Reset version of a manuscript submitted November 15, 1979 to *ACM SIGMOD Record*. Written November, 1979; reset March, 1996. Freely available via `http://www.medg.lcs.mit.edu/doyle`. Author's current address: Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.

Historical Annotations and Humble Databases

Jon Doyle
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
November 15, 1979

As databases and systems of interrelated databases become more common and more complex, our society comes to rely crucially on their accuracy and intelligibility. Stories abound of false information irrevocably ruining someone's credit ratings, employment records, or worse. In trying to deal with such tragedies, society finds that computer systems are designed with the view that they are monolithic, infallible sources of information. This leads to great disrespect and growing resentment of these large information systems. If we are to justify our reliance on these systems while avoiding society's censure, we can take either one of two paths. We might make databases responsible for their contents, but this is impossible with current technology. More practically, with current techniques we can make databases explicitly defer all responsibility to humans. What we can do is to have database systems keep historical information about their inputs and about the computations they perform. This historical information can then be used to construct explanations or justifications of each database entry so that errors can be traced to bad inputs, to faulty programs, or to other databases in a distributed system. In this way, the computer can be prepared with the fact of its own fallibility and irresponsibility, and can help track down its own problems and those of its users. While this may not render intelligible the enormous systems of programs involved, at least their effects will have been isolated to some extent. This may add considerable cost in implementing database systems, but we suggest that in society's view it is too expensive *not* to provide these facilities. With this motivation, we briefly outline methods for constructing such "humble" databases.

The basic idea is simple. The main objective is that no information begin with the database; all information can be traced to some identifiable external source. To achieve this objective, two steps are required. First, each new piece of information entered into the system is annotated with its source. For example, the source specification might mention the time and location of entry, the person entering the data, the physical form number and the location of the form in physical files, if any, and perhaps other relevant information. In highly influential databases in which the security of the system is very important, unforgeable identification codes for the person involved might be used. Second, whenever computations are performed on the database, again each record in the resulting database is annotated with its source, namely the records used in its computation, the time of the computation, the version of the program and compiler used, the computer used, and other relevant information. For example, when two files are merged, each record in the resulting file points back to some record in one of the merged files. When records in one file are used to update those in another (or in itself), the updated records similarly point to the origin records, time, program, etc. (In fact, these annotations might well exist for each field of the records, to show just what other fields of the records were involved in computing the annotated field. This would help to unravel the actions of complex programs as well.)

The cost of keeping this historical information can be held to a minimum in many ways. It may be possible to avoid keeping complete historical annotations of each record of each file involved, for example in merging two files, when the only information necessary to reconstruct the source of a particular record are the two files merged to produce the resulting file. The complete system calls for keeping each version of a database for future reference, but if this is too expensive, one can maintain multiple versions of each record, corresponding to successive updates of the record, and mark one version as the current version. If bidirectional pointers are used to link records with their sources, the consequential effects of updating a record can be computed by tracing forward through the pointers from sources to consequences starting with the superceded record version. (These techniques are described in [1], but have only been explored there in small databases.) If the computer system running the database is careful to permanently archive all versions of the programs and programming systems in use and the durations of their use, then simple version numbers may suffice to mention which program performed some computation. If the computer system does not archive the program versions, it may well be worthwhile for the database system to itself include a copy of the program producing a new database as part of the database.

We have had considerable practical experience with these techniques in current artificial intelligence programs, where they are regularly used both to explain and to automatically update databases. (See [1] for a discussion.) Many AI researchers who have not yet adopted these techniques are still suspicious of the memory requirements of such systems, but those who do use them will never give them up. The explanatory capabilities of these techniques make possible ways of using and debugging a complex system that just could not be done before. The difference in the ease of use and intelligibility of these programs seems as great as when high-level languages superceded assembly language.

To stress our main point again, it may be impossible for systems without historical annotations to do many of the things that we want our databases or computer systems to do, namely to defer responsibility to humans so that their contents may be explained and corrected. The larger and more important our databases become, the more important such humility becomes. The fairness and effectiveness of our databases are at stake, and if society is to trust their accuracy and usefulness, they must be able to trace their contents to responsible sources.

Acknowledgments: I thank Ronni Rosenberg for several suggestions and for valuable criticisms of drafts of this paper. Statements of Gerald J. Sussman and Joseph Weizenbaum provided motivation. I thank the Fannie and John Hertz Foundation for their support via a graduate fellowship.

Reference

1. Doyle, J., A truth maintenance system, *Artificial Intelligence* **12**, in the press, 1979.