

Relational DataBase

Arturo Chinchilla Sánchez, Malcolm Davis Steele

April 8, 2016

Abstract

There are several paradigms of programming, object-oriented paradigm, Imperative, functional, logical, etc. For purposes of this project, is be used the functional programming paradigm, in conjunction with the Racket programming language , which belongs to the family of Lisp, as a functional language that it is. The Functional Paradigm as it's name says, used functions for all the operations and integrates math functions for operate the data. Internet is currently used for multiple purposes, but one of the primary, and which any Internet user is saved, is to do a search on your favorite search engine, search for the name of a song, the name of a plant, sound of an animal or to see that movies are on the undercard of film, from where we get this information? We obtain databases stored on Internet servers, to which we have access through the network. The databases contain information of all kinds, which are related to each other. A database can be from the list of employees of a company to a list of scientific names of trees. But how programming paradigms relate to databases? the answer is very simple, databases are created using computer programs such as programming languages, which use paradigms, to carry out their tasks. So in this project we will create a relational database called "RELDB" using Racket and functional paradigm.

Contents

1	Introduction	4
2	User Guide	5
2.1	Software and Hardware Requirements	5
2.2	How to use:	5
3	Development environment	6
3.1	DrRacket	6
3.2	Git and Github	7
3.3	Zube.io	7
3.4	CodeShare.io	9
3.5	Overleaf	9
4	Data Structures and functions	10
4.1	Lists	10
4.2	List Implementation.	11
5	Student activity log	11
5.1	Arturo's Time-Sheet	11
5.2	Malcolm's Time-Sheet	12
6	Project final status	12
6.1	Issues, Limitations or Challenges	13
6.2	Known Issues	13
7	Conclusions, Suggestions and Recommendations	13

List of Figures

1	Example addt.	5
2	Example ins.	6
3	Example addr.	6
4	DrRacket Interface.	7
5	GitHub Interface.	8
6	Zube Dashboard.	8
7	CodeShare Interfaces.	9

8	Overleaf Interface Code-Compiled.	10
---	---	----

1 Introduction

In computing the functional paradigm is a declarative programming paradigm, based on the use of functions, which are first-class citizens, which go hand in hand with recursion. This paradigm has its beginnings in the simplest programming language, "The Lambda Calculus" which was a system developed in the 1930s for research and application of functions and recursion. Various programming languages are based in this paradigm, such as Lisp, Scheme, Erlang, Rust, Objective Caml, Scala, F # and Haskell. Although the functional paradigm consists only of definitions of functions, actually, many programming languages that use this paradigm, have implemented functions that don't belong to the paradigm, which are called impure functions, and therefore are also called impure languages that incorporates these. Racket is a programming language, which belongs to the large family of Lisp, although it is multi-paradigm, it is generally used in the functional paradigm. Developed in 1994 by PLT Inc. as an pedagogic programming environment based on Scheme, and MrEd, the first virtual machine for Racket was created. Later they developed DrScheme with PLT Scheme as the primary development language. Racket is very flexible, even without the use of dialects. Features such as the use of macros, tail recursion, and much more, allow it to be used to perform all kinds of tasks, from generation graphics web scrapers. Additionally, the powerful macro system allows developers to control all aspects of a language, which was one of the main goals behind its design.

A database is a collection of information containing information on various topics, and they are categorized in different ways, but they are related to each other. All these data are stored for later use. If we see it in this way, a library can be considered a database. Today with the intensive use of electronic devices such as computers, smart phones, etc, besides the Internet, have been created thousands of digital databases, and have been stored on servers which are accessed via the web, for ensure that information reaches more people and of a faster way.

2 User Guide

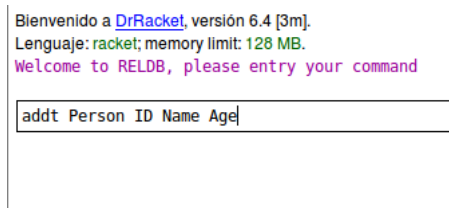
2.1 Software and Hardware Requirements

- Operating System: OS based on Linux (Recommend Linuz Mint 17.2)
- CPU: Intel Dual Core at 2GHz or equivalent
- RAM Memory: 2GB (Recommend 4 or more)
- DrRacket IDE installed

2.2 How to use:

For the Data Base use the following commands:

- addt or addtable for add tables in the Data Base. Example:
addt Person ID Name LastName
addtable Person ID Name LastName
When ID it's the Primary key. Take care, if you don't entry the Primary key the program don't allow you to insert the table. See Fig 1



```

Bienvenido a DrRacket, versión 6.4 [3m].
Lenguaje: racket; memory limit: 128 MB.
Welcome to RELDB, please entry your command

addt Person ID Name Age
```

Figure 1: Example addt.

- ins or insert for insert registers into a table. Example:
ins Person 12345 Dereck Molloy
insert Person 12345 Dereck Molloy
Then the command you give the table name that you want to insert the data. Take care, if you don't give the correct data, the program don't allow you to insert the registers. See Fig 2

```
Bienvenido a DrRacket, versión 6.4 [3m].  
Lenguaje: racket; memory limit: 128 MB.  
Welcome to RELDB, please entry your command  
  
ins Person 123 Dereck 25
```

Figure 2: Example ins.

- addr or addReference for create references.
addr Car Owner Person
addReference Car Owner Person
Note that both tables should exist. See Fig 3

```
Bienvenido a DrRacket, versión 6.4 [3m].  
Lenguaje: racket; memory limit: 128 MB.  
Welcome to RELDB, please entry your command  
  
addr Car Owner Person
```

Figure 3: Example addr.

- remr or removeReference for remove the reference.
remr Car Owner Person
removeReference Car Owner Person
The reference should exist for be removed.
- rr or remover to remove registers of the table
rr Person 12345
remover Person 12345
You should indicate the table name and the Primary key.

3 Development environment

3.1 DrRacket

The Racket platform provide the "DrRacket" tool, which it's an Integrated Development Environment, programmed in Racket language,

which will facilitate the task of programming in Racket. It also offers `raco` a tool for command line that will allow us to install packages or compile libraries. See Fig 4

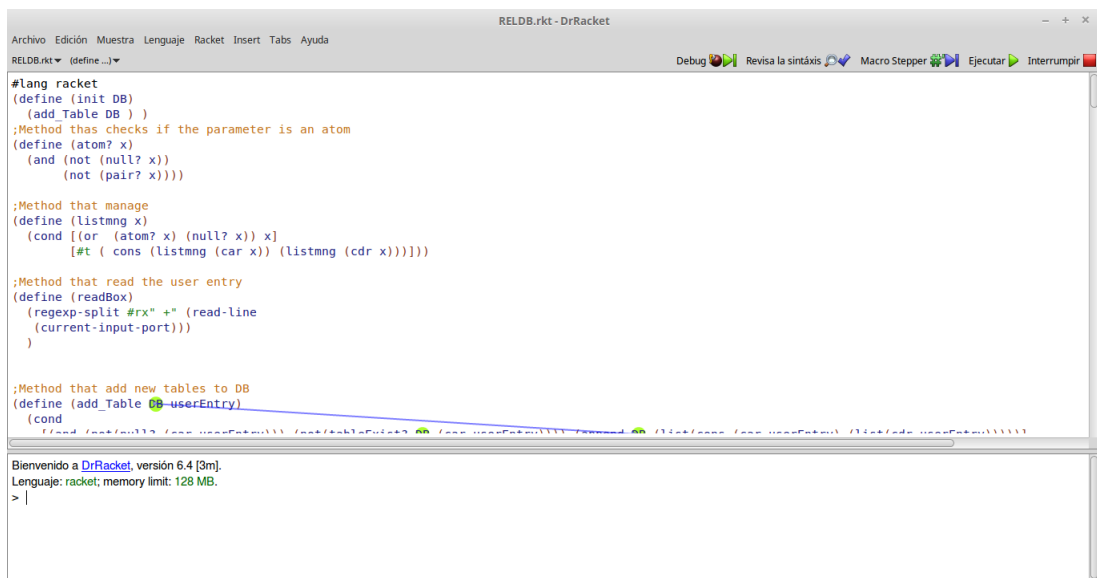


Figure 4: DrRacket Interface.

3.2 Git and Github

To store the code in a practical way and have a versioning control Git is used for creating repositories and GitHub as service and cloud storage. This code manager (Git) let us to create branches for each member of the group and work separately, change the branch work, see and edit code of another branches, make commits for later if you want to go back in your code and when each part needed is done they can be merged into master branch to have a final version of the code. See Fig 5

3.3 Zube.io

Zube is a free online service that provides the ability to organize, divide and manage a number of tasks by using a board, in addition to divide

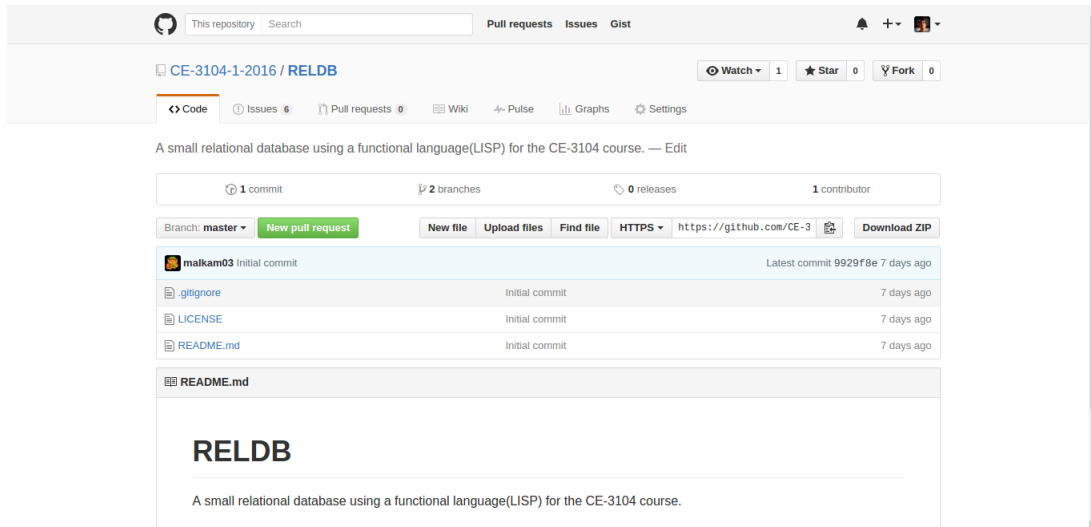


Figure 5: GitHub Interface.

tasks by individual, organized by tasks remaining to be done, in process and finished . besides being able to put delivery dates and stages of the project. One of the possibilities Zube offers is the ability to connect to GitHub. See fig 6

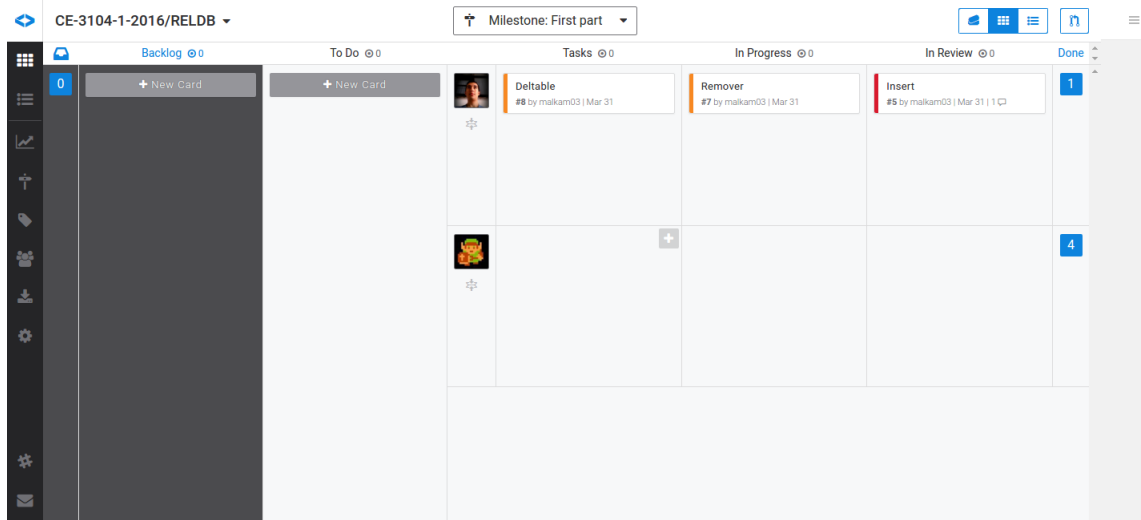


Figure 6: Zube Dashboard.

3.4 Codeshare.io

CodeShare is a free online service that lets you share text in real time, but that is optimized for code sharing programming languages. This tool allows us to create a personalized web address (within your domain "https://codeshare.io/example") where anyone can enter, view and edit the shared material. It also has the option to create chats and video calls. It is a quick and easy way to share ideas and / or information with our workmates. See fig 7



```
1 #lang racket
2 (require math/number-theory)
3 (define entryList '((((Reference_Table Reference_Column) Referenced_Table)
4                       ((Car Owner) Person)((table column) originalTable))
5                       (Table_Name (Table_Format)(Table_Registers))
6                       (Person (ID Name Age) (7115689 Malcolm 21) (1468987 Arturo 22))
7                       (Car (Licence_Plate Owner Brand) (648854 7115689 Nissan) (AC5895 1468987 Audi))
8                       (Mechanic (ID Data Car))
9                       (Person (ID Nombre Age) (7115689 Malcolm 21) (1468987 Arturo 22))))
10 (define badEntry1 '((((Reference_Table Reference_Column)() Referenced_Table)
11                      ((Car Owner) Person()))()
12                      (Table_Name (Table_For)(mat)(Table_Registers))
13                      (Person (ID Name Age)()) (7115689 Malcolm 21) (1468987 Arturo 22))
14                      (Car (Licence_Plate Own()er Brand) (648854 7115689 Nissan) (AC5895 1468987 Audi))
15                      (Person (ID Nombre Age) ()) (7115689 Malcolm 21) () (1468987 Arturo 22))))
16 (define badEntry2 'a)
17
18
19 ;Method to add reference to the DB
20 (define (addReference DB userEntry)
21   (cond
22     [(not (eq? (length userEntry) 3)) (display "The user entry is not valid.") DB]
23     [(eq? DB null) (display "The database is empty.") DB]
24     [(not (tableExist? DB (car userEntry))) (display "The table doesn't exist.") DB]
25     [(not (empty? DB (car userEntry) 0)) (display "The table isn't empty.") DB]
26     [(not (tableExist? DB (caddr userEntry))) (display "The source table doesn't exist.") DB]
27     [(not (columnExist? DB (car userEntry) (cadr userEntry) 0)) (display "The column doesn't exist.") DB]
28     [(reference? (car DB) (car userEntry)) (display "The table already have a reference") DB]
29     ;Other verifications
30     [#t (cons (cons (cons (cons (car userEntry) (cadr userEntry)) (caddr userEntry)) (car DB)) (cdr DB))]
31   )
32 )
```

Figure 7: CodeShare Interfaces.

3.5 Overleaf

Free online tool used to write academic documents. This service allow create, edit and share your documents easily using \LaTeX . Provide two principal windows, in the first you can write your document, obviously using the \LaTeX syntax, and the second you can see your compiled document. \LaTeX also is composed of a large set of macros getting documents have a typographic high quality. Therefore it is widely used for the creation of academic papers, theses and technical books, since the

typographic quality and made with \LaTeX documents is comparable to that of a great scientific publishing. See Fig 8

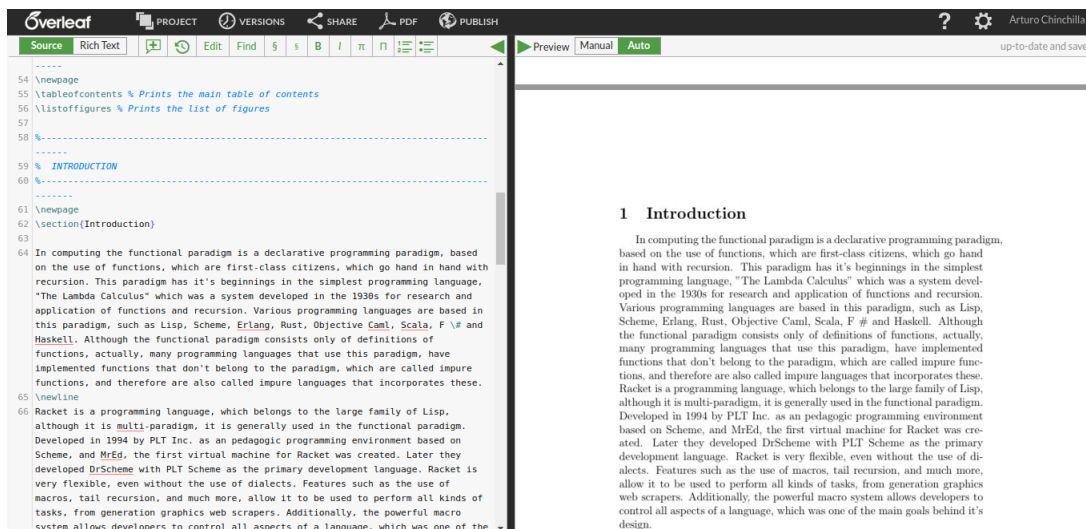


Figure 8: Overleaf Interface Code-Compiled.

4 Data Structures and functions

4.1 Lists

The lists originally were the principal storage structures for programming languages that use the Functional Paradigm, like LISP or SCHEME, this while they were still completely pure, in fact Lisp is named for the acronym "LISt Processing". Actually it's have been modified / adapted to give more flexibility to the programmer, introducing features that not belong to the functional paradigm, making impure the languages. Functions like the use of instantiated variables, the cycles "for" and "while", etc.

4.2 List Implementation.

Since this project is about the functional paradigm, the list is the only data structure used because it could not use impure aspects of language. The database uses this list to store all the data that users enter, the implementation is a list of lists, where the first sub-list is a list of references and all other lists are tables with their respective parameters and records. An examples of such a lists would look like is shown.

1. A null list
'[()]'
The center list is the space for references
2. A list with tables but no references
'[() (Table1 (Param1...Paramx)) (Table2 (Param1...Paramx))]'
The references list is null, and the others tables are full.
3. A list with references and tables
[((Table1 Paramx) Table2) (Table1 (Param1...Paramx)) (Table2 (Param1...Paramx))]
When a the Parameterx of Table 1 makes reference to the Primary Key of the Table2.

5 Student activity log

5.1 Arturo's Time-Sheet

Date	Task Description	Spent Time
04/02/2016	Create a method that gets user entries and put it into a list.	4 hours
04/02/2016	Create a method that creates lists(tables) into a list(DB)	2 hours
04/02/2016	Create a method that insert registers without parameter	6 hours
04/02/2016	Create method that manage the user entry by box entry	3 hours
04/03/2016	Fix the method add_Table, adding the restrictions	0.5 hours
04/03/2016	Create method that manage the insertions	0.5 hours
04/03/2016	Create method that found a referenced table of a given table	5 hours
04/03/2016	Create method that check if a register is in a table	2 hours
04/03/2016	Fix the method mngIns	1 hour
04/05/2016	Fix the method insert_data for uncompleted or more registers	2 hours
04/06/2016	Creating auxiliaries functions	6 hours
04/06/2016	Fix some bugs (Voids elements)	15 hours
04/07/2016	Making the external documentation	12 hours
	Total	59 hours

5.2 Malcolm's Time-Sheet

Date	Task Description	Spent Time
04/02/2016	Making the functions of add and remove references	13 hours
04/03/2016	make auxiliaries functions for references	13 hours
04/04/2016	Create the update method	14 hours
04/05/2016	Fix some bugs for the program	15 hours
04/06/2016	Create the method that show	15 hours
04/07/2016	Create the methods of queries	14 hours
	Total	84 hours

6 Project final status

Despite the nature of the project, and the amount of time this sued, because of the number of verifications, and the learning curve to a new programming language and a new paradigm, the project is accomplished almost entirely. Just leaving aside the storage processes.

6.1 Issues, Limitations or Challenges

- The large amount of verifications to be performed.
- The implementation of REGEX for manage and split the users entries.
- The learning curve of a new Programming Language and new Paradigm.
- Make the update and query functions were a challenge, because it has a great difficult level.

6.2 Known Issues

- Don't implemented the stored procedures.

7 Conclusions, Suggestions and Recommendations

- online tools can be helpful for remote work.
- Tools like the handler code versions is very useful, because in case of any failure in the new version allows us to return to a previous, plus the possibility of linking it to the storage service in the cloud, to share, and work together in a very easy way.
- Zube can be helpful for organize the tasks of the group.
- The amount of verifications tha a Data Base have are exaggerated.
- OverLeaf it's a very easy tool, for write papers and scientific papers.

References

- [1] "*Welcome to Racket*" Docs.racket-lang.org, 2016 [Online]. Available: <http://docs.racket-lang.org/guide/intro.html>. [Accessed: 08- Apr-2016].

- [2] "*Racket Documentation*" Docs.racket-lang.org, 2016 [Online]. Available: <https://docs.racket-lang.org/index.html>. [Accessed: 08- Apr- 2016].
- [3] "*RegErr: Learn, Build, & Test RegEx*" RegExr.com, 2016. [Online]. Available: <http://regexpr.com/>. [Accessed: 08- Apr- 2016].