



# Introduction to Databases

...and how to do stuff with them



Copyright © Newcastle University 2012

This work is licensed under the Creative Commons Attribution License

See <http://software-carpentry.org/license.html> for more information.



## *Preparation Check*

Install sqlite3

Run sqlite3 from command line (check you get a  
sqlite> prompt, Control-D to exit)

Download our example database:

*[http://homepages.cs.ncl.ac.uk/  
stephen.mcgough/swc/experiments.db](http://homepages.cs.ncl.ac.uk/stephen.mcgough/swc/experiments.db)*

**<http://tinyurl.com/8wp7s7a>**

## *What is a Database?*

A database is a way to store and manipulate information that is arranged as tables

Each table has

- Columns (fields) which describe the data
- Rows (records) which contain the data

Databases often confused with DBMS' (DataBase Management System)

- Manage and provide access to data in a database

## *Speaking Database: SQL*

In a spreadsheet, you insert formulas or new sheets to analyse your data

In a database, you give commands, also known as Queries

The database does the analysis your query specifies, and returns the results in a tabular form

Queries are written in a simple, english-like, language called SQL ("Structured Query Language")

## *Dataflow: The building blocks of queries*

SQL is a vast language that provides all sorts of ways of mixing and remixing your data

Each operation extracts or transforms data from a previous operation, to form a pipeline or flow of data that ends with the results you get back

We'll be giving you practical experience with a core set of SQL – probably around 80% of all queries you'll ever do

## *SQLite*

Production-level DBMS' like Oracle and Postgres are complicated

The concepts we'll show you are the same and can be applied to these databases

- Preventing implementation detail getting in the way

SQLite allows us to contain a separate database within a single file


- Great for sending databases to others; not so easy with other DBMS' !
- SQLite slower for huge databases than Oracle, etc.



...

Databases

Introduction to Databases



## *Exercise 1*

**Find out those people that have the last name Pavlov or Sakharov, with the results in ascending order of their login id**

Databases

Introduction to Databases

## *Exercise 2*

**Devise a query to:**

**Get all experiments where number of those involved > 1**

**Order results in ascending order by experiment id**

## *Exercise 3*

**Devise a query to:**

**give a total number of hours for experiments that have more than 4 hours registered...**

**...but don't have an experiment id of 1**

## Why Project.ProjectID = Involved.ProjectID ?

To distinguish between Fields in different tables with the same name we prepend the Field with the name of the Table

Project.ProjectID

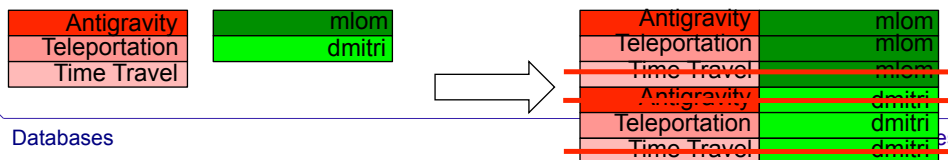
Involved.ProjectID

But why Place.Dept = Staff.Dept?

When we join the two tables together each record from Project is matched with each record from Involved

So we have every project matched with every member involved

But we only want the ones that represent real 'matches'



Databases

## Exercise 4

```
select distinct Project.ProjectName, Involved.Login
  from Project join Involved where
    Project.ProjectID=Involved.ProjectID;
```

**Extend the above query to also retrieve the last name of the person**

**Hint: there's a table we've already looked at that contains this information**

**Bigger hint: you'll need to do another 'join', increase the 'select' filtering and use an additional 'where'**

Databases

Introduction to Databases

## *Exercise 5 – the biggie*

**Devise query to return one table that shows:**

**Project Name | Person Last Name | Experiment Id | Hours  
Worked**