Twitter Thread by Stian Håklev ■ intertwingling Roam graphs

Stian Håklev ■ intertwingling Roam graphs
@houshuang

For people curious about the Roam API and confused by the syntax, or interested in why Conor went with Datomic/Datascript and not a traditional database, this older talk by Roam developer @mark_bastian is a great overview.

He gives great examples using Spiderman of how even modeling something fairly trivial in SQL is much more complex than in Datomic. But the real kicker is when you're trying to interrogate the data to find recursive relationships.

SQL

```
CREATE TABLE people (id INT, name varchar(MAX), gender varchar(1));
CREATE TABLE marriages (id INT, personID INT, spouseID INT);
CREATE TABLE aliases (id INT, personID INT, alias varchar(MAX));
CREATE TABLE children (id INT, parentID INT, childID INT);

-- Populate the people table
INSERT INTO "PUBLIC"."PEOPLE" ("ID", "NAME", "GENDER") VALUES (1, 'Peter Parker', 'M');
INSERT INTO "PUBLIC"."PEOPLE" ("ID", "NAME", "GENDER") VALUES (2, 'Richard Parker', 'M');
INSERT INTO "PUBLIC"."PEOPLE" ("ID", "NAME", "GENDER") VALUES (3, 'Mary Parker', 'F');
INSERT INTO "PUBLIC"."PEOPLE" ("ID", "NAME", "GENDER") VALUES (4, 'Ben Parker', 'M');
INSERT INTO "PUBLIC"."PEOPLE" ("ID", "NAME", "GENDER") VALUES (5, 'May Parker', 'F');
INSERT INTO "PUBLIC"."PEOPLE" ("ID", "GENDER") VALUES (6, 'M');

-- Populate the marriages table
INSERT INTO "PUBLIC"."MARRIAGES" ("ID", "PERSONID", "SPOUSEID") VALUES (1, 2, 3);
INSERT INTO "PUBLIC"."MARRIAGES" ("ID", "PERSONID", "SPOUSEID") VALUES (2, 3, 2);
INSERT INTO "PUBLIC"."MARRIAGES" ("ID", "PERSONID", "SPOUSEID") VALUES (3, 4, 5);
INSERT INTO "PUBLIC"."MARRIAGES" ("ID", "PERSONID", "SPOUSEID") VALUES (4, 5, 4);

-- Populate the aliases table
INSERT INTO "PUBLIC"."ALIASES" ("ID", "PERSONID", "ALIAS") VALUES (1, 1, 'Spider-Man');
INSERT INTO "PUBLIC"."ALIASES" ("ID", "PERSONID", "ALIAS") VALUES (2, 1, 'Spidey');

-- Populate the children table
INSERT INTO "PUBLIC"."CHILDREN" ("ID", "PARENTID", "CHILDID") VALUES (1, 2, 1);
INSERT INTO "PUBLIC"."CHILDREN" ("ID", "PARENTID", "CHILDID") VALUES (2, 3, 1);
INSERT INTO "PUBLIC"."CHILDREN" ("ID", "PARENTID", "CHILDID") VALUES (3, 6, 2);
INSERT INTO "PUBLIC"."CHILDREN" ("ID", "PARENTID", "CHILDID") VALUES (4, 6, 4);
```

Right now the Roam data model (at least that's exposed to developers) is just about pages, blocks, and children with tags. Already you can see how finding the page containing a block with a certain tag etc is useful.https://t.co/jWJnuKu1RG

Reverse Relationships

What if you want to relationships backwards? Say for instance you wanted a list of all pages in the database, with all blocks that referenced those pages? This is possible using the "**_**" character, as follows:

```
window.roamAlphaAPI.q('[
                        :find
                        (pull ?e
                              [:node/title {:block/_refs [:block/string] }])
                        :where
                        [?e :node/title]]')
```

One line version:

```
window.roamAlphaAPI.q('[ :find (pull ?e [:node/title {:block/_refs
[:block/string] }]) :where [?e :node/title]]')
```

This creates a "field" on each page called "_refs" which will contain an array of all blocks whose "refs" array contain this block. The title of the page and name of the linking block is pulled. This gives something like the following:

```
5: Array(1)
    0:
    title: "July 27th, 2020"
    _refs: Array(2)
        0: {string: "Week Starting [[July 27th, 2020]]"}
        1: {string: "Mon [[July 27th, 2020]]"}
```

But imagine when attribute relationships are fully represented

You should be able to model the entire Spiderman story in Roam.
Page title: Peter Parker
Child of:: [[Richard Parker]] [[Mary Parker]]
Aliases:: [[Spidey]]

etc, and do these kind of queries.

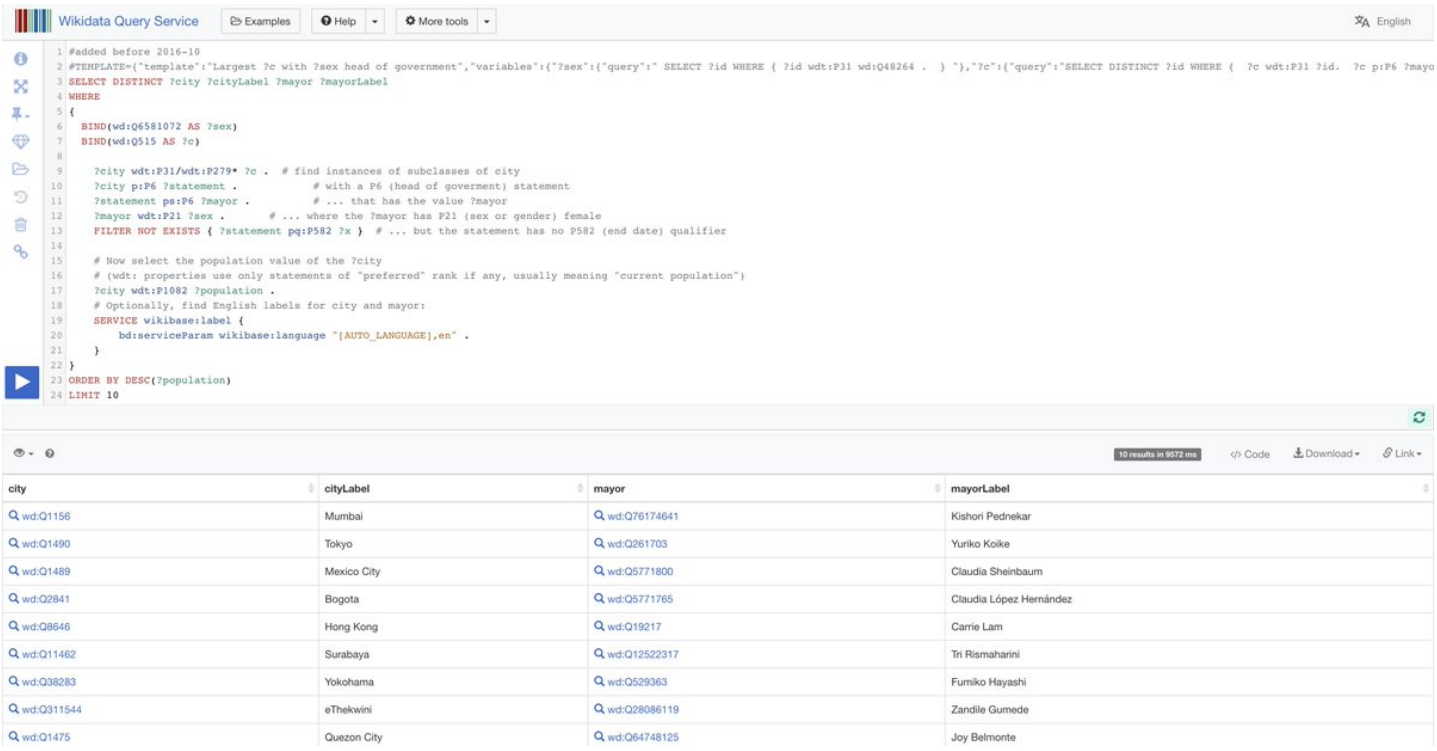
"Show me quotes about operational efficiency in books by authors who used to be in the military"
"Show me companies in Boise, Idaho, founded by women, whose evaluation is lower than 10X ARR"

Of course, this data can also be used as input to timelines, graphs, etc:

"Show me a graph of my sleep quality versus days in which I ate foods that had gluten in them or not" (where [[bread]] has a page with ingredients::).

One thing I'm curious about is how the Datalog system differs from Wikidata and SparQL, the modeling seems to be kind of similar - you have triplets of entities, like :Oslo :is-a-capital-if :Norway (where all three entities have an id), and you can do graph queries.

So you can ask "Largest city with female mayors", but you can also visualize data in all kinds of ways, like dimensions of elements, children of Genghis Khan, or lighthouses in Norway https://t.co/XvxmEVO9vB



What would it look like to integrate Wikidata with Roam in the future, being able to easily pull in and reference data about entities (cities, authors, scientific concepts)... And build our own Wikidata through inter-Roaming... As well as citations (https://t.co/aP7RSyaGl0) ...