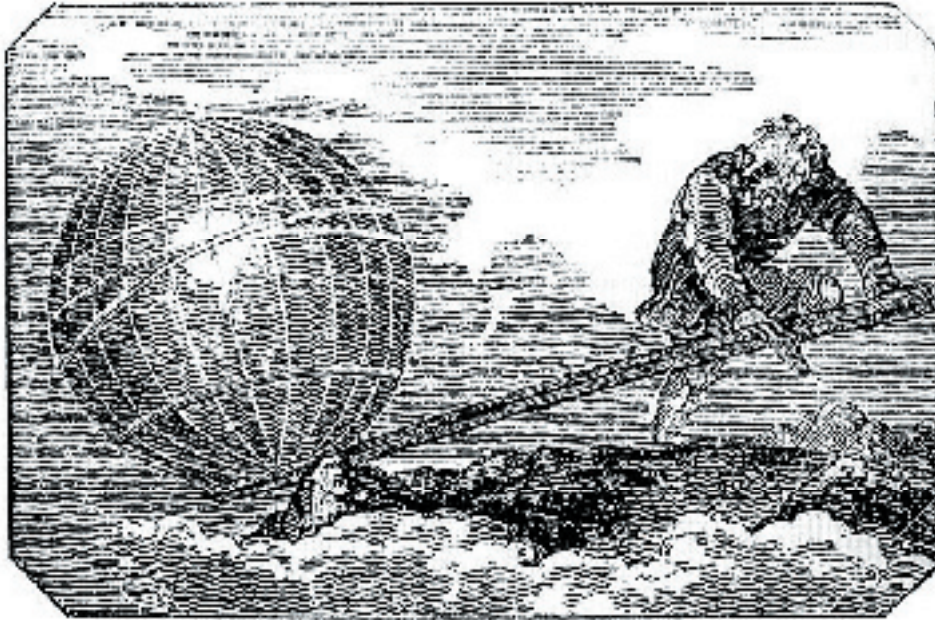




## ► The Perfect Database

By Richard J. Cichelli, SCS President

A version of this article appeared in *Newspapers & Technology* (September, 2006)



<http://www.math.nyu.edu/~corres/Archimedes/Lever/Lever.jpg>

"Give me a place to stand on, and I can move the earth." (The Works of Archimedes with the Method of Archimedes, edited by T. L. Heath, Dover Publications, Inc., New York, 1953, p. xix.)

Elementis Maximus was more specific when he said "Give me a long enough lever and a place to put a fulcrum and I'll move the earth."

Once the perfect tools were levers, now they're database management systems. You notice, however, in the advocacy of levers the devil is in the details. The trick is to abstract out the confusing details and leave the essential understanding.

Is there a perfect database? This article may help you understand the issues. It is about Information Technology (IT), but it is not written for IT staff alone.

As the President and co-owner (with my wife) of Software Consulting Services, a 30-year-old

Software Consulting Services, LLC

630 Selvaggio Drive, Suite 420  
Nazareth, PA 18064

Sales: 1-800-568-8006

Fax: 610-746-7900

E-mail: [sales@newspapersystems.com](mailto:sales@newspapersystems.com)

[www.newspapersystems.com](http://www.newspapersystems.com)

newspaper systems vendor, I have a perspective about information technology that differs from those who purchase it or use it. My role is that of one who pays to have it developed. And once having done this, I have to sell it. The choices I make come long before I have a solution. I then live with those choices and back them up with millions of dollars of my own money as SCS builds systems for the newspaper business.

When a prospect for our systems puts out initial questions about not what we do but how we do it, we hear several recurring refrains. "Do you use QuarkXPress® or Adobe InDesign® for pagination?" "What is the database that you use?"

If you have a vision for how systems should be built, then neither of these questions is likely to be top on your list. Let me tell you how I think about it.

Newspapers need highly functional systems to efficiently run their businesses. Some of the tools that they need are very specific to them. An integrated newspaper system for all but the smallest of newspapers is a highly complex collection of software.

**SCS BUILDS TRUSTED NEWSPAPER SYSTEMS.**



The simple fact is that building newspaper systems is about solving some very large and complex problems. As development proceeds you hope that the software will not grow so dauntingly complex as to preclude providing a functional, installable and maintainable solution. So the question is, "What tools will help you with building, maintaining and installing newspaper software?"

This is an engineering issue. It has to do with problem solving. Problems are either small, solved in a single step, or larger, requiring a solution composed of a number of steps. If you can envision a number of steps, then you can also envision that each step itself might be small and solved immediately or again needing refactoring into a series of even-smaller steps, until you get to those steps for which you have solutions. So the question is, when you are building newspaper systems or for that matter any software systems, "How do you factor big problems into smaller problems that you can solve?"

Understanding this is easier than you might think. You know that your computer has programs and data. The programs examine and modify data. For example, Microsoft Word® can be used to change the data in a Microsoft Word document. Now documents are a type of data, and data is something used to describe nouns. The data is stored in memory (RAM, disks, tapes, etc.). Being big is what you want the memory to be. Besides the memory, the other computer part is the processor. Its job is to change the data based on programs. Being fast is what you want the processor to be. Programs are made up of instructions which are verbs - everything from add, subtract, to sort and save.

Clearly in a world where there are verbs and nouns (or operators and operands), factoring big problems into small ones requires choosing between refining a verb or a noun at each step of the problem-solving process.

Database management systems essentially facilitate a view of the world based on nouns. If someone tells you, "I have the perfect database management system for newspapers," what they are saying is they think they can build complex systems such as those that newspapers use based on a noun-centric view of the world.

Of course, the devil is in the details. Clearly some pieces of a newspaper system can be built

effectively around a database view of information processing technology. And, of course, if there exists such a tool that manipulates the types of data that newspapers deal with (everything from text to pictures, etc.), then this view is likely to yield useful solutions. Managing newsprint inventory is easily done in a database context. Paginating classifieds quickly and optimally is not.

A problem arises, however, when a vendor claims for an enterprise advertising system or enterprise newsroom system that, "everything can be stored in the database." Here you can imagine immediately that such a solution would entail significant complexity due to the fact that it is unlikely that the whole could be understood from a conglomeration of all the pieces.

As a practical matter, I know this to be the case. Any system with over a hundred different types of database entities is considered large. (In database terminology, a database entity is something like the set of customers, or ads or insertion orders.) Our advertising front-end system with its billing and sales management tools has nearly 275 entities - a very large number. Our well-known Layout-8000 ad dummied product has well over 150. So, by now you should appreciate the naïve view that simply having the right database and putting all the data in it isn't appropriate if you want to build, install and maintain newspaper systems.

The problem of building newspaper systems can also be factored by verbs. In this view, one thinks of "taking ads" as opposed to "ad records." If you build an enterprise newspaper system around verbs, you would say your system has a service-oriented architecture. Here are some services that a newspaper system might have: wire capture, ad entry, billing, accounts receivable, ad dummied, classified pagination, etc.

So if I wanted to have a system built around a service-oriented architecture, what database would I choose? The answer is simple.

Service-oriented architectures are about verbs; database-oriented architectures are about nouns. Both are clearly appropriate in some cases; both are clearly inappropriate in others. Neither is appropriate for all cases.

Is this issue something new? No, many vendors have faced it previously. One of the biggest



software vendors, SAP, faced it in the mid-90's. They developed a successor to their mainframe-based R/2 system called R/3. R/3's goal was to allow application distribution. R/3 changed SAP's approach to software from database-oriented to service-oriented.

Service-oriented architectures have a number of desirable attributes. Since the pieces are functionally oriented, new releases of different services can come out independently. Since the subsystems are distributed and based on exchanging and replicating data rather than storing it in a single place, system upgrades are greatly simplified. And, of course, it's much easier in a system built on services to attach components from a number of different vendors. With a service-oriented approach, you get to build an integrated system from "best of breed" technology.

There are numerous technical advantages to a service-oriented architecture as well. Transactions can take place across application servers based on numerous data services. Data can be replicated where required according to the underlying distribution model. Again, buying multiple

applications from different vendors is made much easier. And finally, as noted emphatically by SAP, the Achilles' heel of R/2 was its single database constraint. This was eliminated by the distributed service-oriented model of R/3.

One of the characteristics of well-engineered service-oriented applications is that the components are tied together via messaging. Today the messaging technology of choice is based on XML. Messaging, not a database, is the key to building service-oriented systems (especially those that use the web).

For those who began reading this article anticipating that I would identify Oracle®, MySQL®, MS-SQL®, PostgreSQL®, or some other database-oriented tool as the perfect technology for building newspaper systems, I apologize. At this point, however, I will deliver on my promise to provide the answer to, "What is the perfect database?" The perfect database is the one which, when mentioned along with other magic words by a vendor's salesperson, provides a unique and compelling selling message. If you can't sell what the application does, sell its database.