Content Integration for E-Business

Michael Stonebraker Joseph M. Hellerstein

Cohera Corporation
3953 Point Eden Way
Hayward, CA 94545
http://www.cohera.com

{mike,jmh}@cohera.com

ABSTRACT

We define the problem of **content integration** for E-Business, and show how it differs in fundamental ways from traditional issues surrounding data integration, application integration, data warehousing and OLTP. Content integration includes catalog integration as a special case, but encompasses a broader set of applications and challenges. We explore the characteristics of content integration and required services for any solution. In addition, we explore architectural alternatives and discuss the use of XML in this arena.

1. INTRODUCTION

It is often said that "the web changes everything", and this is particularly true for business computing. But while the web has changed E-Business, E-Business has changed the web as well, by tying it into the existing and developing fabric of business computing systems. In this paper we discuss **content integration**, a set of challenges that arise from the needs of a new, third generation of corporate web usage. To put this generation into context, we consider the previous generations as well:

- **Generation 1: Web Marketing.** As is well known, the web was invented for the exchange of scientific documents. Businesses first used the web in a similar fashion – as a marketing channel for pamphlets, advertisements and other loosely-formatted, human-centric content, published via HTML and HTTP. This generation of web use led to a renaissance in information retrieval that spawned a number of successful companies. Generation 1 presented few new information integration challenges.

- **Generation 2: E-Commerce.** The second generation of web use brought consumer purchasing online, and required the integration of back-end Transaction Processing (OLTP) systems with web front ends. Generation 2 required back office systems be integrated with new front ends, and spawned a thriving enterprise application integration (EAI) business.

- **Generation 3: E-Business Operations.** The third generation is just starting, and entails business-to-business operations. Here, corporate buyers and sellers wish to transact business more efficiently in net marketplaces, private exchanges and procurement sites. The key to this generation is to integrate the content-rich operational software involved in these processes.

1.1 Sea Changes in E-Business

Generation 3 brings three fundamental differences from earlier generation web usage, and we now turn to these sea changes.

- **Integration occurs between enterprises.**
Historically, information integration was a problem that occurred within a single enterprise, and it was solved using data warehousing techniques. As we will see, this model breaks down when the information to be integrated crosses enterprise boundaries.

- **Operational data must be integrated.**
E-Business in Generation 3 primarily deals with operational data, rather than historical (trend) data. The concern in this generation centers on pricing information, orders, fulfillment issues and delivery of services over the web. The shift of the major information integration problem from historical to operational data has profound consequences on the architecture of information integration systems.

- **Structured and unstructured data must be easily integrated.**
  Traditional information integration focused on structured data, and data warehousing systems exclusively dealt with record-oriented data that was carefully (and expensively) mapped by experts. However, the current web requirements entail structured data (e.g. parts catalogs) and unstructured information, (e.g. news reports and product testimonials). Mixed-mode ("semi-structured") data may be expected as well (e.g. contracts and resumes). Queries need to span from structured to unstructured data, and integration has to be handled by non-technical content managers with domain expertise – not DBAs with modeling expertise.

1.2 Vignettes from a Third Generation

To motivate content integration and explore these sea changes in more detail, we now present three vignettes of Generation 3 E-Business information integration.

Integration for Republishing: Consider a large distributor of so-called "MRO" goods – the basic Maintenance, Repair and Operations items purchased by almost any business, including everything from lightbulbs to pencils to forklifts. This MRO distributor maintains an online catalog of the products offered for sale to its customers. Given the breadth of products covered by MRO, a large MRO distributor typically has thousands of suppliers. Hence the distributor must integrate the individual catalogs from each of its suppliers. This entails integrating content-rich operational information from different enterprises. Sometimes this problem is referred to as **catalog management** [9].

Integration of Availability and Pricing: A second example concerns a traveler who must fly to Atlanta tomorrow. Besides an airplane ticket, he requires a place to stay. His request is for a room within ten miles of the airport with a health club at a corporate rate less than $200 per night. Hotel room availability in the Atlanta area is in some fifty data systems (each hotel chain runs their own reservation system), and solving the traveler's problem requires integration of operational content from this myriad of data sources in different enterprises.

Integration for Supply-Chain Management: Our final example concerns a manufacturer of goods. This manufacturer buys subassemblies from its suppliers and produces a finished product. Suppose demand for the product just increased and the manufacturer would like to increase his production rate. Whether this is feasible or not depends on information inside his enterprise, but more importantly on information from each of his suppliers. Whether they can increase production, of course, depends on their individual situation and on the state of each of their suppliers. Hence, efficient product scheduling requires the entire supply chain to share information. Again we see cross-enterprise integration of operational information. Furthermore, there may be various contract documents among the participants in the supply chain, which may deal with changes in production schedules. Such unstructured information must be integrated as well as possible with structured data concerning fulfillment, in order to determine a fair and legal price for a change in production.

These three examples show three different information integration problems that arise in the conduct of E-Business. The first two are both catalog integration problems, with the first being over goods and the second over services. All share the common characteristics that they entail integration of operational information across enterprises.

In this paper we define the content integration problem, and discuss in more detail the characteristics of content integration, which result from the above sea changes. In the process we comment on architectural alternatives and discuss open research problems.

2. NEW CHALLENGES, NEW TERMS

We use the term **content integration** to refer to the integration of operational information across enterprises. Content integration deals with the cross-enterprise integration of items such as product catalogs, product descriptions, product availability, product fulfillment and related information for custom and MRO products for E-Business purposes. We use a new term to reflect a number of new challenges raised in this setting. It is instructive to contrast these challenges with the problems addressed by earlier integration efforts.

Enterprise Application Integration (EAI) is a term used to describe the task of programming across multiple enterprise applications. EAI provides workflow scripting languages and reliable messaging, along with gateways to common enterprise applications (e.g. ERP) and systems (application servers, web servers, etc.) EAI is well suited for routing individual code-intensive tasks via relatively simple messages ("buy this item, charge that account"). By contrast, content integration must not only route large volumes of rich content, but also cleanse, normalize and integrate it into a semantically useful cross-enterprise model. EAI is a programming-centric model, and its imperative style makes it ineffective for the challenges of content integration. The complexity and rapid evolution of data and systems require high-level data modeling, declarative languages and intelligent content integration systems.

The phrase *data integration* was originally used to describe the problem of bringing together historical, intra-enterprise data for business intelligence purposes. Because content integration entails cross-enterprise operational data, it is very different from this more traditional data integration problem. Data warehousing was the technology of choice for historical data integration. We will see that warehousing is fundamentally flawed as a technology for content integration. As a result, it is crucial to use a different term to describe the content integration problem.

Content management is a term that describes the issues around creating, tracking, versioning, storing and disseminating semi-structured and unstructured information owned by an enterprise. Every enterprise involved in even first-generation E-Business has a content management problem. In contrast, content integration deals with the integration issues that arise from content and data having to be shared across enterprises.

3. CHARACTERISTICS

In this section we indicate the main characteristics that describe content integration and content integration solutions in the E-Business space. We break these characteristics into two broad challenges: content mapping and query processing.

3.1 Mapping Content: Access and Transformation

Any federated architecture needs to get data from multiple sources, and provide it to multiple recipients. This presents serious challenges in a heterogeneous, decentralized environment.

Characteristic 1: Content to be integrated comes from owners with varying relationships with the integrator of the content.

Some content owners will allow an integrator to directly access their internal systems, often SAP or another ERP system. This typically requires the content owner to allow integrator-specific code to run on his site. Obviously, this sort of direct relationship can be supported only for a small number of important integrators. Others will be required to have an arms-length relationship, and get information by accessing the web site of the content owner just like anybody else.

Hence, a good content integration solution must support a variety of relationships between the content integrator and the content owners, ranging from scraping web sites to directly accessing internal systems.

There is a cottage industry of companies that write wrappers (or gateways) around ERP and other legacy systems (e.g. Merant, NEON, Attunity, etc.) To complement this technology, web screen scrapers are also needed to wrap websites. Web screen-scraping entails specifying both how to access some data (e.g. by issuing an HTTP GET or POST), and how to parse it when it is returned. Most of the research work in this area has been on automatically developing wrappers for pages ("wrapper induction" [10], "information extraction" [5], etc.) Commercial screen-scraping is not merely intelligent parsing, however – it also includes the intricacies of navigating JavaScript pages, dealing with cookies and passwords, and interfacing with HTTPS-protected sites.

The web screen-scraping work in the literature can help with parsing, but only in the context of easy-to-use tools. We remind the research community that the people writing wrappers are typically non-technical "content managers". Hence what is really needed is an integration of semi-automatic wrapping (since no automatic scheme we have seen is close to foolproof) with simple fix-by-example graphical interfaces. The research community is encouraged to continue working on minimizing the cost per wrapper – which entails intelligence not only in the system, but also in the user interface.

XML certainly ameliorates the problem of writing wrappers, in that it encourages the use of meaningful tags for structuring text. Languages like XSLT also help simplify the parsing and transformation into a standard format. As the commercial world migrates from HTML to XML, it will become easier to develop wrappers. However, we do not expect the need for scraping unstructured data to disappear soon.

Characteristic 2: Content to be integrated comes in different formats.

Every enterprise constructs its content using its own format and semantics. For example, a US supplier quotes product prices in dollars, while a French supplier quotes prices in francs. Moreover, companies often mean very different things by "two day delivery". For some companies it is two calendar days; for others, it is two business days; for yet others (like FedEx) it is two calendar days with Sunday excluded. As such, content integration requires the homogenization (or normalization) of disparate syntactic and semantic data elements, i.e. content integration must deal with semantic heterogeneity.

A commercial content integration system must have the ability to transform objects to different formats, with custom mapping rules to capture semantics as well. To specify transformations, one would want a mechanism where simple transformations could be specified using a simple drag-and-drop GUI, while more complex ones could use a scripting language. Data-driven mappings (via synonym tables, stored semantic taxonomies, and so on) are also important, and form another step in data integration. Ultimately, one must be able to construct general transformations in a conventional programming language. Furthermore, some transformations require a multi-step workflow. A transformation infrastructure that supports all these options is important.

Ease of use is absolutely critical here. A large supply chain entails thousands of suppliers. For example, Home Depot is reputed to have 60,000 suppliers. Specifying 60,000 transformations is a daunting task, and some very high-level mechanism is clearly required. An interesting research area includes approaches to make it easy for content managers (again, people with limited technical skills) to write large numbers of transformations. Also, standards activity, perhaps a generalization of UDDI [14], is another promising direction of exploration.

Characteristic 3: Content requires multiple schemas and multiple taxonomies.

Content integration arises in a variety of E-Business situations, ranging from airline seats to after-market truck parts to steel beams. Obviously, the information that must be recorded about airline seats is different than that recorded about steel beams. As a result, any content integration solution deployed in multiple vertical markets must support a multitude of schemas. This is a failing of many of the early catalog integration systems, which provide a rigid (and often obscenely complex) schema.

Many E-Business schemas require part_name as one of the fields, in addition to a part identifier such as a universal product code (UPC). Unfortunately, in most vertical markets, there is no standard representation for part names. Hence, some vendors refer to black ink as "India ink" while others call it "fountain pen ink, black". Obviously, content integration solutions need to support searching on product names; for example, a user should be able to query an integrated catalog for all vendors that supply black ink. To support such search, it is necessary for the content integrator to map supplier product names into a common semantic representation. In many circles this is called a taxonomy; in some research communities it is more grandly called an "ontology".

Clearly, there are as many taxonomies as there are vertical markets. There are also generalized "horizontal" taxonomies, like the Universal Standard Products and Services Classification (UN/SPSC) codes developed by the United Nations Development Program and Dun & Bradstreet. Moreover, in some markets there are multiple taxonomies for part names as well as for other fields in the schema.

Taxonomies are usually arranged in a semantic hierarchy. For example, in the UN/SPSC, "India ink" is a special kind of "Ink and lead refills", which are in turn a special kind of "Office supplies". In this way, a user who requests information about refills can be given product entries for both ink and lead. As a result, a query to a hierarchical taxonomy of part names should return all parts at the matching levels as well as those below them in the hierarchy. Taxonomies should be browseable and searchable in the same manner as the data itself.

With taxonomies, the data integration task expands to include the integration of the taxonomies. When a new taxonomy is to be added to an integrated model, matches need to be found, conflicts identified, and ambiguities resolved. In most systems today this is a laborious manual task. Semi-automatic schemes that combine system suggestions with user editing are absolutely critical here. This is an interesting research area where elegant ideas are in short supply.

Characteristic 4: Content must be custom syndicated.

Historically, a content owner published the same content to everybody. However, the web allows much more sophisticated "personalization" of content to be syndicated (published) to multiple recipients. We refer to this process as **custom syndication**.

Many sellers have pricing schemes that are buyer-dependent. As a familiar example, airlines typically have discounts for their preferred frequent flyers. In some cases seats are "made available" to top-tier customers even when there are no seats left (by "bumping" less-preferred customers). Hence both pricing and availability can be functionally specified by business rules. Even more complex are package prices for bundles of purchases (e.g. vacations) that may span multiple suppliers and their individual rules. There is some debate as to just how prevalent custom pricing will become for consumer goods – a recent attempt at custom pricing created bad press for one online merchant. But custom pricing is certainly a fact of business in many corporate sectors, and is a basic requirement for a content integrator.

Similarly, the chosen content needs to be formatted correctly for different recipients. Some content integration systems assume a "receiver-makes-right" environment, where it is the job of the integrator to get the data formatted correctly. However, more powerful net markets are legislating formats for various XML content that they receive – a "sender-makes-right" architecture[1]. This places the burden of correct formatting on the suppliers. Of course suppliers would like to sell their products in multiple markets. And the market-makers would prefer that the suppliers participate in their markets. Hence it is in the interests of all parties to get the suppliers "hooked up" into integrators' legislated formats – a problem known as *supplier enablement*.

Tools to help with the problems of custom publishing and supplier enablement are critical to content integration. Like the previous examples, the rules and transformations for custom syndication must be easily specified and checked by non-technical content managers.

3.1.1 Content Mapping: Themes

Clearly there are numerous challenges in mapping content, both in bringing it into a unified framework, and syndicating it out in custom versions. One important theme is that a clean data modeling environment is needed at the integrator, in order to make sense of the task. Content integration fundamentally requires a simple, powerful framework for

[1]We use this networking terminology here for clarity. However the challenge of "making right" in this context is far more complex than in the traditional networking context, which is concerned with issues like agreeing on byte ordering.

internal content representation at the integrator.

But the most important theme here is that we need powerful, easy-to-use tools, to address the broad challenges of cleaning, transforming, combining and editing content. These tools must be targeted at typical, non-technical content managers. In order to be usable, the tools must be graphical and interactive, so that content managers can see the data as it is being mapped. Any automated techniques must be made clearly visible, so that domain experts can edit and adjust the results. The development of semi-automatic content mapping and integration tools represents a new class of systems challenges, at the nexus of query processing, statistical and logical mapping techniques, and data visualization.

These challenges can be more difficult and interesting than traditional systems problems, which focused solely on computer performance and correctness. More systems work should include the ultimate performance figure of merit (and the bottom line for industrial purposes): the cost of a person using the system to perform a task, factoring in both time and expertise.

3.2 Query Processing Issues

In the previous section we covered requirements that touch on setting up a content integration system, both in terms of access and modeling. We now turn our attention to the challenges that arise in making a content integration system perform efficiently and flexibly, taking into account the administrative realities of E-Business.

Characteristic 5: Content to be integrated is largely operational information and is increasingly volatile.

Product availability information is fundamentally volatile. Our second example in Section 1 entailed integration of hotel room availability – when the last room is sold, there are no more. In the airline example of the last section, when the last seat is sold there are no more – unless you are a Platinum member, in which case there are. Hence, any integrator must deal with the volatility of this data element correctly, according to business rules. Stale data is unacceptable in these cases.

In addition, we expect product prices to become increasingly volatile. Obviously commodities like oil have always had volatile prices, as have securities. However, the volatility of prices of other items will only increase as the web allows innovative market mechanisms (other than fixed prices) to be explored. For example, the price of airplane seats is becoming increasingly volatile as airlines try to maximize revenue from seat mining.

The effect of content volatility is profound. Any modern content integration solution must be able to **fetch on demand**, retrieving volatile data from the content owner at the time a user wants to know. This is critical both for time-varying data, and for data that is generated functionally from business rules or software "agents" that automatically generate data like prices. On the other hand, to provide the best response time, a content integration system must also support **fetch in advance** for slowly changing information. A modern content integration solution must often employ both strategies over a single body of content. For example the address of the hotel and its amenities are static data and can be fetched in advance, while room availability is highly volatile and must be fetched on demand.

Fetch on demand to multiple content sources fundamentally requires a federated query system. Such systems decompose queries over global schemas into a collection of local queries. Federators have typically also supported views of tables in the global schema for user convenience. Suppose slowly changing data is defined in a view, the view materialized at one or more sites, and then refreshed at a user-specified interval. In this scenario, slowly changing data is elegantly cached closer to the location of the user who requires it and better response time can be obtained.

In contrast, many vendors are trying to solve content integration problems using data warehousing approaches. Warehousing systems are built solely around the "fetch in advance" paradigm. To deal with volatile data, they suggest refreshing the warehouse more frequently, which is neither scalable nor sufficiently close to real time. This paradigm fundamentally breaks when live information is required.

We see no clean way to extend existing warehousing technology to support fetch on demand. One simple issue is that data warehouses are built on parallel database systems, which typically do not support federated query optimization or execution. An additional (and in some ways more fundamental) roadblock comes from the "Extract-Transform-Load" (ETL) tools used to normalize data and bring it into a warehouse. These tools are engineered around batch processes, and involve non-standard imperative scripting languages that are not easily incorporated into query processing. In essence, the ETL tools gave up on data independence, leading to nasty problems of data lineage through arbitrary code. By contrast, federated systems do not distinguish logically between views that transform data on demand, and materialized views that have been pre-loaded; the query optimizer treats these as alternative physical database designs, and applications are shielded from changes in the caching policy by data independence [8]. There is no reason not to build data warehouses over federated database technology, but after millions of dollars of warehouse projects in the 90's (a large fraction of which failed), most corporations are no longer able to make the shift.

There is a large body of research on materialized views [6]. In our opinion, this work needs to better address basic issues of usability. As used today, materialized views must be specified in advance, adding to the administrative burden. In a dynamic environment, they must be dropped, added and refreshed manually as circumstances change, again leading to an administrative burden. It is of course attractive to try developing automatic techniques for addition, alteration, and deletion of materialized views [1], but the design space seems unattractively large in a full-scale operational environment, especially in the federated context.

As an alternative, it would be interesting to devote more work to flexible mechanisms to perform fetch in advance. Something closer to semantic caching [3] or prefetching seems attractive. Regardless of the nomenclature, the integration of distributed query optimization and intelligent caching is a key systems performance and usability issue in content integration.

Characteristic 6: Content integration solutions must support ad hoc queries using existing and emerging standards.

Long gone are the days when content was subject to a specific mode of access. Now a user must be allowed to ask a variety of ad hoc queries, and to make up her own if she so chooses. Some companies in the XML space have not realized this yet, but the analogy to traditional ad hoc queries is straightforward – users will inevitably evolve to request "custom documents" constructed from existing documents. To support ad hoc access, any serious content integration solution must support a query language. Obviously, the query language must allow any field to be searchable, regardless of whether the data in it is a number, text string or other kind of information. Such general searchability is sometimes given the grand name *parametric search* in marketing documents for information retrieval systems, though most of us are used to this functionality as a basic component of traditional database query languages.

Although it is possible to invent a proprietary query language for content, that would fly in the face of the desire of the entire information technology market to use standard mechanisms. In the query language arena today, this requires the use of the standard SQL language, which is the interface to modern data base systems. The XML standards groups have not been idle in this sphere, and in short order this will also require support for emerging XML-based query access like XQuery [2]. In the meantime it requires support for XPath and XSLT.

We fully expect that a content integration solution must support multiple standard query languages (e.g. SQL and XPath today, SQL and XQuery tomorrow) as well as multiple output formats (e.g. SQL result sets and XML documents). We do envision an eventual merger of these standards, but if the SQL language specifications provide any lessons, such a merger will be long in coming and time-consuming to deliver.

Characteristic 7: Content integrators require information retrieval capabilities, including synonyms and fuzzy search.

Obviously, a content integrator must be able to employ his favorite taxonomy to map product names and other fields into a standard representation. However, when a user issues a query, he should not be required to know the standard names for parts in the taxonomies. Clearly, a query for "India ink" should return the same answer as one for "black ink". Hence, ad hoc query systems should support the use of synonyms of this sort.

However, a further requirement is to support even more flexible querying. A user should be able to ask for "ink, black", "black India ink", "inkpen refills", or "ink". A query for "cordless drills" should fetch similar records to one for "drlls: crdlss".

To support this flexibility, a system must support fuzzy (or approximate) matching of fields. A user of the ad hoc query system must be able to specify "fuzzy" mode to allow this larger set of catalog objects to match his query. Avoid any content integration solution that does not support both synonym search and fuzzy search.

As a result, content integration has requirements that are presently satisfied by traditional information retrieval engines. Object-relational database engines typically have loosely-coupled integrations with text search engines, thereby providing text indexing as a local site capability. Any content integration system that can wrap a web search engine can take advantage of it for information retrieval over the full web (and the information there can be exploited in clever ways for local search, too.) It is also possible to directly implement inverted indexes in a single site data system or to model them on standard B-trees as discussed in [11]. Such a solution presents performance problems if transactional semantics are to be maintained. The best way to mix efficient textual search with structured data search in a data federation is a topic that deserves more careful attention in the research literature.

Characteristic 8: Content integration requires high availability, scalability and load balancing.

The web makes the failure of essentially all applications visible to outsiders. Hence, the outages experienced by eBay were front-page headlines and adversely affected the share price of the enterprise. Put differently, one must assume that a bad outage will land your enterprise on the front page of the New York Times and cost the company 15% of its market capitalization.

To cope with this new reality, web applications must be even more reliable than the traditional mission-critical enterprise applications that preceded them. Hence, the web redefines high availability, and "five nines" (99.999% uptime) is the goal commonly desired. As such, the application can be down about five minutes per year – and preferably those five minutes should be spread nicely across the year.

Current hardware and software components do not have this level of reliability. Achieving reliable systems on less reliable hardware and software requires content replication and site failover. This tactic is known as a "hot standby" and is effective at supporting a high availability environment. Of course, the cost of a hot standby is a doubling of all hardware resources.

A second (much cheaper) tactic is to fragment content and store portions at multiple sites. In this case, the failure of one site only causes a portion of the content to become unavailable. This fragmentation of content results in an intermediate availability between that of a central site and that of a replicated environment, delivering *some of the content all of the time*. A combination of replication and fragmentation can deliver *most of the content all of the time*, and is the design of choice in most high-availability environments.

The goal of any E-Business is to be highly successful. eBay, for example, is running several hundred transactions per second and anticipating an order of magnitude greater volume. B2B exchanges are currently supporting much less volume but hope for more business. The larger MRO exchanges will presumably have large transaction volumes over time.

Obviously, a content integration solution must be architected to scale incrementally, over several orders of magnitude in transaction load. The best solution is an architecture with no limits on scalability; a customer can simply scale the solution by adding more hardware – preferably without a reboot.

To achieve this level of scalability, a content integration solution must support both fragmentation and replication of content. Fragmentation allows the content to be spread over several machines; if additional scalability is required, the data can be repartitioned over more machines, and the transactions dispersed more widely. Replication allows the load to be shifted arbitrarily across machines. In this case, a strategy for load balancing is required to keep all machines equally busy.

As a result, an **adaptive, load-balancing federated query processor** is a required service. Also, a federator must scale to hundreds, if not thousands, of sites, and be able to perform query optimization at this scale including dealing with table fragments, materialized views and replicas. The architecture proposed in Mariposa supports both adaptive load balancing and scalability using an agoric optimizer [13, 4]. Tuning and refinement of agoric optimizers is an interesting topic for research. In addition, there may be even more elegant mechanisms that are possible [7], though none of the other proposals to date have tackled distributed query processing.

In contrast, we see no way for compile-time, centralized cost-based optimizers to provide required scalability or adaptivity. Hence, almost all of today's commercial distributed and heterogeneous systems are unacceptable for serious content integration, and there are precious few contenders in the research literature.

3.2.1 Query Processing: Themes

The content integration arena raises fundamental new challenges in query processing. In terms of interfaces, it requires both standard query support, and the ability to integrate very interactive, fuzzy search interfaces. In terms of the data sources, there is a serious lack of control or predictability in the federated query environment. Multiple kinds of data sources can be queried on the fly. The query system must deal gracefully with the changing needs, capabilities and performance of these sources – issues that lie outside the control of the query system itself. The optimizer has to scale to large numbers of systems and replicas, and continue to choose intelligently as individual changes are made at those systems. In the face of all this volatility, the system must not only be efficient, it must be robust and predictable, scaling gracefully as the cross-business environment scales. This problem is a logical extension of the research agenda first set out by Codd, which separates applications from performance issues via a clean declarative level of indirection. Content integration raises the complexity of bridging that level of indirection – a challenge for the research community, and a business opportunity as well.

4. COHERA'S SOLUTION

The Cohera Content Integration System$^{\mathrm{TM}}$ aims to solve the challenges listed above. Cohera includes a number of components; we give a brief overview here.

- **Cohera Connect**$^{\mathrm{TM}}$ provides access to HTML, XML and text data either over the web, or via a file system. It includes a full-function web browser agent, which can automatically navigate complex web pages, correctly managing issues like DHTML, JavaScript, cookies, passwords, and HTTPS. Cohera Connect comes with an intuitive graphical "training" interface for generating HTML and XML wrappers; these wrappers can operate either on regular expressions or by navigating the Document Object Model (DOM) corresponding to a document. Expert users can also customize wrappers directly with XSLT transformations or JavaScript code. Cohera Connect can present a traditional ODBC or JDBC interface to query applications, or can be configured to directly generate complex XML at its output.

- The **Cohera Workbench**$^{\mathrm{TM}}$ is a graphical content workflow to help content managers model, map, transform and syndicate content. Inspired in part by [12], it supports natural graphical versions of a variety of logical data transformations, and includes rules for detecting data discrepancies and guiding the content manager through the task of fixing them. It includes a semi-automatic taxonomy mapper, which suggests good matches for categories, and highlights conflicts which can be edited or accepted. The Cohera Workbench also includes a content syndication environment for specifying and transforming the content to be generated for different users. All of the Cohera Workbench features are supported within an interactive graphical environment that keeps the content manager "in the loop" during the mapping process.

- **Cohera Integrate**TM is a federated query processing engine, representing the commercial state of the art in distributed query processing. It is based on the agoric, federated query processor architecture of the Mariposa system developed at Berkeley [13]. Cohera Integrate enhances the Mariposa design with a new agoric optimizer, materialized views, and support for multiple kinds of data sources – including native support of relational databases and XML. AltaVista's text search engine is compiled directly into the query engine, and fully modeled by Cohera Integrate's optimizer as an access path for text searching and other IR services. Cohera Integrate provides an object-relational SQL that ships with fuzzy and pattern-match functions, providing a full range of query and search facilities. It also supports XPath queries over integrated XML views of the data. Because of Cohera's scalable agoric optimizer, new compute and cache machines can be added to a Cohera installation incrementally via a graphical interface; the optimizer takes advantage of them as soon as they are added, with no need for downtime.
- Cohera optionally provides full SSL encryption between its components, to allow for secure E-Business communication across public channels.

5. CONCLUSION

In response to the latest wave of business use of the web, this paper has defined a new class of information integration, which we called content integration. It arises from the desire of enterprises to do business over the Internet, and entails integrating operational structured and unstructured data between enterprises.

In this paper we presented characteristics that are shared by the content integration problems that we have seen. These include:

- Varying relationships with content providers
- Semantic heterogeneity
- Multiple taxonomies and schemas
- Custom Syndication
- Increasingly volatile content
- Ad hoc queries
- Information retrieval capabilities
- Serious scalability and availability

Architecturally, we believe that a data federator with materialized views is the best approach to providing elegant content integration. Warehousing solutions can implement few of these features, while traditional distributed DBMSs usually lack materialized views, support for semantic heterogeneity and have unusable compile-time cost-based query optimizers. In contrast, Cohera contains a novel agoric optimizer and supports materialized views and transformations. Moreover, it has intelligent, interactive GUIs for defining transformations, syndication and web-site scraping. Lastly, it includes information retrieval, taxonomies and automatic classification capabilities in an elegant way. As such, it is unique in the marketplace in providing a complete content integration solution.

In this paper, we have also indicated some of the hard research problems that deserve further study. These include the challenges of extreme scalability, adaptive query processing and load balancing, and easy-to-use semi-automatic content management tools. We hope that the research community will embrace this problem area and its various challenges, and help to provide more powerful, more usable solutions.

6. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated Selection of Materialized Views and Indexes in SQL Databases. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 496–505. Morgan Kaufmann, 2000.

[2] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. XQuery: A Query Language for XML. Technical report, World Wide Web Consortium, 2001. http://www.w3.org/TR/xquery.

[3] S. Dar, M. J. Franklin, B. T. Jónsson, D. Srivastava, and M. Tan. Semantic data caching and replacement. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 330–341. Morgan Kaufmann, 1996.

[4] A. Deshpande and J. M. Hellerstein. Decoupled Query Optimization for Federated Database Systems. Computer Science Division Technical Report CSD-01-1136, University of California, Berkeley, Mar. 2001.

[5] D. Freitag. Information Extraction from HTML: Application of a General Machine Learning Approach. In *Proc. 15th National Conference on AI*, pages 517–523, 1998.

[6] A. Gupta and I. S. Mumick. *Materialized Views : Techniques, Implementations, and Applications*. MIT Press, May 1999.

[7] J. M. Hellerstein, M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. Shah. Adaptive Query Processing: Technology in Evolution. *IEEE Data Engineering Bulletin*, 23(2):7–18, June 2000.

[8] J. M. Hellerstein, M. Stonebraker, and R. Caccia. Independent, open enterprise data integration. *IEEE Data Engineering Bulletin*, 22(1):43–49, 1999.

[9] A. Jhingran. Moving Up the Food Chain: Supporting E-Commerce Applications on Databases. *SIGMOD Record*, 29(4):50–54, 2000.

[10] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97*, volume 1, pages 729–737, Nagoya, Japan, Aug. 1997.

[11] C. A. Lynch and M. Stonebraker. Extended user-defined indexing with application to textual databases. In *Fourteenth International Conference on Very Large Data Bases, August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings*, pages 306–317. Morgan Kaufmann, 1988.

[12] V. Raman and J. M. Hellerstein. An Interactive Framework for Data Cleaning. Computer Science Division Technical Report CSD-00-1110, University of California, Berkeley, Sept. 2000.

[13] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 5(1):48–63, 1996.

[14] UDDI Technical White Paper. Technical report, UDDI.org, 2001. http://www.uddi.org/whitepapers.html.