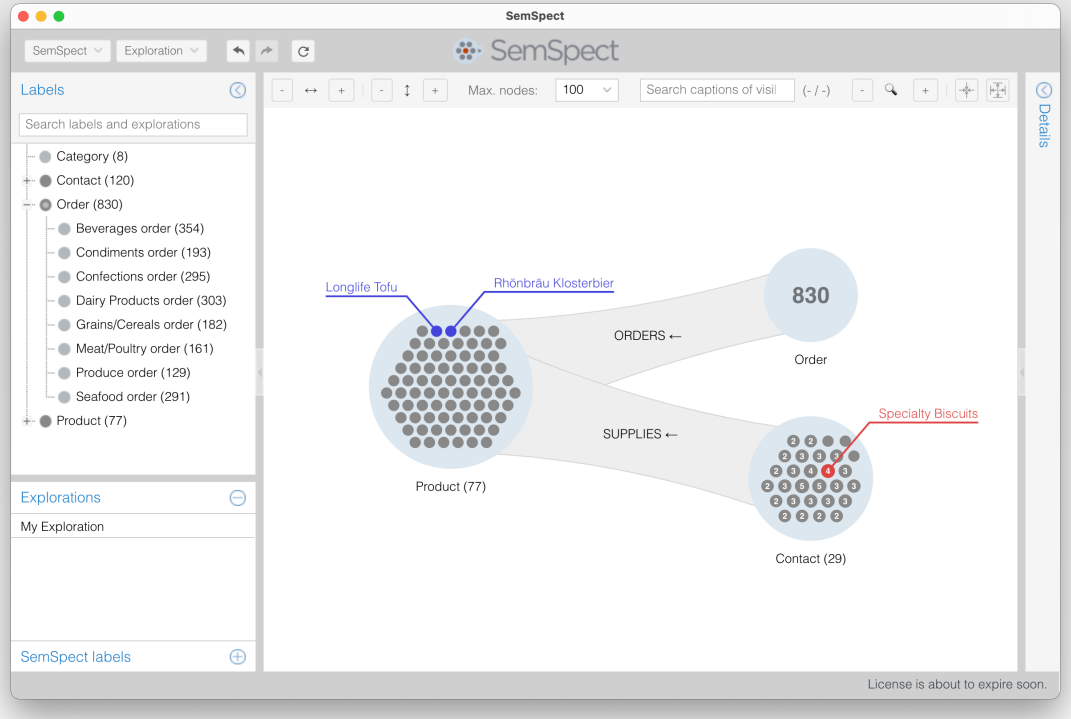
Developing and Refining Schemas for Knowledge Graphs

Tutorial at KGC 2022

Thorsten Liebig & Vincent Vialard

derivo

- SME, est. 2010
- Background KR & reasoning
- Consulting and development of semantic software tools & solutions
- Products:
  - Reasoning:
    - Konclude
    - GraphScale
    - Reseller of RDFox
  - Browsing:
    - SemSpect



Outline

- Motivation
- Demo data import workflow
- LPG and RDFS
- Schema modeling options in LPG and RDFS
- Interactive session with Neo4j

Data Graph

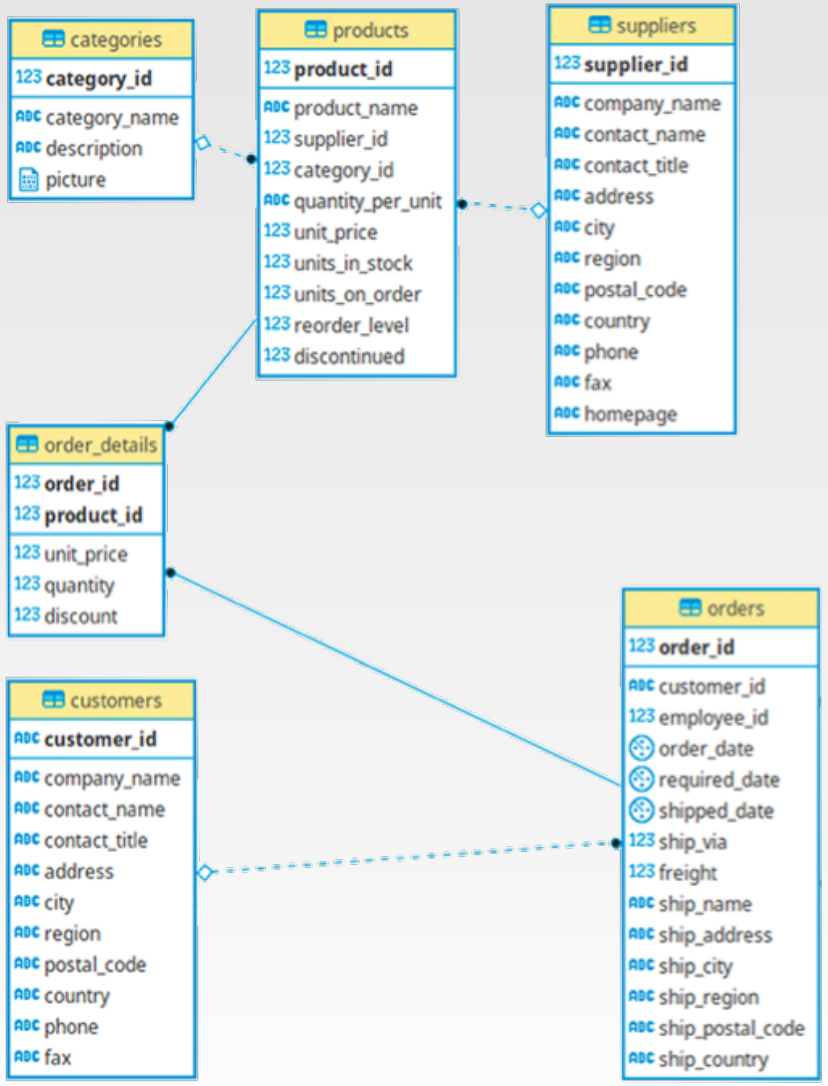
- Real-world data often is a graph
- Graph: collection of data represented as nodes and edges (either represented in the LPG or RDF model)
- Often loaded from relational sources naively into a graph:
  - users often don’t recognize their own data
  - no “views” to look at the data from different perspectives
  - query writing is more complicated than necessary

Northwind Example

orderID	customerID	employeeID	shipVia	freight	shipName	shipAddress	shipCity	shipRegion	shipPostalCode	shipCountry	
10248	VINET	5	3	32.38	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	NULL	51100	France	
10249	TOMSP	6	1	11.61	Toms Spezialitäten	Luisenstr. 48	Münster	NULL	44087	Germany	
10250	HANAR	4	2	65.83	Hanari Carnes	Rua do Paço	67	Rio de Janeiro	RJ	05454-876	Brazil
10251	VICTE	3	1	41.34	Victuailles en stock	2	rue du Commerce	Lyon	NULL	69004	France
10252	SUPRD	4	2	51.30	Suprêmes délices	Boulevard Tirou	255	Charleroi	NULL	B-6000	Belgium
10253	HANAR	3	2	58.17	Hanari Carnes	Rua do Paço	67	Rio de Janeiro	RJ	05454-876	Brazil
10254	CHOPS	5	2	22.98	Chop-suey Chinese	Hauptstr. 31	Bern	NULL	3012	Switzerland	
10255	RICSU	9	3	148.33	Richter Supermarkt	Starenweg 5	Genève	NULL			
10256	WELLI	3	2	13.97	Wellington Importadora	Rua do Mercado	12	Resende	SP		
10257	HILAA	4	3	81.91	HILARION-Abastos	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	Táchira			
10258	ERNSH	1	1	140.51	Ernst Handel	Kirchgasse 6	Graz	NULL			
10259	CENTC	4	3	3.25	Centro comercial Moctezuma	Sierras de Granada 9993	México D.F.	NULL			
10260	OTTIK	4	1	55.09	Ottilies Käseladen	Mehrheimerstr. 369	Köln	NULL			
10261	QUEDE	4	2	3.05	Que Delícia	Rua da Panificadora	12	Rio de Janeiro	RJ		
10262	RATTC	8	3	48.29	Rattlesnake Canyon Grocery	2817 Milton Dr.	Albuquerque	NM			
10263	ERNSH	9	3	146.06	Ernst Handel	Kirchgasse 6	Graz	NULL			
10264	FOLKO	6	3	3.67	Folk och fä HB	Åkergatan 24	Bräcke	NULL	S-		
10265	BLONP	2	1	55.28	Blondel père et fils	24	place Kléber	Strasbourg	NU		
10266	WARTH	3	3	25.73	Wartian Herkku	Torikatu 38	Oulu	NULL			
10267	FRANK	4	1	208.58	Frankenversand	Berliner Platz 43	München	NULL			
10268	GROSR	8	3	66.29	GROSELLA-Restaurante	5ª Ave. Los Palos Grandes	Caracas	DF			
10269	WHITC	5	1	4.56	White Clover Markets	1029 - 12th Ave. S.	Seattle	WA			

orderID	productID	unitPrice	quantity	discount
10248	11	14.00	12	0
10248	42	9.80	10	0
10248	72	34.80	5	0
10249	14	18.60	9	0
10249	51	42.40	40	0
10250	41	7.70	10	0
10250	51	42.40	35	0.15
10250	65	16.80	15	0.15
10251	22	16.80	6	0.05
10251	57	15.60	15	0.05
10251	65	16.80	20	0
10252	20	64.80	40	0.05
10252	33	2.00	25	0.05
10252	60	27.20	40	0
10253	31	10.00	20	0
10253	39	14.40	42	0

Northwind Neo4j Data Import

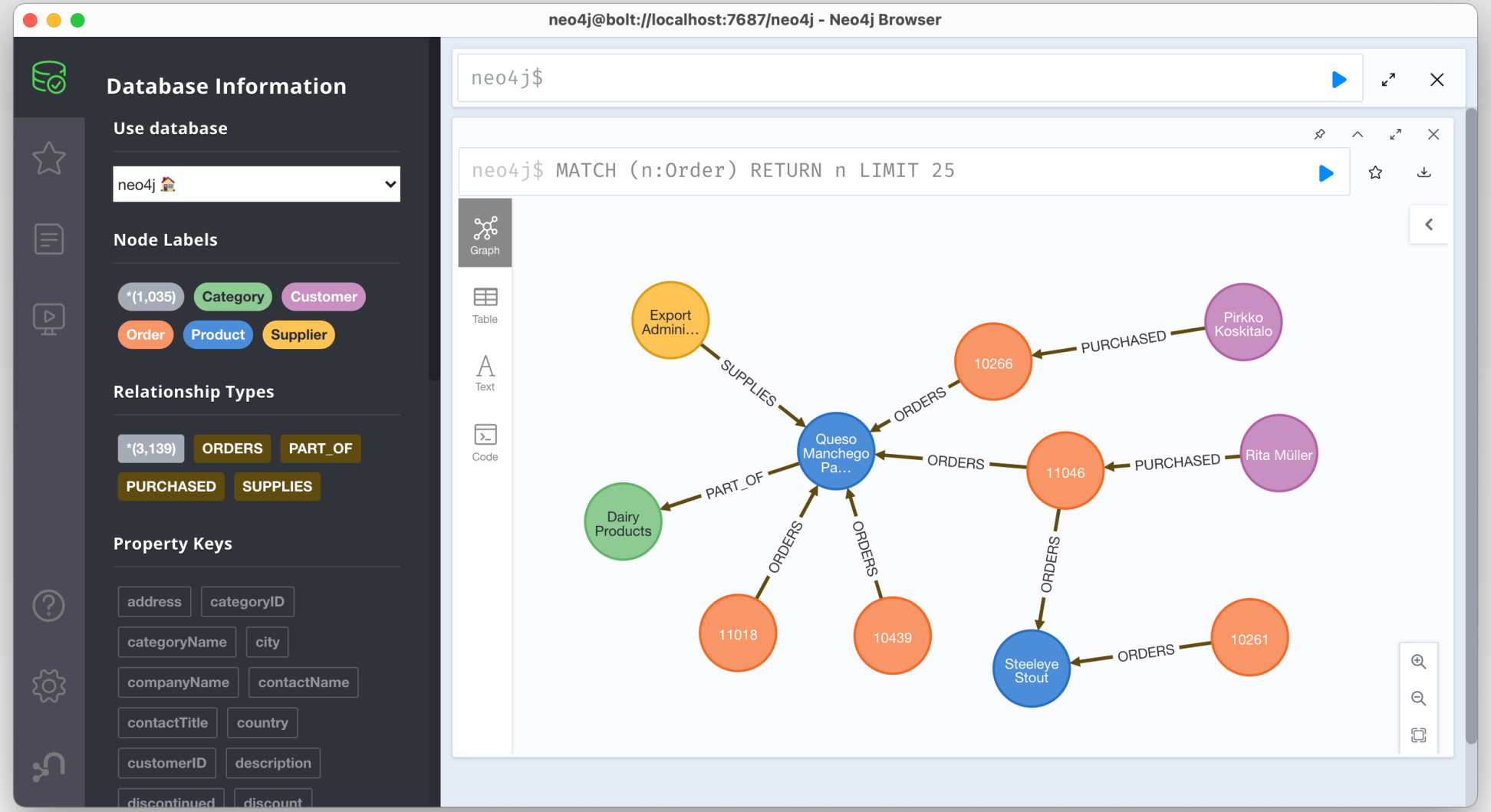



```
LOAD CSV WITH HEADERS
  FROM "order-details.csv" AS row
MATCH (p:Product), (o:Order)
WHERE p.productID = row.productID AND
      o.orderID = row.orderID
MERGE (o)-[details:ORDERS]->(p)
SET details = row,
details.quantity = toInteger(row.quantity);
```

Northwind Neo4j Data Import


```
LOAD CSV WITH HEADERS
  FROM "order-details.csv" AS row
MATCH (p:Product), (o:Order)
WHERE p.productID = row.productID AND
      o.orderID = row.orderID
MERGE (o)-[details:ORDERS]->(p)
SET details = row,
details.quantity = toInteger(row.quantity);
```

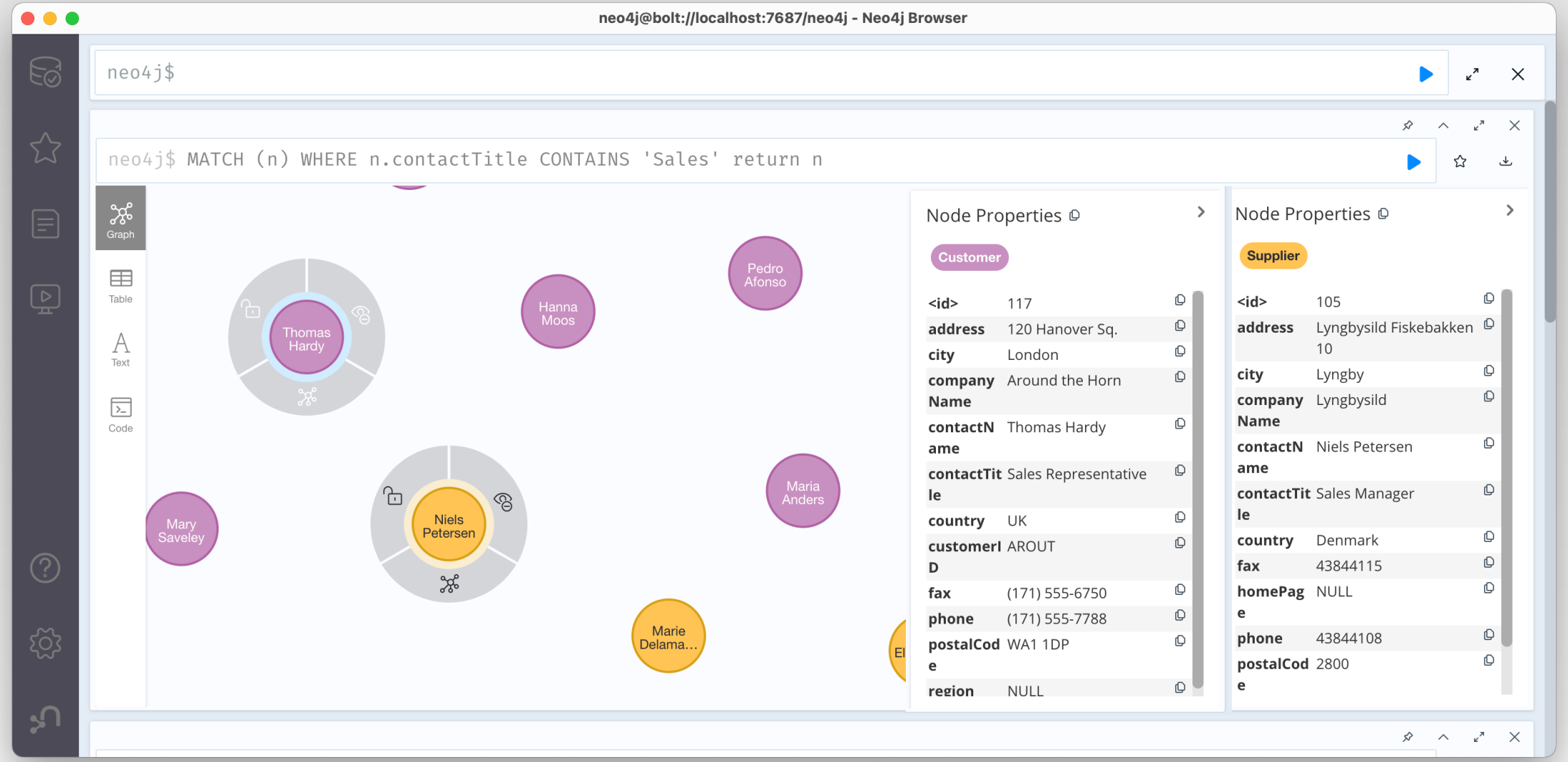
Northwind in Neo4j



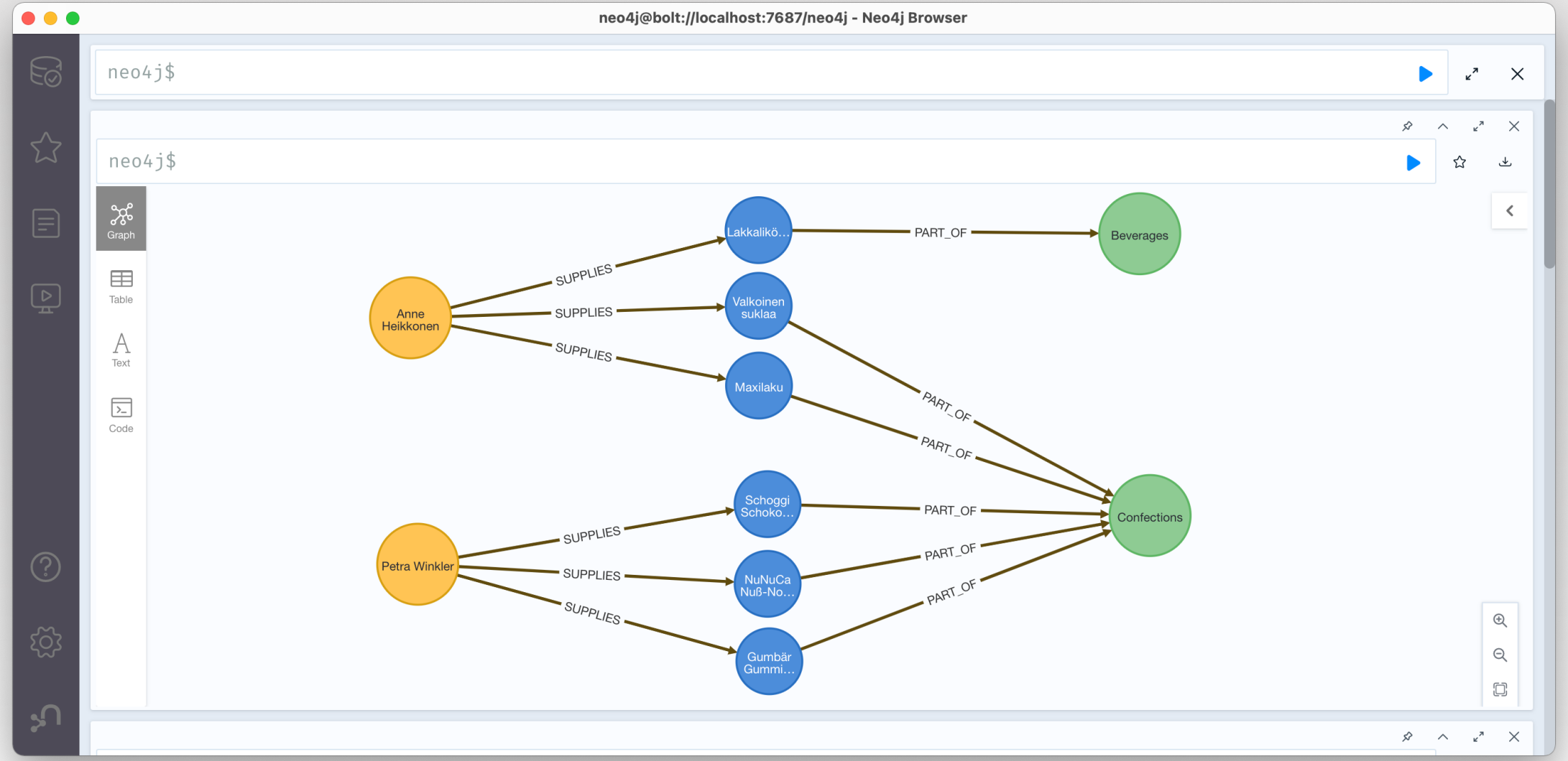
Preparation for Hands-On Experience

1. Download Neoj4 Desktop
2. Import sample project “Northwind”
3. Install APOC plugin
4. Install SemSpect Graph App via Graph App Gallery
5. Open SemSpect and install plugin to Northwind DBMS
6. Start Northwind DBMS

Label Generalization & Attribute Classification



Indirect Node Classification



Knowledge Graph Models

Property Graphs

- LPG = Labeled Property Graph
- Graph theory: labeled multigraph
  - labeled nodes
  - labeled edges
- Graph databases
- NoSQL type
- Query language: gremlin, Cypher, GraphQL

Resource Description Framework

- RDF & RDFS
- General data model:
  - statement about resources
  - subject – predicate – object
- W3C recommendation
- Basis for OWL (Web Ontology Language)
- Query language: SPARQL

LPG to RDF Comparison

LPG

```
(p:Customer {name:"Peter"})
      -[r:ORDERS]->
(o:Order {shipCity:"Bern"})
```

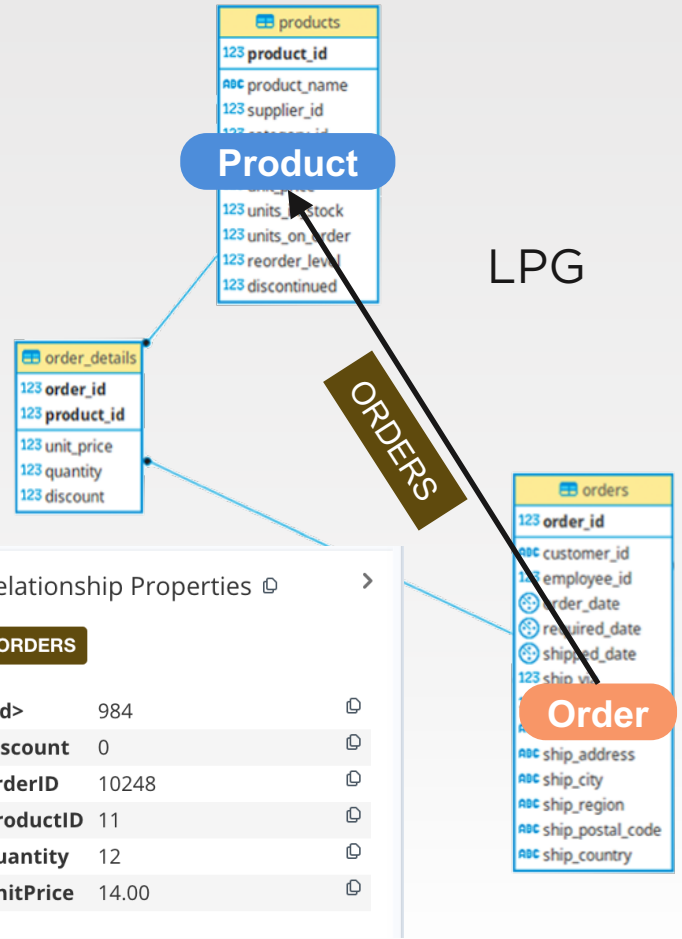
RDF

```
:customer1 rdf:type :Customer .
:customer1 :name "Peter" .
:customer :ORDERS :order 1 .
:order1 rdf:type :Order .
:order1 :shipCity "Bern" .
```

LPG & RDF Data Model Comparison

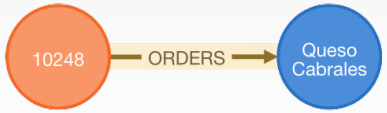
LPG & RDF graph models differ in certain points [Lassila et al.] [Hogan et al.] :

- **edge properties**
- **reification**
- multiple edge instances/values
- datatypes (arrays/lists)
- partitioning
- global identifiers
- schema and formal semantics



LPG

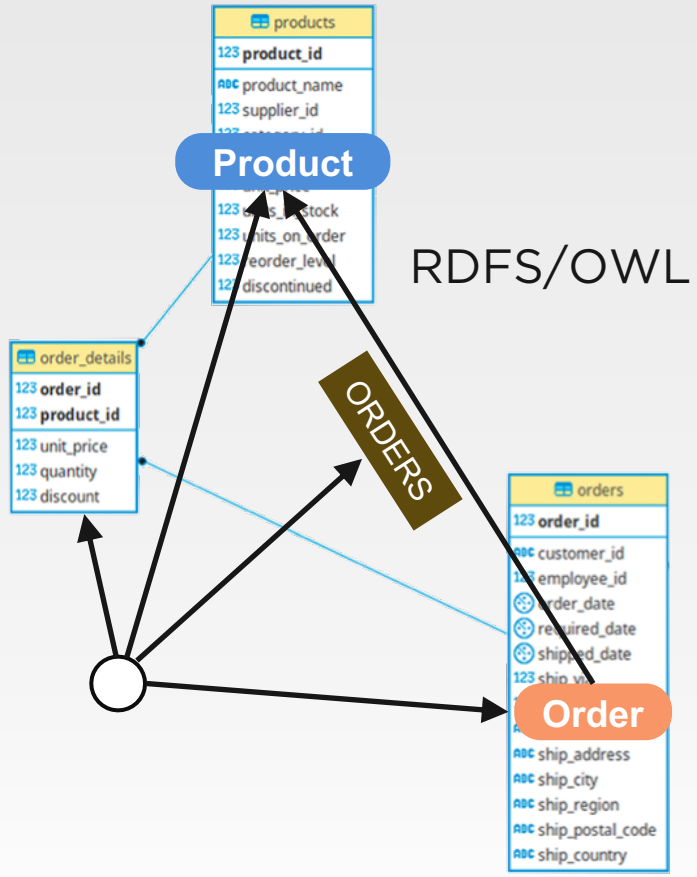
orderID	productID	unitPrice	quantity	discount
10248	11	14.00	12	0
10248	42	9.80	10	0
10248	72	34.80	5	0
10249	14	18.60	9	0
10249	51	42.40	40	0
10250	41	7.70	10	0
10250	51	42.40	35	0.15
10250	65	16.80	15	0.15
10251	22	16.80	6	0.05
10251	57	15.60	15	0.05
10251	65	16.80	20	0
10252	20	64.80	40	0.05
10252	33	2.00	25	0.05
10252	60	27.20	40	0
10253	31	10.00	20	0
10253	39	14.40	42	0



LPG & RDF Data Model Comparison

LPG & RDF graph models differ in certain points [Lassila et al.] [Hogan et al.] :

- **edge properties**
- **reification**
- multiple edge instances/values
- datatypes (arrays/lists)
- partitioning
- global identifiers
- schema and formal semantics



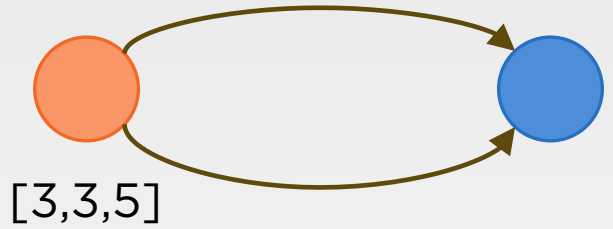
RDFS/OWL

orderID	productID	unitPrice	quantity	discount
10248	11	14.00	12	0
10248	42	9.80	10	0
10248	72	34.80	5	0
10249	14	18.60	9	0
10249	51	42.40	40	0
10250	41	7.70	10	0
10250	51	42.40	35	0.15
10250	65	16.80	15	0.15
10251	22	16.80	6	0.05
10251	57	15.60	15	0.05
10251	65	16.80	20	0
10252	20	64.80	40	0.05
10252	33	2.00	25	0.05
10252	60	27.20	40	0
10253	31	10.00	20	0
10253	39	14.40	42	0

LPG & RDF Data Model Comparison

LPG & RDF graph models differ in certain points [Lassila et al.] [Hogan et al.] :

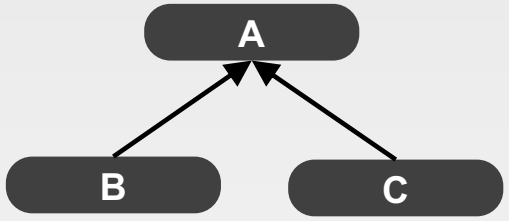
- edge properties
- reification
- **multiple edge instances/values**
- datatypes (arrays/lists)
- partitioning
- global identifiers
- schema and formal semantics



Semantic Schemas in LPG & RDF (I)

Goal: allow defining node categories of different levels of abstraction

- Hierarchy of categories



RDFS has a built-in typing mechanism: classes

```
:n rdf:type :B
        ⇩
:n rdf:type :A
```

Note: Triple Stores differ in evaluating this!

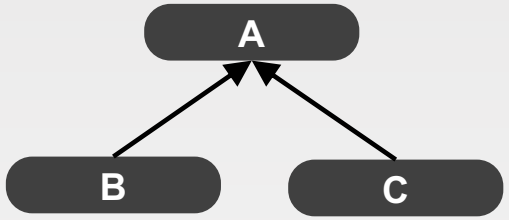
Table 7. The Semantics of Class Axioms

	If	then
cax-sco	T(?c_1, rdfs:subClassOf, ?c_2) T(?x, rdf:type, ?c_1)	T(?x, rdf:type, ?c_2)

Semantic Schemas in LPG & RDF (I)

Goal: allow defining n els of abstraction

- Hierarchy of catego



a) Semantics of axioms is **ignored**

b) Semantics of RDFS(+) is considered by **query-rewriting**

```
:n rdf:type ?c .
        ↓
:n rdf:type ?c1 .
?c1 rdfs:subClassOf* ?c .
```

c) Semantics is evaluated by **materialization**

ing mechanism: classes

```
:n rdf:type :B
        ⇩
:n rdf:type :A
```

Note: Triple Stores differ in evaluating this!

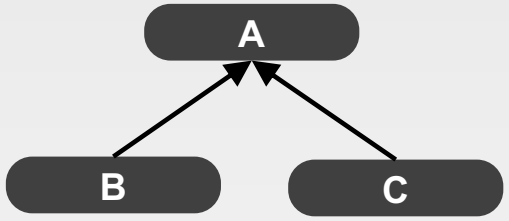
Table 7. The Semantics of Class Axioms

	If	then
cax-sco	T(?c_1, rdfs:subClassOf, ?c_2) T(?x, rdf:type, ?c_1)	T(?x, rdf:type, ?c_2)

Semantic Schemas in LPG & RDF (I)

Goal: allow defining node categories of different levels of abstraction

- Hierarchy of categories



LPG has no hierarchical typing mechanism

```
CREATE (n:B)
```

a) **Ignore** semantics

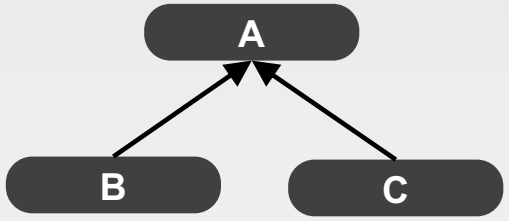
b) **Query-rewriting?** No representation of hierarchies / No rewriting engine
-> model in the graph with Neosemantics

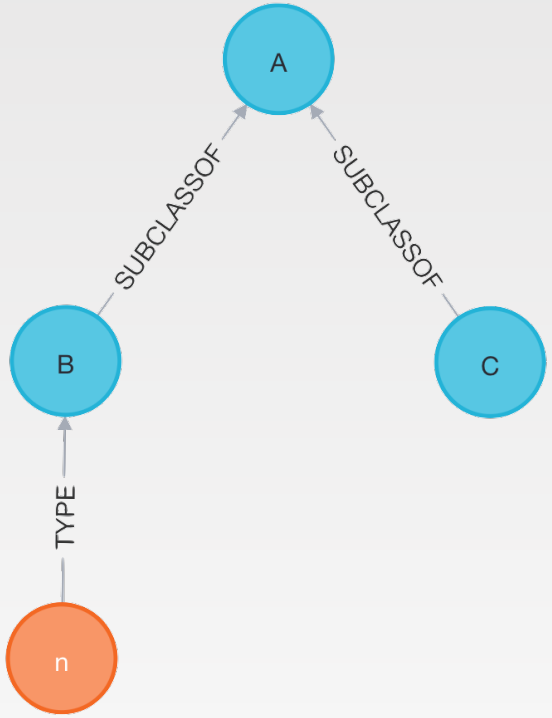
c) **Materialization** using multi-label approach: `MATCH (n:B) SET n:A`

Semantic Schemas in LPG & RDF (I)

Goal: allow defining node categories of different levels of abstraction

- Hierarchy of categories






```
:B rdfs:subClassOf :A .
:C rdfs:subClassOf :A .
:n rdf:type :B .
```

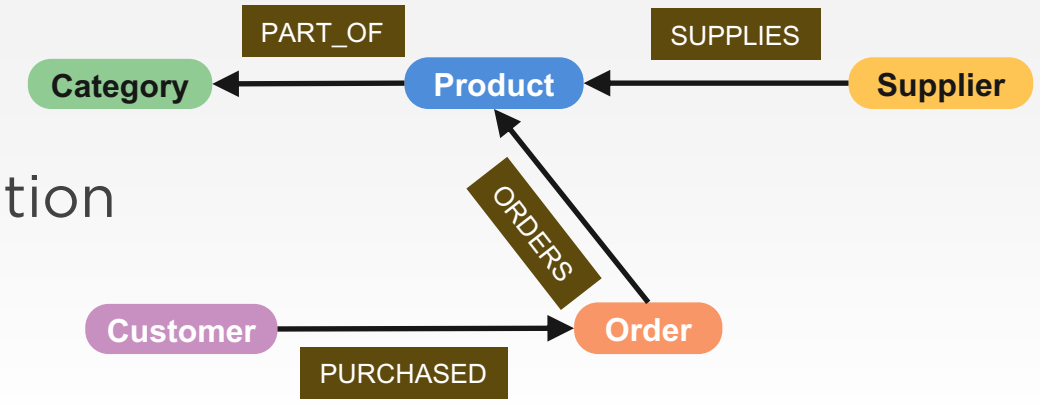
Neosemantics (n10s):

`handleRDFTypes: LABEL, **NODES**, LABELS_AND_NODES`

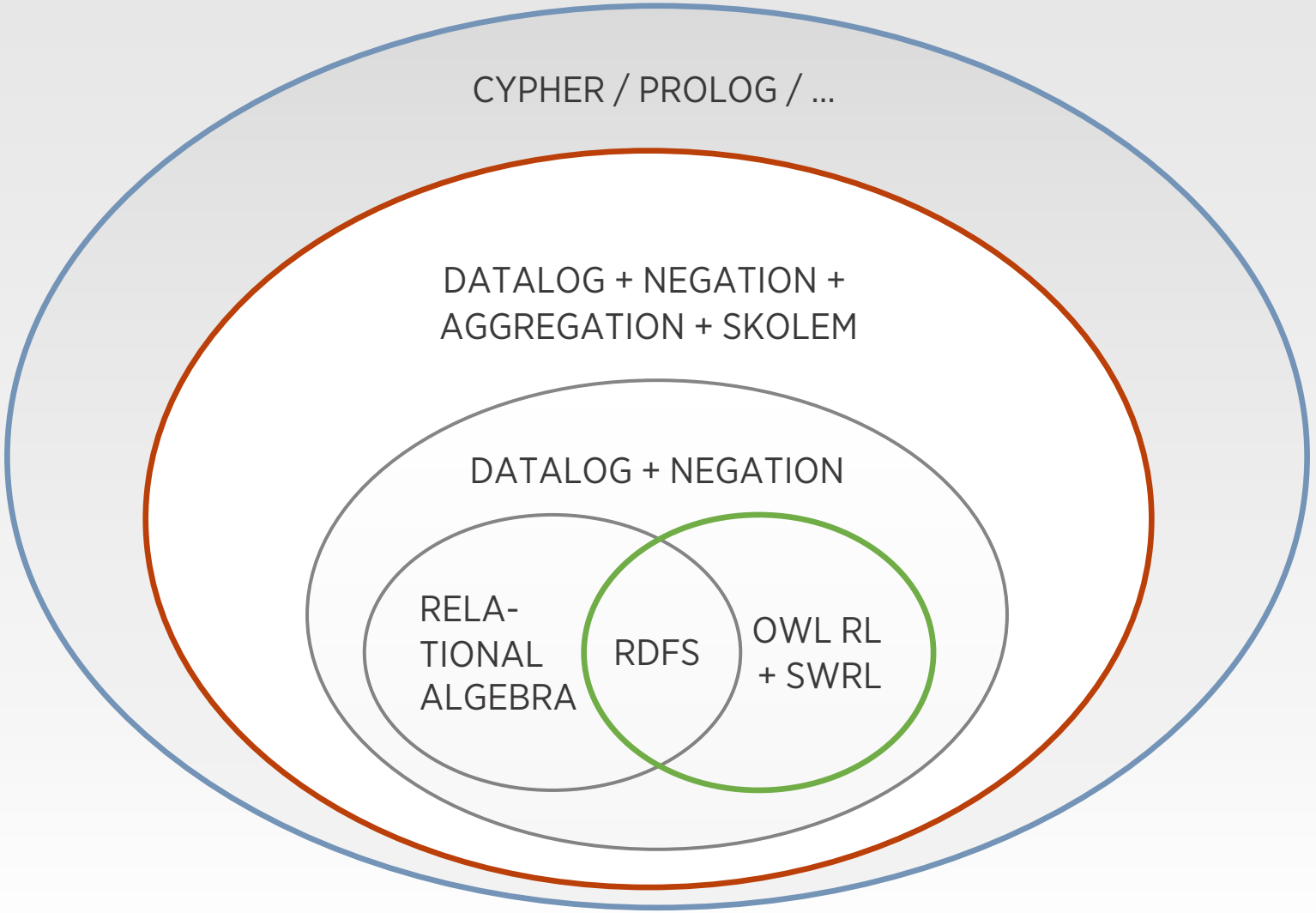
```
MATCH (n)-[:type]->(c) return c
union
match (n)-[:type]->()-[:subclass*]->(c) return c
```

Semantic Schemas in LPG & RDF (II)

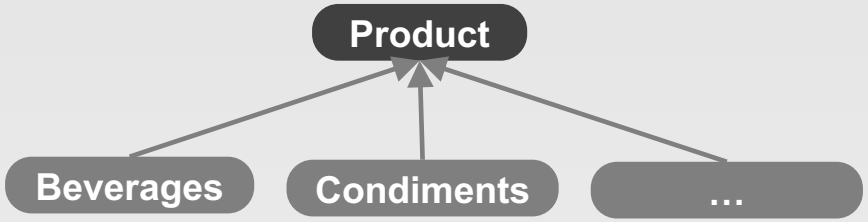
- Hierarchy of node categories
- Hierarchy of relationship types
  (RDFS: subPropertyOf; no such construct in LPG)
- Web Ontology Language:
  - Chains: `PURCHASED o ORDERS -> PURCHASED_PRODUCT`
  - Existentials: `Product and (PART_OF value Food) -> FoodProduct`
- Semantic Web Rule Language (SWRL) [Lawan & Rakib]:
  - Adds math and general chains (shapes)
- Datalog:
  - Adds new nodes, negation and aggregation



Expressivity Map



Adding a more specific category



- First intuition when talking about model refinement.
- Predefined filter
- Provides: easier selection
- Slight conceptual differences between:
  - Main taxonomy & facets
  - Partition & multiple classification
- Eventually big number of subcategories

Example: Single Subcategory

- OWL:

```
Declaration(Class(:DiscontinuedProduct))

AnnotationAssertion(
  rdfs:label
  :DiscontinuedProduct
  "Discontinued Product")

EquivalentClasses( # or SubClassOf
  ObjectIntersectionOf(
    :Product
    ObjectHasValue(:discontinued „true"))
  :DiscontinuedProduct)
```

Example: Single Subcategory

- Datalog:

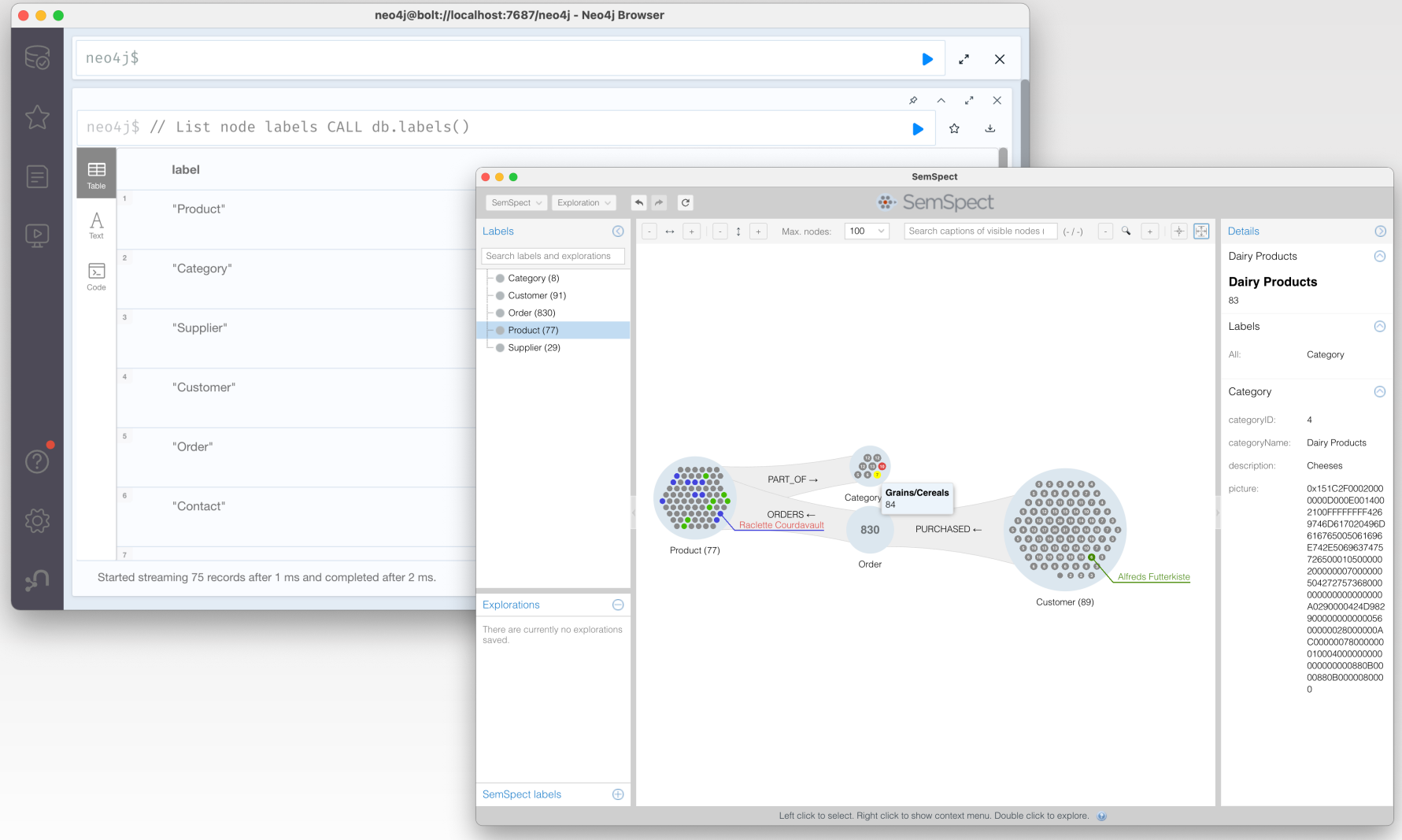
```
owl:Class[:DiscontinuedProduct].

rdfs:label[:DiscontinuedProduct, "Discontinued Product"].

owl:SubClassOf[:DiscontinuedProduct, :Product].

:DiscontinuedProduct[?product] :-
  :Product[?product],
  :discontinued[?product, "true"].
```

Example: Single Subcategory

- Cypher:

```
MATCH (n:Product {discontinued:true})
SET n:`Discontinued Product`;
```

Demo


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
neo4j$
neo4j$ // List node labels CALL db.labels()
Table
Text
Code
label
1 "Product"
2 "Category"
3 "Supplier"
4 "Customer"
5 "Order"
6 "Contact"
7
Started streaming 75 records after 1 ms and completed after 2 ms.
SemSpect
SemSpect
Exploration
Labels
Search labels and explorations
Category (8)
Customer (91)
Order (830)
Product (77)
Supplier (29)
Explorations
There are currently no explorations saved.
SemSpect labels
Max. nodes: 100
Search captions of visible nodes
PART_OF →
Category
Grains/Cereals
84
ORDERS ←
Raclette Courdavault
830
Order
PURCHASED ←
Product (77)
Customer (89)
Alfreds Futterkiste
Details
Dairy Products
Dairy Products
83
Labels
All: Category
Category
categoryID: 4
categoryName: Dairy Products
description: Cheeses
picture: 0x151C2F0002000 0000D000E001400 2100FFFFFFFF426 9746D617020496D 616765005061696 E742E5069637475 726500010500000 200000007000000 504272757368000 000000000000000 A0290000424D982 900000000000056 0000002800000A C00000078000000 010004000000000 000000000880B00 00880B000008000 0
Left click to select. Right click to show context menu. Double click to explore.

Example: Multiple Subcategories

- OWL: use program to generate axioms for each product category

```
Declaration(Class(:ProductCategory_{categoryId}))

AnnotationAssertion(
  rdfs:label
  :ProductCategory_{categoryId}
  "{categoryName} Product")

EquivalentClasses( # or SubClassOf
  ObjectIntersectionOf(
    :Product
    ObjectSomeValuesFrom(:PART_OF :Category_{categoryId}))
  :ProductCategory_{categoryId})
```

Example: Multiple Subcategories

- Datalog: use skolemization and string manipulation functions

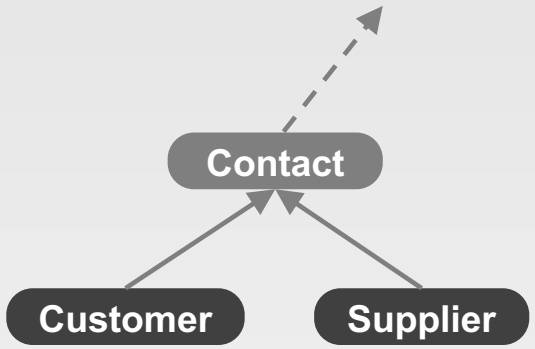
```
owl:Class[?categoryClass],
rdfs:label[?categoryClass, ?label],
owl:SubClassOf[?categoryClass, :Product]
:-
  :Category[?category],
  :categoryName[?category, ?categoryName],
  BIND(CONCAT(?categoryName," Category") AS ?categoryLabel),
  rdfox:SKOLEM("Class", ?category, ?categoryClass).

rdf:type[?product, ?categoryClass] :-
  :Product[?product],
  :PART_OF[?product, ?category],
  rdfox:SKOLEM("Class", ?category, ?categoryClass)
```

Example: Multiple Subcategories

- Cypher: use APOC library to add variable labels

```
MATCH (n:Product)-[:PART_OF]->(c:Category)
CALL apoc.create.addLabels(n, [c.categoryName+' Product'])
YIELD node
RETURN count(*);
```

Adding a more general category



- Group entities of different type that share characteristics (Example: similar entities from different data sources)
- Predefined union
- Provides: better overview & easier selection
- Eventually multi level: iterative process or using a predefined taxonomy (geographical classification, standard industrial classification, ...)

Example: Single Supercategory

- OWL

```
Declaration(Class(:Contact))

AnnotationAssertion(rdfs:label :Contact "Contact")

EquivalentClasses( # or SubClassOf
  ObjectUnionOf(:Customer :Supplier)
  :Contact)
```

Example: Single Supercategory

- Datalog

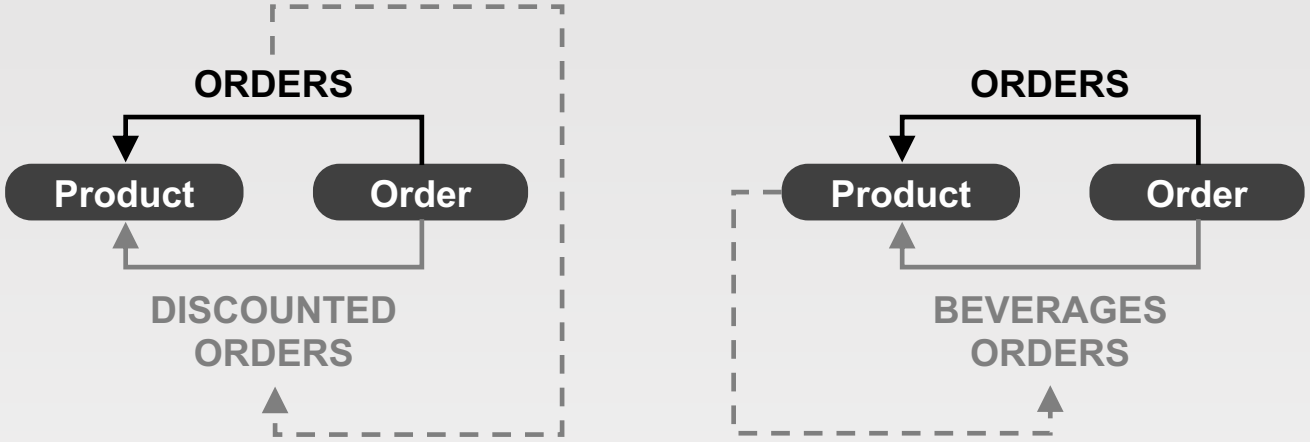
```
owl:Class[:Contact].

rdfs:label[:Contact, "Contact"].

:Contact[?customer] :- :Customer[?customer].
:Contact[?supplier] :- :Supplier[?supplier].
```

Example: Single Supercategory

- Cypher

```
MATCH (n:Customer) SET n:Contact;
MATCH (n:Supplier) SET n:Contact;
```

Adding more specific relations



- Specialized relation based on characteristics of the relation itself or of the entities it relates
- Predefined filter
- Provides: easier selection
- Helps hiding complex annotation pattern in RDF context

Example: Subproperty based on relation attributes

- OWL

```
Declaration(ObjectProperty(:DISCOUNTED_ORDERS))

AnnotationAssertion(rdfs:label :DISCOUNTED_ORDERS
"Discounted orders")

SubObjectPropertyOf(:DISCOUNTED_ORDERS :ORDERS)

# SWRL rules are necessary
# to insert Object Property Assertions
# (similar to datalog)
```

Example: Subproperty based on relation attributes

- Datalog

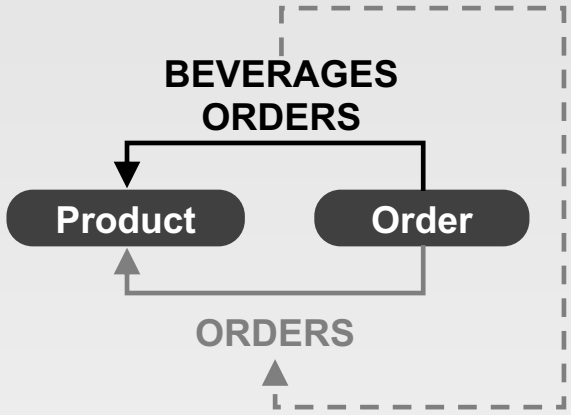
```
owl:ObjectProperty[:DISCOUNTED_ORDERS].

rdfs:label[:DISCOUNTED_ORDERS, "Discounted orders"].

owl:SubObjectPropertyOf(:DISCOUNTED_ORDERS :ORDERS)

:DISCOUNTED_ORDERS[?order, ?product]
:-
  :ORDERS[?order, ?product],
  owl:annotatedProperty[?reification, :ORDERS],
  owl:annotatedSource[?reification, ?product],
  owl:annotatedTarget[?reification, ?order],
  :discount[?reification, ?discount],
  FILTER(?discount > 0).
```

Example: Subproperty based on relation attributes

- Cypher

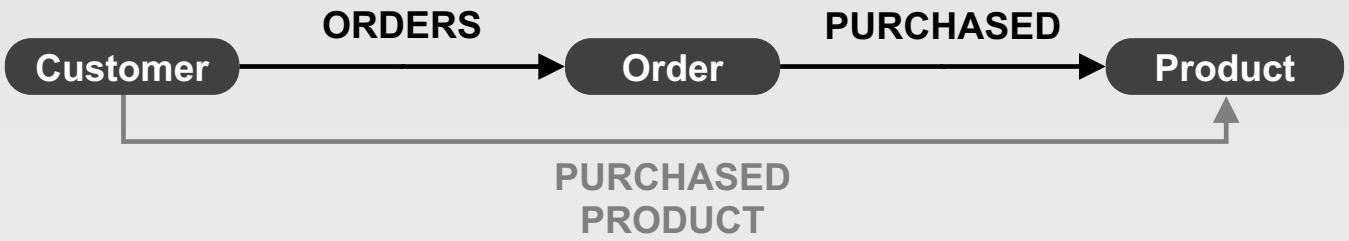
```
MATCH (o:Order)-[r:ORDERS]->(p:Product)
WHERE r.discount <> '0'
CALL apoc.create.relationship(
        o, 'DISCOUNTED_ORDERS‘, {discount:r.discount}, p)
YIELD rel
RETURN COUNT(*)
```

Adding more general relations



- Predefined union
- Provides: better overview & easier selection
- Eventually multi level: iterative process or using a predefined relation taxonomy

Adding a shortcut / materializing a partial query



- Infer a new relation based on existing relations
- Provides: direct access to remote entities
- Examples:
  - Hide details not relevant to the user
    (intermediary nodes introduced by reification)
  - Materialization of transitive relations
    (all parts in a subsystem of a BOM)

Example: Shortcut

- OWL

```
Declaration(ObjectProperty(:PURCHASED_PRODUCT))

AnnotationAssertion(rdfs:label :ORDERS "Purchased
product")

SubObjectPropertyOf(
  ObjectPropertyChain(:PURCHASED :ORDERS)
  :PURCHASED_PRODUCT)
```

Example: Shortcut

- Datalog

```
owl:ObjectProperty[:PURCHASED_PRODUCT].

rdfs:label[:PURCHASED_PRODUCT, "Purchased product"].

:PURCHASED_PRODUCT[?customer, ?product]
:-
  :PURCHASED[?customer, ?order],
  :ORDERS[?order, ?product].

What about the information of the order node?

Let us add the year of the order..
```

Example: Shortcut

- Datalog

```
owl:ObjectProperty[:PURCHASED_PRODUCT].

rdfs:label[:PURCHASED_PRODUCT, "Purchased product"].

:PURCHASED_PRODUCT[?customer, ?product],
owl:annotatedProperty[?annotation, :PURCHASED_PRODUCT],
owl:annotatedTarget[?annotation, ?customer],
owl:annotatedSource[?annotation, ?product],
:orderYear[?annotation, ?year]
:-
  :PURCHASED[?customer, ?order],
  :ORDERS[?order, ?product],
  :orderDate[?order, ?date],
  BIND(SUBSTR(?date,0,4) AS ?year),
  rdfox:SKOLEM(”Annotation", :PURCHASED_PRODUCT, ?customer,
?product, ?annotation).
```

Example: Superproperty

- Cypher

```
MATCH (c:Customer)-[:PURCHASED]->(o:Order)
   -[:ORDERS]->(p:Product)
WITH c, p, COLLECT(DISTINCT(LEFT(o.orderDate,4))) AS y
CALL apoc.create.relationship(
   c, 'PURCHASED_PRODUCT', {year:y}, p)
YIELD rel
RETURN COUNT(*);
```

Wrap-Up

- Adding schema to KGs brings
  - Domain knowledge into the graph
  - Various levels of abstractions/views
  - Query performance
- Various schema options:
  - Hierarchical node categories / relationships
  - OWL rules & SWRL
  - Datalog
- Tool support:
  - SemSpect

References

[Battaglia et al.] Relational inductive biases, deep learning, and graph networks, 2018, https://arxiv.org/abs/1806.01261

[Lassila et al.] Graph? Yes! Which one? Help! Lassila, Schmidt, Bebee, Bechberger, Broekema, Khandelwal, Lawrence, Sharda, Thompson, 1st Workshop on Squaring the Circle on Graphs (SCG2021), SEMANTiCS 2021, https://arxiv.org/abs/2110.13348

[Hogan et al.] Knowledge Graphs, ACM Computing Surveys, Vol. 54, No. 4, Article 71, 2021 https://arxiv.org/abs/2003.02320

[Lawan&Rakib] The Semantic Web Rule Language Expressiveness Extensions – A Survey, 2019 https://arxiv.org/abs/1903.11723