

Design Tip #121 Columnar Databases: Game Changers for DW/BI Deployment?

By Ralph Kimball

Although columnar RDBMSs have been in the market since the 1990s, the recent BI requirements to track rapidly growing customer demographics in enormous terabyte databases have made some of the advantages of columnar databases increasingly interesting.

Remember that a columnar RDBMS is a “standard” relational database supporting the familiar table and join constructs we are all used to. What makes a columnar RDBMS unique is the way data is stored at the physical level. Rather than storing each table row as a contiguous object on the disk, a columnar RDBMS stores *each table column* as a contiguous object on the disk. Typically, these column specific data objects are sorted, compressed, and heavily indexed for access. Even though the underlying physical storage is columnar, at the user query level (SQL), all tables appear to be made up of familiar rows. Applications do not have to be recoded to use a columnar RDBMS. The beneath-the-cover twist from a row orientation to a column orientation is the special feature of a columnar RDBMS.

A columnar RDBMS offers the application designer and the DBA some design advantages, including:

- Significant database compression. Dimension tables filled with many low cardinality repeated attributes compress significantly, sometimes by 95% or even more. Fact tables with numeric measurement values also compress surprisingly well, often by 70% or more. Columnar databases are very supportive of the dimensional modeling approach.
- The query and storage penalties for very wide tables with many columns are greatly reduced. A single column, especially one with low cardinality, adds very little to the overall storage of a table. More significantly, when a table row is fetched in a query, only the named columns are actually fetched from the disk. Thus if three columns are requested from a 100 column dimension, only about 3% of the row content is retrieved. In a conventional row oriented RDBMS, typically the entire row must be fetched when any part of the row is needed. (More exactly, the disk blocks containing the requested data are fetched into the query buffer and much un-requested data in these blocks comes along for the ride).
- Adding a column to a fact or dimension table in a columnar RDBMS is not an especially big deal since the system doesn't care about row widths. No more block splits when you expand a “row.” Now the designer can be much more profligate about adding columns to an existing table. For instance, you can now be comfortable with wider fact table designs, where many of the fields in a “row” have null values. In this way, the designer can regularly add new demographics columns to a customer dimension because each column is an independent compressed object in the physical storage. Be aware, however, that as you add more columns to your fact and dimension tables, you have increased responsibility to keep the BI tool user interfaces simple. A tried and true way to keep the user interfaces simple is to deploy multiple logical views for different BI use cases, where each view exposes a set of relevant fields used together. And, we should point out that adding more columns to a table is not an excuse for adding data to a table that violates the grain!
- As I mention in my design classes, SQL is horribly asymmetric in that it allows very complex computations and constraints within records, but makes it virtually impossible to apply computations or constraints across records. Columnar RDBMSs remove some of this applications pressure, not only because they encourage far wider table designs (thereby exposing many fields within the same row to SQL), but also because columnar RDBMSs specialize in cross column access through bitmap indexes and other special data structures. BI tool designers welcome the extended width of fact and dimension

tables because within-row constraints and computations are a hundred times easier to set up than the corresponding cross-row versions.

Columnar databases pose provocative tradeoffs for IT. In particular, these databases have had a reputation for slow loading performance, which the vendors have been addressing. Before investing in a columnar database, make sure to carefully prototype loading and updating back room tasks. But in any case, these databases offer some interesting design alternatives.