Very Large Databases

How Large? How Different?

David Vaskevitch, Microsoft Corporation, Redmond, WA.

Soon, the world will need far more truly large databases then any of us ever imagined; yet, ironically, without a lot of care, VLDB's as we know them today may be left along the wayside. The way in which we think about, design and build enormous databases will have to completely change if we are to participate in this revolution.

By now everybody, including database people, realizes that the computer world is going through not one but two revolutionary changes. First, of course, there's the whole impact of personal computers, commodity hardware, and ever increasing speed and capacity. Second, there's the impact of the internet with its globalization, ubiquity, and popularization of the very notion of servers. Individually, each of these would be quite a lot to deal with; put them together, and the impact is explosive. This paper deals with the particular impact of this changing world scene on the concepts behind and implementation of very large databases. By the time we're done, the meanings of "very", "large" and "database" will be pretty different than it is today.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 21st VLDB Conference
Zurich, Switzerland, 1995

Very Large Databases Today

Teradata, Tandem, VAXclusters, DB2, Oracle OPS, Sybase Navigator, Informix Massive Parallelism are generally associated with very large databases. They aim to support multi-terabyte databases with high cost, specialized hardware and software. Their customers are large organizations with dedicated technical staff and complex operational environments.

Broadly speaking, these large databases fall into two categories: OLTP and OLAP. The OLTP databases, run by airlines, banks, and other service oriented organizations, process thousands of transactions per second. Yes, most databases handle smaller loads, but it is the very big ones we are talking here. The OLAP databases -- sometimes called data warehouses, DSS's, etc -- are characterized by huge amounts of data and very complex queries.

Databases have traditionally been monolithic and specialized in nature, but VLDB's have taken this trend considerably farther than their smaller brethren. The Teradata and the Tandem systems, two of the leaders in supporting extremely large databases and user loads, use proprietary hardware, operating systems, dbms' and tools. The airlines continue to depend on ACP / TPF an operating system written for their purposes only.

The net result is that very large databases today are unusual. They are in serious danger of becoming both obsolete and irrelevant before they ever get to be successful and popular in the first place. To put this in perspective, let's consider a few datapoints.

On the one hand, it's easy to talk about databases with over 1TB of data. However, when one of the leading industry publications surveyed users of such databases, they had a hard time finding many real ' examples. Examples exist, but they are few and far between. Why?

On the other hand, if we start by thinking about really enormous applications, it's surprising how few of them have really huge databases. For example, SAP, a German company, is one of the leading purveyors of manufacturing and financial accounting software. Their R/3 system is one of the largest application suites ever written on top of an RDBMS. While many of their customers are talking about R/3 db's with 200MB, in the real life today, even 30-50GB is big for an R/3 site. Quite a step down from terabytes.

Of course, many of the world's largest companies have very large applications, larger than R/3 with databases that really are enormous. It is also true that many of these same applications use IMS, TPF, and a variety of either custom or specialized tools and infrastructure to support their load and data requirements. Many of these same companies, when they start thinking about rewriting these applications – which are often 20 or more years old – conclude that the challenge is just to much to undertake. It is in this context that SAP, Peoplesoft, Baan, and others become particularly interesting; it is these high end packaged solutions that become the leading choice for replacing yesterdays largest applications. Since these packaged products are generally specifically written to take advantage of today's tools and today's database technology; it allows us to consider directly how well that technology does at supporting the world of the truly large.

In many ways, an R/3 system typifies the upper limit of what can be achieved with common RDBMS technology on standard platforms: tens of gigabytes, thousands of users, hundreds of simultaneous users. Big, but not huge. It is true that a few customers have managed to build databases much bigger than this, but aside from these exceptional cases, large is normal, but very big is not.

So, overall, very large databases today are hard to build, based on proprietary technology, and pretty unusual in real life. The question is what might make this change in the next few years.

Three Core Shifts

Three core changes will frame our world in the next ten years:

1. Continuing popularity of personal computers within large organizations with an increasing demand for personal empowerment.

2. The boundaries between organizations will disappear as the internet becomes increasing successful. Organizations will need to deal both with each other and with users in their homes.

3. The first two shifts will cause such unbearable pressures on computer capacity that we will, finally, have to figure out how to unlock the power of commodity hardware; the many small will indeed win over the few large.

These three shifts will first make very large databases common, even ubiquitous. They will produce VLDB's that look totally different than what we have grown to expect in this domain.

Empowerment Without Anarchy

Spreadsheets are simple databases. 80% of Excel spreadsheets contain essentially no formulae in them; only sums at the bottoms of columns. These spreadsheets are exactly, literally, small databases. Microsoft's Access desktop database shipments have reached over one million units per month. Clearly tens of millions of individuals build and use databases every day. So, what's new?

The use of real data in actual databases is new. Throughout much of the eighties, for the most part, personal computers were used for everything but database applications. In fact, at Microsoft, until very recently, desktop databases were not even considered part of the *big four* application categories. The standard edition of the Microsoft Office consists of a word processor, a spreadsheet, a presentation graphics package, and a mail client; no database. Yet, now, as it turns out, not only is database right up there along with the other four, but if the truly dominant use of spreadsheets is taken into account, database in popularity is probably second only to word processing. Still, does this new popularity of *personal*, *desktop* databases have anything to do with the biggest of big databases?

The second big thing that is new about desktop databases is that people want to access non-desktop data. In fact, it is the growing availability of non-desktop data that has fueled the success of the desktop databases. A primary reason that desktop dbms's weren't more popular in the eighties is that they could not work with corporate data; at least no easily. Essentially users in the eighties had three choices when it came to real data; the data in the corporate databases. The first choice was to use a desktop data and re-enter, or create from scratch, any data being worked with; not very practical. The second choice was to work with a corporate database directly; working with the real data and being able to manipulate and analyze that data. The third choice, which takes on more importance than we might expect, is to give up and accept not having good data at all. Rather than re-enter data pulled down from a mainframe, the user just accepted the fact that he had to more or less make up data as he went along; derive data independently, or find other sources for the data. Let's consider those last two choices.

There is no question that since 1980 the relational database industry has grown up and now generates several billion in revenues each year. At the same time, most surveys show the vast majority of the world's *business data*, in large organizations, still sitting in non-relational databases. How can both these facts be true? The fact is that the force driving the growth of the RDBMS is our option #2 of the previous paragraph; providing users with access to corporate data primarily for analytic and question answering purposes. Now, here's where things really get interesting.

The eighties was a period in which a small number of users within larger organizations for the first time got access to relational databases which allowed them to analyze and manipulate data in new ways. Those users created sufficient demand to build several large db companies. But those users represented a small minority of the total user population. Most users, even inside large organizations picked option #3 above; they chose to live (suffer?) without very good data all through the eighties and most of the nineties. How do we know? In two ways. First, taking all the RDBMS sites in the world and multiplying by the most optimistic connection counts we can imagine, we still don't end up servicing more than at most a few million users. And, secondly there's the complete explosion in sales of desktop databases. That's where we get to the irony and the opportunity.

Quite suddenly in 1993, sales of desktop databases exploded. Partly this was caused by the introduction of Windows based databases with much improved ease of use. But, surely that can't be the whole story; graphical databases on the Mac and on other platforms had existed for several years. More importantly, WANS and LANs were coming of age, and desktop databases could finally be used to access corporate data.

The problem is that what users expect and what they are getting are pretty different, at least today. Users expect to be able to easily access shared databases containing up-to-date consistent information. And, what they are getting is tools that have that latent capability but central databases that can't keep up with yesterday's demands, let alone tomorrow's. The irony is that the very tools -- desktop databases -- that can make central database truly popular and widely used are the same tools that can easily bring those databases to their knees if not managed.

Now we can put these two pieces together to see what's next. On the one hand we have the large, server based RDBMS's, servicing a relatively small population of users. Most of those users see the RDBMS through pre-built applications with compiled queries. On the other hand we have the tens of millions of desktop database users, extremely eager to access those server based db's, using tools that operate primarily with and around complex dynamic queries. What happens when we mix these two?

The dynamic queries represent a particularly key form of empowerment. Finally, as the first inventors of databases promised, true end users can formulate relatively complex questions on their own and run them against a database. It is surprising how even relatively simple Access, Fox and Paradox queries turn into very complex SQL, but users don't have to know that. Have you ever tried convincing database operational staff that thousands of users all over the organization are going to launch ad-hoc, dynamic queries against a large shared database with no pre-defined control?

Desktop databases represent empowerment, but they also represent anarchy. Our challenge is to reconcile these two polar extremes. If we can do that, very large databases indeed will become quite common, quite quickly.

Empowerment and Size

It is puzzling that there are not more really large databases around. I believe that the same challenge that forces us to choose between empowerment and anarchy stands in the way of these bigger databases. Two drivers, among others, can lead to big, big databases. One is OLAP[1] oriented and the other is OLTP oriented; we're going to consider the former here and the other a little later.

Even medium sized organizations can quickly grow pretty huge OLAP databases. The driving force is history; keeping lots of historical data, at any level of granularity, makes a database grow very quickly. The problem with doing that is justifying the cost. The cost, in turn, has two dimensions: storing the data in the first place, having the processing power to handle complex queries across large databases. The users of desktop databases are going to make both of these problems go away. First those users are going to create a huge amount of demand. And, second, implicit in their tools is a solution the processing problem.

Let's be really clear about the nature of the demand. At one time a GB was a lot of data; a huge amount of data. Today *consumer* machines routinely ship with 1.5GB desktop disks. In fact, server configurations with 1GB of main memory are becoming routine.

Users today have the ability to routinely build and manipulate temporary databases with 1-5GB of data. In fact, they can carry these databases around with them, tucked under their arm. The problem is that the original data sources with the corporate data they need either don't exist or can't be accessed.

The reason that data warehousing, OLAP, and server databases have become so hot in the last two years is exactly because users are demanding access to this type of data.

Within just a few years the personal computer software industry will be shipping ten million desktop databases (including updates) per year. There will be 50-100M desktop users in organizations worldwide, and that won't be

[1] Throughout this paper, OLAP is used to also stand for DSS, Data Warehouse, Analytic DBMS, etc.

the end. All of these users will expect access to complete historical information for all the products and customers their organizations deal with and they will expect that information to be constantly available and reasonably up-to-date. VLDB, here we come.

The Empowerment Challenge

How do we provide database access to tens of thousands of users? Are the architectures we've built today up to the task?

The proprietary hardware and software architectures described near the beginning of this paper are optimized for running a relatively small number of OLAP queries at a time. A Teradata deals best with a few large queries at a time. A Tandem, while well optimized for OLTP, and able to deploy lots of parallelism on behalf of an OLAP query, still is designed to handle only a few large OLAP queries at one time. What happens when we have to handle hundreds of simultaneous OLAP queries? Three basic ground rules change in this environment.

First the core database architecture has to be redesigned to exploit *both* the client and the server simultaneously. In a network with thousands of desktops and dozens of servers the horsepower and storage capacity of the desktops exceeds that on the servers by a wide margin. Yet, no high-capacity, high-throughput, shared database today exploits this power. OODB's have learned how to live in this world, but they aren't designed to support the desktop db's users know and love.

Second, the database needs to exploit high degrees of parallelism on the server as well. Granted Tandem and Teradata already do this moderately well, but only with specialized hardware. Clearly if we really are to serve the needs of hundreds of simultaneous OLAP users, lots of parallelism is the only way to get there.

Thirdly, and a necessary consequence of the first two ground rule changes, heterogeneity becomes the order of the day. On the desktop users will use several databases, spreadsheets, project managers, and PIM's. All of these tools will have server manifestations too. Data will include classical records, but also pictures, multimedia, geographical, and other diverse types; handling all these types of data will require a variety of storage and query managers. The database of the future will run in a variety of places, but will also consist of a wide variety of different types of components, all cooperating smoothly. This is the true meaning of empowerment without anarchy.

Reaching Outside the Organization

1995 is the year of the World Wide Web and the Internet. The Web itself, today, mostly revolves around relatively static pages, but the true meaning of this phenomenon runs much deeper.

Empowerment, tied to the business process reengineering movement pushes decision making to the periphery of very organization. The desktop databases now numbering in the millions are tools that allow newly empowered managers and workers to analyze data to make better decision. However, all that empowerment is still confined *inside* organizations. The meaning of the internet is that organizations must *reach outside themselves.*

In the beginning a manufacturing company focused on MRP- Manufacturing Requirements Planning; clearly an internal application. Next decision support systems reached out to marketing personnel. In the early nineties, notebook computers, allowed the reach to broaden out further to salespeople on the road; still inside the organization. Why stop there?

Why shouldn't a customer be able to place an order directly against the company's systems? Why shouldn't a consumer be able to make a warranty claim directly from his home computer? Home banking? Cash management workstations at customer sites? Consumers making plane and hotel reservations directly?

Many modern manufacturers and retailers – Walmart is a particularly well known example – have found extending the companies boundaries in just this way are central to the whole concept of Just in Time and process re-engineering.

Today these are no longer new ideas; the only question is whether or not they are real. When automated teller machines were first built in the early eighties, many pundits predicted the complete replacement of bank branches within two years. Just twenty four months later, faced with early failures of ATM's, the same pundits forecasted ATM's as never becoming popular. Of course, by the end of the eighties, ATM's had succeeded, and on a broader scale than anybody had originally guessed; today many banking customers never see the inside of a branch ever.

In the same way, electronic transaction services of all kinds are clearly on the way; the journey will just take a little longer than most people expect. Within ten years it's safe to assume that all the services listed above and more will be popular, commonplace, and widely used. What does this mean for OLTP systems?

Today an airline reservation system handling several thousand simultaneous users is a very big deal. However, if the same system had to handle millions of consumers, with all their questions, queries, and so forth, what would the load pattern look like?

Sports Illustrated magazine recently launched an internet server, starting with their annual swimsuit issue. In the first week, several *million* new users accessed the site. Peak load now has to be characterized in terms of hundreds of thousands of users.

These new users will tend to be both demanding and naive at the same time. On the one hand, they will have no training at all in any particularly system. As a result, graphical, easy to understand interfaces will be key. These interfaces place more, *not less*, load on the underlying OLTP system. And, on the other hand, these users over time will expect to be able to ask very complex questions of the system. As competition heats up the ability to answer these questions will become a matter of survival.

Say a customer wants to examine last year's utilities bills? Or perhaps an airline customer wants to consider seven different ways of traveling from one place to another? Perhaps the customer wants to talk to an airline computer, a hotel computer, and a restaurant computer all at the same time?

Going back to the three new ground rules for the OLAP side of the house (1. commodity, 2. parallelism, and 3. heterogeneity) , the internet forces the same ground rules to apply to the OLTP side too.

First, the internet makes it even more compelling to take advantage of desktop power than ever. Trying to handle hundreds of thousands of users without help from the desktop will be impossible. In addition, network latency will require that the desktop do as much as possible. And, tying together diverse data sources simply reinforces this need.

Second, even with the help of the desktop, parallelism is the only way one can even imagine building a server powerful enough to keep up with internet demand. Just the idea of an OLTP system serving hundreds of thousands or even millions of users requires us to think in new ways.

And, finally, heterogeneity takes on a whole new meaning on the internet. Within an organization, most users adhere to at least some standards. However, the moment we open the doors to the outside, all bets are off. Our system will be accessed from more places, with more tools, and in a wider variety of ways than we can even imagine.

The Commodity Phenomenon

While the preceding parts of this paper imply some pretty large changes in the db world, they still belie the true impact of commoditization. Millions of OLTP users is a scary goal. Thousands of OLAP users is right up there in terms of the same scariness -- Terror bytes. However, there is a quality of *sameness* that might lull us into believing that what we are talking about are bigger systems, better systems, but fundamentally the same systems. Yet, the true implication of commodity hardware is that the systems will be bigger, yes, but also very different, too. The easiest way to think about this is by thinking about where these systems will run.

Today there are about 25,000 mainframes in the world. Considering that these machines form the basis for a $75B industry, this is a shockingly small number of computers. By definition these 25,000 machines are installed at fewer than 25,000 sites.

By contrast the worldwide car industry builds about 50M cars *every year*; there are about 600M motorized vehicles running around all told. This is what happens when commodities become established.

The most amazing thing about the internet is just how *dispersed* it is. During the early eighties, MIS people were dumbfounded by how *easy* it was to acquire and run a personal computer; anybody could just do it by and for themselves. Fortunately, these machines could only serve personal needs. Well, guess what, with the internet servers are as easy to buy and install today as personal computers were in the early eighties.

With a little work, and some luck, anybody can buy and install a server for themselves. This changes the entire meaning of the server world. A home page for your office, one for your home, one for your cottage, and even one for your teenage son's bedroom!

These home pages can just as easily be databases as text pages or pictures. That is the world we are truly talking about. But, how, again, does this related to large databases?

Big, even very big, means either handling lots of data, lots of users, or both. Inexpensive family computers, costing about $2,000 now come with 1GB of disk storage. So, before the end of the decade, I could build a server with hundreds of GB's of storage for less than $25K. As soon as my database has any serious pictorial or multimedia content, it's easy to see how it can use up all that capacity quickly. But, I can get there pretty easily with conventional records too; just prowl the internet doing interesting consolidations and suddenly huge databases become routine. And, as to lots of users, for the first time in computer history, this can a *word of mouth* phenomenon. What an idea; if enough people hear about your server, perhaps put links to in their pages, etc, suddenly you, too, can have thousands of users.

Scalability, Distribution

The world we are describing can best be thought of in terms of two attributes: scalability and distribution. Scalability means that very, very big systems will be built out of very, very small systems. The Tiger video distribution system, at Microsoft, is built completely out of Gateway computers, none of which cost over $3,000. We buy tower configurations, take their bases of, and put them sideways on the shelves of a wiring closet. A system with forty of these machines is at least the equivalent of a decent sized Tandem; yet each machine is the same that is found in thousands of homes, schools and churches.

Prologic, a Canadian company, builds complete banking systems running on commodity hardware. In one case, a bank with 300 tellers replaced its central mainframe with about 25 PC based servers. Again, a fairly large system, processing dozens of transactions per second, was built up completely out of small, commodity boxes.

Scalability also means that the same software will run on a detached notebook as runs on the biggest machine. Which brings us to the point of distribution.

By the end of this decade computers will be running at several hundred million sites all over the planet. In fact, at some point there will be billions of workstations, and in the same time frame, there will be first tens, then hundreds of millions of servers. What we are talking about is a highly distributed world. Today only a few examples of such highly distributed systems exist. Of course, the internet is a good example of a non-mission critical distributed system. The world wide credit card and point of sale systems are a good example that is much more critical to the enterprises involved.

Cooperating Components

In order to support the needs of this new world, we all need to think about both databases and applications differently than do today. In fact, I claim that the change is so large as to mark the third major generation in the computer world.

During the fifties and sixties people were learning how to think about computers at all, how to program, and inventing languages and operating systems. Then in a small number of years, online storage, networks, terminals, and databases all came on the scene. Most commercial organizations are still adjusting to an application design world that revolves around the database instead of around the application itself.

The world of the 50's and 60's reached out to individual sites, typically corporate headquarters, revolved around isolated computers, and very limited numbers of users. The world of the 70's and 80's accommodated several sites in an organization, reached beyond the headquarters to large offices, handled hundreds of users, and moderately sized networks of computers and terminals. In both these worlds, computers within an organization were relatively homogeneous with a strong sense of standards de factor and de jure.

During the eighties most larger organizations worked hard to expand the reach of their computer systems to accommodate more users in more locations running more and different applications. A primary strategy for accomplishing this was the distributed database. After all if the application and the system in general revolved around the database, then clearly that database should play a leading role in helping to reach out further.

The problem with distributed databases is that they don't work: they were unmanageable, were homogeneous, and were inconsistent with organizational goals. The world described in most of this paper; a world which I view as inevitable; cannot be built on top of distributed database technology.

The goal of the distributed database is to make highly distributed systems largely transparent through the agency of the database. That is a successful distributed database system would make a collection of geographically distributed computers all look like one big centralized computer running a non-distributed database (modulo some aspects of performance). This is the goal database researchers have been working at for over two decades now.

The problem we all face is that the very large, very ubiquitous databases described early are also very distributed, and these three aspects of the system -- the size, the ubiquity, the distribution -- are all interwoven with each other. And, the essence of the problem is that the fundamental way most of us are approaching the challenge of building these systems revolves around distributed database technology.

Piece by piece, the elements of distributed databases are all important and even necessary. Distributed queries, replication, global catalogs, partitioning strategies and the like, all are still important; we should keep working on them all. It is the overall framework that has to be broader.

Heterogeneity. Of all the characteristics of the new world that have come up, it is this one that is the most key. Databases in the future cannot be monolithic. In fact, applications can't be monolithic either.

The striking thing about the World Wide Web and the Internet is the way in which components from a wide variety of sources written in a wide variety of ways all work together. But this is only the beginning.

For example the challenge of having the desktop and the server really inter-operate, each taking advantage of the other's strengths is really a challenge of cooperating components. Similarly the challenge of servers from two different companies exchanging product order information, running software written by two different developers is again a challenge of cooperating components.

Cooperating components imply that the database itself be built out of components. And, if those components are interchangeable, then desktops can work with servers; query processors for heterogeneous datatypes become possible, and data can be stored in containers of all kinds.

Cooperating components also imply that the challenge of building distributed systems does not need to be solved entirely by the database or even at the database level. The database has a role to play, but much of the interaction between distributed systems may also take place at the business rule, or process level, "above" the underlying and separated database components themselves

Components can be thought of in two different ways. In the first place, much of component thinking has to do with interfaces. Getting the interfaces right, makes it possible to substitute components for each other, providing new services within a fixed framework. Building the database itself in this way means that parts of the database become replaceable, and it is this that supports both distribution and heterogeneity. In the second place, though, thinking about an application in terms of components provides they key for building distributed systems. The components that talk to each other in a distributed system may not be substitutable. Nonetheless by conceptualizing the system in this way we gain the ability to move parts of the system around with some degree of freedom.

Very Large, Very Different, Very Relevant

Today very large database are unusual, used by only a few chosen few, and hard to build. Yet, as we move into a world of empowerment without anarchy, and organizations without boundaries, very large databases need to become common, ubiquitous and easy to build. In order for this to happen, these databases need to be very different than any we are used to today. They need to be highly scalable, very distributed, and most of all built around an architecture of cooperating components. As make this happen, VLDB will mature from a somewhat obscure specialty field, to a discipline central to the computing world of the future. We just have to be careful to be part of this transformation as it happens.