Optimizing Query Performance with the Ontology Total_Num field

Optimizing Query Performance with the Ontology Total_Num field

Introduced in Core i2b2 Version 1.4

When the set finder algorithm creates an SQL query, it does so one panel at a time. First, the SQL is created for the first panel and it runs against the star schema to find a set of patients or encounters. This set is stored in a temporary table. The SQL is then created for the second panel and it runs against the star schema, and before a new temporary table is created it is joined to the temporary table of the prior panel. The resulting join will thus always yield an equal size or smaller patient set, and this is stored in the temporary table. Successive panels continue to be processes in this manner, each time creating an equal-sized or smaller temporary table. When there are no more panels to process, the last resulting temporary table is used to find the number of patients satisfying the query, or is output as a permanent patient set, or is used immediately for further analysis.

Performance is impacted heavily by the creation of the temporary tables. The larger the temporary table that is created, the worse the performance will be of the set finder query as a whole. Therefore, it is important and desirable to limit the number or patients that are placed in the temporary table. This can be achieved by surveying the panels at the beginning of the query as a whole to estimate which panel will likely result in the fewest patients, and create the SQL for this panel first. Since successive temporary tables will always be equal in size or smaller than the first temporary table, performance can often be greatly enhanced by this survey.

The survey for the size of the resulting temporary table for each panel uses the total_num column of the ontology table (or more precisely, the total_num item of the ontology XML message). The actual integer in total_num is used relatively to other total_num integers, and they are added together for each panel. Ordinarily, a good number to start with in this field is the number of patients in the database that have each specific concept. Queries will be storing patient sets in the temporary tables, and the optimized query will place the panel that is estimated in the survey to get the largest set of patients from the star schema last, and the panel that is estimated in the survey to get the smallest set of patients first.

For example, let's say that one wants to perform a query like

Patients over 18 years old

AND

Female patients

AND

Patients with a diagnosis of Rheumatoid Arthritis

AND

Patients who ever took the medication Retuxan

The total_num field of the various concepts in a database of 5 million patients would typically be numbers such as:

Ontology concept	Total_num
> 18 y/o	4500000
Female	3000000
Rheumatoid Arthritis	200000
Retuxan	5000

In the native order of the panels, the SQL will result in the following temporary tables as the query is run:

Patients over 18 years old (temporary table = 4,500,000)

AND

Female patients (temporary table ~ 3,000,000)

AND

Patients with a diagnosis of Rheumatoid Arthritis (temporary table ~ 100,000)

AND

Patients who ever took the medication Retuxan (temporary table ~ 110)

If one examines the total_num integers, it becomes apparent that the Retuxan would be the best panel to create the SQL for first. The SQL for the Retuxan panel will result in a temporary table of 5000 patients, and this will take much less time to write than the native order of the panels, which would create SQL to find the 4.5 million patients over 18 y/o first, and place them in a temporary table. By using the total_num field, the query is reordered as:

Patients who ever took the medication Retuxan (temporary table = 5000)

AND

Patients with a diagnosis of Rheumatoid Arthritis (temporary table ~200)

AND

Female patients (temporary table ~120)

AND

Patients over 18 years old (temporary table ~110

This strategy gives the best chance of creating the smallest temporary tables in successive joins (assuming independence). Note that because the ordering is relative, it is plausible to put best guesses into the total_num field. A reasonable set of guesses may have populated the "greater than 18" field with 4,000,000, and the "female" total_num field with 2,500,000, and the other total_num fields may have been left blank. The resulting order would have been the same for the first two query panels, and random for the last two. This would still result in most of the increased query performance from populating the total_num field manually with a few guesses regarding the query items likely to represent a large fraction of patients in the database.

Addendum for Version 1.6

In Version 1.6, the set-finder begins to support the option of performing a query to find items occurring in the same encounter, besides just occurring in the same patient. The temporary tables may now contain encounter numbers rather than just patient numbers. If many of these types of queries are anticipated, an optimization strategy may include number of encounters for each concept in the total_num field rather than the number of patients.