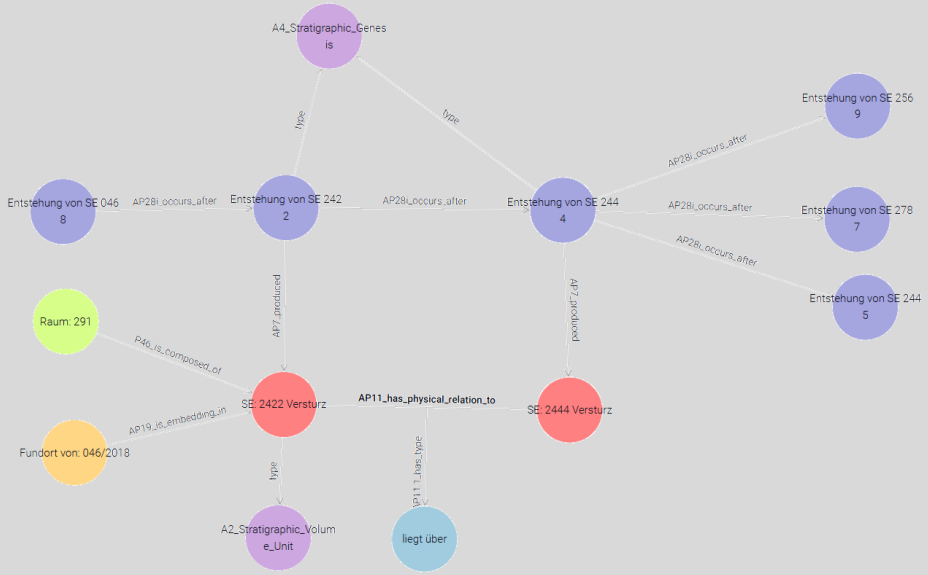
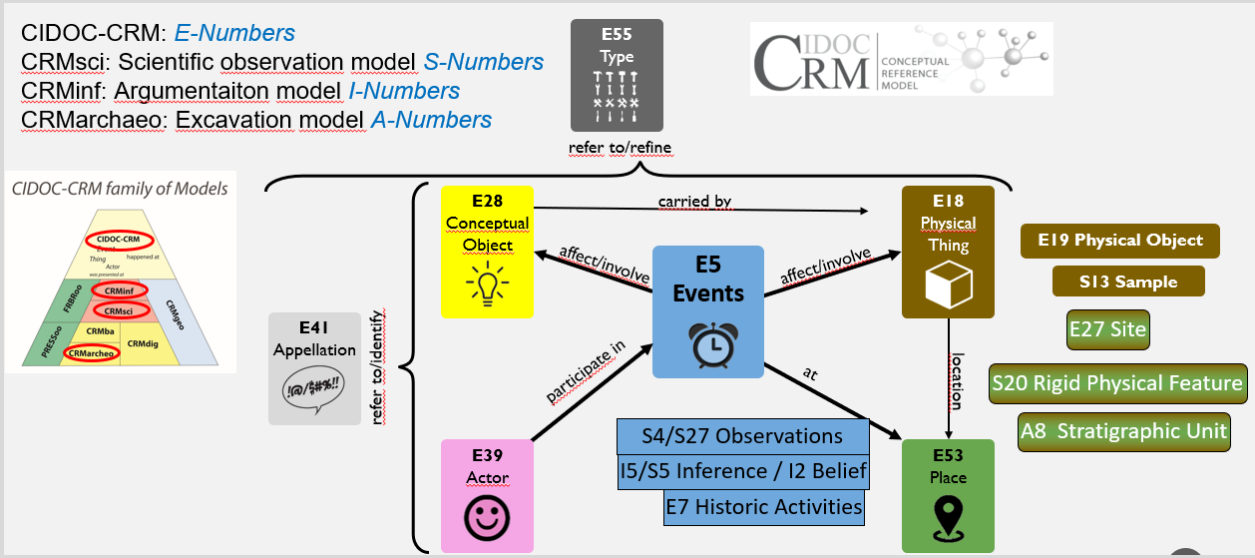
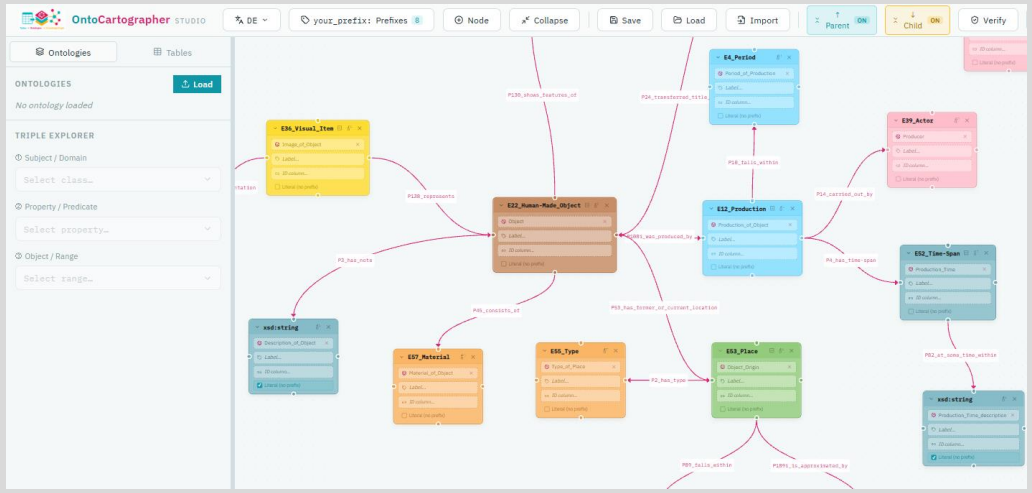


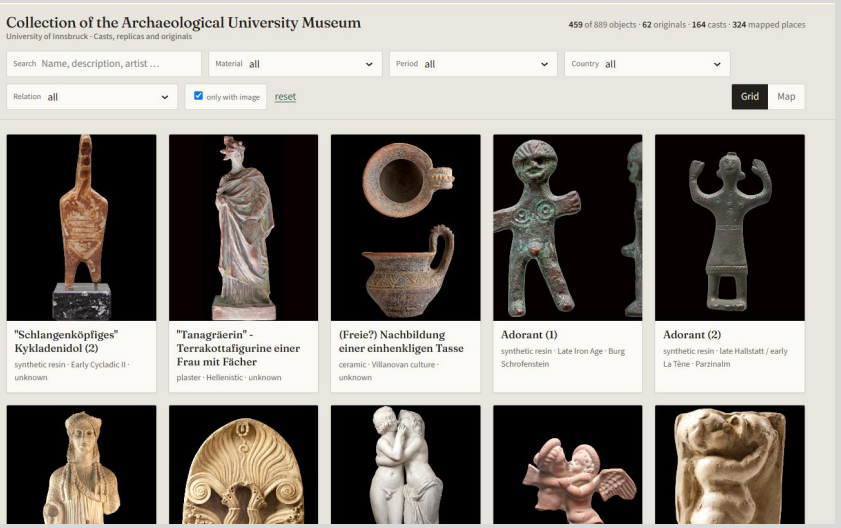


AI ASSISTED GENERATION & QUERY OF KNOWLEDGE GRAPHS



GERALD HIEBEL

BRIGIT DANTHINE



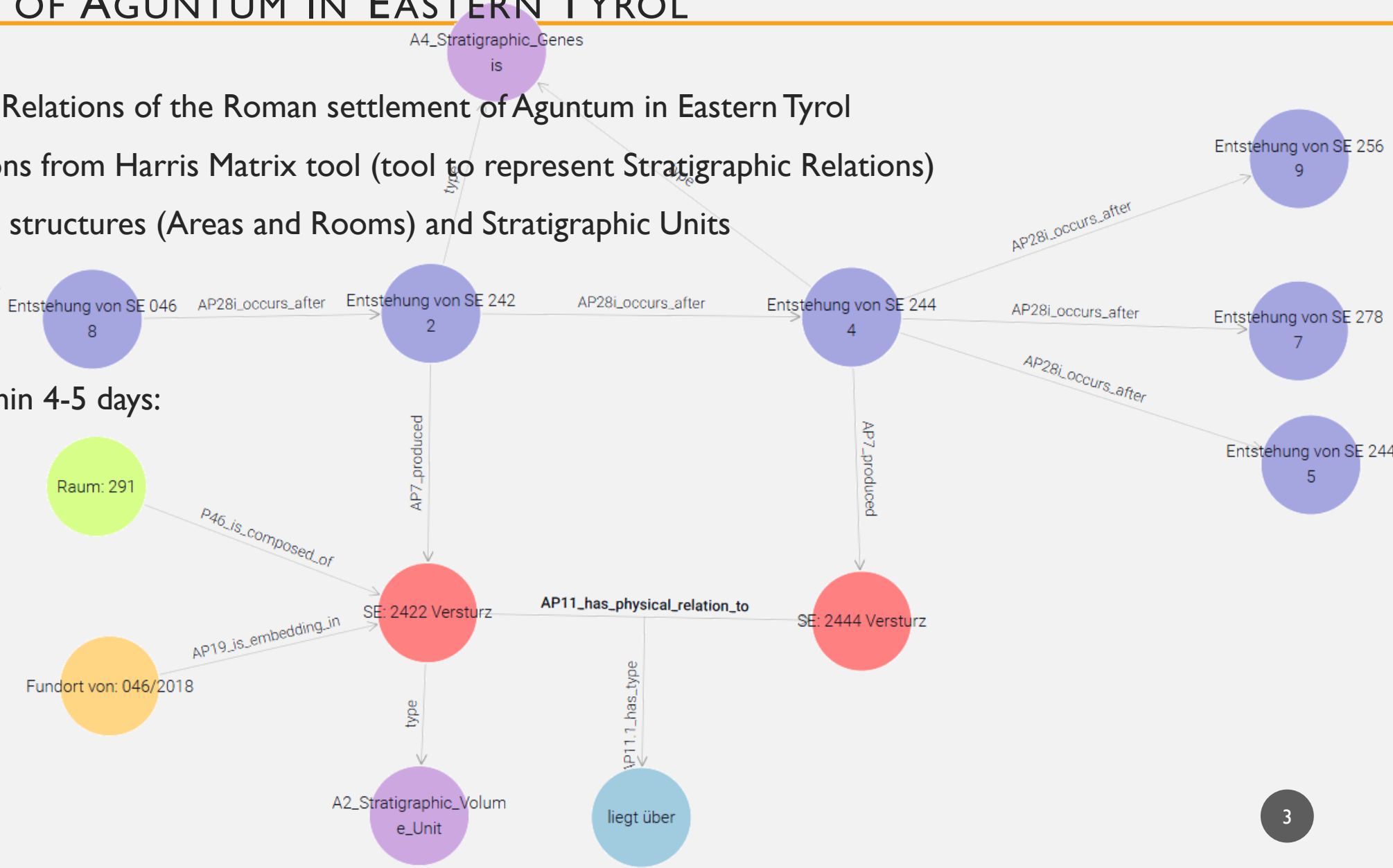
Issues with Knowledge Graphs

- Initial idea: AI assisted creation of a Knowledge Graph for Stratigraphic Relations
- Creation
  - Ontology Knowledge and data knowledge necessary for data modelling
  - very time consuming data entry or complicated data transformations
  - Validation of data
- Query of Knowledge Graph
  - Ontology Knowledge and data knowledge necessary
  - SPARQL is no option for scientists
  - Building of applications from Graph Data

Initial idea: AI assisted creation of Knowledge Graphs for Roman settlement of Aguntum in Eastern Tyrol

Example: Stratigraphic Relations of the Roman settlement of Aguntum in Eastern Tyrol

- Stratigraphic Relations from Harris Matrix tool (tool to represent Stratigraphic Relations)
- Access Database on structures (Areas and Rooms) and Stratigraphic Units
- Excel Sheet on finds
- Creation of KG within 4-5 days:



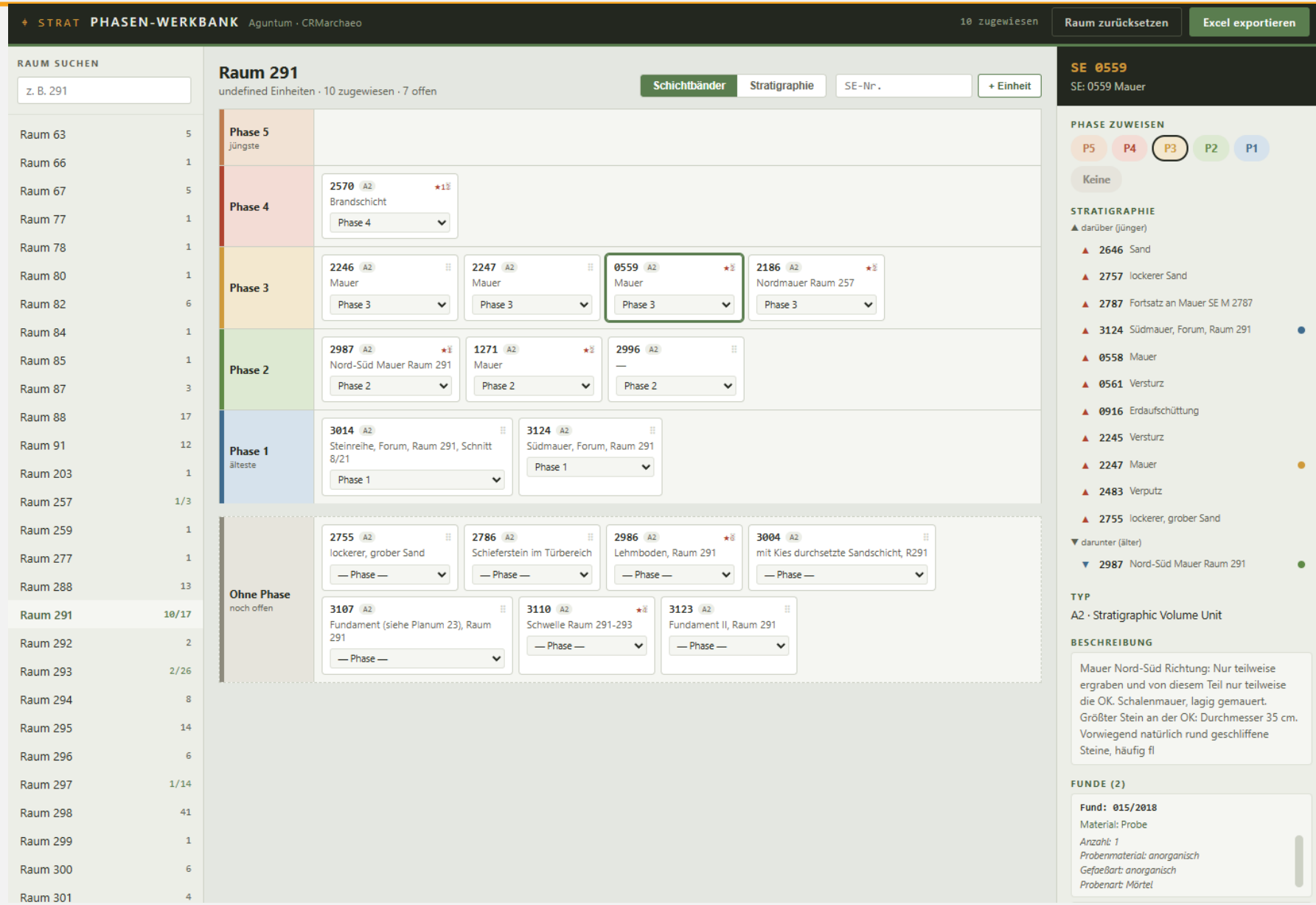
AI ASSISTED CREATION OF APPLICATION

Goal: User Interface to relate Stratigraphic Units to Phases

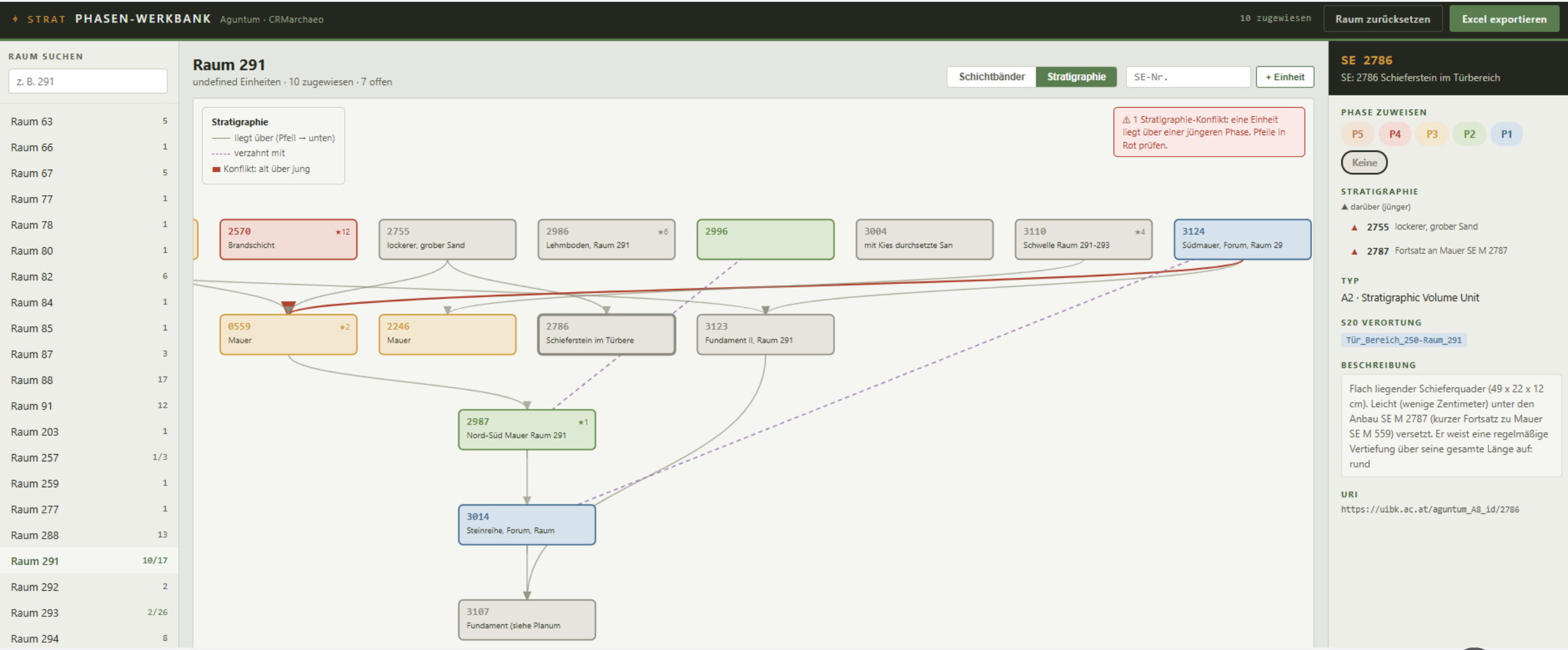
Approach: HTML tool and export relations of Stratigraphic Units to Phases

- Creation of Tool: approx. 1 h with 1 iteration

https://drive.google.com/open?id=1RbF8oLuyDPMcKO-3K4C5dDVsc0C1ybCh&usp=drive_fs



AI ASSISTED CREATION OF APPLICATION



AI Potential for Issues with Knowledge Graphs: Creation

very time consuming data entry

→ creation of structured data, even from scans, webpages or texts with the right prompt

Ontology knowledge and data knowledge necessary

→ can be transferred to AI by feeding sample KG data & data mappings

Complicated data transformations

→ AI writes programs or automates tasks

Validation of data

→ AI writes programs to validate data structure

Data transfer to other applications — network structure vs. structure of the target application (e.g. hierarchical, Site as root element for a database interface or GIS)

→ AI writes programs to transform data to target applications

From your own spreadsheet / database to a CIDOC-CRM knowledge graph

A reproducible workflow: domain data stays in its familiar schema —
the semantic structure grows alongside it, not inside it.

891 objects · 23 CRM classes · ~24,000 triples

The Archaeological University Museum Innsbruck

The largest collection of classical antiquities in western Austria — and mostly plaster casts.

1869
founded as a plaster cast museum

>1,300
casts, copies and originals

2nd
oldest university museum of archaeology in Austria



- Opened in 1869 as the "k.u.k. Gipsmuseum" — the first archaeology classes took place among the casts
- A teaching and study collection: students learn on casts identical in form and size to the ancient works
- Besides casts: Greek pottery, terracottas, Roman sculpture and Latin inscriptions

Why this matters for modelling: most objects are casts, so each record is about two things — the plaster object in Innsbruck and the ancient work it reproduces.

https://drive.google.com/open?id=1d4_7kPAT8dD3E8oDDn9VOUsUGzZXWpua&usp=drive_fs

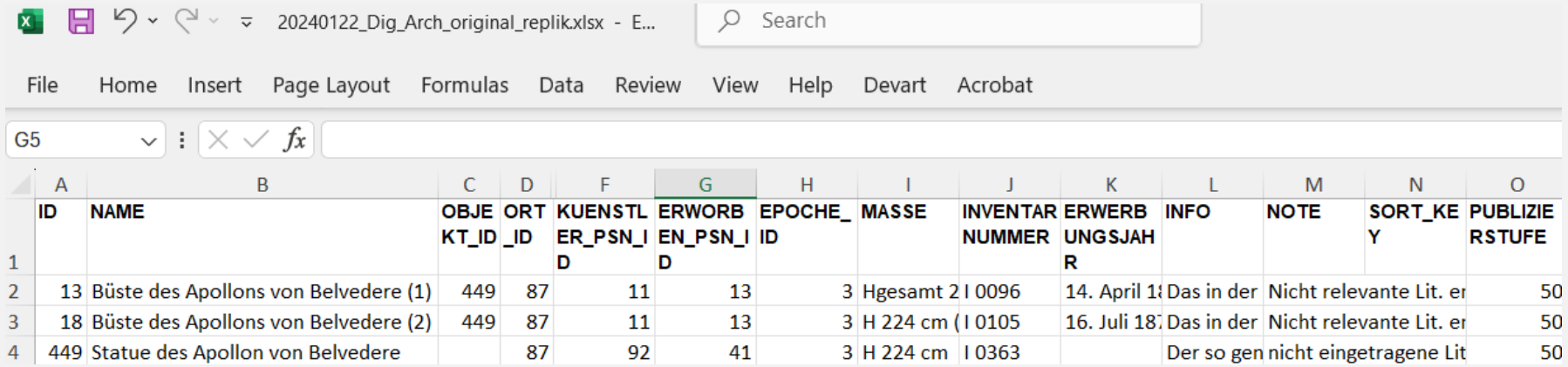
The data: the inventory database as a spreadsheet

Six worksheets exported from the museum's database — the schema the domain experts know and keep.

objekte	**890**	one row per inventory item, 34 columns
Personen	**152**	artists, former owners, dealers
PLH_Orte_Koord	**643**	places with hierarchy and WGS84 coordinates
Material	**76**	material vocabulary — marble, plaster, bronze …
epoche	**44**	periods — Classical, Hellenistic …
Images	**503**	inventory photos with Drive links

- Sheets are linked by IDs — `MATERIAL_ID_IBK` points into Material, `ORT_ID` into the places.
- 164 casts point to a model record via `OBJEKT_ID` — 62 records serve as models.

https://docs.google.com/spreadsheets/d/1_P0LcPIZEX1GK5zWdtnpOlrBKX0NA_c8?rtpof=true&usp=drive_fs



	A	B	C	D	F	G	H	I	J	K	L	M	N	O
1	ID	NAME	OBJEKT_ID	ORT_ID	KUENSTLER_PSN_ID	ERWORBEN_PSN_ID	EPOCHE_ID	MASSE	INVENTARNUMMER	ERWERBUNGSJAHR	INFO	NOTE	SORT_KEY	PUBLIZIERSTUFE
2	13	Büste des Apollons von Belvedere (1)	449	87	11	13	3	Hgesamt 2	I 0096	14. April 1	Das in der	Nicht relevante Lit. er		50
3	18	Büste des Apollons von Belvedere (2)	449	87	11	13	3	H 224 cm (	I 0105	16. Juli 187	Das in der	Nicht relevante Lit. er		50
4	449	Statue des Apollon von Belvedere		87	92	41	3	H 224 cm	I 0363		Der so gen	nicht eingetragene Lit		50

The example: the Apollo Belvedere in three records

Two busts and a full statue — all plaster casts in Innsbruck, all after the same marble statue in the Vatican.



`ID 13 · I 0096`

Büste des Apollons von Belvedere (1)

bust · plaster · H 77 cm with base
acquired 14 April 1876 from the k.k. Österreichisches Museum für Kunst und Industrie

`OBJEKT_ID → 449`



`ID 18 · I 0105`

Büste des Apollons von Belvedere (2)

bust · plaster · H 50 cm
acquired 16 July 1877 from the k.k. Österreichisches Museum für Kunst und Industrie

`OBJEKT_ID → 449`



`ID 449 · I 0363`

Statue des Apollon von Belvedere

statue · plaster · H 230 cm with base
acquired from the Formerei des Louvre, the cast workshop of the Louvre

`model record of 13 and 18`

The original: a Roman copy after a Greek bronze of c. 320 BC, attributed to Leochares — Cortile del Belvedere, Vatican Museums.

One record, two objects

Every row of "objekte" describes the cast in Innsbruck and the ancient original at the same time.

Sheet objekte — record ID 13

the cast in Innsbruck

INVENTARNUMMER	I 0096
MATERIAL_ID_IBK	6 → Gips
MASSE_IBK	H 77 cm (mit Sockel) …
STANDORT	ATRIUM, Raum 01S-21 …
ERWERBUNGSJAHR	14. April 1876
ERWORBEN_PSN_ID	13 → k.k. Österr. Museum …

the ancient original

INVENTARNUMMER_ORIG	1015
MATERIAL_ID_ORIG	1 → Marmor
MASSE	Hgesamt 224 cm
DATIERUNG	hadrianische Kopie nach …
KUENSTLER_PSN_ID	11 → Leochares (?)
AUFBEWAHRUNG_ORT_ID	90 → Cortile del Belvedere
OBJEKT_ID	449 → the model record

In CIDOC CRM: two E22 instances

vam_E22:13 the cast

P45_consists_of **Gips**
P24i_changed_ownership_through **E96 1876**

P130_shows_features_of ↓

vam_E22:449 the model

P45_consists_of **Marmor**
P108i_was_produced_by **E12 → P14 Leochares**
P53_has_former_or_current_location **E53**

The link is P130: what is true of the plaster stays at the cast, what is true of the ancient work is read via OBJEKT_ID.

The tension

Domain experts

think in inventory numbers, acquisition years, find spots — and work in spreadsheets or a database grown over decades.



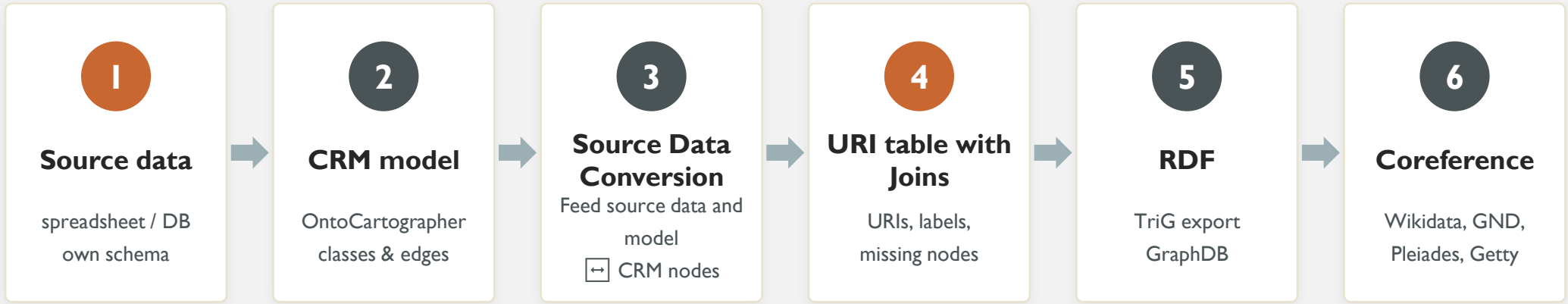
CIDOC-CRM

thinks in events: production, acquisition, time-span — with nodes that exist nowhere as a row in the source data.

The solution: keep both sides apart

Nobody has to give up their schema. One configuration file translates between the two worlds — and it is the only piece rewritten per project.

The workflow in six steps



set up once per project

repeatable on every data correction

Steps 4 to 6 run as scripts — so a correction in the source data costs one more run, not a second round of modelling; step 6 feeds its matches back into step 4.

Archaeological Museum Sample:
https://drive.google.com/open?id=1boGvfpT7meQUMx7KR8sXGITMcqN8sIcV&usp=drive_fs

Step 1 — Source data stays in its own schema

The collection: six worksheets exported from a database, linked by IDs.

Sheet "objekte" — one row per inventory item

```
ID                13
NAME              Büste des Apollons von …
OBJEKT_ID         449        ← reference to the original
MATERIAL_ID_IBK   6      ← reference to sheet "Material"
INVENTARNUMMER    I 0096
ERWERBUNGSKOSTEN  9,00, Kosten der Verpackung …
ORT_ID            87     ← reference to sheet "PLH_Orte_Koord" (Rom)
```

Other sheets

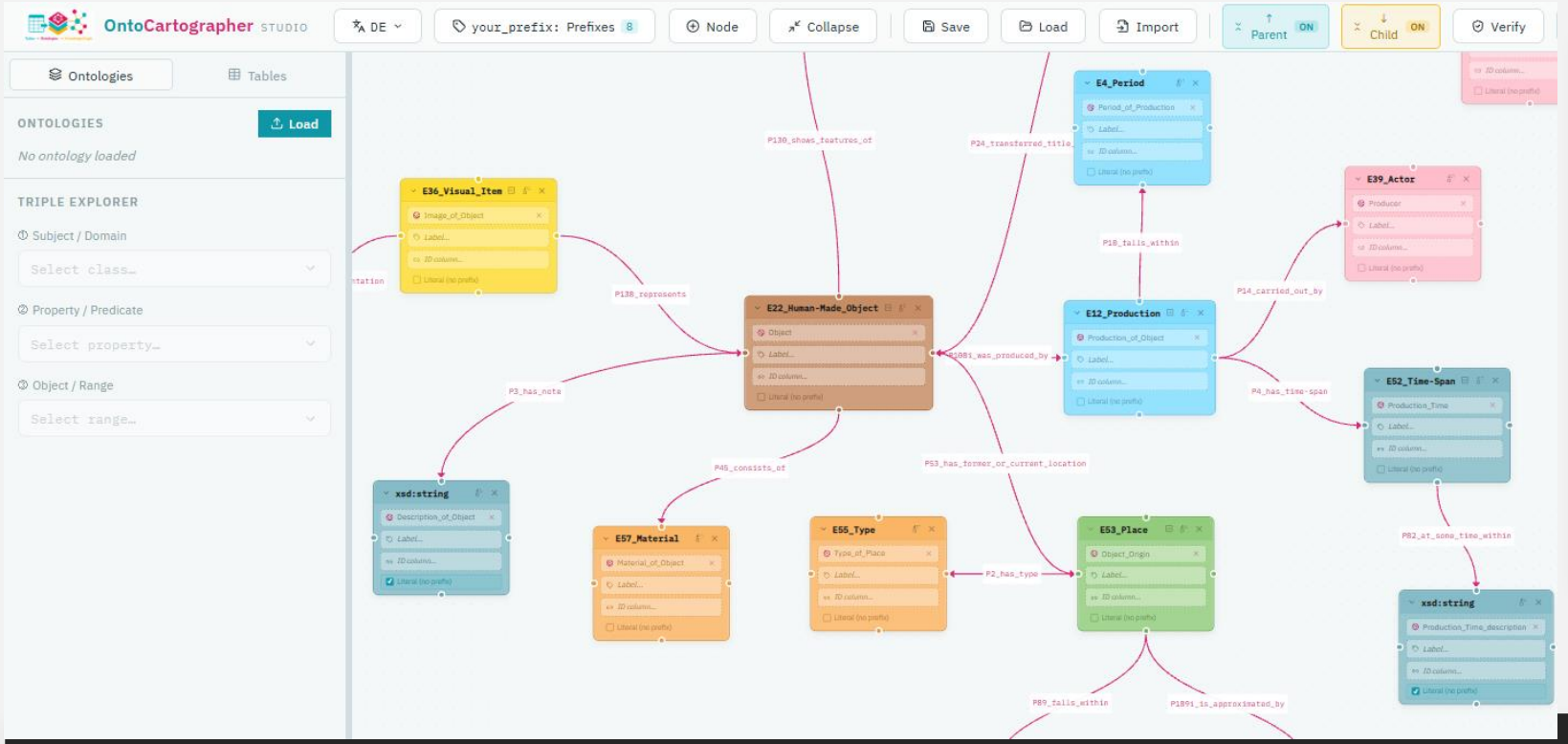
Personen	artists, previous owners
PLH_Orte_Koord	places, hierarchy, coordinates
Material	marble, plaster, bronze …
epoche	Classical, Hellenistic …
Images	filename → Google Drive link

Optional upstream step: Where domain logic has to correct the data — here: the place and artist of the original were recorded only on the replicas — a separate script runs first. It moves values only when the evidence is unambiguous and reports every conflict instead of guessing.

https://docs.google.com/spreadsheets/d/1_P0LcPIZEX1GK5zWdtnpOIrBKX0NA_c8?rtpof=true&usp=drive_fs

Step 2 — The CRM model in OntoCartographer

Classes and properties are connected visually — independently of what the source data looks like.



What is created here

Classes
E22 Object, E12 Production, E39 Actor, E53 Place …

explorerLabel
a speaking name per node — "Producer" rather than just E39

Properties
P108i, P14, P130, wgs:lat …

The key observation: Only some of these nodes exist as a row in the source data. E12 Production, E52 Time-Span, E96 Purchase and SP6 Point are **implicit nodes** — they are generated in the next step.

https://drive.google.com/open?id=1Ha6PMy3XvoH86fQ3jwO6h9LvZvJRckRp&usp=drive_fs

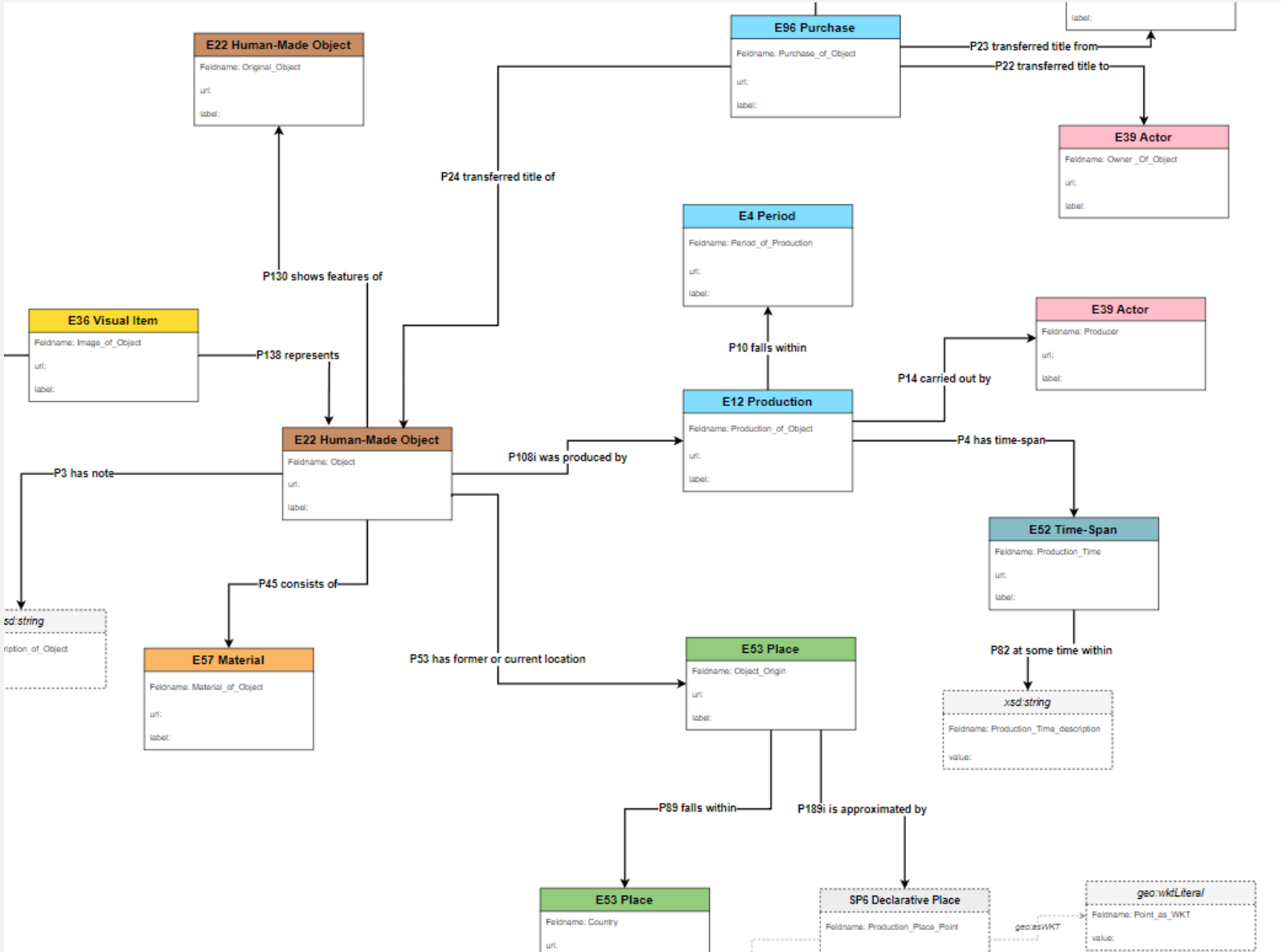
Step 2a — The CRM model in Draw.IO

OntoCartographer json converted in draw.io.

https://drive.google.com/open?id=1QU6J3xHf855wh-8jwM7t2FHggTiaTNsW&usp=drive_fs

Python program and draw.io library

https://drive.google.com/open?id=17hwBNO2arh3VNwyrXSnbNetY4IWN--z8&usp=drive_fs



Step 3 — Source Data Conversion to joined, labeled URI tables

A Python script (build_mapping_table.py) converts source data (xlsx) to joined URI tables (xlsx) with URIs and labels performing joins.

Uses a config file (dig_arch_config.json): For each CRM node you declare where its ID, label or value comes from — including joins across other sheets.

```
"Producer": {
   "via": { "sheet": "Personen",
            "key": "personen_id",
            "from": "KUENSTLER_PSN_ID" },
   "id": "personen_id",  "prefix": "vam_E39",
   "label": "{Name}|{Vorname} {Nachname}"
},
"Production_of_Object": {
   "id": "ID",  "prefix": "vam_E12",
   "label": "Produktion von {NAME}"
},
"Point_as_WKT": {
   "literal": true,
   "value": "POINT({WGS84_long} {WGS_84_lat})"
}
```

dig_arch_config.json — the only project-specific piece of the workflow

python build_mapping_table.py build vam_object_OntoCarto_model.json 20240122_Dig_Arch_original_replik.xlsx dig_arch_config.json Dig_Arch_mapping_from_build.xlsx

https://drive.google.com/open?id=1LXZwlO9Mpa_pXFQebnkRCVw5sFpCl8cc&usp=drive_fs

Join

pull values from other sheets — also as a chain or up a parent hierarchy

Template

{NAME} inserts source values, "a|b" picks the first non-empty option

Prefix

vam_E39 → https://…/vam/E39/ yields clean, resolvable URIs

Condition

create the node only when the required fields are actually filled

Step 4 — The URI table: what the script adds

891 source rows become 41 columns — with URIs, labels and exactly those nodes missing from the source data.

Source data

ID	**13**
NAME	Büste des Apollons von Belvedere (1)
OBJEKT_ID	449

URI table

E22__Object__uri …	vam_E22:13
E12__Production_of_Object__uri	**vam_E12:13**
E12__…__label	Produktion von Büste des …
E22__Original_Object__uri	vam_E22:449

newly created: the production event with its URI and a speaking label

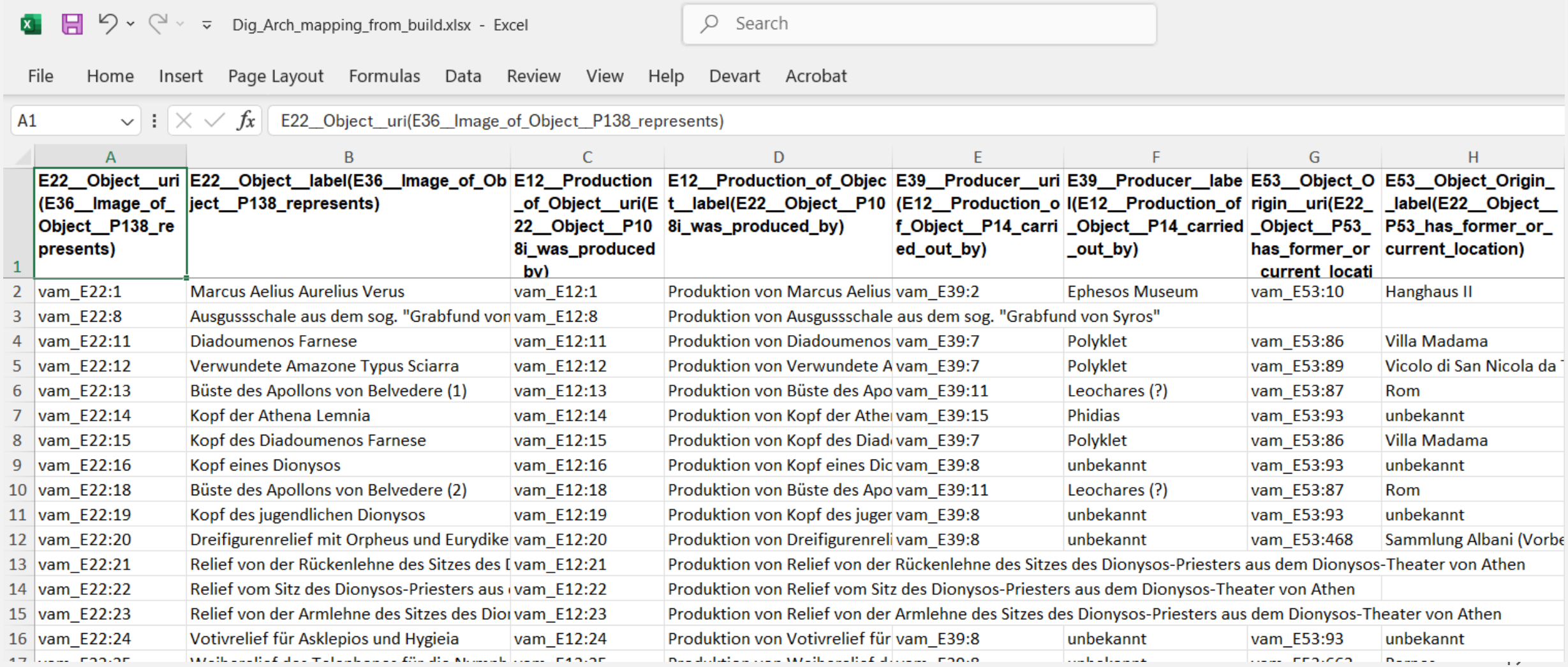
E22__Object__uri(E36__Image_of_Object__P138_represents)

class + role + kind of value · in brackets: the path to it in the graph

Literals get only __value — no more confusing uri with label.

Second sheet "column provenance": source sheet, source column, join path and condition for each of the 41 columns — the table documents itself.

Step 4 — The URI table



	A	B	C	D	E	F	G	H
1	E22__Object__uri (E36__Image_of_ Object__P138_re presents)	E22__Object__label(E36__Image_of_Ob ject__P138_represents)	E12__Production _of_Object__uri(E 22__Object__P10 8i_was_produced by)	E12__Production_of_Objec t__label(E22__Object__P10 8i_was_produced_by)	E39__Producer__uri (E12__Production_o f_Object__P14_carri ed_out_by)	E39__Producer__labe l(E12__Production_of _Object__P14_carried _out_by)	E53__Object_O rigin__uri(E22_ _Object__P53_ has_former_or current_locati	E53__Object_Origin_ _label(E22__Object__ P53_has_former_or_ current_location)
2	vam_E22:1	Marcus Aelius Aurelius Verus	vam_E12:1	Produktion von Marcus Aelius	vam_E39:2	Ephesos Museum	vam_E53:10	Hanghaus II
3	vam_E22:8	Ausgussschale aus dem sog. "Grabfund von	vam_E12:8	Produktion von Ausgussschale aus dem sog. "Grabfund von Syros"				
4	vam_E22:11	Diadoumenos Farnese	vam_E12:11	Produktion von Diadoumenos	vam_E39:7	Polyklet	vam_E53:86	Villa Madama
5	vam_E22:12	Verwundete Amazone Typus Sciarra	vam_E12:12	Produktion von Verwundete A	vam_E39:7	Polyklet	vam_E53:89	Vicolo di San Nicola da
6	vam_E22:13	Büste des Apollons von Belvedere (1)	vam_E12:13	Produktion von Büste des Apo	vam_E39:11	Leochares (?)	vam_E53:87	Rom
7	vam_E22:14	Kopf der Athena Lemnia	vam_E12:14	Produktion von Kopf der Athe	vam_E39:15	Phidias	vam_E53:93	unbekannt
8	vam_E22:15	Kopf des Diadoumenos Farnese	vam_E12:15	Produktion von Kopf des Diad	vam_E39:7	Polyklet	vam_E53:86	Villa Madama
9	vam_E22:16	Kopf eines Dionysos	vam_E12:16	Produktion von Kopf eines Dic	vam_E39:8	unbekannt	vam_E53:93	unbekannt
10	vam_E22:18	Büste des Apollons von Belvedere (2)	vam_E12:18	Produktion von Büste des Apo	vam_E39:11	Leochares (?)	vam_E53:87	Rom
11	vam_E22:19	Kopf des jugendlichen Dionysos	vam_E12:19	Produktion von Kopf des juger	vam_E39:8	unbekannt	vam_E53:93	unbekannt
12	vam_E22:20	Dreifigurenrelief mit Orpheus und Eurydike	vam_E12:20	Produktion von Dreifigurenreli	vam_E39:8	unbekannt	vam_E53:468	Sammlung Albani (Vorbe
13	vam_E22:21	Relief von der Rückenlehne des Sitzes des I	vam_E12:21	Produktion von Relief von der Rückenlehne des Sitzes des Dionysos-Priesters aus dem Dionysos-Theater von Athen				
14	vam_E22:22	Relief vom Sitz des Dionysos-Priesters aus	vam_E12:22	Produktion von Relief vom Sitz des Dionysos-Priesters aus dem Dionysos-Theater von Athen				
15	vam_E22:23	Relief von der Armlehne des Sitzes des Dio	vam_E12:23	Produktion von Relief von der Armlehne des Sitzes des Dionysos-Priesters aus dem Dionysos-Theater von Athen				
16	vam_E22:24	Votivrelief für Asklepios und Hygieia	vam_E12:24	Produktion von Votivrelief für	vam_E39:8	unbekannt	vam_E53:93	unbekannt

Step 5 — Back into OntoCartographer — and out as RDF

One script writes columns, data and prefixes into the project. The export produces TriG for GraphDB.

```
python build_mapping_table.py build \
   mapping.json quelle.xlsx config.json out.xlsx

python inject_into_project.py \
   mapping.json out.xlsx projekt.json --config …
```

- assign columns to nodes
- embed the data
- write the prefixMap
- resolve CURIEs to full URIs

Post-processing: rewrite_prefixes.py enforces your own prefixes and verifies the set of triples stays identical.

```
vam_E22:13 a crm:E22_Human-Made_Object ;
  rdfs:label "Büste des Apollons von Belvedere (1)"@en ;
  crm:P108i_was_produced_by vam_E12:13 ;
  crm:P130_shows_features_of vam_E22:449 ;
  crm:P45_consists_of vam_E57:6 .

vam_E12:13 a crm:E12_Production ;
  rdfs:label "Produktion von Büste des Apollons…"@en ;
  crm:P10_falls_within vam_E4:3 ;
  crm:P4_has_time-span vam_E52:13 .

vam_SP6:87 a crmgeo:SP6_Declarative_Place ;
  wgs:lat "41.8933203"^^xsd:float .
```

ontology-export.trig · ~24,000 triples

https://drive.google.com/open?id=1xuKocJhck7IPiZOW_QXVItiRcWdWTQ3l&usp=drive_fs

Step 6 — Coreferencing to Linked Open Data

Persons, places, materials and periods matched to authorities: scored by script, decided by a person.

```
python coref_extract_context.py \
   export_prefixed.trig coref_context.json
python coref_reconcile.py coref_context.json \
   --classes personen orte material epochen
python coref_apply.py coref_candidates.xlsx \
   mapping.xlsx config.json --provenance …
```

- context per entity: dates, places, roles, objects
- Wikidata, GND, Pleiades, Getty, PeriodO
- candidates scored, reviewed in a match register
- accepted matches flow back into table and config

A loop, not an end point: the LOD identifiers go back into table and config, so the next run of steps 4–6 carries the links.

```
vam_E53:87 skos:exactMatch wd:Q220 ,
                           pleiades:423025 .

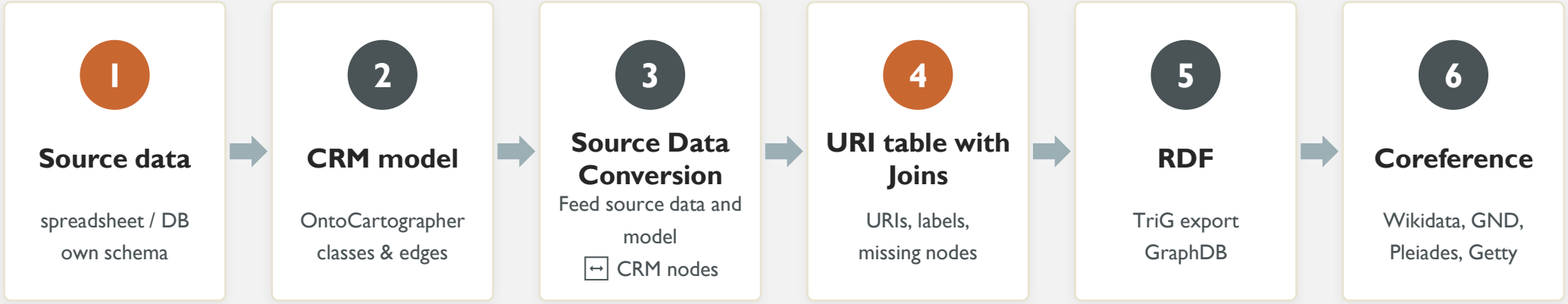
# every match is documented as an event
vam_E13:m0412 a crm:E13_Attribute_Assignment ;
  crm:P140_assigned_attribute_to vam_E53:87 ;
  crm:P141_assigned wd:Q220 ;
  crm:P177_assigned_property_of_type
      skos:exactMatch ;
  crm:P14_carried_out_by vam_E39:reviewer .
```

coref_provenance.trig · excerpt for Rome (place ID 87)

https://drive.google.com/open?id=1aloTi9MPYQojQtPuLLfxtnCiSrtyDRdc&usp=drive_fs

Corrections: one run, not a new model

A typo in the source data costs one run of the scripts — the modelling work is preserved.



set up once per project

repeatable on every data correction

The diff shows what the correction did

```
890 → 889 rows | removed: 1
changed columns: 14
   3  E22__Original_Object__label …
   1  E53__Country__uri …
   1  SP6__…__label …  (+ 10 more)
```

What this means in practice

- A renamed object propagates automatically into every dependent label.
- Source data, config and project live in Git — every state is traceable.
- The diff acts as a sanity check before exporting.

What comes out at the end

891
objects

23
CRM classes

~24,000
triples

1
config file

A queryable graph

SPARQL across material, period, artist, place — and combinations

Linked into the LOD cloud

Wikidata, Pleiades, AAT via an auditable match register

The collection online

A site with photos, facets and a map of the originals — straight from the graph

Transferable: For a different collection the CRM model and the config change — the scripts stay the same. What each project requires is modelling and field mapping, not programming.

AI Potential for Issues with Knowledge Graphs: Query

Ontology knowledge and data knowledge necessary

→ sample data and mappings replace prior CRM training

SPARQL is no option for scientists

→ AI writes the SPARQL query

Development of applications to use graph data

→ AI creates applications

Use of a Chatpod to query data

→ natural language in, graph answers out

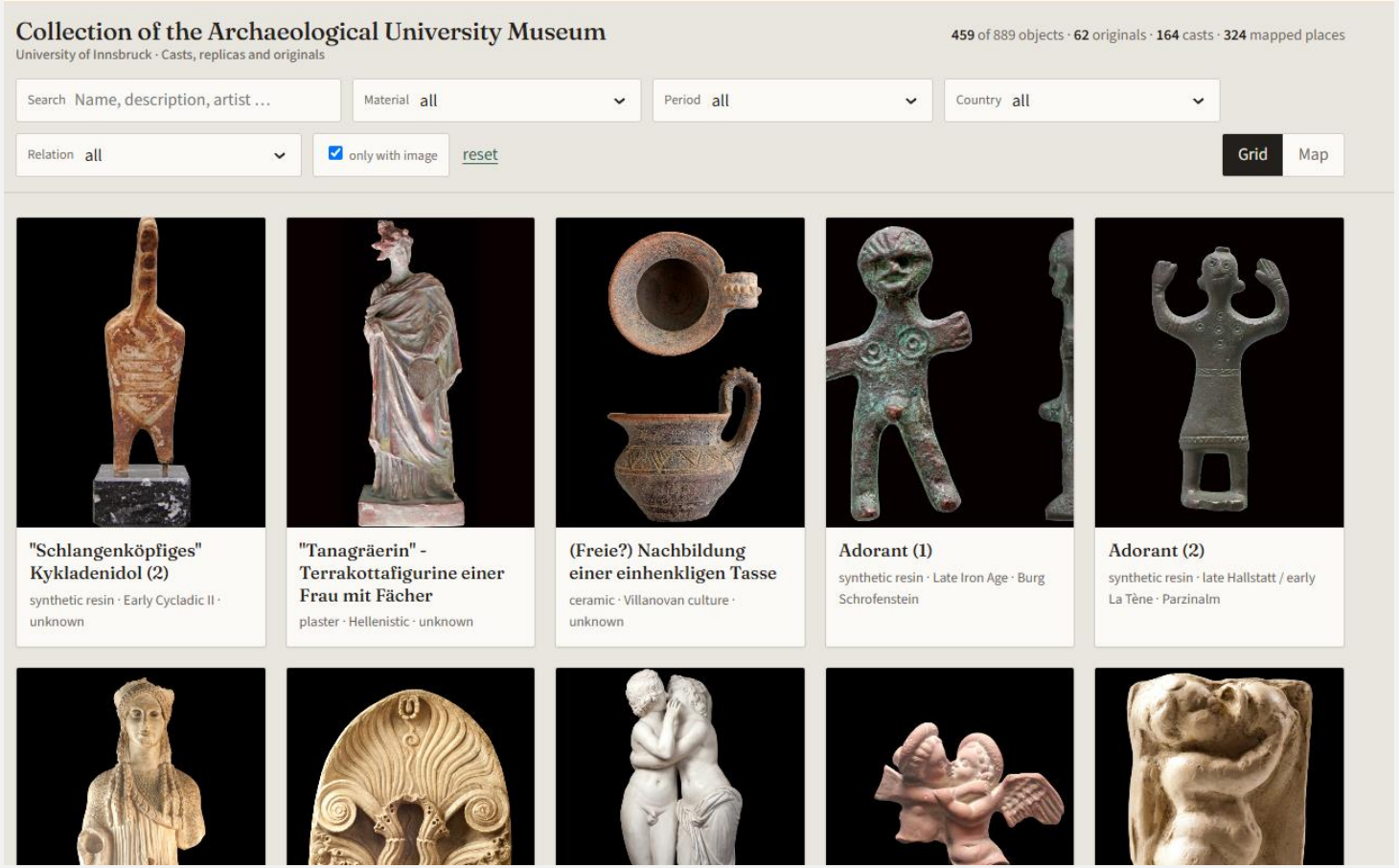
Examples

Archaeological University Museum

collection website — next slide

Query without SPARQL: the collection as a website

The same graph, read by a visitor: facets instead of queries, a map instead of coordinates — one self-contained HTML file generated from the TriG export.



Facets from the graph

Country, period, material and artist come straight from the CRM classes, with live counts

Replica → original

For a cast the map shows where its model comes from, resolved through P130

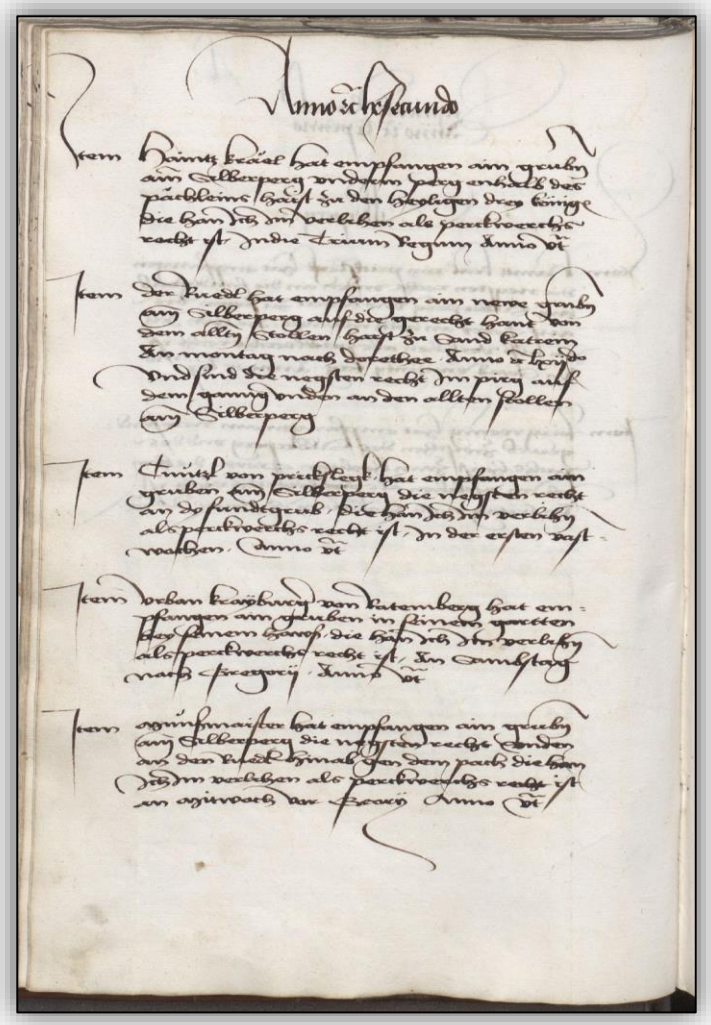
Photos and detail

454 objects with images, acquisition, provenance and the note text

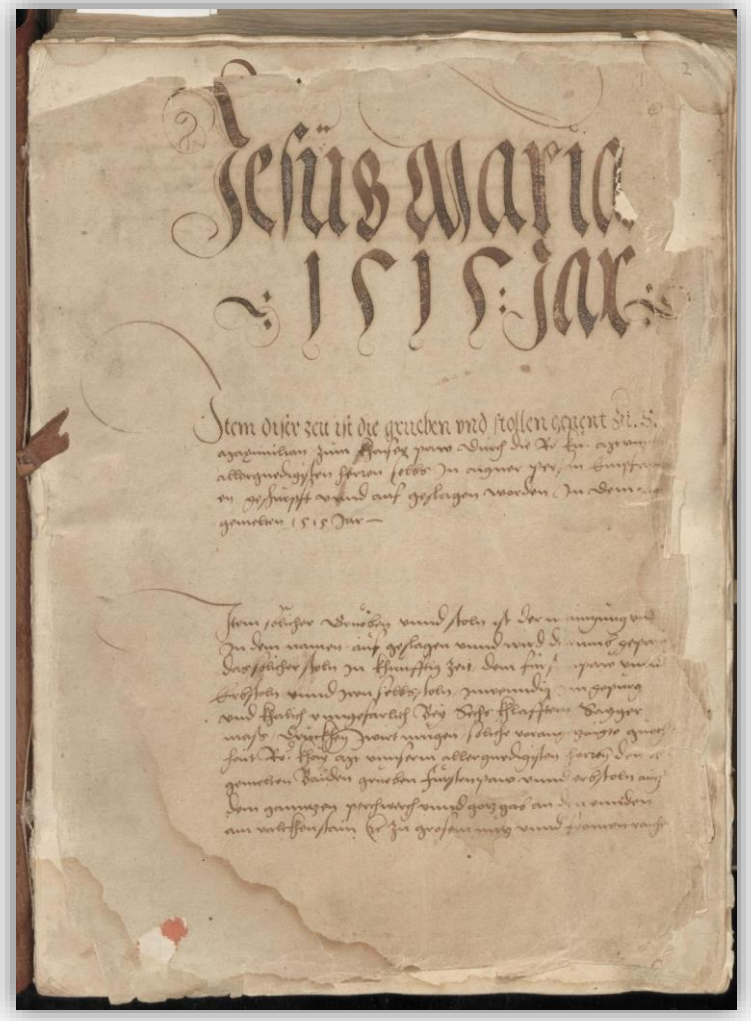
https://drive.google.com/open?id=1d4_7kPAT8dD3E8oDDn9VOUsUGzZXWpua&usp=drive_fs

829 objects · 454 with photo · 467 on the map generated from the exported TriG by one script

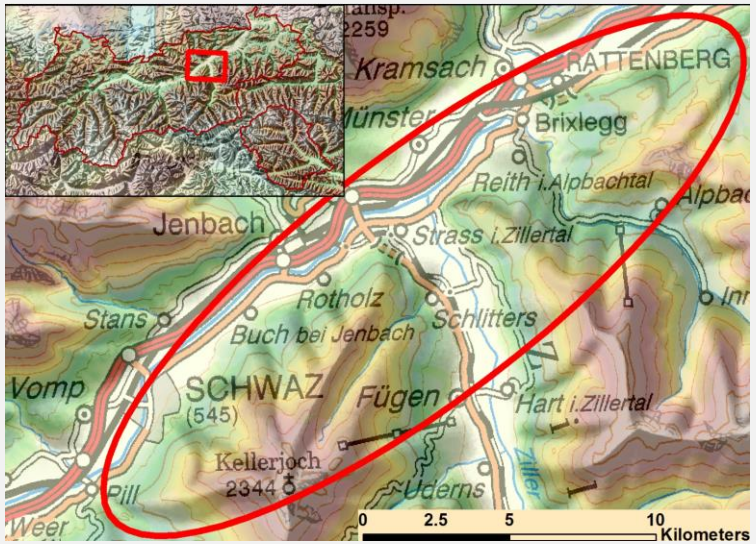
T.M.M.M.T. - Text Mining Medieval Mining Texts
Awards of Mining claims in two mediavel documents

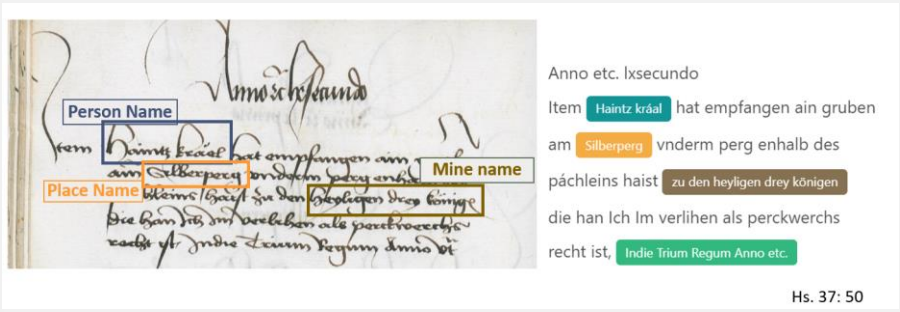


Verleihbuch der Rattenberger Bergrichter, 1460-1463 (Hs. 37)

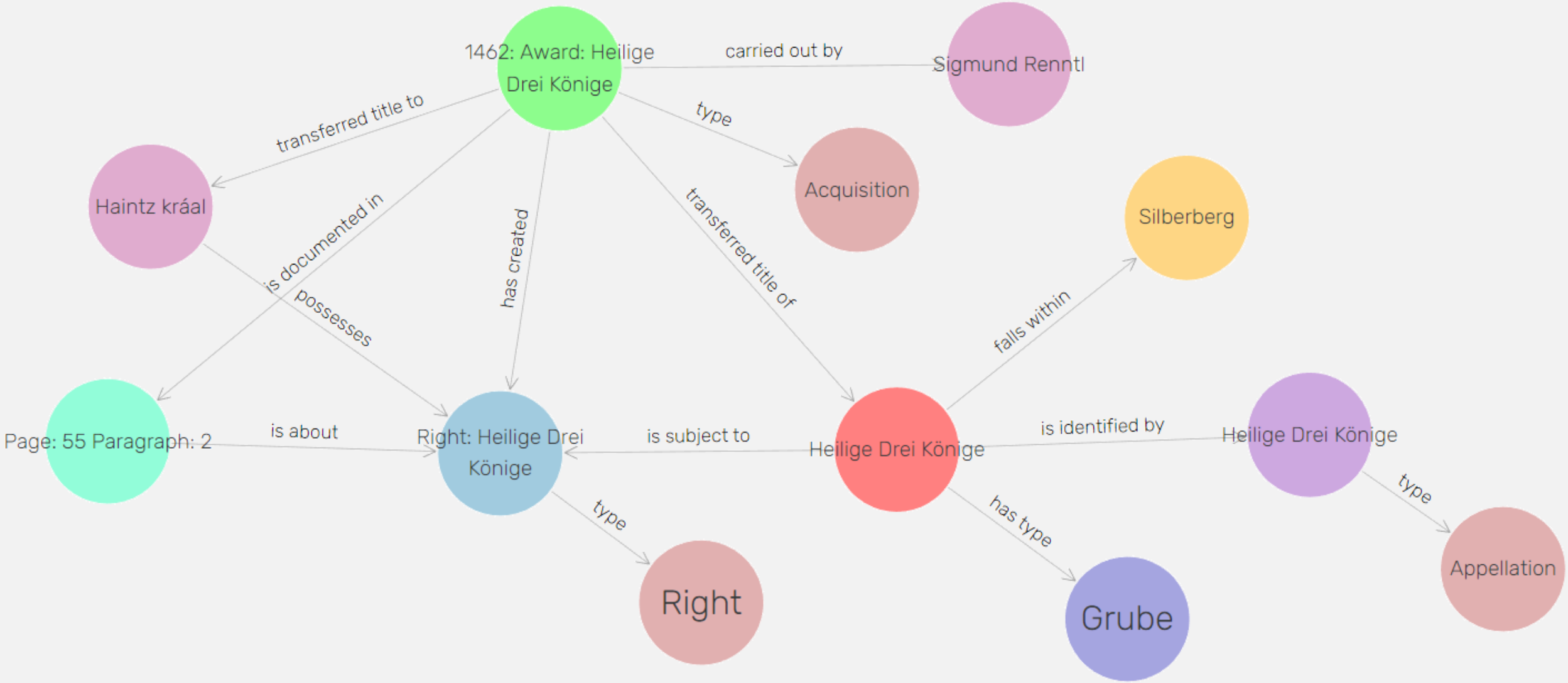


Schwazer Berglehenbuch, ca. 1515 (Hs. 1587)





Knowledge Graph: Mine - Heilige drei Könige

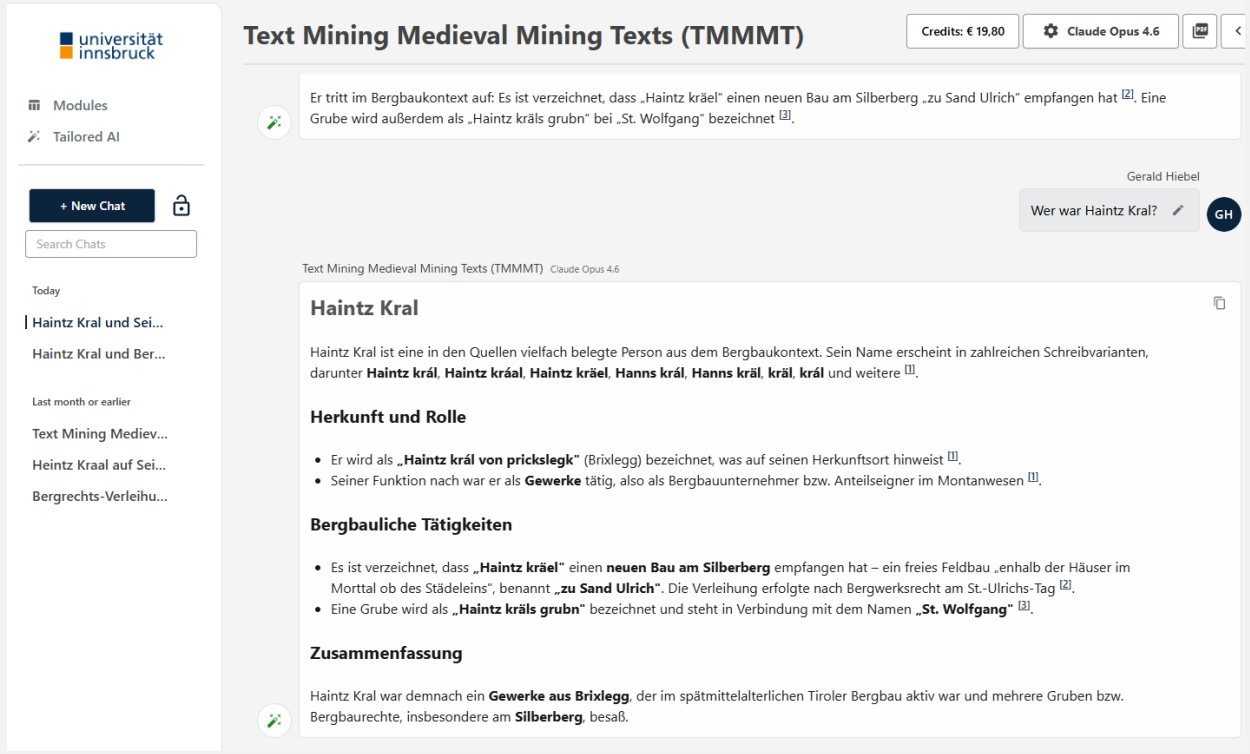


AI ASSISTED CREATION OF APPLICATION



https://drive.google.com/open?id=11kUWNu7P0OzkHDh9i_kPSHiHXXQmsm0q&usp=drive_fs

UIBK Academic AI – RAG Chatpod to Query Knowledge Graph



Wer war Haintz Kral?

Suchte nach Informationen über historische Person >

Auf Basis der Daten lässt sich über Haintz Král folgendes sagen:

Haintz Král von Brixlegg war einer der aktivsten Gewerken im Verleihbuch der Rattenberger Bergrichter (Hs. 37, 1460–1463). Der Knowledge Graph weist ihn mit **15 namentlichen Namensvarianten** aus: *Haintz král, Haintz kráal, Haintz kräel, haintzen králs, Hanns král, chräls, kräl* u.a. — typisch für die uneinheitliche mittelhochdeutsche Schreibung der Zeit.

Herkunft: Er stammt aus **Brixlegg** (in der Quelle: *von prickslegk*), und ist im Knowledge Graph als **Gewerke** typisiert (`E55/Gewerke`), also ein lizenzierter Bergwerksunternehmer. Sein Wohnort wird mit dem Ortspunkt **et90_pl1** (Brixlegg-Bereich) angegeben.

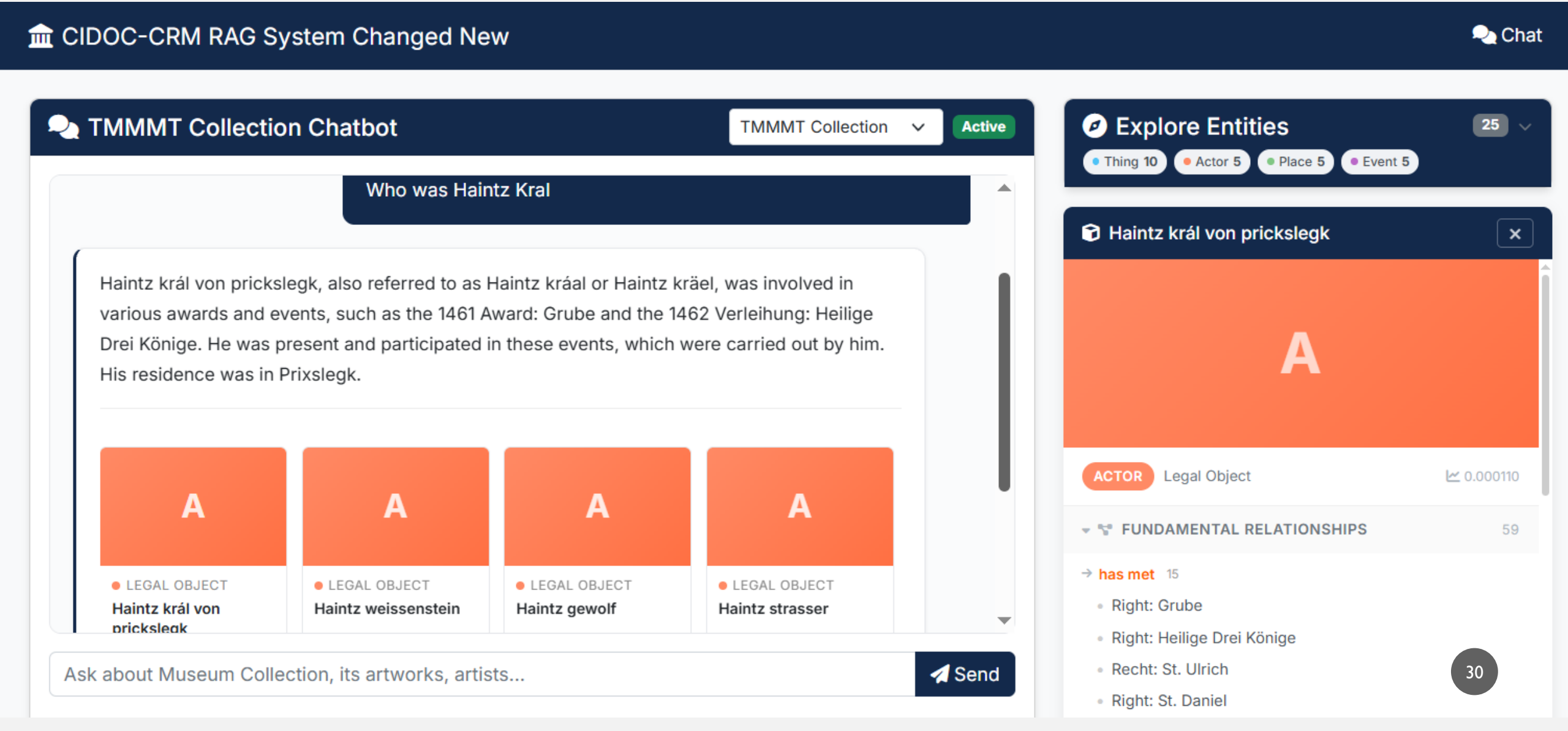
Bergbauaktivität: Er erhielt zwischen 1460 und 1463 insgesamt **ca. 35 Grubenverleihungen** in mindestens 9 verschiedenen Abbaugebieten — eine außergewöhnlich hohe Zahl, die ihn zu einem der bedeutendsten Akteure in der Handschrift macht:

Gebiet	Grubennamen (Auswahl)
Silberperg	Grube (Fundgrube), Heilige Drei Könige, St. Ulrich, St. Lienhard im Haferfeld, St. Oswald, Zum Hammersbach, St. Bartlme, St. Michel
Pintzkaw (Reith im Alpbachtal)	St. Daniel, Hüttschlag
Albsteg	St. Wolfgang, St. Lorentzen, Grube
Einperg (Ainperg)	St. Lienhard, Zum Heiligen Kreuz
Holtzalben (Holzalm)	Fundgrube, Zum Glück
Múlpúhel (Mühlbühel)	St. Lorentzen

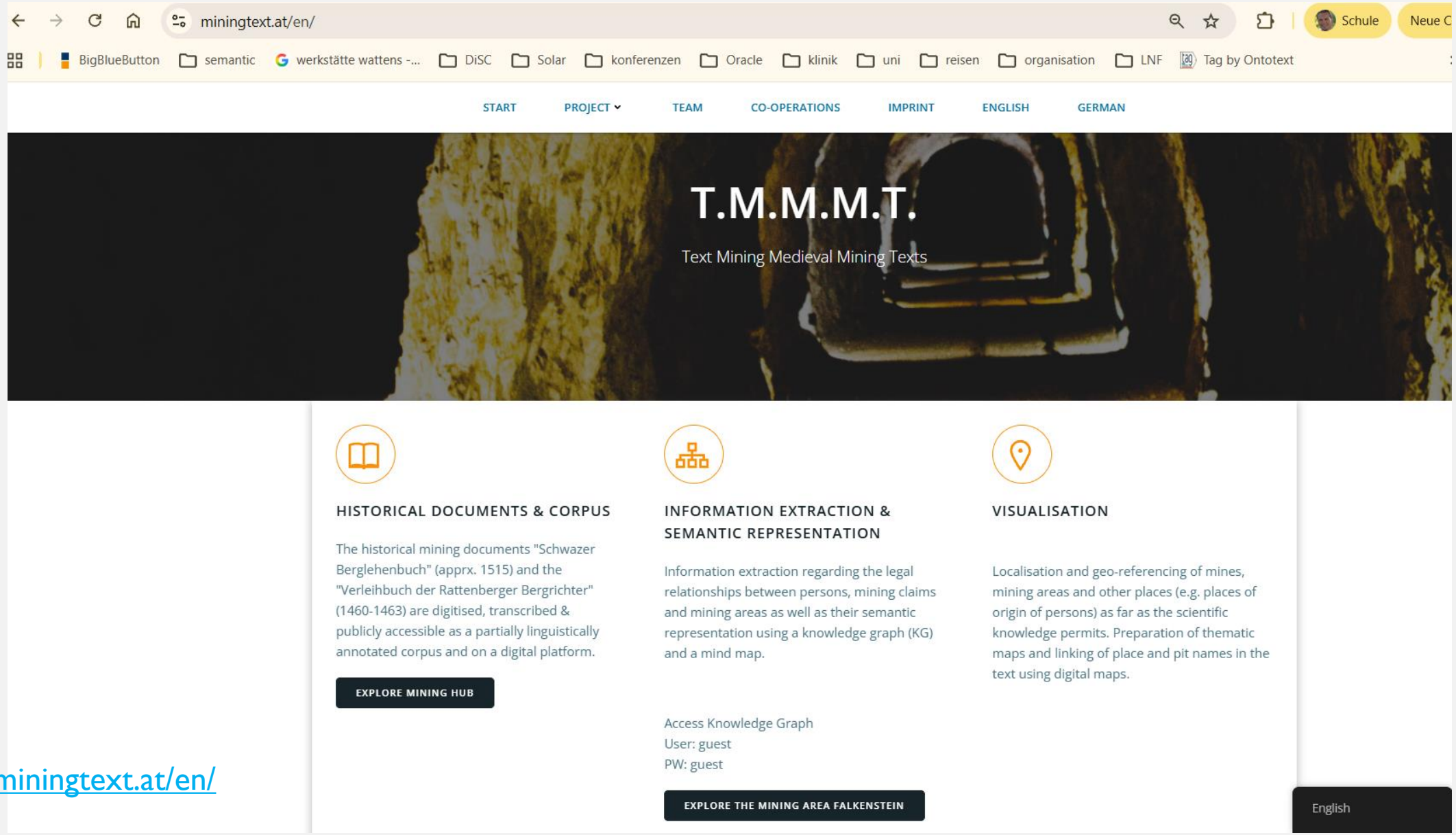
Text Mining Medieval Mining Texts (TMMMT) https://miningtext.at
Academic AI **Chatpod:**
https://uibk.academic-ai.at/join/NbU3UeHNtAcQ4CNWUP4O176BZtCYGtxa8TNFSApC8UyiUnRd

Information Integ. for Prehist. Mining Archaeology (https://www.uibk.ac.at/en/projects/inf-integ4pre-min-archaeo/)
Academic AI Chatpod:
https://uibk.academic-ai.at/join/z9OKWWCGjyGkJt4hGQKYeuvxpKfwWcQLDy6utKUchRXhtxZ3

Nicola Carboni - GRAPH RAG - Chatpod

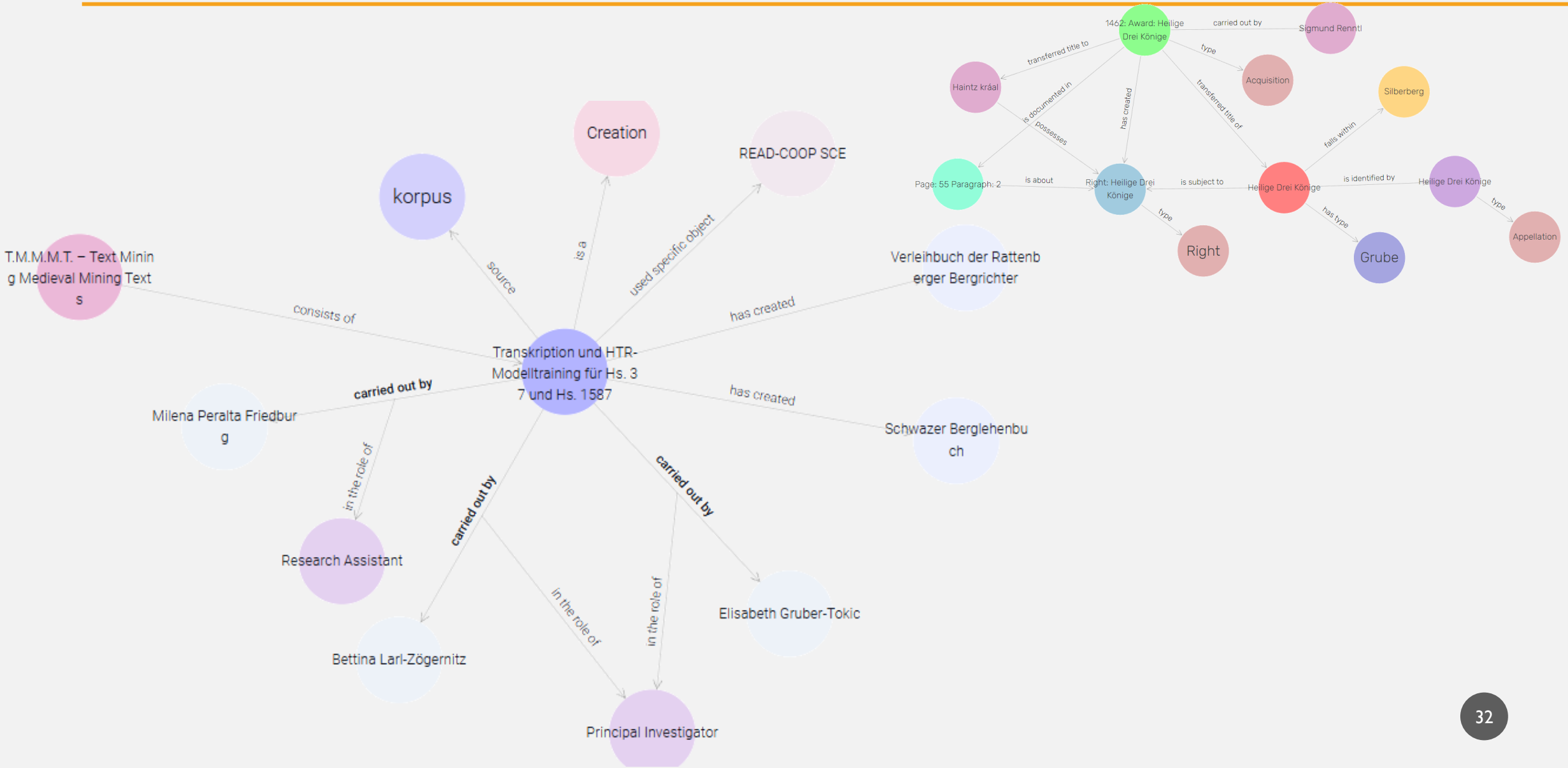


Transforming Texts to RDF and integrate with the Knowledge Graph

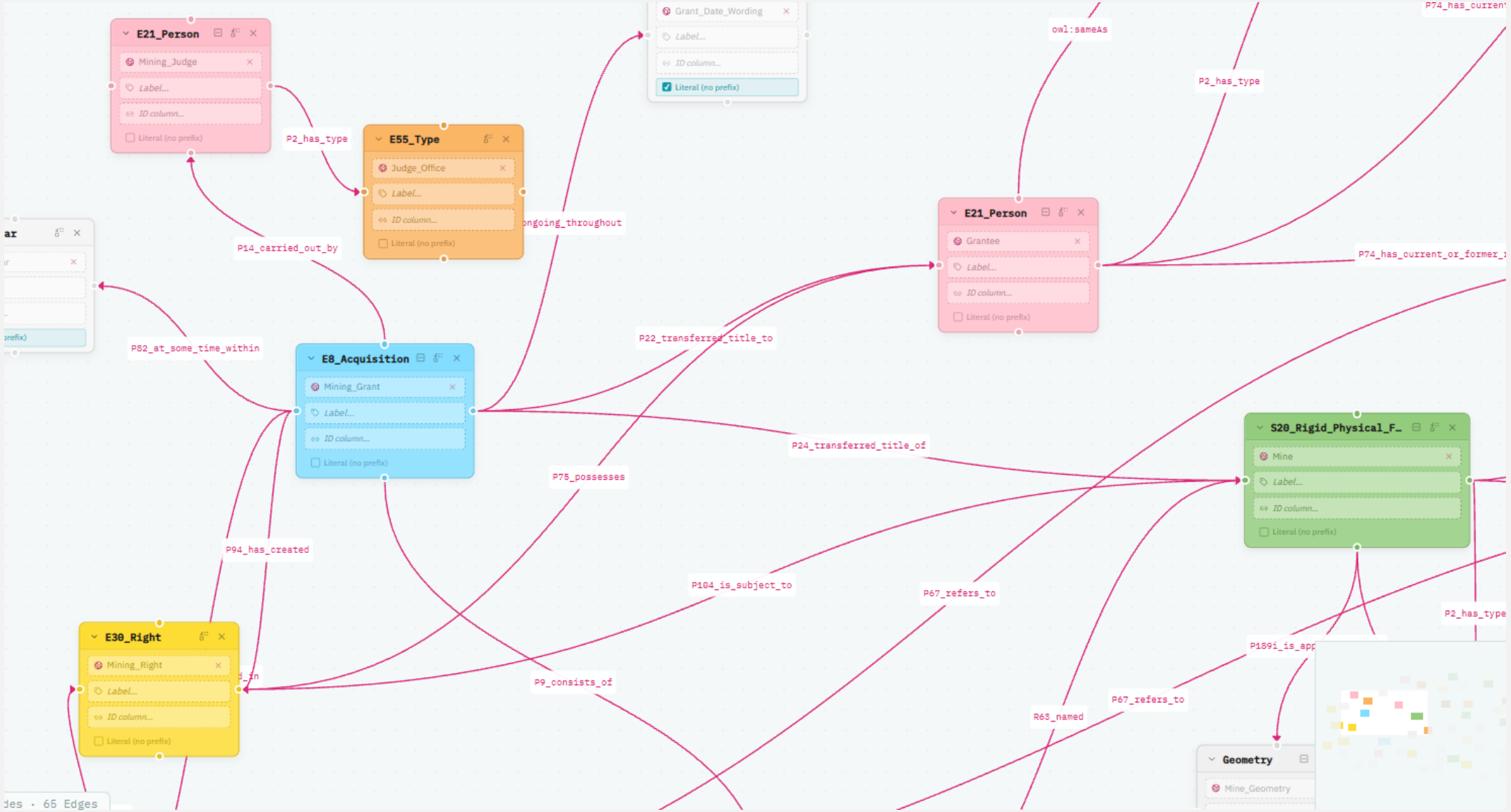


https://miningtext.at/en/

Transforming Texts to RDF and integrate with the Knowledge Graph



Reengineering Existing Knowledge Graph -> OntoCartographer



Reengineering Existing Knowledge Graph -> OntoCartographer

