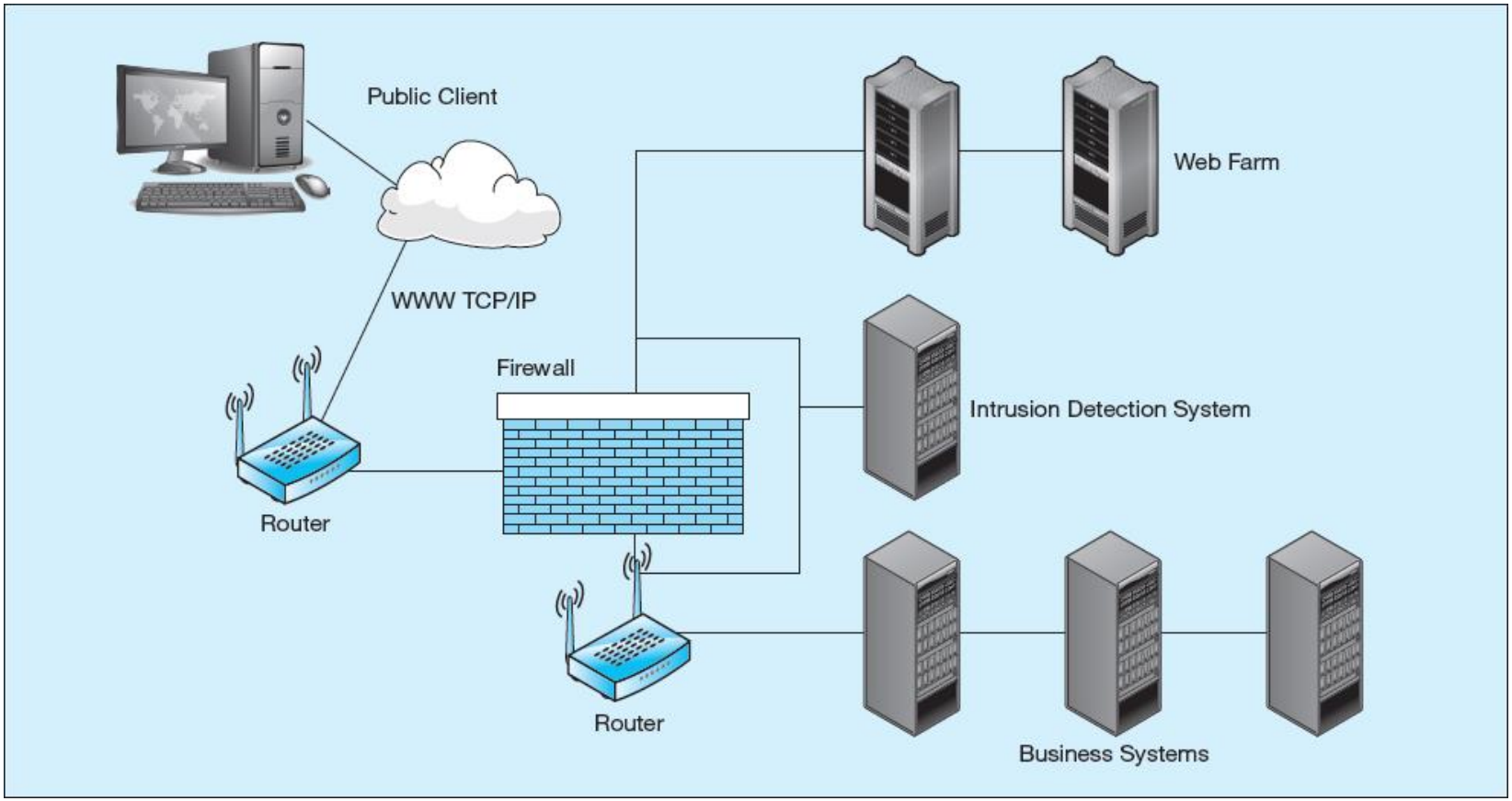


Figure 7-26 Establishing Internet Security



Client–Server Application Security

- Static HTML files are easy to secure
  - Standard database access controls
  - Place Web files in protected directories on server
- Dynamic pages are harder
  - User authentication
  - Session security
  - SSL for encryption
  - Restrict number of users and open ports
  - Remove unnecessary programs

Data Privacy

- W3C Web Privacy Standard
  - Platform for Privacy Protection (P3P)
- Addresses the following:
  - Who collects data
  - What data is collected and for what purpose
  - Who is data shared with
  - Can users control access to their data
  - How are disputes resolved
  - Policies for retaining data
  - Where are policies kept and how can they be accessed

Copyright