**Hong Zhang**

**CIS2710 CH8 Assignment**

**12/03/2013**

***Case Exercises 1:*** Outline the advantages and risks/disadvantages of moving toward an integrated Web-based environment based on a three-tier architecture. What do you think would be the most significant challenges with this integration approach? Which specific technologies would you recommend for implementing this solution?

Three-tier architecture is one of the basic paradigms of software development. This design pattern can be beneficially used in web integration projects, as a basis for the overall solution architecture, with benefits.

3-Tier client-server architectures have 3 essential components:

1. A Client PC

2. An Application Server

3. A Database Server

3-Tier Architecture Considerations:

1. Stored procedures

* Code logic embedded in DBMS
* Improve performance, but proprietary

1. Transactions

* Involve many database updates
* Either all must succeed, or none should occur

1. Database connections

* Maintaining an open connection is resource-intensive
* Use of connection pooling

3 – Tier Pros and Cons:

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Development Issues:   * Complex application rules easy to implement in application server * Business logic off-loaded from database server and client, which improves performance * Changes to business logic automatically enforced by server – changes require only new application server software to be installed * Application server logic is portable to other database server platforms by virtue of the application software | Development Issues:   * More complex structure * More difficult to setup and maintain |
| Performance:   * Superior performance for medium to high volume environments | Performance:   * The physical separation of application servers containing business logic functions and database servers containing databases may moderately affect performance. |

I think the most significant challenge with this integration approach is how to simplify 3-tier application development and administration by providing an extra application server layer to manage communication between components. That means how we can communicate with the database. It needs special software called database-oriented middleware. Middleware is software that allows an application to interoperate with other software without requiring user to understand and code low-level operations. The database-oriented middleware needed to connect an application to a database consists of two parts: an application programming interface (API) and a database driver to connect to a specific type database (e.g., SQL Server or Oracle). The most common APIs are Open Database Connectivity (ODBC) and ADO.NET for the Microsoft platform (VB.NET and C#) and Java Database Connectivity (JDBC) for use with Java programs.

No matter which API or language is used, the basic steps for accessing a database from an application remain surprisingly similar:

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.