Query Optimization

1. Exercise

Due 28.04.2014, 9 AM

submit via email (`Andrey.Gubichev@in.tum.de`)

Exercise 1

Suggest SQL queries for the following problems, then manually translate the queries into execution plans, and execute them using the *tinydb* system.

- Find all students that attended the lectures together with *Schopenhauer*, excluding *Schopenhauer* himself.
- Find all professors whose lectures attended at least two students.

Hint: you can use the function evaluation operator *Chi* in *tinydb*, see an example in the *tinydb*'s source code.

Exercise 2

Implement a program that can parse SQL queries of the following form:

select (*|*attribute*(,*attribute*)*)
from *relation binding*(,*relation binding*)*
where *binding.attribute*=(*binding.attribute*|*constant*)
(**and** *binding.attribute*=(*binding.attribute*|*constant*))*

- make sure that the query is semantically correct, i.e., all relations and attributes exist
- store the query structure in a format suitable for simple plan construction. For example (just a suggestion):

Query:

relations: list of relation names
selections: list of attribute-access/constant pairs
joinconditions: list of attribute-access/attribute-access pairs

Hints for the Runtime System

We use the tinydb runtime system for experiments (links are included in this document). Its C++ version requires a (not very old) C++ compiler (gcc 4.6 is known to work).

Installing the C++ version

1. Make sure that a recent C++ compiler is installed. For POSIX systems, check your distribution or download from the GCC home page. For Window, download gcc 4.5 from MinGW (which is unfortunately a pain) or use our pre-packed bundle. Make sure that g++ is in the PATH and working.

2. go to the unpacked tinydb source code, build by calling `make` (*mingw32-make* under Windows)

3. load a sample database by `cd data && ./loaduni` (call *loaduni.cmd* under Windows)

4. test the example programs (i.e., `./bin/scanexample`)

5. look at the source code in *examples* to see how the system is used