Schussel's

DOWNSIZING JOURNAL

October 1992

In This Issue:

Distributed & Client/Server DBMS: Underpinning for Downsizing Part II of IV

This is the second article in a four part series by our editor on distributed and client/server DBMS architectures and fundamentals.

More details on distributed DBMS

Distributed DBMS are where the interesting action is happening in the large systems DBMS market (mini-computer to super-computer). As SQL emerges as the standard DBMS language, the principal methods DBMS vendors are using to differentiate their products is to add various functions including:

(continued on page 10)

Data Quality and Downsizing: An Interview with Mark Hansen

The following is an interview with Mark Hansen, Principal of QDB Solutions, Inc., a Cambridge-based firm that specializes in data quality. Since data quality is emerging as an important aspect in downsizing database systems, our editor decided to give Mark a call to ask him a few pertinent questions.

GS: As I talk to people about migrating existing applications to client/server environments, I'm finding that data quality

(continued on next page)

Data Quality...

(continued from front page)

(or the cleaning up of old data) represents the most labor intensive and difficult part of the transition. Because of that, I was particularly interested in your views on data quality and how one can use computers to both filter and monitor data. Can you explain the basic principals of computer-based data quality enhancement?

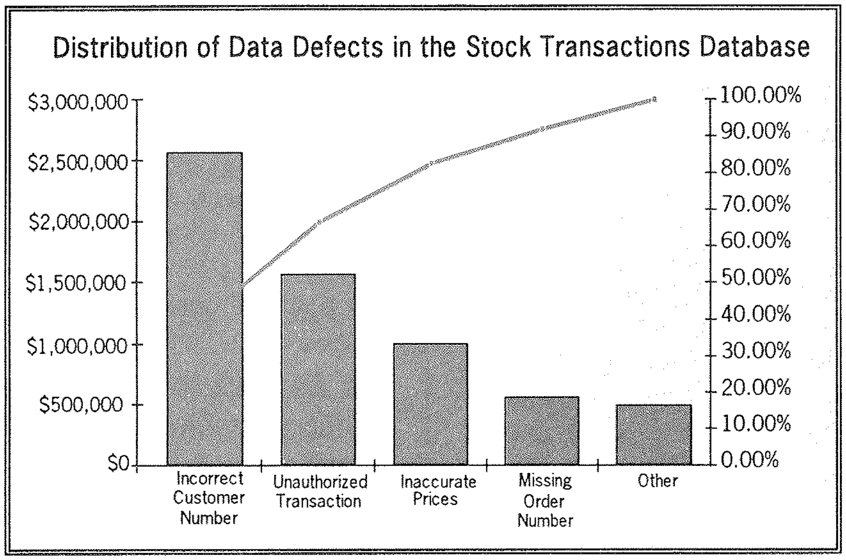
With data quality, as with manufacturing quality, measurement is the key to successful management. You can't manage what you don't measure.

Quality gurus like Dr. Edward Deming teach that to improve quality, we must measure the rates at which a process produces defects and then improve the process to reduce the types of defects observed. We need to continually measure quality, analyze the results of our measurements, and improve the process based on our analysis. This three step process of continuous improvement form the basis for Total Quality Management (TQM).

Working with our customers, I have found that TQM provides the most effective strategy for improving data quality. The key role played by computers in the TQM process is measurement. Unlike manufacturing, where manual inspection is often necessary to measure defects, computers can be used to automate the inspection process that identifies data defects.

We teach our customers to develop data quality metrics - measurements of the impact that poor data quality has on the business (e.g., the total dollar value of a hospital's receivables that are unpaid because of data quality errors). The metrics developed are implemented using software tools that allow a company to continuously monitor its data quality levels.

Computers are also used to drive the analysis and improvement process. When a company's metrics indicate that data quality levels are unacceptable, exception reports are generated to indicate the types of data defects that are occurring and where they are originating. Computer generated quality control charts can be used to determine where to take action first in order to improve data quality. For example, using the pareto diagram at the bottom of this page, a manager can tell that resources should first be applied to correcting the sources of "Incorrect Customer Number" errors. Each of the five bars in the graph represents a type of data defect that occurs in the Stock Transactions Database. The left axis indicates the total dollars associated with each of the defect types. The solid line indicates the cumulative impact of the defects on the overall quality of the database, and should be read against the right axis.



GS: Do there exist any simple ways that a company can determine the quality of its data files before they begin a downsizing migration?

The most straightforward and effective way to determine the quality of your files before beginning a downsizing migration is to conduct a data quality audit. There are two components of such an audit: (1) determining the requirements for data quality in the new environment, and (2) measuring the quality of existing files against those requirements.

To define the data quality requirements, a company needs to consider both the business requirements (e.g., medical claims for pregnant males are invalid) and the technical requirements (e.g., referential integrity). When checking existing files for data quality, many companies run into trouble because the IS organization defines requirements that are too detailed, or not aligned with business needs for the downsized environment. As a result, the organization wastes time checking the wrong problems and overlooking the problems that end-users really care about.

One of the methodologies that I use accelerates a data quality audit by providing a step-by-step process for developing the correct business and technical requirements. Requirements are developed based on interviews with end-users and IS staff.

Once requirements have been developed, files can then be measured against these guidelines. This can sometimes be accomplished by writing code to run directly against the files. However, there are three reasons why this is not the best approach: 1) writing code can be very time consuming; 2) data quality checks are often complex to program, and writing code may tie up some of your most talented development resources; 3) the code produced will not be portable, and if you want to run the same data quality checks on different files (perhaps from different software or hardware environments), you end up with a software conversion project.

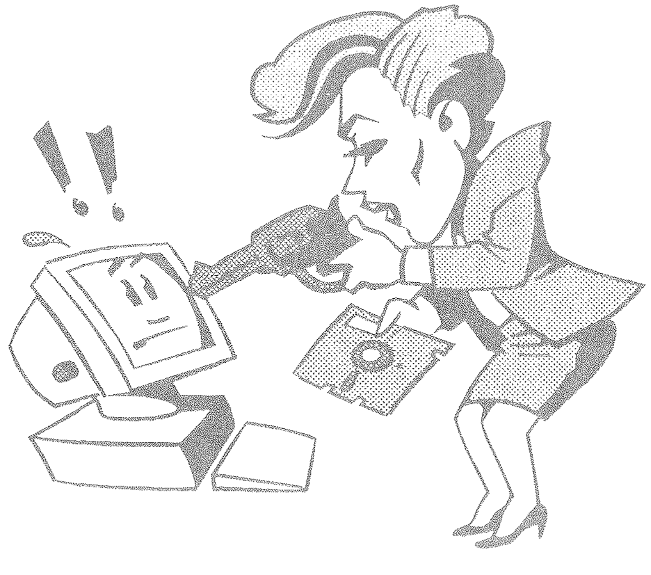
To address these problems our company has developed a software product, QDB/Analyze, for rapidly checking data files against data quality requirements. The product allows you to check for the most common types of problems (duplicates, incomplete data, invalid or corrupt data, improperly formatted data, etc.) using a single point and click Windows interface screen.

More sophisticated checks involving business rules or statistical checks (e.g., the utility bill that is much higher than average for a particular customer) can also be developed through the tool's graphical user interface without writing a line of code.

This software product can also be run against data from any source. So there is no need to port the data quality checks across platforms to analyze multiple data sources. Running under Windows on a 386 or 486 machine, a data quality audit can be conducted quickly and effectively.

Currently, to use QDB/Analyze, the data to be analyzed is downloaded onto the PC from its native environment. In the next release, QDB/Analyze will operate in client/server mode, which will allow the product to run as a client under Windows

(continued on next page)



Data Quality...

(continued from previous page)

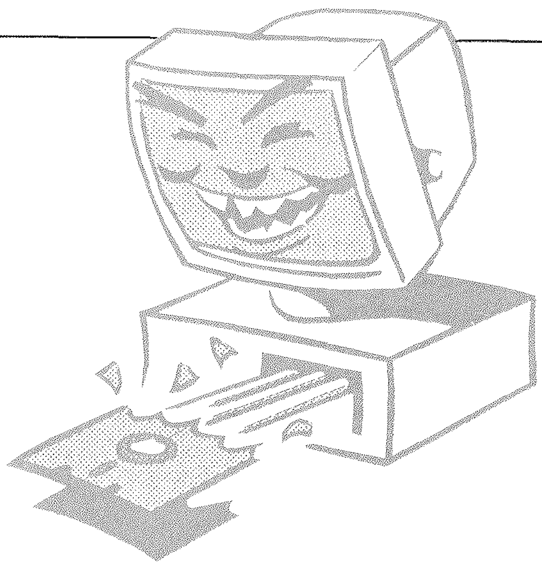
against server data located elsewhere.

GS: Are there certain types of environments that you know of from experience that contain or promote especially large amounts of bad data?

Generally, the more local the data is, the lower its quality. This is intuitive when one considers that quality can be defined as "conformance to specifications." If a database is used by only one person, or by a small department, then its data needs to satisfy the specifications of only a small set of users and applications. Idiosyncrasies in the data are understood and tolerated by this small community.

From a downsizing perspective, the legacy systems that companies use to process transactions or support narrowly defined business functions usually present the most data quality problems. Downsizing involves pulling data out of local environments with older technologies (IMS, IDMS, VSAM) and putting it into shared relational environments.

The trouble is that the old transaction processing systems were designed to do a specific job. A typical transaction system will capture a large amount of data. However, only a small subset of that information has to be correct in order for the transaction to be processed. The remainder of the data collected is not critical to processing the transaction. As a result, it is often incomplete or erroneous.



One insurance company told us that when they loaded data from their claims system into a relational database to be shared by the marketing and actuarial departments, it was discovered that most of the data stored in the "diagnosis code" field was invalid. Because this data element was not critical to processing the transaction, claims processors had been entering default values or leaving it blank. Invalid diagnosis codes may not have been critical to claims processing, but for the actuarial and marketing department, it significantly degraded the value of the data.

As companies move toward downsized, open environments where corporate data resides on servers that are shared across departments, organizations will need to significantly improve the quality of the data that is now stored in legacy systems.

GS: What are the types of data quality issues that a company in the process of downsizing needs to be aware of?

Data quality problems pose serious risks to downsizing projects. The most dangerous problems are invalid keys, undependable data elements, and poorly synchronized data sources. Downsizing projects that integrate data from multiple sources into relational databases in an open systems environment are particularly at risk.

For example, a large manufacturing company that we have worked with recently began a systems integration project designed to downsize decision support applications from the mainframe to a UNIX environment. As part of this process, the company needed to combine IDMS and flat file data feeds from the purchasing, engineering, inventory, cost management, and scheduling departments into a relational database. After spending considerable energy designing the logical and physical structure of the database, they became nervous that data quality problems would jeopardize the project. An investigation of the data feeds revealed each of the three problems mentioned above.

Invalid keys threatened the integration of data from the multiple sources. The financial account number, used to combine multiple feeds, was found to be corrupted or invalid in over 20% of the cases. As a result, project management applications running on the relational database would present inaccurate and incomplete information to business end-users.

Undependable data elements threatened the quality of expense tracking information. The source-code field, used to identify the organization originating a purchase order, was often empty or populated with "dummy" values on the mainframe. As a result, the new UNIX decision support application that reported expenses grouped by organization presented misleading information to users unfamiliar with the idiosyncrasies of the mainframe purchasing system.

Poorly synchronized data sources threatened the accuracy of project management information maintained in the UNIX environment. For example, data from the mainframe scheduling system was used to derive a planned-expense field for each project tracked by the new system. However, because project schedules were often bogged down in administrative approval processes, data from the scheduling feed was found to be out of date by the time it reached the relational database in the downsized environment. As a result, the planned-expense field did not accurately represent the costs associated with its corresponding project.

GS: What are some ideas for establishing and maintaining a data quality program before, during, and after downsizing?

Before beginning a downsizing program, it is critical that a company conduct a data quality audit of the systems that will be downsized. The audit will tell you where data quality falls short of the requirements so that these issues can be addressed early in the process. Companies that do not conduct an audit run the risk of getting half way through a downsizing project, having spent millions of dollars, only to realize that data quality problems will prevent successful completion of the downsizing effort.

During a downsizing program, companies need to continuously monitor data quality (using the filters and metrics developed during the audit) to ensure that, as new data is brought into the downsized environment, it meets the data quality requirements that have been established. Just as software is tested at regular intervals during development, data quality measurements should be taken on a regular basis (e.g., weekly, bi-weekly).

After the downsizing program is completed, companies should move to a complete Total Quality Management (TQM) program for their data. As described earlier, this involves (1) measuring the impact that data quality has on the business in the downsized environment, (2) analyzing the causes of the data quality problems that are having the biggest impact on the business, and (3) working with data suppliers to improve the underlying systems so that fewer data quality problems are created in the future. These three steps should be repeated regularly to maintain a "continuous improvement process" for data quality in your company.

GS: New remote data access initiatives, such as IBM's DRDA and Information Warehouse, are going to make access to existing databases more generally available. What impact do you think data quality will have on the success of these initiatives?

Data quality will have a significant impact on the success of both DRDA and the Information Warehouse initiatives. In both cases, IBM and the other vendors

(continued on next page)

Data Quality...

(continued from previous page)

supporting these initiatives have focused on issues of database design and architectures. I believe it is crucially important to focus on the quality of the data that will be used to populate the databases used to implement these initiatives.

The goal of both DRDA and Information Warehouse is to provide remote access to an organization's distributed data resources from a single source. Accomplishing this requires that an organization's multiple databases be viewed as a logical, consistent whole. Data quality problems arise because most organizations do not design their multiple disparate databases to work together as a logical whole.

...Downsizing projects that integrate data from multiple sources into relational databases in an open systems environment are particularly at risk....

As a result, organizations that buy into these IBM initiatives will need to spend a certain amount of energy preparing their data for DRDA and Information Warehouse. The data quality issues they will need to address fall into two categories: technical and business.

The technical issues involve upgrading an organization's data so that there is a certain amount of integrity across multiple databases. For example, within DRDA, referential integrity applies not just to the tables within a database, but within the entire distributed architecture. This means that if the purchase order table has an account number field, which is used as a foreign key within DRDA to reference an accounts table in the finance database, then there must be referential integrity between those two tables. That is, every account number that appears in a purchase order should also appear in the accounts database.

Since most databases that have grown up separately (e.g., purchasing and finance) will not have enforced such constraints, integrity across systems is likely to be very poor. In short, databases that have grown up separately will not "fit together" well within DRDA or the Information Warehouse.

The business issues involve improving the quality of data stored in local systems so that this data will satisfy the requirements of a broader community with remote access. Even if the databases used within DRDA "fit together" in a technical sense, data quality across the architecture may not be satisfactory from a business perspective. In the insurance company example discussed earlier, DRDA implies that the marketing and accounting departments would have remote access to the claims database. However, the data that is accessed is not of value to these departments because the diagnosis and treatment fields have not been maintained properly.

DRDA and Information Warehouse require that higher standards of data quality be maintained across systems. The owners of a database must not exclusively enforce the integrity and data quality constraints important to a narrow set of users. They must also insure that the data quality requirements of all users with access to the database through DRDA are met. GS

The promise of remote access to data that is offered by these initiatives is tremendous. As companies begin moving in this direction, QDB Solutions looks forward to working with them to improve data quality to the levels necessary to be successful. QDB Solutions provides software, consulting, and education for database data quality management. Dr. Hansen may be reached at QDB Solutions, Three Cambridge Center, Cambridge, MA 02142, (617) 577-9205.

Downsizing: What's Really Going On...

"Just do it."

Some of the best advice I have ever received was that 99% of any job is getting started. I have found that this admonition is doubly true in the mainframe world. MISers have become so accustomed to analyzing choices that result in huge expenditures, that elaborate analysis procedures have been constructed. Such procedures have traditionally been necessary for two valid reasons: 1) to ensure that the best decision is made, and 2) to ensure that if a mistake is made, the company would be well-covered. However, in the field of downsizing, such extended analysis can become "analysis paralysis" since the objects of our attention often change too fast to be captured for analytical consideration.

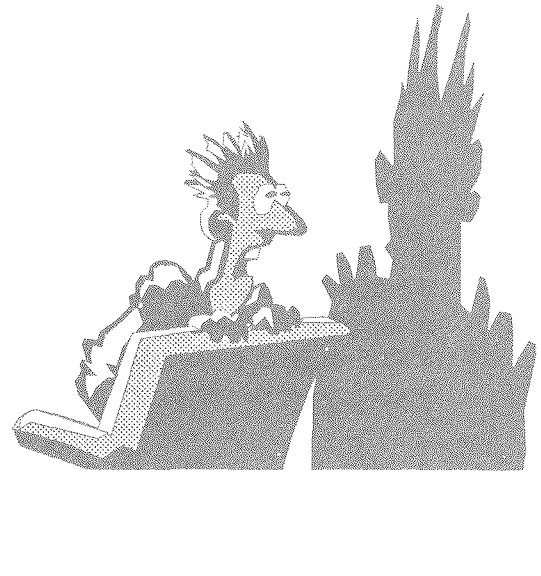
Here is the perfect example of "analysis paralysis": one company I have worked with needed to select a database standard. They, therefore, decided to select a database committee. The first logical step was to decided who would be on the committee. So, a "steering committee" was selected to select the database committee. I think you get the picture. After eighteen months of similar processes, the database committee reached a critical conclusion: they would use Sybase for their database. Shortly after the decision was made, Oracle released version 7.0 which contained many of the critical features that had initially pushed the database committee towards choosing Sybase. Already their decision had become outdated.

I know of another company that in their "analysis paralysis" spent six months deciding that the basic corporate PC purchase should be 25 MHz 386 with a 40 MB hard drive and 2 MB of RAM. Shortly after this decision was made, Windows 3.1 and OS/2 2.0 were both released, and hardware prices dropped by 50%. Their configuration standard is now inadequate.

The moving target

Lengthy analyzes have become counter productive since the cost of buying and trying different solutions is often significantly cheaper than the cost of analysis. The old approaches of the mainframe world were acceptable because of the longevity of most equipment. In downsized environments, however, new technology and competition have shortened product life-

(continued on next page)



Periscope...

(continued from previous page)

cycles to less than six months in certain cases.

Consider how far things have progressed in a few years: it's not that long ago that the greatest dream of my life was to own a PC with a 10 MB hard drive. Now, no one even makes a 10 MB hard drive since 40MB is barely large enough to accommodate some of the new operating systems!

In the past we dealt with one, three, and five year plans. But with hardware and software evolving so rapidly, only the most foolhardy would make specific, iron-clad plans beyond a two year horizon. Imagine our new world: in two years the 686 will have arrived with 20 million transistors (the 486 has only 1.5 million) and operating systems like Windows NT and Novell will routinely support 10, 20, or 100 686s in parallel. Several nationwide wireless networks will have been implemented.

...In two years the 686 will have arrived with 20 million transistors... and operating systems like Windows NT and Novell will routinely support 10, 20, or 100 686s in parallel...

In addition to such advances, new forms of computing will be common. John Sculley of Apple predicts that the use of pen-based, wireless, personal data assistants will be a $2.5 trillion market! So, if you let your downsizing analysis take a year, you'll always be chasing a moving target.

The new "interactive" development approaches

Just as decision making requires new approaches, so does development. For the last 25 years, systems development has required a substantial amount of pre-planning. It was not unusual for a large project to require six months or a year of design prior to the generation of even one line of code. Such planning was necessary because changing even a few data elements after coding had begun could easily affect hundreds of programs and significantly extend the duration of the project.

In fact, most traditional development approaches have historical basis. In the sixties, we would submit a deck of cards for a nightly run. Computer-time was very expensive while human-time was very cheap. After one run, we'd spend an entire day checking code so as to maximize the value of the next nightly test. Using this approach with languages including COBOL, PL/1, and FORTRAN led to "top-down programming," "structured programming," and HIPO among other design techniques.

Now, the situation is reversed. Cycles of computing power are incredibly cheap — it's people that are expensive. In addition, more advanced languages have been developed. Therefore, new techniques and approaches are in order.

Utilizing what we refer to as "interactive programming techniques," development time can be cut by as much as 90%! Such techniques are most effective with non-procedural languages possessing active data dictionaries. In essence, you utilize the language as if it were a prototyping tool, refining the model as you interact with the users. In the time it would normally take you to prototype a system, you'll have a completed application.

The new procedure can best be presented as follows:

1. analyze the existing system
2. interview the users to determine how the system actually works
3. obtain a basic understanding of the system

(continued on page 15)

Schussel Convinces Elephants to Downsize...

At the beginning of September, Ron Peri and I had the chance to visit South Africa for a seminar series on downsizing. We were invited by Hermus Erasmus, Managing Director of Perseus, the leading South African open systems vendor. Two years ago, when Erasmus took the helm at Perseus, the company was operating solely as a distributor and reseller for Data General. Recognizing important trends, he aggressively repositioned the company into the open systems market by focusing on systems integration of various products that target the LAN and UNIX environments. The result has been a 40% per year growth rate.

In South Africa, Peri and I spoke to three groups (a total of approximately 400 people) about downsizing, LANs, and related client/server topics. The presentations were extremely well-received, and there was much interest shown by the attendees in obtaining more product and management information. As a result, Perseus Computer and DCI will be sponsoring a DOWNSIZING EXPO in Johannesburg for the spring of 1993. This event should help to launch downsizing as an area of major interest within South Africa.

I found that in South Africa, the general understanding of downsizing and client/server computing approaches was much less than in the US. While in a typical North American office you can expect to find a PC or Macintosh on almost every desk, the office landscape in South Africa more closely resembles the typical US office of 1985.

Currently, South Africans are paying 30% more for their PCs than do North Americans. Mainframes are also approximately 30% more expensive in South Africa. Therefore, due to their comparatively lower standard of living, the incentive to migrate to LAN-based computing is very great.

The knowledge gap between the US and South Africa is large. Knowledge of the enabling technologies — LANs, LAN O/Ss, open O/Ss, server DBMS, windows tools, etc., is low. I believe that in South Africa, there is a fabulous opportunity for entrepreneurs to educate and establish markets for the new technologies that *Schussel's Downsizing Journal* has been preaching. At the same time, one needs to be very aware of their tumultuous political situation. There is some probability, to what extent no one knows, that the country will descend into civil strife, revolution, or chaos as the necessary political changes proceed.

I hope that South Africa can make the transition to a peaceful, pluralistic government successfully and soon. However, just as the republics of the former USSR are discovering that there is no easy transition to free and open market democracies, so will South Africa will find that there is a high price to be paid for the under-education and discrimination against the majority of its citizens.

Despite the political turmoil, South African computing is relatively progressive. I'm sure that downsized and open systems will begin to dominate in their future as it has in ours. Both Peri and I plan to return next year for DOWNSIZING EXPO - maybe we'll see you there! GS

Distributed & Client/Server...

(continued from front page)

- distributed or client/server computing
- support for object approaches
- addition of database semantics
- addition of more relational functionality (typically semantics)

Distributed database software needs to provide all of the functionality of multi-user mainframe database software, while allowing the database itself to reside on a number of different, physically connected computers. The types of functionality distributed DBMS must supply include data integrity, maintenance through automatically locking records, and the ability to roll-back transactions that have been only partially completed. The DBMS must attack deadlocks to automatically recover completed transactions in the event of system failure. There should be the capability to optimize data access for a wide variety of application demands. Distributed DBMS should have specialized I/O handling and space management techniques to insure fast and stable transaction throughput. Naturally, these products must also have full database security and administration utilities.

The discussion below first focuses on the basic, and then advanced functions for a distributed DBMS. However, it won't be helpful to use this section as a feature checklist since there is a great disparity between performing these functions at a minimum level and accomplishing them at an advanced level.

Basic requirements for a distributed DBMS

- *Location transparency* Programs and queries

DISTRIBUTED DBMS - REQUIREMENTS

1) LOCATION TRANSPARENCY

QUERIES CAN ACCESS DISTRIBUTED OBJECTS (DISTRIBUTED JOIN) FOR BOTH READ & WRITE WITHOUT KNOWING THE LOCATION OF THOSE OBJECTS. THERE IS FULL LOCAL DBMS & DD.

2) PERFORMANCE TRANSPARENCY

A QUERY OPTIMIZER MUST DETERMINE THE BEST (HEURISTIC) PATH TO THE DATA. PERFORMANCE MUST BE THE SAME REGARDLESS OF THE SOURCE NODE LOCATION.

3) COPY TRANSPARENCY

MULTIPLE COPIES OF DATA MAY OPTIONALLY EXIST. IF A SITE IS DOWN, THE QUERY IS AUTOMATICALLY ROUTED TO ANOTHER SOURCE. FAILOVER RECONSTRUCTION IS SUPPORTED.

4) TRANSACTION TRANSPARENCY

TRANSACTIONS THAT UPDATE DATA AT MULTIPLE SITES BEHAVE EXACTLY AS OTHERS THAT ARE LOCAL. THEY COMMIT OR ABORT. THIS REQUIRES A 2-PHASE COMMIT PROTOCOL.

5) FRAGMENT TRANSPARENCY

THE DDBMS ALLOWS A USER TO CUT A RELATION INTO PIECES, HORIZONTALLY OR VERTICALLY, AND PLACE THEM AT MULTIPLE SITES.

6) SCHEMA CHANGE TRANSPARENCY

CHANGES TO DATABASE OBJECT DESIGN NEED ONLY TO BE MADE ONCE INTO THE DISTRIBUTED DATA DICTIONARY. THE DBMS POPULATES OTHER CATALOGS AUTOMATICALLY.

7) LOCAL DBMS TRANSPARENCY

THE DDBMS SERVICES ARE PROVIDED REGARDLESS OF THE LOCAL DBMS BRAND. THIS MEANS THAT RDA AND GATEWAYS INTO HETEROGENEOUS DBMS PRODUCTS ARE NECESSARY.

may access a single logical view of the database; this logical view may be physically distributed over a number of different sites and nodes. Queries can access distributed objects for both reading and writing without knowing the location of those objects. A change in the physical location of objects without a change in the logical view requires no change of the application program. There is support for a distributed JOIN. In order to meet this requirement, it is necessary for a full local DBMS and data dictionary to reside on each node.

- *Performance transparency* It is essential to have a software optimizer create the navigation for the satisfaction of queries. This software optimizer should determine the best path to the data. Performance of the software optimizer should not depend upon the original source of the query. In other words, because the query originates from point A, it should not cost more to run than the same query originating from point B. This type of technology is rather primitive at this time and will be discussed later in this article series.
- *Copy transparency* The DBMS should optionally support the capability of having multiple physical copies of the same logical data. Advantages of this functionality include superior performance from local (rather than remote) access to data, and non-stop operation in the event of a crash at one site. If a site is down, the software must be smart enough to re-route a query to another data source. The system should support *fail over reconstruction*: when the down site becomes live again, the software must automatically reconstruct and update the data at that site.
- *Transaction transparency* The system needs to supports transactions that update data at multiple sites. Those transactions behave exactly as others that are local. This means that transactions will either all commit or abort. In order to have distributed commit capabilities, a technical protocol known as a two-phase commit is required.

...There is a great disparity between performing these functions at a minimum level and accomplishing them at an advanced level...

- *Fragmentation transparency* The distributed DBMS allows a user to cut relations into pieces horizontally or vertically, and place those pieces at multiple physical sites. The software has a capability to recombine those tables into units when necessary to answer queries.
- *Schema change transparency* Changes to database object design need only to be made once into the distributed data dictionary. The dictionary and DBMS automatically populate other physical catalogs.
- *Local DBMS transparency* The distributed DBMS services are provided regardless of the brand of the local DBMS. This means that support for remote data access and gateways into heterogeneous DBMS products are necessary.

IBM's four ways to distribute data

Most vendors have been taking many years to develop software that offers distributed

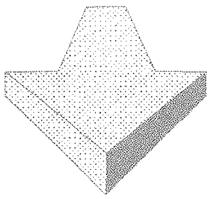
(continued on next page)

downsizing, Upsizing, Rightsizing, or What?

Subtitled: The Politically Correct PC

Lately, as I've been talking to various companies that have successfully accomplished downsizing projects, I am finding that more than ever, people are objecting to the use of the term *"downsizing."* Of course, this is not a new issue. In 1989, when DCI ran its first conference on downsizing, I received strong advice from a prominent consultant not to use the term *"downsizing"* in our title.

What's going on here? Well, there are several issues at hand. Perhaps one of the most obvious problems with *"downsizing"* is that the term is a homonym. In the business field, downsizing means a reduction in the employment force — usually in the form of layoffs. Since this type of downsizing can have frightening connotations (and the fact that it has been implemented at many US firms in the past few years), there is an implied negativity from the mere use of the word.

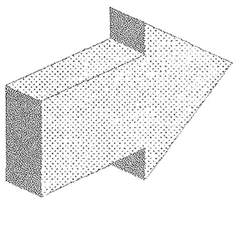


The second problem with the term is that if your current employment is directly related to mainframes, you may (logically) come to the conclusion that your firm's progress towards downsizing its computer system may result in a reduction of its need for you. There is an abundance of history to support this logic since many firms who have employed downsizing as a data processing strategy have also been interested in downsizing personnel.

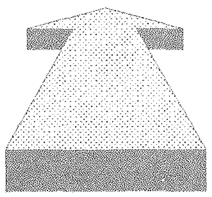
In order to make the concept of downsizing more politically correct (PC), quite a few alternative terms have been proposed for use. Some that I have heard include:

- Upsizing
- Rightsizing
- Smartsizing
- Surroundsizing

However, none of these terms denote or express the same meaning as does *"downsizing,"* as each has subtle differences. With the goal of helping standardize the usage of these alternatives, I offer the following definitions:

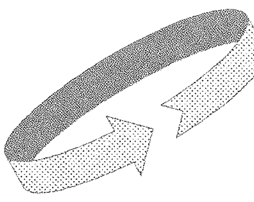


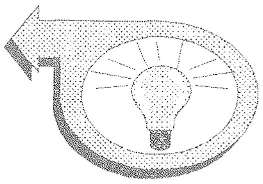
Downsizing — This refers to the aggressive use of PCs and LANs to replace functions that would have traditionally been performed on mainframes or minicomputers. Downsizing frequently, but not always, implies the use of client/server styles of computing — including the new style of server built with microprocessor engines. Definitely, any system that consists entirely of PCs would be accurately described by this term.



Upsizing — This is a word that I've been pushing at DCI conferences. What is meant by upsizing is the movement of file management-based PC systems into the data processing world of SQL and client/server solutions. This movement gives users the robustness and data integrity of the mainframe world, but on the cheaper platforms of networks and PCs.

Rightsizing — IBMers have always objected to my use of the *"D"* word. As an alternative to the idea of migrating applications onto smaller platforms, most of the IBMers I know have used the term *"rightsizing."* They are usually talking about the replacement of large MVS systems with multiple AS/400s rather than with PCs. Many IBM documents have used this term to precisely describe the movement to this IBM mid-range processor. Many in our industry have picked up on the term *"rightsizing"* to mean the proper integration and use of mini-computers into the downsizing formula. My conclusion is that when someone talks about rightsizing, they are probably meaning to specifically include mini-computers and/or UNIX machines within the mix of processors.





Smartsizing— This word is largely a synonym for rightsizing, but in addition implies that mainframes can be included as part of the mix of processors in any final configuration. Smartsizing means "use the best processor for each part of the computing solution."

Surroundsizing— This expression was first suggested to me by Susan Nurse of Informix. Informix's client/server and distributed database solutions are being used in large projects that, until recently, were always performed on mainframes. When selling into that culture, they have found that it doesn't help to antagonize or intimidate your potential customers. One way to gain their trust is to explain that downsizing concepts can be applied to systems *surrounding* the mainframe, without pulling or throwing away the mainframe investment. This kind of approach can be very successful in exposing people to the values of downsizing without scaring them.

So, call it what you will, but the migration towards down-, right-, smart-, or surround-sizing is not abating. GS

Periscope...

(continued from page 8)

and approximate the application

4. review the application with the users, refining the system each time
5. document the system

This rapid approach works because changes that in COBOL might take a few weeks, can be done in a few minutes *while the user watches*. Not only is the process faster, but the final result is far superior. And, at the end of the process, users have the system they want. Lest it seem as if this approach only works for small systems — we have successfully used this approach for a workers compensation system designed to support over five hundred users.

Another example is a case in which the postal service had estimated that one project written on the mainframe in COBOL would take two years and cost $2 million. Several enterprising individuals using these techniques and a non-procedural language called "Magic" produced the bulk of the application in two weeks!

Changes in business

Not only is technology changing, but, in all likelihood, so is your business. And be assured that if your business *isn't* changing, your competitor's business is. In industries like insurance and pharmaceuticals, failure to embrace the new technologies creates a substantial competitive disadvantage.

In conclusion, there is an old adage about buying a house that I believe applies well to the field of downsizing: *The best time to start downsizing was six months ago. The second best time is* ***today***.

Just do it! RP

Author's note: Shortly after writing this column, I was team teaching with Dr. George Schussel and was amazed to hear him relate

(continued on back page)

Periscope...

(continued from previous page)

exactly the same "just do it" message based on his recent visit to a downsized company. The account was a full service bank, the IBM Pacific Employee's Credit Union in San Jose, California. A large IBM 4381 had just been replaced by Novell networks at the branches tied into a centrally located IBM PS/2 Model 95. While most banks are saying "It can't be done," this bank president's advice to George was to tell his client's "just do it, there is no question that downsizing is the best approach. In addition, the cost of a mistake is much less than the cost of delay." (Editor's note: a detailed summary of this visit will be run in the November 1992 issue of SDJ.)

Ron Peri is the President and Founder of Computer Support of North America, a firm which provides downsizing services from initial consultations through software conversion, hardware installation, and outsourced support. Mr. Peri can be reached at Computer Support of North America, Basking Ridge, NJ, 908-766-9200.

Schussel's

Downsizing Journal

October 1992
Volume 2 Issue 2

SDJ is published monthly by:
Digital Consulting, Inc.
204 Andover Street, Andover, MA 01810 USA
(508) 470-3870 FAX (508) 470-0526

Editor: Dr. George Schussel
Managing Editor: Stacey S. Griffin

Subscription rates:
$199 annually for U.S. residents
$225 annually for foreign addresses

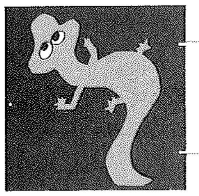
UPCOMING downsizing Events...

One of DCI's newest offerings is **DOWNSIZING EXPO CANADA**. Being held October 19-21, 1992 in Toronto, this premier show features over 50 participating vendors for the exposition, and 50 conference speakers. The conference consists of five tracks: Downsizing, Client/Server, Business Reengineering, Interoperability, and Windows. In addition to Chairman George Schussel, other keynote presenters include: Larry DeBoever, Herb Edelstein, Roel Pieper, Dr. Howard Rubin, Roger Burlton, Cheryl Currid, and Michael O'Beirne.

Implementing Client/Server Applications and Distributing Data, one of the seminars in DCI's Downsizing Seminar Series, is being held in Orlando, November 5-6, 1992. Instructor Herb Edelstein, in this two day course, will cover the following topics: Distributed Systems, RDBMS, Networks, Distributing Data, Database Servers, Distributed and Federated Databases, Distributed Queries, and Transaction Management and Concurrency.

Another seminar from the Downsizing Seminar Series, **Finkelstein's Practical Guide to Client/Server DBMS Computing**, is being held in Toronto, November 23-24, 1992. The seven topics to be covered include: Client/Server Environments, Choosing a Database Server - Feature Comparison, Database Servers: Highlights and Conclusions, Understanding Database Benchmarks, Choosing Front-end Tools, Case Tools for Client/Server Development, and Client/Server Case Studies.

For more information on any of these classes, call DCI at (508) 470-3880.



For your enjoyment and ease of reference, like a chameleon, **SDJ** will be changing its colors monthly. Our changing color scheme will make it easy to spot each new issue at a glance, and quick to reference past issues.