Knowledge Graphs and SPARQL

Agenda

1. A general introduction into RDF
2. Why SPARQL?
3. How to use SPARQL
4. Hands-on

Basics on RDF

RDF (Resource Description Framework):

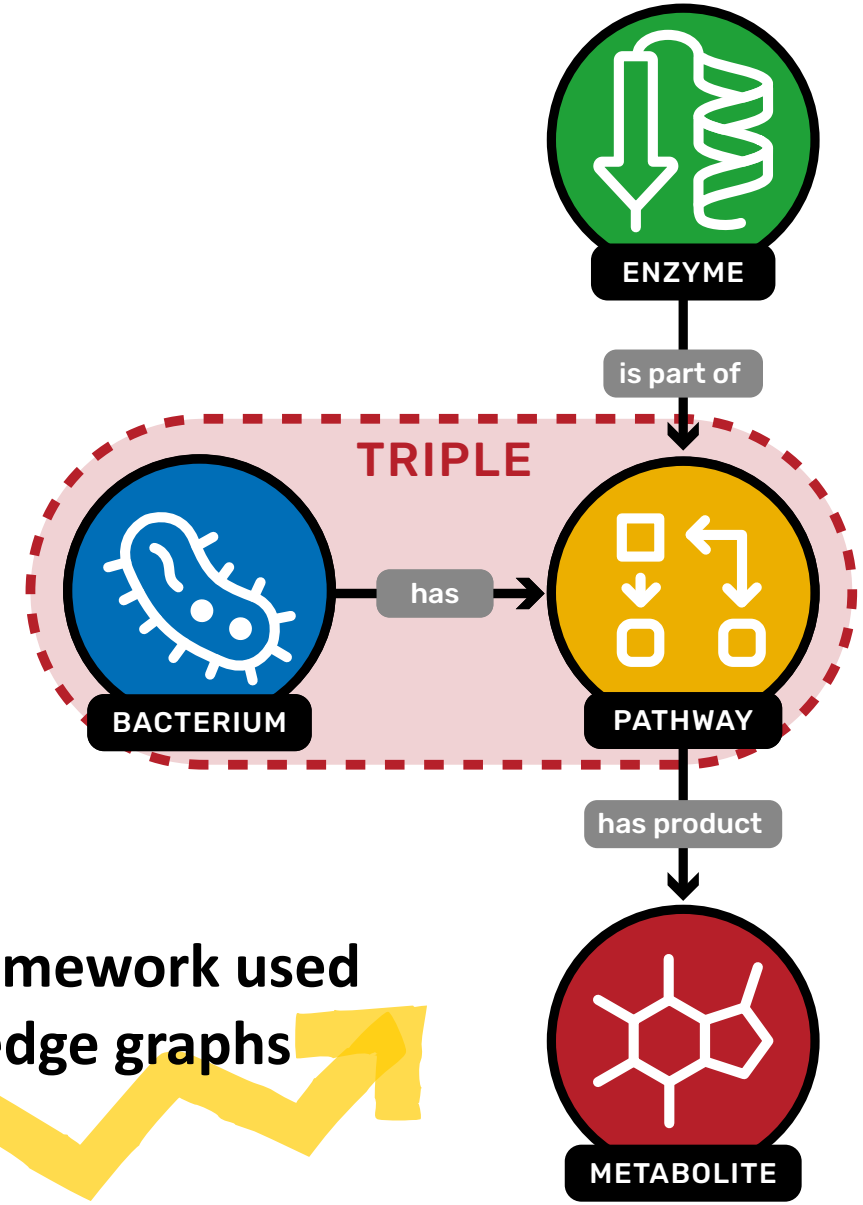
- A framework for describing and interlinking data

Core Concept – Triples:

- Data is represented as triples:
  Subject - Predicate - Object

Graph-based Representation

- Triples are joined to form a graph
- Machine-Readable and Semantic



RDF is the framework used for knowledge graphs

Basics on Knowledge Graphs

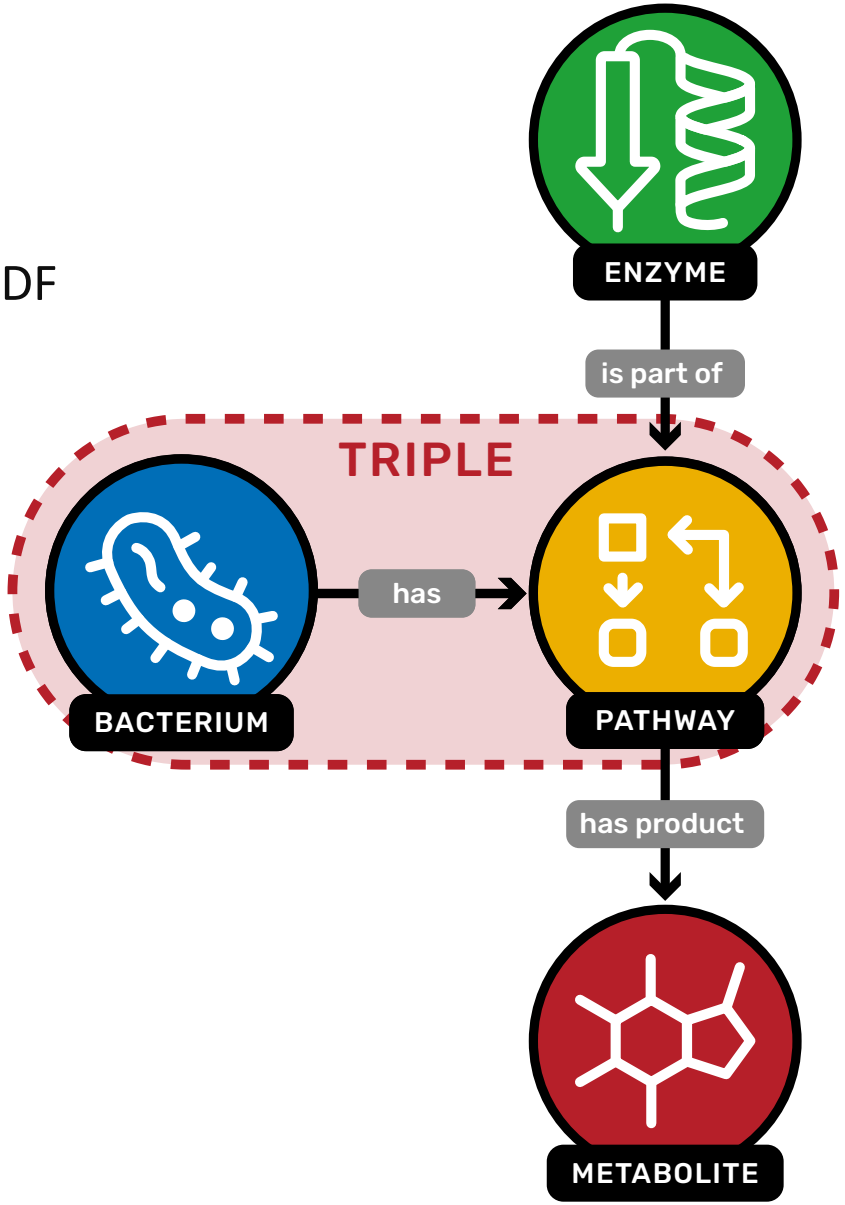
A network of entities and their relationships modelled in RDF

Graph Structure:

- Nodes represent entities (e.g., strains, culture media).
- Edges represent relationships (e.g., “grows on”).

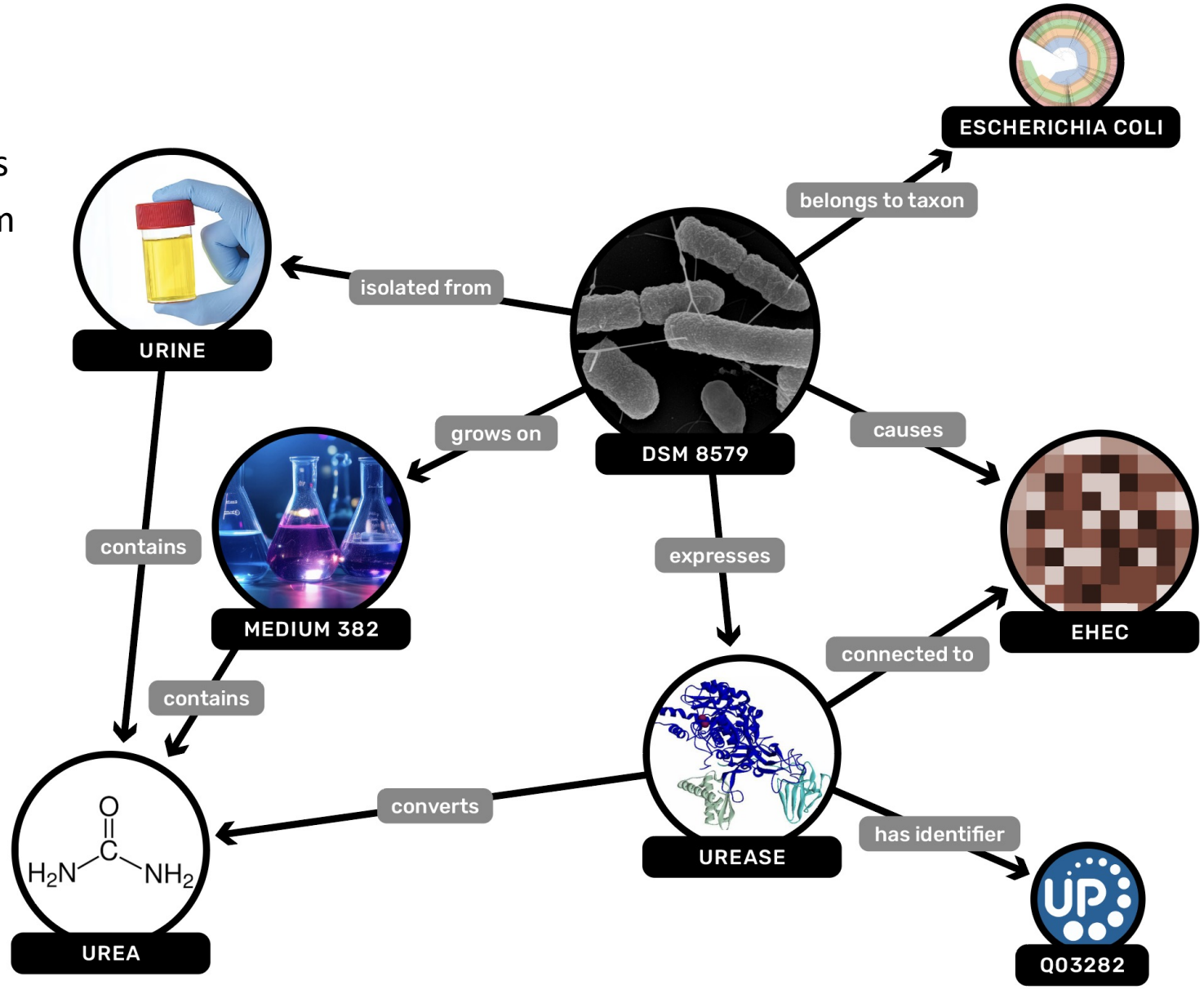
Knowledge comes from ontologies

SPARQL is the language for querying a knowledge graph



Knowledge Graphs

- A simple graph with 10 triples
- Information on one bacterium



Knowledge Graphs

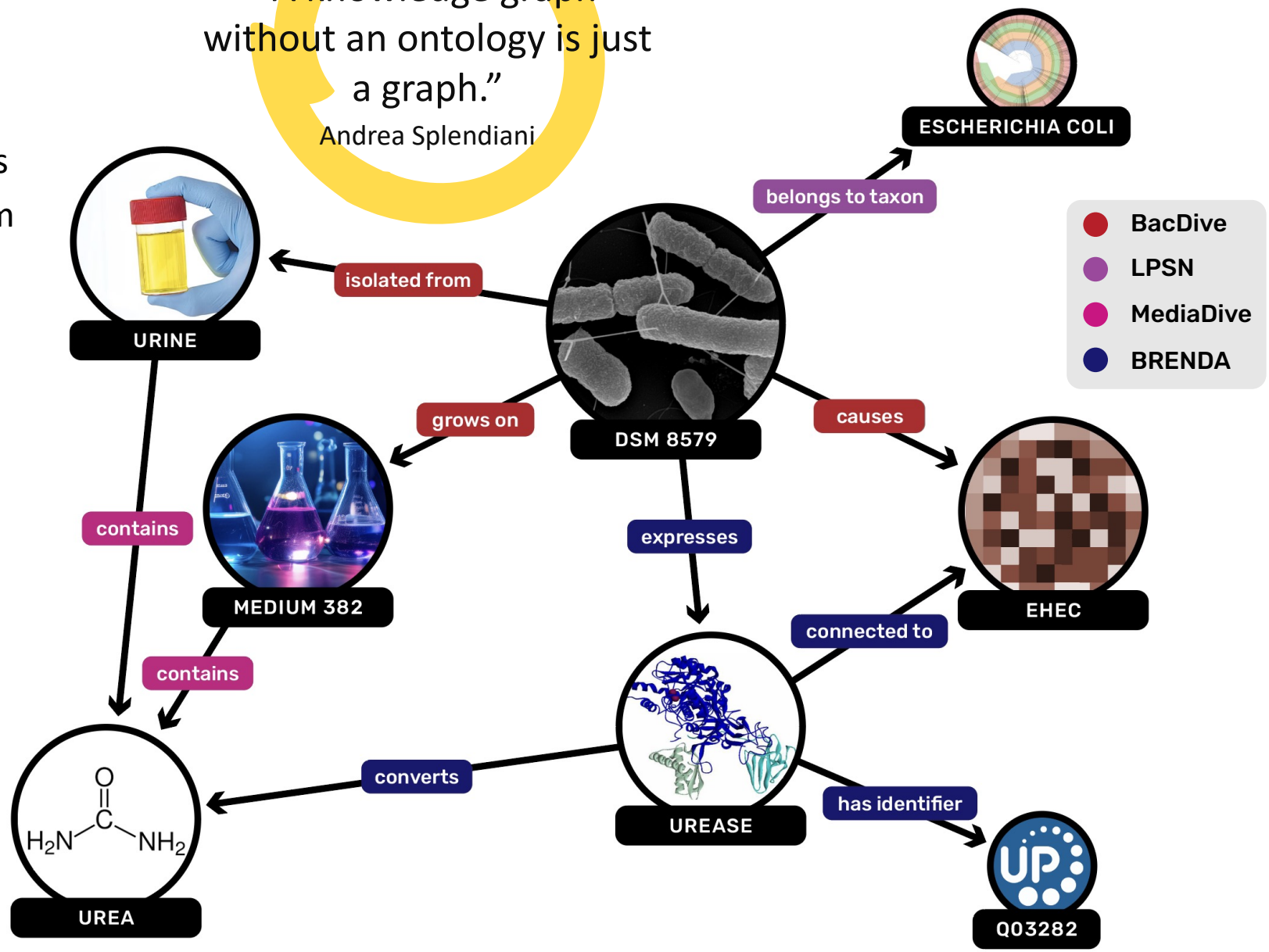
"A knowledge graph without an ontology is just a graph."
Andrea Splendiani

- A simple graph with 10 triples
- Information on one bacterium

Data Integration

- Triples can originate from different databases!

Which entity is the bacterium?



Knowledge Graphs

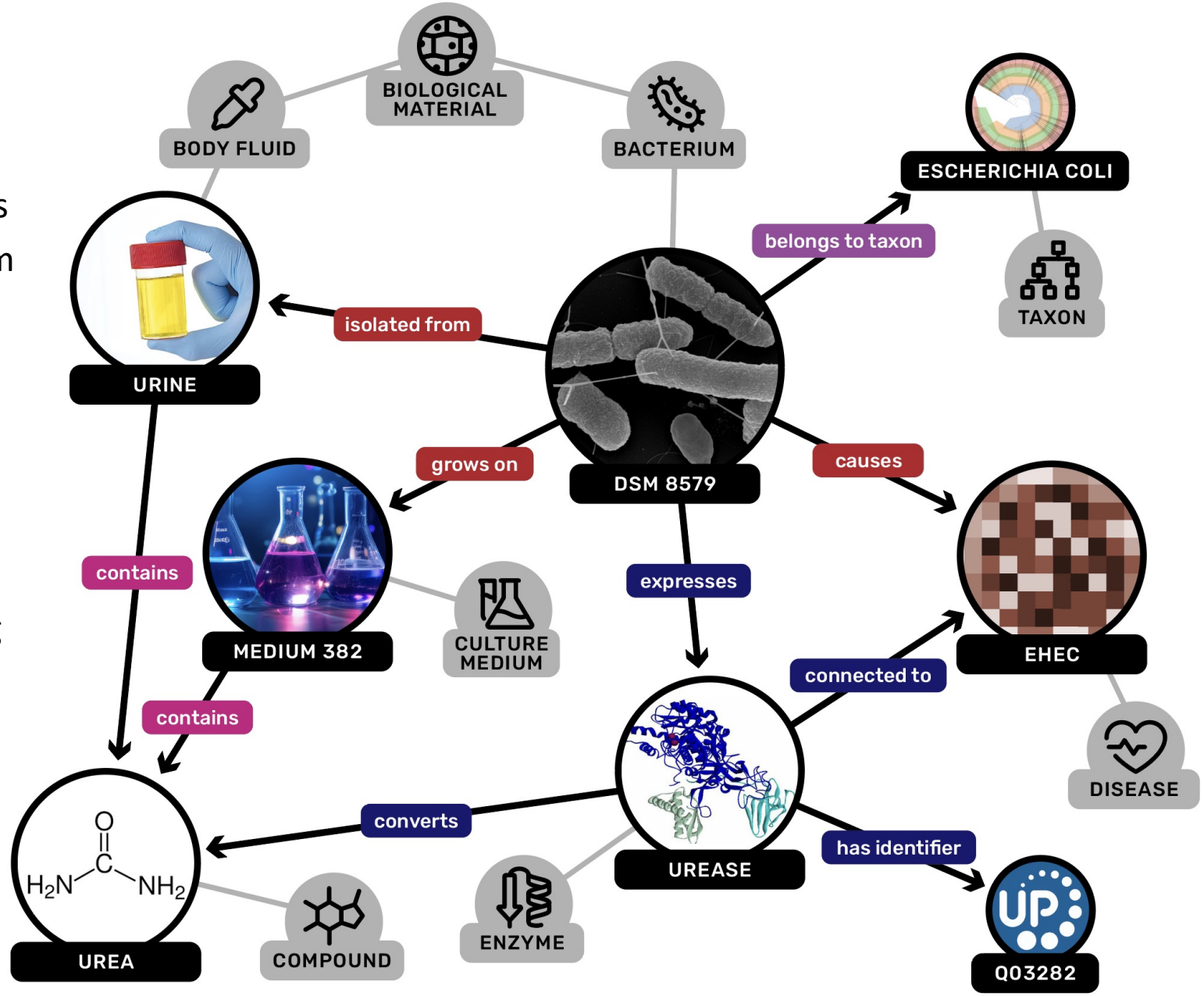
- A simple graph with 10 triples
- Information on one bacterium

Data Integration

- Triples can originate from different databases!

Ontologies

- Ontology terms give meaning to triples
- „Organizing principles“



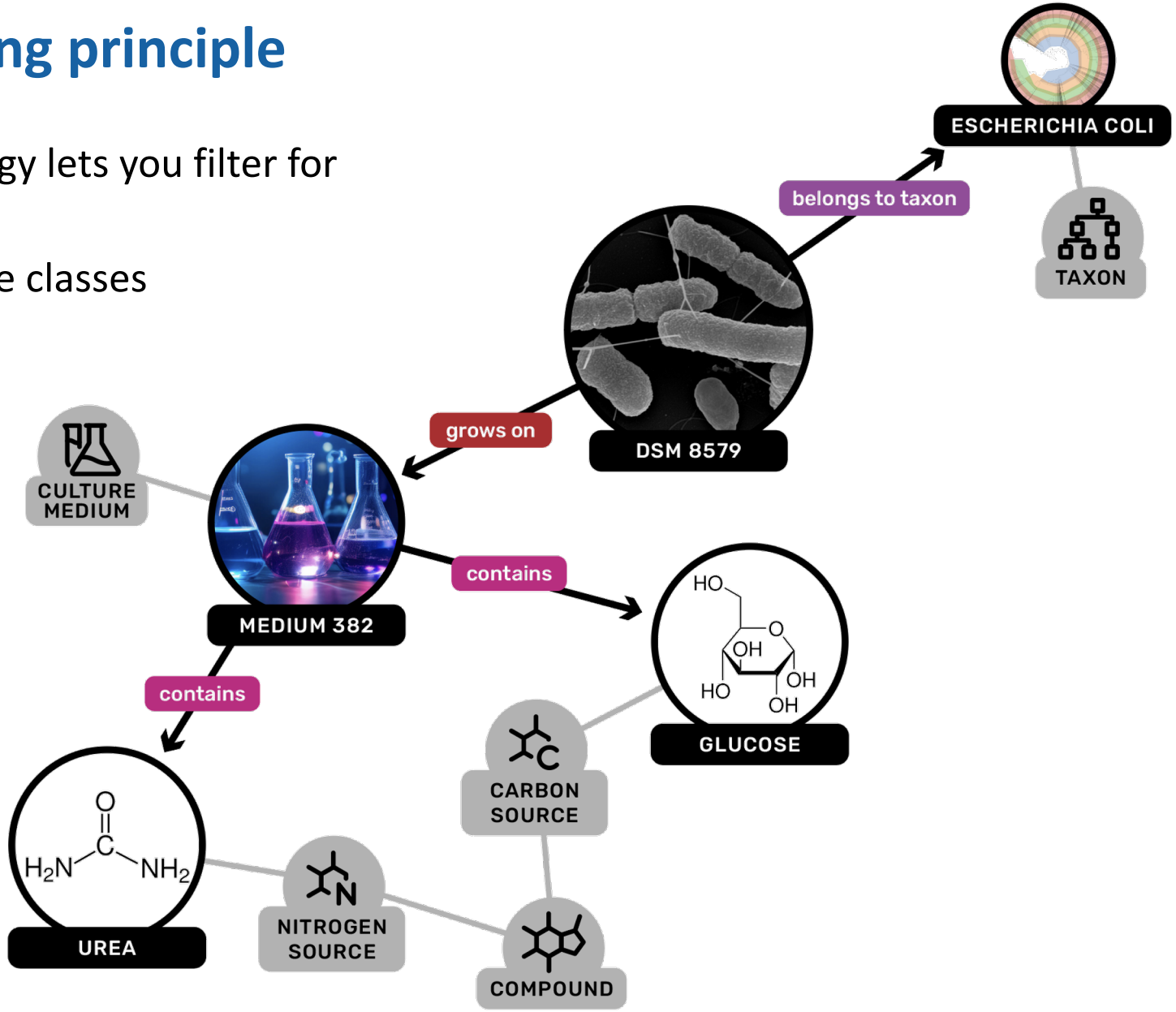
The ontology as guiding principle

- Adding depth to an ontology lets you filter for specific characteristics
- An entity can have multiple classes

Example question:

What are possible nitrogen sources used in media of *E. coli* strains?

Possible with SPARQL

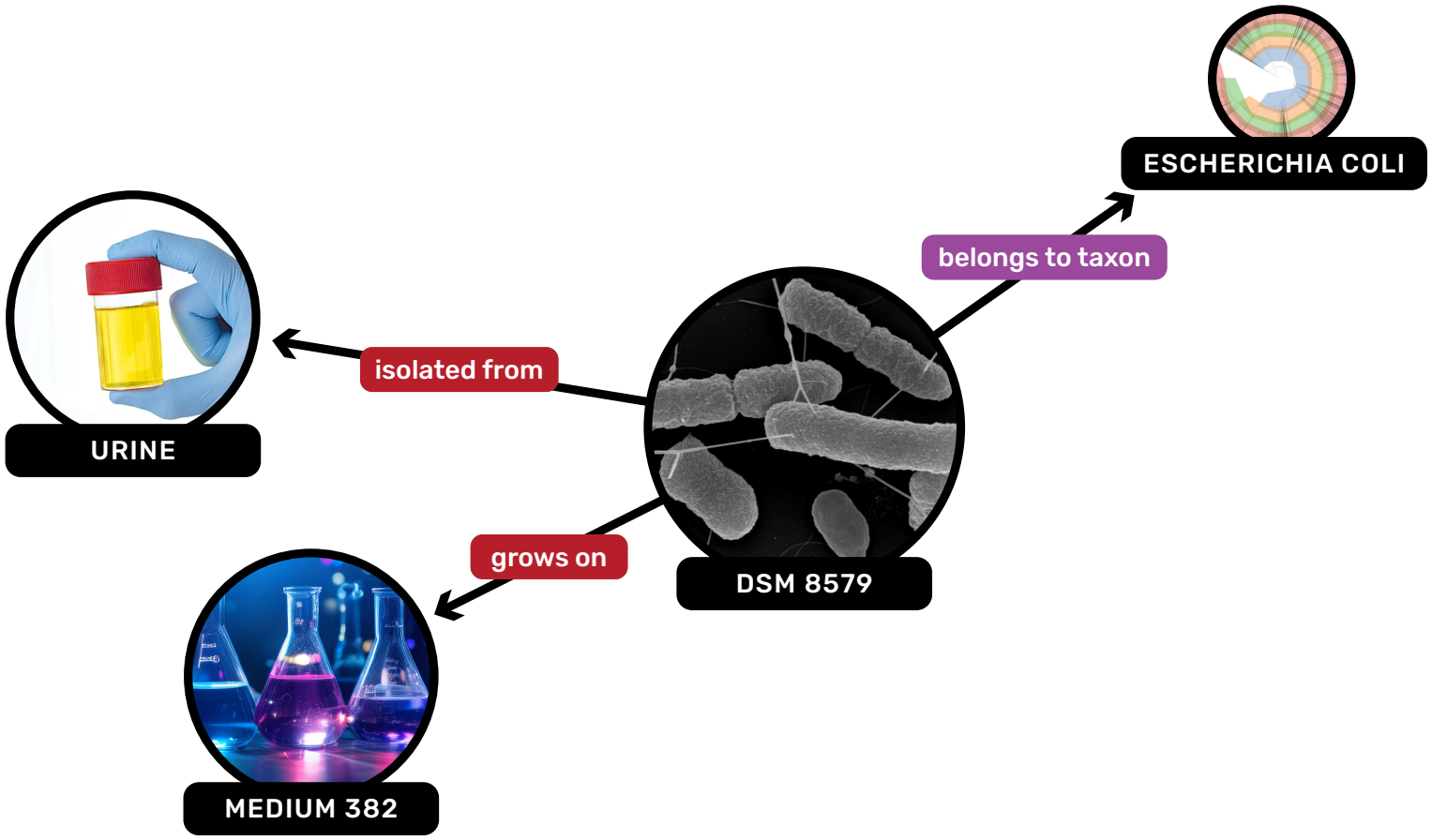


We need unique identifiers!

- to link entities with each other
- to avoid ambiguity
- to be machine-readable
- to be FAIR

RDF is a web standard

→ URIs are used as identifiers



URI for identifying entities

Uniform Resource Identifiers (URIs) uniquely identify resources

`https://doi.org/10.1093/nar/gkae959`

- A link that serves as an identifier
- Don’t have to be human readable
- Redirect/resolve to an entity

“Speaking identifier”

Do we simply use the URL of the resource?

`https://bacdive.de/strain/12345`

PURL

- We use a **PURL** (Persistent URL) instead of the „real“ URL of the resource

Reasons:

1. An entity can have multiple URLS:
   - https://bacdive.de/strain/12345
   - https://**www**.bacdive.de/strain/12345
   - **http**://bacdive.de/strain/12345
   - https://bacdive.**dsmz.**de/strain/12345
2. A URL can contain special characters that are not allowed:
   - https://www.brenda-enzymes.org/enzyme.php?ecno=3.5.1.5
3. A URL might change over time
   - Renaming of a database
   - New backend system and new routing

https://purl.dsmz.de/bacdive/strain/12345

https://purl.dsmz.de/strain/bacdive:12345

Why SPARQL?

- Standardized Query Language for RDF W3C®
- Flexibility for Complex Queries
- Machine-Readable and Reproducible
- Link data from different databases (e.g., BacDive, MediaDive, BRENDA)
- Discover hidden connections and generate new insights

- SPARQL endpoints are generally open

Comparison with traditional APIs

	API	SPARQL
Dynamic and Flexible	Restricted to the structure and filters provided by the API	Fully customizable queries; users can explore data beyond predefined options
Querying Relationships	Typically limited to predefined endpoints or requires multiple calls for complex queries	Easily retrieves data across complex relationships
Cross-Dataset Integration	Requires custom scripts or manual integration between API results	Queries data from multiple sources in a unified way
Reproducibility	Scripts depend on specific API versions, which might change over time	Queries are reusable and version-controlled
Data Context and Semantics	Typically lack semantic awareness and contextual relationships	Leverages semantic data and ontologies for richer queries

How do I access a database with SPARQL?

yummydata.org

Introduction to the SPARQL syntax

Please find example queries and some instructions here:

LeibnizDSMZ/d3-sparql

d3-sparql / queries / introduction.md

Klick here

Getting started: SPARQL 101

Basic syntax:

```
PREFIX d3o: <https://purl.dsmz.de/schema/>

SELECT ?strain ?genus WHERE {
    ?strain a d3o:Strain .
    ?strain d3o:hasGenus ?genus .
}
LIMIT 5
```

Definition of prefixes

Select all variables (start with ?)

A triple, ending with a dot.

Another triple reusing the same variable

Limit to the first 5 results

PREFIX is used to simplify the queries:

```
d3o:Strain
<https://purl.dsmz.de/schema/Strain>
```

Variables are used as wildcards:

```
?strain a d3o:Strain .
Any of type strain .
```

Getting started: SPARQL 101

Getting started #1

```
PREFIX d3o: <https://purl.dsmz.de/schema/>
SELECT * WHERE {
    ?strain a d3o:Strain .
    ?strain d3o:hasGenus ?genus .
}
LIMIT 5
```

	?strain	?genus
1	1	Acetobacter
2	10	Acetobacter
3	100	Phascolarctobacterium
4	1000	Bacillus
5	10000	Myxococcus

<https://purl.dsmz.de/bacdive/strain/10000>

Adding a label and simplifying the query


```
PREFIX d3o: <https://purl.dsmz.de/schema/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
    ?strain a d3o:Strain ;
            rdfs:label ?label ;
            d3o:hasGenus ?genus .
}
```

	?strain	?label	?genus
1	1	Acetobacter aceti 1	Acetobacter
2	10	Acetobacter estunensis 10	Acetobacter
3	100	Phascolarctobacterium faecium 100	Phascolarctobacterium
4	1000	Bacillus subtilis 1000	Bacillus
5	10000	Myxococcus xanthus 10000	Myxococcus

Combine two different fields

Combining optional values #1

```
SELECT ?strain ?label (CONCAT(?species, " ", ?des) AS ?strainName)
WHERE {
    ?strain a d3o:Strain ;
        rdfs:label ?label ;
        d3o:hasSpecies ?species .
    OPTIONAL { ?strain d3o:hasDesignation ?des . }
} LIMIT 5
```

	?strain	?label	?strainName
1	1	Acetobacter aceti 1	Acetobacter aceti B 4114
2	10	Acetobacter estunensis 10	Acetobacter estunensis
3	100	Phascolarctobacterium faecium 100	Phascolarctobacterium faecium QUM 3679
4	1000	Bacillus subtilis 1000	Bacillus subtilis FR1-49con
5	10000	Myxococcus xanthus 10000	Myxococcus xanthus Mx x20

Filtering by text content

Filtering #1

```
SELECT ?strain ?name ?origin WHERE {
    ?location a d3o:LocationOfOrigin ;
        d3o:describesStrain ?strain ;
        d3o:hasCountry ?country ;
                rdfs:label ?origin .

    ?strain a d3o:Strain ;
            rdfs:label ?name .

    FILTER (CONTAINS(?country, "Germany"))
}
```

	?strain	?name	?origin
1	1000	Bacillus subtilis 1000	Friedrichshafen, European Aeronautic Defense and[...]
2	10000	Myxococcus xanthus 10000	Siebengebirge, Germany
3	10001	Myxococcus xanthus 10001	Siebengebirge, Germany
4	100040	Morganella morganii 100040	Schwellenburg Hill near Erfurt, Germany
5	100052	Morganella morganii 100052	valley of the Altmühl River near Badanhausen, Ge[...]

Filtering by numeric content and order results

Sorting #1

```
SELECT ?strain ?name ?temperature  WHERE {
    ?tempObj a d3o:CultureTemperature ;
        d3o:describesStrain ?strain ;
        d3o:hasTemperatureRangeStart ?temperatureStart ;
        d3o:growthObserved 'positive' ;
        d3o:hasTestType 'optimum' ;
        rdfs:label ?temperature .

    ?strain a d3o:Strain ;
            rdfs:label ?name.

    FILTER (?temperatureStart > 30)
}
ORDER BY ?temperatureStart
```

	?strain	?name	?temperature
1	1976	Mycetohabitans rhizoxinica 1976	30.5
2	1977	Mycetohabitans endofungorum 1977	30.5
3	13155	Pseudomonas knackmussii 13155	30.5
4	8581	Mycobacterium vulneris 8581	30.5
5	17835	Celerinatantimonas diazotrophica 17835	31
6	166389	Maioricimonas rarisocia 166389	31
7	23078	Listeria weihenstephanensis 23078	31
8	24748	Amorphus suaedae 24748	31
9	132144	Comamonas guangdongensis 132144	31

Aggregation

Aggregation #1 + #2

```
SELECT (COUNT(?strain) AS ?count) WHERE {
    ?strain a d3o:Strain .
}
```

?count
97,334

```
SELECT (AVG(?temperatureStart) AS ?averageTemperature) WHERE {
    ?tempObj a d3o:CultureTemperature ;
        d3o:describesStrain ?strain ;
        d3o:hasTemperatureRangeStart ?temperatureStart ;
        d3o:growthObserved 'positive' ;
        d3o:hasTestType 'optimum' .
}
```

?averageTemperature
30.3497

Grouping

Grouping #1

```
SELECT (COUNT(?strain) AS ?count) WHERE {
    ?strain a d3o:Strain .
}
```

?count
97,334

```
SELECT ?genus (COUNT(?strain) AS ?count) WHERE {
    ?strain a d3o:Strain ;
            d3o:hasGenus ?genus .
}
GROUP BY ?genus
```

?genus	?count
Abditibacterium	1
Abiotrophia	39
Absicoccus	1
Abyssalbus	1
Abyssibacter	1
Abyssibius	1

Questions

- How many strains are from a location (near or) in Braunschweig?
- What is the lowest growth pH of a strain isolated from soil?
- What is the first (lowest) BacDive-ID of a strain from the CIP collection?
  - FILTER(STRSTARTS(?variable, "prefix"))