INFOKARTA

Aggregation Logistics

"Method" in the Data Kitchen "Madness"

By Michael Kamfonas

Imagine a huge industrial kitchen serving an enormous restaurant where customers have a choice to order ala carte or to request ad-hoc meals that can be prepared from a published list of ingredients. The chefs accommodate a torrent of these, standard and ad-hoc, orders, and deliver an impressive array of high quality products on time. The recipes that combine raw ingredients to produce entrées, main dishes, sauces, garnitures, soups, side dishes are well defined and don't change that often.

In order to meet the strict quality and time demands, chefs, every day, concoct stocks, demi glaze, half boiled pasta, salad mixes and all those laborious core preparations, before their guests arrive. As a result, the staff can keep up producing and serving all those steaming hot extraordinary dishes even during the buzz of high activity.

What preparations should be available beforehand, and at what quantity? Stocking on Holadaise, Espagnole and the few other "mother sauces" from which dozens of sauces are derived and used daily is a practical choice. On the other hand, stocking Sauce Charcutière, Demi-glace, Bourguignonne, Sauce aux Champignons, Sauce Béarnaise etc. individually may require more time and waste more ingredients, but it cuts down order preparation time. These complex and delicate recipes take hours and special skill to prepare, so they are preserved and replenished when needed, notwithstanding quality and sanitary considerations. Fresh-made sauce Béarnaise raises the standard of quality and may be worth the wait while enjoying a conversation, sipping an aperitif or savoring Hors d'oeuvre in a high end restaurant, but such delay is unacceptable if fast service is expected.

A data warehouse serves "data" instead of food. Like recipes, data can be combined in predefined ways that don't change over time, making it possible to "precook" certain ingredients. Indexes, complex table joins and aggregations maintained ahead of time shorten the work of queries satisfying the myriad user requests at peak times.

Aggregate selection and implementation are cost-critical ways to ensure that the solution fits the box after other tuning and performance techniques reach diminishing returns. How about making your users happy while postponing that capacity upgrade to next year? The stakes are high. Improvement potential in capacity and efficiency are measured not in percentages, but in multiples, or even levels of magnitude.

How do we pick the most suitable aggregations? In contrast to haute cuisine, where wisdom has accumulated over centuries of practice and refinement, in the information

realm every situation is a different challenge with many more options and possible choices. Nevertheless, the thinking is strikingly similar:

Define a Strategy: Aggregates closer to the final result have narrower applicability and more of them will be required. Conversely the more basic, closer to base data they are, the wider the spectrum of their applicability, but also the longer the remaining work to be done at query execution time. We have to strike a balance between the gamut of requests helped and the degree by which they are helped.

Broaden the coverage: We often focus aggregates to the needs of "the high demand" requests. However in a high diversity ad-hoc environment, helping a broader (if not the complete) range of requests is likely to be more productive for total cost/performance. Consider that the 5% worse performing requests may consume over 35% of total server resources during stress times.

Effective coverage is expensive: There are limited resources and time for aggregate preparation. As it turns out, aggregates with more effective coverage of all possible requests, are costlier to build and maintain. Conversely, aggregates that are built on top of one another, making maintenance more efficient, leave more gaps in their coverage and many potential user requests are left without aggregate support.

Be ready to adjust as you go: Aggregate needs will change over time as new uses are introduced and existing ones are understood better. Two things are needed for this:

- Use a rational framework to analyze ongoing activity and assess if queries are using the aggregates expected. If there are better aggregates that are feasible what is the cost/benefit impact?
- Ensure that aggregates are implemented in a way that they can be changed without impact to applications. Adjusting the level of an aggregate should not require application changes; it should only require testing. This is not the case, from my experience, in many implementations today.

Exploit available automation: DBMS automatic maintenance and optimization as well as BI aggregate-awareness are capabilities to be exploited to the fullest possible, as long as they keep within acceptable bounds of availability and predictability.

Benchmarks: Generate and run benchmarks to evaluate performance for "envelopes" of query patterns and criteria, not just a handful of queries.

Imagine a systematic approach to the selection and implementation of aggregations in the data warehouse that addresses all of these points. Such is the VDM Aggregation Strategy, an InfoKarta service that assists customers in rationalizing, optimizing, implementing aggregates, but more importantly establishing the process to keep them effective. Customizable VDM tools and templates assist in various steps of the process. The Aggregation Strategy, ideally, will be an integral part of architecting a new data warehouse, but it can also be a time-boxed assessment to validate an existing aggregate design, and/or produce specific recommendations and a follow-up plan.

