DB Connectivity | Meerza Ahmed

Tuesday, October 10, 2023 1:00 PM

Lecture 8: Link | Syllabus
**Class Week 8/15*

Database Systems by Coronel & Morris

Chapter 15: Database Connectivity and Web Technologies

* 15.1 *Database Connectivity*

Database connectivity refers to the mechanisms through which application programs connect and communicate with data repositories. Databases store data in persistent storage structures so it can be retrieved at a later time for processing. The database management system (DBMS) functions as an intermediary between the data (stored in the database) and the end-user's applications.

* 15.1-a *Native SQL Connectivity*

Most DBMS vendors provide their own methods for connecting to their databases. Native SQL connectivity refers to the connection interface that is provided by the database vendor and is unique to that vendor. The best example of this type of native interface is the Oracle RDBMS. To connect a client application to an Oracle database, you must install and configure Oracle's SQL*Net interface on the client computer. Figure 5.1 shows the configuration of the Oracle SQL*Net interface on the client computer.
* Usually, the native database connectivity interface provided by the vendor is not the only way to connect to a database; most current DBMS products support other database connectivity standards, the most common being ODBC.

* 15.1-b *ODBC, DAO, and RDO*

Developed in the early 1990s, Open Database Connectivity (ODBC) is Microsoft's implementation of a superset of the SQL Access Group Call Level Interface (CLI) standard for database access. ODBC is probably the most widely supported database connectivity interface. ODBC allows any Windows application to access relational data sources, using SQL via a standard application programming interface (API).

* As programming languages evolved, ODBC did not provide significant functionality beyond the ability to execute SQL to manipulate relational-style data. Therefore, programmers needed a better way to access data.

To answer that need, Microsoft developed two other data access interfaces:
- Data Access Objects (DAO) is an object-oriented API used to access desktop data bases, such as MS Access and FileMaker Pro. DAO provides an optimized interface that exposes programmers to the functionality of the Jet data engine, on which MS Access is based. The DAO interface can also be used to access other relational-style data sources.
- Remote Data Objects (RDO) is a higher-level, object-oriented application interface used to access remote database servers. RDO uses the lower-level DAO and ODBC for direct access to databases. RDO is optimized to deal with server-based databases such as MS SQL Server, Oracle, and DB2.

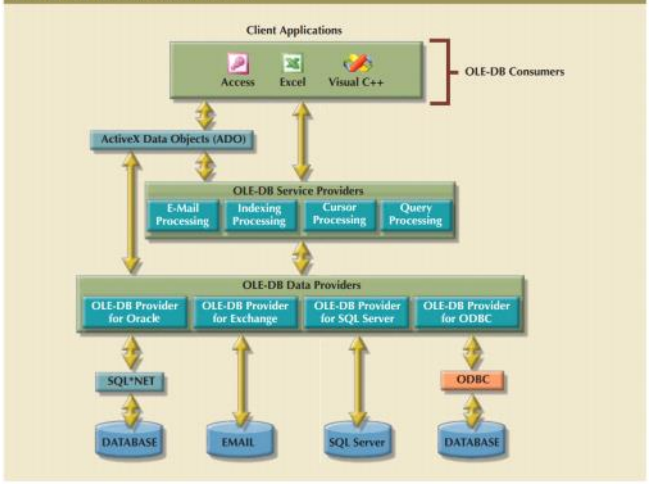
* 15.1-c *OLE-DE*

Although ODBC, DAO, and RDO are used, they do not provide support for nonrelational data. To answer that need and to simplify data connectivity, Microsoft developed Object Linking and Embedding for Database (OLE-DB). Based on Microsoft's Component Object Model (COM), OLE-DB is database middleware that adds object-oriented functionality for access to relational and nonrelational data. OLE-DB was the first part of Microsoft's strategy to provide a unified object-oriented framework for the development of next-generation applications.

OLE-DB is composed of a series of COM objects that provide low-level database connectivity for applications. Because OLE-DB is based on COM, the objects contain data and methods, also known as the interface.

The OLE-DB model is better understood when you divide its functionality into two types of objects :
- *Consumers* are objects (applications or processes) that request and use data. Consumers request data by invoking the methods exposed by the data provider objects (public interface) and passing the required parameters.
- *Providers* are objects that manage the connection with a data source and provide data to the consumers. Providers are divided into two categories: data providers and service providers.
  - *Data providers* provide data to other processes. Database vendors create data provider objects that expose the functionality of the underlying data source (relational, object-oriented, text, and so on)
  - *Service providers* provide additional functionality to consumers. The service provider is located between the data provider and the consumer. The service provider requests data from the data provider, transforms the data, and then provides the transformed
    data to the data consumer. In other words, the service provider acts like a data consumer of the data provider and as a data provider for the data consumer (end-user application). For example, a service provider could offer cursor management services, transaction management services, query processing services, and indexing services.

FIGURE 15.5 OLE-DB ARCHITECTURE



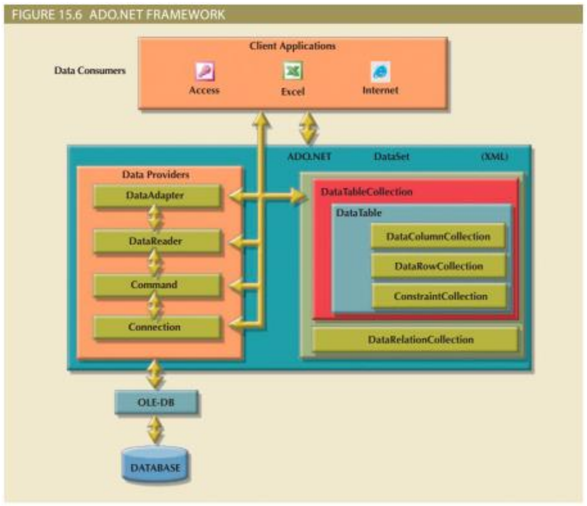
*** 15.1-d** *ADO.NET*

Based on ADO, ADO.NET is the data access component of Microsoft's .NET application development framework. The Microsoft .NET framework is a component-based platform for developing distributed, heterogeneous, interoperable applications aimed at manipulating any type of data using any combination of network, operating system, and programming language

* It is important to understand that the .NET framework extends and enhances the functionality provided by the ADO/OLE-DB duo. ADO.NET introduced two new features that are critical for the development of distributed applications: DataSets and XML support.

* A DataSet is a disconnected, memory-resident representation of the database. That is, the DataSet contains tables, columns, rows, relationships, and constraints. Once the data is read from a data provider, it is placed in a memory-resident DataSet, which is then disconnected from the data provider. The data consumer application interacts with the data in the DataSet object to make inserts, updates, and deletes in the DataSet. Once the processing is done, the DataSet data is synchronized with the data source and the changes are made permanent.
* The DataSet is internally stored in XML format, and the data in the DataSet can be made persistent as XML documents. This is critical in today's distributed environments. You can think of the DataSet as an XML-based, in-memory database that represents the persistent data stored in the data source.

FIGURE 15.6 ADO.NET FRAMEWORK



*** 15.1-e** *Java Database Connectivity (JDBC)*

Java is an object-oriented programming language developed by Sun Microsystems (acquired by Oracle in 2010) that runs on top of web browser software. Java is one of the most-common programming languages for web development. Sun Microsystems created Java as a "write once, run anywhere" environment, which means that a programmer can write a Java application once and then run it in multiple environments without any modification.

When Java applications need to access data outside the Java runtime environment, they use predefined application programming interfaces. Java Database Connectivity (JDBC) is an application programming interface that allows a Java program to interact with a wide range of data sources, including relational databases, tabular data sources, spreadsheets, and text files. JDBC allows a Java program to establish a connection with a data source, prepare and send the SQL code to the database server, and process the result set.

FIGURE 15.7 JDBC ARCHITECTURE

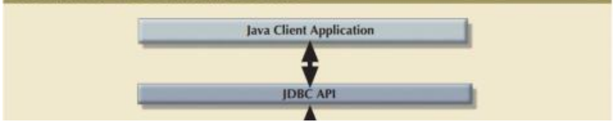
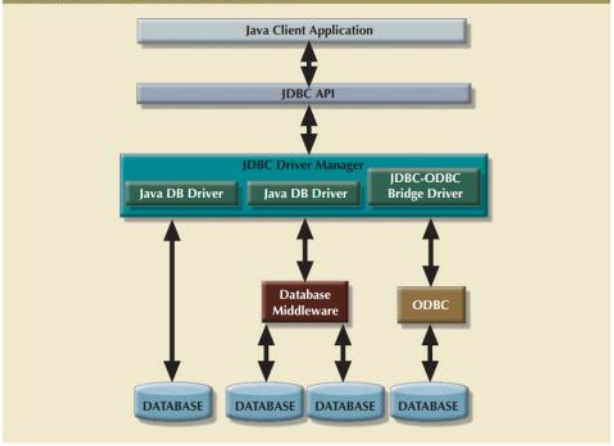


FIGURE 15.7 JDBC ARCHITECTURE



*** 15.2** *Database Internet Connectivity*

Internet database connectivity opens the door to new, innovative services that do the following :

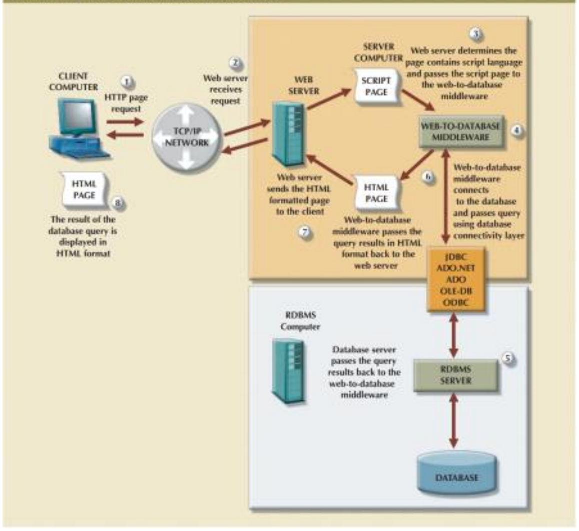
1. Permit rapid responses to competitive pressures by bringing new services and products to market quickly.
2. Increase customer satisfaction through the creation of innovative data services such as mapping data combined with GPS (Global Positioning System) information to provide location-aware services. These applications present end users with information or services located near the users' current location.
3. Allow anywhere, anytime data access using mobile smart devices via the Internet.
4. Yield fast and effective information dissemination through universal access from across the street or across the globe

* The simplicity of the web's interface and its cross-platform functionality are at the core of its success as a data access platform. In fact, the web has helped create a new information dissemination standard. The following sections examine how web-to-database middleware enables end users to interact with databases over the web.

*** 15.2-a** *Web-to-Database Middleware: Server-Side Extensions*

In general, the web server is the main hub through which all Internet services are accessed. For example, when an end user uses a web browser to dynamically query a database, the client browser requests a webpage from the web server. When the web server receives the page request, it looks for the page on the hard disk; when it finds the page, the server sends it back to the client.

FIGURE 15.8 WEB-TO-DATABASE MIDDLEWARE



CLIENT COMPUTER
1
HTTP page request
TCP/IP NETWORK
2
Web server receives request
WEB SERVER
SERVER COMPUTER
SCRIPT PAGE
3
Web server determines the page contains script language and passes the script page to the web-to-database middleware
WEB-TO-DATABASE MIDDLEWARE
4
Web-to-database middleware connects to the database and passes query using database connectivity layer
HTML PAGE
8
The result of the database query is displayed in HTML format
Web server sends the HTML formatted page to the client
HTML PAGE
6
7
Web-to-database middleware passes the query results in HTML format back to the web server
JDBC
ADO.NET
ADO
OLE-DB
ODBC
RDBMS Computer
Database server passes the query results back to the web-to-database middleware
RDBMS SERVER
5
DATABASE

* Trace the web-to-database middleware actions in Figure 15.8 :

1. The client browser sends a page request to the web server.
2. The web server receives and passes the request to the web-to-database middleware for processing.
3. Generally, the requested page contains some type of scripting language to enable the database interaction. The web server passes the script to the web-to-database middleware.
4. The web-to-database middleware reads, validates, and executes the script. In this case, it connects to the database and passes the query using the database connectivity layer.
5. The database server executes the query and passes the result back to the web-to-database middleware.
6. The web-to-database middleware compiles the result set, dynamically generates an HTML-formatted page that includes the data retrieved from the database, and sends it to the web server.
7. The web server returns the just-created HTML page, which now includes the query result, to the client browser.
8. The client browser displays the page on the local computer

15.2-b *Web Server Interfaces*

Extending web server functionality implies that the web server and the web-to database middleware will properly communicate with each other. (Database professionals often use the word *interoperate* to indicate that each party can respond to the communications of the other.) A web server interface defines a standard way to exchange messages with external programs.

Currently, there are two well-defined web server interfaces :

- Common Gateway Interface (CGI)
- Application programming interface (API)

* The Common Gateway Interface (CGI) uses script files that perform specific functions based on the client's parameters that are passed to the web server. The script file is a small program containing commands written in a programming language—usually Perl, C#, or Visual Basic. The script file's contents can be used to connect to the database and to retrieve data from it, using the parameters passed by the web server. Next, the script converts the retrieved data to HTML format and passes the data to the web server, which sends the HTML-formatted page to the client.

* An application programming interface (API) is a newer web server interface standard that is more efficient and faster than a CGI script. APIs are

more efficient because they are implemented as shared code or as dynamic-link libraries (DLLs). That means he API is treated as part of the web server program that is dynamically invoked when needed.

* APIs are faster than CGI scripts because the code resides in memory, so there is no need to run an external program for each request. Instead, the same API serves all requests. Another advantage is that an API can use a shared connection to the database instead of creating a new one every time, as is the case with CGI scripts.

15.2-c *The Web Browser*

The web browser is software such as Microsoft Internet Explorer, Google Chrome, Apple Safari, or Mozilla Firefox that lets end users navigate the web from their client computer. Each time the end user clicks a hyperlink, the browser generates an HTTP GET page request that is sent to the designated web server using the TCP/IP Internet protocol. The web browser's job is to *interpret* the HTML code that it receives from the web server and to present the various page components in a standard formatted way. Unfortunately, the browser's interpretation and presentation capabilities are not sufficient to develop web-based applications. The web is a stateless system—at any given time, a web server does not know the status of any of the clients communicating with it. That is, there is no open communication line between the server and each client accessing it, which of course is impractical in a worldwide web! Instead, client and server computers interact in very short "conversations" that follow the request-reply model.

15.2-d *Client-Side Extensions*

Client-side extensions add functionality to the web browser. Although client-side extensions are available in various forms, the most common are: Plug-ins, Java and JavaScript, ActiveX and VBScript

* A plug-in is an external application that is automatically invoked by the browser when needed. The plug-in is associated with a data object—generally using the file extension—to allow the web server to properly handle data that is not originally supported. For example, if one of the page components is a PDF document, the web server will receive the data, recognize it as a Portable Document Format object, and launch Adobe Reader to present the document on the client computer.

* JavaScript is a scripting language (one that enables the execution of a series of commands or macros) that allows web authors to design interactive sites. JavaScript code is embedded in the webpage and executed after a specific event, such as a mouse click on an object or a page being loaded from the server into memory.

* ActiveX is Microsoft's alternative to Java. ActiveX is a specification for writing programs that run inside the Microsoft client browser, Internet Explorer. Because ActiveX is oriented toward Windows applications, it has low portability. ActiveX extends the web browser by adding controls to webpages, including drop-down lists, a slider, a calendar, and a calculator. Those controls are downloaded from the web server when needed so you can manipulate data inside the browser. ActiveX controls can be created in several programming languages; C++ and Visual Basic are most commonly used. Microsoft's .NET framework allows for wider interoperability of ActiveX-based applications (such as ADO.NET) across multiple operating environments.

* VBScript is another Microsoft product that is used to extend browser functionality. VBScript is derived from Microsoft Visual Basic. Like JavaScript, VBScript code is embedded inside an HTML page and is activated by triggering events such as clicking a link.

15.2-e *Web Application Servers*

A web application server is a middleware application that expands the functionality of web servers by linking them to a wide range of services, such as databases, directory systems, and search engines. The web application server also provides a consistent runtime environment for web applications

* Examples of web application servers include ColdFusion/JRun by Adobe, WebSphere Application Server by IBM, WebLogic Server by Oracle, Fusion by NetObjects, Visual Studio .NET by Microsoft, and WebObjects by Apple. All web application servers offer the ability to connect web servers to multiple data sources and other services. They vary in their range of available features, robustness, scalability, compatibility with other web and database tools, and extent of the development environment.

15.3 *Extensible Markup Language (XML)*

Extensible Markup Language (XML) is a meta-language used to represent and manipulate data elements. XML is designed to facilitate the exchange of structured documents, such as orders and invoices, over the Internet. The World Wide Web Consortium (W3C) published the first XML 1.0 standard definition in 1998, setting the stage for giving XML the real-world appeal of being a true vendor-independent platform. It is not surprising that XML has rapidly become the data exchange standard for e-commerce applications.

* XML is not a new version or replacement for HTML. XML is concerned with the description and representation of the data, rather than the way the data is displayed. XML provides the semantics that facilitate the sharing, exchange, and manipulation of structured documents over organizational boundaries. XML and HTML perform complementary functions rather than overlapping functions. Extensible Hypertext Markup Language (XHTML) is the next generation of HTML based on the XML framework. The XHTML specification expands the HTML standard to include XML features. Although it is more powerful than HTML, XHTML requires strict adherence to syntax requirements