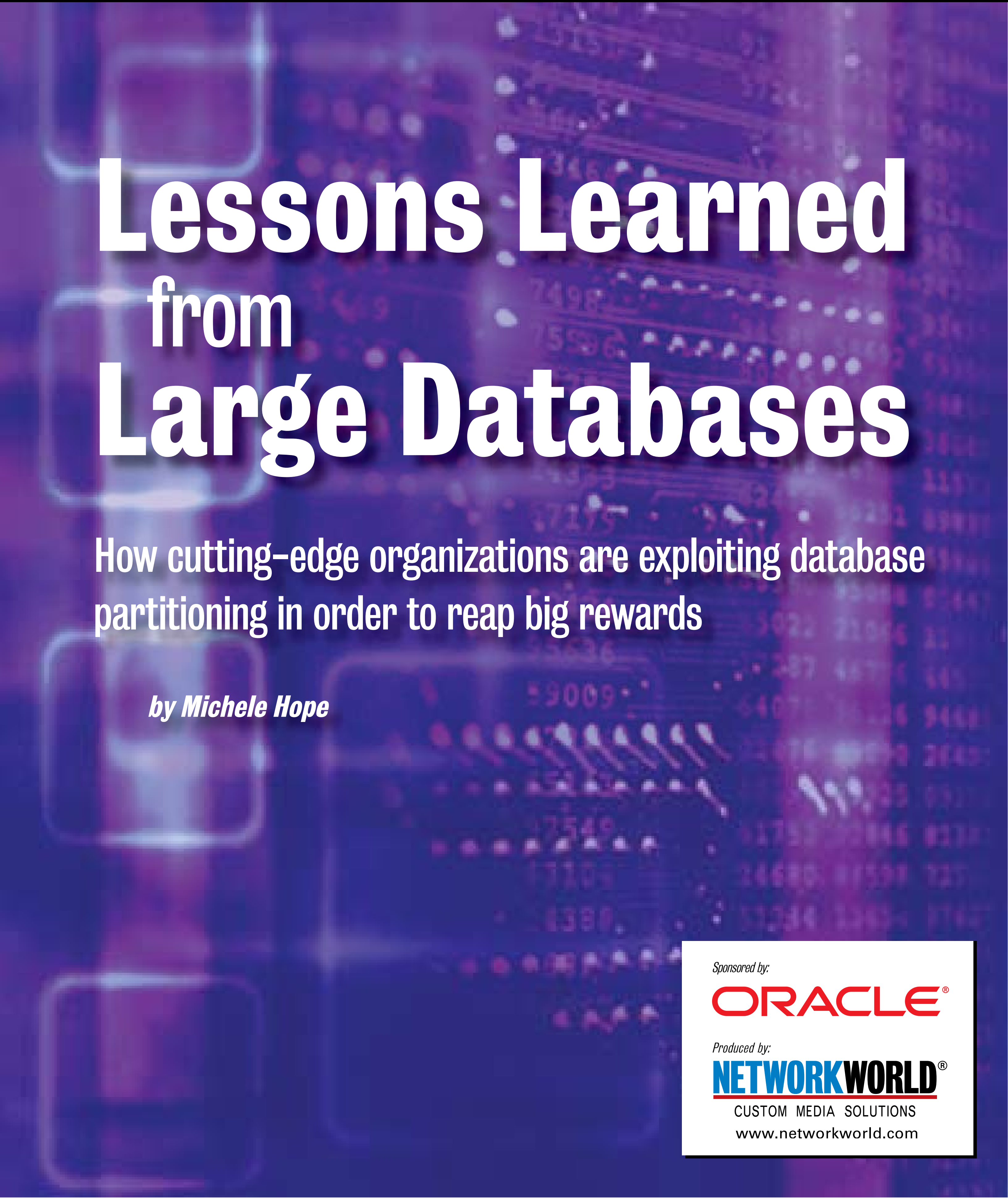
SPECIAL REPORT



Lessons Learned from Large Databases

How cutting-edge organizations are exploiting database partitioning in order to reap big rewards

by Michele Hope

Sponsored by:

ORACLE®

Produced by:

NETWORKWORLD®

CUSTOM MEDIA SOLUTIONS

www.networkworld.com

Table of Contents

Agility. Quickness. Responsiveness to subtle (and not-so-subtle) changes in the market. All are characteristics usually found in abundance within today's cutting-edge organizations. These are companies that have managed to wrestle a leadership position from competitors despite the odds, and have since been able to maintain that position through continued innovation and careful investment in the resources that have gotten them there.

A look "under the hood" of such organizations often reveals a humming IT environment that reflects many of the same characteristics of the larger whole. Typically, what you'll find upon closer inspection is a rich and complex database environment as the primary engine that propels many of the company's strategic and day-to-day decisions.

Yet, as database sizes continue to mushroom from gigabytes into multiple terabytes, it has become increasingly important to look at how well the database "engine" is designed to withstand sudden shifts in business focus, expansive growth and the ongoing strain of user demands put upon it. Economics often enters into the picture, as IT teams struggle to balance the value of making database environments faster, more available and more easy to manage, and scale against the economic reality of having to "make do" with existing resources.

As part of this balancing act, database teams have had to grapple with a number of issues, including how best to make the database design aid — not hinder — the work of multiple concurrent users submitting transactions or querying the system. Service-level agreements (SLA) often prompt database support teams to achieve strict, time-based criteria surrounding database system availability, performance and manageability. In their efforts to use costly data storage systems most efficiently, many database environments also have begun trying to define a larger information life cycle management (ILM) strategy to help them assign the most critical and frequently accessed data to the best-performing storage resources, while older, less critical data is shifted to more cost-effective "nearline" and "archive" storage systems.

Lessons can be learned and applied here from the common practices of large database environments. This special report evaluates the benefits of one such practice: the use of a somewhat little-known database feature called database partitioning that Amazon.com Data Warehouse Manager Mark Dunlap once described as the "'secret sauce' that makes the system work." Database partitioning allows a large, logical database to be divided into distinct, independent parts. This functionality, when implemented correctly, allows IT organizations to see large gains in system manageability, availability and performance.

About the Author

Michele Hope is an independent writer based in Miami. She covers enterprise IT and storage topics for a variety of private clients and IT publications, including *Network World* and *InfoStor* magazines. She can be reached at mhope@thestorage-writer.com.

Why Talk About Database Partitioning?

In the world of DBAs and software developers, much has already been written about the technical steps involved in partitioning large database tables and indices into smaller, more manageable subsets. After all, the act of partitioning databases is not new. Time-based (or "range") partitioning functionality actually has been available for more than 10 years from established enterprise database vendors.

Examples also abound of SQL code used to build or query such tables and partitions, based on various partitioning methods. These types of partitioning exercises can be found readily in everything from books on database fundamentals to database vendor documentation, such as the Oracle Data Warehousing Guide or an IBM Redbook on database strategies for DB2 UDB.

Yet, while these resources readily address the mechanics of database partitioning, they often spend less time exploring the often-instant benefits to be gained in today's real-world environments through the effective use of this feature.

Jonathan Lewis, a recognized Oracle specialist, director of the Oracle U.K. user group and author of the book *Cost-Based Oracle — Fundamentals*, has spent a lot of time looking at how organizations can make their database operations more effective. He readily offers his own caveats on the use of database partitioning in this context.

"With partitioning, you have to know what you're doing and why you're doing it," he says, making a point to reiterate one fundamental of good database design: understanding the data and the users' requirements for visiting it. "If you get it right, database partitioning can mean the difference between the system taking all day to get a job done or half an hour. It can mean the difference between supporting 10 users without partitioning or 100 users with partitioning. It's that kind of degree of difference we are talking about."

Lewis has hit upon one of the most commonly known benefits of proper database partitioning: *dramatic performance gains in database query processing times.* Often, other database performance enhancement techniques work well in conjunction with database partitioning as well, such as parallel processing and, in the case of an Oracle system, the use of materialized views.

Lewis puts it this way: "If you find out the way customers are asking queries, and you make sure you slice the data up into partitions to match those types of queries, you'll get a win. You'll get a certain 'cost-nothing, free-of-charge' performance improvement for running queries."

Noel Yuhanna, senior analyst at Forrester Research, reiterates this point. "Instead of running a COMPASS (Context-Oriented Multi-Format Portal-Aware Search System) query in an hour, partitioning can bring this type of query down to minutes," he says.

However, beyond obvious performance gains database partitioning also can bring some other, unexpected benefits to large database environments:

- **Improved manageability** for behind-the-scenes database operations (backups, optimized load processes for "rolling windows" in data warehouse environments, etc.). In some vendor implementations of database partitioning, data management procedures can be performed on an isolated partition basis, while other partitions remain available and online for production use. Data partitions also can be more easily migrated or copied to other locations.
- **Improved data availability.** With partitioned tables, scheduled downtime can be performed in isolation, only on specific partitions, without requiring a whole table to be put out of commission. Also, by splitting partitions onto separate physical tablespaces, it limits the impact of a storage hardware or disk failure to just individual partitions, as opposed to a wider effect to a large table whose entire data might suddenly become unavailable.
- **A working foundation for data life cycle management.** Organizations concerned with maximizing the investment in their current data storage infrastructure can use database partitions to identify and

easily migrate less frequently accessed or historic data from high-performing storage "tiers" to lower-cost or near-line disk storage, all transparently to the host application.

Database Partitioning: The Basics

Historically, those who have found the most benefit from database partitioning have had one or more of the following characteristics present in their environment:

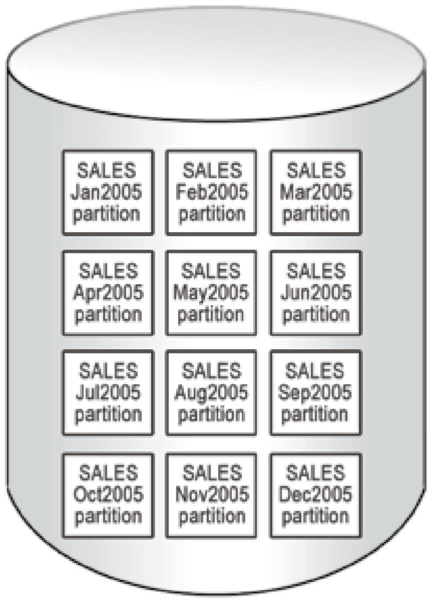
- Database systems supporting mission-critical applications.
- Systems characterized by high-volume OLTP workloads.
- Database systems supporting OLAP, DSS, data warehousing or data mining activities.
- Large database environments (more than 1TB in size).
- Database systems with large tables (more than 100GB in size) consisting of millions of rows of data.

Some proponents of database partitioning see an even broader application for its use. "In large enterprise environments, as soon as you have single objects getting larger than 1 to 10GB, partitioning can make sense," says Hermann Baer, a principal product manager for data warehousing at Oracle. Baer notes occasions where customers have been able to use partitioning to break up data in an important table and thus isolate risk that the overall table will become unavailable. Such an approach, he says, also allows for more rapid recovery of isolated partitions from various types of disruption.

Forrester's Yuhanna has seen partitioning used most of the time in data warehousing environments, where the motivation for use is fairly straightforward. "If you have a large database, you want to manage it in chunks to perform better data management. The major focus of database partitioning over the last decade has been on data warehouses and data marts, where you break down those queries into smaller units so they can run in parallel on different partitions," he says.

"It's divide and conquer. You divide the data and conquer. If you have large amounts of data running on a single processor, that query can be decomposed into smaller units that run concurrently with multiple processors, where each processor goes after a single partition of data," Yuhanna says.

Database vendors have different ways of implementing database partitioning, with some offering more ways to slice and dice the data than others. The most common database partitioning method is to split up a larger, time-based table — like a table containing sales data — into individual partitions by week, month, quarter or year. This is known as range partitioning because it allows a range of dates or numbers to be represented in the partition, with obvious start and end points for the range. The accompanying figure shows a typical sales table partitioned by month.



Sample of range partitioning in a "Sales" table that is separated into individual, monthly partitions.
Source: *Oracle Database 10g A Beginner's Guide* by Ian Abramson et Al (McGraw-Hill/Osborne 2004)

Once the partitions have been defined in a table, queries that run against the table typically are optimized to automatically "prune" unnecessary partitions out of the query process before running the query. For example, if you want to analyze all of the sales made for the last two weeks of April, your query would go to the sales table and automatically process data from just the "April" partition of the table, versus data from all 12 partitions. Users don't need to know about the partitioning structure, either. Queries use the same language, whether the table is partitioned or not. What's different is the underlying efficiency partitioning now builds into the process.

This process can result in significant reductions in processing time. Baer explains it like this: "In the most simple example of a time-based partitioning strategy partitioned by month, you might only want to look at the last month (or one-twelfth) of data in a table. That means the resources to satisfy the business request are also one-twelfth. Staying with this simplistic example, you could then theoretically run 12 times the number of users on that same hardware or make queries run 12 times faster."

One user who presented at Oracle OpenWorld put the use of partitioning in a data warehouse into more practical, real-world perspective. When asked what Oracle database features had been most beneficial for his data warehousing needs, Jay Parmar, the head of data at U.K.-based financial services provider Egg, responded as follows: "From a data warehouse perspective for Egg, parallel queries, materialized views and partitioning have been incredibly useful because we've been able to drive performance from our warehouse and provide the speed of information that my customers are asking for." Egg moved from outsourced data warehousing services with six week latencies to a near real-time, internal data warehousing service with less than 24 hours of latency. "My internal users want data even quicker and faster than ever," he said.

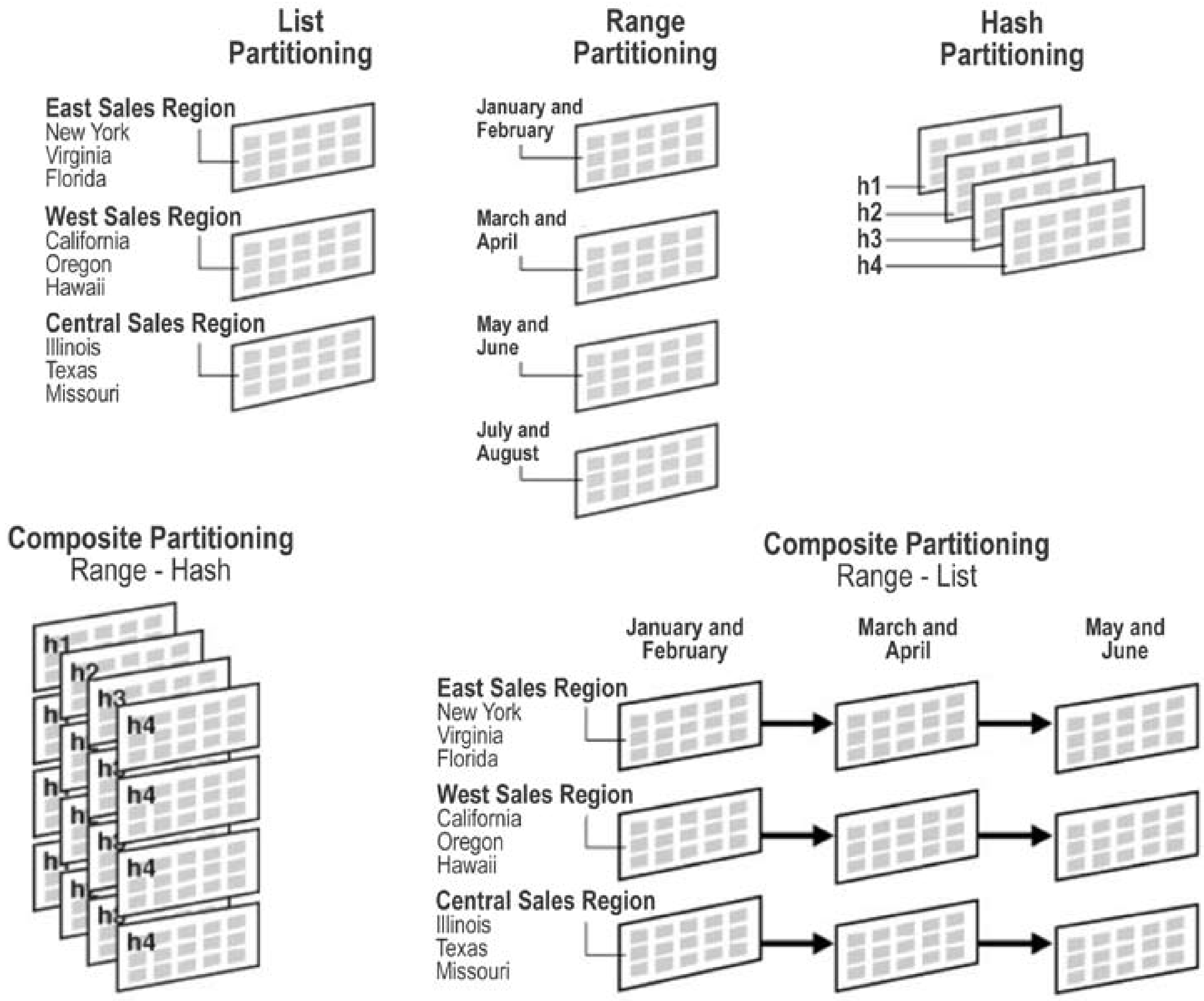
According to Baer, many organizations begin simply with range partitioning as their most common partitioning technique. This technique alone quickly yields them large gains in performance and database operations.

Going Beyond the Basics

As database environments grow and adapt to meet the changing needs of the business, they often are faced with situations where other partitioning methods may prove useful as well. Beyond range partitioning, other partitioning methods also are supported to varying degrees by database vendors as well (see figure, page 6).

These include:

- **List partitioning.** This method allows you to assign a list of named values to each partition. For example, some database tables might be able to be partitioned based on geographical lines, such as region, state, unique retail store identifier, etc. In this case, you could create a list partition for each unique region that places all sales data for the Canton, Ohio, store in one partition and sales data for the Poughkeepsie, N.Y., store in another partition. Likewise, you could partition by region.
- **Hash partitioning.** This method is often used to perform a more specific load balancing or data clustering function that evenly distributes the data in a large table into any number of partitions. Contents of each partition are not always as easy to correlate to their related business context because the partitions are not usually identified with an obvious range or list moniker, such as Sales_West or Last_Month_Sales.
- **Composite partitioning.** This method allows two partitioning methods — such as Range/Hash or Range/List — to be combined within a given table. In these scenarios, a higher-level range partition usually is established with a series of subpartitions beneath it. This type of combined partitioning functionality might be used in situations where data might need to be searched by time and region simultaneously, for example.



Samples of the various partitioning methods available.
Source: Oracle

Experts maintain that the more flexible options you offer customers when it comes to partitioning, the more likely they will be able to mix and match functionality to meet the business needs of their environment. They offer a few other partitioning scenarios to consider when exploring the various vendor implementations of database partitioning.

Scenario #1: Queries by Time and Geography

Both report queries and common database maintenance functions often can be geographically driven as well as time-driven, especially in larger environments. These might lend themselves to some type of composite partitioning strategy that combines range with list partitioning.

"Larger enterprises might not only have time as a common criteria, but also region. The usage pattern for working on Europe's data or America's data, doing things like loading or accessing data, may be different as well. When we are asleep, the guys in Asia are hopefully busy," Baer of Oracle says.

Lewis, the Oracle specialist, also notes this benefit of regionally separated partitions. "Let's say you have a sales table, partitioned by time and region [or list] into two to three regions like Europe, America, etc. If I run a query comparing Europe to America, I can just run it against the specific partitions in the table to which it applies. Then, when everyone goes to sleep in Europe at 9:00 p.m. our time, that's just when America is getting active," he says. "So, at that time, you can do maintenance on the European partitions while those in America aren't ever touched. It allows you to act as if you've got each region in its own separate table for maintenance purposes."

Scenario #2: Ease of Adding, Moving, Deleting Partitions

Many data warehousing and maintenance operations may deal with specific portions of a table. According to Baer, these tend to naturally lend themselves to the use of partitioning, which allows operations to be performed on an isolated partition-by-partition basis, without affecting user access to the rest of the table.

Partitioning has been used in this case to support what's commonly known as a "rolling window" in data warehouse environments, where new data needs to be added on a consistent basis, often via some type of ETL process, and old data simultaneously is moved off the system.

In this situation, it pays to check with your database vendor to see how easy it might be to support wholesale moves, deletes or additions of partitioned data from one tablespace or location to another. This functionality also makes it easy to move into a smarter way of utilizing expensive storage assets, by partitioning more frequently accessed partitions and pairing them with high-performance storage, while less-frequently accessed systems might be moved to larger but slower storage devices.

Lewis describes this process as follows: "For data changing very rapidly, you might decide to partition it and put that onto four times as many disk spindles and stripe the data to get the best possible performance. Then, as time passes and the data becomes older and no longer critically high or active data, you might decide to copy it down from one place to another to get to the slower disk. How do you actually say, 'Here's this 30GB chunk of data, now going from high-speed, busy and active to the lower- speed, less-interested bit?' How do you move that? You want the chance to rearrange it. That's where partitioning helps."

Because your mileage may vary in these wider applications of partitioning, depending on the database vendor package you are using, it pays to check first with the vendor about how far it may go in supporting some of these more advanced partitioning scenarios.

When Smaller is Better

The concept of database partitioning is not accompanied by much sizzle or flash, but is something savvy environments readily embrace as a critical design component for growth management and streamlining even the most critical and intensive database operations.

Despite its effectiveness in high-end settings, the current use of this functionality presents a strange conundrum. Those "in the know" take it for granted as a sound design practice they would not think to exclude when building or supporting data warehouse and decision support environments. The Data Warehousing Information Center, supported by Larry Greenfield of LGI Systems, lists partitioning as "probably the second most common method of speeding up queries." The center also goes on to suggest that anyone wanting to learn about data warehousing should take care to start learning about some "fundamental technical topics" that include "how database partitioning can be used in conjunction with logical structures."

Likewise, author Scott Ambler of Ambysoft, in his book *Agile Database Techniques*, published by John Wiley and Sons, recommends table partitioning as one design strategy to help support more "performance-friendly" reporting.

Customers that have cut their teeth on the process and seen its results are quick to note the dramatic changes it's made in their own operations. Brian Kalmar, a solution architect at national food retailer Giant Eagle, noted recently in an interview at Oracle OpenWorld that database partitioning had been one of two essentials to achieving the levels of performance required from the company's four Oracle-based data warehouses.

"We run our machines very hard. We throw a large number of parallel processes at the machines and we try to get our data back as quickly as possible because it's critical for our business decisions," Kalmar says. "Partitioning and parallelism are the keys to data warehouse performance. Breaking out a table that's billions and billions of rows into smaller chunks allows us to get at the data quickly. Parallelism, throwing a number of child worker processes at a SQL query or

Partitioning in Action at Amazon.com

A few years back, the WinterCorp, a large database consulting and market research firm, produced a report (click here), "Field Experience with Large-Scale Data Warehousing on Oracle," that outlined the experiences of several large-scale Oracle data warehousing customers.

One company it profiled was Amazon.com, which even then was heralded as one of the largest and fastest-growing businesses on the Web. At the time, the company's online inventory of books and other products already amounted to millions of items, being purchased by millions of customers in more than 200 countries around the world.

Even then, Amazon's large data warehouse environment already had mounted to 10TB of user data and was growing at a phenomenal rate of 100% each year. Average user load was somewhere between 55 and 60 concurrent users, while peak query load reached somewhere in the range of 4,300 queries per day. The largest of 600 Oracle tables at the time contained more than 10 billion rows of Web click stream data. (The most recent VLDB Survey from WinterCorp showed that Amazon's data warehouse had since grown to 25TB by 2005.)

As part of their initial research, WinterCorp Founder and President Richard Winter joined with WinterCorp Vice President of Engineering Rick Burns to interview Amazon Data Warehouse Program Manager Mark Dunlap. During the interview, they asked how he had been able to support the needs of such a high-performance Oracle system. Following is an excerpt of some of their findings.

"Mr. Dunlap identified Oracle's ability to support a high level of concurrency as a key to the success of the Amazon warehouse. Amazon also relies extensively on Oracle partitioning, external tables, the SQL MERGE statement, and Oracle's analytical and windowing functions. . .The warehouse environment is supported by fewer than two DBA full-time equivalents."

The report goes on to note the following regarding Amazon's use of database partitioning:

"Mr. Dunlap also credits Oracle composite partitioning with contributing in a major way to performance. First, the many subpartitions enable highly parallel query. Second, the time-based major partitions create a 'free' index. The Oracle optimizer uses the information in the partition key to avoid accessing partitions that cannot contain data relevant to a given query. Since many queries pertain to the most recent month, week or day of data, the partition elimination is a powerful optimization. Mr. Dunlap views partitioning as the 'secret sauce' that makes the system work."

Interestingly enough, Amazon also credited partitioning with allowing a more trouble-free migration of its large data warehouse from one physical data center and hardware platform to another. This involved moving the database across country to another Oracle instance running on a different hardware platform. The WinterCorp report stated the following about this experience:

"The data was actually exported from the existing Oracle system, transferred via high-speed network to the new data center, and imported into the new Oracle system without ever writing it into a flat file. This was possible because of partitioning features of Oracle, in which partitions of the database can be independently moved from one system to another. About five partitions were moved in each export operation. The migration occurred with essentially no disruption of user activities and was virtually transparent to end-users."

Amazon presents a good case of demanding customer needs that resoundingly met their match with the proper use of database partitioning.

Source: WinterCorp.

DML statement, allows us to get the data much quicker also."

Yet, despite such widespread support for partitioning, Yuhanna estimates that only 10% of companies currently use it. Why? Is it because its benefits are often carefully buried in the depths of operating manuals and SQL code snippets? Or, because companies simply don't realize this type of functionality is already available to varying degrees within most database vendors' enterprise-level offerings?

More likely, one culprit behind its slow adoption is the fact that most organizations simply do not have the luxury of looking "under the hood" of high-end database environments to see what really makes them tick. This report offers a high-level glimpse into the benefits such environments have achieved through the simple application of one operational feature like database partitioning. The ability to support more users, more processing and more back-end operations without a performance hit is almost always the end result of their efforts.

Conclusion

With the number of multi-terabyte installations expected to double in the next few years, and annual data growth approaching somewhere between 30% and 100%, partitioning the database to "divide and conquer" is a process many of today's cutting-edge organizations already have accepted without question. "There is not a single large customer we speak with who is not using partitioning for data warehousing," Baer says. "Their automatic assumption is, 'If I have large objects, I'll need partitioning.'"

The performance benefits and effectiveness in scaling such partition-minded systems are probably the biggest reasons organizations tend to purchase this functionality, he says. Baer also says it's only a matter of time before other enterprise organizations also will be attracted to this unsung feature for many of the same reasons.

Whichever way you approach the equation, the end result of effective partitioning boils down to one simple truth: To make something big (and getting bigger) act as if it were small. That seems as good a building block as any toward achieving a more agile, adaptive IT environment.