A New Way to Store Knowledge

HTML | TXT | PDF

by Breck Yunits

May 21, 2024

All tabular knowledge can be stored in a single long plain text file.

The only syntax characters needed are spaces and newlines.

This has many advantages over existing binary storage formats.

Using the method below, a very long scroll could be made containing all tabular scientific knowledge in a computable form.

*

There are four concepts to understand:

- measures
- concepts
- measurements
- comments

Measures

First we create measures by writing parsers. The parser contains information about the measure.

The only required information for a measure is an id, such as `temperature`.

An example measure:

```
temperatureParser
```

Concepts and Measurements

Next we create concepts by writing measurements.

The only required measurement for a concept is an id. A line that starts with an id measurement is the start of a new concept.

A measurement is a single line of text with the measure id, a space, and then the measurement value.

Multiple sequential lines of measurements form a concept.

An example concept:

```
id Earth
temperature 14
```

Comments

Unlimited comments can be attached under any measurement using the indentation trick.

An example comment:

```
temperature 14
 > The global mean surface air temperature for that p
  - NASA
    https://earthobservatory.nasa.gov/world-of-change/
```

*

The Complete Example

Putting this all together, all tabular knowledge can be stored in a single plain text file using this pattern:

```
idParser
temperatureParser

id Earth
temperature 14
 > The global mean surface air temperature for that p
  - NASA
    https://earthobservatory.nasa.gov/world-of-change/
```

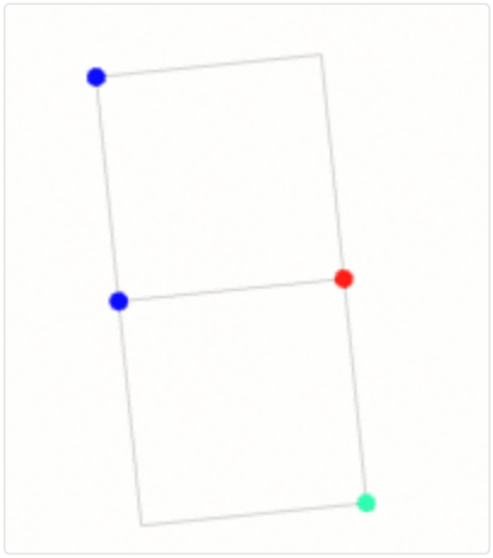
*

Once your knowledge is stored in this format, it is ready to be read—*and written*—by humans, traditional software, and artificial neural networks, to power understanding and decision making.

Edit history can be tracked by git.

*

A Visualization



Blue dots are measure ids. The first blue dot is a measure definition (aka a parser). The red dot is a measurement value. The blue-red pair is a measurement, as well as a concept. The cyan dot is a comment. View Source

*

Prior Art

Modern databases[1] were designed before git[2], fast filesystems[3], and the Tree Notation stack[4], all requirements of this system.

GNU Recutils[5] deserves credit as the closest precursor to our system. If Recutils were to adopt some designs from our system it would be capable of supporting larger databases.

*

Initial Implementation and Experimental Evidence

ScrollSets is the name of the first implementation of the system above. It is open source and dedicated to the public domain.

ScrollSets are used to power the open source website PLDB.io. PLDB currently has over 300 measures, over 4,000 concepts and over 150,000 measurements, contributed by over 100 people, dozens of software crawlers, and a couple of artificial neural networks.

If printed on a single scroll, the PLDB ScrollSet would be over one kilometer long.

*

Enhancements

- For pragmatic reasons, it is best to split your data into 1 file per concept and combine concept files at runtime.
- The utility and joy of this system improves as your parser language improves. The parser language powering ScrollSets is currently called Grammar, and is largely influenced by ANTLR[6] and Racket[7].
- It is *very* helpful to have a `sortIndex` attribute on your measures to automatically prioritize[8] the measurements in your source and output files. The impact of this simple enhancement hints at interesting signs of dense information packing achieved by this method, which may have implications for the weights and training of artificial neural networks.
- Computed measures are measurements not stored statically, but derived at runtime from other measurements. They are very useful and easy to add with a few lines of parser code.
- You generally always want to add a type attribute to your measures, which gives you error checking, among other things.
- Measures can be nested. This means it is best to be restrictive in what characters are allowed in measure ids to integrate with a broad set of software tools. For example, you can nest a `minParser` under `temperatureParser` to generate a `temperature_min` column name in a generated TSV.
- It is useful to have measures whose values are foreign keys, such as a list of `ids`.

*

Conclusion

Measurements loosely map to nucleotides; concepts to genes; parsers to ribosomes.

This system might also have broad use.

*

Citations

[1] SQL: Donald D. Chamberlin and Raymond F. Boyce

[2] Git: Linus Torvalds, Junio Hamano, et al

[3] M1: Apple

[4] Tree Notation: Breck Yunits et al

[5] GNU Recutils: Jose E. Marchesi

- Recutils and our system have debatable syntactic differences, but our system solves a few clear problems described in the Recutils docs:
  - "difficult to manage hierarchies". Hierarchies are painless in our system through nested parsers, parser inheritance, parser mixins, and nested measurements.
  - "tedious to manually encode...several lines". No encoding is needed in our system thanks to the indentation trick.
  - In Recutils comments are "completely ignored by processing tools and can only be seen by looking at the recfile itself". Our system supports first class comments which are bound to measurements using the indentation trick.
  - "It is difficult to manually maintain the integrity of data stored in the data base." In our system advances parsers provides unlimited capabilities for maintaining data integrity.

[6] ANTLR: Terence Parr et al

[7] Racket: Matthias Felleisen, Matthew Flatt, Robert Bruce Findler, Shriram Krishnamurthi, et al.

[8] Prettier: James Long et al

*

Thanks

Thank you to everyone who helped me evolve this idea into its simplest form, including but not limited to, A, Alex, Andy, Ben, Brian, C, Culi, Dan, G, Greg, Jack, Jeff, John, L, Liam, Hari, Hassam, Jose, Matthieu, Ned, Nick, Nikolai, Pavel, Steph, Tom, Zach, Zohaib.

⁂