

INFORMATION INTEGRATION NEEDS A HISTORY LESSON

Peter Buneman
University of Edinburgh

Draft of September 9, 2006

For 25 years the problem of information (a.k.a. data) integration has been discussed and worked on, especially by the database research community. Despite their efforts, the fruits of the research have been slow to come to market. Why is this? I suggest three reasons:

1. The tools and principles created by database research such as schema integration techniques, advanced query languages and treatment of missing data, are too sophisticated for mass consumption.
2. The real problem is not really a database problem, but is the hard graft required of domain experts in: understanding the schemas; reconciling terminologies (ontologies), object identifiers or co-ordinate systems; and in cleaning the data.
3. By taking a narrow view of data integration, we have made the problem more difficult and ignored other related and important issues.

In short there is no silver bullet for data integration, but there are a number of things we, as database researchers, can do to help with the problem. In this note I want to explore point 3 and suggest that a broader view of data integration in which we pay attention to the *process* by which databases are constructed may bear fruit.

The traditional view of database integration is that one is given two or more databases A, B, \dots and one wants to construct a new database G which represents the combined information in A, B, \dots . This involves first finding a schema for G , and then creating an instance of G . Because the data models used to construct A, B, \dots may differ, this involves work on both models and query languages. Also, data integration necessarily involves a treatment of inconsistency and incompleteness, and these qualities need to be represented in G , even if they were not in explicit A, B, \dots .

Data Quality. In practice, the problem is not so simple. The end users (the people who want G) typically know quite a bit about A, B, \dots and want to know, when they look at G , something about the reliability or accuracy of the data they see. In particular they often want to know *where* the data has come from. Even though they want a simple unified view of the data, the trust they place in it is determined by its *provenance* (Yes, you knew I would utter the “P-word” before long.) This means that our data integration tools should convey provenance, but what does this mean? A simple answer is that we should simply keep the query Q that created G . This is seldom satisfactory for a number of reasons. First, keeping Q and G is not enough. There are cases in which one also needs A, B, \dots . Second the intended provenance may not be captured by Q as the Q may be a transformed version of some other query, which produces the same output but creates different provenance. Third, it is quite unlikely our end-user will understand Q , especially when it has been generated automatically by sophisticated data integration software. What we need to find are simple descriptions of provenance with which data elements in G are annotated.

Thus, data integration should convey provenance. This is not only true for the sophisticated tools being developed in database research, but also for *curated* databases, which are constructed with a great deal of manual effort in annotating, classifying and correcting existing data. Molecular biology provides abundant examples of such databases. Much of the data in these databases has been copied from other sources, so curated databases also represent a form of data integration, but they are not views. They represent a great deal of costly added information, and they cannot be regenerated on the fly, from other databases. Once again, provenance is lost in the generation of curated databases, partly because the databases are not

designed to record provenance and partly because the provenance is not captured by automatically (as it could be) in the simple act of manually copying data from one database to another.

Provenance as an aid in data integration. So far, I have argued that provenance, as an important part of data quality, should be recorded in data integration. What I want to suggest now is that it could actually be of help in – and might even simplify – the process of integration. Here are some examples:

- Some notion of trust in the sources could be built into data integration tools, e.g.: only use source *B* if source *A* does not contain the required information.
- Data integration necessarily involves a treatment of incomplete information, which is usually presented as null values. It would be useful if such null values could be annotated with their provenance – the source that did not deliver them. If the previous sentence doesn't make sense, consider the null values generated in an outer join.
- Data integration necessarily involves a treatment of *inconsistent* information. Suppose sources *A* and *B* disagree on some value (e.g. a key constraint is violated in a union of two tables from these sources.) Suppose, however, that they both convey provenance information and they tell us where the offending items have been copied from. As suggested above, if they have been copied from different places, some preference mechanism might operate; if they have been copied from the *same* place, in all likelihood one of the copies is stale – it has been copied from an old version.

Summary. Provenance information should be conveyed and recorded in data integration as an essential part of data quality. Moreover, keeping record of provenance may greatly facilitate the process of data integration.