Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations   1. Thesis and Dissertation Collection, all items

1997

Database access from the Web

Dean, Andrew S.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/31929

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DATABASE ACCESS FROM THE WEB
by

Andrew. S. Dean

March, 1997

Thesis Advisor:	C. Thomas Wu
Thesis Co-Advisor:	Monique P. Fargues

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE DATABASE ACCESS FROM THE WEB	5. FUNDING NUMBERS
6. AUTHOR A. S. Dean	

7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
    The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)
    Determining the best method for granting World Wide Web (Web) users access to remote relational databases is difficult. Choosing the best supporting Web/database link method for implementation requires an in-depth understanding of the methods available and the relationship between the link designer's goals and the underlying issues of Performance and Functionality, Cost, Development Time and Ease, Serviceability, Flexibility and Openness, Security, State and Session.

    This thesis examined existing methods for enabling Web-client access to remote relational databases and found that most fall within the general categories of Common Gateway Interface scripts, Server Application Programming Interfaces, Web-enabled Database Management Systems exporting query results in Hypertext Markup Language, and independent client-based processes such as Java applets. To determine the best database access category we compared each one to the underlying link issues and conducted a case study for the IEEE Signal Processing Society.

    The results of this thesis are: (1) a taxonomy of existing Web\database linking methods, (2) a thorough listing and examination of the underlying issues as they relate to each link method, (3) recommendation and specification of the proper link method and hardware/software support system for the case study linkage problem.

14. SUBJECT TERMS Internet, Relational Database, World Wide Web			15. NUMBER OF PAGES 130
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

DATABASE ACCESS FROM THE WEB

A. S. Dean

Lieutenant Commander, United States Navy

B.B.A., University of Texas, 1982

Submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1997

Author: _____

A. S. Dean

Approved by: _____

C. Thomas Wu, Thesis Advisor

_____

Monique P. Fargues, Thesis Co-Advisor

_____

Ted Lewis, Chairman

Department of Computer Science

iii

ABSTRACT

Determining the best method for granting World Wide Web (Web) users access to remote relational databases is difficult. Choosing the best supporting Web/database link method for implementation requires an in-depth understanding of the methods available and the relationship between the link designer's goals and the underlying issues of Performance and Functionality, Cost, Development Time and Ease, Serviceability, Flexibility and Openness, Security, State and Session.

This thesis examined existing methods for enabling Web-client access to remote relational databases and found that most fall within the general categories of Common Gateway Interface scripts, Server Application Programming Interfaces, Web-enabled Database Management Systems exporting query results in Hypertext Markup Language, and independent client-based processes such as Java applets. To determine the best database access category we compared each one to the underlying link issues and conducted a case study for the IEEE Signal Processing Society.

The results of this thesis are: (1) a taxonomy of existing Web\database linking methods, (2) a thorough listing and examination of the underlying issues as they relate to each link method, (3) recommendation and specification of the proper link method and hardware/software support system for the case study linkage problem.

TABLE OF CONTENTS

I. INTRODUCTION

A.   OBJECTIVES AND RESEARCH QUESTIONS

The main objective of this thesis is to exam the issues surrounding linkage of a *database management system* (DBMS) to the World Wide Web (Web) in support of dynamic data access. All DBMS's in the DBMS/Web linkage discussion are assumed to follow the relational model (still the most widely used)[Plain96] unless otherwise stated. Morever, only currently available and utilized methods, protocols and standards are examined and thus, those links requiring radical changes to the Web or the DBMS's themselves are not covered.

Secondary objectives supporting this thesis include an investigation, analysis, and creation of a taxonomy of existing dynamic DBMS linking methods, a discussion of related issues and tradeoffs, and the design of a solution to a prototypical case study. The case study offers the opportunity to further investigate the alternative link methods, weigh them against the related connectivity issues, and illustrate the decision processes relevant to designing and implementing a Web/DBMS link.

B.   THESIS ORGANIZATION

Chapter II introduces the database and Internet concepts which serve as the foundation to the discussions in this thesis. Chapter III surveys common Web/DBMS link software components and discusses how they work together in building links. Chapter IV lists link related issues and analyzes required tradeoffs between them which underlie every link approach. Chapter V introduces and analyzes a case study while Chapter VI redefines the case study problems in terms of user needs and functional requirements. The chapter

1

then discusses the potential use of the specific linkage approaches previously compared and concludes by outlining the actual design for implementing the most appropriate of the potential approaches. Chapter VII contains the thesis summary and conclusion, an analysis of data obtained, and thoughts for future work and potential applications. Appendices summarize two separate discussions between the thesis author and case study personnel.

II. BACKGROUND

A.   A HISTORY OF DATABASE MANAGEMENT SYSTEMS

The term *database* [Rob95] refers to both the electronic storage of data (raw facts) in such data structures as linked lists, trees and hash tables, and the associated *meta-data* or data about the raw facts. A DBMS [Rob95] includes the database plus the combination of programs controlling access and allowing manipulation of the data. Generally, DBMS's are categorized by the functions supported and the architecture with which they maintain data relationships. Functionally, DBMS's range from the local, stand-alone DBMS's found on personal computers and mainframes to the distributed, shared DBMS's supporting the large numbers of multiple users and simultaneous queries commonly associated with large data processing applications.

1.   Evolution of DBMS Architectures

Beginning with the use of flat-files in the 1950's and continuing with the relational and object-oriented DBMS's of today, DBMS architects have sought to provide robust solutions to ubiquitous data storage and manipulation needs [Burleson94]. As time passed, each subsequent method brought additional functionality and solutions to the problems or constraints associated with its predecessors.

a.   *Flat-files*

Flat-file systems were the original DBMS [Burleson94]. Primitive direct data access methods used by these early systems included simplistic sequential searching of sorted

3

listings or, at best, hashing algorithms that mapped files to specific disk or tape storage addresses. Problems associated with these methods included an inability to recognize, establish, or manage data relationships, the development of "information islands" within organizations as different departments stored duplicate information in different types of file structures written in different languages, and the tight coupling of data and programs that always led to maintenance, concurrency, and recovery problems.

b.   *The Relational Model*

To address flat-file system problems, Dr. E. F. Codd, a researcher at IBM in the 1960's, developed the relational database model in which data resides in two-dimensional tables representing *entities* such as Customers, Inventory, Airplanes, etc [Burleson94]. Each row, or *tuple,* in the table holds data pertaining to one particular entity instance (Susy, hammer, B-52, etc). Additionally, one column in the table contains entity-unique data called a *primary key* that has two purposes. First, it serves to uniquely identify specific tuples within the table, or; secondly, when the primary key column is redundant to two or more tables, it provides links, or *relationships,* between the tables that can be navigated and manipulated with ease.

Improvements offered by relational databases included a simple conceptual framework, data independence in which data relationships were no longer hard-linked by the supporting data structures, control of relationship cardinality via the enforcement of constraints upon addition and deletion of data, and declarative data access utilizing relational set theory encapsulated within the mechanics of *structured query language* (SQL). SQL

initially offered only three classes of operators [Burleson94] SELECT, which shrunk the tables vertically by eliminating rows; PROJECT, which shrunk the table horizontally by removing columns; and JOIN, which dynamically linked two tables together via common column values.

Problems associated with the relational model include the inability to store *business rules* that constrain and affect the behavior of data, and persistent difficulties in handling recursive relationships.

c.   *The Object Model*

The Object Model of the 1980's and 1990's seeks to address the relational model's drawbacks and, by utilizing the properties of encapsulation, abstraction, and polymorphism, to more effectively model and replicate relationships in the real world [Burleson94]. By storing the behavior of the data within the DBMS alongside the data and their relationships, object technology promises to be less maintenance-intensive since all changes to data attributes and behaviors are handled within the DBMS instead of by the programmer.

2.   Interconnecting DBMS's

As the amount of stored data increased, the need arose for different entities in different geographic locations to share data in a collaborative fashion. It was at this point that the limitations of monolithic DBMS's became apparent, leading DBMS architects to create the initial distributed DBMS's of the 1970's which later evolved into the inter-connected networks of distributed DBMS's in use today.

a. Essentials of a Distributed DBMS

All successfully interconnected DBMS's share certain characteristics. At the highest level, distributed DBMS's should easily handle geographic dispersion and heterogeneous hardware/software. To be ideal they must also include the following traits [Burleson94]:

- *local autonomy,* meaning all data in the distributed network is owned and managed on a local basis independent of the overall system.

- *no reliance upon a central site*, i.e, all sites are equally "remote" with no one site having governing authority over another's data dictionaries or security.

- *continuous operation* supporting seamless, around-the-clock access.

- *location independence,* since end-users don't care or need to know about the physical location of DBMS's comprising the system.

- *hardware/operating system independence* with no queries tied to specific proprietary systems.

- *network independence* where network protocols are invisible to the end-user and differences in topologies are accommodated.

- *DBMS independence* allowing non-DBMS-specific retrieval and update operations.

- *data integrity management* of concurrent updates, ensuring the ability to handle transaction-type computing.

b. Client/Server

Because of the similarity between desired traits for distributed DBMS's and those of the client/server paradigm, today's distributed DBMS products predominantly use client/server [Orfali94] to link decoupled entities just long enough for them to accomplish pre-specified units of work. *Client/server* (Figure 1) is predicated upon two independent and autonomous processes working together (usually over a network) [Orfali94]; *client* processes which request specific services, and *server* processes which respond to the requests.
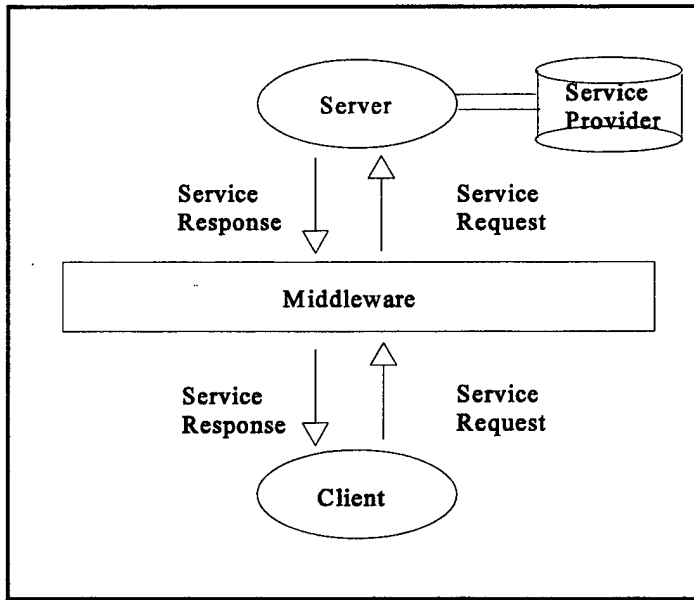
Figure 1. Client/server architecture

Client/server seeks to optimally distribute these processing activities via many-to-many relationships over different computer platforms, using the best characteristics of each platform to the most advantageous end. The anticipated benefit of client/server is the ability to abstract hardware and software concerns and focus on developing and building user-friendly, cost-effective systems.

All client/server systems have the following distinguishing characteristics [Orfali94]:

- *service*: client/server primarily involves a two-process relationship with the server on one machine providing services to the client which is (most probably) running on a separate machine.

- *shared resources*: one server can service many clients at the same time and regulate their access to shared resources.

- *asymmetrical protocols*: clients always initiate the many-to-one relationship by requesting a service from the passively waiting server.

- *transparency of location*: the server process can reside on either the same machine as the client or on a different machine across the network.

- *message-based exchanges*: clients and servers interact via a loosely-coupled message passing mechanism.

- *encapsulation and integrity*: the server uses centrally-maintained code to perform the services in a manner transparent to the client.

- *scalability*: client/server systems can be scaled horizontally by adding new clients, and vertically by migrating to newer, faster and more machines.

- *tier*: client/server implementations can be classified by the location of the split between user interface, application logic, and data repository functions.

8

c.   Tier

The three basic client/server functions of data reservoir, user interface, and application logic are predominantly classified and divisible into two or three tiers [Finkelstein96]. *Two-tier* client/server systems are recognized by the division of the three DBMS's functions between the client and only one server. In other words, all logic not actually on the client is co-located on the server. Two-tier implementations are themselves divisible into fat and thin client implementations [Orfali94], with *fat clients* holding both presentation logic and application logic and *thin clients* simply having presentation logic alone.

With *three-tier* implementations a thin client presents DBMS interaction results while residing separately are the data reservoir on a data server and the application logic on the application server. The benefits of spreading the three functions among multiple tiers include [Centura96] faster performance because complex application logic can be executed on more powerful machines, and the improved ability to support end users by adding additional software or hardware to improve performance in those sites sustaining large numbers of simultaneous queries.

B.   EMERGENCE OF THE WORLD WIDE WEB

The *World Wide Web*, also known as the Web or WWW [Marshall95], is a client/server implementation comprised of all the inter-networked (*sic* Internet) software and hardware information-retrieval components required to for around-the-clock access by anybody to any available information, located anywhere in the world. The WWW was

originally developed at CERN in March 1989 to facilitate information dissemination and sharing between collaborative groups across the world. At that time most users worked for the government or large universities and information was available primarily via anonymous *File Transfer Protocol* (FTP) using a simple line-mode browser. In January of 1993, access was made much easier with release of the alpha version of the X-MOSAIC browser and WWW use exploded to the high levels evident today.

1.   Web Architecture

As previously stated, the architecture of the Web simply represents another step in the evolution of client/server technology. Distinguishing features of the Web client/server version include the extreme thinness of clients utilized simply to render documents for viewer consumption and the sheer number and variance in platforms participating in the network. Evident from the beginning, though, is that many of the Web's underlying technologies directly mirror the desired traits for distributed DBMS implementations.

a.   Client/User-Interface Architecture

From the user point of view the Web is a series of documents written in *hyper-text markup language* (HTML) [Schulzrinne96]. A Web document, also known as an HTML document, consists of special "tags" embedded in the document text which direct the user interface application in rendering the document into usable format.

At the client end is the *browser*, a universally supported, ultra-thin user interface that supports the user view of the Web as a series of pages to be quickly rendered

into a digestible format. A special attraction of the browser is that it is easily and inexpensively deployed to very large numbers of users.

Web pages are connected in many-to-many relationships by *universal resource locator* (URL) [Schulzrinne96] specified *hyperlinks* exemplified by the fictional URL "**http://www.nps.navy.mil/~asdean/index.html.**" A hyperlink contains all data necessary for the browser to locate and render the document since it indicates the connection method (*hyper-text transfer protocol, e.g.* **http**), the server's name (**www.nps.navy.mil**), the resource's name (**~asdean/index.html**) and the resource data type (**.html**). Resources named "index.html" usually indicate the *homepage* [Lynnworth96], a term referring to a Web site's initial visitor starting page that includes welcoming data and hyperlinks leading to other pages located at the site or elsewhere in the Web.

b. Server/System Architecture

From the system viewpoint the Web consists of a world-wide network utilized to briefly connect clients to multitudes of servers, each of which is holding HTML documents made available for access. The network supports Web client/server implementations by establishing a connection between the requesting application (browser) and the responding application (Web server) when users click upon URL's. During the connection, the server receives information from the browser regarding connection method (FTP, HTTP, etc.), resource URL code, resource data type, and client-specific browser type/model data. It then responds to the request with a protocol version status line, resource

11

access success/error code, relevant entity meta-information and the appropriate HTML resource, and then closes the connection.

2. Standard Internet Protocols

In the Internet protocol hierarchy [Stallings94] shown in Figure 2, a client request flows down from the browser application to the *application* layer, then on to the *transport* and *Internet* layers till it reaches the *network access* layer just prior to crossing the physical infrastructure.

To ensure client requests work properly, the browser must work with the host machine's operating system and connection hardware to ensure client requests are in proper format and to correctly process each server response.

HTTP (1.0) [Schulzrinne96], as the application layer protocol, determines the beginning, duration, and ending of a single communication session. For this reason, a solid understanding of HTTP's stateless nature and its ultimate effect on Web communications is absolutely vital to this discussion of Web/database connectivity solutions. An analogy for *stateless* communications would be a telephone conversation where the called party hangs up after answering each of the caller's questions, and when called again, has no recollection of previous answers and conversations. Because of statelessness, each client/server session must stand alone as a complete entity unless special measures are taken to retain records of the transaction somewhere in the loop.
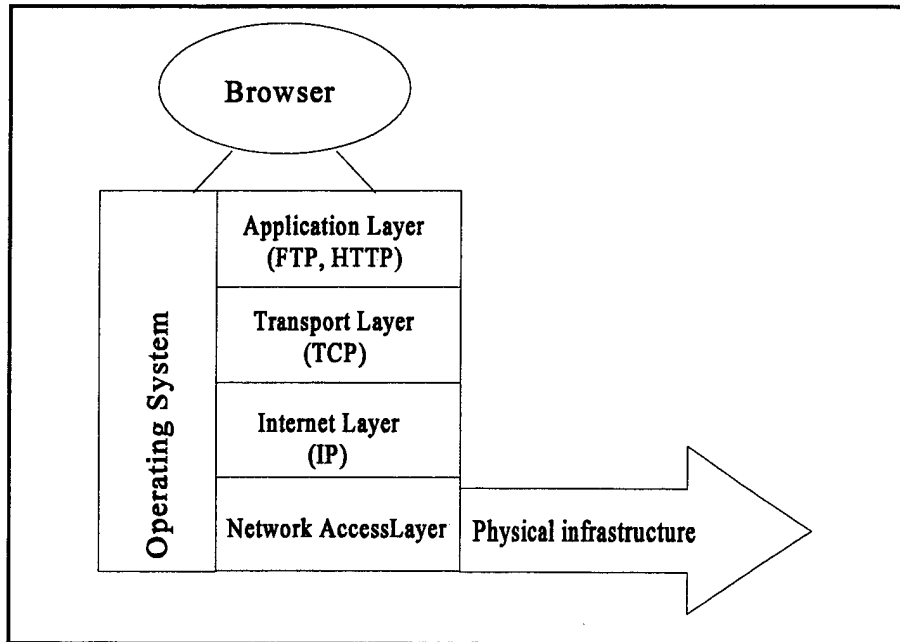
Figure 2. Internet Protocol Hierarchy

Below the application layer, and above the network card or modem, is the Transmission Control Protocol/Internet Protocol (TCP/IP) [Jordan94] transportation layer and the Internet layer that work together to facilitate all WWW traffic by abstracting individual network hardware protocols. TCP/IP is characterized by *connection-less* (no-dedicated conduit) communications where each message is composed of multiple uniformly-sized packets of data, each of which making its way alone to and from the server. When the packets arrive and are re-assembled at the server, they allow reconstruction of the original message. This process works the same in reverse when the server application sends its response back to the client.

3. Current Web Status

Estimates of current Web usage vary, but Informix Corporation [Informix96]holds that there will be over 200 million Web users by the year 2000 and that the Web-related software market will grow from $260 million in 1995 to $4 billion by the end of 1996, a sixteen-fold increase. Along with this explosive growth in use is a corresponding revolution in content since Web pages have changed from static representations of archived files to extravagant multimedia productions. These newer sites allow intuitive user navigation of Web sites and offer such innovative mixed-media combinations of color, sound, and animation that sites not providing this level of service tend to be avoided [Informix96].

C. CONVERGENCE OF DBMS's AND THE WEB

DBMS products have now demonstrated the capability of managing mixed-media in a collaborative workflow environment [Frank95]. Vendors are selling *rapid application development* (RAD) tools incorporating pre-written question/answer "wizards" that abstract underlying technical details and guide developers through the development process. These tools, which are common for either monolithic or LAN-based client/server environments, have not completely solved all the problems associated with the uneasy linkage of proprietary versions of SQL and other protocols and so they limit DBMS's ability to cheaply scale up to the inter-networked level of distributed computing. It is because of this that DBMS's need to find an easy method of disseminating rich content on a massive scale without the high costs associated with proprietary solutions.

From the Web perspective, it is content that is king, and site builders have irrevocably moved from presenting static text to allowing the user to interact with rich, dynamic and fluid multimedia sites. *Interactive* [Stormcloud96] in this sense means collecting and processing user input that is more complex than simply choosing from among a set of static links. Throughout the world and throughout the day, a growing number of users demand interactive access with photographs, graphics, text, audio and video, all of which must be archived for real-time retrieval and editing. The problem with this demand is that traditional HTTP servers are incapable of supporting this level and type of interaction [Rowe96], let alone facilitating quick, easy, and innovative customization of content.

Given the current and projected levels and methods of Web use, the marriage of DBMS's and the Web is a natural one. The Web is a compelling platform for database-driven interactive applications because it offers platform independence and instant global access along with the possibility of significantly lowering development, deployment, training, maintenance, and upgrade costs. Concurrent with DBMS products handling new data-types and developing new distributed processing capability is the Web-driven demand for increasing quantities of these new data-types along with new, richer ways to manage and interact with them.

The bottom line is that DBMS's need for low-cost, widespread connectivity and the Web's need for robust tools to manage dynamic interaction with multimedia content has caused the two technologies to converge. Key to this convergence is discovering and

15

creating a link between the two that leverages their respective strengths without unduly

restricting each other's ability to address and resolve important issues.

III. COMPONENTS IN THE WEB/DBMS LINK

A. INTRODUCTION

A Web/DBMS link *component* is any identifiable (e.g., self-contained and/or standalone) software program utilized to dynamically generate data. These components, though wide-ranging in complexity and purpose, all have the common goal of enabling Web/DBMS linkage.

This chapter begins with an explanation of the basic data access cycle common to all approaches and an introduction to a fictional database that will be used throughout the chapter for illustration purposes. The following sections explore a taxonomy of components commonly used to build Web/DBMS links. The taxonomy is grouped into client, server and middleware categories and we will provide a brief discussion of the mechanics of each one. We conclude the chapter with an introduction to the main Web/DBMS approach categories that we further discuss in Chapter III when the issues surrounding Web/DBMS links are more fully explored.

1. The Data Access Cycle

All Web/DBMS links have the common data access cycle since, regardless of which software products are used to build a Web/DBMS link, all approaches fundamentally rely upon a message-based form of interaction called *remote procedure calls* (RPC) [Orfali94] for data access. As demonstrated in (Figure 3), on the calling (client) side of the RPC is the browser serving as the *graphical user interface* (GUI) that accepts user inputs and renders

HTML documents. Each remote call commences when the user clicks on a URL displayed by the browser, opens a connection with the remote server site via middleware communication processes, and sends a request for data.

The remote server, which listens passively for client connection calls and data requests, completes the cycle by parsing, interpreting and acting upon the request, encoding the response in HTML for browser rendering, sending the results back to the client via middleware, and finally closing the connection.
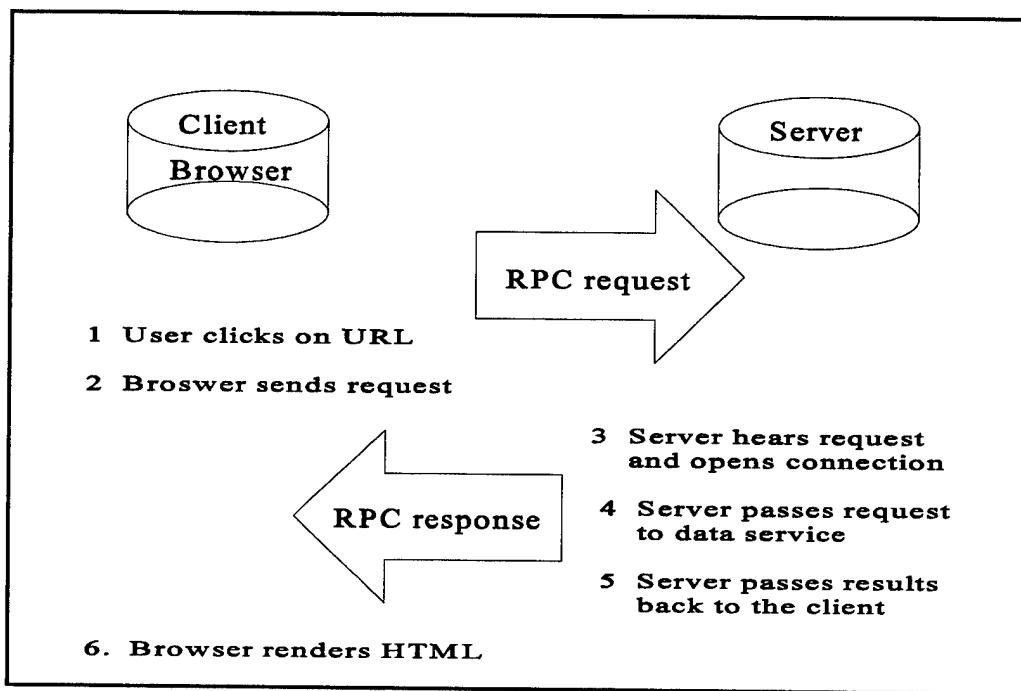
Figure 3.   Data Access Cycle

2.   Example Database

To illustrate how respective component characteristics and mechanics are used to execute the data cycle, a relational database comprised of two tables will be used throughout

this chapter. The database pertains to a fictional campus food stand called "THAI SHACK" that is owned and operated by its owner "Ralph Thai." Ralph manages his menu and customer lists with a Web-linked database containing "MENU" (Table 1) which tracks menu-item ingredients, and "CUSTOMERS" (Table 2) used to track customer menu preferences. For both tables the primary key is "Name."

Name (key)	Gender	Graduation-Date	Favorite
Suzy	Female	Mar 1997	Tiger Cry
Dewey	Male	Oct 1998	Spicy Soup
Gidget	Female	Dec 1998	Red Curry

Table 1. CUSTOMERS

Name (key)	Meat	Coconut	Hot Peppers
Tiger Cry	Beef	No	Yes
Spicy Soup	Chicken	Yes	Yes
Red Curry	Chicken	No	No

Table 2. MENU

B.   CLIENT COMPONENTS

1.   Browsers

In the simplest method of Web/database interaction the tasks of a "standard" Web browser are limited and the data access cycle begins with the user clicking on a URL to render fill-in-the-blank "forms." The forms capture the user input for such pre-fabricated database queries as the following SQL pseudo-code:

19

```
SELECT      Names
FROM        CUSTOMERS
WHERE       Favorite = "X"
```

The cycle continues when the user selects a specific Favorite (for example "Red Curry"), and clicks on a submit button to send the data request to the server. At the server the "X" is replaced with the selected favorite, the query is executed, and the results are formatted in HTML. The cycle is complete when the browser renders the HTML-formatted data returned by the server.

Advantages of using standard browsers include simple, standardized functionality and universal Web server support without the additional costs associated with proprietary software. A major limitation in relying upon standard browsers (as opposed to fat clients, discussed next) is the dependence upon the pre-fabricated queries stored on the server, and the large number of network accesses required to receive client input, return error messages for incomplete or inaccurate entries, and then respond again with the correct query results.

a.   *Plug-ins and Fat-clients*

In an attempt to limit the number of network accesses, tailor the query process for each user, and reduce server workload, some developers have created variations on the standard browser. These variations, called browser plug-ins, rely upon the browser knowing about a server-side database's existence and interacting with it in some proprietary fashion to fetch and interpret results on the client side. *Plug-in's* [Netscape96] are software modules that extend the standard browser's capabilities. Plug-in's accomplish such tasks as guiding the user through the query process or performing rudimentary business logic with respect to

20

form completion. They also ensure data type/query language compliance prior to actually submitting the request to the network. An example would be if a THAI-SHACK database user input "Susy" in a fill-in form field supporting a query based upon Graduation Date. The plug-in would know that "Susy" is not an alphanumeric and could provide immediate feedback to prevent the erroneous database query from crossing the network.

Fat-clients take browser variations to the extreme and are exemplified by the classical client/server distributed DBMS, whereupon the DBMS runs on the client side and needs no WWW support other than to provide a connection to the server-side data reservoir [Orfali94]. An example of this situation is when an entire query similar to the following SQL pseudo-code:

```
SELECT      Names
FROM        MENU
WHERE       Meat = "CHICKEN"
```

is created on the client side, checked for syntactical and data-type errors, and then a fetch routine listing specific tuples to be gathered is sent across the network to the server-side data reservoir. These fat-clients, or *database front-ends*, usually require proprietary licensing agreements and are restricted in their capability to work with anything other than vendor-specific products except for lowest common denominator middleware products [DataRamp96] like ODBC (discussed in the middleware section of this chapter). The advantage to using fat-clients is that DBMS/user interaction no longer depends upon pre-formatted queries and the supported functionality approaches that of a stand-alone database on a PC. The disadvantage with fat clients is that they routinely cost more, are problematic

with respect to upgrades and maintenance due to wide dispersion of multiple copies. Additionally, fat clients can, require users in some cases to have a high degree of familiarity with database specifics and query languages.

b.   *Client Record Storage*

Some browsers incorporate the ability to store records in client permanent memory. When users click on a URL, downloaded pages are stored (cached) permanently in client memory so subsequent accesses only require the downloading of updates. The usefulness of caching is inversely proportional to the richness and interactive/dynamic nature of page content since, unlike static text pages, those pages containing dynamically generated information such as links to other URL's or multimedia files that were generated in response to queries do not readily lend themselves to caching. This is true because the data that was gathered and used to build the page may have been changed or deleted subsequent to the last access. Utilizing the example database, the results of the SQL pseudo-code:

```
SELECT      Names
FROM        CUSTOMERS
WHERE       Favorite = "Red Curry"
```

once cached would not change from one page load to another despite database entries which may have changed relevant information held at the server.

To counteract the static nature of caching, browsers can store, along with the page itself, another record type commonly known by the Netscape Corporation's term "cookie." *Cookies* [Netscape96-2] record such data as time stamps or other indications of previous user input, and whenever the users make subsequent requests, the information

written in the cookie is transparently sent to the server to facilitate interactive responses. With respect to the example database, the browser could compare the date and time of cached Red Curry query results against a server-supplied time of last database update and, if cached results have been superseded, new results could be generated. This guarantees that users are always presented with information reflecting the most recent database state.

2. Independent Processes

Essentially a hybrid plug-in, *independent processes* are client-resident entities working within the browser to dramatically extend client functionality in the Web/DBMS link. Despite their dependence on browser hosting, these processes are considered independent since they are self-contained and not part of the browser itself.

a. *Java Applets*

Created by Sun Microsystems, Java applets [Hamilton96] are small, self-contained, object-oriented, platform-independent, multi-threaded processes downloaded by the client browser to deploy application logic services on the client side of the connection. Applets come over the Internet in partially-compiled Java-specific binary code that is fully interpreted upon download so they can immediately start running on a virtual machine inside the browser. The *virtual machine* [Hamilton96] is a software process residing inside the browser application's memory space that emulates the instruction set of specific computer hardware. Since the Java byte code is interpreted instead of compiled, and the virtual machine is a plug-in, they offer the benefit of complete portability. In other words, the

specific type and instruction set of client hardware is irrelevant since the applets run on the virtual machine inside the browser instead of on the client host.

Applet hosting on the client computer can solve many of static HTML's limitations. This is especially true for database applications since applets enable database servers to concentrate on serving data while the independent process authenticates users, provides business logic such as data-range checking or form completion, and then obtains data from the DBMS on the server.

Returning to the example database, an applet would execute upon download to capture a user's data request and error-check such complex tailored queries as the following SQL pseudo-code:

```
SELECT     Names
FROM       CUSTOMERS
WHERE
           SELECT     *
           FROM       MENU
           WHERE      CUSTOMER.Favorite = MENU.Name
                      and
                      Meat = "Chicken"
                      and
                      Graduation Date's < "Red Curry"(erroneous input)
```

When the applet error-checks the request for data type and SQL syntax agreement, it notices that Red Curry is not a proper entry for the Graduation Date column. The applet then informs the user to make changes, and once Red Curry is changed to a proper date such as "Dec 1997," the applet builds a JOIN between the CUSTOMER and MENU

tables, sends the data request to the server, and aids the browser in rendering the returned output.

The primary benefits of applets are portability and the savings in lag-time associated with conducting all business and application logic prior to network transmission. The portability benefit comes at the potential expense of a 30-fold speed deficit to compiled processes [Yourdon96] since there is an additional layer of processing required to translate between virtual and actual host machines.

An additional downside associated with applets concerns security and the fear of malevolent processes potentially accessing client files and hardware. These possible security infractions include [Dean96] stealing passwords, copying or destroying files, spying on client hardware and software configurations, attacking flash BIOS, or copying more and larger malicious programs from Internet-connected computers.

b.   Active-X

Active-X [Murdoch96] is very similar to Java in concept and, as an evolutionary technology based on object linking and embedding (OLE) technology, is Microsoft's approach to thin-client Internet computing. On a Web page, Active-X can accomplish a wide variety of tasks and extend normal HTML coding by incorporating into Web applications small object-oriented programs called "custom controls" [Microsoft96]. These controls can seamlessly handle repetitive and/or complicated functions such as menus, scroll bars, buttons, pre-loaders, and timers. The true power of these controls, though, comes from their ability to perform traditional server-side functions by downloading at runtime

25

functional modules from the network and contacting other hosts outside of the client/server link. Active-X is envisioned by Microsoft [Murdoch96] to be a method of standardizing web site development and making data available transparently no matter where it is located. On the downside, there is a strong security concern with Active-X that mirrors the concerns with Java.

C. SERVER COMPONENTS

1. The Relational Database

The relational database is the basis for most Web/DBMS links [Plain96] and it remains fundamentally the same regardless of whether it is a stand-alone or Web-connected application. Web/DBMS link applications differ in what the relational database can offer, and it is the implementation of SQL within the application that determines robustness of query requests and such additional functions as protection of data integrity, user authentication, or limited data access based upon multiple security levels.

a. Extended HTML/Embedded SQL

Some database applications utilize Web pages seeded with special HTML tags and the SQL code required to conduct pre-formatted database queries and create the corresponding responsive Web page [Frank96]. When a data request is received by the HTTP server, it utilizes a special program or set of macros to map specific extended-HTML form fields to corresponding place holders located within an SQL statement. The same program then submits the SQL query to the database server, formats the results in HTML, and releases the results to the Web server.

26

With embedded SQL [Frank96], it is the query itself and form fields that are sent by the browser. A special server-side parsing application strips an embedded SQL query and related data from the browser-generated input stream, executes the query, merges the results with the original or stored Web page, and then passes the output back to the Web server for HTML encoding and return to the client. Almost invariably the use of embedded SQL code requires extensions to standard HTML tags to signal the HTTP Web server that a database query must be conducted.

With respect to THAI SHACK, a query similar to the following SQL pseudo-code:

```
SELECT      Names
FROM        X
WHERE       Y = "Z"
```

would be stored in the server-side DBMS. The user would be offered the opportunity to see and select food made with different items, and if they selected "Coconut," then X becomes MENU, Y becomes Coconut and Z becomes Yes. These entries are swapped into the correct place in the query statement, the query is conducted, and the results formatted into HTML and passed to the HTTP server for return to the client.

Although various HTML extension and embedded SQL codes implementations are proprietary, and therefore limit the openness of the total solution, they are useful in customizing page content in response to a user's identifier or computer Internet address captured by the Web server [Gundavaram96]. Additionally, they perform faster than applications requiring the execution of more complicated methods and can be easier for

developers to use since they dispense with requirements for writing, compiling, and maintaining multiple programs.

One note of caution with the use of HTML extensions is that Web pages containing proprietary tags may not be properly displayed by some browsers [McGee96]. This means users may miss some of the content, since only those pages conforming to the standards issued by the WWW Consortium (and do not contain proprietary extensions) can be universally read.

b.   Templates

Another way to lighten developer workload is to reduce the amount of time spent writing the HTML code used to format database output. Some DBMS products are "Web enabled," meaning both data and the programs for embedding HTML code into the data output are located within the database [Dobson95]. In operation, when the browser calls the HTTP server with its request, a stored program is actually called instead of a specific HTML page. The stored program executes a database query, reads in a generic template serving as the basis for the Web page, and populates areas marked by predefined tags with values retrieved from the query. Once the program has completed the request query and HTML formatting, it sends it to the HTTP Web server for shipment back to the browser.

In the case of templates, the results of the previous Coconut query are individually mapped to specifically-marked places on pre-formatted pages stored in their entirety within the database. This allows additional flexibility in page presentation since a

query that results from inputting Yes under Coconut, for example, could map to a page with a palm tree motif, while the results of No under Coconut would not.

Advantages of Web-enabled databases are the familiarity of the development environment, bypassing the use of scripting languages (discussed in the following section), and the simplification of web page organization since the web page logic, graphics, HTML encoder, and other resources are all stored together. The primary disadvantage of templates are their proprietary nature.

2. Database Gateways

The function of the basic HTTP server is to listen for incoming connection requests and either return a file or pass the request on to another application [Rowe96]. Since database queries are not included in these rudimentary services, developers have created interface programs to stand between the HTTP server and the database and facilitate the enhanced translation and communication capabilities. Approaches to these interface programs range from open interface standards called *gateways* to proprietary implementations known as *application program interfaces* (API's).

a. Common Gateway Interface

Common Gateway Interface (CGI), called the "poor man's client/server" [Varney96], is an open interface standard enabling clients to initiate and pass information to server-based programs that generate responsive information in the form of dynamically created Web pages [Gundavaram96]. CGI's key service is providing client access to information that would otherwise be unreachable without the Web server running these

external programs. With a simple file, the Web server's job is to respond to the http request by returning a file and when CGI is involved, its additional responsibility is to generate the file's content dynamically. As illustrated in Figure 4, clicking on the URL hyperlink executes a stored CGI program and begins the access process.

BROWSER	Input data	SERVER	Input data	CGI PROCESS
Accept data input Submit to server	▷	Fork CGI process Forward data input	▷	Parse input Process input
Display HTML	HTML ◁	Send HTML	HTML ◁	Create HTML Return to server
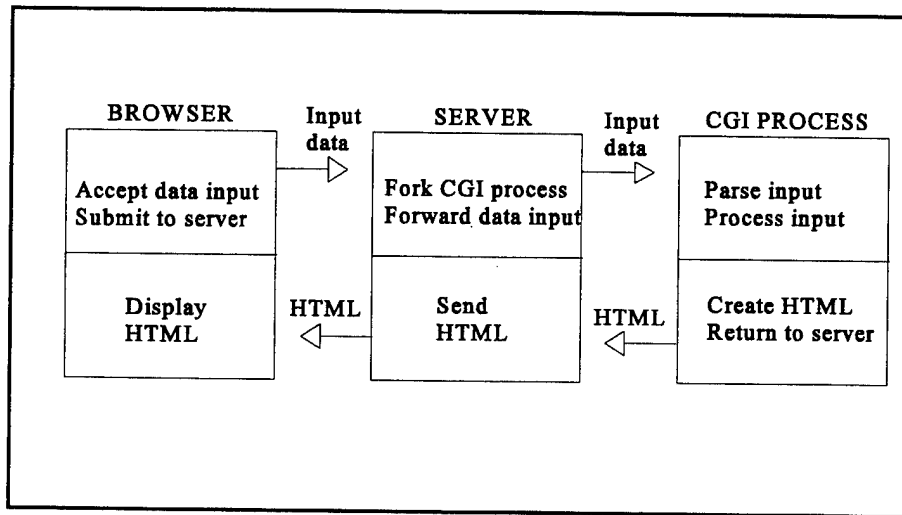
Figure 4. The CGI Process

It is the Web server itself that runs the program by forking a new process, passing client-submitted data to the new process, giving up CPU cycles to the new process, formatting the returned results in HTML, sending results to the client and then killing the process [Gundavaram96]. CGI is *single threaded,* meaning servers must create a separate process for each request received, with increased numbers of requests leading to increased numbers of processes.

Sites utilizing CGI must provide guidance to the HTTP server and differentiate between static pages and those generated dynamically. One convention is to have static pages end with the ".html" suffix and dynamic pages end with ".cgi." so that a site

would have a static homepage provide welcoming information, site "how-to's," and form

fill-ins along with links to the other pages at the site offering more robust interaction.

In the example case, a user desiring to "see the names of female customers

who eat beef" would utilize the static homepage "THAI-SHACK.html" to access a

dynamically generated page called "THAI-SHACK-PROFILE.cgi." Once the user completes

the fill-in form and clicks on "Submit," the browser assigns the values Gender=Female and

Meat=Beef, sending the query input to the HTTP server which in turn spawns a CGI process

and hands off the input. The CGI process would generate a query similar to the following

SQL pseudo-code:

```
SELECT      Names
FROM        CUSTOMERS
WHERE

            SELECT *
            FROM        MENU
            WHERE       CUSTOMER.Favorite = MENU.Name
                        and
                        Gender = "Female"
```

Query results are then formatted into HTML by the CGI process and passed back to the

HTTP Web server for transmission to the client.

One benefit of using CGI is the large base of pre-written programs available

for free from the Internet and other sources. Other benefits are extreme flexibility in

providing page content, almost universal support among commercial servers (ergo, highly

portable), and compatibility with almost any database application language. In fact, CGI

programs can be written in any language that can create an executable for the given operating

31

system [Lynnworth96] despite the fact that most CGI programs are UNIX based and therefore the CGI programs are normally Unix shell scripts or Perl scripts [Yager96]. These scripts are unique in that they are runtime "interpreted languages" as opposed to compile time "compiled languages" that generate binary executable files in the host machine's native hardware language code [Brenner96]. Care must be taken with respect to choice of languages since interpreted languages run more slowly than compiled languages and can thus slow server performance. Tediously created CGI applications can't handle rich data types easily [Gaffney96] and they require the programmer to code in both HTML and the supporting scripting language to make processes that contains the database application logic required to execute the query. These processes offer poor performance due to the continual forking of a new process with each request and the repeated database opening/closing that overtaxes server CPU cycles. CGI is adequate for simple queries but is unable to handle rigorous compound transactions where the results of one operation are required to determine the next one.

b. API's and DLL's

One way around the performance penalty of open gateways and interpreted languages is to use native code (proprietary to specific vendors) to provide translation services. This natively compiled code, or API, provides the fastest way to access a database via the Web [Rowe96] because it does not need a run-time interpreter and because, for each request, it can load and run much faster than interpreted code and remain in memory if desired. An API works by providing a library of function calls, usually written in C or C++

32

[Rowe96], that allow a link programmer to make function calls on the database without knowing the specifics of the database implementation.

An example API is the Microsoft *dynamic link library* (DLL). A DLL is one or more functions created from the ground up to provide run-time support to the server. DLL's are compiled, linked, and stored separately from the processes that use them and are loaded (once) into the same memory space as the server. When a user clicks on a URL that requires the DLL, the server first checks to see if it is already in memory. If not, it is loaded and used, but if it is so, then there is zero overhead associated with loading. Because of this co-location in memory, DLL's have access to all the resources available to the HTTP server and, since DLL's only need to be loaded once and they avoid the added overhead of repeated loadings or request calls to external scripts, DLL's gain a five to ten-fold increase in performance over CGI [Microsoft96-2].

Data access is accomplished in a manner similar to the CGI methodology discussed earlier.

3. Providing Security

Surrounding all data access methods is the need for security. This need transcends all components of the Web/DBMS link primarily because the Internet was originally created to support a benign network of mutually-supportive academic and government entities. Internet protocols are thus inherently insecure. Because of this, web servers and client hosts require Web/DBMS link components to additionally provide robust protection capabilities, but without overwhelming usability and performance requirements.

33

The most common security vulnerabilities associated with the Internet [Hughes96] include:

- *hazardous traffic*: meaning traffic associated with any attempt by unauthorized users to enter and harm the Web site.

- *lack of confidentiality*: the Internet protocols assume networked computers will only acknowledge traffic meant for them and ignore all other.

- *unidentified systems*: since there is no reliable manner to permanently bind a specific computer to a specific IP address, it is difficult to reliably identify networked computers.

- *unidentified users*: most user authentication schemes rely on a password that is passed in the clear and thus available for copying and re-use.

- *unidentified servers*: it is difficult to reliably identify the actual server performing an application service.

- *modified data*: because of the open nature of the Web, it is easy for malevolent entities to capture, copy and alter data packets prior to arrival at ultimate destination.

Most Web applications address this situation by using server-side components to prevent the unwanted release of personal data or unauthorized data manipulation both at the server and en route to or from the client.

a. Firewalls and Proxy Servers

For the Web site itself, the most common way to provide basic user authentication and authorization security procedures is with firewalls and specialized HTTP servers called proxy servers [Hughes95]. In their simplest form, *firewalls* are screening mechanisms that discriminate between valid and invalid users or processes depending upon passwords or client host Internet addresses. The successful [Hughes96] firewall utilizes the philosophy of "those not known to be a friend are considered foe and denied access."

Considering the example database, if Ralph wanted to ensure only he could make changes to MENU or CUSTOMERS, a firewall could be programmed to forbid access to anyone who did not access THAI-SHACK from Ralph's computer and type his password in correctly.

Proxy servers contain additional functionality extending basic firewall security by straddling the firewall to "focalize application traffic" [Hughes96] via processes that listen for and respond to client page requests coming over the Internet. Conventional use of a proxy server [Hughes95] would have it compare the user's identity to a list of pre-approved users with passwords, client TCP/IP addresses, or client host names against a pre-approved list on file in the server.

Once the authentication and authorization service has been completed, the proxy server forwards the request to the database application for processing and then returns the query results via the HTTP server to the client. Proxy servers can also be used to perform such other duties as caching heavily used documents that don't need to be created on-the-fly

or converting the database output into HTML, and supplementing audit capabilities by logging transactions as they occur.

D.   MIDDLEWARE COMPONENTS

As shown in Figure 1, facilitating the interaction between client and server processes is *middleware*, a vague term covering all distributed software such as communication stacks, distributed directories, authentication services, remote procedure calls, and queuing services [Orfali94]. Middleware is the "glue" that provides inter-operability between proprietary entity or process protocols that would otherwise be unable to communicate. What middleware does not include is the DBMS or other applications providing the actual service, the user interface or any logic that supports the application.

Middleware can be divided into the two main classes of *service-specific functions* which enable a particular client-requested service and *general functions* such as communication stacks, the RPC's, queuing services, and other services that comprise the enabling substrate for the service-specific functions [Orfali94].

Database middleware represents more than half of the entire middleware market [Freeman96] and can be either open or proprietary. Developers must therefore choose a particular middleware component based upon whether their link will be optimized for a specific application or standardized to provide "lowest common denominator" [Orfali94], or cross-vendor, access to multiple applications. In other words, to provide broad-based support middleware must forgo some of the performance and functionality enhancements that vendor-focused solutions are able to provide.

36

This section looks at the most prevalently utilized Web/DBMS link middleware components and discusses how they try to finesse the limitations of standard Internet middleware protocols and add the concepts of state and connection-oriented communications to database transactions.

1. Protecting Data En-route

If basic firewall authorization and authentication schemes don't provide enough security, then encrypting the data prior to passing it over the wire may be required. Server Sockets Layer (SSL) [Hughes95], developed by Netscape, is an encryption scheme that provides a secure client and server connection along with guaranteed message authenticity and server identity authentication services. What SSL does is scramble the data at the application layer prior to passing it down to the network layer. This renders any intercepted packet unreadable by unauthorized readers.

S-HTTP is a standardized form of the HTTP protocol [Hughes95] that also encrypts data above the network layer in order to guarantee data integrity, non-repudiation and user authentication. Working alone or together, SSL and S-HTTP and components like them are the most prevalent middleware method of protecting users.

2. Common Object Request Broker Architecture And Internet Inter-ORB Protocol

The basic Common Object Request Broker Architecture (CORBA) is a message-based middleware specification [Vinoski93] that provides high-level object-oriented support of applications inter-networking across diverse architectures and infrastructures. In

CORBA's case the high-level support is an additional abstraction layer superimposed over underlying network and application layer protocols. CORBA is implemented with client-side *proxies* and server-side *skeletons* via an "object-dot-method" nomenclature that allows a client object to call the procedures of an object located on the server. The server-based object responds by invoking the requested procedure and returning the results in CORBA specified format.

The Internet Inter-ORB Protocol (IIOP) [House96] addition to the CORBA specification offers the advantage of having Internet-based CORBA server objects retain state and bypass HTML and CGI. This offers the advantage of reducing HTTP server overhead and extending database application functionality but comes at the cost of some extra CPU cycles used for messaging formatting, data-copying and parameter marshaling [Schmidt96]. It has been suggested that this may limit CORBA IIOP's use in high-speed applications [House96], but for the average Internet connection, ease-of-use and data-type extensibility overcome any performance penalties.

3.   Open Database Connectivity

Some vendor-specific middleware protocols are rapidly becoming de-facto standards for Web/database connectivity. These proprietary "standards" are utilized primarily in a three-tier architecture and stand between the database and the API or gateway to enable cross-vendor and multi-DBMS access by Web servers.

Open Database Connectivity (ODBC), a proprietary (Microsoft Corporation) specification [Lambert96], is one standard that attempts to solve the problems associated

38

with accessing data from multiple proprietary databases in the heterogeneous Web/DBMS

environment. As illustrated in Figure 5, the ODBC architecture has four major components:
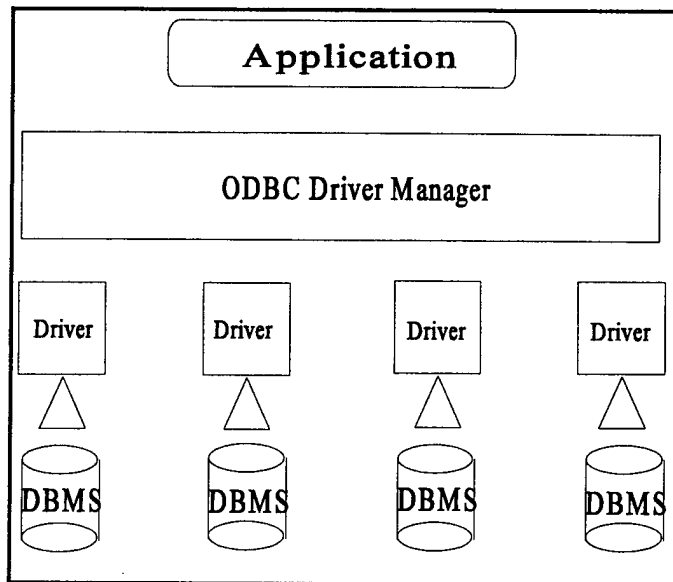
Figure 5.   ODBC Architecture

the *data source*, which can be almost any DBMS; the *driver*, which modifies application

requests to conform to data source syntax; the *driver manager*, which loads specific data

source drivers when needed; and the ODBC *application*, which performs functions required

for submitting queries and retrieving results.

Together these components provide a *call-level interface* (CLI) that allows Web site

developers to make calls in a generic database interface language and have ODBC translate

the call into specific database driver syntax.

ODBC problems abound, with the most serious one being that the standard is

controlled by Microsoft and thus constantly evolving. Furthermore, experience has shown

that the ODBC driver/layer interfaces are difficult to build and maintain, introduce a lot of

39

overhead to simple SQL queries, and can never be as fast as native API's that bypass the additional layers of any CLI [Orfali94].

4.   Java Database Connectivity

Similar to ODBC in nature, JAVA Database Connectivity (JDBC) is a specification and API of the JavaSoft division of Sun Microsystems [Linthicum96]. JDBC was built to give Java a database connection and also to be implemented as an additional layer on top of ODBC so it can work with ODBC-compliant databases without change. JDBC consists of a series of abstract Java interfaces addressing standard database query requirements and it uses a simple two-tier architecture (Figure 6) for applet-database connectivity in which applets executing within the browser environment connect back to a database server located somewhere else on the Internet.
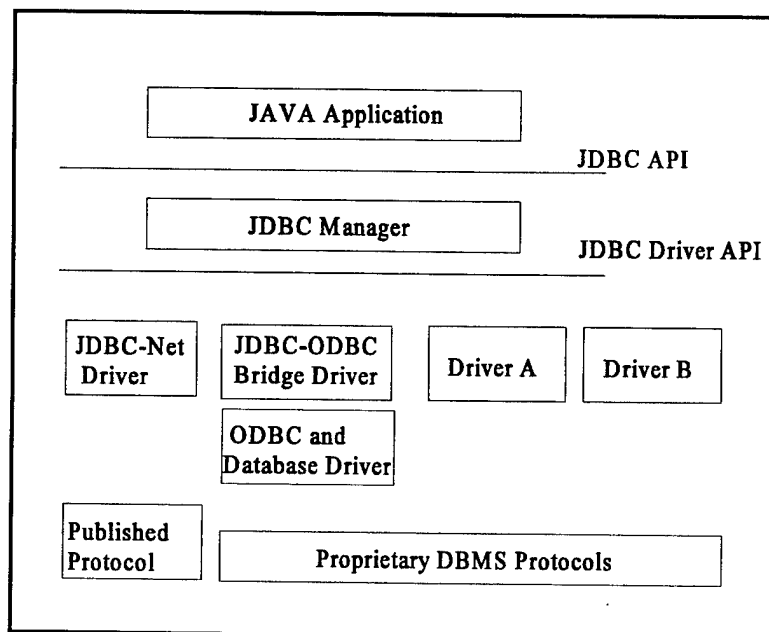
Figure 6.   JDBC Architecture

The JDBC API connects the Java application to the JDBC manager, which in turn uses the Driver API to connect the JDBC manager to either a JDBC-specific database driver or to the ODBC database driver via a JDBC-ODBC bridge. In addition to traditional two-tier implementations, JDBC also provides three-tier access to database servers in which an applet can call a middle tier service layer (such as a CGI or API process) to make the call to the database server. By sitting on top of ODBC, JDBC is well positioned to provide database-neutral communication services, but it must address other issues before it can be considered a mature technology. These issues include database and host computer security, database recovery from interrupted or dropped transactions, and performance issues relating to the addition of another abstraction layer.

E.   COMBINING THE COMPONENTS

After looking at the most common link components, a choice must be made regarding which of them to use when building a Web/DBMS link. The correct combination of client, server and middleware link components is ultimately in the eye of the builder, since the choice of which to use depends upon such questions as the stated purpose of the site, anticipated usage levels, available resources, and the emphasis each developer places on the different and competing link issues.

Given the large number of components available, the numerical possibilities for combining them are quite high, but if an assumption is made that the client will utilize a simple browser with the possible addition of a widely used plug-in, then the number of possible combinations drops. In light of this assumption, closer inspection shows most

41

Web/DBMS link server implementations actually fall into one of the following general categories [Reichard96]:

- CGI gateway

- API (including ODBC)

- Web-enabled DBMS's

- Independent processes

- Other solutions

Chapter IV introduces issues surrounding all Web/DBMS links and discusses the tradeoffs required between them when choosing one of these general link implementation approaches.

IV. WEB/DBMS LINK ISSUES AND TRADEOFFS

A.   MARRIAGE OF TWO TECHNOLOGIES

The technology behind the Web/DBMS link basically represents a new marriage of old technologies. As stated in Chapter I, the WWW is a client/server implementation built upon insecure, connection-less, and stateless protocols, and most Web-connected databases are variations of the relational database model. Since each of these two technologies' strengths and weaknesses are well understood, the novelty and excitement surrounding the Web/DBMS link results from connecting these technologies in manners at odds with their underlying precepts. Examples of this inherently non-symbiotic linkage include the usage of relational databases in an operating environment lacking total control of access method and duration and the use of native WWW protocols to support interactive sessions of long duration.

The creation of a Web/DBMS link additionally involves deciding and committing to such mutually exclusive development paths as the use of tediously hand-written code specifically tailored to user needs, or the use of integrated RAD software packages that may entail limitations to flexibility or complexity. Other issues include considering whether the site should grant secure or secured access to remote users, the use of freeware instead of commercial products to save money, and the ability of normal site administrators to support the final link product instead of requiring the expense of hiring of skilled technicians. The

one constant throughout all link approaches is that each has its own advantages and weakness relative to these issues and so any approach chosen must involve trade-offs.

B. OVERARCHING ISSUES

1. Performance and Functionality

Time delay between user input and system output is an issue that can be affected by such diverse factors as network load, server computational power, and limitations placed upon the number of simultaneous client accesses. Regardless of the cause, as delays increase more and more users begin to consider the delay excessive and start to seek alternative interaction methods [VisualWave96].

In the early stages of the Internet, high-performance content delivery was easily accomplished with quick and simple file servers that could also provide caching of heavily-used pages that, since they were static text, did not change from user to user. As the Web has evolved, though, server capabilities have had to change in response to the need to dynamically generate web pages and tailor page content for each user. These changes in server capabilities must still keep Web/DBMS link access time down, and do it without negatively impacting flexibility in design or content complexity. In other words, the ability to handle additional media and data types or unpredictable peak loads in usage must come without loss of usability or functionality.

The DBMS itself can affect performance [Francett96]. In addition to the DBMS's ability to support multiple simultaneous users, the DBMS must be able to generate pages comprising disparate and rich data types without perceptibly slowing. For this reason, some

corporations are choosing to provide only text-based data instead of richer data utilizing graphics or other such bandwidth and computation-intensive types [Francett96]. Finally, the machinations required to retain DBMS state information in the face of standard Internet protocols can also negatively impact site throughput.

2. Cost

These evolutionary changes in server capabilities must also be made with an eye on the resources required to implement the change. Since money is needed to buy hardware and software, create content, and maintain the site as it grows and reacts to changes in access levels or content, it is fair to say that funding availability affects a Web site throughout its life. If funds are limited, tradeoffs must be made between fund availability and the integration of solution approaches. For instance, a content provider looking for low-cost solutions could initially choose to piece together shareware and freeware components. With additional resources, though, it is possible to use the more expensive integrated Web/DBMS development and maintenance suites or even to commission highly expensive consultant-provided turnkey solutions. Basically, the more money that is available, the easier it is to rapidly field a complete solution.

3. Development Time and Ease

Competition is fierce among content providers trying to attract ever larger numbers of visitors, and so any Web site desiring to continually improve must have rapid development and revision ability. Given adequate funding, integrated RAD Web-server development products can be used to dramatically reduce site development time by offering "wizards" that

insulate developers from arcane scripting languages and quickly guide them through the creation of complex applications. These Fourth Generation Language (4GL) GUI tools such as Borland Corporation's Delphi are intuitive to use and are re-defining the word "rapid," but since each one can lock a developer into a specific set of proprietary languages and protocols, the decision of which of them to use is a paramount issue that can significantly impact Web/DBMS link viability.

4.   Serviceability

Along with content evolution, the Web-associated protocol standards are also continually evolving. Because of this, the maintenance of Web server and client software must be considered up-front and early to ensure that addressing and incorporating these evolutionary changes doesn't become a money and time sink. Basically, the easier the Web/DBMS link is to service, the easier it is to ensure costs remain low. Site owners must therefore have the ability to reconfigure or upgrade the Web/DBMS link easily, and to make content changes quickly without high levels of training or hiring the services of outside consultants.

5.   Flexibility and Openness

Once cost, schedule and performance requirements have been defined, the Web/DBMS link builder should look to the creation of an "open" link using flexible Web/DBMS products that support cross-commercial and cross hardware/operating system architectures. The benefits of this flexibility, or openness, is multi-vendor support, platform independence, and the ability to rely upon standard browsers that avoid proprietary (and

therefore expensive) client software. Openness is indicated by the number of different database models and specific DBMS's supported, along with the ability to "multi-home" data and spread access across multiple DBMS's using different application logic protocols. On the opposite of openness are single-vendor turnkey products requiring total commitment to one link vendor, one hardware instruction set, and thus one set of proprietary protocols.

6. Security

Contrasting with the need for openness and requirements for users to query a DBMS and submit their own content is the need to protect Web site assets and provide this world-wide access to only those users with proper permission. Traditionally, databases have relied largely on the operating system for security [Bina94], but linking to the Web changes that and requires either the DBMS or another system to substitute in this role and deal appropriately with requests from unknown or unauthorized individuals.

No Web-related issue has received more press than security, and industry concerns include user authentication and authorization, multi-level access based upon privileges, prevention of identity spoofing or forced entry, navigation control while users are within the Web site, and of course, database security and the protection of privacy or personal data. That this protection of data within the DBMS must be guaranteed for a Web/DBMS link approach to be viable is certain. What is equally certain is that this protection cannot come at the expense of the product's ability to support client requests.

7. State and Session

State is the other troublesome issue associated with Web/DBMS linkage and we cannot avoid addressing it since the WWW was built with the stateless HTTP session-layer protocol placed over the connection-less TCP/IP transportation-layer protocol. As previously discussed, state addresses the question "where in a series of related computations (e.g., "session") are we?" In traditional client/server the session begins at logon and ends at logoff, and almost all advances in the development of the relational database model presupposed an ability to progress through a transaction (session) in a serial fashion, with each step being cognizant of what had previously transpired. HTTP's inability to support retention of place or state within a transaction has led site builders to such "work-around's" as limiting users to one-shot DBMS queries, creations such as Netscape Corporation's "cookies" to store a record of state on the client end, or the pushing of user identity and session state through hidden intra-web site hyperlinks. Regardless of how the site builders address the state issue, they must address it.

C. DBMS ISSUES

1. Interface Usability and Control

In contrast to previously discussed issues, certain Web/DBMS link issues such as usability, transactions, concurrency and data type extensibility are specific and internal to the DBMS. The DBMS portion of the link should be the least problematic since the relational database has been in use since the 1960's, but problems can still arise if the underlying issues are not properly considered prior to development. Foremost among these issues is interface

"usability" and the level of control users have over their own sessions. The interface should support intuitive Web site navigation and personal tailoring of robust database queries, rapidly giving end users their data in desired formats.

This issue of usability as it relates to the intricacy of database queries to be supported (i.e., the decision of whether the client will conduct simple searches or whether initial query results will be utilized in forming subsequent queries in a cascading fashion) affects Web sites from the outset. To be independent of the server, complex query support requires the use of a fat client, and yet the two "standard" browsers (Microsoft Internet Explorer and Netscape Navigator) that currently dominate the Web market [Ayre96] are thin clients. A decision must be made with respect to the level of DBMS interactivity end users will enjoy and how that interactivity will be supported.

2.   Transactions

Properly supporting complex interactivity requires the builder to know and understand the limitations associated with specific transactions. A database transaction can be one of two types: conversational and pseudo-conversational [Finkelstein96]. In *conversational transactions* a dedicated server process is created for each browser user and maintained until the end-user terminates the connection or a time-out occurs. In other words, the unit of work to be accomplished is extended to include the entire duration spent on the transaction. The advantage of this approach is that the server maintains information about the user's previous data requests and proceeds without interference from other transactions since the database is reserved for the exclusive use of the transaction for its entire duration.

The disadvantages are the serial nature of transactions and the resulting requirement for server dedication of limited resources to each browser client that has recently submitted a request regardless of whether any further processing is needed. This can quickly overload the server and degrade performance, especially if numerous transactions are initiated and then abandoned, leaving the server to wait for them to time out.

Pseudo-conversational transactions partition the work into subunits. This division of work allows the database to be reserved and released more often, but with the penalty of the added effort required to ensure inter-woven transactions do not adversely affect each other's accuracy and data integrity.

With either type of transaction the continual opening and closing of the database has an effect on server performance.

3.   Concurrency

Concurrency [Burleson94] addresses the problem of users affecting each other's transactions by the reading or writing of data during the other's transaction. Problems that can arise include *dirty reads* in which a retrieved record is held by another transaction with intent to update, *non-reproducible results* when queries are run against a database in the middle of an update, or *bad pointers* arising during active updates when previously indexed records no longer exist due to inserts or deletes. In all cases, the number of simultaneous DBMS connections that will be allowed affects the granularity and complexity of DBMS concurrency. The Web/DBMS link should support the same level of transaction concurrency

required in other distributed database applications and do it with a well-conceived locking procedure.

Locking protects data integrity and ensures that a database transaction properly retrieves and updates information by limiting the number of transactions having database access at the same time. Locking can take place at different levels, e.g. rows, tables, or even the entire database, and can be either exclusive to one process or shared by two or more processes depending upon the nature of database access required.

4. Extensibility

The final DBMS-specific issue, extensibility, addresses the inherent capability of the DBMS to easily and quickly adapt to new data forms. As the Web continues to evolve, new data types such as graphic files or audio clips will become more prevalent and, if keeping pace with other sites is an issue, the developers' chosen DBMS must extend to include the ability to store and retrieve these data types. As mentioned before, restriction in data types can lead to restrictions in site growth and adaptability.

D. RELATING THE ISSUES

1. Tradeoffs

The challenge to all site builders is to design Web/DBMS links producing the relevant information (no more or less), at the right time (no delay), in the appropriate form, and at an acceptable cost [Jeffery95]. In a perfect world there would be no downside in financial, performance, or lost opportunity terms to having every link address every link issue adequately.

In truth, though, some of the issues conflict with each other, requiring a site builder to consciously decide to give one or the other more emphasis. Some of the more common tradeoffs include:

- *cost versus development time/ease*: the absolute cheapest way to create a dynamic web site is to use freeware and shareware in "hand programming" a tailor-made link. The problem is that the money saved on software could ultimately be spent on labor since programming the complete solution can be time intensive, require a lot of programming experience, and the testing and de-bugging can also be difficult. Integrated 4GL development packages avoid these problems but in general are quite expensive to purchase initially.

- *openness versus performance*: the best performance comes from code that is vertically integrated from the beginning to offer a complete solution with no extra functionality or abstraction layers. As the capability to support more protocols and functions are added, more abstraction layers are added that can adversely impact performance speed.

- *cost versus serviceability*: closely related to the first bullet, the more hand programming involved in the initial creation of a link, the more hand programming involved in the servicing and upkeep of the site. Time is money.

- *security versus performance*: just as more abstraction layers can negatively impact performance, the addition of more gate-guard requirements can also

be of significance. This is especially true when the additional layers are themselves computation intensive or require additional use of network links.

- *transactions control versus performance*: time and effort spent on controlling simultaneous DBMS access to prevent mutual interference between clients can delay the production and transmission of results.

- *data-type versus performance*: as previously mentioned, the richer the data type, the more network bandwidth and server computational power required to maintain rapid results.

2. Assessing the Combination Categories

Now that the more common trade-offs have been highlighted, a close look can now be taken at how these common tradeoffs apply to each of the aforementioned general link categories. This will allow conclusions to be drawn on the relative advantages, disadvantages, and tradeoffs associated with each one. The first three categories discussed assume the use of a standard browser client. Then the use of a plug-in to facilitate independent process is discussed and finally, a brief discussion of fat clients and certain proprietary middleware solutions concludes the chapter.

a. CGI Gateway

CGI's primary advantages are its ubiquitous support among Web servers and programming languages, and the large body of previously-written software modules that can be pasted together by experienced programmers to provide tailored solutions to almost any link problem. CGI's biggest downfall, aside from the requirement for strong programming

skills, is the performance deficit it displays in relation to code written and compiled to support a DBMS in its native language, and the load it places on the Web server to spawn a process and open and close the DBMS. Use of CGI is a tradeoff of performance for cost.

b. *API (including ODBC)*

API's are the new wave in Web/DBMS linkage offering dramatic improvements in performance. These speedy links come at the expense of openness and the lack of universal standards and Web server support. To utilize a specific link, proprietary software products must be purchased, thus locking the site builder into a long-term relationship with a specific vendor. The middleware "API's" like ODBC attempt to address this proprietary nature of API's but they impose the performance burden of an additional software abstraction layer. API's represent a tradeoff between performance and openness, and they come at a greater financial expense than CGI.

c. *Web-enabled DBMS's*

Utilizing the DBMS itself to produce the HTML file for the Web server to ship is an innovation that offers promise. Templates and special HTML code or embedded SQL can all be used alone or together to relieve the server of the computational burden of gateways. Some of these solutions are proprietary and they can cause problems for standard browsers that don't recognize and properly handle special tags. The primary benefit of these methods is the retention of the standard browser for the client and the centralizing of the solution at the server. Financially these solutions can be prohibitive for the small enterprise.

d. *Independent Processes*

With the addition of a browser plug-in, the size and capability of the client software is moving back in the direction of the traditional distributed DBMS fat client. The gain in putting some of the application logic back on the client is the spreading of work across all participating entities and the associated possibility for a reduction in network transmissions. These processes represent the newest link solution and are therefore the least understood with respect to their impact on all the issues. Initial indications are that additional client capability comes at the expense of performance and the possible compromise of security.

e. *Other Solutions*

In this category are fat clients and special middleware like CORBA that successfully resolve state and session issues at the expense of openness. Problems primarily revolve around the decentralized nature of these solutions requiring special client-based components and functionality. These components bring with them all the traditional problems of upgrade and serviceability that accompany any non-standard browser based solution.

3. The Ideal Solution

In the absence of technological limits, the ideal WWW/database link would offer the scalability of traditional client/server systems along with browser independence, robust queries on a secure database, speedy response times, 4GL rapid application development techniques, persistent state and session management, and multiple, cross-DBMS support that

allowed for easy integration with emerging technologies. In reality, due to the tradeoffs that must be made among the underlying issues, no single linkage approach offers a perfect solution. The remainder of this thesis, by covering the steps in designing a Web/DBMS link for a prototypical case study, serves to further illustrate the complexity and nuances surrounding each issue, and the tradeoffs between them.

V. CASE STUDY

This case study focuses on the process whereby The Institute of Electrical and Electronics Engineers (IEEE) Signal Processing (SP) Society reviews the format and content of technical papers submitted for publishing in the Society's journal, *IEEE Transactions on Signal Processing*.

The Society currently receives approximately 700 papers a year from authors desiring their papers to be published in the Society's journal [Moura96]. For each of these 700-plus papers, the Society requires two or three qualified reviewers to be found, and their services enlisted, to review the paper's format and content. The administrative logistics required to find, solicit, and support enough reviewers to complete approximately 2100 reviews per year is difficult.

A.   BACKGROUND

1.   Current Review Process

As outlined in Appendix A and illustrated in Figure 7, authors desiring publishing must first suggest an appropriate Editors' Information Classification Scheme (EDICS) number for their paper based upon its subject matter. The EDICS [IEEE96] is a SP sub-subject numbering system which is used to categorize and map a paper's subject matter to the most appropriate sub-area within the overall SP subject. It is also used for assigning areas of responsibility to respective Associate Editors (AE's). Of note, each EDICS is

associated with several AE's and each AE is responsible for several EDICS, so overlaps are possible.

Once the author has chosen an appropriate EDICS code for his paper, he mails seven copies of the manuscript, abstract and references to the Society. Then, based upon the suggested EDICS code, Society staff chooses one of the code's associated AE's and forwards to him six of the seven manuscript copies.
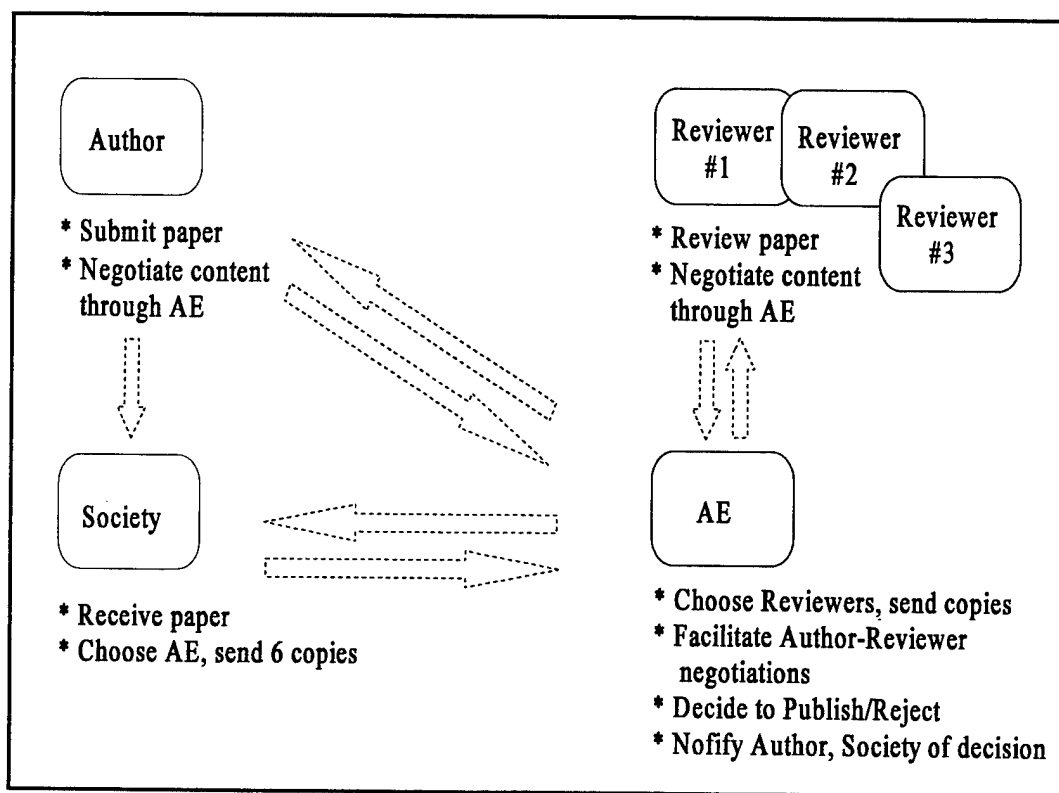
Figure 7. Paper Submission and Review Process

Upon receiving the paper copies, the AE must find the two or three reviewers that most appropriately match the paper's content and who also have the time and inclination to participate in the review process. Although the Society will sometimes suggest the names

of some potential reviewers, most of the time the AE alone is responsible for identifying and negotiating the services of qualified reviewers for each of the papers he is assigned.

The most common method utilized by AE's for reviewer identification is to build an ad-hoc list of names from personal acquaintances, word-of-mouth references, conference attendee lists, published paper references, journal subscription lists, etc. Then, to solicit reviewers from the lists, the AE's conduct a blanket mailing of solicitations. This mailing is generally expected to yield a fifty-percent success rate in terms of responders willing to review the paper [Fargues96]. When enough of the contacted reviewers have acknowledged the solicitation favorably, the AE mails them copies of the manuscript. Of note, some reviewers use students in a mentor-apprentice relationship to perform "ghost reviews" for the reviewers subsequent check.

Each of the reviewers reads the manuscript, annotates comments and suggested changes, and then mails the papers back to the AE for forwarding to the author (who remains "blind" to the identities of his paper's reviewers). The author responds to the suggested changes by either incorporating them outright or debating them (via the AE) with the reviewers. This "negotiation stage" may go through several iterations prior to all parties reaching complete agreement. Once the author and reviewers have finally agreed upon the paper's content, the AE must decide whether to publish or reject the paper and then notify the author and the Society of his decision.

Currently, about half of the submitted papers are eventually published after negotiations lasting 14 months on average but, in some cases, extending up to two years.

The remaining papers are rejected for reasons including incompatibilities with format or length restrictions, paper-Journal audience mismatch, work overlap between authors, or even disagreement over the importance and relevance of the paper's material.

2. Changes In Work

Several changes to the current process are being implemented. First, to alleviate some of the AE's workload, the Society plans to wait until the AE has provided a list of reviewers and then it will mail the manuscript copies directly to the reviewers. This will be done instead of forwarding six copies of each manuscript to the AE for his subsequent re-mailing to reviewers and therefore allows the Society to monitor time-lag associated with reviewer assignment.

The second change [Moura96] contemplated by the Society is to create the position of "designate reviewer." A *designate reviewer* is an approved reviewer who enters into a relationship with an AE agreeing to accept, for prompt review, a limited number of manuscripts for review per year. It is hoped that this pre-agreement will greatly streamline the reviewer solicitation process, protect the reviewers from being over-utilized, and additionally serve to create a database of reviewers with relevant biographical and review-related data.

The third, and relatively minor, planned change is to require the reviewers to pre-agree to completing their review within six weeks of manuscript receipt in order to reduce the length of the negotiation stage and help ensure that manuscripts are published on a timely basis.

a. *New Database*

Lastly, as discussed in Appendix A, the Society has hired an outside consultant to create a Microsoft Access95 database application to hold data relevant to the manuscripts and personnel associated with their review. The application is currently being implemented on a standalone PC (that is not dedicated solely to the database) but it will eventually be moved to a dedicated PC server so all of the appropriate Society staff can have unrestricted access. The database is designed to do/aid the following:

- Log in submitted manuscripts as they arrive

- Automatically generate a receipt letter to send to the author

- Automatically generate a list of AE's responsible for the manuscript's EDICS code

- Track AE assignments and limit AE responsibilities to no more than three manuscripts per month

- Provide AE's with suggested reviewer names

- Keep statistics such as author and reviewer periodicity, or in other words, the number of papers an author has submitted or the number of times a reviewer has been used.

3. Database System Support

The new application can be broken into the three main sub-components of hardware, software, and the data/data organization contained within the software.

a.   Hardware

The database is implemented on the following computer hardware:

- CPU: 133 MHZ Pentium
- RAM: 16 MB
- Hard Drive: 1 GB
- CD-ROM
- TAPE B/U
- MODEM
- NETWORK INTERFACE CARD   (Ethernet)

The computer is currently connected to an internal, cross-departmental, Ethernet-based, Windows peer-to-peer LAN that is connected to the Internet. This connection may change slightly in the near future since the journal staff wants to separate from the enterprise-level LAN for privacy reasons and to facilitate an impending move to another physical location.

b.   Software

Software support for the reviewer database includes Access95, a Microsoft Corporation implementation of the relational database model, and the Microsoft Windows95 operating system. The current plan is to give only three staff members password-enabled access to the database. An additional requirement exists for remote access via modem due to one staff member requiring the ability to work from home.

c.   Database

The relationships captured by the new relational database are illustrated in Figure 8. Although only three main entities (PAPERS, PEOPLE, and EDICS) exist, the relationships between them are somewhat complex because of the multiple many-to-many

relationships between PAPERS and PEOPLE. Since all personnel related to the review process are members of the PEOPLE entity, a single paper can have many relationships with the PEOPLE entity; one for each author, one for each reviewer, and one for the AE. Additionally, each author, reviewer and AE can be associated with more than one paper.

With respect to PEOPLE and EDICS, each AE can be responsible for multiple EDICS codes while each EDICS can be associated with multiple AE's. The same many-to-many relationship should be included between the reviewers and the respective EDICS codes but neither the conversations recorded in the Appendices nor careful scrutiny of [IEEE96-2] reveal this relationship to exist within the database. The resulting effect of this "non-relationship" is that no capability currently exists for categorizing and sorting reviewers according to EDICS.
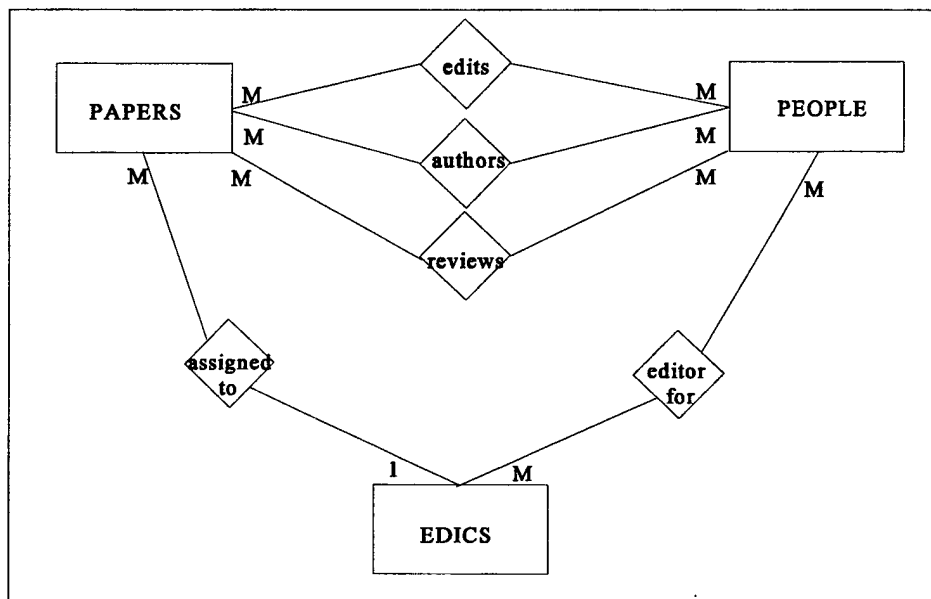
Figure 8. E-R Diagram

The three main database tables are named to match the relational entities. The first of these called PAPERS, is for the submitted manuscripts; the second, called PEOPLE, is for the review-related personnel, reviewer, author, and AE, and the third; called EDICS, contains all journal-related information. Tables 3, 4, and 5 list the actual database tables [IEEE96-2] and their respective fields. Explanatory notes are included for some of the fields as appropriate.

Field	Notes
Id#	Primary Key
Title	
Contact Author	Foreign Key (PEOPLE.ID#), primary negotiation author
Authors	Foreign Key (PEOPLE.ID#), up to 6 additional authors
Date Rec'd	
EDICS	Foreign Key (EDICS.Code#)
Status	AE/reviewer assignment complete, paper type/accept, etc.
Last Changed	date status changed last
Over-length	yes/no, yes = auto reject
Inc Format	yes/no, yes = auto reject (incorrect format)
AE	Foreign Key (PEOPLE.Editor)
Acc Type, Pub Type	correspondence/paper
#Pages	
SDATE2	date AE assigned
Rev Date	
Pub Date	
Copyright Date	
Artwork Rec'd	yes/no
Bio Rec'd	yes/no
#Figures	
DISC	media type (diskette, email, unknown)
#Rev	negotiation revisions
Pub Page	#pages in final version
ABSTR	yes/no, yes=abstract to be printed
Abstr Rec'd	date

Table 3. PAPERS

NOTE: A record is created for each manuscript received whether it is kept for review or rejected outright by the staff for format reasons.

Field	Notes
Id#	Primary Key
First	
Last	
Title	
Salutation	
Phone	
Address	actually 5 fields, ADD1...ADD5
FAX	
Email	
Editor	yes/no
Term Start	
Term End	
Reviewer	yes/no
Contact Author	yes/no

Table 4. PEOPLE

NOTE: approximately 40 AE's are activated on a daily basis for each of three

Transactions (Digital, Audio, Image).

Field	Notes
Code#	Primary Key
Transaction	(The three Transaction journals share the same code numbers but the subject matter of each code number changes from Transaction to Transaction)
Description	Subject of the EDICS code
Editors	Repeating Field

Table 5. EDICS

4. Other Plans

The database is not the only change the Society has planned for the review process (Appendix A). Also being considered is the acceptance of electronically submitted abstracts and reference lists that can be forwarded to the AE's to offer them a head start in identifying reviewers. Once reviewers are identified, the AE will simply send their names to the Society so it can, in turn, mail each reviewer a manuscript copy. It is hoped that this new procedure will reduce the amount of time that manuscripts spend awaiting reviewer assignment or in transit between the AE and reviewer, thereby speeding up the entire review process.

As an aside, in January of 1997 the Society will also begin publishing letters to the editor on the Internet with the hope of eventually offering online subscription/access to the journal itself. What is not known at this time is how the current income from subscription fees will be replaced. Since the subscription fees are the lifeblood of the journal, this is a very serious concern.

B. CASE ANALYSIS

1. Review Process Problems

The current reviewer solicitation process and the resultant time drag it imposes upon the entire review process is one of the top areas targeted for improvement by the Society [Fargues96]. Soliciting reviewers imposes an inordinate administrative burden upon AE's and they are simply spending too many hours in relation to their other professional responsibilities in an attempt to find and communicate with enough new and current reviewers. Due to the self-generated nature of their reviewer lists, multiple AE's are

67

soliciting and utilizing the same reviewers repeatedly, thereby causing the reviewer-workload mismatch which burns out the most willing of reviewers.

Despite the AE's best attempts to properly support the reviewer solicitation process, the sheer volume of papers submitted and the ensuing difficulty in finding enough qualified reviewers is causing many AE's to short-circuit the solicitation process. As a result, the AE's simply mail copies of the manuscripts to potential reviewers who they hope will accept the responsibility, and the first time many reviewers know of a paper's existence is when it comes in the mail [Fargues96].

In summation, the following list details the problems with the current reviewer solicitation process that have been recognized by the Society [Kowalczyk96] as being among the first that it wants to address:

- The continual shrinking of the qualified reviewer pool due to reviewer overload,

- The virtual impossibility of gaining access to individuals outside of the small pool of previously-published individuals and AE personal acquaintances,

- The fact that AE's usually do not have much administrative support and therefore tend to sit on the manuscripts,

- The fact that many knowledgeable and potentially willing reviewers such as Ph.D. candidates, newly hired faculty, and commercial industry representatives remain unidentified by AE's and are therefore not asked to

participate due to lack of publishing experience or association with separate social/professional circles [Fargues96],

- The fact that some authors publish regularly but are either unwilling to review the work of others or are not identified as potential reviewers (these members need to be identified and encouraged to help out),

- The fear that a general call to the public to identify additional reviewers may generate too many responses from unqualified people or those whose qualifications would be difficult to vet, thereby overwhelming the small journal staff.

2.   Solving the Problems

As with all problems, there are multiple solutions to those pertaining to the review process. Though the merits and cost/benefit ratios of each solution are debatable, what can't be debated is that wasted time equals wasted resources, resources which could be used by the Society in other venues. In recognition of this, the Society has made the following modifications and additions to the current reviewer solicitation process.

First, a full effort is being made to expand the database of *designate reviewers* to include every reviewer known to the AE's and other Society members. The thought is that centralizing separately compiled reviewer lists allows the Society to gain control and standardize the reviewer qualification process, facilitate AE solicitations for reviewer names, and provide a central point of contact for record update by the authors, reviewers and AE's.

This will additionally help the Society track reviewer work assignments to prevent reviewer overload and burn out and ensure author participation on both sides of the review process.

a.   *Attracting More Reviewers*

The Society is also interested in investigating methods for identifying qualified candidate reviewers who are unknown to the Society. By registering these candidates and (once properly vetted) adding them to the pool of reviewers, a win-win situation is created in which the Society can increase the size of its reviewer pool while simultaneously allowing SP subject-matter experts who are unknown to the Society to break out from obscurity. It is by identifying themselves and becoming valued, knowledgeable partners in the publishing process that both the Society and the new reviewers receive value.

The anticipated procedure for vetting these candidate reviewers would be for them to submit biographical information to the Society, have the staff conduct an initial review and then forward the applicant's information to an appropriate AE. The AE would then verify the applicant's qualifications and inform the Society whether the applicant should be added to the reviewer database. Specific biographical information to be captured would include all of the fields in the PERSON table plus those extra fields shown below in Table 6.

Field	Notes
Education Level Attained	MS, PHD, etc
Certifying Educational Institution	
IEEE Member Number/status	can be targeted for membership if not already a member
Biography	suggestions include papers/articles published, special expertise, etc
References	with Contact Information
EDICS Categories Wanting to Review	

Table 6. Candidate Reviewer Biographical Data

The most important information that the Society needs regarding the potential reviewers are references and proof of educational certifications to ensure the motivation, authenticity, and credibility of the individuals.

3. Using The Web

One possibility for implementing the aforementioned changes to the reviewer solicitation process would be to link the new reviewer database to the Internet. This would allow potential reviewers to submit their own biographical data while offering the Society the potential of dramatically increasing the size of the reviewer pool by tapping into the continued growth of the Internet. At a minimum, this link would relieve Society staff of the mundane biographical data-entry tasks associated with new reviewers. Of more importance, it could also reduce the workload associated with soliciting new reviewers via such traditional search strategies as blanket application mailings, exhorting current AE's and reviewers to "rush" new reviewers, and conducting telephone polls.

A Web link has yet to be created for the new database due to concerns [Kowalczyk96] over maintaining data integrity (*sic.* purity) and maintaining the sanctity of the review process. Protecting the confidentiality of reviewer identities from the authors dictates that only the office staff, actually responsible for maintaining the database and ensuring that data entries are correct, should have read/write access to reviewer, author and AE data.

AE access to the new database via the Internet is also considered problematic since AE's are only appointed on a voluntary basis for a short term lasting just long enough to oversee the review of 25 manuscripts. In addition to a valid fear of malevolent hacking, it is this revolving door of appointed AE's (of whom a high percentage are authors) that has caused the Society to demand total separation of the new database and Internet to date. The concern is that just one mishap is one too many given the limited number of personnel available to "fix" bad or incorrect entries, regenerate lost information, and handle the political fallout associated with the revealing of reviewer identities.

Despite these concerns, the Society's fears are, in fact, not unique and are actually a subset of the Web/DBMS link issues discussed in Chapter III. Using the Web as a communication link between potential reviewers and the database can actually be a viable tool that the Society staff can use with confidence. This link will specifically alleviate prospective reviewer identification problems by allowing potential reviewers to submit their own personal data to the database for Society review. Also, by additionally enabling the AE's to rapidly find the appropriate reviewers for a particular paper, the link aids the Society

in reducing the administrative burden associated with centralizing the manuscript review process.

In summary, the creation of a Web/database link is a viable solution to the Society's problems with the current reviewer solicitation process. The remainder of this thesis discusses the issues, functionality and additional system support required for implementing this link within the guidelines of the mandatory access constraints. It also details required changes to the currently implemented database which must be made to facilitate the use of a Web/database link.

VI. DESIGNING THE LINK

When currently used systems no longer meet user needs, changes must be implemented. This chapter assumes the Society's concurrence with the technical viability of Chapter V's recommendation to change its current reviewer solicitation and designation process by modifying its manuscript reviewer database and linking it to the Web. The chapter provides a framework for implementing the recommended changes and begins by discussing the expected benefits and required constraints upon the Web/database link. Following this discussion is a listing of the advantages and disadvantages of using each of Chapter III's general link approaches. Finally, the chapter concludes by first selecting the general link approach which best addresses the Society imposed constraints, and then specifying and discussing the system design to implement the link.

A.   USER REQUIREMENTS

1.   Expected Benefits

To ensure that the Society's needs and concerns are properly addressed, a thorough investigation of the expected benefits and user-imposed constraints is required. Primary among the benefits expected from linking the database to the Web is the Society gaining access to a larger pool of potential reviewers. This larger pool will reduce workload requirements for the individual reviewer while simultaneously decreasing the amount of time AE's spend finding and enlisting the services of new reviewers [Fargues96]. Additionally, by combining the separate AE reviewer-contact lists the Society will gain:

- centralized administrative oversight and control of reviewer qualification process

- reduction of AE administrative load

- removal of the redundancy between AE lists

- facilitation of the AE turnover process.

In other words, centralization of the reviewer database will remove the AE reviewer-assignment bottleneck from the review process and, via the quick identification of specific reviewers in response to paper content, ease the building of reviewer teams. This ability could eventually facilitate the creation of reviewer mentor/apprentice relationships when appropriate. The end result will be a reduction in the 14-month lag from manuscript submission to publication.

Another benefit of creating a database link and hosting a reviewer-specific web site is the fact that it is not a radical departure from currently existing IEEE websites including the main IEEE page (www.ieee.org) and the SP homepage (www.ieee.org/sp). The new site could take advantage of existing IEEE-related Internet traffic as it is a fairly simple matter to make the new site an extension linked to the other sites.

2. Specific In-variants

Due to the small size of the journal's staff, the limited nature of its supporting budget, and the Society's strong concern for its intellectual and computational assets, certain in-variants were stated [Kowalczyk96] from the onset for the Web/database link.

76

First among equals is the absolute imperative to preserve and protect database integrity from inadvertent mistakes or malicious intent. Of almost equal importance are protecting the sanctity and confidentiality of the review process and ensuring that the Society's intellectual assets (*sic.* reviewer contact lists) are protected from competitors. This, in turn, requires that limits be placed upon the AE's ability to search and access reviewer names and contact information. The AE's access must be limited to just the particular author-reviewer grouping for which the AE is accountable.

Also understood is that the Society desires complete control over the solution design and content to ensure that any new system implemented is completely compatible with the Society's currently used hardware and software systems. Furthermore, flexibility and scalability for future content and size growth must be guaranteed and the link project must be inexpensive to maintain and deploy or it will not be implemented.

Finally, when new reviewers register with the Society, registration must be accomplished in such a manner to facilitate quick and accurate Society verification of submitted qualifications. Care must also be taken to ensure the new reviewers are familiar with proper review procedures to protect all parties' legal and privacy interests.

3. Information Flow

a. Web Input/Output

As illustrated in Figure 9, the Web site will initially funnel two separate data streams into the database. The first stream is the prospective reviewer's biographical data

as submitted via an HTML form, while the second contains AE queries submitted to the database as they search for reviewers associated with a certain EDICS code.
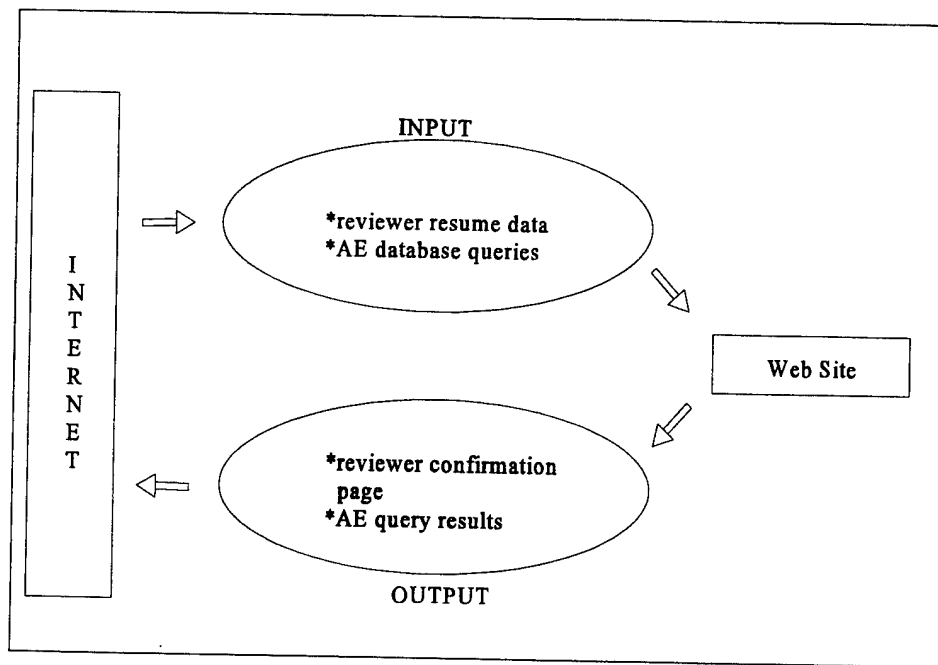
Figure 9. Web Site Data Flow

Output of the Web site will also consist of two things. The first is a simple HTML page confirming to the prospective reviewer that their data has been accepted for investigation and that they will be contacted within a certain time frame regarding their status. The second output will be for the AE's only and consists of the results of their submitted query.

4.   Summary

The overall requirement for the Web site is that it act as a clearing house for collecting and disseminating information related to the reviewer solicitation and paper review process with the end goal of making this process easier for all parties involved.

B.   CHOOSING AN APPROACH

1.   Issues for Consideration

When looking at the case analysis section in the previous chapter and the in-variants discussed in this chapter, one thing becomes clear: the Society's concerns regarding connection of its database to the Web are actually real-life manifestations of the overarching link issues of Performance and Functionality, Cost, Development Time and Ease, Serviceability, Flexibility and Openness, Security, State and Session discussed in Chapter IV. This section reviews these issues as they relate to the case study.

a.   *Overarching Issues*

It is understood that the most important issue of concern when choosing a link approach for this case study is the guaranteed security of the Web site and the database. No matter how well the other issues are addressed, it is a given that if security is not guaranteed, no Web link will be created. Therefore, the approach recommended by this thesis holds security in the forefront of all design decisions.

With Society funds and available personnel a scarce commodity, the cost of the chosen link approach must include all additional system support purchases plus the consultant fees for implementation. Therefore, the development of the link must be rapid and easy, and the actual application must easily support continual re-design as the site changes and grows over time. Finally, the need to limit expenditures and ensure compatibility with currently used systems also requires the link to utilize and leverage as much of the installed system components and user expertise as possible.

The easiest issues to grapple with are the performance and functionality issues. Performance wise, with the limited number of AE's who would have access to the database at any given time, a high performance server on the order of a catalog or auctioning service is not needed and a simple system will suffice. Additionally, completing and submitting a form to a Web site or executing a simple database query are two common functions which are well within the capabilities of most Web software programs.

From a database perspective, the issues of transaction monitoring and concurrency limitations are relatively moot since they are "write" related issues, and as originally envisioned, all case study database writing will be done by the prospective reviewers. Since each resume constitutes a separate record, the prospective reviewers will not be accessing the same records and the requirement for concurrent processing is non-existent. With respect to the AE's, although they will sometimes be accessing the same records simultaneously, concurrency is still not an issue since the records are related to currently vetted reviewers and all AE queries will be "read" only.

2.   Approach Alternatives

To properly address these issues, the Society can choose a Web/database link approach from the general approach alternatives discussed in Chapter III:

- Common Gateway Interfaces (CGI's)

- Application Program Interfaces (API's) including Open Database Connectivity (ODBC)

- Web-enabled Database Administrative Systems (DBMS's)

- Independent processes

- Other solutions.

A brief scan of the list reveals the following: use of a CGI gateway can be quickly discarded (despite its offered flexibility) because of its potentially deleterious effect upon security and the level of programming skills required to use it. With respect to independent processes, the writing of Java applets or Active-X controls can also be discarded, as writing them can be too hard for the non-programmer and the computer industry has yet to completely address all concerns regarding security. The final category to quickly be discarded is the catch-all category of "Other Solutions." This category includes the proprietary approaches which suffer from limited openness and security qualities, or are too expensive to deploy and maintain on a limited budget.

The options actually constituting viable alternatives are the use of API's (including ODBC) and Web-enabled DBMS's. Either one offers a valid approach to addressing the Society's link requirements, but as will be shown later in this chapter, API's offer more flexibility.

One option the Society could take is to convert its new Access95 database into a Web-enabled database [Simpson96] which actually generates and exports static HTML files directly to a Web server and on to the Web. This is accomplished via a free add-on called Internet Assistant for Access95, which is downloadable from Microsoft's Internet site. One disadvantage to this method is the labor intensive nature of continually creating new static files every time that the database is updated. Despite the additional labor, if the Society only

wants to export static information to the Web without offering the capability of updating database tables or enumerating dynamic data, then the Internet Assistant for Access95 could be one avenue worth pursuing. Since the implementation is trivial and the offered capability is not within the purview of this thesis, the actual use of Internet Assistant for Access95 will not be explored further.

3. Recommended Approach

We recommend a solution falling within the category of "API (including ODBC)." The primary reason (outside of the potential for air-tight security) is that it offers complete compatibility with the Society's installed software/hardware/user-expertise base and therefore eases the migration of the Society's new database to the Web. This compatibility also serves to limit the amount of additional cash outlays and the training time required to bring the link to fruition.

C. SYSTEM SPECIFICATION

This section specifies required components and the system design for the recommended approach. Sufficient detail is provided to serve as a framework for Society implementation.

1. System Support

a. *Software Applications*

Given that the Society is currently implementing its reviewer and paper tracking database on a IBM-clone PC with all Microsoft products, the link should be designed with the Microsoft software applications listed in Table 7.

Application	URL of Location Extra Reference Material
Access95	www.microsoft.com/msaccess
Windows NT Server (3.51 or later)	www.microsoft.com/ntserver/default.asp
Internet Information Server	www.microsoft.com/iis/default.asp
Internet Database Connector, or dbWeb (choose one)	www.microsoft.com/accessdev/itk/idcfaq.htm www.microsoft.com/intdev/dbweb

Table 7. Microsoft Internet Related Software Applications

Because Access95 is already being used by the Society for its new database, it is the obvious choice for the database portion of the link. The added benefits of using Access95 are the user level password security it offers to protect data, and the fact that it already contains an ODBC driver as shipped.

We recommend switching from the Windows95 operating system to Windows NT Server (3.51 or later). Windows95 is not sufficient for Web site hosting because it is a workstation operating system which does have any networking capability outside the Microsoft peer-to-peer networking service first introduced with Windows for Workgroups 3.11.

Windows NT Server is a complete multi-threaded, multi-tasking operating system offering centralized user access control, file-level security locks, and robust support for remote modem access and Internet connectivity [Minasi96]. NT Server scales easily to support large numbers of users while offering single-point administration of all users and resources. The Society's security concerns are particularly well addressed because of NT Server's C-2 security capability which precludes users from granting access rights. NT Server also has the ability to monitor all security related actions and keep a log for

subsequent review. Finally, NT Server offers the ability to selectively grant access to users (and groups of users with the same needs) for all methods of access including interactive (user is physically sitting at the computer), over a network, and via a modem. Access type is also controllable so while Society staff can be given full read, write, and create access, AE's can be given password-controlled read-only access while prospective reviewers can be given write access only.

For the Society staff who require at-home access to the new database, NT's Remote Access Server (RAS) offers the user all of the capabilities that they would enjoy while actually sitting at the computer. The data security capability of RAS is quite strong and includes the capability to encrypt all transmitted data. Access security via encrypted passwords and telephone callback features is also quite strong.

Internet Information Server (IIS) is the only Web server that integrates fully with Windows NT Server. IIS runs all of the standard Internet services (Web, FTP, Gopher) and allows the site manager to grant/restrict, monitor and log user access to HTML pages, data files, and database records. By seamlessly integrating with the security features of NT Sever, IIS provides the ability to require users to input identification and password information prior to gaining access to "privileged" information. It can even support the use of SSL if required. Finally, IIS works with all standard client browsers hosted on almost any platform, thereby offering the universal access necessary for the Society's world-wide users.

With respect to linking the Web server to the database, Microsoft's Internet Database Connector (IDC) and dbWeb, as illustrated in Table 8, are both API applications which offer complete Web server/database connectivity and "wizard" publishing technology. The wizards work via a point and click metaphor to abstract much of the mundane programming required to build a link. Where the two applications differ, though, is that they are targeted towards two essentially separate types of users.

TRAIT	IDC	dbWeb
Wizard interface shields users from complexities of Web/database interface programming	yes	yes
Primary Audience	developer	developer/power user
Uses ISAPI architecture to run as a server process	yes	yes
Can publish data from any ODBC compliant database	yes	yes
Creates static pages	yes	**no**
Creates dynamic pages in response to user input	yes	yes
Creates files with SQL statements the developer can modify	yes	**no**

Table 8. IDC and dbWeb

IDC is a developer-oriented core technology of the Internet Information Server that provides connectivity between the Server and any ODBC-compliant database. It also serves as a foundation for developing custom database applications in concert with Internet Information Server. IDC provides a great deal of flexibility by allowing developers to directly program custom SQL statements, create their own HTML templates, or generate static database information "snapshots" which can be quickly loaded by the server in response to client requests.

dbWeb, while also offering connectivity between the Server and any ODBC-compliant database, uses its wizards to additionally abstract the intricacies of SQL programming. By offering this additional abstraction service, it prevents the user from having to actually do any programming in SQL. Also, dbWeb does not allow the generation of static HTML files and is therefore a little less robust than IDC.

b. Platform Selection

Compiling the list of hardware required to use the above applications is very easy since most of the required hardware components are the ones already owned by the Society and listed in Chapter V. The only hardware item needed is some additional RAM in order to bring the system total up to the minimum recommended by Microsoft.

c. Cost of Upgrading

Upgrading the current system is quite inexpensive since all the current hardware plus Access95 can be kept. The major software expense is replacing Windows95 with Windows NT Server (3.51 or later) since IIS and IDC are free add-ons to NT Server. The hardware upgrade is an inexpensive matter of inserting two more SIMMS (assuming current configuration is two 8MB SIMMS). Table 9 lists the additional system support items required and includes an estimation of each item's cost based upon market rates advertised in the San Jose Mercury during the month of December 1996.

Item	Cost
Microsoft NT 3.51 (5-client license)	$750.00
Internet Information Server	free
Internet Database Connector	free
16 MB RAM	$75.00

Table 9. Upgrade Costs

Note: the number of recommended client licenses, although low, is sufficient because they actually enumerate the number of simultaneous connections the server can support.

d.   *Migrating to the Web*

The switching of the database from one operating system to another is trivial since NT Server supports all of the Windows95 applications the Society is currently using. Still, migrating the current database to the Web requires a choice between two options. The first option is to put aside vulnerability concerns about granting Web access to the database and to simply use the single database application for staff use, prospective reviewer resume submission, and AE search queries. This requires extra care when implementing the link so that integrity control is never lost and no data will need to be re-generated.

The other (and better) option is to actually have two separate database applications, one for staff use only, and the other a replicate for all Web-related use. Complete physical separation of the two databases offers greater security than trying to protect sub-portions of the database from net users. This separation and replication is possible because Access95 can actually import table data from external sources. Examples of external data sources include tables from other Access95 databases located on a network,

87

data from other programs such as Microsoft Excel, Microsoft FoxPro, Paradox, and Microsoft SQL Server, as well as data from other programs and files. In the Society's case, importing data creates a copy of the information in one Access95 database to a new table in another Access95 database so the external data source's format is not altered. The Society staff can therefore continue to use its original program, while Web users use the other Access95 database to add or edit the copy of their own personal data.

Although imported data can't be appended to existing tables (except when importing spreadsheet or text files), once a table has been imported, an append query can be performed to add the data to a pre-existing table. When importing from another Access95 database, the user can choose to import all or just a subset of the database in a single operation. Additionally, one of the best capabilities of Access95's import feature is that if the need exists to perform an import operation on an ongoing basis, the user can automate the process of importing data using macros or Visual Basic for Applications (VBA) code.

In the future, if the Society decides to use another database or scale up to a different server suite, Access95 can also export data and the information needed to access that data to programs and databases which support the Open Database Connectivity (ODBC) protocol. Options include, but are not limited to, Microsoft SQL Server which is the logical extension to the use of Access95 in large scale operations. Of note, the information required to define and access an ODBC data source varies depending on the requirements of its ODBC driver. The developer must therefore create a data source definition for each ODBC database he wants to import from, export to, or link to the Access95 database.

2. System Design

The required software will work together as illustrated in Figure 10. The IIS will listen on the Internet for incoming client requests and pass them off to NT Server. The Server will ascertain which URL is requested and forward resume submission requests to the secondary Access95 database while the AE-generated reviewer-name queries will be forwarded to the primary database.
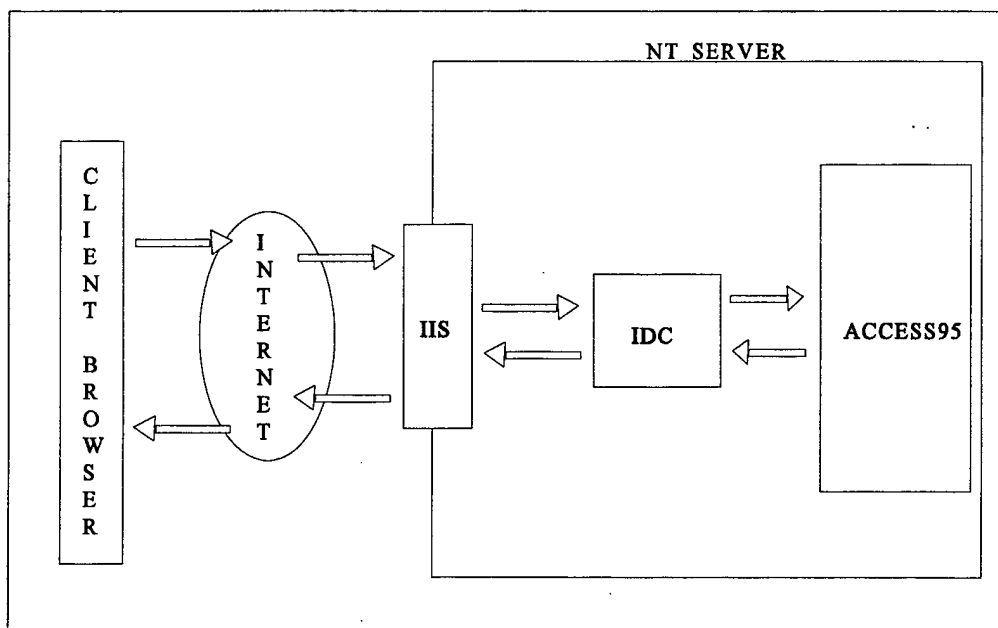
Figure 10. Software Hierarchy

a. Sample Queries

The SQL queries that the database needs to support are very simple. After entering proper user authentication data based upon name and password, the AE will be presented with a form which allows him to choose (via a list box) one of the EDICS codes. The database will then perform an SQL query similar to the one below:

```
SELECT      First, Last
FROM        PEOPLE
WHERE       Reviewer = "yes"
            And
            EDICS_Reviews = "EDICS#"
```

The database responds with a simple list of reviewer names, each of which is a hyperlink to specific reviewer contact data (EDICS# is the code number selected by the user). Of more importance, the current database must be changed to incorporate a new field in the PEOPLE table that links the reviewers to the EDICS they review.

Once the names have been returned as hyper-links, the AE can then scroll down the list, click on a few reviewer hyper-links and copy or save the contact data. The actual SQL code for executing this would be similar to the following pseudo-code in which "First" and "Last" represent the hyper-link data.

```
SELECT      *
FROM        PEOPLE
WHERE       First = "First"
            And
            Last = "Last"
```

3.   Potential New Features

Once the extra system support has been installed and the link between the database and the Web is operating correctly, the Society may choose to expand the site's capabilities. Example additions include enabling NT's FTP service so members can access and download large review instruction files or authors can download paper format templates. The potential would then exist for authors to submit their original papers and reviewers to download, review, and resubmit the annotated papers via FTP instead of by postal mail. The initial use

90

of the FTP service is less important than the fact that NT's FTP capability exists and offers exciting new opportunities for the Society to expand upon the functionality that is already offered via other sites linked to the main IEEE homepage.

The incorporation of an AE email address library categorized by EDICS codes that can facilitate communications between review personnel (Society Staff, reviewers, AE's) is another potential use of the Web site. Implementation would be a simple matter of extending the web site by adding another form page which would incorporate the appropriate SQL query.

As the Internet continues to grow, the Society may even want to consider creating a subset of the online "virtual" magazine discussed in Chapter V's "Other Plans" section. Online publishing of letters to the editor can be fully supported with the system recommended by this thesis. More importantly, this will serve a test bed for the creation of a complete online magazine in which all paper submission, review and publishing occurs on the Web. The bottom line is that the Society must resolve to grapple with Web/DBMS link issues and begin its transition to providing content on the Web if it is to remain relevant in the electronic age.

4. Impact of Planned Software Upgrades

Finally, according to information located on its Web site, Microsoft is planning to release major upgrades to its software in the first quarter of CY97. We have investigated the press releases surrounding these upgrades and when this occurs, the functions currently offered by the IDC, dbWeb and Internet Assistant will be embedded in NT Server and

91

Access97 respectively. This will not change any of this thesis' recommendations and will,

in fact, actually serve to further ease the Society's transition to the Web.

VII. SUMMARY AND CONCLUSION

A.   SUMMARY

The objectives of this thesis were to

- create a taxonomy of existing dynamic Web\database linking methods

- examine the issues surrounding this linkage of a database to Web

- investigate a case study linkage problem

En-route to meeting these objectives, a thorough review of currently available and utilized methods, protocols and standards for connecting a relational database to the Web was conducted. As work progressed, the information collected through the course of this review revealed that the myriad of link products currently available actually fall within a small number of general categories, each of which has it own strengths and weaknesses with respect to the issues underlying the connection of a database to the Web.

The use of a case study and an analysis of its associated problems offered an opportunity to investigate the potential use of each general link category in a prototypical situation. The ensuing discussion of an actual design for implementing the most appropriate of the potential approaches for the case study revealed the validity of the taxonomy and universal nature of the issues. The suggested design delivered all of the benefits expected from the creation of a Web/database link while addressing all the underlying link issues.

Finally, the database technology research conducted in this thesis may actually become rather quickly dated due to the speed with which Internet-related technologies are being developed. This is true because database, operating system, and middleware product vendors all approach the link issues from different directions and they are all in competition to strengthen the connectivity between the Web and database applications. With the marketplace driving the rapid pace, the possibility of new categories for link approaches being developed is quite high.

B.   ADDITIONAL RESEARCH AND DEVELOPMENT

Although this study has amply illustrated the viability of the different Web/database link approaches and even provided a framework for implementing one of these link approaches, the possibility exists for further research. By exploiting the power of the Internet, a change can be made from the postal delivery of paper media to video presentation of electronic content from points both internal and external to a user's computer. As a result, it may no longer be necessary for people to gather in one geographic location to create and maintain a database which is linked to the Web. This thesis provides a framework and cornerstone for ongoing work which may achieve the goal of developing a DBMS which is completely disconnected from geographical limitations.

APPENDIX A. SUMMARY OF CONVERSATION

On Wednesday, 27 November 1996, the following information was transcribed by

LCDR Dean during a telephone conversation with:

Mercy Kowalczyk,
Executive Director
IEEE Signal Processing Society
445 Hoes Lane
PO Box 1331
Piscataway, NJ. 08855-1331

A.   THE REVIEW PROCESS

1.   Current Situation

The Society's current manuscript review process is as follows:

- Authors desiring publishing must suggest an appropriate EDICS code for

  their paper based upon the manuscript's subject matter and mail seven copies

  of their manuscript, abstract and references to the Society.

- Based upon the suggested EDICS code, Society staff chooses one of the

  code's associated AE's and forwards six of the seven manuscript copies.

- The AE compiles a list of potential reviewers from related publication

  reference lists, word-of-mouth recommendations, and personal acquaintances,

  and then attempts to enlist their services. Of note, some reviewers use

  students to perform "ghost reviews" that reviewers subsequently check.

- Once two or three contactees agree to review, the AE mails them copies of the manuscript.

- The reviewers read the manuscript, annotate suggested changes, and mail the papers back to the AE for forwarding to the author who remains "blind" to his paper's reviewer identities.

- The author responds to the suggested changes by either incorporating them outright or debating them (via the AE) with the reviewers. This may take several iterations prior to reaching total agreement.

- Once the author and reviewers agree on the paper's content, the AE must decide whether to publish or reject the paper, notify the author, and inform the Society of his decision.

2. Problems

Several problems are associated with the current review process:

- AE's usually do not have much administrative support and therefore tend to sit on the manuscripts.

- Since reviewers are extremely hard to find, the tendency is for multiple AE's to repeatedly contact the same reviewers.

- Many knowledgeable and potentially willing reviewers remain unidentified by AE's due to lack of publishing or separate social/professional circles.

- A general call to the public to identify additional reviewers may generate too many responses from unqualified people or those whose qualifications would be difficult to vet.

- Some authors publish regularly but are either unwilling to review the work of others or are not identified as potential reviewers. These members need to be identified and encouraged to help out.

3. Changes Underway

Several changes to the current process are being implemented. First, instead of forwarding six copies of each manuscript to the AE for his subsequent re-mailing to reviewers, the Society plans to wait until the AE has provided a list of reviewers and then mail manuscript copies directly to the reviewers.

Secondly, the Society has hired an outside consultant to create a Microsoft Access95 database application to do/aid the following:

- Log in submitted manuscripts as they arrive

- Automatically generate a receipt letter to send to the author

- Automatically generate a list of AE's responsible for the manuscripts EDICS code.

- Track AE assignments and limit AE responsibilities to no more than three manuscripts per month

- Provide AE's with suggested reviewer names.

Finally, reviewers will be required to pre-agree to complete the review within six weeks of manuscript receipt to ensure manuscripts are published on a timely basis.

4. Future Plans

The Society is considering the acceptance of electronically submitted abstracts and reference lists that can be forwarded to the AE's and thereby offer them a head start in identifying reviewers. Once identified, the AE will send the reviewer names to the Society which will mail them manuscript copies. It is hoped that this new procedure will reduce the amount of time that manuscripts spend awaiting assignment or in transit and thus speedup the entire review process.

Allowing AE's Web access to the new database is also being considered and thus the Society's support for this thesis as an initial investigation into the relevant issues. The Web link has yet to be started due to concerns over maintaining data integrity and reviewer confidentiality from the authors. This is exasperated by the fact that, in some cases, AE's have responsibility for the very EDICS code to which their own papers have been assigned. Because of these concerns, the following in-variants for any Web/database link were stated:

- AE's need to search for and access reviewer names and contact information contained in the new database must be limited to protect database integrity and the confidentiality of the review process.

- Only Society staff and the accountable AE shall know of a particular author/reviewer grouping.

- The Society's intellectual assets must be protected from competitors.

- New reviewers must be able to register with the Society in a manner that will facilitate Society verification of qualifications and familiarity with review procedures.

APPENDIX B. USER QUESTIONNAIRE

On December 2, 1996, this questionnaire was faxed to:

Mercy Kowalczyk,
Executive Director
IEEE Signal Processing Society
445 Hoes Lane
PO Box 1331
Piscataway, NJ. 08855-1331

The purpose of the questionnaire is to re-address (in detail) certain points first brought to light in Appendix A, specifically those concerning the new database and the potential use of a Web/database link facilitating the solicitation of new manuscript reviewers. Her responses on December 5, 1996 are written below each question in *italics*.

A.   HARDWARE

- What computer hardware is the new database implemented on?

  CPU: 133 MHZ Pentium
  RAM: 16 MB
  CD-ROM: Yes
  Hard Drive: 1 GB
  TAPE BU: Yes
  MODEM: Yes
  NETWORK INTERFACE CARD: Yes

- Is the computer currently connected to the Internet?

  Yes, via an internal cross-departmental Ethernet-based Windows For Workgroups 3.11 peer-to-peer LAN. This may change since the Journal staff desires to separate from the enterprise level LAN for privacy reasons and to facilitate an impending move to another physical location.

- How much hard drive disk space is available for additional Web/database link software?

  Unknown

B. SOFTWARE

- Assumed: the reviewer database is implemented with Microsoft Access95.

  Yes

- Assumed: Windows95 is the operating system.

  Yes

- Is any other software being used to implement the database?

  No

- Will the computer be dedicated solely to the database?

  It is not now. The database may be moved to a dedicated PC server so the appropriate staff can have unrestricted access. Ultimate plan is to give only three staff members password enabled access to the database. An additional requirement will be for remote D.B.A. access due to one staff member requiring the ability to work from home.

C. DATABASE

- What data are held in the database for each reviewer, author, and Associate Editor (AE), and how are they individually identified?

  Note: the fields listed below differ from the ones actually documented in [IEEE96-2] but are included for transcription accuracy purposes.

NAME   *(Last, First) ==>Primary Key*
AUTHOR   *(yes/no)*
REVIEWER   *(yes/no)*
AE   *(yes/no)*
MAIL ADDRESS
TELEX
PHONE
EMAIL
FAX
MISC   *(this is where the associated Editor's Information Classification Scheme (EDICS) codes are entered)*

- How many of each type of person will be entered in the database?

Approximately 40 AE's activated on a daily basis for each of three Transactions (Digital, Audio, Image). This equals approximately (?) for the whole database.

- What other TABLES are in the database and what are their fields?

There are at least two other tables. Neither one appears to be normalized as described by Ms. Kowalczyk.

For the EDICS table (Primary Key unknown) the fields are:
TRANSACTION   *(the three Transaction Journals share the same code numbers but the subject matter of each code number changes from Transaction to Transaction)*
EDICS CODE
EDICS SUBJECT
AE NAME

For the MANUSCRIPT table (Primary Key unknown) the fields are:
AUTHOR
TITLE
CONTROL NUMBER ==>Primary Key (5-digit number preceded by A, I, D to signify associated Transaction)
DATE RECEIVED
INITIAL PAGE COUNT
AE ASSIGNED
DATE FORWARDED TO AE
DATE SENT TO REVIEWERS

REVIEWER NAMES
PUBLISH DECISION   *(yes/no)*
DECISION REASON
DATE OF DECISION
DATE TO BE PUBLISHED
FINAL PAGE COUNT
DATE FINAL MANUSCRIPT RECEIVED

NOTE: a record is created for each manuscript received whether it is kept for review or rejected outright by the staff for format reasons.

- Are EDICS/AE relationships captured by the database?

  Yes, by a JOIN(?) between the EDICS and PERSON tables

- Are AE/reviewer relationships captured by the database?

  Yes, by a JOIN(?) between the MANUSCRIPT and PERSON tables

- What specific data for reviewers, authors and AE's must be <u>accessible</u>

  <u>only by office staff</u>?

  All

- What specific information must be captured on people signing up to be a

  reviewer?

  All fields in the PERSON table plus:
  EDUCATION
  IEEE MEMBER NUMBER/STATUS
  BIOGRAPHY? (suggestions include papers/articles published, special
  expertise, etc.)
  REFERENCES WITH CONTACT INFORMATION
  EDICS CATEGORIES WANTING TO REVIEW

- What are the top pieces of information the Society would like to have on newly signed up reviewers to ensure credibility, authenticity, and motivation of the individuals?

  References, educational certifications, and which EDICS categories they are willing to review for. The latter is important because there are trends and fads in DSP subject matter such as fuzzy logic, wavelets, and multimedia while other areas such as underwater are rather esoteric, all leading to the continual changing of EDICS specific workload.

- What information will be made available to the Internet?

  None, total separation of database and Internet is desired. Anticipated use is to have potential reviewers submit information and then have the staff conduct an initial review, forward the applicant's information to an appropriate AE, and have the AE vet the applicant and inform the Society whether the applicant should be added to the reviewer database.

  The problem with giving AE's access to the database is that they are only appointed (on a voluntary basis) for a short term (long enough to oversee the review of 25 manuscripts). Since there is a revolving door of AE's and a high percentage of them are authors, access becomes problematic. Furthermore, due to the small nature of the staff and supporting budget, it is absolutely imperative that the database not be sullied by inadvertent mistakes or malicious intent. The two absolutes are:
      1- to preserve/protect database integrity
      2- to protect sanctity and confidentiality of review process

- Will the information be tailored to specific types of accessors?

  No. This question is irrelevant since the Society does not desire to connect its database to the Internet.

- Are there any statistics that the database must keep?

  Author, reviewer periodicity, ergo, how many times reviewer has been used.

- Can I have a copy of the database tables with fictitious data?

  Possibly forthcoming, time-line unknown

- ASSUMED: Web connected database information will be READ only and

  no direct WRITE capability will be granted.

  This is the crucial matter that is addressed by this thesis. Although Mercy feels personally that the answer should be FALSE, and that no link should be allowed between her database and Internet, she is willing to discuss the matter as she recognizes that others within the IEEE organization may feel differently.

- ASSUMED: Individually submitted reviewer information will be checked

  and entered by office staff to maintain database integrity.

  FALSE, after initial Society review for completeness, the data will be forwarded to an appropriate AE for vetting. This will potentially require an additional field in PERSON table to enter the reviewer's qualifying AE's name.

D. FINANCIAL

- If the decision is made to make certain information available from the

  Internet, will additional money be available, and if so, how much?

  Unknown, depends upon the worthiness of the project, will not be a large sum.

E. PEOPLE

- Who will access the database in the office?

  Appropriate staff

- Who will maintain the database and ensure that the data is correct?

  Appropriate staff

- Who does the Society want to <u>access the database from the Internet</u>?

  Nobody (see discussion in next question)

- Who does the Society <u>not want to access the database from the Internet</u>?

  As stated in section C. above, this a contentious point. Mercy feels that everybody should be excluded. What is not clear at this time is whether there truly are a lot of people out there who are ready, willing and able to review and yet they are unknown to the Society. As an aside, a lot of people are uncomfortable putting personal information on the Internet and this may impact the willingness of potential reviewers to submit a resume over the Internet.

F.   TIMING

- Does a deadline exist for Internet connectivity?

  No

G.   MISCELLANEOUS

- How is an EDICS code assigned to a newly submitted paper?

  Suggested by the author

- Does the Society foresee extensions to the currently planned use of the database as a central reservoir of reviewer information. In other words, what else is planned?

  In January of 1997 the Society will begin publishing the letters to the Journal on the Internet with the hope of eventually offering online subscription/access to the Journal itself. What is not known at this time is how the current income from subscription fees will be replaced. Since the subscription fees are the lifeblood of the Journal, this is a very serious concern.

  Finally, Ms. Kowalczyk re-iterated her desire and willingness to aid this case study's research phase, and also stated her understanding and willingness to work within the time constraints of the thesis completion schedule.

LIST OF REFERENCES

[Ayre96]   Ayre, R., Mace T., *Just Browsing*, PC Magazine, March 1996.

[Bina94]   Bina, E., et al., *Secure Access to Data Over the Internet*, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, IL, 1994.

[Brenner96]   Brenner, S. E. & Aoki, E., *Introduction to CGI/PERL*, M&T Books, New York, NY, 1996.

[Burleson94]   Burleson D. K., *Managing Distributed Databases, Building Bridges to Database Islands*, John Wiley & Sons New York. NY, 1994.

[Centura96]   *Challenges in Next-Generation Client/Server Application Development and Deployment*, Centura Corporation, http:// www.gupta.com /centweb /wp_int /wp_inet.htm, October 1996.

[DataRamp96]   DataRamp Inc., White Paper, http:// dataramp.com /white /white.htm, May 1996.

[Dean96]   Dean, D., and others, *Java Security: From HotJava to Netscape and Beyond*, Princeton University, Princeton NJ, 1996.

[Dobson95]   Dobson, S.A., Burrill, V.A., *Lightweight Databases*, Rutherford Appleton Laboratory, Chilton, DIDCOT, UK, 1995. http:// www.cis.rl.ac.uk /people/sd/sd_refs.html, October 1996.

[Elmsari94]   Elmsari R., Navathe, S. B., *Fundamentals of Database Systems,* 2nd Edition, Benjamin/Cummings Publishing Company, Redwood City, CA, 1994.

[Fargues96]   Conversation between Professor M. Fargues and LCDR A. Dean, October 1996.

[Finkelstein96]   Finkelstein, R., *The Need for Versatile Application Development Tools In The Changing World of Client/Server and INET Development*, Performance Computing Inc., http:// www.arborsoft.com /papers /finkTOC.html, October 1996.

[Francett96]   Francett, B., *DBMS Response Time Tempers Web Strategies*, Software Magazine, September 1996.

[Frank95]   Frank, M., *Database and the Internet*, DBMS Online, December 1995. http:// www.dbmsmag.com, October 1996.

[Freeman96]   Freeman, E., *Middleware: Link everything to anything*, Datamation, October 1996.

[Gaffney96]   Gaffney, J., *Ilustra's Web Datablade Module*, About Informix White Papers, http:// www.informix.com, October 1996.

[Gundavaram96]   Gundavaram, *S., CGI Programming on the World Wide Web*, O'Reilly and Associates, Sebastopol, CA, 1991.

[Gutierrez96]   Gutierrez, D. D., *Java-Boost Your Databases*, Databased Advisor, November 1996.

[Hamilton96]   Hamilton, M. A., *Java and the shift to Net-Centric Computing*, Computer, August 1996

[House96]   House, B., *The New Life of Middleware*, Databased Advisor, October 1996.

[Hughes95]   Hughes, L. J., *Actually Useful Internet Security Techniques*, New Riders Publishing, Indianapolis, IN, 1995.

[Hughes96]   Hughes, L. J., *Guarding the Castle, Internet Security*, a supplement to INFO Security News, October 1996.

[IEEE96]   *IEEE Transactions on Signal Processing*, IEEE Signal Processing Society, Piscataway, NJ, 1996.

[IEEE96-2]   *User Manual of Publications Administration Database System*, IEEE Signal Processing Society, Piscataway, NJ, 1996.

110

[Informix96]   *The Web Changes Everything*, About Informix White Papers, Informix Corporation, http://www.informix.com, October 1996.

[Jeffery95]   Jeffery, K. G., *Database: Introduction to Problems*, Rutherford Appleton Laboratory, Chilton, DIDCOT, UK., 1995, http://www.cis.rl.ac.uk/people/sd/sd_refs.html, October 1996.

[Jordan94]   Jordan, L., Churchill, B., *Communication and Networking For The PC*, New Riders Publishing, Indianapolis IN, 1994.

[Lambert96]   Lambert, C., *ODBC: Architecture, Performance, and Tuning*, Microsoft Corporation, http:// www.microsoft.com /odbc /wpapers /odbcperf.htm, October 1996.

[Lemay96]   Lemay, L., *teach yourself Web Publishing with HTML 3.0*, Sams.net Publishing, Indianapolis, IN, 1996.

[Linthicum96]   Linthicum, D. S., *The JDBC Connection*, Internet Systems, October 1996.

[Lynnworth96]   Lynnworth, A., *excerpt from Delphi 2 Unleashed*, http:// www.href.com, October 1996.

[Marshall95]   Marshall, V.A., The World Wide Web, Rutherford Appleton Laboratory, Chilton, DIDCOT, UK, 1995. http:// www.cis.rl.ac.uk /people /sd /sd_refs.html, October 1996.

[McGee96]   McGee, M., Web Pages: A Programmer's Perspective, Microsoft Corporation, http:// www.microsoft.com /intdev /prog-gen /webpage.htm, November 1996.

[Microsoft96]   *Custom Controls*, Microsoft Corporation, http:// www.microsoft.com /msdn/sdk/platforms/doc/sdk/win32/ctrl/src/controls_4, October 1996.

[Microsoft96-2]   *Internet Server API Overview*, Microsoft Corporation, http:// www.microsoft.com /win32dev/apiext/isalegal.htm, October 1996.

[Microsoft96-3]   *Microsoft dbWeb, in a Nutshell*, Microsoft Corporation, http:// www.microsoft.com /intdev /dbweb/dbweb.htm, October 1996.

[Minasi96]   Minasi, Mark, et al., *Mastering Windows NT Server 3.51*, Sybex
                    Network Press, San Francisco, CA, 1996.

[Murdoch96]   Murdoch, J., *Active-X and Thin-Client Computing*, Databased
                    Advisor, October 1996.

[Netscape96]   *Inline Plugins*, Netscape Corporation, http:// home.netscape.com
                    /comprod /products /navigator /version_2.0 /plugins /index.html,
                    October 1996.

[Netscape96-2]   *Persistent Client State- HTTP Cookies*, Netscape Corporation, http://
                    home.netscape.com /newsref/ std /cookie_spec.html, October 1996.

[Orfali94]   Orfali, R., et al, *Essential Client Server Survival Guide*, John Wiley
                    & Sons, New York, NY, 1994.

[Plain96]   Plain, S. W., *Web Database Tools*, PC Magazine, September 1996.

[Ranade96]   Ranade, J., *Network Security Data and Voice Communications*,
                    McGraw-Hill, New York, NY, 1996.

[Reichard96]   Reichard, K., *Web Servers for Database Applications*, Internet
                    Systems, October 1996.

[Rob95]   Rob, P., Coronel, C., *Database Systems*, 2d ed, Boyd & Fraser
                    Publishing Company, Danvers MA, 1995.

[Rowe96]   Rowe, J., *Building Internet Database Servers with CGI*, New Riders
                    Publishing, Indianapolis, IN, 1996.

[Schmidt96]   Schmidt, D. C., and others, *Measuring the Performance of
                    Communication Middleware on High-Speed Networks*, presented to
                    the Symposium on Communication Architectures and Protocols
                    (SIGCOMM), Stanford University, ACM, http:// www.cs.wustl.edu
                    /~schmidt/corba-research.htm, October 1996.

[Schulzrinne96]   Schulzrinne, H., *World Wide Web: Whence, Whither, What Next?*,
                    IEEE Network, March/April 1996.

112

[Simpson96]   Simpson, A., Olson E., *Mastering Microsoft Access for Windows 95*, Sybex Inc., San Francisco, CA, 1994.

[Stallings94]   Stallings, W., *Data and Computer Communications*, 4th ed, MacMillan Publishing Co., New York, NY, 1994

[Stormcloud96]   *Adding Dynamic Intelligence to your Web site*, Stormcloud Development Company, http:// www.ndev.com:80 /ndc2 /products /reviewer.htp, October 1996.

[Varela94]   Varela, C. A., Hayes, C. C., *Providing Data on the Web: From Examples to Programs*, Department of Computer Science, University of Illinois at Urbana-Champaign, IL, 1994.

[Varney96]   Varney, S. E., *Datawebs! Link the Web to Your Legacy Data and Apps*, Datamation, April 1996.

[Vinoski93]   Vinoski, S., *Distributed Object Computing With CORBA*, Hewlett-Packard Company, Chelmsfird, MA, 1993.

[VisualWave96]   *Building Client/Server Web Applications, How Fast can your business respond?* ParcPlace-Digitalk Inc., http:// www.parcplace.com, October 1996.

[Yager96]   Yager, T., *NT and the Net*, Byte Magazine, July 1996.

[Yourdon96]   Yourdon, E., *Java, the Web, and Software Development*, Computer, August 1996

[Wall91]   Wall, L., Schwartz, R. L., *Programming PERL*, O'Reilly and Associates, Sebastopol, CA, 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code CS 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943

4. Dr. C. Thomas Wu, Code CS/KA 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943

5. Dr. Monique P. Fargues, Code EC/Fa 1
   Electrical and Computer Engineering Department
   Naval Postgraduate School
   Monterey, California 93943

6. LCDR A. Dean 2
   628 E. Bismark
   Seguin, Texas 78155