Scalable Integration of Data Collections on the Web

Raghu Ramakrishnan
Avi Silberschatz

Technical Report #1376

June 1998

Scalable Integration of Data Collections on the Web

Raghu Ramakrishnan **Avi Silberschatz**

CS Dept., University of Wisconsin-Madison, and Lucent Bell Labs
email: raghu@cs.wisc.edu, avi@bell-labs.com
The contact author is Raghu Ramakrishnan

Abstract

The Web today is essentially a collection of HTML objects, and the primary mode of querying is through HTML links. However, large *data collections*—not just individual HTML pages or objects—are becoming increasingly accessible over the Web, and traversing HTML links is inadequate as a retrieval and query mechanism. We argue that our model of the Web must change fundamentally to support a *collection of tables* instead of just a *collection of HTML objects* in order to handle large data collections. This model is a natural extension of distributed relational databases, but with quantum shifts in the degree of autonomy and number of nodes. Supporting this model offers numerous challenges for the database research community. As networks become the backbone of information exchange, how we meet these challenges can determine whether database technology continues to be an integral part of information management or becomes "roadkill on the information superhighway."

In this paper, we argue the importance of supporting tables on the Web in order to deal with data collections, and outline the new challenges that this goal raises. In particular, we discuss the following issues and how they are affected by the new environment: (1) Creating and managing integrated data repositories and sharing collections across an organization, (2) Query optimization and execution.

We also propose a novel *Publish-Register-Subscribe* framework as a solution to one of the problems that we identify, namely data integration on the Web: (1) Consumers can publish *container descriptions* for desired data, and search for and *register* into existing containers, (2) Producers can publish descriptions of data sources, and register their sources into published containers. In this approach, the task of creating a complex network of information is gracefully distributed by enabling each producer to contribute to one or more containers without knowledge of other sources of data.

1 Introduction

The World-Wide Web has dramatically increased the number of data sources available to users in a variety of domains, and offers a powerful distribution vehicle for organizations that want to publish or co-ordinate the use of multiple data sources. It is especially attractive in situations where the sources are largely autonomous, evolve dynamically, and therefore do not lend themselves to data warehousing approaches to integration.

The Web today is essentially a collection of HTML documents, and the primary mode of querying is through hypertext-like links. However, large *data collections*—not just individual HTML pages or objects—are becoming increasingly accessible over the Web, and traversing HTML links is inadequate as a retrieval and query mechanism. Typical collections include image repositories,

experimental datasets in numerous specialized formats, and text corpora, as well as structured datasets in DBMSs.

We argue that the model of the Web must change fundamentally from a *collection of HTML objects* in order to provide adequate support for large data collections; while this model has proved to be very successful, it must be enriched to handle data collections, and we argue that the appropriate extension is to support *collection of tables* on the Web (in which some records might contain URLS of HTML objects).

In order to meet the challenges of querying tables over the Web, DBMSs must be brought "out of their boxes" [SZ98] and made accessible over the Web in ways that go well beyond simply providing (form-based or SQL-based) gateways to conventional DBMSs. While others have considered how the link-oriented structure of HTML documents can itself be queried in SQL-like languages [MMM97, LSS96, KS95], or restructured using relational views [AMM97], we discuss the issues involved in handling large numbers of tabular data sources. Of course, the ability to also query the link-oriented network of HTML documents in tabular terms, as these complementary research efforts demonstrate, further underscores the importance of relational querying over the Web. (Incidentally, while we talk about the Web, our discussion is equally relevant to any kind of network, such as corporate intranets.)

This model is a natural extension of distributed databases, but with quantum shifts in the degree of autonomy and number of nodes, and supporting it offers numerous challenges for the database research community. As networks become the backbone of information exchange, how we meet these challenges can determine whether database technology continues to be an integral part of information management or becomes "roadkill on the information superhighway."

In this paper, we argue why the Web must be viewed as a collection of tables in order to deal with data collections, and outline the new challenges that this view raises. In particular, we discuss the following issues and how they are affected by the new environment: (1) Creating integrated data repositories and sharing collections across an organization, (2) Searching for relevant data, (3) Query optimization and execution.

We propose an approach to integrated data access over the Web that relies upon co-operation between *consumers* of data and *producers* of data, and by giving greater responsibility to data producers, enables many evolving data sources to be meaningfully and dynamically configured in response to specific consumer requirements. We refer to such intelligently integrated, co-operatively managed repositories of data as *gestalts.*

We describe the organization of this paper at the end of the next section, after first presenting our view of the Web as a collection of tables.

2 The Web as a Collection of Tables

We now make the case that the Web must be modelled as a collection of tables in order to handle large data collections of arbitrary type. More precisely, the Web is a collection of objects identified by URLS, and we argue that (1) tables should be identifiable through URLs, (2) records can contain URLs, and (3) any collection of objects should be modelled as one or more *metadata* or *summary* tables, with some records containing the URLs of objects that they describe. In varying degrees, each of these three desiderata already appear in practice, and our purpose here is to argue that the proliferation of such data on the Web will necessitate significant extensions of DBMS technology.

To begin with, consider that the Web contains large collections of data objects of various types, including multimedia repositories and text corpora. How can we query such *collections* of data, rather than just retrieve individual objects through an HTML link?

In the field of DBMSs, the relational model has proved to be the most suitable way to think of data collections from the standpoints of ease of querying and data distribution. Object-relational extensions still retain the essential tabular abstraction of data collections. We believe that the only viable approach to querying data collections on the Web is to model each collection as a set of tables, and to view the Web itself as a loosely federated distributed database. Emerging standards such as OLE-DB [B96] already aim to support tabular interfaces to data sources such as spreadsheets.

An objection to this approach might be that it does not encompass collections of non-tabular data, such as multimedia repositories. We claim that in fact such repositories must be queried using associated tabular "metadata". A collection of images, for example, could be modelled as a table of records, where each record describes an image and contains a pointer (e.g., URL) to the image object. The "description" of the image could be a feature vector extracted by image processing algorithms, information such as source and type of image, or a combination: the essential point is that a collection of images is queried by querying the associated table, followed by retrieving specific images corresponding to records in the answer set. (An image collection might well be modelled using more than one table; this is not central to our discussion.)

Indeed, we observe that such tabular metadata is already ubiquitous. Collections of imagery or multimedia objects almost always have associated metadata that is tabular, and individual objects of interest are identified by first analyzing the metadata. This pattern of access does not hold for a major class of objects on the Web today, namely HTML files and objects referenced solely through links in such files; this class of objects is *not* our focus.

2.1 New Web Infrastructure

We now discuss the infrastructure required to adequately support Web sites containing tabular data or any data repository with tabular summaries. The commercial state-of-the-art is that vendors are developing:

- Gateways to DBMS products, and interface standards such as ODBC.
- Middleware products such as Datajoiner that allow relational queries over multiple databases; however, the number of databases involved is assumed to be small.
- Easy to use HTML form-based interfaces for generating queries and for returning answers.

In other words, the focus is on making traditional DBMSs more easily accessible over the Web, which we expect will inevitably lead to rapidly increasing numbers of DBMSs on the Web. We believe that considerable additional research is needed to complement this trend in two ways:

1. Develop a comprehensive framework for organizing and managing the increasing numbers of DBMSs and other structured data sources becoming accessible through the Web. This includes searching for relevant sources, and creating loosely-coupled federations of related sources.

2. Develop the query processing technology to operate in this environment, which poses many novel challenges.

In the rest of this paper, we outline the capabilities that we believe are missing, and that the DB research community should address:

1. Creating semantically meaningful *integrated views* of data from multiple tabular sources is a hard problem that becomes much harder as the number of sources grows; current approaches to relational database integration, where the integrated views are built in a centralized fashion, break down very quickly.

2. Novel profiling issues arise. Both consumers and providers of data must provide profiles that, in addition to describing the data involved (through conventional schemas and statistics), must specify such factors as desired *currency* and *ranking criteria.*

3. Evaluating queries that span a large number of networked sources raises many problems; for example, ranked answers; exploiting hierarchies of sources; greatly extended forms of caching; using approximate reasoning techniques; identifying and using appropriate statistics such as user and source profiles; and spatial reasoning (often, queries over localized regions are required). Most importantly, perhaps, we must give up the assumption that the DBMS has total control over the storage and retrieval of all data.

4. Dealing with changes, both in the sources that become (un)available and in the data that they contain, becomes very important.

5. User interfaces for DBMSs must evolve dramatically to deal with the look and feel of Internet browsers. This means that visual presentations of data and visual exploration will become extremely important, and that references to data outside the DBMS must be handled gracefully through some form of extensibility such as Netscape's "plug-ins".

The problem of creating semantically integrated views is a good starting point for the rest of our discussion for several reasons. First, it is a central problem in its own right. Second, it is implicit in several of the other issues listed above. For example, a query in this environment is in general executed against such a view, and understanding how such views are created offers insight into query characteristics. Similarly, user and data profiles are closely related to the creation of integrated views and query evaluation over such views. We can even think of searching for relevant data sources as a first step towards creating an integrated view that can be queried to obtain the desired information.

The remainder of this paper is organized as follows. In Section 3, we propose a novel approach to creating integrated views over large numbers of autonomous, distributed tables. We call these views *gestalts* to emphasize that they differ from views in a traditional DBMS setting in important ways. We motivate our approach to creating integrated views in Section 5. In Section 6, we discuss the new challenges raised by data collections over the Web within the context of the Gestalt framework.

3 The Gestalt Approach to Integration

A fundamental question that arises is how users can find data sources that might provide information relevant to their needs. The problem is similar to that addressed by search engines such as Alta Vista, but has some additional aspects:

- Keyword search does not utilize information in DBMS catalogs, i.e., the structural and semantic information in schema descriptions and data dictionaries.
- Since the goal is to ask relational queries over collections of tables, "finding" relevant data goes beyond simply locating interesting tables; we must also map the query onto these newly identified tables.

Extending keyword search to incorporate an understanding of DBMS catalog structure, and extending DBMS APIs to provide Web "crawlers" with the appropriate entry points, are necessary but relatively straightforward extensions to existing technology. (Of course, just as with document searches, the task can be made ever more challenging by seeking to exploit domain ontologies and other semantic information aggressively.)

We therefore focus on the second issue, namely how to map a query onto a collection of "related" tables. This is really just the semantic data integration problem in another guise: the user has an abstract table in mind, and issues the query over this table, and we must now map this query onto a collection of concrete tables in one or more databases.

In the traditional approach to database integration, an integrator defines the desired integrated view by examining all the sources to be integrated. This is clearly not feasible for more than a few sources; certainly, it is not feasible at the scale of the number of sources available on the Web. In contrast (and as a complement) to this "pull" style to defining integrated views, we propose a "push" approach in which:

1. The integrator **publishes** a description of desired data, e.g., a relational schema plus a data dictionary for the attribute names involved. This creates a container for the desired data, which we refer to as a **gestalt**.
2. Owners or collectors of data then define local views to compute tuples of the desired form, and **register** these local views into the gestalt.
3. A gestalt can itself be used to define views that are registered into other gestalts, leading to a hierarchy of gestalts.

To register a data source into a gestalt, an owner has to understand his or her data source and the gestalt description, but does *not* have to understand other data sources. This is the step at which semantic integration is done. By distributing the responsibility for this step, our approach allows the creation of gestalts with any number of registered sources! A gestalt created in this manner can be regarded as a view definition that contains the union of a large number of views.

We illustrate gestalts through an example:

1. **Publishing Requests-For-Data:** We regard the definition of a schema for an integrated view as the creation of a *container for desired data*, or a *Request-for-Data (RFD)*. The RFD is *published* by the *consumer* of this information.

   We treat such an integrated view specially in some respects, (e.g., we can associate detailed visualization information with it exploiting DEVise's existing visualization capabilities) and we propose a very different approach to creating such integrated views.

   A container in our framework, or RFD, is similar to a relational view, but differs in some important respects:

- The integrated view is created in a very different way from the usual centralized approach in which one integrator defines the view in its entirety: the integrator just defines the schema in our framework, and the actual view definition is created co-operatively by the (owners of) the underlying data sources.
- Containers form the basis for change notification and management, as described below, and must be capable of receiving and managing notifications of changes at the underlying changes. In this respect, we can think of a container as an *agent*, and a federation of containers as a large-scale *network of information agents.*
- In the context of the Web, it is likely that additional presentation information will be stored with containers for the use of Web browsers and other visualization tools used to render the data in the container.

We therefore call our containers or RFDs **gestalts** in order to distinguish them from conventional RDBMS view definitions.

As an example, we might define a gestalt called FighterPlanes(PlaneId, BaseAirfield, Range, Speed, Level). There may well be collections of fighter planes in several databases, for example, an Air Force database and a Navy database. (Of course, we expect to see far more than two databases!) These collections must now be made accessible through the gestalt.

2. **Subscribing to gestalts:** A consumer of information may want to search for relevant gestalts. If there is already a gestalt called StrikePlanes(PlaneId, Range, Speed, Level, Squadron), the creator of FighterPlanes may want to subscribe to the StrikePlanes gestalt. Even though there are some differences (e.g., the BaseAirfield attribute is missing and the extra attribute Squadron is present), StrikePlanes tuples can be easily transformed into FighterPlanes tuples (adding null values in the BaseAirfield attribute). By registering for the StrikePlanes gestalt, the FighterPlanes gestalt accomplishes two things:

   (a) Queries over FighterPlanes will retrieve tuples from StrikePlanes as well.

   (b) FighterPlanes will be notified of changes to StrikePlanes.

3. **Registration of data sources into gestalts:** In order for a query on the FighterPlanes gestalt to retrieve information from the Air Force and Navy databases, these databases must be registered into the gestalt. First, we must define views, say FP_AF and FP_Navy, over these databases to generate tuples with the same schema as FighterPlanes. There may be no one relation in the Navy database containing all the fields of FighterPlanes; we may have to join two tables, for example, to compute this information. Once these views are registered, FighterPlanes is defined to be the union of FP_AF and FP_Navy.

4. **Queries over gestalts:** At any time, a user can issue a query against a gestalt table, e.g., FighterPlanes, treating it just like a table in a database. Queries are answered by generating queries against each registered data source and combining the results at the integration agent.

5. **Change management:** In the Web environment, new data sources could come on-line, and old sources might be discontinued. In addition, the data in a given source may be updated. How can such changes be dealt with? In our example, changes to the Navy database that affect the view FP_Navy are relevant to the FighterPlanes gestalt, and there must be mechanisms to notify the gestalt about such changes, and for the gestalt to request updated information. Local trigger mechanisms, available in current commercial DBMSs, can be used to detect changes to the Navy database that affect the view FP_Navy. These must be complemented

by messaging and monitoring services that allow the Navy source to notify the FighterPlanes gestalt, and by mechanisms that allow the gestalt to maintain and process a history of such change notifications.

4 Other Approaches to Integration

Integrating heterogeneous databases has received much attention. While the Gestalt framework shares some goals with many of these efforts, it differs significantly from each. In particular, our work is different from related efforts in: (1) Our separation of gestalt creation from the registration of data sources; (2) Our focus on creating/maintaining/querying integrated views over very large numbers of sources. These differences are crucial to success in the Web environment, with large numbers of autonomous, dynamically evolving data sources.

Some of the major related projects are Garlic at IBM Almaden, [LMH97, STR97], Hermes at U. Maryland, [Ada96], Information Manifold at AT&T Research, [LOR96], Kleisli at U. Pennsylvania, [Bun95], Disco [RTV96] at INRIA, Infomaster [GKD97], Lorel and Tsimmis [Pap95, PV97, HPG96] at Stanford, SIMS [AKH96] at USC-ISI, InfoSleuth [Bay97] at MCC, and Rodin/COD at UW Madison. [Alb96].

Like us, these projects follow the mediation philosophy of wrapping data sources and providing mediated access through an integration agent [Wie89, Wie90]. Garlic and Hermes have objectives similar to ours, but there are many differences in the approach taken; specifically, we emphasize integrated queries spanning large numbers of networked sources. Hermes has not emphasised query optimization issues. Garlic has done some of the best work on distributed query optimization over networked heterogeneous sources, but does not consider issues of scaling with number of sources or cooperative processing in a network of autonomous DBMSs. Neither project has dealt with change notification/management.

Information Manifold concentrates on access to networked sources and uses source descriptions to guide query decomposition. Our Gestalt approach refines this by decoupling integrated view definition (i.e., the description of the gestalt, and its publication) from the registration of sources into the integrated view, and thereby enables scalable integration of large numbers of sources. The IM results can be used to support semantic query optimization for relational queries in the gestalt framework; we additionally consider issues of scale, ranked queries, and change notification/management. The SIMS project has also concentrated on semantic query optimization in a traditional distributed environment with few sources.

Infomaster, like Information Manifold, uses source descriptions to identify sources containing data relevant to a query and to translate results into a common form; query planning is done using a model-elimination resolution theorem prover. Of the related work discussed here, it comes closest to Gestalts in that it uses the concept of *base* or *reference* relations to uniformly describe all source relations. In other words, a source can be described in terms of just the base relations, analogous to how a source definition is registered into a Gestalt independent of other sources.

Infosleuth is an agent-based infrastructure for integrated querying. Broker agents maintain information about sources, and are used to decompose queries and to assemble the results. The difference with respect to us is again in our approach to constructing integrated views in a distributed fashion, and in our emphasis on scale with respect to the number of sources, and change management.

W3QS [KS95], WebLog, [LSS96] and WebSQL [MMM97] are all SQL-like query languages for searching the web. For example, WebSQL models the contents of the web in terms of two relations:

Doc(URL, title, type, length, text, modification-time)
Anchor(current, references, label).

The first relation contains one tuple per object on the web, and the second relation contains one tuple per link (in some HTML document *current*, referring to another object). This makes it possible to issue SQL-like queries that fetch documents based on selection criteria that refer to fields like *title* and *length*, and on selection criteria that specify paths of links leading to and from desired documents. The issues addressed by these efforts complement, but do not overlap, the goals of this proposal: our focus is on *tabular* data sources accessible over the web. (Of course, the sources we are concerned with could include tables like Doc and Anchor, containing links to images, videos and other complex objects, and this complicates issues such as cost estimation.)

Kleisli and Lorel focus on integrating semi-structured data and data having complex types (e.g., lists, nested collections), and thus are complementary to our approach. The Disco project also supports registration of remote data sources, and supports a query language based on OQL. The emphasis thus far appears to be on single-site processing, and issues of scaling to a large number of sources have not been addressed.

Tsimmis is studying an integration approach based on warehousing, rather than on-demand query evaluation. Our work in Rodin explored the use of explanations and semantic integration [MIR94, MIR93, JA97, explain] of tabular data.

Finally, the Mariposa project [Sto96] at Berkeley, while not addressing integration, is investigating distributed query optimization in a dynamic environment like the Web, where the set of data sources and their capabilities are changing constantly. They assume that each query deals with a few sources (as in the traditional case), and that each source varies the cost it "charges" a query based on its current load. This framework enables other sites to evolve (e.g., by replicating data that is expensive to access) over time so as to make the overall system more efficient.

5 Motivating Applications

Effective integration and interactive analysis of large, heterogeneous datasets is very important in a variety of environments, for applications ranging from logistics and battlefield awareness to federations of scientific data sources (e.g., NASA's WSIPS federation) to supply-chain management (e.g., Andersen Consulting study [Mad97, PS98]) to on-line directory services (e.g., Yahoo, www.INJersey.com).

In this section, we describe some of these applications and the state-of-the-art in terms of the underlying technology, and outline how our proposed gestalt framework has the potential to change the way such applications are designed, implemented and maintained.

5.1 Overview of Applications

A rapidly growing trend in manufacturing is on-line supply-chain management. Organizations depend upon subdivisions or other organizations for many services, parts, etc., and there are currently fairly rigid and centralized approaches to setting up these "supply chains" and routing

requests and responses through them. A recent study (CDDN) from the U.S. Census Bureau estimates that the inventory in these supply chains (maintained in order to provide timely responses to anticipated future requests) is about 1 *trillion* dollars, with a typical order fulfillment time of 6-9 weeks. It estimates that savings ranging from 10% to 65% can be achieved for a variety of tasks by taking a more flexible approach to supply chain management. A report from Anderson Consulting [Mad97] outlines a new approach to supply chain management to realize the potential savings.

Key aspects to the more flexible approach described in the Andersen report are on-line data services, on-line collaboration between consumers and producers via a "capacity reservation and demand commit model," and Internet-based routing of requests. In essence, technology makes information necessary for decision making instantly available. Whereas earlier operational frameworks for supply-chain management focused on reasonable decision-making with incomplete knowledge and predictions, increasingly the emphasis is on how to use up-to-date, comprehensive information effectively.

Another potential application is Internet commerce. As an example, companies such as WIZnet maintain an on-line supplier catalog library that is accessible over the Internet. Earlier this year, WIZnet reported [Cha97] that its database contained over 61,000 catalogs representing 57,000 suppliers and more than 6,000,000 products. These catalogs are currently organized as HTML pages, and support primarily variants of keyword search. A far better solution would be for WIZnet to maintain one or more Catalog gestalts into which individual suppliers could register information from their catalogs.

A number of federal agencies have also articulated the need for integrating large numbers of data sources. NIMA (National Imagery and Mapping Agency) has an ongoing project called DAGS that aims to connect a large number of sources of geospatial and temporal information, including satellite imagery and associated metadata. Analysts might want to use DAGS, for example, to study events in a certain location over a certain time period, using information from diverse sources. DARPA has several programs whose objective is to integrate large numbers of heterogeneous sources to build and maintain models of battlespaces. NASA has recently begun to explore alternatives to the EOSDIS paradigm of a few centralized data repositories (DAACs), and is actively pursuing the development of a federation of loosely-coupled data sources called WSIPS. Essentially, anyone generating and maintaining a significant data collection would become a member of the federation, as opposed to this data being archived through one of a small number of DAACs. The number of WSIPS could grow very large (easily tens and possibly hundreds). Each WSIP member is responsible for maintaining their data and for registering it into the federation. NSF has launched an initiative on networks of knowledge bases.

Clearly, there are many applications that could be enabled if we can manage and access large numbers of autonomous data sources uniformly. Next, we consider how the gestalt approach can help address these problems.

5.2 Gestalt-Based Solutions

We consider the supply-chain application in more detail to illustrate how gestalts could be used to improve the state of the art. The Andersen report presents the following example scenario for competitive bidding:

1. Manufacturer creats a request for quotes (RFQ), and notifies suppliers over the Internet.

2. Suppliers analyze the RFQ on-line, create a bid in response, and submit the bid (over the Internet) to the manufacturer with price and conditions.

3. Manufacturer analyzes all bids and chooses a winner.

There could be outstanding bids for many items, with many potential suppliers competing. RFQs may well be generated automatically based on current inventories (as per the information in the manufacturer's databases). The sets of bids and responses represent valuable data, and are potentially very large in size; i.e., they need to be managed in a DBMS. In fact, each supplier may have a large number of bids in response to the RFQs of a given manufacturer (e.g., General Motors), stored in the supplier's database.

Given current technology, RFQs would be stored in the manufacturer's DBMS, and suppliers would be allowed to query this database through a convenient form-based interface over the Internet. Individual bids could also be submitted through such an interface; intuitively, each bid results in the insertion of a record in some table of bids maintained by the manufacturer.

The main limitation of this approach is that if a supplier wants to submit bids in response to many RFQs, there is no way to essentially say "Here is my table of bids." Each bid, or record in the bid table, must be submitted individually.

If the manufacturer defines a gestalt table to hold all bids, each supplier can register a local table of bids into this gestalt, solving the problem. Another benefit is that the local views (defining the bids) are evaluated when the manufacturer considers bids. This allows the computation of the bid parameters to be deferred until actually required. For example, a supplier of wheels could make the bid dependent on the price they have to pay (to a supplier further down the chain) for spokes, and this is evaluated as of the time the manufacturer opens the bid.

This example has many natural variations. For instance, Amazon.com, the online bookstore, encourages other sites to act as "re-sellers" by maintaining reviews of a specialized category of books and directing purchase orders to Amazon.com. The company also allows individual reviews for a book to be registered into their central database through a form-based interface. However, there is no way for a re-seller site to include a pointer to the set of all reviews maintained at that site. Again, the problem is easily solved if Amazon.com maintains a gestalt of reviews; each re-seller then registers their tables of reviews into this gestalt, and a user who visits Amazon.com will be able to see all relevant reviews for a book, regardless of whether the review is at the company or at a re-seller.

As another example, the popularity of on-line information directories such as Yahoo continues to grow rapidly. Today, these directories are essentially created and maintained by a central organization. Yahoo (the site) is obviously maintained by Yahoo (the company), but a number of organizations want to create directories tailored to their services or needs. Junglee is a company that creates such directories for a fee; for example, they have created a site for the Washington Post.

The gestalt framework would allow such directories to be populated and maintained in a decentralized fashion. For example, to create a directory of services for a city, an organization might create gestalt tables for restaurants, for movie theaters, for job and real estate advertisements, etc. (Indeed, many examples of such directories can be found on the web, including the Junglee site and sites such as www.INJersey.com, which provides a directory for the Jersey area.)

Now, restaurant owners, theater owners, etc. can be informed about the gestalt, and given the opportunity to register their information into the gestalt. Regional gestalts can then be registered

into gestalts covering larger areas, such as state-wide and country-wide gestalts, possibly several at each level maintained by different organizations. The advantage of this approach is that the gestalt designer can focus on the overall structure, and the task of maintaining consistency of local databases is distributed across many participants.

Note that by refining currently available technology for creating dynamic HTML forms from database contents, the look and feel of a site such as Yahoo can be essentially preserved, even if the site is created and maintained using the decentralized Gestalt approach.

6 Additional Problems to be Investigated

Thus far, we have argued for treating the Web as a collection of tables, and proposed an approach to creating integrated views, called gestalts, over a large collection of such tables. As indicated in Section 2.1, there are many other issues to address. In this section, we discuss some of these additional problems in more detail, using the terminology of the Gestalt framework for concreteness.

6.1 Semantics for Gestalt Networks

The semantics of cycles in gestalt definitions is not clear. For example, Gestalt A at site 1 may be defined in terms of a data source at site 2 that is itself a gestalt, say Gestalt B. What if Gestalt B is defined (possibly indirectly) in terms of A? Since several users contribute to gestalt definitions over time, it is possible for such cycles to develop inadvertently. One approach is to consider such definitions to be recursive, with a least fixpoint semantics. This is unsatisfactory because, first, it seems unlikely that this is actually intended; second, the fixpoint semantics will be very expensive to compute in this highly distributed environment.

We can disallow such cycles at registration time (in which case registration mechanisms are no longer straightforward) or define an execution semantics under which cycles are handled in some reasonable (and efficient) manner. A promising approach is to use node priorities to obtain a unique traversal order over the reachable nodes in a gestalt definition, and to use this order to define a corresponding non-recursive query for any query over a gestalt. We will investigate this further to provide a rigorous foundation for query processing in the Gestalt framework.

6.2 Search and Registration

An owner of a data source must be able to search for gestalts into which the source can potentially be registered (e.g., suppliers would like to find RFQs for which they can bid). Conversely, creators of gestalts must be able to search for sources to register into the gestalts. For example, a manufacturer might want to inform important suppliers about recently posted RFQs. In some applications, the gestalt creator might actually want to register some sources, to populate the gestalt table initially. For example, the creatory of a directory gestalt such as INJersey might want to populate the gestalt with information about some well-known restaurants and theaters.

How can we support such search capabilities? If gestalts and sources are both logically viewed as tables accessible via the web, the search services must understand the associated catalogs and data dictionaries. Standard keyword based search is inadequate, but it is not clear what the right approach should be.

Once relevant gestalts or sources are identified, mechanisms must be provided to register a source into a gestalt. While this is conceptually straightforward, it becomes tricky once issues such as authorizations are considered. (Just consider the interactions between authorizations and view definitions in SQL-92, and consider the impact of distributing location and ownership of view definitions across several DBMSs!)

6.3 Profiling Users and Data Sources

User requirements and the types of data sources cover a very wide spectrum in the Web environment, and query processing and consistency maintenance must be carried out cooperatively by several autonomous DBMSs. The role of statistics and additional summary information will be central to effective data processing. The problems to be addressed include:

- Techniques to maintain summary statistics for widely dispersed data. The challenge is to maintain reasonably accurate information about a large range of external sources as they evolve, with small overhead costs.
- Techniques to describe aspects such as desired levels of currency and response. For example, what are the externally defined gestalts whose changes a user wishes to track? At what level of detail (individual tuples, relation level, aggregate summaries such as counts, etc.) should these changes be tracked?
- Use of shared ontologies in developing data dictionaries. This can greatly facilitate searching for relevant data.
- Protocols to describe the availability of local computational resources for processing external requests. This is essential if cooperative query evaluation by autonomous query engines is to be effective.

6.4 Distributed Query Processing Over Many Sources

Queries can operate on both local and remote data sources. At remote sites, if software is available that can provide query profiling and/or evaluation services, the optimizer must exploit this; otherwise, it must retrieve complete relations and essentially do the rest of the query evaluation at the site where it is executing. When accessing data at remote sites that do provide query profiling and/or evaluation, we must (in the usual cost-based manner) produce plans that "push" selections and projections to remote sites, and consider various join strategies (ship relations between sites, Bloomjoins).

As a simple scenario that illustrates the dramatic potential benefits of doing database-style optimization for queries over the Web, suppose that we want to retrieve 5% of a 500KB file, and that disk bandwidth is 1MB/s and that Internet bandwidth is 1KB/s. The time to ship the entire file from a remote site is 500 secs, whereas the time to ship the result of the selection is 25 secs. We have run such queries using DEVise, and the numbers are entirely as expected. Note that such remote-site query processing benefits both the DEVise user (who wants answers fast) and the owner of the site (who sees much smaller files being shipped, in exchange for a modest CPU cost). Thus, there is motivation for sites to download and utilize our query handling software for remote sites.

The above points illustrate the need to incorporate standard distributed query processing strategies into the web environment. This environment and the gestalt framework together introduce some novel challenges as well:

- *Queries with ranked results:* The importance of queries whose answers are *ranked*, or sorted, cannot be underestimated in this setting. (When was the last time you looked at more than the first couple of pages of results produced by a search engine such as Altavista?) For example, a manufacturer probably wants to find the lowest bidders first, and may be satisfied after finding the first few answers. Some results on evaluating such queries are presented in [HHW97, CK97].

  Ranking also arises because of queries over multimedia sources, which are common on the web. (Although we deal with a framework of tabular sources, we fully expect many such sources to be really metadata for a collection of documents or images; a query such as "Find similar images" over a tabular source containing image descriptions (as feature vectors of some kind) will return ranked results.) Fagin gives an interesting analysis of such ranked queries in [Fag96], which provides a good starting point for a detailed investigation of this problem.

  The notion of ranking might also be tied to factors such as reliability (e.g., if a node provides delayed stock quotes, we might prefer to access a source that provides real-time quotes, although the gestalt definition is a union over both). Yet another notion of ranking is tied to expected cost: we want to evaluate subqueries in an order that minimizes the time taken to produce the initial set of results, since additional results can often be computed while the initial set is being evaluated by the user.

- *Scalability with number of sources in a query:* How should we schedule each query in a large union of queries? As the above discussion of ranking reveals, there are many possibilities. If several queries are concurrently issued, partial results obtained from one could be used to restrict computation in others. For example, if we want the lowest cost bidder, and we obtain a $100 bid for item A from vendor V, we are no longer interested in bids for A that cost more; this information can be "pushed down" into local queries evaluating bids from other vendors.

- *Spatial Distribution of Data:* Given the highly distributed nature of data, and the large number of network nodes that are capable of query processing, we believe that novel mechanisms involving query shipping based on data proximity will become increasingly important. If queries can be described with references to networked data, a DBMS may well process a query by handing it off to another (co-operating) DBMS. For example, consider a query issued at Wisconsin over data in New Jersey. If there is an idle site in New York with a high bandwidth connection to New Jersey, we may request this site (which is autonomous and has no direct ties to the Wisconsin and New Jersey sites, beyond being a Gestalt engine) to evaluate the query.

  If a query requires all maps of the Chicago region, and map servers are also distributed geographically (as is the case with many NIMA databases, for example), there is a close relationship between deciding where to execute the query and the spatial index used to identify the relevant subset of data. As another example, the information about data sources at a container node might indicate that a particular data source contains relatively few matches for a particular query (e.g., relatively few days with temperature below 32F in Seattle, but many such days in Madison).

The combination of ranking and spatial queries creates additional challenges. For example, if we are interested in information about Madison, WI in December 1997, information about nearby locations and time periods is also likely to be of interest, but information about neighboring locations becomes less interesting as we move further away. The known techniques for handling ranked queries do not cover this class of queries.

Distributed query optimization and the use of spatial indexes in this context present interesting challenges.

- *Systems issues:* Many new systems issues arise as we investigate query evaluation in this new environment. For example, if remote sites ship data faster than the query computation can consume it, the buffer manager must provide the infrastructure to "buffer" (in memory or disk) varying amounts of an incoming stream, and to consume multiple input streams (generated by different subqueries) asynchronously. [DFJST96, FZ97, AFB97]

6.5 Wrappers for Tabular Data Sources

Middleware tools such as DataJoiner allow users to issue relational queries against data stored in several DBMSes at different sites. We seek to provide similar functionality with queries over gestalt definitions, but with an emphasis on queries spanning many DBMSs. Another differing point of emphasis is that we consider it important to provide access to tabular datasets on the Web in operating system files, not just data in traditional DBMSs. There are already many such datasets, and the number is growing. Further, some sites have realized that they need to provide a query capability to avoid being swamped by unnecessarily large file retrievals, e.g., the NCDC weather information repository.

It is in this context that we now briefly discuss software that "wraps" a tabular data source, i.e., provides external access through a uniform interface. This is an area where there is considerable ongoing activity (see, e.g., [STR97]), and we make just two comments: (1) It is important to provide an interface to ask about estimated costs for a query, user/data profiles, etc., in order to enable intelligent query optimization. Current standards such as OLE-DB do not go far enough in this direction. (2) Wrappers must be non-intrusive; for example, "wrapping" data by loading it into a DBMS is not always acceptable, for a variety of reasons ranging from cost to the need to execute other applications on the data. Nonetheless, for tabular data, the ability of the wrapper code to do some query processing at the remote site can be very valuable.

With this motivation, we propose software (essentially, a simple relational query engine) that sites can download and install to provide such query capability. The owner of the data will "register" all local tabular datasets with this software, indicate which tables/fields are to be indexed, etc., and the necessary index and catalog information will be constructed "on the side", without in any way affecting the original data; data records are identified using "handles" supported by the local data representation (e.g., file offsets). Thus, existing applications can run unchanged, and the data is not imported into a DBMS. However, there is now an alternative way to get at the data, through our software's query interface.

Such a **remote wrapper** package has been implemented in the DEVise system [LR+97], and runs as a CGI at a remote site, The interface is as follows: It accepts an ASCII query string and parameters that indicate whether the query is to be executed or profiled. If the query is profiled, a string containing estimated cost and result size is sent back; if it is executed, the answers are sent back as an ASCII stream. The OLE-DB *rowset* interface can be used to implement versions of our

software, but it is a strictly lower level interface (since it does not cover query strings, profiling vs. execution, etc.).

6.6 Dynamic Data Environments

The Web is dynamic by nature. New data sources could come on-line, and old sources might be discontinued. In addition, the data in a given source may be updated. How can such changes be dealt with? In general, we believe that given the heterogeneous, loosely-coupled nature of gestalt federations, an asynchronous message passing approach to change notification is appropriate. Once notified of a change, a gestalt must deal with it somehow (by re-evaluating the gestalt definition if it is materialized, for example). There are two basic aspects to the problem:

1. Gestalts that draw information from a source must describe the changes that they are interested in.

2. A source must be able to notify all gestalts that it is registered into of relevant changes.

In an environment where many independently evolving nodes collaborate, in dealing with changes at one node we cannot, in general, assume that another node affected by these changes will be prepared to immediately respond. Rather, we provide a mechanism for nodes to register profiles at other nodes; when there is a change at node N that is relevant to node M (as determined by M's profile at N), N will notify M. M can deal with this change notification using a policy that is local to M, and which N need not worry about.

While the appropriate general approach to change notification and change management may be asynchronous replication based on message-passing, we anticipate that there will be a need for a synchronous notification capability. For example, consider an example where a user wishes to be notified of changes in a stock's price. In general, such changes might be communicated through asynchronous messages; however, once a significant change is detected, further rapid change might be anticipated and the user might want to monitor this data source continuously.

Results in warehouse view maintenance and distributed trigger implementations are available in the literature and are clearly relevant. However, the degree of autonomy across sites and the number of sources envisaged require us to re-evaluate existing solutions, and develop new solutions if needed.

In order to maintain a consistent view of data from multiple sources, gestalts must provide another important feature: the ability to receive and maintain a history of change notifications from registered sources, and to reason about the implications of these changes. For this, the query capabilities must be extended to cover sequences (since histories of changes are time sequences), in addition to relations. (More ambitiously, if a gestalt maintains a history of actual changes at a particular source, this involves reasoning about how the data evolved at time.)

Another motivation for considering sequences is active sources whose contents are frequently updated. A gestalt that monitors such a source might want the source to simply send it a stream of updates, and these are naturally modeled as sequences. This approach, in contrast to the asynchronous message passing approach, places a greater demand on the gestalt site's resources. If the number of monitored sources is large, we cannot afford to allocate each source space in the buffer pool, and must develop techniques to "poll" active sources, rather than have the sources "push" change information to the gestalt.

In general, both change management and query evaluation need to balance *push* versus *pull* modes of execution. While there is some initial work in this area [AFB97, FZ97], this is clearly an important area for further research. For example, profiles can be used to estimate how quickly answers to a query must be evaluated, and parts of the answers (perhaps those that are needed first for a ranked query) can be cached. When evaluating a query, intelligent caching can be used to utilize existing results in the cache and to modify the query to compute only the rest of the answers. Again, work in this area is preliminary; for example, [DFJST96] shows the benefits of intelligent buffering through a simulation study, but does not address the issues in designing and implementing an intelligent cache.

6.7 User Interfaces

The typical database user can no longer be assumed to be an SQL expert, or indeed to have the services of an SQL expert available to him or her. When a query is issued, often the desired presentation of the result is some graphical form. For example, a user may want to see piecharts, scatterplots, or more sophisticated and customized representations; indeed, the description of how the data is to be presented should be considered as part of the query specification.

Specifying how data is to be presented must be intuitive, and presentations must support interactive exploration (e.g., drill down to specific records from a scatterplot). If references to external objects are included in the retrieved answer records, it must be possible to use the appropriate viewers (e.g., a Netscape browser to view an HTML object identified by a URL) seamlessly. Conversely, presentations of tabular information must be accessible through standard Internet browsers.

These challenges argue for a new generation of presentation and retrieval tools that combine features of relational query interfaces and Internet browsers [LR+97], and developing such tools is a major challenge.

6.8 Agent Based Architecture

To realize the Gestalt framework, an agent based architecture seems necessary in order to decentralize the decision making. We briefly outline such an architecture:

- **Consumer Agents:** These agents are associated with clients or users. Their role is to enable a client to describe desired information, and to then intelligently locate and subscribe to sources of such information.
- **Provider Agents:** These agents are associated with data providers. Their role is to advertise available data, to control access to this data, and to register the data into gestalts that require this data.
- **Search Agents:** These agents are used by both Consumer and Provider Agents, and directly by users, to search for relevant gestalts and data sources. They are similar to Web search engines, but are designed to search gestalt and data source catalogs that are accessible through the Web.
- **Query Agents:** These agents are associated with nodes (gestalts or sources), and process queries at that node. In doing so, they must interact with other query agents at connected

nodes, and use intelligent control strategies to (1) avoid swamping the network with overlapping requests, and (2) to restrict the computation by sharing information obtained across subcomputations.

- **Change Agents:** These agents are associated with nodes, and are responsible for change notification and management: (1) Changes at the node should be propagated to nodes that consume its data. (2) If the node depends on data from other nodes, change notifications from these other nodes must be processed.

Consumer and provider profiles must guide the activities of these agents, and the design of these profiles is an important research objective. The particular division of responsibilities implicit in the above list of agents is not as important as the basic approach taken: in this agent based architecture, responsibility for maintaining data currency and querying has been completely decentralized, and agents are expected to co-operatively address their own tasks as well as tasks executed on behalf of other nodes in the network. This has several implications: It places limits on the forms of data consistency and currency that are feasible for such widely shared data collections. It raises a host of functionalities for which agent interfaces must be defined and for which corresponding interfaces do not exist in a conventional DBMS. It makes explicit the fact that no agent can assume that it has control over the activities of other agents, and raises issues of local resource management in the face of external requests and limited co-operation.

Clearly, what we have outlined here is less an architecture than a skeleton that highlights important unresolved issues.

7 Conclusions

We have argued that the Web should be modelled as a collection of tabular data sources in order to provide adequate support for querying large data collections of any type, and described a number of issues that cannot be handled using current techniques.

The Web is becoming the dominant means of accessing data. DBMSs cannot remain specialized data management "boxes" that are attached to the Web through simple gateways. We must take DBMS technology "out of the box" and enhance it and apply it directly to change the way data is organized, maintained and queried on the Web; we cannot afford to let the current paradigm of hyperlinked HTML objects continue to be the mode of retrieval.

In this paper, we make a contribution towards realizing this new generation of highly distributed data sources by developing a scalable paradigm for creating integrated views over many sources. Since querying in a relational model fundamentally involves view definition as the means of identifying relevant data, this is a necessary first step. However, a number of other issues remain to be resolved, and we have tried to make the case that this represents a fundamental new direction for the database community to explore.

8 Acknowledgements

We thank Miron Livny for many stimulating discussions on the role of tabular metadata.

9 Bibliography

References

[AFB97] Swarup Acharya, Michael J. Franklin, Stanley B. Zdonik. Balancing Push and Pull for Data Broadcast. in Proc. SIGMOD, pp. 183-194, 1997.

[Ada96] S. Adali et al. Query caching and optimization in distributed mediator systems. In *Proc. SIGMOD*, 1996.

[Alb96] J. Albert. Schema and Data Integration in Heterogeneous Multidatabase Systems. In *Ph.D. Thesis, University of Wisconsin-Madison*, 1996.

[AKH96] Y. Arens, C.A. Knoblock and C-N. Hsu. *Query Processing in the SIMS Information Mediator.* Advanced Planning Technology, editors, Austin Tate, AAAI Press, Menlo Park, CA, 1996.

[explain] T. Arora, R. Ramakrishnan, W. Roth, P. Seshadri, and D. Srivastava. Explain: Generating explanations for deductive systems. In *Proc. DOOD*, 1993.

[AMM97] P. Atzeni, G. Mecca, P. Merialdo. To Weave the Web. In *Proc. VLDB*, 1997.

[Bay97] R.J. Bayardo Jr. et. al. InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. Proc. ACM SIGMOD, 1997.

[B96] J.A. Blakeley. OLE DB: A Component DBMS Architecture. Proc. ICDE, pp. 203–204, 1996.

[Bun95] P. Buneman et al. The Kleisli Project. *http://sdmc.iss.nus.sg/kleisli/kleisli/kleisli.html.*

[CK97] Michael J. Carey, Donald Kossmann. On Saying "Enough Already!" in SQL. *Proc. SIGMOD, pp. 219–230*, 1997.

[Cha97] R.E. Chalmers. Speed Product Design with Supplier Management Software. In *Integrated Design and Manufacturing*, pages 26–30, October 1997.

[DFJST96] S. Dar, M.J. Franklin, B.T. Jonsson, D. Srivastava, M. Tan. Semantic Data Caching and Replacement. In *Proc. VLDB*, 1996.

[Fag96] R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1996.

[FZ97] Michael J. Franklin, Stanley B. Zdonik. A Framework for Scalable Dissemination-Based Systems. in Proc. OOPSLA, pp. 94-105, 1997.

[GKD97] M.R. Genesereth, A.M Keller and O. Duschka. Infomaster: An Information Integration System. *Proc. SIGMOD*, 1997.

[GRSY97] J. Goldstein, R. Ramakrishnan, U. Shaft, and J. Yu. Processing Queries by Linear Constraints. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1997.

[GRS97] J. Goldstein, R. Ramakrishnan, and U. Shaft. Compressing Relations and Indexes. In *Proc. ICDE*, 1997.

[HPG96] Laura M. Haas, Yannis Papakonstantinou, and Ashish Gupta. Capabilities-based Query Rewriting in Mediator Systems. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, 1996.

[HHW97] J.M. Hellerstein, P.J. Haas, H. Wang. Online Aggregation. *Proc. SIGMOD, pp. 171–182*, 1997.

[KS95] D. Konopnicki and O. Shmueli. A Query System for the World Wide Web. In *Proc. VLDB, pp. 54–65*, 1995.

[LSS96] V.S. Lakshmanan, F. Sadri, and I.N. Subramanian. A Declarative Language for Querying and Restructuring the Web. In *Intl. Workshop on Research Issues in Data Engineering*, 1996.

[LOR96] Alon Levy, Joanne Ordille, and Anand Rajaraman. Querying Heterogeneous Information Sources Over the Web. In *Proc. VLDB*, 1996.

[LR+97] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki and K. Wenger. DEVise: Integrated Querying and Visual Exploration of Large Datasets. *Proc. SIGMOD*, 1997.

[Mad97] S. Maddila. Supply Chain Opportunities. *Report, Andersen Consulting*, 1997.

[MMM97] A. Mendelzon, G. Mihaila, and T. Milo. Querying the World Wide Web. In *Intl. Journal on Digital Libraries 1:1*, pages 54–67, 1997.

[MIR93] R. Miller, Y. Ioannidis, and R. Ramakrishnan. The role of information capacity in schema integration. August 1993.

[MIR94] R. Miller, Y. Ioannidis, and R. Ramakrishnan. Translation and Integration of Heterogeneous Schemas: Bridging the Gap Between Theory and Practice. *Information Systems*, 19(1):3–31, January 1994.

[Pap95] Y. Papakonstantinou et al. Object Exchange Across Heterogeneous Information Sources. In *Proceedings of the IEEE Conf. on Data Engineering*, 1995.

[PV97] Yannis Papakonstantinou and Vasilis Vassalos. Describing and Using Query Capabilities of Heterogeneous Sources. In *Proceedings of the International Conference on Very Large Data Bases*, pages 256–265, 1997.

[PS98] PeopleSoft White Paper. *http://www.peoplesoft.com/products_and_services/scc/index.html.*

[JA97] R. Ramakrishnan, J. Albert, Y. Ioannidis. Conjunctive query equivalence of keyed relational schemas. In *Proc. ACM PODS*, 1997.

[RTV96] L. Raschid, A. Tomasic and P. Valduriez. Scaling Heterogeneous Databases and the Design of Disco. In *Proceedings of Intl. Conf. on Distributed Computer Systems*, 1996.

[STR97] Peter M. Schwarz, Mary Tork Roth. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. VLDB, pp. 266–275*, 1997.

[SZ98] Abraham Silberschatz, Stanley B. Zdonik. Database Systems—Breaking Out of the Box. in *SIGMOD Record 26(3): 36-50*, 1997.

[SLR94] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. Sequence query processing. In *Proc. ACM SIGMOD*, pages 430–441, Minneapolis, MN, May 1994.

[Sto96] M. Stonebraker, P.M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Straelin, and A. Yu. *Mariposa: A Wide-Area Distributed Database System.* VLDB Journal, 5(1): 48-63, 1996

[Wie89] Gio Wiederhold. The Architecture of Future Information Systems. In *Proc. of the Intern. Symp.on Database Systems for Advanced Applications, KISS and IPSJ, Seoul Korea, pp.7,* April 1989.

[Wie90] G. Wiederhold. Future architectures for information processin g systems. In *Rishe, Navathe, and Tal (eds) : Proc. Parbase-90, Int. Conf. on Databases, Parallel Architectures, and their Implementation, IEEE Computer Society Press, March , pp. 160-176.*, 1990.

[LMH97] Edward L. Wimmers, Jun Yang, Laura M. Haas, Donald Kossmann. Optimizing queries across diverse data sources. In *Proc. VLDB, pp. 276–285*, 1997.