Benchmarking Advanced Features in Database Systems

András Gábor

Database systems play a fundamental role in information technology ever since the early days. Most often relational databases are used. There has been always a need for benchmarks that helped in deciding on the appropriate RDBMS for a project.

The traditional database benchmarks measure the performance and stability of the relational capabilities in DBMSs. As today most products perform the most basic relational features reliably and fast enough to fulfill most application needs, the database development efforts have shifted towards providing full-featured DBMSs. However developers don't tend to use these features, because of lack of knowledge and mistrust.

The usage of the implemented widely-usable features could save development time, increase maintainability and evolvability and sometimes performance as well. In this paper we will enumerate such features like object relational and other SQL:1999 extensions and propose a benchmark approach to rate them. The good benchmark examples and results can popularize these functionalities.

The proposed features benchmark approach gives bases for comparing different implementations. This is done via specifying typical application scenarios where a concrete set of features can be used beneficially. In these application scenarios multiple implementations can be used traditional ones and those using one or more features. Every implementation can then be tested for performance and this gives a comparison metric. However the final rating must include an adjustment factor representing the assumed development and maintenance benefits gained from the usage of the incorporated features. This simple benchmark framework can be used to specify tests for different functionalities.

In our paper we also explain the following available features: collection types, object types and object tables, virtual private database, domain indexes, table functions, aggregate clauses and analytical functions.

Then we specify a concrete benchmark for collection types, where we will have two implementations: one with a traditional database model and the other using nested tables as collection types. We define the transactions to be run against these databases and the distribution of their workload. We also specify the adjustment factors for each implementation. We estimate what results we may get and how to interpret them. For instance a badly performing nested table (a collection type) implementation can show that in our system the implementation of this feature is not mature enough.

After specifying the benchmark in theory we show some actual benchmark results achieved in different size databases and different database products (e.g.: Oracle, Postgres).

Finally a conclusion and future work plan is provided.