



Size matters

5th October 2000

There is a continuing emphasis, in all forms of business intelligence environment, on providing improved performance...

The notion of an "information worker" has been transformed by the increased emphasis on customer service and the growing need for customer (and business partner) self-help. The term - once restricted to senior executives and business planners - has grown to encompass line managers and all customer-facing staff.



For obvious reasons there is a continuing emphasis, in all forms of business intelligence environment, on providing improved performance. To run queries faster, to run more queries simultaneously, to run more complex queries. One way to achieve this is by investing in more, and faster, hardware. However, a more fruitful approach is to achieve this through software. And one of the most useful ways of achieving this is by reducing the size of the database, not in logical terms, but physically. If, for example, you have two databases that contain the same data, and one requires 100 Gb and the other requires 50 Gb then, all other things being equal, it will take less time to search through the latter than the former.

Of course, there are also other benefits of a smaller database. For example, it takes less time to load it and it requires less maintenance and tuning. Also, as we shall see, ultra-small databases are small enough to be e-mailed as an attachment. You don't believe us? Read on.

One of the most useful ways of achieving this is by reducing the size of the database...

The first and most obvious way of reducing the size of a database is to use compression techniques. But the problem with this is that the different fields in a data row all have different attributes and it is therefore not possible, using conventional databases, to optimise the compression. Indeed, a least common denominator approach is the best that can usually be achieved.



Another way to reduce the size of a database is to reduce the number of indexes. Typically, the indexes in a data warehouse take up as much space as the data itself, in effect doubling the size of the database. However, in conventional environments, the more indexes you remove the slower queries will run.

But all is not lost, one approach that eliminates the need for indexes is to use column instead of row-based storage. This has three effects. First, since storage is by column, there is no need to create indexes since each column is, in effect, an index on that table (though you may want to define specific indexes to increase performance for particular types of query). Secondly, because data is stored by column, you can implement compression in an optimal manner for each column, since the fields within the column share a common datatype. And, thirdly, you can use vertical rather than horizontal partitioning (that is, partitioning by column rather than row). This has the advantage that the partitioning cannot become unbalanced because each column has the same number of entries.

There are several examples of column-based databases of which the most well known is Sybase Adaptive Server IQ Multiplex. Others include Freedom Intelligence from the company of the same name and Nucleus Server from Sand Technology. All three of these products are marketed as suitable for us as a front-end to a data warehouse (for example, Freedom Intelligence markets its product as a query

accelerator), and Sand and Freedom Intelligence in particular limit themselves to this market in order not to compete with the data warehousing big boys. Sybase, of course, is in a different position and certainly does market into this sector.

Column-based storage and tokenisation are techniques used to further reduce the size of a database...



Freedom Intelligence and Nucleus Server also apply an additional technology to help further reduce the size of the database. This is a process known as tokenisation. Put briefly, the aim of tokenisation is to separate data values from data use. This has the effect of reducing data requirements and further improving performance. As a practical example, in a customer table you might have many customers in Michigan and each of them would have "Michigan" stored as a part of their address. To store this several hundred, or even thousands of times, is wasteful. Tokenisation aims to minimise this redundancy by storing a binary value (token) instead. The way that Freedom Intelligence and Nucleus Server treat these tokens is different but, in essence, they are mapped into an array that is then compressed. It should be noted that the resulting compressed data is never decompressed but, instead, is read directly in its compressed form.

Sybase IQ also uses tokenisation but only for low cardinality fields. That is, those with a limited number of unique fields (up to 1,500). For high cardinality fields it uses bitwise indexing, and otherwise it uses column stores (that is, it works on the basis that the column is self-indexing) or Btrees when multiple columns will be required in the same query.

Two further features of Sybase IQ distinguish it from the other two vendors mentioned, which make it more suitable for deployment as a data warehouse. First, it includes substantial parallel capabilities, notably by supporting multiple read nodes and a separate write node. Uniquely, these can reside on a combination of Unix and Windows NT platforms. The other major feature of Sybase IQ is that it supports flat schemas. These are much simpler to implement, and require less maintenance than star or snowflake schemas.

Indeed, the undoubted leader in this market is QueryObject ...

However, column-based storage and tokenisation are not the only way to achieve reduced database sizes. Indeed, the undoubted leader in this market is QueryObject System Corporation with its QueryObject System product.



The founders of QueryObject System Corporation originally met when they worked together on the guidance systems for Cruise missiles and had to compress terabytes of topographical data into a far smaller space. Using fractal geometry they managed to achieve a compression ratio of 10,000:1. Subsequently they adapted this abstraction for commercial applications, initially working on a prototype with Chase Manhattan Bank, which resulted in compressing 30 million records into a 10Mb file. The Bank validated that the representation was complete and exact. To give a more recent example, one current user of the product has created a QueryObject on a single diskette that allows its users to view and analyse details of some 78 million households across the US.

We are not going to attempt to describe the intricacies of fractal geometry but all the evidence suggests that this is a very important advance. Moreover, as noted above, databases of this size are small enough to be attached to e-mails. We leave it to the reader to imagine how this can be used within their organisation. As we began: size matters.

This uses fractal geometry rather than tokenisation to achieve its size reduction. The QueryObjects created with this system can be sufficiently small that it is feasible to e-mail them, for example, even when using a conventional modem.