Knowledge Graphs

Knowledge Engineering and Business Intelligence
SS24

MSc Computer Science

Camerino, 29/04/2024

Dr. Emanuele Laurenzi

slido

What are Knowledge Graphs in your opinion?

ⓘ Start presenting to display the poll results on this slide.

Knowledge Graph (KG) Definition

- A KG is a directed labeled graph in which domain-specific meaning are associated with nodes and edges.
- A node could represent any real-world entity,
  - for example, people, companies, computers,etc.
- An edge label captures the relationship of interest between the two nodes
  - for example, a friendship relationship between two people; a customer relationship between a company and person; or a network connection between two computers.
- Meaning of nodes and edges can be expressed in a:
  - Human interpretable language such as English -> knowledge is easily understood and verifiable by humans.
  - Machine interpretable language -> formal specification language such as first-order logic -> knowledge is computed and automated by machines.

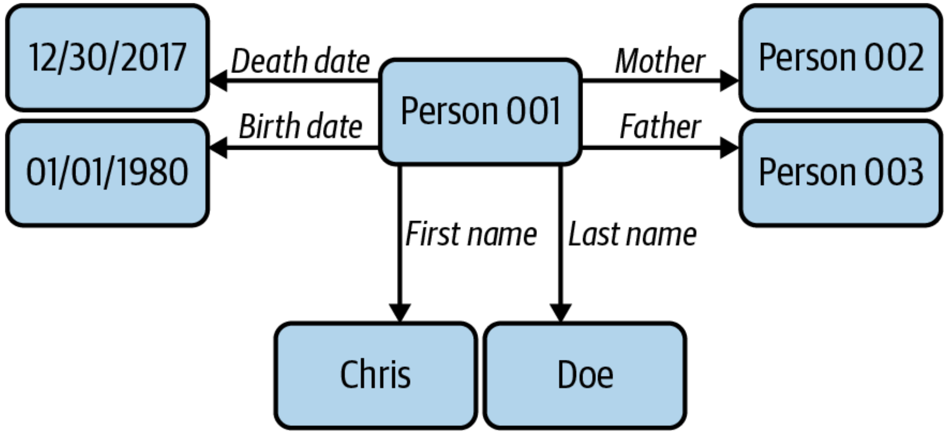
Source: https://onlinelibrary.wiley.com/doi/10.1002/aaai.12033

Example for human interpretability of a KG

– Who is the mother of whom?

ID	First name	Last name	Mother	Father	Birth date	Death date
001	Chris	Doe	002	003	01/01/1980	12/30/2017
002	Jane	Doe	104	124	03/03/1952	06/07/2015
003	John	Doe	343	322	04/06/1950	-

Tabular representation



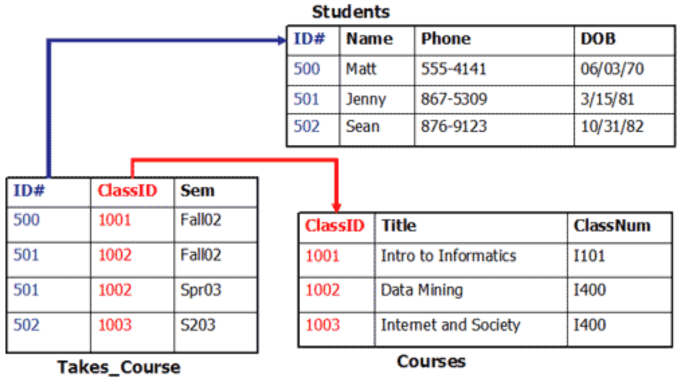
Graph representation

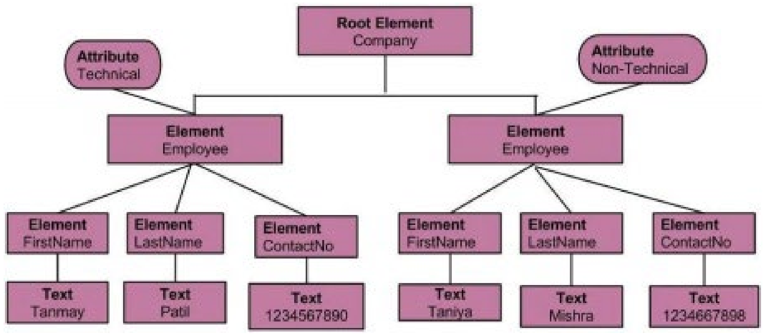
Higher flexibility in the way data is stored

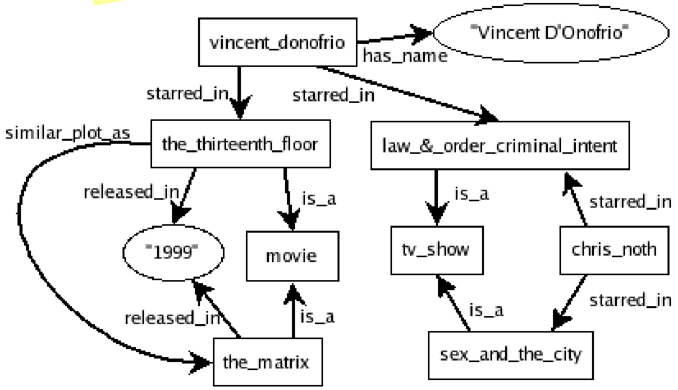
Tabular representation of data.

Tree-like representation of data.

Graph-like representation of data. Loops, self-referencing and multiple relations to the same node are allowed.



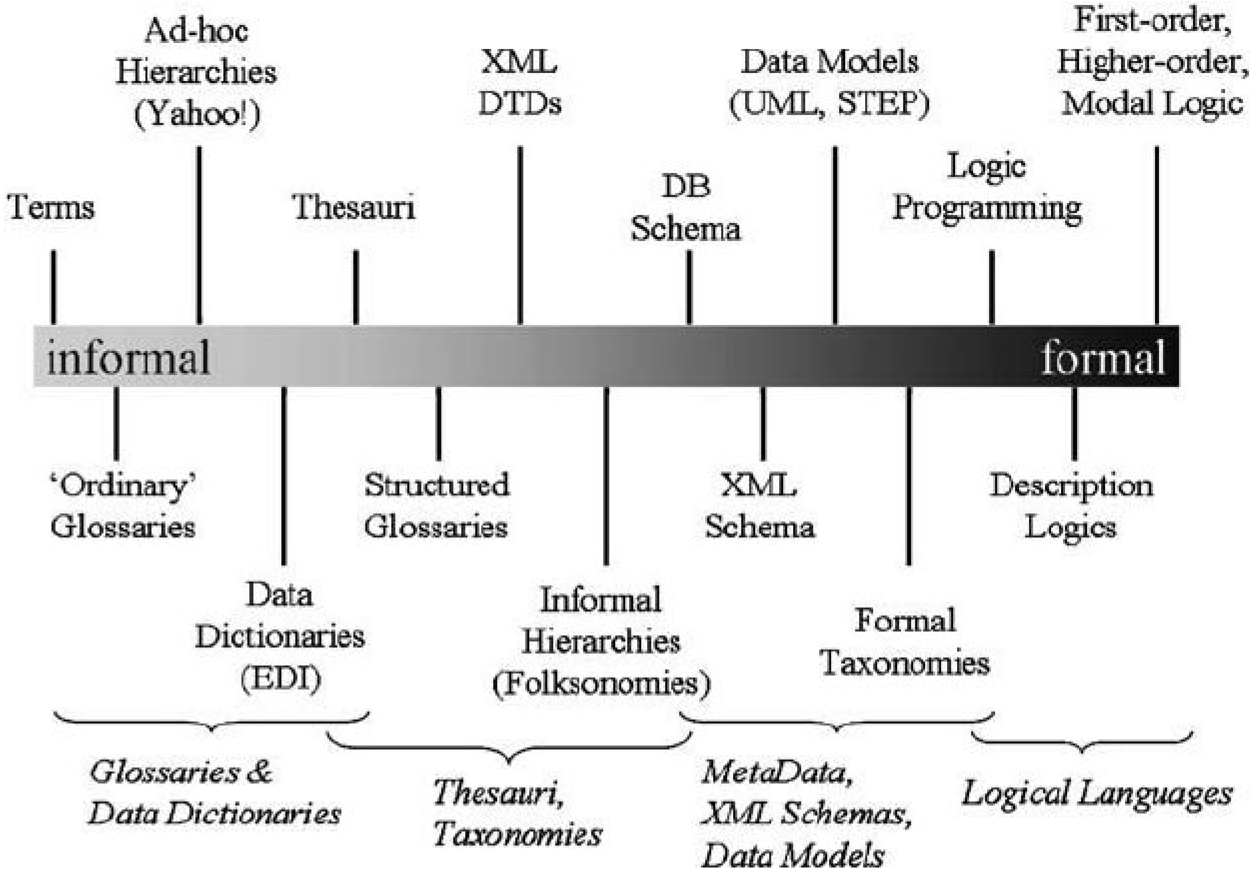




Tables *(SQL)* → *flexibility* → **Trees** *(XML, JSON)* → *flexibility* → **Graphs** *(RDF, LPG)*

Logical Foundations

- A knowledge graph can be regarded as a lightweight ontology.
- “An ontology is a **formal**, **explicit specification** of a shared **conceptualization**” (Studer at all. 1998).
- There exist several logic-based languages (i.e., knowledge representation formalisms) for the specification of ontologies, from less to more expressive ones.
- A knowledge graph is typically represented with a low expressive formalism.
- **The formalism makes the knowledge machine-interpretable** and enables automation, aka machine reasoning.



https://link.springer.com/chapter/10.1007/978-3-540-92673-3_0

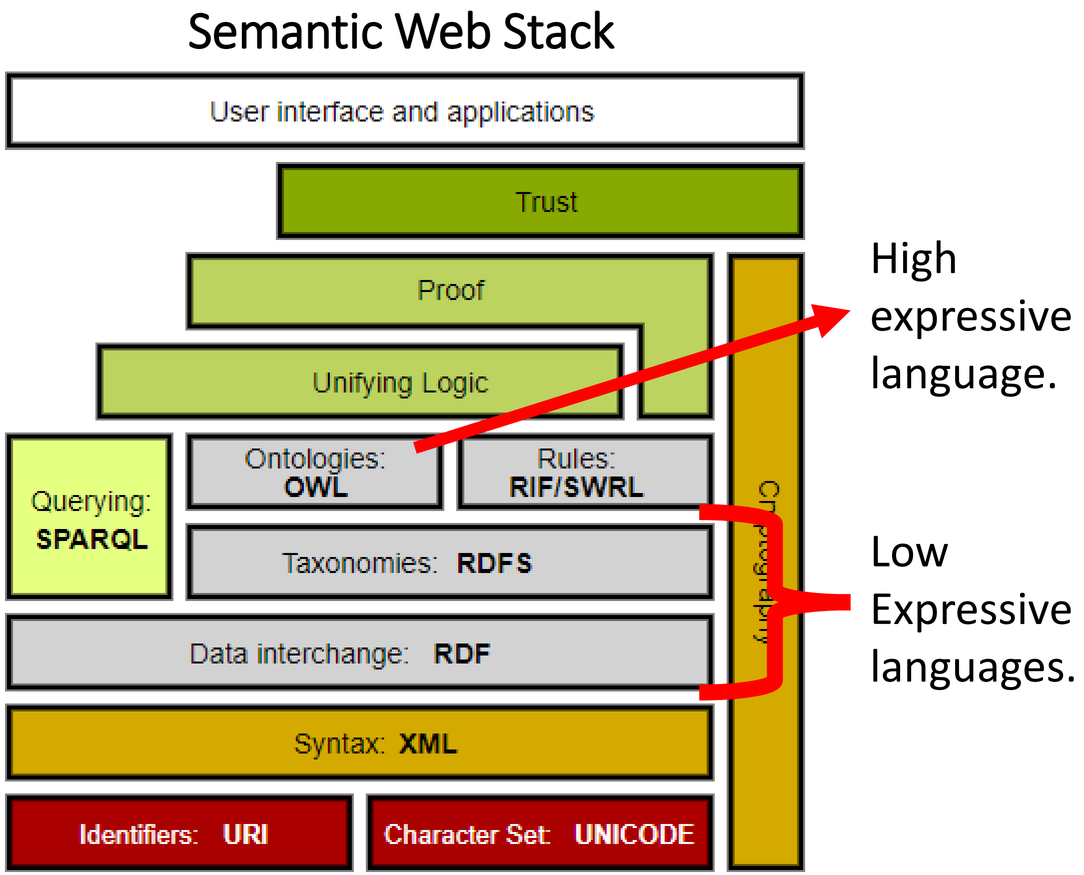
Source: https://arxiv.org/ftp/arxiv/papers/1401/1401.3858.pdf

The expressive formalism of a language

- The **expressive formalism** (also referred to expressive power, expressiveness or expressivity) of a language is the breadth of ideas that can be represented and communicated in that language.
- The more expressive a language is, the greater the variety and quantity of ideas it can be used to represent.
- More specifically,
  - The expressivity is defined by the (logical) elements (like ***and, or, not,*** **etc**) that a language provides; more elements imply more expressivity.
  - The higher the expressivity, the harder and the longer to answer decision problems.

W3C - World Wide Web Consortium (W3C)

- The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web.
- There are many more standards than what the Sematnic Web Stack shows.

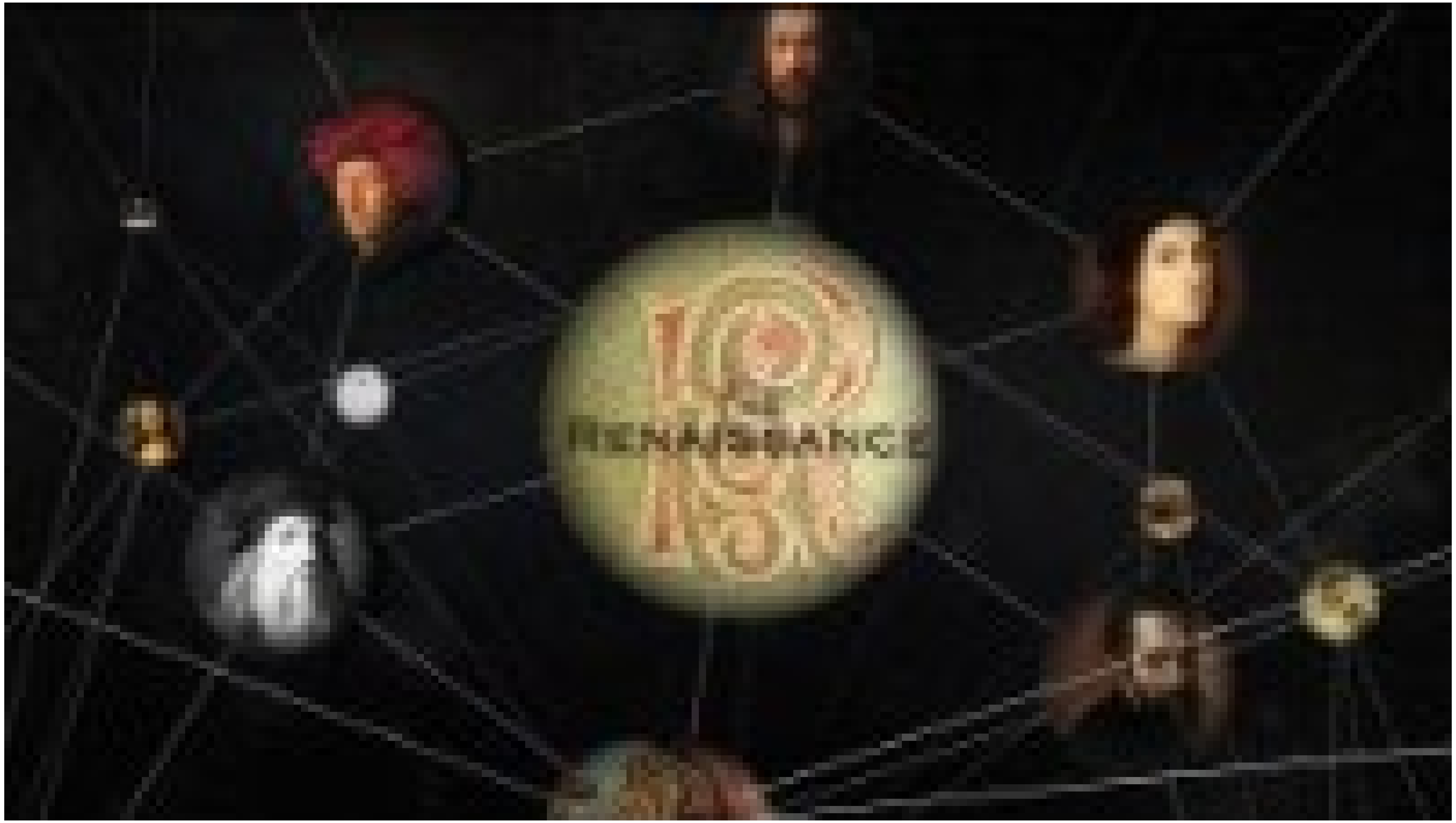


https://www.w3.org/2004/Talks/0611-sb-wsswintro/slide18-0.html

What’s the value of Knowledge Graphs for organizations?

Discussion.

The Knowledge Graph – According to Google



https://www.youtube.com/watch?v=mmQl6VGvX-c

“Why can’t I have relevant data at my fingertips whenever they are needed, the way Google does it for the web?”

Company leaders ask.

https://info.cambridgesemantics.com/hubfs/The_Rise_of_the_Knowledge_Graph.pdf

Parallel history of knowledge-based technology and data management technology merging into the data fabric

Semantic nets | Frame-based systems | Expert systems | AI winter | The web | Semantic Web | Knowledge graph | Data fabric

1960 | 1970 | 1980 | 1990 | 2000 | 2010 | 2020

Files

Hierarchical

RDBMS & SQL

OLAP data warehouse

Hadoop

Mapreduce

ElasticSearch

Spark

Cloud data warehouse

NoSQL movement

Data lake

Towards a data architecture truly scalable; durable from one application to the next.

https://info.cambridgesemantics.com/hubfs/The_Rise_of_the_Knowledge_Graph.pdf

Knowledge Graph in Enterprises

- A Knowledge Graph in enterprises:
  - is a central data element in the organizational data management infrastructure.
  - is a repository for organization-wide master data AND integration hub for various legacy data sources, e.g., relational databases or data streams.
- It consists of a collection of interlinked descriptions of concepts, entities, relationships and events, exploitable for the support of decision-making in businesses.
- Data is put in context via the semantic meta-data (or schema), enabling meaningful reasoning, retrieval, sharing and integration of knowledge.

Things
Real world objects and abstract concepts with **unique identifiers**

Relationships
How things are connected; First-class citizens

Semantics
The meaning of the data is encoded alongside the data (**metadata**)

Hogan et al. (2020), Fensel et al. (2020), Ehrlinger & Wöß (2016) and Jetschni & Meister (2017)

http://www.semantic-web-journal.net/system/files/swj2149.pdf

Basic Benefits of Knowledge Graphs in Enterprises

Unifying: Heterogenous internal and external data are integrated seamlessly.

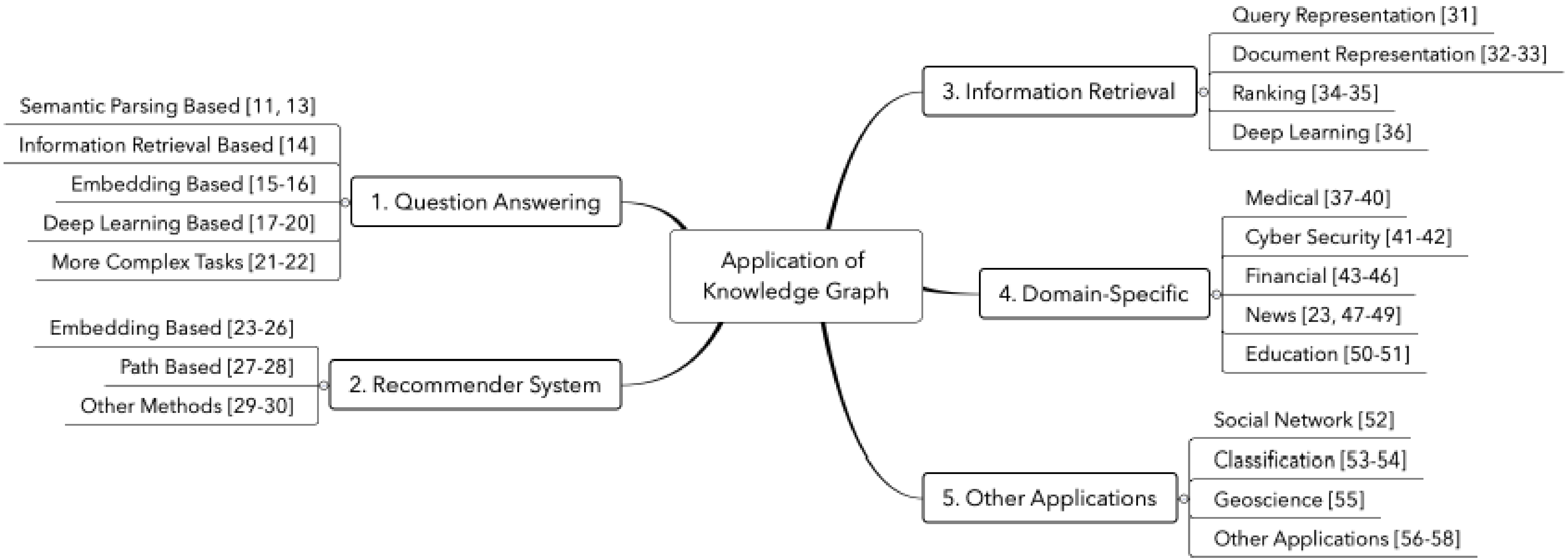
Flexible: Data and schema can be easily extended and connected.

Semantic: Data and its meaning are available in the same place.

Searchable: Unique IDs make all meta-levels searchable, sharable & accessible.

Trustworthy: Provenance information provides traceability and lineage.

Application fields of Knowledge Graph



https://iopscience.iop.org/article/10.1088/1742-6596/1487/1/012016

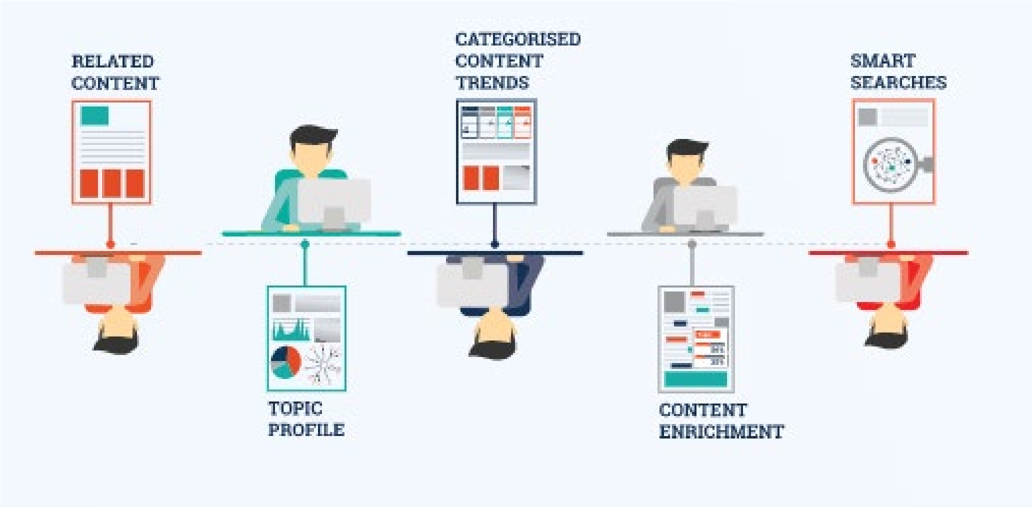
Examples for use of Knowledge Graphs in Enterprises

In Media

– The BBC case

https://youtu.be/9-g9A6zqFVw

– Content reuse and repurposing



In Engineering and Manufacturing

Smart Manufacturing Planning & Execution

SIEMENS

- ✓ AI-based knowledge graph application for automated, skill-based allocation of machines to production requests
- ✓ Cost & time savings by supporting planners & line operators in validation of manufacturing plans
- ✓ Enables realization of low-volume orders

Materials Science Knowledge Graph

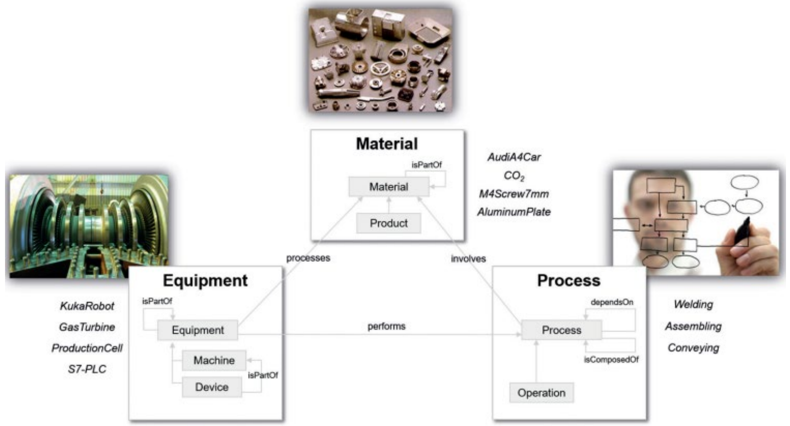
BOSCH

- ✓ Smart business application for material research & development
- ✓ One-stop knowledge hub for materials and chemical component information
- ✓ Meaningful & actionable insights surfaced through a user-friendly interface

Turbine Spare Parts Management

SIEMENS energy

- ✓ Smart and targeted maintenance of spare parts of large gas turbines
- ✓ Preventive maintenance resulting in reduced turbine downtimes
- ✓ Increased business user and customer satisfaction
- ✓ Savings of thousands of hours on manual effort



https://metaphacts.com/images/PDFs/case-studies/metaphacts-Case-Study-Smart-Manufacturing-at-Siemens-with-metaphactory-Knowledge-Graph-Platform.pdf

https://metaphacts.com/resource-hub

In Pharma & Life Sciences

Drug Development & Drug Repurposing

Swiss multinational healthcare company

- ✓ Target discovery dashboard connecting & transforming proprietary & public information into explicit knowledge
- ✓ Data scientists, immunologists & systems biologists gain access to actionable insights for drug discovery & repurposing

Omics Data Management

Large German pharmaceutical company

- ✓ One-stop knowledge hub for gene expression data helping data stewards in bridging the gap between business and IT
- ✓ Bioinformaticians benefit from intuitive exploration of gene sequencing data for specific diseases and time frames

Clinical Analytics and Informatics Dashboard

American multinational pharmaceutical corporation

- ✓ Intelligent dashboard providing an integrated view over a data mesh of proprietary & public data sources
- ✓ Accelerated & optimized drug discovery & development through contextualized data & reasoning

https://metaphacts.com/resource-hub

In Cultural Heritage

Reference Data Services

Swiss Art Research Infrastructure

University of Zurich[UZH]

- ✓ Unified access to scholarly established, high-quality, yet extendable reference data
- ✓ Research & cultural heritage institutions can integrate own terminology into existing, multilingual thesauri & make them publicly accessible

Knowledge System Evolution

MPIWG
Max Planck Institute for the History of Science

- ✓ Reconstruction of the transformation process of the original treatise 'Tractatus de sphaera' by Johannes de Sacrobosco
- ✓ Exploration of the evolutionary path of the scientific system pivoted around cosmological knowledge

Performing Arts Archive

sapa

- ✓ Publicly available archive of data around important performing arts events in Switzerland
- ✓ Accessible through an intuitive end-user interface

https://www.performing-arts.ch

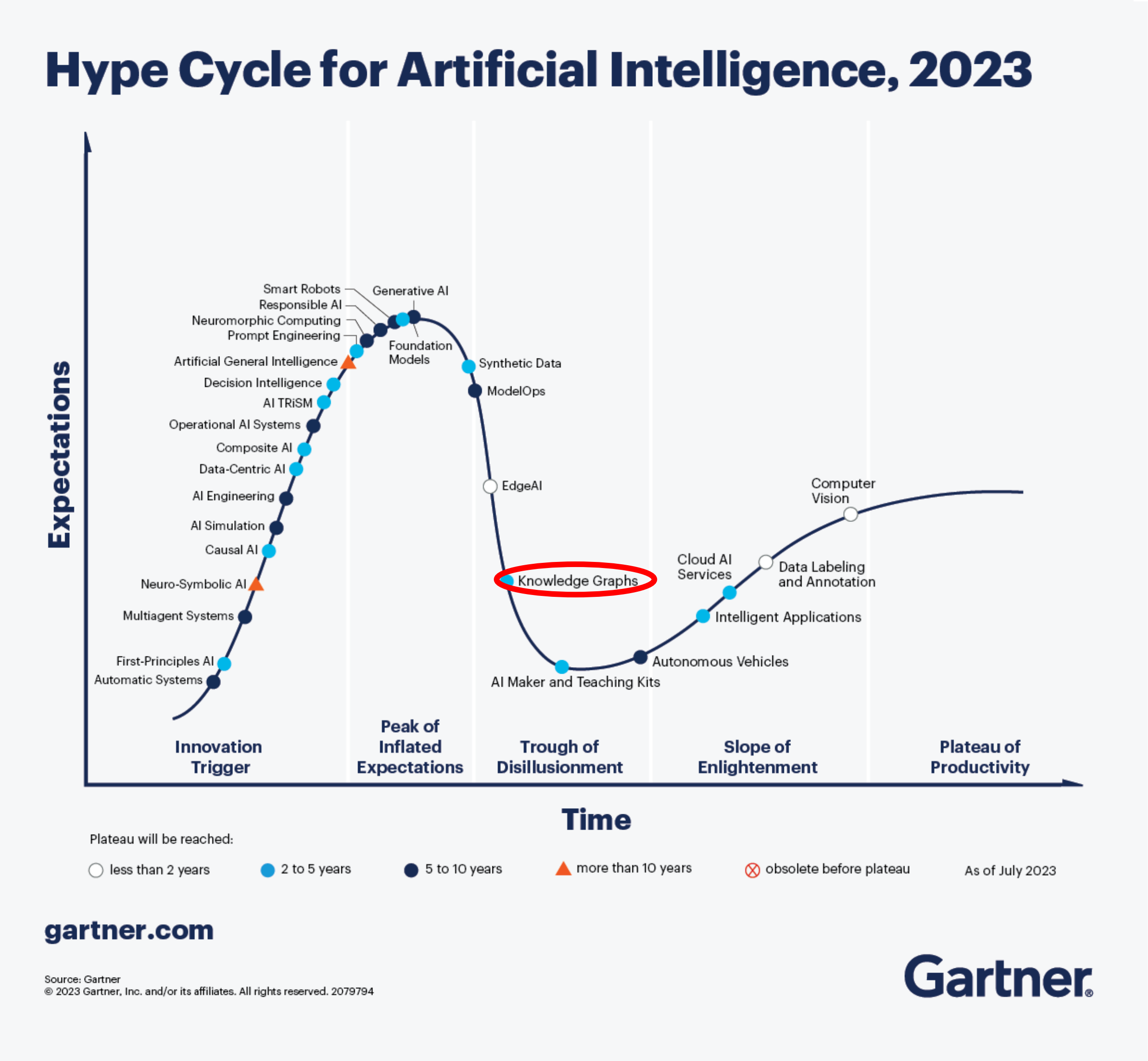
https://metaphacts.com/resource-hub

Additional resources with case studies and white papers

- Case studies:
- https://www.ontotext.com/knowledge-hub/case-studies/
- https://www.stardog.com/resources/#filter=.case-studies
- White papers:
- https://www.ontotext.com/knowledge-hub/white_paper/
- https://www.stardog.com/resources/#filter=.whitepapers

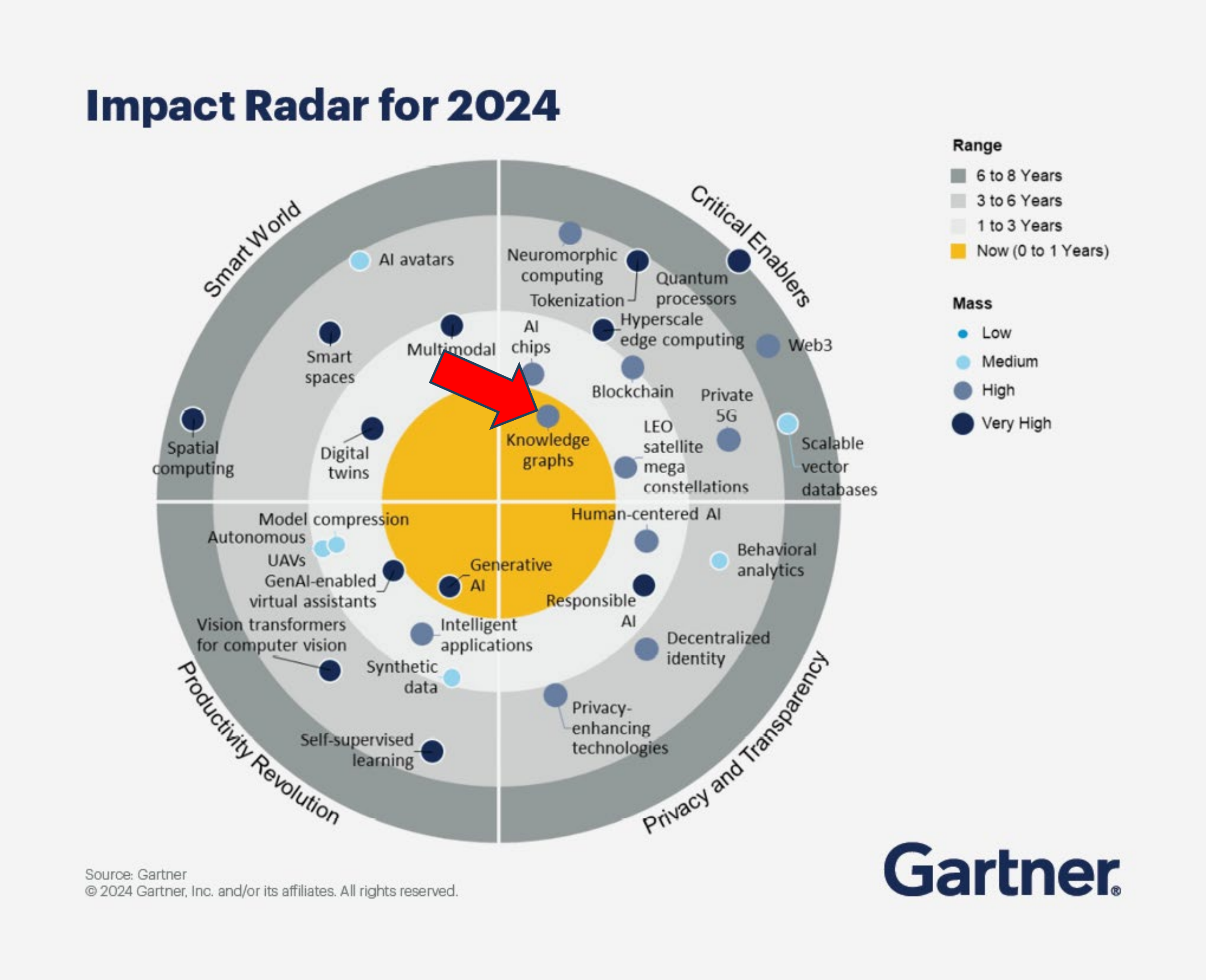
Trends for Knowledge Graphs

- The 2023 Gartner Hype Cycle™ for Artificial Intelligence (AI) identifies innovations and techniques that offer significant and even transformational benefits.
- Gartner Hype Cycle methodology gives you a view of how a technology or application will evolve over time, providing a source of insight to manage its deployment within the context of specific business goals.



https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2023-gartner-hype-cycle

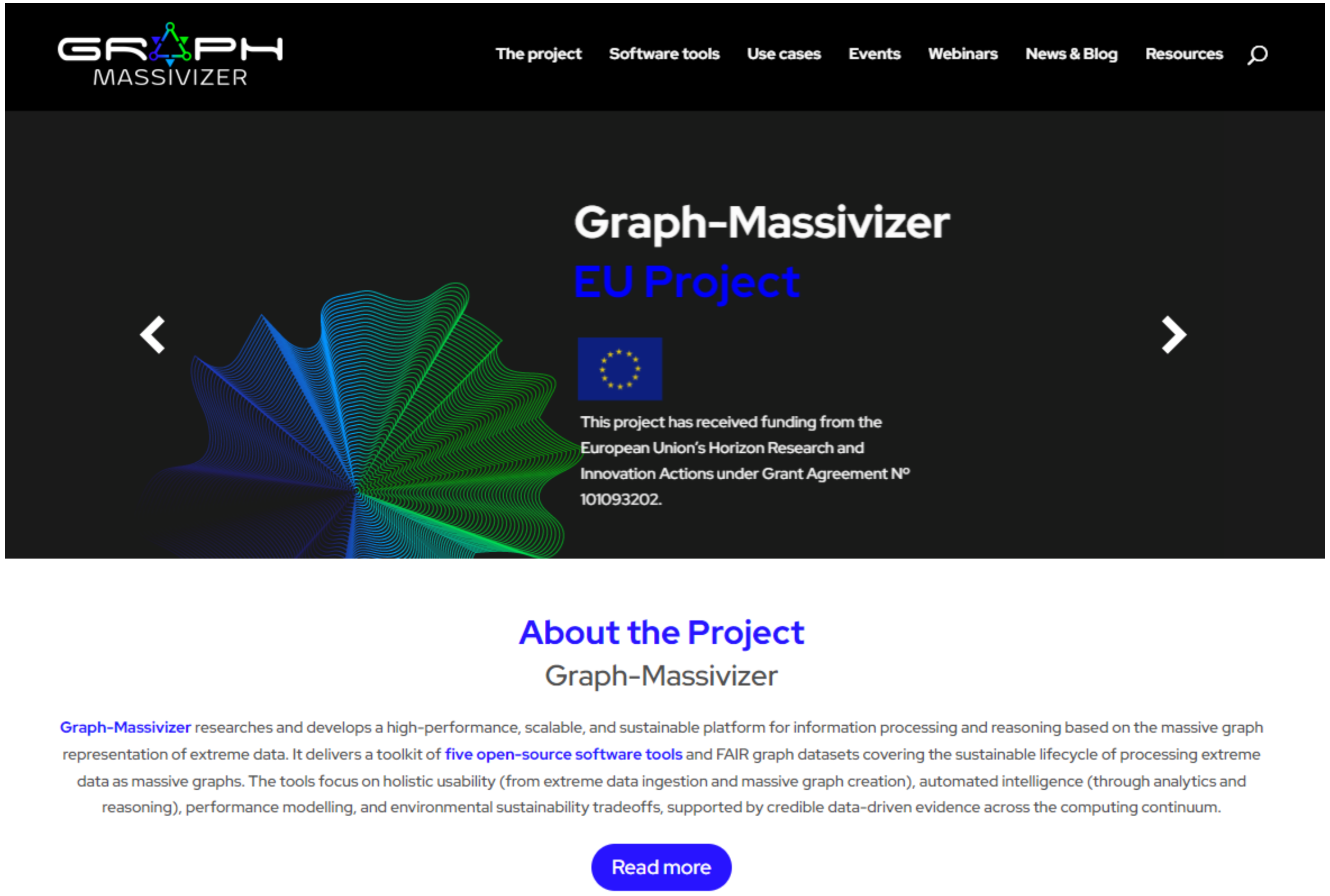
The **Gartner Emerging Tech Impact Radar** highlights the **technologies and trends** with the greatest potential to disrupt a broad cross-section of markets.



https://www.gartner.com/en/articles/30-emerging-technologies-that-will-guide-your-business-decisions

Some initiatives /1

https://www.knowledgegraph.tech/



https://graph-massivizer.eu/

Some initiatives /2

HORIZON EUROPE

AI powered Data Curation & Publishing Virtual Assistant

Fact Sheet

Project description

DE EN ES FR IT PL

AI-based automation helps citizens curate their personal health data

By 2030, European citizens should be in full possession of their personal health data. Currently, this data is scattered across different clinics, surgeries or hospitals and across medical devices or personal health apps. There is also a lot of information in paper form. Most of the data cannot be used by advanced algorithms supporting preventive and personalised medicine. In this context, the EU-funded AIDAVA project will maximise automation of data curation and publish unstructured and structured, heterogeneous data using a virtual assistant powered by AI. Central to the project is the concept of the FAIR Guiding Principles, which require data to be findable, accessible, interoperable and reusable.

Show the project objective

Fields of science

social sciences › sociology › industrial relations › **automation**
natural sciences › computer and information sciences › **knowledge engineering**
medical and health sciences › clinical medicine › oncology › **breast cancer**
medical and health sciences › health sciences › **personalized medicine**
natural sciences › computer and information sciences › artificial intelligence › machine learning › **deep learning**

Project Information

AIDAVA
Grant agreement ID: 101057062

DOI
10.3030/101057062

Start date
1 September 2022

End date
31 August 2026

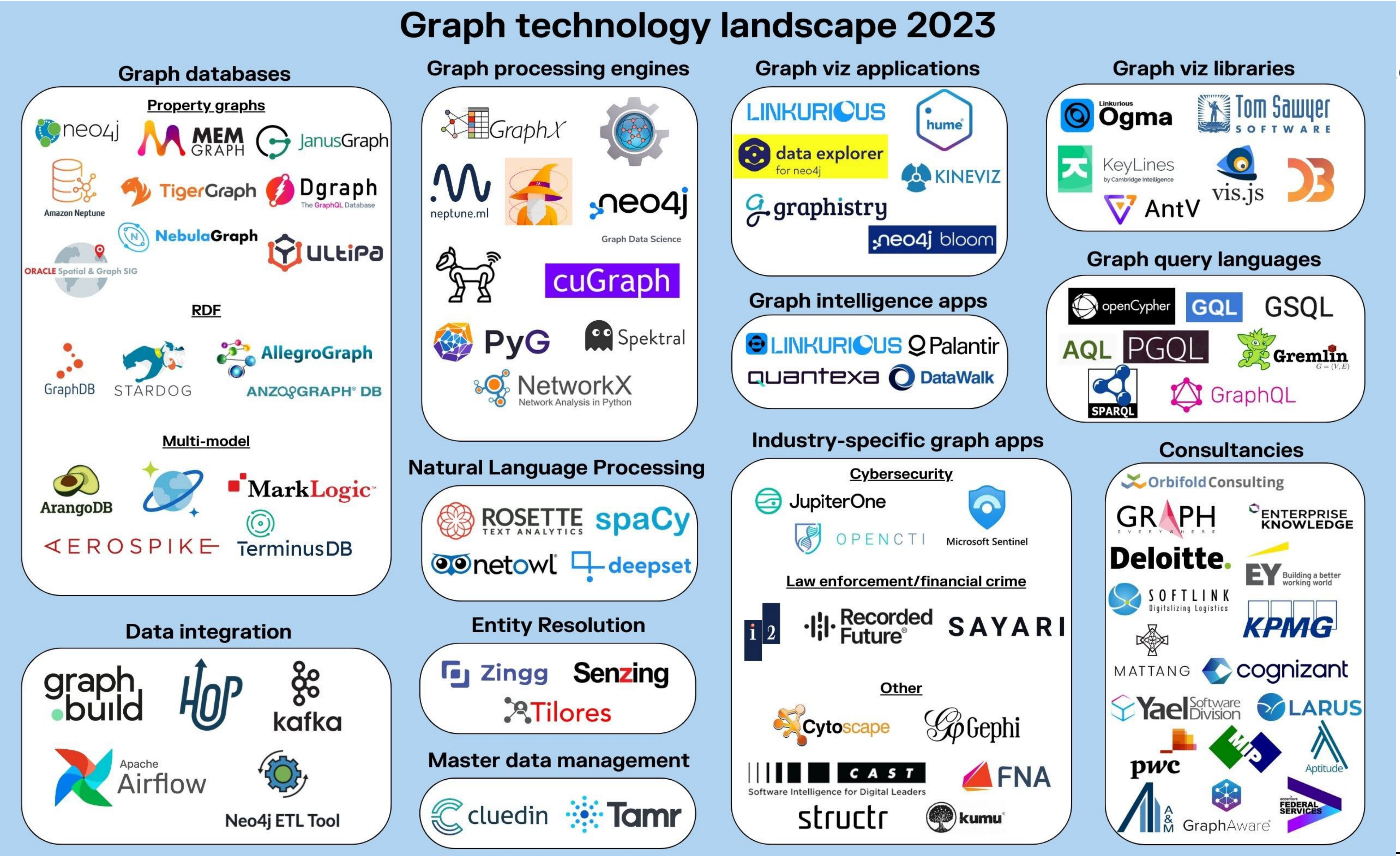
Funded under
Health

Total cost
€ 7 720 618,75

EU contribution
€ 7 720 615

Coordinated by
UNIVERSITEIT MAASTRICHT
Netherlands

https://cordis.europa.eu/project/id/101057062



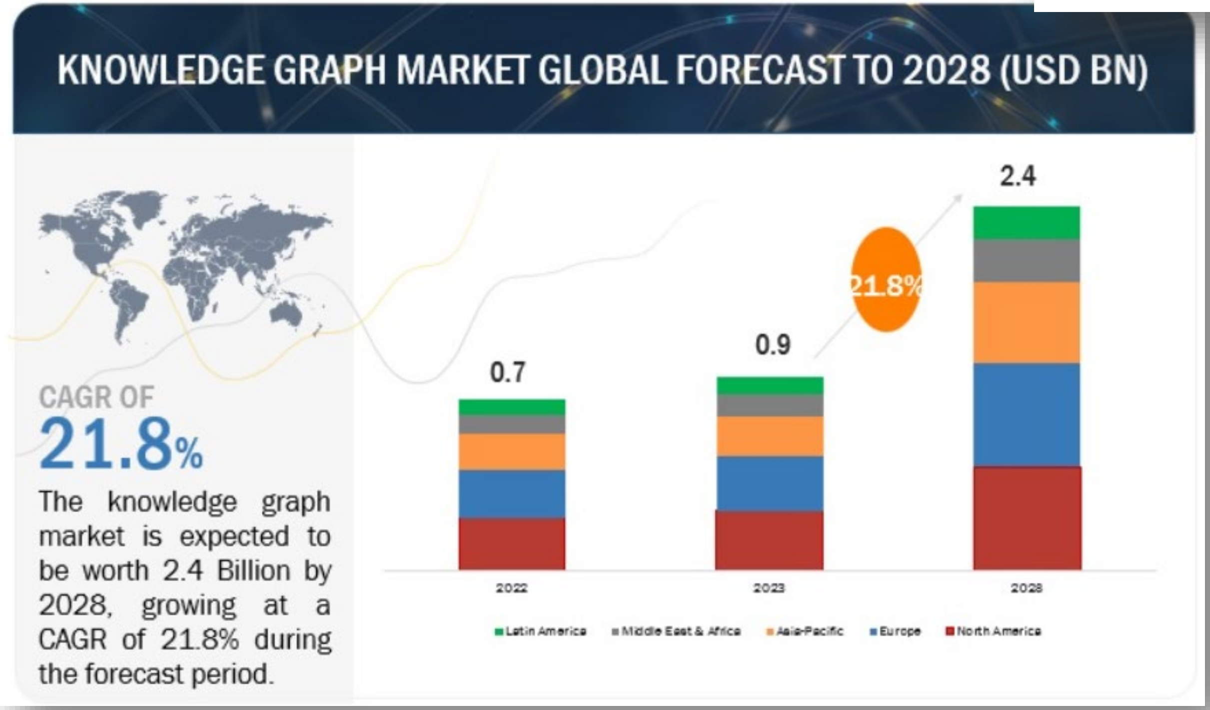
https://linkurious.com/blog/introduction-graph-technology-landscape/

Market forecast about Knowledge Graphs

MMR

Global knowledge graph industry is highly fragmented, competitive, diverse, driven by emergence of start-ups and innovation.

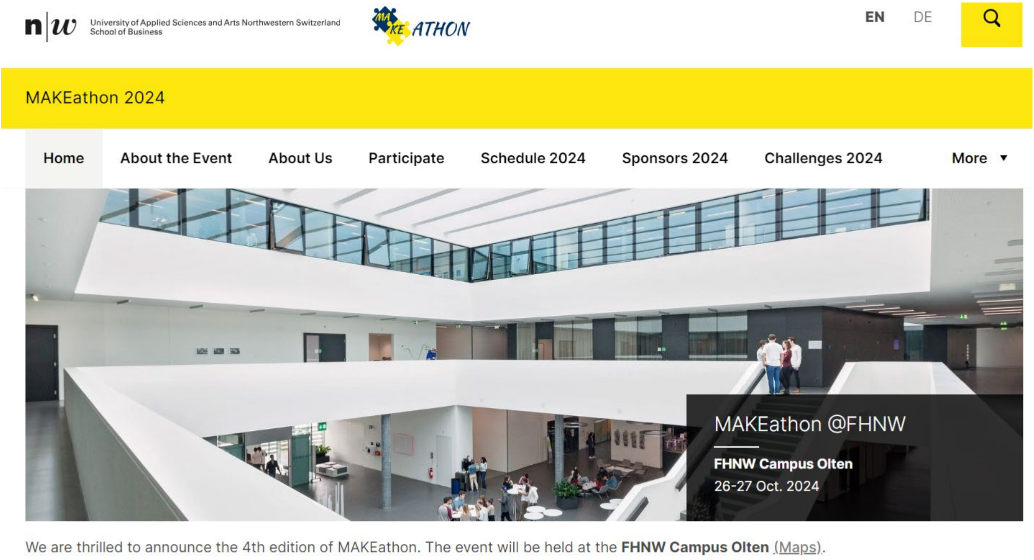
The integration of AI and machine learning is a transformative trend for the knowledge graph

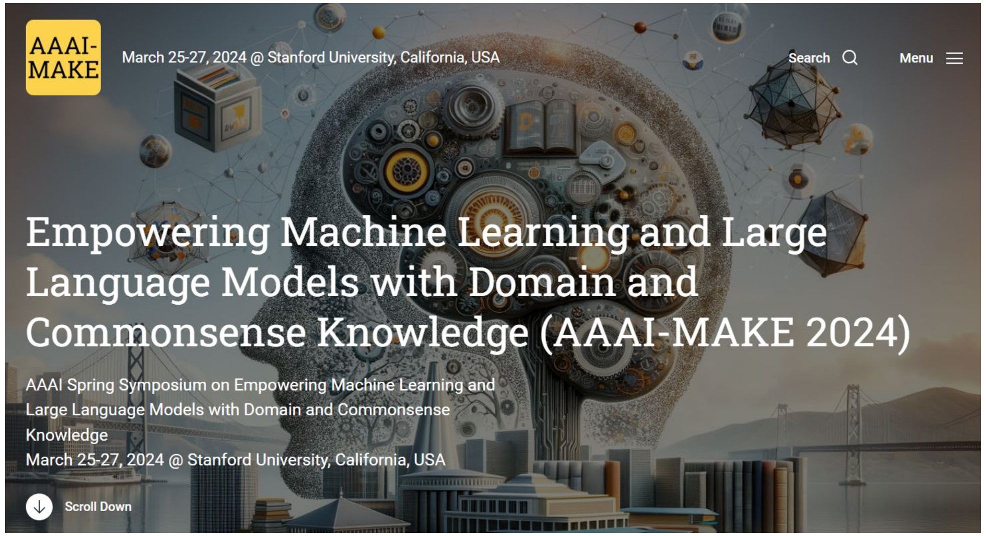


Forecasted investments by Application (2023 to 2029)	Semantic search Recommendation Systems Data integration Knowledge Management AI and machine learning

https://www.marketsandmarkets.com/Market-Reports/knowledge-graph-market-217920811.html
https://www.maximizemarketresearch.com/market-report/knowledge-graph-market/221742/

Scientific…and slowly also industry trend: Knowledge Graphs + Machine Learning





https://makeathonfhnw.ch/

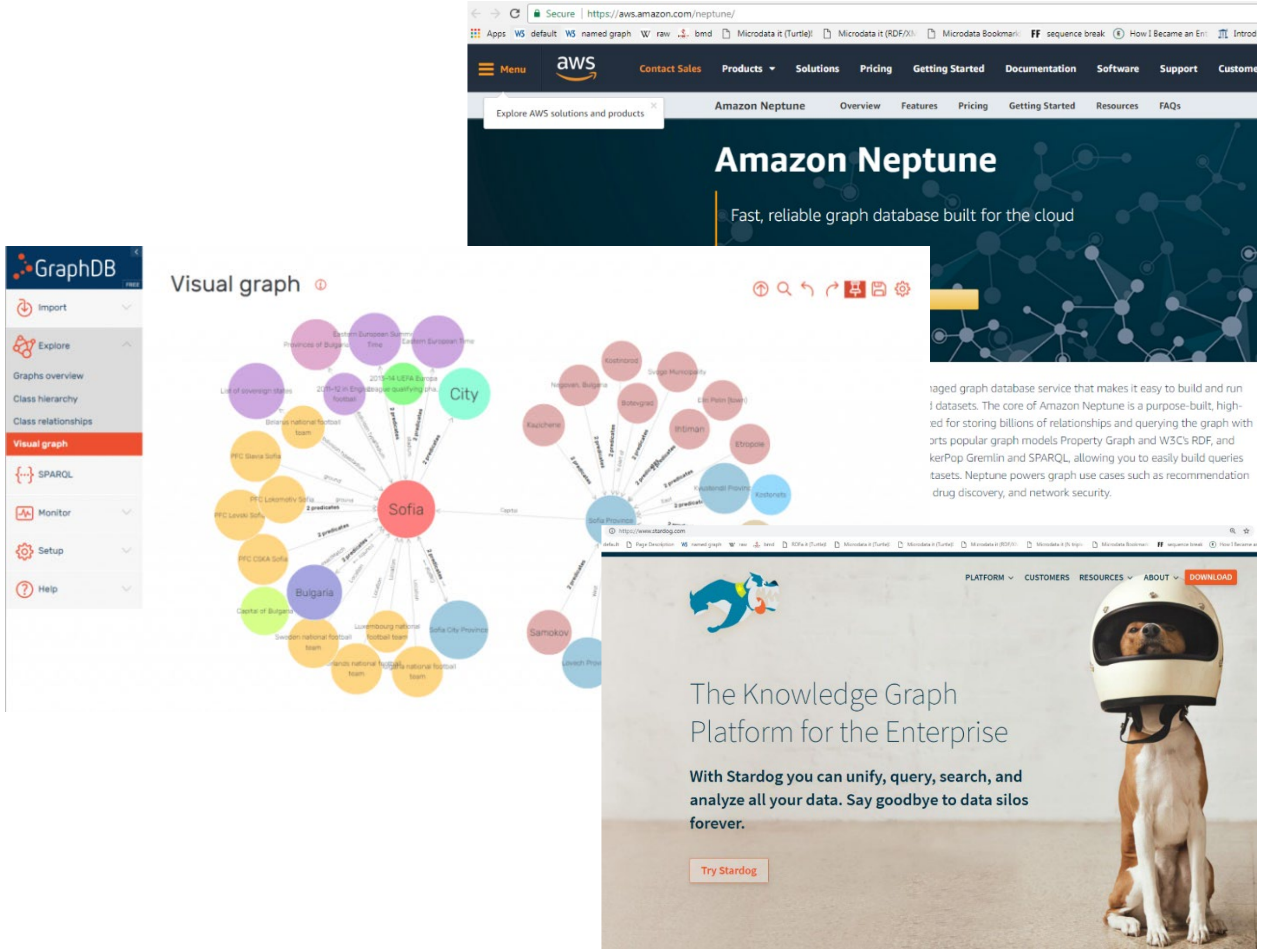
https://www.aaai-make.info/

https://hybridaims.com/

Popular Knowledge Graphs Technologies for W3C standards

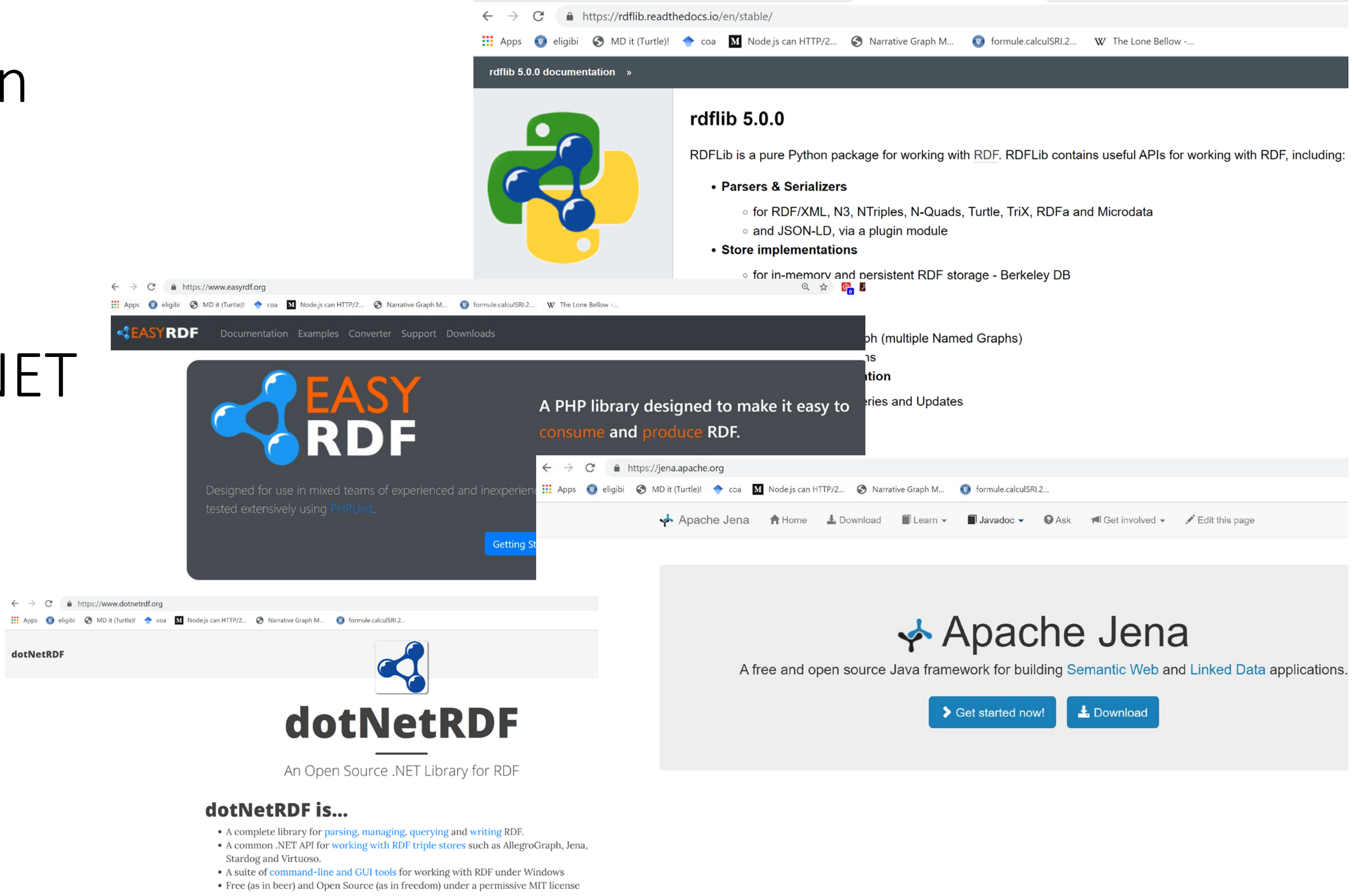
Graph Databases Servers

- Neptun (part of AWS)
- GraphDB
- Stardog
- …



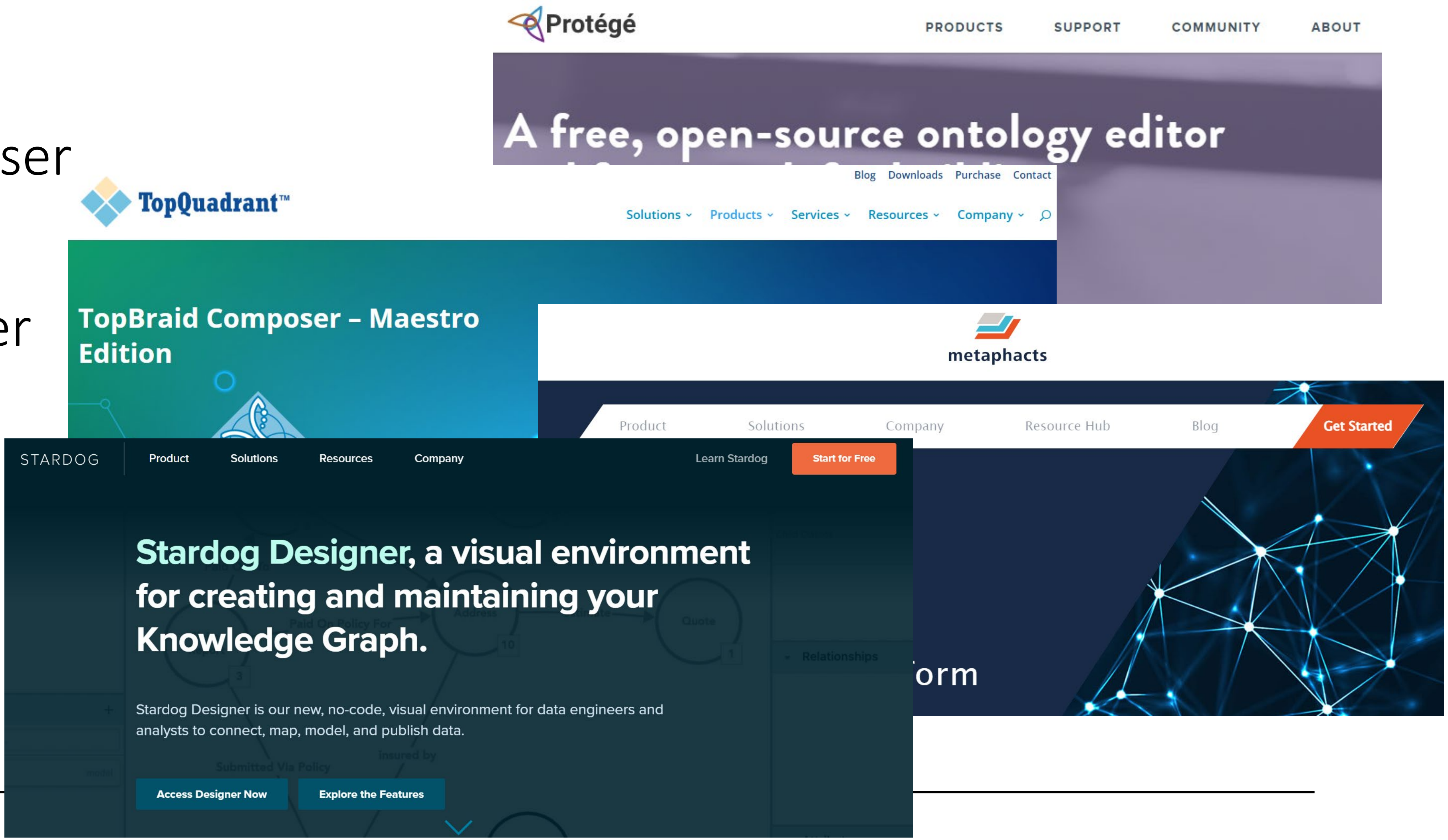
Programming Libraries for Knowledge Graphs

– RDFLib for Python
– EasyRDF for PHP
– Jena for Java
– dotNetRDF for .NET



Tools for Ontology Engineering

– Protégé
– TopBraid Composer
– Metaphactory
– Startdog Designer

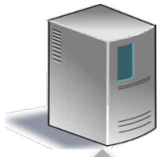


Knowledge Representation and Reasoning

A pragmatic view.

Knowledge-Representation and Reasoning

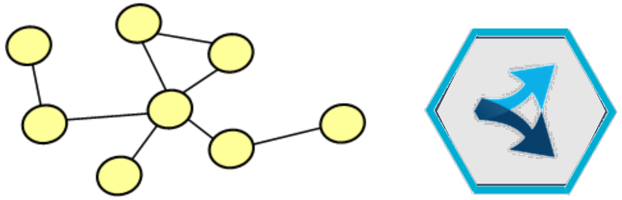
Graph Database/Triplestore and inference engine



consistent

Storing, integration, querying, reasoning.

Knowledge representation formalism

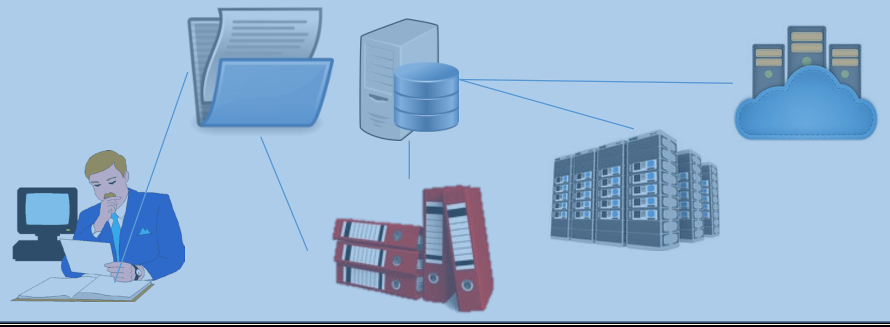


machine-interpretable knowledge

RDF, RDF(S), OWL, SPARQL, SHACL…

consistent

Representation of reality



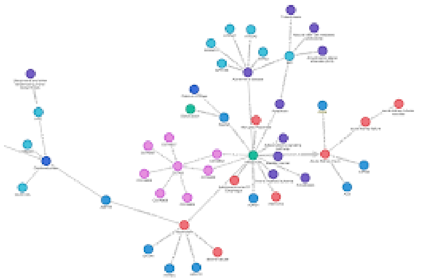
Not formally represented

Adapted from lecture of Prof. Dr. Knut Hinkelmann

Representation of Reality.

Exercise in Pair (5+5 min)

- Describe your family tree to your nearest classmate. If possible, mention what your family component(s) likes and/or where he/she works or study.
  - Switch roles after 5 min – the one who speaks listens and vice-vera.
- Constraints:
  - The one who listens has to note his understanding down in the following triple structure:
    - Subject - Predicate - Object
    - E.g., Emanuele - works at - FHNW
  - **Do not** specify the types for the subjects or objects.
    - Each subject and object should stick to a representation of individuals (or instances). Do not make abstractions like class mother, or father etc. These shall be defined in the predicates,
    - e.g. hasMother, hasFather etc.



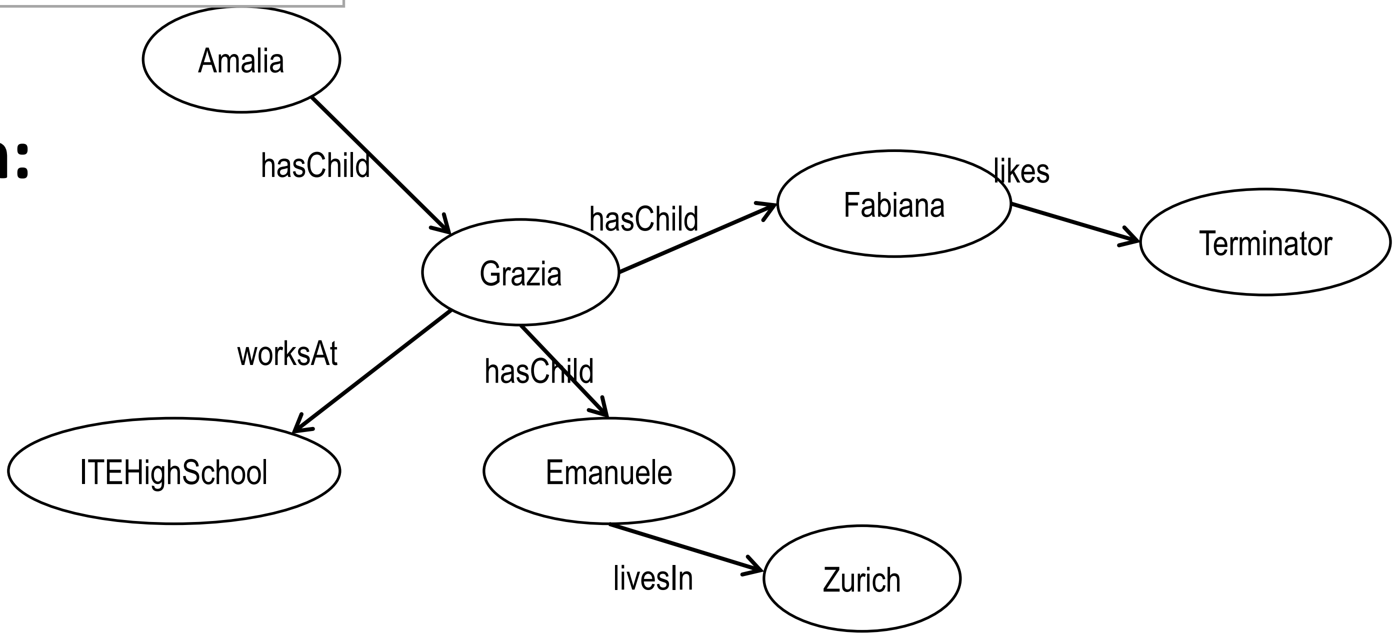
Individual Exercise (5 min)

- Take the triples that were listed by your classmate and create a graph.
  - Every subject and object is a node;
  - Every predicate is an edge (i.e., a link or relation);
  - Nodes are connected by edges.

A possible solution

Subject	predicate	Object
:Grazia	:worksAt	:ITEHighSchool.
:Fabiana	:likes	:Terminator.
:Emanuele	:livesIn	:Zurich.
:Amalia	:hasChild	:Grazia.
:Grazia	:hasChild	:Fabiana, :Emanuele.

The resulting graph:

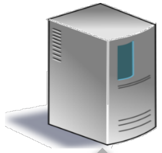


RDF

Resource Description Framework

Knowledge-Representation and Reasoning

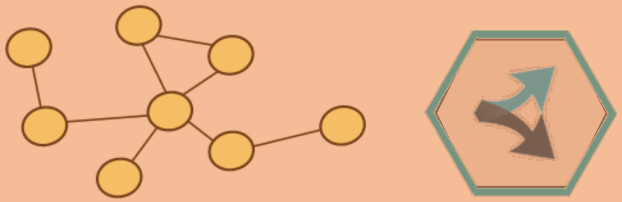
Graph Database/Triplestore and inference engine



Storing, integration, querying, reasoning.

consistent

Knowledge representation Formalism

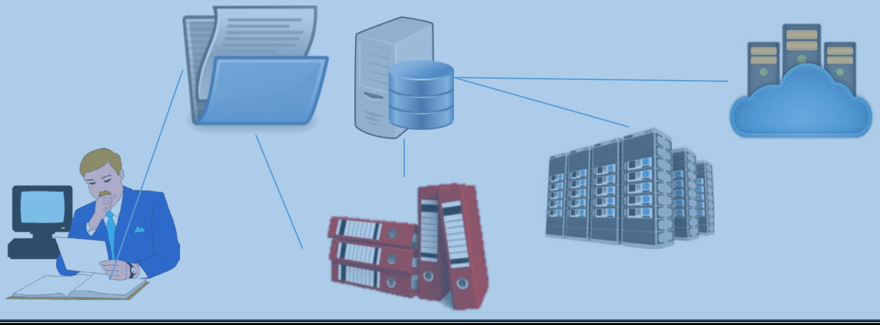


machine-interpretable knowledge

RDF, RDF(S), OWL, SPARQL, SHACL…

consistent

Representation of reality



Not formally represented

Adapted from lecture of Prof. Dr. Knut Hinkelmann

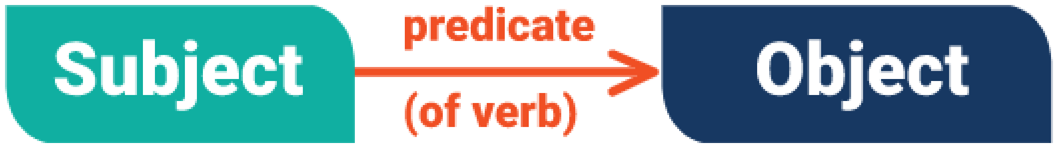
About RDF

– It stands for Resource Description Framework.
– A [World Wide Web Consortium] (W3C) standard.
– It is used to describe and exchange information/data model in the Web.
– Key data structure: RDF graphs.
– Graphs are a set of statements.
– A statement is also called triple.
– Each statement or triple consists of 2 nodes connected by a predicates:
  – *Subject-Predicate-Object*
– Every resource is identified by a URI (Universal Resource Identifier), i.e., an object in the "web", e.g.
  - *https://www.wikidata.org/wiki/Q12418*

https://www.w3.org/TR/rdf11-concepts/

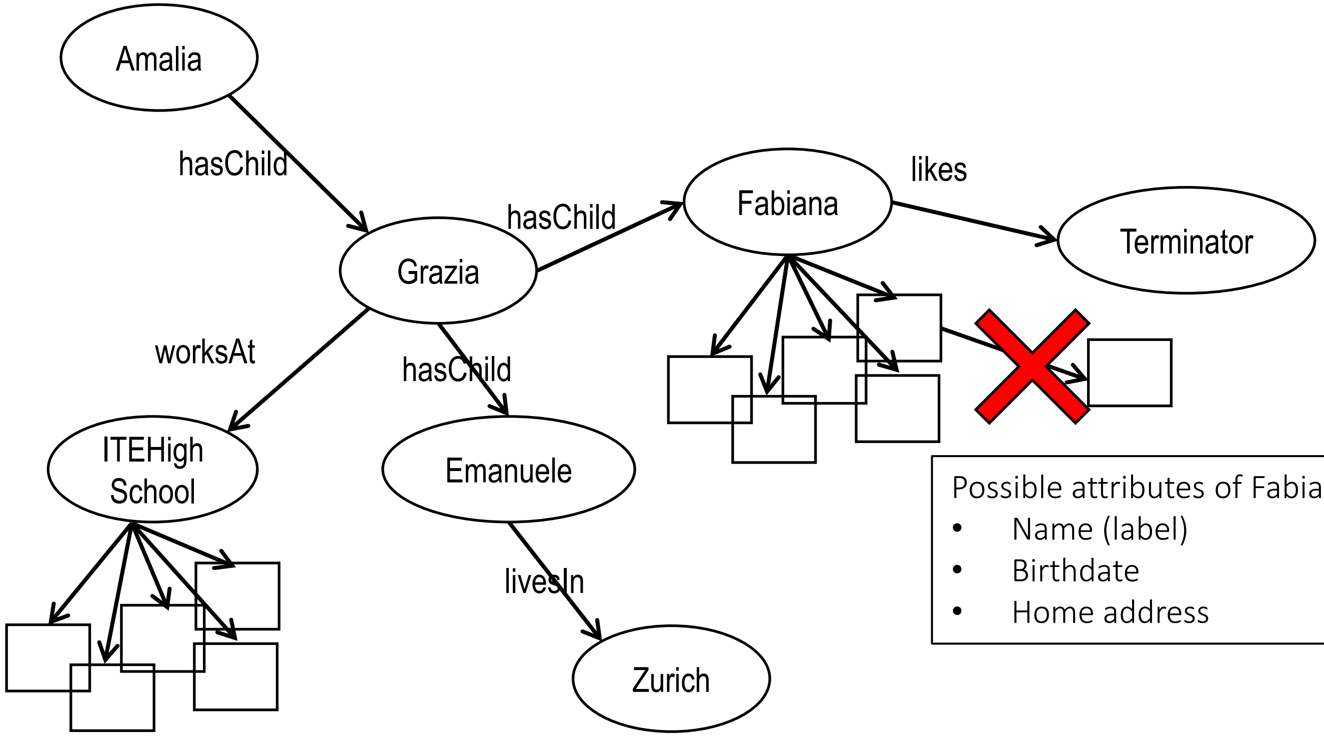
RDF Triples: Three components

- The subject, is a resource (URI) or a blank node.
- The predicate, is a resource (URI).
- The object, is a resource (URI), a literal or a blank node.
  - A literal is a terminal node = data values (strings, numbers, etc).



https://www.w3.org/TR/rdf11-concepts/

Example of a RDF Graph with Data Values



Possible attributes of Fabiana:
- Name (label)
- Birthdate
- Home address

Data values are terminal nodes! They can only occur at the end of a Turtle statement =>

:Fabiana :hasAge 34.

Not allowed:

34 :ageOf :Fabiana.

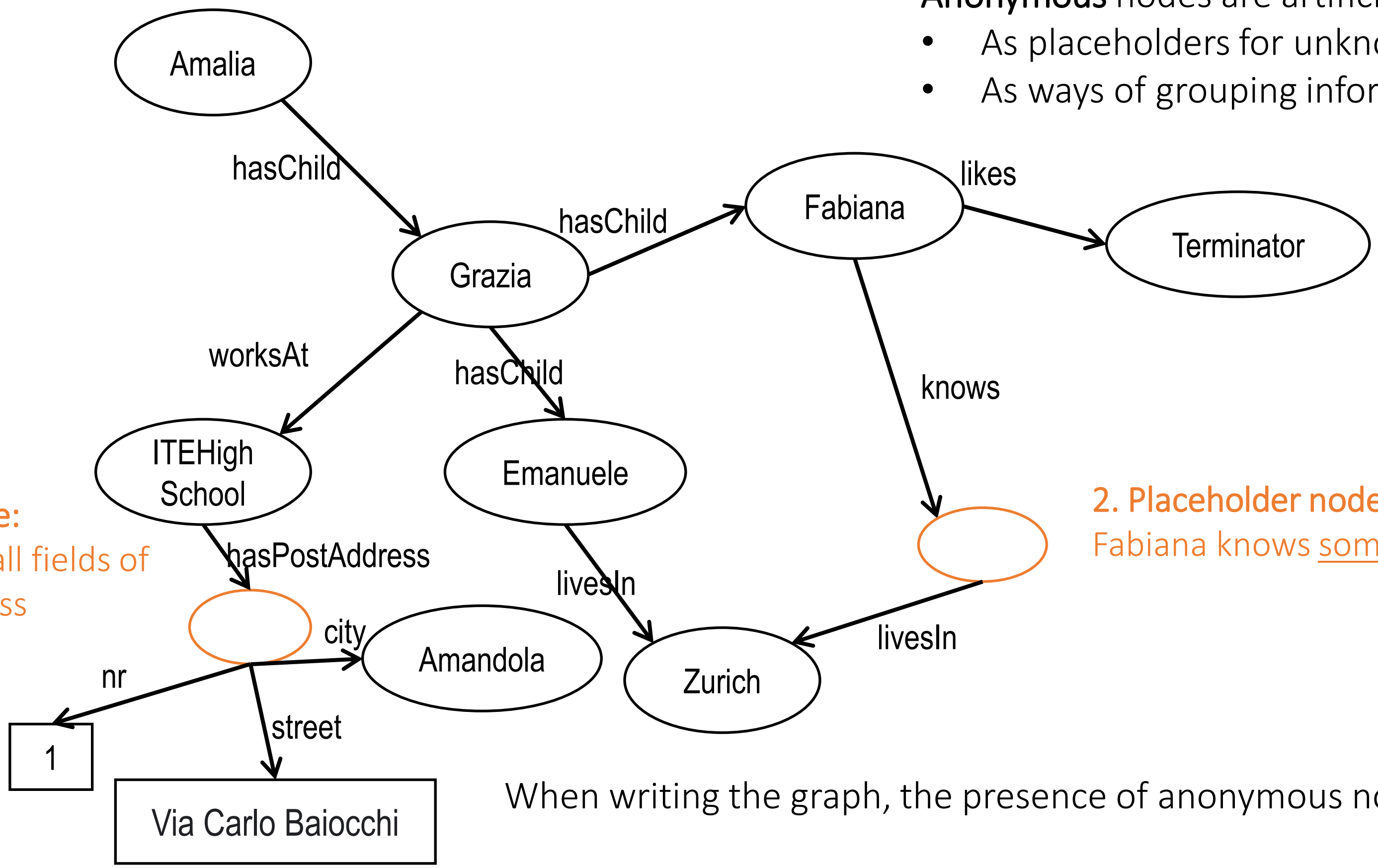
Data values are terminal nodes ("hanging fruits" that can be attached to any ID node).

Adapted from lecture of Prof. Dr. Buchmann

Anonymous (blank) nodes

Anonymous nodes are artificially introduced:

- As placeholders for unknown IDs
- As ways of grouping information ("data structures")



1. Grouping node: Keeps together all fields of the postal address

2. Placeholder node: we can express that Fabiana knows someone who lives in Zurich

When writing the graph, the presence of anonymous nodes is indicated with [...]:

1. :ITEHighSchool :hasPostAddress [:nr 1; :street "Via Carlo Baiocchi"; :city :Amandola].
2. :Fabiana :knows [:livesIn :Zurich].

Adapted from lecture of Prof. Dr. Buchmann

Syntax for RDF Graphs

- RDF statements come in the form of graphs.
- Syntax for writing RDF graphs (data format/notation for storing RDF data):
  - Turtle (Terse RDF Triple Language) – the most human readable format, the most used.
  - JSON-LD
  - RDF/XML
  - …

https://www.w3.org/TR/2014/REC-turtle-20140225/

Formal Foundation

– Every “statement” (or triple) is a logical predicate (similar to Prolog).

statement

Subject (ID)	predicate (ID)	Object (ID)
:Grazia	:worksAt	:ITEHighSchool.
:Fabiana	:likes	:Terminator.
:Emanuele	:livesIn	:Zurich.
:Amalia	:hasChild	:Grazia.
:Grazia	:hasChild	:Fabiana, :Emanuele.

:Emanuele :livesIn :Zurich.

⇔

livesIn(Emanuele, Zurich)

Knowledge Graphs are object-oriented.
Prolog is predicate oriented.

– Syntactical delimeters to write more complex phrases:

Emanuele :livesIn :Zurich;
:worksAt :FHNW;
:hasWorkAddress [:street “Riggenbachstrasse”; :nr 16].

Formal Foundation

– Syntactical delimeters to write more complex phrases:

Subject (ID)	predicate (ID)	Object (ID)
:Grazia	:worksAt	:ITEHighSchool.
:Fabiana	:likes	:Terminator.
:Emanuele	:livesIn	:Zurich.
:Amalia	:hasChild	:Grazia.
:Grazia	:hasChild	:Fabiana, :Emanuele.

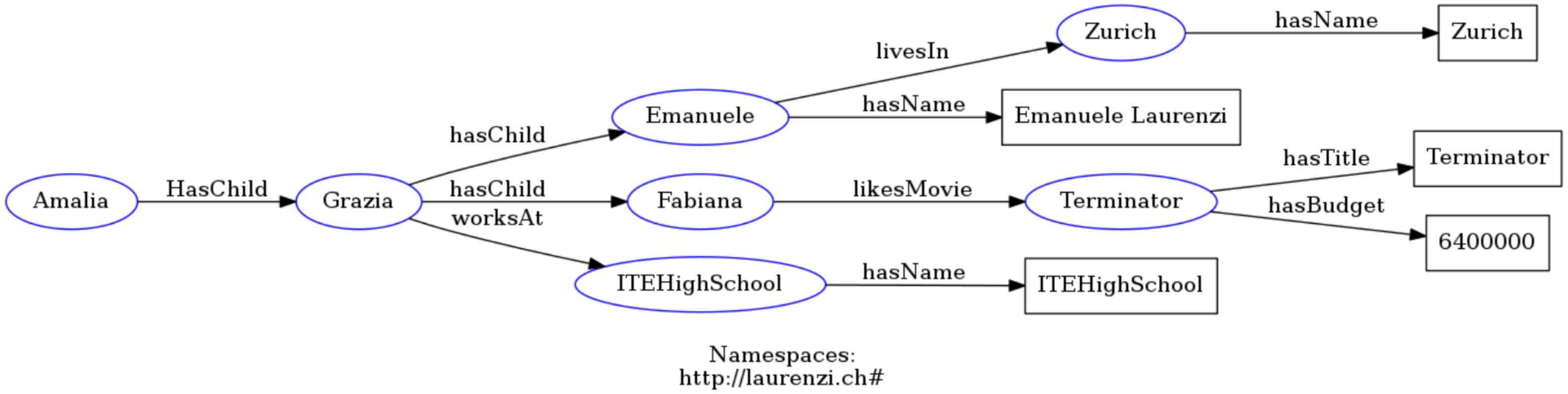
Emanuele :livesIn :Zurich;
:worksAt :FHNW;
:hasWorkAddress [:street “Riggenbachstrasse”; :nr 16].

Adapted from lecture Prof. Dr. Buchmann

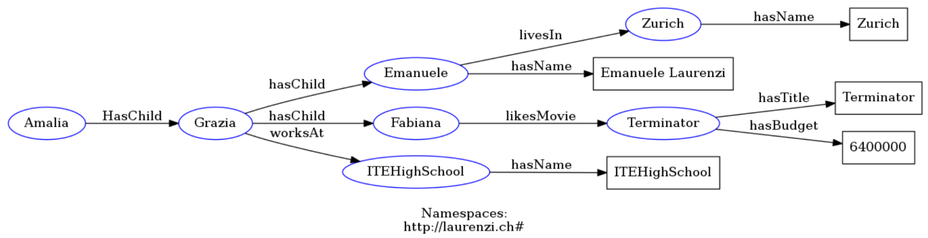
Turtle File – Family Tree

```
@prefix : <http://laurenzi.ch#>.

:Amalia
  :HasChild :Grazia ;
.
:Emanuele
            :livesIn :Zurich ;
            :hasName "Emanuele Laurenzi" ;
.
:Fabiana
            :likesMovie :Terminator ;
.
:Terminator
           :hasTitle "Terminator";
           :hasBudget "6400000" ;
.
:Grazia
            :hasChild :Emanuele ;
            :hasChild :Fabiana ;
            :worksAt :ITEHighSchool ;
.
:ITEHighSchool
            :hasName    "ITEHighSchool";
.
:Zurich
            :hasName "Zurich" ;
.
```



https://www.ldf.fi/service/rdf-grapher



Knowledge Representation Formalism. Individual Exercise (5 min)

- Turn your statements in Turtle notation.
- Use https://www.ldf.fi/service/rdf-grapher to validate the statements and to automatically create the graph.

Entity Types = Classes

- We distinguish between:
  - Concrete things (individual objects) in the domain:
    - Amalia, Grazia, Emanuele, Fabiana, Zurich, ITE High School, Terminator.
  - Set of individuals sharing properties called **classes**:
    - Person, Movie, City, High School.
- Individual objects that belong to a class are referred to as instances of that class.
- The standard relationship **rdf:type** or **a** are used to state that a resource is an instance of a class.

Kinds of Nodes

- All nodes in previous examples are identifiers (IDs, also called URIs, IRIs)

- Other kinds of nodes in RDF graphs:
  - data types (integers, booleans etc.)
  - entity types (classes)
  - anonymous nodes (placeholders, helpers)

Kinds of Nodes: An Example

```
:Emanuele      a             :Lecturer, :Italian, :Person;
               rdfs:label    "Emanuele Laurenzi";
               :hasAge       36;
               :married      false;
               :hasParents   :Omar, :Grazia;
               :authorOf     [a :Book;  :yearOfPublication  2022;
                                :hasTitle "Domain-Specific Conceptual
                                Modeling"@en];
               :livesIn      [:street "Gablerstrasse"@de;  :nr 47];
               :bought       [:product :Bananas;
                             :quantity [rdf:value 1; :unit "kilogram"];
                             :price [rdf:value 2.3; :currency  "CHF"]].
```

Comma devides different values for the same attribute and different objects.

Semicolon devides different attributes.

https://www.w3.org/TR/turtle/
Section 2.5 Literals
Quoted Literals (String) and Numbers

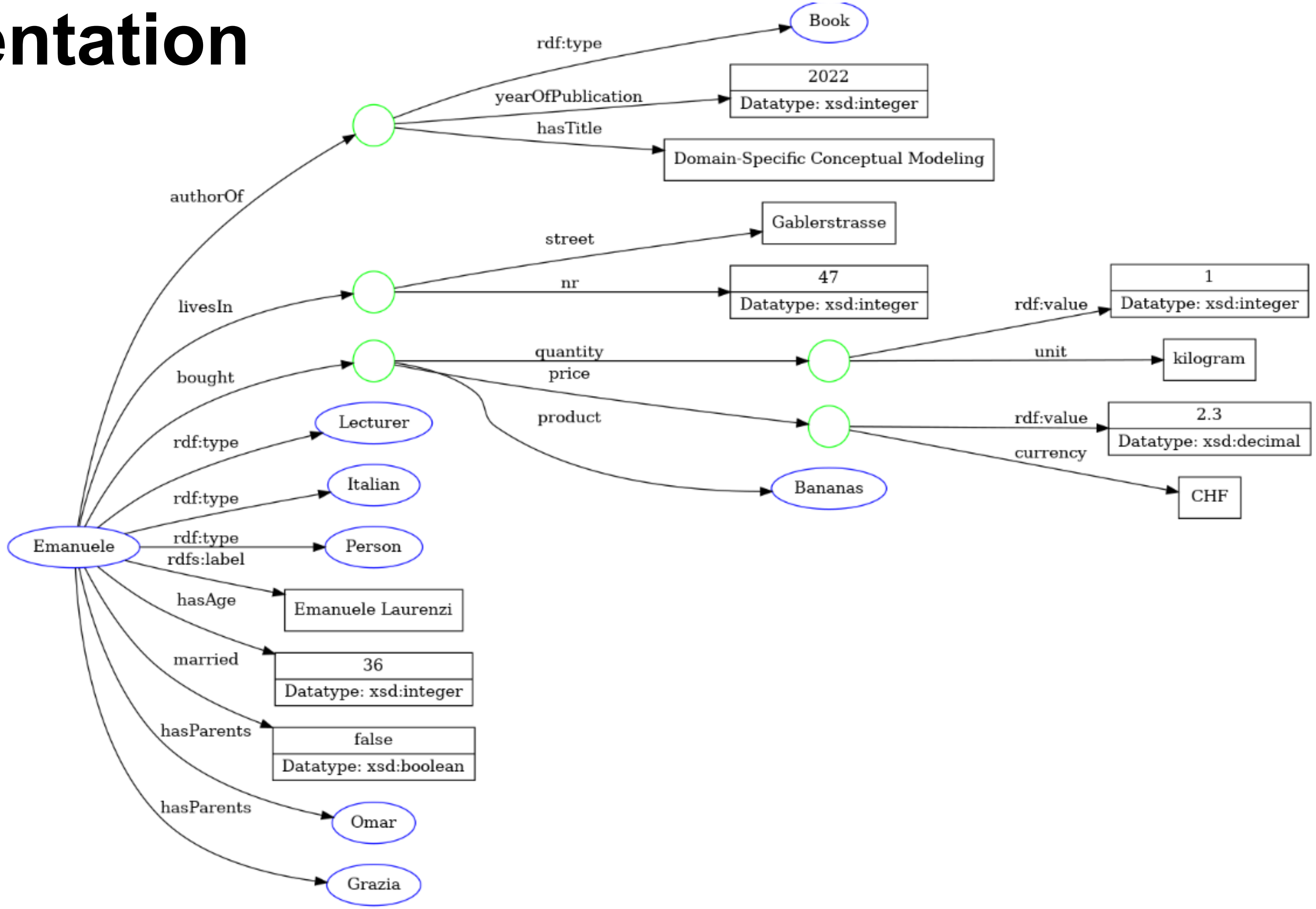
– Datatypes (Integer, Decimal, Double, String) are automatically associated based on the data value of the literal, e.g.
  36 is associated to Integer,
  false/true is associated to Boolean,
  4.3 is associated to a Decimal,
  1.663E-4 is associated to a Double,
  “Gablerstrasse” is associated to String.
  The String data type wants the “ ”
– XML Schema datatypes can be added to the value, e.g.,
  “Gablerstrasse”^^xsd:string
– Language tag can be attached to the String datatype, e.g.
  “Gablerstrasse”@de OR “Hi there!”@en
– Datatypes are predefined in XSD (XML Schema Definition).
– Specify the namespace for XSD when using datatypes:

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

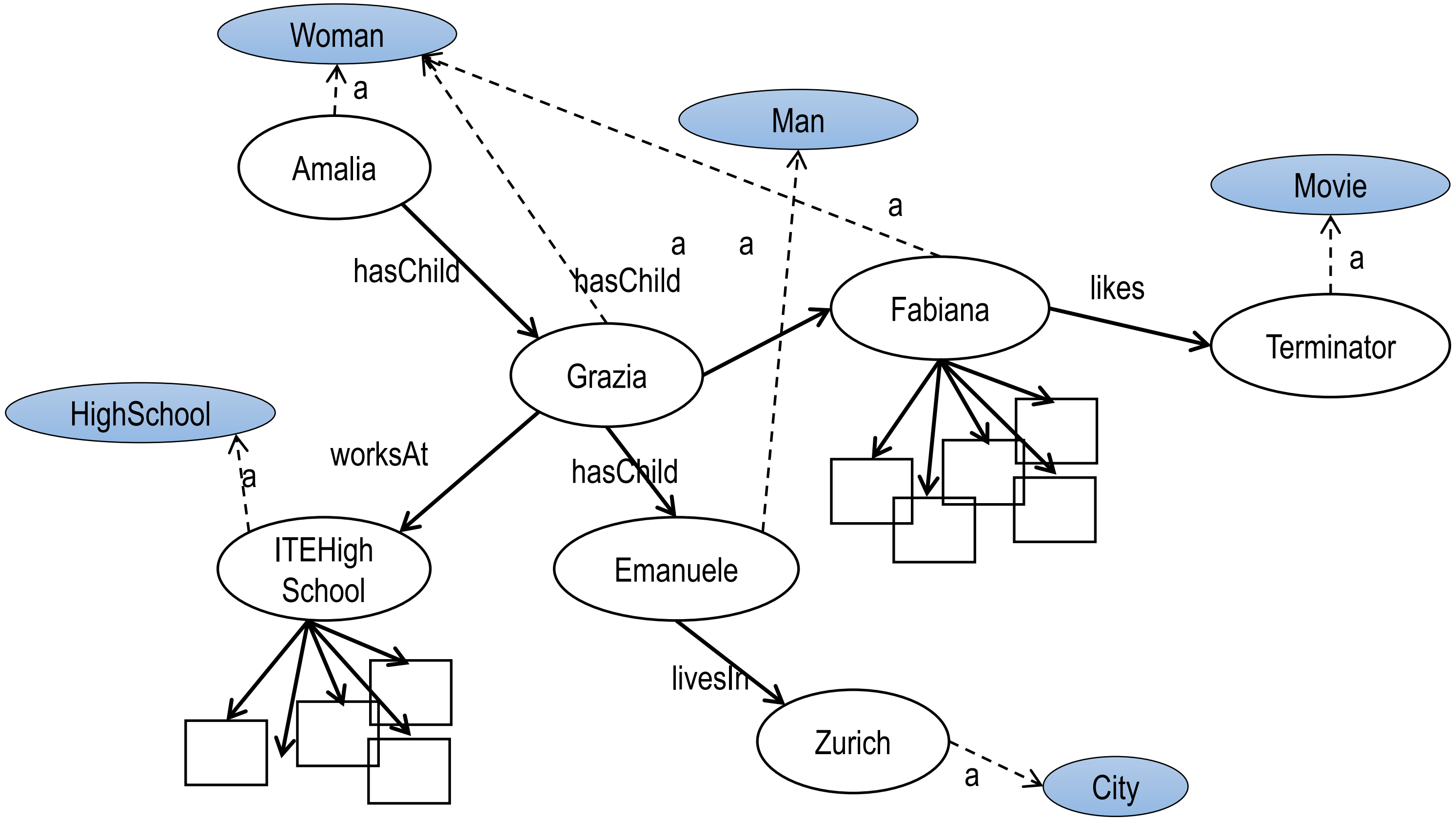
– Notice the semi-colon “;” after each row and the full-stop “.” after the last statement that refers to the subject “Emanuele”.

Graph Representation

- Where are the terms for nodes and edges from?



Entity Types
(WHAT are the things mentioned in the graph?)



Now we know that Terminator is a movie and not the online shop about electrical product.

https://terminator.ae/

Provenance of Terms in RDF Statements

Possible sources:

– the **creator of the graph can improvise all terms** (similarly to how we are free to decide what JSON fields or XML tags we can use).

– terms **can be picked from public sources** (public graphs, public ontologies).

– our own terms **can be freely combined** with terms from other provenances.

Where did we take this type from?

```
:Grazia      a          :Person;
             :worksAt   :ITEHighSchool;
             :hasChild  :Emanuele.
:Emanuele    :livesIn   :Zurich.
```

Where did we take these nodes from?

Where did we take these properties from?

Adapted from lecture of Prof. Dr. Buchmann

Prefix and Namespaces (URI or IRI)

No label. Default prefix.

Label that is associated to a URI and is called namespace prefix or simply prefix

Directive to associate a prefix with an URI

Namespace, URI, or IRI

@prefix : <http://laurenzi.ch#>.
@prefix dbr: <http://dbpedia.org/resource/>.

The **prefix declarations** is mandatory for all prefixes used in a graph.
The provenance of terms determines the namespace prefixes that shall be declared.

- Default prefix. Improvise our own terms and IDs , e.g.,
  - @prefix : <http://laurenzi.ch#>.
- Use an ID of a public source, e.g.,
  - @prefix dbr: <http://dbpedia.org/resource/>.
- You can also add a label to characterize your dataset, e.g.,
  - @prefix el: <http://laurenzi.ch#>.
    - This helps to identify a graph or its scope in the whole Web. E.g.,
      - in dbr:The_Terminator, *dbr* identifies the Graph of DBPedia, which describes Wikipedia pages.

Examples of Public Sources

- Schema.org – public ontology (we can take from there properties and types)
- DBPedia – public Knowledge Graph (offers Wikipedia information in graph form, we can also take from there IDs)

Example written with terms of varying provenance.

@prefix : <http://laurenzi.ch#>. (default prefix for my own terms)
#Alternative -> @prefix el: <http://laurenzi.ch#>. (prefix "el"for my own terms)
@prefix s: <http://schema.org/>. (prefix for terms from Schema.org)
@prefix dbr: <http://dbpedia.org/resource/>. (prefix for DBPedia terms)

:Emanuele a s:Person; → https://schema.org/Person
s:worksFor dbr:University_of_Applied_Sciences_and_Arts_Northwestern_Switzerland;
s:homeLocation dbr:Zürich.

https://schema.org/worksFor
https://schema.org/homeLocation
https://dbpedia.org/page/Zürich
https://dbpedia.org/page/University_of_Applied_Sciences_and_Arts_Northwestern_Switzerland

Deferencing

- **Dereferencing** = accessing the URL address obtained from **namespace + local ID/term**
- It can return something useful about a term, typically:
  - A Webpage about the term
  - A subgraph with all information available about the term
  - Nothing at all because it's a **global identifier** and not an address.

For example:

-> http://laurenzi.ch/ + Emanuele = http://Laurenzi.ch/Emanuele

It returns nothing.

@prefix : <http://laurenzi.ch#>.

@prefix dbr: <http://dbpedia.org/resource/>.

@prefix s: <http://schema.org/>.

:Emanuele **s:**homeLocation **dbr:**Zürich.

-> http://dbpedia.org/resource/ + Zürich = http://dbpedia.org/resource/Zürich

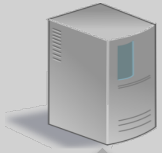
If you try to access it in a browser (through HTTP) it returns information about the city of Zurich.

Querying an RDF Graph

Queries are mostly about navigating the graph in search of some patterns.

Knowledge-Representation and Reasoning

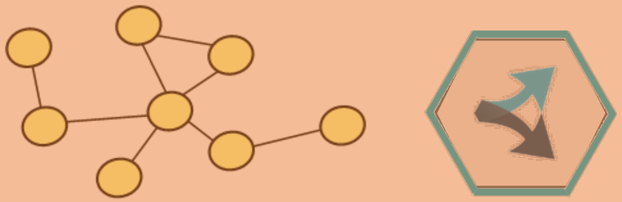
Graph Database/Triplestore and inference engine



Storing, integration, querying, reasoning.

consistent

Knowledge representation Formalism



machine-interpretable knowledge

RDF, RDF(S), OWL, SPARQL, SHACL…

consistent

Representation of reality

Not formally represented

Adapted from lecture of Prof. Dr. Knut Hinkelmann

SPARQL Query Syntax

A World Wide Web Consortium (W3C) standard.

SPARQL similar to select-from-where syntax (like SQL):

- *PREFIX*: prefix information. The left-hand side of ":" can contain an acronym to denote the name of the Turtle file.

```
prefix
      : <http://www.fhnw.ch#>
```

- *SELECT*: the entities (variables) you want to return.

```
select ?X ?Y ?A
```

- *WHERE*: the (sub)graph you want to get the information from.

```
where  { ?X friend ?Y. ?Y age ?A.
```

- additional constraints on objects, using operators.

```
          FILTER  ?A > 25. }
```

From lecture of Prof. Dr. Holger Wache

SPARQL

- It provides facilities to:
  - Extract information in the form of URIs, blank nodes, plain and typed literals.
  - Extract RDF subgraphs.
  - Construct new RDF graphs based on information in the queried graphs.
- Feature
  - Matching graph patterns.
  - Query terms – based on Turtle syntax.
  - Terms delimited by "<>" are relative URI references.
  - Data description format – Turtle.

From lecture of Prof. Dr. Holger Wache

Popular SPARQL Forms

- SELECT
  - returns all, or a subset of the variables bound in a query pattern match.
- CONSTRUCT
  - returns an RDF graph constructed by substituting variables in a set of triple templates.

Our focus

- DESCRIBE
  - returns an RDF graph that describes the resources found.
- ASK
  - returns whether a query pattern matches or not.

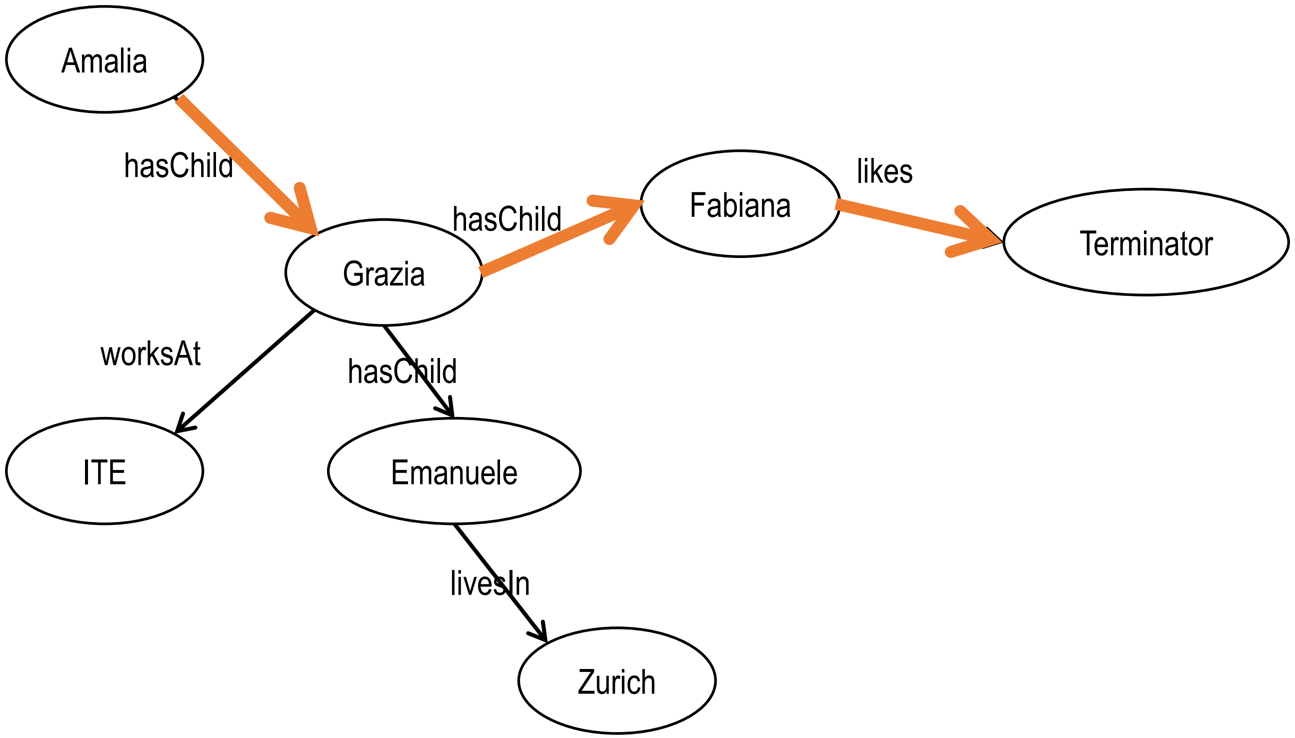
From lecture of Prof. Dr. Holger Wache

Hands-on with SPARQL

- Download zip familytree_without_schema.zip -> unzip it
- Launch GraphDB
- Create a graph database Family Tree
- Load .ttl file familytree_without_schema to GraphDB

SELECT

to 1. navigate along a graph path, 2. unknown length and 3. navigate a path in reverse



1. Navigating along a graph path

SELECT ?x WHERE
{:Amalia **:hasChild/:hasChild/:likesMovie** ?x}
(what does Amalia's grandchild like?)

2. Navigating a path of unknown length

SELECT ?x WHERE
{:Amalia **:hasChild+/:likesMovie** ?x}
(what do all Amalia's descendants like?)

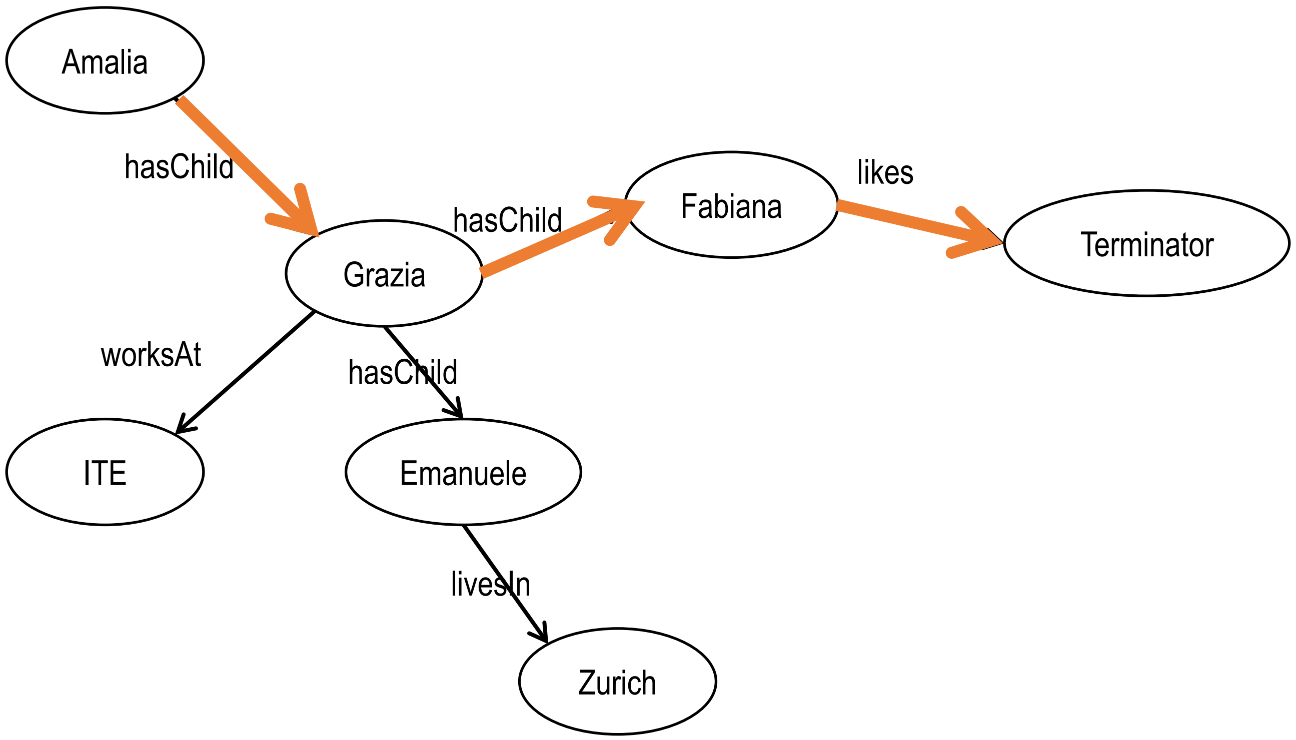
3. Navigating a path in reverse:

SELECT ?x WHERE
{?x **^:likesMovie/^:hasChild/^:hasChild** :Amalia}

Adapted from lecture Prof. Dr. Buchmann

SELECT

to 4. navigate a path and 5. to retrieve potential missing information



4. Step-wise navigation of a path
(it can retrieve intermediate nodes and edges)

```
SELECT ?a ?b ?x WHERE
{:Amalia     :hasChild   ?a.
?a           :hasChild   ?b.
?b           :likesMovie ?x}
```

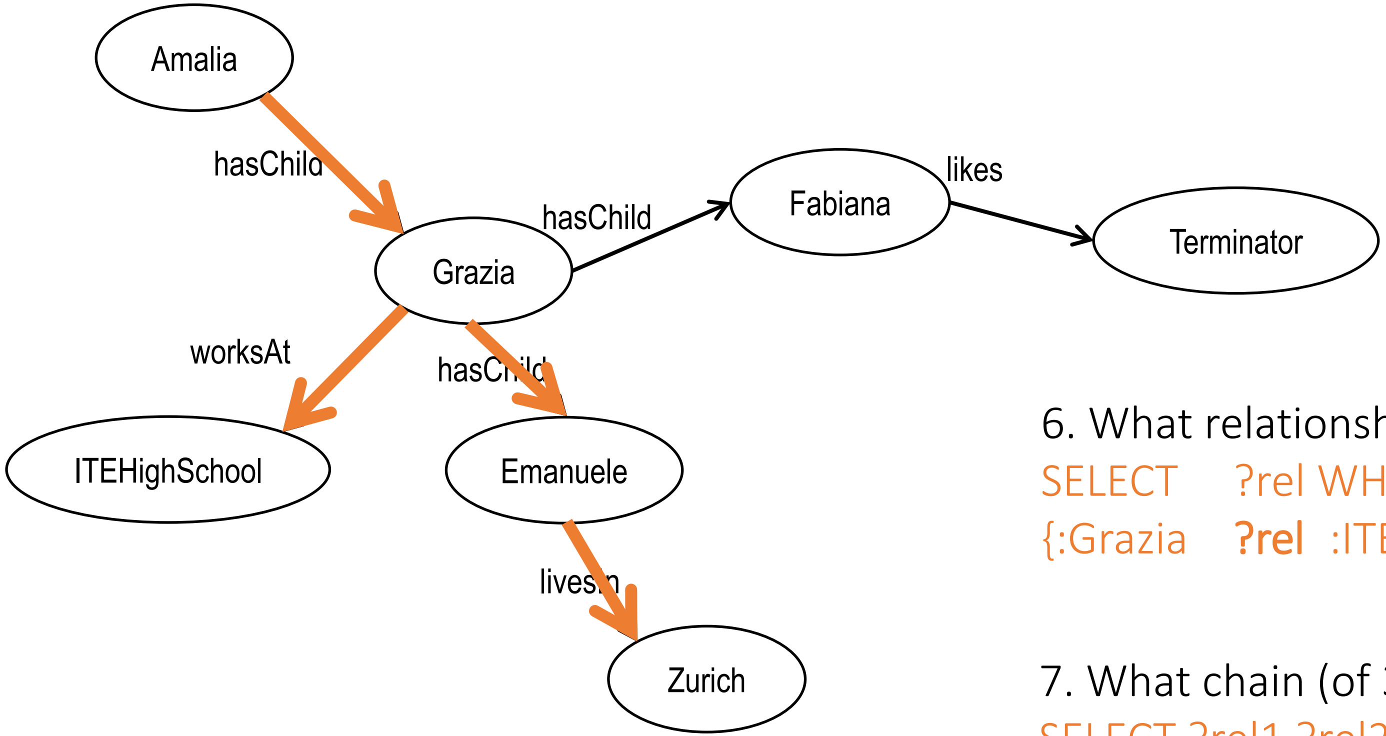
**the decomposition into statements complies with the Turtle syntax*

5. Retrieving potentially missing information
(give me all parents and, IF AVAILABLE, their work place)

```
SELECT ?a ?c WHERE
{?a  :hasChild  ?b.
OPTIONAL
      {?a  :worksAt  ?c}}
```

Adapted from lecture Prof. Dr. Buchmann

SELECT to discover relationships



6. What relationship exists between Grazia and ITEHighSchool?

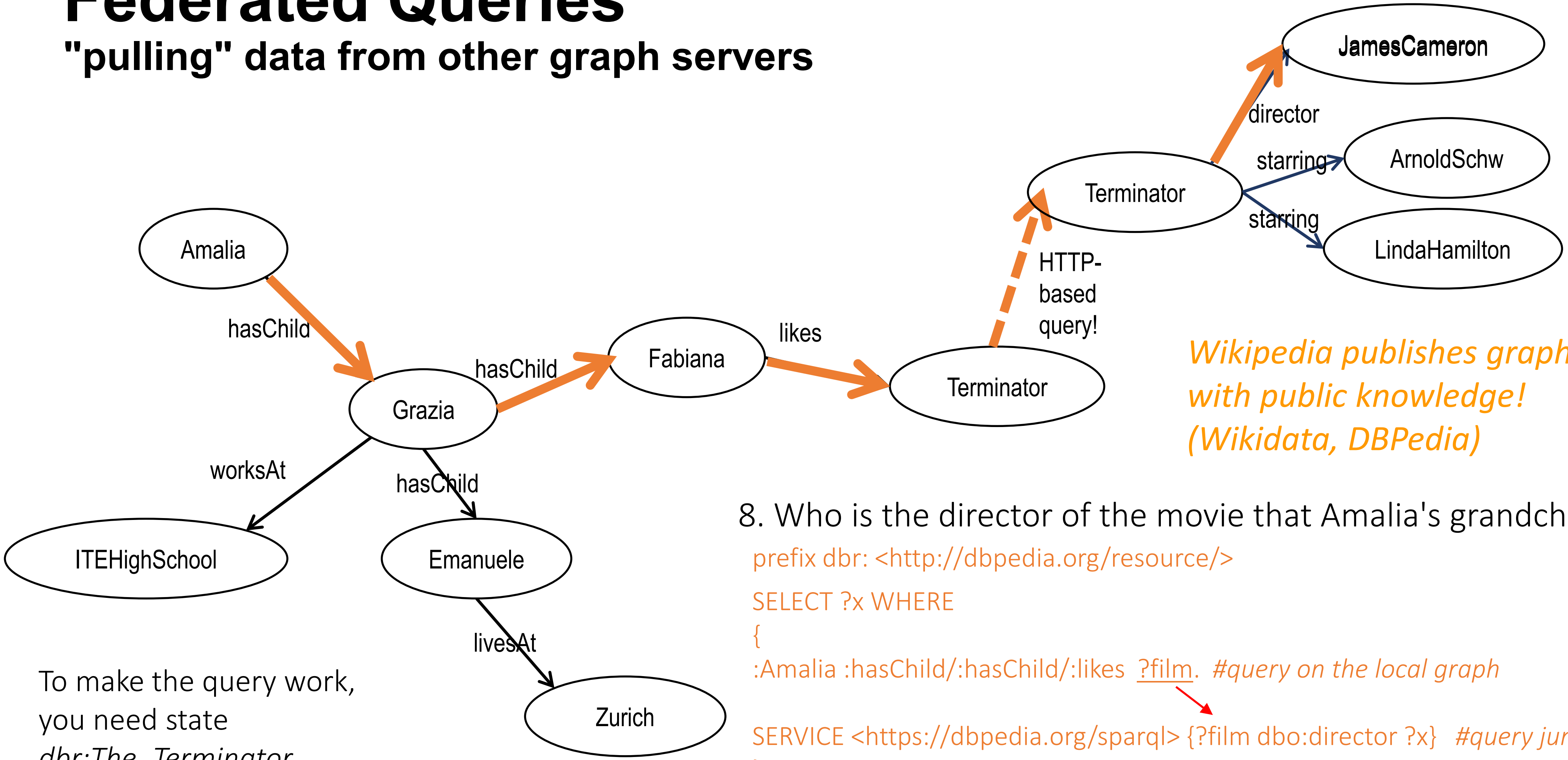
SELECT ?rel WHERE
{:Grazia **?rel** :ITEHighSchool}

7. What chain (of 3 relationships) exists between Amalia and Zurich?

SELECT ?rel1 ?rel2 ?rel3 WHERE
{:Amalia **?rel1** ?intermediary1.
?intermediary1 **?rel2** ?intermediary2.
?intermediary2 **?rel3** :Zurich}

Adapted from lecture Prof. Dr. Buchmann

Federated Queries
"pulling" data from other graph servers



Wikipedia publishes graphs with public knowledge! (Wikidata, DBPedia)

8. Who is the director of the movie that Amalia's grandchild likes?

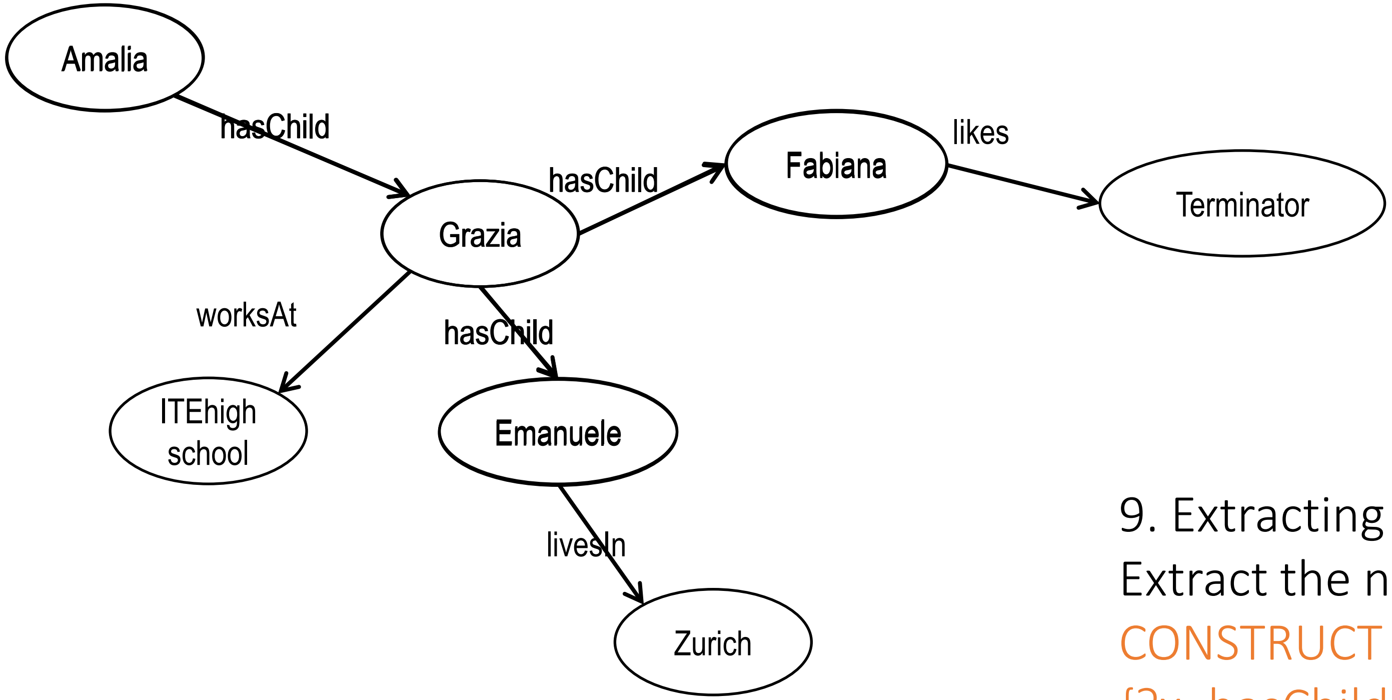
```
prefix dbr: <http://dbpedia.org/resource/>
SELECT ?x WHERE
{
:Amalia :hasChild/:hasChild/:likes  ?film.  #query on the local graph

SERVICE <https://dbpedia.org/sparql> {?film dbo:director ?x}   #query jumps to public graph
}
```

To make the query work, you need state *dbr:The_Terminator* in your dataset.

Adapted from lecture Prof. Dr. Buchmann

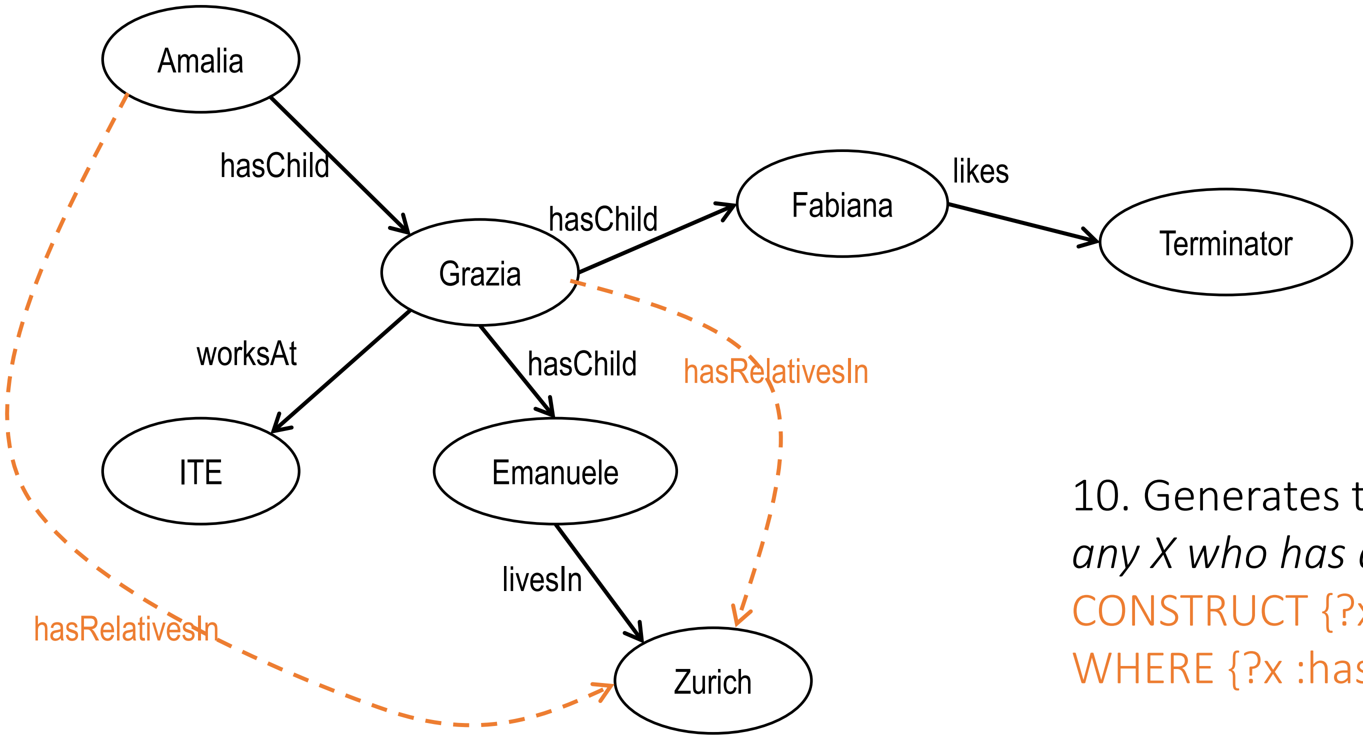
CONSTRUCT to extract a subgraph



9. Extracting a subgraph
Extract the network of child relationships
CONSTRUCT WHERE
{?x :hasChild ?y}

Adapted from lecture Prof. Dr. Buchmann

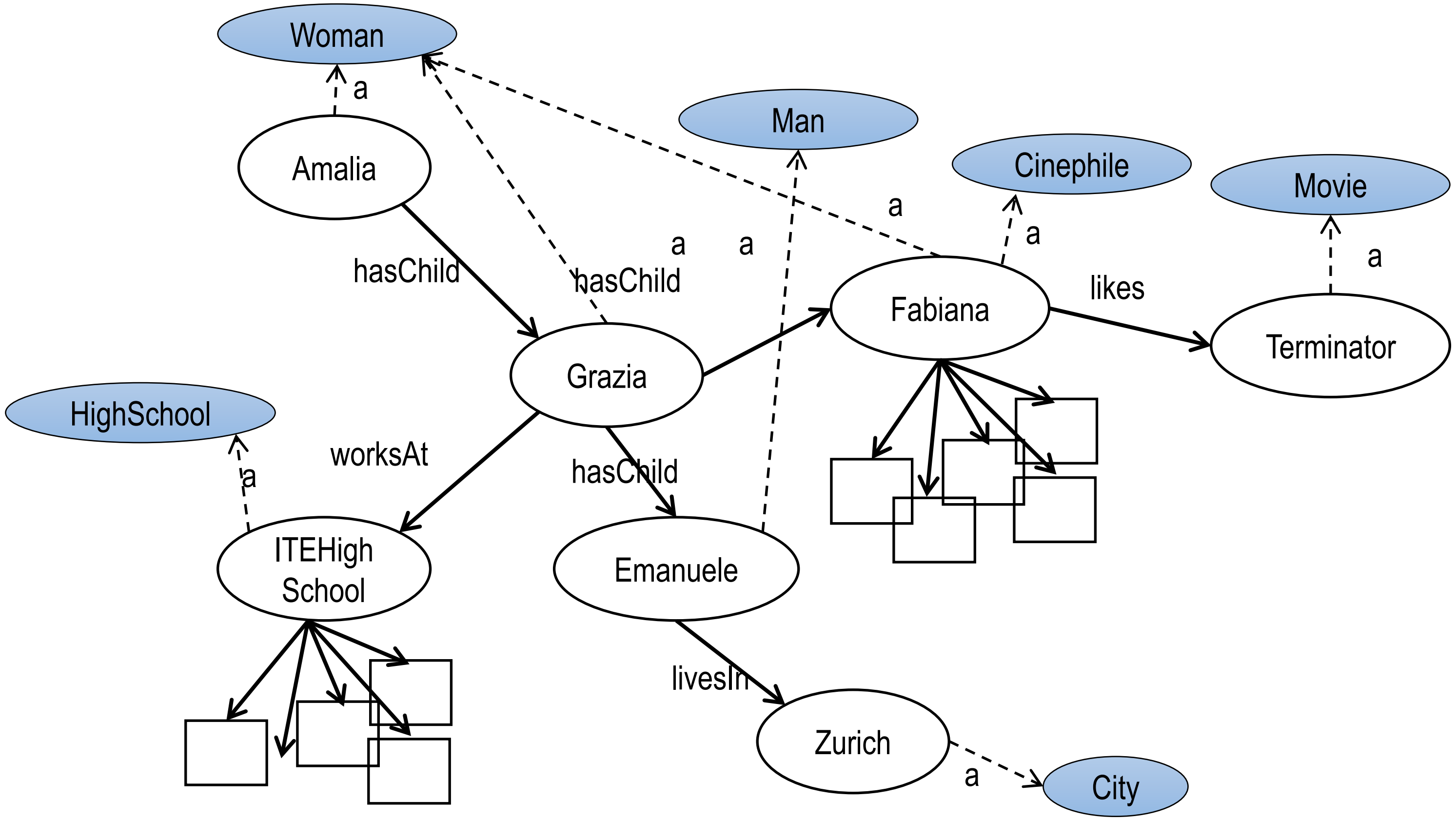
CONSTRUCT for Machine Reasoning
supported directly in the query language!



10. Generates the statement "X hasRelatives in Y" for *any X who has descendants living in Y*:
CONSTRUCT {?x :hasRelativesIn ?y}
WHERE {?x :hasChild+/:livesIn ?y}

Adapted from lecture Prof. Dr. Buchmann

Entity Types
(WHAT are the things mentioned in the graph?)



Types can be attached to the nodes:

- a standard relationship denoted "a" is used for this purpose.
- some types can be generated through deductive reasoning, e.g.:
  - *11. if X likes Terminator, she/he is a Cinephile:*

CONSTRUCT {?x a :Cinephile}
WHERE {?x :likes dbr:The_Terminator}

  - *12. if X likes movies, she/he is a Cinephile:*

CONSTRUCT {?x a :Cinephile}
WHERE {?x :likes/a :Movie}