

Introducing PelletDb

Expressive, Scalable Semantic Reasoning
for the Enterprise

<http://clarkparsia.com/whitepapers/>

CLARK  PARSIA

We solve the world's most complex information problems.

Copyright ©2009, Clark & Parsia LLC. All Rights Reserved.

This document is available under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States License.

<http://creativecommons.org/licenses/by-nc-nd/3.0/us/legalcode>.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose.

PelletDb is a registered trademark of Clark & Parsia LLC. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Produced with Xe₂La₂X; body set in Milo Serif 10/15. Cover design and C&P product logos by [Kate Krizan Design](#).

EXECUTIVE SUMMARY — This white paper describes PelletDb, the first and only owl 2 reasoner for Oracle Database 11g, which provides scalable and correct owl reasoning. Built on [Pellet](#), PelletDb provides an integrated solution for owl 2 reasoning in the enterprise. PelletDb is especially optimized for Oracle Database 11g, enhancing both the native reasoning capabilities of Oracle Database Semantic Technologies and the scalability of Pellet. PelletDb significantly improves performance compared to other systems and offers convenient access to Pellet’s advanced reasoning features, including inference explanation. This white paper describes PelletDb’s benefits; gives a usage overview; and offers a roadmap for future development.

1 PELLETDB BENEFITS

PelletDb®, an integrated system combining Clark & Parsia’s owl reasoner Pellet and Oracle® Database Semantic Technologies, provides the only scalable owl 2 reasoning system for enterprise semantic applications.¹ PelletDb and Oracle customers benefit from more complete insights into their semantic data by using PelletDb’s expressive, sound-and-complete owl 2 reasoning as a plug-in to Oracle Database 11g. Developers are freed from in-memory constraints on inference and are able to create unlimited result sets, while also benefiting from the persistence, security, and other enterprise features provided by Oracle Database 11g.

Scalable and efficient owl reasoning is the main benefit of PelletDb. It makes *expressive, sound-and-complete owl 2 reasoning realistic for most cases*, especially when schema reasoning is required. If your schema and instance data can be accommodated in main memory, then PelletDb can read both from Oracle Database 11g into memory via SPARQL and SQL. If your instance data is too large for main memory, sound-and-complete schema reasoning can be performed by PelletDb and persisted in Oracle Database 11g. Then the native inference engine in Oracle Database Semantic Technologies can perform instance reasoning using its instance data with the schema inferences computed by PelletDb.

¹ For more information about owl 2, see [the W3C OWL Working Group](#).

Pellet's OWL capabilities are useful for integration and analysis problems where schema-level reasoning is complex, but overall data sizes are not extremely large. The complete reasoning results provided by PelletDb enhance the capabilities of decision support systems and analytical tools that use OWL ontologies to represent and reason about domain knowledge. PelletDb's optimized interface with Oracle Database 11g provides scalable and efficient reasoning, performance improvements up to an order of magnitude,² and allows customized inferencing for faster storage of inferences to Oracle Database 11g.

1.1 INTEGRATION DETAILS

PelletDb provides an optimized implementation of the approach described in the Oracle white paper, *Oracle Semantic Technologies Inference Best Practices with RDFS/OWL*, which motivates and explains the use of sound and complete OWL reasoners with Oracle Database 11g via the *Jena* Adaptor for Oracle Database. The idea is to load the ontology schema from Oracle Database 11g to the in-memory PelletDb reasoner in order to compute a class subsumption tree; and then to use the Oracle Database native inference engine on the combination of the complete class subsumption tree and instance-level data.

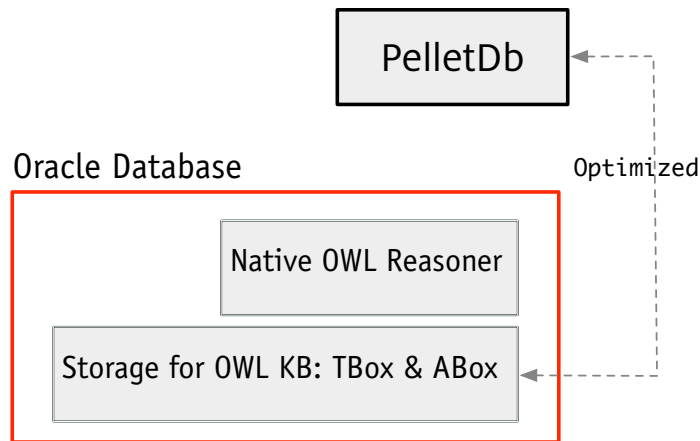


Figure 1: Architecture of PelletDb

There are several optimizations and improvements implemented in PelletDb to make the integration work efficiently. For example, loading data from a relational database to Pellet using the standard Jena interface is very inefficient: Pellet makes a large number of simple queries to the underlying database. PelletDb includes an optimized loader, which is only available in PelletDb, that minimizes the number of queries sent to the database, thus sig-

² See the [published benchmarks](#) at the PelletDb site.

nificantly reducing the loading time. In addition, if only the schema is to be read from the database, that can now be accomplished efficiently regardless of the size of the instance data. For example, loading the schema from LUBM(10,0)³ can be done on a commodity computer in 3 seconds, whereas this operation takes 10 minutes with the standard Pellet loader.

PelletDb also provides efficient mechanisms to retrieve inferences from PelletDb to save into Oracle Database 11g. PelletDb can be configured to compute only a subset of inferences relevant for an application. Further, many kinds of trivial inference (for example, every class is a subclass of itself), when added to the database, will reduce the efficiency of the Oracle Database 11g reasoner since it will have to deal with unnecessary instance data. PelletDb automatically filters such trivial inferences minimizing the amount of data that sent to the database improving overall system performance.

PelletDb reasoning covers the full expressivity of OWL 1 & 2: in fact, the combination of Pellet and Oracle Database Semantic Technologies covers the entire OWL 2 family of languages: DL, EL, QL and RL. PelletDb provides access to Pellet's reasoning services, including consistency checking, concept satisfiability, classification, realization; as well as non-standard reasoning services like SPARQL-DL conjunctive query answering, datatype reasoning, SWRL rules reasoning, inference explanation, and incremental reasoning. For example, PelletDb will detect all inconsistencies in an ontology and will compute *all* legal OWL inferences.⁴ PelletDb also provides automated explanations for the inferences it computes. If an inconsistency is detected, then the set of asserted statements causing the inconsistency can be retrieved, in order to help pinpoint and resolve the problem(s).

Even when completeness of results is not crucial for the application domain, getting *more* results can improve the quality of analysis and of the resulting decisions. PelletDb is particularly useful for applications where the modeling needs exceed the expressivity of OWL 2 RL.

2 USING PELLETDDB

The normal PelletDb usage pattern is to (1) load the data from Oracle Database 11g to PelletDb; (2) compute all requested inferences; and then (3) save the inferences to Oracle Database 11g. This pattern can be employed in one of two PelletDb reasoning modes:

PELLETDDB REASONING (MODE 1) In this mode, PelletDb loads both the schema and the instance data from Oracle Database 11g; it then computes and saves all inferences back to Oracle Database 11g. After all the inferences are computed, Oracle Database 11g can be queried without needing to do additional reasoning.

³ LUBM is a de facto standard benchmark for RDF & OWL reasoners and databases. ⁴ PelletDb does not literally compute all inferences, since there can be infinite trivial inferences; rather, PelletDb computes all inferences about named concepts or individuals.

ORACLE & PELLETDDB REASONING (MODE 2) When instance data is too large to fit into memory, PelletDb can be configured to extract *only* the schema from Oracle Database 11g, compute additional schema inferences, and save those inferences back to Oracle Database 11g. Oracle Database 11g can then be used to perform instance reasoning using the schema inferences computed by PelletDb.

PelletDb's two modes of operation have advantages and disadvantages. For example, performing full reasoning in memory ensures the completeness of all query results but cannot scale to datasets that do not fit into available memory. For such datasets, the combination of PelletDb reasoning in-memory together with instance reasoning in Oracle Database 11g provides a viable means to achieving more complete inference and query results than either solution can offer alone.

2.1 API EXAMPLES

In this section we describe in detail how to use PelletDb in Java programs. PelletDb includes convenience classes and functions to handle typical tasks. The first step is to create a `PelletDb` instance that will connect to an Oracle database:

```
// Provide the JDBC arguments for Oracle connection and
// the name of the RDF model in Oracle
PelletDb pelletdb = new PelletDb(jdbcURL, user, pwd, modelName);
```

The default configuration is to perform Mode 2 reasoning; that is, PelletDb will perform schema reasoning only. However, if instance data is small enough for in-memory reasoning, then PelletDb can be configured for Mode 1 reasoning, that is, to read instance data into memory directly:

```
// Configure PelletDb to load both the schema and the instance data
pelletdb.setConfiguration(PelletDbConfiguration.LOAD_ABOX_CONFIGURATION);
```

After the `PelletDb` instance is created, a single method call will load the data, compute the inferences, and save the results to Oracle Database 11g:

```
// load the data, compute and save the inferences
pelletdb.saveInferences();
```

Alternatively, for finer-grained control—for purposes of logging, debugging, etc.—the call to `saveInferences` can be replaced with three calls:

```
pelletdb.loadData();
pelletdb.computeInferences();
pelletdb.saveInferences();
```

The computed inferences will be saved, by default, to the same model that the data is loaded from. However, this may not be possible in cases where the user does not have write permission on the data model. In those cases, the inferences can be saved to a separate model:

```
Model inferenceModel = ...
pelletdb.saveInferences( inferenceModel );
```

Data stored in Oracle Database 11g may be updated after the inferences are computed and saved. In this case, we can reload the data to the existing PelletDb instance and save the new inferences as before:

```
pelletdb.reload();
pelletdb.saveInferences( inferenceModel );
```

Finally, all of the resources used by PelletDb can be freed after all tasks are completed:

```
pelletdb.close();
```

2.2 QUERYING PELLETDDB

The schema inferences computed by PelletDb can be queried in memory as well as in Oracle Database 11g.⁵ For example, if we would like to check for any contradictions in the class definitions, we can query PelletDb to retrieve unsatisfiable classes:

```
// Unsatisfiable classes are subclasses of owl:Nothing
StmtIterator i = pelletdb.listStatements(null, RDFS.subClassOf, OWL.Nothing);
// then iterate through the results...
```

Since an unsatisfiable class likely indicates a serious modeling or data problem, PelletDb applications should check for unsatisfiable classes and for inconsistencies at appropriate intervals after the schema or instance data is modified.

Querying the in-memory PelletDb model in this way has the benefit that, since inferences like classification results have already been computed and cached, no expensive reasoning

⁵ SPARQL queries can be performed in Oracle Database using the Jena Adaptor with a Java application or by embedded SPARQL query graphs in SQL queries.

computation is performed. The in-memory model can also be directly accessed and used like any other Jena model:

```
InfModel model = pelletdb.getPelletModel();
```

2.3 EXPLAINING INFERENCE

PelletDb can generate an explanation for any kind of inference it computes, including any inconsistencies that it detects. An explanation is a minimal set of asserted OWL axioms that is sufficient to cause the inference. Explanations can be used to understand why an inference is computed as well as to debug and repair problems in the schema, instance data, or both. To generate explanations, the first step is to enable explanation support on the PelletDb instance:

```
// Explanation should be turned on BEFORE data is loaded into PelletDb
pelletdb.setExplanationsEnabled(true);
```

Now we can extend the earlier example for detecting unsatisfiable classes in order to retrieve an explanation for each unsatisfiable class:

```
StmtIterator i = pelletdb.listStatements(null, RDFS.subClassOf, OWL.Nothing);
while( i.hasNext() ) {
    Statement stmt = i.nextStatement();
    Model explanation = pelletdb.explain(stmt);
    // process the explanation
}
```

Explanations can be generated on-demand for *any* kind of inference. For example, if two classes are inferred to be equivalent, PelletDb can provide an explanation:

```
// Why is Human equivalent to Person?
Model explanation = pelletdb.explain(Human, OWL.equivalentClass, Person);
```

2.4 CONFIGURATION

PelletDb provides fine-grained control of data loading and of the kind of inferences that are computed. Eagerly computing all possible inferences may not be useful in every case because only certain kinds of inferences are typically relevant for an application. Further, RDF

and OWL prescribe some trivial inferences that may not be useful at all;⁶ including these trivial inferences in the results makes the output unnecessarily verbose.

PelletDb provides a two-level approach to inference customization. First, an inference selector chooses the set of statement types that will be computed. These results will be complete with respect to OWL semantics. Second, a filter is used to exclude trivial inferences (or any triple pattern that is irrelevant) from the results.

For example, PelletDb by default returns only direct subclass inferences. So if we have the subclass inferences `B rdfs:subClassOf C` and `C rdfs:subClassOf D`, then the inference `B rdfs:subClassOf D` will *not* be included in the results. However, if required, the inference extractor can be customized to output the transitive closure of subclass inferences as follows:

```
// Get the default set of inference types computed
EnumSet<StatementType> inferenceSelector =
    PelletDbInferenceSelector.getDefaultSelector();

// Select all subclass inferences instead of only direct subclass inferences
inferenceSelector.remove(StatementType.DIRECT_SUBCLASS);
inferenceSelector.add(StatementType.ALL_SUBCLASS);
```

PelletDb by default also filters RDF triples of the form `?X rdfs:subClassOf ?X` since such inferences trivially hold for any class regardless of its definition. For completeness, these inferences can be included in the final result:

```
// Create a default inference filter
PelletDbInferenceFilter inferenceFilter = new PelletDbInferenceFilter();

// Remove rdfs:subClassOf from the list of reflexive predicates
// that are filtered
inferenceFilter.getReflexivePredicates().remove(RDFS.subClassOf);
```

There are other customization options provided in `PelletDb` and `PelletDbConfiguration` to control other aspects of reasoning. Please see the [PelletDb Java documentation](#) for further details.

For a more detailed discussion about developing OWL applications with Pellet, see the Clark & Parsia tutorial, *A Programmer's Introduction to Pellet: How to Build OWL Ontology-based Semantic Applications*.

3 PELLETDDB ROADMAP

In future PelletDb releases we will introduce additional features to more fully exploit the integration with Oracle Database 11g introduced in PelletDb, including:

⁶ For example, every class is equivalent to, and is a subclass of, itself.

INTEGRITY CONSTRAINT VALIDATION A future release of PelletDb will include an integrity constraints validation system, which will allow users to use OWL as an expressive schema language for semantic data validation.⁷ Users will be able to specify data validation constraints using new and existing OWL ontologies; PelletDb will then automatically generate SPARQL queries to validate RDF and OWL data using any database that supports SPARQL evaluation. Validation errors (i.e., constraint violations) will be automatically explained so that errors can be debugged and repaired.

OWL 2 QL & EL REASONERS The OWL 2 family of [profiles](#) includes EL, QL, and RL. Oracle Database 11g Semantic Technologies includes core OWL 2 RL support; and PelletDb includes complete support for OWL 2 DL. A future version of PelletDb will offer optimized support for both OWL 2 QL & EL, giving PelletDb and Oracle Database 11g customers a comprehensive suite of OWL 2 reasoners. This expanded coverage of OWL 2 profiles will extend the utility of PelletDb into a variety of useful areas including database integration and health care life sciences applications.

EXTENDING INTEGRITY CONSTRAINTS A future release of PelletDb will extend its data validation facilities beyond semantic data in RDF and OWL to include validation of relational data stored in Oracle Database 11g—using the same OWL ontology for validating RDF, OWL, and relational data. With this extension PelletDb will be capable of validating data from relational to semantic forms, presenting a single unified framework, based on OWL, for enterprise data validation.

4 ADDITIONAL INFORMATION & RESOURCES

PELLETDDB For more information about obtaining PelletDb, see [the PelletDb site](#), which includes licensing, technical support, and related information. For immediate access to sales and licensing information, contact us at sales@clarkparsia.com OR +1 202 408 8770. PelletDb requires access to an Oracle Database 11g installation.

ABOUT CLARK & PARSIA Clark & Parsia LLC is an R&D firm based in Washington, DC, with customers in government, enterprise, health care & life sciences, aerospace, financial services, oil & gas, IT, and defense industries.

We focus on infrastructure and application development for OWL, automated reasoning systems, RDF-based integration, ontology tool development, automated planning, machine learning and data mining, etc. Our licensable portfolio of semantic technologies includes:

PELLET the leading OWL reasoner for automated reasoning applications

JSPACE SPARQL-based Faceted Browser for semantic applications & data

⁷ There's an ongoing [discussion](#) of the utility of OWL-based integrity constraints in enterprise applications on the C&P weblog.

HOTPLANNER an HTN planner deeply integrated with Pellet for automated planning applications

OWLSIGHT Web 2.0, in-browser ontology browser & editor

PRONTO Pellet extension for reasoning probabilistically with information uncertainty

XACML-DL XACML policy analysis tool for security policy analysis and design-time decision support

ABOUT ORACLE Oracle (NASDAQ: ORCL) is the world's largest enterprise software company. For more information about Oracle, please visit the [Oracle Web site](#).

ORACLE DATABASE SEMANTIC TECHNOLOGIES Oracle Database Semantic Technologies⁸ are licensed as part of Oracle Spatial 11g, an option for Oracle Database 11g Enterprise Edition. They include advanced semantic data management capabilities that are superior to other commercial and open source RDF systems, including: (1) native support for RDF, RDFS, OWL, and SPARQL standards; (2) bulk and incremental loading of triples; (3) DML access to RDF/OWL data and ontologies; (4) native inference using OWL and RDFS semantics and user-defined rules; (5) mixed querying of relational and RDF/OWL data and ontologies using SPARQL-like graph query embedded in SQL; (6) programming Java APIs and SPARQL queries in Java through a Jena Adaptor; and (7) ontology-assisted querying of relational data.

The Oracle Database native inference engine supports the semantics of OWL, RDFS, and user-defined rules. It stores inferred results persistently, ahead of query time to minimize on-the-fly computation and to accelerate queries. In-database reasoning allows inferencing on the largest instance data sets and persistent storage of the largest possible inferred result sets. Oracle Database OWL reasoning supports proof generation for inferred triples and model and entailment validation to detect inconsistencies. See the Oracle Database Semantic Technologies Developer's Guide for details.

5 CONCLUSION

In this white paper,⁹ in Section 1, we've described the benefits of PelletDb, an integrated automated reasoning tool that combines the best of Pellet and Oracle Database Semantic Technologies. PelletDb's benefits include sound-and-complete OWL 2 reasoning based on the optimized integration of Pellet and Oracle Database Semantic Technologies. PelletDb provides access to the advanced reasoning and information management features of Pellet and Oracle Database Semantic Technologies. We've also discussed in Section 2 some basic methods

⁸ For more information about obtaining Oracle Database Semantic Technologies, see http://oracle.com/technology/tech/semantic_technologies. ⁹ Thanks to Xavier Lopez, Bill Beaugard, Zhe Wu, Ian Horrocks, Kevin Newman, Mike Smith, and Andrea Westerinen for timely feedback and valuable guidance.

for using PelletDb in Java programs, as well as briefly examined the future course of PelletDb development. For more information, please see Section 4.

Please see the Clark & Parsia [White Paper Series](#) for additional strategic and product studies and other semantic technology briefings.

CLARK & PARSIA, LLC

Washington, DC

June 30, 2009