Databases - Lesson 1

An Introduction to Databases

So… What Exactly is a Database?

- A database is a place where you put stuff.
  - That's really all there is to it - you can all leave now.
- More precisely, a database is an organised collection of data. The database is implemented to support processes on the data contained within. These processes produce information from the data.

Data + Processing = Information

- If you are a climate scientist, you may want to model rainfall in Sydney. You will gather data, and process it to produce information.
  - Data are raw facts about something that have no meaning on their own. For example, the rainfall on a given day, or geographic coordinates.
  - Information is data that has been processed and can convey something useful in a given context. For example, climate trends are information for climatologists, the weather on a given day is data.

This Seems… Unnecessary

- So that's it? There's not much to it eh?
- Why don't we just store the data we want to keep in a list, or Excel, or better, in Outlook!
- By this definition, isn't a text file or a spreadsheet already a database?
  - Technically, yes. They are called "shitty databases". ;)

- This is where we get to the why.

Why we Care About Databases

- It’s hard finding and keeping track of everything.
  - Good luck traversing 100 sheets in your Excel doc, each with 1,000 columns and 50 million rows.
- It’s hard to prevent errors.
  - Now, try using the above Excel doc without creating or finding any errors.
- It’s hard to keep data up to date.
  - Have fun changing the data in the above Excel doc!

Why we Care About Databases

- A database will handle all of this for you.
  - All you need to worry about is what data needs to be put in your database, and what you want to receive when you get data out.
- Databases know that computers die, and so makes it very easy to make lots of copies.
- Databases will allow for thousands of people to access the data contained at the same time. This is actually quite remarkable.

Database Pick ‘n’ Mix

- Databases come in many types.
  - Hierarchical
    - This is old school and won’t be covered here.
  - Relational
    - This is the currently the most common type of database in both enterprise and at home.
  - NoSQL
    - These are databases that don’t use SQL. We won’t be covering these here. But we will talk about what SQL is a bit later.

You are All Already Database Users

- Pretty much any website you visit is essentially a database with an ad filled face.
- Your music player stores information about your music in a database.
- Your phone's contacts are in a database.
- Your GPS stores maps in a database, not as a big-ass JPEG file.
- Google ***is*** a database. And I don't just mean the search engine.

What are Databases Like Internally?

- Normally, databases are just specially structured files on a computer.
- Big databases need to be very efficient and the structure matters.
  - Many complex algorithms are used to manage data.
    - Z-curve, query optimisation, shadow paging, write-ahead logging…
  - The data is stored internally in complex data structures.
    - B-trees, B+-trees, ISAMs, heaps...

DataBase Management Systems

- DataBase Management Systems (DBMSs) wrap around these database files and data structures and provide an interface for computer software to work with the database.
- Functionality to Insert, update, delete and query the data are provided by the DBMS.
- The DBMS keeps the data organised.

DataBase Management Systems

- There are many DBMSs in the world.
  - PostgresQL (this will be our primary focus)
    - Very flexible free/open source DBMS.
  - MySQL/MariaDB
    - Free/open source DBMS.
  - SQLite
    - Many small computer programs will use SQLite if they need a simple database embedded inside.
  - Oracle
    - Some enterprises pay millions for this every year. Not necessarily any better than the others.

The Data Model

- In a relational database, the data is organised into tables and rows. This is called the data model.
  - A table represents a type of object, or type of ‘thing’ - trees, people, cars etc.
  - Each table consists of any number of rows.
    - Each row is an example of that type of object.
    - If we had a table for ‘trees’, each row would be an individual tree.
  - There are relationships between tables and rows.

The Data Model

- Each table contains any number of columns. These columns store facts about the objects being modelled.
  - If we are modelling trees, each one would have a height, a weight, a colour a name, an age and many other facts.
  - Each of these columns has a data type (number, string, binary data, date etc).
  - Each column can have a number of properties - we'll get into this later.

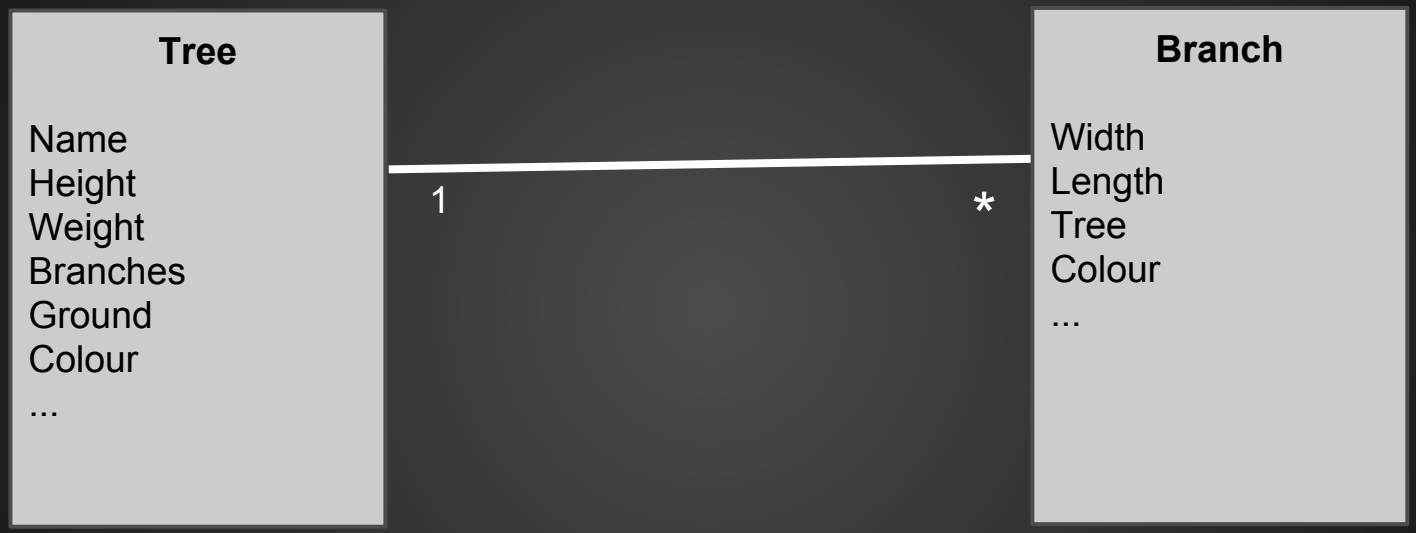
Types of Relationships

- There are three types of relationships between database tables. This is called ‘multiplicity’.
  - One-to-one relationship
    - For example: Each person has one and only one brain. Each car has one and only one engine.
  - One-to-many relationship
    - For example: Each tree has many branches. A branch cannot belong to more than one tree.

Types of Relationships

  - Many-to-many relationship
    - For example: A doctor sees any number of patients, and each patient sees any number of doctors.
- Relationships can be diagrammed. Each table becomes a square and relationships are lines linking squares. The multiplicity is represented at both ends of the line as a number or an asterisk (meaning ‘many’).

Relationship Diagrams



- In this example, each tree (1) has any number of branches (*). Both tables have several data columns.

Relational Databases

- Everything in life is related to everything else.
  - The ground has a tree growing out of it. The tree turns carbon dioxide into oxygen. You breath in oxygen, breathing out carbon dioxide and vapor. Vapor becomes a cloud and it rains. The ground gets wet and the tree grows. Someone chops down the tree and makes old fashioned wooden statuettes of fish. You go Antiquing for wooden statuettes of fish. You die during an unfortunate smelting accident and get reincarnated as a tree… The cycle repeats.

Relational Databases

- If the previous (wholly serious) example was modelled as a database, we would need:
  - Some way of inserting trees, people.... wooden fish statuettes… into our environment.
  - Some way of updating the data when changes occur in the environment.
  - Some way of deleting things from the environment.
  - Some way of inspecting the environment, to find out what's going on.
- This is where we talk about language.

Structured Query Language

- Structured Query Language (SQL) is a programming language that allows you to communicate with the relational DBMS and manipulate data in your database.
- Very similar to English, but very strict.
- SQL is a recognised standard, but unfortunately, DBMS vendors modify the SQL used in their database. :(

Structured Query Language

- Most basically, when you write SQL you write ‘statements’ or more appropriately ‘queries’. Each of these is on a separate line.
- Each query consists of a number of ‘clauses’ and ‘operators’.

- Lets look at an example query.

An Example Query

```
SELECT name, age
FROM trees
WHERE height > 10
    AND age >= 4
ORDER BY name, age;
```

- The above is called a ‘SELECT’ query. It’s purpose is to retrieve information.
- The ‘SELECT’, ‘FROM’, ‘WHERE’, and ‘ORDER BY’ are ‘clauses’. The ‘WHERE’ clause has several ‘operators’ like ‘>’, ‘>=’ and ‘AND’.

The ‘SELECT’ Clause

```
SELECT name, age
```

- The ‘SELECT’ clause allows you to declare which columns in a table you would like to retrieve. There is more complex behaviour you can do, but this is a good start.
- In our example, we are selecting two columns of data.

The ‘FROM’ Clause

```
FROM trees
```

- The ‘FROM’ clause declares which tables data will be retrieved from. Clearly, the columns we mentioned in the ‘SELECT’ clause must exist in these tables or you will die!
- In our example, we are retriving information from our ‘trees’ table.

The ‘WHERE’ Clause

```
WHERE height > 10
    AND age >= 4
```

- The ‘WHERE’ clause is like a search query. It allows us to tell the database that only some data is to be retrieved and what conditions must be met for this to happen.
- The ‘WHERE’ clause involves an expression. The expression contains operators.

The ‘WHERE’ Clause

- The expression in the ‘WHERE’ clause is matched performed against each row in the database tables being queried.
- SQL queries can return any number of rows. Some special types of queries must return only one row. We might look at these later.

Operators

- Many operators are used in SQL ‘WHERE’ clauses, here’s a few to get you going.
  - x = y
    - Is x equal to y?
  - x > y
    - Is x greater than y?
  - x < y
    - Is x less than y?
  - x >= y
    - Is x equal to or greater than y?

Operators

- x <= y
  - Is x less than or equal to y?
- x != y
  - The opposite of x = y. Is x not equal to y?

- Operators can be combined into complex expressions with ‘AND’ and ‘OR’:
  - AND
    - The expression is true if both sides are true.
  - OR
    - The expression is true if at least one side is true.

The ‘ORDER BY’ Clause

```
ORDER BY name, age
```

- Sometimes we want to sort the information retrieved to make it more useful. In SQL, sorting can be performed over the columns returned by a query, with any number of columns used in the sort.

Some Notes

- Usually, SQL queries are terminated by a semicolon (;).
- Many tables can be joined together in a query, so you can pool together different types of data using their relationship. We'll discuss this later.
- The more complex the query, the slower the query will be.

That’s Really Not All

- There are many other types of queries that SQL provides you. ‘SELECT’ statements are not all you can do. We need to learn how to manipulate data inside the database.

- But… that’s for another day.