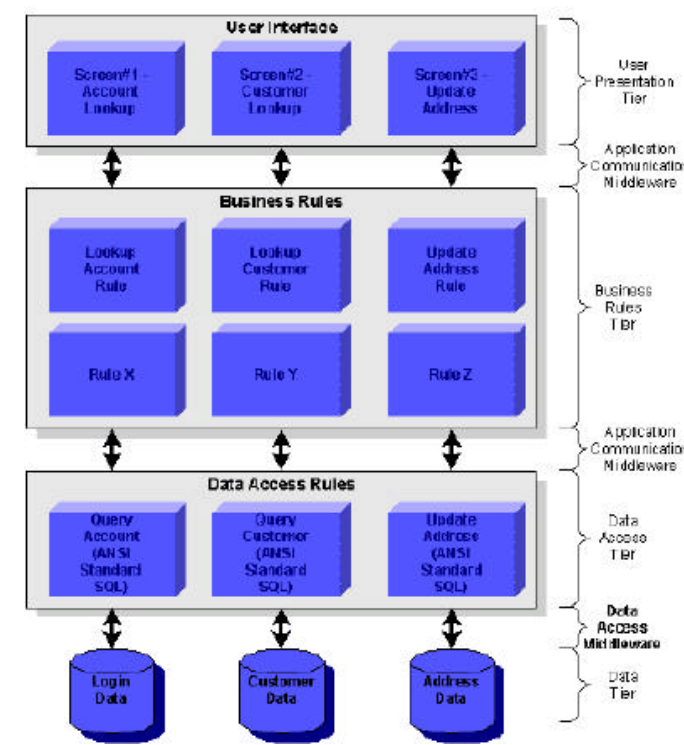
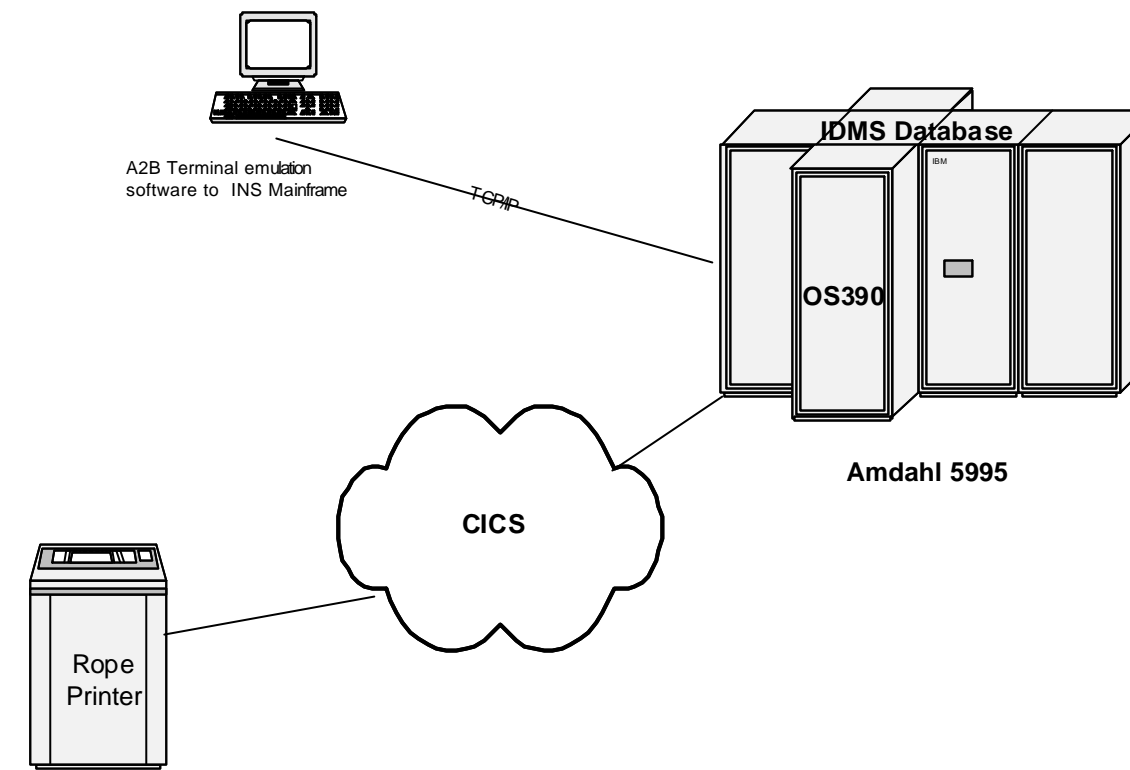


Database Runtime Patterns



J2EE connection to Oracle Database



The following implementation guidelines are for Data Access Middleware.

Guideline 1: Use OLE DB or JDBC to separate business rules from the data access tier.

Rationale: Use of OLE DB or JDBC will isolate the business rule from the data access tier when data modeling or database technology changes occur. The OLE DB and JDBC Call Level Interfaces (CLIs) are designed especially for object oriented languages. Much less software and related maintenance is required on the client side when the middleware is server based.

Standards: The standards in this section pertain to data access middleware.

Standard 1: Use OLE DB or JDBC database access middleware when accessing a database.

Rationale: Use OLE DB or JDBC to access a database instead of vendor specific database middleware. OLE DB and JDBC allow flexibility in programming. A database can be easily modified or relocated. If a change is needed, the change is made to the OLE DB or JDBC configurations, not to each data access program or tool. These technologies are widely supported by the industry and make an application more adaptable to changes in database or other technology requirements.

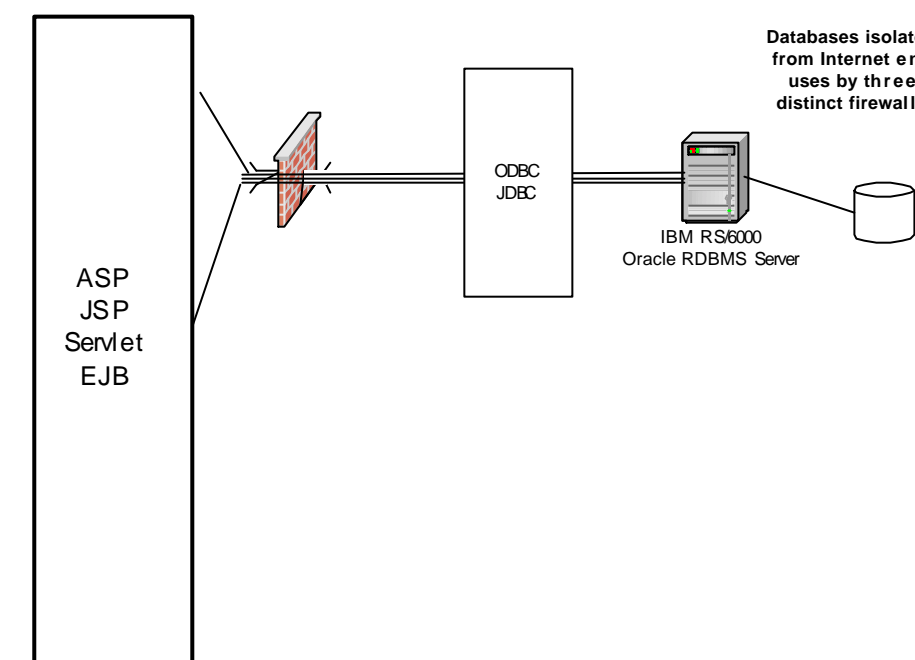
Standard 2: Implement a server-based OLE DB or JDBC solution as opposed to a workstation-based OLE DB, ODBC, or JDBC implementation.

Rationale: A server-based solution is easier to administer. Database changes and additions are easier to manage, since updates are made to database middleware servers, not every workstation that requires access.

Standard 3: Use domain name system (DNS) alias names when accessing databases through OLE DB and JDBC.

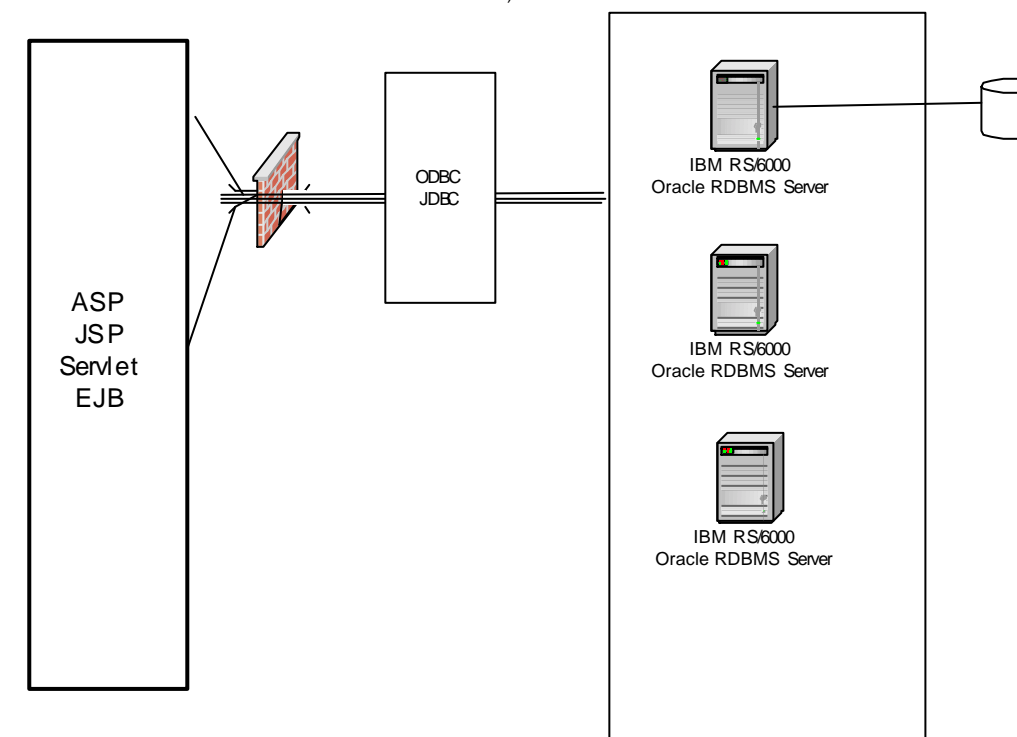
Rationale: If the database location changes or if the server name changes, the DNS configuration is changed, and no changes are needed to each client configuration.

Database Access from application server through firewall (as in current environment)



Databases isolated from Internet end uses by three distinct firewalls

Database Access from application server through firewall (as in current environment)



Oracle real time application cluster

Database-Specific Middleware Drivers

Structured query language (SQL) is a query language used to query and retrieve data from relational databases. The industry standard for SQL is ANSI Standard SQL. SQL drivers are implemented by each RDBMS vendor to enable database access to its proprietary database (e.g., SQL*Net is Oracle's SQL driver, Open Client is Sybase's SQL driver). Vendors may add extensions to the SQL language for their proprietary databases.

Open Database Connectivity (ODBC) Drivers

Open database connectivity (ODBC) drivers are the middleware used to connect database access rules to relational databases through the use of a generic application program interface (API). ODBC drivers are vendor-provided and allow databases to be connected and used by a generic interface. The ODBC drivers enable access to data and provide insulation between a program and the specific RDBMS language used by each database. Database access tools and programs do not have to be customized for each database, because an ODBC configuration file maintains the database connections. ODBC is language-independent, so many different programming languages can use it.

Technology Component 3: OLE DB

OLE DB is part of the COM+ Microsoft specification called Object Linking Embedding Database (OLE DB). OLE DB provides specifications for an object-oriented API to access relational and object-oriented databases. Microsoft introduced OLE DB as a universal technology to query and update data in all databases of an enterprise, regardless of where and how the data is stored. Both relational and object-oriented databases are supported by the OLE DB specifications.

Technology Component 4: Java database connectivity (JDBC) drivers

JDBC is a database connectivity driver developed by a subsidiary of Sun Microsystems. It is very similar to ODBC, only it is developed specifically for use by Java programs. JDBC is language-dependent because it requires the use of the Java programming language. With JDBC, Java programs can execute SQL queries against SQL-compliant databases. Since Java programs can execute on most of the major platforms, applications written in Java do not have to know the specific data access calls of each database being accessed. In this context, it can be referred to as database-independent.

Practical Enterprise Architecture	Data Architecture			
	Database Runtime pattern			
	SIZE	FSCM NO	DWGNO	REV
	John Wu		SHEET	9 OF 12