Analytical Processing: A comparison of multidimensional and SQL-based approaches

Hyperion®

Contents

What is Analytical Processing? 1

Comparing SQL-based and Multidimensional Analytical Processing 3

Analytical Advantages of Multidimensional Databases 5

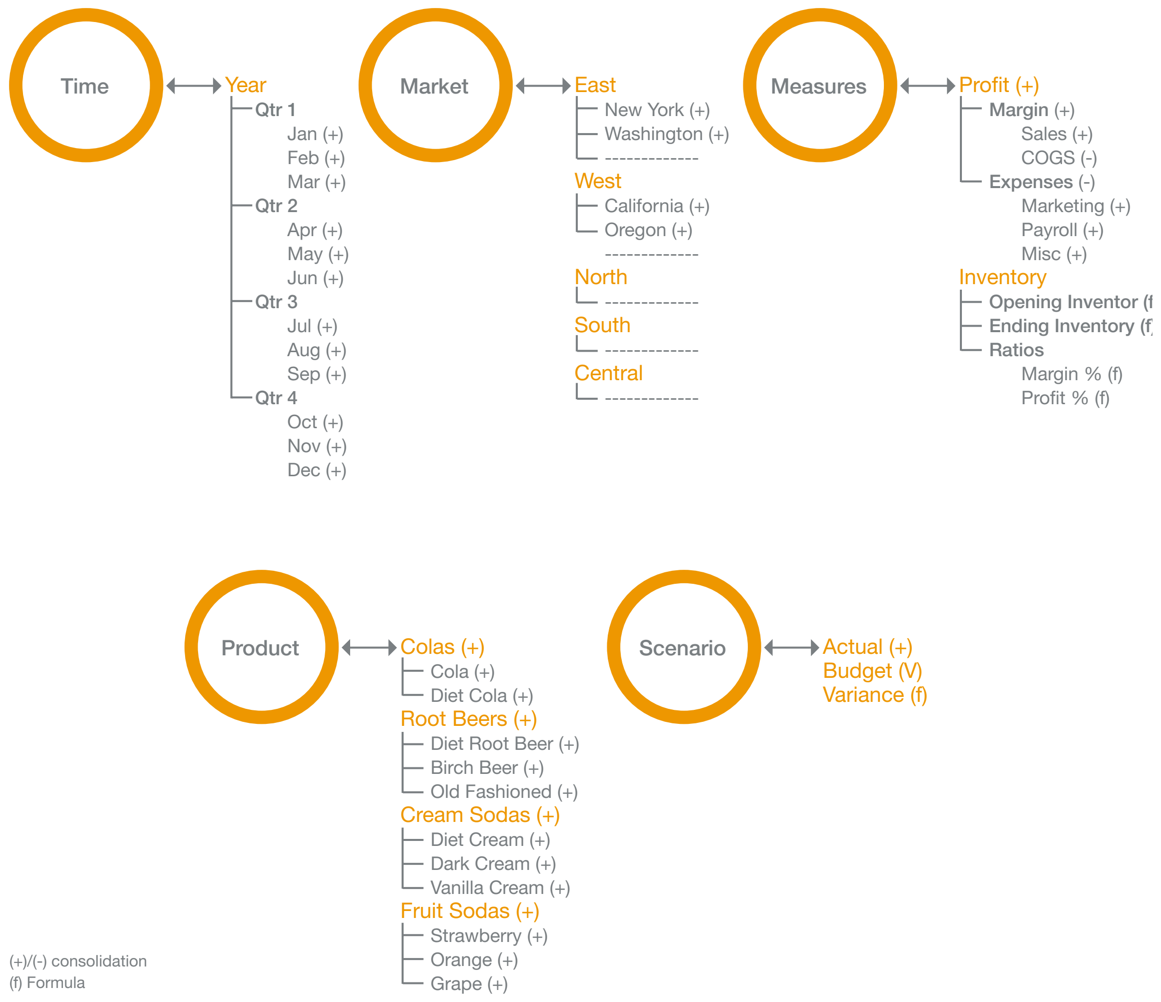
- Summarization 5
- Native Support for Business Analysis 7
- Richer Analysis Through Sophisticated Calculations 7
- Iterative Analysis 9
- Real-Time Analysis 9

Conclusion 10

Relational database management systems have matured into powerful transaction processing solutions capable of handling volumes of data in the terabytes. With this explosive growth in the size of databases came the need for sophisticated analysis techniques to synthesize raw data into actionable information. To satisfy these requirements, multidimensional databases such as Hyperion® Essbase® OLAP Server evolved to provide a platform optimized for analytical, rather than transactional, processing. This paper discusses analytical processing, comparing SQL-based analysis with multidimensional systems and discussing the limitations of recent SQL extensions intended to incorporate OLAP (online analytical processing) functionality.

What is Analytical Processing?

Consider a soft drink manufacturer that wants to analyze financial data to better understand current performance and identify areas of opportunity. The basis for analysis is sales and expense data for each product sold, during each month of the year, in each state in the U.S. Numerical items like sales and cost of goods sold are referred to as measures, and ways of looking at these items—such as by product, by market and by time—are called dimensions.



A typical business view showing how financial data is organized for analytical processing.

The company has a number of processing requirements to support its analysis needs, which are:

Summarizations. Data for analysis is generally loaded at the lowest level of detail. In the figure, for example, Sales, Cost of Goods Sold (COGS), Marketing, Payroll and Miscellaneous Expenses are extracted from transactional systems for each individual product, in each month and state. The company needs to be able to analyze data at various hierarchical levels of detail, so states "roll up" into regions, time into quarters and products into categories. Measures can also be hierarchical—for example, Sales and COGS "roll up" into Margin. Additive summarizations like this are also referred to as consolidations. Nonadditive summarizations such as *Profit = Margin – Expenses* or *Margin% = Margin/Sales* are also common.

Native Support for Business Analysis. Beyond simple summarization, there are a number of more sophisticated analytical tasks that are quite common:

- Time-balance computations. Summarizing sales values for each quarter (a simple aggregation) is different from summarizing inventories, which requires an awareness of the relationship between different time periods. For example, the quarterly value for Opening Inventory is normally equal to the Opening Inventory value for January, while Ending Inventory for a particular year is the same as the Ending Inventory for the last quarter. These computations should be performed transparently at every level within time dimensions.

- Expense reporting: The ability to track actual and budgeted measures and monitor the variance between them is a fundamental business requirement. Budgeted measures represent expenses, but are typically analyzed as positive quantities rather than debits. Tagging such measures as expenses allows an intuitive handling of variance reporting.

- Automatic conversion from one currency denomination to another. For companies with global operations, creating a single business view of performance requires the ability to integrate financial data in multiple local currencies.

- Time series computations, such as year-to-date sales, and complex aggregations involving rankings, moving ratios, median values, standard deviation, etc.

Richer Analysis Through Sophisticated Calculations. Hierarchies are often insufficient to model business relationships. For instance, while the regional sales for a product is the sum of sales across all states in the region, the same may not be true for payroll or budget measures. Payroll can be a function of inflation and cost of living indices, while budget may be a constant ratio of a product's sales performance within a region. An analytical processing solution must allow for complex calculations with a rich range of mathematical, financial, statistical and forecasting functions.

Iterative Analysis. While the data to be analyzed is fairly static, the analysis process itself is iterative and must allow for unlimited "what-if" scenarios. This involves changing data relationships or updating values to determine how various changes would affect the bottom line. These user-defined changes must be propagated to summaries in real time in order to allow accurate, ad hoc analysis.

Real-Time Responses. Support for a combination of static and dynamic computations—both simple consolidations and sophisticated calculations—is essential. Pre-computing all possible summaries improves query performance significantly, but can increase the window when information is unavailable.

Comparing SQL-based and Multidimensional Analytical Processing

Although the individual transaction details that support analysis may come from relational database management systems, the business view analysts use is a *multidimensional* description of information. Numerical items or *measures* (e.g., sales) are described by various categories or *dimensions* (e.g., time, geography, etc.), which are organized into *hierarchies* (e.g., month/quarter/year).

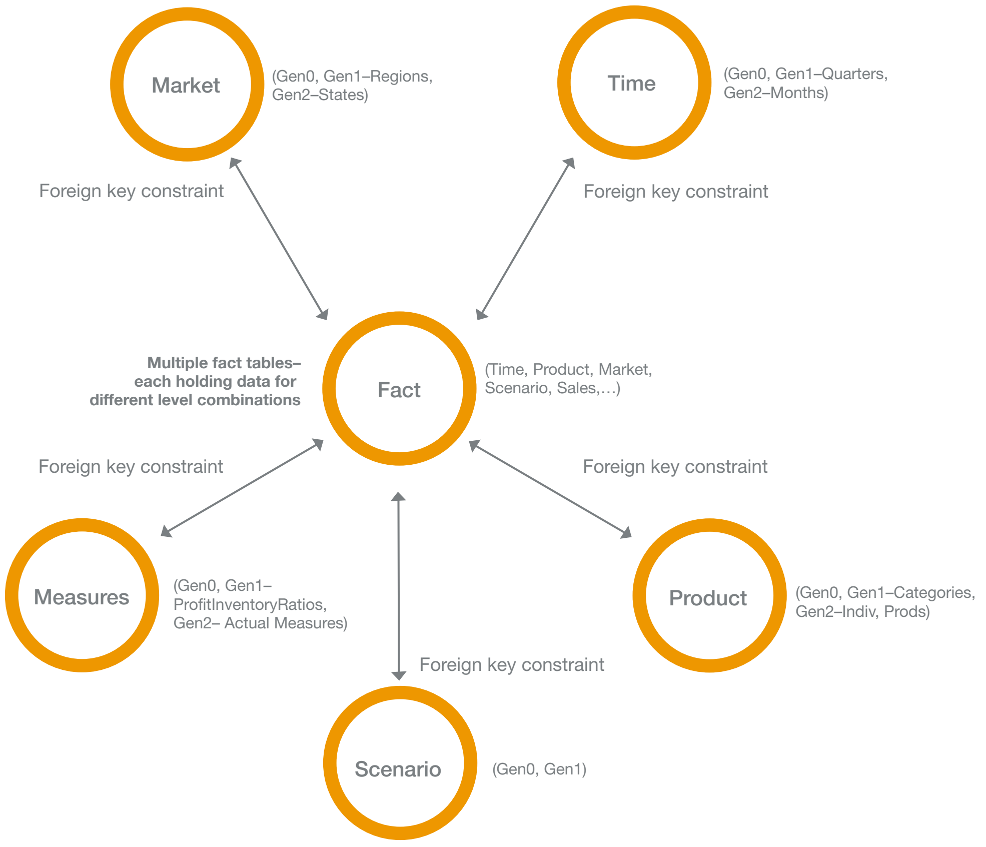
SQL-based relational database systems are based on set theory model, which is two-dimensional in nature. Dimensions and hierarchies are usually captured in multiple separate tables and processed through primary and foreign-key relationships. As a result, changes to dimensions and hierarchies must be explicitly monitored, and appropriate SQL queries must be issued to modify existing summaries whenever there is an update.

Access to relational data is value-based, which does not lend itself to preserving dimensional locality in the way information is physically stored. For example, January, February and March are dimensional siblings likely to be accessed together for analysis. However, relational databases store information in sequential rows of transaction after transaction, with no awareness of the groupings by which information will be retrieved and analyzed.

Because relational databases lack native support for analytical processing features such as hierarchies, iterative analysis and dimensional operations, a layer of logic outside the database is required to translate the multidimensional business model into a two-dimensional data model. In other words, dimensions are an application concept, as opposed to being supported by the data model itself.

Figure 2 shows a relational schema for the soft drink company's financial data. Dimension tables (Measures, Year, Product, Market and Scenario) capture hierarchical information required for zoom-ins and rollups. A separate table, commonly referred to as the fact table, holds data values that are related to the dimension tables through primary-key/foreign-key relationships. This kind of schema is commonly referred to as a star schema, because of the way dimension tables radiate out from the fact table. A variant of the star-schema—called a snowflake schema—is also widely used to separate dimensional attributes into separate tables.

In contrast, multidimensional databases organize data in a way that preserves dimensional locality. Access to data items is by position—determined by each data item's dimensional coordinates—rather than by value. Every measure is stored as a set of numeric values indexed by the members of each dimension. Each point in the multidimensional "dataspace" is mapped to a corresponding point on disk where the cell of information is stored. This is possible because dimension members are fairly static and the number of possible points in the multidimensional space is known.


Market
(Gen0, Gen1–Regions, Gen2–States)
Time
(Gen0, Gen1–Quarters, Gen2–Months)
Foreign key constraint
Foreign key constraint
Multiple fact tables– each holding data for different level combinations
Fact
(Time, Product, Market, Scenario, Sales,…)
Foreign key constraint
Foreign key constraint
Measures
(Gen0, Gen1– ProfitInventoryRatios, Gen2– Actual Measures)
Product
(Gen0, Gen1–Categories, Gen2–Indiv, Prods)
Foreign key constraint
Scenario
(Gen0, Gen1)

A typical star schema used to represent multidimensional data in a relational database.

Such a storage mechanism is more efficient for analysis than relational tables, which require that dimensional coordinates be enumerated for every cell value. In a multidimensional database, only a numerical value is stored for each cell, since the physical location of that cell is described by the dimension coordinates. The multidimensional business view is physically preserved in the data model, ensuring dimensional locality and enabling more efficient computations of rollup, zoom-in and slice/dice operations.

Analytical Advantages of Multidimensional Databases

Multidimensional databases such as Hyperion Essbase offer a number of advantages over relational databases in addressing the analytical processing requirements described earlier in this document.

Summarization

Additive consolidations are performed in SQL by using the ANSI-SQL ROLLUP operator. This operator assumes a two-level hierarchy with only one parent and several children. Hence multilevel hierarchies are captured through explicit joins with dimension tables. Assuming all of the hierarchical relationships illustrated on page 1 are additive, the following SQL is required to generate the summaries:

```
SELECT Time.GEN0 as Time0, Time.GEN1 as Time1, Time.GEN2 as Time2,
     Scenario.GEN0 as Scenario0,
     Market.GEN0 as Market0, Market.GEN1 as Market1,
     Market.GEN2 as Market2,
     Product.GEN0 as Product0, Product.GEN1 as Product1,
     Product.GEN2 as Product2,
     sum(fact.sales) as sumsales,
     ... (other additive consolidations)
FROM fact, Time, Scenario, Market, Product
WHERE
     (fact.Time = Time.GEN2) AND
     (fact.Scenario = Scenario.GEN0) AND
     (fact.Market = Market.GEN2) AND
     (fact.Product = Product.GEN2)
GROUP BY ROLLUP(Time.GEN0, Time.GEN1, Time.GEN2),
         ROLLUP(Scenario.GEN0),
         ROLLUP(Market.GEN0, Market.GEN1, Market.GEN2),
         ROLLUP(Product.GEN0, Product.GEN1, Product.GEN2)
```

This SQL-based approach to summarization has a number of limitations from an analytical standpoint:

Sophisticated calculations are cumbersome or impossible with SQL. ANSI SQL constructs such as ROLLUP and CUBE support intra-row operations involving additive consolidations only. In other words, summarization along the measures dimension is the only kind of consolidation that can be expressed, and this requires that the entire dimension appear as column values in the fact table. Non-additive intra-row consolidations such as *Variance = Actual-Budget* are too cumbersome to express in SQL. Inter-row calculations (i.e., calculations involving two or more rows, such as: *SalesContributionOfATimePeriod = SalesInTheTimePeriod/SalesInTheQuarter)* are impossible to express with any declarative ANSI SQL construct.

Multidimensional SQL operators are implemented inefficiently. SQL queries like the one above are very expensive and cannot be executed in real time for even the smallest data sets. In most relational database systems, ROLLUP operations are implemented by splitting the query into several smaller units that are processed one at a time and unioned together. Materializing the entire query through the creation of summary tables is an option, but is generally impractical because of the storage requirements. Selective materialization of summaries is another alternative, but this means each query must be analyzed before execution to determine which summary tables may be used. Relational databases claim to optimize queries by automatically routing them to the appropriate summary tables, but identifying which summary tables cover what portions of a query is a complex task. It is not surprising, then, that empirical results demonstrate this feature works best only for very simple queries.

To optimize query performance, database administrators must pay careful attention to data distribution in each table, build the right kinds of indices with the appropriate sort orders, ensure the proper join method (nested loops vs. sort merge vs. hash join) is used, allocate sufficient temporary storage and determine how summaries are to be organized (in one table or split across tables by level). This last requirement, in particular, is a complex function of how often hierarchies are expected to change, which summaries need to be refreshed more often and which summaries will be queried frequently. Administrators often must choose between sluggish response times if the right summaries are not created, or lengthy windows of unavailability while summaries are created and updated.

One reason for SQL's inefficient handling of multidimensional operations is that transaction semantics of relational systems are more tuned to handle OLTP workloads. Analytical processing differs from transactional processing, in that it is characterized by a combination of intensive read activity, small "trickle" updates and long-running calculations. Transactional algorithms for cache management, serialization, redundancy models, join operations and indexing methods are not well-suited for analytical workloads.

Managing summary data with SQL is a painful, time-intensive process. Summaries are typically separated into different tables for each combination of dimension levels. This allows the system to route queries to the appropriate summary tables using application logic instead of relying on the relational optimizer, and simplifies determining which summaries must be refreshed whenever dimensional or data updates occur. Unfortunately, this approach stops scaling even for relative small analytical applications. For example, with six dimensions that each have four levels of consolidation, there would be a total of 4**6 (4096) possible summary tables to manage.

In contrast to relational systems, multidimensional databases such as Hyperion Essbase have a number of characteristics that make summarization easier and more powerful.

Hyperion Essbase enables symmetric dimensional calculations. Hierarchies are an integral part of the Hyperion Essbase multidimensional data model. Members can be associated with unary mathematical symbols (such as +, –, *, / and %), which automatically determine the parent/child relationship. There is no distinction between inter-row and intra-row calculations. Every dimension is treated symmetrically, which enables all types of sophisticated consolidations to be expressed in the data model itself. Hyperion Essbase publishes a default calculation order, but also allows users to specify their own order for asymmetric consolidations.

Hyperion Essbase provides native support for multidimensional operators. Rolling up data is a native multidimensional operation. Cells are stored and accessed in a predefined order, which preserves spatial locality and optimizes rollup performance. Additionally, Hyperion Essbase's patented categorization of dimensions into dense and sparse based on data distribution enables more efficient storage and near uniform response times for every point in the multidimensional space.

Hyperion Essbase's efficient storage structures, coupled with native support for consolidations, make it possible to pre-compute most summaries in a reasonably short time. Hyperion Essbase transparently handles query requests for summaries that have not been pre-computed by dynamically calculating summaries as required. Unlike a relational database, selective pre-computation does not require users to know whether or not specific summary tables exist when they form queries. HyperionEssbase's efficient storage layout provides better I/O throughput (one page of a Hyperion Essbase multidimensional space may contain 10,000 cells, as opposed to a page of a relational data fact table that may contain only 100 rows), which guarantees faster computations and makes dynamic summarizations more practical.

Hyperion Essbase transparently manages summaries. Regardless of the number of dimensions or levels in each dimension, summary management is completely transparent with Hyperion Essbase. Unlike relational systems, there is no need to separate summaries into different tables. Internal benchmarks show that implementing the OLAP Council's APB-1 benchmark using a relational database with the most recent ANSI SQL-99 extensions consumes up to 150 times more storage for summarization than does Hyperion Essbase. And Hyperion Essbase, without any special tuning and using minimal resources, builds summaries up to 100 times faster than a relational system—even when the relational system has parallelism enabled and all resources dedicated to summary computation.

Hyperion Essbase automatically handles updates to dimensions and data values. Most changes happen in real time with minimal user intervention, although support for deferred updates also exists. Contrast this with relational database administrators who must track each and every update and issue the appropriate SQL to incorporate updates into existing summary tables.

Native Support for Business Analysis

ANSI SQL supports only time series computations and aggregation operations involving rankings, moving ratios and standard deviation. There is no declarative support for other accounting analysis such as time-balance calculations, variance reporting, native currency conversions and a rich repertoire of custom business analytic functions. All of these functions must be implemented at the application level with a relational database.

In contrast, Hyperion Essbase allows any type of business calculation to be described in a complete and intuitive calculation language. Even an average business user who is a database novice can easily create sophisticated calculations and formulas composed of operators and built-in functions. Formulas can be attached to a dimensional member and executed whenever a point containing that member is encountered in the multidimensional space. For example, the opening inventory for each product may be defined as being equal to the ending inventory for the prior month. This relationship can be expressed in Hyperion Essbase as `Opening Inventory = @PRIOR("Ending Inventory")`, where `@PRIOR` is a Hyperion Essbase function that understands the sequencing of months, quarters and years. Similarly, `profit%` can be expressed as `(Profit % Sales)`.

Several other member relationship operations are supported; for example, the sales contribution of cream sodas to the beverage category can be expressed as: `@SUMRANGE(Sales, @Children(CreamSodas))` / `@SUMRANGE(Sales, @LevMbrs(Product, 0))`. `@SUMRANGE` describes the sum of a measure over a specified range, `@Children(CreamSodas)` is a member relationship function that translates into the range (DietCream, DarkCream, VanillaCream) and `@LevMbrs(Product, 0)` identifies all leaf-level members along the product dimension.

Hyperion Essbase supports over 200 predefined routines including specialized mathematical functions such as Avg, Min and Max; financial functions such as NetPresentValue, RateOfReturn and CompoundGrowth; statistical functions such as Median, StdDev and Rank; moving ratios, linear regression and correlations; powerful forecasting functions and functions that reference member relationships created by dimension hierarchies (as in the above example). Every one of these functions has unrestricted access to any point in the dimensional space, significantly enhancing the expressive power of the calculation language. Expense reporting is as easy as tagging members that require expense computation, making all subsequent consolidations involving those members "expense aware." Finally, Hyperion Essbase has native support for automatic currency conversion to different denominations, an important requirement for most business analyses.

Richer Analysis Through Sophisticated Calculations

With recent ANSI SQL extensions that operate on multiple ranges of rows, many relational vendors have claimed to be able to perform *all* of the OLAP functionality delivered by multidimensional products such as Hyperion Essbase. In truth, all that these SQL OLAP aggregates have enabled is the ability to express in a single query several intra-row aggregations that previously needed to be split across several queries. The inefficiencies of aggregation operations, the inability to express non-additive intra-row aggregations and the lack of support for even some simple business requirements still plague SQL-based analysis. Further progress in this area is constrained by the simple fact that a row-column based model cannot easily support a multidimensional view of data.

Relational implementations of analytical processing claim to manage sparsity simply because fact tables only contain available data points. However, the lack of knowledge about each dimension's domain—except through explicit lookups of dimension tables—makes it practically impossible to perform operations that are a function of data points being absent. For example, standard deviation of sales across all products is sensitive to products with no sales to report. The concept of logical aggregation groups in the SQL99 standard is an attempt to address this deficiency, but it still suffers from domain sequencing limited to what is available through data types.

The following examples illustrate the rich analytical capability of Hyperion Essbase and its multidimensional calculation language. These queries either cannot be written in procedural SQL or require extensive SQL coding.

1. Marketing expenses are captured at the product family and region level. This data must be allocated down to each individual product and state, based on sales contribution. Input data is available as follows:

		Sales	Marketing
New York	Cola	300	Not available
	Diet cola	200	Not available
Boston	Cola	100	Not available
	Diet cola	400	Not available
	Colas	500	Not available
East	Cola	400	Not available
	Diet cola	600	Not available
	Colas	1000	200

The requirement can be expressed as a formula:

```
Marketing = Sales / (@mdancestval (2, Market, 2, Product, 2, Sales)*
                    @mdancestval (2, Market, 2, Product, 2, Marketing))
```

This formula is applied to every cell that has Marketing as a dimensional coordinate. `@mdancestval ( )` is a Hyperion Essbase function that returns ancestral members at a particular generation from a specified dimension. In this example, ancestral values from the second generation of the Market and Product dimensions are required. The last argument to `@mdancestval( )` identifies the measure. This formula returns the following results:

		Sales	Marketing
New York	Cola	300	60
	Diet cola	200	40
Boston	Cola	100	20
	Diet cola	400	80
East	Cola	400	80
	Diet cola	600	120

2. Payroll expenses are a function of each geographic area. Payroll for a state in the Eastern or Western region is calculated as 15% of the state's sales contribution. Payroll for the Central region is calculated as 11% of sales. For all other regions, payroll expenses equal sales * profit for a particular product in January. This is specified with the following formula.

```
Payroll = If (@isidesc (East) OR @isidesc (West))
             Sales * 0.15
          else if (@isidesc (Central))
                        Sales * 0.11
          else
             Sales * Margin->Jan
```

The `@isidesc()` function operates on a dimensional hierarchy, returning all descendants of a member including the member itself. The formula above is executed on all cells having Payroll as a dimensional coordinate. Members along the Market dimension for each cell are used to determine the value of Payroll. The `Margin->Jan` statement is a cross-dimensional reference, a powerful feature of Hyperion Essbase calculations that is not supported by declarative SQL.

Iterative Analysis

Analytical processing requires an iterative refinement of data and hierarchies to gain business insight. Common operations include changing member relationships, adding or removing members, changing consolidation types, loading new data for certain member combinations and adding new levels to a hierarchy. Relational systems claim to propagate updates to summary tables whenever base tables are updated, but this places several constraints on the type of SQL allowed in summary table construction. In particular, summary tables created using CUBE and ROLLUP operators cannot be automatically refreshed! This means almost all data and dimension updates must be tracked by the administrator and manually propagated to relevant summaries. Hyperion Essbase, on the other hand, propagates dimensional changes transparently and in real time. Data updates trigger the required recalculations during the next refresh phase. The recalculations are extremely efficient and largely incremental (i.e., only cells requiring recalculation are refreshed). This enables efficient and sophisticated "what-if" analysis in real time.

Real-Time Analysis

Selective pre-computation of summaries in relational systems requires the SQL optimizer to determine whether or not summary tables can handle a given query. This is a complicated task, and a user query must often be virtually identical to the query that created the summary table in order to trigger the optimizer. Hyperion Essbase allows administrators to specify which cells will be pre-computed and which will be dynamically calculated at query time. Additionally, Hyperion Essbase's powerful reporting language allows effortless construction of complex multidimensional spreadsheets. On-the-fly calculations, currency conversions, value-based ordering and top-n/bottom-n listings are just some of the functions that can operate on data retrieved by Hyperion Essbase.

Hyperion Headquarters
Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, CA 94089

tel: 408 744 9500
fax: 408 744 0400

info@hyperion.com
www.essbase.com
www.hyperion.com

Conclusion

Meaningful business analysis requires a multidimensional view of data that has proven inefficient and cumbersome to express using relational databases and SQL. Relational databases have achieved transaction efficiency, but the multidimensional data model is superior and more intuitive for analysis than the two-dimensional relational model.

In an attempt to retrofit multidimensionality on top of relational databases, SQL has recently been enhanced with new operators such as CUBE and ROLLUP. Support for statistical and windowing aggregate functions such as moving average, and for pre-computed summary tables have also been included in recent ANSI SQL standards Despite all of these efforts, multidimensional databases continue to offer significant advantages due to their ease of use, superior performance, naturally expressive calculation language and rich analytical processing capabilities.

1700_0800KS