the 'Export to database...' option in the 'Extras' menu. This command prompts for the name of the log file to be exported (the last log file written is given as the default). After pressing the 'OK' button it invokes the `dabconv` utility to convert the log file into a special delimited format which is then uploaded to the database using the `dabload` shell script. Thus to provide an interface to other databases, only these two external utilities must be rewritten.

Note that the database interface has been designed especially for log files written by the kind of test macros created with the test configuration dialog. It will *not* work with ordinary log files created manually or with some handcoded test macro.

The `dabconv` program converts the DABTool log file to a standardized data structure which makes it easier to extract specific parameters and relate them to each other. The data structure currently consists of two tables:

- The `TEST` table contains for each test run the corresponding test id (which is also used as the primary key of the table), a descriptive comment and the random seed.

- The `PARAM` table contains the test results and other parameters such as various statistics about the generated problem instances. The first field in this table is the test id which matches the corresponding field in the `TEST` table. It is followed by a collection of index fields for numbering the parameters taken from the log file (these numbers are actually generated by the test macro, not by the DABTool program itself). The numbering scheme is described in more detail below. The last two fields contain the actual parameters as key/value pairs. Typically a selection of certain related parameters will be selected from the database to perform some kind of regression analysis or compute average parameter values. A plethora of different parameters is supported, too many to list them here; most of these can be found in the `IOParam.dat` file included in the installation of the DABTool program, which is also used by the `dabconv` program to guide the translation process.

The index fields in the `PARAM` table provide a hierarchical numbering of the logged parameters which allows to relate them to each other. There are five such fields, from which four are actually used in the current implementation; the fifth field is reserved for future extensions. All entries in the index fields are nonnegative; only nonzero entries are significant. The meaning of the four numbering levels currently supported is as follows:

- Level 1: running number of the generated area graph.

- Level 2: running number of the requirements (unit, random, random areas, etc.) generated for a given graph.

- Level 3: running number of the generated ensemble assignment (unit/SFF/GFF) for a given problem instance (graph/requirements pair).

- Level 4: running number of the coloring/clique algorithms applied to an ensemble assignment.