

Chapter 12: A Biased Take on a Moving Target: Data Integration

by Michael Stonebraker

I will start this treatise with a history of two major themes in data integration. In my opinion, the topic began with the major retailers in the 1990s consolidating their sales data into a data warehouse. To do this they needed to extract data from in-store sales systems, transform it into a predefined common representation (think of this as a global schema), and then load it into a data warehouse. This data warehouse kept historical sales data for a couple of years and was used by the buyers in the organization to rotate stock. In other words, a buyer would figure out that pet rocks are “out” and barbie dolls are “in.” Hence, he would tie up the barbie doll factory with a big order and move the pet rocks up front and discount them to get rid of them. A typical retail data warehouse paid for itself within a year through better buying and stock rotation decisions. In the late 1990s and early 2000’s there was a giant “pile on” as essentially all enterprises followed the lead of retailers and organized their customer-facing data into a data warehouse.

A new industry was spawned to support the loading of data warehouses, called extract, transform, and load (ETL) systems. The basic methodology was:

- a) Construct a global schema in advance.
- b) Send a programmer out to the owner of each data source and have him figure out how to do the extraction. Historically, writing such “connectors” was a lot of work because of arcane formats. Hopefully, this will become less problematic in the future with more open source and standardized formats.
- c) Have him write transformations, often in a scripting language, and any necessary cleaning routines
- d) Have him write a script to load the data into a data warehouse

It became apparent that this methodology scales to perhaps a dozen data sources, because of three big issues:

1. A global schema is needed up front. About this time, there was a push in many enterprises to write an enterprise-wide schema for all company objects. A team was charged with doing this and would work on it for a couple of years. At the end of this time, their result was two years out of date, and was declared a failure. Hence, an upfront global schema is incredibly difficult to construct for a broad domain. This limits the plausible scope of data warehouses.
2. Too much manual labor. A skilled programmer is required to perform most of the steps in the methodology for each data source.

3. Data integration and cleaning is fundamentally difficult. The typical data warehouse project in the 1990’s was a factor of two over budget and a factor of two late. The problem was that planners underestimated the difficulty of the data integration challenge. There are two big issues. First, data is dirty. A rule of thumb is that 10% of your data is incorrect. This results from using nicknames for people or products, having stale addresses for suppliers, having incorrect ages for people, etc. The second is that deduplication is hard. One has to decide if Mike Stonebraker and M.R. Stonebraker are the same entity or different ones. Equally challenging is two restaurants at the same address. They might be in a food court, one might have replaced the other in a stand-alone location or this might be a data error. It is expensive to figure out ground truth in such situations.

In spite of these issues, data warehouses have been a huge success for customer facing data, and are in use by most major enterprises. In this use case, the pain of assembling composite data is justified by the better decision making that results. I hardly ever hear enterprises complaining about the operational cost of their data warehouse. What I hear instead is an incessant desire by business analysts for more data sources, whether these be public data off the web or other enterprise data. For example, the average large enterprise has about 5000 operational data stores, and only a few are in the data warehouse.

As a quick example, I visited a major beer manufacturer a while ago. He had a typical data warehouse of sales of his products by brand, by distributor, by time period, etc. I told the analysts that El Nino was widely predicted to occur that winter and it would be wetter than normal on the west coast and warmer than normal in the Northeast. I then asked if beer sales are correlated to temperature or precipitation. They replied “I wish we could answer that question, but weather data is not in our warehouse”. Supporting data source scalability is very difficult using ETL technology.

Fast forward to the 2000’s, and the new buzzword is *master data management (MDM)*. The idea behind MDM is to standardize the enterprise representation of important entities such as customers, employees, sales, purchases, suppliers, etc. Then carefully curate a master data set for each entity type and get everyone in the enterprise to use this master. This sometimes goes by the mantra “golden records”. In effect, the former ETL vendors are now selling MDM, as a broader scope offering. In my opinion, MDM is way over-

hyped.

Let me start with “Who can be against standards?” Certainly not me. However, MDM has the following problems, which I will illustrate by vignette.

When I worked for Informix 15 years ago, the new CEO asked the Human Resources VP at an early staff meeting “How many employees do we have?” She returned the next week with the answer “I don’t know and there is no way to find out?” Informix operated in 58 countries, each with its own labor laws, definition of an employee, etc. There was no golden record for employees. Hence, the only way to answer the CEO’s question would be to perform data integration on these 58 data sources to resolve the semantic issues. Getting 58 country managers to agree to do this would be challenging, made more difficult by the fact that Informix did not even own all the organizations involved. The new CEO quickly realized the futility of this exercise.

So why would a company allow this situation to occur? The answer is simple: business agility. Informix set up country operations on a regular basis, and wanted the sales team up and running quickly. Inevitably they would hire a country manager and tell him to “make it happen”. Sometimes it was a distributor or other independent entity. If they had said “here are the MDM golden records you need to conform to”, then the country manager or distributor would spend months trying to reconcile his needs with the MDM system in place. In other words, MDM is the opposite of business agility. Obviously every enterprise needs to strike a balance between standards and agility.

A second vignette concerns a large manufacturing enterprise. They are decentralized into business units for business agility reasons. Each business unit has its own purchasing system to specify the terms and conditions under which the business unit interacts with its suppliers. There are some 300 of these systems. There is an obvious return on investment to consolidate these systems. After all it is less code to maintain and the enterprise can presumably get better-consolidated terms than each business unit can individually. So why are there so many purchasing systems? Acquisitions. This enterprise grew largely by acquisition. Each acquisition became a new business unit, and came with its own data systems, contracts in place, etc. It is often simply not feasible to merge all these data systems into the parent’s IT infrastructure. In summary, acquisitions screw up MDM.

So what is entailed in data integration (or data curation)? It is the following steps:

1. *Ingest*. A data source must be located and captured. This requires parsing whatever data structure is used for storage.

2. *Transform*. For example, Euros to dollars or airport code to city name.
3. *Clean*. Data errors must be found and rectified.
4. *Schema integration*. Your wages is my salary.
5. *Consolidate (deduplication)*. Mike Stonebraker and M.R. Stonebraker must be consolidated into a single record.

The ETL vendors do this at high cost and with low scalability. The MDM vendors have a similar profile. So there is a big unmet need. Data curation at scale is the “800 pound gorilla in the corner.” So what are the research challenges in this area?

Let’s go through the steps one by one.

Ingest is simply a matter of parsing data sources. Others have written such “connectors”, and they are generally expensive to construct. An interesting challenge would be to semi-automatically generate connectors.

Data transformations have also been extensively researched, mostly in the last decade or so. Scripting/visualization facilities to specify transforms have been studied in [8, 3]. Data Wrangler [6] appears to be the state of the art in this area, and the interested reader is encouraged to take a look. In addition, there are a bunch of commercial offerings that offer transformation services for a fee (e.g. address to (lat, long) or company to canonical company representation). In addition, work on finding transformations of interest from the public web is reported in [1].

Data cleaning has been studied using a variety of techniques. [2] applied functional dependencies to discover erroneous data and suggest automatic repairs. Outlier detection (which may correspond to errors) has been studied in many contexts [5]. [12, 11] are query systems to discover interesting patterns in the data. Such patterns may correspond to errors. [10] have studied the utilization of crowd sourcing and expert sourcing to correct errors, once they have been identified. Lastly, there are a variety of commercial services that will clean common domains, such as addresses of persons and dates. In my opinion, data cleaning research MUST utilize real-world data. Finding faults that have been injected into other-wise clean data just is not believable. Unfortunately, real world “dirty” data is quite hard to come by.

Schema matching has been extensively worked on for at least 20 years. The interested reader should consult [7, 4, 9] for the state of the art in this area.

Entity consolidation is a problem of finding records in a high dimensional space (all of the attributes about an entity – typically 25 or more) that are close together. Effectively this

is a clustering problem in 25 space. This is an N^2 problem that will have a very long running time at scale. Hence, approximate algorithms are clearly the way to proceed here. A survey of techniques appears in [5].

In my opinion, the real problem is an end-to-end system. Data curation entails all of these steps, which must be seamlessly integrated, and enterprise-wide systems must perform curation at scale. An interesting end-to-end approach that appears to scale well is the Data Tamer system [10]. On the other hand, data curation problems also occur at the department level, where an individual contributor wants to integrate a handful of data sources, and the Data Wrangler system noted above appears to be an interesting approach. There are commercial companies supporting both of these systems, so regular enhancements should be forthcoming.

Hopefully, the next edition of the Red Book will have a collection of seminal papers in this area to replace this (self-serving) call to action. In my opinion, this is one of the most important topics that enterprises are dealing with. My one caution is “the rubber has to meet the road”. If you want to work in this area, you have got to try your ideas on real world enterprise data. Constructing artificial data, injecting faults into it, and then finding these faults is simply not believable. If you have ideas in this area, I would recommend building an end-to-end system. In this way, you make sure that you are solving an important problem, rather than just a “point problem” which real world users may or may not be interested in.

Commentary: Joe Hellerstein

6 December 2015

I agree with Mike’s assessment here in general, but wanted to add my perspective on the space, relating to the “department level” problem he mentions in passing.

Based on experience with users across a wide range of organizations, we’ve seen that data transformation is increasingly a user-centric task, and depends critically upon the user experience: the interfaces and languages for interactively assessing and manipulating data.

In many of today’s settings, the right outcome from data transformation depends heavily on context. To understand if data is dirty, you have to know what it is “supposed” to look like. To transform data for use, you need to understand what it is being used for. Even

in a single organization, the context of how data is going to be used and what it needs to be like varies across people and across time. Add this to Mike’s critique of the idea of a “golden master”—it simply doesn’t make sense for many modern use cases, especially in analytical contexts.

Instead, you need to design tools for the people who best understand the data and the use case, and enable them to do their own data profiling and transformation in an agile, exploratory manner. Computer scientists tend to design for technical users—IT professionals and data scientists. But in most organizations, the users who understand the data and the context are closer to the “business” than the IT department. They are often less technically skilled as well. Rather than reach for traditional ETL tools, they tend to fall back on manipulating data in spreadsheets and desktop database packages, neither of which are well suited for assessing data quality or doing bulk transformation. For large datasets they “code in Microsoft Word”: they describe their desired workflow in a natural language spec, wait for IT to get around to writing the code for them, and then when they get the results they typically realize they don’t quite work. At which point their need for the data has often changed anyhow. No surprise that people often assert that 50-80% of their time is spent in “preparing the data.” (As a footnote, in my experience modern “data scientists” tend to wrangle data via ad-hoc scripts in Python, R or SAS DataStep, and are shockingly lax about code quality and revision control for these scripts. As a result they’re often worse off over time than the old-school ETL users!)

Business users reach for graphical tools for good reason: they want to understand the data as it is being transformed, and assess whether it is getting closer to a form that’s useful for their business problem. As a result, the unattended algorithms from the database research literature have done too little to address the key issues in this space. I believe the most relevant solutions will be based on interfaces that enable people to understand the state of their data intuitively, and collaborate with algorithms to get the data better purposed for use.

This presents a significant opportunity for innovation. Data transformation is a perfect Petri Dish for exploring the general topic of visualizing and interacting with data. This is an area where ideas from Databases, HCI and Machine Learning can be brought together, to create interactive collaborations between algorithms and people that solve practical, context-specific problems with data. Backing this up we need interactive data systems that can do things like provide instantaneous data profiles (various aggregates) over the results of ad-hoc transformation steps, and speculate ahead of users in real time to suggest multiple alternative transformations that could be useful.¹ Some of the topics from the Interactive Analytics chapter are relevant here, particularly for big data sets. I’ve been happy to see more work on visualization and interaction in the database community in recent years; this is a great application area for that work.

¹Heer, J., Hellerstein, J.M. and Kandel, S. “Predictive Interaction for Data Transformation.” CIDR 2015.

References

- [1] Z. Abedjan, J. Morcos, M. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani. Dataxformer: Leveraging the web for semantic transformations. In *CIDR*, 2015.
- [2] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *ICDE*, 2013.
- [3] T. Dohzen, M. Pamuk, S.-W. Seong, J. Hammer, and M. Stonebraker. Data integration through transform reuse in the morpheus project. In *SIGMOD*, 2006.
- [4] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *VLDB*, 1997.
- [5] I. F. Ilyas and X. Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2012.
- [6] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *CHI*, 2011.
- [7] R. J. Miller, M. A. Hernández, L. M. Haas, L.-L. Yan, C. H. Ho, R. Fagin, and L. Popa. The clio project: managing heterogeneity. *SIGMOD Record*, 30(1):78–83, 2001.
- [8] V. Raman and J. M. Hellerstein. Potter’s wheel: An interactive data cleaning system. In *VLDB*, 2001.
- [9] M. T. Roth and P. M. Schwarz. Don’t scrap it, wrap it! a wrapper architecture for legacy data sources. In *VLDB*, 1997.
- [10] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.
- [11] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: automatically generating query visualizations. In *VLDB*, 2014.
- [12] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. In *VLDB*, 2013.