Resistance is Futile: The Web Will Assimilate Your Database

Susan Malaika
IBM Santa Teresa Laboratory
Email: malaika@us.ibm.com

Abstract

Developers of relational database systems have made few concessions to the Web, preferring that Web applications access relational tables through SQL (or other programming interfaces) via intermediate gateway software. Result sets from database queries continue to be delivered to applications for further processing one row at a time. This article describes ways that relational databases are accessed on the Web through HTML and XML, and some system integration issues.

1 Introduction

Many of the relational database activities for the Web are in the following categories:

- *Store and manage Web resources in relational databases:* Storing Web related information, such as hypertext documents and hypertext links, in relational databases, and then making the information accessible through the Web.
- *Query and access Web resources using relational database techniques:* Making Web resources resemble structured data, often by augmenting them with metadata or by building indexes, and then querying, searching or mining the data using database query languages which can operate in conjunction with query optimizers.
- *Store and manage relational data on the Web:* Transforming tabular data in relational databases into formats immediately accessible on the Web and managing the transformed relational data in caches and data stores around the network.
- *Query and access relational data using Web techniques:* Accessing tabular data in relational databases from Web clients via server gateways or through software running in the clients, which is the main topic of this article.

2 The Web and Relational Databases

2.1 The Web

The Web provides easy human access to a variety of resources held on private and public networks. It is made up of client and server systems that communicate through agreed protocols. Servers deliver information in response

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

to client requests, usually without either server administrators or client users being aware of intermediate routing and caching systems that exist between clients and servers. Information delivered from servers can be created in advance by humans or programs *(static content)*, or at the time clients request information *(dynamic content)* and often dynamic and static content are combined. The next three sections describe major elements of the Web in the context of databases.

2.1.1 Universal Resource Locators (URLs)

URLs support a global naming scheme, that humans or programs can employ, for accessing resources such as files held at servers. A URL [7] encompasses the name of a resource as well as the protocol used to access a resource in an extensible way such as:

- *ftp://server:port/resourcename* where FTP is the protocol name used to retrieve a file.
- *jdbc:dbsystem://server:port/resourcename* where *dbsystem* is the name of a database system (such as Informix, Sybase). JDBC (Java Database Connectivity [12]) is a programming interface for Java applications to access relational databases. Because there is no agreed communication protocol from clients to relational database systems, the protocol name jdbc:dbsystem is not supported in general purpose products. The name of the client communication software (the driver name) and the database access request are defined within the client Java application, in addition to the JDBC URL.
- *http://server:port/virtualdirectory/owa/packagename.storedprocedurename* where the latter part of the URL is a database stored procedure request over the HTTP protocol, destined to an Oracle database through an Oracle Web server [6].
- *http://server:port/db2www/?langenv=dtw_sql(perform)&sql="select+*+from+opera"* where the latter part of the URL (known as a *database query URL*) is an SQL request over the HTTP protocol, destined to a DB2 [3] database through a Web server and Net.Data [19]. The result set is delivered in an HTML page.

2.1.2 Hypertext Markup Language (HTML)

HTML is a document markup language whose definition is expressed in an SGML DTD (Standard Generalized Markup Language Document Type Definition [8] [10]). Clients request HTML documents by specifying their URLs, and servers deliver documents to clients. In addition to markup tags for document structure (such as $<p>$ for a paragraph) and for presentation (such as $<hr>$ for a horizontal rule), HTML contains useful features such as:

- *Forms:* providing a means for humans and programs to supply input parameters to servers often to generate and tailor dynamic content. Input parameters are frequently delivered to server applications as name and value pairs separated by ampersands as in the database query URL example above where the variables are *langenv* and *sql*.
- *Hypertext Links* making it possible for humans and programs to navigate easily within and between static and dynamic content delivered from different servers. These unidirectional links refer to URLs and require no modifications in target resources. Link integrity is not usually monitored, other than through occasional use of link checker software.

Authors can incorporate product specific database constructs in HTML documents, that do not appear in the standard HTML DTD. Database constructs for HTML are usually transformed into regular HTML at servers just before delivery to clients. Here are some examples:

- *HTML Database Markup Tags :* With *Cold Fusion* [1], HTML documents can be extended with database markup tags and database queries to dynamically populate portions of the document with relational database content as illustrated below:

$$< cfquery\, name = "composer\_query"\ datasource = "world\_opera" >$$

$$select\ *\ from\, opera\, where\, comp = "Mozart"$$

$$< /cfquery >$$

- *HTML Database Programming Extensions :* With *Net.Data*, database queries can be incorporated in an HTML document without additional markup tags, using percent signs as delimiters, as illustrated below:

$$\%function(dtw\_sql)composer\_query()\{select\ *\ from\, opera\, where\, comp = "Puccini"\%\}$$

2.1.3 Hypertext Transfer Protocol (HTTP)

HTTP [11] is a client server, content neutral, Internet communication protocol that runs over TCP/IP. A server URL is associated with each HTTP request. There are a number of HTTP request types including:

- *GET* To retrieve static or dynamic content which could be HTML documents, multimedia files, code fragments such as Java applets or ActiveX controls. Any input parameters are sent from the client to the server as part of the corresponding URL.
- *POST* To retrieve dynamically generated resources. Parameters are sent separately and not in the URL. Pages created through POST are not usually cached in intermediate caching servers.
- *PUT and DELETE* To add, replace or delete resources held on servers. Web standards do not describe the behavior of server systems should multiple users attempt to update resources concurrently, nor is the concept of a collection of updates (a transaction) defined.

One of HTTP's virtues is statelessness which means that servers are not required to associate consecutive requests from the same client and to maintain client context on the server side. Statelessness makes HTTP servers simpler to write, makes intermediate network caching of the results of an HTTP request more straightforward, makes independent client cache navigation possible, and is well suited to occasional access from humans when surfing or when running applications that bypass intermediaries such as human agents who use applications intensively.

Some JDBC drivers communicate with their corresponding JDBC server gateways over HTTP, however there is no agreement between products on the precise format.

In general, access to Web resources is permitted, unless explicitly forbidden. Web clients can filter documents received by checking the content or by referring to third party sites. Few tools are required to run and manage a simple low volume Website made up of one or more HTTP servers along with HTML documents and other Web resources. However, where a Website has static or dynamic content that changes frequently with high volume requirements, then more sophisticated tools and people are currently required.

2.2 Relational Databases

This section highlights some aspects of relational databases that influence Web integration. Relational database systems provide controlled computer program access to tabular data. Each table has distinct rows whose order is immaterial. There are a number of related set based data manipulation languages which are generally embedded in other programming languages. These languages have public descriptions such as SQL and SQLJ [13]. Metadata describing the structure of tables has to be supplied to the database system before tables can be populated and accessed. Additional tables can be defined in terms of views, e.g., by omitting portions of tables and by combining multiple tables. Database metadata and views are also expressed in terms of programming interfaces. Other than by using a programming interface, there is no consensus for many database features such as:

- how relational data and corresponding metadata are stored within a database system, making navigation via a database tool the only option
- how relational data is exchanged between heterogeneous systems such as dissimilar data management systems and with the Web
- how diverse relational database systems communicate across networks
- how relational database resources are named and how database query URLs are constructed

Security is given much emphasis in relational databases. Access is prevented unless specifically permitted and users who access databases are mostly identified in advance. Any routes through intermediate database systems traversed between clients and database servers are usually predefined. Rules governing concurrent database update attempts are defined and transactions are supported. Access to relational databases is stateful, and clients are expected to connect to a database system, perform a number of database requests using the same connections, and then disconnect. Thus, database systems maintain client context at the server side. Because of the elaborateness of database systems, the number of computer instructions to retrieve an item from a database is usually one or two orders of magnitude greater than that for retrieving a file from a typical Web server (excluding instructions that have to do with communications).

Tools are required to create and manage a database system, even one with few users. The effort of populating and managing a database is greater than that for a Website. There are many more people with the ability and tools to create HTML documents and related Web resources that include navigation elements for other people to follow, than there are people who are themselves able to navigate through existing relational databases.

3 Methods of Integration

In this section, we classify the current methods for Web database integration according to the format of information sent to clients (HTML documents or relational result sets).

3.1 HTML Documents (Server Side Processing)

Relational database information is delivered to clients across HTTP formatted in HTML, from Web database gateway software that acts as the interface between a Web server and a database system. HTML documents can be modified many times before they reach their ultimate human or software destination, e.g., via HTML template gateway software to resolve database references, via server and client side script processors, via server directive processors. By the time, an HTML document that includes relational data reaches a client, any relational table structure information is lost and is difficult to reconstruct.

3.1.1 HTML Template Gateways

A document author creates an extended HTML document, known as a template, that contains database access requests, result set formatting and programming constructs, often using product specific markup tags and programming notation.

Database template software acts as a gateway between an HTTP (Web) server and a relational database system, transforming templates into regular HTML documents before passing them on to a Web server for delivery to a client. Examples of database template tools include Cold Fusion and Net.Data. The most frequently cited benefit of template tools is is the ease with which database result sets are formatted. Features of template processing include:

Database constructs for HTML templates

- *Embedded SQL requests:* See the examples in the Web HTML section above.
- *Embedded result sets:* Templates contain support for defining the style and layout of relational result sets in HTML, without the template author making provision for processing each row individually.
- *Database query URLs:* Database query URLs can be used in hypertext links to generate database result sets. Examples were included in the URL section of this article.

General constructs for HTML templates

- *Programming constructs:* Template tools usually include their own programming constructs which means that *presentation services* code is combined with *business services* code that processes data.
- *Variable substitution:* Most tools make it possible for HTML forms input parameters and database content to be substituted in resulting database query and output documents. Variable substitution notation in use currently includes delimiters in the form of hash, dollar, brackets, ampersands, semicolons (depending on the product) such as $\$(input\_composer)$, #input_composer#, &input_composer; Here is a variable substitution example from an Adabas database template:

$$< sqlexec\, name ='' \, query1"$$

$$query = "select\, title, first\_perf\ from\, opera\, where\, comp\ =\ '\$input\%'" >$$

$$< /sqlexec>$$

- *Embedded HTML fragments:* Fragments of HTML can be incorporated into an HTML template by naming a file through a URL.
- *User Interaction Linkage:* Template tools have conventions for linking query and update documents with result sets documents, making it possible for users to navigate through a series of related documents. Generally each document contains references to the next appropriate document either through ACTION URLs in an HTML FORM, or in hypertext links. Some template tools make it possible to include a series of database queries and results documents in a single file with multiple entry points.
- *Context services:* Template tools include schemes to specify that user sessions be maintained across user requests, such as through HTML extensions for cookies.
- *File Name Extensions:* Template tools have their own usage for file extensions (.mac, .htx etc.) and the usual .htm or .html file extension is used at the time the page is transformed.

3.1.2 Server Side Script and Program Gateways

A programmer creates an application, in a scripting or programming language, that receives input parameters from clients across the network, via Web servers, issues database access requests, and produces an HTML document as output which is delivered to clients via Web servers.

Examples of server side scripting and programming include C, Perl and Rexx CGI programs [14], Java servlets [16], Active Server Pages [17] and LiveWire [18]. Where an interpretive scripting language is used, the difference between HTML templates and server side scripting becomes indistinct, but can be characterized by an emphasis on markup tags in a template environment and on script language constructs in a scripting environment. In general templates are easier to learn and use, but scripts are more flexible.

General HTML templates, in particular variable substitution notation, can be used in conjunction with regular server side programs. Thus, a server application substitutes variables in a template typically when it has completed its processing. Using templates makes it possible for separate individuals to design output documents (templates) while programmers write the code to populate the template.

3.2 Relational Result Sets (Client Side Processing)

A programmer writes a client application, e.g., a Java or ActiveX application, that communicates with a relational database across the network. An intermediate gateway is often used. Relational result sets are delivered to the client application, one row at a time, for further processing and formatting.

With Java, the client portion of an application can be pre-installed on a client system or downloaded from a Web server using HTTP, to run as an applet. An applet is a collection of Java classes that run within the context of another application such as a Web browser. For security reasons browsers prevent applets from performing a number of functions, such as:

- Communicating with software running on a system that is not the same as the one running the Web server from where the applet was downloaded
- Invoking non-Java code (native methods)
- Writing to disk

Some browsers have user options to override these restrictions in Intranet environments, but most browsers enforce the checks. Alternatively, signed applets can be used to overcome restrictions. Applets can access relational databases, subject to the security restrictions, through JDBC or SQLJ. Often, a separate JDBC gateway processes inbound JDBC requests and routes them to the appropriate database system.

4 Some System Integration Issues

This section describes system issues that affect the integration of relational database systems with the Web.

4.1 Thread and Process Management

Web servers provide ways of pooling resources across client requests each with its own performance characteristics. Here are just some of the ways operating system threads and processes are handled:

- *Single process per request:* such as the CGI, where one process is created and dismantled for each client request incurring performance overheads but which nevertheless works well in low volume environments.
- *Multiple long running processes:* such as Fast CGI [15] where a pool of processes is created, and optionally initialized with database connections when the server starts. Each process is used by multiple clients serially. Memory leaks do become apparent in this environment.
- *Single thead per request:* such as the mainframe CICS transaction processing system [19] when it responds to client requests (including HTTP).
- *Multiple long running threads:* such as Web server APIs (GWAPI, ISAPI, NSAPI) where a pool of threads is created and optionally initialized. Each thread is used by multiple clients serially. Side effects include memory overwrites and memory leaks becoming noticeable. (Java servlets can run in single-threaded or multi-threaded processes.)

Web database gateway software often tries to hide the differences between these server environments from template and script authors, and to give the illusion that each template or script is running alone. Database management systems typically manage their own thread and process structures internally, maintaining user contexts for as long as possible while minimizing server resource usage, to reduce request, session initiation and termination overheads. Some Web database gateway software also provides similar long term threading and process management structures. Long lasting resource pooling mechanisms, through Web servers and gateways, can have unexpected effects on database management system internal structures that are tuned for inbound user contexts that endure for minutes rather than days.

4.2 Security

- *Authentication:* There is no single way for performing authentication on the Web. By the time a Web request reaches a database system, the request has usually been authenticated.
- *Access Control:* It is often impossible to predict who all the users of a Web application will be in advance, and the numbers of occasional users may be very large, Usually database access control in Web database applications is performed with general purpose user identifiers rather than specific user identifiers associated with individuals. Most of the time, database gateway connection pools described in the next section require general purpose user identifiers to run efficiently.
- *Encryption:* Communication flows between clients and servers can be encrypted. SSL (Secure Sockets Layer) is the most commonly used encryption method on the Web.
- *Firewalls:* Firewalls are one or more computer systems that monitor and prevent certain kinds of inbound and outbound communication requests, usually between Intranets and the Internet. Few firewalls support database communication protocols because the protocols are not well understood outside database products. Thus, JDBC applets are suitable for Intranets, but they cannot be used easily by people behind firewalls to access servers outside firewalls unless the applets use regular Internet protocols to communicate with JDBC gateways.

4.3 Context or State Management

A Web database gateway has two interfaces, one for the Web client (often the human user) through a Web server, and one for the database system. Maintaining context on behalf of a client is called *session management*, and maintaining context on behalf of a database system is called *connection management*.

- *Session Management:* Users expect session support and do not want to type in the same information repeatedly such as their names. In order to support sessions, gateway software has to be able to identify when a session starts and when it ends, how to transmit a session identifier between a client and server, where to store any session state (client or server), and how to handle unexpected session errors. A number of types of session identifiers are widely used on the Web, such as URL extensions, cookies, hidden fields in HTML forms. Often gateway software provides explicit support for sessions.
- *Connection Management:* Database systems assume that consecutive requests from a single thread or process, within one database connection and with the same user identifier, are issued on behalf of one individual, and hence they bypass certain pieces of processing. Web database gateways often take advantage of this assumption by creating a database connection pool with threads or processes that are connected to the database system for the lifetime of the gateway or Web server. An element in the pool (a thread or process) is usually associated with one user session and cannot be accessed by other users. Where Fast CGI or Web server APIs are used, the Web server resource pool can become the database connection resource pool by automatically connecting the Web server processes or threads to the database system at Web server startup.

4.4 Transactions (Unit of Work)

Transactions with ACID (Atomicity, Consistency, Integrity and Durability) characteristics are a feature of database management systems. Transactions ensure that only consistent data is externally visible, even when multiple users are updating the same data concurrently, when errors occur in the system and when multiple databases at multiple sites are involved. Because of the concurrency control mechanisms for transactions and the resources they consume, and because of the nature of many applications, it is advisable to keep transactions short and in particular not to extend the duration of a transaction beyond one user interaction. The same considerations prevent ACID transactions from being used across organisations and companies. In Web database applications, it is usual for a transaction to be initiated and terminated by the Web database gateway. Typically, one user session is made up of multiple transactions. There have been attempts to introduce transaction identifiers into HTTP, making it possible to extend the notion of a transaction to resources on the Web, but HTTP transaction identifiers are not in general use yet. In Oracle, database query URLs can include transaction initiation and termination information.

4.5 Caching

The primary purpose of caching on the Web is to avoid transmitting Web resources unnecessarily across networks. Caching can occur at many points and they include:

- At a client: A Web browser cache that is typically associated with a single client
- Near a group of clients: A Web proxy cache which is typically associated with and placed near a group of clients accessing many servers
- Near a group of servers: A Web reverse proxy cache which is typically associated with and placed near a group of servers being accessed by many clients
- At a server: This can be similar to database caching where the main reason for caching is to avoid disk accesses. Dynamic content can also be cached to eliminate processing time needed for repeated generation.

LRU (Least Recently Used) algorithms are often used in proxy caches in combination with heuristics such as if a resource was updated recently, then it is more likely to be updated again in a short time interval, or prefetching according to previous access patterns. It is also possible for clients to refresh pages automatically at intervals. Web caching uses information transmitted as part of the HTTP protocol to determine whether to cache the resource and how long to cache it for. However, proxies are also being used to augment, modify or filter the information transmitted between clients and servers transparently. For example, the Webcosm [21] link manager can be configured as a proxy (known as an *intermediary* or a *facade*), to incoporate links into existing documents without changing the documents and taking into account users' interests. Currently, there are few systems that try to provide consistency between relational database content and with the content of Web caches.

5 XML and Relational Databases

XML (Extensible Markup Language [9]) is a modification of SGML [8] intended to make it straightforward to define domain specific markup tags (vocabularies) and to develop programs that process documents containing these tags. Processing programs can understand the specific tags or can be general purpose. HTML is an SGML (not XML) application with its own vocabulary of tags that is useful in a wide range of documents, as well as features such as FORMS that are used in Web applications. Fragments of XML and HTML are likely to appear in a single document. Some important aspects of XML are being able to:

- define and use application-specific markup tags (vocabularies) in XML documents
- create new or use existing document type declarations (DTDs) allowing software to validate the format of corresponding XML documents
- create or use existing stylesheets making it possible to present the same information in different ways
- specify a variety of link types between documents
- combine multiple vocabularies from different sources in a single document through appropriate naming (namespaces)
- refer to DTDs, stylesheets, namespaces etc. in XML documents through URLs

Examples of XML vocabularies are Chemical Markup Language, Information and Content Exchange (for exchanging information between Websites) and Web Interface Definition Language (analogous to IDL). Vocabularies for relational databases include some of the constructs in HTML templates for databases, such as database query and result set tags. The availability of XML formatted result sets makes the database templates approach (server side programming) and result sets approach (client side programming) described earlier more alike as both can manipulate XML fragments. It also makes it easier to exchange relational data with other systems in unplanned ways.

There are application design considerations when database updates arise some time, minutes say, after an XML document was generated from a database. Update decisions can be made based on information which may have already changed in the corresponding database. In the absence of check-in checkout and related constructs in database systems, applications have to detect and recover from the inconsistencies. Some XML formatted result sets persist for long periods, weeks or months say, such as in catalog applications, and the XML formatted information is refreshed or updated intermittently. The generated XML can then be stored in a cache or database for further manipulation and transformation [20].

XML style sheets describe the rules for translating XML tagged information into another format such as into another XML vocabulary or into HTML for display. Initially intended for transformations into human display formats, style sheets or database view mechanisms can form the basis for transformations between XML vocabularies at servers, between servers, as well as at clients. Multiple transformations can take place before data reaches its destination.

Creating and following links between diverse pieces of electronic information, now known as hypertext and hypermedia, was described by Vannevar Bush in 1945 [2]. Links are what make the Web a Web. Relational databases incorporate the notion of links mainly through programs matching key and attribute values within databases of the same type, or through referential integrity. Federated database software supports similar notions across heterogeneous databases.

XML linking makes it possible for applications that produce and process XML tagged information to manage links externally from both source and target resources, to support many-to-many links, to materialise linked to resources in a variety of ways, at different times in accordance with context information, to name and construct multiple navigational views on a set of resources.

In effect indexes created for Internet search engines and for relational databases are collections of externally managed links that provide direct access to resources containing specific terms, with particular names, or with particular attributes. However, relational databases do not typically expose their indexes for general use nor do they have universal naming schemes. Although few database tools automatically generate hypertext links for database result sets, hypertext links can play a part in database applications through linking primary and foreign key values and drilling down into categories of information when data mining.

6 Conclusions

The success of the Web illustrates the advantages of agreed extensible protocols and naming conventions, general purpose software and independence from underlying data structures. Simplifying and adapting well established and widely used standards, together with decentralized collaborative development, are hallmarks of Web protocols and software [5].

Currently, programming effort is required to make relational data accessible on the Internet and on Intranets. Disparate database markup extensions, variable substitution symbols, naming schemes and database query URLs demonstrate that there is no general agreement for Web related database access. The output from database programming interfaces currently requires manipulation before it can be presented to humans and to heterogeneous systems. Reaching consensus on Web based input and output formats (and ways of defining the formats) for established database interfaces is a step towards making relational data more easily accessible from the Web in a generic way.

Increased emphasis is being placed on database systems that manage structured and semi-structured data and which participate on networks as Web resources. With unanimous support for data exchange in XML, there is considerable opportunity for improving the integration of database data with other Web resources in a universal way, including better compatibility between diverse Web integration methods through XML vocabularies for relational databases, and automated conversions to other vocabularies.

The length of time that multiple copies of the same information exist in electronic form outside the control of database management systems, continues to grow. Data is materialised in formats that are not necessarily related to the way it is stored in relational databases. Instead, it is transformed into self defining linked fragments making it more readily accessible to humans or to general purpose software often held in network caches where further transformations occur and where the beginnings of network based collaborative data management and exchange systems are taking place. Irrespective of the facilities incorporated into current database systems, the Web and related software will eventually assimilate their content, thereby providing new data management opportunities.

Apology

In an article on archiving the Internet [4], Brewster Kahle claimed that the average lifetime of a URL is just 44 days making it likely that some URLs listed below will not be available at the time you read this article. I regret any inconvenience caused.

References

[1] **http://www.allaire.com/** : Burke, J., Web Databases with Cold Fusion, McGraw Hill, 1997

[2] **http://www.isg.sfu.ca/%7educhier/misc/vbush** : Bush, V., As We May Think, Atlantic Monthly, pp 101-108, July 1945

[3] **http://www.mkp.com/books_catalog/books.htm** : Chamberlin, D., Using the New DB2, Morgan Kaufmann, 1996

[4] **http://www.sciam.com/0397issue/0397kahle.html** : Kahle, B., Preserving the Internet, Scientific American, March 1997

[5] **http://www7.conf.au** : Khare, R. and Rifkin, A., The origin of (document) species, Proceedings of the 7th International WWW Conference in Australia, April 1998

[6] **http://www.oracle.com/** : Papaj, R. and Burleson, D., Oracle Databases on the Web, Coriolis Books, 1997

[7] **http://www.w3.org/Addressing/URL** : The URL at the W3 Consortium

[8] **http://www.sil.org/sgml/** : The SGML pages at SIL

[9] **http://www.w3.org/XML** : eXtensible Markup Language at the W3 Consortium

[10] **http://lcweb.loc.gov/global/internet/html.html** : HTML at the Library of Congress

[11] **http://www.w3.org/Protocols/Overview.html** : HTTP at the W3 Consortium

[12] **http://java.sun.com/products/jdbc/** : JDBC at Sun

[13] **http://www.oracle.com/st/products/jdbc/sqlj/** : SQLJ at Oracle

[14] **http://hoohoo.ncsa.uiuc.edu/cgi/** : Common Gateway Interface at NCSA

[15] **http://www.fastcgi.com/** : Fast CGI

[16] **http://java.sun.com/products/jdk/1.2/docs/ext/servlet/** : Java servlets at Sun

[17] **http://www.microsoft.com/** : Microsoft Active Server Pages

[18] **http://www.netscape.com/** : Netscape Livewire

[19] **http://www.software.ibm.com/** : CICS, DB2 and Net.Data at IBM

[20] **http://www.darmstadt.gmd.de/oasys/projects/** : The OASYS multimedia projects

[21] **http://www.webcosm.com/** : Webcosm link management