Dashboard Performance – Data Do's and Don'ts

September 22, 2015 - Rebecca Gow

In the age of high-speed Internet, no one wants to wait for a page to load. So imagine, you've just launched a beautiful Logi Info dashboard…but it's taking so long to render that nobody is seeing its value. What can you do to alleviate slow response times, or better yet, prevent poor performance from the beginning? Here are our best practices to consider:

1. Everything in Moderation

Aim to give your users the data they need, *no less and no more*. When troubleshooting performance issues, it's important to remember that Logi Info can only render a report definition as fast as its slowest query or connection. The fewer queries in your report definition, the less each query returns and the simpler its structure, the faster the results.

Here's where to look when considering where to limit the data you send to your Info app:

- Volume of data returned by each query
- The number of tables or individual sources joined in each query
- The number of distinct data sources or data layers used by a definition

Understand what questions will be asked of the data and how the answers might be prepared before the data reaches Info:

- If your users are only interested in sales for North America, can you create a database view that is filtered for that region?
- If your dashboard receives data from a dozen spreadsheets prepared by different teams in the company, can these be combined into one file?

What about data analysts or other advanced users who want everything *and the kitchen sink*, even if that's millions of rows of data? Info's DataLayer.ActiveSQL can help if you are presenting a large volume of data to the Analysis Grid. Or consider giving this audience an option to download directly in their preferred format, such as Excel or CSV file, especially if they won't otherwise be frequent users of your app.

Process SQL	Database server took 0.002 seconds to process the query	.035
Processed Row Count	77	.035
Processed Data	View Memory Stream Data (12,004 bytes)	.035

When troubleshooting, Info's debugger trace report can tell you at a glance how long a query took to run, how many rows were returned and the size of the data file generated.

2. Make the Database Do the Heavy Lifting

Take advantage of what databases are designed to do, and do better than any Web technology. Where possible, handle aggregations, calculations and other complex data operations in the database, before Info ever connects to it. Again, here's where it helps to know what questions your users will be asking: i.e. whether they'll want to see revenue aggregated by fiscal year or the median age of support tickets logged last week. Based on that understanding, you might consider creating a view or table in which these figures are already calculated, or do the work in your query.

Obviously, this doesn't apply for sources such as text or Twitter feeds, where there's no out-of-the-box way to do the work in advance. Info's Calculated Columns and Aggregate Columns features can be invaluable for this situation, but you can have too much of a good thing. A single report definition that includes dozens of calculated columns and conditional operations is often a chief culprit behind slow rendering – each of these operations adds overhead on top of data loading. Aim to use these features sparingly and on the least rows possible.

3. Give Data Room to Breathe

As a general rule, host your data sources and your Info application on separate servers. This may seem an obvious tip, but it's not uncommon to start small with limited resources early on in an organization's analytics project or when testing a product. Down the road, mystery performance issues such as intermittent timeout errors and the same query executing over drastically different times can often be traced to a fight for resources on a strapped server.

Dream big when planning where your application will live and grow. Provision both database and Web servers for your deployment, and consider how your database and other sources will expand as your audience does. This is one practice best applied early – infrastructure limits can be hard to fix once you're already in production.

Also consider other demands on the data that may impact your queries:

- What other databases or applications share the server, and therefore its processing power and memory?
- Who else will be querying the data you need?
- How often will the data be updated? At what times of day?

Answering these questions early can help you forecast when you'll need to add resources as your dashboard takes off. Filtering and processing data at its source and giving it space early in your Logi Info project can not only ward off headaches in production, but set the stage for expansion as your application wins a wider audience.

Rebecca Gow is a Senior Solutions Architect on the Professional Services team at Logi Analytics

* Reprinted with permission from Logi Analytics.