Beauty and the Beast: The Theory and Practice of Information Integration

Laura Haas

IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120
laura@almaden.ibm.com

Abstract. Information integration is becoming a critical problem for businesses and individuals alike. Data volumes are sky-rocketing, and new sources and types of information are proliferating. This paper briefly reviews some of the key research accomplishments in information integration (theory and systems), then describes the current state-of-the-art in commercial practice, and the challenges (still) faced by CIOs and application developers. One critical challenge is choosing the right combination of tools and technologies to do the integration. Although each has been studied separately, we lack a unified (and certainly, a unifying) understanding of these various approaches to integration. Experience with a variety of integration projects suggests that we need a broader framework, perhaps even a theory, which explicitly takes into account requirements on the result of the integration, and considers the entire end-to-end integration process.

Keywords: Information integration, data integration, data exchange, data cleansing, federation, extract/transform/load.

1 Introduction

"For us…growth is a way of life. So we'll continue to grow"[1]. Nearly two thirds of CEOs surveyed recently said that growth is the key priority, requiring quick delivery of new products and services in response to rapidly changing market conditions [2]. Yet only 13% felt that their business was well-positioned to react quickly [2]. They stressed the need to capture and understand all available information to make rapid business decisions, but today that is not such an easy task. In fact, 68% of the CEOs listed the integration of disparate applications and infrastructure as a key issue for their business, one that slows them down and stops the flow of information [2]. Meanwhile, customers tell us that 30% of their people's time is spent just looking for the information they need to do their jobs.

Why is information so hard to find? Partly, this is due to the increasing volumes of information available on line. But there is a second, deeper problem, and that is the fragmentation of information, and the proliferation of information sources. Even

[1] Mukesh Ambani, chairman and managing director of Reliance Industries, India's largest private sector company, as quoted in [1].

within a relatively controlled environment such as an enterprise Information Technology (IT) organization, customers report many database instances, often hidden behind applications, not to mention document repositories and other sources of unstructured information. For example, analysts report [3] that 79% of companies (of all sizes) have more than two document stores, while 25% have more than fifteen. Information is not only hard to find, but further complications such as overlapping, conflicting and incomplete information are inevitable. Almost any business with multiple business units has multiple sources of customer information, for example – often with conflicting information for the same customers.

Information integration is the database community's answer to these problems. The goal of information integration is to enable rapid development of new applications requiring information from multiple sources. This simple goal hides many challenges, from identifying the best data sources to use, to creating an appropriate interface to (or schema for) the integrated data. Much research has focused on how best to do the integration itself, for example, how to query diverse sources with differing capabilities and how to optimize queries or execution plans. Other issues concern how to cleanse information to get a consistent view, how to deal with uncertainty and trace data lineage, and how to identify the same object in different data sources (a problem known by various names, including entity resolution). There has been a lot of progress on individual challenges, but information integration remains a difficult task. We believe that one reason for that is that these challenges are inter-related, part of the overall process of integration, and yet have been largely considered in isolation. Thus, the separate solutions do not always work well together. Perhaps more importantly, the solutions that are relevant to a particular integration task depend heavily on the application requirements regarding data qualities (*e.g.*, currency, consistency, completeness) and quality of service (*e.g.*, response time, availability, resources consumed). We lack a clear view of information integration that positions the various technologies relative to each other and relative to the application requirements for the integration problem that must be solved.

The rest of this paper is structured as follows. Section 2 elaborates on the overall information integration challenge, and presents an extended example of a real integration problem to motivate our suggestions for future work. In Section 3, we briefly survey the research underpinnings of information integration, showing how the research applies to our example, while Section 4 reviews the state of the art in the industry today, showing what products are available for use. Section 5 comes back to the issue of unification and the end-to-end information integration problem. We pose a new challenge to the research community with both theoretical and systems implications, and explore several possible approaches. Finally, the paper concludes in section 6.

2 Information Integration Illustrated

There is no one integration problem; the challenges vary depending on the environment. In this paper, we focus on information integration within an enterprise.

This environment typically includes a broad mix of sources, many structured (*e.g.*, relational or other databases), but increasingly many unstructured (*e.g.*, document repositories, web pages, email). The uses for the integrated data are likely to vary greatly, from mission-critical applications to exploratory queries and everything in between. A broad range of technologies is used to handle this range of needs. In this section, we first provide an overview of the integration process in the enterprise context, and then illustrate it through an extended example.

2.1 The Information Integration Process

Research on information integration has focused on particular aspects of integration, such as schema mapping or replication, individually. But for businesses, information integration is really a process, with four major tasks: *understanding*, *standardization*, *specification* and *execution*.

Understanding. The first task in information integration is to understand the data. This may include discovering relevant information (including keys, constraints, data types, and so on) and analysing it to assess quality and to determine statistical properties (for example, data distributions, frequent values, inconsistent values). During this task the integrator may look for relationships among data elements (such as foreign keys, or redundant columns) and possibly (for unstructured data) meaning. Metadata is central to this phase, though used in all. Both tools and end users leverage it to find and understand the data to be integrated. It is also produced as the output of analysis, to be exploited by later tasks in the process.

Standardization. This task typically leverages the work of the previous task to determine the best way to represent the integrated information. This includes designing the "target" or integrated schema, deciding at the field level what the standard representation should be (*e.g.*, will full names be represented as first name followed by last name, or last name comma first name?), and even defining the terminology and abbreviations to use ("str" vs. "st" for "street"). In addition to these rules on how data is represented, other rules that specify how to cleanse or repair data may be provided. Issues here include how to handle inconsistent or incomplete data (for example, if we find multiple phone numbers for the same person, should we keep all of them, or only the most recent?) and how to identify data that refers to the same objects (for example, is John Q Public the same person as John Public?).

Specification. In this step, the artifacts that will control the actual execution are produced. As a result, the techniques and technologies used for specification are intimately linked to the choice of execution engine(s). For example, mapping tools specify the relationship between source(s) and target(s), and then typically can generate a query or other executable artifact (*e.g.*, XSLT) that would produce data in the desired target form. Often, however, the specification is part of actually configuring an integration engine to do the desired integration. Thus, determining the execution engine should be thought of as part of specification.

Execution. This is where the integration actually happens. Integration can be accomplished via materialization, federation and/or indexing. ***Materialization*** creates and stores the integrated data set; this may be thought of as eager integration. There are many techniques for materialization. *Extract/Transform/Load (ETL)* jobs extract data from one or more data sources, transform them as indicated in the job script, and then store the result in another data source. *Replication* makes and maintains a copy of data, often differentially by reading database log files. *Caching* captures query results for future reuse. ***Federation*** creates a virtual representation of the integrated set, only materializing selected portions as needed; it can be thought of as lazy integration. Federation is a form of ***mediation***; in general, mediation refers to an integration technique in which requests are sent to a "mediator" process which does routing and translation of requests. ***Search*** takes a different approach, creating a single index over the data being integrated. This is commonly used for unstructured data, and represents a partial materialization, since typically the index identifies relevant documents, which will be fetched dynamically at the user's request.

Note that these tasks are interdependent, and existing tools often support (pieces of) several of these tasks. They may be overlapped in practice; for instance, it is not necessary to have a complete understanding before starting to standardize. Likewise, a particular integration may not require all of the subtasks for any task, and in really simple cases, some tasks may seem to vanish altogether.

The integration process is iterative, and never-ending. Change is constant; there is always another source to deal with, a new application with new requirements, an update to some schema, or just new data that requires analysis.

2.2 An Extended Example

Consider a typical integration problem. A major company, Grande, acquires a small company, Chico, with less than fifty employees. Chico has three products, several "databases" per product (ranging from design docs scattered about the file system to requirements docs in a document management system to relational databases tracking line items and owners, and so on), two orders databases (one for mail orders, one for the web), a defect tracking database for support, and other information sources. Several Chico IT staff members quit in the transition, so knowledge about the data is lost.

The combined enterprise needs to ensure that Chico continues to do business during the transition (so their sales, support and development databases and processes must continue to operate). But the duplication of databases, processes and IT staff is costly, so they also need to consolidate their operations reasonably quickly. In the meantime, they have immediate needs to correlate information across the old and new systems. For example, Chico's customers overlap with Grande's. The new, bigger Grande wants to send mail to all existing customers who might be interested in Chico's products, but not to those who already have them. They may want to quickly get to a single phone number for support across the combined product set.

For our example, we'll focus on this latter scenario. The support representative answering the phone needs to be able to check customer entitlement quickly, *i.e.*, he

must be able to look up a customer and discover the level of service for which the customer has paid. Ideally, this would be solved by providing the support person with a single list of customers, duplicates removed, information merged, or the

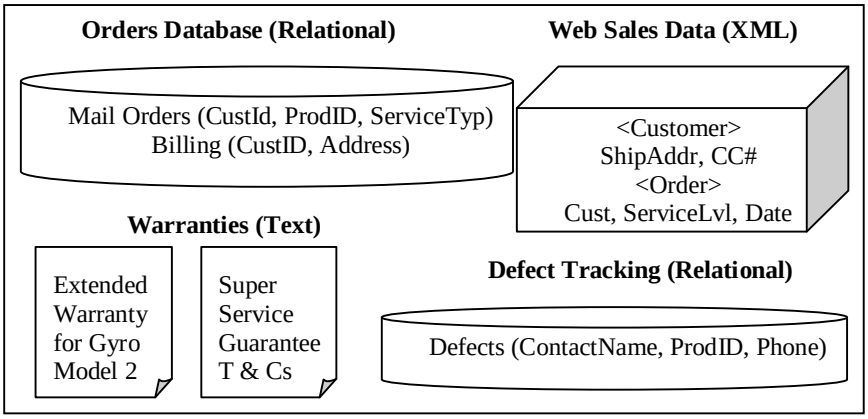

Fig. 1: Chico's customer-related data is spread over multiple data sources, in multiple formats. Only partial schemas are shown; the real data would have many more tables, and columns, as well as richer XML document types. Gathering full information on customer entitlement requires detailed knowledge of the sources, a complex join and understanding the service types.

equivalent. But that is not so easy, as customer information for Chico is scattered across multiple tables; there is no single customer list, even for Chico alone. Further, Chico checked entitlement by looking up the product registration number in the orders database(s) to see if the customer bought support, and if so, at what level. Grande was more focused on customer accounts, with a certain level of service for a customer based on overall sales (*e.g.*, Gold vs. Platinum customers). So not only is the information about customers organized differently, but the semantics of customer entitlement are also different for the two companies. Eventually, the combined company will want to settle on a single scheme, but in the short term, they just want to continue to support both customer sets, smoothly.

Janet Lee, a Grande IT architect, is asked to set up this combined customer support system. Janet is not familiar with the Chico systems, of course, and because of the loss of Chico staff, she will not have the benefit of their expertise. She needs first to *understand* what information is available to her. She will need to find the Chico customer information and information on what types of support exist. This information is spread over order, billing and defect tables in multiple databases and in the document management system that tracks contracts (Figure 1). She will probably need to talk to someone in sales to understand the Chico support semantics, and she will likely want to inspect or analyze the relational data so that she knows what she is up against in terms of *standardization*. For that task, she will need to specify how to represent various data, such as address (Chico doesn't store state, and the address is all in a single field, where Grande has state, city and zip all in separate fields). She

will design a merged representation, and define the rules that determine what happens when there is disagreement (for example, when the same customer appears in both databases with different information).[2] She may also need to write rules to determine when data refers to the same customer. Janet then needs to *specify* how to do the integration. Although there are some tools that will support an abstract, nonprocedural way of doing this, for example, mapping tools, even these today tend to be associated with a particular integration engine.

So Janet now faces the question of how to *execute* the actual integration. She could, for example, choose to materialize the combined customer list. To do that, she'll need to define an Extract/Transform/Load (ETL) job, deciding how often it needs to run, or whether it can be run once, and then refreshed incrementally at regular intervals. A differential refresh might be better handled using a replication product – and if the transformations needed are simple enough, the replication engine might be *all* that is needed. In any case, she will need to set up one or both products and configure them to reach the actual data sources, and to run her job or jobs. Alternatively, she could choose to federate the various data sources, providing a single (virtual) view of the combined data. In this case, she will need to set up and configure the federation software to reach the data sources, and then define appropriate views over the data. In a customer support environment, a good search capability is typically critical. It is possible that Janet could return the information needed by the support representative just via search. She will want to evaluate that possibility, and possibly set up search software, configuring crawlers, getting an index built, and so on. Of course, a combination of these various integration engines may be the best approach – materializing critical information for the first, fast check, keeping it up-to-date via replication, using federation to "drill down" if the customer has further questions about their account, and using search as a way to retrieve details of the relevant service plans.

How will Janet decide? To make her decision, she must take a number of factors into account. She will think about the requirements of the task: how quickly must the information be returned, how many end users will be using the system simultaneously, whether it will be needed 24x7, and so on. Other requirements apply to the data quality: how current must the data be, how complete, how accurate. Janet will also think about the physical constraints on the solution, for example, how much storage space is available, the processing power at her disposal, a limit on total cost, perhaps. Finally, the policies Grande has in place for security or to comply with relevant industry regulations will also affect the solution. For purposes of this paper, we will refer to these varied types of requirements – qualities of service, qualities of data, physical constraints and policies – as the *solution desiderata*. These desiderata are critical to determining the best techniques to use for a particular scenario; however, only Janet's experience allows her to make the decision – there are no studies or formal guidance on what desiderata require which integration techniques.

To summarize, in order to integrate enough data for this one critical but simple scenario, Janet must go through quite a complex process. She will need to develop an understanding of the data and its semantics. She will need to assess the quality and

[2] Note that the order of these may vary a bit, for example, some tools would allow Janet to write these "cleansing" rules as part of specifying the integration.

degree of overlap of the data, identifying common customers and merging and standardizing their information while dealing with any inconsistencies. She will need to design the integrated view or target schema. Finally, she will need to choose one or more integration engines to deploy, configure them to reach the various data sources, and create the instructions needed (the ETL script, view, or program) for them to instantiate the target schema. In our simple example, Janet is dealing with primarily relational data. If some of the data is unstructured, her task is harder still. Fortunately, the research and development communities have made great strides towards tools to address these challenges. In the next two sections, we examine some of the highlights of this work.

3 Research in Information Integration

There are thousands of papers relevant to information integration. Many focus on some aspect of one of the stages of the process described above, *e.g.*, discovering primary keys (one piece of understanding the data). Others propose broader solutions for specific environments, for example, querying deep web data sources. It is beyond the scope of this paper to survey the literature (see [4] for an excellent introduction). Instead, we categorize the work into four broad areas, one for each step in the integration process, and provide a few pointers to work in each category, to give a feeling for the accomplishments to date. Not all of the literature is amenable to this crude categorization, as we also briefly illustrate. Despite the weighty body of literature, the information integration challenge is far from solved, especially in the enterprise context.

In the area of understanding the data and data sources, there has been much recent cross-disciplinary work (spanning data management, information retrieval, statistics and machine learning). Key areas of focus include structure discovery [5], which aims to determine the schema for data automatically, data summarization and analysis [6, 7], to determine characteristics such as value distributions and dependencies, text analytics [8], which tries to find specific concepts in text, and source selection [9, 10], which chooses the best data source(s) to answer a particular query.

Research on standardization has focused around several aspects of reconciling different data sets [11, 12]. A key challenge is *entity resolution* (often known as semantic resolution or deduplication), the problem of determining when two data objects refer to the same real-world entity [13]. Other aspects under study include dealing with inconsistent data [14, 15], and how to measure quality and incorporate it in systems [16]. In general, if data can be inconsistent, there can be uncertainty, sparking a renewed surge of interest in probabilistic databases [17].

In specification, the major topics of interest have been schema mapping and schema matching [18, 19]; although work on dataflow systems [20] and workflow [21] is also relevant, these technologies have not typically been applied to information integration by the research community (though they are used in enterprises). Model management [22] takes a broad view of managing and manipulating schemas. Schema mapping tools such as Clio [23] help the user align a target schema with (potentially multiple) source schemas, allowing a nonprocedural specification and typically

generating the runtime artifacts needed to populate the target schema from the source(s). Dataflow programming could be used as a more procedural way to specify how to create the target data; workflow tools are similarly procedural, but centered on the operations rather than the data.

There are many ways to integrate information. As described in our example, materialization, federation, search, as well as "application integration" techniques (workflow or business process integration, hard-wired code, composing Enterprise Java Beans, and so on) all may apply to the execution step. Initially, the research community focused on integration via materialization, with emphasis on data transformation [24], and replication [25, 26]. In the early 1980's, attention shifted to querying across distributed databases [27, 28], and more recently, to mediation [29, 30, 31] approaches. The Garlic project [32] explored a form of mediation now known as federation, which extended a relational query processor [33], and thus fit easily into enterprise environments. Database theory has made strong contributions in this area, both formalizing and extending these basic techniques [4, 34, 35]. While search [36] was initially conceived of as a way to find unstructured information, it has rapidly become a means of information integration [37], though with radically different properties than either materialization or federation. While those integration techniques allow for precise queries spanning data from multiple sources with structured results composed of data from multiple sources, search poses an imprecise query to one or more sources, and returns a ranked list of results, each typically from a single source. This form of integration is "good enough" for some integration scenarios, requires much less work for the initial three integration tasks in our process, and may also be used as an aid to understanding the data.

Of course, not all work fits nicely into one of these categories. For example, many papers are now focusing on integration in the context of the world-wide web [38]. These papers often tackle multiple steps, but in this narrower context. Likewise, much research has been done on integration in the context of bioinformatics [39]. Specialized integration languages [40] and the use of domain ontologies [41] have gained some traction in this community.

This discussion is far from comprehensive, but gives a flavor for both the broad range of problems and the types of approaches that have been taken. The results have led to great progress in the tools available to the industry, as we show in the following section. However, while research has solved subsets of the overall problem, there is little today in the way of complete and unified solutions.

4 The State of Information Integration in Practice

Out in the marketplace, tools for integrating information are proliferating. Many smaller companies sell products addressing one or more of the integration steps we have enumerated. Meanwhile, larger companies, most notably IBM and Informatica, are consolidating tools for the various steps into powerful platforms for information integration and access [42, 43]. Rather than trying to cover all the products on the market, we will instead describe in some detail the present market leader, namely, IBM Information Server [44].

Understand:

- WebSphere Information Analyzer
- WebSphere Business Glossary
- Rational Data Architect

Standardize:

- Rational Data Architect
- WebSphere QualityStage

Specify:

- Rational Data Architect
- Each execution engine

Execute:

- WebSphere DataStage
- WebSphere Federation Server
- WebSphere Replication Server
- WebSphere Data Event Publisher

Operational platform:

- Connectors to databases, applications, and web sources
- WebSphere Metadata Server and Metadata Workbench
- WebSphere Information Services Director

Fig. 2: Individual products comprising IBM Information Server, listed by the integration task they support. Additional products in the suite provide a common platform for the products listed to run on. That platform includes connectivity to a broad range of sources, shared metadata, and the ability to invoke the various products as services, from a range of different programming environments.

IBM Information Server (IIS) is a platform for information integration. It consists of a suite of products (Figure 2) that together cover all the integration tasks. There are multiple products for any task. For example, IIS includes three products, each aimed at a different level of understanding of the data. WebSphere Information Analyzer analyzes source data, discovering schema elements such as primary and foreign keys, and checking adherence to integration and quality rules defined by the user; it supports understanding the physical data. It provides detailed profiling of the data in each column (cardinality, nullability, range, scale, length, precision, etc.). WebSphere Business Glossary lets the user define and manage a vocabulary for their enterprise, and link the terms to the underlying data (providing business-level understanding). It is designed for business users and subject-matter experts to define data standards and record business terminology definitions, rules and taxonomies. It is useful both for understanding and standardization in our framework.

IBM Rational Data Architect (RDA) is a full-function data modeling tool that can be used with any database management system. RDA supports understanding of data at the logical level. It allows the design and exploration of logical schemas, including relationship discovery (*i.e.*, finding foreign keys), and production of physical schemas. RDA also incorporates the Clio [23] mapping capabilities, which are useful in conjunction with an integration project. Thus, RDA also spans our understanding and standardization tasks. In fact, from a mapping, RDA can generate out the

artifacts needed by the federation engine, hence it handles specification for federation as well.[3]

IIS includes WebSphere Metadata Server to capture the insight gained (and standardization decisions made) in using these products and to make that knowledge available to the other tools in the suite. Metadata Server provides a unified repository for metadata access and analysis. It provides import and export to twenty common modeling and business intelligence tools, as well as being leveraged by the products of the IIS suite.

A key component of standardization is data cleansing, provided for IIS by WebSphere QualityStage. QualityStage allows the user to set the formats for data records, examine, correct and enrich data fields, and match and link records that may represent the same object. The user can specify rules to determine which values should "survive" merging of similar records. A graphic interface allows the user to set up the rules for this cleansing in a dataflow design metaphor, and to tune the rules by observing their impact on a dataset. The dataflow design metaphor of QualityStage is also exploited by WebSphere DataStage, one of several products in the suite aimed at execution. The same graphic interface allows the user to design complex transformation logic visually, exploiting a large library of transforms (shared with QualityStage). DataStage is used for materialization. It can be invoked in batch or real-time, and can extract, transform and load vast volumes of information.

IIS also includes other integration engines, most notably WebSphere Federation Server, which allows query access to heterogeneous data sources as if all the data were in a single (virtual) database. Federation Server supports full SQL and SQL/XML [45] access, with optimized query plans and materialized query tables for caching data. It can be configured graphically using Rational Data Architect, or using a Wizard in its own control center. Other integration engines include several replication products, which allow synchronization of multiple copies of data. Each product addresses a specific set of requirements. For example, one product focuses on flexible scheduling, SQL-based transformation, and support for a variety of configurations to handle typical business intelligence and application integration scenarios, while another focuses on high throughput, low latency replication, for high availability or workload distribution. The suite also provides event-publishing capabilities (allowing database changes to be captured as XML messages and put on a queue for an application to interpret). Event publishing is often used as a way of integrating applications (by sending messages between them), and also as a way to feed information to ETL engines to trigger a job. For example, using WebSphere Event Publisher with WebSphere DataStage allows DataStage ETL jobs to be fed a stream of data values to transform and load, so that it can integrate information in real-time, driven by changes to the data, as opposed to batch processing.

This is a wide range of products, but even IIS is not sufficient for all integration needs. In particular, while several of the products deal with both structured and unstructured data, as a whole they offer more features to deal with structured datasets.

[3] In fact, it handles specification for any integration that can be done by an SQL engine today. Incorporating additional Clio capabilities would give it the ability to also handle XML to SQL, SQL to XML and XML to XML transformations.

Hence, the various engines are increasingly interoperating with related IBM products for content federation [46] and enterprise search and text analysis [47, 48].

This brief description is offered as an example of the types of products that exist today. For each product mentioned, there are many competitive products that could be substituted. Each typically has its own unique strengths. Further, a particular function (*e.g.*, finding relationships among data records) may be present in many products. Different vendors will package functionality differently, depending on the strengths of their products and the market niche they expect to address with them.

Looking back to the extended example from Section 2.2, Janet's pain becomes more concrete. Which products should she use for which steps in the integration? Will Rational Data Architect and WebSphere Federation Edition provide better results than WebSphere QualityStage and DataStage? How can she know that the products she chooses will meet the application desiderata?

5 The "Big I" Challenge

Despite the many products available today, there are still many opportunities for research in each of the basic functions needed. We don't have an ultimate answer on how to tell that two pieces of data refer to the same object, for example. We are still learning how to automate schema integration. The wealth of research pointed to in Section 3 shows how rich an area integration is for new discoveries.

However, in working with customers such as Janet over the last few years, we have come to believe that there is a more global problem that needs to be addressed. The issue that we see is that there is no theoretical – nor much practical – guidance for the many Janets of the world on how to make these choices. This is problematic, because the wrong choice can lead to bad results: orders of magnitude difference in performance, lack of flexibility to accommodate changes in the company's processes, or just difficult, time-wasting implementations.

More concretely, what is wrong with today's products, from the consumer standpoint? There are too many, with too much functional overlap. (We have described a relatively simple situation, in which they were all IBM products. In general, the poor customer would be choosing from many products from many different companies, much less compatible with each other than the IBM suite). Once an integration approach is chosen, it is hard to switch; for example, the work done to use federation today would have to be largely redone to move to a materialization approach, as might be desired if the data were really massive or response time were critical, for example. This is too bad, as federation is much better for rapid prototyping to show the benefits of integration than materialization, which typically requires months (and sometimes years) of effort before the benefits are visible. When you consider that integration is an ongoing effort (new sources and new requirements for information arrive constantly), flexibility becomes a major issue.

The products are also too hard to use together to support a complete integration scenario. Most of the industrial-strength integration products available today have many knobs that must be set to configure them to meet a particular set of requirements. They are typically too focused on their own functionality and not on

smooth interoperability with other tools needed in the integration process. Hence, they may be difficult or expensive to use in combination. The emergence of product suites such as IIS as described above is a start at addressing this problem, but there is still progress needed, especially when dealing with products from multiple vendors.

Ultimately, the time until the customer realizes value from the integration project (and her investment in the tools) is too long, and the costs are too high. Because of the complexity of these decisions, even for a simple case such as that in our example, consulting engagements are frequently needed in even the best-staffed IT departments in order to deliver a successful integration. Bringing in a specialist makes the project more likely to succeed, but with high associated cost. It also adds another step to the process – finding the right expert.

We need enormous advances in integration technology to get beyond these issues. In an ideal world, integration would happen almost automatically. In this world, the user would specify what information he wants, what he wants the result to look like, and what properties he needs the result to have, and the system would "do it" (though probably with user involvement at various points to verify key decisions, etc). This is hardly a new vision. We are basically arguing for nonprocedural information access. In a simpler time, relational databases were the answer to the same quest. The difference is that in today's world, the information needed for a new application is likely to come from multiple heterogeneous sources.

More concretely, we would like to see the various integration execution engines converge, so that, to the user, there is no visible difference between an ETL engine, a replication engine, federation or even search in terms of the first three steps of the integration process. Understanding, standardization and specification should be done the same way, regardless. The execution will just happen, based on the solution desiderata: the qualities of service desired, the constraints, and so on. The user might be blissfully unaware of what execution engine (or engines) does the actual integration under the covers.

To reach this information integration "nirvana", a number of advances in technology are needed. We must raise the level of abstraction significantly above where it is today, where characteristics of individual products become primary concerns for the integrator. A critical challenge is to represent *all* the information needed for the task. Today, an important component of that information is found only in the user's head, or, in a well-disciplined IT department, perhaps in one or more requirement documents. This is the information on the solution desiderata. Because it is not available in a machine interpretable format, we have no way for an integration tool to consider those requirements and hence, automate integration.

The level of abstraction needs to rise in other ways, as well. For example, it must support logical or even business-level descriptions of what information is needed, as opposed to concrete physically-linked descriptions such as column and table names, or existing view definitions. Janet should be able to say she wants customer information and the system should find and deliver it. That will require much richer metadata than we have today, much of which will need to be derived automatically. To support the rising level of abstraction, more sophisticated techniques are needed to automate the various parts of the process, from discovery to entity resolution to configuration and tuning.

There are challenges here for both the theoretical and the more systems-oriented research communities. From the theoretical perspective, we lack a deep understanding of what fundamental operations are needed to integrate information. While Bernstein [22] has suggested a set of operations, we do not know if they are complete, nor do we have precise semantics for them. Is there a unifying theory of integration that subsumes the separate problems of data integration and data exchange? We have posited that achieving our goal requires being able to represent the solution desiderata. What role do these properties or characteristics of the result play, what aspects can be represented and how? We need a model of these desiderata, and how they relate to the integration task. We have wished for fully automatic integration. How close to our goal can we possibly get?

From a systems research perspective, there are several approaches one might take to this challenge. Perhaps the simplest is to start by building an "Integration Advisor", on the model of today's physical design advisors [49, 50]. This tool would lead the user through the various integration steps, asking for input on the desiderata, and then recommend the appropriate engines, and perhaps even generate the configuration information needed. This would simplify the integration process greatly. However, there are still many issues to be addressed in creating this tool, such as what input must the user provide, what really are the tradeoffs among the different integration approaches, which desiderata matter, and so on. Another approach would be to start with a language to express the integration desired (covering data plus desiderata), and then build a system to interpret (or compile) that request against the tools and engines on hand today. In other words, this approach would treat the current set of integration engines as given, and the result of compilation would be a script that invoked one or more of them to accomplish the integration. Alternatively, the system could compile to a new engine with a complete set of operators for integration. In this last case, we are returned to the questions of what is the model of information and what are the basic operations that we posed above to the theory community, as presumably this system would be the interpreter of some subset of those operations.

These are big challenges, and they hide a raft of further interesting problems. How can we deal with uncertainty in a general way within the integration context? How can we exploit the results of the discovery algorithms that are being developed to tell us more about the data? Can we extend our theories of integration to include uncertainty and newly produced knowledge? How much can we model, and how much must we just make informed engineering choices?

We have focused in this paper on an information-centric view of integration. But in fact, the most common form of integration today is still enterprise application integration (EAI). Confronted with a myriad of choices of tools and techniques, many customers fall back on the most popular alternative: writing a special-purpose application, possibly exploiting some workflow or other process support tools. We call this application-level, procedural style of integration *process integration*. Integration at the application level is unfortunate, for several reasons. First, it is not clear that writing a special-purpose application will be simpler even for an easy first project as in our example. All of the initial hard work to understand the data and standardize on an integrated representation will need to be done anyway, and an integration approach chosen, *i.e.*, whether to materialize, federate and/or search.

Without the use of at least some information integration tools, a lot of code will be written to accomplish the integration [51]. Second, when the code for the integration is in the application, it can only be optimized by the programmer, so performance will be only as good as the programmer [52]. Third, it may be harder to reuse the work done for this application when the next application over the same data or data sources comes along.

We will never be able to stop programmers from writing code to do integration if they want to do it. But we can ask how far we can and should go in terms of simplification. What if we could not only relieve customers from deciding whether ETL or federation was the answer, but also unify many of the basic tools that are used in the application for integration (for example, business process integration, message queuing, and so on)? Can we replace all the process integration and information integration techniques with a single integration engine? This is what we refer to as the "Big I" vision: a single engine for all integration needs, which takes a nonprocedural specification of those needs and automatically chooses the right approach or combination of approaches.

6 Conclusion

In this paper, we have presented a snapshot of the world of information integration as it stands today. We have made great progress in both the theoretical foundations of information integration and in the algorithms and tools that support it. Still, information integration remains a daunting task. There are many improvements needed: to the basic integration engines themselves, to the tools for understanding, standardizing and specifying what is needed, and to the theory behind them. These improvements will simplify certain aspects of the task, but they will not, by themselves, eliminate the many choices that must be made by a talented expert today. We therefore posed a challenge to the research community: can we move beyond the individual techniques for integration to a fundamental understanding of what integration is, and armed with that understanding, can we build a single integration engine that automatically uses the right techniques for the right situation? We hypothesized that the key to achieving this goal may lie in being able to represent and reason about the full set of desiderata for the integrated system.

There is plenty of work to do, and many areas we could not touch on in this paper. We focused on the problem of integration within the enterprise. In recent years, much research has focused on the exciting world of data outside the enterprise, on the worldwide web. Much of this work is applicable within the enterprise, though it typically requires significant adaptation to work effectively with the constraints and issues of that environment. More recently, research is emerging (again) on integration of personal information, for example the information on your laptop [52]. New challenges and techniques will doubtless be found in this environment as well.

Acknowledgments. I am grateful to my colleagues who have worked with me on integration and related issues over the years at the IBM Almaden Research Center (IBMers and visitors both), and to my many colleagues in IBM's Information

Management division who introduced me to the real-world challenges. In particular, I would like to thank Lee Scheffler and Mike Beckerle for many deep and stimulating discussions on our shared vision of a simpler world of integration, which Lee christened "the Big I". Finally, I'd like to thank Phokion Kolaitis, Ron Fagin, and Mike Beckerle for reading and improving earlier drafts of this work.

References

1. Jacob, K.J.: Betting on Brain Power. The Week. Feb 2, 2003. Available at http://www.the-week.com/23feb02/biz2.htm
2. IBM Business Consulting Services: Your Turn, The Global CEO Study 2004. Available from http://www.bitpipe.com/detail/RES/1129048329_469.html
3. Moore, C., Markham, R.: The Future of Content in the Enterprise. Forrester Report (2003)
4. Lenzerini, M.: Data Integration: A Theoretical Perspective. PODS (2002) 233-246
5. IEEE Data Eng. Bull. Special Issue on Structure Discovery, 26:3 (2003)
6. Barbará, D., DuMouchel W., Faloutsos, C., Haas, P. J., Hellerstein, J. M., Ioannidis, Y. E., Jagadish, H. V., Johnson, T., Ng, R. T., Poosala, V., Ross, K. A., Sevcik, K. C.: The New Jersey Data Reduction Report. IEEE Data Eng. Bull. 20:4 (1997) 3-45
7. Ilyas, I. F., Markl, V., Haas, P. J., Brown, P., Aboulnaga, A.: CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. SIGMOD (2004) 647-658
8. Doan, A., Ramakrishnan, R., Vaithyanathan, S.: Managing information extraction: state of the art and research directions. SIGMOD (2006) 799-800
9. Gravano, L., García-Molina, H., Tomasic, A.: GlOSS: text-source discovery over the Internet. ACM Transactions on Database Systems (TODS) 24:2 (1999) 229-264
10. Powell, A. L., French, J. C., Callan, J., Connell, M., Viles, C. L.: The impact of database selection on distributed searching. SIGIR (2000) 232-239
11. Hernández, M. A., Stolfo, S. J.: Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. Data Min. Knowl. Discov. 2:1 (1998) 9-37
12. Johnson, T., Dasu, T.: Exploratory Data Mining and Data Cleaning. John Wiley (2003)
13. Koudas, N., Sarawagi, S., Srivastava, D.: Record Linkage: Similarity Measures and Algorithms. SIGMOD (2006) 802-803
14. Lembo, D., Lenzerini, M., Rosati, R.: Source inconsistency and incompleteness in data integration. KRDB (2002)
15. Bertossi, L. E., Chomicki, J.: Query Answering in Inconsistent Databases. Logics for Emerging Applications of Databases (2003) 43-83
16. Naumann, F., Gertz, M., Madnick, S. E.: Proc. Information Quality (MIT IQ Conference), Sponsored by Lockheed Martin, MIT, Cambridge, MA, USA (2005)
17. IEEE Data Eng. Bull. Special Issue on Probabilistic Data Management, 29:1 (2006)
18. Miller, R. J., Haas, L. M., Hernández, M. A.: Schema Mapping as Query Discovery. VLDB (2000) 77-88
19. Rahm, E., Bernstein, P. A.: A survey of approaches to automatic schema matching. VLDB J. 10:4 (2001) 334-350
20. Johnston, W. M., Hanna, J. P., Millar, R. J. Advances in dataflow programming languages. ACM Comput. Surv. 36:1 (2004) 1-34
21. Rinderle, S., Reichert, M., Dadam, P.: Flexible Support of Team Processes by Adaptive Workflow Systems. Distributed and Parallel Databases 16:1 (2004) 91-116
22. Bernstein, P.A.: Applying Model Management to Classical Meta Data Problems. Proc. CIDR (2003) 209-220
23. Haas, L. M., Hernández, M. A., Ho, H., Popa, L., Roth, M.: Clio grows up: from research prototype to industrial tool. SIGMOD (2005) 805-810

24. Shu, N. C., Housel, B. C., Taylor, R. W., Ghosh, S. P., Lum, V. Y.: EXPRESS: A Data EXtraction, Processing, amd REStructuring System. ACM Trans. Database Syst. 2:2 (1977) 134-174
25. Breitbart, Y., Komondoor, R., Rastogi, R., Seshadri, S., Silberschatz, A.: Update Propagation Protocols For Replicated Databases. SIGMOD (1999) 97-108
26. Kemme, B., Alonso, G.: A new approach to developing and implementing eager database replication protocols. ACM Trans. Database Syst. 25:3(2000) 333-379
27. Dayal, U., Hwang, H.-Y.: View Definition and Generalization for Database Integration in a Multidatabase System. IEEE Trans. Software Eng. 10:6 (1984) 628-645
28. Lohman, G. M., Daniels, D., Haas, L. M., Kistler, R., Selinger, P. G.: Optimization of Nested Queries in a Distributed Relational Database. VLDB (1984) 403-415
29. Wiederhold, G. Mediators in the architecture of future information systems. IEEE Computer 25:3 (1992) 38-49
30. Papakonstantinou, Y., Gupta, A., Haas, L. M.: Capabilities-Based Query Rewriting in Mediator Systems. PDIS (1996) 170-181
31. Levy, A. Y., Rajaraman, A., Ordille, J. J.: Querying Heterogeneous Information Sources Using Source Descriptions. VLDB (1996) 251-262
32. Roth, M. T., Schwarz, P. M., Haas, L. M.: An Architecture for Transparent Access to Diverse Data Sources. In Dittrich, K. R., Geppert, A. (eds.): Component Database Systems. Morgan Kaufmann Publishers (2001)175-206
33. Haas, L. M., Kossmann, D., Wimmers, E. L., Yang, J.: Optimizing Queries Across Diverse Data Sources. VLDB (1997) 276-285
34. Fagin, R., Kolaitis, P. G., Miller, R. J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336:1 (2005) 89-124
35. Kolaitis, P. G.: Schema mappings, data exchange, and metadata management. PODS (2005) 61-75
36. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. 38:2 (2006)
37. Meng, W., Yu, C., Liu, K.: Building efficient and effective metasearch engines. ACM Comput. Surv. 34:1 (2002) 48-89
38. Chang, K. C.-C., Cho, J.: Accessing the web: from search to integration. SIGMOD (2006) 804-805
39. Leser, U., Naumann, F., Eckman, B. A.: Data Integration in the Life Sciences (DILS 2006). Lecture Notes in Computer Science, Vol. 4075. Springer-Verlag, Berlin Heidelberg New York (2006)
40. Buneman, P., Davidson, S. B., Hart, K., Overton, G. C., Wong, L.: A Data Transformation System for Biological Data Sources. VLDB (1995) 158-169
41. Blake, J. A., Bult, C. J.: Beyond the data deluge: Data integration and bio-ontologies. Journal of Biomedical Informatics 39:3 (2006) 314-320
42. http://www-306.ibm.com/software/data/integration/
43. http://www.informatica.com/
44. http://www-306.ibm.com/software/data/integration/info_server/overview.html
45. ISO/IEC 9075-14:2003 Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML). International Organization for Standardization (2003)
46. http://www-306.ibm.com/software/data/integration/db2ii/editions_content.html
47. http://www-306.ibm.com/software/data/integration/db2ii/editions_womnifind.html
48. Ferrucci, D., Lally, A.: UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering 10:3-4 Cambridge University Press, New York (2004) 327-348

49. Zilio, D. C., Rao, J., Lightstone, S., Lohman, G. M., Storm, A., Garcia-Arellano, C., Fadden, S.: DB2 Design Advisor: Integrated Automatic Physical Database Design. VLDB (2004) 1087-1097
50. Agrawal, S., Chaudhuri, S., Kollár, L., Marathe, A. P., Narasayya, V. R., Syamala, M.: Database Tuning Advisor for Microsoft SQL Server 2005. VLDB (2004) 1110-1121
51. Saracco, C., Englert S., Gebert, I.: Using DB2 Information Integrator for J2EE Development: A Cost/Benefit Analysis. May 2003. On IBM Developerworks, available at www.ibm.com/developerworks/db2/library/techarticle/0305saracco1/0305saracco1.html
52. Halevy, A. Y., Franklin, M. J., Maier, D.: Principles of dataspace systems. PODS (2006) 1-9