CM3010: Databases and Advanced Data Techniques Summary

Arjun Muralidharan

March 5, 2022

Contents

List of Figures

List of Listings

1 Working with Data

Learning Outcomes

- ✓ Find, describe and evaluate data sources
- ✓ Understand the forms data comes input
- ✓ Evaluate data-related access and reuse rights

1.1 Data Sources

Data can be:

- **New data**. Can be added as you go, or requiring an initial bulk entry. Having a small amount of data when you start can make testing easier and performance isn't a problem.
- **Pre-existing data**. Might need extraction, conversion and cleaning before it can be used in a new database.
  - **Internal "legacy" data**. Data in existing, in-house sources or systems that you need to re-use.
  - **External data**. Useful if you want to eliminate costs for data entry and quality checks, or need delegated expertise from a third party. However, you have no control over data quality and data structure, and data may be incomplete or ambiguous, and you need to trust the third party.

1.1.1 Ordering some data: What's on the menu?

- Trevor Munoz, 'What IS on the menu'
- Trevor Munoz, 'Refining the problem'
- NYPL Menus

NYPL's menus data scores 3-4 stars on TimBLs 5 Start Linked Open Data Scale. It uses JSON instead of the W3C standards RDF and SPARQL. **Data curation** involves moving data from it's original site of creation to a stable, accessible environment. Additional value can be created by creating secondary and tertiary resources such as indexes of data.

An important concept is "Strings vs. Things", as computers can only see strings, while humans identify things (Captain Cook is the same as James Cook, but computers can't know that at first glance). NYPLs data is largely composed of strings, not things, as identical dishes are counted separately due to naming differences.

The way NYPL asks for data in its interface conflates "transcribing" and "naming", making it easier for users to enter names but at the cost of semantic precision.

Cleaning might be made easier with tools such as OpenRefine which can help normalize strings (removing spaces and such) and grouping. It also provides a Python client library.

1.1.2 What does your data look like?

When we use data, we need to consider the use of the data, for example what the data represents, what attributes we're interested in, and other metadata. For example, if we are making a database of books, we might want to store the contents and title of the book, but also the look, colour and weight depending on what we are using the data for.

1.2 Linking and using data

The Pratt institute founded the Linked Jazz project, which creates a *Semantic Web* of Jazz musician data. It uses **linked open data (LOD)**, a semantic technology, that gives visibility to information that is often hidden behind institutions. Semantic webs which are linked as Linked Data are created using standard formats such as RDF, GRDDL, POWDER, SPARQL and so on.

1.2.1 Licensing, sharing and ethics

Access to data is often placed under legal restrictions. But there are cases where open data is published, for example to drive sales, for the common good, for contractual requirements or interoperability.

Reasons to *not* publish data are restrictions on the source data (e.g. medical records), control of use (e.g. artistic data), value of the data (e.g. if the data is the product),

Researchers increasingly plan the release of their data in journals or repositories along with the published work. This release needs to be governed by some terms. Depending on the region, data will be treated differently, e.g. in the US, creativity is emphasised, so raw data can usually be easily published, in Australia, originality is more important. In the EU, a database has copyright if there was some intellectual judgement needed in assembling it (see also the EU database directive).

Three conditions commonly found in licences are *attribution*, *copyleft*, and *non-commerciality*.

- An *attribution* requirement means that the licensor must be given due credit for the work when it is distributed, displayed, performed, or used to derive a new work.
- A *copyleft* requirement means that any new works derived from the licensed one must be released under the same license, and only that licence.
- The intent of a *non-commercial* licence is to prevent the licensee from exploiting the work commercially. Such licences are often used as part of a dual-licensing regime (see 'Multiple licensing', below), where the alternative licence allows commercial uses but requires payment to the licensor.

Problems with attribution are *attribution stacking*, i.e. the need to attribute all preceding works. Problems with copyleft are that the derived work cannot be licensed under its own copyleft licence.

Important licenses include the creative commons licenses, including CC0 for public domain works, Open Data Commons, Open/Non-Commercial Government Licence or GNU/GPL. Sometimes, multiple licensing can be used so the recipient can decide under what license they will use the data. See How to license research data.

1.3 Data Structure

There are different sorts of structure:

1. Structures in **programming environments**
2. Structures in **data models**
3. Structures in **serialization** (data formats)
4. Structures in **exchange protocols**
5. Structures in **User Interfaces**

The general **shapes available** are:

1. Tables
2. Trees
3. Graphs
4. Media (raw data)
5. Documents / objects

1.3.1 Tables

In tables consisting of rows and columns, each row represents a thing, and each column describes a type of information that we ascribe to a thing.

A table as normally understood is a rectangular array with the following properties:

1. it is column-homogeneous - in other words, in any selected column the items are all of the same kind, whereas items in different columns need not be of the same kind;
2. each item is a simple number or a character string (thus, for example, if we look at the item in any specified row and any specified column, we do not find a set of numbers or a repeating group).
3. all rows of a table must be distinct (duplicate rows are not allowed);
4. the ordering of rows within a table is immaterial;
5. the columns of a table are assigned distinct names and the ordering of columns within a table is immaterial.

This list is important in determining if a table is in an acceptable form for a database use; many spreadsheets and open data sources do not comply with this form.

A table with n columns represents a n-ary relation between the rows and the columns (each row is related to n columns).

1.4 Trees

Trees can represent data in hierarchical form and suited for *heterogenous data.* It is an arrangement of nodes, with each node in the tree having zero or more child nodes, and exactly one parent node (with exception of the root). The data in child nodes is usually of a distinct subtype of the parent node. HTML is a tree structure. Trees have a more rigid order of elements but each node can store a different number of child elements, unlike a table.

1.4.1 Other

A **graph** is a more general form of tree, where each node can be connected to other nodes. The web in general is organised as a graph.

More complex forms involve **blobs**, used for inaccessible data for storage, such as raw media. **Features** are searchable information derived from blobs. **Documents** are rich, but not interrelated data structures, for example the musical notation for a piece of music.

2 Relational Databases

A **relation** in the context of databases roughly corresponds to a table. It represents the definition of the table along with the values stored in it. Rules:

Everything is a relation

- All operations use the relational models
- All data is represented and accessed as relations
- Table and database structure is accesses and represented as tables

The system is unaffected by its implementation

- If the hardware changes
- If the operating system changes
- If the disks are replaced
- If data is distributed

The *relational model* is not the same as the *entitiy-relationship model.* An ER model helps model concepts, often as part of relational database design.

2.1 Drawing Databases

Entities A thing we want to model that can be uniquely identified. An entity is drawn using a *box* with a word in it.

Attributes Information that descrives an aspect of an entity. An attribute is drawn using an *ellipse*, connected by a line to an entity.

Relationship A connection or dependency between two *entitites.* A relationship is drawn as a *rhombus* connecting two entities using lines.

2.2 Joins

1. **Cross Join:** $A \cup B$
2. **Inner Join:** $A \cap B$
3. **Outer Join:** $A \bigoplus B$
4. **Left Join:** $A \cap \overline{B}$
5. **Right Join:** $\overline{A} \cap B$

2.3 Normal Forms

2.4 ACID

Some principles guard us against errors, often calles **ACID** principles.

Atomicity A group of operations should either *all* be performed, or *none at all.* They only make sense as a group. For example, a bank transfer only makes sense if all steps (checking balances, deducting from one account, adding to the other) are performed together. If a transaction fails, we **roll back** to the last consistent state.

Consistency The database is never in an inconsistent state as a result of groups of operations being processed. This means that any intermediate states during the execution of a group of operations should not be available for any other group of operations. For example, when two bank transfers occur, the second transfer cannot operate on any intermediate states that happen while the first transfer is executing. It needs to use the state either before or after the first transaction.

Isolation Sets of operations can only be executed one at a time.

Durability A completed set of operations will have it's result physically committed.

2.5 Transactions and Serialization

In SQL, ACID principles can be enforced with the `TRANSACT` command. This groups commands as a single, atomic transaction.

Listing 1 Transactions in SQL

```sql
START TRANSACTION;

SELECT ...
INSERT ...
UPDATE ...

COMMIT;

DELETE ...

ROLLBACK;

-- Rolls back to after the last commit
```

The data definition language can cause problems. If a certain language processes transactions mostly in memory for speed gains, then checkpoints may not be as frequent as transactions. Table locking is not absolute.

Serialisability is if interleaved transactions produce the same result as if they were processed sequentially.

Concurrent transactions need a **locking strategy**, so either a transaction locks the database from write access while another is reading, or the reading is done on an old state of the database. Either strategy is valid.

2.6 User rights

A well-designed user policy will guard a database from malicious or erroneous manipulation.

Listing 2 User Rights in SQL

```sql

CREATE USER Arjun
        IDENTIFIED BY '298UFHASD'
        PASSWORD EXPIRE;

CREATE ROLE Astronomer;

GRANT Astronomer TO Arjun;

GRANT SELECT, INSERT
        ON Planets, Moons
        TO Astronomer
WITH GRANT OPTION; -- allows the user to pass their privileges to other users

REVOKE ALL
        ON Planets, Moons
FROM Astronomer
```