HyperSQL: Web-based Query Interfaces for Biological Databases*

Mark Newsome[1], Cherri Pancake[1], and Joe Hanus[2]
Department of Computer Science[1], Department of Botany and Plant Pathology[2]
{newsome | pancake}@research.cs.orst.edu, hanusj@mgd.cordley.orst.edu
Oregon State University, Corvallis, OR 97331
http://mgd.cordley.orst.edu/hyperSQL

*This work is sponsored by NSF Grants BIR 9503712 and 9402192.

Abstract

HyperSQL is an interoperability layer that enables database administrators to rapidly construct browser-based query interfaces to remote Sybase databases. Current browsers (i.e., Netscape, Mosaic, Internet Explorer) do not easily interoperate with databases without extensive "CGI" (Common Gateway Interface) programming. HyperSQL can be used to create forms- and hypertext-based database interfaces for non computer experts (e.g., scientists, business users). Such interfaces permit the user to query databases by filling out query forms selected from menus. No knowledge of SQL is required because the interface automatically composes SQL from user input. Database results are automatically formatted as graphics and hypertext, including clickable links which can issue additional queries for browsing through related data, bring up other Web pages, or access remote search engines.

Query interfaces are constructed by inserting a small set of HyperSQL descriptors and HTML formatting into text files. No compilation is necessary because commands are interpreted and carried out by our special gateway, positioned between the remote databases and the Web browser. Feedback from developers who have used the initial release of HyperSQL has been encouraging. At present, query interfaces have been successfully implemented for three major NSF-sponsored biological databases: Microbial Germplasm Database, Mycological Types Collection, and Vascular Plants Types Collection.

1. Motivation

Accessing information stored in biological databases helps speed research and is essential to solve complex problems requiring data from multiple sources. Such databases are not likely to be centrally located, because they are typically maintained under the control of individual scientists/collectors who know the limitations of the data.

Many large-scale biological collections are stored in Sybase databases (e.g., Genome Database, Microbial Germplasm Database, National Fungus Collection, Nature Conservancy Database). Sybase is a commercial relational database management system that employs SQL (Structured Query Language). SQL was adopted as the standard query language in the scientific community because it is an industry standard, is well-defined and has a solid theoretical foundation in set theory, and is implemented in successful and proven commercial products (i.e., Sybase, Oracle, Informix) that run on high-end UNIX workstations capable of managing large volumes of data.

Researchers in the biological sciences have not had an easy time retrieving information stored in SQL databases. There are four primary reasons for this difficulty. First, SQL requires that users remember too much information about database organization. Before users can formulate queries, they must remember or look up the names of tables and data items, which requires knowledge of the database's proprietary command language. Moreover, users must determine the logical meanings of the data items, which can be frustrating because of cryptic naming conventions and the use of synonyms. Often there is little more available than an uncommented listing of the database schema. These problems are exacerbated when tables in the database contain a large number of data categories.

Second, users are forced to use cryptic command-line tools (i.e., Sybase isql) intended for database administrators. These are based on proprietary command languages[1], present information in a low-level format, and are unforgiving of mistakes. Furthermore, to access

[1] Although the vendors of the most commonly used SQL databases (i.e., Sybase, Oracle, Informix) adhere to the SQL standard, they use incompatible tools, programmatic interfaces, and command languages.

1060-3425/97 $10.00 © 1997 IEEE

Oracle Ex. 1006, p. 1
Oracle v. DeBellis
IPR2024-00221

databases at remote sites, scientists must learn additional connection utilities (e.g., telnet, ftp).

Third, SQL itself is terse, error prone, and unfriendly. It is easy to misspell a keyword or data name, omit a "join" clause, or worse, construct a query that returns unintended results [Sme95]. We believe it is senseless to expect scientists—who have neither the time nor the desire to be experts in computer science—to become proficient in SQL programming and database theory.

Last, there is a lack of meaningful feedback during the query process [JV85]. The chances of a casual user formulating a query correctly on the first attempt are slim. With command-line SQL, the user must bear the burden of determining the exact nature of an error. Messages are cryptic and generally do not reveal the source of the problem.

Since SQL's inception in the 1970s, database vendors have virtually ignored the scientific community, instead building support tools for business and financial users (i.e., inventory, accounting, stock analysis). The need for query assistance tools targeting scientific users has been expressed by researchers at number of biocomputing workshops and conferences [FJP90, ZI94]. To fill the need, we worked closely with scientists from the Department of Botany and Plant Pathology to develop HyperSQL.

HyperSQL is a development tool for constructing Web-based query interfaces to SQL databases. Our goal is to make it easier for scientists to retrieve information from remote scientific databases using common Web browsers (Netscape, Mosaic, Internet Explorer), which lack the capability to interoperate directly with SQL databases. Our software makes it possible to layer browser-based query interfaces on top of normal SQL. No modifications to either the Web browser or the SQL database are required.

With our software, forms and hypertext-based interfaces eliminate direct exposure to SQL and low-level database tools. The user queries SQL databases by entering search criteria into forms. No knowledge of SQL is required because the interface automatically generates SQL queries from the user's input. Since the form is largely independent of SQL, it can use discipline-specific (rather than computer-related) terminology. The results of the query are formatted as hypertext, so they can include hyperlinks that can be selected to browse the database for related information, bring up other Web pages, or access remote search engines.

For query assistance tools, browsers offer several important features. They employ a conceptually simple hypertext model that even users with little or no computer training can easily learn and use. The browser serves to replace a number of low-level tools (i.e., telnet, ftp, gopher) and provide unified means of displaying data in a variety of formats (i.e., text, tables, images, sound, movies); furnish the features necessary to create database front-ends—tables and forms (i.e., text boxes, push buttons, radio lights, pulldown menus, scrolled lists)—and operate independently of both location and computer platform. Finally, they are available for free or at a low cost. The remainder of this paper is organized as follows. Section 2 demonstrates how HyperSQL interfaces are used; the following section shows how they are constructed. Section 4 shows the organization of query files, while Section 5 discusses the architecture and how HyperSQL is implemented. Section 6 summarizes related work by others. The last section discusses the current status of HyperSQL and our future plans.

2. How HyperSQL Query Interfaces are Used

HyperSQL query interfaces are constructed by inserting a small set of HyperSQL descriptors [NPHM96] and HTML (Hypertext Markup Language) [BLCL+94] into text files. No compilation is necessary because commands are interpreted and carried out by our special gateway, positioned between the remote databases and the browser software. Operation of the gateway is transparent and users need not be aware of its presence.

A HyperSQL query interface is composed of a **query menu**, **query form**, **results screen**, and one or more **browse screens** (Figure 1). The **query menu** serves as a starting point for users, presenting introductory information about the database and displaying a list of queries, organized by subject. Clicking on one of the query selections brings up a **query form**. HyperSQL automatically prompts the user for a password if one is required to gain access to the database.

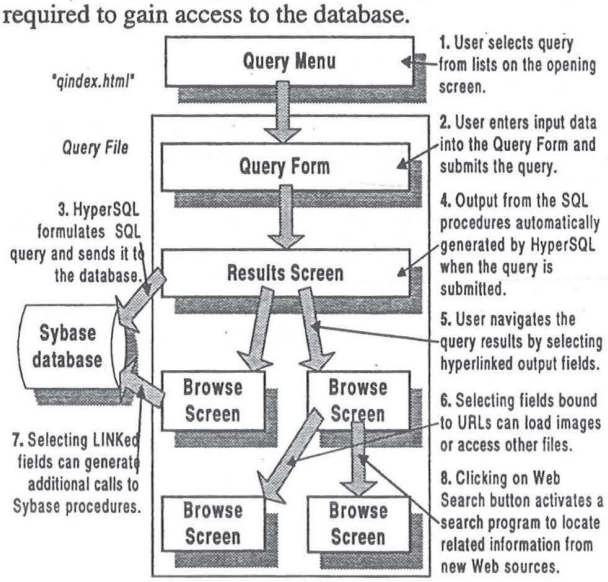


Figure 1. Operational model of the HyperSQL query interface

Oracle Ex. 1006, p. 2
Oracle v. DeBellis
IPR2024-00221

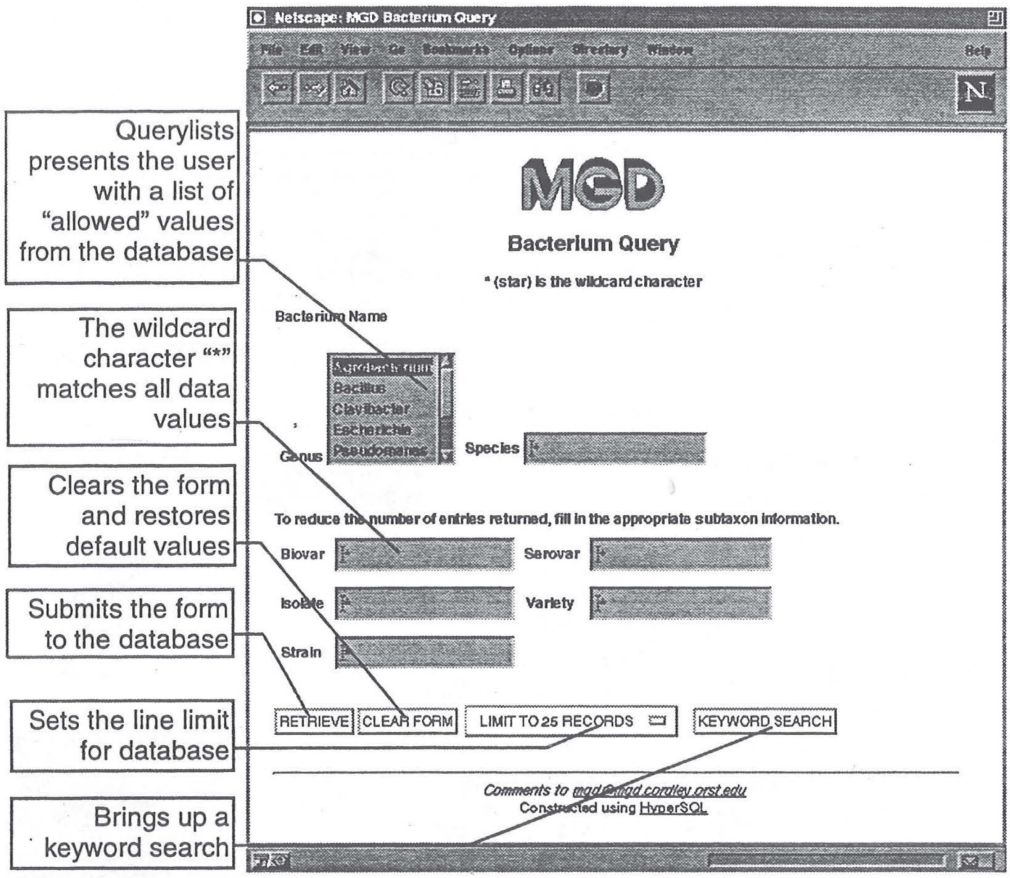


*Figure 2. Example of a **query form** (shown in a Netscape browser)*

The user enters search criteria on the **query form** (Figure 2). He/she can fill in as much information as available, or none at all; by default all text fields are supplied with wild-cards ("*"), meaning "match everything." A row of standard buttons are automatically provided at the bottom of the form. The `Retrieve` button submits the form and returns results. `Clear` erases any information entered into the form, replacing them with default values.

The pulldown menu located next to the `Clear` button is used to limit the number of results returned from the query; by default output is limited to 25 records. Clicking the `Keyword Search` button brings up a form that accepts a text string to be located by searching all data fields. For later reuse, forms containing favorite or frequently used sets of values can be saved by selecting "`Add to Bookmarks`" (or equivalent) in the browser.

HyperSQL query forms supports "query refinement," allowing the user to fill in fields incrementally by selecting from "querylists" (lists of allowed values). Querylists are useful when the user does not know what the database contains, or simply wants to select from a list of values rather than typing.

HyperSQL also supports keyword search, making it possible to locate records that contain keyword(s) within any data field (Figure 2). This mechanism provides an alternate way to begin; results are presented in the same hypertext-style format as results from a database search. Keyword search is particularly useful when the information desired is located in unexpected places or embedded in descriptions, notes, or memo fields. Query results are displayed as a list of "headlines" on the **results** screen (Figure 3). Headlines are single line summaries giving a general identifier for each database record that matched the query criteria. At this point, users may choose to explore one of the headlines (by clicking on it), or may return to the query form to revise the search. (It is always possible to return to the previous screen using the `Back` key provided on the browser). All forms and output screens can also be saved or printed using the browser's facilities.

Oracle Ex. 1006, p. 3
Oracle v. DeBellis
IPR2024-00221

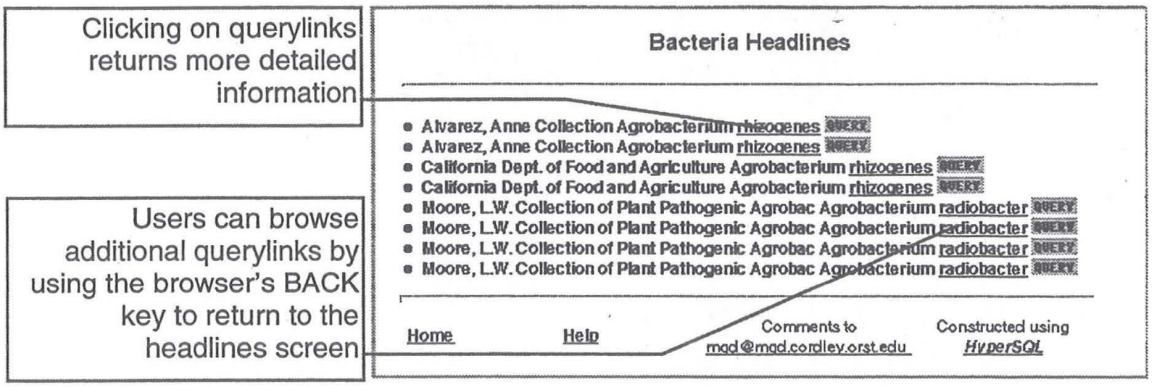


*Figure 3. Example of a **results** screen*

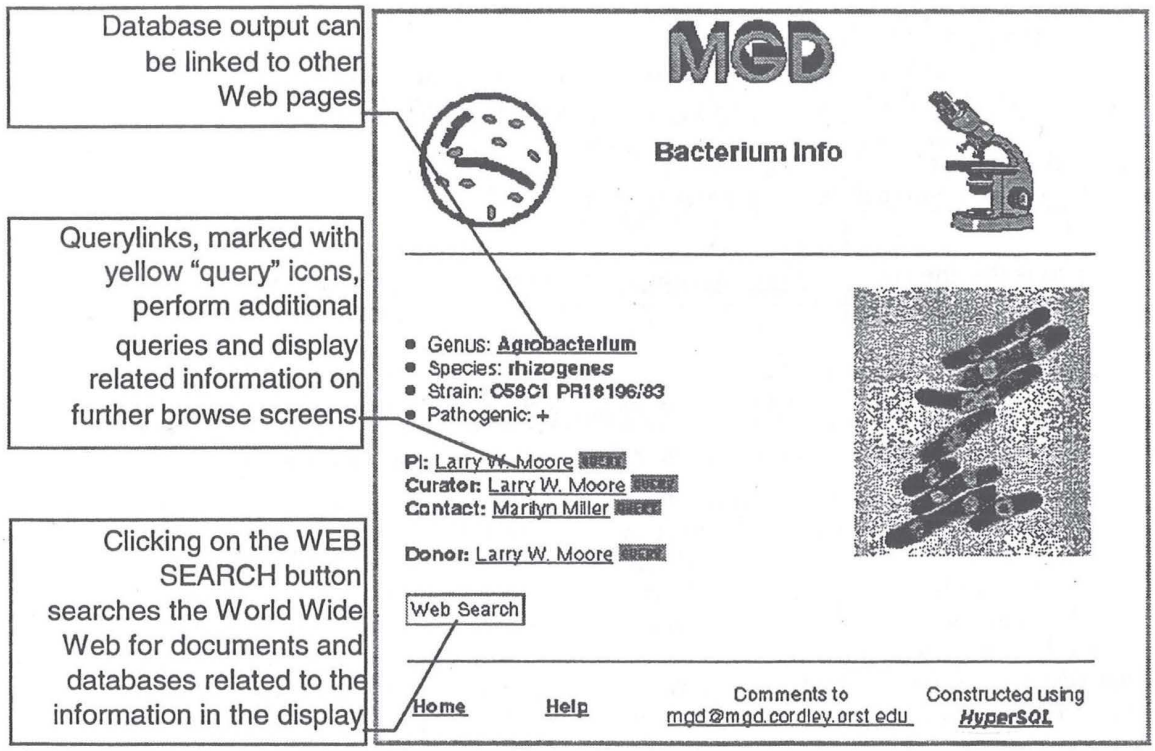


*Figure 4. Example of a **browse** screen*

Clicking on a headline brings up a **browse** screen with more detailed information (Figure 4). A **browse** screen also provides hyperlinks to access Web documents or other databases and software. In addition, HyperSQL introduces a special style of link called a *querylink* (marked with a yellow "query" icon, as shown in Figure 4), which activates a pre-formulated query to obtain more information from the database. Results from querylinks are displayed on additional **browse** screens. At the bottom of the **browse** screen is a Web search button that can be clicked to search the World Wide Web for documents and databases related to the information displayed on the screen.

3. How HyperSQL Query Interfaces are Constructed

This section presents an overview of how HyperSQL query interfaces are constructed. The first subsection discusses how **query forms** are built, followed by a subsection explaining how SQL is generated from the **query forms**. The final subsection describes how output is formatted on **results** and **browse** screens.

Oracle Ex. 1006, p. 4
Oracle v. DeBellis
IPR2024-00221

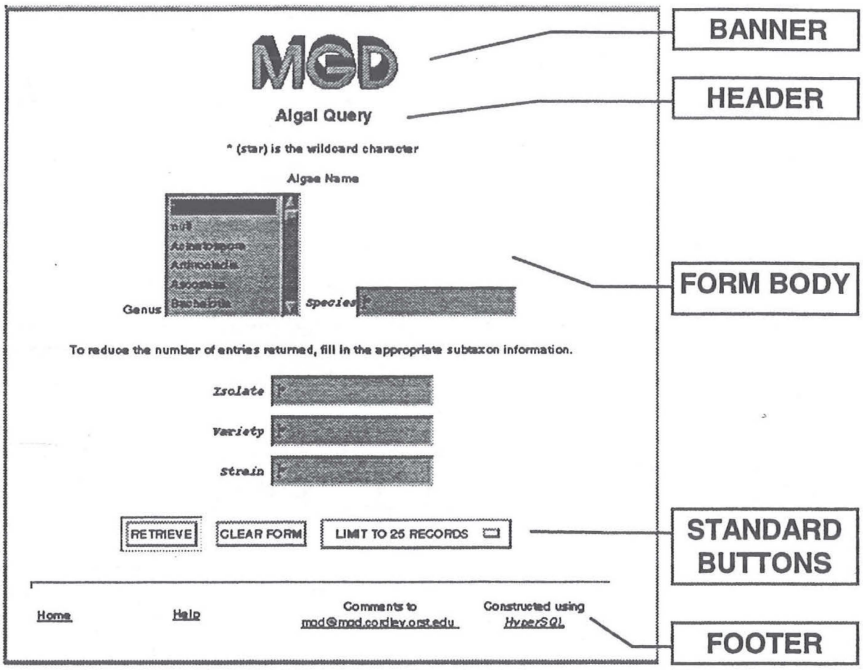


*Figure 5. Layout of a sample **query form***

3.1 Building a Query Form

Query forms are composed of six sections: banner, header, form body, standard buttons, and a footer; Figure 5 shows where these regions appear in the example query form. The layout of the **query form** is specified in the query file, using "query form" descriptors [NPHM96]. As their names suggest, banner, header, and footer descriptors are used to define what should appear in predefined decoration areas. Since the decoration descriptors accommodate any HTML text, the programmer can include hyperlinks, graphical imagemaps, tables, and formatted text in those areas. If a password is required for the database, HyperSQL automatically prompts for the password before bringing up the query screen.

The *banner* area is located at the top of the screen. It typically displays a logo or database name and is used on all screens in the interface. In contrast, the header (located below the banner) usually varies to reflect the nature of the information on each screen.

The *form body* section specifies what prompts and data entry fields should be presented to the user. HyperSQL's `INPUT` descriptor offers a variety of styles, including radio lights (lists of choices controlled by diamond-shaped buttons), buttons, pulldown menus, and querylists (scrollable lists that are filled with information acquired automatically from the database), as well as simple text entry fields. A row of *standard buttons* automatically appears below the form body. The function of the buttons was described in Section 2.

The *footer* area, located at the bottom of the query form, can be used for a variety of purposes: to display links to the query menu, or related pages; to identify the author and version of the software; or to furnish a button for sending feedback to the developers.

3.2 How SQL is Generated

When the user clicks the `Retrieve` button on the query form, the contents of the form are sent to the HyperSQL interpreter for processing. HyperSQL first establishes a connection to the remote database, using information from the environment section of the query file. (Since connections are short-lived, HyperSQL logs into the host computer and database for every query submission.) Depending on options specified in the "SQL" section of the query file, HyperSQL can either (a) invoke a specified SQL procedure already stored in the database's procedure cache, using a list of values from the query form, or (b) dynamically compose SQL code from the user input and the set of query directives specified in the query file. After transmitting the code to the database, HyperSQL awaits the results.

Errors from the underlying database and the network layer are trapped automatically and displayed prominently at the top of the screen. For debugging, HyperSQL provides a debugging descriptor which can be activated to display all SQL code generated by the HyperSQL interpreter.

Oracle Ex. 1006, p. 5
Oracle v. DeBellis
IPR2024-00221

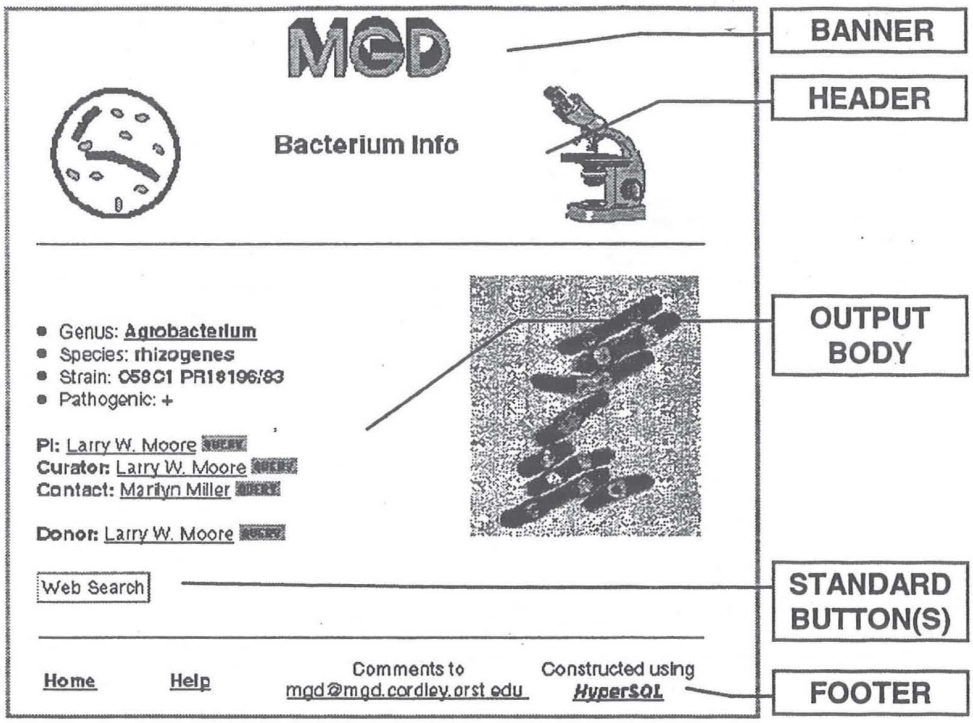


*Figure 6. Layout of a sample **browse** screen*

3.3 Formatting Database Output

Query results are formatted as hypertext, using "output" descriptors specified in the query file. There are two types of output screens, **results** and **browse**, controlled by separate sections of the query file. A **results** screen presents a list of single lines, summarizing the results returned from the database or keyword search engine. Data items in the output which cross-reference other information in the database are formatted as querylinks, a special kind of hyperlink designed to retrieve related or more detailed information from the database.

When the user selects a querylink, HyperSQL automatically invokes the associated SQL procedures with selected querylinks, passing the specified data fields as parameters. Since all querylinks on a results screen are statically associated with the same set of SQL procedures, the code is pre-stored in the database's procedure cache to improve response time. **Browse** screens display the results from following querylinks.

The same output form descriptors are used for both results and browse screens: *banner*, *header*, *form body*, and *footer*. Figure **6** show these regions on an example output screen. Note that output forms are very similar to query forms, except that the *standard buttons* are different and the form body displays output instead of blanks for entering input.

4. Query File Organization

A HyperSQL-based query interface is composed of a set of text files defining the screens. The query menu, which contains only HTML, resides in a file named `qindex.html`. The remainder of the files are called query files and their names have `.hsql` extensions. As a collection they provide complete specifications for a query interface (e.g., Bacterium Query, Fungal Query). Each file contains a combination of HyperSQL descriptors and embedded HTML, organized into five sections (Figure **7**). A query file must include exactly one each of the environment, query, and SQL sections, plus one or more output (results and browse) sections. The sections, delimited by BEGIN and END statements, must appear in order:

1. **ENVIRONMENT**: contains information for connecting to the target database and sets the banners for all interface screens.
2. **QUERY**: describes the layout of the query form. It specifies formatting for the form header, input fields on the form body, and the footer.
3. **SQL**: controls how SQL code is generated from information entered on the query form. Alternatively, this section can invoke a SQL procedure stored in the database, or an external script or program.

Oracle Ex. 1006, p. 6
Oracle v. DeBellis
IPR2024-00221

4. **RESULTS**: describes the layout of query output providing a list of result records. It specifies formatting for a header, query results, and footer.

5. **BROWSE**: describes the layout of screens containing detailed information. It specifies formatting of a header, data fields, and footer.

```
ENVIRONMENT BEGIN
  USEDATABASE mgd LOGIN=MGD PASSWORD=PROMPT;
  BANNER "<center><img src=/HYPERSQL/gif/mgd.gif></center>";
ENVIRONMENT END
QUERY BEGIN
  HEADER "<center> <H2>Bacterium Query</H2></center>";
  INPRINT "<B>Bacterium Name</B><br>";
  INPRINT "  Genus "; INPUT genus   TYPE=TEXTINPUT DEFAULT="Escherichia";
  INPRINT "Species "; INPUT species TYPE=TEXTINPUT DEFAULT="*";
  FOOTER "<hr>Feedback to mgd@mgd.cordley.orst.edu";
QUERY END
SQL BEGIN
  SUB genus   WHERELIST AS upper(Organism.genus)   like upper('$');
  SUB species WHERELIST AS upper(Organism.species) like upper('$');
  FROMLIST    Organism,  Headline_org;
  SELECTLIST  Headline_org.mgd_germplasm_record_num, headline;
  WHERELIST   Organism.mgd_germplasm_record_num = Headline_org.mgd_germplasm_record_num
        AND   Organism.germplasm_type = 'Bacteria';
SQL END
RESULTS BEGIN
 HEADER "<h1><center>Bacterium Headlines</center></h1>";
 OUTPRINT "<br>DATA_FIELD=mgd_germplasm_record_num DATA_FIELD=headline";
 LINK headline TO mgdGetBacterium(mgd_germplasm_record_num) BROWSE_SCREEN=1;
 FOOTER "<hr>Feedback to mgd@mgd.cordley.orst.edu";
RESULTS END
BROWSE 1 BEGIN
 HEADER "<h1><center>Bacterium Screen</center></h1>";
 OUTPRINT SUPPRESS_IF_EMPTY "<li>Genus: <b>DATA_FIELD=Genus</b>";
 OUTPRINT SUPPRESS_IF_EMPTY "<li>Species:<b>DATA_FIELD=Species</b>";
 OUTPRINT "DATA_FIELD=Organism_Role:<b>DATA_FIELD=Researcher_o<br>";
 OUTPRINT "DATA_FIELD=Collection_Role:<b>DATA_FIELD=Researcher_c<br>";
 FOOTER "<hr>Feedback to mgd@mgd.cordley.orst.edu";
BROWSE 1 END
```

Figure 7. Sample HyperSQL query file

5. HyperSQL Implementation

Figure 8 shows the architecture of HyperSQL. The software components of a HyperSQL query interface include the target Sybase database, the HyperSQL gateway, an HTTP (Hypertext Transfer Protocol) daemon, a set of screens (displayed by the browser), and an end-user, assumed to be a scientist or other person who may be unfamiliar with SQL and the target database. The initial version of HyperSQL supports databases using Sybase System 10 [Syb94b], a de facto standard among large-scale scientific databases. Sybase provides multi-database management capabilities and a programmatic interface based on SQL and the Open Client Library [Syb94a], a standard network communication protocol. The Open Client architecture makes it possible for the HyperSQL gateway and target Sybase databases to reside on different computers across the Internet. However, other database vendors have not adopted the interface and we plan to rewrite the communication interface using DBI (Database Interface) [Bun95], a freeware effort under development by a number of third-party database programmers. By taking advantage of DBI we will be able to support other databases, including Oracle, Informix, INGRES, mSQL, Empress, C-ISAM, DB2, Quickbase, and Interbase.

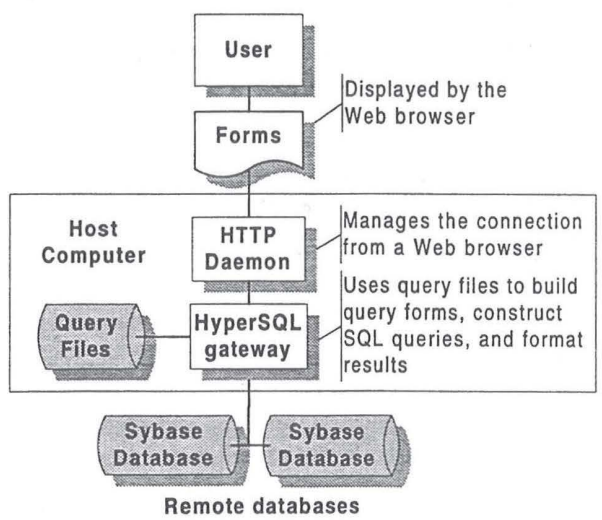


Figure 8. HyperSQL architecture

Oracle Ex. 1006, p. 7
Oracle v. DeBellis
IPR2024-00221

The core of the HyperSQL software is the *HyperSQL gateway*, composed of an interpreter and a communications back-end. The interpreter carries out the database operations described in the query files, transparently composing SQL queries, establishing RPCs (Remote Procedure Calls) to the remote databases, and formatting database results according to query file specifications (Figure **9**). The back-end module contains interface to the Open Client Library. Other communications drivers, such as the Microsoft's Open Database Connectivity and OpenLink's UDBC (Universal Database Connectivity) [Ope95] can be incorporated into the gateway by writing a replacement back-end.

The number of HyperSQL gateways in operation at a given time is limited only by host computer memory, the number of processes running, and the number of users simultaneously logged into the database. Each database transaction (submitting a query form or clicking on a querylink) invokes a new copy of the gateway, which terminates automatically after processing the query results. This style of interaction was chosen over maintaining a continuous login session, because it is not clear in a browser-based interface when a given user-session ends (i.e., interface screens may reside in the browser's cache indefinitely).

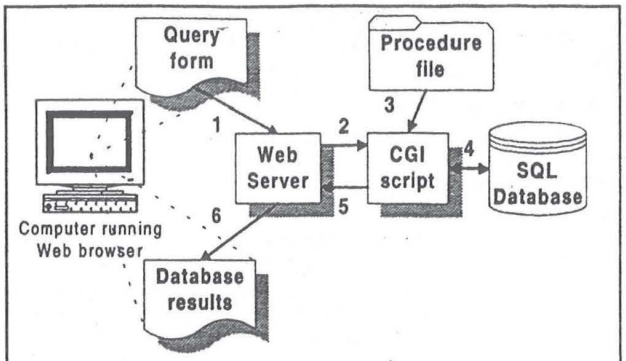
An *HTTP daemon* [NCSA95], available from NCSA and other sources, processes browsers' HTTP (Hypertext Transfer Protocol) requests received on a dedicated machine port (80 hex). The daemon automatically calls the gateway when HyperSQL services are required to build a screen or submit a query. There is a restriction that the daemon and gateway reside on the same machine, because the daemon invokes the gateway as a UNIX child process. The query file must also be accessible to the gateway software.

6. Related Work

Tool developers have constructed a variety of toolkits for constructing browser-based interfaces to remote databases. We have classified the toolkits into three categories, based on how query interfaces are specified: (1) interoperability languages (GSQL, WDB, WebinTool, W3-MSQL), (2) schema-based tools (Zelig, Genera/Web, UMASS Information Navigator), and (3) GUI-based environments (Cold Fusion, dbWeb, Sapphire).

6.1 Interoperability languages

Several toolkit developers use a language approach to interfacing Web browsers to remote databases. Developers construct query files by embedding special directives into HTML (Hypertext Markup Language) text files. Since all of the languages provide minimal features for building forms and generating queries, they differ primarily in number of features, syntax, and programming



1. The Web browser submits the query form to the Web server.
2. The Web server invokes the specialized CGI script commonly written in C or Perl), passing it the contents of the form.
3. The script reads the local procedure file describing how to formulate a query and format the results.
4. The script passes the query to the database and awaits results.
5. The CGI script formats the results into HTML and sends it to the Web server.
6. The Web server sends the results to the browser for display.

Figure 9. How CGI-based query interfaces work

"friendliness". The tools employ a special CGI interpreter to intercept the directives, carry out the corresponding database operations, and format database results into HTML. Figure **9** illustrates how this process works.

The NCSA's (National Center for Supercomputing Applications) GSQL [Ng95] is one of the first Web-to-database interface tools—all the other tools discussed here are considered derivatives of this work. Using GSQL, developers build forms-based query interfaces using a combination of HTML and GSQL commands embedded into "procedure" files. The tool introduced the concept of query directives, used by developers to specify how SQL is to be generated when query forms are submitted. By avoiding vendor-specific SQL extensions and providing a replaceable communications "backend," GSQL can be easily adapted to a variety of SQL databases. GSQL currently supports Oracle, Sybase, and Interbase, but lacks the ability to access remote databases. A restriction of GSQL is the requirement that the software reside on the same machine as the database.

A serious limitation of GSQL is its lack of output formatting. All results are returned as text; no provisions are provided for taking advantage of the browser's capability to display graphics, play sounds, or link output to other Web documents. Furthermore, there are no means of limiting the volume of database output, or linking database fields to related information in the database.

Similar to GSQL, WDB [Ras94] allows developers to specify query interfaces using a set of high-level form definition files, each describing a different view of the database. The WDB script, written in Perl, uses the definition fields to dynamically generate query forms

Oracle Ex. 1006, p. 8
Oracle v. DeBellis
IPR2024-00221

(where users enter query constraints) and format query results. The definition files permit embedded HTML, as well as calls to Perl functions, for additional formatting. WDB also supplies a utility for extracting the database schema from the database and automatically creating a working template form definition file. Like GSQL, the software must reside on the same machine as the database. WDB currently supports Sybase, Informix, and Mini-SQL, a freely available database which implements a subset of SQL.

WebinTool [Ins95] query interfaces are composed of a web page template and a set of macro directives (prefixed by periods), embedded directly into HTML text files. WebinTool offers a number of enhancements to GSQL: arithmetic operations, variable translation (i.e., uppercase, lowercase, escape, unescape), control directives (.IF .ELSEIF .ELSE, .STOP, .ABORT) and file directives for reading in a file for display (.READFILE). WebinTool's GSQL-style query directives (.SELECT, .FROM, .WHERE, .UNION, .SORT) specify how SQL is to be generated from user input from query forms. References to local and output variables are prefixed with a csh-style '$'; a different prefix, "@", is used to dereference input variables.

Improving on GSQL, WebinTool provides comprehensive support for formatting database output. Template directives (.TPLDEF ... .ENDTPL) specify how output column variables are to be printed (left/right adjusted; maximum length). Data items in the query results can be linked to related information in the database (.ARG LINK). The current version of WebinTool is restricted to machine containing the database software. It supports INGRES, Informix and Mini-SQL, and can support other SQL-based databases by replacing the backend.

Of all the GSQL-style languages, W3-MSQL [Hug95] contains the smallest number of descriptors. Programmers must understand UNIX and C-style concepts; for example, file seek and fetch operations and how to allocate and free handles. Unlike most of the other GSQL-style languages, W3-MSQL provides an "if" construct included primarily to alter output formatting based on the values of data fields returned from the database. W3-MSQL supports Mini-SQL databases and, like the other tools, must be executed from the same machine running the database software.

6.2 Schema-based tools

Each of the tools described in this section is driven by programmer-supplied "schemas," although the meaning of the term is used somewhat differently in each case. Zelig [VH94] uses "schema-based" specifications to guide forms generation. Here "schema" refers to a set of directives embedded into HTML files (coded as comments). The authors take a novel approach to monitoring user behavior in query interfaces. They embed an expert system, written in OPS5, into the interface to accumulate statistics on database access and monitor interface usage patterns. The rule-based module offers the database administrator advice on modifications to the underlying data structures for optimizing access time and storage requirements of the database system.

Another schema-based tool is Genera/Web [Let94]. Using Genera, developers specify query interfaces entirely in the tool's proprietary object-oriented schema language by defining and expressing relationships between "entities," groups of related information in the database. An entity is a collection of fields from one or more tables, views, or other entities, grouped together in a logical set. Fields are permitted to contain links to other databases. From the schema, Genera's compiler generates the HTML code for the query interface and the corresponding SQL procedures to be invoked by the query interface.

Genera, designed for Sybase databases, eliminates the need for developers to code directly in SQL and HTML, but at the expense of having to learn a complex and proprietary schema language. For developers who are already proficient with SQL, Genera introduces an unnecessary level of abstraction, which can be frustrating when the SQL generated from the schema produces undesired results. Although developers can edit the generated SQL stored in the database, Genera automatically overwrites all of the SQL code any time the compiler is invoked—even when simple cosmetic changes are made to the interface.

The third schema-based tool is the UMASS Information Navigator, [Hud95] designed to support easy navigation of relational databases. Hudson's tool generates query forms from a meta-data "schema," which contains information about the organization of the database. After entering search criteria into forms, the tool returns results which in turn can be clicked on to bring up more detailed information in the local database, on other Web pages, or from other databases. The Navigator is ideal for storing an entire Web-based information system. Because it generates Web pages dynamically, the Navigator avoids the "stale link" problem, a major source of frustration of Web users.

A configuration file contains the opening screen information, including the views contained in the meta-database. The Navigator, which supports both Oracle and Basis+, also permits users to use the results of relational queries as starting points for searching using full-text retrieval engines such as WAIS or INQUERY.

6.3 Query Interface Construction Tools

The tools presented in this section are designed to assist programmers in furnishing Web access to ODBC-compliant (Open Database Connectivity) software—which

Oracle Ex. 1006, p. 9
Oracle v. DeBellis
IPR2024-00221

includes SQL databases and spreadsheets—on Windows 95 and NT platforms. (Similar tools are not yet available on the UNIX platform.)

Cold Fusion [All95] is a CGI-based tool that reads template files to build query interfaces to Windows-based databases. In response to user queries, Cold Fusion creates dynamic HTML pages by mixing HTML tags and Database Markup Language (DBML) from the templates. DBML dictates how queries are formulated and database results are displayed. Using query interfaces created with Cold Fusion, end-users can both update and query databases by interacting with forms displayed on browsers. Cold Fusion provides a number of features targeting financial applications, including special formatting of currency, date, and time fields.

Aspect Software Engineering's dbWeb [Lau95] is a data-driven gateway between ODBC data sources and NT-based web servers implemented as a multithreaded NT service. Forms can be created for both updating and retrieving information from databases using dbWeb's administration tools. The tools generate a combination of HTML and special "tags," (similar to Cold Fusion's DBML) which specify the database operations to be performed by the gateway. At runtime, the gateway intercepts the tags, performs the specified database operations, and wraps the results in HTML for the browser. Database information which contains pointers to related data in the database are formatted as "smartlinks" in the browser, permitting users to navigate and browse the database by clicking links.

Bluestone's Sapphire/Web [Blu95], is a GUI-based programming tool for creating Web-based database interfaces for the three major SQL databases: Sybase, Oracle, and Informix. Sapphire, available for both the X Window System (not yet released) and Windows 95/NT, is similar in operation to Powersoft's PowerBuilder, in that both tools permit the user to build a database interface using direct manipulation. The tool generates C/C++ source code which can be modified or customized by the interface developer.

Sapphire provides no direct support for designing HTML-based forms. Developers can create HTML forms manually using a text editor or by using HTML authoring tools such as HotMetal, WebMagic, or the Internet assistant built into Microsoft Word. Application "objects" such as stored database procedures, dynamic SQL, functions, executables, files, and other objects (i.e., OLE) are dragged and dropped into the "Bind Editor," where HTML elements such as text input fields or drop-down menus are "bound" to arguments; database results are also bound to HTML elements for formatting. Once the interface is defined, the tool generates C and C++, which can be customized manually by programmers, who can introduce conditional processing or change the default method of populating the data. Although the interface does not entirely eliminate programming, it does offer a convenient means of assembling the components of CGI-based database interfaces.

When we started our work, only GSQL and Genera existed. Although GSQL demonstrated how browsers could interoperate with relational databases, it was limited to simple input forms, lacked means for formatting and cross-referencing output to related information, and was unsupported. The drawback to Genera was having to learn its high-level schema language; the tool automatically generates both the interface and SQL. The high degree of automation has a drawback—no provisions are made for queries for which the precise SQL is already known; the programmer must unnecessarily translate SQL into schema language.

Furthermore, most of the subsequently developed tools we have examined do not meet all of our users' requirements for publishing pre-existing, large-scale scientific databases, which are traditionally found on high-end UNIX workstations. In particular, the tools:

- do not connect to remote databases (must be run on the machine running the database) or cannot operate beyond a PC-based local area network;
- do not help users fill out the query form (e.g., displaying a list of allowed values);
- furnish no means of restricting access to sensitive data (e.g., password protection);
- lack means of cross-referencing database results to related data (i.e., querylinks) to help the user visually scan for information of interest.

7. Current Status and Future Work

At the time this paper was prepared, HyperSQL had been successful in building query interfaces in support of three major biological databases:

- Microbial Germplasm Database (MGD) [HNMP95]: searchable information describing microbial germplasm maintained in research collections throughout the U.S.
- Mycological Types Collection (MTC) [PHS96]: searchable descriptions of fungi type specimens.
- Vascular Plants Types Collection (VPTC) [LHP96]: searchable descriptions of vascular plants collections.

These databases have been accessed from more than 14 countries, including Brazil, Canada, Germany, Russia, Sweden, and the US. Feedback from programmers and database end-users have been encouraging. Additional databases are expected to come online using our software in the near future. We are also considering adapting HyperSQL to support Oracle and possibly other databases using DBI (Database Interface) [Bun95].

Oracle Ex. 1006, p. 10
Oracle v. DeBellis
IPR2024-00221

We have begun development of a next-generation Web tool called QueryDesigner, which will enable non computing professionals to locate, connect, and build personalized query interfaces to remote SQL databases. No knowledge of SQL, HTML, or HyperSQL will be required. Current information on QueryDesigner and HyperSQL, plus a list of supported databases are available at `http://mgd.cordley.orst.edu/hyperSQL`.

References

[All95] Allaire Corporation. *Cold Fusion White Paper*, 1995. [Online]. Available: `http://www.allaire.com/`

[BLCL+94] Tim Berners-Lee, Robert Cailliau, Art Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The World Wide Web. *Communications of the ACM*, 37(8):76-82, August 1994.

[Blu95] Bluestone Corporation. *Sapphire/Web manual*, 1995. [Online]. Available: `http://www.bluestone.com/products/sapphire/`

[Bun95] Tim Bunce. *The Database Interface DBI: Specifications*, 1995. [Online]. Available: `http://www.hermetica.com/technologia/DBI/`

[FJP90] James C. French, Anita K. Jones, and John L. Pfaltz. Scientific database management (final report). Technical Report TR-90-21, University of Virginia, Department of Computer Science, August 1990.

[HNMP95] Joe Hanus, Mark Newsome, Larry Moore, and Cherri Pancake. "The Microbial Germplasm Database." [Online]. Available: `http://mgd.cordley.orst.edu/cgi-bin/mgd`

[Hud95] Richard L. Hudson. *UMass Information Navigator*. Office of Information Technology, 1995. [Online]. Available: `http://www.crocker.com/~hudson/navigator.html`

[Hug95] David J. Hughes, *W3-MSQL Users Manual*. Hughes Technology Pty, Ltd., 1995. [Online]. Available: `http://hughes.com.au/product/w3-msql/manual-1/w3-msql.htm`

[Ins95] BBSRC Roslin Institute. *WebinTool*, 1995. [Online]. Available: `http:// anita.jax.org/webintool-0.921/docs/user-guide.txt`

[JV85] Matthias Jarke and Yannis Vassiliou. A framework for choosing a database query language. ACM Computing Surveys, 17(3):313-340, May 1995.

[Lau95] Jim Laurel. *dbWeb White Paper*. Aspect Engineering, Inc. August 1995. [Online]. Available: `http://www.aspectse.com/`

[Let94] Stanley Ian Letovsky. *Web/Genera*. John Hopkins Medical Institutes, Baltimore, 1994. [Online]. Available: `http://gdbdoc.gdb.org/letovsky/genera/genera.html`

[LHP96] Aaron Liston, Joe Hanus, Eric T. Peterson, "Vascular Plants Types CollectionDatabase". [Online]. Available: `http://www.orst.edu/dept/botany/herbarium/vasc_plant.html`

[Lin94] Paul Lindner. GopherSQL, A Gopher Interface to Relational Databases. In *Internet Gopher Conference*, Minneapolis, Minnesota, April 1994.

[NCSA95] NCSA. HTTPd documentation, [Online]. Available: `http:// hoohoo.ncsa.uiuc.edu/docs/Overview.html`

[Ng95] David Ng. *GSQL - a Mosaic-SQL gateway*. NCSA, Champaign, IL, 1995. [Online]. Available: `http://ox.ncsa.uiuc.edu/w/gsql/0gsql.html`

[NPHM96] Mark Newsome, Cherri Pancake, Joe Hanus, and Larry Moore. *HyperSQL User's Guide and Language Reference*, Revised February 1996 [Online]. Available: `http://mgd.cordley.orst.edu/hyperSQL`

[Ope96] OpenLink Software, Inc., 10 Burlington Mall Rd., Suite 265, Burlington, MA 01803, OpenLink ODBC Technical White Paper. [Online]. Available: `http://www.rockhopper.com/openlink`

[PHS96] Sherry Pittam, Joe Hanus, Joey Spatafora, "Mycological Types Collection Database." [Online]. Available: `http://mgd.cordley.orst.edu/hyperSQL/hsql/mct.html`

[Ras94] Bo Frese Rasmussen. WDB: A Web Interface to Sybase. In *Fourth Annual Conference on Astronomical Data Analysis Software and Systems*, Baltimore, September, 1994.

[Sme95] John B. Smelcer. User errors in database query composition. *International Journal of Human-Computer Studies*, 42:353-381, 1995.

[Syb94a] Sybase, Inc. 6475 Christie Avenue, Emeryville, CA 94608. Open Client-Library/C Reference Manual, Revised January 15, 1994.

[Syb94b] Sybase, Inc. 6475 Christie Avenue, Emeryville, CA 94608. Sybase SQL Server Reference Manual, Volume 1, Revised February 1, 1994.

[VH94] Carlos A. Varela and Carolyn C. Hayes. Zelig: A Schema-Based Generation of SoftWWW Database Applications. In *First World Wide Web Conference '94: Mosaic and the Web*, Chicago, September, 1994. NCSA.

[ZI94] Maria Zemankova and Yannis E. Ioannidis. Scientific Databases—State of the Art and Future Directions. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 752-753, 1994. Panel.

Oracle Ex. 1006, p. 11
Oracle v. DeBellis
IPR2024-00221

Wendt
S
H31
IN852
V.30
4

Proceedings of the Thirtieth Annual Hawaii International Conference on System Sciences

Edited by
JAY F. NUNAMAKER, JR.
RALPH H. SPRAGUE, JR.

1997

Vol IV

INFORMATION SYSTEMS – INTERNET and the DIGITAL ECONOMY

IEEE Computer Society Press

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Oracle Ex. 1006, p. 12
Oracle v. DeBellis
IPR2024-00221

Proceedings of the

Thirtieth Hawaii International Conference on System Sciences

Volume IV

Information Systems Track — Internet and the Digital Economy

KURT F. WENDT LIBRARY
COLLEGE OF ENGINEERING

APR 2 2 1998

UW-MADISON, WI 53706

Oracle Ex. 1006, p. 13
Oracle v. DeBellis
IPR2024-00221

Overview

Volume I — *Software Technology Track*

Object-Oriented Methods for Distributed Applications
Distributed Objects Platforms
Parallel and Distributed Algorithms
Multi-Threaded Systems: Issues, Solutions and Future
Coordination Models and Languages
Software Engineering for Distributed Systems
Distributed Persistent Architectures
Partitioning and Scheduling
Distributed Virtual Reality Environments
Distributed Multimedia and Collaboration over the Web

— *Computer Architecture Task Forces*

Combining General Purpose and Multimedia in One Package: Challenges and Opportunities
Configurable Computing: Concepts and Issues
Network Storage Architecture: Abstracting the Storage Interface
Wireless Networking - Reaching the Extra Mile
Interaction of Architecture and Compilation Technology for High-Performance Single-Processor Design
Architectural Trends for Shared-Memory Multiprocessors
Compiling for Processors with Instruction-Level Parallelism

Volume II — Information Systems Track: Collaboration Systems and Technology

A Vision for a New Discipline
Group Support Systems
Groupware Development
The Dynamics of Business Engineering
Organizational Memory
Groupware: User Experiences
Negotiation Support Systems
Distributed Group Support Systems
Technology-Supported Learning

Volume III — *Information Systems Track: Organizational Systems and Technology*

Measuring the Effectiveness of Emerging Technologies
Systems in Technology Transfer
Risks in Information Systems
Methods and Tools for Information Systems Development
Information Technology and Market Structure
Strategic and Competitive Information Systems
Creativity/Innovation in Information Systems Organizations
Organizational Decision Support Systems
Information Systems in Global Business
Management of Technology
Data Warehouses and Information Delivery Systems
Information Technology and Social Accountability

Volume IV — Information Systems Track: Internet and the Digital Economy

Computer-Mediated Inter-Organizational Systems
Telework and Organizational Connectivity
Telemedicine
Intelligent Agents on the Internet
Electronic Commerce
Virtual Collaboration
Internet and the Information Superhighway
Workflow Technology: Organizational and Interorganizational Dimensions
Formal Aspects of Digital Commerce

Volume V — Advanced Technology Track:

Engineering Complex Computer Systems
Techniques for Safety Critical Software Development
Emerging Paradigms for Intelligent Systems
Logic Modeling
Modeling and Decision Technologies
Complex and Chaotic Systems
Restructuring the Electric Power Industry: Emerging Issues, Methods and Tools

Volume VI — Digital Documents Track:

Genres in Digital Documents
Digital Documents in Organizations and the Workplace
Hypermedia in Information Systems and Organizations
Multimedia Information Systems

Proceedings of the

Thirtieth Hawaii International Conference on System Sciences

Volume IV

Information Systems Track — Internet and the Digital Economy

Edited by Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr.

Sponsored by
The University of Hawaii
College of Business
Administration

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

Oracle Ex. 1006, p. 15
Oracle v. DeBellis
IPR2024-00221

IEEE Computer Society Press
10662 Los Vaqueros Circle
P.O.Box 3014
Los Alamitos, CA 90720-1314

Copyright © 1997 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved.

Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries may photocopy beyond the limits of US copyright law, for private use of patrons, those articles in this volume that carry a code at the bottom of the first page, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Other copying, reprint, or republication requests should be addressed to: IEEE Copyrights Manager, IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society Press, or the Institute of Electrical and Electronics Engineers, Inc.

IEEE Catalog Number 97TB100234
ISBN: 0-8186-7743-0 (Softbound)
ISBN: 0-7803-4911-3 (Microfiche)
ISSN: 1060-3425
PR7734

Additional copies may be ordered from:

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: +1-714-821-8380
Fax: +1-714-821-4641
Email: cs.books@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: +1-908-981-1393
Fax: +1-908-981-9667
misc.custserv@computer.org

IEEE Computer Society
13, Avenue de l'Aquilon
B-1200 Brussels
BELGIUM
Tel: +32-2-770-2198
Fax: +32-2-770-8505
euro.ofc@computr.org

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN
Tel: +81-3-3408-3118
Fax: +81-3-3408-3553
tokyo.ofc@computer.org

Editorial production by Penny Storms
Cover by Joseph Daigle / Studio Productions
Printed in the United States of America by Sony Electronic Publishing Services

The Institute of Electrical and Electronics Engineers, Inc.

Table of Contents

Volume IV

Information Systems Track — Internet and the Digital Economy

Preface ix
Introduction x
Reviewers xii

Computer-Mediated Inter-Organizational Systems 2
Coordinators: Theodore Clark, Ho Geun Lee, and Donna B. Stoddard

Electronic Market Intermediary: Transforming Technical Feasibility into Institutional Reality 3
H.G. Lee

Electronic Commerce for Agricultural Transactions: Role of Intermediaries and Accurate Pricing 13
T. Matsuda, T.H. Clark, and H.G. Lee

Web-Enabled Cooperation in Tourism - A Case Study from the Region of Lake Constance 21
N. Tschanz and S. Klein

The Danish EDI Bandwagon Gaining Momentum 31
J. Damsgaard

Adoption of EDI by Small Business: Are the Advocates in Tune with the Views of Small Business - A Pilot Study 42
R.C. MacGregor, P. Waugh, D.J. Bunker, and J.F. Courtney

The Impact of Electronic Commerce on the Travel Industry An Analysis Methodology and Case Study 48
M. Bloch and A. Segev

Interorganizational Information Systems in Support of Manufacturing Networking 59
S.A. Sherer

Overcoming Barriers to Distributed Interorganizational Systems 69
C. Van Slyke, M.B. Prescott, and M. Kittner

Telework and Organizational Connectivity 80
Coordinators: Tung Bui, Kunihiko Higa, and Jerome Yen

Fragmented Lives: How Do Teleworking Parents Juggle Work and Children Care? 81
D. Hardwick and J.W. Salaff

Oracle Ex. 1006, p. 17
Oracle v. DeBellis
IPR2024-00221

Information Systems Managers' Perceptions of the Advantages and Disadvantages of Telecommuting: A Multivariate Analysis 94
T.S. Ellis and R.L. Webster

An Exploratory Analysis of Factors Toward the Emergence of Teleworking in Malaysia 99
A.A. Kadar Hamsa, L.C. Oniyirimba, and S. Ahmad

An Adoption Model of Telework for Organizations 110
B. Shin, K. Higa, and O.R. Liu Sheng

Group Decision and Negotiation Support in Telemedicine: An Application of Intelligent Mobile Agents as Non-Human Teleworkers 120
T. Bui and S. Sankaran

Analysis on Media Choice by Two Telework Groups 130
B. Shin and K. Higa

Development of Telework Research from the Finnish Perspective 131
A. Luukinen

Telemedicine 135
Coordinators: Olivia R. Liu Sheng, Kunihiko Higa, and Grace Au

A Group Decision Support System Framework for Medical Decision Making Incorporating Cognitive-Aid Structures and Cognitive Appropriation 136
G.R. Rao, B.A. Suresh, and M. Turoff

Organizational Adoption and Diffusion of Technological Innovation: A Comparative Case Study on Telemedicine in Hong Kong 146
K. Higa, O.R. Liu Sheng, P.J-H. Hu, and G. Au

Suggesting a Diffusion Model of Telemedicine -Focus on Hong Kong's Case 156
K. Higa, B. Shin, and G. Au

Object Oriented Teleconsultations in Global PACS Using Multi-Thread Java 166
Y-P. Yu and R. Martinez

Intelligent Agents on the Internet 177
Coordinators: Jerome Yen, Kar Yan Tam, and J. Leon Zhao

Intelligent Spider for Internet Searching 178
H. Chen, Y-M. Chung, M. Ramsey, C.C. Yang, P-C. Ma, and J. Yen

Intelligent Agents for Matching Information Providers and Consumers on the World-Wide-Web 189
J.K.W. Lee, D.W. Cheung, B. Kao, J. Law, and T. Lee

Virtual Proxy Servers for WWW and Intelligent Agents on the Internet 200
S. Wu and C-C. Liao

A Generic Virus Detection Agent on the Internet 210
J-S. Lee, J. Hsian, and P-H. Tsang

An Interactive Scheduling Agent on the Internet 220
B.P-C. Yen

Intelligent Agent Based Contract Process in Electronic Commerce: UNIK-AGENT Approach 230
J.K. Lee and W. Lee

Human Agents and Intelligent Agents: An Experiment on the Internet 242
J. Yen

Oracle Ex. 1006, p. 18
Oracle v. DeBellis
IPR2024-00221

wendt

AMN 4274

Electronic Commerce 253
Coordinator: Rick Oliver

Corporate Policies for Electronic Commerce 254
R.W. Oliver

A Framework for Targeting Banner Advertising on the Internet 265
K. Gallagher and J. Parsons

Banking on the Internet and its Applications 275
G. Yan, J.C. Paradi, and S. Bhargava

Virtual Collaboration 286
Coordinator: Alan R. Dennis

Groupware and the Internet: Charting a New World 287
A.R. Dennis and B.C. Wheeler

Designing a System for Cooperative Work on the World-Wide Web: Experiences with the BSCW System 297
R. Bentley and W. Appelt

Internet and the Information Superhighway 308
Coordinator: Robert W. Blanning and David King

Some Hypermedia Ideas for the WWW 309
M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian, and H. Oinas-Kukkonen

An Intranet Solution for a Real-Time GPMS in Newspaper Production 320
B. Hedin, F. Fällström, and V. Ionesco

HyperSQL: Web-based Query Interfaces for Biological Databases 329
M. Newsome, C. Pancake, and J. Hanus

Resource Discovery and Intelligent Image Retrieval in a Distributed Environment 340
S. Ram, S. Venkatsubramanyan, S. Marsh, and G. Ball

Developing Internet Security Policy for Organizations 350
S. Lichtenstein

Workflow Technology: Organizational and Interorganizational Dimensions
Coordinators: Pai-Chun Ma, Karl Reiner Lang, and Ted Stohr

Metagraph Transformations and Workflow Analysis 359
A. Basu and R. Blanning

Dynamic Workflow Management: A Framework for Modeling Workflows 367
M.M. Kwan and P.R. Balasubramanian

Hypermedia Document and Workflow Management Based on Active Object-Oriented Databases 377
G. Kappel, S. Rausch-Schott, S. Reich, and W. Retschitzegger

Using a Simulator for Testing and Validating a Newspaper Production Decision Support System 387
F. Fällström, S. Nordqvist, B. Hedin, and V. Ionesco

WWWorkflow: World Wide Web Based Workflow 397
C.K. Ames, S.C. Burleigh, and S.J. Mitchell

A Technology Adaptation Model for Business Process Automation 405
E.A. Stohr and J.L. Zhao

Oracle Ex. 1006, p. 19
Oracle v. DeBellis
IPR2024-00221

Workflow Reengineering: A Methodology for Business Process Reengineering Using Workflow Management Technology 415
S.M. Bitzer and M.N. Kamel

The Chautauqua Workflow System 427
C. Ellis and C. Maltzahn

Formal Aspects of Digital Commerce 438
Coordinators: Steve O. Kimbrough and Ronald M. Lee

The Role of Electronic Commerce Technologies in Just-In-Time Replenishment 439
R.B. Johnston and R.P.W. Lee

Electronic Brokering for Assisted Contracting of Software Applets 449
W.N. Robinson

Facilitating International Electronic Commerce by Formalising the Incoterms 459
A. Foekens, A. Mitrakas, and Y-H. Tan

A Messenger Model for Navigating Among Bureaucratic Requirements 468
R.M. Lee

Speech Acts in Electronic Communication with Special Reference to KQML and ANSI X12 478
M.A. Covington

Relativized Deontic Modalities for Contractual Obligations in Formal Business Communication 485
Y.U. Ryu

Modeling the Dynamics of Transferable Obligations in Business Procedures 494
Y-H. Tan and W. Thoen

On Electronic Commerce, Subatomic Semantics and Caesar's Stabbing 502
S.O. Kimbrough

Table of Contents — Volumes I, II, III, V, and VI 512

Author Index 536

Oracle Ex. 1006, p. 20
Oracle v. DeBellis
IPR2024-00221