{ bareclad™ }

The Pain and Pleasure of Building a Database Engine

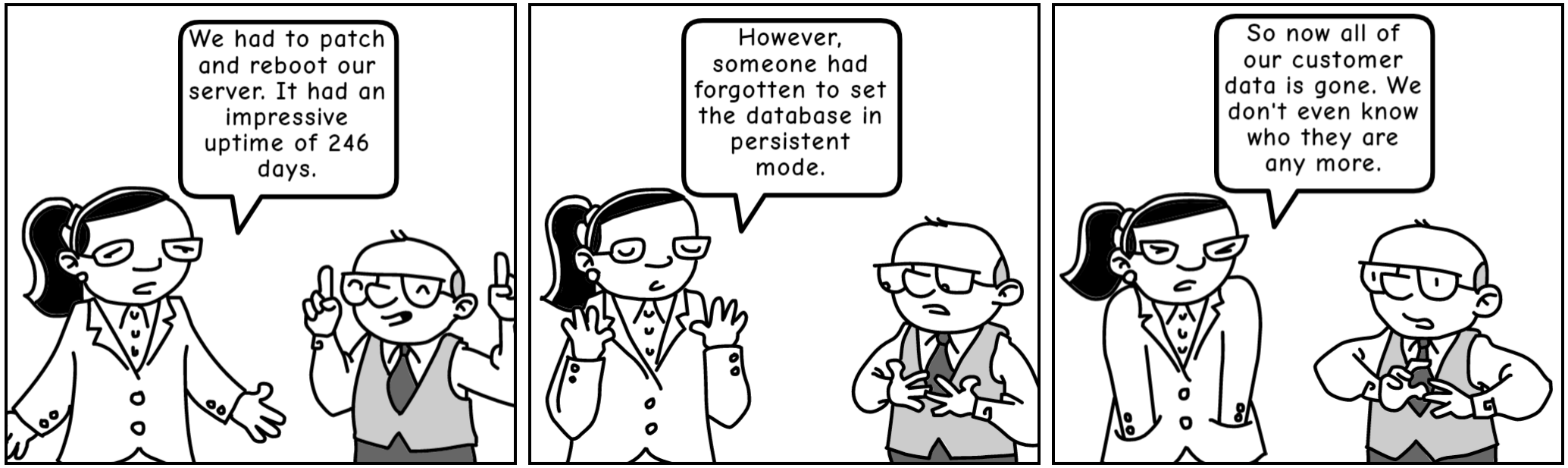
LARS RÖNNBÄCK

The Knowledge Gap 2022

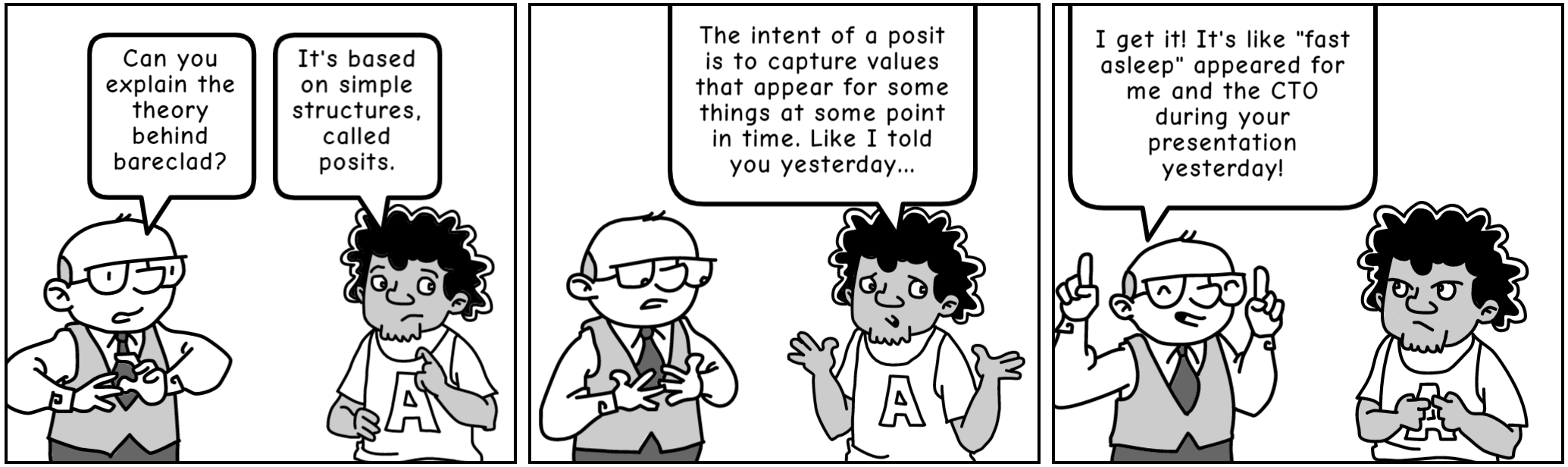
Comic strips made in https://makebeliefscomix.com – Bill Zimmerman

up ? to change


What have you been up to?
Well, I decided to create a database engine completely from scratch.
Wow, that must take a whole team of experts?
Umm... It's just me right now.
I don't think he knows what he's getting himself into.
I suppose I'll figure it out along the way...


We had to patch and reboot our server. It had an impressive uptime of 246 days.
However, someone had forgotten to set the database in persistent mode.
So now all of our customer data is gone. We don't even know who they are any more.


Java, C++, C, Scala, Go, Haskell, Swift, .NET, Rust, Node.js, WASM, Python, Ruby, SQL, Perl, Pascal, Basic, Kotlin, Julia, Lisp, Elm...
I need the BEST possible language.
But how do you decide? How do you find good and reliable snippets? How do you know if it's "alive"?
I know, I'll use the CHOAS metric:
Clueless Has Opinion About Subject
...and Rust is the winner!

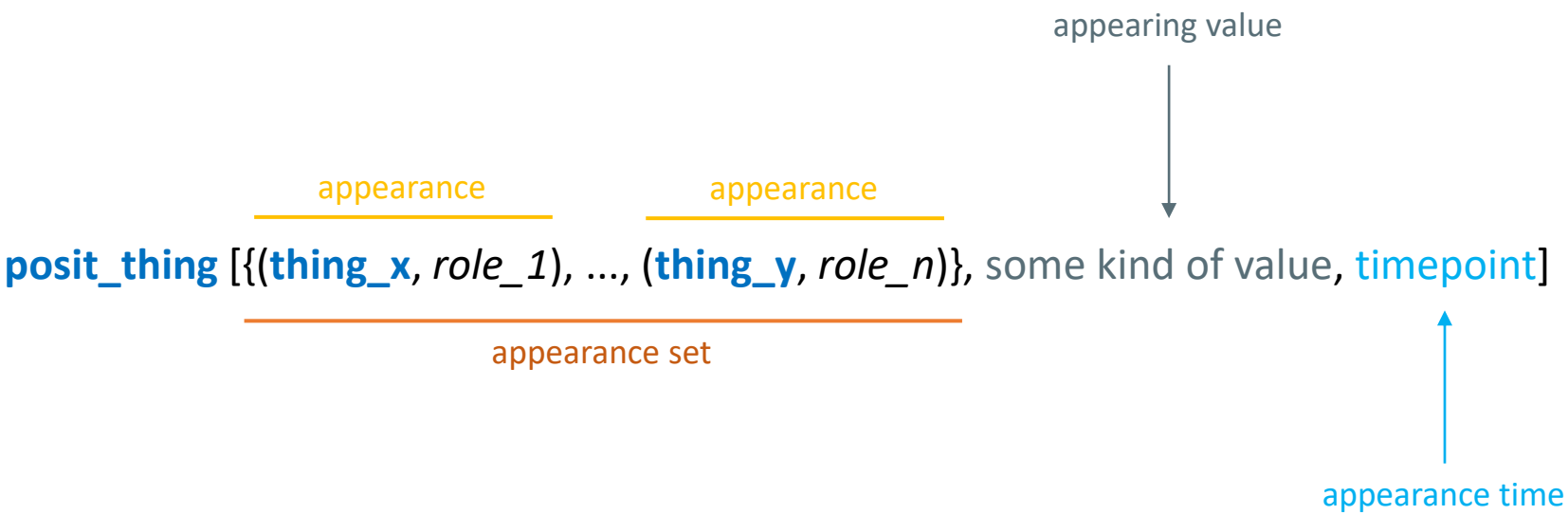

Can you explain the theory behind bareclad?
It's based on simple structures, called posits.
A
The intent of a posit is to capture values that appear for some things at some point in time. Like I told you yesterday...
A
I get it! It's like "fast asleep" appeared for me and the CTO during your presentation yesterday!

posit_thing [{(**thing_x**, *role_1*), ..., (**thing_y**, *role_n*)}, some kind of value, timepoint]

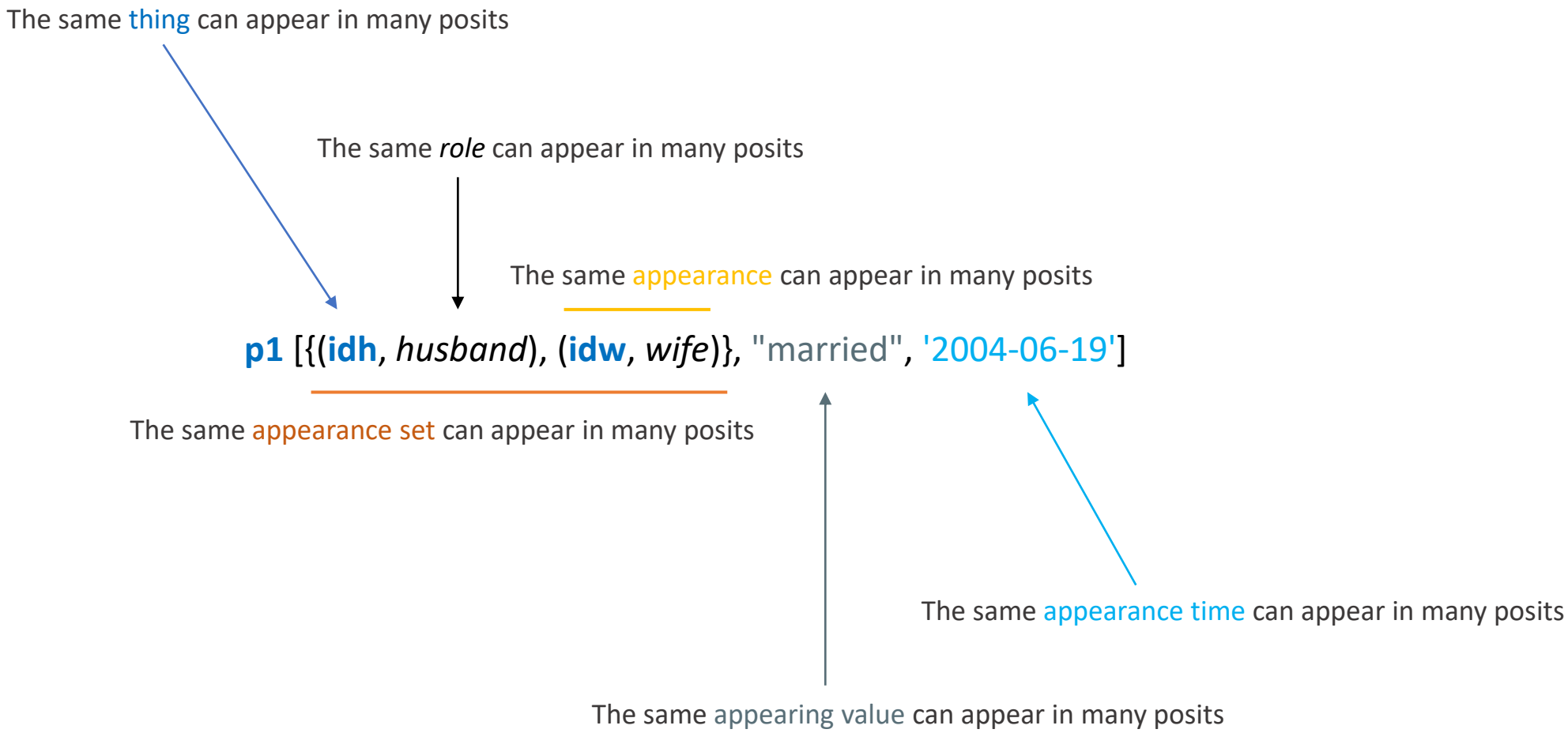
posit – an atomic data piece

---

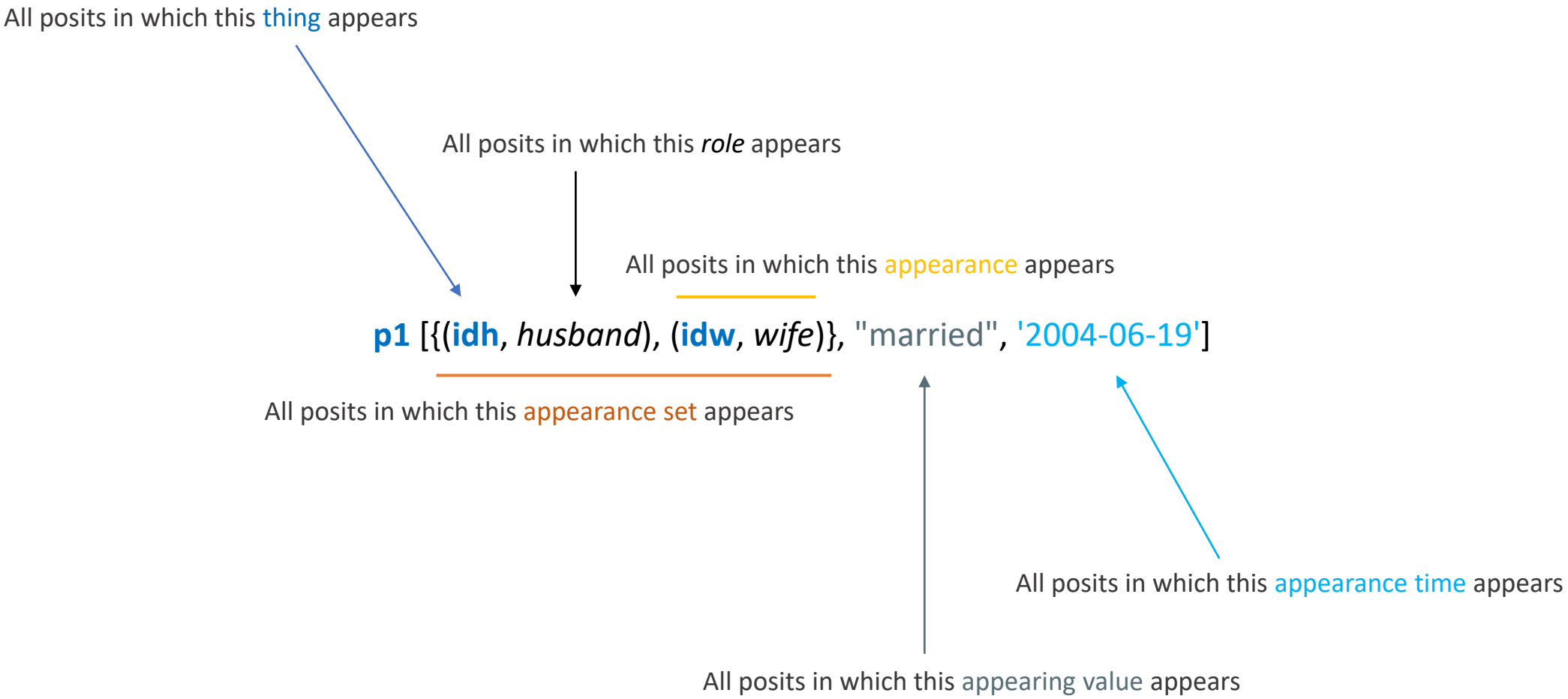
posit_thing [{(**thing_x**, *role_1*), ..., (**thing_y**, *role_n*)}, some kind of value, timepoint]


appearing value
appearance
appearance
posit_thing [{(thing_x, role_1), ..., (thing_y, role_n)}, some kind of value, timepoint]
appearance set
appearance time

p1 [{(**idh**, *husband*), (**idw**, *wife*)}, "married", '2004-06-19']


The same thing can appear in many posits
The same role can appear in many posits
The same appearance can appear in many posits
p1 [{(idh, husband), (idw, wife)}, "married", '2004-06-19']
The same appearance set can appear in many posits
The same appearance time can appear in many posits
The same appearing value can appear in many posits

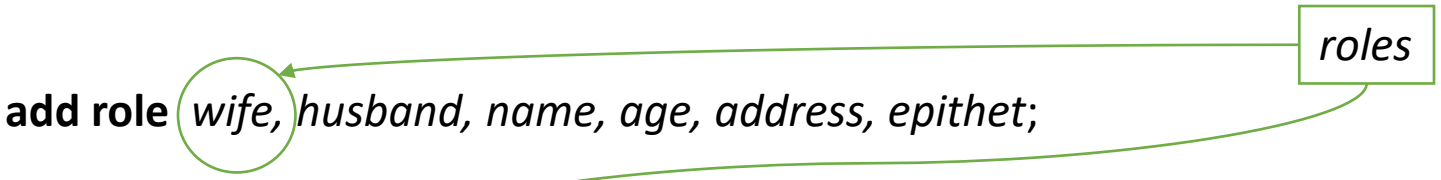
MANY WAYS TO SEARCH!



Traqula

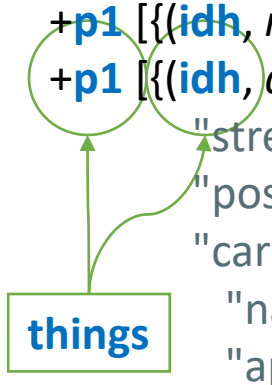

What do you mean, it's not going to understand SQL?
Well, it turns out that there are a lot of things you cannot do in SQL.
But the query language is not completely unlike SQL and it's got a bloody good name, Traqula.

```
add role  wife, husband, name, age, address, epithet;

add posit
  +p1 [{(+idw, wife), (+idh, husband)}, "married", '2004-06-19'],
  +p1 [{(idh, name)}, "Lars Samuelsson",      '1972-08-20'], /* A name I used to go by */
  +p1 [{(idh, address)}, {
            "street": "The Road of Karlberg",
            "postal code": 11335,
            "care_of": {
              "name": "Lars Roenbaeck",
              "apartment": 1501
            }
          }, '2007-09-01'],
  +p1 [{(idh, age)}, 3.14159265358979323846264338327950288419716939937510582097494459, '2003-02-20'],
  +p2 [{(idh, epithet)}, "/* liker of comments */, ], and  ""spaces""", @BOT];
```

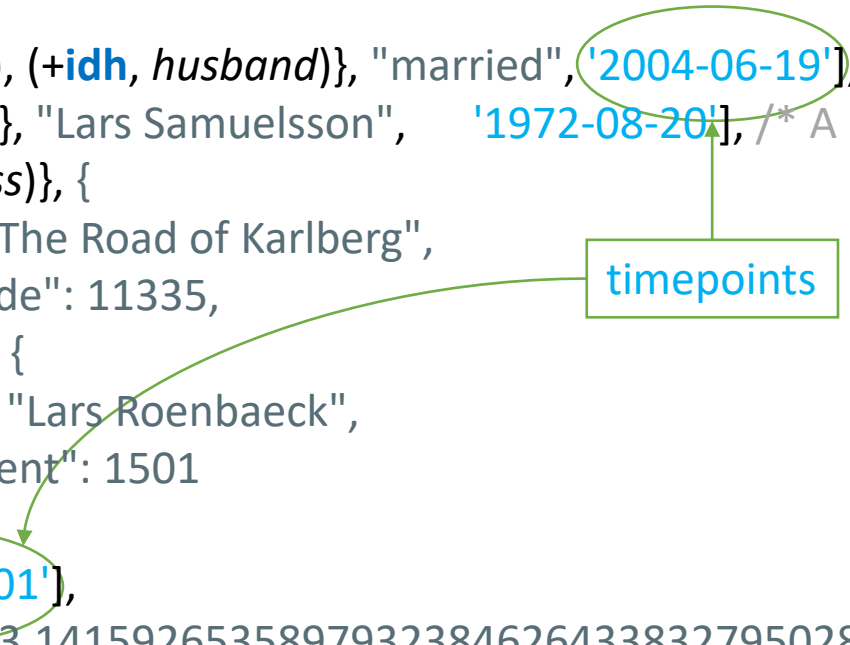



```
add role wife, husband, name, age, address, epithet;

add posit
  +p1 [{(+idw, wife), (+idh, husband)}, "married", '2004-06-19'],
  +p1 [{(idh, name)}, "Lars Samuelsson",     '1972-08-20'], /* A name I used to go by */
  +p1 [{(idh, address)}, {
          "street": "The Road of Karlberg",
          "postal code": 11335,
          "care_of": {
            "name": "Lars Roenbaeck",
            "apartment": 1501
          }
        }, '2007-09-01'],
  +p1 [{(idh, age)}, 3.14159265358979323846264338327950288419716939937510582097494459, '2003-02-20'],
  +p2 [{(idh, epithet)}, "/* liker of comments */, ], and  ""spaces""", @BOT];
```

```
add role  wife, husband, name, age, address, epithet;

add posit
  +p1 [{(+idw, wife), (+idh, husband)}, "married", '2004-06-19'],
  +p1 [{(idh, name)}, "Lars Samuelsson",      '1972-08-20'], /* A name I used to go by */
  +p1 [{(idh, address)}, {
        "street": "The Road of Karlberg",
        "postal code": 11335,
        "care_of": {
          "name": "Lars Roenbaeck",
          "apartment": 1501
        }
      }, '2007-09-01'],
  +p1 [{(idh, age)}, 3.14159265358979323846264338327950288419716939937510582097494459, '2003-02-20'],
  +p2 [{(idh, epithet)}, "/* liker of comments */, ], and  ""spaces""", @BOT];
```

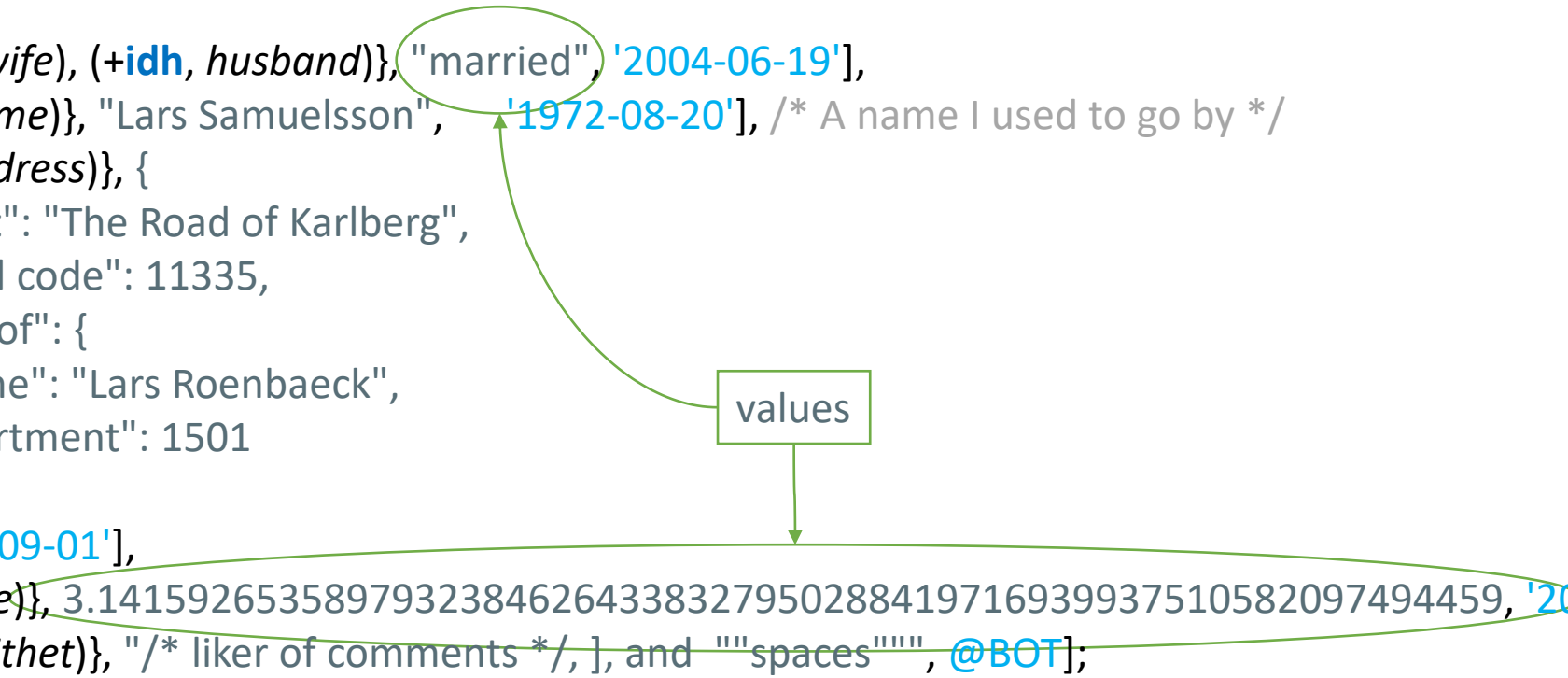


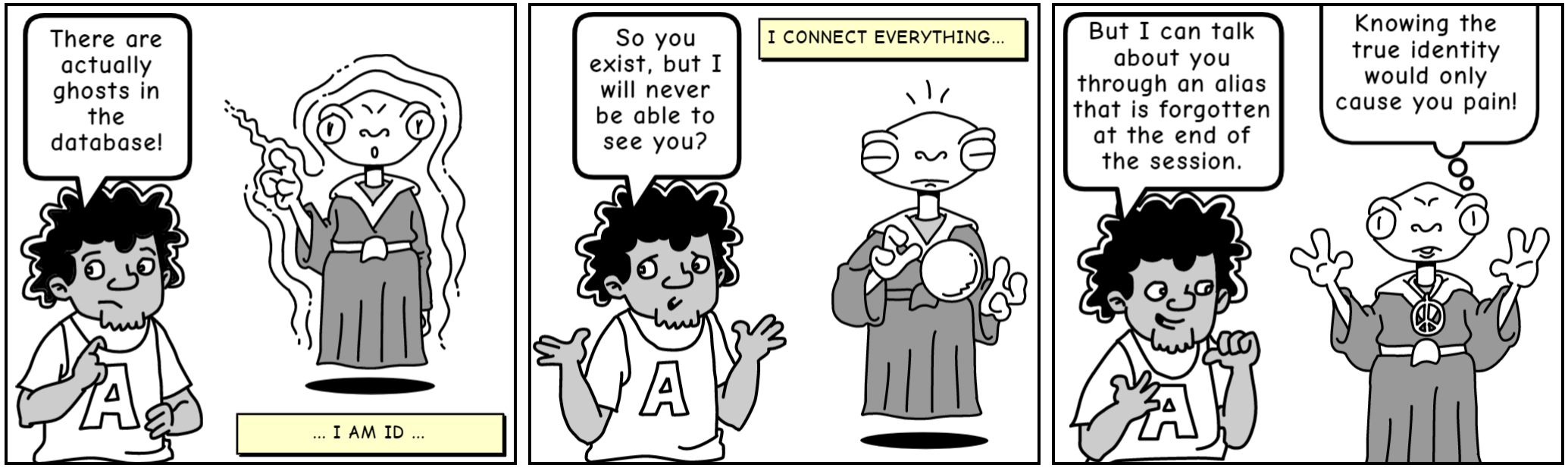
```
add role  wife, husband, name, age, address, epithet;

add posit
  +p1 [{(+idw, wife), (+idh, husband)}, "married", '2004-06-19'],
  +p1 [{(idh, name)}, "Lars Samuelsson",      '1972-08-20'], /* A name I used to go by */
  +p1 [{(idh, address)}, {
            "street": "The Road of Karlberg",
            "postal code": 11335,
            "care_of": {
              "name": "Lars Roenbaeck",
              "apartment": 1501
            }
          }, '2007-09-01'],
  +p1 [{(idh, age)}, 3.14159265358979323846264338327950288419716939937510582097494459, '2003-02-20'],
  +p2 [{(idh, epithet)}, "/* liker of comments */, ], and  ""spaces""", @BOT];
```



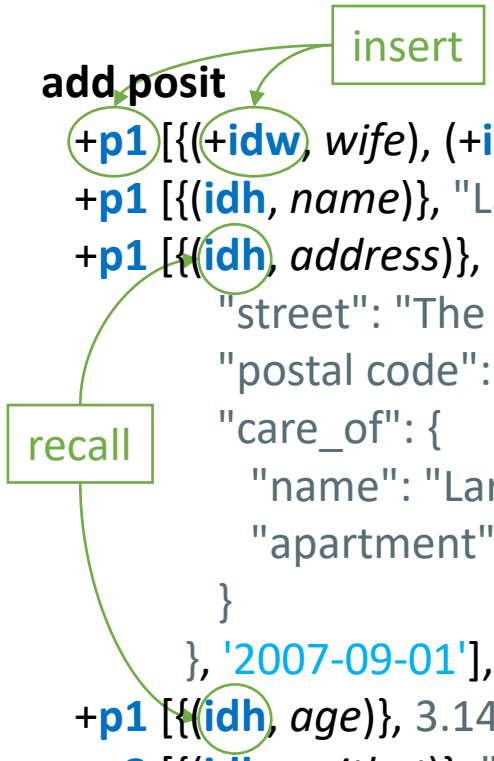

It looks like you decided to keep only two decimal places in the database?
I thought all prices were given like that...
Why do we have to care about what a price looks like today and what it may look like in the future? Can't the database just pick whatever fits best given how it looks?
If it looks like a dog, barks like a dog, and smells like a dog...
Look-alike datatypes might not be such a bad idea! When do we get that?

```
add role  wife, husband, name, age, address, epithet;

add posit
  +p1 [{(+idw, wife), (+idh, husband)}, "married", '2004-06-19'],
  +p1 [{(idh, name)}, "Lars Samuelsson",     '1972-08-20'], /* A name I used to go by */
  +p1 [{(idh, address)}, {
            "street": "The Road of Karlberg",
            "postal code": 11335,
            "care_of": {
              "name": "Lars Roenbaeck",
              "apartment": 1501
            }
          }, '2007-09-01'],
  +p1 [{(idh, age)}, 3.14159265358979323846264338327950288419716939937510582097494459, '2003-02-20'],
  +p2 [{(idh, epithet)}, "/* liker of comments */, ], and  ""spaces""", @BOT];
```




There are actually ghosts in the database!
... I AM ID ...
So you exist, but I will never be able to see you?
I CONNECT EVERYTHING...
But I can talk about you through an alias that is forgotten at the end of the session.
Knowing the true identity would only cause you pain!

```
add role  wife, husband, name, age, address, epithet;

add posit
  +p1 [{(+idw, wife), (+idh, husband)}, "married", '2004-06-19'],
  +p1 [{(idh, name)}, "Lars Samuelsson",      '1972-08-20'], /* A name I used to go by */
  +p1 [{(idh, address)}, {
          "street": "The Road of Karlberg",
          "postal code": 11335,
          "care_of": {
            "name": "Lars Roenbaeck",
            "apartment": 1501
          }
        }, '2007-09-01'],
  +p1 [{(idh, age)}, 3.14159265358979323846264338327950288419716939937510582097494459, '2003-02-20'],
  +p2 [{(idh, epithet)}, "/* liker of comments */, ], and  ""spaces""", @BOT];
```



```
add role  birth date;

add posit
   [{(idw, name)}, "Anneli",    '1972-02-13'],
   [{(idw, age)}, 30, '2002-02-13'],
   [{(idw, birth date)}, '1972-02-13', '1972-02-13'];

add posit +p1 [{(idh, name)}, "Lars Roenbaeck", '2004-07-01'];
add posit
   [{(p1, posit), (idh, ascertains)},   100%, @NOW],
   [{(p2, posit), (idh, ascertains)},  -100%, @NOW];


add posit
   [{(+c, name)}, "Person", @NOW],
   [{(idw, thing), (c, class)}, "belongs", '1972-02-13'];
```

```
add role  birth date;

add posit
   [{(idw, name)}, "Anneli",    '1972-02-13'],
   [{(idw, age)}, 30, '2002-02-13'],
   [{(idw, birth date)}, '1972-02-13', '1972-02-13'];

add posit +p1 [{(idh, name)}, "Lars Roenbaeck", '2004-07-01'];
add posit
   [{(p1, posit), (idh, ascertains)},   100%, @NOW],
   [{(p2, posit), (idh, ascertains)},  -100%, @NOW];


add posit
   [{(+c, name)}, "Person", @NOW],
   [{(idw, thing), (c, class)}, "belongs", '1972-02-13'];
```

p1 holds five posit things here

so this expands to five posits

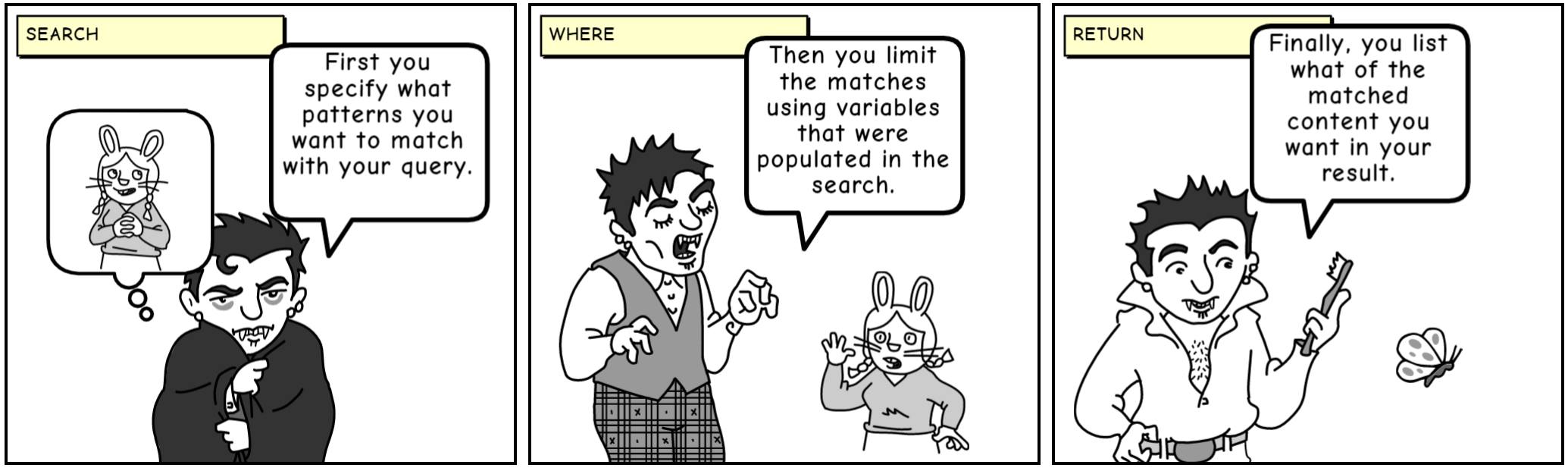
```
add role  birth date;

add posit
   [{(idw, name)}, "Anneli",    '1972-02-13'],
   [{(idw, age)}, 30, '2002-02-13'],
   [{(idw, birth date)}, '1972-02-13', '1972-02-13'];

add posit +p1 [{(idh, name)}, "Lars Roenbaeck", '2004-07-01'];
add posit
   [{(p1, posit), (idh, ascertains)},   100%, @NOW],
   [{(p2, posit), (idh, ascertains)},  -100%, @NOW];

add posit
   [{(+c, name)}, "Person", @NOW],
   [{(idw, thing), (c, class)}, "belongs", '1972-02-13'];
```

assertions (metadata)

reserved roles

classification (peridata, i.e. peripheral data)

reserved roles


SEARCH
First you specify what patterns you want to match with your query.
WHERE
Then you limit the matches using variables that were populated in the search.
RETURN
Finally, you list what of the matched content you want in your result.

```
search
   [{(idw, wife), (*, husband)}, "married", t] as of @NOW,
   [{(idw, name)}, w, *] as of t
where
   t >= '2001-01-01'
aggregate
   for w { number_of_wives_with_name = count(idw) }
return
   w,
   number_of_wives_with_name;
```

Peril	bareclad	relational database	What do you mean by that?
Boundaries	🤝	🤝	Where a thing begins and ends.
Vocabulary	🤝	🤝	What words we use to talk about things.
Classification	✓	🤝	How we categorize things that have similarities.
Intrinsics	✓	🤝	How values are represented (data types).
Uncertainty	✓	🤝	How certain we are of the statements we have.
Imprecision	🕘	✓	Data types that can store imprecise values.
Identification	🕘	❌	Search criteria for recollecting a thing.
Evolution	✓	❌	Everything can change over time.
Fallibility	✓	❌	Corrections may need to be made.
Negation	✓	❌	Talking about the opposite of a statement.
Relativity	✓	❌	Conflicting opinions are not unheard of.
Evaluation	⚠	⚠	A value is subjectively determined on the way in.
Interpolation	⚠	⚠	A value is subjectively experienced on its way out.
Insufficient context	⚠	⚠	A situation can never be captured in its entirety.

🤝 must be universally agreed upon
✓ managed by the database engine
❌ not managed by the database engine
⚠ cannot be managed by a database engine
🕘 upcoming feature

Roenbaeck / bareclad

Public

Sponsor | Pin | Unwatch 4 | Fork 0 | Star 7

Code | Issues 1 | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 3 branches | 0 tags | Go to file | Add file | Code

Roenbaeck Added mem check example — 72892f1 — 3 hours ago — 171 commits

.github	Create FUNDING.yml	2 years ago
.vscode	Added future dependencies and new VSCode setup	6 months ago
benches	All (relevant) set operations implemented	4 months ago
sql	"Smart" Time type replaces TimeType trait	4 months ago
src	Added mem check example	3 hours ago
traqula	Goodbye dollar signs	4 months ago
.gitignore	Removed SQLite database file (.gitignored now)	6 months ago
Cargo.lock	Traqula variables now have an extended memory	4 months ago
Cargo.toml	Traqula variables now have an extended memory	4 months ago
LICENSE	Dual license options	4 months ago
LICENSE.Apache-2.0	Dual license options	4 months ago
LICENSE.MIT	Add license file	4 months ago
README.md	Update README.md with license section	4 months ago
Traqula.png	A script language needs a script logo	5 months ago
Traqula.svg	A script language needs a script logo	5 months ago
bareclad.json	Great, but scary macro gone. Strict SQLite tables.	4 months ago
bareclad.png	Small variation in color	5 months ago
bareclad.svg	Small variation in color	5 months ago
bareclad_mini.png	Small variation in color	5 months ago
bareclad_mini.svg	Small variation in color	5 months ago
relational_model.png	Visualising implicit connection	5 months ago

README.md

{ bareclad™ }

Bareclad is a new type of database, based on transitional modeling. There are a number of posts on the topic that can be found in our blog on anchormodeling.com. The database engine will provide and combine features from relational, graph, columnar, and name-value pair databases. Modeling is bottom-up and respects the uncertain and subjecive nature of reality.

About

Based on transitional modeling, bareclad is a new type of database that provides features from relational, graph, columnar, and name-value pair databases.

www.bareclad.com

database | graph | relational | nvp | columnar

Readme
View license
7 stars
4 watching
0 forks

Releases

No releases published
Create a new release

Sponsor this project

ko-fi.com/bareclad

Packages

No packages published
Publish your first package

Contributors 2

Roenbaeck Lars Rönnbäck
Kestrer

Languages

Rust 100.0%


Hey, I heard you developed a new metric called CHOAS. Let's try it on "Data Warehouse".
A
Oh dear me, what am I looking at here?
Erm... I am afraid "Data Warehouse" is getting a terrible CHOAS score right now.
A
We'll toss the DW next sprint, and build a "Data Meshcastle" instead.
I know that look, our boss is going try on the emperor's new clothes again...