

Theory & Practice of Data Cleaning

Installing the SQLite System

SQLite Download Page

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

Documentation

[sqlite-doc-3170000.zip](#) Documentation as a bundle of static HTML files.
(5.28 MiB) (sha1: bf3b479067d6eb36c8230b668a45994c9212d6cc)

Precompiled Binaries for Android

[sqlite-android-3170000.aar](#) A precompiled Android library containing the core SQLite together with appropriate Java bindings, ready to drop into any Android Studio project.
(4.65 MiB) (sha1: 27f1a8391dc9586dc71dfe2b100551e1ff33e754)

Precompiled Binaries for Linux

[sqlite-tools-linux-x86-3170000.zip](#) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqldiff](#) program, and the [sqlite3_analyzer](#) program.
(1.77 MiB) (sha1: ba887504b36833ad380cdb57bb5684bd36335faa)

Precompiled Binaries for Mac OS X (x86)

[sqlite-tools-osx-x86-3170000.zip](#) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqldiff](#) program, and the [sqlite3_analyzer](#) program.
(1.11 MiB) (sha1: 6eed355bf4b59c4a4443a4f037adf16b1e6e96ba)

Precompiled Binaries for Windows

[sqlite-dll-win32-x86-3170000.zip](#) 32-bit DLL (x86) for SQLite version 3.17.0.
(431.40 KiB) (sha1: a97cebc176b3daa453189f2c0b7cf2a5a70f9c92)

[sqlite-dll-win64-x64-3170000.zip](#) 64-bit DLL (x64) for SQLite version 3.17.0.
(715.39 KiB) (sha1: deba09d3c18bf4cdf9e0c3af7f7e7147d9f4fab9)

[sqlite-tools-win32-x86-3170000.zip](#) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqldiff.exe](#) program, and the [sqlite3_analyzer.exe](#) program.
(1.54 MiB) (sha1: 66cff70dc901eb6015a5c425cd7ec527f02628a0)

Universal Windows Platform

Theory & Practice of Data Cleaning

SQL(ite) by Example

swcarpentry/sql-novice-survey x Bertram

← → ↻ GitHub, Inc. [US] https://github.com/swcarpentry/sql-novice-survey

requirements.txt Updating Python requirements file 9 months ago

setup.md Fixing link in setup instructions 9 months ago


README.md

sql-novice-survey

An introduction to databases and SQL using Antarctic survey data. Please see <https://swcarpentry.github.io/sql-novice-survey/> for a rendered version of this material, [the lesson template documentation](#) for instructions on formatting, building, and submitting material, or run `make` in this directory for a list of helpful commands.

Maintainer(s):

- [Abigail Cabunoc Mayes](#)
- [Sheldon McKay](#)

© 2017 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)  [Contact GitHub](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)

Databases and SQL

Securehttps://swcarpentry.github.io/sql-novice-survey/

HomeCode of ConductSetupReferenceEpisodes ▾Extras ▾License

Search...

Databases and SQL

In the late 1920s and early 1930s, William Dyer, Frank Pabodie, and Valentina Roerich led expeditions to the [Pole of Inaccessibility](#) in the South Pacific, and then onward to Antarctica. Two years ago, their expeditions were found in a storage locker at Miskatonic University. We have scanned and OCR the data they contain, and we now want to store that information in a way that will make search and analysis easy.

Three common options for storage are text files, spreadsheets, and databases. Text files are easiest to create, and work well with version control, but then we would have to build search and analysis tools ourselves. Spreadsheets are good for doing simple analyses, but they don't handle large or complex data sets well. Databases, however, include powerful tools for search and analysis, and can handle large, complex data sets. These lessons will show how to use a database to explore the expeditions' data.

🌟 Prerequisites

- Unix shell plus SQLite3 or Firefox SQLite plugin.
- [survey.db](#)

Schedule

00:00	Selecting Data	How can I get data from a database?
00:15	Sorting and Removing Duplicates	How can I sort a query's results? How can I remove duplicate values from a query's results?
00:35	Filtering	How can I select subsets of data?
00:55	Calculating New Values	How can I calculate new values on the fly?
01:05	Missing Data	How do databases represent missing information? What special handling does missing information require?
01:35	Aggregation	How can I calculate sums, averages, and other summary values?
01:55	Combining Data	How can I combine data from multiple tables?
02:35	Data Hygiene	How should I format data in a database, and why?
03:05	Creating and Modifying Data	How can I create, modify, and delete tables and data?

SQL: Structured Query Language

- Based on **relational algebra** and tuple **relational calculus**.
- Consists of a data definition language, data manipulation language, and data control language.
- Scope of SQL includes data **insert, query, update** and **delete**, schema creation and modification, and data access control.
- SQL is a **declarative** language (4GL), but also includes procedural elements.

SQL: Structured Query Language

- A typical SQL query has the form

select A_1, A_2, \dots, A_n
from r_1, r_2, \dots, r_k
where P

- A_i s represent attributes
 - r_i s represent relations
 - P is a predicate
- This query is equivalent to the relational algebra expression
$$\pi_{A_1, A_2, \dots, A_n}(\sigma_P(r_1 \times r_2 \times \dots \times r_k))$$
- The result of an SQL query is a relation (set of tuples) with a schema defined through the attributes A_i s.
- The **select** clause corresponds to the projection operation of the relational algebra; it is used to list the attributes to be output in a query result.

Theory & Practice of Data Cleaning

SQLite by Example (cont'd)

Queries and Simple Data Cleaning in SQLite

Software Carpentry: Databases & SQL

1. Selecting Data
2. Sorting and Removing Duplicates
3. Filtering
4. Calculating New Values
5. Missing Data
6. Aggregation
7. Combining Data
8. Data Hygiene
9. Creating and Modifying Data
10. Programming with Databases

The screenshot shows a SQLite shell window and a text editor window. The SQLite shell window displays the following commands and output:

```
[SQLite :)]$ sqlite3 survey.db
SQLite version 3.8.10.2 2015-05-20 18:17:19
Enter ".help" for usage hints.
sqlite> .schema
.schema
CREATE TABLE Person (id text, personal text, family text);
CREATE TABLE Site (name text, lat real, long real);
CREATE TABLE Visited (id text, site text, dated text);
CREATE TABLE Survey (taken integer, person text, quant text, reading real);
sqlite> select * from person;
select * from person;
dyer|William|Dyer
pb|Frank|Pabodie
lake|Anderson|Lake
roe|Valentina|Roerich
danforth|Frank|Danforth
sqlite> .mode column
.mode column
sqlite> .header on
.header on
sqlite> select * from person;
select * from person;
id      personal    family
-----
dyer     William    Dyer
pb       Frank      Pabodie
lake     Anderson   Lake
roe      Valentina  Roerich
danforth Frank      Danforth
```

The text editor window shows the following SQL code:

```
CREATE TABLE Survey(
  taken integer,
  person text,
  quant text,
  reading real
);
INSERT INTO "Survey" VALUES(619,'dyer','rad',9.82);
INSERT INTO "Survey" VALUES(619,'dyer','sal',0.13);
INSERT INTO "Survey" VALUES(622,'dyer','rad',7.8);
INSERT INTO "Survey" VALUES(622,'dyer','sal',0.09);
INSERT INTO "Survey" VALUES(734,'pb','rad',8.41);
INSERT INTO "Survey" VALUES(734,'lake','sal',0.05);
INSERT INTO "Survey" VALUES(734,'pb','temp',-21.5);
INSERT INTO "Survey" VALUES(735,'pb','rad',7.22);
INSERT INTO "Survey" VALUES(735,NULL,'sal',0.06);
INSERT INTO "Survey" VALUES(735,NULL,'temp',-26.0);
INSERT INTO "Survey" VALUES(751,'pb','rad',4.35);
INSERT INTO "Survey" VALUES(751,'pb','temp',-18.5);
INSERT INTO "Survey" VALUES(751,'lake','sal',0.1);
INSERT INTO "Survey" VALUES(752,'lake','rad',7.19);
```

Annotations in the image include:

- A red arrow pointing to the `sqlite3 survey.db` command with the text "loading survey 'database'" in a red box.
- A red arrow pointing to the `CREATE TABLE Survey` command with the text "This is a different file for 'creating' the database in the first place .." in a red box.

Theory & Practice of Data Cleaning

Recursion in SQL

Recursion in SQL

- Traditionally, SQL, First-order Predicate Logic, Relational Algebra, don't allow *recursion* (or iteration) => Datalog to the rescue!
- Since SQL:1999 standard recursion is allowed using CTEs (*common table expressions*):

```
WITH RECURSIVE (...)  
SELECT ...
```

The screenshot shows a SQLite shell window with the following content:

```
-- family.sql  Bot (14,0)  (SQL[ansi])  
[SQLite-notes :>] $ sqlite3  
SQLite version 3.8.11.1 2015-07-29 20:00:57  
Enter ".help" for usage hints.  
Connected to a transient in-memory database.  
Use ".open FILENAME" to reopen on a persistent database.  
sqlite> .read family.sql  
174.0  
Alice  
...Bob  
.....Dave  
.....Emma  
...Cindy  
.....Fred  
.....Gail  
  
sqlite>
```

Two blue boxes highlight recursive queries:

- Box 1:

```
WITH RECURSIVE  
works_for_alice(n) AS (  
VALUES('Alice')  
UNION  
SELECT name FROM org, works_for_alice  
WHERE org.boss=works_for_alice.n  
)  
SELECT avg(height) FROM org  
WHERE org.name IN works_for_alice;
```

 A blue arrow points from this box to the value 174.0 in the output.
- Box 2:

```
WITH RECURSIVE  
under_alice(name,level) AS (  
VALUES('Alice',0)  
UNION ALL  
SELECT org.name, under_alice.level+1  
FROM org JOIN under_alice ON org.boss=under_alice.name  
ORDER BY 2 DESC  
)  
SELECT substr('.....',1,level*3) || name FROM under_alice;
```

 A red arrow points from this box to the hierarchical output of names.

The bottom of the window shows the prompt:

```
--:**-- *shell*  Bot (18,7)  (Shell:run)
```

Benoit [in 1] Mandelbrot



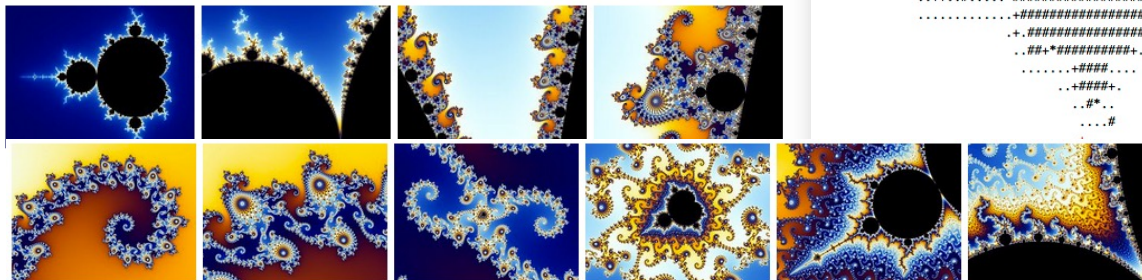
At a TED conference in 2010.

Born	20 November 1924 Warsaw, Poland
Died	14 October 2010 (aged 85) Cambridge, Massachusetts, United States
Residence	Poland · France · United States
Nationality	Polish · French · American
Fields	Mathematics · Aerodynamics

Recursion in SQL

- ASCII-art version of famous Mandelbrot fractal in SQL with recursion (right)
- Zooming into the Mandelbrot set (below)

Source: [Wikipedia]



```
*shell*
New Open Recent Print Copy Search Preferences Help
family.sql mandelbrot.sql SQLite-notes<2> sudoku.sql sudoku.dlv test.dlv
WITH RECURSIVE
xaxis(x) AS (VALUES(-2.0) UNION ALL SELECT x+0.05 FROM xaxis WHERE x<1.2),
yaxis(y) AS (VALUES(-1.0) UNION ALL SELECT y+0.1 FROM yaxis WHERE y<1.0),
m(iter, cx, cy, x, y) AS (
  SELECT 0, x, y, 0.0, 0.0 FROM xaxis, yaxis
  UNION ALL
  SELECT iter+1, cx, cy, x*x-y*y + cx, 2.0*x*y + cy FROM m
  WHERE (x*x + y*y) < 4.0 AND iter<28
),
m2(iter, cx, cy) AS (
  SELECT max(iter), cx, cy FROM m GROUP BY cx, cy
),
a(t) AS (
  SELECT group_concat( substr(' .+##', 1+min(iter/7,4), 1), '' )
  FROM m2 GROUP BY cy
)
SELECT group_concat(rtrim(t),x'0a') FROM a;

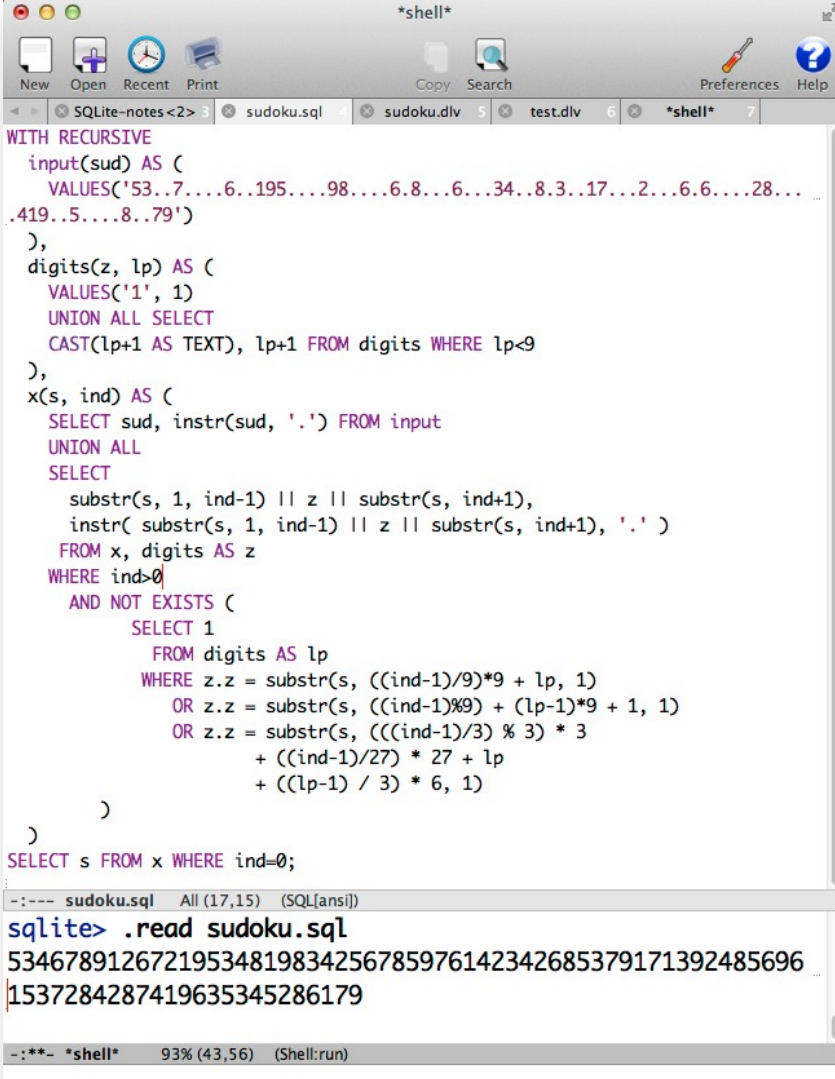
U:--- mandelbrot.sql All (18,0) (SQL[ansi])

sqlite> .read mandelbrot.sql

...#
..#*..
..+###+.
.....+###+. +
..##+*#####+.++++
+.#####+.
.....+#####+.+
..+.#.....*#####+.
...+#####+.#####
...+*#####+.
#####
.....+#####+.
...+*#####+.
..+.#.....*#####+.
.....+#####+.+
.....+#####+.+
..+#####+.
..##+*#####+.++++
.....+###+. +
..#*..
...#
```

Crazy, Recursive SQL (cont'd)

- Sudoku in SQL!
 - Basic idea:
 - For each blank cell, find a digit 1..9 that doesn't violate the complex IC (another "denial constraint"):
- WHERE ...
- AND NOT EXISTS ...



```
WITH RECURSIVE
  input(sud) AS (
    VALUES('53..7....6..195....98....6.8...6...34..8.3..17...2...6.6....28...
.419..5....8..79')
  ),
  digits(z, lp) AS (
    VALUES('1', 1)
    UNION ALL SELECT
      CAST(lp+1 AS TEXT), lp+1 FROM digits WHERE lp<9
  ),
  x(s, ind) AS (
    SELECT sud, instr(sud, '.') FROM input
    UNION ALL
    SELECT
      substr(s, 1, ind-1) || z || substr(s, ind+1),
      instr( substr(s, 1, ind-1) || z || substr(s, ind+1), '.' )
    FROM x, digits AS z
    WHERE ind>0
      AND NOT EXISTS (
        SELECT 1
        FROM digits AS lp
        WHERE z.z = substr(s, ((ind-1)/9)*9 + lp, 1)
          OR z.z = substr(s, ((ind-1)%9) + (lp-1)*9 + 1, 1)
          OR z.z = substr(s, (((ind-1)/3) % 3) * 3
            + ((ind-1)/27) * 27 + lp
            + ((lp-1) / 3) * 6, 1)
      )
  )
SELECT s FROM x WHERE ind=0;
```

-- sudoku.sql All (17,15) (SQL[ansi])

```
sqlite> .read sudoku.sql
53467891267219534819834256785976142342685379171392485696
1537284287419635345286179
```

-- *shell* 93% (43,56) (Shell:run)