’Computing skills for Biologist’ – Chapter 10
2021-03-11 Ingimar Erlingsson (CC BY-SA 4.0)

- Book, ch 10: Computing Skills for a Biologist; by Stefano Allesina & Madlen Wilmes

- What is a relational database ?
- Why use a relational database ?
- Structure and Design
  1) ER EER diagram
- Getting started with SQLite
  1) Create a database - Creating tables
  2) CRUD-statements: Basic SQL
  3) Exporting db to a csv-file
  4) Backup & Restoring

Computing skills for Biologist – Chapter 10
2021-03-11 (1h) Ingimar Erlingsson

- Graphical User Interfaces for SQLiteGUI:s)
- Other popular RDBMS
- References and Reading
- Other Resources and then some Akronyms

What is a relational database ?

- A database is a structured collection of data.

1) A program , an **RDBMS**, to manage the data – SQLite is such a RDBMS
   - Data is stored in a specific binary format
2) Data are arranged in **Tables**;
   - A Table is composed of Records & Fields [rows & columns]
   - Each field contains a certain type of data (e.g ’TEXT’, ’INTEGER’,’FLOAT’ etc)
3) The ability to connect different tables through related fields
4) A language to query the data, **SQL**
   - **CRUD** operations; Create/Retrieve/Update or Delete the data

Why use a relational database (1)

- Large data sets, using relational db can greatly improve performance

  1) Each field **contains data of a specific type**, and the software stores its values in the most efficient way […] **read into memory without any conversion.**

  2) Redundancy - in a text file (such as csv) you often repeat yourself across the different rows

  3) In a Relational db – You store values in different tables and you make references between the tables (linking the tables)
     - Using a relational db minimizes (avoids) entering redundant information

Why use a relational database (1)

- Ability to to create ’Index’ [if you want to]
  1) ’indexing system’ speeds up data retrieval operations
  2) Efficient when it comes to ’large data sets’
  3) The index contains one or more columns in order , ’we can find the desired value much faster because our search is performed on an ordered column’
  4) Downside using Index :
     - An index takes additional storage space
     - The index needs to be updated or rebuilt every time the data are modified

Why use a relational database (3)

- Integrity and Security

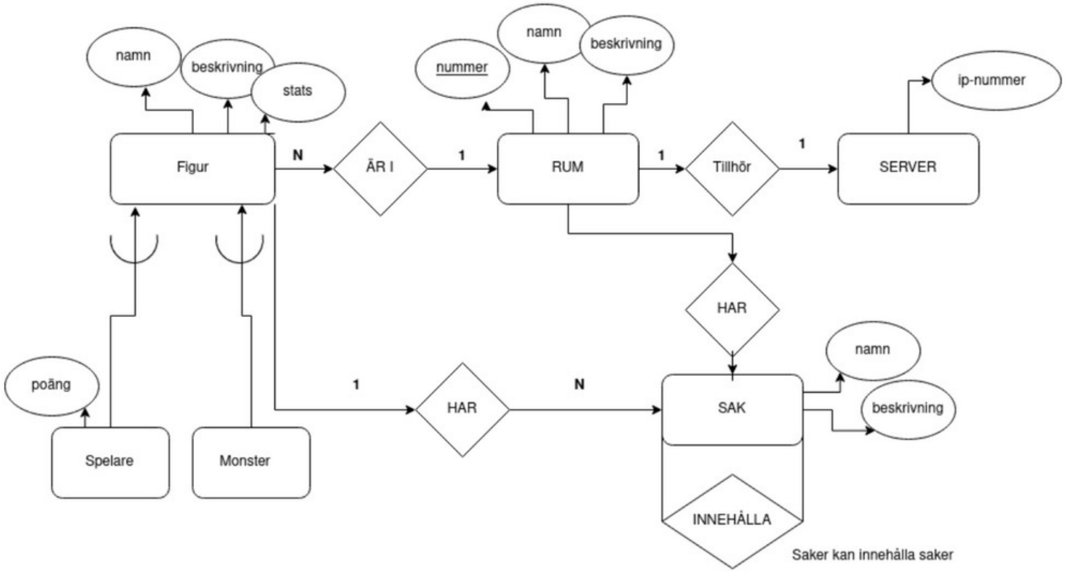
  1) Adding contstraints:
     - Reject: to add data whenever a field is missing (’NOT NULL’)
     - Reject: to add data that does not satisfy a given requirment
  2) Security when it comes to sensitive data (often a task in the Role of a ’DBA’ in larger org.)
     - You can have different privileges for different users
       - Some users can **only read** data but can not modify it
       - Some users can only **see partial records** etc

Structure & Design (1)

- When you start Designing your DB, start with a ’Conceptual Data Model’
  1) Use ER-modelling or EER-modelling
     - **ER** = Enitity Relationship , **EER** = Enhanced ER
- It is often easier to start ’to think in ER’ than ’to think in Tables’
  1) When creating the ER-diagram, you are not creating the Tables .
     - You start with the ER and then ’translate’ that diagram to Tables
     - draw.io is a good tool to use drawing ER-diag. (draw.io → https://app.diagrams.net/)
- Wiki-links to ER and EER
  1) ER: https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
  2) EER: https://en.wikipedia.org/wiki/Enhanced_entity%E2%80%93relationship_model

Structure &Design (2)

EER-diagram



EER- diagram by
Ingimar Erlingsson

Structure and Design (3)

- Designing
  1) Tables : Records and Fields
  2) Tables can be linked to other Tables, linking is done via ’fields’
  3) Primary Key (PK) and Foreign Key (PK) – best to use the data_type ’INTEGER’
  4) Pay attention to the Normal Forms (1NF,2NF,3NF,4NF and 5NF) -se page 353
- SQLite
  1) Serverless (most RDBMS are server-client systems)
  2) Zero-configuration database system, therefore ’easier to install’
  3) A database in a single file
- I am running SQLite (3.31.1) on Ubuntu 20.04.2 LTS
  1) sqlite3 --version
     - 3.31.1 2020-01-27

Getting started with SQLite (1)

- Creating a database from scratch (the ’create’-statement is a DDL-statement)

1) $ **sqlite3** homework_2.db
   - Where ’homework_2.db’ is the name of my database, stored in the FS
2) Create the table ’site’ and the table ’sampling’ – (obs, ’snake_case’ is best-practice)
   - Usage of PRIMARY KEY, data type = INTEGER (Foreign Key in samling ’site_id’ could be named ’site_id_FK’)
3) **sqlite>** CREATE TABLE **site**(

```
"site_id" INTEGER PRIMARY KEY,
"site_name" TEXT,
"x_coord" REAL,
"y_coord" REAL);
```

4) **sqlite>** CREATE TABLE **sampling**(

```
"sampling_id" INTEGER PRIMARY KEY,
"site_id" INTEGER,
"date" TEXT,
"temperature" TEXT,
"humidity" TEXT);
```

5) **check your design** with the command ’.schema’ or ’.schema <table>’.
   - **sqlite> .schema**
   - **sqlite> .schema site**

Getting started with SQLite (2)

- Inserting data (**C**RUD)
  1) **sqlite**> INSERT INTO site (site_id,site_name,x_coord,y_coord) VALUES (1,'stockholm',18.0685808,59.3293235);
  2) **sqlite**> INSERT INTO sampling (sampling_id,site_id,date,temperature,humidity) VALUES (1,1,'2021-03-10',-4,77);
- Retrieving data (C**R**UD)
  1) **sqlite**> SELECT * FROM site;
     - Result will be without headers → ’1|stockholm|18.0685808|59.3293235
  2) **sqlite**> .header on
  3) **sqlite**> SELECT * FROM site;
     - site_id|site_name|x_coord|y_coord
     - 1|stockholm|18.0685808|59.3293235
  4) **sqlite**> .mode column AND **sqlite**> .width 10 10 10 10
  5) **sqlite**> SELECT * FROM site;
     - site_id   site_name   x_coord   y_coord
     - ---------- ---------- ---------- ----------
     - 1   stockholm   18.0685808 59.3293235
  6) To fetch a subset (not valid here, only 1 record) using ’WHERE
     - **sqlite**> SELECT * FROM site WHERE site_name=’stockholm’;

Getting started with SQLite (3)

- Updating data (CR**U**D)
  1) **sqlite**> UPDATE site SET site_name='STOCKHOLM' WHERE site_id=1;
- Deleting data (CRU**D**)
  1) **sqlite**> DELETE FROM <mytable> WHERE <condition>;

- DML-statement 'DROP' (delete a Table or a View)
  1) **sqlite**> DROP TABLE site;
  2) **sqlite**> DROP VIEW <myview>;

Getting started with SQLite (4)

- Exit sqlite
  1) sqlite> .quit
- The database/the file is in your path
  1) homework_2.db (in my system it is 12 kilobyte)
- (1) Start sqlite and then open the file/db
  1) $ **sqlite3**
  2) sqlite> .open homework_2.db
  3) sqlite> .tables
     - Output → ’sampling’ and ’site’
- (2) Start sqlite and open the file/db
  1) $ **sqlite3 homework_2.db**
  - sqlite> .tables
     - Output → ’sampling’ and ’site’

Getting started with SQLite (4)

- Import a CSV file to SQLite (page 344)

  1) sqlite> **.mode** csv
  2) sqlite> **.import** Lohr2015_data.csv lohr
  3) sqlite> .save **lohr.db**
  4) sqlite> **.schema**
  5) ”Note that all the columns in the.csv file have been imported as **TEXT**, though we know that some of the values are numeric.”

- Then open the file using SQLite

  1) $ sqlite3 lohr.db

10.7 Exporting Tables and Views

- Exporting tables and Views to a csv-file (here ’,’-separated, se line 2)

1) sqlite> .mode list
2) sqlite> .separator ,
3) sqlite> .output homework_2_20210311.csv
4) sqlite> select * FROM site;
5) sqlite> **.output**
6) ”Note that all the columns in the.csv file have been imported as **TEXT**, though we know that some of the values are numeric.”

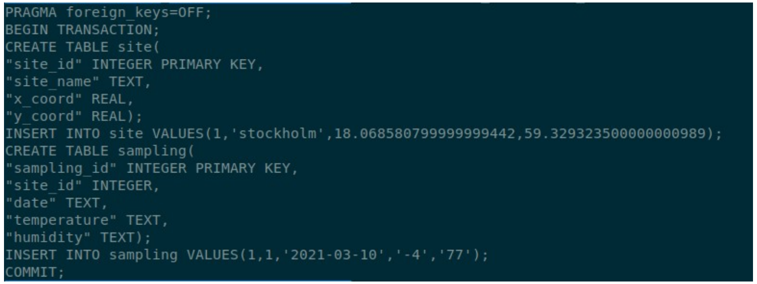
10.7 Backing up & Restoring

- Backup

1) .output [PATHTOFILE]/<name-of-sqlitedumpfile>.sql
2) sqlite> .output homework_sqlitedump_20210311.sql
3) sqlite> .dump
4) sqlite> .exit

- A textfile is created

1) With the db-structure (DML-commands) and the data (DDL-) are saved.
2) $ more homework_sqlitedump_20210311.sql



```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE site(
"site_id" INTEGER PRIMARY KEY,
"site_name" TEXT,
"x_coord" REAL,
"y_coord" REAL);
INSERT INTO site VALUES(1,'stockholm',18.068580799999999442,59.329323500000000989);
CREATE TABLE sampling(
"sampling_id" INTEGER PRIMARY KEY,
"site_id" INTEGER,
"date" TEXT,
"temperature" TEXT,
"humidity" TEXT);
INSERT INTO sampling VALUES(1,1,'2021-03-10','-4','77');
COMMIT;
```

2 GUI:s

- I have not tried out these ones for SQLite
  1) http://www.sqlitebrowser.org/
  2) http://www.sqlitebrowser.org/dl/
     - GNU/Linux (Arch Linux, Fedor, openSUSE,Debian, Ubuntu, FreeBSD)
     - Windows, macOS
  3) SQLiteStudio → sqlitestudio.pl (https://sqlitestudio.pl/)
     - https://github.com/pawelsalawa/sqlitestudio/releases
     - GNU/Linux (download tar.xz-file)
     - Windows, macOS

RDBMS

Other Popular Relational Database Management Systems

- PostgreSQL (https://www.postgresql.org/)
  1) You can install ’PostGIS’ → A ’Geospatial Database extender’ to PostgreSQL
- MySQL (https://www.mysql.com/)
  1) Now the property of Oracle Corporation (https://www.oracle.com/se/mysql/)
  2) 2 products : The Enterprise Edition and the **Community Edition** (CE)
     - https://www.mysql.com/products/community/
- MariaDB (https://mariadb.org/)
  1) MariaDB Foundation
  2) ”It’s made by the original developers of **MySQL** and guaranteed to stay open source.” (/about/)
- Mimer (https://www.mimer.com/)
  1) Swedish company, ’Mimer evolves from a research project at **Uppsala University**’ (/about-us/)

Resources

- http://computingskillsforbiologists.com/
- http://computingskillsforbiologists.com/downloads/exercises/#sql
- git clone https://github.com/CSB-book/CSB.git
- https://sqlite.org : sqlite.org/books.html → https://www.tutorialspoint.com/sqlite/index.htm
- Överkurs: versioning your database, https://www.liquibase.org/
- Överkurs: Data Cleaning and Transforming : https://openrefine.org/

- 3 Books
  1) **Relational Database Design** by Jan L Harrington (ISBN-13: 978-0128043998 , ISBN-10: 012804399)
  2) **Databasteknik** av Thomas Padron-McCarthy & Tore Risch (ISBN 9789144069197)
     - Thomas P → http://www.databasteknik.se/nekotronic/cv.html (teacher at oru.se)
  3) **SQL-introduktion** av Mikael Segerlund & Folke Stridman (pris 100kr)
- Wiki-Links to RDBMS

1)	SQLite:	https://en.wikipedia.org/wiki/SQLite	(stable release 3.34.1	[20 jan 2021])
2)	MySQL:	https://en.wikipedia.org/wiki/MySQL	(stable release 8.0.23	[18 jan 2021])
3)	MariaDB:	https://en.wikipedia.org/wiki/MariaDB	(stable release 10.5.9	[22 feb 2021])
4)	PostgreSQL:	https://en.wikipedia.org/wiki/PostgreSQL	(stable release 13.2	[11 feb 2021)]
5)	Mimer:	https://en.wikipedia.org/wiki/Mimer_SQL	(stable release 11.0.5A	[1 mar 2021])

Akronyms

- **SQL**
  1) Structued Query Language: For ’data query’, ’data manipulation’ and ’data access control’
  2) Different standards – linked to ANSI (American …) and ISO (International …) - https://en.wikipedia.org/wiki/ISO/IEC_9075
  3) ”SQLite understands most of the standard SQL language. But it does omit some features while at the same time adding a few features of its own. ” : source https://sqlite.org/lang.html (2021-03-11)
- **CRUD**: 4 basic operations of persistent storage
  1) Create, Retrieve [or Read], Update, Delete
- **DDL and DML**
  1) **DDL**: ’Data Definition Language’ or ’Data Description Language’
     - Such as ’Create’- , ’Drop’-, ’Alter’, ’Truncate’-statements
  2) **DML:** ’Data Manipulation Language’
     - ’SELECT … FROM … WHERE …’ / ’INSERT INTO … VALUES …’ / ’UPDATE … SET … WHERE …’ / ’DELETE FROM … WHERE …’
- ER, EER, ERD → Entity-Relationship, Extended ER, ER-Diagram