crystal decisions™

Compound OLAP

An OLAP Architecture for the Real World

Overview

This document is all about a development which we believe defines a new direction for OLAP, and hence for Business Intelligence.

Contents

Introduction

When OLAP (On-line Analytical Processing) emerged in 1994, it served to define a previously fragmented marketplace. Various systems which had been available for some time, like Decision Support, EIS, and Business Modeling came together under the umbrella of OLAP. OLAP identified the common ground between products which had previously been seen as distinct, and effectively created a new market sector. In time, it has become apparent that OLAP on its own provides little benefit, and that what is really needed is an OLAP system together with the tools to allow distribution and exploration of the OLAP data. This combination, with some other elements, is the foundation of Business Intelligence systems.

OLAP is nevertheless a central part of Business Intelligence, and the key idea in OLAP is multi-dimensionality. This is a recognition that an information consumer, a person, thinks about data and information in a multi-dimensional way. A *person* wants to compare the performance of the Paris office, across a range of product lines and time periods, with that of the San Francisco operation. A *person* cares nothing for how the corresponding data might be stored on the machine – multi-dimensionality is like a *human information schema*.

The problem which faces OLAP vendors is the delivery of information in this human schema in an efficient way. Until now, none of the attempts made to map the way the information is stored on the machine to the way the user wants to see it have been wholly satisfactory. There have always been too many compromises, and most vendors have been forced, by their existing technology investment, to offer solutions which work well in certain situations, but very badly in others.

This white paper is all about a development which we believe defines a new direction for OLAP, and hence for Business Intelligence. With a single and simple technological idea, many of the shortcomings of OLAP simply vanish, and OLAP becomes practicable in even the biggest, most complex applications.

Existing Approaches to OLAP

At the time the OLAP market was defined through Codd & Date's seminal paper, a number of suppliers were out in the marketplace doing something similar to OLAP. All these vendors rushed to show that their products conformed to Dr Codd's rules; when some more rules were introduced, the vendors just showed they met those too. But of course the existing players were converging on OLAP, whatever it was exactly, from different starting points. Since the Codd & Data Paper was commissioned by Arbor Software, it might be thought that the OLAP definition would best describe Arbor's product Essbase, but nevertheless other vendors with very different approaches were also able to claim compliance. Thus was born the gulf between Relational OLAP and Multi-dimensional OLAP (the latter term is misleading since all OLAP is inherently multidimensional), which has the source of much spurious and specious argument in the OLAP world since.

Crystal Decisions' product, Crystal Holos, in fact had a foot in both camps because it had both MOLAP and ROLAP capabilities.

Relational OLAP

The idea behind Relational OLAP comes from the true observation that the majority of corporate information, which may become the subject of the human schema requirement already discussed, is located in relational databases. RDBMS have an impressive pedigree, are capable of holding almost any amount of data, and have a complete infrastructure of knowledge, skilled individuals, recognition and awareness behind them. Organizations storing mission-critical information rightly want an absolutely reliable, proven system, and a good relational database provides it.

But relational databases are optimized for transaction performance in a way which does not meet the requirements of OLAP access. The normalization process in the design of the database almost guarantees poor performance in the context of OLAP, and there is a danger that OLAP queries on operational systems will have an extremely adverse effect on the primary function of the database.

Suppliers who have chosen the relational OLAP route find it necessary to create elaborate database schemas, effectively to simulate multi-dimensionality in a relational environment. A key weakness of this technique is the maintenance burden – it is difficult and time-consuming to accommodate the kind of day-to-day changes in product lines, for example, which happen in practically every business.

This complexity, slow response to change, the generally poor performance, and the read-only access provided by ROLAP means that, while there are certainly applications for which the ROLAP approach is valid, for the majority of mainstream applications, pure ROLAP is not a contender.

Multi-Dimensional OLAP

So-called MOLAP systems stored data in a form which is optimized for multi-dimensional access. The challenge is met head-on – "we have a multi-dimensional problem, so let's create a multi-dimensional solution". The data is stored in a proprietary file format, which is as compact as possible while providing fast access to data. Unlike ROLAP, typical business calculations which may involve the execution of business rules on large sets of data become feasible when the many data elements which need to be accessed can be reached quickly, and the results can be written back to the OLAP data store.

Some early systems loaded all data into memory, giving extremely fast access. But now OLAP is being used for very large scale applications, it is simply impractical and uneconomic to consider memory-based storage, despite the rapid and continuing drop in memory prices.

Disk-based storage is naturally slower, but has a higher capacity. To overcome the potential performance penalty, the more sophisticated MOLAP systems optimize performance through careful use of file system characteristics, and through caching. It is possible to exploit the caching strategy of the underlying operating system to boost performance, or a better approach is to create a custom caching scheme which can make intelligent forecasts of multi-dimensional data proximity and sequencing.

The first versions of Crystal Holos used a memory-based OLAP approach, supplemented later with a range of disk-based options which utilize these special-purpose multi-dimensional caching strategies.

One of the biggest problems with MOLAP is the requirement for large-scale transfer and hence duplication of operational data. Since this will often come from relational database sources, there is a stark choice – either replicate often, accepting the load on the RDBMS, or settle for out-of-date data in the multi-dimensional system.

Hybrid OLAP

Hybrid OLAP (HOLAP) is obviously enough a combination of ROLAP and MOLAP. The aim is to use each approach in the role for which it is most effective. MOLAP is used for high-speed access to summarized data, while the underlying lowest-level data is held in the relational database. Crystal Holos was the first and arguably still is the only system to deliver a full hybrid solution, although some vendors have recently made me-too product announcements.

But while hybrid OLAP can't fail to be better than either MOLAP or ROLAP singly, there is another dimension to the problem at the Enterprise OLAP level. What is needed is still hybrid approach, but with a whole new range of flexibility in applying OLAP to real-world problems. This is the background to Compound OLAP.

Compound OLAP Architecture

The Challenges

So far, we have looked in fairly abstract terms at the variety of approaches to delivering this necessary human schema for Business Information. In the real world, the situation is complicated by a number of factors.

Data volumes are now almost unlimited

The biggest challenge in OLAP now is the sheer volume of data which needs to be handled. Large organizations are usually awash with data, and OLAP techniques are being used to try to make sense of these vast data volumes. But most OLAP systems impose limits on data volumes, and cannot scale to the level now demanded of them.

For very high capacity applications, there can be constraints in hardware and in time – can existing hardware be upgraded sufficiently to handle the potential data volumes, and will the necessary processing fit within the overnight batch window available?

Data changes and grows

So data starts big but, almost more difficult for some OLAP approaches, it then changes and grows. Consider a sales organization offering 10,000 product lines. In some types of business, 5% of these could change month-on-month – 500 discontinued and 500 new products each month. Some approaches to OLAP cannot deal efficiently, or anything like it, with this degree of change.

So the structure of the data changes, but in particular it grows. Time-based data grows all the time, self-evidently, and in any organization the trend is to capture more operational data, not less. A successful Business Intelligence application which began with fairly limited scope will almost always grow, encompassing more data and more areas of the business.

Real-world data is sparse

Some approaches to OLAP would work well if a very large proportion of the possible data values did in fact exist. But in a sales organization for example, some product lines will not be sold in some regions, in certain periods. The net effect of the factors, when taken across all the dimensions of the OLAP cube, can be that only a tiny fraction of the possible data values actually exist. In real-world Holos applications, it is not uncommon to see millions of actual data cells within trillions of possible cells – about one cell in every million actually in use.

The data explosion

The problem of 'data explosion' is a surprising and counter-intuitive characteristic of multi-dimensional data. If the data is sparse, and is subject to hierarchical roll-ups particularly in several dimensions of the OLAP cube (for example local-office figures are summed to give a regional figure, then to a sales area, then to a country and then to the world), then the amount of calculated, rolled-up data can be orders of magnitude larger than the source data. While the source data might be handled comfortably by the hardware and storage in use, it is possible that the calculated data could be far too large. This unavoidable characteristic of hierarchical multi-dimensional data has a significant influence in choosing how to deal with enterprise-level volumes of data, and is a major challenge for OLAP vendors.

The Compound OLAP Solution

The ideas behind Compound OLAP are really very simple. The user's view, the human schema, required of an OLAP data store is just a giant 'cube' which apparently contains all available data and calculated results. The trap, into which some vendors have fallen, is to use this human requirement as the model for the physical implementation in the storage of the computer – for reasons already discussed, at the enterprise scale this is not desirable or even feasible. The solution, almost self-evident in retrospect, is to allow OLAP cubes to be combined, in a way analogous to Joins in the relational database world. If this joining mechanism is made sufficiently flexible, some quite startling advantages just fall out. At month-end, it becomes possible simply to add-in another cube of data and attach it transparently to the existing data. If data volumes become unmanageable for a single machine, just perform the cube processing on another machine. And if ROLAP and MOLAP cubes are combined, the resulting compound cube can merge the best characteristics of both – speed of access for MOLAP, and information down to the most detailed level for ROLAP.

Flexibility in cube compounding

The whole concept of Compound OLAP relies on providing complete flexibility in the way component cubes can be combined. In the Compound OLAP Architecture using in Holos the joining would appear to be simple – either cubes are joined 'next to each other', or 'one beneath another'. The terms *racking* and *stacking* are used for these methods. There is no restriction on the type of joined

cube – since Holos offers relational, memory and disk-based cubes, there is limitless flexibility in achieving exactly the characteristics required of the Compound OLAP data store.

This ability to combine cubes of different types, in different relationships would on it own be a major development in OLAP. But the final element of Compound OLAP is perhaps the most important – compound cubes can contain other compound cubes, repeated to any required depth.

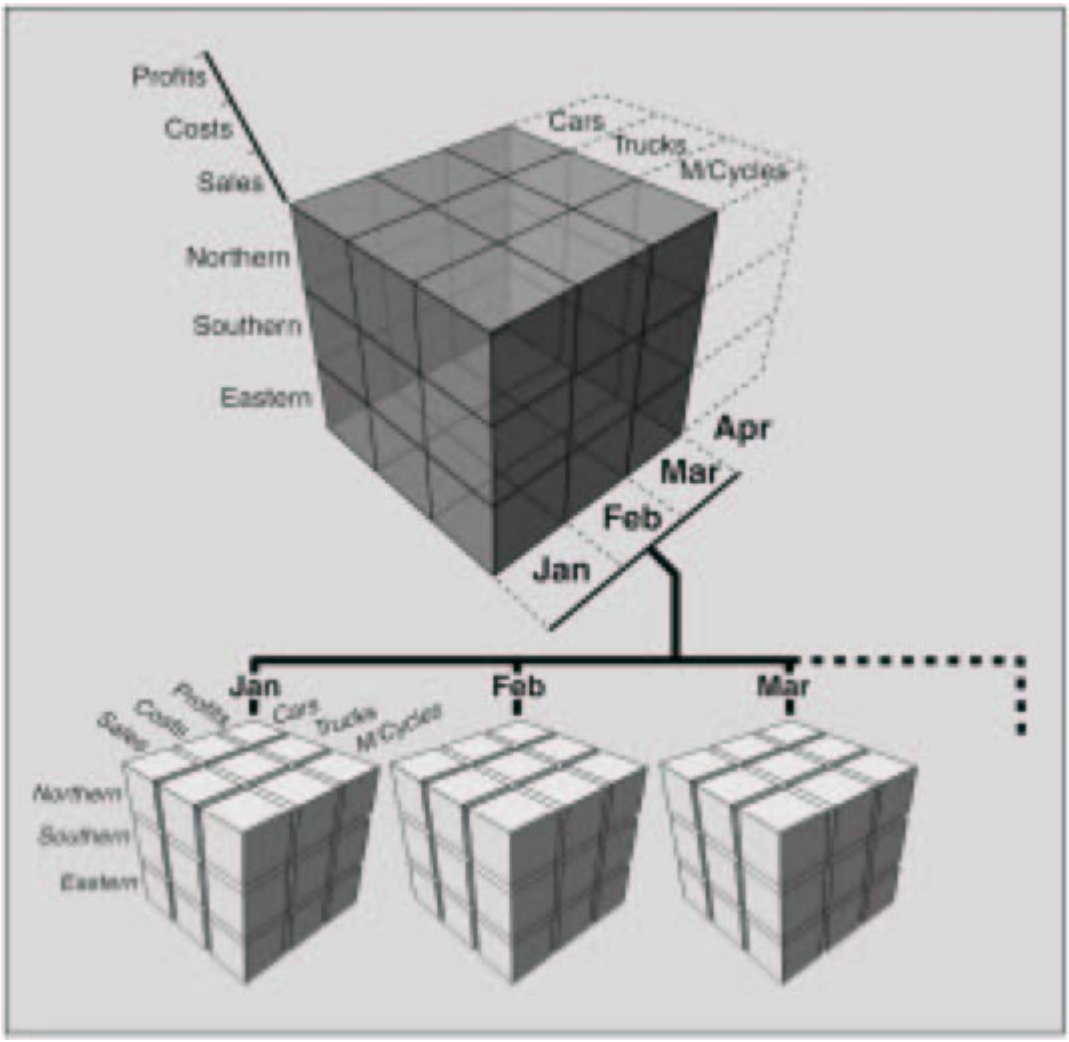
In the following three pages, we illustrate and describe this flexibility in cube compounding in greater detail.

Racking

When cubes are racked together they are joined side by side along a "backbone" dimension.

For example, consider three cubes containing account data dimensioned by region, product and line item. The first holds figures for January, the second for February and the third for March. Compound OLAP Architecture allows these to be racked together by introducing a time dimension to act as a backbone. Each of the monthly structures would contribute the data for one field in the backbone dimension.

The resulting compound cube would behave exactly like a normal cube dimensioned by region, product, line item and time, but would contain no data of its own – instead it would *point* to data in the underlying cubes.

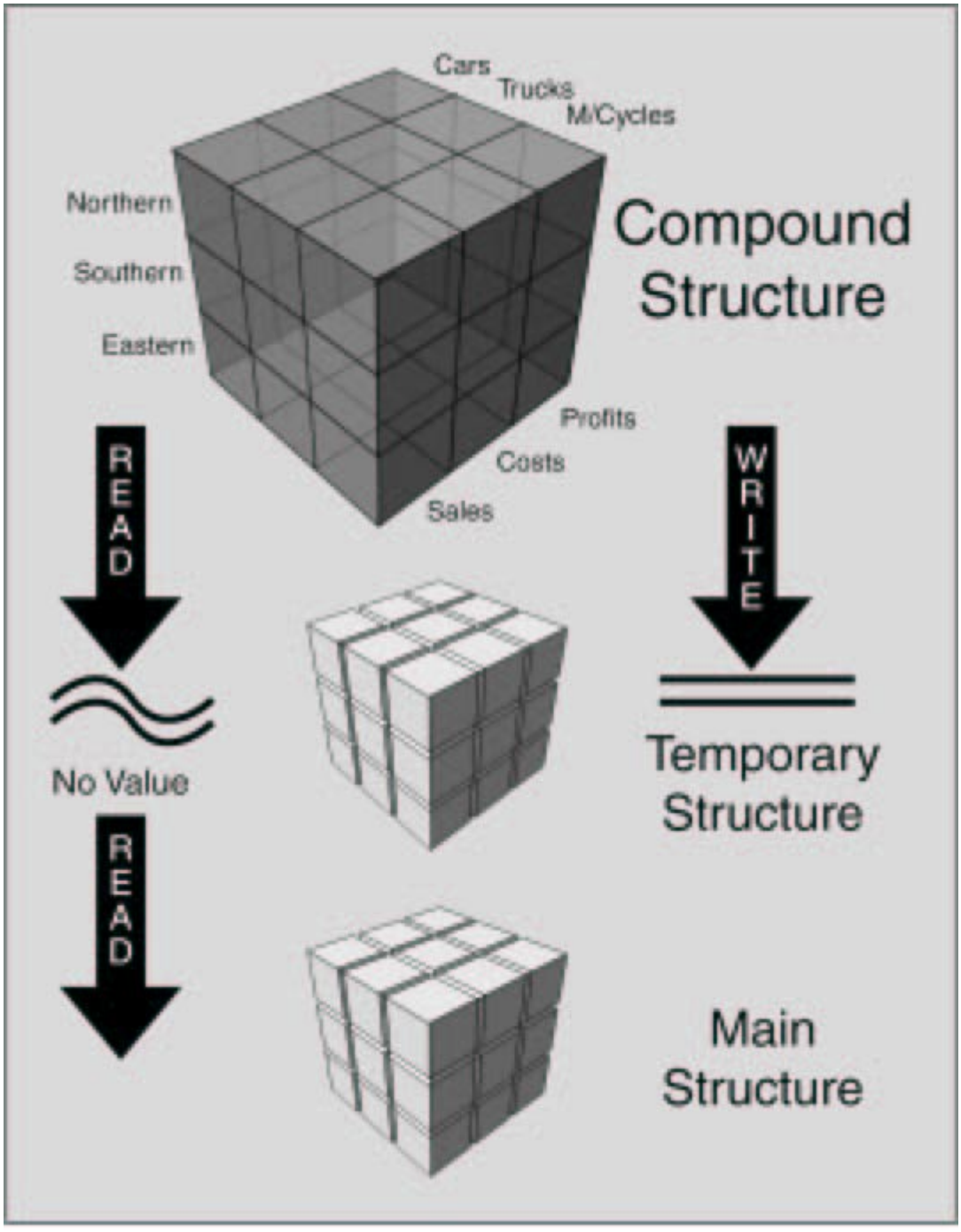


Compound OLAP Racking

Stacking

When two structures are joined together by "stacking" the resulting compound cube acts like a data valve. When data is read from the compound cube, Holos first interrogates the top cube of the stack. If the data is found it is returned, if not the bottom cube is interrogated. By contrast, when data is written to the compound cube it is only ever written to the top cube.

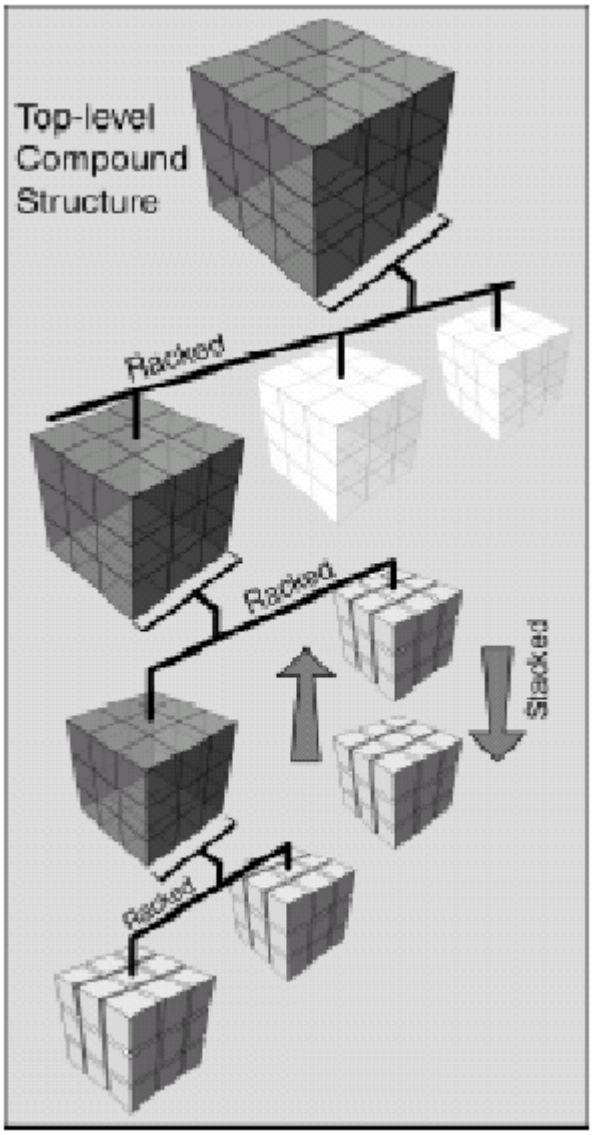
One use of this is in the provision of transaction control over end-user changes to data.



Compound OLAP Stacking

Joining joined cubes

The third and perhaps most important element of the compound mechanism is the freedom to combine compound cubes into further compound cubes, with no practical limit on the level of nesting. In combination with the stacking and racking capabilities, this allows extremely complex real-world situations to be represented efficiently in multi-dimensional storage. For special situations, such as the flexibility in handling the time dimension discussed below, it is even possible to join cubes to themselves.



Joining cubes to any level

The benefits of language

Not directly part of Compound OLAP, but an important part of its implementation in Holos, is the ability to create Compound OLAP cubes under the control of the Holos language. This means, for example, that 'real' cubes which contain data and which already exist can be combined into Compound cubes at 'run-time' in the application, perhaps in response to the user's actions. This is an unparalleled level of flexibility.

The Benefits

The Compound OLAP approach elegantly solves many of the challenges which arise in large-scale OLAP applications. There are some key benefits, to some extent interrelated:

- Scalability
- Parallel processing
- Separation of the logical and physical OLAP representation
- Representation of the complexities of real world data

In addition, there are some more minor benefits, which can nevertheless be important in particular applications:

- Live data updates
- Handling the time dimension
- Improving developer productivity

Scalability

The dramatic increase in the volume of data which OLAP systems need to handle has made the ability to scale, to handle the data volumes, to apply as much processing power as is necessary to achieve satisfactory performance, absolutely vital. With its multi-cube architecture, the design of Crystal Holos has always reflected the fact that a more granular approach to multi-dimensional data storage is necessary. As a result, Holos has always had the capability to cope with large volumes of data. However, until now this was often at the expense of the requirement to develop sophisticated data-handling applications to glue the underlying cubes together for the end-user. Compound OLAP changes this by providing a completely scalable and adaptable infrastructure for joining OLAP cubes together.

The complete scalability of Compound OLAP arises from the freedom to divide a multi-dimensional problem, however large, into manageable cubes, and then to process those cubes in series or in parallel efficiently on the available hardware.

Parallel OLAP processing

Most of the major OLAP engines claim to be multi-threaded. However, this ability is limited to querying data only. Loading & pre-calculation of data can

only ever utilize one processor. As data volumes grow and more is demanded of over-night batch windows, this can be a serious limitation.

Compound OLAP breaks this restriction and allows the loading and pre-calculation of data to be spread over many processors, either inside a single multi-processor system or across a loosely clustered network of machines.

Because each of the underlying physical cubes can be accessed and manipulated independently it is possible for each of them to be sent to a different CPU for processing before recombination through the compound cube. This makes Crystal Holos the only OLAP tool which can make full use of SMP and MPP hardware for data processing

In addition to utilizing specific parallel hardware, Compound OLAP can also use loosely clustered machines on a network to process cubes in parallel. This is because each of the physical cubes is stored in a separate file which can be independently distributed around a network for processing. This "shared nothing" approach is particularly advantageous because it is often memory usage and not CPU usage which is the bottleneck in OLAP processing

Logical/physical OLAP separation

The split between logical and physical design is standard in the world of relational databases. But until now there has been no engine with a sufficiently flexible data architecture to provide this in the OLAP arena. So far, this has not been a serious issue as the volumes of data in OLAP systems has been relatively small, but with the continued growth of OLAP data volumes, this is likely to change in the near future. With the Compound OLAP Architecture, Crystal Holos is the only OLAP engine available which is ready for this change.

Compound OLAP allows an arbitrarily complex underlying combination of cubes (the physical implementation) to be joined together into a compound cube (the logical view) which is accessed by application developers and end-users alike.

One of the major advantages of this separation in Compound OLAP is that it allows the underlying design of the data structures to be changed without disturbing the view of the data which is accessed at the higher level. This type of re-design is not uncommon and can stem from either growing data volumes of or from an initial misunderstanding of the characteristics of the data.

Handling real world data

In the real world, OLAP data has a wide range of characteristics. Sometimes it is dense, sometimes sparse, it can have many dimensions or just a few, the dimensions can be very hierarchical or flat or so on.

As the size of individual OLAP cubes grows to tens or hundreds of gigabytes, the idea that it is possible to use a generic, homogeneous container to hold the data in a large cube is just not credible. Compound OLAP uniquely gives the flexibility to use the particular characteristics of data sets to optimize the design of OLAP data storage.

Live update

Most OLAP systems allow the incremental update of data in their cubes. Unfortunately, due to the complex inter-relationships inside multi-dimensional

data these updates can quickly cause significant degradation in the performance of both the updates themselves and of queries to the resulting cubes. In addition, it is usually the case that the cubes must be taken off-line for the often-lengthy duration of the update process.

With Compound OLAP these problems disappear. Each time the data needs to be updated the underlying cubes can be independently created or refreshed, either on the production machine itself or more likely on a separate processor. Once the data is ready they can be snapped back into the main compound cube with a minimum of disruption.

Because Compound OLAP allows nested levels of compound cubes, the data can be broken down as finely as is required to allow each section of the data to be independently updated.

Handling the Time dimension

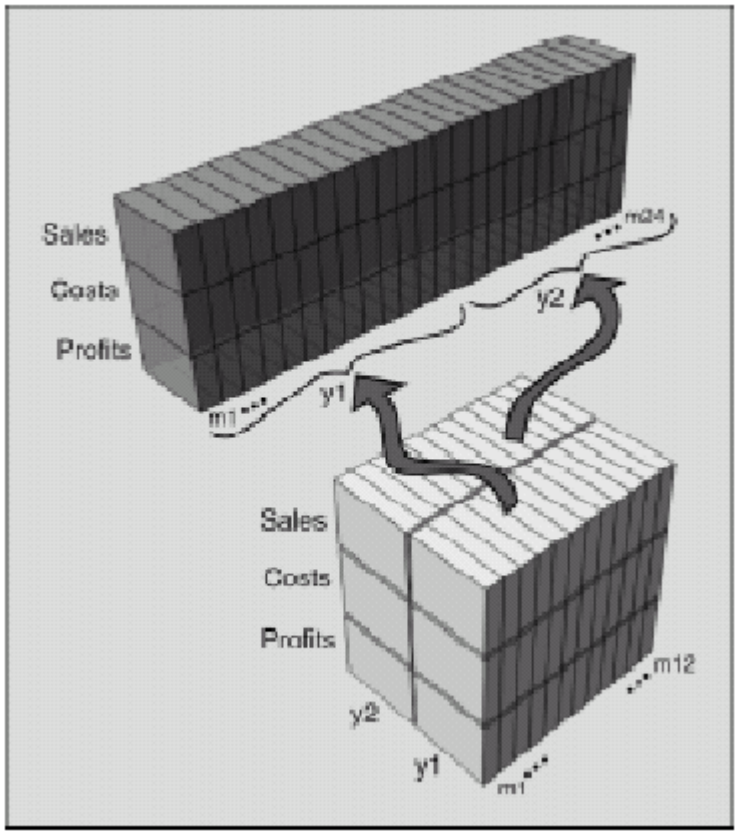
One of the first design problems which almost all OLAP systems encounter is the question of how to represent the time dimension in the cube.

If there is more than one year's worth of monthly data, it can be represented either as a single long time dimension or as two dimensions – one containing the twelve months of the year and the other containing a field for each year covered by the data. In the example shown in the diagram, this is represented by either a 24 month dimension or a 12 month dimension and a dimension containing the fields 'y1' and 'y2'.

Unfortunately, it is almost certainly that both views will be required sooner or later. The two time dimension approach is ideal for year-on-year analysis and the one dimension approach is ideal for trend analysis

Usually the year-on-year approach is chosen as the main storage mechanism with the trend view being provided either by wholesale data duplication or by clever application work to produce trend analyses "on the fly" as the user requests them.

Uniquely, Compound OLAP provides an extremely elegant solution to the problem. Due to the flexibility of the cube combination in Compound OLAP it is possible for a single compound cube to slice up a year-on-year style cube and re-present it as a trend style cube. Both the application developer and the end-user can make use of both views but



Elegant solution to the 'two time dimension' problem

there is no data duplication as the compound cube has no data of its own.

In addition, because compound cubes support full read/write access to the data, it is possible to use both to support calculations on the data. This is extremely helpful because while the year-on-year view is ideally suited for yearly variance analysis, it is quite complex to use it to perform the rolling 12 month calculations which are made trivial by the trend view.

Improving developer productivity

With enough effort, many of the problems of real-world OLAP can be overcome by skilled developers working in any given OLAP tool. Developer time is at a premium though, and the better the tools the developer works with, the better the finished application is likely to be and the more quickly it will be delivered. The Compound OLAP Architecture offers the developer a flexible and expressive means of creating a multi-dimensional representation of the real-world business situation. The correspondence between the OLAP model and the real situation can be close, and can be achieved quickly without time-consuming 'work-arounds'.

Possible Objections to Compound OLAP

There are some obvious potential objections to Compound OLAP – that it introduces extra complexity, and that it might have an adverse effect on performance. Experience in implementing large-scale OLAP solutions suggests that neither objection is valid.

Complexity

The most obvious criticism of Compound OLAP is that it is complex, and imposes an additional burden on application developers. In fact, designing efficient multi-dimensional representation of real-world situations is a challenging task, and the hardest thing an application developer will do in the creation of an OLAP system. The Compound OLAP Architecture in Holos provides the application developer with the tools needed to express the multi-dimensional situation, and avoids time wasted on 'working around' the limitations of a restricted system.

The parallel processing capability again has the potential to make life more complicated for the application developer. But in the Holos implementation of Compound OLAP, a parallel processing toolkit is provided to handle the necessary co-ordination.

Performance

At first sight, it might appear that the layering of data in nested cubes would lead to a degradation of performance. Surprisingly, in practice the reverse is true. The compound cubes act as high-level indices into the underlying data, which in turn relieves some of the pressure on the indices inside the physical cubs and gives improved performance.

In general, the better management of sparsity and more logical organization of the multi-dimensional storage possible with Compound OLAP leads to performance which is better than equivalent single cube solutions.

Real-World Application

An Automobile Company

A well-known automobile company, in its home market operation, faced exactly the kind of problem the Compound OLAP architecture was designed to solve. The company had about 3 million owners in their home country, and a fairly typical organizational structure with sales information recorded at salesman level, then aggregated up through sales outlet, dealer, region, and HQ. One of the goals of the intended application was to allow dealers to obtain information about new car registration and so to adapt their marketing strategies to suit sales trends in their local areas.

At first sight, the OLAP cube required had more than a million million potential cells, was extremely sparse, with many dimensions and deep hierarchies. This potential data structure suffered badly from the 'data explosion' problem, with

many more cells populated as a result of the hierarchies than there were original base data cells.

Using the Compound OLAP Architecture in Holos 6, a solution was found which involved creating one OLAP cube for each dealer, with aggregations and other processing pre-calculated. These pre-calculated cubes were combined into one large compound cube using the Compound OLAP Architecture. Higher-level aggregations were performed on-the-fly, at the time they were needed.

The data in the system is updated monthly. These updates are achieved simply through the addition of extra, racked, structures along the time dimension, and do not involve any re-load of existing data.

The Compound OLAP Architecture provided a solution to two key problems facing large-scale OLAP applications – the ability to handle very large data volumes, and a solution to multi-dimensional data explosion.

Conclusion

OLAP vendors have been struggling to meet the requirements of large-scale enterprise applications, with Hybrid OLAP offering the best solution so far. The new Compound OLAP Architecture introduced in Crystal Holos 6 makes it possible, for the first time, to face the challenges presented by even the very largest and most complex OLAP applications.

Recently, the fragmented approaches to OLAP arising from each vendor's existing technology base have begun to converge on Hybrid OLAP. The target has now moved on, to Compound OLAP.

Compound OLAP is effectively a 'paradigm shift' which redefines the OLAP marketplace, and brings many more applications within OLAP's scope. The power that Compound OLAP brings to Business Intelligence is such that it is difficult to see credible OLAP products of the future doing without it.