****

Statement

****

A Grand Challenge: Integrating Information

Jeffrey D. Ullman
2/19/02

People have been talking about information integration --- warehousing, mediators, middleware --- for over a decade. Likewise, process integration is in a similar state. A few companies, e.g., SAP, provide backends that try to encompass an enterprise, and there are other companies, e.g., e.piphany, that try to make it by writing their own integrators between their software and every likely information system their clients might be using. However, despite "rumors" at DARPA and elsewhere that the information-integration problem had been solved, the fact is it has *not* been solved, and it needs to be solved.

* Information integration is important --- it impacts strongly on our prosperity and security.

* Information integration will not be a reality without advances in a number of areas of computer science.

Here are two examples of the sorts of things we cannot do and should be able to do:

1. If HP really buys Compaq, they are going to have to integrate hundreds of databases. Logically, they should mesh, e.g., an HP employee should be the same as a Compaq employee. Yet there will be subtle but important differences between the two companies, and also among "employee" databases of the same company. Is a consultant an employee? The benefits department thinks not, but health-and-safely better think "yes." Is a retiree an employee? The reverse probably is true.

2. The 9-11 event was preceded by four guys not affiliated with an airline enrolling in four different flight schools to fly the same kind of plane, and each had connections to Al Qaeda, as could be traced by the movement of funds and perhaps other information. We can't easily integrate the schools' enrollment databases, let alone connect it to the banking system.

Technology Challenges

Setting aside for the moment the obvious issues of privacy that point (2) raises, solutions bring in technology from a number of areas of computer science:

A. Data Modeling.
There are lots of models, none particularly well suited, e.g., relational or object-oriented models. Recently, "semistructured models" have evolved into XML, which is has attracted a lot of interest as a way to store data that can be shared. However. about 0.001% of all data is today in XML, and it is still open what data-integration needs it can fill.

B. Programming languages.
In a sense, integration is achieved by writing code that converts information from one source into that of another. The conversion has to be written in some language by someone who understands the domain. There have been some experiments with logical languages, and tree-based languages that work on XML data are exciting. All these languages are radically different from the C++ paradigm.

C. Intelligent Systems.
Humans aren't terribly good at explaining, or writing down in some specialized language, the relationship between two sources of similar data. There are opportunities for software that looks at the content and, using any of a number of interesting clues, can figure out or suggest what the relationship is.

D. System Architecture.
As a simple example, would HP and Compaq be better off (a) translating the employee information of Compaq to the terms of HP's databases (b) do exactly the opposite (c) create a warehouse in which both databases are translated and copied (d) build a querying system that has its own data model and that translates queries into requests to the HP and/or Compaq DB's as needed (e) other?

E. Cryptography and Access Control.
Again, without wanting to address the issue of who decides what are legitimate uses of integrated data, an ambitious system, e.g., one that integrates medical records and information of all kinds, or a system that could be used to mine for unusual behavior by terrorists, cannot be allowed to exist without strong guarantees against misuse. How do we make sure that we know who has obtained information, and what they have

obtained? How do we deal with discoveries that should not be revealed at all, e.g., discovering that a prospective employee has a prediliction toward a certain form of cancer, while still allowing the discovery of patients that would, for reasons of physiology, respond well to a new drug? How do we distinguish two terrorists meeting to plan a bombing from two people having an "affair," so the system exposes the first and not the second.

The Real Challenge

The technical questions above are themselves important and useful, but what I'm really thinking about is how we respond to the terrorist threat by *building* an information system big enough and all-encompassing enough that machines can recognize the interesting coincidences, like the four guys in flight school, while not innundating analysts with similar sounding coincidences that are meaningless. I have less in the way of ideas here, but I doubt we'll learn anything much until we try to build. It's another ARPANET --- we'll have to start small and see where it leads.

****

Vita

****

Jeff Ullman is the Stanford W. Ascherman Professor of Engineering in the Department of Computer Science at Stanford. He received the B.S. degree from Columbia University in 1963 and the PhD from Princeton in 1966. Prior to his appointment at Stanford in 1979, he was a member of the technical staff of Bell Laboratories from 1966-1969, and on the faculty of Princeton University between 1969 and 1979. From 1990-1994, he was chair of the Stanford Computer Science Department. He has served as chair of the CS-GRE Examination board, Member of the ACM Council, Chair of the New York State CS Doctoral Evaluation Board, on several NSF advisory boards, and is past or present editor of several journals.

Ullman was elected to the National Academy of Engineering in 1989 and has held Guggenheim and Einstein Fellowships. He is the 1996 winner of the Sigmod Contributions Award, the 1998 winner of the Karl V. Karlstrom Outstanding Educator Award, and the 2000 winner of the Knuth Prize. He is the author of 16 books, including widely read books on database systems, compilers, automata theory, and algorithms.

His research interests include information integration, warehouse design, and data mining. In the mid-1980's his NAIL project, which developed many of the fundamental ideas behind deductive databases --- ideas that are now being used in a number of information-integration systems, and the SQL3 recursion standard.

More recently, he worked on data-cube design, developing a method now used in at least two commercial systems for selecting views of a data cube to materialize, in order to optimize the response rate to a given mix of queries. He has also begun a project called MIDAS (Mining Data at Stanford) to address a number of problems involved with extraction of information from very large bodies of text, including the Web.