Sirenial

Type-safe SQL

Martijn van Steenbergen
Dutch Haskell Users Group Day
17 April 2010

Design goals

- Moderately type-safe
- Simple types (simple type errors)
- Modular
- Predictable SQL generation
- Work with existing databases
- Transparent merging of similar SELECT queries

Example

```
getTownName :: Ref Db.Town -> Query String
getTownName townId = do

  [townName] <- select $ do

    t <- from Db.tableTown
    restrict (t # Db.townId .==. expr townId)
    return (t # Db.townName)

  return townName
```

```
> execute $ getTownName 1
*** Executing query:
      select t0.townName
      from towns t0
      where (t0.id = 1)
"'s gravenmoer"
```

Example

```
getTownName :: Ref Db.Town -> Query String
getTownName townId = do
  [townName] <- select $ do
    t <- from Db.tableTown
    restrict (t # Db.townId .==. expr townId)
    return (t # Db.townName)
  return townName
```

Select monad

Query monad

```
> execute $ getTownName 1
*** Executing query:
      select t0.townName
      from towns t0
      where (t0.id = 1)
"'s gravenmoer"
```

Types

```
module Sirenial.Tables where
  newtype Table t   = Table { tableName  :: String  }
```

```
module TownTable where
  data Town
  tableTown = Table "towns"                :: Table Town
```

Types

```
module Sirenial.Tables where
  newtype Table t   = Table { tableName  :: String  }
  data    Field t a = Field { fieldTable :: Table t
                            , fieldName  :: String  }
```

```
module TownTable where
  data Town
  tableTown = Table "towns"                :: Table Town

  townName  = Field tableTown "townName" :: Field Town String
```

Types

```
module Sirenial.Tables where
  newtype Table t   = Table { tableName  :: String  }
  data    Field t a = Field { fieldTable :: Table t
                            , fieldName  :: String  }
  newtype Ref t     = Ref   { getRef     :: Integer }


module TownTable where
  data Town
  tableTown = Table "towns"                :: Table Town

  townName  = Field tableTown "townName" :: Field Town String
  townId    = Field tableTown "id"       :: Field Town (Ref Town)


module TownQueries where
  import qualified TownTable as Db
```

Merging queries

```
> execute $ (,,) <$> getTownName 1
                 <*> getTownName 2
                 <*> getTownName 3

*** Executing query:
      select t0.id, t0.townName
      from towns t0
      where t0.id in (1,2,3)
("'s gravenmoer","'s-graveland","'s-gravendeel")
```

Merging queries

```
> execute $ (,,) <$> getTownName 1
                 <*> getTownName 2
                 <*> getTownName 3

*** Executing query:
      select t0.id, t0.townName
      from towns t0
      where t0.id in (1,2,3)
("'s gravenmoer","'s-graveland","'s-gravendeel")

> execute $ for [11..20] getTownName
*** Executing query:
      select t0.id, t0.townName
      from towns t0
      where t0.id in (11,12,13,14,15,16,17,18,19,20)
["'s-heerenbroek","'s-heerenhoek","'s-hertogenbosch","'t
goy","'t haantje","'t harde","'t loo oldebroek","'t
veld","'t waar","'t zand nh"]
```

Merging queries (2)

```
getTownIdsForAd   :: Ref Db.Ad -> Query [Ref Db.Town]

getTownNamesForAd :: Ref Db.Ad -> Query [String]
getTownNamesForAd adId = do
  tids <- getTownIdsForAd adId
  for tids getTownName
```

Merging queries (2)

```
getTownIdsForAd   :: Ref Db.Ad -> Query [Ref Db.Town]

getTownNamesForAd :: Ref Db.Ad -> Query [String]
getTownNamesForAd adId = do
  tids <- getTownIdsForAd adId
  for tids getTownName

> execute $ getTownNamesForAd 202
*** Executing query:
      select t0.adId, t0.townId
      from ads_weeks t0
      where (t0.adId = 202)
*** Executing query:
      select t0.id, t0.townName
      from towns t0
      where t0.id in (890,986)
["hilversum","jabeek"]
```

Merging queries (3)

```
> execute $ getTownNamesForAd 202
["hilversum","jabeek"]
> execute $ getTownNamesForAd 230
["jabeek",”zwolle”]
```

Merging queries (3)

```
> execute $ getTownNamesForAd 202
["hilversum","jabeek"]
> execute $ getTownNamesForAd 230
["jabeek",”zwolle”]

> execute $ for [202,230] getTownNamesForAd
*** Executing query:
      select t0.adId, t0.townIdCopy
      from ads_weeks t0
      where t0.adId in (202,230)
*** Executing query:
      select t0.id, t0.townName
      from towns t0
      where t0.id in (890,986,2454)
[["hilversum","jabeek"],["jabeek","zwolle"]]
```

Try it yourself

- Sirenial is available online: http://code.google.com/p/sirenial/
- Feedback is much appreciated!