# Structures Inherent to Data Storage Technologies

This unit provides a detailed discussion of the third Principal of Data Storage:

> All Data Stores Impose an Underlying Data Structure

Remembering our definition of Data as "information in digital form," we can see that it would be impossible to store this information digitally without imposing some for of organization. At a minimum, any information that we want to place in our data store must be converted into an organized collection of 1s and 0s to be stored digitally. This, in and of itself, represents a rudimentary data structure.

Unpacking the complexities of the data structure impositions of various data storage technologies can be difficult. This is because these structures are both imposed and present possibilities and limitation at different stages of the research lifecycle. Additionally, the degree to which structures are imposed varies widely from technology to technology.

For example, one of the simplest and most common approaches to storing data is a filesystem approach wherein collections of simple ASCII files containing data (most commonly in comma separated value format) are stored in an organized manner on the computer’s filesystem. This is a legitimate approach to data storage, which has the distinct advantage of placing a very low barrier for interaction with the data. There are literally several hundred current software packages, both free and paid, available for reading .csv files on all computer operating systems. But these software packages do little more than read the files, and performing searchers, organizing, and otherwise working with the data requires non-trivial effort every time you engage with the data. Additionally, the software provides no systems for data management in a team environment where multiple researchers are working simultaneously on the same data, nor does it have systems for checking and otherwise maintaining integrity of the data.

On the other side of the complexity universe stand Relational Database Management Systems (RDBMS) such as MySql and Postgres. These systems provide robust data integrity checking, transactional management of data interaction that accounts for frequent and multiple updates my multiple users, and provide rapid, multithread querying. But these systems store data in non-human-readable and proprietary binary forms of data that unreadable without specialized software.

The following sections discuss a variety of currently available and widely adopted data storage technologies. Each is presented under the heading of the type of data structure the system optimally supports.

## Tabular/Columnar Data

\*\*delimited text files\*\*

A delimited file is a simple, ASCII file that holds tabular data by placing each observation on a single line and separating columns/variables by a defined delimiter such as a tab, a semicolon, or a comma. The most common form of delimited file is the Comma Separated Value (.csv) file, which uses a character delimiter, usually as semicolon rather than a comma, oddly, to separate variable values. The figure below shows a snapshot from a .csv file.

Advantages: Ubiquitous software; human readable; simple.

Disadvantages: No data management and integrity checks; exponential degradation of query performance with size of data.

[insert figure]

\*\*binary data files\*\*

Data manipulation and analysis systems and scripting languages typically support environment specific binary data formats for storing tabular data. R, for example support the R Data Structure (.rds) data format.

Advantages: Fast input and output of larger datasets; small storage media footprint;

Disadvantages: Data can only be accessed by more expert users proficient in the required environment and/or language; No data management and integrity checks.

\*\*Excel files\*\*

Advantages: Robust user interface for basic analysis and manipulation. Reasonably ubiquitous access to software.

Disadvantages: Proprietary data structure; No data management or integrity checks.

\*\*RDBMS\*\*

Many users choose to store their tabular data in a Relational Database Management System (RDBMS) in what is known as a ‘wide table’ or ‘long table’ format in order to capitalize on the data management and integrity checking functionality of RDBMS. “Wide” or “long” table forma describes a relational database with no relationships. All data is stored in a single, “wide” table that functions just like a spreadsheet.

Advantages: System level data management and integrity.

Disadvantages: Actual data not human readable; implementation of software requires a server and server administration effort.

## Tree Data

Tree data is generally best managed using a “Document Store” or “noSQL” data store solution. Currently, the most widely adopted of these systems include platforms such as “s, y, & z”. Document Stores provide systems for indexing collections of documents that contain hierarchical data represented in formats such as XML and JSON. The figure below shows the same, simple dataset represent as both DML and JSON.

[insert figure]

These files can be prepared using general text editors or using specialized software that validates the tree structure and hierarchical schema of documents. The text preparation software packages are generally created and maintained separately from the document store index and query engines, so there is a great deal of flexibility in this universe. Putting aside minor differences in available document store platforms, all are roughly equivalent in terms of usability and performance. As such, we provide the following, combined assessment:

Advantages: Best solution for hierarchical data;

Disadvantages: Limited built-in support for data management and data integrity checking. However, text-based XML and JSON files can be managed using Git or other version control systems, which will ensure data integrity.

## Graph Data