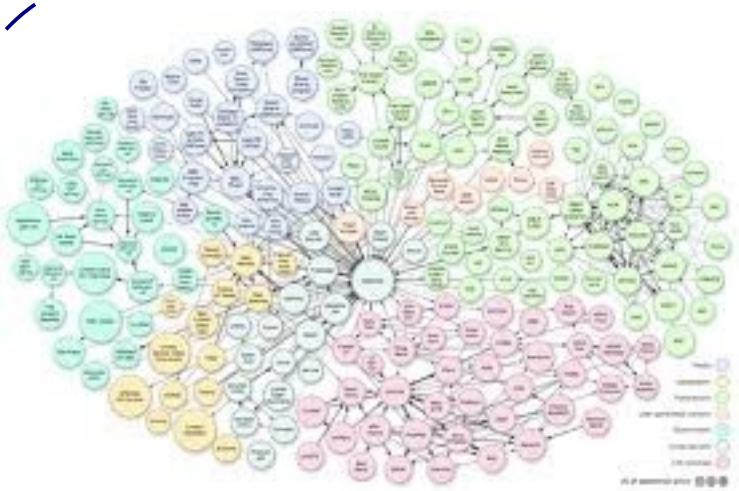
Property-based Typing for RDF-Access

– A Vision –

Stefan Scheglmann
Gerd Gröner

Publishing Linked Data on the Web - Status

1. Huge collections of open and governmental data are published as Linked Data



representation formalism?

2. Data enrichment
   - Meta data
   - Micro formats

for semantic search and exploration

RDF

Data model with formal semantics

- Flexible data representation
- Extendable
- Deal with incomplete data

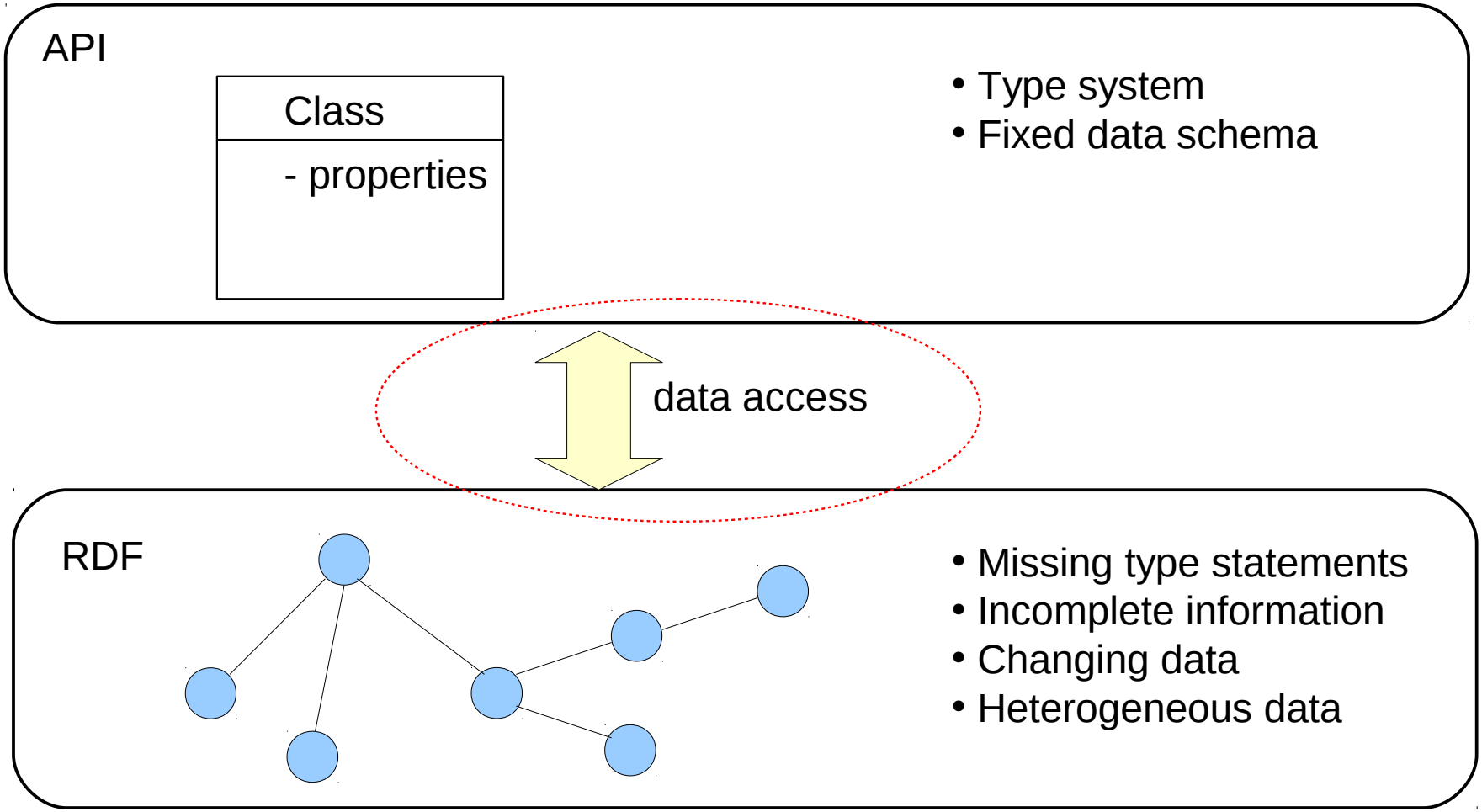
Linked Data

Best practice to publish structured data on the web

Observation: successful publishing and exploration of data on the Web

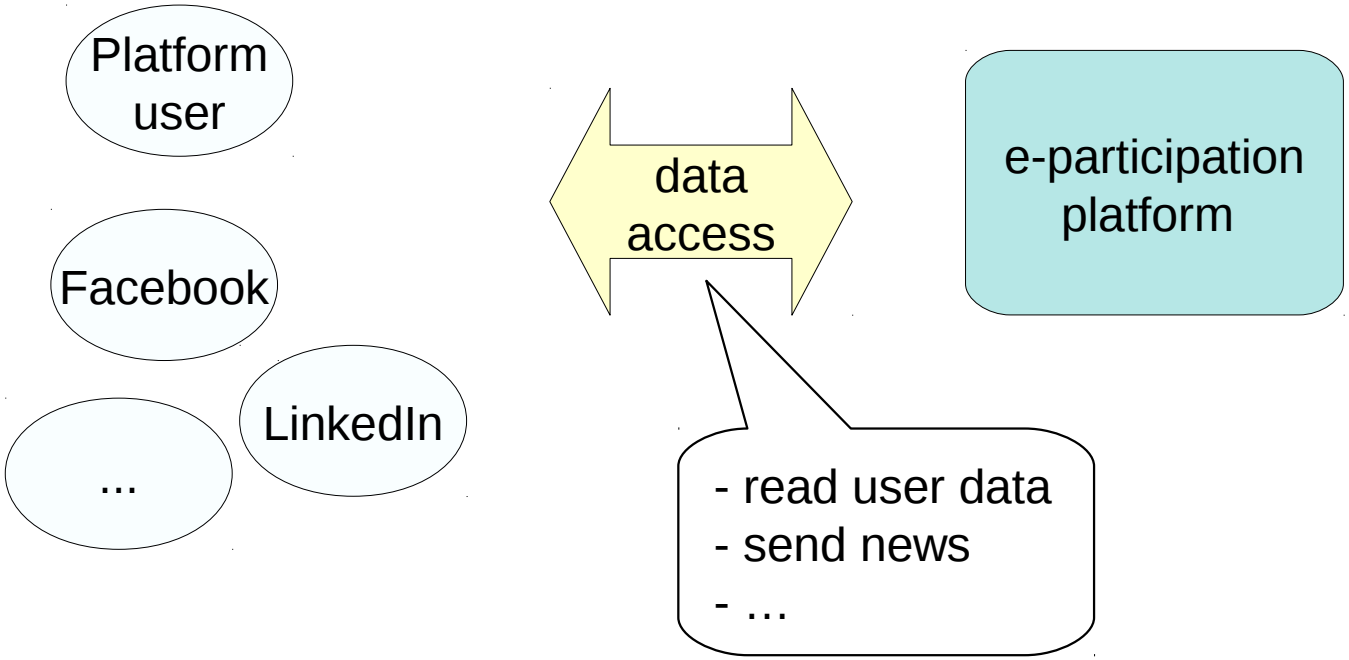
What is the problem?

- The flexible data model makes it difficult for applications to access the data



Example - Scenario

Users:
(Accounts, available user data)



Example

alice rdf:type ex:citizen.
bob rdf:type ex:citizen.
dave rdf:type ex:citizen.
eric rdf:type ex:citizen.

alice foaf:name "Alice".
alice foaf:mbox "alice@home.com".
alice foaf:interest <http://wikipedia.org/Literature>.

bob foaf:name "Bob".
bob foaf:mbox "bob@mail.com".
bob foaf:interest <http://wikipedia.org/Cinema>.

candy foaf:name "Candy".
candy foaf:mbox "candy@coldmail.com".

dave foaf:name "Dave".
dave foaf:mbox "dave@online.com".

API with methods:

Send mail to all citizen with ***name*** and ***mbox***!
→ Candy?

Modify/read ***interest*** of all citizen!
→ Dave?

Example (2)

alice rdf:type ex:citizen.
bob rdf:type ex:citizen.
dave rdf:type ex:citizen.
eric rdf:type ex:citizen.

alice foaf:name "Alice".
alice foaf:mbox "alice@home.com".
alice foaf:interest
<http://wikipedia.org/Literature>.

eric foaf:name "Eric".
eric ex:member
<http://cinemafriends.com>.
<http://cinemafriends.com> ex:interest
<http://wikipedia.org/Cinema>.

API with methods:

Find interest of citizen!
→ Eric?

How should a solution look like?

API with methods:

Send mail to all citizen with ***name*** and ***mbox***!
→ Candy?

There might be missing type statements

- access data based on properties (e.g., to define methods)s

Modify/read ***interest*** of all citizen!
→ Dave?

There might be missing properties

- access data with
  - Required and
  - optional properties

Find interest of citizen!
→ Eric?

There might be implicit properties.

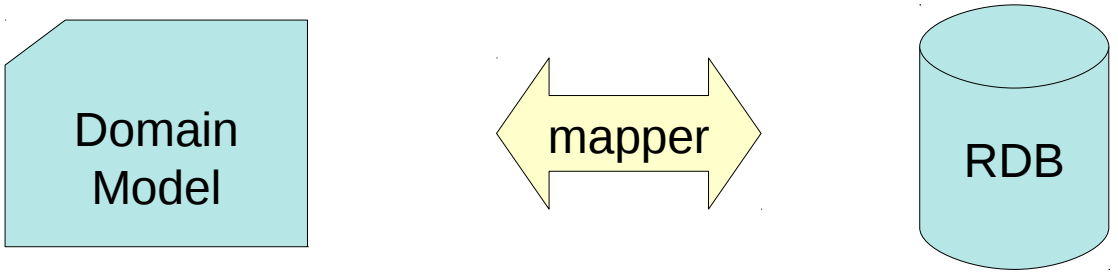
- access data with properties that are derived (by reasoning or navigation)

How to achieve a possible solution?

1. Object Triple mapping

Background: object relational mapping

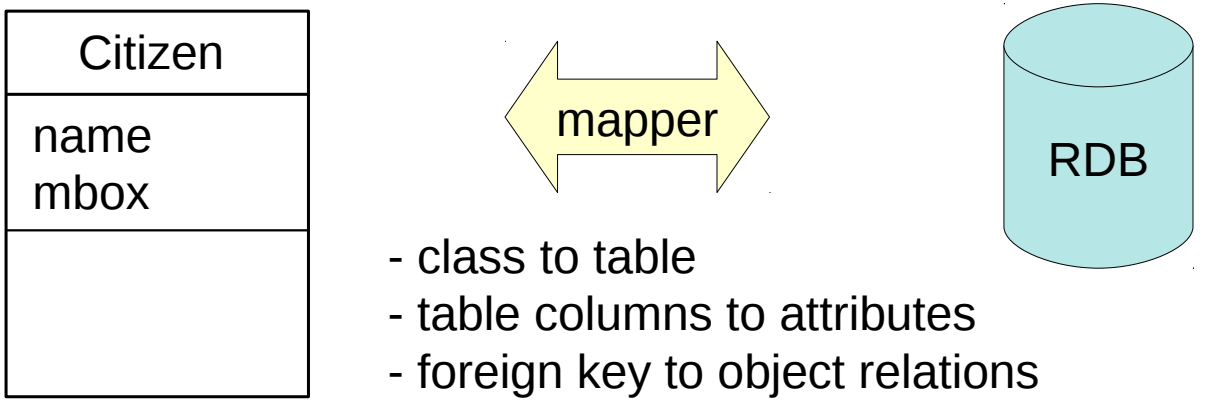
Design patterns:



Application development language

data

Example: Class to table



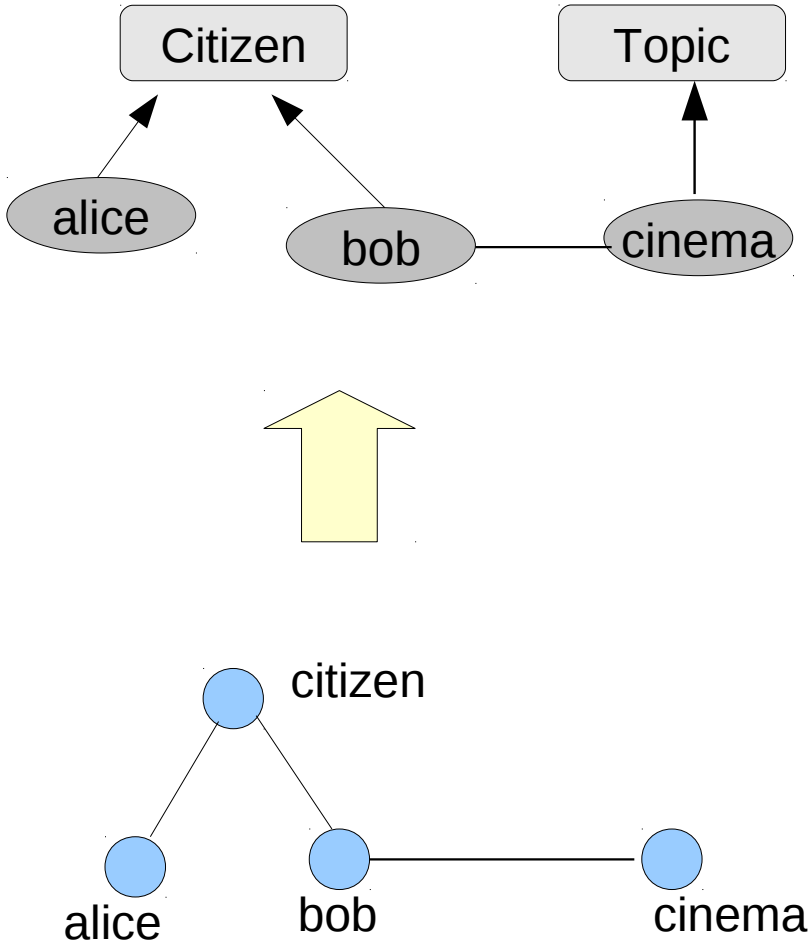
- class to table
- table columns to attributes
- foreign key to object relations
- every tuple to an object

Map Triples to Objects (2)

Basic Idea

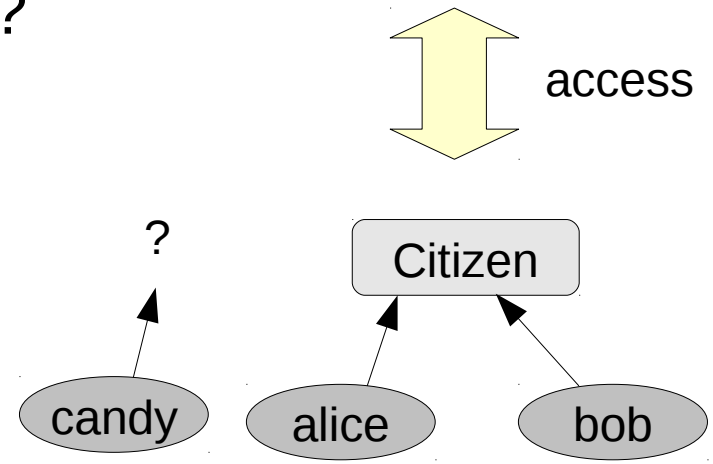
1. Define a class for each type

2. Triples to objects
   - resources to objects
   - predicates to attributes or relations



Map Triples to Objects (3)

… but does this solve the problem?



1. We need types for classes.

2. We can not state properties as optional

How to achieve a possible solution?

1. Object Triple mapping

2. Class definitions like in OWL

Constructs for property specifications of classes

$$Citizen \sqsubseteq \exists name.Name \sqcap \exists mbox.Mail$$

$$Citizen \sqsubseteq \forall interest.(Topic \sqcup TopicGroup)$$

$$\exists name.Name \sqcap \exists mbox.Mail \sqsubseteq Citizen$$

$$\exists name.\top \sqsubseteq Citizen$$

… does this solve our problem?

- not possible in RDF
- not realized in Linked Data

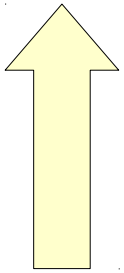
How to achieve a possible solution?

1. Object Triple mapping
2. Class definitions like in OWL
3. Extract schema

- **1. Step: Derive schema**

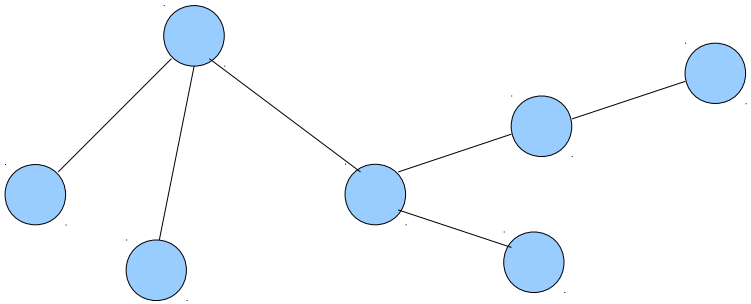
Schema is based on property statements

group instances that share some properties



schema extraction

RDF data



2. Step: Use Type Provider

- component that provides types, properties, and methods in programs
  - i.e., programmers get type information of entities (like databases) they work with

- F# offers built-in type providers and allows user defined providers

… does this solve our problem?

→ Schema is a snapshot

How to achieve a possible solution?

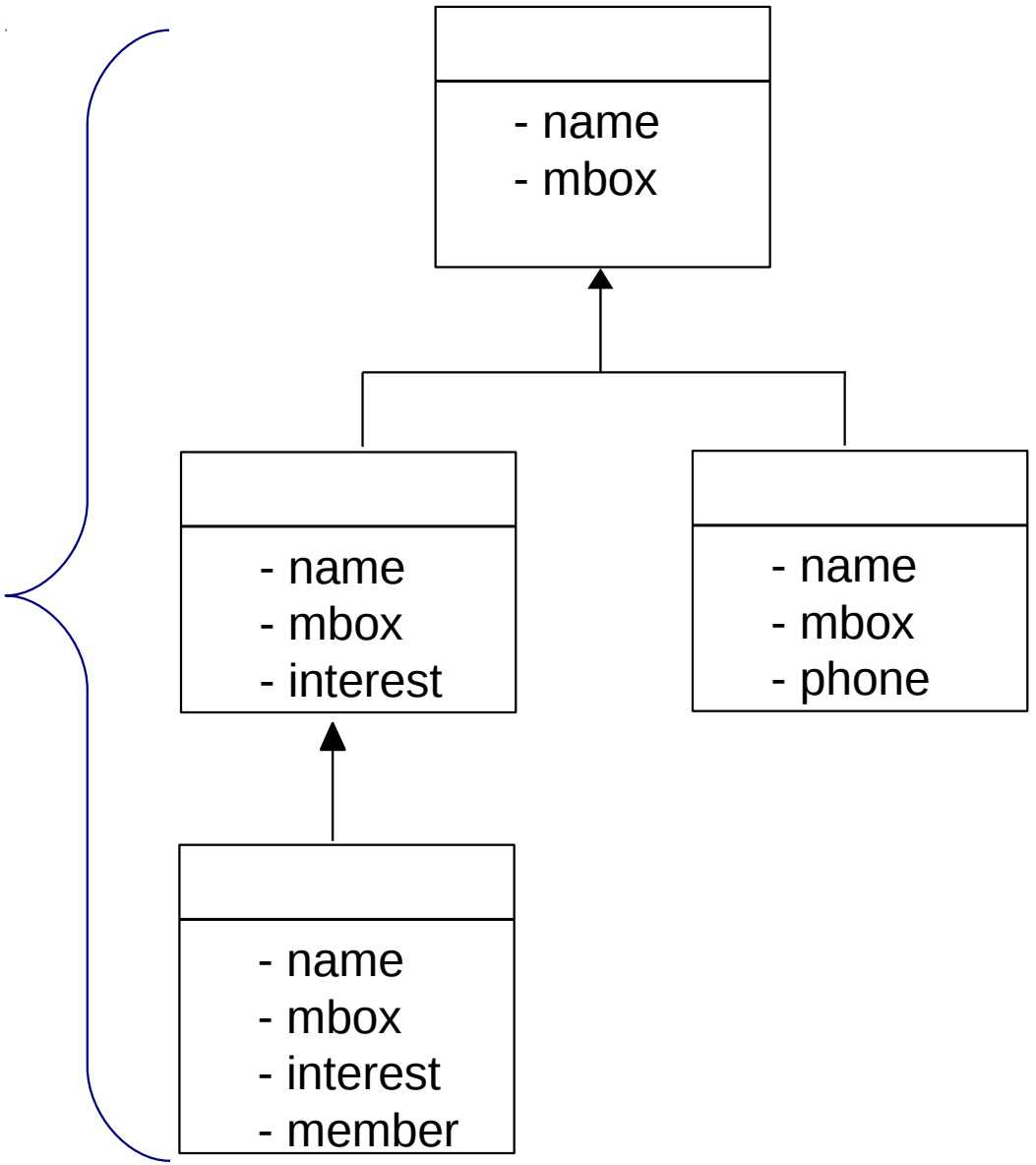
1. Object Triple mapping
2. Class definitions like in OWL
3. Extract schema

How should a solution look like?

Requirements for a solution

Property-based Types:

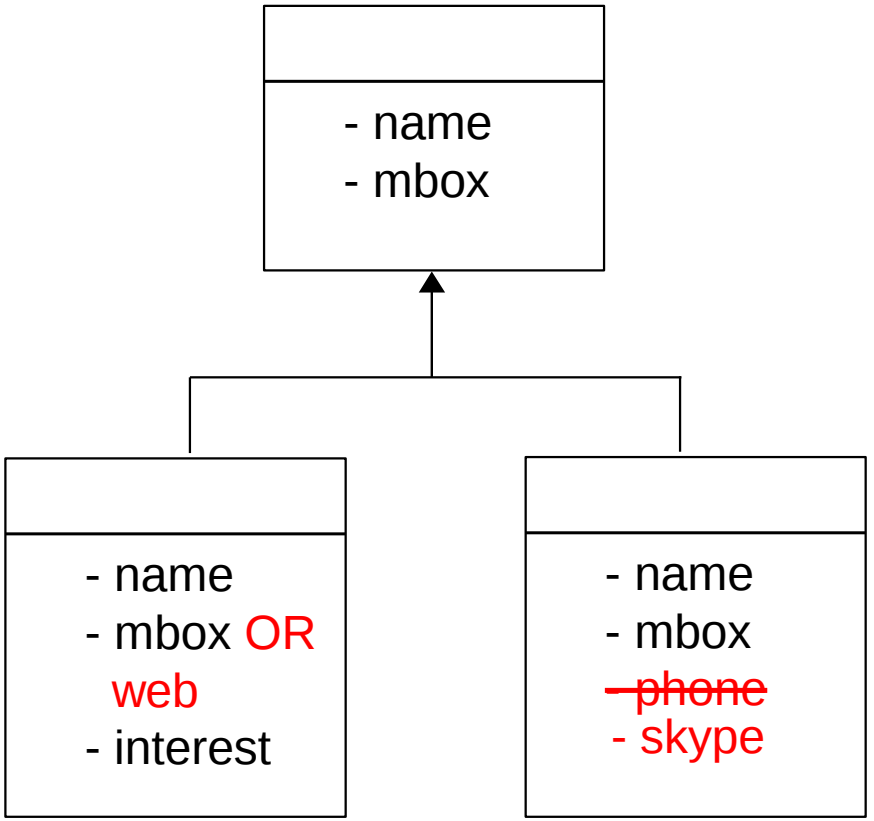
- hierarchies
- specialization



Requirements for a solution (2)

Flexible Type System

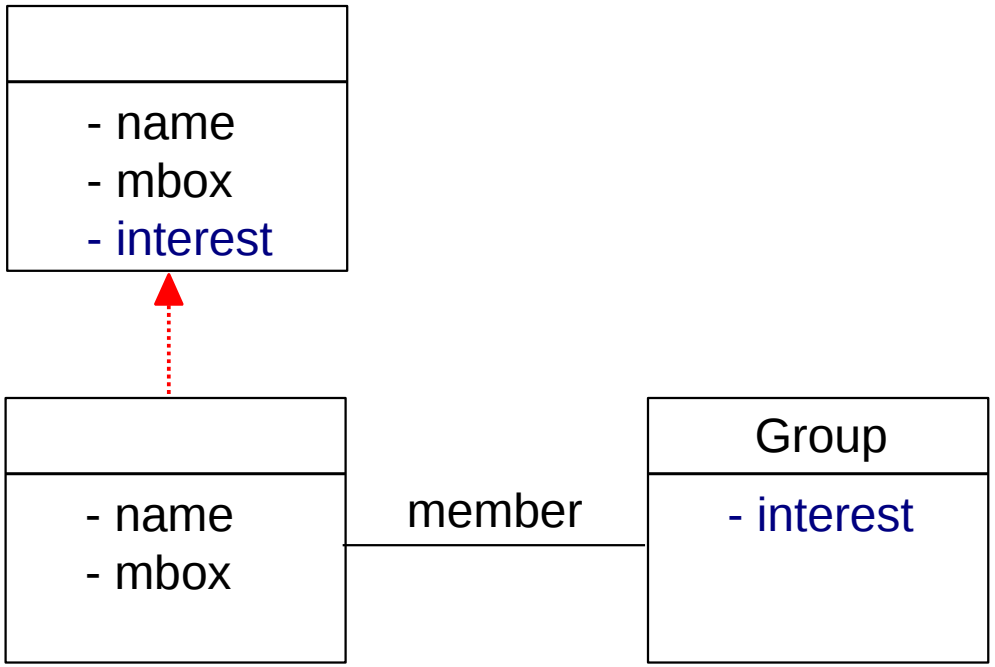
- Extendable



Requirements for a solution (3)

Inference:

- Derive properties
- Derive type



… or the trade-off between persistent data access and flexible representation

Build on types and extracted schema	Incomplete and flexible RDF data representation
• Object Triple mapping • Class specifications (for incomplete data) • Schema extraction and type provider (e.g., F#)	• Missing type statements • Property-centric • Extendable

… how to bring these “worlds” together?

Outline

How to achieve a possible solution?

1. Object Triple mapping
2. Class definitions like in OWL
3. Extract schema

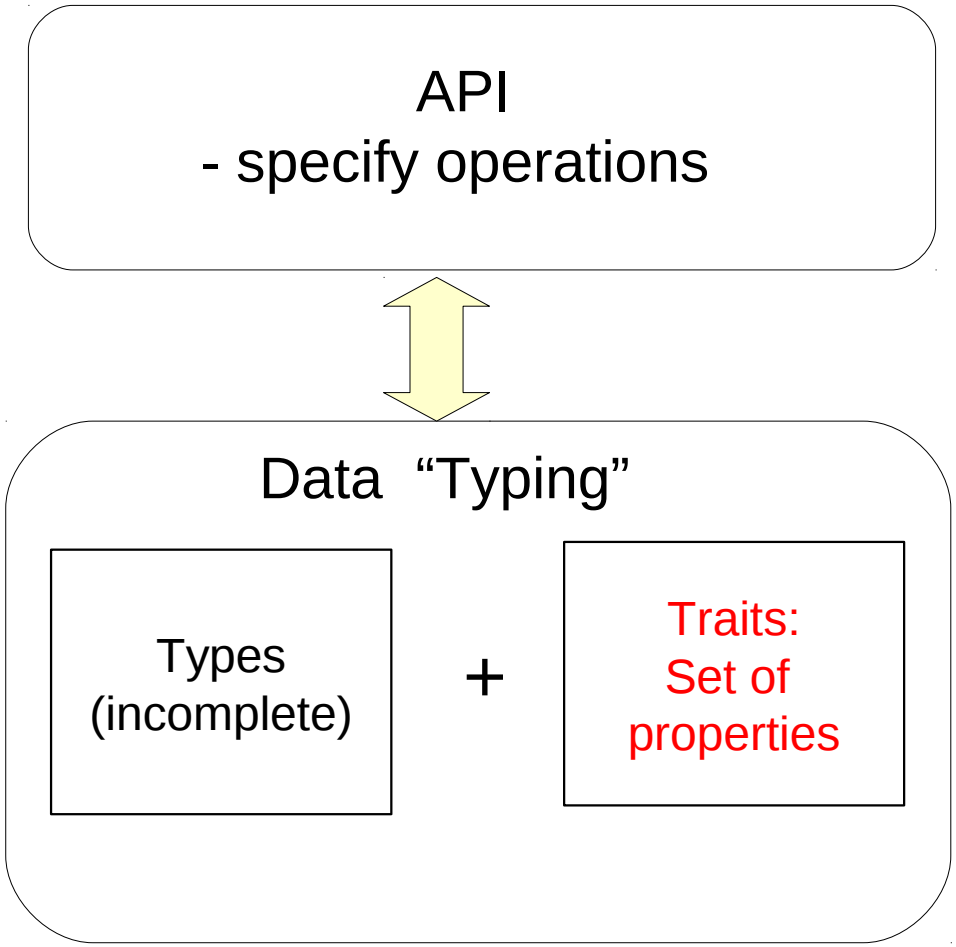
How should a solution look like?

Our vision / statement

New ***Programming Paradigm***:

"Property-based typing for RDF data access"

Basic idea:



Example (Property-based Access)

alice rdf:type ex:citizen.
bob rdf:type ex:citizen.
dave rdf:type ex:citizen.

alice foaf:name "Alice".
alice foaf:mbox "alice@home.com".
alice foaf:interest <http://wikipedia.org/Literature>

bob foaf:name "Bob".
bob foaf:mbox "bob@mail.com".
bob foaf:interest <http://wikipedia.org/Cinema>

candy foaf:name "Candy".
candy foaf:mbox "candy@coldmail.com".

dave foaf:name "Dave".
dave foaf:mbox "dave@online.com".

Traits:
- foaf:mbox
- foaf:name

~~Conclusion~~ … Discussion

New ***Programming Paradigm***:
"Property-based typing for RDF data access"

- What are the challenges in a property-based approach for programming the semantic Web?
- Which are the implications that solutions of these challenges could have?

Discussion: Challenges

Type hierarchies (property-based):

1. How to build a reliable type hierarchy based on property sets?
   i. Schema extraction
   ii. Formal concept analysis
   (for occurring property combinations)

2. How to combine property-based types and type statements (rdf:type)?

Incompleteness

1. How to interpret a missing property?
   - ➔ Missing or unknown?
2. How to achieve type inference (like in RDFS)?

Contracts

- Method and type contracts (based on property sets)

1. How to declare optional and required properties?
2. Can we support global and local contracts?

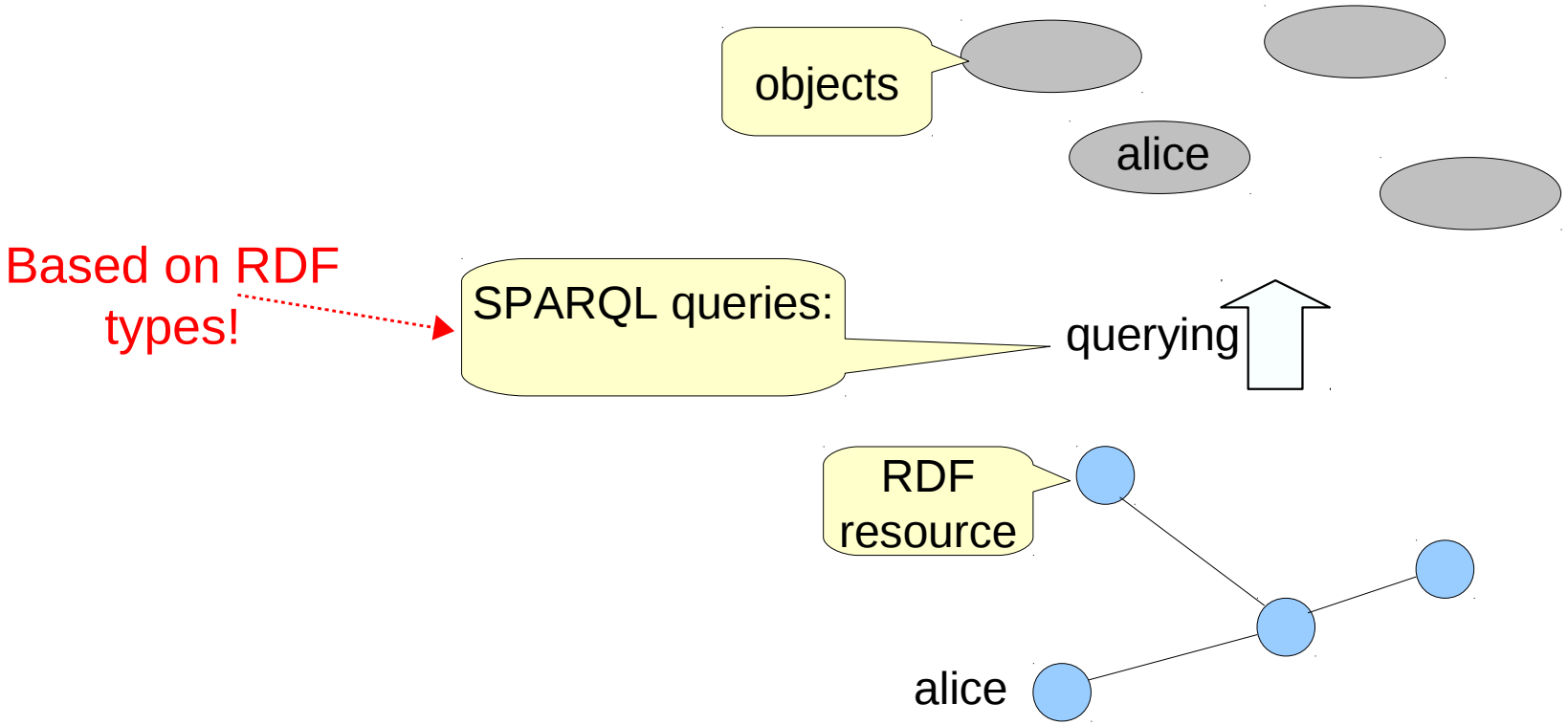
Requirements for Programming Access

- Efficient support for processing large data sets
- Alignment (or even integration) of programming languages to data models on the Web
- Flexible type system
  - Type refinement / specialization
  - Extendable types
- Deal with incompleteness
  - Missing types
  - Missing properties

Object-oriented semantic Web programming

ActiveRDF

- Object-oriented API for managing RDF data
  - Does not rely on a schema
- Basic Idea: represent RDF resources as objects



Object Triple Mapping:

- Idea: extend object relational mapping to object triple mapping
- Steps:

  1. Build a domain model in terms of classes (similar to a schema)
  2. Data mapper: mapping between resources and objects