A Protégé 4 Backend for Native OWL Persistence

June 25, 2009

Jörg Henss
Joachim Kleb
Stephan Grimm

FZI Research Center for Information Technology
at the University of Karlsruhe
Germany

Motivation

- Why do we need a Persistence Backend for Protégé 4?
  - Storage
  - Maintenance
  - Collaborative Work
- Why do we need a *new* Persistence Backend for Protégé 4?
  - Native support for OWL
  - It was missing ;-)

Nativeness

- By **nativeness** we understand:
  - Mapping OWL language constructs one-to-one to storage layer
- Triple Structure
  - RDF-Store
  - CLOS model
- Axiomatic view
  - Restrictions, cardinalities
  - OWL acts on **objects** not on **nodes**
    - E.g. blank nodes are only recognizable via URI in RDF
  - An object model for OWL is required

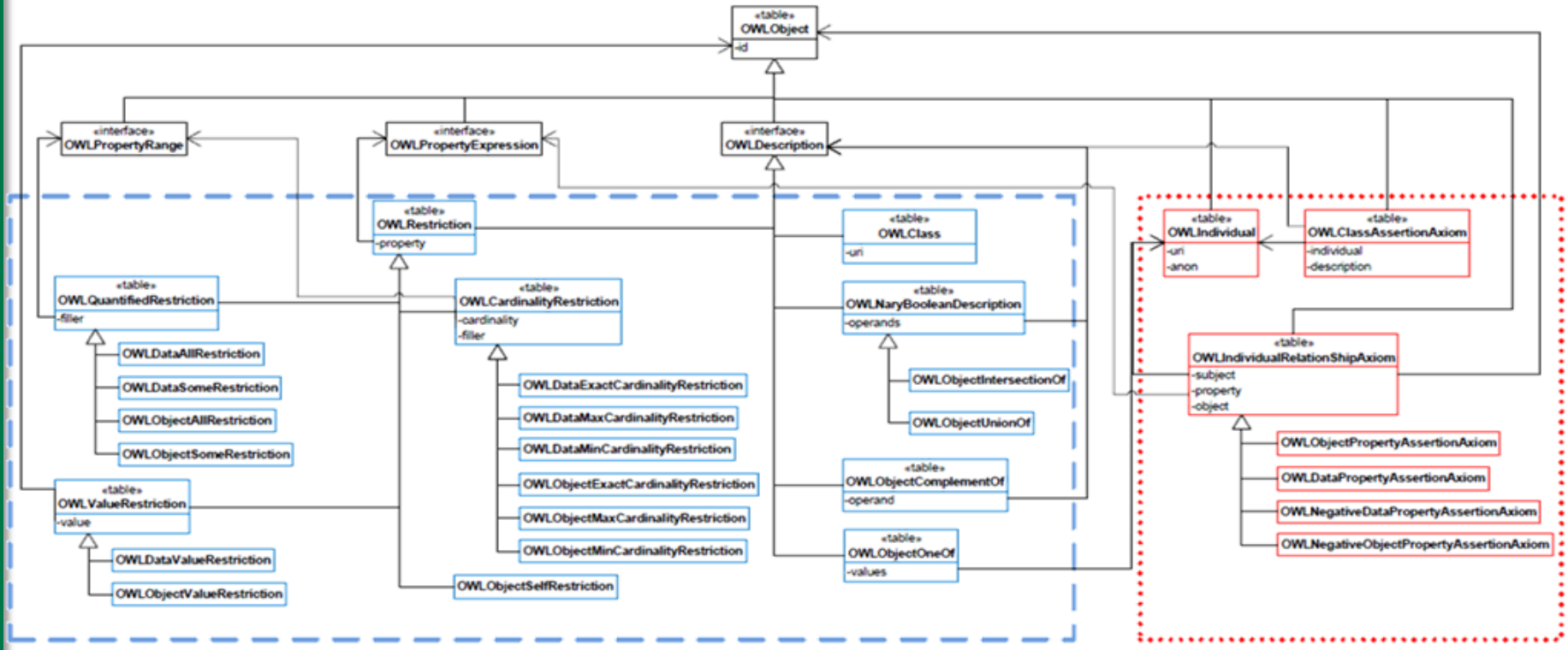
Schema Representation

- OWL as Objects
  - Concepts, Individuals, etc.
- OWL-API as Object Model for OWL
  - Java based API for OWL
  - Maintained by University of Manchester
  - OWL 2 ready
  - Protégé 4 is based upon
- Use of Object-Relational mapping for persistence
  - Stores object information in database
  - Restriction on necessary parts for Ontology Persistence
    - E.g. minimisation of redundancy

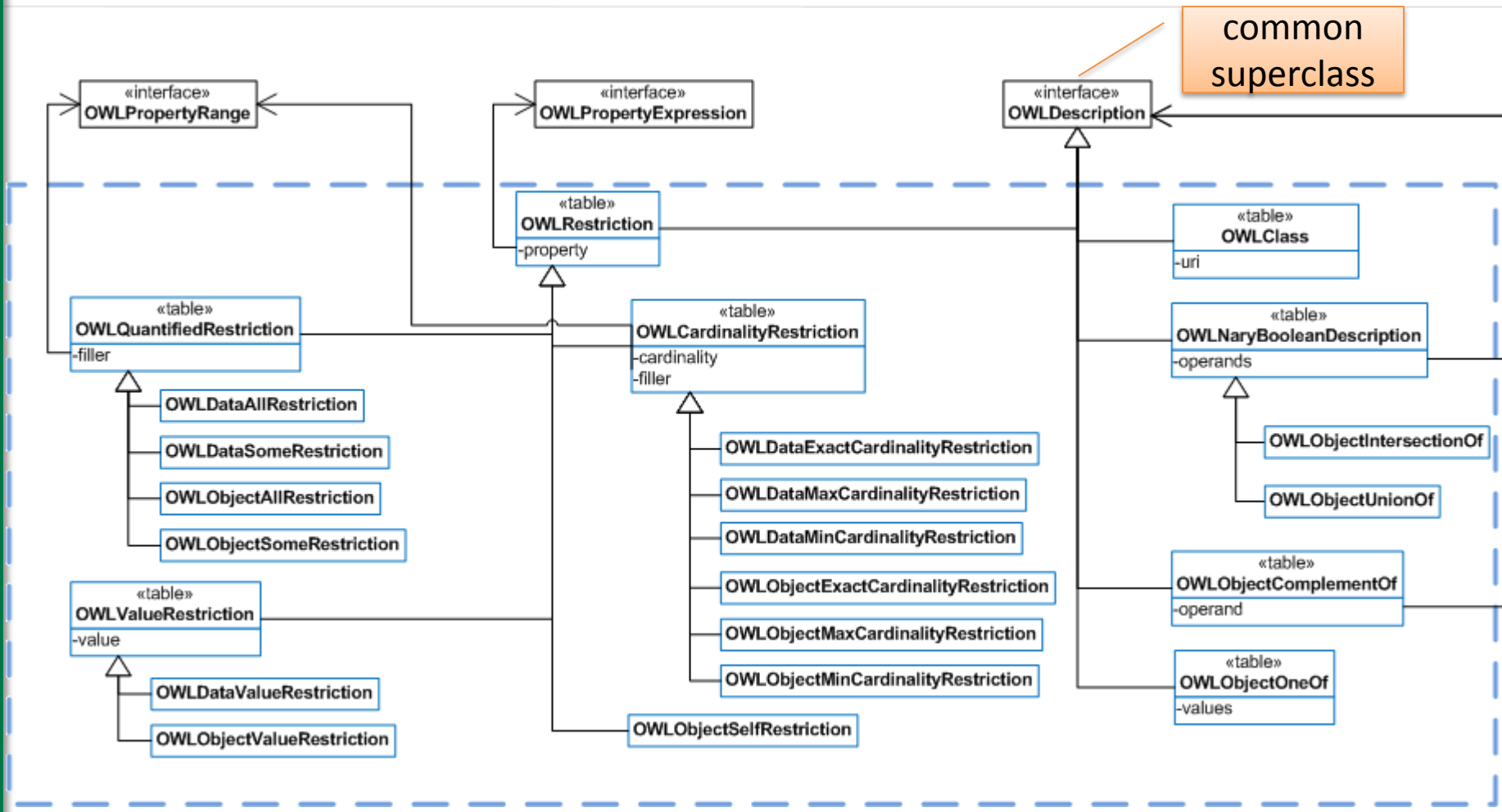
Mapping Paradigms

- Possible Strategies
  - One Java Class One Table
  - One Inheritance Tree One Table
  - One Inheritance Path One Table
  - Mixed forms
- Our Strategy
  - Mixed form
    - One class one table
    - One inheritance tree one table
  - Results in 56 tables

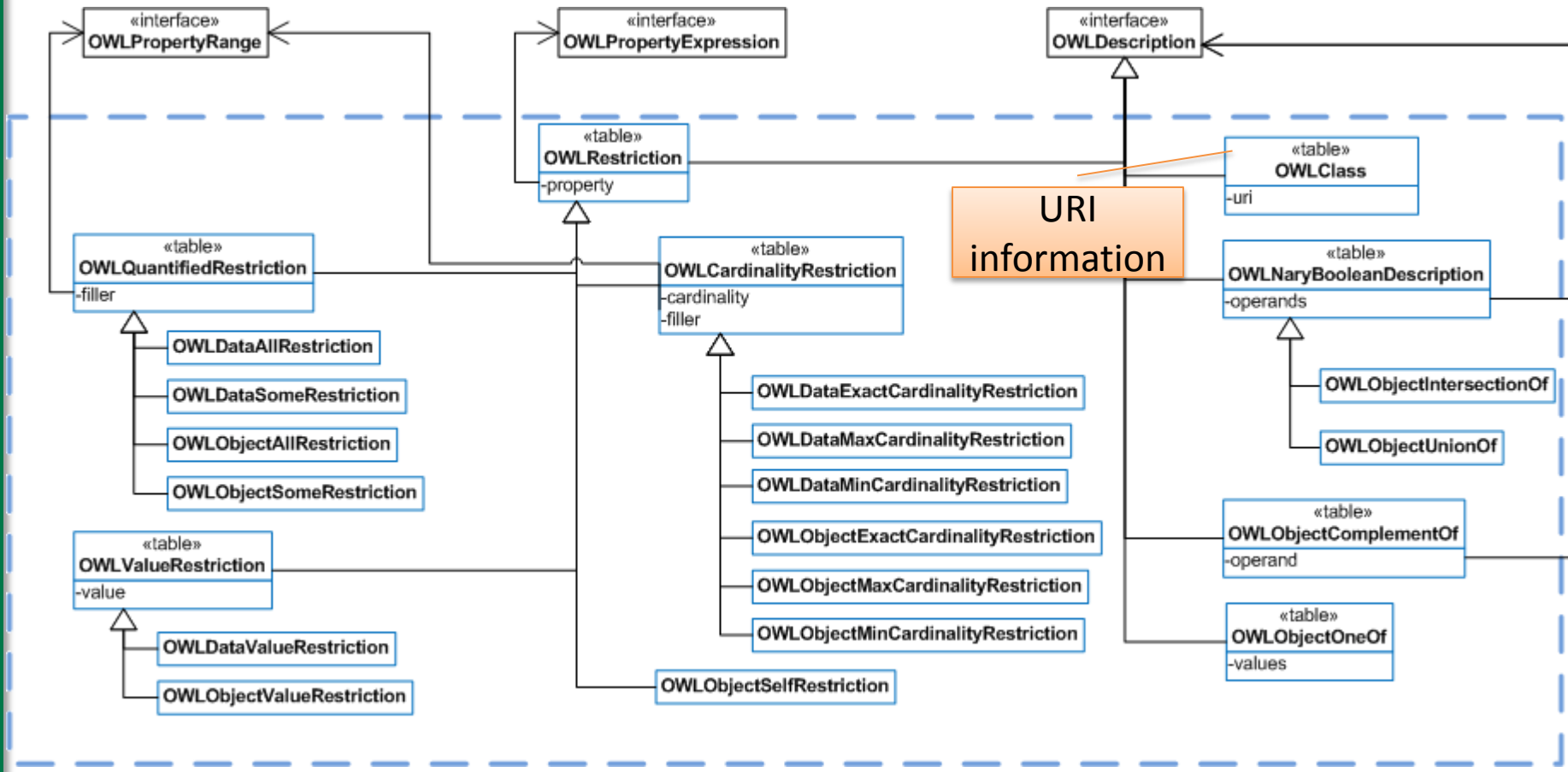
Relational Model



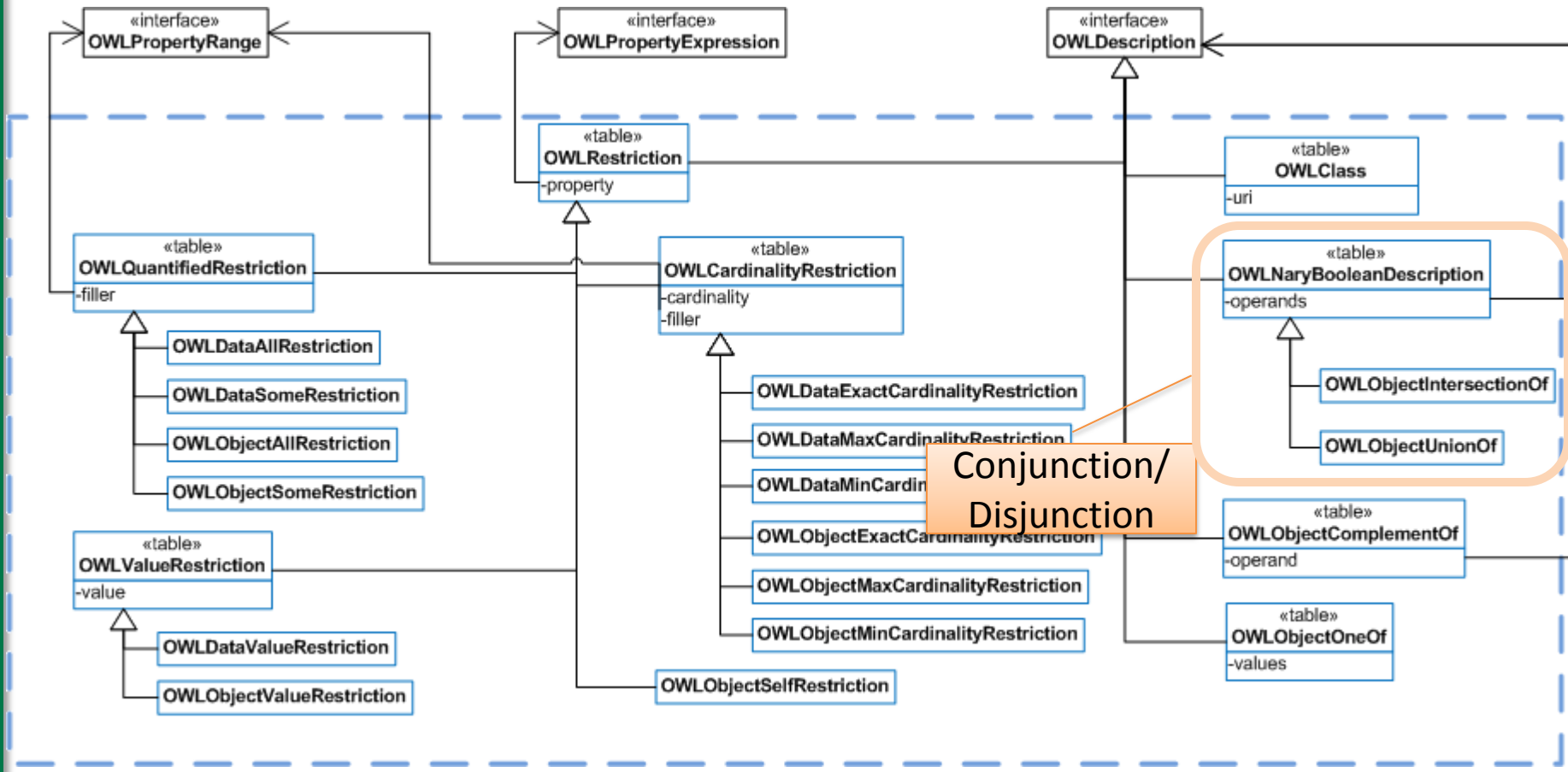
Complex Classes



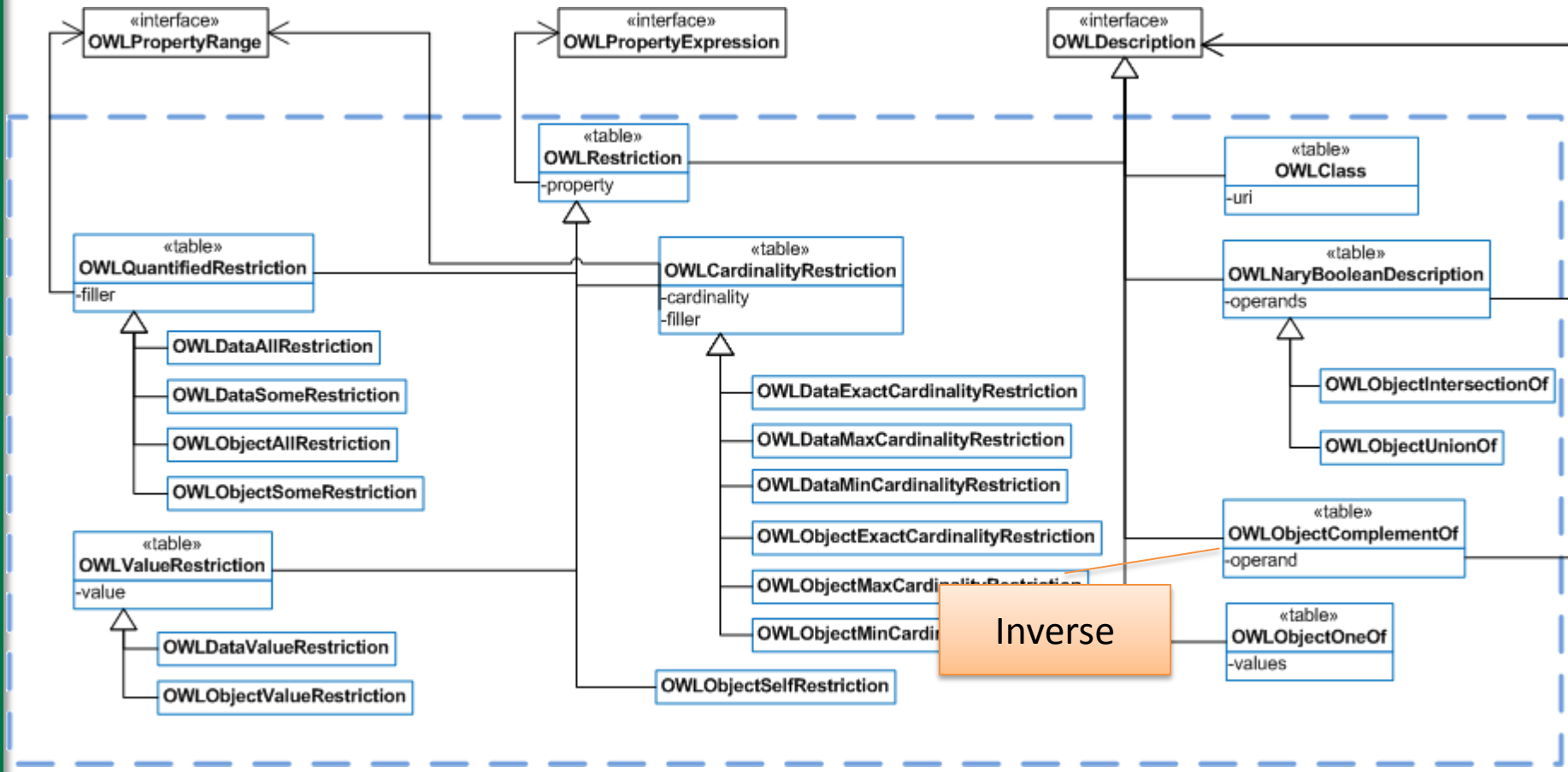
Complex Classes



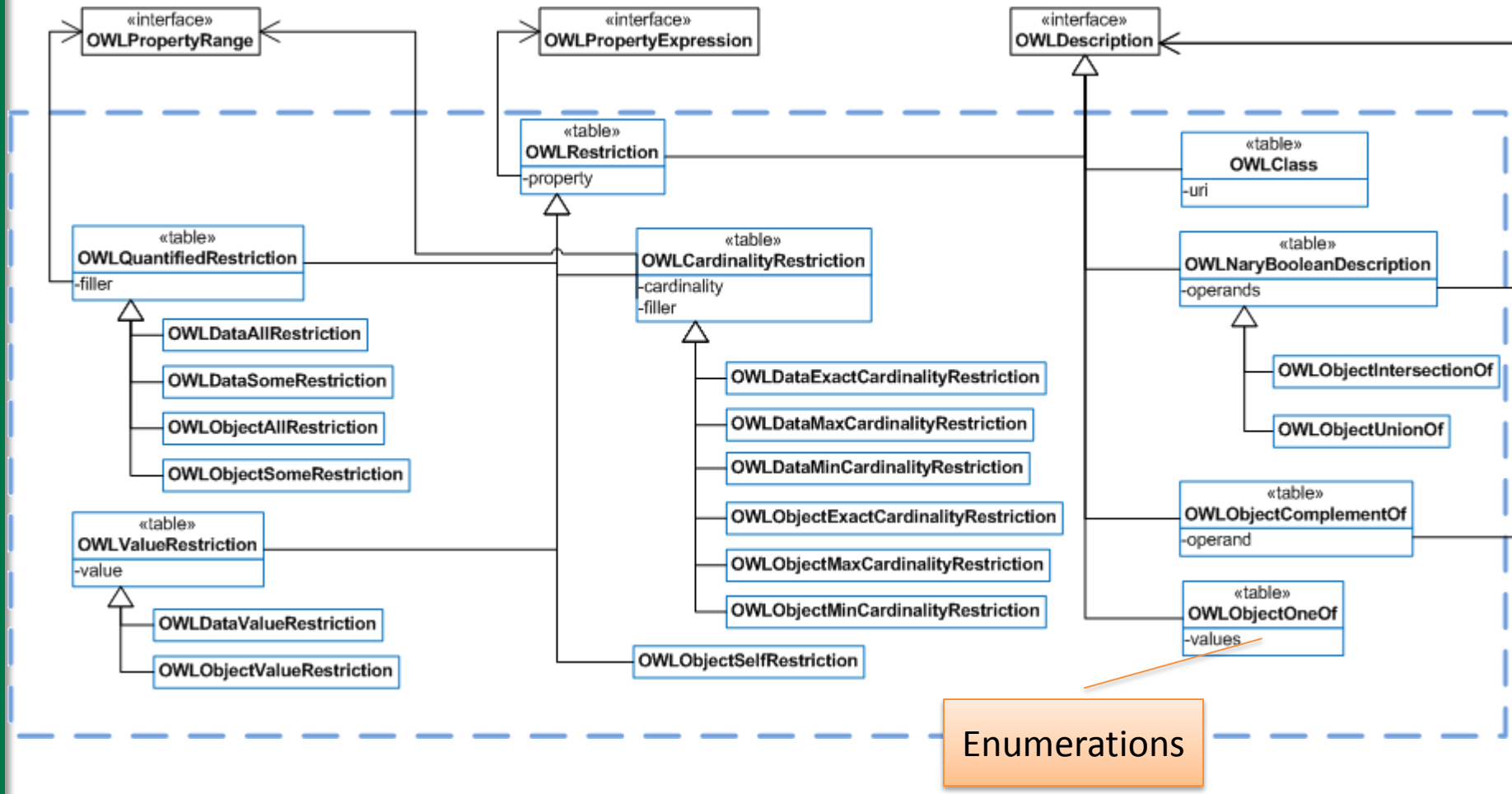
Complex Classes



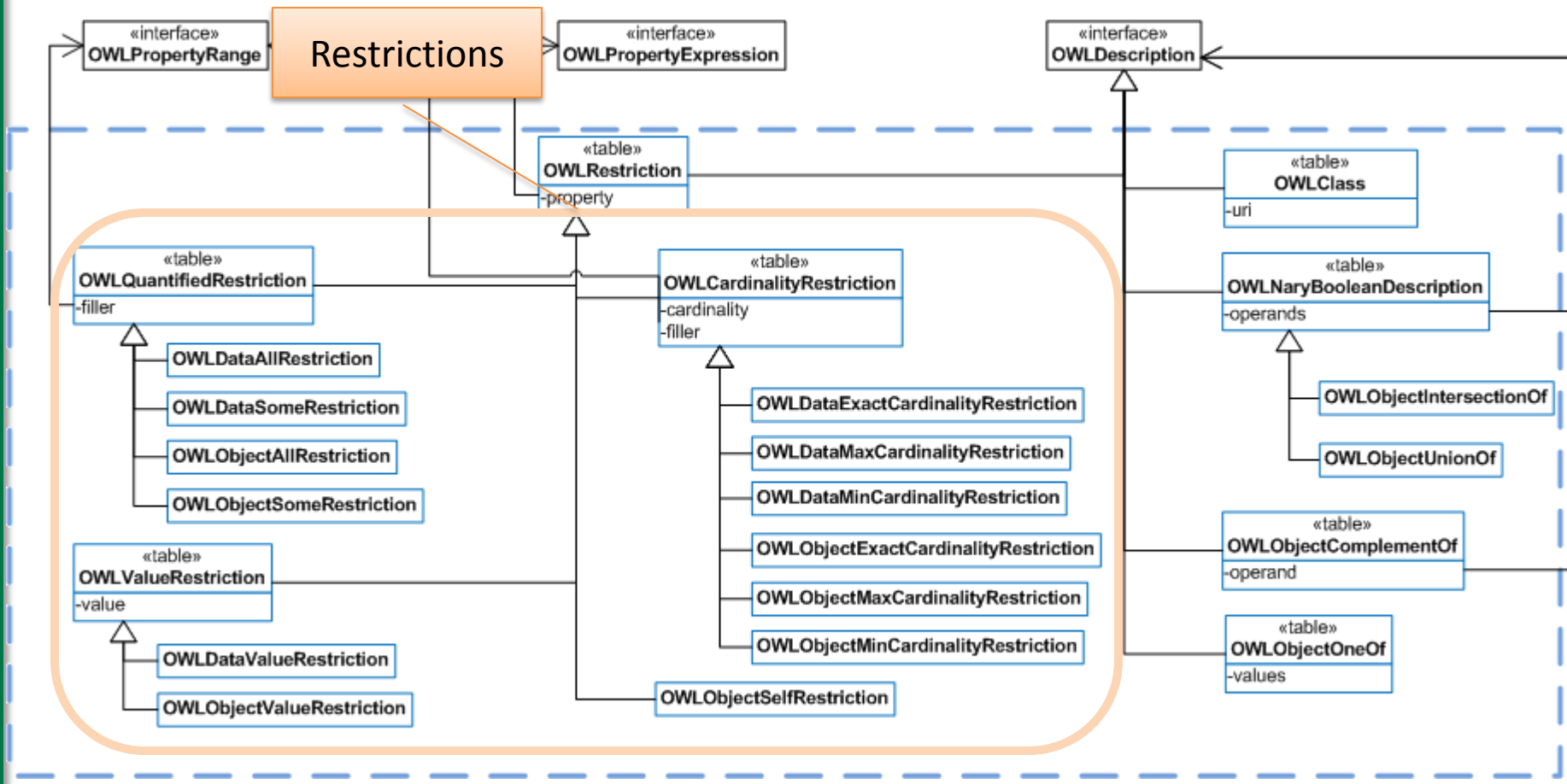
Complex Classes



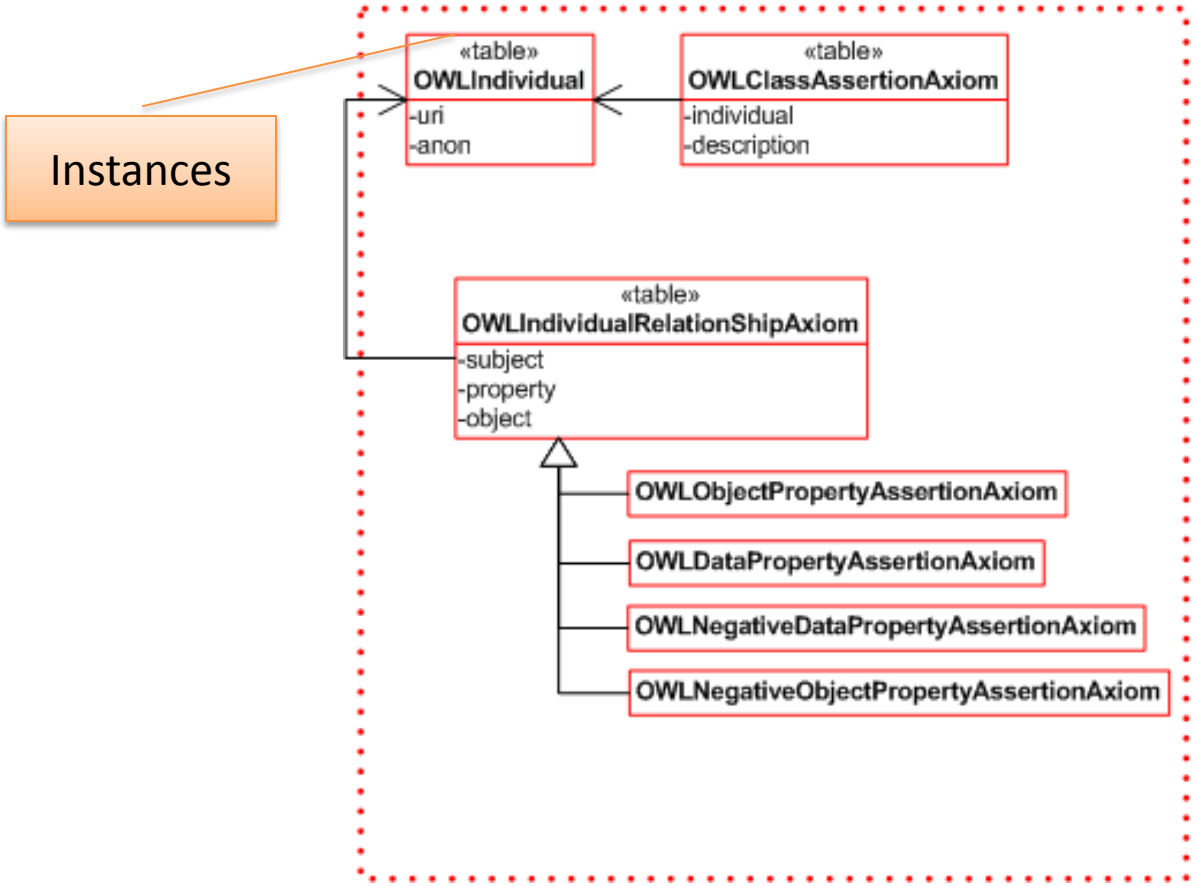
Complex Classes



Complex Classes

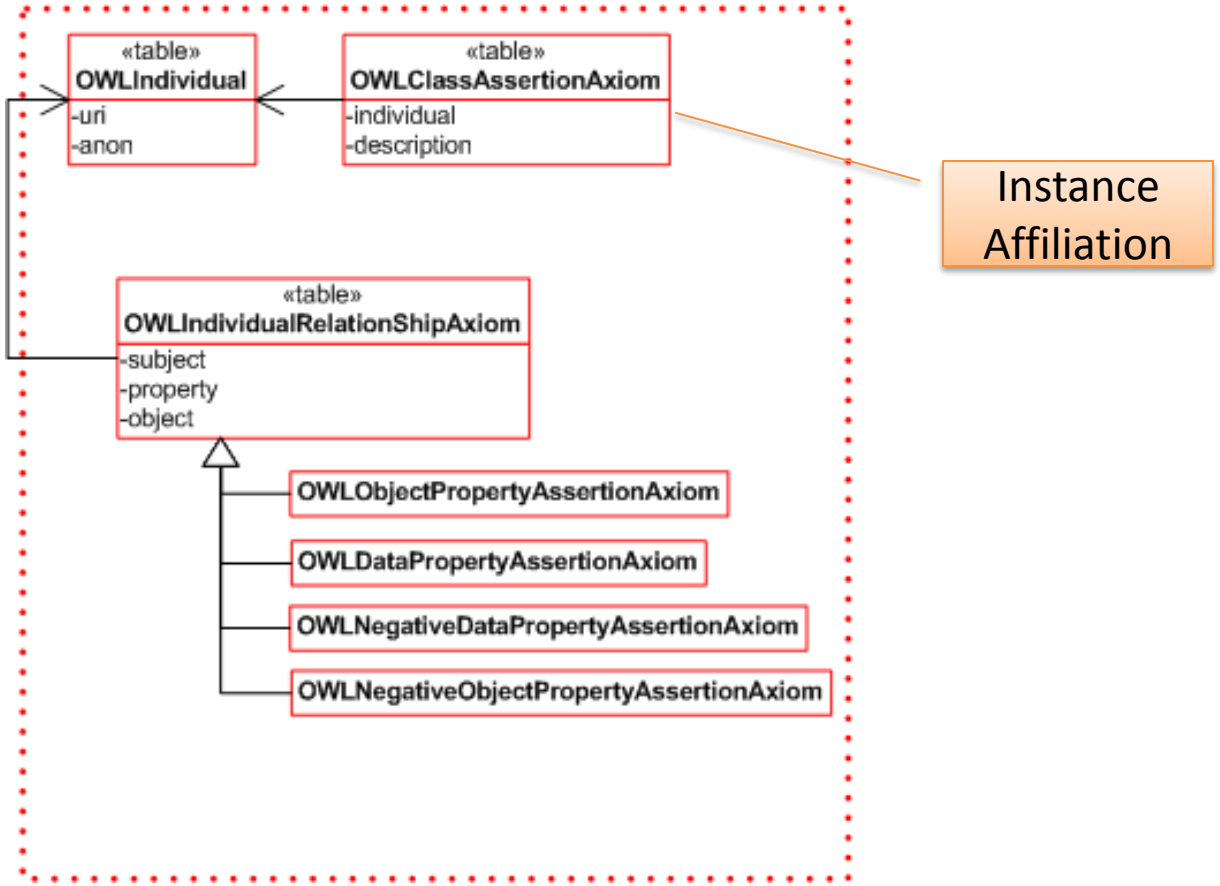


ABox


Instances
«table»
OWLIndividual
-uri
-anon
«table»
OWLClassAssertionAxiom
-individual
-description
«table»
OWLIndividualRelationShipAxiom
-subject
-property
-object
OWLObjectPropertyAssertionAxiom
OWLDataPropertyAssertionAxiom
OWLNegativeDataPropertyAssertionAxiom
OWLNegativeObjectPropertyAssertionAxiom

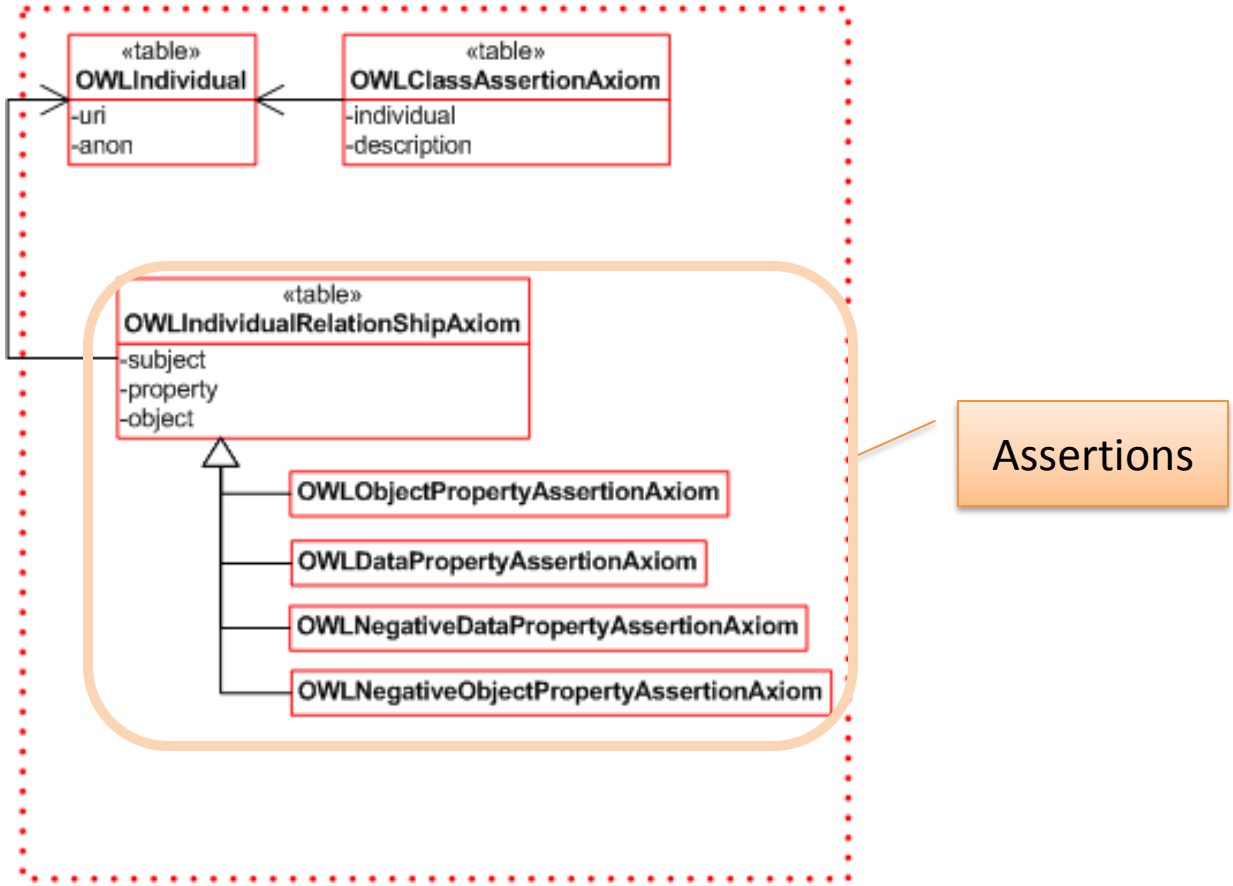
ABox

FZI


«table»
OWLIndividual
-uri
-anon
«table»
OWLClassAssertionAxiom
-individual
-description
Instance Affiliation
«table»
OWLIndividualRelationShipAxiom
-subject
-property
-object
OWLObjectPropertyAssertionAxiom
OWLDataPropertyAssertionAxiom
OWLNegativeDataPropertyAssertionAxiom
OWLNegativeObjectPropertyAssertionAxiom

ABox

FZI

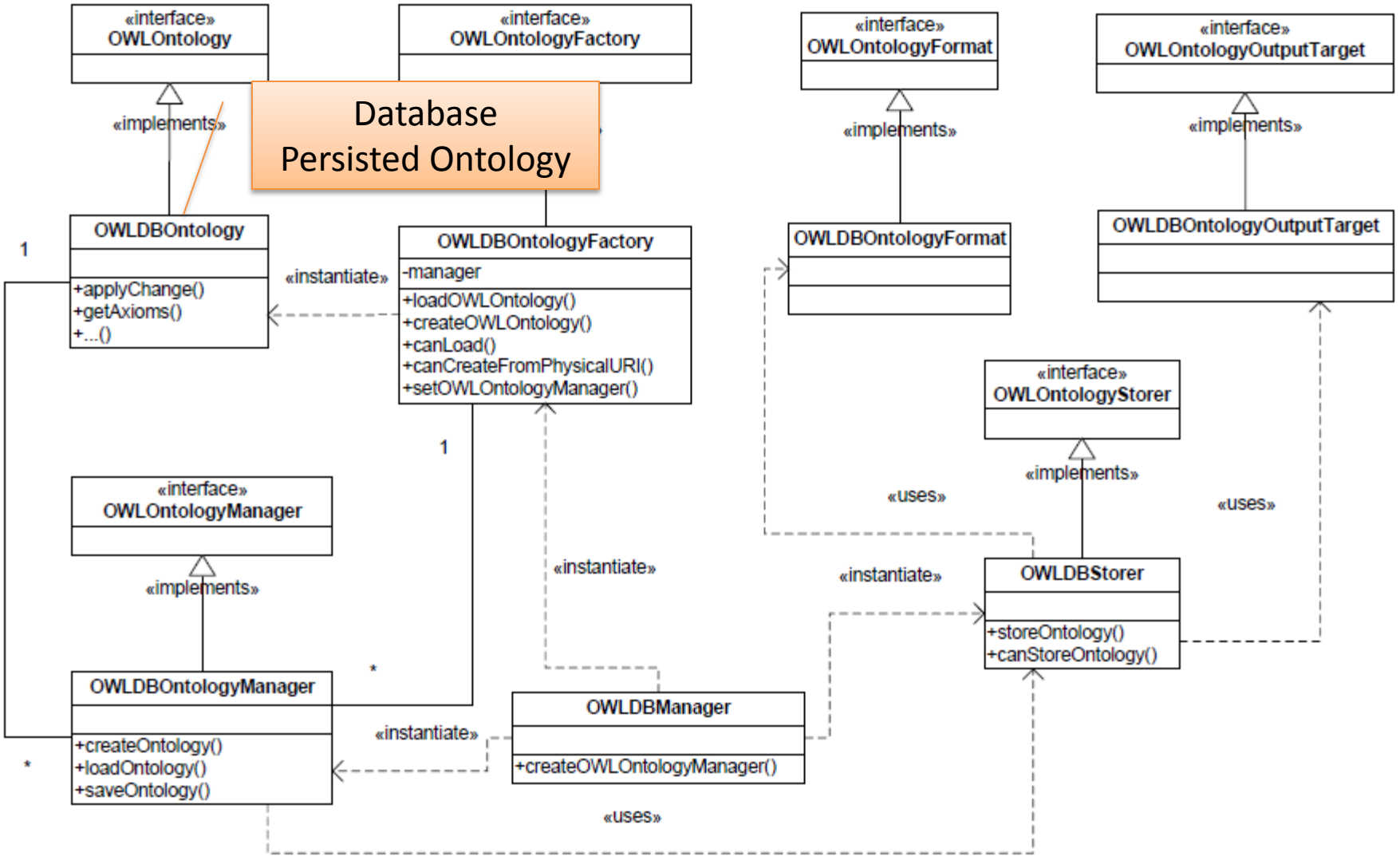

«table»
OWLIndividual
-uri
-anon
«table»
OWLClassAssertionAxiom
-individual
-description
«table»
OWLIndividualRelationShipAxiom
-subject
-property
-object
OWLObjectPropertyAssertionAxiom
OWLDataPropertyAssertionAxiom
OWLNegativeDataPropertyAssertionAxiom
OWLNegativeObjectPropertyAssertionAxiom
Assertions

Comparison to other systems

- Most systems focus on optimisation techniques for reasoning. In contrast we focus on direct manipulation.
- No in-Memory parsing necessary.
- Highly similar to other systems on schema level, *e.g.* SOR.
- Direct manipulation
  - Complete ontology is editable on database level.
- Instance information persistence is similar to triple stores
- Ensures all functionalities of OWL-API

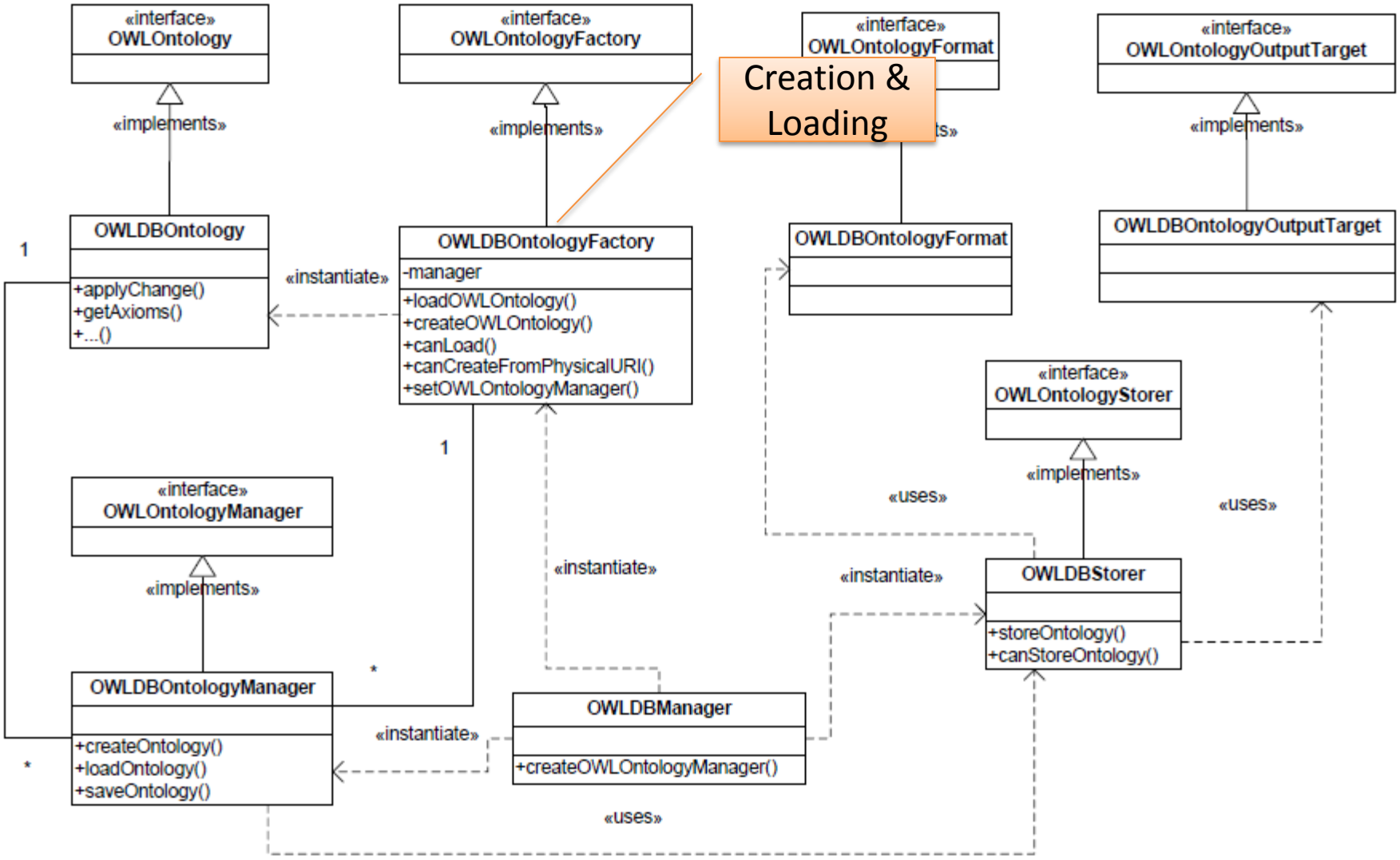
OWL-API

FZI


«interface»
OWLOntology
«interface»
OWLOntologyFactory
«interface»
OWLOntologyFormat
«interface»
OWLOntologyOutputTarget
«implements»
«implements»
«implements»
Database
Persisted Ontology
OWLDBOntology
+applyChange()
+getAxioms()
+...()
OWLDBOntologyFactory
-manager
+loadOWLOntology()
+createOWLOntology()
+canLoad()
+canCreateFromPhysicalURI()
+setOWLOntologyManager()
OWLDBOntologyFormat
OWLDBOntologyOutputTarget
«instantiate»
1
1
«interface»
OWLOntologyStorer
«implements»
«uses»
«uses»
«interface»
OWLOntologyManager
«implements»
«instantiate»
«instantiate»
OWLDBStorer
+storeOntology()
+canStoreOntology()
*
OWLDBOntologyManager
+createOntology()
+loadOntology()
+saveOntology()
OWLDBManager
+createOWLOntologyManager()
«instantiate»
*
«uses»

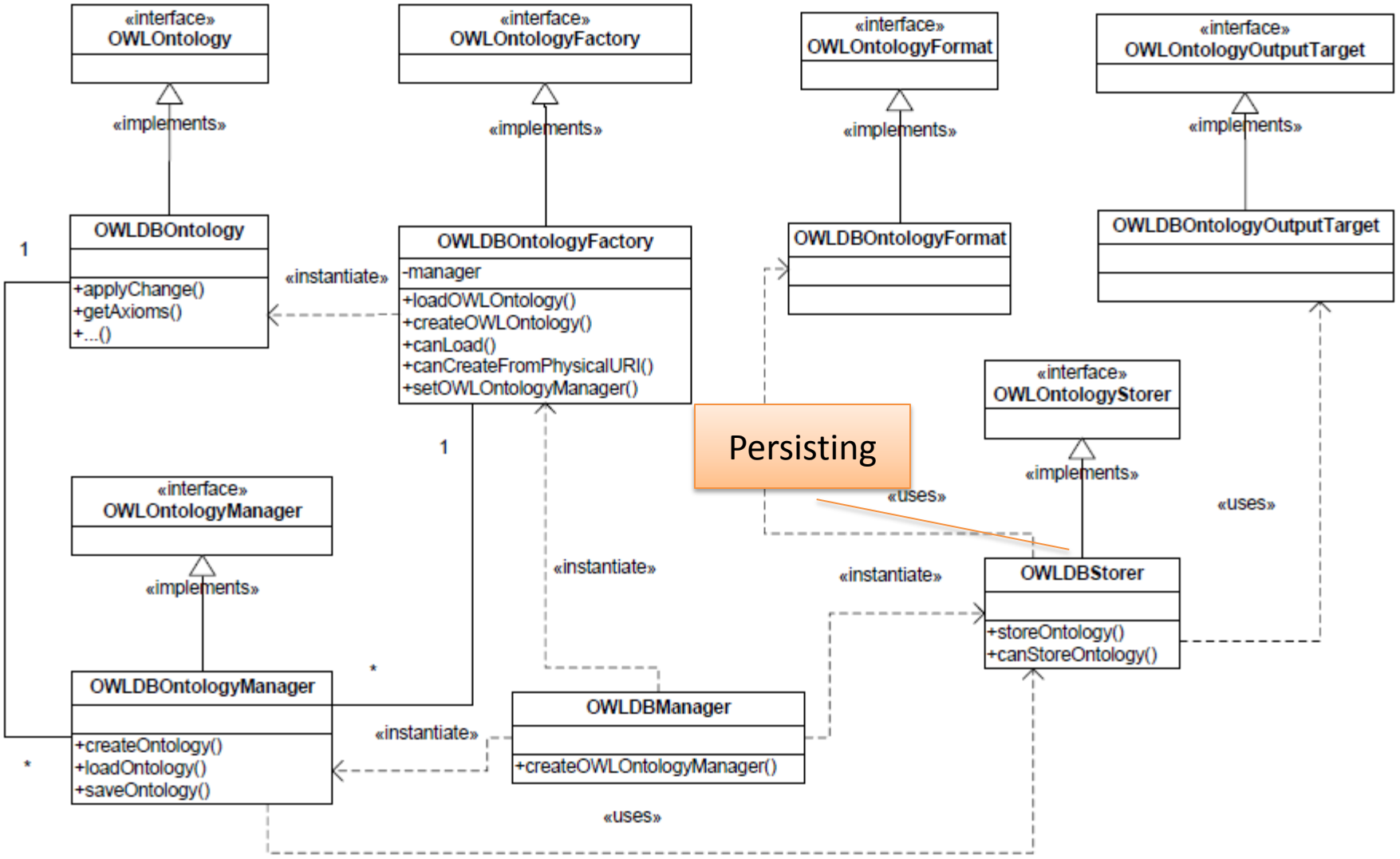
OWL-API

FZI


«interface»
OWLOntology
«implements»
OWLDBOntology
1
+applyChange()
+getAxioms()
+...()
«interface»
OWLOntologyFactory
«implements»
OWLDBOntologyFactory
-manager
+loadOWLOntology()
+createOWLOntology()
+canLoad()
+canCreateFromPhysicalURI()
+setOWLOntologyManager()
«instantiate»
1
Creation & Loading
«interface»
OWLOntologyFormat
OWLDBOntologyFormat
«interface»
OWLOntologyOutputTarget
«implements»
OWLDBOntologyOutputTarget
«interface»
OWLOntologyStorer
«implements»
«uses»
«uses»
«instantiate»
OWLDBStorer
+storeOntology()
+canStoreOntology()
«interface»
OWLOntologyManager
«implements»
OWLDBOntologyManager
+createOntology()
+loadOntology()
+saveOntology()
*
*
«instantiate»
OWLDBManager
+createOWLOntologyManager()
«instantiate»
«uses»

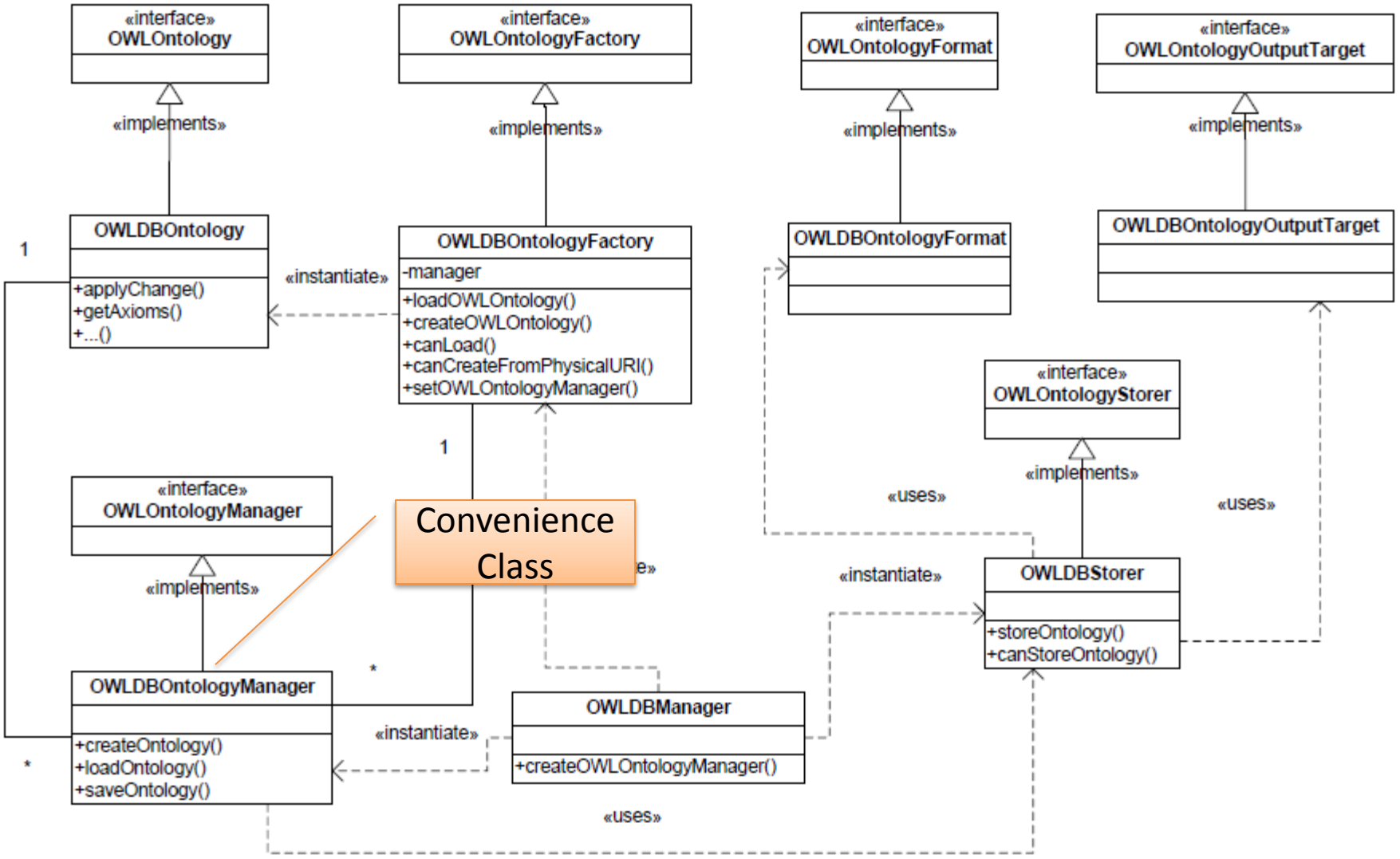
OWL-API

FZI

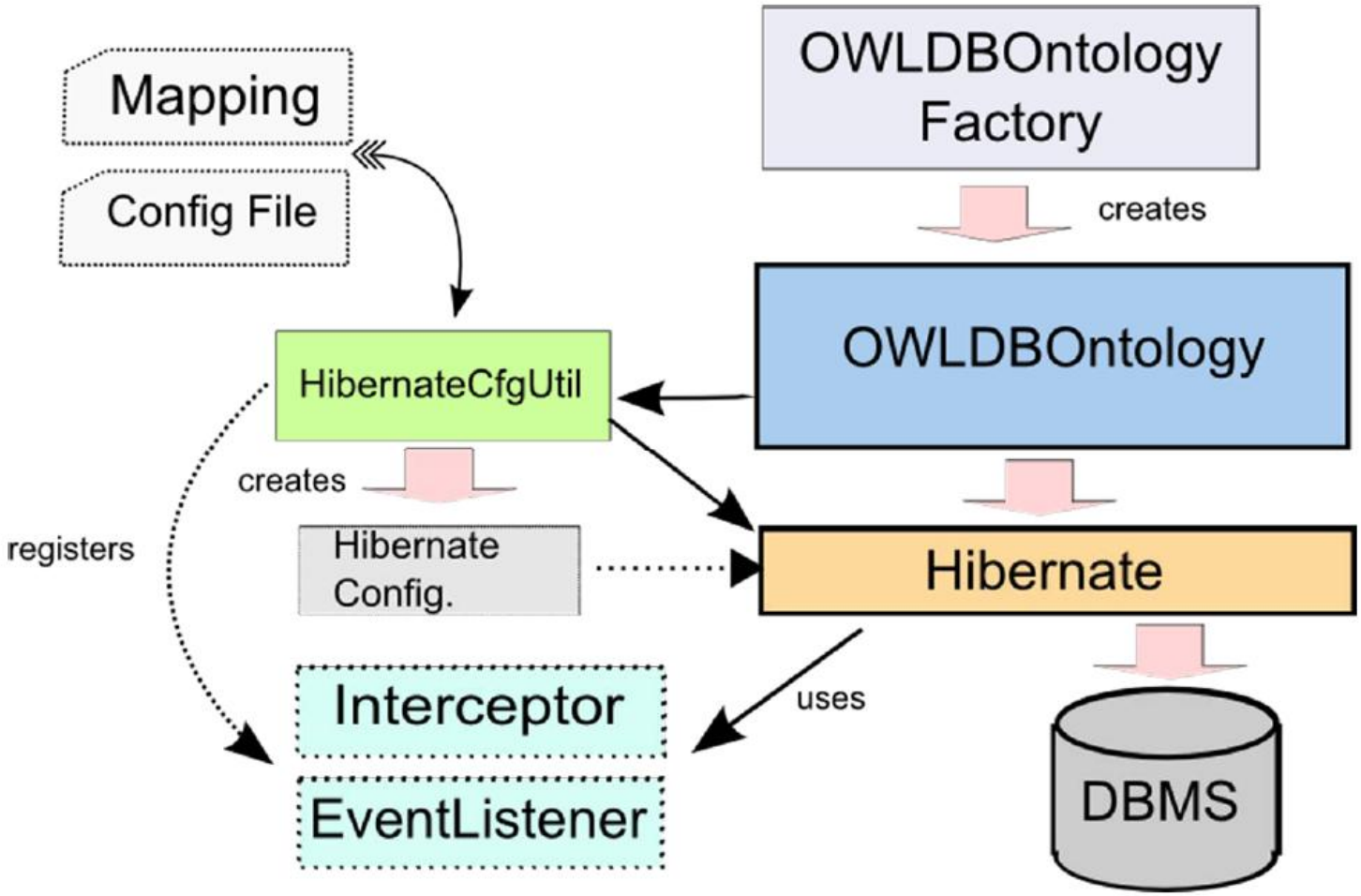

«interface»
OWLOntology
«interface»
OWLOntologyFactory
«interface»
OWLOntologyFormat
«interface»
OWLOntologyOutputTarget
«implements»
«implements»
«implements»
«implements»
OWLDBOntology
+applyChange()
+getAxioms()
+...()
1
«instantiate»
OWLDBOntologyFactory
-manager
+loadOWLOntology()
+createOWLOntology()
+canLoad()
+canCreateFromPhysicalURI()
+setOWLOntologyManager()
OWLDBOntologyFormat
OWLDBOntologyOutputTarget
1
Persisting
«interface»
OWLOntologyStorer
«implements»
«uses»
«uses»
«interface»
OWLOntologyManager
«implements»
«instantiate»
«instantiate»
OWLDBStorer
+storeOntology()
+canStoreOntology()
*
OWLDBOntologyManager
+createOntology()
+loadOntology()
+saveOntology()
*
«instantiate»
OWLDBManager
+createOWLOntologyManager()
«uses»

OWL-API

FZI


«interface»
OWLOntology
«interface»
OWLOntologyFactory
«interface»
OWLOntologyFormat
«interface»
OWLOntologyOutputTarget
«implements»
«implements»
«implements»
«implements»
OWLDBOntology
+applyChange()
+getAxioms()
+...()
1
«instantiate»
OWLDBOntologyFactory
-manager
+loadOWLOntology()
+createOWLOntology()
+canLoad()
+canCreateFromPhysicalURI()
+setOWLOntologyManager()
OWLDBOntologyFormat
OWLDBOntologyOutputTarget
«interface»
OWLOntologyStorer
«implements»
1
«interface»
OWLOntologyManager
«implements»
Convenience Class
«uses»
«uses»
«instantiate»
OWLDBStorer
+storeOntology()
+canStoreOntology()
*
OWLDBOntologyManager
+createOntology()
+loadOntology()
+saveOntology()
*
«instantiate»
OWLDBManager
+createOWLOntologyManager()
«uses»

Architecture

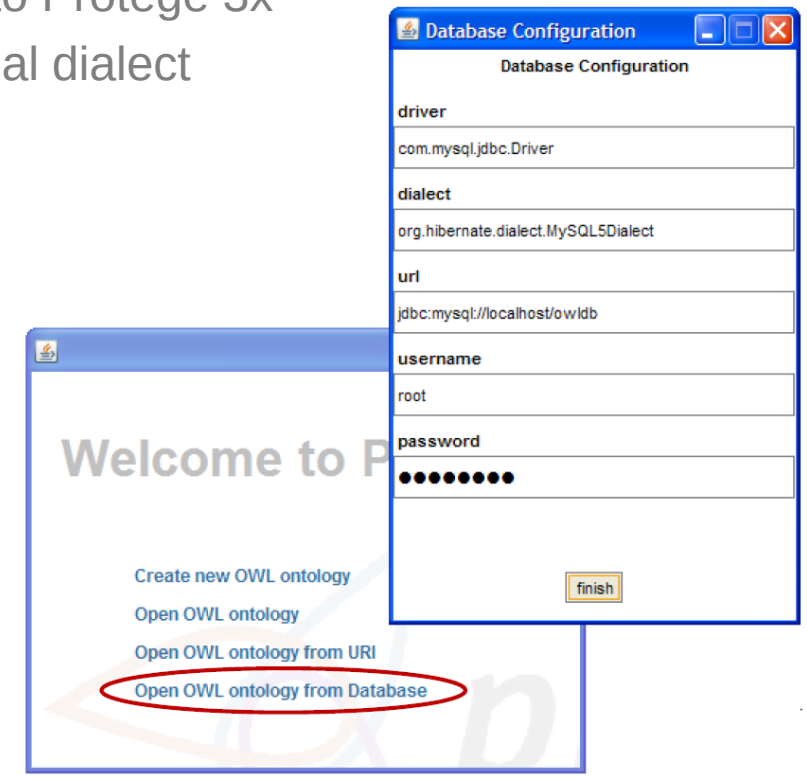


Benefits

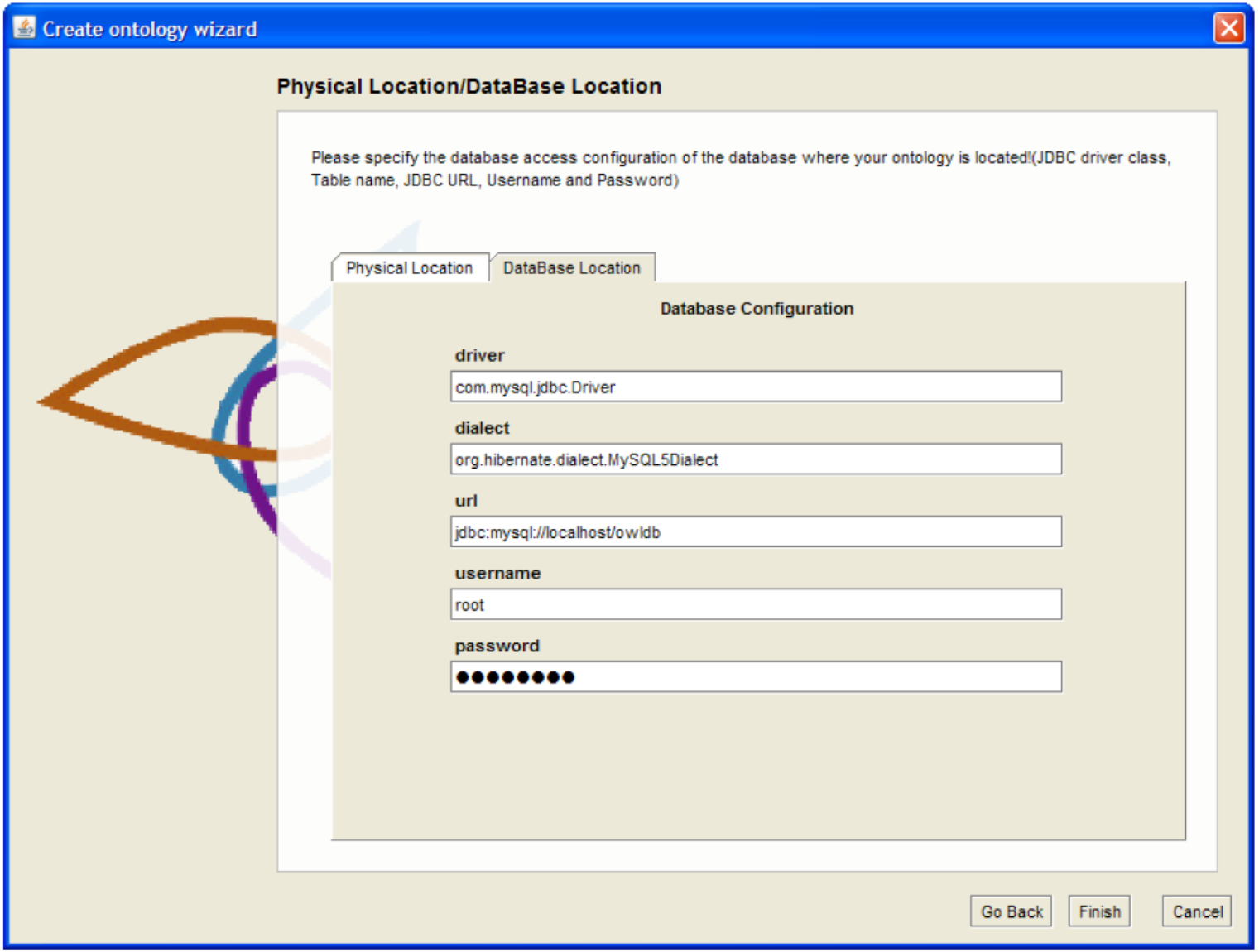
- Minimisation of necessary joins compared to triple stores
  - Better retrieval
- Management facilities of RDBMS
  - Query optimisation
  - Transactions
  - Caching, etc.
- OWL 2 compatible
  - Mapping approach also usable for another API
- Modularisation via owl:import
  - Several ontologies possible
- Seamless integration into the OWL-API
  - Non-invasive

Seamless Protégé Integration

- Open Dialog
  - Similar to Protégé 3x
  - Additional dialect



Protégé Integration



Conclusion

- No change in interaction regarding the in-Memory implementation of Protégé (as well as in interaction with the OWL-API)
- No changes on the OWL-API object implementations (non-invasive)
- Project files desirable
- Still Prototype
- Download address
  - http://www.fzi.de/downloads/ipe/owldb.zip

- Part of the German Theseus Research Project

Questions?