Transaction Processing Systems

- Business Transaction
  - Interaction in real world
  - Usually between enterprise and person
  - Or maybe between enterprises

- Transaction Program
  - Performs function on (shared) database

- Online Transaction Processing System
  - Runs collection of transaction programs

- Our notion of “Information System”
The ACID Properties

- **Atomicity**
  - All *commit* or nothing *abort*

- **Consistency**
  - Map good states to good states

- **Isolation**
  - Concurrent transactions *serializable*

- **Durability**
  - Committed transactions are not lost

Difficult in centralized database
Especially difficult in distributed system
Atomicity

• All (commit) or nothing (abort)

• Example: transfer money between two bank accounts

• Some actions (“launch the missile”) are not recoverable

• Compensating transactions?
Consistency

• Each transaction takes valid states to valid states:
  • Satisfy integrity constraints
  • Sometimes the only notion of “valid” state is “a state that could have been produced by executing a sequence of transactions"
Isolation

- Each transaction behaves as if it were executed in isolation at some instant in time
- AKA serializability
- Consistency + Isolation implies the data remains consistent even when multiple transaction programs execute concurrently
Durability

- The effect of a committed transaction will not be lost.
- So data must be on stable storage before commit.
- Usually done with a log (or journal) that must be forced before commit.
- Crash recovery using the log.
Resource Manager

- How ACID transactions are implemented
- Allocate resources to program executing a transaction
  - e.g. a locked record is a resource
- Reclaim resources in appropriate state on commit or abort

This is the meaning of “Resource Management Layer” in [ACKM04]
Three Layers of an Info System

client

presentation layer

application logic layer

resource management layer

information system
Presentation Layer

- Controls how the information system presents information to external entities and accepts it from them.
- External entities are users (UI) or other information systems (API)
Application Logic Layer

- The program
- Business process
- Business logic
- Business rules
• The *data layer* as discussed above
Top Down Design

1. define access channels and client platforms
2. define presentation formats and protocols for the selected clients and protocols
3. define the functionality necessary to deliver the contents and formats needed at the presentation layer
4. define the data sources and data organization needed to implement the application logic
Top Down Architecture

Top-down design

- PL-A
- PL-B
- PL-C

AL-A
AL-B
AL-C
AL-D

RM-1
RM-2

top-down architecture

- PL-A
- PL-B
- PL-C

AL-A
AL-B
AL-C
AL-D

RM-1
RM-2

Copyright Springer Verlag Berlin Heidelberg 2004
Bottom Up Design

1. define access channels and client platforms
2. examine existing resources and the functionality they offer
3. wrap existing resources and integrate their functionality into a consistent interface
4. adapt the output of the application logic so that it can be used with the required access channels and client protocols
Bottom Up Architecture

bottom-up design

<table>
<thead>
<tr>
<th>PL-A</th>
<th>PL-B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PL-C</td>
</tr>
</tbody>
</table>

AL-A  AL-B  AL-C  AL-D
wrapper  wrapper  wrapper

bottom-up architecture

<table>
<thead>
<tr>
<th>PL-A</th>
<th>PL-B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PL-C</td>
</tr>
</tbody>
</table>

AL-A  AL-B  AL-C  AL-D
wrapper  wrapper  wrapper

legacy system  legacy system  legacy system
1-Tier Architecture

client

presentation layer

application logic layer

resource management layer

information system
1-Tier - Remarks

- System is necessarily monolithic
- May be highly efficient
- No stable service interface API
  - That’s what screen scrapers are for
- Problem of Legacy Systems
2-Tier Architecture
2-Tier - Advantages

• Added flexibility in presentation layer
  • e.g. multiple specialized presentation layers add no complexity to application

• Encouraged stable, published APIs
  • So clients could be developed
Server Organization: 2-Tier

- server's API
- service interface
  - service
  - service
  - service
  - service
- resource management layer
2-Tier - Disadvantages

• A single server doesn’t scale
• Integration of multiple services must be done at client
Integration by Client

Client

Application logic

Presentation layer 1

Presentation layer 2

Server 1

Application logic layer

Resource management layer

Server 2

Application logic layer

Resource management layer
3-Tier Architecture

- **Client**
  - Presentation layer
- **Application Logic layer**
- **Resource Management layer**

Information System

Middleware
3-Tier - Advantages

- Scalability at application layer
  - Multiple application servers
- Application Integration
  - Do it in the middle tier
- Encourage stable, published APIs for resource management layer
App Integration in Middle Tier

- **client**
  - presentation layer

- integration logic
- application logic layer

- middleware

- **wrapper**
  - 1-tier
  - 2-tier
  - 3-tier

- resource management layer
Inductively, N-Tier Architecture

- **Client**
  - **Presentation Layer**

- **Integration Logic**
  - **Application Logic Layer**
  - **Middleware**

- **Tier Analysis**
  - **1-Tier**
    - **Wrapper**
  - **2-Tier**
    - **Wrapper**
  - **3-Tier**
    - **Resource Management Layer**
  - **4-Tier**
  - **5-Tier**

- **Layer Breakdown**
  - **Client**
  - **Presentation Layer**
  - **Application Logic Layer**
  - **Resource Management Layer**
  - **Middleware**
  - **Integration Logic**

- **Tier Comparison**
  - 3-Tier
  - 4-Tier
  - 5-Tier
N-Tier in the Enterprise

- Remote clients
- Internet
- Firewall
- LAN
- Internal clients
- Web server cluster
- LAN
- Middleware and application logic
- LAN
- Gateway
- LAN, gateways

Resource management layer:
- Database server
- File server
- Application

Additional resource management layers:
- Wrappers and gateways
Communication

- 1-way messaging
- synchronous RPC
- asynchronous RPC
I-Way Messaging

- Request
- Invoking execution thread
- Invoked execution thread
- (no response)
Synchronous RPC

invoking execution thread

request

invoked execution thread

response

blocking period
Synchronous RPC Issues

- Forcing caller to wait may reduce parallelism and waste resources
- Connection management issues
- Round-trip time issues
Asynchronous RPC
Asynchronous RPC

- Tolerates application (but not queue) failures
- Advantages for application integration
- Advantages for connection management
- Message-Oriented Middleware (MOM) and Message Brokers
Types of Middleware

- RPC-based systems
- TP Monitors
- Object Brokers
- Object Monitors
- Message-Oriented Middleware (MOM)
- Message Brokers
RPC Systems

- The foundation for most of the others
- We’ll cover in some depth
- Requires support infrastructure
  - Interface Definition Language (IDL) compilers, stub generators
  - directory services for binding
  - etc.
TP Monitors

- Most established form of middleware
- TP-Lite
  - A 2-Tier architecture
  - Allow application logic as stored procedures in a database
- TP-Heavy
  - 3-Tier architecture
  - Implements transactional RPC
  - Coordinator for distributed transactions
Objects

- Object Broker (OMG CORBA)
  - Like an object-oriented RPC system
- Object Monitor
  - Like an object-oriented TP Monitor
Queues

• Message-Oriented Middleware
  • Queues (and transactional queues) to support asynchronous messaging

• Message Broker
  • All of the above
  • Ability to run application logic for message routing