Easter holidays

next lecture Friday 25. April

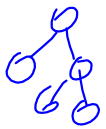
- Presentation on API

# Status 4. April 2014

- Generic Scenario and Use Case ✓
- Verbal expression of query ✓
- Decision on
  Protégé 3.5 with SWRL SQWRL OR Protégé 4.3 with SPARQL ✓
- Working on: apply SWRL // SPARQL on ontology
  Andreas, Rozina
- Upcoming: Interface to a programm to work with the knowledge
  - OWL-API
  - Jena-API ... } 28. Apr. ... Mai
- Final: Presentation of your integrated solution
  Mid May

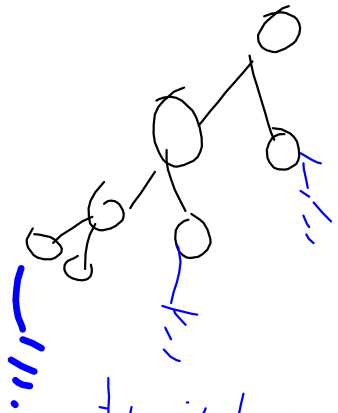
# Real time analysis

(1) (Marcel) Continuous C-SPARQL



Example:
train network
← schedule, train set
↳ "real time info"
Programm

(2) knowledge space



typical update
1-/day 1/hour

API
=

Program with real-time information
1/min
1/5 min

http://cwi.unik.no/wiki/Applying_SWRL_to_your_ontology

- Book (Ontologies & Rules): Media:UNIK4710_Chapter_6.pdf

## SWRL examples

- Is the person running?

```
Person(?person) ∧ hasSpeed(?person, ?speed) ∧ swrlb:greaterThanOrEqual(?speed, 10) → hasStatus(?person, running)
```

- Is Susana walking?

```
Person(Susana) ∧ hasSpeed(Susana, ?speed) ∧ swrlb:greaterThanOrEqual(?speed, 1) → hasStatus(Susana, walking)
```

- which songs the person likes

```
Person(?person) ∧ hasPreference(?person, ?prefer) ∧ Music(?music) ∧ hasStyle(?music, ?prefer) → like(?person, ?music)
```

- LowRisk state range for the persons who have a passive action and a heart rate between 81 – 120

```
Person(?p) ∧ LowRiskState(?n) ∧ Passive(?y) ∧ hasAction(?x, ?y) ∧ hasHeartRateOf(?x, ?z)∧
swrlb:greatherThanOrEqual(?z, 81)
∧ swrlb:lessThanOrEqual(?z, 120) → hasHealthCondition(?x, ?n)
```

## Ontology examples

- OWL ontologies Code :
  - http://www.co-ode.org/ontologies/
  - http://protege.stanford.edu/plugins/owl/ontologies.html
  - http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary

## RDF examples

- RDF Code: http://justinian.leibnizcenter.org/MetaLex/metalex-cen.owl

Query in Protégé 4 :- SPARQL

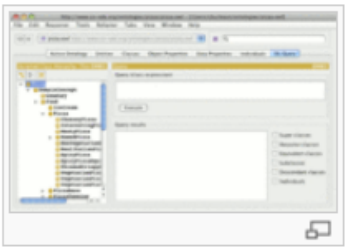
- syntax based

(2) - DL Query tab

- more graphics based

(2)

## DL Query tab



The DL Query tab provides a powerful and easy-to-use feature for searching a classified ontology. It is a standard Protégé 4 plugin available both as a tab and also as a view widget that can be positioned into any other tab. The query language (*class expression*) supported by the plugin is based on the Manchester OWL syntax, a user-friendly syntax for OWL DL that is fundamentally based on collecting all information about a particular class, property, or individual into a single construct, called a frame.

more intuitiv

(7)

http://answers.semanticweb.com/questions/26501/owl2query-plugin-for-protege-43

But this, does: PREFIX pre0:http://www.onto-au.edu/ontologias/estudiante-generico# SELECT ?x ?y WHERE { ?x pre0:tieneHabilidad ?y .}

This is valid even if you have checked your ontology IRI - prefix line in the upper grid.

/wiki/DLQueryTab

class in the query like this:

Of course, these are extremely simple queries; the Manchester syntax is much more capable.

Literal constants can be expressed with type by using ^^ and then the type:

**hasAge value "21"^^long**

Or, a more general expression that uses type:

**hasAge some int**

Following are just a few more examples to get you going:

**hasChild some Man**

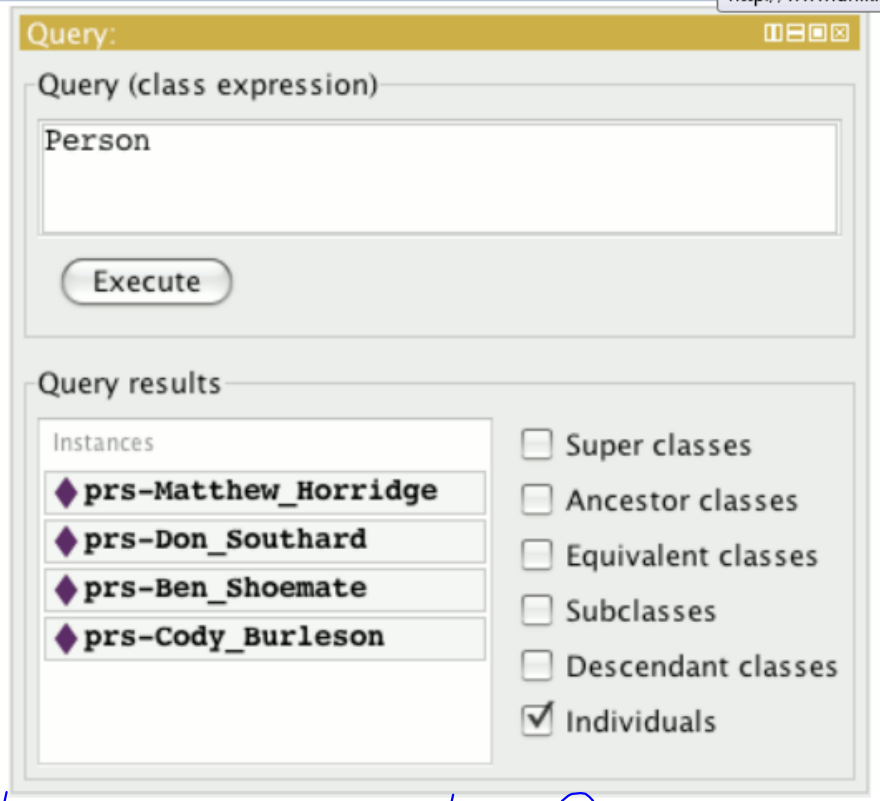
**hasSibling only Woman**

**hasCountryOfOrigin value England**

**hasChild min 3**

**hasChild exactly 3**

**hasChild max 3**




1) translate verbal query → DL Query

↳ 2.1. what is the syntax in SPARQL

2.2. API to use the DL Query

# First

- Protêgê 4.3
- Reasoner
  - The «quality test» of your onthology.
  - Crtl+R
  - Inconsistency
  - Broken Rules
    - Ann and Per have the same mother.
    - They are not siblings.
      - Illegal rule

Hermit

# SPARQL Syntax

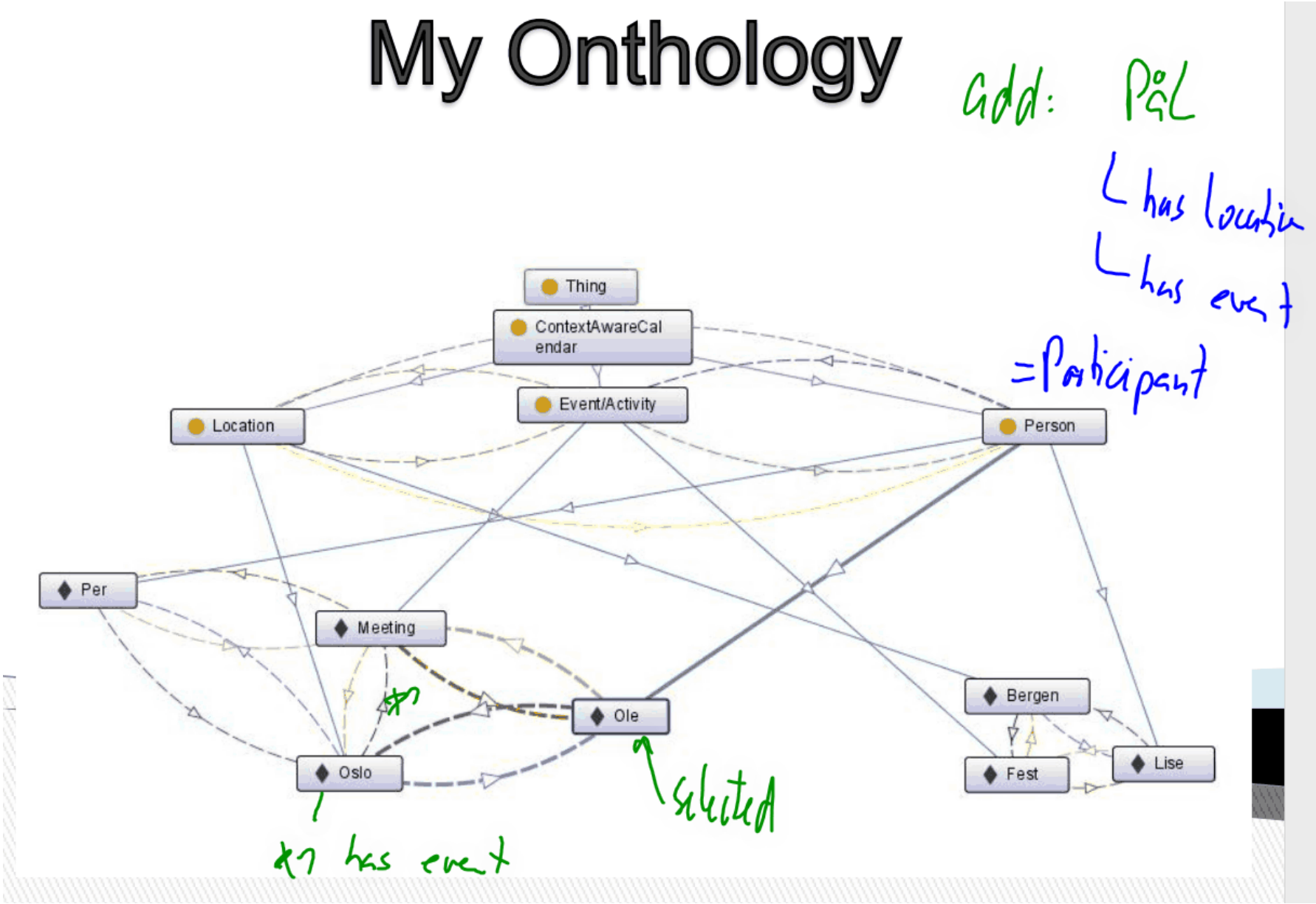
- PREFIX
  - What Schema to use?
    - Classes with attributes.
- SELECT
  - What do you want?
- WHERE
  - Matches your criteria
- Ex.
  - SELECT{cars}
  - WHERE{Colour is «Blue»}

RDF: RFDS: ...
Can create own ":"

Select all people

select all people with age

# My Onthology



# Find all Instances/Individuals

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/andreas/ontologies/2014/2/untitled-ontology-7#>
SELECT *
WHERE { ?x rdfs:subClassOf*  :ContextAwareCalendar .
        ?y rdf:type ?x }
```

" * " all available

Move Cursor

lists all subclasses

| x | |
|---|---|
| Person | Per |
| Person | Lise |
| Person | Ole |
| Location | Oslo |
| Location | Bergen |
| Event/Activity | Fest |
| Event/Activity | Meeting |

Same as previous, asks only for people

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/andreas/ontologies/2014/2/untitled-ontology-7#>
SELECT *
                WHERE { ?x rdfs:subClassOf* :ContextAwareCalendar .
                        ?y rdf:type ?x .
                         ?y rdf:type :Person .
                       }
```

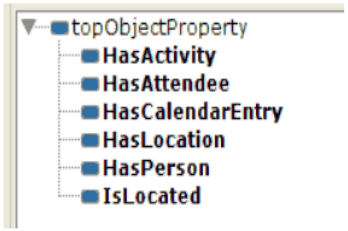
| x | |
|---|---|
| Person | Per |
| Person | Lise |
| Person | Ole |

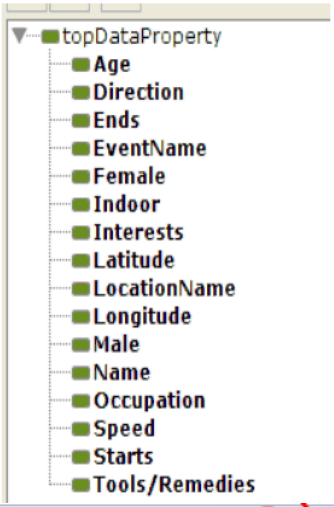
# Who is in Oslo?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT  distinct ?instance ?property ?object
WHERE { ?instance a ?Person .
        ?instance ?property ?object .
        ?property a ?HasLocation .
        ?object a ?Oslo . }
        ORDER BY DESC(?property)
```

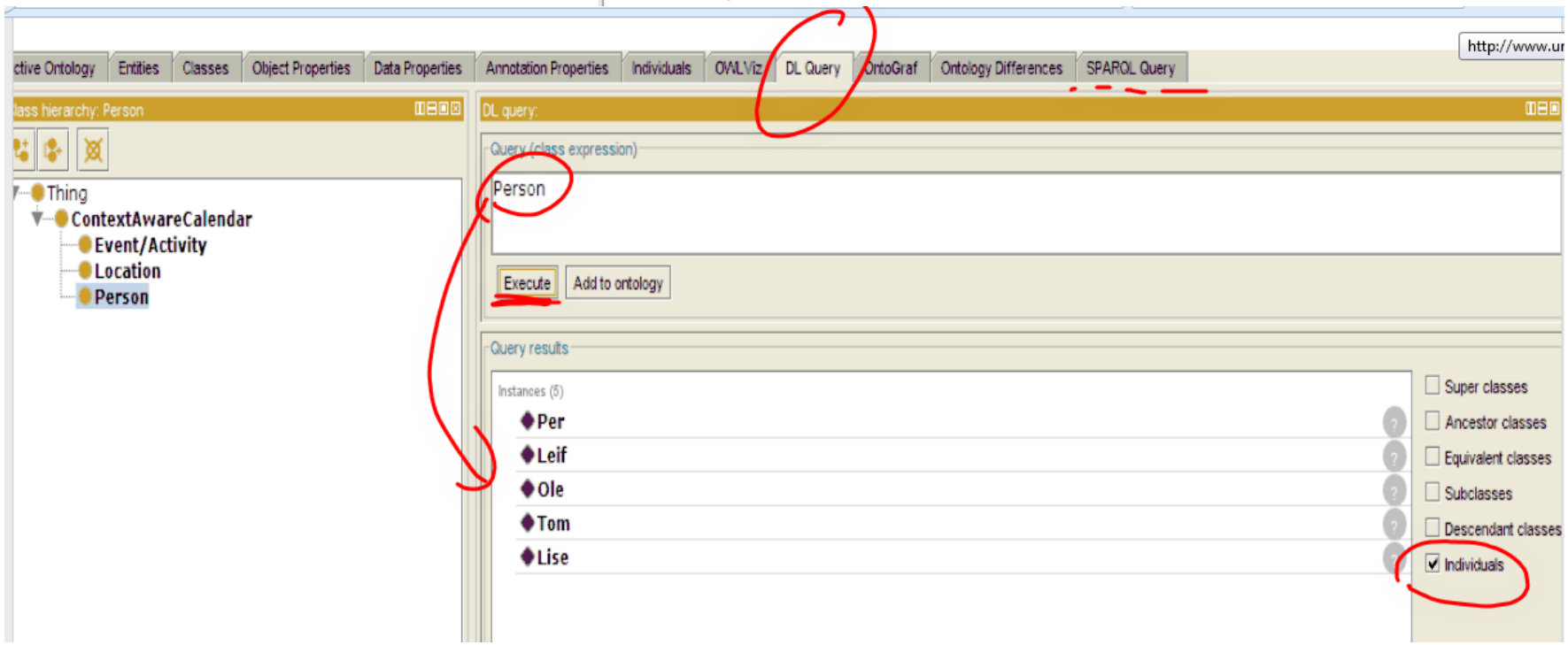
what is what?

| instance | property | object |
|---|---|---|
| Meeting | IsLocated | Oslo |
| Fest | IsLocated | Bergen |
| Bergen | HasPerson | Lise |
| Oslo | HasPerson | Ole |
| Oslo | HasPerson | Per |
| Per | HasLocation | Oslo |
| Ole | HasLocation | Oslo |
| Lise | HasLocation | Bergen |
| Ole | HasCalendarEntry | Meeting |
| Per | HasCalendarEntry | Meeting |
| Lise | HasCalendarEntry | Fest |
| Fest | HasAttendee | Lise |
| Meeting | HasAttendee | Per |



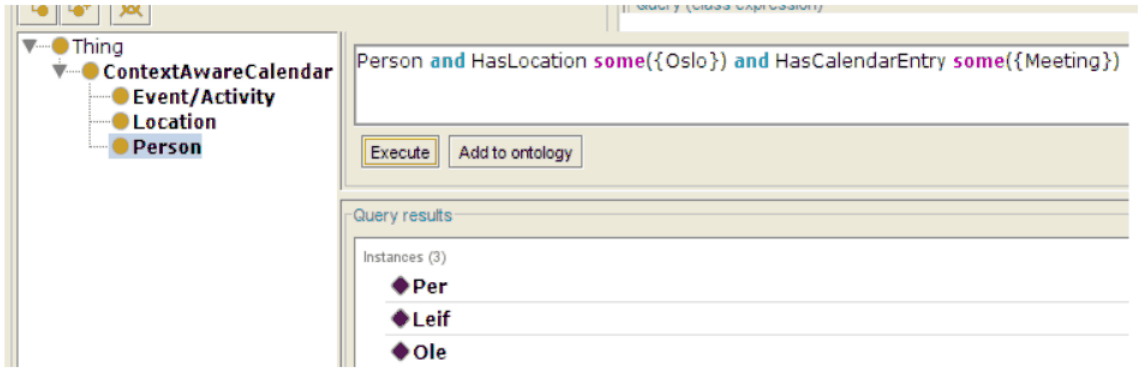




Note: earlier in lectures




some({Oslo}, {Bergen})

all people in Oslo or Bergen



Query (class expression)

```
Person and HasLocation some({Oslo}) and HasCalendarEntry some({Meeting})
or HasCalendarEntry some({Fest})
```

Execute | Add to ontology

Query results

Sub classes (0)

Instances (4)

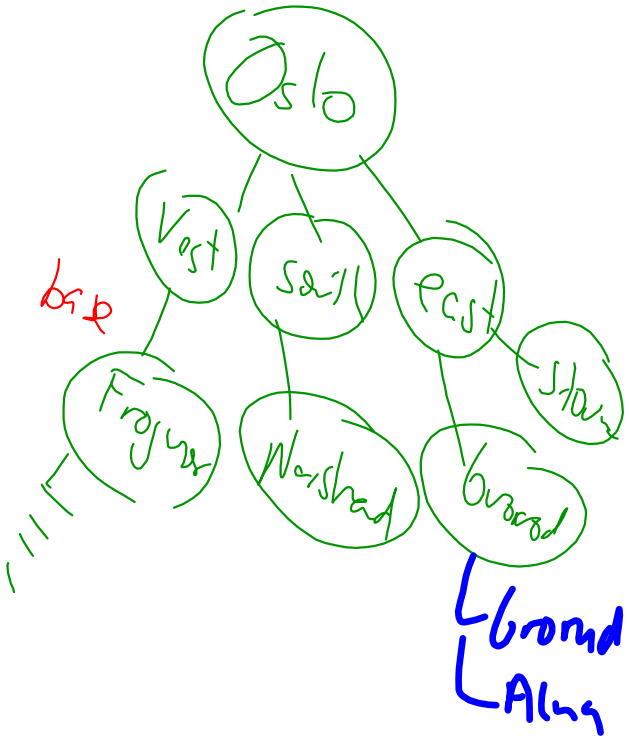
- Per
- Leif
- Ole
- Stein

Stein was added afterwards to knowledge base

Some ( {Meeting}, {party}, ... )

or

And syntax?



Next step: understand query syntax