Содержание

HaskellDB

Nikita Yurchenko

National University of Radioelectronics

30.12.2016

Обзор

- HaskellDB - Библиотека комбинаторов для языка Haskell, которые реализуют операторы реляционной алгебры типобезопасным и декларативным способом.
- Попытка DSL (Domain Specific Language) на базе TH (Template Haskell).
- Можно работать с БД не зная SQL, а только реляционную алгебру и Haskell.
- Изначально написанная Эриком Мейером библиотека использовала бекенд ADO и была игрушкой теоретиков (впрочем, ей она и осталась).

Плюсы и минусы

С изменяемыми данными (в т.ч. БД) в Haskell из-за чистоты языка все традиционно обстоит довольно плохо.

+ Интересно и необычно
+ Вкусный Haskell
+ Вкусная реляционная алгебра
+ Ошибки сводятся на нет благодаря типам
+ MySQL, PostgreSQL, SQLite

- Никому не нужно
- Плохо переваривает особенности конкретных СУБД
- Тормозит
- Проблемы с ленью языка

Функции реляционной алгебры

```
1 restrict $ emps!E.xid .>. constant 2
2 project $ E.name << emps!E.name
3 intersect query1 query2
4 union query1 query2
5 minus query1 query2
6 divide query1 query2
```

Агрегатные и вспомогательные функции

```
1  top 10
2  unique
3  project $ E.xid << variance (emps!E.xid)
4  project $ E.xid << count (emps!E.xid)
5  project $ E.xid << avg (emps!E.xid)
6  project $ E.xid << _min (emps!E.xid)
7  restrict $ emps!E.xid .==. (cast "INT" (constant "2"))
8  order [(desc emps E.xid), (asc emps E.name)]
```

Типы

Rel	Типизированное отношение
Attr	Типизированный атрибут
Query	Тип запроса (создается в монаде функциями)
Table	Самый непосредственно относящийся к самой БД тип
Expr	Тип выражения, инкапсулирует PrimExpr

Подключение к БД (описание БД)

Предположим, имеется некоторая SQLite БД с таблицей emp(id, name). Создадим файл createLayout.hs, скомпилируем и запустим его

```
1  module Main where
2  import ...
3
4  dbdescr = DBInfo "Base" (DBOptions False mkIdentPreserv
5      TInfo "emp" [
6          CInfo "id" (IntT, False)
7        , CInfo "name" (StringT, False)]
8    ]
9
10 main = dbInfoToModuleFiles "." "Base" dbdescr
```

[2] Подключение к БД (простой запрос)

```
 1  module Main where
 2  import ...
 3  import qualified Base.Emp as E
 4
 5  getAllEmps db = query db $ do
 6    emps <- table E.emp
 7    project $ copyAll emps
 8
 9  withDb = sqliteConnect "DB.db"
10
11  main = do
12    res <- withDb getAllEmps
13    print $ res
```

Изменение данных

Расширим наш Main.hs, добавим новые функции

```
1  addEmp :: Int -> String -> Database -> IO ()
2  addEmp id name db = insert db E.emp $
3      E.xid <<- id
4    # E.name <<- name
5
6  updEmp :: Int -> String -> Database -> IO ()
7  updEmp id name db = do
8    update db
9           E.emp
10          (\e -> e!E.xid .==. constant id)
11          (\e -> E.name << constant name)
```

Ссылки

- github.com/zelinskiy/haskelldb
- hackage.haskell.org/package/haskelldb
- hrisdone.com/posts/haskelldb-tutorial
- https://github.com/MasseR/haskelldb-example
- citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.3828&rep=rep1&type=pdf