

Data transformation

There are two ways to write error-free programs; only the third one works.

Alan Perlis

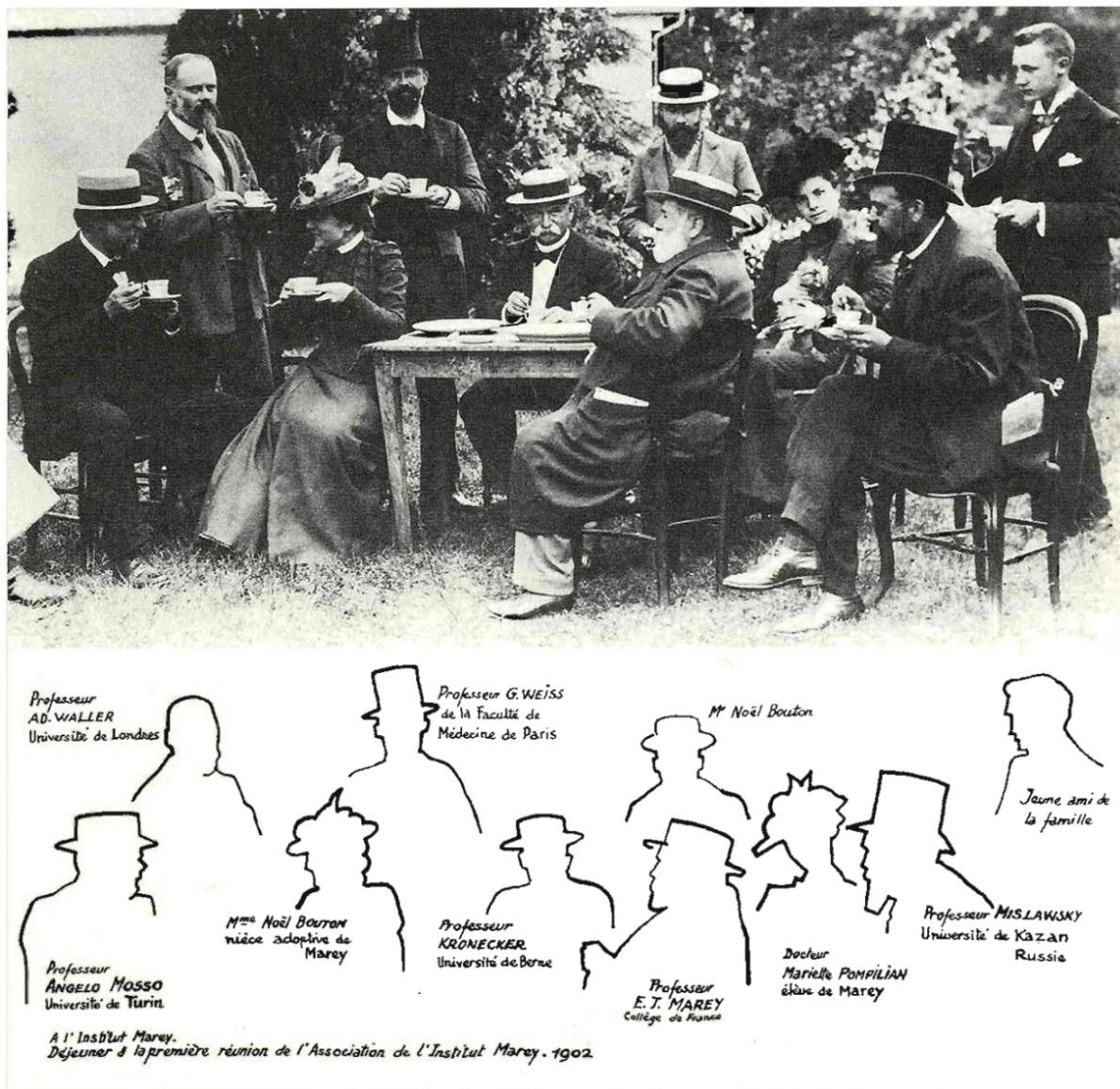


Figure 1: This picture and mapping of a 1902 reunion of colleagues of E.J. Marey, the great scientific photographer, provides names and affiliations linked to hat-head outlines that run parallel with the image. The diagram allows label-captions of greater detail than labels placed on the picture itself. (From "Beautiful Evidence" by Edward Tufte, page 42)


Vision

Over 3 iterations, your goal is to build a local file search system¹ that:

- Indexes files on your device, including documents, media, and binaries.
- Uses filename, content, and metadata for searching.
- Feels fast and responsive, with results appearing as you type.

In this iteration, you will lay the foundation of your search infrastructure, which will enable you to implement ever more advanced information retrieval features. You crack your knuckles, prepare a tasty dose of dehydrated water, and get to work.

Outline

1. Crawl your local content, filter out unwanted data, and store it in some database. Designing your own indexing format is outside the scope of this class, so you'll be offloading that part to your preferred DBMS.
2. Your database schema will determine how you process your data at query time. Focus on the quality of your data, instead of trying to prematurely denormalize it for some imagined use case. If you favor NOSQL DBMSs and believe Relational DBMSs are obsolete, we strongly suggest you read [this paper](#)  .
3. When building a search engine, every bit of information is important, even if not immediately useful. Consider keeping all available metadata for future use cases (e.g., extensions, tags, timestamps).
4. Carefully consider what you return when the user types a query in your search bar. File previews are crucial for user satisfaction (e.g., return the first 3 lines of a document)².
5. Don't try to apply design patterns preemptively. First, strive for a deep understanding of the problem you're solving. As you develop the system and gain a clearer picture of its requirements, the applicability of design patterns will become self-evident.
6. For this iteration, focus on content search over textual files.

¹Google Desktop

²Contextual search snippets were one of the main innovations of Google Search back in 1998. Their competitors were mostly just showing the first hundred or more characters of the document, which often included a mix of HTML, JavaScript and other irrelevant text.

Search Engine: Iteration 1

Software Design 2025

March 7, 2025

Score	Criteria
5/10	(Passing - Basic Functionality): Code compiles and runs, but with significant limitations. File traversal may be incomplete. Text extraction is basic (e.g., only '.txt'). Database connection is established, but data insertion or retrieval might be broken or very limited. Basic SQL queries are attempted, but likely incorrect. Little to no error handling.
6/10	(Below Average): File traversal works recursively. Data is inserted into the database, but may have errors (e.g., incorrect data types, missing data). Basic single-word searches work using SQL queries, but multi-word searches are likely broken. Limited error handling.
7/10	(Average): File traversal is robust. Single-word and multi-word searches work using appropriate SQL queries (using the DBMS's full-text search features). Basic error handling.
8/10	(Good): All features from 7/10 are implemented correctly and efficiently. Good database schema design with appropriate data types and indexes. SQL queries are well-written and efficient. Good error handling, including database-related errors. Code is well-structured, following basic OOP principles. The indexer can be configured at runtime (e.g., file ignore rules, root directory).
9/10	(Excellent): The index builder keeps track of its progress and generates a report at the end, formatted as a text file.
10/10	(Outstanding): Adherence to good software engineering practices (commit messages, commit granularity, coding style, etc.).

 **Important: The deadline for this iteration is Week 5.**