M1 Project - Databases

Simon Mauras

March 29, 2016

Abstract

This report describes the implementation of a small SQL-based data exchange engine. It is part of the evaluation of a master's degree lecture on databases at ENS Lyon.

1 Implementation

Here is a quick description of the architecture of the implemented tool:

File	Description
`Makefile`	Makefile, run `make` to install
`src/main.ml`	Handles I/O and calls to the other modules
`src/sigs.ml`	Definitions of types and useful functions
`src/lexer.mll`	OCamllex source file (lexer)
`src/parser.mly`	OCamlyacc source file (parser)
`src/skolemizer.ml`	Sanitize the mapping (remove existencial variables)
`src/sql_generator.ml`	Generate SQL queries from a mapping

The lexer and the parser are straighforward since a BNF grammar was given. However we choose to handle `SOURCE`, `TARGET` and `MAPPING` as keywords, so they must not be used as names.

The skolemization is done with two passes, the first one to get the name of the existencial variables and the second one to rename them. We use sets an maps rename variables, and tail-recursion to speed-up the execution.

For the generation of SQL queries we had to handle carefully several points. If a source relation is used twice in one rule, then we must rename it. If a constant appears on the left hand side of a rule, then it is a constraint on the rows selected from the source.

2 Installation

Our data exchange tool has the following dependancies:

- OCaml environment (`ocamlbuild`, `ocamlc`, `ocamllex`, `ocamlyacc`)
- OPAM packages `ocamlfind` and `sqlite3`

For debian-like systems:

```
apt-get install opam
opam install ocamlfind sqlite3
make
```

3 Functionalities

Here is a small description of the functionalities of the implemented tool. By default the program `DataExchangeSQL` reads the description of a mapping on the standard input and write a sequence of SQL queries on the standard output. The following options are available:

Option	Description
`-help`	Display the list of options
`-input <file>`	Read the description of the mapping from `<file>`
`-output <file>`	Write the sequence of SQL queries in `<file>`
`-sqlite3 <file>`	Execute the queries on the SQLite3 database `<file>`