DBMS
DATABASE & CLIENT/SERVER SOLUTIONS

■ SERVER SIDE ■

BY DAN BULOS

How to Evaluate OLAP Servers

A COMPREHENSIVE CHECKLIST FOR EVALUATING THE LATEST OLAP SERVER PRODUCTS.

If you haven't heard of online analytical processing (OLAP) by now, chances are you will soon. Several recent articles have defined OLAP as a technology and examined the products in this market space. (See "The Truth about OLAP," on page 40 of this issue.) This column discusses the pragmatic issue of developing a framework for evaluating OLAP servers.

Before evaluating the specific features of various OLAP servers, you must first determine the mission of analytical processing within your organization. OLAP applications are not focused on data management; rather, an OLAP server is a calculation engine that creates new information from existing data through transformations and formulas. This is in contrast to online transaction processing (OLTP) applications, in which the database collects data for control purposes. It is also different from data warehousing applications, in which the database stores the data for query and distribution purposes.

In this column, I discuss eight categories of features that you can use as a framework for evaluating OLAP servers: database architecture, calculations, query language, back-end integration, front-end integration, security, server development environment, and performance. I have also provided a quick reference checklist (see Table 1, page 98) that you can use for your OLAP server product evaluation.

Determining what information you need to create from your data and which business problems you need to solve will help guide you in the OLAP selection process. In the OLAP arena, technical features are important only if they help provide the information you need to answer strategic questions about the direction of your business. The criteria in Table 1 are targeted at OLAP servers only. OLAP client software is a huge topic in its own right and outside the scope of this article. The evaluation criteria are independent of whether you use relational or multidimensional database technology to implement the OLAP server.

Database Architecture

Whether you use relational or multidimensional database technology, an OLAP server provides a multidimensional view of data. The basic units of storage in a multidimensional view are dimensions (made up of members) and cubes (made up of dimensions).

Capacity. While it would be nice, it is unrealistic to require an OLAP server to have an unlimited number of dimensions, each with an unlimited number of members, all contained within a database of unlimited size. The important issue is whether the OLAP server has sufficient capacity to represent the business problems you need to solve.

Data Types. Dimension members are either text or calendar period data types. Cubes are generally numeric data types because analytical processing is a numerical operation. This does not mean that other data types are irrelevant to OLAP servers. Other data types, such as dates, binary large objects (BLOBS), or text strings may also be required in an OLAP system. For most OLAP applications, support for text strings is usually the most important of these complex data types. A simple example of a text string is dimension labels. A member of the product dimension might be a stock keeping unit (SKU) represented as a numeric value. A long description of that SKU might be more appropriate for reporting purposes.

You can also use text strings multidimensionally. For example, a planner might be required to enter a written justification of a sales forecast by product and customer. This would require a free-form text field.

Multiuser Write Access. If your OLAP applications are not read-only, you must evaluate how the server implements multiuser write access. The two key functions to evaluate are the locking mechanism and read consistency. The characteristics of an update in OLTP are very different than those in OLAP. OLTP is characterized by a large volume of small transactions. While OLAP generally has fewer transactions, the ripple effect of database recalculation causes each transaction to affect a larger part of the database. When a user writes to an OLAP database, not only do the input values change, but any stored derived values that depend on the input values must be recalculated. This increases the scope of the lock and affects which portion of the database is inaccessible during the write. How the server maintains read consistency is important not only during the write but during the solve process (when the server calculates derived values).

Access to Multiple Databases. OLAP applications are problem-oriented. Rather than having a single enterprise-wide database, companies generally implement OLAP databases at the departmental level or to address a specific cross-functional issue (such as pricing or cost allocations). However, the fact that marketing has one database and finance has another does not mean that the two databases do not need to interact. Information derived in one database can be important to the other. How the server facilitates information sharing among databases may be important for your particular application.

Database Reorganization. Some databases are sensitive to structural changes. The addition and deletion of objects may require a reorganization of the database. The frequency and duration of reorganizations can require DBAs to bring the database

.....

Dan Bulos is president of Symmetry Corp., an OLAP consulting firm based in San Rafael, California. He has 17 years of experience working with OLAP systems. You can reach Dan via the Internet at bulos@ix.netcom.com.

off-line for maintenance. How long the database is unavailable during these maintenance periods will have an impact on the user community.

Calculations

The ability to perform calculations is the heart of any OLAP server. The management and maintenance of calculations are crucial for effective problem-solving. An OLAP server that simply gives you the ability to write program code in order to perform the calculations is insufficient. The server should have objects (such as hierarchies and models) that store the calculation logic, and routines that execute the calculations based on the objects. You should not consider OLAP servers that do not have a strong and flexible calculation paradigm. I discuss the key calculation features in the next few sections.

Hierarchies. To manage large amounts of data, OLAP servers aggregate data along hierarchies. Not only do hierarchies provide a mechanism for aggregating data, they also provide a technique for navigation. The ability to navigate data by zooming in and out of detail is key.

Not every OLAP application looks at data the same way. A financial application may have a geographical and legal entity consolidation. A book publisher's product hierarchy may be based on category (mystery, suspense, sci-fi, and so on), but may also need to classify products by type (hardcovers, trades, and paperbacks). The first example demonstrates a need for multiple hierarchies (geographical and legal) applied to the same dimension (organization). The second example demonstrates the need for a single hierarchy in which a member (book) can have multiple parents (category, type).

Some hierarchies need to be unbalanced. This means that there can be deeper levels of aggregation in some sections of the hierarchy than in others. A retailer might have more product categories in the hardware department than in the appliances department, for example.

An OLAP server must be able to do more than just sum data along hierarchies. Maximum, minimum, average, and percent of total (or any higher level aggregate) are just some examples of functions that can be applied to hierarchies. To promote flexibility, an OLAP server should supply other functions for navigating the hierarchy. These functions are parents, children, ancestors, descendants, siblings, tops, and bottoms. In addition, an OLAP server should be able to combine these functions using set operators (such as union and intersection) to perform calculations on subsets.

An important feature of hierarchies, particularly in planning systems, is the ability to accept and process data at any level. For example, a packaged goods company collects actual sales by universal product code (UPC), but might forecast sales at the UPC level for one product line, at the product class level for another product line, and at the product group level for another product line. OLAP servers that always start the aggregation at the lowest level (UPC) would require that an arbitrary UPC be selected to store the forecast for the product group or product class. This is an adequate yet unsatisfying solution. Comparing actual versus forecast at the UPC level when the forecast is performed at the product class level produces meaningless results.

Time Calculations. Most OLAP systems have a time dimension. How the OLAP server manages and uses a time dimension is critical to the successful implementation of your OLAP application. Time is a very special dimension because it is sequential; that is, January always comes before February. Rarely is data sorted across time (seasonal analysis is an exception). Many analytical queries are based on comparing time periods: sales this period versus the previous period, or versus the same period last year. An OLAP server should have as much calendar knowledge as possible without locking you into a specific view of the calendar.

The calendar was not designed for analytical processing. Months do not all have the same number of days; weeks do not neatly roll up into months. (To overcome this problem, many manufacturers have a year made up of 13 four-week periods.) Once a calendar is defined, the OLAP server must be able to perform calculations on the data. Computing a more aggregate period (quarters) from a less aggregate period (months) is not always straightforward. While most data is simply summed, many measures require other functions. For example, the number of employees in Q1 is not the sum of the number of employees in January, February, and March. Some companies decide to use an ending balance, others use an average balance. The sequential nature of the time dimension requires OLAP servers to understand time attributes such as the first and last period in an aggregate period. Other calculations an OLAP server should be able to perform include time-series functions such as year-to-date, moving averages, and moving totals.

Models. Not all the calculations required of an OLAP server are hierarchical aggregations. Much of the data is derived from formulas (which are sometimes very complex). A group of formulas is called a model. When you are evaluating support for other calculations in OLAP applications, remember that models are necessary for marketing applications as well as financial applications. Calculating inventory turnovers for a retailer can be as complex as calculating return-on-assets. The OLAP server you choose should have a rich library of functions (financial, marketing, and algebraic). Keeping track of the dependencies among formulas should not be the application designer or user's responsibility. Rather, the server should have an automatic ordering ability to ensure the proper calculation of formulas.

While it is rare that an application uses simultaneous equations in its model, certain classes of applications (such as capital budgeting) require it. This should not be an automatic checklist item. Investigate whether your organization needs this feature.

Dimension Transformations. The transformation of one dimension into another dimension is a useful facility for an OLAP server. Certain business problems require that data dimensioned by one set of dimensions be mapped to data that is dimensioned by a different set of dimensions. For example, a company that has purchased a new subsidiary might need to map the new subsidiary's general ledger accounts to the company's own general ledger.

Database Recalculation. OLAP databases can get quite large and the relationships among derived members can get very complex. The nature of OLAP systems is that data is added incrementally (usually by calendar period). If the OLAP server does not support minimal recalculation, either the entire database must be recalculated or you must write application code to calculate the appropriate portion of the database. You must write different code for different situations. You must also maintain this code when there are structural changes to the database. An OLAP server that performs minimal recalculation saves time and frustration.

Streaming Calculations. The strategy of computing and storing every possible derived value can cause OLAP databases to grow to multigigabyte sizes. On the other hand, minimizing storage requirements by computing every derived value at query time can slow database server response time. In most applications, a mixed strategy is best. Values that are either accessed often or computed from a large set of values (for example, worldwide total sales for all products and customers) are best calculated and stored. Values that are seldom accessed or computed from a small number of values are often best calculated at query time. Forecast variance, for example, might depend on two values only (forecast and actuals) and it's unlikely that every combination of product and customer variance would be accessed.

Another strategy for balancing storage requirements versus

TABLE 1. OLAP Server Product Evaluation Checklist

(1) FEATURE WEIGHT	(2) PRODUCT FEATURES	(3) FEATURE RANK	(4) FEATURE SCORE

	Database Architecture
	• Capacity
	• Data types
	  • Text strings
	  • Dates
	  • BLOBs
	• Multiuser write access
	• Access to multiple databases
	• Database reorganization		

	Calculations
	• Hierarchies
	  • Structures
	  • Functions
	  • Navigation
	  • Can process data at any level
	• Time
	  • Understands first and last period in an aggregate period
	  • Year-to-date
	  • Moving averages
	  • Moving totals
	• Models
	  • Auditing and debugging
	  • Financial functions
	  • Marketing functions
	  • Algebraic functions
	  • Automatic ordering
	  • Simultaneous equations
	• Dimension Transformations
	  • "Tables, program code"
	• Database recalculation
	• Streaming calculations		

	Query Language • Subsets based on data values • Subsets based on hierarchies • Subsets based on models • Subsets based on time dimension • Sorts children of a parent within their parent • Heterogeneous granularity		

	Back-End Integration
	• Data loading
	  • Reads legacy data
	  • Can select data directly from an RDBMS
	  • Can directly read spreadsheets
	  • Can specify the range of spreadsheet cells to be read
	  • Provides logging of rejected records
	  • Manages dimension domains
	  • Externally maintains hierarchy parentage relationships
	  • Can decode a field value in a data source into a dimension member
	• SQL drill-through
	  • Supported through embedded SQL
	  • Maps the data warehouse and OLAP server dictionaries
	  • Provides administrative tools to manage dictionary mapping		

TABLE 1. OLAP Server Product Evaluation Checklist

(1) FEATURE WEIGHT	(2) PRODUCT FEATURES	(3) FEATURE RANK	(4) FEATURE SCORE

	Database Architecture
	• Capacity
	• Data types
	  • Text strings
	  • Dates
	  • BLOBs
	• Multiuser write access
	• Access to multiple databases
	• Database reorganization		

	Calculations
	• Hierarchies
	  • Structures
	  • Functions
	  • Navigation
	  • Can process data at any level
	• Time
	  • Understands first and last period in an aggregate period
	  • Year-to-date
	  • Moving averages
	  • Moving totals
	• Models
	  • Auditing and debugging
	  • Financial functions
	  • Marketing functions
	  • Algebraic functions
	  • Automatic ordering
	  • Simultaneous equations
	• Dimension Transformations
	  • "Tables, program code"
	• Database recalculation
	• Streaming calculations		

	Query Language • Subsets based on data values • Subsets based on hierarchies • Subsets based on models • Subsets based on time dimension • Sorts children of a parent within their parent • Heterogeneous granularity		

	Back-End Integration
	• Data loading
	  • Reads legacy data
	  • Can select data directly from an RDBMS
	  • Can directly read spreadsheets
	  • Can specify the range of spreadsheet cells to be read
	  • Provides logging of rejected records
	  • Manages dimension domains
	  • Externally maintains hierarchy parentage relationships
	  • Can decode a field value in a data source into a dimension member
	• SQL drill-through
	  • Supported through embedded SQL
	  • Maps the data warehouse and OLAP server dictionaries
	  • Provides administrative tools to manage dictionary mapping		

ucts within departments will produce a report with all the departmental sales at the top and the individual product sales at the bottom. This is not the desired result. The children of a parent (products) should be sorted within their parent (department).

Analysis cannot occur in a vacuum: Analytical queries require a contextual framework for the analysis. Detailed data is often compared with aggregates of associated or nearby data to provide a perspective. The ideal OLAP server query language has a high degree of heterogeneous granularity, which is the ability to embody varying levels of hierarchical detail in a single query. This can be along a single dimension such as time (query the sales forecast by month for the current year, by quarter for next year) or along multiple dimensions such as vendor and geography (query sales by individual store and retail chain in California as well as by all retail chains by state). This approach empowers a single statement to perform the work of several statements in a less powerful query language.

Back-End Integration

To paraphrase a famous quote, "No system is an island." Interoperability is crucial to providing an OLAP system that supports the information needs of business users. Therefore, OLAP applications must be integrated with the enterprise's overall database architecture. This means that users must be able to retrieve and process data from a variety of data sources. The effort required to incorporate new data into an OLAP application directly affects its responsiveness to changing business requirements.

Data Loading. The primary function of OLAP applications is not data collection, but data analysis. OLAP applications are typically bulk-loaded from other sources on a periodic basis. Basic data sources include ASCII data files (from in-house legacy systems or purchased from external sources), SQL databases, and spreadsheets. The ease of loading data and the ability to process differing layouts of data seriously affect an application's initial development time and its ongoing responsiveness to changing requirements.

Most corporate data resides in legacy systems. Extracting data from these systems can be a tedious and painful experience. Many legacy systems can dump data in a single format only, which may not be readable by all OLAP servers. For example, many legacy systems produce data files with packed, zoned, or over-punched numeric fields. Some systems produce a file with a transaction code field to distinguish the differing record layouts in the same file. If your OLAP server cannot read or process the data in the format supported by your legacy system, you will need to write an intermediate program to digest the data and produce a readable data file for the OLAP server. If your company has put the effort into replacing legacy systems with RDBMS systems, the OLAP server should be able to take advantage of them by selecting data directly from an RDBMS table or view.

With spreadsheets becoming a growing source of corporate data, selecting an OLAP server that can directly read spreadsheets can also save you time and effort during data loading. Remember, spreadsheets generally have a lot of disparate data on a single sheet. The data load facility should let you specify the range of cells to be read.

Regardless of the data source, the data-loading facility should include the following four features. First, it should provide logging of rejected records (with or without explanations). A log makes it easier to pinpoint problems and reprocess any rejected records. Next, the facility should also be able to manage dimension domains, as well as add and delete dimension members. Third, the data loading facility should be able to maintain hierarchy parentage relationships externally (for example, what region the Jersey City branch rolls into). And fourth, you should be able to decode a field value in a data source into a dimension member when loading data, or you will be forced to preprocess the data. For example, you should be able to take into account that one source system calls the sales office in New York "NY" and another system calls the office "N.Y."

SQL Drill-Through. If a data warehouse is part of your enterprise's systems architecture, SQL drill-through is a critical OLAP feature. A data warehouse reduces data duplication and gives an application access to transaction detail that is not appropriate for an OLAP database. SQL drill-through gives the OLAP server access to data in a data warehouse on an as-needed basis.

The most common way to provide SQL drill-through is via embedded SQL. This enables application developers to program the appropriate SELECT statement for a given situation. While this is a competent solution, it requires that the application designer anticipate every given situation. This is a high maintenance solution.

A better implementation of SQL drill-through maps the data warehouse dictionary and the OLAP server dictionary, providing a greater degree of flexibility and responsiveness. The mapping can exist in either the data warehouse or the OLAP server. This implementation allows the OLAP server to generate the required SQL statement without either developer or user intervention. Administrative tools should be available to manage the dictionary mapping regardless of where it is implemented.

Front-End Integration

Today's IT environment is highly heterogeneous. Different users have different needs and use different client tools. To be successful, an OLAP server must interface with a variety of client applications. Depending on the OLAP application, bringing data to the desktop can be a matter of developing a custom application or buying an off-the-shelf spreadsheet, query tool, or report writer. The amount of effort required to bring analytical processing information to a desktop has an impact on the application's success, both with the development staff and the user community.

Spreadsheets. The analytical tool on virtually every user's desktop is the spreadsheet. Most users should be able to incorporate data from the OLAP server into their personal spreadsheets regardless of whether the spreadsheet is their primary navigation tool.

OLAP Proprietary Tools. Because of their strategic nature, many OLAP systems require a custom development effort to target an application to a specific problem. Budgeting and executive information systems are two examples of OLAP applications that are generally custom projects. Many OLAP vendors have developed proprietary client applications. The positive aspect of a proprietary client is that the level of integration between the client and the server is very high. If the proprietary client is the only information delivery vehicle available from a given vendor, however, you are locked into a single front end.

Third-Party Tools. Most OLAP servers have a published proprietary applications programming interface (API) for interfacing with client applications. While proprietary APIs are adequate interfaces because of the detailed coding required to use each different API, in-house developers and third-party tools vendors may find it tedious to interface their client applications to each OLAP vendor's proprietary API. Because of this, there are few off-the-shelf third-party tools that interface to OLAP servers. (However, a new class of query tools is appearing on the market. These "cube navigators" are specifically designed for OLAP applications and include products such as Andyne's Pablo and Cognos' Powerplay.)

Some OLAP server vendors have developed or are in the process of developing plug-in components to facilitate interfacing client applications to their servers. "Componentware" isolates developers from the intricacies of a proprietary API, thereby increasing application developer productivity. Examples of OLAP-based componentware include Microsoft Visual Basic's custom controls, OLE containers, and Powersoft PowerBuilder's class li-

braries. Because it is almost impossible to predict what tools your users will want to use, the key to successful front-end integration is to select an OLAP vendor that has made a commitment to working with other vendors to provide a spectrum of choices.

Security

While the strategic nature of OLAP systems frequently makes OLAP data highly confidential, security is not as much of an issue for OLAP systems as for OLTP systems. OLAP security typically involves access to the entire database, not access to specific information within a database. In an OLTP environment, there are "more fingers in the pie"; that is, a greater population is introducing data into the OLTP system than in OLAP. However, in OLAP applications such as budgeting, where a large population of users is introducing data into the system, the OLAP server should be able to restrict access by user to the cell level. The security definition and management facility should be able to leverage the hierarchical relationships defined in a given OLAP application. For example, in an application that has a hierarchy of organizations by geographical region, you might want to limit access of the western region's data to users in the western region.

Server Development Environment

A productive development environment lets developers bring applications online quickly and perform system maintenance with greater reliability. OLAP servers contain strategic information about the enterprise, and the timely delivery of this information may be as mission-critical as for any OLTP application.

Most OLAP servers have some sort of programming environment — a 4GL, a stored procedure language, or a scripting language. Important features in the programming language include argument passing, local variables, looping constructs, and control branching. Application maintenance can be a more difficult job than the initial development of an OLAP application because of ongoing changes. To help manage application changes, an OLAP server should provide good cross-referencing tools.

In addition to cross-referencing programs, you should be able to cross-reference database objects that control processing (such as hierarchies and models). You should be able to examine the impact of adding, deleting, or renaming dimensions or their members before you actually make the change. For example, you might want to see what other derived data will be affected by changing how the average selling price is calculated.

Performance

While you may be tempted to use performance as an "objective" measure for selecting an OLAP server, remember: Unlike hardware systems or database servers used for OLTP, OLAP systems are about effectiveness (how to make better decisions), not efficiency (how to make faster decisions). In contrast to OLTP applications, where the database size grows as a function of transaction volume, OLAP applications grow as a function of time periods stored. OLAP applications do not grow by factors of three or four without a major change in scope. For example, if business volume doubles, it will double the size of the OLTP database, but it will barely affect the size of the OLAP database because the number of products and customers has not changed. An OLAP server's performance does not have to be the best, only sufficient. For example, if the general ledger closes on Tuesday night and financial performance reports must be ready on Wednesday morning, does it really matter whether the database took three hours or four hours to process the information? The performance of any database system is sensitive to database design. An OLAP server that has tuning tools will let you improve the performance with fewer iterations and less guesswork.

Final Reminder

Keep your business goals firmly in mind as you evaluate OLAP servers. Remember that technical features are important only if they help provide the information you need to answer strategic questions about the direction of your business. ■

Reprinted with permission from **DBMS** Magazine, August, 1995, Vol. 8, No. 9