

Discrete Mathematics

Practical Assignment 2



Table of Contents

1	Assignment	3
2	Resources	8
3	Delivery and grading	8
3.1	Delivery	8
3.2	Grading	9
4	Appendix: SQLite environment setup.....	Error! Bookmark not defined.
4.1	Installation	Error! Bookmark not defined.
4.2	Usage	Error! Bookmark not defined.
5	Appendix: Relational Data using R	Error! Bookmark not defined.
5.1	Introduction	Error! Bookmark not defined.
5.2	Relational data with R warm-up	Error! Bookmark not defined.
5.3	Running a R script	Error! Bookmark not defined.



1 Assignment

In this assignment, you will convert your expert system developed in Prolog during assignment 1, into a relational database. The desired outcome is a representation of your expert system queries using the mathematical notation of relations (see: Relational-Logic.pdf).

Also, you should build a database where rules, facts, and queries must be mapped to tables, tuples and queries. The database should be a translation of the knowledge base created in Prolog. The queries should have the same functionality as the rules of inference in Assignment 1.

You can implement your solution using a traditional SQL database to create tables, tuples and queries. Instructions on how to set up your environment can be found in the section 2 (SQLite environment setup).

Or you can go beyond and earn extra points. You can implement the solution using the R language, a programming language and software environment for statistical computing and data science. R has a relational data package that allows you to define tables, tuples, queries. More information is available in the section 3 (Relational data using R). Note that the choice is mutually exclusive: You should choose between SQLite or R, not both.

The information about the requirements and grading can be found in the section “Delivery and grading”.

2 SQLite environment setup

2.1 Installation

For the sake of simplicity and portability, we have chosen SQLite to be our relational database. SQLite is a public domain SQL database engine with implementations for several operating systems. You can download a precompiled binary for your OS using the link below (see the sections of precompiled binaries). The installation process is quite simple: you download and extract the zip file into your preferred folder.

Please download the precompiled binary for your operational system.

<https://sqlite.org/download.html>



2.2 Usage

The SQLite project provides a simple command-line utility named **sqlite3** (or **sqlite3.exe** on Windows) that allows the user to execute SQL statements and manage SQLite databases. Here you can see the basics to do the exercise like create the database, use the interactive console, exporting and importing data. For more information, please refer to the “Resources” section.

2.3 Executing the interactive console

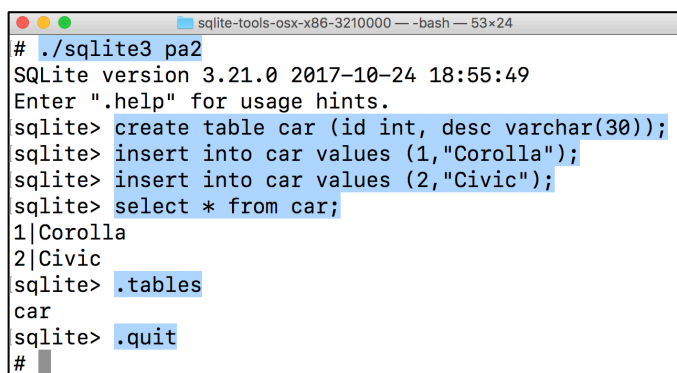
To use the interactive console, you can type **sqlite3 <db>** at the command prompt inside your SQLite installation folder. The parameter **<db>** is the database name. If the database does not exist, an empty one will be created. If the database name is omitted, SQLite will use an “in-memory” temporary database. Thus, if you want to reuse the database, we recommend you to specify a database name.

For example:

```
# sqlite3 pa2
```

2.4 Using the interactive console

You can use the interactive console to execute SQL statements and manage your database. The figure below shows an example of usage:



```
sqlite-tools-osx-x86-3210000 — -bash — 53x24
# ./sqlite3 pa2
SQLite version 3.21.0 2017-10-24 18:55:49
Enter ".help" for usage hints.
sqlite> create table car (id int, desc varchar(30));
sqlite> insert into car values (1,"Corolla");
sqlite> insert into car values (2,"Civic");
sqlite> select * from car;
1|Corolla
2|Civic
sqlite> .tables
car
sqlite> .quit
#
```

First, the command **sqlite3 pa2** was executed, where **pa2** is the database name. SQLite printed its current version and enabled the interactive mode (see the **sqlite>** prompt). Some SQL statements were executed: creating a table, inserting data into a



table, and selecting data from a table. Note that all SQL statements are followed by the semicolon (;) character.

The tool also has special commands starting with a (.) dot. For example, the command **.tables** list all tables of the database, and the **.quit** command leaves the console. Type **.help** to get a list of all special commands.

2.5 Running scripts from command-line

When working with databases, it is common storing a series of SQL commands (DDL statements, queries) into a script file. You can execute a script file from the SQLite command-line by using the **.read** command. For example:

```
# sqlite3 pa2 ".read script.sql"
```

The command above will open the **pa2** database and execute the statements of the **script.sql** file. The output will be printed on your console. Note that you can use output redirection to save the output to a file. For example:

```
# sqlite3 pa2 ".read script.sql" > output.txt
```

The example above will store the output produced by the execution of the statements into the file **output.txt**.

2.6 Exporting your database

This step is necessary before handing your assignment into VLO. To export your database, you should use the **sqlite3** command from your command-line interface.

Syntax: `sqlite3 <your-db-name> .dump > <exported-file>`

For example:

```
# sqlite3 pa2 .dump > pa2.db
```

This command will export the **pa2** database, generating a text file called **pa2.db** in the current folder. This file contains the schema creation statements and the insert statements to populate the database.



2.7 Importing a database

To grade your work, your teacher will import your file into his/her environment in the following manner. Make sure that your file is working correctly on a different machine before you hand it in.

To import data from a file, you should use **sqlite3** command with the **-init** parameter followed by your dump file. For example:

```
# sqlite3 newdb -init pa2.db
```

This command will create a database called **newdb** and populate the newly created database, by executing the statements of the file **pa2.db**.

3 Environment setup for using R

3.1 Introduction

You might opt to implement your project using the R language. This section provides information on how to download and learn R. It is not intended to be complete, and you should invest time on reading the documentation and make experiments with this development environment beforehand. As a reward, you will learn more about this language that is widely used among statisticians and data miners for developing statistical software and data analysis.

R is a free language and development environment. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. You should first install version 3.6 on your local machine.

<https://www.r-project.org/>

To get acquainted with R and its library to deal with relational data, you can use the following user guide, more specifically section 13.

<https://r4ds.had.co.nz/index.html>

3.2 Relational data with R (warm-up)

This section aims at making a straightforward experiment using relational data using R. Use your terminal to invoke the R interactive console tool.

```
# r
```

The interactive console of R will open:

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
```





Copyright (C) 2019 The R Foundation for Statistical Computing

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

>

You can quit the console at any time by typing `quit()`.

To describe tables, you should install the *tibble* package that is part of the core *tidyverse*.

> `library(tidyverse)`

3.2.1 Creating tables using Tribble

The code below describes two tables: one table called *country* with an id and a description, and another called *person* with an id, name and a foreign key to the *country* table, indicating the nationality of the person.

```
> country = tribble(~id,~name,"BRA","Brazil","NLD","Netherlands","FRA","France")
> person = tribble(~id,~name,~country,1,"John","NLD",2,"Marcio","BRA",3,"Anna","FRA")
```

A variable holds a reference to the table. You can print the content of each one by typing the name of the variable in the console, as follows:

```
> country
# A tibble: 3 x 2
  id  name
  <chr> <chr>
1 BRA  Brazil
2 NLD  Netherlands
3 FRA  France
```

3.2.2 An example of inner join

The statement below is an example of an inner join between *person* and *country*. Note the usage of the function `inner_join` and the operator `%>%` to relate two tables.

```
> result = person %>% inner_join(country, by = c("country" = "id"))
```

The variable *result* holds the outcome of the inner join.

```
> result
# A tibble: 3 x 4
  id name.x country name.y
  <dbl> <chr> <chr> <chr>
1 1 John NLD Netherlands
2 2 Marcio BRA Brazil
3 3 Anna FRA France
```



You can print the result just by typing the variable name or using the print function.

3.3 Running an R script

You can save the set of statements in a text file with the .r extension. To execute the file, use the *Rscript* executable, passing the script file as a parameter.

```
# Rscript query.r
```

The output of the program execution will be printed in your console.

4 Resources

Text Book: Discrete Mathematics with Applications, International Edition
Chapter 8.1 – Relations on Sets

Relational Logic
Available in the Course Portal

SQLite Website (download, guides, reference manual)
<https://sqlite.org/>

R for Data Science. *Garrett Golemund* and *Hadley Wickham*
Section 13 – Relational Data
<https://r4ds.had.co.nz/relational-data.html>

5 Delivery and grading

5.1 Delivery

You hand in your work in separate files:

- A manual in the PDF format, including information about your expert system (*manual.pdf*). This manual should explain each query, more specifically, formal description and the correlation with the rules of your expert system.
- For SQLite Implementations:
 - The exported database. The file should have the name: **pa2.db.txt**
 - The set of SQL queries should have the name: **pa2.sql.txt**. The file **pa2.sql.txt** that contains all the SQL queries you exported
- For R implementations:
 - A single program called **pa2.r.txt** that includes the definition of tables, tuples and queries. There should be a clear distinction between these three sections.



The source code will be used to test the validity of your implementation step by step. Thus, there should not be any syntax errors in any file. All files should be uploaded separately. The three files should be handed in by the same student. See: all the rules regarding practical assignments in the Study Manual (section “Grading practical assignments and rules”).

Zip files will not be accepted.

5.2 Grading

Passing grade (5.5) Minimal requirements	+ 1 Organization	+3,5 Extra
<p>This database is a translation of the knowledge base created in Prolog.</p> <p>A manual is present, and there is a description of what each query does.</p> <p>The manual has a table containing the logical correspondence between rules, facts and propositional logic queries and tables, tuples and relational data queries.</p> <p>At least seven different queries are written down as a mathematical relation using relational logic.</p> <p>For an SQL implementation, there is an exported database called pa2.db and a file pa2.sql.txt that contains at least 10 SQL queries OR</p> <p>For an R implementation, there is a file called pa2.r.txt that contains at least 10 queries.</p> <p>All files are semantically and syntactically correct.</p>	<p>The queries have useful comments and describe what they are doing.</p> <p>The queries are well organized in a text file.</p> <p>It is clear which query is related to which rule of inference in the PA1 program.</p> <p>It is clear how the database is built up and how the tables/entities of the relational database correspond to the knowledge base of PA1.</p>	<p>a) Use of extra types queries, like join, outer join or subqueries.</p> <p>b) These extra types of queries are written down in mathematical notation.</p> <p>c) There is a description of what these queries do in the manual</p> <p>Items a, b, and c are documented in the manual.</p> <p>When the script is executed, the output clearly states which query is being executed, followed by the result. The layout of the output is unambiguous and easy to read</p> <p>The R language was used to describe tables, tuples and build queries.</p> <p>The R script is well-organized. There is a clear distinction between the definition of tables, tuples and queries.</p>

