COMP1531

6.1 - Data

Overview

- Data storage
- Data transfer
- Data modelling
- Lab05 chat

Storing Data: Persistence

Persistence: When program state outlives the process that created it. This is achieved by storing the state as data in computer data storage

What is storage?

- CPU cache?
- RAM?
- Hard disk? (we usually mean this one)

Storing Data: Persistence

Most modern backend/server applications are just source code + data

Storing Data: In practice

A very common and popular method of storing data in python is to "pickle" the file.

- Pickling a file is a lot like creating a .zip file for a variable.
- This "variable" often consists of many nested data structures (a lot like your iteration 2 data)

Storing Data: In practice

Let's look at an example

pickle_it.py

unpickle_it.py

Standard Interfaces

In any field in engineering, we often have systems, components, and designs built by different parties for different purposes.

How do all of these systems connect together? Through standard interfaces

Standard Interfaces

Data Interchange Formats

When it comes to **transferring data**, we also need common interface(s) that people all send or store data in universal ways to be shared between applications or shared over networks.

Three main interchange formats we will talk about:

- JSON
- YAML
- XML

Data Interchange Formats

When it comes to **transferring data**, we also need common interface(s) that people all send or store data in universal ways to be shared between applications or shared over networks.

Three main interchange formats we will talk about:

- JSON
- YAML
- XML

JSON

JavaScript Object Notation (JSON) - TFC 7159

A format made up of braces for dictionaries, square brackets for lists, where all non-numeric items must be wrapped in quotations. Very similar to python data structures.

JSON

Let's represent a structure that contains a list of locations, where each location has a suburb and postcode:

```
{
        "locations": [
        {
            "suburb" : "Kensington",
            "postcode" : 2033
        },
        {
            "suburb" : "Mascot",
            "postcode" : 2020
        },
        {
            "suburb" : "Sydney CBD",
            "postcode" : 2000
        }
    ]
}
```

Note:

- No trailing commas allowed
- Whitespace is ignored

JSON - Writing & Reading

Python has powerful built in libraries to write and read json.

This involves converting JSON between a python-readable data structure, and a text-based dump of JSON

json_it.py

unjson_it.py

YAML

YAML Ain't Markup Language (YAML) is a popular modern interchange format due it's ease of editing and concise nature

```yaml
---
locations:
- suburb: Kensington
  postcode: 2033
- suburb: Mascot
  postcode: 2020
- suburb: Sydney CBD
  postcode: 2000
```

Same example from previous slide

Note:

- Like python, indentation matters
- A dash is used to begin a list item
- very common for configuration(s)

XML

eXtensible Markup Language (XML) is more of a legacy interchange format being used less and less

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <locations>
      <element>
         <postcode>2033</postcode>
         <suburb>Kensington</suburb>
      </element>
      <element>
         <postcode>2020</postcode>
         <suburb>Mascot</suburb>
      </element>
      <element>
         <postcode>2000</postcode>
         <suburb>Sydney CBD</suburb>
      </element>
   </locations>
</root>
```

XML

Issues with XML include:

- More verbose (harder to read at a glance)
- More demanding to process/interpret
- More bytes required to store (due to open/closing tags)

While you will find very few modern applications choose to use XML as an interchange format, many legacy systems will still use XML as a means of storing data

Data

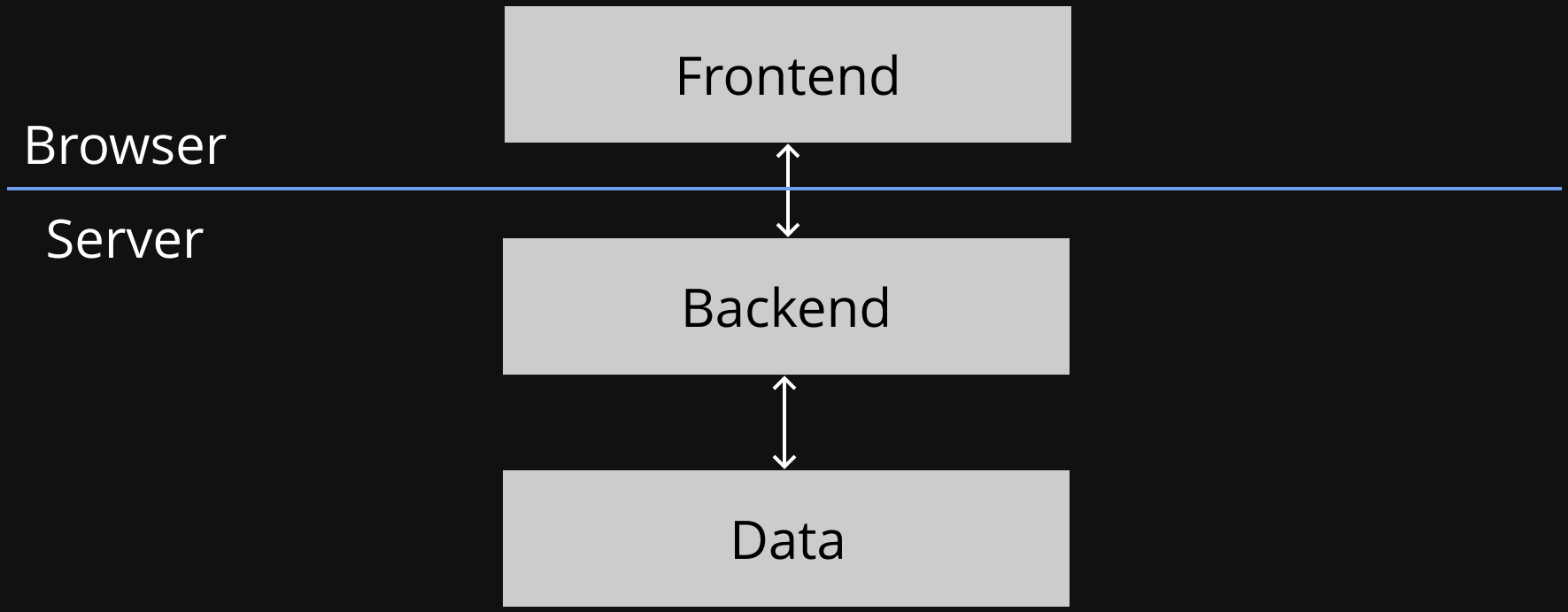
Data: Facts that can be recorded and have implicit meaning

Data is one of the fastest and most rapidly growing areas within software.

From **data (raw)** was can find **insights (information)** that allow us to make **decisions**.

Data Layer

Data Layer: The part of the tech stack that provides persistence



Databases

Data is only as powerful as you can store and access it. Study COMP3311 to learn more about efficient data storage

Data Modelling

When working in teams and organisations, data is only as useful as you can explain and represent it.

This is what we call data modelling

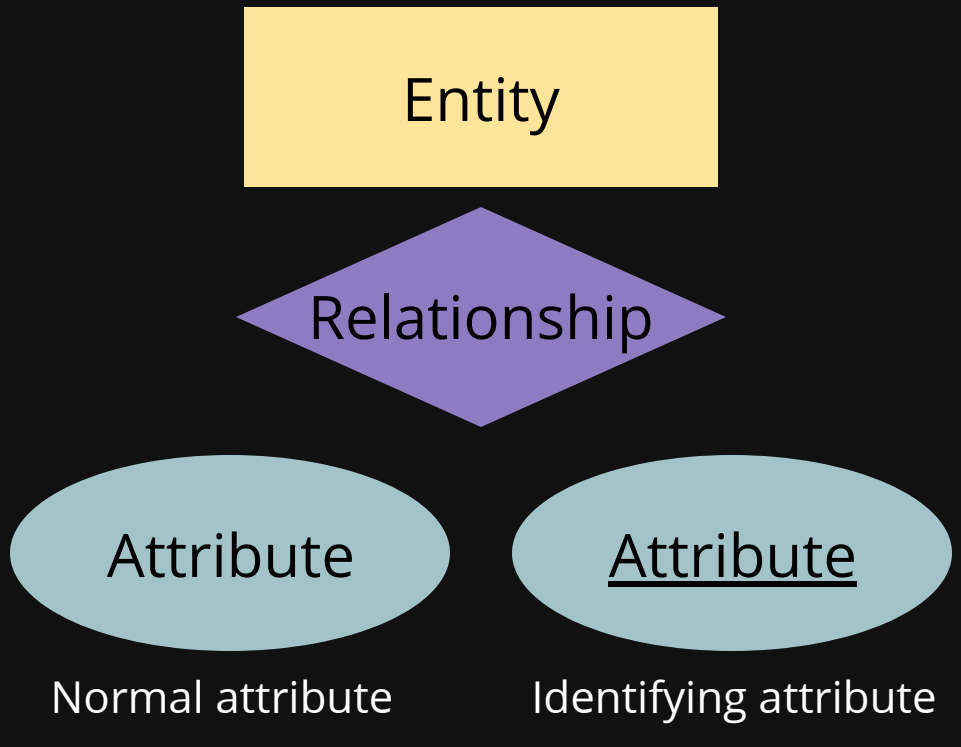
In this course we will model data with **ER diagrams**

ER Diagrams

ER Diagrams: Graphical tool for modelling data

Many different conventions for ER diagrams exist.

Key features made up of:



ERD Example

A hospital wants to build a database system to manage the information of patients. Here is some information:

A patient is identified by his/her patient ID. For each patient, we want to record his/her name, phone number, date of birth and age. A room is identified by its room ID and we also record its type and capacity. A patient must be assigned to exactly one room and a room can be assigned to multiple patients. Some rooms may be empty. A staff is identified by his/her staff ID, and we also need to know his/her name, salary, gender and role. A staff can work in zero or more rooms. And there should be at least one staff that works in a room.

Draw an ER diagram to represent this

ERD Multiplicity

Multiplicity is information about a relationship:

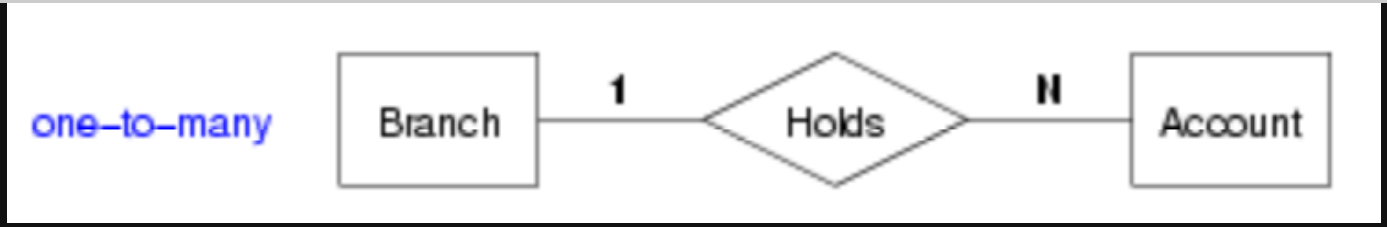
- **Cardinality**
  - Denoted with "1" or "N"/"M"
- **Participation**
  - A requirement to participate
  - Denoted with a bold line

Multiplicity = Cardinality + Participation

ERD Cardinality

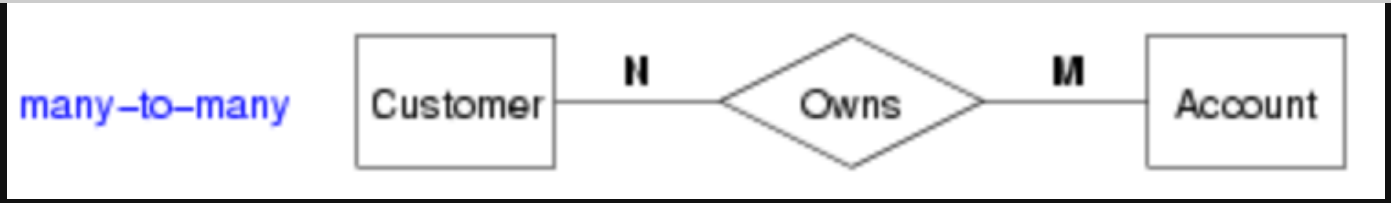
A manager is associated with **one** branch

A branch is associated with **one** manager



An account is associated with **1** branch

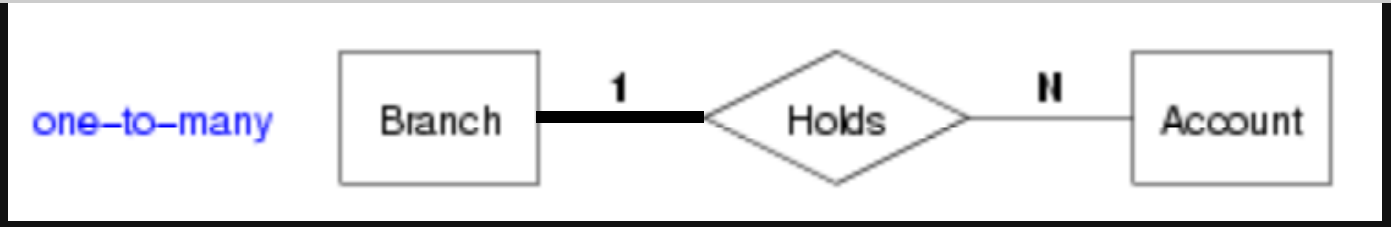
A branch is associated with **N** (many) accounts



A customer is associated with **M** (many) accounts

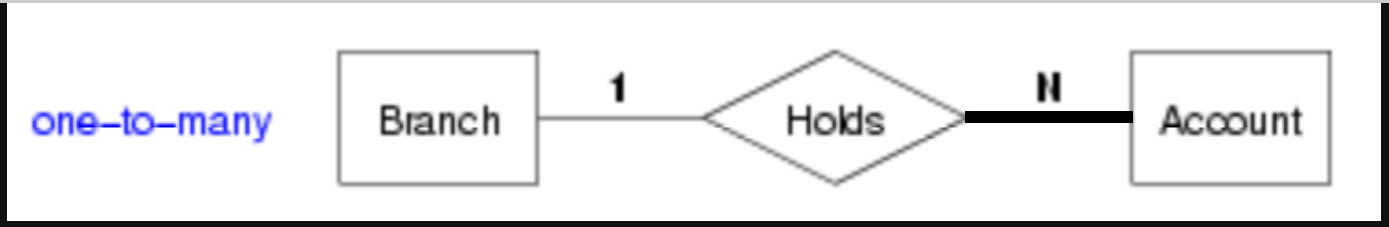
An account is associated with **N** (many) customers

ERD Participation



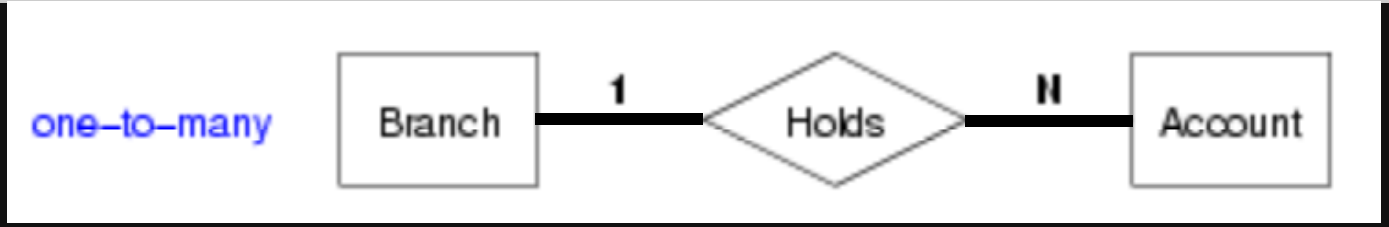
An account is associated with **0** or **1** branch

A branch is associated with **1..N** (many) accounts



An account is associated exactly **1** branch

A branch is associated with **0..N** (many) accounts



An account is associated exactly **1** branch

A branch is associated with **1..N** (many) accounts

ERD Participation

Re-draw the previous ER diagram taking into account multiplicites

Bonus: Write a dummy JSON structure that stores data consistent with the data model