Kernmodule 3 Datavisualisatie 3

Vorige keer...

- Voor visualisaties van data heb je *gestructureerde data* nodig
- *gestructureerde data* sla je op in een database, bv. een *relationele database*
- Om data uit een database te halen, gebruik je **SQL:**

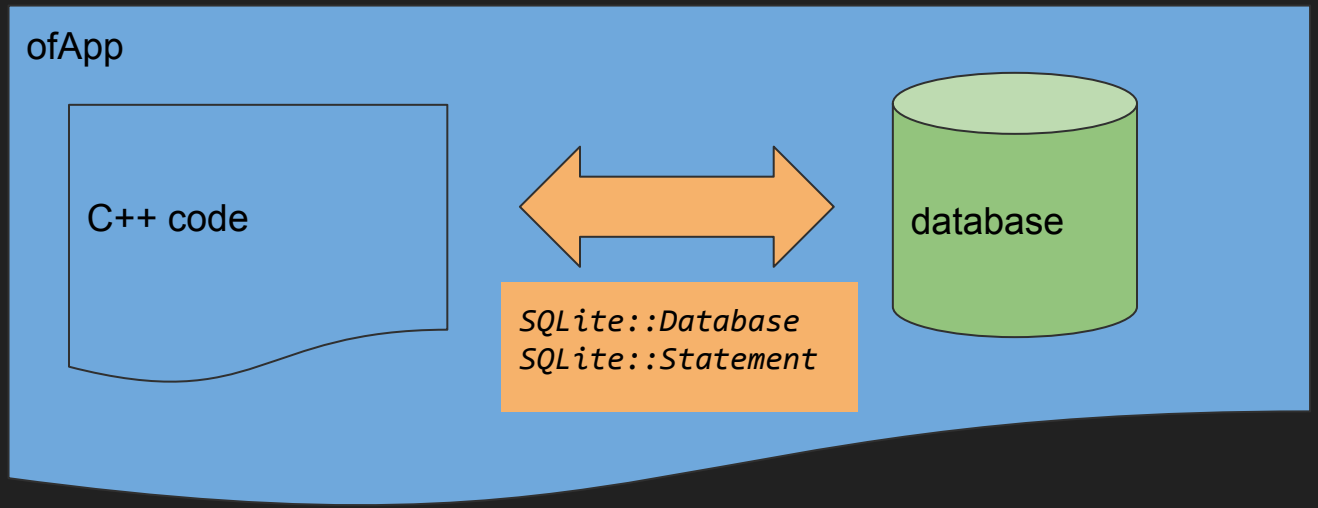
  Structured Query Language

- Binnen deze lessen maken we gebruikt van SQLite, via een addon: `ofSQLiteCPP`

- Deze addon levert een serie C++ classes om tegen de database te praten (in SQL…)

SQLite in openFrameworks

De SQLite extensie voor openFrameworks wordt *ingebouwd* in je ofApp.
De twee belangrijkste classes uit de *driver* zijn `SQLite::Database`, en `SQLite::Statement`



SQLite in openFrameworks

De `ofxSQLiteCPP` is een *wrapper* rond de SQLite functionaliteit.

Verschillende SQL / Database concepten zijn ingepakt als C++ classes

De database zelf: `SQLite::Database`
gebruik je om de database te openen, en te sluiten

Een *query* (bv een SELECT): `SQLite::Statement`
gebruik je om data op te halen

SQLite gebruiken betekent in oF dus gebruik maken van deze objecten, en de methodes die daar in zitten.

SQLite in de Rosling-app

Data opvragen: het `SELECT` statement

Als je (in SQLite Browser) de Roslingdb database opent, kun je queries uitproberen:

```
SELECT year,nl,au FROM population WHERE year>=1945
```

year	nl	au
1945	9	12
1968	11	24
2005	17	32

Dit geeft een *resultset* terug: een soort tijdelijke tabel, met nul, één of meerdere rijen. De kolommen in de *resultset* corresponderen met de kolommen uit het `SELECT` statement

SQLite::Statement

De SQLite::Statement *class* gebruik je om tegen de SQLite database te praten

```
SQLite::Statement query(*db, "SELECT * FROM population WHERE year<=1945");
```

Dit levert dus ook een *recordset* op - *meerdere* rijen.

Om daar data uit te halen, wandel je stap-voor-stap door de *resultset:*

```
while (query.executeStep()) {
…
}
```

Data uit de resultset halen

De kolommen in je resultset kun je op index (getal), of op naam (string) uit de *query* ophalen:

```
query.getColumn("nl");
```

Wat je hier terugkrijgt is een `SQLite::Column` class. De *data* die er in die kolom zat kun je er weer uithalen met speciale methodes:

```
query.getColumn("au").getDouble(); // .getString(), .getInt()
```

CSV imports

Databases aanmaken

Een database aanmaken gaat het snelst door data te *importeren*.

Een handig import-formaat is een **CSV** file (**C**omma **S**eparated **V**alue): een eenvoudig tekst-formaat met een "row" per regel, en komma's tussen de kolommen:

```
"Month", "1958", "1959", "1960"
"JAN",  340,  360,  417
"FEB",  318,  342,  391
"MAR",  362,  406,  419
"APR",  348,  396,  461
"MAY",  363,  420,  472
```

Opdracht volgende keer

Maak een nieuwe ofApp die gebruik maakt van `ofxSQLiteCPP`

Maak met SQLiteBrowser een nieuwe database aan (in de bin/data directory van de ofApp die je net gemaakt hebt), en importeer het CSV bestand `huishoudens.csv` uit de klas-repository.

Deze database bevat nu een tabel met de samenstelling van Nederlandse huishoudens van 1995 tot 2016: het totaal aantal huishoudens, het aantal één- en meerpersoonshuishoudens, met en zonder kinderen, getrouwd en niet getrouwd.

Ontwerp (schets) meerdere visualisaties op basis van deze data. Daarnaast bouw je met deze data een eenvoudige interactieve visualisatie op basis van de *RoslingDB* code uit de les.