

SQLSourcery

The logo for SQLSourcery, featuring a horizontal bar with a teal segment on the left and an orange segment on the right.

:: Project Design Presentation
:: Kant, Adrian
:: Murphy, Taylor

(provide
 #%project-begin
 (all-from-out cs-4620))

Summary / Motivation



Idea: An ORM for mapping structures into a SQL database

Programmer Motivation:

- ❑ Saving functional program state without writing I/O boilerplate code
- ❑ Gives benefit of partial program execution save from interruptions

Language Format: Racket-like language behavior with added constructs

Language Concepts



Programmers can load in previously saved structures, add structures to the database, modify structures, and delete structures from the database.

Language accomplishes this through:

- ❑ Underlying numeric ID's per structure that map to primary keys in db
- ❑ At runtime, structure use reference these ID's

Programmers must:

- ❑ Only use existing records in database
- ❑

Grammar

SQLSourceryProgram = Database (Definition | SourceryStruct | Expression)...

Database = (sourcery-db String)

Definition = (define Variable Expression)
| (define (Variable Variable...) Expression)

SourceryStruct = (sourcery-struct Variable [(Variable Type)...])

Expression = Variable
| Value
| (Primitive Expression...)
| (Variable Expression...)
| (cond [Expression Expression]... [Expression Expression])
| (cond [Expression Expression]... [else Expression])

Scoping



Same as BSL

Program Example

```
(sourcery-db "path/to/database.sqlite")  
(sourcery-struct user [(name String) (grade Integer)])  
  
(define original-users (sourcery-load user))  
(define new-users (list (user-create "Matthias" 1)  
                        (user-create "Ben L" 0)))  
  
(define users (append new-users original-users))  
  
(define (is-failing? u)  
  (= (user-grade u) 0))  
  
(define (up-grade u)  
  (user-update u (user-name u) (+ 1 (user-grade u))))  
  
(define failures (filter is-failing? users))  
  
(set! users (sourcery-filter is-failing? users))  
  
(sourcery-map up-grade failures)
```

;; Use existing database or create
;; map users to db with given types

;; loads state from db
;; add two records to db

;; does not change database

;; User -> Boolean
;; Purpose: determine if failing

;; User -> User
;; Purpose: updates user grade

;; does not update DB

;; Remove from db, users still
;; tracks the db structures

;; throws an error: a given user does
;; not exist

References and Database Mapping

```
(define (up-grade u)
  (user-update u (user-name u) (+ 1 (user-grade u))))
```

```
(define ben (user "Ben L" 2))
;; ben - reference to database: ID: 1 - values ("Ben Lerner" 2)
```

```
(define bean (user-update ben "Bean Learner" 2))
;; ben and bean - reference to database: ID: 1 - values ("Bean Learner" 2)
```

```
(grade-up ben)
;; ben and bean - reference to database: ID: 1 - values ("Bean Learner" 3)
```

```
(grade-up bean)
;; ben and bean - reference to database: ID: 1 - values ("Bean Learner" 4)
```

ben

bean

```
;; with both print same thing - (user "Bean Learner" 4)
```

Github Link



<https://github.com/adjkant/sql-sourcery>