



ORDI

Ontology Representation and Data Integration Framework

Ontotext Lab, Sirma Group

Jul, 2006

ORDI Introduction

- **Ontology Representation and Data Integration (ORDI)**
 - Nature: **ontology middleware in Java**
 - Licence: open-source, under **LGPL**
 - **D2.3 of DIP** implementing the conceptual framework of D2.2
 - Ontology Management and Working Group (**OMWG**)
- **Objectives:**
 - Support for **interoperability** across of heterogeneous repositories and reasoners
 - **Integration of databases** and other structured data-sources
 - **RDF <> WSML** interoperability is major objective
 - Conversion of WSML to and from WSML-Triples (WSML-RDF)
- <http://www.ontotext.com/ordi/>
 - For download and documentation
 - The distribution includes sample applications

Related Syntaxes

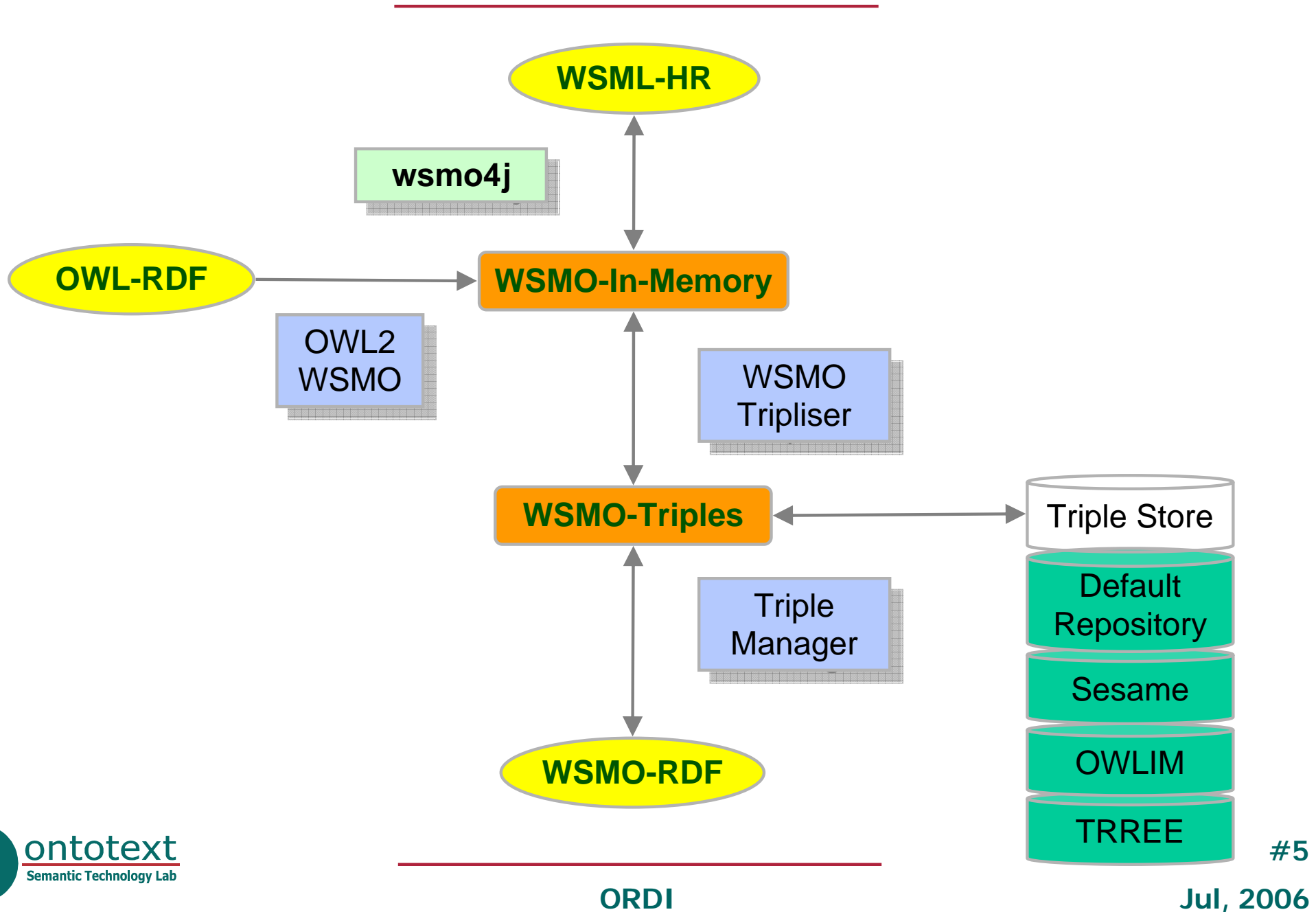
- A WSML document in either:
 - **WSML XML** syntax or
 - **WSML human readable** syntax,
- **OWL in RDF**: the standard RDF XML syntax. RDF syntaxes different than XML (e.g. N-Triples and N3) will also be supported (through the existing RDF parsers).
 - RDFS subset which is a proper sub-language of OWL DLP is considered.
- **WSMO-RDF**: a WSMO/L document serialized according to:
 - **WSMO RDF Schema**: an ORDI-specific RDFS/OWL ontology (meta-schema) derived from the WSML mapping to OWL (**D32@WSML**). It will be similar to the RDFS schema for OWL and dependent on it.

Representations

We distinguish two internal representations of the ontology data:

- **WSMO-In-memory:** WSMO-API compliant model (e.g. WSMO4J). This representation is not specific for ORDI;
- **WSMO-Triples:** representation of WSMO elements as RDF triples according to the WSMO RDF Schema. This is an internal representation allowing us to store WSMO (and other data aligned to its model) into a RDF triple repository for efficient query and management. The WSMO- RDF syntax is a serialization of this representation

wsmo4j & ORDI representations

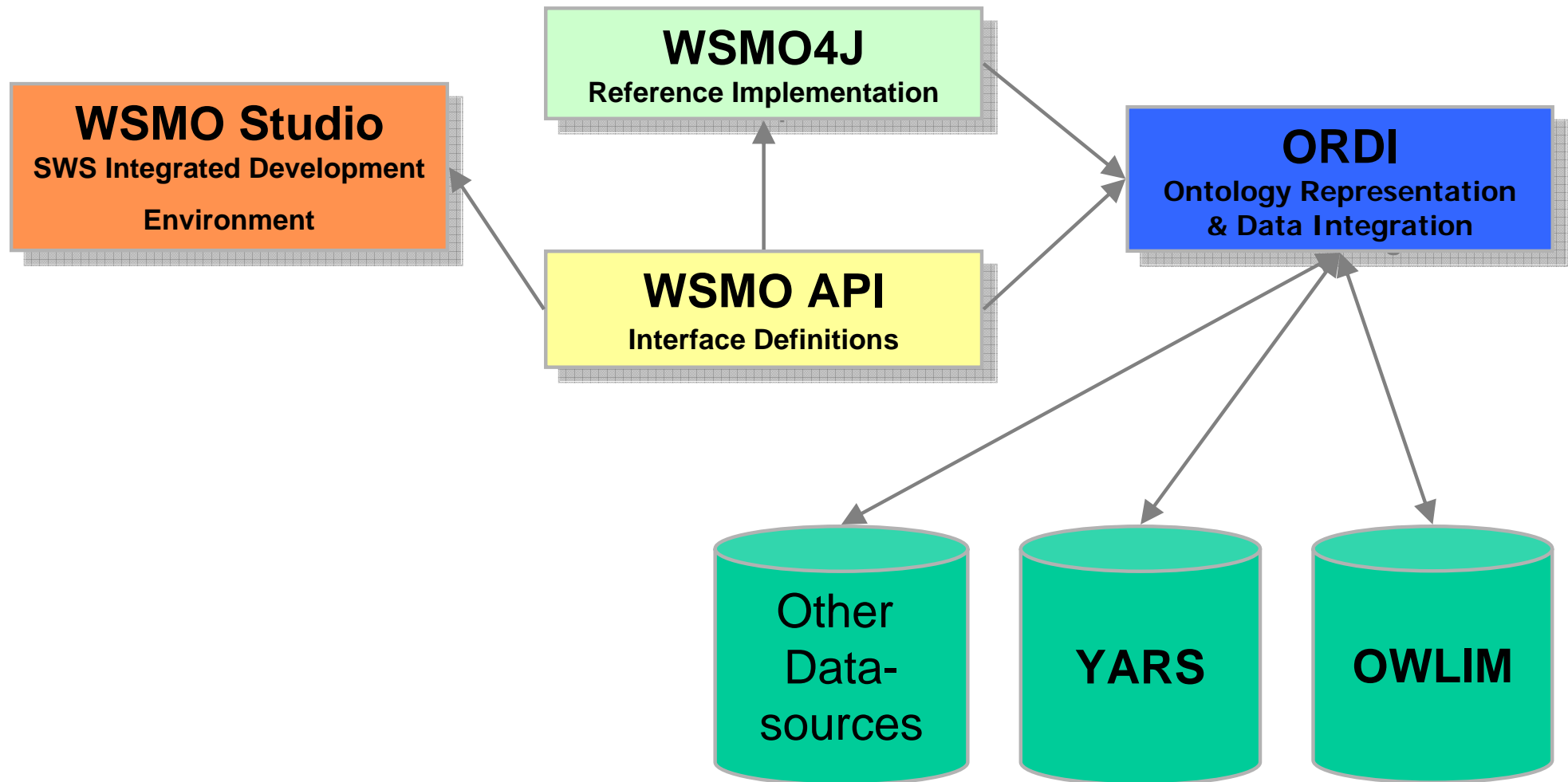


ORDI and wsmo4j

ORDI and wsmo4j were designed to **complement each other**:

- wsmo4j includes a WSMO **representation and management API**, coupled with a reference implementation
- **ORDI is an extension of wsmo4j**
 - while wsmo4j is self-sufficient
- The major dependency between the two is the **org.omwg.ontology** package, which defines the ontology primitives of WSMO.
- Starting with version 0.3 the **OWL Import** is not part of ORDI; it has been moved to the wsmo4j code-base
 - It allows import of OWL through parsing of the most popular RDF/XML syntax and transformation directly into WSMO-In-Memory format.

Ontology & SWS Management Architecture



Functionality and Modules

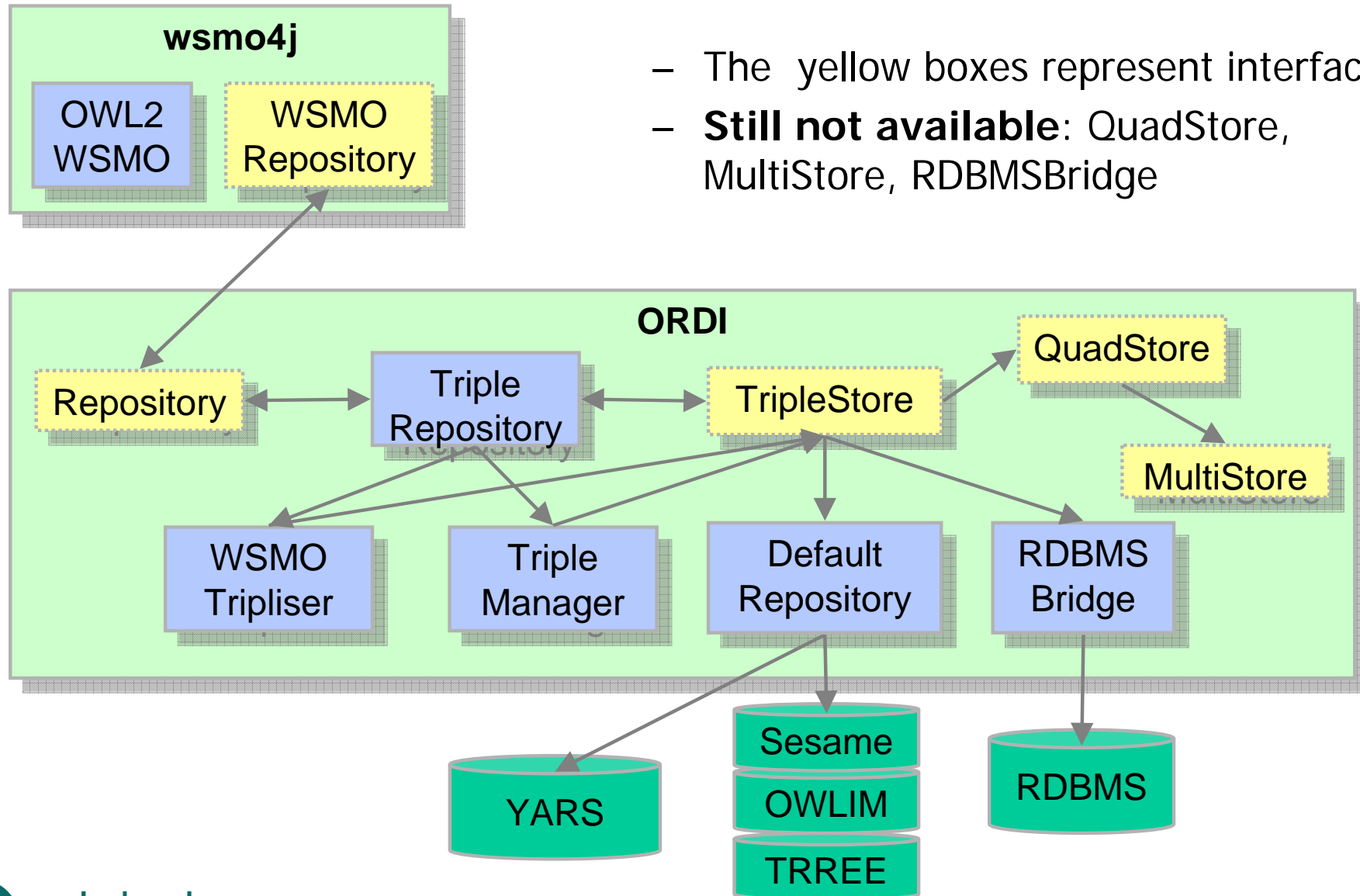
ORDI, as a package, contains the following modules:

- **ORDI API** - all the APIs necessary to work with ORDI, at present it is a tiny extension of the WSMO API.
- **ORDI Implementation** - an implementation of the interfaces with the following major parts:
 - A default repository implementation, based on Sesame. It uses the WSMO Tripliser;
 - An RDF/XML Parser implementation for WSMO-RDF;

ORDI Facts

- **ORDI and wsmo4j** were designed to complement each other:
 - **ORDI is an extension of wsmo4j**
 - while wsmo4j is self-sufficient
 - ORDI extends several WSMO API interfaces, most notably WSMORepository
- ORDI is a **high-performance scalable WSMO Repository** (amongst its other functionality)
 - It stores and manages an RDF-representation of WSML
 - This includes all WSMO elements (e.g. Goals and WS)
 - Using Sesame, OWLIM and TRREE
 - The latest version of ORDI (v0.4) provides SPARQL support

ORDI Architecture



- The yellow boxes represent interfaces
- **Still not available:** QuadStore, MultiStore, RDBMSBridge

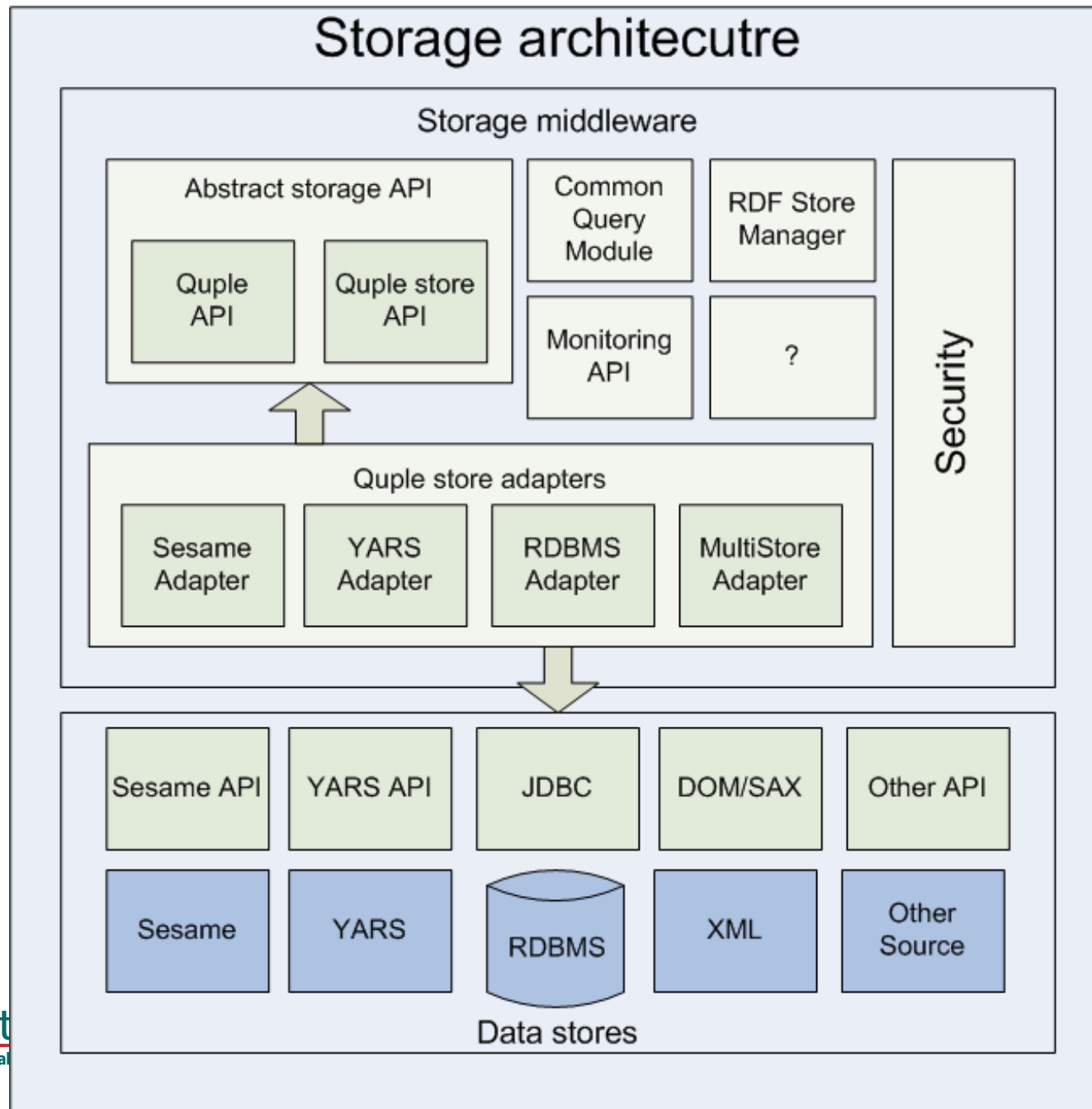
ORDI Facts

- **ORDI and wsmo4j** were designed to complement each other:
 - **ORDI is an extension of wsmo4j**
 - while wsmo4j is self-sufficient
 - ORDI extends several WSMO API interfaces, most notably WSMORepository
- ORDI is a **high-performance scalable WSMO Repository** (amongst its other functionality)
 - It stores and manages an RDF-representation of WSML
 - This includes all WSMO elements (e.g. Goals and WS)
 - The latest version of ORDI (v0.4) provides **SPARQL support**
 - Using Sesame, OWLIM and TRREE
 - It can scale to **billions of triples**
 - Upload and inference speed: 4,000-150,000 statm./sec.

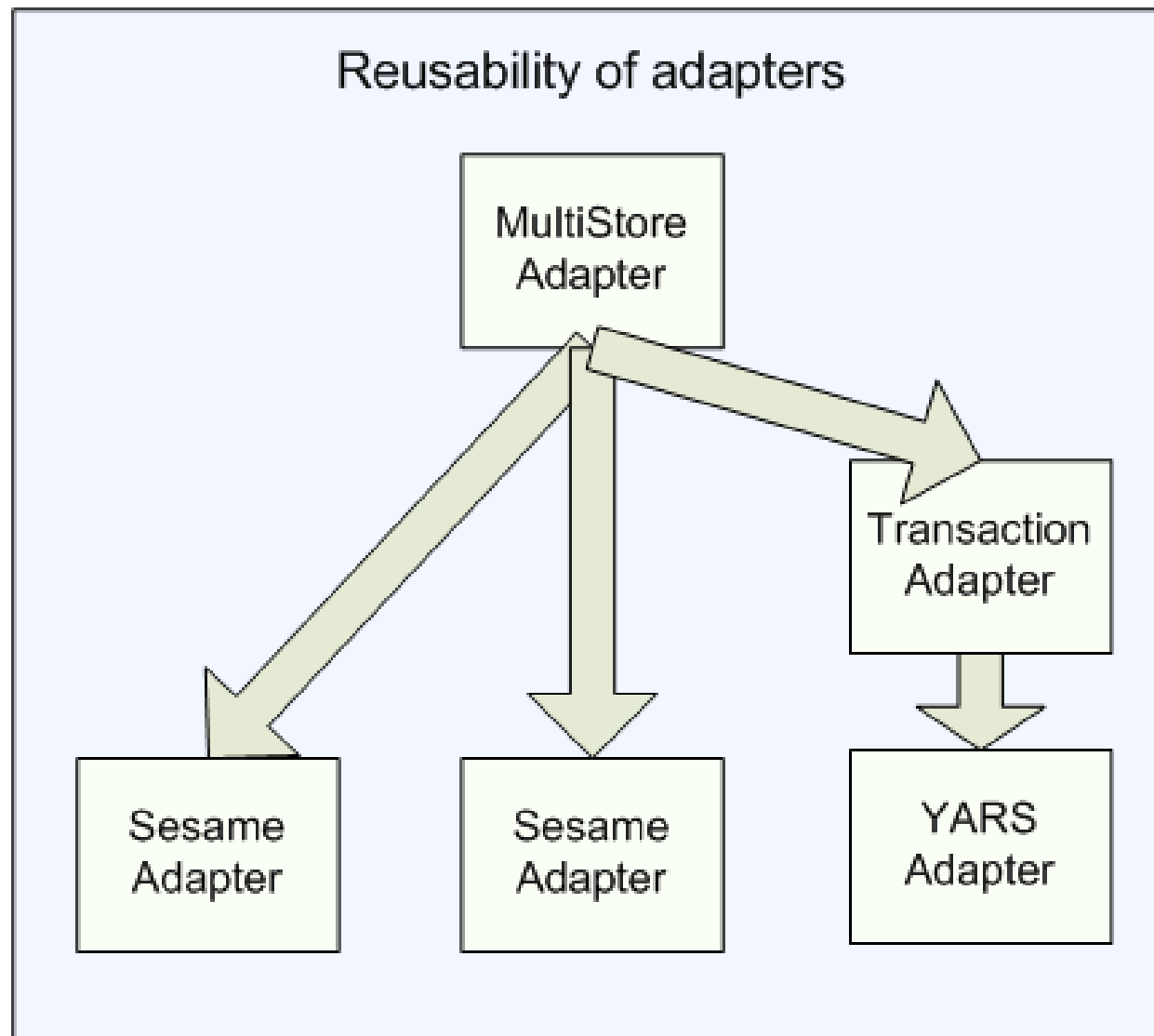
ORDI Architecture Development Plans

- Java middleware (as at present), consisting of
 - APIs allowing plug-ins/customization
 - Management components and facilitators
 - Default/references implementations
- Abstract Storage API
 - It is a logical view – its various aspects may or may not be supported in the different “physical” implementations
 - Quple = triple + reification + context support
 - Quple API: OO model interfaces
 - Quple Store API: storage isolation
 - Simple interfaces for quple storage/retrieval
- Quple Store Adapters
 - Can be stacked on top of each other

ORDI Architecture Overview



ORDI Architecture Plans - Adapters



ORDI Architecture Plans (II)

- Querying
 - Unified query support - generic SPARQL support (on top of an arbitrary QupleStore implementation)
 - Custom query support, including preparation/optimization
 - Arbitrary Query Language support
- Batch-copy
 - Support for efficient copy, to be used for replication/import
- Multi-Store
 - Allows for consolidation of several datasources, each of which wrapped as QupleStore
 - Universal but not always efficient integration approach

Related tools: TRREE Engine

- TRREE is a **Triple Reasoning and Rule Entailment Engine**
- It supports efficient RDF management:
 - Representation, indexing and storage of RDF graphs
 - Retrieval wrt to triple patterns
 - Forward-chaining of entailment rules
 - Materialization of and invalidation of implicit triples
- The supported **semantics can be customized** through different sets of rules:
 - The complexity which can be efficiently supported this way is in the range of OWL DLP/Horst; it is above WSMML Core
- TRREE is in the hart of OWLIM

Related tools: OWLIM

- OWLIM is a high-performance **OWL semantic repository**,
 - It is packaged as a Storage and Inference Layer (SAIL) for the **Sesame RDF database**
- OWLIM uses the **TRREE engine**
 - The reasoning and query evaluation are performed in-memory,
 - A reliable persistence strategy assures data preservation, consistency and integrity
- OWLIM provides full support for **RDF(S)** and **OWL DLP**
 - It supports **OWL Horst**, which is more expressive than OWL DLP and 100% compliant with RDFS
- OWLIM is proven to scale up to **tens of millions of statements**, maintaining upload speed of tens of thousands of statement/sec.
 - It can manage millions of statements even on commodity desktop hardware.

ORDI

Developed within the
Ontology Management Working Group
<http://www.ontotext.com/ordi/>