An Overview of Databases and Data Storage

Elise Hellwig

2024-04-11

Workshop Description

This workshop provides a broad overview of the various technologies for storing and organizing different collections of data. We will discuss how data structure and data types impact your storage options, when you should use a database, and which platforms you might consider for your research. This workshop is a general lecture with case studies and Q&A (no laptops necessary). This workshop is a prerequisite for DataLab's Intro to SQL for Querying Databases Workshop (event), and Spatial SQL with SpatiaLite Workshop (event).

Learning Objectives

After completing this workshop, you will be able to answer the following questions:

- What are the most common data structures people use?
- Which data structures lend themselves to answering different types of research questions?
- What is a database?
- What are the differences between relational and noSQL databases?
- What factors impact what type of data store I should use for my research?

Prerequisites

No prerequisites or software required. This workshop is designed for researchers with active or planned data collection and storage.

Credits

All images were created by Elise Hellwig unless otherwise noted. Movie information courtesy of IMDb (https://www.imdb.com). Used with permission for non-commercial purposes only.

References

Henderson, Harold V., and Paul F. Velleman. 1981. "Building Multiple Regression Models Interactively." *Biometrics* 37 (2): 391–411. https://doi.org/10.2307/2530428.

IMDb. 2023. "IMDb Non-Commercial Datasets. IMDb Developer." November 12, 2023. https://developer.imdb.com/non-commercial-datasets/.

Tingeborn, Jonas. 2023. "Imdb-Sqlite: Imports IMDB TSV Files into a SQLite Database." https://github.com/jojje/imdb-sqlite.

Wikimedia. 2024. "Wikidata Query Service. Wikidata." March 5, 2024. https://query.wikidata.org/.

Wikipedia contributors. 2024. "House of Tudor." In *Wikipedia*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=House_of_Tudor&oldid=1212771076.

Data Storage Considerations

When deciding on a method for storing your data (a data store), you have many options. While none of the options will likely be perfect, some are undoubtedly more suited to your situation than others. To figure out which data store will best suit your needs, it is important to consider the following four factors: (1) your data's structure, (2) the ways you anticipate interacting with your data, (3) your data management needs, and (4) the computing resources you have access to.

Figure 2.1. Diagram outlining four factors to consider when choosing a data store.

Data Structure

Your data's structure describes how your data is organized. While the same information *can* be encoded in multiple data structures, certain types of information tend to lend them selves to certain data structures. Additionally, different data structures can make it easier or harder to ask specific research questions. Many data stores impose structure on the data they store. To make your life easier, choose a data store whose structure aligns with the types of research questions you want to ask.

Key Questions

- What types of research questions will I ask?

Data Interaction

Generally speaking, collecting data is only an early step on the road to scientific discovery. Once you collect data, that often needs to be cleaned and reformatted before it can be analyzed. Sometimes you have to repeat this process many times, either for different groups of data, or for different types of analysis. In many cases, you may not need all of your data to answer every research question. Some data stores have built in tools to make data manipulation faster, easier, and reproducible. If you spend a lot of time reformatting your data, consider a data store that has this functionality.

Key Questions

- What kinds of analysis will I do?
- How does my data need to be formatted?

Data Management

Data management concerns the logistics of working with data. It encompasses the challenges of people (including you) interacting with your data over time and space. There are different considerations for data management when you are working on a project alone, as a collaboration, or archiving data for future use. However, similar best practices for data management will benefit them all.

Data Management is a crucial part of valid and reproducible research. Even if you are the only person who will ever interact with your data, you must manage your data in a way that you in the future will be able to recall what past you did, and why. If you collect or modify your data over time, you need to make sure you keep track of those changes, and make sure you are always working with the most up to date version of your data. This could be because you are adding new data or correcting errors in older data. This is doubly important if data gets entered by hand. Some data stores keep track of changes to your data for you and even provide some quality control tools. Others require you to track versions of your data and do quality control manually.

However, science is a collaborative process, so it is more than likely you won't be the only one working with your data. Any time more than one person needs to access or modify a data set, data management gets more complicated. By default, most computers only allow one person to modify a file at a time, and anyone who has access to that file can edit it. Some data store software has methods for dealing with the conflicts or inconsistencies in data that come from allowing multiple people to modify data so you don't have to.

Finally, there will come a time when you move on to other projects. If other people still need to use your data, or if your data is part of a public resource you may need to select or change to a data store that is publicly accessible or can be maintained by other members of the project team.

Key Questions

- When and how is the data collected?
- Who needs access to it? one or more people?
- How long does the data need to persist?

Computing Resources

Data set size can vary widely. A data set containing information about patients of an early stage drug trial may only contain a few dozen observations. On the other side of the spectrum web scraping social media sites can easily generate terabytes or petabytes of data. Different types of data store software have different limits in terms of how much data they can accommodate. Additionally, some software is free for small data sets, but costs money to use for larger ones. UC Davis provides free or discounted software licenses to researchers. Your lab also may have access to software or data storage specific to your area of study. Finally, if you work with data that contains sensitive information, like medical data, you will need to make sure the data store you select is sufficiently secure.

Key Questions

- How much data do I have? Is any of it sensitive?
- Do I have access to external or shared storage space?
- What software to I have licenses for?

Data

> Data (noun): Quantities, characters, or symbols... considered collectively.
>
> - Oxford English Dictionary

Information becomes data when we give it some sort of structure. If you are doing research, chances are you are going to be working with data in some form or another. Even if you do not use traditional spreadsheets, you are certainly working with some form of information. And if you have translated information into a form that is more easily shared or stored for later use, you are working with data. That could be the rows and columns of a table, or something more human readable, like the grammar and syntax of natural language in text form.

One of the challenges of working with data is figuring out a useful way to store it. Thankfully, we have mostly left behind the 20th century's favorite way of storing data: paper in filing cabinets. Modern digital methods definitely take up less space but also require more technical knowledge. There are many more types of data stores available now as well. This means you will need to make a decision on which one to use. **Ultimately, the best way to store your data will depend on the data you want to store and the questions you want to ask with it.**

When describing data, we generally talk about two main characteristics: data type and data structure.

Data Types

Your data's type tells the computer what sorts of operations make sense. Common data types include:

- Integers
- Decimal
- Floating point numbers
- Categories (small, medium, large)
- Characters (text)
- Boolean values (TRUE and FALSE)
- Dates and times

All data, including individual data points, have a type. However, a given piece of information can be stored using multiple data types. For example, you can store the value of 1 as the integer 1, the decimal 1.0, the text "1", the Boolean TRUE, or the date 1900-01-02 (days after January 1, 1900). The underlying information is the same (1), but the computer will treat each of these pieces of data differently. It will not allow you to add "1" to another number, and it may have issues combining the integer or decimal 1.0 with text. Text itself can only be stored as characters.

Data Structures

> A data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.
>
> Wikipedia

Unlike with data types, a single piece of data cannot have a data structure. This is because data structure is about capturing relationships between data points. We will discuss four of the most common data structures: tabular/columnar, graph, tree, and spatial data. In practice, data often exhibits traits from multiple data structures.

Table 1: A subset of the ‘mtcars‘ data set, containing information about 32 cars from the 1974 *Motor Trend* Magazine.

	mpg	cylinders	horsepower	weight	manual	origin
Mazda RX4	21.0	6	110	2.620	1	Japan
Mazda RX4 Wag	21.0	6	110	2.875	1	Japan
Mercedes-Benz 240D	24.4	4	62	3.190	0	Europe
Mercedes-Benz 230	22.8	4	95	3.150	0	Europe
Mercedes-Benz 280	19.2	6	123	3.440	0	Europe
Mercedes-Benz 280C	17.8	6	123	3.440	0	Europe
Mercedes-Benz 450SE	16.4	8	180	4.070	0	Europe
Mercedes-Benz 450SL	17.3	8	180	3.730	0	Europe
Mercedes-Benz 450SLC	15.2	8	180	3.780	0	Europe
Toyota Corolla	33.9	4	65	1.835	1	Japan
Toyota Corona	21.5	4	97	2.465	0	Japan
Lincoln Continental	10.4	8	215	5.424	0	USA
Chrysler Imperial	14.7	8	230	5.345	0	USA
Dodge Challenger	15.5	8	150	3.520	0	USA

Data is information translated into a form to store for later use. The type of information you work with does not dictate the type of data structure you must use. However, some types of information lend themselves to particular data structures over others.

When trying to decide what data structure to use to represent information from your research, the key is to focus on what sorts of questions you would like to ask. Each data structure will make answering some types of questions easier or harder. If you want to ask disparate types of questions, you may need to represent your data using multiple structures. Because data structure has such a large impact on how you interact with your data, the structure you choose is just as important as the analysis you do afterward.

Columnar/Tabular

Tabular or columnar data is made up of rows and columns, collectively referred to as a table. By convention, each row represents an observation, and each column a measurement or aspect of that observation. Tabular data is incredibly common, especially in the sciences. Many researchers, regardless of field store at least some of their data in tabular form. Tables work well when all of the objects in your study can be summarized with the same set of characteristics. Many study designs require this, which is one of the reasons why tabular data is so common.

The table below contains data from the US magazine *Motor Trend* in 1974 (Henderson and Velleman 1981). In the table below, each row is an observation of a car model released between 1973 and 1974. The magazine recorded the make, model and origin of each car and measured the car's fuel efficiency (mpg), horsepower, weight, and number of cylinders in the engine.

Other examples of data sets that lend themselves to a tabular data structure include California almond yields (lb/acre) for farms around the state, climate data over the past 50 years, snow accumulation (inches) for Lake Tahoe, and biomarkers for people participating in a study on a chronic illness.

Tabular data lends itself to answering questions about the average value or spread of a particular measurement over the entire data set. It can also help investigate trends in a particular variable, or how two variables relate to each other. In the case of the car fuel efficiency data, we could ask about the relative fuel efficiency of cars from different parts of the world or if there is any relationship between a car's weight and its horsepower. The tabular data structure also makes sorting observations very easy.

Example Research Questions General Research Questions

- What is the average value of a particular measurement across a data set?
- What is the range or spread of values in a data set?
- Are there trends in the values of a variable?
- How to two columns in a data set relate to each other?

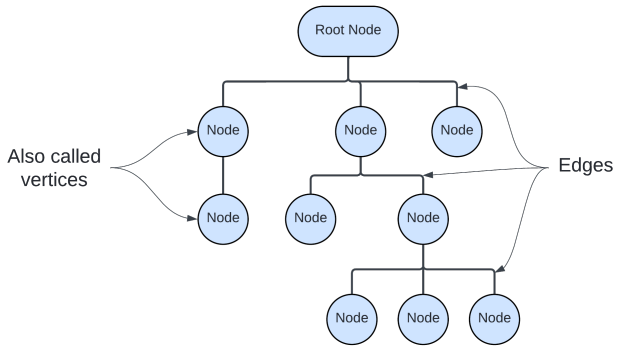
Topic Specific Examples

- What is the is the average yield (lb/acre) for California almond farms?
- What is the normal range of values for a vitamin A blood test in humans?
- Have summer temperatures increased in the Bay Area, and if so, by how much?
- Are there compounds present in blood we can use to detect rabies in urban wildlife?

Trees

Trees are another commonly used data structure. Tree structured data explicitly capture hierarchical relationships between entities. Trees are made of up nodes (or vertices) which represent entities in the data, and edges which represent relationships between those entities. The number of edges separating an entity from a common root node defines that entity's place in the hierarchy. When viewing tree structured data on a computer, it often looks very "nested", like sections and subsections of a document. In fact, tree data is sometimes referred to as document structured data because of this.

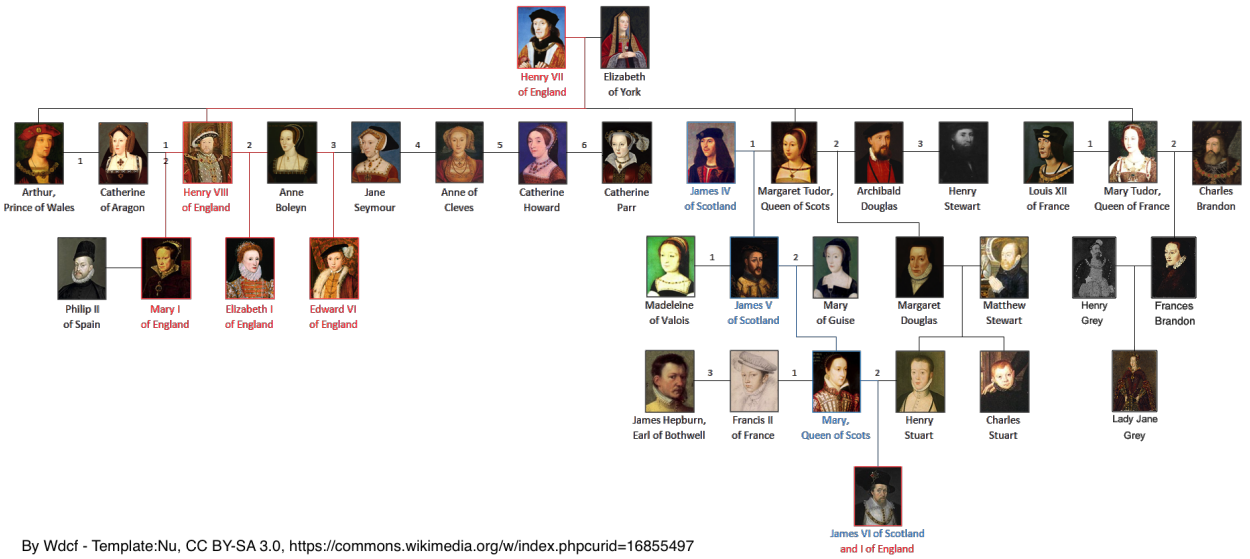
Figure 3.1. An generic example of tree structured data. Intersections are labeled as nodes and the nodes are connected with edges.



In general, tree data facilitates asking questions about particular subsets of the data. For example, you might want to know how closely two entities are related, which branches of the tree have the most nodes, or all of the observations that fall under a particular subtree.

Family trees are a common example of tree structured data. In family trees, the nodes represent people in the family. Horizontal lines represent marriages or couplings, and vertical lines represent a connection to offspring. In the family tree of the Tutor Dynasty in England below (Wikipedia contributors 2024), it is easy to determine things like who had the most wives (Henry VIII), which people shared a generation, and who was the last member of the dynasty (James I of England). Questions like what was the average age at death for the Tudors would be harder to answer.

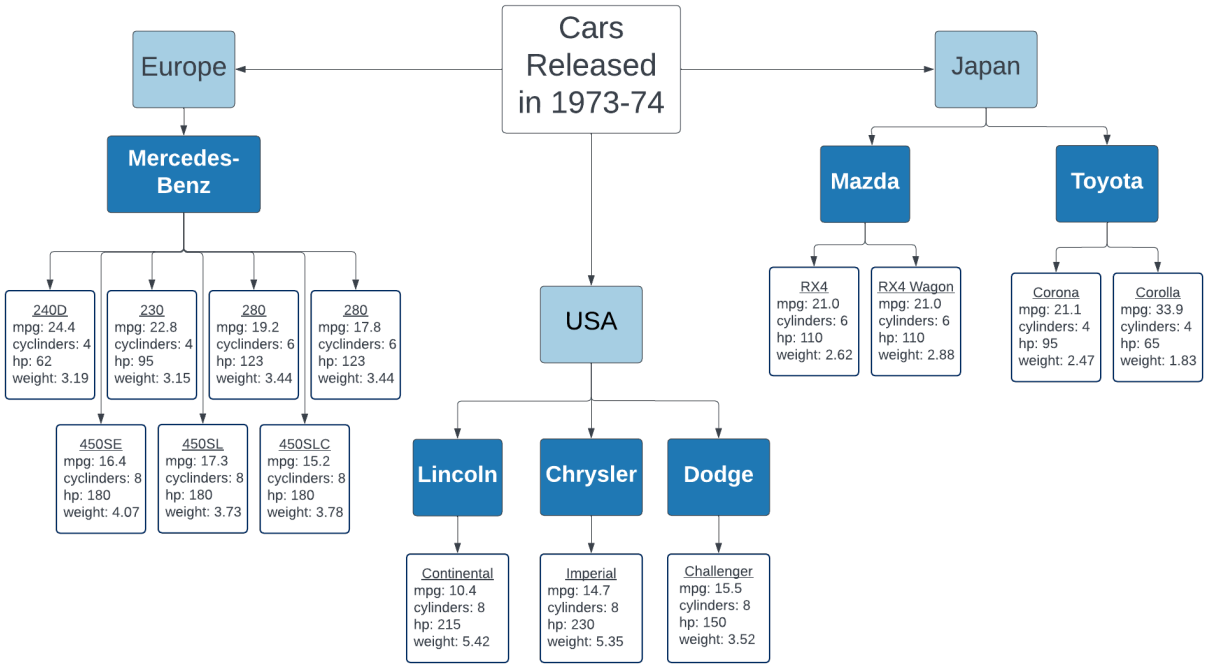
Figure 2.2. Family tree of the principal members of the house of Tudor in England. Red indicates monarch of England. Blue indicates monarch of Scotland.



By Wdcf - Template:Nu, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.phpcurid=16855497

We can also represent the car fuel efficiency data with a tree data structure. This makes it much harder to answer some of the questions mentioned in the previous section. On the other hand, the tree structure makes it very easy to see which car manufacturer produced the greatest number of models in 1973-74 and how many different models came out of Japan during that time period.

Figure 3.3. A tree representation of the subset of the `mtcars` *data set displayed in Table 3.1.*



Example Research Questions General Research Questions

- How many nodes separate two entities?
- Which elements fall under a particular subtree?
- Text or directory searching

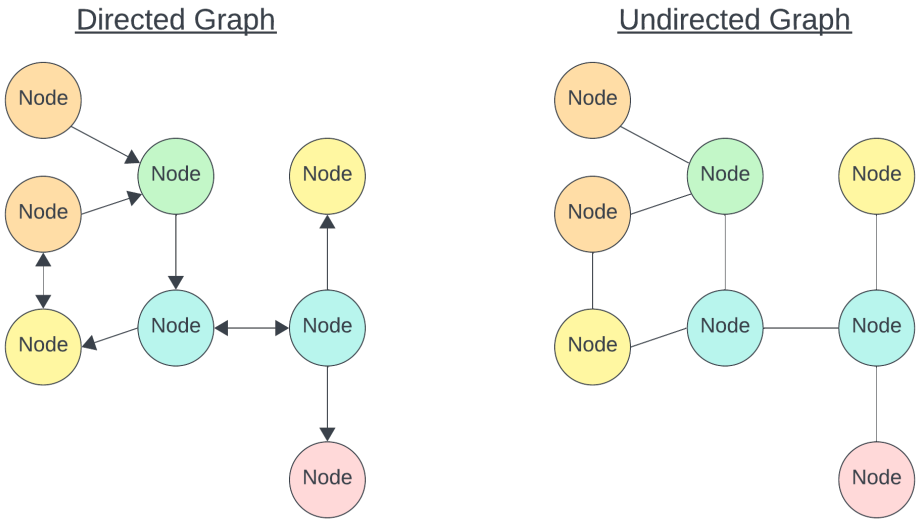
Topic Specific Examples

- Are elephants more closely related to sloths or rhinoceroses?
- How many species of dinosaurs were there and are there any alive today?
- Which journals published the most articles mentioning AI this year?

Graphs

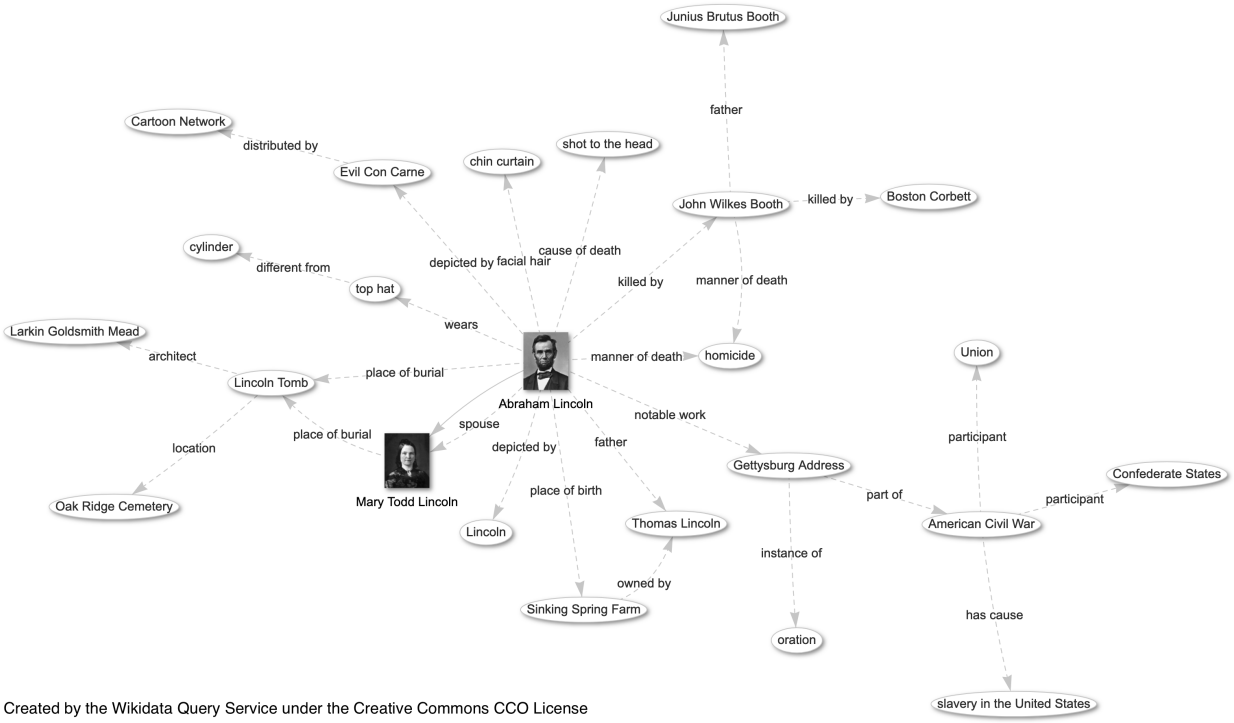
Graph data, also called network data, is similar to tree data in many ways. It also explicitly captures information about relationships between entities using nodes and edges. However, unlike trees, graph data sets have no inherent center or root node, and there is no implicit hierarchy or nesting. Additionally, the relationships between the vertices can be directional, and that direction does not have to be consistent throughout the graph.

Figure 3.4. Two examples of generic graph data, one a directed graph and one and undirected graph. The directed graph has arrows connecting the nodes, while the undirected graph simply has lines.



Graph structured data tends to be more common in the social sciences and humanities. A common example of a graph in popular culture is a social network. Transportation data, like roads and flight paths, also lend themselves to the graph structure. In the digital world, we can use graphs to map out the link connections between Wikipedia pages, as seen below with Abraham Lincoln (Wikimedia 2024).

Figure 3.5. A graph of relationships between Abraham Lincoln's Wikipedia page and other pages linked to it. Each linkage contains a label describing the relationship between the two pages



Graphs make it easy to ask questions about connectivity. For example we might want to know how many flight connections we will need to make between Honolulu, Hawaii and Cairo, Egypt, or how many degrees of separation there are between Abraham Lincoln and Kevin Bacon on Wikipedia. We can also ask about which parts of the graph are the most connected and which are the least connected.

Example Research Questions General Research Questions

- How many edges separate two entities?
- How many paths are there between two nodes?
- Which parts of the graph are most or least connected?

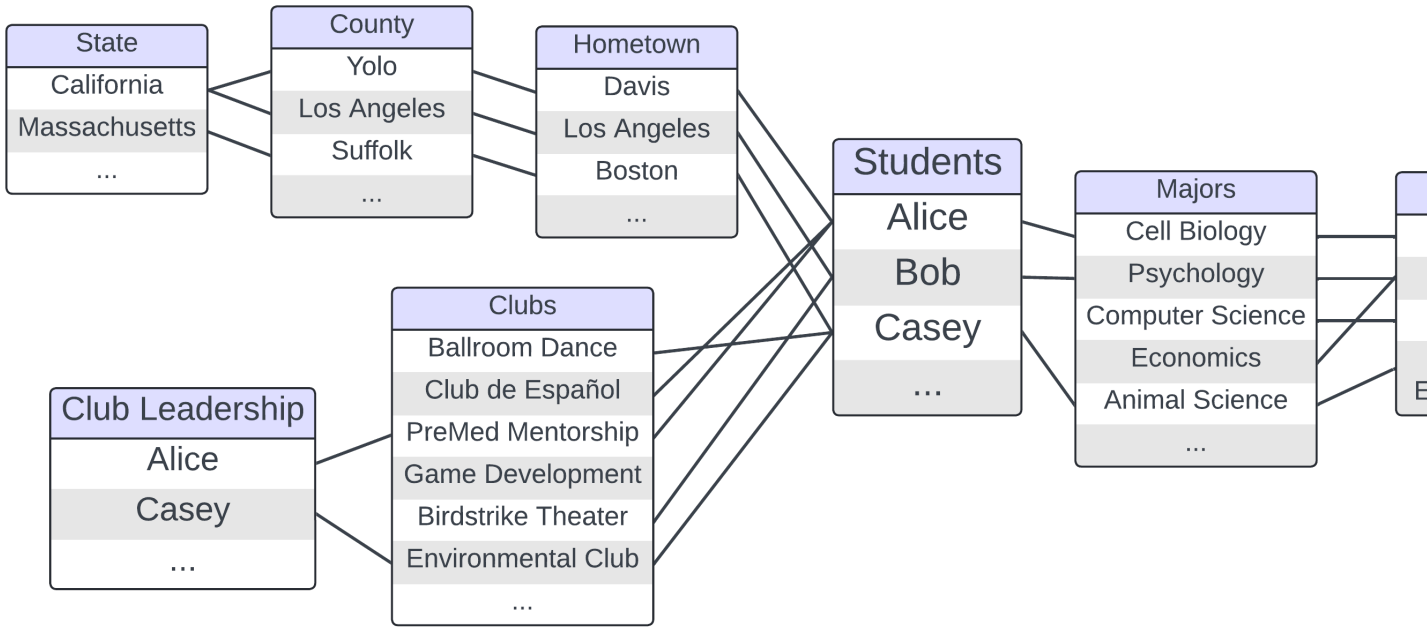
Topic Specific Examples

- How many flight connections needed to fly between Honolulu, Hawaii and Cairo, Egypt?
- Would adding a public transit line between San Francisco and Santa Rosa relieve highway congestion?
- Which jobs in a hospital will most likely expose you to COVID?

Relational Data

In relational data, also called category or group membership data, the organizing principle is group membership. Each group in the data structure has a list of members. Those members can then be linked to members of other groups. For example, Alice is a member of the group "Students". She is linked to Davis in the "Hometown" group, indicating she is from Davis. She is also linked to Cell Biology in the "Majors" group and Club de Español and PreMed Mentorship in the "Clubs" group.

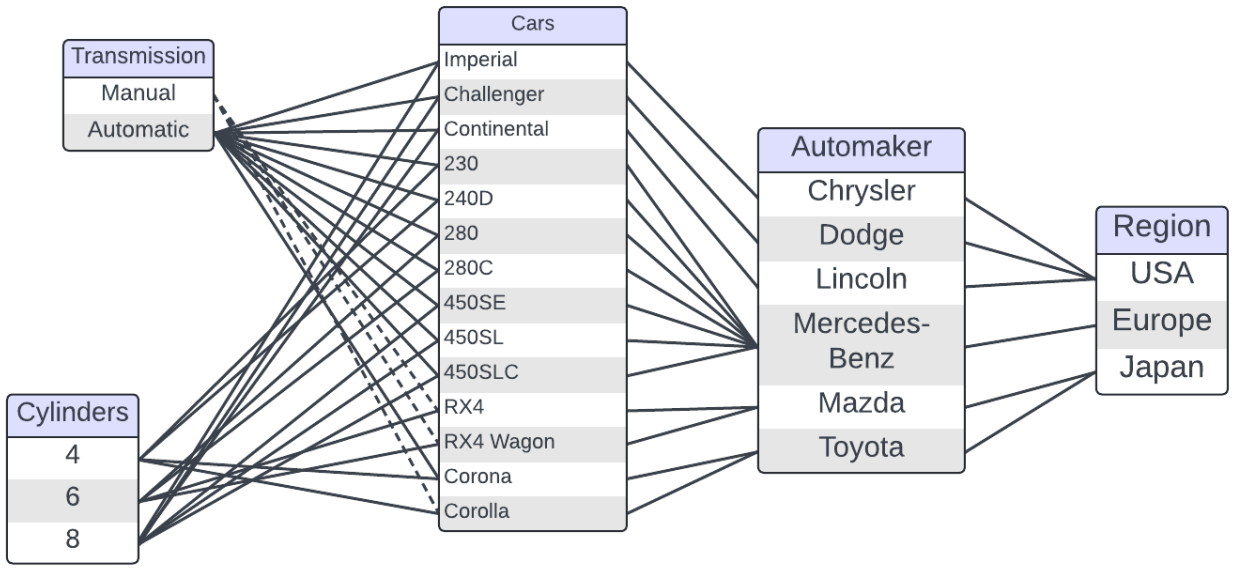
Figure 3.6. A relational diagram of characteristics of fictional first year students at UC Davis.



This data structure has some similarities to graph and tree data. Both tree data and relational data share some level of nestedness in their data structure. However, for tree data every relationship is a nested relationship. By contrast, with relational data, entities can be a part of a group, but they can also be linked to entities in other groups in a non-hierarchical way. Like in graph data, relationship connections link entities in relational data. However, those entities are also nested in groups, something that isn't captured in graph data. Additionally, neither graph nor tree data allows entities to appear multiple times in a data set. However, that is exactly what you see in Figure 3.6. Alice and Casey are members of both the "Students" group and the "Club Leadership" group. This helps capture the multifaceted-ness of some entities, like people!

Like we did in the Tree Data section, we can structure the `mtcars` fuel efficiency data set in a relational diagram. This doesn't give us much more more information than we had with previous structures. In fact, there is a lot of information we had to leave out because it couldn't be captured in the relational structure. Specifically, the relational structure does not handle numeric data well. Because of this, researchers often combine relational data with other data structures in practice.

Figure 3.7. A relational representation of the subset of the `mtcars` *data set displayed in Table 3.1.*



Example Research Questions General Research Questions

- Which groups are the largest or smallest?
- Do certain memberships tend to co-occur?
- How many groups does an entity tend to belong to?

Topic Specific Examples

- Which US state has the most government agencies related to health and medical care?
- Do students majoring in Environmental Science and Management tend to join student groups related to outdoor recreation?

Data Stores

> A data store is repository for persistently storing and managing collections of data which include not just repositories like databases, but also simpler store types such as simple files, emails etc.
>
> Wikipedia

Once you have decided on a data structure, the next decision to make is what type of data store to use for data storage. A data store is a repository for storing and managing data. While this workshop covers exclusively digital data stores, for most of human history, data stores were analog. Even today, we still use analog data stores like libraries, museums, and filing cabinets. Data stores have two primary functions. First, they need to facilitate people accessing the data (reading) and making changes to it (writing). Second, they need to preserve this access over a period of time.

Data Store Definitions

Data Store: Where your data is saved and how you interact with it.

GUI: Graphical User Interface, a software application that allows you to interact with the computer via icons, buttons, and windows. Often referred to in contrast to a command line interface which only allows you to interact with the computer through text commands.

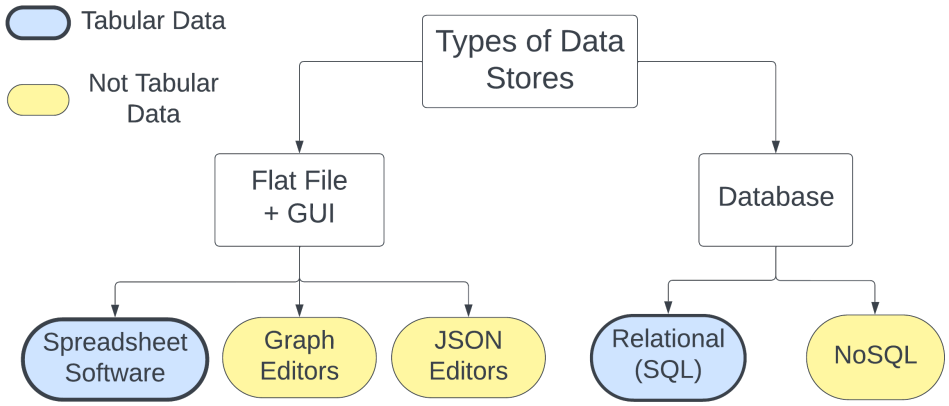
Query: commands that create, read, update, or delete information from a data store, generally entered through a command line interface.

Transaction: the unit of work in a data store. The database keeps track of the order of transactions submitted by each user, to prevent conflicts. A transaction succeeds or fails as a unit, so if an update affects multiple parts of a database, the update only succeeds if every individual part of it is allowed.

Types of Data Stores

Regardless of your data's structure you have two general options for how to store your data digitally: in flat files using a GUI or in a database.

Figure 4.1: A tree diagram enumerating different ways of managing your data, in part based on the data structure.



Probably the most common type of data store is a flat file. A **flat file** is stand alone file on your computer that is not linked to data elsewhere. You can store tabular data in .xlsx and .csv files, tree data in .xml and .json files, and graph data in .GML and .DOT files. Generally, people interact with data stored in flat files via a GUI. Most GUIs are specific to file type or data structure. For example, spreadsheet applications like Microsoft Excel or Google Sheets can edit xlsx and csv files, and applications like Graphviz can visualize and modify DOT files.

In comparison, a **database** is an organized collection of data that is linked or related in some way. Users interact with a database through specialized software called a database management system, often abbreviated DBMS. Most databases do not have a GUI. This means you will be interacting with the database through a query language instead of clicking on buttons. If you already use a programming language like R or python, this will be very familiar to you. If not, don't worry. Query languages, like SQL, are designed to be simpler and easier to use than most programming languages, but also incredibly powerful.

Database management systems have several advantages accessing flat files with a GUI.

1. **Structure** - Databases can store data with structures ranging from low to high complexity.
2. **Interaction** - Database queries make data interaction and formatting reproducible. If you use the same query on the same database, you will get the same result.
3. **Management** - Databases allow multiple people to interact with data at the same time without locking people out or introducing conflicts. They also provide fine-grained controls on who can access, modify, and delete data.
4. **Computing Resources** - Many GUIs have fairly restrictive size limits on how big a file can be. Databases do not generally have these restrictions, or if they do, the size limit is much higher.

Databases can also formalize some processes that should be done on flat files but often are not. The first of these is version control. When working with data it is vitally important to define which version of the data is the correct and up to date. This is particularly important when multiple people have the data stored locally

on their computers. Databases provide a definitive version of the data that users can access remotely. This ensures that everyone working on a project uses the same data.

The second is administration. Because databases can set controls over who has access to what, you can control who has the ability to add data to and modify the database. You also need to designate someone to be in charge of fixing things if something in the database breaks. This is also true for flat files but because everyone has the same access it is often not clear whose job it is.

Some databases also provide tools to prevent mistakes and maintain data integrity. This includes restricting data to specific data types or specific sets of values. This means if you type a "w" instead of a "2", the database will tell you to check your data and try again.

If you are running into any of the following issues, a database may be the solution to your problems.

- Your data set has many repeated values
- Your data set contains information about many unrelated or loosely related studies
- You are sharing data with multiple people and everyone seems to have a different version
- Your data set is so large, your computer slows down or freezes when trying to view it
- You have to keep making the same corrections to your data over and over again
- Excel keeps converting your columns to the wrong data type
- You have so many pivot tables
- You have hundreds (or thousands) of data files you need to share with your collaborators
- You think PowerQuery is neat but wish people would stop changing the underlying data while you are using it.
- You have a hard time finding and extracting the information you need to answer a particular research question
- Processing your data requires you to spend a lot of time moving data between spreadsheets by hand.

Spreadsheet Software

Using spreadsheet software with tabular data files is very a common form of digital data store. If you have used Microsoft Excel or Google Sheets, you have used spreadsheet software. Like some databases, spreadsheets impose a tabular data structure on the data they store. Despite this, a spreadsheet is not a database. Like most modern software, spreadsheet applications have graphical user interfaces, making the barrier to entry very low. Spreadsheet software also offers some built in analysis tools, so you don't have to learn a new application to do basic statistics and data visualization.

Spreadsheet software also suffers from some drawbacks. They offer limited storage capacity, making it difficult to work with large data sets. Excel and LibreOffice have a maximum row number of a little over 1 million and Google Sheets limits you to 10 million cells. Software operations will also slow down if you get close to their data capacity. This means analyzing large data sets is much more tedious than analyzing small ones.

Spreadsheets also store data in ways that cause a lot of data duplication and make data entry susceptible to typos. In the spreadsheet visualized below, we have data on the Die Hard movie franchise (IMDb 2023). Specifically, we have information about the actors that starred in the series and the movies in which they appeared.

Looking at the last three columns, there is a lot of data duplication, meaning our data set takes up more space on our hard drive than it needs to. Additionally, if we have to change the information about a single movie or actor, we would have to make that change in many different places, and potentially miss one or make a mistake. If we accidentally left out an "l" in Bruce Willis's last name, we would have to modify five cells to fix our data.

Table 2: Information from IMDB about the Die Hard movie franchise and the people who acted in those films.

title_id	person_id	title	name	characters	prem
tt0095016	nm0000246	Die Hard	Bruce Willis	[""John McClane""]	
tt0095016	nm0000614	Die Hard	Alan Rickman	[""Hans Gruber""]	
tt0095016	nm0000889	Die Hard	Bonnie Bedelia	[""Holly Gennaro McClane""]	
tt0095016	nm0001817	Die Hard	Reginald VelJohnson	[""Sgt. Al Powell""]	
tt0099423	nm0000246	Die Hard 2	Bruce Willis	[""John McClane""]	
tt0099423	nm0040472	Die Hard 2	William Atherton	[""Thornberg""]	
tt0099423	nm0000889	Die Hard 2	Bonnie Bedelia	[""Holly McClane""]	
tt0099423	nm0001817	Die Hard 2	Reginald VelJohnson	[""Al Powell""]	
tt0112864	nm0000246	Die Hard with a Vengeance	Bruce Willis	[""John McClane""]	
tt0112864	nm0000460	Die Hard with a Vengeance	Jeremy Irons	[""Simon""]	
tt0112864	nm0000168	Die Hard with a Vengeance	Samuel L. Jackson	[""Zeus""]	
tt0112864	nm0001295	Die Hard with a Vengeance	Graham Greene	[""Joe Lambert""]	
tt0337978	nm0000246	Live Free or Die Hard	Bruce Willis	[""John McClane""]	
tt0337978	nm0519043	Live Free or Die Hard	Justin Long	[""Matt Farrell""]	
tt0337978	nm0648249	Live Free or Die Hard	Timothy Olyphant	[""Thomas Gabriel""]	
tt0337978	nm0702572	Live Free or Die Hard	Maggie Q	[""Mai""]	
tt1606378	nm0000246	A Good Day to Die Hard	Bruce Willis	[""John McClane""]	
tt1606378	nm2541974	A Good Day to Die Hard	Jai Courtney	[""Jack McClane""]	
tt1606378	nm0462407	A Good Day to Die Hard	Sebastian Koch	[""Komarov""]	
tt1606378	nm0935541	A Good Day to Die Hard	Mary Elizabeth Winstead	[""Lucy""]	

Finally, most spreadsheet applications make it difficult to work with multiple people on a single data set. Generally, GUIs only allow one person to edit a file at a time, and setting user permissions is cumbersome. They also make it easy to collect many versions of a given data set. So unless you are diligent about naming, you may end up with people working on different different versions of your data. Google Sheets offers shared access and some version control, but still fails to deliver on storage capacity, and data integrity.

Spreadsheet Software Decision Factors

1. **Structure:** tabular
2. **Interaction:**
   - Graphical User Interface
   - Some offer built in analysis and visualization tools
   - reformatting is hard
3. **Management:**
   - Managing multiple users is challenging
   - Easy to introduce typos or erros when adding data
4. **Computing Resources:**
   - Software is easy to use and widely available
   - GUI limits data size
   - Lots of data duplication makes data larger

Examples

- Microsoft Excel
- Google Sheets
- Numbers
- LibreOffice Calc
- OpenRefine

- Open Science Framework

Relational Database Management System

Relational Database Management Systems (RDBMSs), or relational databases, are one type of database. When people use the word "database", nine times out of ten they are referring to a relational database. Like spreadsheets, RDBMSs impose a tabular structure on the data they contain. A basic RMDBS is a collection of tables connected by shared columns called keys. However, they can be more complex.

Users often interact with an RDBMS through the Structured Query Language (SQL) instead of a graphical user interface, though there are exceptions. SQL is a programming language, but it is one specifically designed for interacting with relational databases. SQL standardizes your interactions with a database, and makes them reproducible. This means it is much easier to ensure everyone is working with the same set of data. Working with SQL is a skill, and will take some time to acquire. However, because of SQL's narrow scope, is easier to learn than a more complex programming language like python or R.

To learn about using SQL in practice, check out DataLab's Intro to SQL (reader, workshop) and Spatial SQL (reader, workshop) resources.

Relationship keys

A relationship key is a column or pair of columns that links two tables. One example of a relationship key you may use is a student or employee ID number. You can use that number to link information about you in one part of the University to another part of the University. This is what allows you to pay for meal plans via the Student Accounting department and you can use your meal plan through Dining Services. The student ID number ensures that the money you spend gets credited to the correct account.

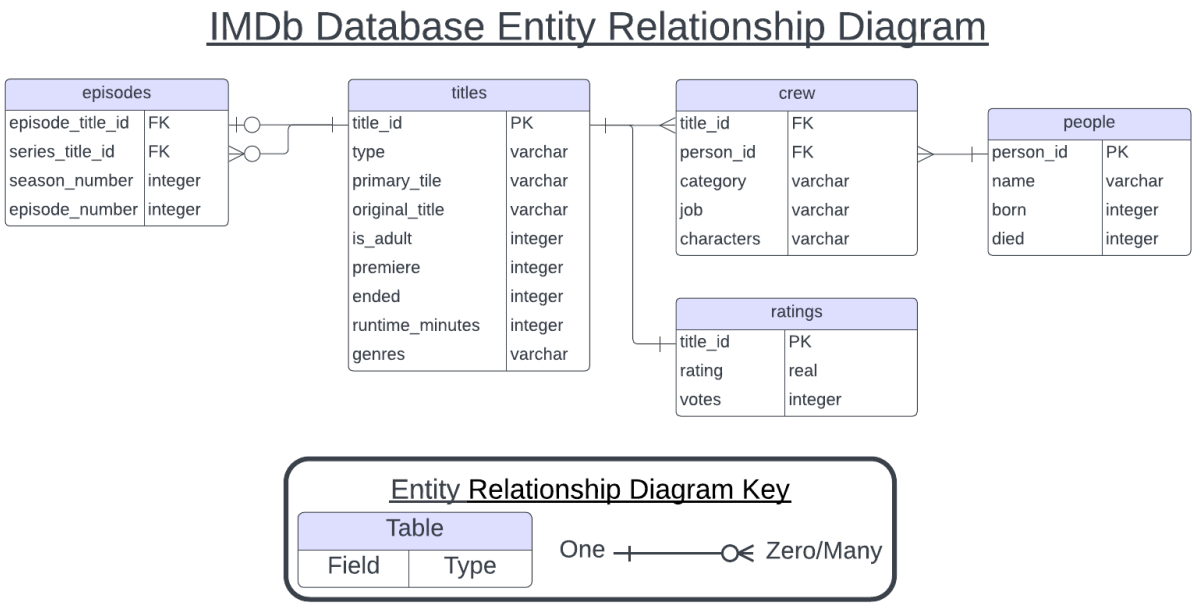
There are two types of relationship keys: primary keys and foreign keys. A primary key, sometimes called an index, is a column that uniquely identifies each row in a particular table. It is generally the first column in the table. A foreign key is a column in one table whose values correspond to a primary key in another table. Student ID would be a primary key in a table that lists all of the students at the university with some basic information about them. It would be a foreign key in a table that lists all food and beverage sales at a particular on campus eatery.

Entity Relationship Diagrams

An Entity Relationship Diagram (ERD) provides a visual summary of of the information in a database without needing to interact with it. ERDs are a standardized way to depict the structure of a relational database. Each box in the diagram represents a table, with the table title at the top and the column names listed below it. The column data type appears next to each column name.

Connector lines link primary/foreign key pairs. These lines specify which tables can be combined, and which keys should be used to combine them. The connector lines also specify what type of relationship between the two tables.

Figure 4.2. A partial Entity Relationship Diagram for the IMDB non-commercial database (IMDb 2023; Tingeborn 2023).



The first type of relationship is a one-to-one relationship. Two tables are one-to-one when one observation in table A only corresponds to one observation in table B. This is the relationship between the titles and ratings tables in the IMDB database. The next type of relationship is a one-to-many relationship. In this case, one row in table A may be linked to many rows in table B. This is the relationship between the title and crew tables in the IMDB database.

Finally, there is the many-to-many relationship. In this case more one row in table A can correspond to multiple rows in table B AND one row in table B can correspond to multiple rows in table A. The classic example of this is cities and zip codes. Many cities contain more than one zip code, but some zip codes also span multiple cities.

Why Use Relational Databases?

There are four primary reasons to use a RDBMS as a data store:

1. Reduce data duplication
2. Speed up and standardize accessing and updating the database
3. Ease of reformatting data
4. Ensure data integrity

In our Die Hard movie spreadsheet example, we had 260 cells. However, in a relational database, storing that same data only requires 156 cells. That is a 40% reduction in data size. For a multi-gigabyte data set, 40% makes a huge difference. Additionally, much of that information was stored as numbers instead of text, which makes its size on even smaller. Finally, we selected a subset of columns from each table, which means we didn't need to work with ALL of the data just because we wanted to work with some of it. Subsetting like this is much harder in traditional spreadsheet software.

The source database of our Die Hard data, IMDB, is massive relative to spreadsheet software capabilities. The titles table alone has 10 million observations, and the crew table has over 60 million. Despite this, extracting the Die Hard movie franchise information using an SQL query took about half a second.

The SQL query used to extract the Die Hard data is also reproducible. If I want to collaborate with a colleague on my analysis of Die Hard, I don't have to send the data set to my collaborator. Instead I can provide them access to to the IMDB database and my SQL query. If I find problems in the data at a later

time, I don't need to send a whole new data set, which could introduce confusion about which data to use. All I need to do is update the database, and tell my colleague to rerun their SQL query. No additional computer storage necessary.

With an RDBMS, even though the data you analyze may have duplicates, you don't need to modify each duplicate value, in the case that one requires a correction. All changes to the data can be made to the singleton values in their original tables. Correcting Bruce Willis's misspelled last name only requires fixing his name once in the "people" table. Then, any time you extract data from the database table, that change will automatically propagate to the new data. While it may be easy to successfully fix 5 errors, it will be much harder to fix 50, or 5,000. This is especially true if they are scattered throughout your data set and not just in a single column.

Relational databases also have a built-in system for managing multiple people interacting with the same data. Whenever someone submits a transaction to a RDBMS, the database puts that transaction in a first come, first served queue. If the transaction is successful, the entire database is updated and any future transaction will work off the new data. If the transaction fails, nothing is updated and the data stays as is for the next transaction in line.

RDBMS Decision Factors

1. **Structure:** tabular, relational
2. **Interaction:**
    - SQL (Structured Query Language)
    - Data manipulation is easy and reproducible
3. **Management**
    - Built in data quality control and multi-user management
    - Standardizes data updates
    - Provides definitive version of your data
4. **Computing Resources**
    - Fewer size restrictions
    - Requires some specialized setup
    - Thousands of transactions per day

RDBMS Software

Because of relational databases' widespread utility, there are many software options to choose from when creating a database. While all of them work off the same basic SQL, each one puts its own particular spin on the language in terms of additional functionality. All of the SQL database software options in the table below have wide community and/or professional support. There are many more RDBMS applications available for more specific use cases (ex. for use with Amazon AWS), but these are the most common and widely supported. All of them provide some level of support for spatial data. However, that level of support varies, so if you have specific requirements, it's best to do additional research before making a decision.

NoSQL Databases

Not Only SQL, or NoSQL, databases are another type of database. They are primarily defined by what they do not do, namely they do not store data in relational tables, as a RDBMSs would. Instead, NoSQL databases do not necessarily impose external structure on the data they contain. In fact, NoSQL databases can contain data from multiple structures, including

- documents
- key-value pairs

Table 3: A comparison of some of the more common software implementations of SQL databases.

Software	Support	Cost	License
SQLite	Extensive documentation and community support	Free	Public Dom
PostgreSQL	Technical documentation, with robust community support	Free	Open Sourc
MySQL	Large community base, No professional support without a paid subscription	Free	Partially Op
Microsoft SQL Server	Robust professional and community support	Paid	Proprietary
Oracle	Robust professional and some community support	Paid	Proprietary
Microsoft Access	Not supported, do not use	NA	Proprietary

- graphs
- trees
- wide column tables

This makes NoSQL databases are much better at storing tree and graph data, as those data structures do not fit well into relational tables. Like RDBMSs, NoSQL users interact with their database via a query language. However, the exact one depends on the database software. This is in part because different data structures lend themselves to different types of operations.

Why Use NoSQL Databases?

1. Compatible with multiple, non-tabular, data structures
2. Data transactions are much faster than other data stores
3. Adding storage capacity is easier than for other data stores

One of the strengths of RDBMSs is that we can easily combine data from different parts of the database using joins. However, functionality does come with a cost. Using joins means relational databases must move a lot of data around, which can slow down operations. Because NoSQL databases do not have this feature, they can respond much faster to requests for information. While a RDBMS can support a few thousand transactions per day, a NoSQL database can run millions of transactions a day.

This lack of data joins also means NoSQL databases can also store data in in many different locations without having it affect the database's response time. This means a single NoSQL database could run on just as easily multiple on multiple machines as one machine. The practical implication of this is that adding more storage space or processing power is much easier. You can add an additional computer instead of needing to upgrade the one you already have.

One common use case for NoSQL databases is for social network sites like Facebook and Twitter/X. Social media contains a lot of text data that does not fit neatly into the tabular structure mandated by relational databases. Additionally, minimizing response time is critical for a company who wants to convince more people use their website or platform. Finally, social media generates massive amounts of data, which means companies like Twitter/X and Meta regularly need to expand their storage capacity, something that is much easier with NoSQL databases.

NoSQL Decision Factors

1. **Structure:** documents, trees, graphs, key-value pairs, wide column tables
2. **Interaction:**
    - Via various query languages
    - Data manipulation is easy and reproducible
3. **Management**
    - Multi-user management

Table 4: A comparison of some of the more common software implementations of NoSQL databases.

Software	Data Structure	Query Functionality	Transactions	Cost	Spatial Data
MongoDB	Tree/Document	SQL-similar	Multi-document	Free	Vector
Couchbase	Tree/Document	SQL-similar	Multi-document	Free	Vector
Cassandra	Column Family	SQL-similar	Limited	Free	Vector (DataStax)
HBase	Column Family	Limited	Multi-row	Free	
Redis	Key-value	Limited	Limited	Free	Point only
Neo4J	Graph	Graph traversal focused	SQL-like	Free	Vector
ArangoDB	Tree/Document, graph, key-value	Read/update focused	SQL-like	Free	Vector
DynamoDB	Document, key-value	Basic	SQL-like	Paid	Point only
Elastisearch	Document	Text search focused	Limited	Free	Vector

- Standardizes data updates
- Provides somewhat definitive version of your data

4. **Computing Resources**
    - Fewer size restrictions
    - Requires some specialized setup
    - Millions of transactions per day

Example Software

Not every NoSQL database supports every data structure. Some support multiple structures while others specialize in one. They generally also low cost to implement, though you can always pay for additional features and support. Similar to RDBMSs, all the most common NoSQL database software provides some support of spatial data. However, that support is limited to vector data and the functionality available varies fairly widely.

Choosing the Right Data Store

You've just learned a bunch of information about data structures and data stores. Now the challenge is applying that information to your own data. Remember the four factors to consider when deciding on a type of data store:

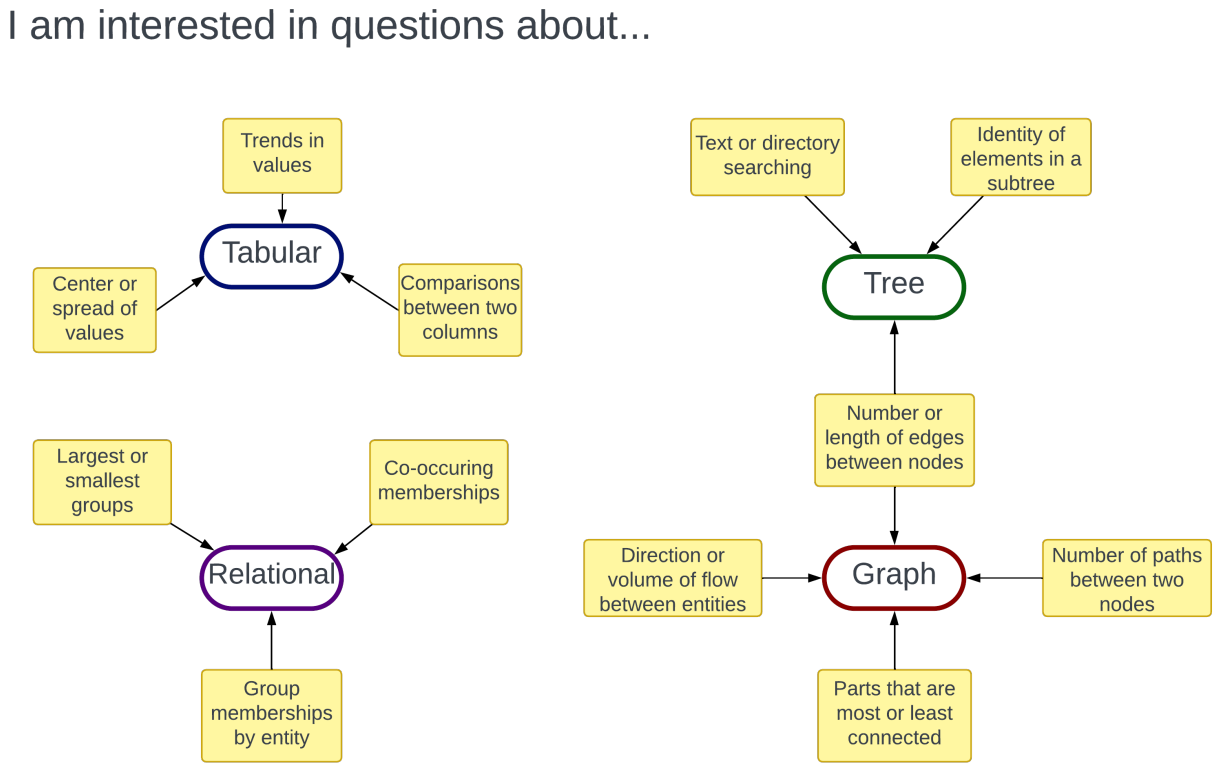
1. Data Structure
2. Data Interaction
3. Data Management
4. Computing Resources

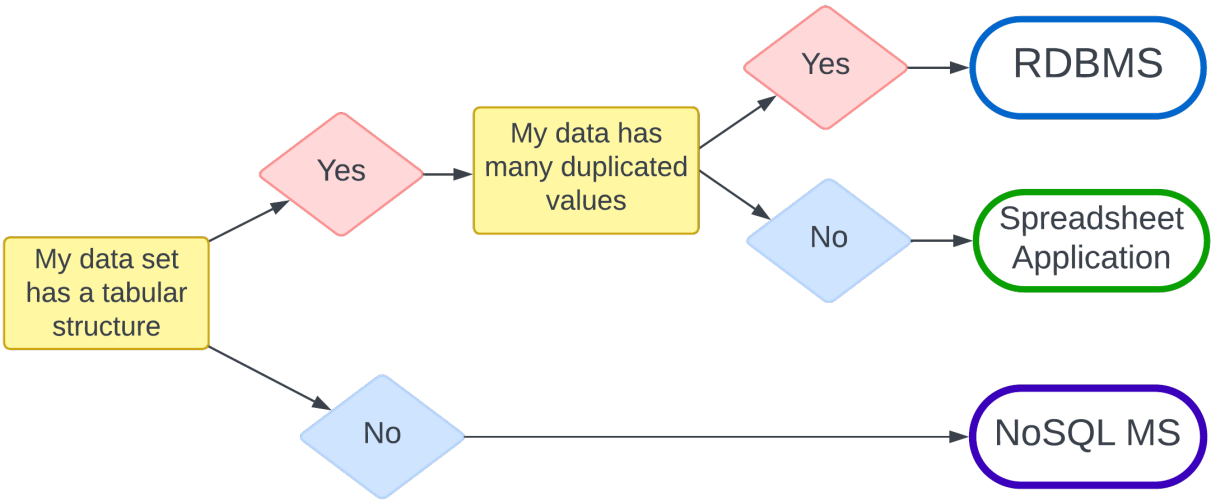
Decision Flow Charts

While each factor compasses a wide range of considerations, the flow charts below can act as a starting point for deciding on the best data store type for your research.

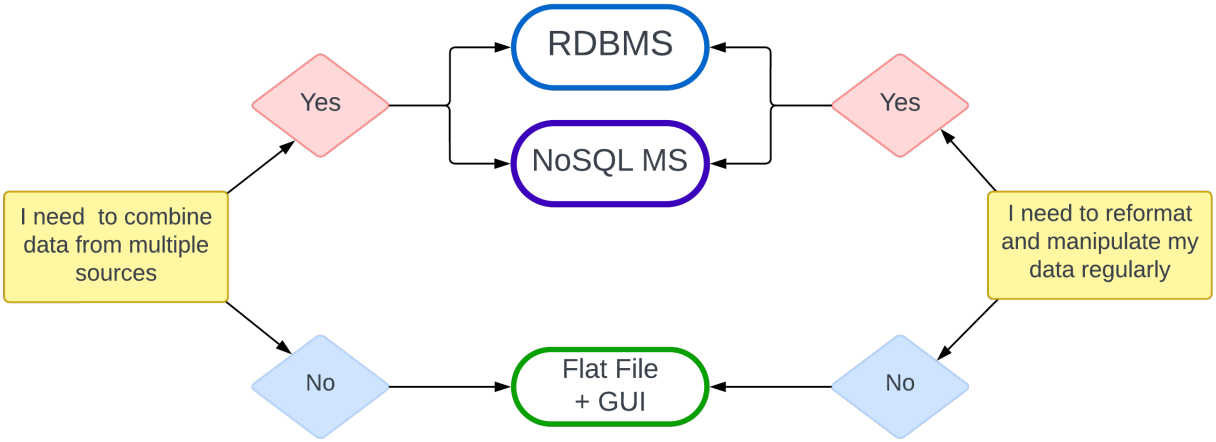
You may find that various parts of your data require different answers to the questions in these decision charts. Or, your answers to the questions may change over time. You may want to ask different questions of your data, and therefore need to change how you structure your data. You may start working with more collaborators and need provide other people access to your data. Reproducibility experts will sometimes assert that you only store one "definitive" version of your data. However, that plan rarely survives first contact with a research project. If you want understand a subject deeply, you need to ask many different types of questions about it. And this necessitates storing your data in multiple forms.

1. Data Structure

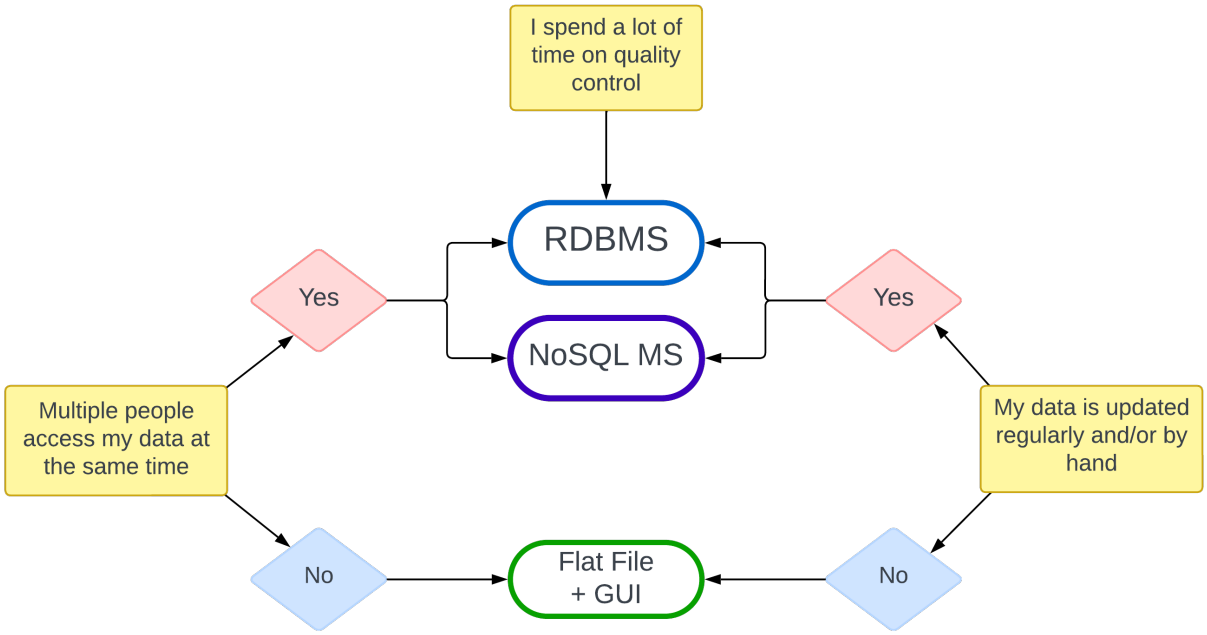




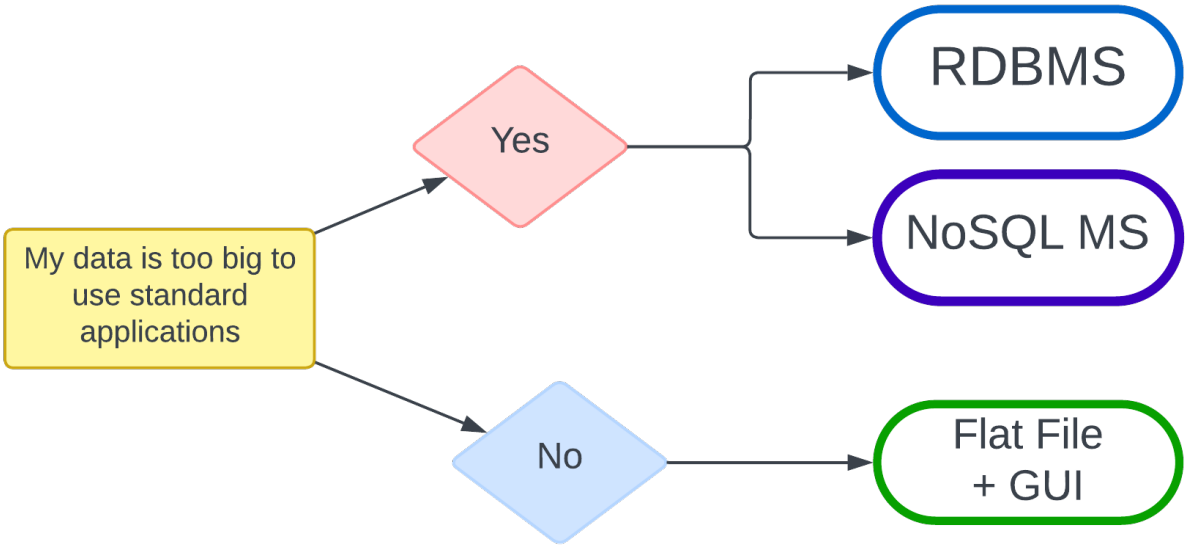
2. Data Interaction



3. Data Management



4. Computing Resources



Additional Resources

This workshop is just the beginning. If a database seems like a good data store for your current (or next) research project, the resources below can help you get started.

General Database Information

- An Overview of Databases and Data Storage Reader
- Intro to Databases (for people who don't know a whole lot about them)
- Database Management System (DBMS) Tutorial
- Creating a database from scratch
- NoSQL Explained
- Introduction to NoSQL

Learning Query Languages

- Intro to SQL for Querying Databases Reader
- Spatial SQL Reader
- W3Schools SQL Tutorial
- W3Schools PostgreSQL Tutorial
- W3Schools MongoDB Tutorial

Campus Database Resources

- UC Davis DataLab Office Hours:
  - Mondays, 1-2pm via Zoom
  - Tuesdays 1-2pm in person (Shields Library, room 360)
- Research Data Services, and the research services librarians

- Campus Data Services Catalog
- Research Data Management Topic Guide
- UC Davis Cloud Services, includes Amazon Web Services, Microsoft Azure, and private cloud services
- Research Electronic Data Capture (REDCap)

Assessment

Describe a project you are working on and specify a particular research question from that project. Alternatively, select from one of the three projects below.

What data structure and data store combination would be the best option for storing your data? What would be a good second option? What option would a a bad option? Explain your reasoning in each case.

Research Question 1: You want to investigate the potential effects of climate change on agriculture in the state of California. You are collecting almond flowering data from almond farms across the state and combining that with weather station data to see how changes in winter and spring temperatures affect the almond flowering dates. You collect digital flowering records via email, but some are only in hard copy and will be sent through the mail.

Research Question 2: You are interested in how the structure of peoples' social circles impacts the way a disease spreads through a population. You have created 1000 digital populations of 5000 people with different levels of connectivity between individuals and groups. You are the only one in your lab comfortable working on computer simulations, so this is likely to be a solo project.

Research Question 3: You are investigating how the meaning of the word footprint has expanded with the advent of the environmental movement and the internet. You are text mining the most prominent journals over the past 50 years to see how the context in authors use the world "footprint" has changed in that time. You are working with several collaborators to curate the list of journals and compile the data.