Knowledge Graphs Everywhere

Knowledge graphs beyond the hype: Getting knowledge in and out of graphs and databases

What exactly are knowledge graphs, and what’s with all the hype about them? Learning to tell apart hype from reality, defining different types of graphs, and picking the right tools and database for you want to be like the Arx operative. Facebook, Google, and LinkedIn are the world.

Is The Enterprise Knowledge Graph Finally Going To Make All Data Usable?

Acquire Lattice Data over the weekend: The startup was working to transform the way businesses deal with paragraphs of text and other information that lives outside neatly structured databases. These engineers are uniquely prepared to assist Apple with building a next-generation internal knowledge graph to power Siri and its next generation of intelligent products and services.

Broadly speaking, the Lattice Data deal was an acquisition. Apple paid roughly $10 million for each of Lattice’s 20 engineers. This is generally considered to be fair market value.
What is a Knowledge Graph?
What is a Knowledge Graph?

The original “Knowledge Graph” (Google, 2012):
Many knowledge graphs, many technologies

There are a number of widely used publicly available knowledge graphs:

... and a variety of technologies for working with them:
What is special about Knowledge Graphs?

A Knowledge Graph is a data set that is:

- **structured** (in the form of a specific data structure)
- **normalised** (consisting of small units, such as vertices and edges)
- **connected** (defined by the – possibly distant – connections between objects)

Moreover, knowledge graphs are typically:

- **explicit** (created purposefully with an intended meaning)
- **declarative** (meaningful in itself, independent of a particular implementation or algorithm)
- **annotated** (enriched with contextual information to record additional details and meta-data)
- **non-hierarchical** (more than just a tree-structure)
- **large** (millions rather than hundreds of elements)
(Counter-)Examples

Typical knowledge graphs:

• Wikidata, Yago, Freebase, DBpedia (though hardly annotated)
• OpenStreetMap
• Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:
• Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
• WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
• Global data from schema.org: maybe not very connected
• Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections secondary

Primarily not knowledge graphs:
• Wikipedia: mostly unstructured text; not normalised; connections (links) important but secondary (similar: The Web)
• Relational database of company X: structured and possibly normalised, but no focus on connections (traditional RDBMS support connectivity queries only poorly)
(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)
Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

Debatable cases:
(Counter-)Examples

Typical knowledge graphs:
- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

Debatable cases:
- Facebook’s social graph: structured, normalised, connected, but not explicit
  (emerging from user interactions, without intended meaning beyond local relations)
Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

Debatable cases:

- Facebook’s social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
Typical knowledge graphs:
- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

Debatable cases:
- Facebook’s social graph: structured, normalised, connected, but not explicit
  (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
(Counter-)Examples

**Typical knowledge graphs:**
- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

**Debatable cases:**
- Facebook’s social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections secondary
(Counter-)Examples

**Typical knowledge graphs:**
- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

**Debatable cases:**
- Facebook’s social graph: structured, normalised, connected, but not explicit
  (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections secondary

**Primarily not knowledge graphs:**
(Counter-)Examples

**Typical knowledge graphs:**
- Wikidata, Yago 2, Freebase, DBpedia *(though hardly annotated)*
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori *(presumably; we can’t really know)*

**Debatable cases:**
- Facebook’s social graph: structured, normalised, connected, but not explicit
  (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content *(descriptions)*; explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores *(Lucene, MongoDB, etc.)*: structured, but not normalised; connections secondary

**Primarily not knowledge graphs:**
- Wikipedia: mostly unstructured text; not normalised; connections *(links)* important but secondary *(similar: The Web)*
(Counter-)Examples

Typical knowledge graphs:
- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

Debatable cases:
- Facebook’s social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections secondary

Primarily not knowledge graphs:
- Wikipedia: mostly unstructured text; not normalised; connections (links) important but secondary (similar: The Web)
- Relational database of company X: structured and possibly normalised, but no focus on connections (traditional RDBMS support connectivity queries only poorly)
Wikidata
A Free Knowledge Graph

**Wikidata [CACM 2014]**

- Wikipedia’s knowledge graph
- Free, community-built database
- Large graph
  (August 2019: >733M statements on >58M entities)
- Large, active community (>12,000 active editors in July 2019)
- Many applications

Freely available, relevant, and active knowledge graph
Many applications (1)

As of today, Wikidata content has been used in many ways.

Wikipedia & the Wikimedia community:
- Wikipedia inter-language links (see any Wikipedia page)
- Data displays in pages (auto-generated info boxes, article placeholders, result tables, . . .)
- Quality checks & edit-a-thons

External re-uses of data:
- Application-specific data-excerpts (e.g., Eurowings in-flight app)
- Data integration and quality control (e.g., Google checks own KG against Wikidata)
- Authority control & identity provider (VIAF, Open Streetmaps, DBLP, . . . link their content to Wikidata)
- Data-driven journalism (individual analyses as well as data-driven information portals)
Who is Grover Cleveland

OK. Check it out:

Grover Cleveland
22nd and 24th president of the United States

Stephen Grover Cleveland was an American politician and lawyer who was the 22nd and 24th President of the United States. He won the popular vote for three presidential elections – in 1884, 1888, and 1892 – and was one of two Democrats to be elected president during the era of Republican political domination dating from 1861 to 1933. He was also the first and to date only President in American history to serve two non-consecutive terms in office.

See More on Wikipedia

<table>
<thead>
<tr>
<th>Date of birth</th>
<th>March 18, 1837</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birthplace</td>
<td>Caldwell</td>
</tr>
<tr>
<td>Date of death</td>
<td>June 24, 1908</td>
</tr>
<tr>
<td>Deathplace</td>
<td>Princeton</td>
</tr>
</tbody>
</table>
Many applications (2)

As of today, Wikidata content has been used in many ways.

**In research:**
- Test data for KG-related algorithms
- Training data for machine-learning approaches
- Wikidata as a subject of study (social dynamics, internationality, biases, . . .)

**Uses by Wikidata community:**
- Software-supported error and vandalism detection
- Feature-based integration with other datasets
- Data-driven statistics as a measure of progress
What is Wikidata?

Wikidata is often described as “the free knowledge base that anyone can edit” or the “knowledge graph of Wikipedia”
What is Wikidata?

Wikidata is often described as “the free knowledge base that anyone can edit” or the “knowledge graph of Wikipedia”

It is useful to distinguish several of these aspects:

Wikidata is …

• … a Wikimedia project like Wikipedia and Wikimedia Commons; represented and supported by the Wikimedia Foundation (WMF)
• … a dataset that can be downloaded and freely used and distributed
• … a website through which the data can be viewed and modified
• … a community of volunteer editors that creates and controls all content
What is Wikidata?

Wikidata is often described as “the free knowledge base that anyone can edit” or the “knowledge graph of Wikipedia”

It is useful to distinguish several of these aspects:

**Wikidata is ...**

- ... **a Wikimedia project** like Wikipedia and Wikimedia Commons; represented and supported by the Wikimedia Foundation (WMF)
- ... **a dataset** that can be downloaded and freely used and distributed
- ... **a website** through which the data can be viewed and modified
- ... **a community** of volunteer editors that creates and controls all content

“And like all uses of the word ‘community,’ you were never quite sure what or who it was.” – Terry Pratchett (Jingo, 1997)
Tim Berners-Lee

British computer scientist

instance of

human

1 reference

employer

CERN

start time 1984
end time 1994
position held Fellow

0 references

+ add reference
Tim Berners-Lee (Q80)

Instance of

- human

Employer

- CERN
  - Start time: 1984
  - End time: 1994
  - Position held: Fellow

Award Received

- Queen Elizabeth Prize for Engineering
  - Point in time: 2013
  - Together with: Robert Kahn, Vint Cerf, Louis Pouzin, Marc Andreessen
Why does Wikidata use abstract (numeric) QIDs and PIDs rather than something more readable?
Wikidata’s IDs

Why does Wikidata use abstract (numeric) QIDs and PIDs rather than something more readable?

International
• Identifiers work for any language and cultural backgrounds

Stable
• Labels can change without IDs changing
• Multiple entities can have the same label
• IDs of deleted entities are never used again

Convenient
• Numeric IDs are quite short
• Uniform format is practical

How to find the ID of an item?
Main methods:
(1) Use the auto-completing search bar on wikidata.org
(2) Go to the item’s Wikipedia page and select “Wikidata item” from the sidebar

Several other projects have started to use Wikidata IDs for tagging and inter-linking.
Wikidata statements

Wikidata’s basic information units

• Built from Wikidata items (“CERN”, “Vint Cerf”), Wikidata properties (“award received”, “end time”), and data values (“2013”)

• Based on directed edges (“Tim Berners-Lee employer→ CERN”)

• Annotated with property-value pairs (“end time: 1994”)
  – same property can have multiple annotation values (“together with: Robert Kahn, Vint Cerf, …”)
  – same properties/values used in directed edges and annotations

• Properties have pre-defined datatypes

• Items and properties can be subjects/values in statements

• Multi-graph
**Elizabeth Taylor (Q34851)**

Elizabeth Rosemond Taylor | Liz Taylor | Dame Elizabeth Rosemond Taylor

British-American actress

**instance of:** Elizabeth Taylor is a(n) human

### Human relationships

#### Own statements

<table>
<thead>
<tr>
<th>Name</th>
<th>Relationship</th>
<th>end time</th>
<th>start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larry Fortensky</td>
<td>construction worker and seventh husband of Elizabeth Taylor</td>
<td>1996-10-31</td>
<td>1991-10-36</td>
</tr>
<tr>
<td>John Warner</td>
<td>Republican politician and Secretary of the Navy from the United States</td>
<td>1982-11-07</td>
<td>1976-12-04</td>
</tr>
<tr>
<td>Richard Burton</td>
<td>Welsh actor</td>
<td>1975-10-10</td>
<td>1976-07-29</td>
</tr>
<tr>
<td>Richard Burton</td>
<td>Welsh actor</td>
<td>1954-03-15</td>
<td>1974-06-26</td>
</tr>
<tr>
<td>Eddie Fisher</td>
<td>American entertainer and singer</td>
<td>1964-03-06</td>
<td>1959-05-12</td>
</tr>
<tr>
<td>Mike Todd</td>
<td>American theatre and film producer</td>
<td>1958-03-22</td>
<td>1957-02-02</td>
</tr>
<tr>
<td>Michael Wilding</td>
<td>English television and film actor</td>
<td>1957-01-30</td>
<td>1952-02-21</td>
</tr>
</tbody>
</table>

#### From related entities

**Links**

- Wikidata page
- Official website
- Wikipedia article
- Reasonator

**Identifiers**

- SFDb person ID: 75200
- Elonet person ID: 224907
- PORT person ID: 7869
- AllMovie artist ID: p70015
Representing Knowledge Graphs
Once upon a time, there was the World Wide Web:
- A huge network of many connections,
- Lacking meaning and data.

"Let's define the meaning of hyperlinks and add data values!"

{Resource Description Framework (RDF)}

Once upon a time, there were document databases:
- Huge collections of objects described by attributes,
- Lacking connections and relationships.

"Let's add a type of 'edge' object that can connect documents!"

{Property Graph}
Tales of two graphs

Once upon a time, there was … the World Wide Web

- A huge network of many connections,
- lacking meaning and data

“Let’s define the meaning of hyperlinks and add data values!”

→ Resource Description Framework (RDF)
Tales of two graphs

Once upon a time, there was … the World Wide Web

• A huge network of many connections,
• lacking meaning and data

“Let’s define the meaning of hyperlinks and add data values!”

⇒ Resource Description Framework (RDF)

Once upon a time, there were … document databases

• Huge collections of objects described by attributes,
• lacking connections and relationships

“Let’s add a type of ‘edge’ object that can connect documents!”

⇒ Property Graph
RDF

Conceived as a data exchange format for describing data about (Web) resources.

- Defined by a W3C standard (patent-free & freely accessible)
- Current version: RDF 1.1, 2014
- Roughly: directed, edge-labelled graphs

Example of such a graph:

- Dresden
- Melitta Bentz
- Coffee filter
- 1873-01-31
- born in
- population
- invention
- inventor
- birthdate
- 547,172
Encoding RDF Graphs

RDF uses IRIs as identifiers, and data literals for encoding values:

- Dresden
- Melitta Bentz
- 547,172
- Coffee filter
- 1873-01-31

Properties:
- born in
- invention
- inventor
- birthdate

Markus Krötzsch, EDBT Summer School 2019
Working with Knowledge Graphs
RDF uses IRIs as identifiers, and data literals for encoding values:
Encoding RDF Graphs

RDF uses **IRIs** as identifiers, and **data literals** for encoding values:

This graph can be encoded in by listing edges (“triples”), one per line:

```xml
```
Encoding RDF Graphs More Nicely

The popular **Turtle syntax** supports several simplifications to enhance readability:

```turtle
PREFIX ex: <http://example.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

ex:Melitta-Bentz rdfs:label "Melitta Bentz"@en ;
   ex:birthdate "1873-01-31"^^xsd:date ;
   ex:invention ex:coffee-filter ;
<https://www.dresden.de/#uri> ex:population 547172 .
```

(Shown here: abbreviation IRIs using PREFIX, continued triples with semi-colon, simplified number syntax – several further features exist)
Property Graph

Conceived as a data management paradigm for adding a graph view to object DBMS.

- Data model as represented by Apache Tinkerpop Java library
- Various interpretations (Java API, relational DB extension, RDF extension)
- Directed, labelled multi-graphs + attribute-value maps for nodes and edges

Example of such a graph:

- **id1**
  - **name**: Melitta Bentz
  - **birthdate**: female
  - **inventor**
    - **name**: coffee filter

- **id2**
  - **born**: 1873-01-31
  - **inventor**

- **id3**
  - **name**: Dresden
  - **population**: 547172

- **id4**
  - **inventor**

- **id5**
Property Graph: Specifics and Unspecifics

Edge and vertex labels
- arbitrary strings (unlike IRIs in RDF: no keys!)

Keys used in key-value maps
- arbitrary strings (subsidiary; no relation to graph vertices)

Values used in key-value maps
- Implementation-specific; for example:
  - OpenCypher: INTEGER, FLOAT, STRING, BOOLEAN (+list type)
  - SAP HANA Graph: SQL datatypes (+own extensions)
  - Amazon Neptune: RDF-like datatypes (+own extensions)
- No relation to graph: cannot refer to graph vertices

Limitations that hurt when working with knowledge graphs:
- “Inner” key-value maps cannot refer to vertices in “outer” graph
- Relationship types and keys cannot be described in graph
- Lack of standard; lack of exchange syntax (knowledge sharing difficult)
Property Graph: Specifics and Unspecifics

Edge and vertex labels
- arbitrary strings (unlike IRIs in RDF: no keys!)

Keys used in key-value maps
- arbitrary strings (subsidiary; no relation to graph vertices)

Values used in key-value maps
- Implementation-specific; for example:
  - OpenCypher: INTEGER, FLOAT, STRING, BOOLEAN (+list type)
  - SAP HANA Graph: SQL datatypes (+own extensions)
  - Amazon Neptune: RDF-like datatypes (+own extensions)
- No relation to graph: cannot refer to graph vertices

Limitations that hurt when working with knowledge graphs:
- “Inner” key-value maps cannot refer to vertices in “outer” graph
- Relationship types and keys cannot be described in graph
- Lack of standard; lack of exchange syntax (knowledge sharing difficult)
Wikidata, RDF, and SPARQL
Wikidata in RDF

Wikidata is internally stored in the document-centric form using a JSON format.

**Data is converted to RDF for several purposes:**

- Offering complete data dumps for external uses
- Providing entity-specific linked data exports via a Web API
- Importing data into Wikidata’s SPARQL query service
Wikidata in RDF

Wikidata is internally stored in the document-centric form using a JSON format.

Data is converted to RDF for several purposes:

- Offering complete data dumps for external uses
- Providing entity-specific linked data exports via a Web API
- Importing data into Wikidata’s SPARQL query service

Wikidata’s graph view has many commonalities with RDF:

- Based on directed, labelled, multi-graph
- Properties have own identity in graph
- Order and in-page context of statements does not matter
Wikidata in RDF

Wikidata is internally stored in the document-centric form using a JSON format.

Data is converted to RDF for several purposes:
- Offering complete data dumps for external uses
- Providing entity-specific linked data exports via a Web API
- Importing data into Wikidata’s SPARQL query service

Wikidata’s graph view has many commonalities with RDF:
- Based on directed, labelled, multi-graph
- Properties have own identity in graph
- Order and in-page context of statements does not matter

However, there are also some important differences:
- Wikidata statements can have annotations and references
- Wikidata property types do not correspond to XML Schema types
- Wikidata IDs are not immediately IRIs
Encoding statements in RDF (1)

Tim Berners-Lee

British computer scientist

award received
Queen Elizabeth Prize for Engineering
  point in time
  2013
  together with
  Robert Kahn
  Vint Cerf
  Louis Pouzin
  Marc Andreessen

Note: For prefix declarations, see https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format
Encoding statements in RDF (1)

Where to store the annotations?

Note: For prefix declarations, see
We can encode statements in the style of reification:

```
wd:Q80
Tim Berners-Lee

p:P166 award received
wds:Q80-...
(statement node)

ps:P166 award received
wd:Q4273323
Queen Elizabeth Prize...
```

```
wdref:30b...
reference node

prov:wasDerivedFrom

wdt:P166 award received
```
We can encode statements in the style of reification:
We can encode statements in the style of reification:

We can encode statements in the style of reification:
We can encode statements in the style of reification:

```
wd:Q80
Tim Berners-Lee
```

```
wd:Q4273323
Queen Elizabeth Prize...
```

```
wd:Q214129
Robert Kahn
```

```
wd:Q92743
Vint Cerf
```

```
"2013"^^xsd:gYear
```

```
p:P166 award received
```

```
pq:P585 point in time
```

```
pq:P1706 together with
```

```
prov:wasDerivedFrom
```

```
wdref:30b9...
```

```
wdref:30b9...
```

```
p:award received
```

```
ps:award received
```

```
wdt:award received
```

```
wd:statement node
```

```
wd:Q80...
```

```
wd:Q80-
```

```
wd:Q4273323
```

```
wd:Q214129
```

```
wd:Q92743
```

```
wdref:30b9...
```

```
prov:wasDerivedFrom
```

```
wdref:30b9...
```

```
p:award received
```

```
ps:award received
```

```
wdt:award received
```

```
wd:statement node
```

```
wd:Q80-
```

```
wd:Q4273323
```

```
wd:Q214129
```

```
wd:Q92743
```

```
wdref:30b9...
```

```
prov:wasDerivedFrom
```

```
wdref:30b9...
```

```
p:award received
```

```
ps:award received
```

```
wdt:award received
```

```
wd:statement node
```

```
wd:Q80-
```

```
wd:Q4273323
```
We can encode statements in the style of reification:

wd:Q80 Tim Berners-Lee

wdt:P166 award received

wd:Q4273323 Queen Elizabeth Prize...

p:P166 award received

ps:P166 award received

"2013"^^xsd:gYear

pq:P585 point in time

wd:Q214129 Robert Kahn

... wd:Q92743 Vint Cerf

pq:P1706 together with

prov:wasDerivedFrom

wdref:30b9... reference node

...
Finishing the RDF encoding

Statements in Wikidata:
- Constitute the largest part of the RDF data
- RDF-encoding introduces over 50K RDF properties

Encoding other parts of Wikidata:
- Labels, descriptions, aliases are encoded as RDF literals (with language), linked from subject with rdfs:label, schema:description, and skos:altLabel, respectively
- Sitelinks are encoded using property schema:about (from article page URL to Wikidata entity IRI)
Finishing the RDF encoding

Statements in Wikidata:

- Constitute the largest part of the RDF data
- RDF-encoding introduces over 50K RDF properties

Encoding other parts of Wikidata:

- Labels, descriptions, aliases are encoded as RDF literals (with language), linked from subject with rdfs:label, schema:description, and skos:altLabel, respectively
- Sitelinks are encoded using property schema:about (from article page URL to Wikidata entity IRI)

Available RDF data:

- Full dumps are generated weekly (currently >7.9B triples, 55GiB Turtle.gz)
  For download see https://dumps.wikimedia.org/wikidatawiki/entities/

- Generate smaller partial dumps (by Benno Fünfstück, experimental):
  https://tools.wmflabs.org/wdumps/

- Linked data exports are provided through content negotiation
  Alternatively, directly use data URLs like http://www.wikidata.org/wiki/Special:EntityData/Q80.nt
Querying with SPARQL
What are the ten largest cities with a female mayor?
What are the ten largest cities with a female mayor?

<table>
<thead>
<tr>
<th>cityLabel</th>
<th>mayorLabel</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokyo</td>
<td>Yuriko Koike</td>
<td>13929286</td>
</tr>
<tr>
<td>Mexico City</td>
<td>Claudia Sheinbaum</td>
<td>8918653</td>
</tr>
<tr>
<td>Hong Kong</td>
<td>Carrie Lam</td>
<td>7409800</td>
</tr>
<tr>
<td>Baghdad</td>
<td>Zekra Alwach</td>
<td>6960000</td>
</tr>
<tr>
<td>Surabaya</td>
<td>Tri Rismaharini</td>
<td>4975000</td>
</tr>
<tr>
<td>Yokohama</td>
<td>Fumiko Hayashi</td>
<td>3748482</td>
</tr>
<tr>
<td>Rome</td>
<td>Virginia Raggi</td>
<td>2873494</td>
</tr>
<tr>
<td>Taichung</td>
<td>Lu Shiow-yen</td>
<td>2803894</td>
</tr>
<tr>
<td>Chicago</td>
<td>Lori Lightfoot</td>
<td>2722389</td>
</tr>
<tr>
<td>Guayaquil</td>
<td>Cynthia Viteri</td>
<td>2698077</td>
</tr>
</tbody>
</table>
Where are people born who travel to space?

(colour-coded by gender)
<table>
<thead>
<tr>
<th>Date</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>33</td>
<td>22</td>
<td>18</td>
<td>26</td>
<td>28</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>26</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>27</td>
<td>21</td>
<td>31</td>
<td>23</td>
<td>28</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>25</td>
<td>33</td>
<td>25</td>
<td>26</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>23</td>
<td>32</td>
<td>18</td>
<td>19</td>
<td>17</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>28</td>
<td>32</td>
<td>20</td>
<td>24</td>
<td>33</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>22</td>
<td>25</td>
<td>16</td>
<td>22</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>28</td>
<td>19</td>
<td>25</td>
<td>22</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>25</td>
<td>29</td>
<td>29</td>
<td>27</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>19</td>
<td>14</td>
<td>25</td>
<td>25</td>
<td>29</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>34</td>
<td>28</td>
<td>23</td>
<td>22</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>41</td>
<td>33</td>
<td>27</td>
<td>30</td>
<td>20</td>
<td>20</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>35</td>
<td>26</td>
<td>29</td>
<td>21</td>
<td>25</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>23</td>
<td>27</td>
<td>23</td>
<td>22</td>
<td>28</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>18</td>
<td>32</td>
<td>33</td>
<td>24</td>
<td>26</td>
<td>17</td>
<td>15</td>
</tr>
</tbody>
</table>
Which 19th century paintings show the moon?
Which films co-star more than one future head of government?

<table>
<thead>
<tr>
<th>Star in the Dust</th>
<th>1956 film by Charles F. Haas</th>
<th>2</th>
<th>Clint Eastwood, mayor; George Wallace, Governor of Alabama</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Two Who Stole the Moon</td>
<td>1962 Polish film by Jan Batory</td>
<td>2</td>
<td>Jarosław Kaczyński, Prime Minister of Poland; Lech Kaczyński, Mayor of Warsaw</td>
</tr>
<tr>
<td>Ragasiya Police 115</td>
<td>1968 film by B. R. Panthulu</td>
<td>2</td>
<td>M. G. Ramachandran, Chief Minister of Tamil Nadu; Jayalalithaa, Chief Minister of Tamil Nadu</td>
</tr>
<tr>
<td>Québec : Duplessis et après…</td>
<td>documentary</td>
<td>2</td>
<td>Bernard Landry, Premier of Quebec; René Lévesque, Premier of Quebec</td>
</tr>
<tr>
<td>Q3541438</td>
<td>1994 film by Claude Lanzmann</td>
<td>2</td>
<td>Ariel Sharon, Prime Minister of Israel; Ehud Barak, Prime Minister of Israel</td>
</tr>
<tr>
<td>Batman &amp; Robin</td>
<td>1997 American superhero film based on the DC Comics character Batman</td>
<td>2</td>
<td>Arnold Schwarzenegger, Mr. Freeze / Governor of California; Jesse Ventura, Governor of Minnesota</td>
</tr>
</tbody>
</table>
SPARQL Basics

The **SPARQL query language** *(W3C, 2011)* is used to query RDF graphs.

- **Graph patterns** are RDF graphs with variables (in Turtle syntax)
- **Can be combined with various operators** *(UNION, MINUS, OPTIONAL)*
- **Filter expressions** express additional conditions

**Example:** On Wikidata, find women born after 1921 that have an article on English Wikipedia but no image, ordered from youngest to oldest:

```sparql
SELECT ?person ?born
WHERE
{
  ?person eg:instanceOf eg:human ;
  eg:gender eg:female ;
  eg:birthdate ?born .
  FILTER (YEAR(?born) >= 1921) # date after 1921
  ?wikipage schema:about ?person ; # article about ?person
    schema:isPartOf <https://en.wikipedia.org/> . # on Wikipedia En
  MINUS { ?person eg:image ?image }
}
LIMIT 100 ORDER BY DESC(?born)
```
The **SPARQL query language** (W3C, 2011) is used to query RDF graphs.

- **Graph patterns** are RDF graphs with variables (in Turtle syntax)
- Can be combined with various **operators** (UNION, MINUS, OPTIONAL)
- Filter expressions express additional conditions

**Example:** On Wikidata, find women born after 1921 that have an article on English Wikipedia but no image, ordered from youngest to oldest:

```sparql
SELECT ?person ?born WHERE {
  ?person wdt:P31 wd:Q5 ; # instance of human
  wdt:P21 wd:Q6581072 ; # gender: female
  wdt:P569 ?born . # date of birth: ?born

  FILTER (YEAR(?born) >= 1921) # date after 1921
  ?wikipage schema:about ?person ; # article about ?person
    schema:isPartOf <https://en.wikipedia.org/> . # on Wikipedia En

  MINUS { ?person wdt:P18 ?image } # minus person with image
}
LIMIT 100 ORDER BY DESC(?born)
```
Path expressions

SPARQL can express reachability queries along paths described by regular expressions.

**Example:** On Wikidata, find all kinds of rock music with a French label:

```
SELECT ?genre ?genreLabel
WHERE {
  ?genre wdt:P279+ wd:Q11399 ; # (indirect) subclass of: rock music
  rdfs:label ?genreLabel. FILTER ( LANG(?genreLabel)="fr" )
}
```

- Further path operators: * (zero or more), | (alternative), ^ (converse edge direction), / (concatenation), ...
- Can be combined and nested
- However: SPARQL cannot count or return paths
Other notable features

Grouping and aggregates
- Including count, max, min, string concatenation, ...

Variable assignments
- Example: `BIND (SUBSTR(?label, 0, 1) as ?initial)`

Subqueries
- Use whole SPARQL `SELECT` blocks inside patterns
SPARQL on Wikidata

Wikidata SPARQL Query Service (WDQS):

- Official query service since mid 2015
- User interface at https://query.wikidata.org/
  - Query editing support (auto-completion, suggestions, examples)
  - Support for many different result visualisations
  - With library of example queries that helps to learn SPARQL
- All the data (7.9B triples), live (latency<60s)
- Very liberal configuration:
  - 60sec timeout
  - No limit on result size
  - No limit on parallel queries, but CPU-time budget per client

- Extra SERVICEs in SPARQL (geo, Wikipedia API, labels, ...)

For details, see [ISWC 2018].
**Wikidata** is currently the largest and fastest-growing free knowledge graph.

There are two predominant ways of encoding knowledge graphs:

- **RDF**: W3C labelled graph standard; highly normalised; many datatypes; various DBMS (BlazeGraph, Virtuoso, Amazon Neptune, Stardog, ...)
- **Property Graph**: graph-structured object DB; distinct graph and object layer; diverse implementations (Neo4j, SAP Hana Graph, Amazon Neptune, ...)

Complex data (as in Wikidata) is encoded in RDF by reification.

**SPARQL** is a powerful RDF query language used for live queries on Wikidata.

*Wikidata is an exciting resource with many uses in research.*


WDQS  Wikidata Query Service https://query.wikidata.org/; see also SPARQL documentation links on that page

WDQS  Wikidata Query Service https://query.wikidata.org/; see also SPARQL documentation links on that page