



Networking Activity D

Task 3.5 Development of a Novel Cache Prototype and its Rapid Implementation

Deliverable 3.5.1

Design Study for a Novel Cache Prototype

Jörg Holetschek
Botanischer Garten und Botanisches Museum Berlin-Dahlem,
Freie Universität Berlin

18 August 2005

The Synthesys cache prototype

Aim

Currently, the BioCAsE user interface (available at search.biocase.org) can be used to query distributed data providers. When a search is performed, a query is sent off to all relevant providers, the results are collected and list of hits is produced by the interface. This in-time-distributed query has some disadvantages regarding performance and functionality.

The Synthesys cache is supposed to overcome these and to provide a faster search process as well as additional search opportunities. It will hold a snapshot of the most important fields of all records available in the BioCAsE network (and even more). In the first phase of this development, a prototype data model for the cache has been developed at the Botanical Museum in Berlin.

Design considerations

It is based on the GBIF index as provided by the GBIF secretariat in Copenhagen, which is kept in a MySQL database at a server in Berlin. The updates made to the index in Copenhagen are reproduced to this instance via MySQL replication. Therewith, it's kept up-to-date with no noticeable delay.

However, this index data model has been developed for the GBIF data portal and doesn't quite meet the requirements for the Synthesys cache. For Synthesys, the main needs are:

- The cache must contain all fields that are search criteria in the user interface and are to be viewed in the result list.
- Furthermore, it must enclose all fields that are required to identify a certain record within the network (provider name, collection name and unit id).
- Queries on the cache have to run in a reasonable time. The most frequently used search criteria are the identification and geography (country), so the focus of performance considerations is on these queries.

Design decisions

To meet these, a different data model was chosen. The needed portions of the GBIF indexed are extracted via SQL scripts and stored in 4 tables. After some rapid prototyping, the following decisions have been made (amongst others):

- The main table contains all units in the network. This results in a very large table (> 1GB) and even larger indexes, which lead to poor performance when searches are done directly done to this table.
- Therefore the fields that are most likely to be searched are kept in extra tables. These are very small, so the can be kept in memory by the MySQL server and searches are extremely fast.
- These lookup-tables don't contain a record for every unit, but a one record for every possible value. So, for example, the country table contains only 239 entries. Even the table which holds all identifications (as specified in the unit record) has only about 1 million entries and a size of 23MB, which means that it will completely reside in the server's memory in most cases.
- The main table contains references to the lookup tables and all the fields that are not suitable for moving to single tables (e.g. latitude/longitude). Indexes are kept for only the key fields that are used in JOINS.
- For the lookup table for identification, an index on the identification column would grow larger than the table itself would only be valuable for equality searches (e.g. "="

Abies alba”) or pattern searches with wildcards at the end of the pattern (“Abies%”). Moreover, since the table is kept in memory, a table scan can be performed within a short time. So a decision has been made not to index this column.

- Once the cache is created, its content is only read. So it’s possible to make all tables “compressed” in MySQL. By this, they get read-only, and their size is reduced enormously. The main table, for example, is shrunk from 2.3GB to 1.3GB, the identification lookup table from 40MB to 23MB. So full table scans are sped up considerably, regardless whether they’re performed in memory (for identification lookup table) or on disk (for the main table).
- Usually, the query optimizer of the SQL server decides how to execute a query. But for the cache, wrong decisions would be disastrous for the performance of a query. So the queries should use the STRAIGHT_JOIN option, which instructs the SQL server to perform the join exactly in the specified order. Therewith it can be ensured that the server will process the lookup tables first and a full table scan of the main table is avoided.

Results

With the first prototype of the cache, the performance results of sample searches are quite encouraging:

- The most frequent type of search is based on the identification. In most cases, they run very fast (in less than one second), even if a pattern search (e.g. “%tessmannii%”) is performed.
- Searches that combine search criteria that are kept in lookup tables (e.g. identification and country) take slightly longer, but also run very fast.
- Searches that combine criteria that are kept in lookup tables with criteria kept in the main table (e.g. gathering date) run fast, as long as the criteria that affects the lookup table is not too unspecific.
- Searches that only contain criteria kept in the main table can take some time.

Generally, the problem of unspecific searches can be tackled by using the LIMIT clause. Therewith, also rather unspecific queries can be processed in a reasonable time.