# Chapter 1

# Introduction

The invention of the Internet and the emergence of the World-Wide Web revolutionized people's access to digital data stored on electronic devices. Today, we take for granted the ability to specify a search query into a browser or smartphone, and tap into millions of documents and into databases of local businesses, recommendations and coupon offers. Similarly, we also assume we can order a computer configured just the way we want, and receive it within *days*, even if the computer requires assembling parts from dozens of companies scattered around the world. To provide such services, systems on the Internet must efficiently and accurately process and serve a significant amount of data. But unlike traditional data management applications, such as maintaining a corporation's payroll, the new services require the ability to *share* data among multiple applications and organizations, and to *integrate* data in a flexible and efficient fashion. This book covers the principles of *data integration*, a set of techniques that enable building systems geared for flexible sharing and integration of data across multiple autonomous data providers.

## 1.1 What is Data Integration?

We illustrate the need for data integration with two examples, one representing a typical scenario that may occur within a company and the other representing an important kind of search on the Web.

**Example 1.1:** Consider FullServe, a company that provides Internet access to homes, but also sells a few products that support the home computing infrastructure, such as modems, wireless routers, voice-over-IP phones and espresso machines. FullServe is a predominantly American company and decided recently to extend its reach to Europe. To do so, FullServe acquired a European company, EuroCard, that

**Employee Database**
`FullTimeEmp`: ssn, empId, firstName, middleName, lastName
`Hire`: empId, hireDate, recruiter
`TempEmployees`: ssn, hireStart, hireEnd, name, hourlyRate

**Resume Database**
`Interview`: interviewDate, name, recruiter, hireDecision, hireDate
`CV`: name, resume

**Training Database**
`Courses`: courseID, name, instructor
`Enrollments`: courseID, empID, date

**Services Database**
`Services`: packName, textDescription
`Customers`: name, id, zipCode, streetAdr, phone
`Contracts`: custID, packName, startDate

**Sales Database**
`Products`: prodName, prodId
`Sales`: prodName, customerName, address

**HelpLine Database**
`Calls`: date, agent, custId, text, action

Figure 1.1: Some of the databases a company like FullServe may have. For each database, we show some of the tables and for each table, some of its attributes. For example, the employee database has a table `FullTimeEmp` with attributes ssn, empId, firstName, middleName and lastName.

is mainly a credit card provider, but has recently started leveraging its customer base to enter the Internet market.[1]

The number of different databases scattered across a company like FullServe could easily reach 100. A very simplified version of FullServe's database collection is shown in Figure 1.1. The Human Resources Department has a database storing information about each employee, separating full-time and temporary employees. They also have a separate database of resumes of each of their employment candidates, including their current employees. The Training & Development Department has a database of the training courses (internal and external) that each employee went through. The Sales Department has a database of services and its current subscribers and a database of products and their customers. Finally, the Customer-care Department maintains a database of calls to their help-line center and some details on each such call.

Upon acquiring EuroCard, FullServe also inherited their databases, shown in Figure 1.2. EuroCard has some databases similar to those of FullServe, but given their

[1]The names of these companies and the combination of businesses in which they are involved are purely fictional, but not unimaginable in today's economy.

**Employee Database**
`Emp`: ID, firstNameMiddleInitial, lastName, salary
`Hire`: ID, hireDate, recruiter

**Resume Database**
`Interview`: ID, date, location, recruiter
`CV`: name, resume

**Credit Card Database**
`Customer`: CustID, cardNum, expiration, currentBalance
`CustDetail`: CustID, name, address

**HelpLine Database**
`Calls`: date, agent, custId, description, followup

Figure 1.2: Some of the databases of EuroCard. Note that EuroCard organizes its data quite differently than FullServe. For example, EuroCard does not distinguish between full-time and part-time employees. FullServe records the hire data of employees in the resume database and the employee database, while EuroCard only records the hire date in the employee database.

different location and business focus, there are some obvious differences.

There are many reasons why data resides in multiple databases throughout a company, rather that sitting in one neatly organized database. As in the case of FullServe and EuroCard, databases are acquired through mergers and acquisitions. When companies go through internal restructuring they don't always align their databases. For example, while the services and products divisions of FullServe are currently united, they probably didn't start out that way, and therefore the company has two separate databases. Second, most databases originate from a particular group in a company that has an information need at a particular point in time. When the database is created, its authors cannot anticipate all information needs of the company in the future, and how the data they are producing today may be used differently at some other point in time. For example, the training database at FullServe may have started as a small project by a few employees in the company to keep track of who attended certain training sessions in the company's early days. But as the company grew, and a training department was created, this database needed to be broadened quite a bit. As a result of these factors and others, large enterprises typically have dozens, if not hundreds of disparate databases.

Let us consider a few example queries that employees or managers in FullServe may want to pose, all of which need to span *multiple* databases.

- Now that FullServe is one large company, the Human Resources Department needs to be able to query for all of its employees, whether in the U.S or in Europe. Because of the acquisition, data about employees is stored in multiple databases: two databases (for employees and for temps) in the American side of the company and one on the European side.

- FullServe has a single customer support hot-line, where customers can call about any service or product they obtain from the company. It is crucial that when a customer representative is on the phone with a customer, he sees the entire set of services the customer is getting from FullServe, whether it be Internet service, credit card or products purchased. In particular, it is useful for the representative to know that the customer on the phone is a big spender on his credit card, even if he's calling about a problem with his Internet connection. Obtaining such a complete view of the customer involves obtaining data from at least three databases even in our simple scenario. The work of the representative is one component of a broader set of tasks known as *Customer Relationship Management*, for finding new customers and tracking their satisfaction and value over time.

- FullServe wants to build a web site to complement its telephone customer service line. On the web site, current and potential customers should be able to see all the products and services FullServe provides, and also select bundles of services. Hence, a customer must be able to see her current services and obtain data about the availability and pricing of any other service. Here too, we need to tap into multiple databases of the company.

- To take the previous example further, suppose FullServe partners with a set of other vendors to provide *branded* services. That is, you can get a credit card issued by your favorite sports team, but the credit card is actually served by FullServe. In this case, FullServe needs to provide a web service that will be accessed by other web sites (e.g., those of the sports teams) to provide a single login-point for customers. That web service needs to tap into the appropriate databases at FullServe.

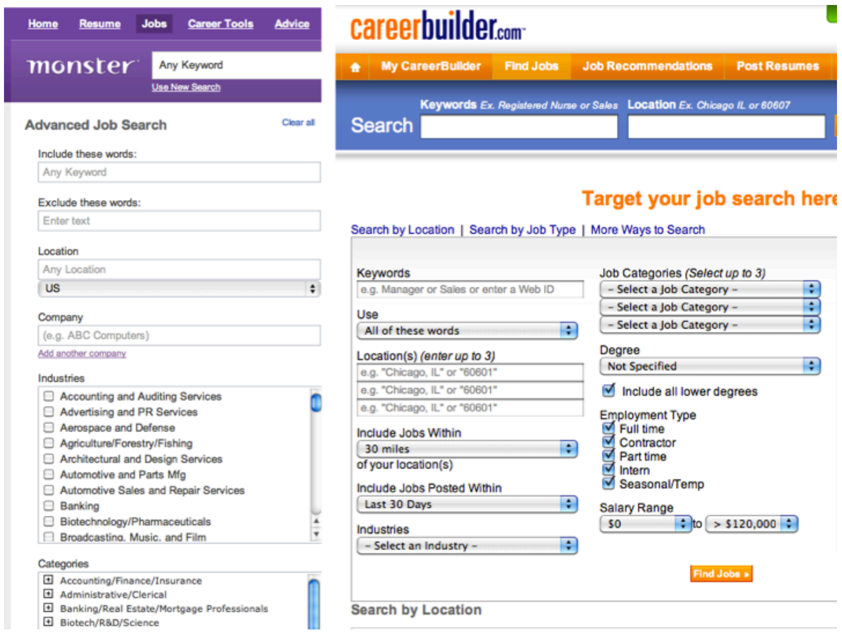
- Governments often change reporting or ethic laws concerning how companies can conduct business. To protect themselves from possible violations, FullServe may want to be proactive. As a first step, the company may want to be aware of employees who've worked at competing or partner companies prior to joining FullServe. Answering such a query would involve combining data from the HR database with the resume database. The additional difficulty here is that resumes tend to be unstructured text, and not nicely organized data.

- Combining data from multiple sources can offer opportunities for a company to obtain a competitive advantage and find opportunities for improvement. For example of the former, combining data from the help-desk database and the product database will help FullServe identify issues in their products and services early on. Discovering trends in the usage of different products can enable FullServe to be proactive about building and maintaining inventory levels. Going further, suppose we find that in a particular area of the country FullServe is receiving an unusual number of calls about malfunctions in their service. Taking a more careful look at this data may reveal that the services were installed by agents who have taken a particular training course, which was later found to be lacking. Finding such a pattern in the data requires combining data from the training database, help-desk database and subscriptions database, all residing in very different parts of the company.

**Example 1.2:** Consider a very different kind of example where data integration is needed. Suppose you are searching for a new job, and you'd like to take advantage of resources on the Web. There are thousands of web sites with databases of jobs (see Figure 1.3 for a few examples). In each of these sites, you'd typically see a form that requires you to fill out a few fields (e.g., job title, geographical location of employer, desired salary level), and will show you job postings that are relevant to your query. Unfortunately, each form asks for a slightly different set of attributes. While the HotJobs form on the top left of Figure 1.3 asks for keywords describing the job, location and job category, the one on the bottom asks for an employer and a job type. The USAJobs form on the right allows you to select a location and job category from a menu of options and lets you further specify your salary range.

Consequently, going to more then a handful of such web sites is tiresome, especially if you're doing this on a daily basis to keep up with new job postings. Ideally, you would like to go to a *single* web site to pose your query, and have that site integrate data from all relevant sites on the web.

More generally, the Web contains millions of databases, some of which are embedded in Web pages and others that can be accessed through web forms. They contain data in a plethora of domains, ranging from items typically found in classified ads and products to data about art, politics and public records. Leveraging this incredible collection of data raises several significant challenges. First, we face the challenge of schema heterogeneity, but on a much larger scale: millions of tables created by independent authors and in over 100 languages. Second, extracting the data is quite difficult. In the case of data residing behind forms (typically referred to as the *deep web* or *invisible web*) we need to either crawl through the forms intelligently or be able to pose well-formed queries at run time. For data that is embedded in web pages,



Any Keyword

Advanced Job Search Clear all

Include these words: Any Keyword

Exclude these words: Enter text

Location: Any Location; US

Company: (e.g. ABC Computers)
Add another company

Industries
- ☐ Accounting and Auditing Services
- ☐ Advertising and PR Services
- ☐ Aerospace and Defense
- ☐ Agriculture/Forestry/Fishing
- ☐ Architectural and Design Services
- ☐ Automotive and Parts Mfg
- ☐ Automotive Sales and Repair Services
- ☐ Banking
- ☐ Biotechnology/Pharmaceuticals
- ☐ Broadcasting, Music, and Film

Categories
- ⊞ Accounting/Finance/Insurance
- ⊞ Administrative/Clerical
- ⊞ Banking/Real Estate/Mortgage Professionals
- ⊞ Biotech/R&D/Science

Keywords *Ex. Registered Nurse or Sales* Location *Ex. Chicago IL or 60607*

Keywords: e.g. Manager or Sales or enter a Web ID

Use: All of these words

Location(s) *(enter up to 3)*: e.g. "Chicago, IL" or "60601"; e.g. "Chicago, IL" or "60601"; e.g. "Chicago, IL" or "60601"

Include Jobs Within: 30 miles of your location(s)

Include Jobs Posted Within: Last 30 Days

Industries: – Select an Industry –

Job Categories *(Select up to 3)*: – Select a Job Category –; – Select a Job Category –; – Select a Job Category –

Degree: Not Specified

✓ Include all lower degrees

Employment Type
- ✓ Full time
- ✓ Contractor
- ✓ Part time
- ✓ Intern
- ✓ Seasonal/Temp

Salary Range: $0 to > $120,000

Figure 1.3: Examples of different forms on the web for locating jobs. Note that the forms differ on the fields they request and formats they use.

extracting the tables from the surrounding text and determining its schema is challenging. Of course, data on the Web is often dirty, out of date and even contradictory. Hence, obtaining answers from these sources requires a different approach to ranking and data combination.

While the above two examples illustrate common data integration scenarios, it is important to emphasize the the problem of data integration is pervasive. Data integration is a key challenge for the advancement of science in fields such as biology, eco-systems, and water management, where groups of scientists are independently collecting data and trying to collaborate with each other. Data integration is a challenge for governments who want their different agencies to be better coordinated. And lastly, mash-ups are now a popular paradigm for visualizing information on the web, and underlying every mash-up is the need to integrate data from multiple disparate sources.

To summarize, the goal of a data integration system is to offer uniform access to a set of data autonomous and heterogeneous data sources. Let us expand on each of these:

- **Query**: the focus of most data integration systems is on querying disparate data sources. However, updating the sources is certainly of interest.

- **Number of sources:** data integration is already a challenge for a small number of sources (less than 10 and often even 2!), but the challenges are exacerbated when the number of sources grows. At the extreme, we would like to support web-scale data integration.

- **Heterogeneity:** a typical data integration scenario involves data sources that were developed independently of each other. As a consequence, the data sources run on different systems, some of them are databases but others may be content management systems or simply files residing in a directory. The sources will have different schemata and references to objects, even when they model the same domains. Some sources may be completely structured (e.g., relational databases), while others may be unstructured or semi-structured (e.g., XML or content management systems).

- **Autonomy:** the sources do not necessarily belong to a single administrative entity, and even when they do, they may be run by different sub-organizations. Hence, we cannot assume that we have full access to the data in a source or that we can access the data whenever we want and considerable care needs to be given to respecting the privacy of the data when appropriate.

## 1.2 Why is It Hard?

In order to approach the problem of data integration effectively, it is important to first examine the reasons for which it is hard. Later in the book we cover these challenges in more detail, but here we give a high-level description of the challenges. We classify the reasons into three different classes: systems reasons, logical reasons and social reasons. We examine each one separately.

### 1.2.1 Systems reasons

The systems challenges that occur in data integration are quite obvious and appear very early in the process of trying to build an integration application. Fundamentally, the challenge is to enable different systems to talk seamlessly to each other. Even assuming that all systems are running on the same hardware platform and all the sources are relational database systems supporting SQL standard and ODBC/JDBC, the problem is not easy. For example, while SQL is a standard query language for relational databases, there are some differences in the way different vendors implement it, and these differences need to be reconciled.

Executing queries over multiple systems efficiently is even more challenging. Query processing in distributed databases (i.e., databases where the data is partitioned into multiple nodes) is already a hard problem. The saving grace in distributed databases is that the data is *a priori* distributed to the different nodes by one entity and in an organized and known fashion. In data integration we are faced with a pre-existing collection of sources, and the organization of the data in the sources is much more complex and not necessarily well known. Furthermore, the capabilities of each of the data sources in terms of the query processing powers they have can be very different. For example, while one source may be a full SQL engine and therefore may be able to accept very complex queries, another source may be a web form and therefore only accept a very small number of query templates.

### 1.2.2 Logical reasons

The second set of challenges has to do with the way data is logically organized in the data sources. For the most part, structured data sources are organized according to a schema. In the common case of relational databases, the schema specifies a set of tables, and for each table a set of attributes with their associated data types. In other data models the schema is specified by tags, classes and properties.

Human nature is such that if two people are given *exactly* the same requirements for a database application, they will design very different schemata.[2] Hence, when data comes from multiple sources, it is very likely to look very different.

We can see several differences when we compare the schemata of the databases of FullServe and EuroCard:

- EuroCard models temporary employees and full-time employees in the same database table while FullServe maintains two separate databases. This may be because FullServe uses an outside agency to contract and manage its temporary employees.

- Even when modeling employees, FullServe and EuroCard do so differently and cover slightly different attributes. For example, EuroCard uses IDs (corresponding to national ID cards) to identify employees, while FullServe records the social-security number but also assigns an employee ID (since social-security numbers are not always reliable for IDs). FullServe records the hireDecision and

[2]This aspect of human nature is well known by database professors when they attempt to detect cheating.

hireDate attributes in their resume database, but EuroCard simply assumes these attributes can be obtained by querying the employee database with the appropriate ID. On the other hand, EuroCard records *where* each interview was conducted, while FullServe does not.

- Even when modeling the exact same attribute, FullServe and EuroCard may use different attribute names. For example, the help-line database of FullServe uses the attributes text and action to record the same as the attributes description and followup in EuroCard's database.

Finally, the representation of the data can be significantly different as well. For example, in FullServe's employee database, the company records the first, middle and last name of every employee in a separate field. In the Training database, it only records the full name of the employee in a single field, and as a result, names often occur in various formats, such as (First, Last), or (Last, First Initial), etc. Consequently, it may be hard to match records from these two databases. The same problem occurs between the Sales and Services databases. While the Sales database only records a field for the name and a field for the address, the Services database records much finer-grained information about each subscriber. Of course, the units of measure will differ as well: FullServe uses American dollars for prices, while EuroCard uses Euros. Since the exchange rate between the currencies constantly changes, the correspondence between the values cannot be set apriori.

Data from multiple sources can only be integrated if we bridge this so called *semantic heterogeneity*. In fact, semantic heterogeneity turns out to be the main bottleneck in data integration.

### 1.2.3 Social and administrative reasons

The last set of reasons we mention are not so technical as the previous ones, but are often as hard and can easily be the reason a data integration project fails. The first challenge may be to *find* a particular set of data in the first place. For example, it may turn out that EuroCard did not save the resumes of their employees electronically, and a special effort is required to locate and scan them all.

Even when we know where all the data is, owners of the data may not want to cooperate with an integration effort. When owners come from different companies or universities, there are some obvious reasons for not wanting to share, but even within an enterprise, owners are often reluctant. In some cases, the reason is that their data is part of a carefully tuned and mission-critical function of the enterprise. Allowing additional queries from the data integration system can put a prohibitively heavy load on their system. In other cases, the reason is that some data can only be viewed by particular people within the enterprise, and the data owners are worried (for a good reason!) that the data integration system will not be able to enforce these restrictions. Finally, in some cases people create *data fiefdoms* – here, access to the data means more power within the organization. For example, the head of the Sales Department may not want to share the data on sales representatives' work as it may reveal some internal problems.

It is worth noting that in a few circumstances — for instance, those involving medical records or law enforcement — there may be legitimate legal reasons why a data owner cannot share data. Because such situations arise, the problem of anonymizing data remains a hot topic in computer science research.

While we cannot expect to solve the social problems with technology alone, technology can help data owners capitalize on some of the benefits of data integration, thereby providing additional incentive to participate. For example, an important benefit of participating in a data integration project is that one's data can be reached by many more people and have broader impact (e.g., if the data is included in relevant search results by a web search engine). As another example, a data integration system can be designed so the attribution of the data is always clear (even when results are assembled from multiple sources), therefore ensuring that appropriate credit is given to the data owners.

### 1.2.4 Setting expectations

Data integration is a hard problem, and some say it will never be completely solved (thereby guaranteeing a steady stream of readers for this book). Before we can discuss solutions to different aspects of data integration, it is important to set the expectations appropriately.

In the ideal world, we would like to provide a data integration system access to a set of data sources, and have the system automatically configure itself so it can correctly and efficiently answer queries that span multiple sources. Since this ideal goal is unlikely to be achieved, we focus on two more modest goals. Our first goal is to build tools that *reduce the effort* required to integrate a set of data sources. For example, these tools should make it easy to add a new data source, relate its schema to others, and automatically tune the data integration system for better performance.

Our second goal is to improve the ability of the system to answer queries in uncertain environments. Clearly, if the data integration system must always return correct and complete answers, then uncertainty cannot be tolerated at all. But in applications where data integration facilitates exploratory efforts (e.g., on the Web), the system should be able to answer queries under uncertainty. For example, when we look for jobs on the web, it's ok to find 29 out of 30 relevant sources, or to return answers that don't completely satisfy the user query.

In some sense, user effort and accuracy are competing goals in setting up a data integration application. The more time we spend, the more accurate our system is going to be. Hence, in cases where we are willing to trade off lower accuracy for decreased setup time and labor, our goal can be stated as follows: *reduce the user-effort needed to obtain high-quality answers from a data integration system.*

In the next chapter we describe a reference architecture for a data integration system. Then, in the following chapter we discuss how practitioners *not* using this architecture do their work — and why the data integration architecture would be useful to them. The rest of the book then considers each of the components of the architecture in detail, and moves on to describe a series of advanced data models and alternative architectures.

## Exercises

1. What distinguishes *data integration* from standard keyword search, as with a Web search engine?

2. How might data integration as described in this chapter help reduce concerns about release of a particular source's private information, e.g., a person's criminal record?

3. Many people advocate the creation of community standards to "solve" the data integration problem. Which of the "hard" aspects of the problem does this address?

4. Data integration across multiple organizations is harder than integration within an organization. What are the additional difficulties that are introduced when we integrate across organizations?

## Bibliographic Notes

Data integration has been a subject of research since the early 1980's, beginning with the Multi-Base System [285]. Since then, there has been considerable research and commercial development in the area. The paper on *mediators* [438] helped fuel the field and ultimately obtain more government funding for research in the area. The emergence of the World-Wide Web and the large number of database available online lead to considering large-scale data integration [179]. Özsu and Valduriez provide a book length treatment of distributed databases [369].

As of the late 1990's, data integration is a well established field in the enterprise market typically referred to as *Enterprise Information Integration* (EII) [223]. There is also a significant market for data integration on the Web. In particular, numerous companies created domain-specific interfaces to a multitude of data sources in domains such as jobs, travel and classifieds. These sites integrate data from hundreds to thousands of sources.