

Client-Server Development in VB

by

Jonathan Zuck

Vice President, Client-Server Technology

Advanced Paradigms, Inc.

(703) 578 - 1400 / (703) 578-1405 FAX

CIS: 76702,1605

Implementing Client-Server

Client-Server Defined

Advantages to Client-Server

Migration Strategies

Logical Architecture

Physical Architecture

Tools, APIs and Libraries

Access Engine Considerations

Building a Client-Server Application

Implementing Client-Server

Client-Server Defined

Advantages to Client-Server

Migration Strategies

Logical Architecture

Physical Architecture

Tools, APIs and Libraries

Access Engine Considerations

What is Client-Server?

Client-Server computing is a processing architecture in which a single application is partitioned among multiple processors which cooperate in a unified manner to complete the unit of work as a single task.

Advantages of Client-Server

Division of Labor

Division of Integrity

Long-term Economic Advantages

Data Access Flexibility

Migration Strategies

Client-Host Computing (front-end)

Direct Access to Mainframe Data

Localize non-Critical Information

Full Distributed Processing

Client-Host Computing

Advantages

- GUI Environment
- Preservation of legacy system
- Data Access Shortcuts

Disadvantages

- Not Really Distributed
- Preservation of legacy system
- Limited Flexibility

Access to Mainframe Data

Advantages

- GUI Interface

- More Efficient Data Access

- Preservation of Legacy Data Schema

Disadvantages

- Inverted processing

- Preservation of Legacy Data Schema

- non-Portable

Logical Architecture

Independent of Physical Architecture

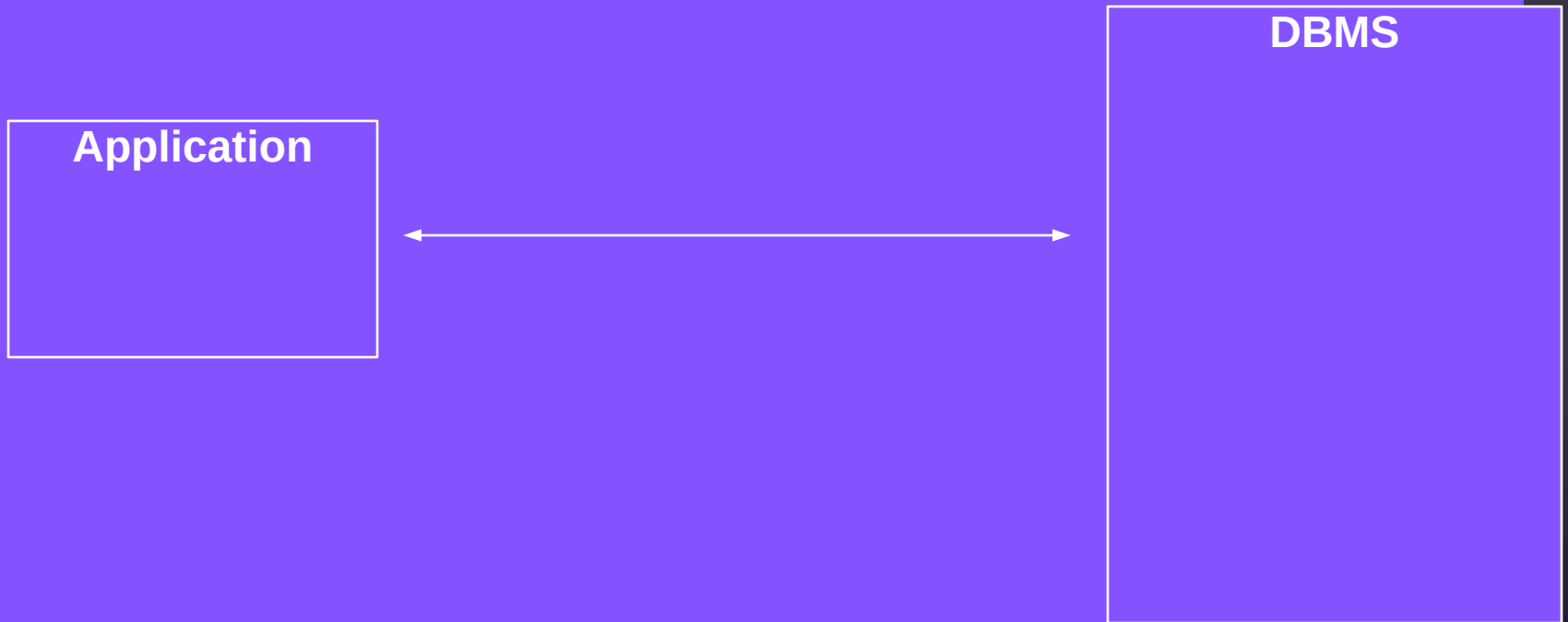
Independent of Toolset

Services Orientation

Object-Orientation

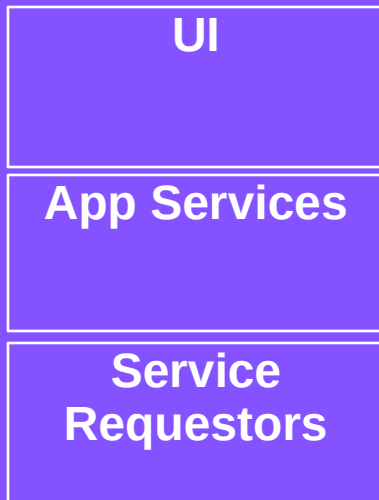
Start with too many layers

Logical Architecture Core



Logical Service Orientation

Application



Server



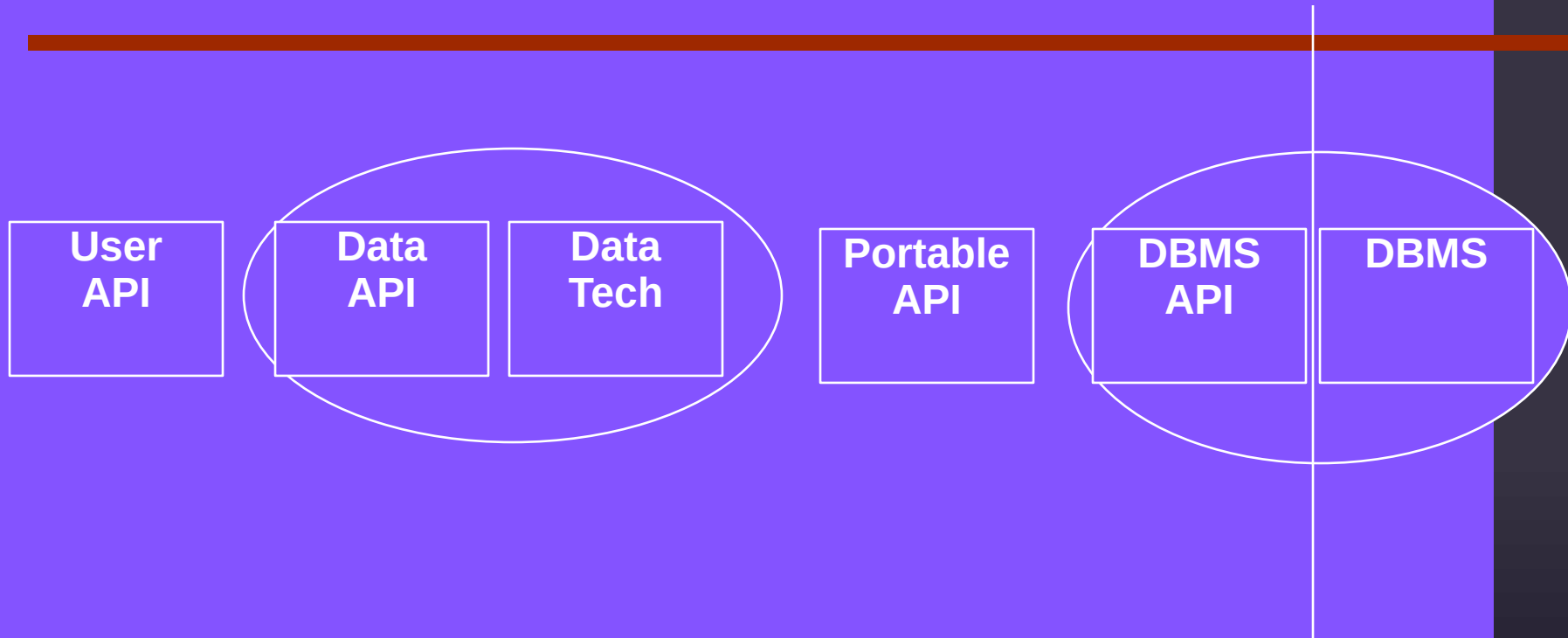
Physical Architecture

Network Architecture

Communication Protocols

Tools Architecture

Tools Architecture



Jet Engine Considerations

Scalable Solution

Tight Integration with VB

Flexible Data Access

Overhead

Non-service Orientation

Implementing a CS Application

Database Design

- Normalization

- Business Rules

Views Design

Forms Implementation

Using Case Tools

Schema Development

Relation diagramming

Data Dictionary

Repository Creation

Database Generation

Data Flow Diagramming

Examples

ERwin

InfoDesigner

Normalization

Eliminate Duplicate Information

Accomodate Future Changes to Structure

Minimize Impact on Front-End

First Normal Form

Flat

No repeating groups

Second Normal Form

Non-key Data dependent on Key data
Uniquely identify a row with a key

Third Normal Form

Second Normal Form

Fields independent of each other

Denormalization

Don't be Inflexible

Be Aware of Performance

Don't make too many tables

Indexing

Primary Keys

Likely Search Candidates

Likely Sort Candidates

SQL Server for NT

Stored Procedures

Triggers

Extended Stored Procedures

Stored Procedures

**Pre-compiled
Security Benefits**

Triggers

Database Driven Stored Procedures Enforce Referential Integrity

```
create trigger cancelClass on class for delete as  
begin  
    DELETE attendee FROM attendee, deleted  
    where deleted.class_id = attendee.class_id  
end
```

Extended Stored Procedure

Written in C

Access to any DLL function

Writing ESPs in Visual Basic?

ESPs in Visual Basic

xp_cmdshell stored procedure

write an ODBC app in VB

“exec xp_cmdshell “myapp.exe”, no_output

Creating an ODBC Data Source

ODBC Data Manager

Naming your Database Source

Attaching from Access

Attach Table

Fastest Access to Data

Relatively Transparent to VB

Creating Views

Flat Files (courses, customers, etc.)

One to Many Forms

Many to Many Forms

Gotchas

Connection Management

Use SQL Passthrough

Locking Schemes