Retrieving data

SQL SELECT

What we will look at in this lecture

- How is data organised in a database?
- What is a table?
- What are types?
- SELECT statement (SQL)
- WHERE clause - specifying criteria
- ORDER BY
- * (wildcard for “all columns”)
- Boolean expressions with AND, OR, comparisons
- LIKE

What we will not learn

How to solve Sudokos using SQL and SELECT

```
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; /* works in sqlite version > 8.2 */
```

Organisation of data

In order to understand how to fetch data, we need to know how data is organised in a database.

Data are organised in a database in something called tables. In our example database there is currently one table called books.

Data in two or more tables can be related. For instance, a table can hold information that is further described in a second table.

We will look at that when we talk about normalisation.

For now, we have only one single table.

Organisation of data inside a table

A table has rows of data. A row contains data in fields (fields are also known as columns). Fields have types and can have modifiers (or constraints).

Our simple book table is arranged as such:

```
Author text, Title text, ISBN text primary key, Publisher text
```

It means that each row has the following fields and types (and modifiers)

```
fieldname    type     modifier
Author       text
Title        text
ISBN         text     primary key
Publisher    text
```

Field types

You can read about the type system of SQLite3 here: https://www.sqlite.org/datatype3.html

Basically we can think of these different types:

INTEGER

REAL

TEXT

BLOB

As a reference/comparison the PostgreSQL types are here: http://www.postgresql.org/docs/9.0/static/datatype.html

Other types that would be handy

We’d also like to have the concepts of Boolean, Date and Time.

SQLite provides functions for the dates and times but uses integers for the actual data. Boolean is used by SQLite using the integers 1 for true and 0 for false.

Modifiers/Constraints

A field can have either a value of a type or the special value NULL.

If we don’t want to permit NULL values we can use the modifier NOT NULL which will cause a constraint violation if someone tries to insert the value NULL for such a field.

PRIMARY KEY is a constraint for a unique column. All primary key values must be different from each other.

We’ll get back to constraints in a later lecture.

SELECT

In order to retrieve data using SQL, we use the SELECT statement. It has the following structure in its simplest form:

```
SELECT column[,column]* FROM table [WHERE criteria];
```

For instance, returning to our books example, we could do:

```
sqlite> SELECT title FROM books;
Life
Love
Guns
Code
Cars
```

SELECT more than one field/column

We can retrieve more than one field:

```
sqlite> SELECT author,title FROM books;
John Smith|Life
James Woody|Love
Joan Carmen|Guns
Johnanna Boyd|Code
Eva Peron|Cars
```

The WHERE clause

Often we want to specify a criteria for the data we want to retrieve. What if we only want the books that have “Bonnier” for publisher?

```
sqlite> SELECT author,title FROM books WHERE Publisher='Bonnier';
John Smith|Life
James Woody|Love
Joan Carmen|Guns
Johnanna Boyd|Code
```

Headers

Warning: SQLite specific!

We can include the column names of our result by setting the headers flag:

```
sqlite> .headers on
sqlite> SELECT author,title FROM books WHERE Publisher='Bonnier';
Author|Title
John Smith|Life
James Woody|Love
Joan Carmen|Guns
Johnanna Boyd|Code
sqlite>
```

Format output

Warning: SQLite specific!

```
sqlite> .mode column
sqlite> SELECT author,title FROM books WHERE Publisher='Bonnier';
Author      Title
----------  ----------
John Smith  Life
James Wood  Love
Joan Carme  Guns
Johnanna B  Code
sqlite>
```

More formatting

Warning! SQLite specific!

```
sqlite> .width 14 14
sqlite> SELECT author,title FROM books WHERE Publisher='Bonnier';
Author          Title
--------------  --------------
John Smith      Life
James Woody     Love
Joan Carmen     Guns
Johnanna Boyd   Code
sqlite>
```

Ordering of the result

We can add an ordering clause at the end of our query like so:

```
sqlite> SELECT author,title FROM books WHERE Publisher='Bonnier'
   ...> ORDER BY Title;
Author          Title
--------------  --------------
Johnanna Boyd   Code
Joan Carmen     Guns
John Smith      Life
James Woody     Love
sqlite>
```

Reverse order

```
sqlite> SELECT author,title FROM books WHERE Publisher='Bonnier'
   ...> ORDER BY Title DESC;
Author          Title
--------------  --------------
James Woody     Love
John Smith      Life
Joan Carmen     Guns
Johnanna Boyd   Code
sqlite>
```

SELECT all fields

Sometimes we want to fetch all columns from a table. We can use * in order to say “all columns”.

```
sqlite> SELECT * FROM books;
Author          Title           ISBN          Publisher
--------------  --------------  ------------  ----------
John Smith      Life            0-0-0-0-0-1   Bonnier
James Woody     Love            0-0-0-0-0-2   Bonnier
Joan Carmen     Guns            0-0-0-0-0-3   Bonnier
Johnanna Boyd   Code            0-0-0-0-0-4   Bonnier
Eva Peron       Cars            0-0-0-0-0-5   Books R us
sqlite>
```

Boolean expressions

We can create criterias using AND, OR combined with <, >, =

```
sqlite> SELECT * FROM books WHERE Publisher='Books R us' OR Title='Life';
Author          Title           ISBN         Publisher
--------------  --------------  -----------  ----------
John Smith      Life            0-0-0-0-0-1  Bonnier
Eva Peron       Cars            0-0-0-0-0-5  Books R us


sqlite> SELECT * FROM books WHERE Publisher='Bonnier' AND
ISBN  > '0-0-0-0-0-2';
Author          Title           ISBN         Publisher
--------------  --------------  -----------  ----------
Joan Carmen     Guns            0-0-0-0-0-3  Bonnier
Johnanna Boyd   Code            0-0-0-0-0-4  Bonnier
```

Like it or not

We can select text according to a substring expression using LIKE:

```
sqlite> SELECT * FROM books WHERE Title LIKE "c%";
Author          Title           ISBN         Publisher
--------------  --------------  -----------  ----------
Johnanna Boyd   Code            0-0-0-0-0-4  Bonnier
Eva Peron       Cars            0-0-0-0-0-5  Books R us
sqlite>
sqlite> SELECT * FROM books WHERE Author LIKE "jo%";
Author          Title           ISBN         Publisher
--------------  --------------  -----------  ----------
John Smith      Life            0-0-0-0-0-1  Bonnier
Joan Carmen     Guns            0-0-0-0-0-3  Bonnier
Johnanna Boyd   Code            0-0-0-0-0-4  Bonnier
```

A few words about LIKE

In SQLite the like expression is case insensitive. In PostgreSQL it is not (PostgreSQL has the ILIKE operator for case insensitive matching).

There is a performance hit when using LIKE. There are much faster matching mechanisms in various databases.

Summary

A database consists of tables. Tables have rows. Each row has columns with name and type. Data is inserted into a table row by row.

To retrieve data from a table we use SELECT. The basic form is:

```
SELECT <column>[,<column>]* FROM <table> [WHERE <condition>];
```

For instance:

```
SELECT Author,Title FROM books WHERE Title = 'Cars';
```

The condition can be complex, e.g. `Title='Cars' AND Publisher='Bonnier';`

What's up next?

Next lecture, we'll look at the SQLite3 database and how to get started with it.

If you want to prepare, install SQLite3 on your computer.

Ubuntu:

```
sudo apt-get install sqlite3
```

MacOS:

Follow the instructions here (make sure to install at least version 3):

http://www.tutorialspoint.com/sqlite/sqlite_installation.htm

Read

http://www.w3schools.com/sql/sql_select.asp

http://zetcode.com/db/sqlite/select/

https://en.wikibooks.org/wiki/Structured_Query_Language/Snippets#Basic_Syntax (The basic syntax is enough for this course, and note that you have to refer to the SQLite3 manual to see what constructs are supported!)

https://www.sqlite.org/lang_select.html (in particular SELECT_CORE)