MY472 - Week 9
Relational Databases and SQL

Friedrich Geiecke
Outline

- **Relational** vs non-relational databases
- The SQ Language
- Coding session
Relational vs non-relational databases
Databases

- **Database system**: An organized collection of data that is stored and accessed via a computer.

- **Relational databases**: Data stored in multiple tables to avoid redundancy. Tables are linked based on common keys.

- **Non-relational databases**: Data stored in a way that is not based on tabular relations (e.g. MongoDB uses JSON like documents).
Relational vs non-relational databases

From: Codewave Insights
Relational databases

- Relational database management systems (RDBMS): MySQL, PostgreSQL, SQLite, MariaDB, etc.
- Database as a Service (DBaaS): Amazon RDS, Google Cloud SQL, Microsoft Azure SQL Database
- DBaaS at a scale: Amazon RedShift, Google BigQuery, Microsoft Azure
Some vocabulary

<table>
<thead>
<tr>
<th>Relational database term</th>
<th>SQL term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation</td>
<td>Table</td>
</tr>
<tr>
<td>Tuple, record</td>
<td>Row</td>
</tr>
<tr>
<td>Attribute, field</td>
<td>Column</td>
</tr>
</tbody>
</table>


Keys

- Primary key: A column or set of columns (composite key) which uniquely identifies each row/record in the table
- Foreign key: A primary key of another table
Structured Query Language
SQL: Structured Query Language

- **Language** designed to define, control access to, manipulate, and query relational databases
- Initially written SEQUEL (Structured English Query Language), but later changed to SQL because of trademark issues
- Pronounced both S-Q-L and SEQUEL today
- It is a **nonprocedural/declarative language**: User defines what to do, inputs, and outputs, but not the control flow; how the statement is executed, is left to the **optimizer**
- How long SQL queries depends on optimization that is opaque to user
- Performance will vary, but generally faster than standard data frame manipulation in R (and much more scalable)
Some components of common SQL queries

- The result of a SQL query is a table
- `SELECT` columns
- `FROM` a table in a database
- `WHERE` rows meet a condition
- `GROUP BY` values of a column
- `ORDER BY` values of a column when displaying results
- `LIMIT` to only X number of rows in resulting table
- Always required: `SELECT` and `FROM`; rest are optional
- `SELECT` can be combined with operators such as `SUM`, `COUNT`, `AVG`...
- To merge multiple tables, use `JOIN`
SQL query example

SELECT name, account_id FROM client;

SELECT * FROM client WHERE gender = 'F';
SQL JOINS

From: https://upload.wikimedia.org/wikipedia/commons/9/9d/SQL_Joins.svg
**SQL JOIN example**

```
SELECT client.name, account.balance
FROM client JOIN account
ON client.account_id = account.id;
```
Coding session
Coding session

- See 01-sql-intro.Rmd
- See 02-sql-join-and-aggregation.Rmd

General information on how to connect to SQL databases with R: https://db.rstudio.com/getting-started/