TUGAS AKHIR – KI141502

IMPLEMENTASI PENYIMPANAN DATA BERBASIS ONTOLOGI PADA RDBMS

STIEVEN WIRAKASA
NRP 5111100058

Dosen Pembimbing I
Umi Laili Yuhana, S.Kom, M.Sc.

Dosen Pembimbing II
Nurul Fajrin Ariyani, S.Kom, M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

ONTOLOGY BASED DATA STORAGE IMPLEMENTATION ON RDBMS

STIEVEN WIRAKASA
NRP 5111100058

Supervisor I
Umi Laili Yuhana, S.Kom, M.Sc.

Supervisor II
Nurul Fajrin Ariyani, S.Kom, M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji dan syukur kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Implementasi Penyimpanan Data Berbasis Ontologi Pada RDBMS".

Tugas Akhir ini diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini, penulis berharap apa yang penulis telah kerjakan dapat memberikan manfaat bagi perkembangan ilmu pengetahuan serta bagi penulis.

Keberhasilan dalam penyelesaian tugas akhir ini tidak terlepas dari bantuan berbagai pihak. Maka dari itu, pada kesempatan ini penulis ingin menyampaikan rasa terima kasih yang tulus dan sebesar-besarnya kepada semua pihak, terutama kepada:

1. Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan baik
2. Keluarga tercinta yang sudah dengan sabar membesarkan penulis hingga saat ini dan senantiasa mendoakan.
3. Ibu Umi Laili Yuhana, S.Kom, M.Sc selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Ibu Nurul Fajrin Ariyani, S.Kom, M.Sc. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo

Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.

6. Bapak Dr. Agus Zainal Arifin, S.Kom., M.Kom. yang telah memberi nasihat selama penulis ketika perwalian dan memotivasi penulis selama perkuliahan berlangsung.
7. Teman-teman *sugoi daebak*: Steven, Ebenhaezer, Tracy, Ericko, Ruben, Lucky, Peter, Ivan, Kevin yang sudah menemani penulis berjuang selama pengerjaan Tugas Akhir.
8. Teman-teman IGS 2011 yang senantiasa saling mendukung satu sama lain agar bisa menyelesaikan Tugas Akhir bersama-sama.
9. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Januari 2016

LEMBAR PENGESAHAN

IMPLEMENTASI PENYIMPANAN DATA BERBASIS ONTOLOGI PADA RDBMS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
STIEVEN WIRAKASA
NRP: 5111 100 058

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Umi Laili Yuhana, S.Kom, M.Sc. .....
   NIP: 197906262005012002 (Pembimbing 1)

2. Nurul Fajrin Ariyani, S.Kom, M.Sc. .....
   NIP: 051100124 (Pembimbing 2)

SURABAYA
JANUARI,2016

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI PENYIMPANAN DATA BERBASIS ONTOLOGI PADA RDBMS

Nama Mahasiswa	**: STIEVEN WIRAKASA**
NRP	**: 5111100058**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Umi Laili Yuhana, S.Kom, M.Sc.**
Dosen Pembimbing 2	**: Nurul Fajrin Ariyani, S.Kom, M.Sc.**

Abstrak

Dari hasil penelitian ini dapat disimpulkan bahwa ontologi dapat disimpan dengan baik pada basis data Oracle. Data ontologi yang tersimpan pada basis data Oracle pada basis data Oracle juga dapat digunakan sebagai konteks dalam aplikasi dengan baik.Di era dimana informasi semakin memegang peranan penting ini, ontologi mulai semakin digunakan untuk membangun aplikasi untuk domain tertentu. Ontologi memungkinkan pengguna untuk menangkap semantik dokumen. Kinerja sistem mampu meningkat secara drastis dengan ekstraksi domain informasi yang spesifik. Sementara ontologi semakin berkembang, muncul masalah baru yaitu semakin besar ontologi yang dihasilkan dan data yang disimpan, semakin lama waktu yang dibutuhkan untuk menyelesaikan proses inference dan query data. Lamanya waktu untuk menyelesaikan proses query dikarenakan kakas bantu pembangun ontologi membaca file teks seperti XML, OWL, RDF, N-Triple, dan lain lain.

Dalam tugas akhir ini penulis memanfaatkan fitur-fitur penyimpanan data dan pemrosesan query yang dimiliki Oracle Semantic untuk menyimpan skema dan data ontologi ke dalam bentu006B tabel relasional. Namun demikian pada proses pemasukkan data ontologi ke dalam tabel membutuhkan waktu yang lama mengingat pasangan instance dan properti atau data triple harus dimasukkan satu per satu melalui sintak INSERT. Untuk mempermudah memasukkan skema dan data ontologi,

penulis membangun sebuah aplikasi yang bisa digunakan untuk mengkonversi file ontologi seperti RDFS, OWL, XML kedalam tabel relasional.

Pengujian aplikasi dilakukan dengan memperhatikan aspek fungsionalitas, kemampuan query melakukan inference, waktu eksekusi query dan usablity (kemudahan). Hasil pengujian menunjukkan bahwa pengembalian (query) data ontologi yang tersimpan pada basis data relasional membutuhkan waktu eksekusi yang lebih cepat dibandingkan dengan pengambilan data ontologi yang tersimpan dalam bentuk file teks. Hasil query menunjukkan bahwa Oracle Semantic mampu menangani pengambilan data ontologi inference dari beberapa karakteristik properti yang terdapat pada Ontologi. Penyimpanan skema dan data ontologi juga bisa dilakukan lebih cepat dengan menggunakan aplikasi yang telah dibangun oleh penulis.

Kata kunci: Ontologi, OWL, RDF, XML, Query, Semantik, Oracle Semantic

ONTOLOGY BASED DATA STORAGE IMPLEMENTATION ON RDBMS

Student's Name	**: STIEVEN WIRAKASA**
Student's ID	**: 5111100058**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Umi Laili Yuhana, S.Kom, M.Sc.**
Second Advisor	**: Nurul Fajrin Ariyani, S.Kom, M.Sc.**

Abstract

In the era where information holds an important role, Ontology is becoming more to be used to build an application for a particular domain. Ontology enables the user to capture the semantic document. The Performance of the system is able to rise drastically by extraction of domain-specific information. While the use of Ontology is growing, a new problem arises, the greater the generated ontologies and data are stored, the longer it takes to complete the process of querying the data. The length of time to complete the query process due tools for ontology read text files such as XML, OWL, RDF, N-Triple, and others.

In this thesis, the author utilizes the features of data storage and query process owned by Oracle Semantic to store ontology data and schema into a relational database. However, in the process of ontology data entry into a relational table takes a long time considering the partner instance and property or triple the data must be entered one by one through INSERT syntax. To make it easier to enter the data and schema ontologies, the authors build an application that can be used to convert files such as RDFS ontologies, OWL, the XML into relational tables.

The application testing carried out by considering the functionally, inferencing ability, query execution time and ease of usability. The testing results showed that retrieval of ontology data which is stored in a relational table requires a faster execution time than ontology data retrieval which is

stored in text files. The query results showed that Oracle Semantic is able to handle inferred ontology data retrieval of some property characteristic contained in Ontology. The ontology schema and data storing can also be done much faster and easily by using application that have been built by the author.

Keywords: OWL, RDF, XML, N-Triple, Query, Semantik, Oracle.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ontologi mulai semakin digunakan untuk membangun aplikasi untuk domain tertentu. Ontologi memungkinkan pengguna untuk menangkap semantik dokumen. Kinerja sistem mampu meningkat secara drastis dengan ekstraksi domain informasi yang spesifik.

Penggunaan ontologi sangatlah luas, mulai dari kesehatan, keamanan, *World Wide Web*, dan lain lain. Untuk menginterpretasikan informasi dan melakukan penalaran, kita perlu menyimpan Ontologi dengan cara yang benar, konsisten, terukur dan efisien untuk mengambil kembali informasi yang tersimpan dalam ontologi.

Saat ini, ontologi semakin berkembang dengan adanya kakas bantu pembangun ontologi, seperti protégé, OWLGrEd, dan lain lain. Ontologi yang dihasilkan oleh kakas bantu itu berupa file teks, seperti XML, RDF, dan OWL. Sementara ontologi semakin berkembang, muncul masalah baru yaitu semakin besar ontologi yang dihasilkan dan data yang disimpan, semakin lama waktu yang dibutuhkan untuk menyelesaikan proses *query* data.

Pada kakas bantu berbasis ontologi, seperti protégé, menggunakan file teks sebagai cara untuk menyimpan data berbasis ontologi. Setiap kali ontologi di*query* oleh kakas bantu tersebut, kakas bantu akan membaca file teks, seperti XML maupun RDF. Kakas bantu akan membaca semua teks untuk membuat skema ontologi dan menyimpannya dalam memori, kemudian memetakan data berbasis ontologi ke dalam ontologi. Namun semakin besar skema ontologi dan data

berbasis ontologi yang disimpan, semakin besar memori yang dibutuhkan untuk menyimpan dan tentunya semakin lama waktu untuk melakukan proses *query* data.

Pada tugas akhir ini, penulis mengimplementasikan teknik untuk menyimpan ontologi ke dalam sebuah RDBMS. Dengan memanfaatkan *fitur* yang dimiliki oleh RDBMS Oracle, diharapkan proses *query* data ontologi menjadi lebih cepat. Dalam praktiknya, skema model dan data ontologi akan disimpan ke dalam beberapa tabel yang memiliki indeks, sehingga diharapkan ontologi dapat di*query* dengan cepat oleh aplikasi [1]. Dengan menerapkan metode ini, aplikasi-aplikasi yang menggunakan ontologi untuk menyimpan konteks akan lebih layak untuk diimplementasikan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana memodelkan penyimpanan skema ontologi ke dalam RDBMS?
2. Bagaimana memodelkan data berbasis ontologi ke dalam RDBMS?
3. Bagaimana mengoptimalkan proses *query* data berbasis ontologi dalam RDBMS?

1.3 Batasan Masalah

Permasalahan yang dibahas pada tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi dibangun berupa aplikasi desktop untuk mengembalikan hasil *query* data berbasis ontologi.
2. RDBMS yang digunakan dari Oracle.
3. Tidak membangun ontologi melainkan lebih fokus terhadap kinerja *query*.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini adalah mengimplementasikan metode penyimpanan ontologi pada RDBMS untuk mempercepat proses pengembalian data atau informasi pada aplikasi berbasis ontologi.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini untuk mengimplementasikan metode penyimpanan ontologi ke dalam RDBMS sehingga aplikasi-aplikasi yang berbasis ontologi dapat melakukan *query* dengan waktu yang lebih cepat.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini.

2. Studi literatur
   Pada tahapan ini dilakukan proses pengayaan wawasan dengan mempelajari literatur-literatur yang telah ada seputar topik Tugas Akhir ini.

3. Analisis dan desain perangkat lunak
   Proses analisis mengenai modul yang dibuat diawali dengan pengamatan terlebih dahulu terhadap sistem informasi akademik yang sudah ada pada berbagai perguruan tinggi di Indonesia. Berdasarkan hasil pengamatan tersebut

dirumuskan aktivitas-aktivitas apa saja yang umum terdapat pada proses penilaian. Kemudian hasil rumusan tersebut akan dijadikan acuan terhadap spesifikasi kebutuhan modul. Spesifikasi kebutuhan modul tersebut kemudian akan digunakan sebagai acuan dalam membuat desain modul.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini direalisasikan spesifikasi kebutuhan yang dihasilkan oleh tahapan sebelumnya, sehingga menjadi sebuah purwarupa modul sistem informasi yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahapan ini dilakukan uji coba terhadap purwarupa sistem informasi yang sudah dibuat pada tahap sebelumnya. Pengujian dan evaluasi dilakukan dengan menggunakan bantuan dari beberapa orang penguji. Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian antara spesifikasi kebutuhan dan purwarupa sistem informasi serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian yang disusun secara sistematis seperti berikut ini:

Bab I Pendahuluan

Bab ini menjelaskan tentang latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir ini, serta batasan masalah dan metodologi yang digunakan dalam pembuatan Tugas Akhir ini.

Bab II Tinjauan Pustaka

Bab ini menjelaskan istilah-istilah khusus serta dasar-dasar teori yang digunakan dalam pembuatan Tugas Akhir ini.

Bab III Analisis Dan Perancangan Sistem

Bab ini membahas rancangan sistem yang dibangun, mulai dari tahap penggalian kebutuhan, tahap perancangan kebutuhan fungsional sistem, sampai pada tahap perancangan arsitektur sistem.

Bab IV Implementasi

Bab ini membahas tahap implementasi dari rancangan yang telah dibuat pada bab sebelumnya.

Bab V Uji Coba Dan Evaluasi

Bab ini membahas tentang perencanaan uji coba serta hasil uji coba yang dilakukan terhadap sistem. Evaluasi dari hasil uji coba tersebut juga dibahas di dalam bab ini.

Bab VI Kesimpulan Dan Saran

Bab ini menyampaikan kesimpulan dari hasil pengerjaan Tugas Akhir yang dilakukan dan saran untuk pengembangan Tugas Akhir ke depannya.

(halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan dari istilah-istilah khusus dan dasar-dasar teori yang digunakan dalam Tugas Akhir. Istilah dan dasar teori yang dibahas juga mencakup teknologi-teknologi yang digunakan pada Tugas Akhir.

2.1 Ontologi

Ontologi adalah suatu konseptual yang formal dari sebuah domain tertentu yang dipakai bersama oleh kelompok orang. Sedang menurut Tom Grubber, ontologi merupakan teori tentang makna dari suatu obyek, properti dari suatu obyek, serta relasi obyek tersebut yang mungkin terjadi pada suatu domain pengetahuan. Ontologi sangat penting karena dapat digunakan menerangkan tentang struktur suatu disiplin ilmu [2]

Secara teknis sebuah ontologi direpresentasikan dalam bentuk *Class*, *property*, dan *instance*. Berikut penjelasan class, property, dan instance yang akan digunakan dalam tugas akhir ini:

1. *Class*

   Konsep (atau makna) suatu domain. *Class* adalah kumpulan dari elemen dengan properti yang sama. Suatu *Class* dapat mempunyai turunan *subclass* yang menerangkan konsep yang lebih spesifik. Dalam Class, juga terdapat yang disebut Disjoint Class. *DisjointWith* merupakan relasi dari kedua kelas yang tidak dapat memiliki anggota yang sama, misalkan kelas manusia dan kelas hewan tidak memungkinkan adanya individu atau instance yang terdapat pada dua kelas tersebut.
2. Properti

   Merupakan relasi dari instance atau *Class* yang terdapat dalam ontologi. Properti digolongkan menjadi dua yaitu *datatype property* atau properti tipedata dan *object property* atau properti objek.

a. Property Objek
   Properti objek digunakan untuk menghubungkan subjek dan objek yang berupa individual. Pada properti objek
b. Properti Tipedata
   Propeerti tipedata digunakan untuk menghubungkan individual dengan data literal seperti tahun kelahiran, tahun kematian, dan lain lain.

3. *Instance*
   Individual yang telah dibuat (diciptakan). *Instance* dari sebuah *Subclass* merupakan instant dari suatu *superClass* [3].

Dalam properti terdapat karakteristik yang berbeda-beda, seperti:

- *Functional*:
  Properti yang hanya dapat memiliki satu anggota yang masuk di dalam range untuk domain apapun atau biasa disebut sebagai relasi many to one.
  Contohnya, Budi hasFather Ali.Adi hasFather Ali. Budi dan Ali hanya boleh memiliki satu relasi “hasFather”. Sedangkan Ali dapat memiliki dimiliki oleh banyak relasi “hasFather”
- *Inverse functional*:
  Properti yang memiliki relasi one to one. Contoh Inverse functional property ialah “hasID”. Dimana setiap orang hanya memiliki satu ID dan setiap ID hanya boleh dimiliki oleh satu orang.
- *Transitive*
  Properti yang menyatakan relasi seperti contoh berikut "x hasSibling y" and "y hasSibling z" implies "x hasSibling z".
- *Symmetric*
  Properti yang menyatakan relasi seperti contoh berikut "x hasBloodRelation y" maka “y hasBloodRelation x”.
- *ASymmetric*
  Properti yang menyatakan relasi seperti contoh berikut jika "x hasBloodRelation y" maka tidak berlaku “y hasBloodRelation x”.
- *Reflexive*
  Properti yang menyatakan relasi "x PropertyA x" selalu benar.

- *Irreflexive*
  Properti yang menyatakan relasi "x PropertyA x" selalu salah [2].

2.2 Inference

Pada web semantik, sumber dari informasi tambahan dapat didefinisikan melalui kosakata atau kumpulan set aturan. Kedua pendekatan ini memanfaatkan teknik representasi pengetahuan. Secara umum, ontologi fokus pada metode klasifikasi, menekankan pada defines Class, subClass, tentang bagaimana resource individual dapat dikaitkan dengan suatu Class, dan karakterisasi relasi antara Class dan instance. Rules, sebaliknya, fokus pada mendefinisikan sebuah mekanisme umum pada menemukan dan menghasilkan relasi baru berdasarkan yang sudah ada [4].

2.3 Java

Java di sini dapat diartikan sebagai sebuah platform perangkat lunak maupun sebagai sebuah bahasa pemrograman. Sebagai sebuah platform, Java memungkinkan sebuah pengembangan perangkat lunak yang dapat dijalankan dalam lingkungan apapun. Java telah digunakan dalam berbagai lingkungan seperti perangkat *embedded*, perangkat bergerak, maupun server dan komputer personal. Sedangkan bahasa pemrograman Java adalah bahasa yang bersifat umum, *concurrent,* berbasis kelas, dan berorientasi objek. Bahasa ini juga bersifat *strongly* dan *statically typed*. Java juga adalah bahasa pemrograman yang secara relatif dapat dianggap sebagai *high-level* (tingkat tinggi), yang berarti bahasa pemrograman Java memiliki tingkat abstraksi cukup tinggi dari tingkat bahasa instruksi komputer seperti bahasa *assembly* [4].

Agar sebuah aplikasi Java dapat dijalankan, maka file dengan ekstensi Java akan dikompilasi menjadi sebuah *bytecode*. Untuk menjalankan *bytecode* ini, dibutuhkan *Java Runtime Environment* (JRE). JRE berisi sebuah *Java Virtual Machine*

(JVM) dan *library* (pustaka) yang dibutuhkan dalam menjalankan sebuah aplikasi Java. JVM inilah yang memungkinkan sebuah aplikasi Java untuk bisa dijalankan dalam lingkungan yang berbeda-beda.

2.4 SPARQL

SPARQL (singkatan rekursif dari SPARQL Protokol dan RDF *Query* Language) adalah bahasa *query* RDF dan bahasa *query* untuk database yang mampu mengambil dan memanipulasi data yang disimpan dalam kerangka Resource Description Format. SPARQL dibuat oleh RDF Data Access Working Group (DAWG) dari World Wide Web Consortium, dan dianggap sebagai salah satu teknologi kunci dari web semantik [2].

2.5 OWL

OWL adalah bahasa ontologi yang baru untuk sebuah web semantik, dikembangkan oleh World Wide Web Consortium (W3C) kelompok kerja Web Ontologi. Pada mulanya OWL didesain untuk merepresentasikan informasi tentang kategori dari sebuah objek dan bagaimana objek tersebut berhubungan. OWL dapat juga menyediakan informasi tentang objek itu sendiri.

Sebagai hasil usaha yang dilakukan oleh kegiatan Web Semantik W3C, OWL harus dapat cocok dengan visi Web Semantik yaitu bahasa yang dikelompokkan bersama-sama dengan XML dan RDF. OWL yang diharapkan menjadi salah satu bahasa ontologi, harus dapat merepresentasikan bagian-bagian yang berguna dalam sebuah ontologi.

Dalam usahanya untuk mendukung kemampuan dan skenario yang telah disepakati, OWL menggunakan kemampuan RDF untuk penjabaran statis dan kemampuan struktur Class dan property dari skema RDF dan menyisipkannya ke dalamnya.

OWL dapat mendeklarasikan Class, dan mengorganisasikan Class tersebut ke dalam hirarki sub-Class, sama seperti skema RDF. Class OWL dapat dijelaskan sebagai kombinasi logical (irisan, gabungan, komplemen) dari Class lainnya, atau sebagai penjelasan satu-persatu dari objek yang dimaksud, melebihi kemampuan skema RDF.

OWL dapat juga mendeklarasikan property, mengorganisasikan property tersebut ke dalam hirarki "subproperty", dan menyediakan domain dan range untuk property tersebut, seperti pada skema RDF.

Domain dari property OWL adalah Class dalam OWL, dan range dapat berupa Class dalam OWL atau tipe data yang dideklarasikan dari luar seperti string atau integer. OWL dapat menetapkan bahwa property tersebut adalah transitif, asimetrik, fungsional atau bertolak belakang dengan property lainnya [5].

Bahasa OWL menyediakan 3 sub-bahasa yang didisain untuk komunitas dari pengguna yang spesifik.

- OWL Lite mendukung pengguna terutama yang membutuhkan hirarki klasifikasi dan fitur batasan yang sederhana. Sebagai contoh, sementara OWL mendukung batasan kardinalitas, hanya memperbolehkan nilai kardinalitas 0 atau 1. Seharusnya dapat menjadi lebih sederhana untuk menyediakan kakas bantu yang mendukung OWL Lite dari pada *relatives* yang lebih ekspresive, dan menyediakan *migration path* yang cepat untuk thesauri dan taxonomi yang lain.
- OWL DL mendukung pengguna yang menginginkan keekspresifan yang maksimum tanpa kehilangan kelengkapan komputasional dan *decidability* dari suatu sistem penalaran. OWL DL mencakup semua konstruksi bahasa OWL dengan restriksi sepeti separasi tipe (sebuah kelas tidak bisa menjadi individual atau properti, sebuah properti tidak dapat menjadi

class atau individual). OWL DL dinamakan sesusai dengan deskripsi logika (Description Logics). OWL DL didisain untuk mendukung segmen usaha deskripsi logika dan memiliki properti komputasional untuk sistem penalaran.

- OWL Full digunakan untuk pengguna yang menginginkan keekspresifan yang maksimal dan kebebasan sintaksis RDF dengan tanpa jaminan komputasional. Sebagai contoh, dalam OWL Full sebuah *class* dapat diperlakukan secara bersamaan sebagai sebuah koleksi dari individual dan sebagai individual yang berdiri sendiri. Perbedaan penting lainnya dari OWL DL adalah sebuah *owl:DatatypeProperty* dapat ditandai sebagai *owl:InverseFunctionalProperty*. OWL Full memperbolehkan sebuah ontologi untuk menambahkan arti dari kosakata yang belum didefinisikan. Hal ini tidak memungkinkan semua perangkat lunak penalaran untuk menggunakan setiap fitur dari OWL Full [6].

2.6 Protégé

Protégé adalah editor ontologi dan sistem akuisisi pengetahuan yang gratis. Aplikasi yang dikembangkan oleh protégé digunakan dalam pemecahan masalah dan pembuatan keputusan dalam sebuahdomain.Protégé dikembangkan oleh sebuah organisasi yang bernaung di bawah Stanford, yang mengambil spesialisasi di bidang ontologi.

Protégé merupakan sebuah alat yang digunakan untuk membuat sebuah domain ontologi, menyesuaikan form untuk entri data, dan memasukkan data.Berbagai format penyimpanannya seperti OWL, RDF, XML dan HTML. Protégé menyediakan kemudahan *plug and play* yang membuatnya fleksibel untuk pengembangan prototype. Protégé dibuat dengan menggunakan bahasa pemrograman Java. Semua alat-alat dalam protégé dapat digunakan melalui

Graphical User Interface (GUI) dengan menyediakan Tab untuk masing-masing bagian dan fungsi standar.

Class Tab dalam editor ontology berfungsi untuk mendefinisikan Class dan hirarki Class, property dan nilai property tersebut, relasi antara Class dan property dari relasi tersebut [3].

2.7 Basis Data Oracle

Pada tahun 1977, Larry Ellison, Bob Miner, and Ed Oates memulai laboratorium konsultasi pengembangan perangkat lunak dimana beruabh menjadi Relational Software, Inc. (RSI). Pada tahun 1983, RSI menjadi Oracle Systems Corporation yang kemudian berubah menjadi Oracle Corporation.

Sebuah model relational adalah dasar dari sebuah sistem manajemen basis data (RDBMS). Pada dasarnya, sebuah RDBMS memindahkan data ke dalam basis data, menyimpan data, dan mengambil kembali data tersebut sehingga aplikasi dapat memanipulasi data tersebut. Sebuah RDBMS dibedakan menjadi beberapa tipe operasinya yaitu:

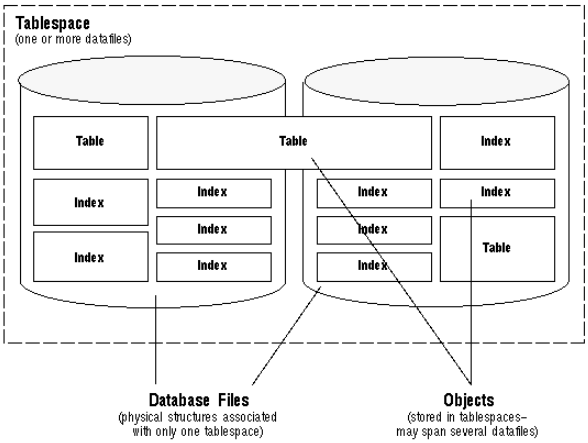
- *Logical operations*
  Sebuah aplikasi yang menentukan konten apa yang siperlukan. Misal, sebuah aplikasi meminta nama seorang pegawai atau menambah data pegawai ke dalam sebuah tabel.
- *Physical operations*
  RDBMS menentukan bagaimana sesuatu harus diselesaikan dan menjalankan operasi tersebut. Misal setelah sebuah aplikasi melakukan *query* pada suatu tabel, basis data itu mungkin menggunakan sebuah indeks untuk mencari baris yang diminta, membaca data ke dalam

memori, dan melakukan langkah langkah lain sebelum mengembalikan hasil *query* kepada pengguna. RDBMS menyimpan dan mengambil kembali data agar physical operations menjadi transparan terhadap aplikasi basis data.

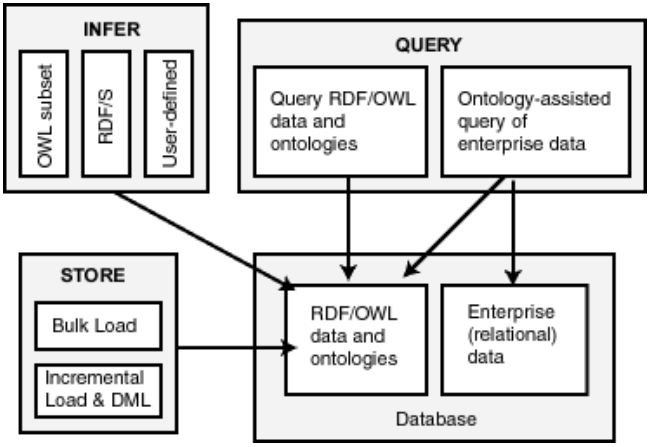
Basis data Oracle adalah RDBMS. Sebuah RDBMS yang mengimplemntasikan fitur berbasis objek seperti *inheritance, polymorphism*, dan *self-defined type* yang disebut sistem manajemen basis data objek relational (ORDBMS). Basis data Oracle telah memperluas model relational menjadi model objek relasional sehingga dapat memungkinkan untuk menyimpan model bisnis yang rumit ke dalam RDBMS

Dalam basis data Oracle, sebuah basis data dibagi menjadi satu atau lebih unit penyimpanan logical yang disebut *tablespace*. Data dari suatu basis data secara kolektif disimpan dalam *tablespace* suatu basis data. Setiap *tablespace* di dalam basis data Oracle terdiri dari satu atau lebih file-file sistem operasi yang disebut *datafiles*. Sebuah tablespace dari suatu datafiles secara fisik menyimpan data basis data terkait dalam disk.

Basis data Oracle dapat digunakan untuk menyimpan data semantik dan ontologi, untuk melakukan *query* data semantik dan untuk melakukan *query* berbasis ontologi dari data relasional perusahaan, dan untuk menggunakan *inference* yang sudah disediakan atau yang digunakan pengguna(self-defined) untuk memperluas kekuatan *query* dari data semantik. Gambar 2.1 menunjukan kemampuan Oracle semantic.



Gambar 2.1 Ilustrasi Tablespace, datafile, dan basis data



Gambar 2.2 Kemampuan *Oracle Semantic* [7]

Seperti yang ditunjukan Gambar 2.2, Basis data memuat data semantik dan ontologi (model RDF/OWL), serta data relasional tradisonal. Untuk memasukkan data semantik, *bulk loading* merupakan pendekatan yang sangat efisien, meskipun pengguna dapat memasukkan data secara bertahap (*incremental*)

dengan statemen *INSERT*. Pengguna juga dapat melakukan *query* ontology-assisted dan data relasional tradisional untuk mencari relasi dari data-data tersebut. Dalam Oracle terdapat beberapa prosedur dan fungsi pada SEM_APIS *Package Subprogram* yaitu:

2.7.1 SEM_APIS.CREATE_ENTAILMENT

Prosedur ini digunakan untuk membuat sebuah *entailment* atau index aturan yang dapat digunakan untuk melakukan OWL atau RDF inference, juga termasuk menggunakan aturan yang didefinisikan pengguna. Pada Tabel 2.1 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.3.

Tabel 2.1 Penjelasan parameter prosedur SEM_APIS.CREATE_ENTAILMENT

Nama parameter	Tipe Data	Deskripsi
entailment_name _in	**Varchar2**	**Nama *entailment* yang akan dibuat**
models_in	**SEM_MODEL**	**Satu atau lebih nama model ontologi. Tipe data SEM_MODEL memiliki definisi lain yaitu *Table Of Varchar2(25)***
rulebases_in	**SEM_RULEBASES**	**Satu atau lebih nama rulebase. Tipe data SEM_RULEBASES memiliki definisi lain yaitu**
passes	**Number**	**Jumlah putaran dimana mesin**

	Default: **SEM_APIS.** **REACH_CLOSURE**	**inference harus dijalankan.**
inf_components_in	**VARCHAR2** **Default: null**	**Sebuah string yang dipisahkan koma, untuk melakukan selektif atau inferencing berbasis komponen.** **Jika parameter bernilai null, maka inference akan menggunakan sebuah set komponen *default***
Options	**Varchar2** **Default: Null**	**Sebuah string yang dipisahkan koma sebauh *option* untuk mengontrol proses inference dengan melakukan *overriding*.** **Untuk mengaktifkan sebuah *option*, masukan '*nama-option*=T', untuk menonaktifkan, masukkan '*nama-option* = F'.** **Contoh option:** **Col_Compress, Distance, Dop, Entail_Anyway, Hash_Part, Inc, Opt_Sameas, Raw8, Proof, And User_Rules.**
Delta_in	**Sem_Models** **Default: Null**	**Jika inference incremental dijalankan, tentukan satu atau dua model dimana akan dijalankan untuk**

		melakukan inference incremental. Tipe data parameter ini adalah SEM_MODEL dimana memiliki definisi lain yaitu *Table of Varchar2(25)*
Label_gen	**Rdfsa_Labelgen** **Default: Null**	**Sebuah instance dari mdsys.rdfsa_labelgen, mendefinisikan logika untuk membuat label *Oracle Label Security* (OLS) untuk *Triple* yang ter*infered*.**

```
sem_apis.create_entailment(
        entailment_name_in => 'contracts_inf',
        models_in          => SDO_RDF_Models('contracts'),
        rulebases_in       => SDO_RDF_Rulebases('contracts_rb'),
        options            => 'USER_RULES=T',
        label_gen          => sem_rdfsa.LABELGEN_PREDICATE);
```

Gambar 2.3 Memanggil *procedure* Sem_Apis. Create_entailment

2.7.2 SEM_APIS.CREATE_RULEBASE

Prosedur ini digunakan untuk membuat sebuah rulebase pada yang akan digunakan untuk melakukan *inference Triple* yang tersimpan pada model ontologi. *Rulebase* yang dibuat merupakan rulebase yang didefinisikan user. Setelah *rulebase* dibuat, pengguna dapat menambahkan aturan pada *rulebase*. Untuk menggunakan aturan yang berada di dalam *rulebase* dalam sebuah *query* dari data RDF, *rulebase* dapat dipakai dalam fungsi tabel SEM_MATCH. Pada Tabel 2.2 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.4.

Tabel 2.2 Penjelasan parameter prosedur SEM_APIS.CREATE_RULEBASE

Nama parameter	Tipe Data	Deskripsi
rulebase_name	**Varchar2**	**Nama *rulebase* yang akan dibuat**

```
EXECUTE SEM_APIS.CREATE_RULEBASE('family_rb');
```

Gambar 2.4 Memanggil *Procedure* SEM_APIS.CREATE_RULEBASE

2.7.3 SEM_APIS.CREATE_SEM_MODEL

Prosedur ini digunakan untuk membuat sebuah model ontologi atau semantik yang digunakan untuk menyimpan data semantik atau ontologi pada basis data ontologi. Sebelum model ontologi dibuat melalui prosedur ini, tabel yang digunakan untuk menyimpan refrensi data semantik sudah dibuat terlebih dahulu. Pada Tabel 2.3 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.5.

Tabel 2.3 Penjelasan parameter prosedur SEM_APIS.CREATE_SEM_MODEL

Nama parameter	Tipe Data	Deskripsi
model_name	**Varchar 2**	**Nama model ontologi**
Table_name	**Varchar 2**	**Nama tabel yang menyimpan refrensi untuk data semantik untuk model ontologi yang akan dibuat**

column_name	**Varchar 2**	**Nama kolom yang bertipe SDO_RDF_TRIPLE_S pada tabel *table_name***
Model_tablespace	**Varchar 2** **Default: NULL**	**Nama *tablespace* untuk tabel dan objek basis data lain yang digunakan oleh Oracle untuk mendukung model ontologi ini.** **Nilai default dari kolom ini adalah tablespace yang ditentukan dalam memanggil SEM_APIS.CREATE_SEM_NETWORK**

```
EXECUTE SEM_APIS.CREATE_SEM_MODEL('articles', 'articles_rdf_data', 'triple');
```

Gambar 2.5 Memanggil Prosedur SEM_APIS.CREATE_SEM_MODEL

2.7.4 SEM_APIS.CREATE_SEM_NETWORK

Prosedur ini digunakan untuk membuat struktur yang penyimpanan data semantik. Prosedur ini digunakan untuk membuat tabel sistem dan objek basis data lain yang akan digunakan untuk teknologi semantik. Sebelum procedur ini dipanggil, tablespace dan objek basis data yang lain harus sudah dibuat. Ukuran file datafile yang dipakai untuk membuat tablespace bergantung pada besar data semantik.

Untuk memanggil prossedur ini, perlu memiliki hak akses user dengan hak akses DBA dan hanya dapat digunakan satu kali dalam suatu basis data. Untuk menghapus struktur penyimpanan data semantik, diperlukan hak akses DBA dan memanggil prosedur SEM_APIS.DROP_SEM_NETWORK. Pada Tabel 2.4 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.6.

Tabel 2.4 Penjelasan parameter prosedur SEM_APIS.CREATE_SEM_NETWORK

Nama parameter	Tipe Data	Deskripsi
tablespace_name	**Varchar2**	**Nama tablespace yang akan digunakan untuk menyimpan data semantik**

```
CREATE TABLESPACE rdf_tblspace
 DATAFILE '/oradata/orcl/rdf_tblspace.dat' SIZE 1024M REUSE
 AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED
 SEGMENT SPACE MANAGEMENT AUTO;
. . .
EXECUTE SEM_APIS.CREATE_SEM_NETWORK('rdf_tblspace');
```

Gambar 2.6 Memanggil Prosedur SEM_APIS.CREATE_SEM_NETWORK

2.7.5 SEM_APIS.ALTER_MODEL

Prosedur ini digunakan untuk melakukan perubahan pada model. Untuk sementara, prosedur ini digunakan untuk memindahkan suatu model ke dalam tablespace yang dituju. Pada Tabel 2.5 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.7

Tabel 2.5 Penjelasan parameter prosedur SEM_APIS.ALTER_MODEL

Nama parameter	Tipe Data	Deskripsi
model_name	**Varchar2**	**Nama model yang akan dipindahkan**

Command	**Varchar2**	**Harus berisi *string* "MOVE"**
Tablespace_name	**Varchar2**	**Nama tablespace yang dituju**
Parallel	**Number(38)** **Defult: NULL**	**Tingkat sifat parallel yang terkait dengan operasi eksekusi parallel.**

```
EXECUTE SEM_APIS.ALTER_MODEL('family', 'MOVE',  'my_tbs');
```

Gambar 2.7 Memanggil prosedur SEM_APIS.ALTER_MODEL

2.7.6 SEM_APIS.ALTER_ENTAILMENT

Prosedur ini digunakan untuk melakukan perubahan pada model. Untuk sementara, prosedur ini digunakan untuk memindahkan suatu model ke dalam tablespace yang dituju. Pada Tabel 2.6 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.8.

Tabel 2.6 Penjelasan Parameter Prosedur SEM_APIS.ALTER_ENTAILMENT

Nama parameter	Tipe Data	Deskripsi
Entailment_name	**Varchar2**	**Nama model yang akan dipindahkan**
Command	**Varchar2**	**Harus berisi *string* "MOVE"**
Tablespace_name	**Varchar2**	**Nama tablespace yang dituju**

Parallel	**Number(38)** **Defult: NULL**	**Tingkat sifat parallel yang terkait dengan operasi eksekusi parallel.**

```
EXECUTE SEM_APIS.ALTER_ENTAILMENT('rdfs_rix_family', 'MOVE',  'my_tbs');
```

Gambar 2.8 Memanggil Prosedur SEM_APIS.ALTER_ENTAILMENT

2.7.7 SEM_APIS.DROP_ENTAILMENT

Prosedur ini berfungsi untuk menghapus *entailment* atau index aturan yang ada juga termasuk index aturan atau entailment yang dibuat melalui prosedur SEM_APIS.CREATE_ENTAILMENT. Pada Tabel 2.7 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.9.

Tabel 2.7 Penjelasan Parameter Prosedur SEM_APIS.DROP_ENTAILMENT

Nama parameter	Tipe Data	Deskripsi
Entailment_name	**Varchar2**	**Nama *entailment* yang akan dihapus atau *drop***

```
EXECUTE sem_apis.drop_entailment('owltst_idx');
```

Gambar 2.9 Memanggil prosedur SEM_APIS.DROP_ENTAILMENT

2.7.8 SEM_APIS.DROP_RULEBASE

Prosedur ini berfungsi untuk menghapus rulebase yang tersimpan pada skema MDSYS dan hanya pemilik rulebase yang dapat menggunakan prosedur ini. Jika rulebase sudah dihapus, maka rulebase yang telah terhapus tidak dapat dipanggil oleh

fungsi SEM_MATCH. Pada Tabel 2.8 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.10.

Tabel 2.8 Penjelasan Parameter Prosedur SEM_APIS.DROP_RULEBASE

Nama parameter	Tipe Data	Deskripsi
Rulebase_name	**Varchar2**	**Nama rulebase yang akan dihapus atau *drop***

```
EXECUTE SEM_APIS.DROP_RULEBASE('family_rb');
```

Gambar 2.10 Memanggil prosedur SEM_APIS.DROP_RULEBASE

2.7.9 SEM_APIS.DROP_SEM_MODEL

Prosedur ini digunakan untuk menghapus atau melakukan perintah *drop* suatu model ontologi. Prosedur ini akan digunakan untuk menghapus model dari *view* MDSYS.SEM_MODEL$ dan prosedur ini hanya satu-satunya cara untuk menghapus model pada skema MDSYS. Perlu diingat bahwa hanya pemilik model yang dapat menghapus model tersebut. Pada Tabel 2.9 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.11.

Tabel 2.9 Penjelasan Parameter Prosedur SEM_APIS.DROP_SEM_MODEL

Nama parameter	Tipe Data	Deskripsi
model_name	**Varchar2**	**Nama model yang akan dihapus atau *drop***

```
EXECUTE SEM_APIS.DROP_SEM_MODEL('articles');
```

Gambar 2.11 Memanggil prosedur SEM_APIS.DROP_SEM_MODEL

2.7.10 SEM_APIS.DROP_SEM_NETWORK

Prosedur ini digunakan untuk menghapuskan semua struktur yang digunakan sebagai penyimpanan data semantik. Untuk menghapuskan struktur yang digunakan sebagai penyimpanan data semantik, diperlukan hak akses DBA untuk memanggil prosedur ini. Pada Tabel 2.10 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.12.

Tabel 2.10 Penjelasan Parameter Prosedur SEM_APIS.DROP_SEM_NETWORK

Nama parameter	Tipe Data	Deskripsi
model_name	**Varchar2**	**Nama model yang akan dihapus atau *drop***

```
EXECUTE SEM_APIS.DROP_SEM_NETWORK;
```

Gambar 2.12 Memanggil prosedur SEM_APIS.DROP_SEM_NETWORK

2.7.11 SEM_APIS.DROP_USER_INFERENCE_OBJS

Prosedur ini digunakan untuk menghapus semua rulebase atau entailment yang dimiliki user DB. Untuk memanggil prosedur ini, perlu memiliki hak akses yang mencukupi untuk menghapus aturan dan rulebase pada user DB yang ditentukan. Pada Tabel 2.11 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.13.

Tabel 2.11 Penjelasan Parameter Prosedur SEM_APIS.DROP_USER_INFERENCE_OBJS

Nama parameter	Tipe Data	Deskripsi
uname	**Varchar2**	**Nama user DB. Nama user DB ini yang diminta *case-sensitive*, Contoh** **user DB1 :herman** **user DB2 :Herman** **kedua user DB ini berbeda.**

```
EXECUTE SEM_APIS.DROP_USER_INFERENCE_OBJS('SCOTT');
```

Gambar 2.13 Memanggil prosedur SEM_APIS.DROP_USER_INFERENCE_OBJS

2.7.12 SEM_APIS.GET_TRIPLE_ID

Prosedur ini digunakan untuk mendapatkan nomor id dari suatu *Triple* yang ada pada model ontologi jika *Triple* itu ada. Namun jika *Triple* itu tidak ada dalam model ontologi tersebut maka prosedur ini akan mengembalikan nilai null. Prosedur ini memiliki dua format yang memiliki perbedaan yaitu terdapat pilihan untuk memasukkan nilai parameter pertama dengan nama model atau nomor id. Pada Tabel 2.12 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan untuk memanggil prosedur ini dapat dilihat pada Gambar 2.14.

Tabel 2.12 Penjelasan Parameter Prosedur SEM_APIS.GET_TRIPLE_ID

Nama parameter	Tipe Data	Deskripsi

Model_id Atau Model_name	**Varchar2 Atau number**	**Nomor id atau nama model ontologi. nilai dari parameter ini harus sama dengan kolom model_id atau model_name pada view MDSYS.SEM_MODEL$**
Subject	**Varchar2**	**Subjek dari *Triple* yang akan dicari. Nilai parameter ini harus sama dengan nilai pada kolom value_name pada tabel MDSYS.RDF_VALUE$**
Property	**Varchar2**	**Property dari *Triple* yang akan dicari. Nilai parameter ini harus sama dengan nilai pada kolom value_name pada tabel MDSYS.RDF_VALUE$**
Object	**Varchar2**	**Objek dari *Triple* yang akan dicari. Nilai parameter ini harus sama dengan nilai pada kolom value_name pada tabel MDSYS.RDF_VALUE$**

```
SELECT SEM_APIS.GET_TRIPLE_ID(
  'articles',
  'http://nature.example.com/Article2',
  'http://purl.org/dc/terms/references',
  'http://nature.example.com/Article3') AS RDF_triple_id FROM DUAL;
```

Gambar 2.14 Memanggil prosedur SEM_APIS.GET_TRIPLE_ID

2.7.13 SEM_APIS.GET_MODEL_NAME

Prosedur ini digunakan untuk mengembalikan nama model yang sesuai dengan nilai parameter model_id yang dimasukkan pada prosedur. Nilai dari parameter ini harus sesuai pada kolom

model_id pada view MDSYS.SEM_MODEL$. Pada Tabel 2.13 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.15 .

Tabel 2.13 Penjelasan Parameter Prosedur SEM_APIS.GET_MODEL_NAME

Nama parameter	Tipe Data	Deskripsi
model_id	**number**	**Nomor id model ontologi**

```
SELECT SEM_APIS.GET_MODEL_NAME(1) AS model_name FROM DUAL;
```

Gambar 2.15 Memanggil Prosedur SEM_APIS.GET_MODEL_NAME

2.7.14 SEM_APIS.GET_MODEL_ID

Prosedur ini digunakan untuk mendapatkan nomor id model dengan memasukkan nama model debagai nilai parameter. Prosedur ini merupakan kebalikan dari prosedur SEM_APIS.GET_MODEL_NAME. Pada Tabel 2.14 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.16.

Tabel 2.14 Penjelasan Parameter Prosedur SEM_APIS.GET_MODEL_ID

Nama parameter	Tipe Data	Deskripsi
model_name	**Varchar2**	**Nama model ontologi**

```
SELECT SEM_APIS.GET_MODEL_ID('articles') AS model_id FROM DUAL;
```

Gambar 2.16 Memanggil Prosedur SEM_APIS.GET_MODEL_ID

2.7.15 SEM_APIS.IS_TRIPLE

Prosedur ini digunakan untuk melakukan pengecekan apakah *Triple* yang dicari ada pada model ontologi. Prosedur ini hamper sama dengan prosedur SEM_APIS.GET_TRIPLE_ID namun prosedur ini mengembalikan nilai *True* jika *Triple* ada pada model ontologi, *False* jika *Triple* tidak ada. Pada Tabel 2.15 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.17.

Tabel 2.15 Penjelasan Parameter Prosedur SEM_APIS.IS_TRIPLE

Nama parameter	Tipe Data	Deskripsi
Model_name	**Varchar2**	**Nama model ontologi. nilai dari parameter ini harus sama dengan kolom model_id atau model_name pada view MDSYS.SEM_MODELS$**
Model_id	**number**	**Nomor id model ontologi. nilai dari parameter ini harus sama dengan kolom model_id atau model_name pada view MDSYS.SEM_MODELS$**
Subject	**Varchar2**	**Subjek dari *Triple* yang akan dicari. Nilai parameter ini harus sama dengan nilai pada**

		kolom value_name pada tabel MDSYS.RDF_VALUE$
Property	**Varchar2**	**Property dari *Triple* yang akan dicari. Nilai parameter ini harus sama dengan nilai pada kolom value_name pada tabel MDSYS.RDF_VALUE$**
Object	**Varchar2**	**Objek dari *Triple* yang akan dicari. Nilai parameter ini harus sama dengan nilai pada kolom value_name pada tabel MDSYS.RDF_VALUE$**

```
SELECT SEM_APIS.IS_TRIPLE(
  'articles',
  'http://nature.example.com/Article2',
  'http://purl.org/dc/terms/references',
  'http://nature.example.com/Article3') AS is_triple FROM DUAL;
```

Gambar 2.17 Memanggil Prosedur SEM_APIS.IS_TRIPLE

2.7.16 SEM_APIS.LOOKUP_ENTAILMENT

Prosedur ini digunakan untuk mengambil nama *entailment* berdasarkan model ontologi dan *rulebase* yang ada pada *entailment*. Untuk mendapatkan index *rulebase*, model ontologi dan *rulebase* harus disebutkan semuanya. Pada Tabel 2.16 akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.18.

Tabel 2.16 Penjelasan Parameter Prosedur SEM_APIS.LOOKUP_ENTAILMENT

Nama parameter	Tipe Data	Deskripsi
Models	**Sem_models**	**Satu atau lebih nama model. Tipe data SEM_MODEL**

		yang memiliki *Table Of Varchar2(25)*
Rulebases	**Sem_rulebases**	**Satu atau lebih nama *rulebase*. Tipe data SEM_RULEBASE yang memiliki *Table Of Varchar2(25)***

```
SELECT SEM_APIS.LOOKUP_ENTAILMENT(SEM_MODELS('family'),
  SEM_RULEBASES('RDFS','family_rb')) AS lookup_entailment FROM DUAL;
```

Gambar 2.18 Memanggil Prosedur SEM_APIS.LOOKUP_ENTAILMENT

2.7.17 SEM_APIS.MERGE_MODELS

Prosedur ini digunakan untuk memasukkan konten dari suatu model ke dalam model tujuan, dan melakukan *update* tabel tujuan. Prosedur ini Pada Tabel 2.17, akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.19 dan Gambar 2.20.

Tabel 2.17 Penjelasan Parameter Prosedur SEM_APIS.MERGE_MODEL

Nama parameter	Tipe Data	Deskripsi
source_model	**Varchar2**	**Nama model ontologi**
destination_model	**Varchar2**	**Nama model ontologi tujuan**
rebuild_apptab_index	**Boolean Default True**	**True: membangun ulang index pada model ontologi tujuan setelah model digabungkan. False: Tidak membangun index pada model tujuan.**

drop_source_model	**Boolean** **Default False**	**Jika nilai parameter *True*, maka model *source_model* untuk dihapus setelah dua model digabungkan.** **Jika nilai parameter *False*, Maka model *source_model* tidak dihapus**
options	**Varchar2** **Default: Null**	**String option yang akan melakukan overriding perilaku default dari prosedur.**

```
EXECUTE SEM_APIS.MERGE_MODELS('M1', 'M2');
```

Gambar 2.19 Memanggil Prosedur SEM_APIS.MERGE_MODEL dengan 2 parameter

```
EXECUTE SEM_APIS.MERGE_MODELS('M1', 'M2', null, null, 'DOP=4');
```

Gambar 2.20 Memanggil Prosedur SEM_APIS.MERGE_MODEL dengan 4 parameter

2.7.18 SEM_APIS.REMOVE_DUPLICATES

Prosedur ini digunakan untuk menghapuskan *Triple* yang ganda atau *Triple* yang berjumlah dua atau lebih dari sebuah model. Ketika *Triple* duplikat dihapus, semua informasi dalam baris yang terhapus tidak dapat diambil kembali, termasuk informasi dalam kolom selain kolom *Triple* dan jika model ontologi kosong atau tidak ada *Triple* yang terduplikasi, maka prosedur ini tidak akan menjalankan proses penghapusan. Pada Tabel 2.18, akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.21.

Tabel 2.18 Penjelasan Parameter Prosedur SEM_APIS.REMOVE_DUPLICATE

Nama parameter	Tipe Data	Deskripsi
model_name	**Varchar2**	**Nama model ontologi**
threshold	**float** **default 0.3**	**Sebuah nilai yang menentukan berapa banyak *Triple* duplikat yang ada untuk melakukan operasi penghapusan.** **rumus :** ***(total-triples - total-unique-triples + 0.01) / (total-unique-triples + 0.01)*** **Semakin tingginya threshold, semakin banyak *Triple* duplikat yang diperlukan.**
rebuild_apptab_index	**Boolean** **Default True**	**True: membangun ulang semua index yang dapat digunakan pada model ontologi tujuan setelah model digabungkan.** **False: Tidak membangun index apapun.**

```
EXECUTE SEM_APIS.REMOVE_DUPLICATES('family');
```

Gambar 2.21 Memanggil Prosedur SEM_APIS.REMOVE_DUPLICATE

2.7.19 SEM_APIS.RENAME_ENTAILMENT

Prosedur ini digunakan untuk mengganti nama *entailment* pada ontologi. Pada Tabel 2.19, akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.22.

Tabel 2.19 Penjelasan Parameter Prosedur SEM_APIS.RENAME_ENTAILMENT

Nama parameter	Tipe Data	Deskripsi
old_name	**Varchar2**	**Nama entailment yang lama**
New_name	**Varchar2**	**Nama entailment yang baru**

```
EXECUTE sem_apis.rename_entailment('owltst_idx', 'my_owltst_idx');
```

Gambar 2.22 Memanggil prosedur SEM_APIS.RENAME_ENTAILMENT

2.7.20 SEM_APIS.RENAME_MODEL

Prosedur ini digunakan untuk mengganti nama model ontologi yang ada pada basis data oracle, namun untuk memanggil prosedur SEM_APIS.RENAME_MODEL, harus memiliki hak akses pemilik model ontologi ini. Misal jika terdapat user SCOTT memiliki model ontologi *ONT*, maka hanya user SCOTT yang dapat memanggil prosedur ini untuk mengganti nama model ontologi *ONT*. Pada Tabel 2.20, akan dijelaskan parameter yang dipakai pada prosedur ini, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.23.

Tabel 2.20 Penjelasan Parameter Prosedur SEM_APIS.RENAME_MODEL

Nama parameter	Tipe Data	Deskripsi
old_name	**Varchar2**	**Nama model ontologi yang lama**
New_name	**Varchar2**	**Nama model ontologi yang baru**

```
EXECUTE sem_apis.rename_model('model1', 'model2');
```

Gambar 2.23 Memanggil Prosedur SEM_APIS.RENAME_MODEL

2.7.21 SEM_APIS.SWAP_NAMES

Prosedur ini digunakan untuk menukar nama model ontologi yang ada pada basis data oracle. Misalkan, terdapat 2 model ontologi pada suatu user, yaitu model1 dan model2. Model1 adalah model ontologi kesehatan sedangkan model2 merupakan nama model ontologi keluarga, sehingga jika prosedur ini dipanggil dengan mengisikan kedua nama model ini ke dalam parameter, maka nama model ontologi kesehatan akan menjadi model2 dan nama model ontologi keluarga menjadi model1. Untuk penjelasan parameter dapat dilihat pada Tabel 2.21, sedangkan perintah untuk memanggil prosedur ini dapat dilihat pada Gambar 2.24.

Tabel 2.21 Penjelasan Parameter Prosedur SEM_APIS.SWAP_NAMES

Nama parameter	Tipe Data	Deskripsi
Model1	**Varchar2**	**Nama model ontologi yang pertama**

Model2	**Varchar2**	**Nama model ontologi yang kedua**

```
EXECUTE sem_apis.swap_names('test', 'production');
```

Gambar 2.24 Memanggil Prosedur SEM_APIS.SWAP_NAMES

2.8 Penilitian Terkait

Penelitian yang terkait dengan Tugas Akhir ini adalah penelitian oleh Christian Candrabiantara dalam Tugas Akhirnya yang berjudul "Rancang Bangun Aplikasi Visualisasi Silsilah Keluarga Berbasis Ontologi". Dalam penelitian tersebut Christian Candrabiantara, yang selanjutnya akan disebut peneliti, mencoba membangun aplikasi yang digunakan untuk memvisualisasikan silsilah keluarga berbasis ontologi. Ontologi dapat digunakan untuk melakukan penalaran atau *inference*, sehingga peneliti tidak perlu memasukkan semua data silsilah keluarga, melainkan data silsilah keluarga yang dibutuhkan saja. Adapun posisi penelitian tersebut dalam Tugas Akhir ini adalah sebagai referensi dari penggunaan ontologi pada silsilah keluarga.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan proses perancangan sistem yang dibuat. Perancangan ini dibagi menjadi dua proses utama, yaitu:

1. Desain ontologi keluarga
2. Kasus penggunaan
3. Desain penyimpanan data basis data oracle
4. Perancangan Antarmuka Aplikasi.

3.1 Desain ontologi keluarga

Ontologi yang dipakai dalam Tugas Akhir ini adalah ontologi keluarga. Dengan menggunakan protégé, dapat diketahui jumlah aksioma, logical aksioma, properti objek, property data, Class, dan individual yang terdapat dalam suatu ontologi. Jumlah dari jenis-jenis aksiom yang telah disebutkan dapat dilihat pada Gambar 3.1.

Metrics

Axiom	2845
Logical axiom count	2186
Class count	58
Object property count	80
Data property count	9
Individual count	508
DL expressivity	SROIF(D)

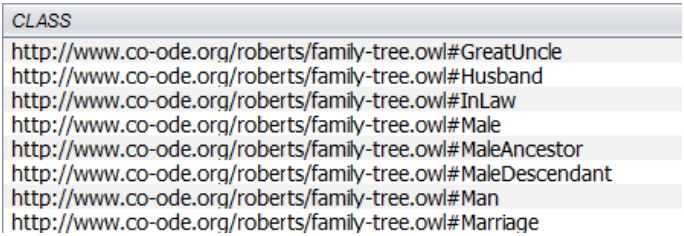
Gambar 3.1 Metrik Ontologi Keluarga

Berikut penjelasan dari setiap metric yang ada pada ontologi ini:

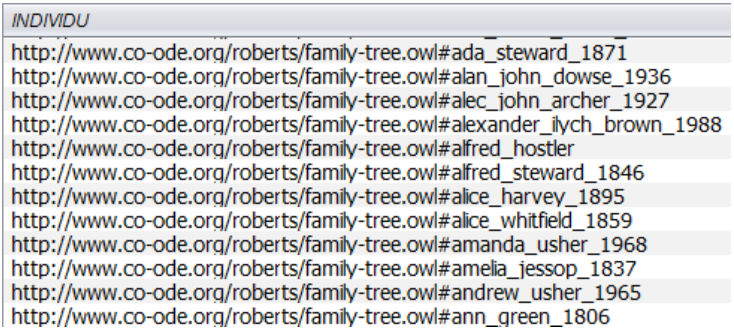
3.1.1 Kelas

Tujuan utama dari ontologi adalah melakukan klasifikasi hal-hal dalam bentuk semantik atau makna. Dalam OWL, hal ini dapat dicapai dengan penggunaan kelas dan subkelas, yang merupakan *instance* dimana di OWL disebut individual. Individual merupakan anggota dari suatu Kelas dapat disebut sebagai ekstensi dari suatu kelas.

Sebuah kelas dalam OWL merupakan sebuah klasifikasi dari individual ke dalam kelompok yang memiliki kesamaan karakteristik. Jika sebuah individu merupakan member dari sebuah kelas, maka hal ini seperti memberitahu sistem bahwa individu tersebut termasuk dalam sebuah klasifikasi semantik yang ada pada kelas OWL. Contoh Class dalam Ontologi keluarga dapat dilihat pada Gambar 3.2, Sedangkan Contoh Individual dapat dilihat pada Gambar 3.3.



Gambar 3.2 Contoh Kelas pada ontologi keluarga



Gambar 3.3 Contoh Individual pada ontologi keluarga

Selain kelas dan individu, terdapat subclass dan disjoin class. Subclass merupakan kelas turunan dari kelas yang lain. Misalnya kelas *male* merupakan subclass. Disjoin merupakan relasi dua kelas yang tidak berhubungan, misalkan kelas *male* dan kelas *female* memiliki hubungan disjoin, sehingga instance atau individu yang ada pada salah satu kelas tidak dapat termasuk pada kelas yang lainnya. Contoh subclass pada ontologi keluarga ini ditunjukan pada Gambar 3.4, sedangkan contoh disjoin class ditunjukan pada Gambar 3.5.

SUPERCLASS	SUBCLASS
http://www.co-ode.org/roberts/family-tree.owl#Female	http://www.co-ode.org/roberts/family-tree.owl#Sex
http://www.co-ode.org/roberts/family-tree.owl#Male	http://www.co-ode.org/roberts/family-tree.owl#Sex
http://www.co-ode.org/roberts/family-tree.owl#Marriage	_:m6mORABNENC4P0H11H25H28HC45E60e5c89d58E151e955942558E45E7c0e
http://www.co-ode.org/roberts/family-tree.owl#Person	_:m6mORABNENC4P0H11H25H28HC45E60e5c89d58E151e955942558E45E7c08
http://www.co-ode.org/roberts/family-tree.owl#Sex	http://www.co-ode.org/roberts/family-tree.owl#DomainEntity

Gambar 3.4 Contoh Subclass pada ontologi keluarga

CLASS1	CLASS2
http://www.co-ode.org/roberts/family-tree.owl#Female	http://www.co-ode.org/roberts/family-tree.owl#Male
http://www.co-ode.org/roberts/family-tree.owl#Marriage	http://www.co-ode.org/roberts/family-tree.owl#Person
http://www.co-ode.org/roberts/family-tree.owl#Marriage	http://www.co-ode.org/roberts/family-tree.owl#Sex
http://www.co-ode.org/roberts/family-tree.owl#Person	http://www.co-ode.org/roberts/family-tree.owl#Sex

Gambar 3.5 Contoh Disjoin class pada ontologi keluarga

3.1.2 Properti

Sebuah properti memiliki relasi biner. Dua jenis properti dapat dibedakan yaitu, properti object dan properti tipe data. Perbedaan dari dua properti ini dilihat dari relasi yang dibuat oleh properti. Properti tipe data merelasikan antara instance dari sebuah kelas dan sebuah data “mati”/literal seperti tanggal kelahiran, nama panggilan dan lain lain. Sedangkan Properti object merelasikan kedua instance dari kelas. Contoh properti tipedata ditunjukan pada Gambar 3.6 dan Gambar 3.7 sedangkan contoh properti objek ditunjukan pada Gambar 3.8.

Properti object juga dapat membatasi suatu relasi dengan domain dan range. Sebuah domain dari suatu properti membatasi individual dimana suatu properti dapat digunakan. Jika sebuah properti merelasikan sebuah individual dengan individual yang lain

dan properti tersebut memiliki suatu kelas (*Class*) sebagai domainnya, maka individual tersebut atau subjek harus termasuk dalam kelas tersebut. Sedangkan range dari sebuah properti membatasi individual dimana individual tersebut merupakan objek dari relasi tersebut. Jika properti tersebut merelasikan sebuah satu individu dengan individu yang lain dan properti tersebut memiliki sebuah kelas sebagai rangenya, maka individual lainnya atau objek properti tersebut harus termasuk dalam kelas tersebut. Contohnya, properti *hasFather* memiliki domain yaitu *Person*, sedangkan rangenya yaitu *Man*, Sehingga individu subjek harus termasuk dalam kelas person dan objek individu harus termasuk dalam kelas *Man.*Contoh properti objek, domain dan range ditunjukan pada Gambar 3.9.

Pada property juga terdapat subproperti, dimana sebuah properti merupakan turunan dari sebuah properti yang lain. Contohnya, properti *hasWife* merupakan turunan dari properti *isSpouseOf*. Dalam Oracle semantic, semua *statement* akan berupa *Triple* dimana terdapat subjek, predikat, dan objek suatu statemen. Untuk memasukkan *Triple* yang mengandung subPropertyOf, subjek *Triple* berisi properti turunan, predikat *Triple* berisi subPropertyOf dan objek *Triple* merupakan properti atasannya. Contoh dapat ditunjukan pada Gambar 3.10

TIPEDATA
http://www.co-ode.org/roberts/family-tree.owl#alsoKnownAs
http://www.co-ode.org/roberts/family-tree.owl#formerlyKnownAs
http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear
http://www.co-ode.org/roberts/family-tree.owl#hasDeathYear
http://www.co-ode.org/roberts/family-tree.owl#hasFamilyName
http://www.co-ode.org/roberts/family-tree.owl#hasFirstGivenName
http://www.co-ode.org/roberts/family-tree.owl#hasMarriageYear
http://www.co-ode.org/roberts/family-tree.owl#hasName
http://www.co-ode.org/roberts/family-tree.owl#knownAs

Gambar 3.6 Contoh properti tipedata pada ontologi keluarga

SUB	TIPEDATA	OBJ
http://www.co-ode.org/roberts/family-tree.owl#caroline_cox	http://www.co-ode.org/roberts/family-tree.owl#alsoKnownAs	Catherine
http://www.co-ode.org/roberts/family-tree.owl#caroline_lavinia_t...	http://www.co-ode.org/roberts/family-tree.owl#alsoKnownAs	Maria
http://www.co-ode.org/roberts/family-tree.owl#elizabeth_blancha...	http://www.co-ode.org/roberts/family-tree.owl#formerlyKnownAs	elizabeth hutchinson
http://www.co-ode.org/roberts/family-tree.owl#william_george_b...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1901
http://www.co-ode.org/roberts/family-tree.owl#iris_ellen_archer_...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1906
http://www.co-ode.org/roberts/family-tree.owl#martha_cotton_...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1806

Gambar 3.7 Contoh *Triple* dengan peroperti tipedata

PROPERTI
http://www.co-ode.org/roberts/family-tree.owl#hasAncestor
http://www.co-ode.org/roberts/family-tree.owl#hasAunt
http://www.co-ode.org/roberts/family-tree.owl#hasAuntInLaw
http://www.co-ode.org/roberts/family-tree.owl#hasBrother
http://www.co-ode.org/roberts/family-tree.owl#hasBrotherInLaw
http://www.co-ode.org/roberts/family-tree.owl#hasChild
http://www.co-ode.org/roberts/family-tree.owl#hasDaughter
http://www.co-ode.org/roberts/family-tree.owl#hasFather
http://www.co-ode.org/roberts/family-tree.owl#hasFatherInLaw
http://www.co-ode.org/roberts/family-tree.owl#hasFemalePartner
http://www.co-ode.org/roberts/family-tree.owl#hasForeFather

Gambar 3.8 Contoh properti objek pada ontologi keluarga

PROPERTI	DOMAIN	RANGE
http://www.co-ode.org/roberts/family-tree.owl#hasAncestor	http://www.co-ode.org/roberts/family-tree.owl#Person	http://www.co-ode.org/roberts/family-tree.owl#Person
http://www.co-ode.org/roberts/family-tree.owl#hasAunt	http://www.co-ode.org/roberts/family-tree.owl#Person	http://www.co-ode.org/roberts/family-tree.owl#Woman
http://www.co-ode.org/roberts/family-tree.owl#hasBrother	http://www.co-ode.org/roberts/family-tree.owl#Person	http://www.co-ode.org/roberts/family-tree.owl#Man
http://www.co-ode.org/roberts/family-tree.owl#hasChild	http://www.co-ode.org/roberts/family-tree.owl#Person	http://www.co-ode.org/roberts/family-tree.owl#Person
http://www.co-ode.org/roberts/family-tree.owl#hasDaughter	http://www.co-ode.org/roberts/family-tree.owl#Person	http://www.co-ode.org/roberts/family-tree.owl#Woman

Gambar 3.9 Contoh properti objek, domain, dan range

PROPERTI1	PROPERTI2
http://www.co-ode.org/roberts/family-tree.owl#isForefatherOf	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf
http://www.co-ode.org/roberts/family-tree.owl#isBrotherOf	http://www.co-ode.org/roberts/family-tree.owl#isDirectSiblingOf
http://www.co-ode.org/roberts/family-tree.owl#hasAunt	http://www.co-ode.org/roberts/family-tree.owl#isBloodRelationOf
http://www.co-ode.org/roberts/family-tree.owl#hasFatherInLaw	http://www.co-ode.org/roberts/family-tree.owl#hasParentInLaw
http://www.co-ode.org/roberts/family-tree.owl#hasMother	http://www.co-ode.org/roberts/family-tree.owl#hasForeMother
http://www.co-ode.org/roberts/family-tree.owl#hasForeMother	http://www.co-ode.org/roberts/family-tree.owl#hasAncestor
http://www.co-ode.org/roberts/family-tree.owl#hasMother	http://www.co-ode.org/roberts/family-tree.owl#hasParent
http://www.co-ode.org/roberts/family-tree.owl#isMalePartnerIn	http://www.co-ode.org/roberts/family-tree.owl#isPartnerIn
http://www.co-ode.org/roberts/family-tree.owl#hasFamilyName	http://www.co-ode.org/roberts/family-tree.owl#hasName
http://www.co-ode.org/roberts/family-tree.owl#isUncleInLawOf	http://www.co-ode.org/roberts/family-tree.owl#isInLawOf
http://www.co-ode.org/roberts/family-tree.owl#hasWife	http://www.co-ode.org/roberts/family-tree.owl#isSpouseOf
http://www.co-ode.org/roberts/family-tree.owl#hasHusband	http://www.co-ode.org/roberts/family-tree.owl#isSpouseOf
http://www.co-ode.org/roberts/family-tree.owl#alsoKnownAs	http://www.co-ode.org/roberts/family-tree.owl#knownAs
http://www.co-ode.org/roberts/family-tree.owl#isSecondCous...	http://www.co-ode.org/roberts/family-tree.owl#isBloodRelationOf
http://www.co-ode.org/roberts/family-tree.owl#hasGreatGran...	http://www.co-ode.org/roberts/family-tree.owl#hasGreatGrandParer
http://www.co-ode.org/roberts/family-tree.owl#isNephewOf	http://www.co-ode.org/roberts/family-tree.owl#isBloodRelationOf

Gambar 3.10 Contoh relasi subproperti dari 2 properti

3.1.3 Karakteristik Properti

Berikut karakteristik properti yang terdapat pada ontologi keluarga:

3.1.3.1 Karakteristik Properti Transitif

Suatu properti dapat memiliki karakteristik transitif. Dan jika properti tersebut transitif, maka jika terdapat *Triple* dengan relasi yang transitif, contoh, A *hasAncestor* B, dan terdapat *Triple* lain, misalnya B *hasAncestor* C, dapat disimpulkan bahwa A *hasAncestor* C. Contoh properti objek yang memiliki karakteristik transitif dapat dilihat pada Gambar 3.11.

PROPERTI_YG_TRANSITIVE
http://www.co-ode.org/roberts/family-tree.owl#isForefatherOf
http://www.co-ode.org/roberts/family-tree.owl#hasAncestor
http://www.co-ode.org/roberts/family-tree.owl#isRelationOf
http://www.co-ode.org/roberts/family-tree.owl#hasForeFather
http://www.co-ode.org/roberts/family-tree.owl#isForemotherOf

Gambar 3.11 Contoh properti objek yang transitif

3.1.3.2 Karakteristik Properti simetrik

Properti yang dapat disebut sebagai properti yang simetrik, jika properti tersebut dapat memiliki relasi dua arah, misalnya Andy hasSiblings Chelsea, maka dapat disimpulkan bahwa Chelsea hasSiblings Andy. Perlu diingat bahwa domain dan range suatu properti tetap diperhatikan. Pada contoh diatas, domain dan range properti hasSiblings adalah Person, sehingga Andy dan Chelsea termasuk dalam domain maupun range. Contoh properti objek yang memiliki karkteristik simetrik dapat dilihat pada Gambar 3.12.

PROPERTI_YG_SIMETRIK
http://www.co-ode.org/roberts/family-tree.owl#isInLawOf
http://www.co-ode.org/roberts/family-tree.owl#isSpouseOf
http://www.co-ode.org/roberts/family-tree.owl#isFirstCousinOnceRemovedOf
http://www.co-ode.org/roberts/family-tree.owl#isSecondCousinOf
http://www.co-ode.org/roberts/family-tree.owl#isRelationOf
http://www.co-ode.org/roberts/family-tree.owl#isThirdCousinOf
http://www.co-ode.org/roberts/family-tree.owl#isDirectSiblingOf
http://www.co-ode.org/roberts/family-tree.owl#isBloodRelationOf
http://www.co-ode.org/roberts/family-tree.owl#isSiblingOf
http://www.co-ode.org/roberts/family-tree.owl#isSiblingInLawOf
http://www.co-ode.org/roberts/family-tree.owl#isFirstCousinOf

Gambar 3.12 Contoh properti objek yang simetrik

3.1.3.3 Karakteristik Properti Fungsional

Suatu properti dapat memiliki karakteristik fungsional jika subjek tersebut memiliki hanya 1 relasi tersebut dengan suatu objek, misalnya jika Andy *hasMother* Lea, maka Andy tidak mungkin memiliki relasi hasMother dengan individu lain. Contoh properti objek yang memiliki karakteristik fungsional dapat dilihat pada Gambar 3.13

PROPERTI_YG_FUNGSIONAL
http://www.co-ode.org/roberts/family-tree.owl#hasMarriageYear
http://www.co-ode.org/roberts/family-tree.owl#hasSex
http://www.co-ode.org/roberts/family-tree.owl#hasDeathYear
http://www.co-ode.org/roberts/family-tree.owl#hasFather
http://www.co-ode.org/roberts/family-tree.owl#hasFemalePartner
http://www.co-ode.org/roberts/family-tree.owl#hasFamilyName
http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear
http://www.co-ode.org/roberts/family-tree.owl#hasMother
http://www.co-ode.org/roberts/family-tree.owl#hasMalePartner

Gambar 3.13 Contoh properti objek yang fungsional

3.1.3.4 Karakteristik Properti *inverseOf*

Jika sebuah properti memiliki relasi *inverseOf* dengan properti lain, maka jika suatu properti memiliki subjek *X* dan objek *Y*, maka properti yang lain memiliki subjek *Y* dan objek *X*. Misal jika properti *hasWife* memiliki relasi *inverseOf* dengan properti *isWifeOf*, maka tripe Andy *hasWife* Chelsea memiliki arti yang sama dengan *Triple* Chelsea *isWifeOf* Andy. Contoh relasi *inverseOf* dari dua properti objek yang terdapat pada ontologi keluarga dapat dilihat pada Gambar 3.14.

PROPERTIA	PROPERTIB
http://www.co-ode.org/roberts/family-tree.owl#hasGreatAunt	http://www.co-ode.org/roberts/family-tree.owl#isGreatAuntOf
http://www.co-ode.org/roberts/family-tree.owl#hasAncestor	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf
http://www.co-ode.org/roberts/family-tree.owl#hasGrandfather	http://www.co-ode.org/roberts/family-tree.owl#isGrandfatherOf
http://www.co-ode.org/roberts/family-tree.owl#hasGrandParent	http://www.co-ode.org/roberts/family-tree.owl#isGrandParentOf
http://www.co-ode.org/roberts/family-tree.owl#hasGreatGrandParent	http://www.co-ode.org/roberts/family-tree.owl#isGreatGrandParentOf
http://www.co-ode.org/roberts/family-tree.owl#isParentInLawOf	http://www.co-ode.org/roberts/family-tree.owl#hasParentInLaw
http://www.co-ode.org/roberts/family-tree.owl#hasChild	http://www.co-ode.org/roberts/family-tree.owl#isChildOf
http://www.co-ode.org/roberts/family-tree.owl#hasWife	http://www.co-ode.org/roberts/family-tree.owl#isWifeOf
http://www.co-ode.org/roberts/family-tree.owl#hasHusband	http://www.co-ode.org/roberts/family-tree.owl#isHusbandOf
http://www.co-ode.org/roberts/family-tree.owl#hasGrandmother	http://www.co-ode.org/roberts/family-tree.owl#isGrandmotherOf
http://www.co-ode.org/roberts/family-tree.owl#hasMotherInLaw	http://www.co-ode.org/roberts/family-tree.owl#isMotherInLawOf

Gambar 3.14 Contoh relasi inverseOf dari dua properti objek

3.2 Desain penyimpanan data dalam basis data oracle

Pada bagian ini akan dibahas desain penyimpanan data dalam basis data Oracle. Dalam basis data Oracle, data semantik memiliki sebuah struktur data yang secara efektif dimodelkan menggunakan graf yang berarah. Adapun desain yang dibahas dalam pengerjaan Tugas Akhir ini adalah desain data, dan desain rule.

3.2.1 Desain data

Dalam Oracle Semantic, Semua *Triple* akan *diparsing* dan disimpan dalam basis data sebagai entri dalam tabel yang berada di dalam skema MDSYS. Sebuah *Triple*, yang di dalamnya mengandung subjek, predikat, dan objek, disimpan sebagai sebuah objek dalam basis data. Sebagai hasilnya, sebuah dokumen memiliki beberapa *Triple* dalam beberapa objek basis data.

Semua subjek dan objek dari *Triple* dipetakan ke dalam node dalam sebuah jaringan data semantik, dan properti dipetakan ke jaringan *link* yang memiliki node awal dan node akhir, masing-masing sebagai subjek dan objek. Beberapa kemungkinan tipe dari node yaitu, URI dan data literal. Berikut persyaratan yang berlaku untuk spesifikasi URI dan penyimpanan data semantik dalam basis data:

- Sebuah subjek harus merupakan URI.
- Sebuah properti harus merupakan URI.
- Sebuah objek dapat berupa tipe apapun, seperti URI dan literal (null tidak didukung dalma Oracle Semantic).

Basis data Oracle memelihara beberapa tabel dan *view* dalam Skema MDSYS untuk menyimpan metadata yang berelasi dengan data semantik. Berikut daftar tabel dan *view* yang terdapat pada Skema MDSYS ditunjukan pada Tabel 3.1.

Tabel 3.1 Daftar tabel dan view dalam skema MDSYS

Nama	Informasi yang terdapat pada tabel / *view*
RDF_VALUE$	**Subjek,properti(predikat), dan objek yang digunakan untuk merepresentasikan *statement*(*Triple*)**
SEM_MODEL$	**Semua model yang didefinisikan dalam basis data**
SEMM_*model-name*	***Triple* yang tersimpan dalam suatu model**
SEM_RULEBASE_INFO	**Rulebase yang tersimpan dalam basis data**
SEM_RULES_INDEX _DATASET	**Objek basis data yang digunakan untuk *entailment***
SEM_RULES_INDEX _INFO	**Rule index atau *entailment***

SEMI_*entailment-name*	***Triple* dalam suatu *entailment***
SEMR_*rulebase-name*	***Rules* atau aturan-aturan yang terdapat pada suatu *rulebase***

3.2.1.1 Model

Setiap ontologi yang ada akan disimpan dalam suatu model ontologi yang telah didefinisikan pada basis data Oracle. Pada view *MDSYS.SEM_MODEL$* memuat segala informasi yang berhubungan dengan model yang telah didefinisikan dalam basis data. Ketika sebuah model dibuat menggunakan *procedure SEM_APIS.CREATE_SEM_MODEL*, parameter yang diminta yaitu nama model ontologi, serta nama tabel dan kolom yang mengandung refrensi kepada data semantik.

Sebelum model ontologi dibuat dengan memanggil *procedure SEM_APIS.CREATE_SEM_MODEL*, perlu membuat tabel yang digunakan untuk menyimpan refrensi data semantik. Sedangkan parameter nama kolom adalah nama kolom yang memiliki tipe SDO_RDF_TRIPLE_S. *procedure* ini akan menambahkan model pada *MDSYS.SEM_MODEL$*.

Sedangkan untuk menghapus model menggunakan *procedure* SEM_APIS.DROP_SEM_MODEL, dengan parameter nama model. Harus diingat bahwa dalam membuat model ontologi tidak boleh menambahkan ke dalam view *MDSYS.SEM_MODEL$* melalui *SQL Insert, update, delete* dikarenakan pembuatan model ini menyangkut refrensi ke view dan tabel lainnya pada skema MDSYS. Contoh tabel yang menyimpan refrensi dan penggunaan *procedure* ditunjukan pada Kode Sumber 3.1, sedangkan penjelasan view MDSYS.SEM_MODEL$ dapa dilihat pada Tabel 3.2

```
--Contoh Create Tabel
CREATE TABLE articles_rdf_data  -- nama tabel
(id NUMBER,
triple SDO_RDF_TRIPLE_S);       -- nama kolom
```

```
-- Panggil Procedure untuk membuat model
EXECUTE SEM_APIS.CREATE_SEM_MODEL(
'articles',              --nama model
'articles_rdf_data',     --nama tabel
'triple');               --nama kolom bertipe

-- Panggil Procedure untuk menghapus model
EXECUTE SEM_APIS.DROP_SEM_MODEL('articles');
```

Kode Sumber 3. 1 *Query* membuat tabel dan model

Tabel 3.2 Kolom *view* MDSYS.SEM_MODEL$

Nama Kolom	Tipe Data	Deskripsi
Owner	Varchar2(30)	Skema dari user DB
Model_Id	Number	Nomor id unik model ontologi yang secara otomatis dibuat oleh oracle.
Model_name	Varchar2(25)	Nama model
Table_name	Varchar2(30)	Nama tabel yang dapat memuat refrensi data semantik untuk model
Coloumn Name	Varchar2(30)	Nama kolom yang memiliki tipe SDO_RDF_TRIPLE_S Dalam tabel yang dapat memuat refrensi data semantik pada model
Model_tablespace_name	Varchar2(30)	Nama tablespace yang digunakan untuk menyimpan *Triple* pada model

Ketika sebuah model dibuat, sebuah view untuk *Triple* yang terkait dengan model, juga dibuat dalam skema MDSYS. Nama view ini dalam format SEMM_*model-name* dan hanya dapat dilihat oleh user DB atau pemilik model dan user lain yang

memiliki hak akses atas view ini. Penjelasan setiap kolom pada view MDSYS.SEMM_*nama-model* ditunjukan pada Tabel 3.3.

Tabel 3.3 Kolom MDSYS.SEMM_*nama-model*

Nama Kolom	Tipe Data	Deskripsi
P_value_id	**Number**	**Value_id untuk nilai text dari predikat pada *Triple*. Termasuk Primary Key**
Start_node_id	**Number**	**Value_id untuk nilai text dari subjek dari suatu *Triple*. Termasuk Primary Key**
Canon_end_node_id	**Number**	**Value_id untuk nilai text dari bentuk kanonikal dari objek dari suatu *Triple*. Termasuk Primary Key**
End_node_id	**Number**	**Value_id untuk nilai teks dari suatu objek pada *Triple***
Model_id	**Number**	**Id untuk graf RDF dimana *Triple* disimpan.**
Link_id	**Varchar2(71)**	**Nilai *identifier Triple* unik**

3.2.1.2 Statement

Dalam basis data Oracle, penyimpanan data *Triple* suatu ontologi dapat dibagi menjadi menjadi dua, yaitu menyimpan pada tabel yang dibuat pada saat sebelum memanggil prosedur SEM_APIS.CREATE_SEM_MODEL. Tabel ini biasanya memiliki dua kolom, yaitu kolom id dan kolom *Triple*. Tipe data masing-masing kolom yaitu number dan SDO_RDF_TRIPLE_S. Tabel ini digunakan untuk menyimpan objek *Triple* dengan tipe

data SDO_TRIPLE_S. Kolom *Triple* ini digunakan untuk menyimpan referensi data semantik sebenarnya.

SDO_RDF_TRIPLE merupakan suatu tipe data objek yang merepresentasikan data semantik atau ontologi dalam format *triple* sehingga tipe data ini digunakan untuk menampilkan data *triple*, sedangkan SDO_RDF_TRIPLE_S merupakan tipe data objek yang digunakan untuk menyimpan data semantik pada basis data Oracle. Tipe SDO_RDF_TRIPLE_S memiliki referensi kepada data, karena data semantik atau ontologi yang sebenarnya disimpan pada skema RDF utama, dalam hal ini skema MDSYS. Tipe data ini memiliki beberapa *method* untuk mengambil data *triple* atau sebagian data *triple*. Atribut-atribut yang terdapat pada tipe data objek SDO_RDF_TRIPLE dan SDO_RDF_TRIPLE_S dapat ditunjukan pada Gambar 3.15 dan Gambar 3.16, Sedangkan *method* yang dapat dipakai untuk mengambil nilai tripel ditunjukan pada Gambar 3.17.

```
SDO_RDF_TRIPLE (
  subject VARCHAR2(4000),
  property VARCHAR2(4000),
  object VARCHAR2(10000))
```

Gambar 3.15 Atribut SDO_RDF_TRIPLE

```
SDO_RDF_TRIPLE_S (
  RDF_C_ID  NUMBER,  -- Canonical object value ID
  SEM_M_ID NUMBER,  -- Model ID
  RDF_S_ID  NUMBER,  -- Subject value ID
  RDF_P_ID NUMBER, -- Property value ID
  RDF_O_ID  NUMBER)  -- Object value ID
```

Gambar 3.16 Atribut SDO_RDF_TRIPLE_S

```
GET_TRIPLE() RETURNS SDO_RDF_TRIPLE
GET_SUBJECT() RETURNS VARCHAR2
GET_PROPERTY() RETURNS VARCHAR2
GET_OBJECT() RETURNS CLOB
```

Gambar 3.17 *Method* dari tipe data objek SDO_RDF_TRIPLE

Untuk memasukkan data semantik yang berupa *triple* ke dalam sebuah tabel model. Tabel model ini merupakan tabel yang memiliki kolom dengan tipe data objek SDO_RDF_TRIPLE_S. Terdapat dua jenis konstruktor SDO_RDF_TRIPLE_S yang ditunjukan pada Gambar 3.18 dimana perbedaan kedua konstruktor ini adalah objek dari konstruktor kedua memiliki tipe data CLOB

```
SDO_RDF_TRIPLE_S (
  model_name VARCHAR2, -- Model name
  subject    VARCHAR2, -- Subject
  property   VARCHAR2, -- Property
  object     VARCHAR2) -- Object
  RETURN     SELF;

SDO_RDF_TRIPLE_S (
  model_name VARCHAR2, -- Model name
  subject    VARCHAR2, -- Subject
  property   VARCHAR2, -- Property
  object     CLOB) -- Object
  RETURN SELF;
```

Gambar 3.18 Konstruktor untuk melakukan INSERT *Triple*

Tabel MDSYS.RDF_VALUE$ menampung informasi tentang subjek, properti, dan objek yang dipakai untuk merepresentasikan RDF statement. Tabel ini menyimpan teks (URI maupun data literal) secara unik untuk tiga buah informasi, menggunakan baris yang terpisah untuk setiap masing masing bagian dari *Triple*.

Basis data Oracle melakukan pengelolaan tabel MDSYS.RDF_VALUE$ secara otomatis, sehingga user tidak boleh mengubah data dari tabel ini secara langsung. Misalnya, perintah SQL Insert, Update, Delete. Berikut kolom yang terdapat pada tabel MDSYS.RDF_VALUE$ ditunjukan pada Tabel 3.4.

Tabel 3.4 Kolom tabel MDSYS.RDF_VALUE$

Nama Kolom	Tipe Data	Deskripsi
Value_id	**NUMBER**	**Nomor id unik yang secara otomatis**
Value _type	**Varchar(10)**	**Tipe teks informasi yang yang tersimpan dalam kolom Value_name. Value yang memungkinkan yaitu:** **1. Ur untuk URI** **2. PL untuk data literal** **3. BN untuk blank node** **4. PL@ untuk data literal dengan language tag** **5. TL untuk *typed* literal** **6. TLL untuk typed long literal (sebuah long literal memiliki 4000 byte atau lebih)**
Vname _prefix	**Varchar2(512)**	**Jika panjang nilai leksikal tidak lebih dari 4000 byte, maka kolom ini akan menyimpan prefix dari nilai lexical. Contoh prefix untuk nilai leksikal dari <http://www.w3.org/2002/07/owl#Restriction> adalah http://www.w3.org/2002/07/owl#**
Vname _suffix	**Varchar2(512)**	**Jika panjang nilai leksikal adalah 4000 byte atau kurang, maka kolom ini dapat menyimpan *suffix* dari sebagian nilai leksikal. Jika menggunakan nilai leksikal yang disebutkan pada deskripsi VNAME_PREFIX, *suffix* nilai leksikal tersebut adalah *Restriction***

Literal _type	**Varchar2(4000)**	**Untuk setiap data literal yang bertipe dengan value_type *TL* seperti *date*, integer, string dan lain lain. Contoh:** **http://www.w3.org/2001/XMLSchema#date**
Language_type	**Varchar2(80)**	**Kolom ini digunakan untuk menyimpan sebuah literal dengan tag *language* (seperti *fr* untuk perancis). Literal yang disimpan memiliki value_type PL@. jika tidak maka kolom ini berisi *null***
Canon_id	**Number**	**No. Id untuk nilai leksikal yang kanon**
Collision_ext	**Varchar2(64)**	**Digunakan untuk mengatasi collision untuk nilai leksikal**
Canon _collision _ext	**Varchar2(64)**	**Kolom ini digunakan untuk mengatasi nilai leksikal yang kanon**
Long_value	**Clob**	**Kolom ini akan terisi jika panjang dari nilai leksikal melebihi 4000 byte, jika tidak, maka kolom ini memiliki nilai *null***
Value _name	**Varchar2(4000)**	**Kolom ini terkomputasi secara otomatis. Jika panjang nilai leksikal kurang dari atau 4000 byte, maka nilai dari kolom ini adalah rangkaian dari VNAME_PREFIX dan VNAME_SUFFIX.**

Pada basis data Oracle, untuk melakukan pencarian berbasis semantik, diharuskan menggunakan SEM_MATCH. SEM_MATCH adalah fungsi tabel yang memiliki atribut-atribut seperti yang ditunjukan pada Gambar 3.19. Pada fungsi tabel SEM_MATCH, atribut *query* wajib diisi sedangkan atribut lain dapat diisi null. Atribut *query* merupakan sebuah *String* dengan satu atau lebih pola *triple*, dimana biasanya mengandung variabel.

Sebuah pola *triple* adalah sebuah *triple* dari sebuah atribut yang bernama atom. Setiap atom dapat berupa variabel (seperti ?x), nama *qualified* seperti rdf:type yang dapat dipanjangkan berdasarkan *namespace*nya dan nilai dari atribut *alias*, atau sebuah teks URI yang lengkap seperti <*http://www.w3.org/1999/02/22-rdf-syntax-ns#type*>. Sebagai tambahan, atom ketiga dapat berupa numerikal seperti 3.14, sebuah literal seperti herman, atau literal bertipe seperti *"123"^^xsd:int*. Fungsi tabel ini mengembalikan sebuah objek tipe *ANYDATASET* dengan elemen yang bergantung pada variabel *input*.

Atribut model digunakan untuk mengidentifikasikan model ontologi atau model ontologi yang akan digunakan. Tipe data atribut ini adalah SEM_MODELS, dimana memiliki definisi lain yaitu *TABLE OF VARCHAR2(25)*.

Atribut *rulebase* digunakan untuk mengidentifikasi satu atau lebih *rulebase* dimana aturan atau *rule* dapat diaplikasikan pada *query*. Tipe data atribut ini adalah SEM_RULEBASES dimana memiliki definisi lain yaitu *TABLE OF VARCHAR2(25)*.

Atribut *aliases* digunakan untuk mengidentifikasi satu atau lebih *namespace* dengan tambahan namespace dasar seperti rdfs, rdfs, yang digunakan untuk ekspansi dari nama *qualified* dalam atribut *query*. Tipe data atribut ini adalah SEM_ALIASES dimana memiliki definisi lain yaitu, *TABLE OF SEM_ALIAS*. Setiap elemen SEM_ALIAS mengidentifikasikan sebuah ID dari namespace dan nilai namespace. Tipe data SEM_ALIAS memiliki definisi yaitu : namespace_id dengan tipe data VARCHAR2(30) dan namespace_val dengan tipe data VARCHAR2(4000).

Atribut *Filter* digunakan untuk mengidentifikasikan kriteria seleksi tambahan yang ada. Jika atribut ini tidak berisi null, maka harus berisi sebuah *String* dalam bentuk klausa *where* tanpa menggunakan kata kunci *where*. Misalnya (height => 6), untuk membatasi hasil *query* yaitu hanya orang dengan tinggi badan lebih dari 6.

Atribut *index_status* digunakan agar jika *entailment* yang bersangkutan tidak memiliki status valid, *query* tetap dapat

dijalankan. Jika status entailment bukan valid dan atribut ini berisi null, maka *query* tidak dapat dijalankan atau mengembalikan *error*. Jika atribut ini berisi *INCOMPLETE* atau *INVALID*, *query* dapat dijalankan namun hasil *query* dapat mengalami penurunan akurasi.

Atribut Option mengidentifikasikan opsi yang dapat mempengaruhi hasil *query*. Nilai atribut opsi diekspresikan dengan beberapa kata kunci seperti:

- ALLOW_DUP=T
  Opsi ini menghasilkan hasil *query* dimana memperbolehkan adanya duplikasi dari baris-baris hasil *query*.
- HINT0={<hint-string>}
  Opsi ini menentukan satu atau lebih kata kunci dengan *hint* untuk mempengaruhi hasil *query*.
- INF_ONLY=T
  Opsi ini digunakan untuk menampilkan graf hasil inference pada model dan *rulebase* yang ditentukan.

```
SEM_MATCH(
  query         VARCHAR2,
  models        SEM_MODELS,
  rulebases     SEM_RULEBASES,
  aliases       SEM_ALIASES,
  filter        VARCHAR2,
  index_status  VARCHAR2,
  options       VARCHAR2
 ) RETURN ANYDATASET;
```

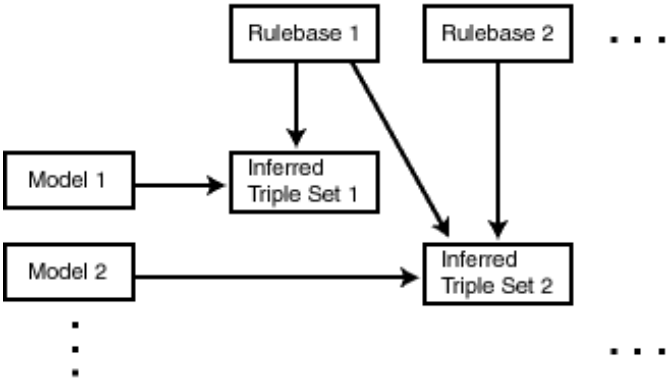
Gambar 3.19 Atrribut Fungsi Tabel SEM_MATCH

3.2.2 Desain *Rule*

Inference merupakan suatu kemampuan untuk membuat deduksi logis berdasarkan aturan (*rule*). Inference memungkinkan pengguna untuk membangun kueri yang dapat melakukan pencocokan semantik berdasarkan relasi antara data dengan data

yang lain. Inference melibatkan pemakaian aturan, baik yang disediakan Oracle maupun aturan yang dibuat sendiri, yang ditempatkan dalam *rulebases*.

Pada Gambar 3.20 menunjukkan sebuah set *Triple* sedang diinference dari model data dan aturan yang tersimpan dalam rulebase. Dalam ilustrasi ini, basis data Oracle dapat memiliki beberapa model semantik, rulebase, dan set *Triple* yang telah disimpulkan. Sebuah set *Triple* dapat disimpulkan menggunakan satu atau lebih rulebase.



Gambar 3.20 Ilustrasi Inferencing [7]

```
EXECUTE SEM_APIS.CREATE_RULEBASE('family_rb');

INSERT INTO mdsys.semr_family_rb VALUES(
  'grandparent_rule',
  '(?x :parentOf ?y) (?y :parentOf ?z)',
  NULL,
  '(?x :grandParentOf ?z)',
  SEM_ALIASES(SEM_ALIAS('','http://www.example.org/family/')));
```

Gambar 3.21 Contoh membuat rulebase dan rule [7]

Sebuah rule atau aturan merupakan objek yang dapat dipakai untuk menarik kesimpulan dari *inference* data semantik. Sebuah aturan diidentifikasi dengan sebuah nama dan terdiri dari:

- Sebuah Pola *If* untuk *antencedent.*
- Sebuah kondisi filter yang opsional yang dapat dipakai untuk membatasi subgraph yang cocok dengan *antecedent.*
- Sebuah Pola *Then* untuk *consequent.*

Sebuah *rulebase* merupakan objek yang berisi *rule*. Berikut *rulebase* yang disediakan oleh Oracle yaitu:

- RDFS.
- RDF (subset dari RDFS).
- OWLSIF.
- RDFS++.
- OWLPRIME.
- SKOSCORE.

Untuk membuat rulebase sesuai yang diinginkan user, user dapat menggunakan *procedure* SEM_APIS.CREATE_RULEBASE dan masukkan aturan yang diinginkan ke dalam tabel MDSYS.semr_<*nama-rulebase*> dengan perintah insert sepeti yang ditunjukan pada Gambar 3.21 Contoh membuat rulebase.

Pada Gambar contoh insert *rule*, dapat dilihat bahwa aturan ini dapat membuat hasil inference dari data semantik keluarga menjadi suatu *statement* atau *Triple* data semantic yang baru. Untuk ekspresi atau sintak yang dipakai pada *antecedent* dan *consequent* adalah sintak SPARQL. Pada aturan '*grandfather_rule*' terdapat antecedent dan consequent yang digunakan untuk mencocokan subgraf yang ada pada data semantik dengan kondisi *if* aturan ini. Jika data semantic ini memenuhi syarat *antecedent*, maka data semantik tersebut dapat digunakan pada consequent. Berikut kondisi if dan then dari aturan *'grandfather_rule'*:

$Kondisi\ If$: $(?x : parentOf\ ?y)\ (?y : parentOf\ ?z)$
$Kondisi\ Then$: $(?x : grandParentOf\ ?z)$

Jika ada dua *Triple* data semantik yaitu, (Ali, parentOf, Lisa) dan (Lisa, parentOf, Nadia), maka melalui inference dari rule ini,

terdapat *Triple* baru yaitu (Ali, grandparentOf, Nadia). Untuk penjelasan kolom-kolom pada tabel MDSYS.semr_<*nama_rulebase*> ditunjukan pada Tabel 3.5.

Tabel 3.5 Kolom tabel MDSYS.SEMR_<*nama-rulebase*>

Nama Kolom	Tipe Data	Deskripsi
Rule_name	**Varchar2(30)**	**Nama rule / aturan**
Antecedents	**Varchar2(4000)**	**Pola *If* untuk sebuah antecedent. Merupakan kondisi dimana aturan / rule dapat dicapai melalui antecedent ini.**
Filter	**Varchar2(4000)**	**Kondisi yang dipakai untuk membatasi subgraf yang cocok dengan antecedent. Dapat bernilai null**
Consequent	**Varchar2(4000)**	**Pola *Then* untuk sebuah *consequent.* Merupakan hasil akhir dari hasil perbandingan graf dengan antecedent pada aturan ini**
Aliases	**SEM_ALIASES**	**Satu atau lebih *namespace* yang dipakai. Digunakan untuk menyingkat prefix dari URI, misal *http://www.family.com/hasSon* dapat disingkat menjadi *fam:hasSon***

Pada basis data Oracle, semua informasi yang berkaitan dengan *rulebase* disimpan dan diolah dalam *view* MDSYS.SEM_RULEBASE_INFO. Daftar kolom-kolom pada *view* MDSYS.SEM_RULEBASE_INFO ditunjukan pada Tabel 3.6.

Tabel 3.6 Kolom *view* MDSYS.SEM_RULEBASE_INFO

Nama Kolom	Tipe Data	Deskripsi
Owner	**Varchar2(30)**	**Pemilik(user) dari suatu *rulebase***
Rulebase_Name	**Varchar2(25)**	**Nama *rulebase***
Rulebase_view_name	**Varchar2(30)**	**Nama dari *view* yang digunakan untuk sql statement untuk memodifikasi *rulebase***
Status	**Varchar2(30)**	**Dapat berisi *VALID* jika rulebase valid.** **Dapat berisi *INPROGRESS,* jika *rulebase* sedang dibuat.** **Dapat berisi *FAILED,* jika *rulebase* sistem mengalami kegagalan ketika *rulebase* dibuat**

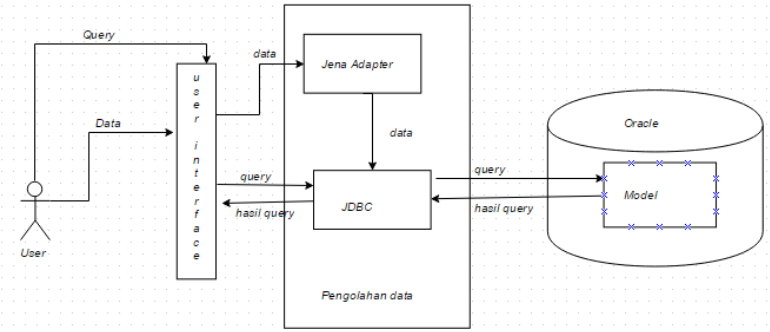
Sebuah index aturan (rule index) atau yang disebut *entailment* merupakan sebuah objek yang mengandung *Triple* yang belum terkomputasi yang dapat disimpulkan dengan mengaplikasikan sebuah set *rulebase* tertentu pada model ontologi tertentu. Jika sebuah *query* SEM_MATCH mengacu pada suatu rulebase, sebuah entailment harus ada atau *exist* untuk setiap kombinasi rulebase-model dalam *query*.

Untuk membuat sebuah *entailment*, harus menggunakan *procedure* SEM_APIS.CREATE_ENTAILMENT dan untuk menghapus (*drop*) sebuah *entailment* menggunakan *procedure* SEM_APIS.DROP_ENTAILMENT. Ketika sebuah entailment dibuat, sebuah *view* untuk *Triple* yang terkait dengan entailment, akan dibuat dalam skema MDSYS. View ini memiliki nama dalam format SEMI_*nama-entailment*, dan view ini hanya dapat dilihat

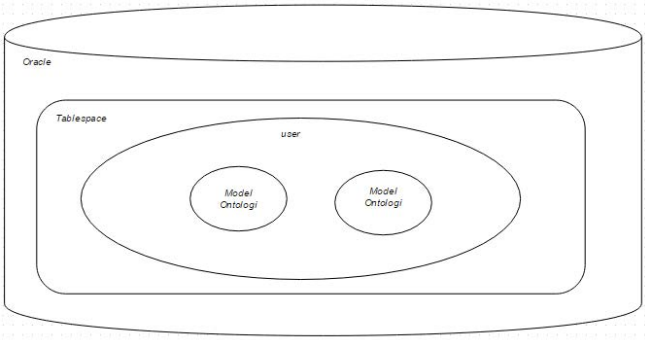
oleh user DB dari entailment dan user DB yang memiliki hak akses. Setiap view SEMI_*nama-entailment* memuat satu baris untuk *Triple* (disimpan sebagai link) dan memiliki kolom yang sama dengan *view* dari SEMM_*nama-model* seperti yang ditunjukkan pada Tabel 3.3.

3.3 Arsitektur Aplikasi

Aplikasi ini dirancang sebagai alat bantu dalam menyimpan data ontologi ke dalam basis data Oracle, dimana pada pembangunan aplikasi diperlukan adanya hal lain sebagai suatu penghubung yang dapat menghubungkan aplikasi dengan basis data Oracle. Pada Gambar 3.22 ditunjukkan melalui aplikasi, pengguna aplikasi dapat memasukkan data ataupun melakukan *query* ke dalam basis data Oracle. Jika aplikasi menerima masukkan data ontologi, berupa *triple* atau file ontologi, maka data akan diproses dahulu melalui Aena Adapter for Oracle Database. Setelah melalui jena adapter, data akan masuk ke dalam model ontologi pada basis data Oracle melalui JDBC sebagai penghubung antara aplikasi dan basis data Oracle. Sedangkan untuk masukkan *query* pada aplikasi, akan langsung melalui JDBC dan akan melakukan *query* pada basis data Oracle. Hasil *query* akan dikembalikan dalam bentuk *vector* dari data tabular.



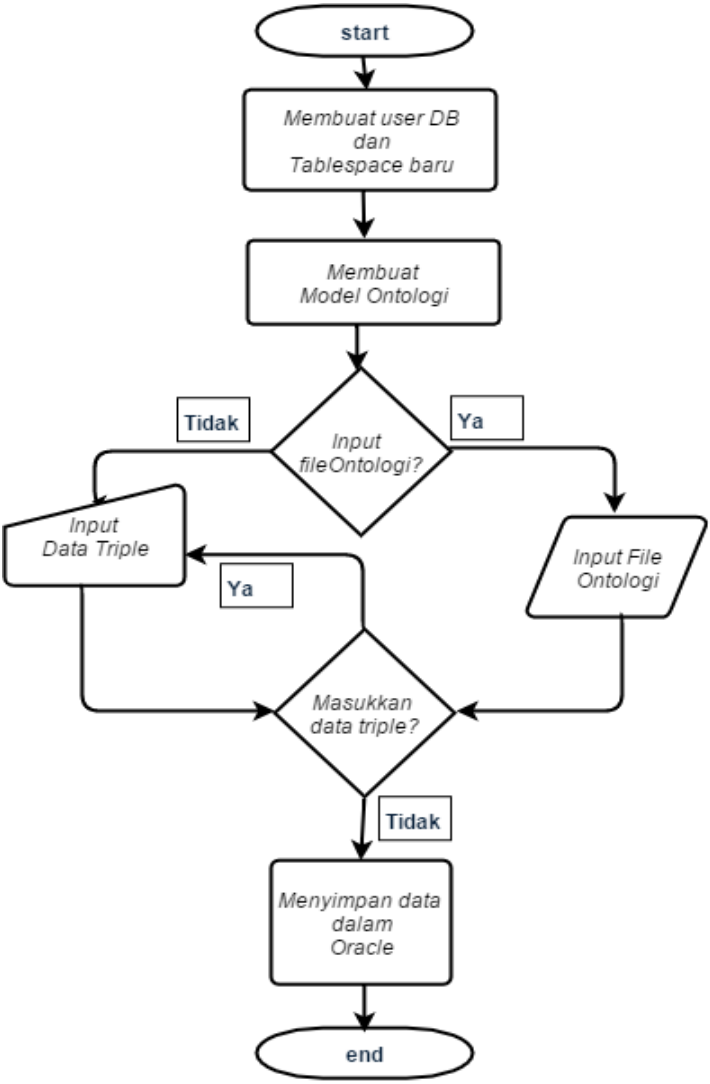
Gambar 3.22 Arsitektur aplikasi



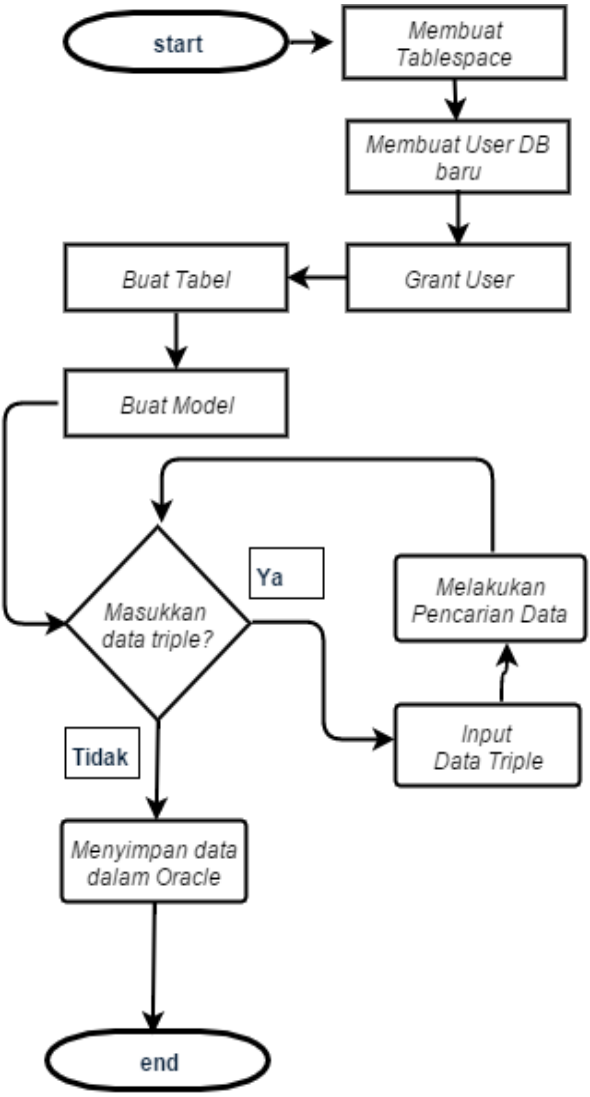
Gambar 3.23 Ilustrasi Hubungan antara Basis Data Oracle, Tablespace, User DB dan Model Ontologi.

Dalam basis data Oracle, terdapat beberapa istilah tempat penyimpanan. Basis data Oracle memiliki user yang memiliki hak akses ke dalam suatu tablespace yang dimilikinya. Satu user memiliki beberapa tablespace yang terdiri dari beberapa model ontologi, sehingga sangat wajar jika hanya user tersebut yang dapat melakukan beberapa subprogram atau prosedur seperti yang telah dijelaskan pada subbab 2.7. Contoh ilustrasi penyimpanan basis data Oracle dapat ditunjukkan pada Gambar 3.23. Perlu diketahui bahwa untuk menggunakan aplikasi, diperlukannya fitur semantik yang sudah diaktifkan terlebih dahulu pada basis data Oracle.

Pada Gambar 3.24 dan Gambar 3.25 dapat ditunjukkan terdapat perbedaan proses yang perlu dilakukan ketika menggunakan aplikasi untuk menggunakan fitur-fitur yang ada pada Oracle Semantic. Perbedaan proses yang dimaksudkan adalah terdapat kemudahan untuk melakukan proses pembuatan user, tablespace, tabel dan model ontologi. Pada aplikasi, terdapat fitur memasukkan file langsung ke dalam basis data dan juga dapat memasukkan data ontologi secara satu per satu.



Gambar 3.24 Diagram Alur Penggunaan Aplikasi



Gambar 3.25 Diagram Alur Penggunaan Oracle Semantic

3.4 Kasus Penggunaan Sistem

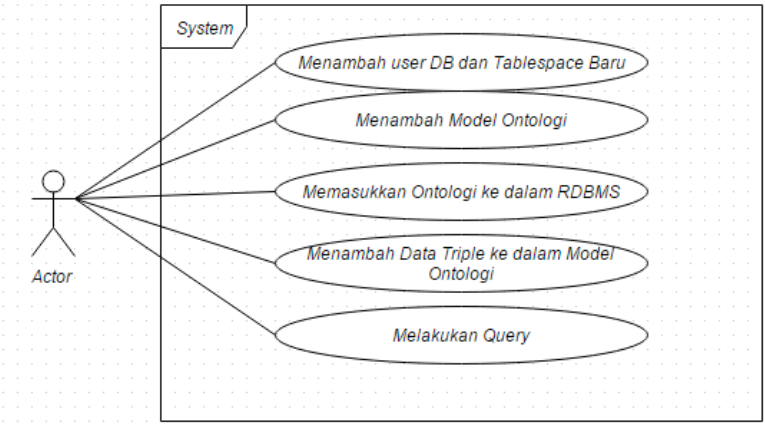
Bagian ini akan menjelaskan tahap perancangan kasus-kasus penggunaan yang terdapat pada sistem dengan mengacu pada kebutuhan-kebutuhan fungsional yang sudah direncanakan. Setiap kasus penggunaan yang direncanakan akan disertai dengan spesifikasi kasus penggunaan, diagram sekuens serta diagram aktivitasnya.

3.4.1 Aktor

Definisi dari aktor dalam kasus penggunaan adalah pihak-pihak yang terlibat dan berinteraksi dengan sistem. Dalam perancangan kasus penggunaan Tugas Akhir ini, yaitu pengguna Aplikasi.

3.4.2 Diagram Kasus Penggunaan

Berikut ini adalah diagram kasus penggunaan yang dirancang untuk memenuhi spesifikasi kebutuhan fungsional yang sudah dirancang pada Gambar 3.26. Berdasarkan diagram kasus penggunaan yang ditunjukkan pada Gambar 3.26, terdapat lima buah kasus penggunaan merupakan bentuk generalisasi kasus penggunaan.



Gambar 3.26 Kasus Penggunaan Aplikasi

3.4.2.1 Kasus Penggunaan Menambah User DB dan Tablespace baru

Dalam kasus penggunaan ini, aktor hanya perlu memasukkan data-data yang diminta untuk menambah user baru dan *tablespace* baru, yaitu *url JDBC*, password user "sys", nama pengguna, *password* pengguna, dan nama tablespace. Rincian dari kasus penggunaan ini dapat ditunjukkan pada Tabel 3.7 dan dan alur dari kasus penggunaan ini dapat digambarkan dengan diagram aktivitas pada Gambar 3.27

Tabel 3.7 Spesifikasi Kasus Penggunaan Menambah User DB dan Tablespace Baru

Komponen	Deskripsi
Nama	Menambah User DB dan Tablespace baru
Nomor	UC-001
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor menambah nama user DB baru dan tablespace baru pada basis data Oracle
Tipe	Fungsional
Aktor	User
Kondisi Awal	User DB dan Tablespace belum dibuat
Kondisi Akhir	User DB dan Tablespace telah dibuat
Alur Normal	1. Aktor memasukkan data yang diminta seperti password sys, Alamat jdbc, nama user, password user dan nama tablespace baru 2. Aktor menambah user DB dan tablespace baru. 3. User DB dan tablespace baru dibuat oleh Aplikasi
Alur Alternatif	3.1. User DB atau tablespace telah dibuat sebelumnya. 3.1.1. Perangkat lunak mengeluarkan pesan gagal 3.1.2. Aktor dapat mengisikan nama user DB atau tablespace yang baru.

3.4.2.2 Kasus Penggunaan Menambah Model ontologi baru

Dalam Kasus penggunaan ini pengguna dapat menambahkan model ontologi ke dalam basis data Oracle. Model ontologi akan disimpan ke dalam tablespace pengguna, sehingga pengguna lain tidak dapat mengakses model ontologi tersebut. Rincian dari kasus penggunaan ini dapat ditunjukkan pada Tabel 3.8 dan alur dari kasus penggunaan ini dapat digambarkan dengan diagram aktivitas pada Gambar 3.28.

Tabel 3.8 Spesifikasi Kasus Penggunaan Menambah Model Ontologi Baru

Komponen	Deskripsi
Nama	Menambah model ontologi baru
Nomor	UC-002
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor menambah model ontologi baru dengan user DB yang diinginkan
Tipe	Fungsional
Aktor	User
Kondisi Awal	Model ontologi belum dibuat
Kondisi Akhir	Model ontologi telah dibuat
Alur Normal	1. Aktor memilih memasukkan data yang diminta 2. Aktor menekan tombol buat model 3. Aplikasi menambahkan model ontologi baru pada basis data.
Alur Alternatif	3.1. Nama model ontologi yang akan dibuat sudah ada sebelumnya dalam basis data 3.1.1. Perangkat lunak mengeluarkan pesan gagal 3.1.2. Aktor dapat mengisikan nama ontologi yang lain

3.4.2.3 Kasus Penggunaan Memasukkan Ontologi ke dalam RDBMS

Dalam kasus penggunaan ini pengguna dapat memasukkan data ontologi yang telah dibuat oleh pengguna ke dalam RDBMS Oracle. Data yang dimasukkan dapat berupa N-TRIPLE, RDF, dan lain lain. Rincian dari kasus penggunaan ini dapat ditunjukkan pada Tabel 3.9 dan alur dari kasus penggunaan ini dapat digambarkan dengan diagram aktivitas pada Gambar 3.29.

Tabel 3.9 Spesifikasi Kasus Penggunaan Memasukkan Ontologi ke dalam RDBMS

Komponen	Deskripsi
Nama	Memasukkan ontologi ke dalam RDBMS
Nomor	UC-003
Deskripsi	Kasus penggunaan ini merupakan kasus penggunaan dimana aktor memasukkan data ontologi dalam bentuk file ke dalam Aplikasi sehingga data ontologi tersebut dapat disimpan dalam basis data Oracle
Tipe	Fungsional
Aktor	User
Kondisi Awal	Data ontologi belum ada
Kondisi Akhir	Data ontologi telah tersimpan
Alur Normal	1. Aktor memasukkan data yang diminta 2. Aktor menekan tombol pilih file 3. Aktor memilih file ontologi yang akan disimpan pada basis data ontologi 4. Aktor menekan tombol simpan ontologi 5. Sistem menyimpan data ontologi yang tersimpan pada file ontologi 6. Data ontologi tersimpan
Alur Alternatif	

3.4.2.4 Kasus Penggunaan Menambah data *Triple* ke dalam Model Ontologi

Dalam kasus penggunaan ini pengguna dapat menambahkan triple ke dalam model ontologi pada database. Dengan membaca model ontologi yang tersimpan pada basis data Oracle, sistem akan memuat semua properti objek yang terdapat pada model ontologi tersebut, sehingga pengguna dapat memasukkan informasi baru berupa data *Triple* atau *statement* ke dalam model ontologi pada basis data. Rincian dari kasus penggunaan ini dapat ditunjukkan pada Tabel 3.10 dan alur dari kasus penggunaan ini dapat digambarkan dengan diagram aktivitas pada Gambar 3.30.

Tabel 3.10 Spesifikasi Kasus Penggunaan menambah *Triple* ke dalam model ontologi

Komponen	Deskripsi
Nama	Menambah data *Triple* ke dalam Model Ontologi
Nomor	UC-004
Deskripsi	Kasus penggunaan ini merupakan kasus penggunaan dimana aktor memasukkan data user dan model ontologi, dan memasukkan *Triple* baru ke dalam basis data.
Tipe	Fungsional
Aktor	User
Kondisi Awal	*Triple* belum dimasukkan
Kondisi Akhir	Triple telah dimasukkan
Alur Normal	1. Aktor memasukkan data yang diminta 2. Aktor menekan tombol *Load model* 3. Aplikasi memuat semua properti objek yang ada pada model ontologi. 4. Aktor mengisikan data subjek dan objek dari *Triple*. 5. Aktor memilih properti objek pada *combobox*

	6. Aktor menekan tombol *insert Triple* 7. *Triple* baru akan tersimpan dalam model ontologi yang diinginkan
Alur Alternatif	

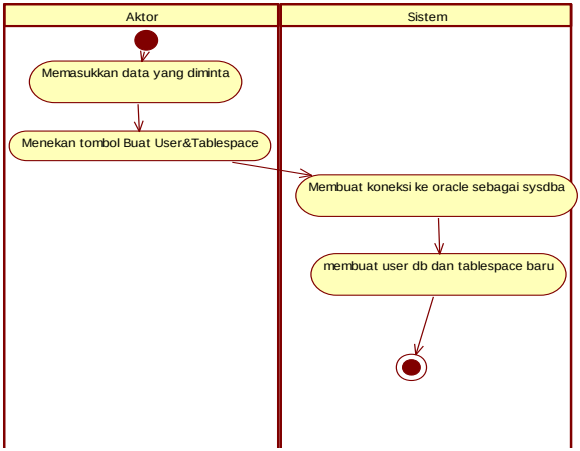
3.4.2.5 Kasus Penggunaan Melakukan *query*

Dalam kasus penggunaan ini, pengguna dapat melakukan *query select* pada ontologi dalam basis data Oracle tanpa sepenuhnya menggunakan *syntax* SPARQL. Dengan menginputkan *syntax* pencarian dari Oracle Semantic maka akan muncul table yang menunjukan hasil pencarian dari Oracle semantik. Rincian dari kasus penggunaan ini dapat ditunjukkan pada Tabel 3.11 dan alur dari kasus penggunaan ini dapat digambarkan dengan diagram aktivitas pada Gambar 3.31

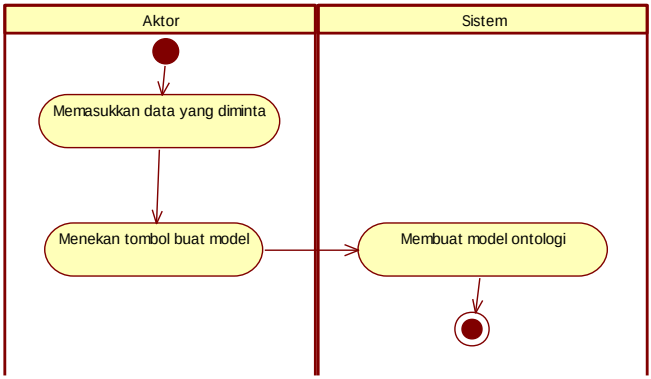
Tabel 3.11 Spesifikasi Kasus Penggunaan Melakukan *Query*

Komponen	Deskripsi
Nama	Melakukan *Query*
Nomor	UC-005
Deskripsi	Kasus penggunaan ini merupakan kasus penggunaan dimana aktor melakukan *query* pencarian data semantik sesuai dengan sintak *query* yang ditentukan
Tipe	Fungsional
Aktor	User
Kondisi Awal	Hasil *query* masih kosong
Kondisi Akhir	Hasil *query* ditampilkan
Alur Normal	1. Aktor memasukkan *query* pada *textfield* 2. Aktor memasukkan user dan password Oracle sesuai dengan nama model pada *query* 3. Aktor menekan tombol *Execute* 4. Aplikasi melakukan *query select* pada Oracle

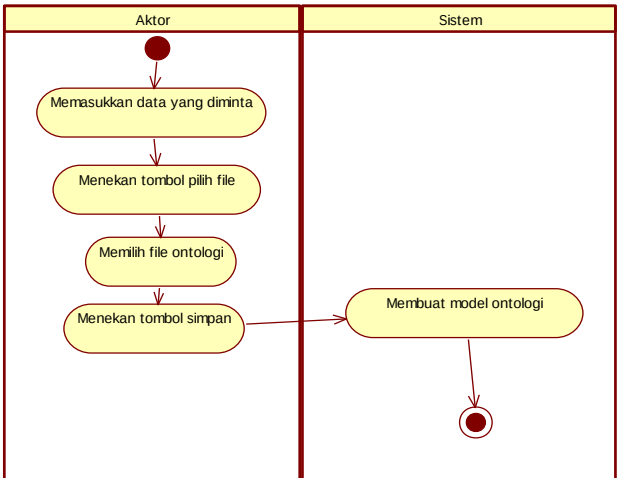
	5. Aplikasi menampilkan hasil *query* select
Alur Alternatif	



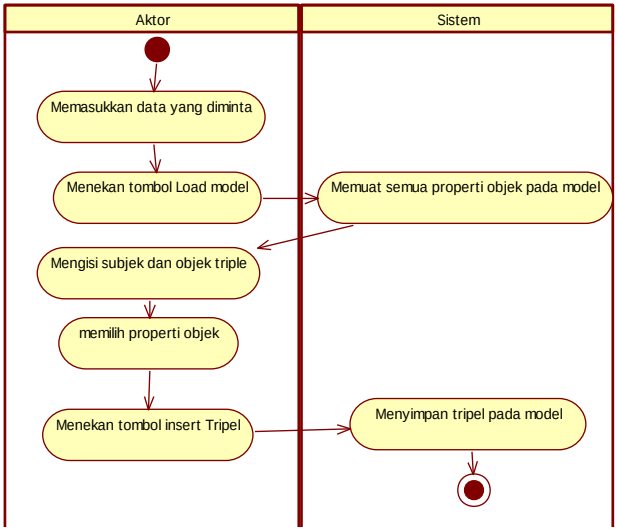
Gambar 3.27 Diagram Aktivitas membuat user DB dan tablespace baru



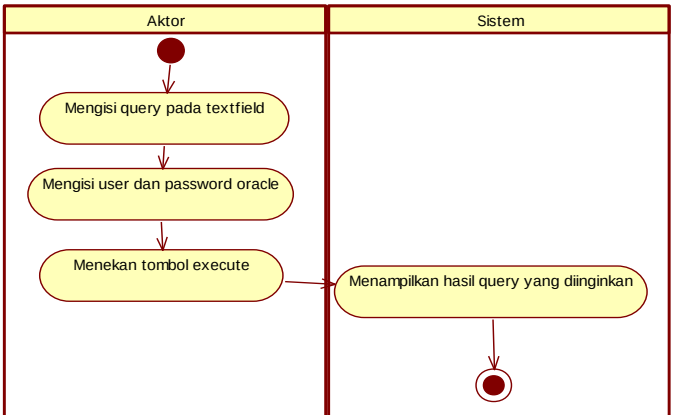
Gambar 3.28 Diagram Aktivitas Menambah Model Ontologi Baru



Gambar 3.29 Diagram Aktivitas Memasukkan Ontologi ke dalam RDBMS



Gambar 3.30 Diagram Aktivitas Menambahkan *Triple* ke dalam Basis data



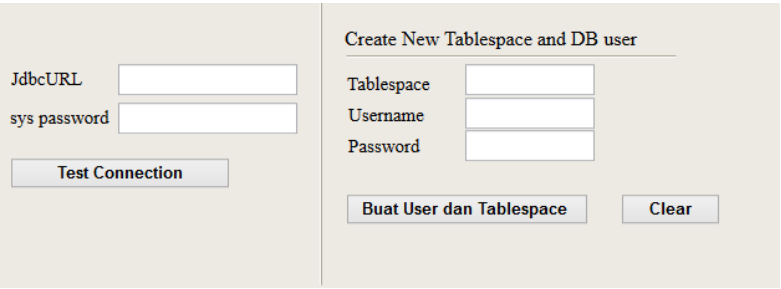
Gambar 3.31 Diagram Aktivitas Melakukan *Query*

3.5 Perancangan Antarmuka Pengguna

Pada bagian ini akan dibahas mengenai rancangan antarmuka bagi pengguna untuk memenuhi kasus penggunaan yang sudah dirancang.

3.5.1 Halaman Menambah User DB dan Tablespace baru

Halaman ini akan digunakan oleh pengguna untuk menambahkan User DB dan tablespace baru ke dalam Oracle. Pada halaman ini pengguna memasukkan data yang diminta seperti url JDBC, password sys, nama pengguna baru, password pengguna baru dan nama tablespace yang baru. Setelah data dimasukkan Aplikasi dapat membuat pengguna DB dan tablespace baru untuk digunakan dalam membuat model ontologi. Rancangan halaman antarmuka bisa dilihat pada Gambar 3.32 dan penjelasan mengenai elemen-elemen yang terdapat pada halaman ini dapat dilihat pada Tabel 3.12.



Gambar 3.32 Rancangan Antarmuka Halaman UserDB dan Tablespace baru

Tabel 3.12 Spesifikasi Atribut Antarmuka Halaman UserDB dan Tablespace baru

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/ Ke-luaran
1	*szJdbcText*	*TextField*	Memasukkan Url jdbc untuk membuka koneksi ke dalam basis data Oracle	*String*
2	*szSYSpassText*	*TextField*	Mengisi password user sys sehingga dapat masuk sebaga *sysdba*	*String*
3	*CrDbUserButton*	*Button*	Membuat Tablespace dan user DB baru pada basis data Oracle	*Event Clicked*
4	*crtblspcText*	*TextField*	Mengisi nama Tablespace baru	*String*
5	*crUsrText*	*TextField*	Mengisi nama User DB baru	*String*

6	*crPassText*	*TextFiel d*	Mengisi password untuk user DB baru	*String*
7	*testconButton*	*Button*	Melakukan pengecekan koneksi ke basis data Oracle apakah url jdbc dan password sys benar	*Event Clicked*
8	*Jlabel11*	*Label*	Menampilkan pesan gagal atau berhasil	*String*
9	*clearText*	*Button*	Menghapus isi textfield tablespace, user dan password baru	*Event Clicked*

3.5.2 Halaman Menambah Model Ontologi Baru

Halaman ini akan digunakan oleh pengguna untuk Menambah model baru yang akan disimpan ke dalam basis data Oracle. Data yang diminta yaitu url JDBC, nama user, dan password user. Setelah itu Aplikasi dapat menambahkan model dan tabel ke dalam Oracle. Rancangan halaman antarmuka bisa dilihat pada Gambar 3.33 dan penjelasan mengenai elemen-elemen yang terdapat pada halaman ini dapat dilihat pada Tabel 3.13.

Tabel 3.13 Spesifikasi Atribut Antarmuka Halaman Buat Model

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan /Ke-luaran
1	*szJdbcText1*	*TextField*	Mengisi URL jdbc	*String*

2	*testconButton*	*Button*	Melakukan cek koneksi aplikasi dengan basis data Oracle	*Event Clicked*
3	*usrText*	*TextField*	Mengisi nama user DB	*String*
4	*passText*	*TextField*	Mengisi password user DB	*String*
5	*modelText*	*TextField*	Mengisi nama model ontologi yang akan dibuat	*String*
6	*crModelButton*	*Button*	Membuat model ontologi baru pada user DB yang diinginkan	*Event Clicked*

Gambar 3.33 Rancangan Antarmuka Halaman Menambah Model Ontologi Baru

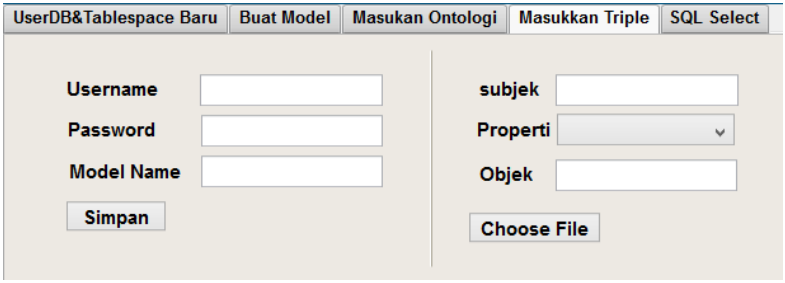
3.5.3 Halaman Menambah Data *Triple* ke dalam model ontologi

Halaman ini digunakan oleh pengguna untuk melakukan insert *Triple* ke dalam model ontologi yang tersimpan dalam basis data Oracle. Data yang diperlukan adalah *query* select, username dan password. Rancangan antarmuka aplikasi dapat dilihat pada Gambar 3.34 dan penjelasan mengenai elemen-elemen yang terdapat pada halaman ini dapat dilihat pada Tabel 3.14

Tabel 3.14 Spesifikasi Atribut Antarmuka Halaman Masukkan Triple

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Ke-luaran
1	*modelText1*	*TextField*	Mengisikan nama model ontologi yang akan dimuat	*String*
2	*usrText1*	*TextField*	Mengisi nama user db yang akan dipakai	*String*
3	*passText1*	*TextField*	Mengisi password user db	*String*
4	*insSbjText*	*TextField*	Mengisi subjek dari *Triple* yang akan disimpan dalam basis data	*String*
5	*pdtBox*	*Combo Box*	Menampilkan list properti objek yang terdapat pada model ontologi yang telah dimuat	*Vector*
6	*insObjText*	*TextField*	Mengisi objek dari *Triple* yang akan ditambahkan pada basis data	*String*
7	*loadButton*	*Button*	Memuat semua data pada model	*Event Clicked*

			ontologi yang diinginkan ke aplikasi	
8	*insertTripleButton*	*Button*	Menyimpan subjek, predikat, dan objek yang telah diisikan ke dalam model ontologi pada basis data oracle	*Event Clicked*



Gambar 3.34 Rancangan Antarmuka Halaman Memasukkan *Triple*

3.5.4 Halaman Memasukkan Ontologi

Halaman ini digunakan oleh pengguna untuk memasukkan data ontologi dalam bentuk file misalnya, RDF, ke dalam model ontologi yang tersimpan pada basis data Oracle. Rancangan antarmuka Aplikasi dapat dilihat pada Gambar 3.35 dan penjelasan mengenai elemen-elemen yang terdapat pada halaman ini dapat dilihat pada Tabel 3.15.

Tabel 3.15 Spesifikasi Atribut Antarmuka Halaman Memasukkan Ontologi

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Ke-luaran
1	*szJdbcText2*	*TextField*	Mengisi URL jdbc	*String*
2	*usrText3*	*TextField*	Mengisi nama user DB	*String*
3	*passText3*	*TextField*	Mengisi password user DB	*String*
4	*FilepathText1*	*TextField*	Mengisi *path* dari file ontology	*String*
5	*chooseOntButton1*	*Button*	Memilih file ontologi yang akan dimuat pada basis data Oracle	*Event Clicked*
6	*inpOntButton*	*Button*	Memuat Data ontologi ke dalam basis data Oracle	*Event Clicked*

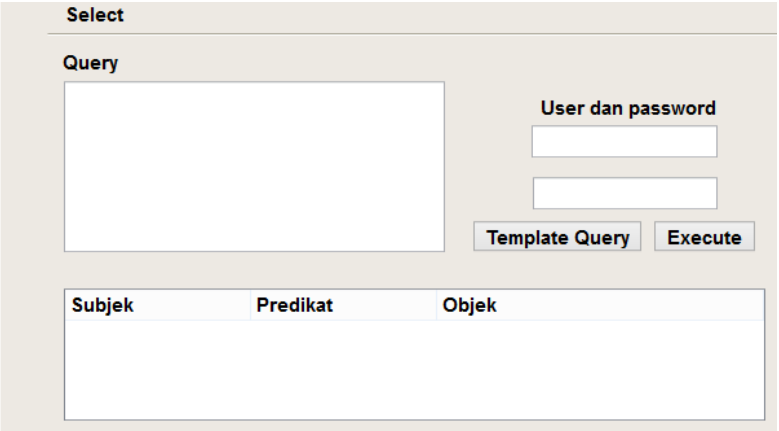
Gambar 3.35 Rancangan Antarmuka Halaman Memasukkan *Triple*

3.5.5 Halaman Melakukan *Query*

Halaman ini digunakan oleh pengguna untuk melakukan *query* select pada ontologi sesuai dengan format pencarian (select pada Oracle semantic. Rancangan antarmuka Aplikasi dapat dilihat pada Gambar 3.36 dan penjelasan mengenai elemen-elemen yang terdapat pada halaman ini dapat dilihat pada Tabel 3.16.

Tabel 3.16 Spesifikasi Atribut Antarmuka Halaman SQL Select

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Ke-luaran
1	*queryTextArea*	*TextArea*	Mengisikan *query* yang akan dijalankan pada basis data oracle	*String*
2	*jTable*	*Table*	Menampilkan hasil *query* yang telah dijalankan	
3	*usrText2*	*TextField*	Menampilkan daftar kelas yang bisa diakses	*String*
4	*passText2*	*TextField*	Memilih kelas untuk diakses	*String*
5	*temQueButton*	*Button*	Menampilkan template *query* select pada Textarea	*Event Clicked*
6	*executeSQLButton*	*Button*	Menjalankan perintah *query* select yang telah diisikan pada textarea	*Event Clicked*



Gambar 3.36 Rancangan Antarmuka Halaman SQL Select

[halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tahap implementasi yang dilakukan berdasarkan rancangan-rancangan yang telah dibahas pada bab-bab sebelumnya. Selain itu juga akan dibahas lingkungan yang digunakan untuk melakukan implementasi sistem.

Proses implementasi yang dilakukan dibagi dalam tiga aspek. Aspek yang pertama adalah pengimplementasian teknologi Oracle semantic pada basis data Oracle 11g release 2 yang dimana harus diaktifkan terlebih dahulu sebelum dapat berinteraksi dengan data semantik dalam basis data. Aspek yang kedua adalah pengimplementasian cara penyimpanan data semantik pada basis data Oracle. Aspek yang ketiga adalah implementasi antarmuka sistem, yang berefleksi terhadap rancangan kasus penggunaan dan rancang antarmuka sistem. Aspek ini direalisasikan dalam bentuk tiap halaman yang nanti akan digunakan oleh sistem sebagai antarmuka.

4.1 Lingkungan Implementasi

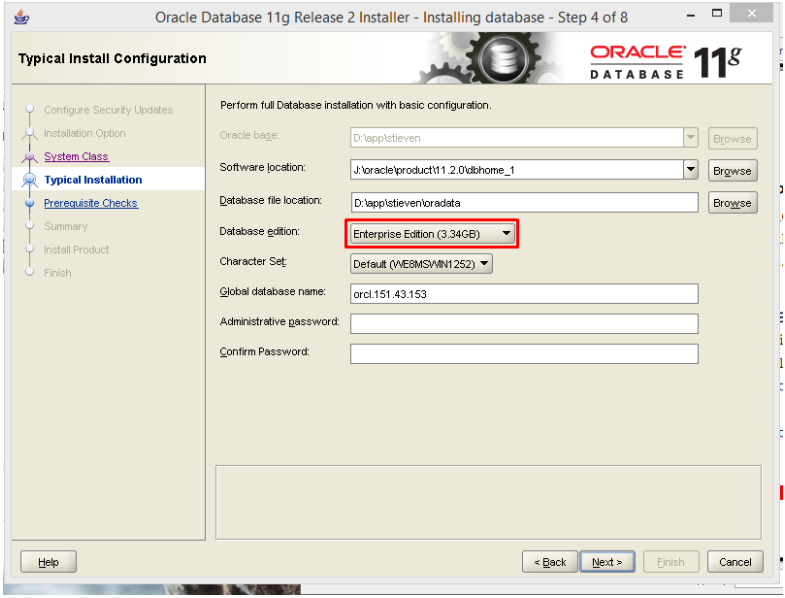
Pengimplementasian sistem menggunakan beberapa kakas bantu yang membangun sebuah lingkungan implementasi. Adapun rincian kakas-kakas bantu yang digunakan dalam pengimplementasian sistem ini adalah sebagai berikut.

1. Sistem Operasi Windows 8.1 Professional 64 bit
2. NetBeans IDE 8.0.2 (IDE)
3. Jena Adapter Library (v2.6) sebagai konektor java dengan jdbc
4. Oracle 11g release 2 with Semantic Technology
5. Java Development Kit (JDK) 8 update 25
6. Oracle SQL Developer v.4.1.0.19.07
7. Protégé 4.3

Pada Basis data Oracle, Fitur semantik belum aktif secara langsung setelah instalasi basis data Oracle, sehingga penulis harus melakukan persiapan basis data Oracle dan *library* yang akan digunakan untuk membangun Aplikasi.

4.1.1 Persiapan Implementasi Basis Data

Pada basis data Oracle, khususnya versi 11g release 2, telah disediakan format penyimpanan data semantik namun fitur semantik hanya dapat digunakan jika edisi basis data Oracle yang dipakai adalah Enterprise edition seperti yang ditunjukan pada Gambar 4.1.

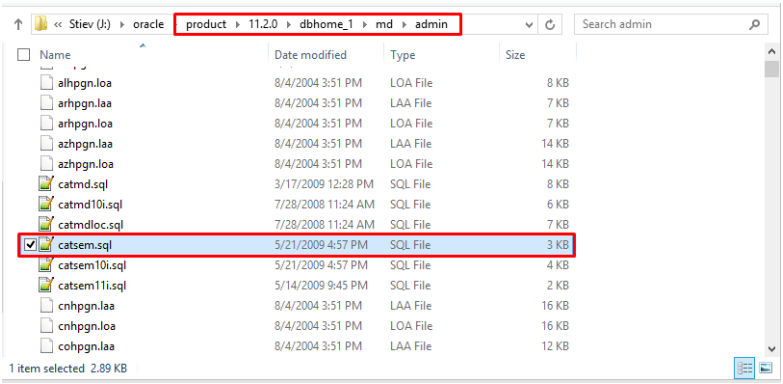


Gambar 4.1 Instalasi Basis data Oracle 11g release 2

Setelah instalasi basis data Oracle selesai, diperlukan menjalankan file sql yang telah disediakan oleh basis data Oracle seperti yang ditunjukkan pada Gambar 4.2.

Untuk menjalankan file catsem.sql, menggunakan Sql*Plus dengan akses user *sys as sysdba*. Kemudian menjalankan

catsem.sql dengan perintah seperti yang ditunjukan pada Kode Sumber 4.1. Perintah ini digunakan untuk membuka user mdsys dan membangun jaringan semantik pada basis data Oracle sehingga jika sudah ada jaringan semantik, atau tipe semantik, atau tabel atau paket PL/SQL, maka script catsem.sql akan mengeluarkan error.



Gambar 4. 2 File untuk mengaktifkan fitur

```
SQL> Sqlplus sys/12345678 as sysdba
SQL> alter user mdsys identified by mdsys account unlock;
SQL> @%ORACLE_HOME%\md\admin\catsem.sql
```

Kode Sumber 4.1 Perintah untuk mengaktifkan fitur semantik

Setelah fitur semantik telah aktif, Diperlukan suatu tablespace untuk menyimpan data logikal dalam komputer. Berikut contoh perintah untuk menambahkan tablespace yang baru yang ditunjukan pada Kode Sumber 4.2. Setelah membuat tablespace baru, perlu membuat user DB baru sebagai pemilik tablespace. Hal ini dikarenakan user MDSYS merupakan pusat dari segala hal yang berhubungan dengan semantik. Perintah membuat user DB baru ditunjukkan pada Kode Sumber 4.3.

```
create bigfile tablespace <nama tablespace>
datafile '?/dbs/<nama datafile untuk
<tablespace>01.dat'
size 512M reuse
autoextend on next 512M
maxsize unlimited
extent management local
segment space management auto;
```

Kode Sumber 4.2 Perintah Membuat Tablespace Baru

```
create user <nama user db> identified by <password>
default tablespace <nama tablespace>;

grant create session, resource to <nama user>;
```

Kode Sumber 4.3 Perintah Membuat user baru

Setelah user DB dan tablespace dibuat, masuk sebagai user yang telah dibuat untuk membuat tabel dan membuat model ontologi dengan akses user yang akan membangun model semantik pada Perintah untuk membuat tabel dan model ontologi dapat dilihat pada Kode Sumber 4.4. Untuk penjelasan lebih lanjut prosedur SEM_APIS.CREATE_SEM_MODEL dapat dilihat pada Tabel 2.3.

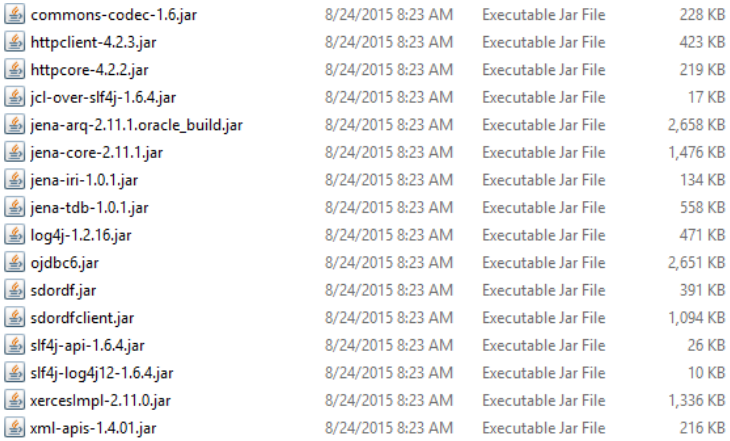
```
connect <nama user>/<password>;
create table <nama tabel>
(id number,
triple sdo_rdf_triple_s);

execute sem_apis.create_sem_model('<Nama Model>',
'<nama model>', 'triple');
```

Kode Sumber 4.4 Perintah membuat

4.1.2 Persiapan Implementasi Aplikasi

Untuk membangun aplikasi, diperlukan beberapa hal seperti, *library* agar aplikasi dapat mengakses koneksi basis data Oracle. Pada aplikasi ini, diperlukan sebuah *adapter*, yaitu Jena Adapter for Oracle, dimana aplikasi berbasis java dapat mengimplementasikan fitur-fitur jena adapter pada basis data Oracle dengan fitur semantik. Library yang diperlukan pada aplikasi ini dapat ditunjukan pada Gambar 4.3.



commons-codec-1.6.jar	8/24/2015 8:23 AM	Executable Jar File	228 KB
httpclient-4.2.3.jar	8/24/2015 8:23 AM	Executable Jar File	423 KB
httpcore-4.2.2.jar	8/24/2015 8:23 AM	Executable Jar File	219 KB
jcl-over-slf4j-1.6.4.jar	8/24/2015 8:23 AM	Executable Jar File	17 KB
jena-arq-2.11.1.oracle_build.jar	8/24/2015 8:23 AM	Executable Jar File	2,658 KB
jena-core-2.11.1.jar	8/24/2015 8:23 AM	Executable Jar File	1,476 KB
jena-iri-1.0.1.jar	8/24/2015 8:23 AM	Executable Jar File	134 KB
jena-tdb-1.0.1.jar	8/24/2015 8:23 AM	Executable Jar File	558 KB
log4j-1.2.16.jar	8/24/2015 8:23 AM	Executable Jar File	471 KB
ojdbc6.jar	8/24/2015 8:23 AM	Executable Jar File	2,651 KB
sdordf.jar	8/24/2015 8:23 AM	Executable Jar File	391 KB
sdordfclient.jar	8/24/2015 8:23 AM	Executable Jar File	1,094 KB
slf4j-api-1.6.4.jar	8/24/2015 8:23 AM	Executable Jar File	26 KB
slf4j-log4j12-1.6.4.jar	8/24/2015 8:23 AM	Executable Jar File	10 KB
xerceslmpl-2.11.0.jar	8/24/2015 8:23 AM	Executable Jar File	1,336 KB
xml-apis-1.4.01.jar	8/24/2015 8:23 AM	Executable Jar File	216 KB

Gambar 4.3 Daftar library untuk membangun aplikasi.

4.2 Implementasi Basis Data

Pada Oracle Semantic, terdapat sintaks sintaks yang dipakai untuk membangun semantik atau ontologi. Data yang telah dimasukkan hanya dapat dihapus dari model ontologi namun data yang tersimpan di dalam tabel MDSYS.RDF_VALUE$ akan tetap tersimpan hanya saja data yang telah dihapus tidak dapat dipakai.

Berikut implementasi penyimpanan elemen-elemen ontologi pada Oracle Semantic, seperti:

1. Kelas
2. Properti

3. Karakteristik properti
4. Instance
5. Rule

```
-- Syntax Insert Triple
INSERT INTO <nama tabel> VALUES ( <no id>,
SDO_RDF_TRIPLE_S( <nama model>,
<subject>,
<predikat>,
<Objek>));
```

Kode Sumber 4.5 *Query* insert manual pada Oracle semantic

4.2.1 Memasukkan Class

Sintak sql ini menyimpan data semantik yang berupa URI ke dalam Oracle Semantic. Dalam elemen Class, terdapat elemen *Disjoint Class* dan *SubClass*. Disjoint Class menunjukkan perbedaan kelas misal kelas *male* dan *female*. Sedangkan Subclass menunjukan suatu *Class* merupakan *subclass* dari Class yang lain, seperti *Female* merupakan subclass *Sex*. Sintak yang dipakai untuk membuat kelas sepeti yang ditunjukan pada Kode Sumber 4.6

```
-- insert Class
INSERT INTO family1_rdf_data VALUES (17,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/Person',
'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',
'http://www.w3.org/2000/01/rdf-schema#Class'));

--insert Disjoint Class
INSERT INTO family1_rdf_data VALUES (17,
SDO_RDF_TRIPLE_S('family',
'http://www.co-ode.org/roberts/family-tree.owl#Person',
'http://www.w3.org/2002/07/owl#disjointWith ',
'http://www.co-ode.org/roberts/family-tree.owl#Sex'));

--insert subclass
```

```
INSERT INTO family1_rdf_data VALUES (17,
SDO_RDF_TRIPLE_S('family',
'http://www.co-ode.org/roberts/family-tree.owl#Female',
'http://www.w3.org/2000/01/rdf-schema#subClassOf ',
'http://www.co-ode.org/roberts/family-tree.owl#Sex'));
```

Kode Sumber 4.6 *Query* Insert Class, Disjoin Class, SubClass

Dapat dilihat pada contoh *Query* yang ditunjukan pada Kode Sumber 4.1, pada penambahan kelas person, subjek dari triple adalah nama dari suatu kelas, dengan *type* sebagai predikat dan objeknya berisi *Class*. Sedangkan pada penambahan *Triple* disjoint class, subjek dan objek merupakan kelas yang tidak sama dan predikatnya yaitu *DisjointWith*. Untuk subclass, subjeknya merupakan subkelas dari objek dengan predikat *subClassOf*.

4.2.2 Memasukkan Properti

Dalam elemen properti, terdapat dua jenis dua properti, yaitu properti tipe data dan properti objek. Dalam elemen properti objek, terdapat elemen subproperti. Properti objek merupakan relasi yang menghubungkan 2 *instance* atau individual. Pada Properti memiliki 2 elemen sampingan yaitu domain dan range. Domain membatasi subjek dari suatu relasi, sedangkan range membatasi objek dari relasi tersebut.Sedangkan Subproperti merelasikan antara 2 properti objek yang memiliki hubungan misalkan, FatherOf merupakan subproperti dari ParentOf. Sintak SQL penambahan data triple properti ditunjukan pada Kode Sumber 4.7.

```
--Insert Properti
  INSERT INTO family1_rdf_data VALUES (13,
  SDO_RDF_TRIPLE_S('family',
  'http://www.example.org/family/Cathy',
  'http://www.example.org/family/sisterOf',
  'http://www.example.org/family/Jack'));
```

```
--insert Range dari properti
INSERT INTO family1_rdf_data VALUES (24,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/sisterOf',
'http://www.w3.org/2000/01/rdf-schema#range',
'http://www.example.org/family/Person'));

-- Insert Domain dari properti
INSERT INTO family1_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/sisterOf',
'http://www.w3.org/2000/01/rdf-schema#domain',
'http://www.example.org/family/Female'));

--Insert Subproperty
INSERT INTO family1_rdf_data VALUES (23,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/sisterOf',
'http://www.w3.org/2000/01/rdf-schema#subPropertyOf',
'http://www.example.org/family/siblingOf'));

-- Insert properti tipe data
INSERT INTO family1_rdf_data VALUES (30,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/Cathy',
'http://www.example.org/family/height',
'"5.8"^^xsd:decimal'));
```

Kode Sumber 4.7 *Query* Insert Properti, Subproperti, properti tipe data.

4.2.3 Memasukkan Karakteristik Properti

Dalam elemen-elemen karakteristik properti, ada beberapa macam, seperti properti Transitive, properti Simetrik, properti Fungsional, dan properti inverse of. Untuk properti inverse of, merelasikan antara 2 properti objek misal *hasPartner* dan *isPartnerOf.* Sedangkan properti transitif, simetrik, dan fungsional

memiliki bentuk *Triple*, dengan subjek *Triple* adalah property objek dengan predikat *type*. Berikut contoh sintak insert pada Oracle. Sintak perintah sql untuk memasukkan karakteristik properti objek dapat ditunjukkan pada Kode Sumber 4.8.

```
-- Insert inverse of
INSERT INTO family1_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/issisterOf',
http://www.w3.org/2002/07/owl#inverseOf',
'http://www.example.org/family/hassister'));

-- Insert Functional
INSERT INTO family1_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.co-ode.org/roberts/family-tree.owl#hasSex',
'http://www.w3.org/1999/02/22-rdf-syntax-ns#type,
'http://www.w3.org/2002/07/owl#FunctionalProperty'));

--Insert Transitive
INSERT INTO family1_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.co-ode.org/roberts/family-tree.owl#hasForeFather',
'http://www.w3.org/1999/02/22-rdf-syntax-ns#type,
'http://www.w3.org/2002/07/owl#TransitiveProperty'));

--Insert Symmetric
INSERT INTO family1_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.co-ode.org/roberts/family-tree.owl#hasForeFather',
'http://www.w3.org/1999/02/22-rdf-syntax-ns#type,
' http://www.w3.org/2002/07/owl#SymmetricProperty'));
```

Kode Sumber 4.8 *Query* Insert Karakteristik properti.

4.2.4 Memasukkan Instance

Instance atau individu merupakan instance dari sebuah kelas. Dalam kode sumber 4.6 dapat dilihat bahwa James_bright_1761 merupakan instance dari kelas

NamedIndividual, dengan predikat triple yaitu http://www.w3.org/1999/02/22-rdf-syntax-ns#type.

```
--Insert Instance
INSERT INTO family1_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.co-ode.org/roberts/family-tree.owl# james_bright_1761',
'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',
'http://www.w3.org/2002/07/owl#NamedIndividual'));
```

Kode Sumber 4.9 *Query* Insert Instance

4.2.5 Memasukkan *Rule*

Dalam Basis data Oracle, *rule* dapat ditambahkan ke dalam model ontologi sehingga basis data Oracle dapat melakukan *inferecing*. Aturan ini terbagi menjadi 2 yaitu *rulebase* dan *rule*. Rulebase merupakan kumpulan dari rule yang ada. Contoh sintak perintah SQL yang dipakai untuk menambah aturan atau rule dan rulebase dapat dilihat pada Gambar 3.21.

4.3 Implementasi Antarmuka Sistem

Implementasi antarmuka sistem dilakukan dengan menggunakan tab dari tabpane untuk masing-masing halaman. Berikut ini akan dijelaskan mengenai implementasi antarmuka sistem yang sudah direalisasikan.

4.3.1 Halaman Menambah User DB dan Tablespace

Halaman ini merupakan halaman yang digunakan untuk melakukan kasus penggunaan menambah user DB dan Tablespace baru. Realisasi dari halaman ini dapat ditunjukkan pada Gambar 4.4.

4.3.2 Halaman Menambah Model Ontologi Baru

Halaman ini merupakan halaman yang digunakan untuk menambahkan model ontologi baru dari user DB. Realisasi dari halaman ini dapat ditunjukkan pada Gambar 4.5.

4.3.3 Halaman Menambah data Triple ke dalam Model Ontologi

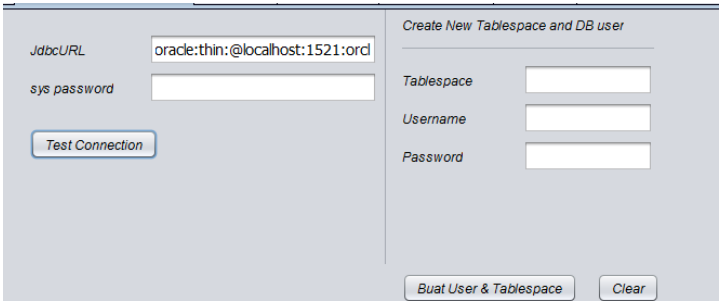
Halaman ini merupakan halaman yang digunakan untuk menambah *triple* ke dalam model ontologi. Realisasi dari halaman ini dapat ditunjukkan pada Gambar 4.6.

4.3.4 Halaman Memasukkan Ontologi

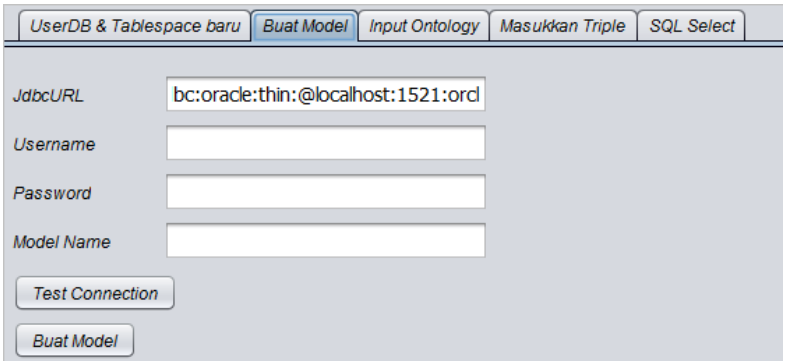
Halaman ini adalah halaman yang digunakan untuk memasukkan atau menyimpan data ontologi dalam bentuk RDF, OWL, N-TRIPLE dan lain lain. Realisasi dari halaman ini dapat ditunjukkan pada Gambar 4.7.

4.3.5 Halaman Melakukan *Query*

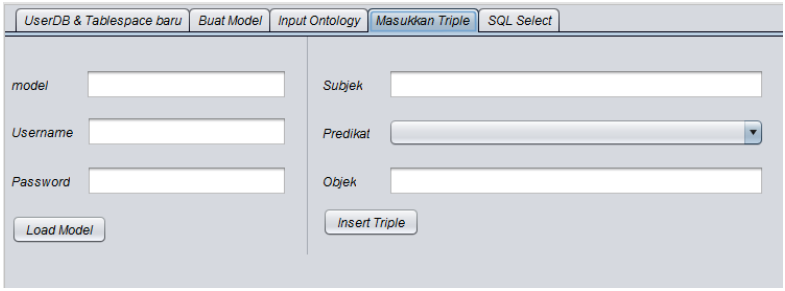
Halaman ini adalah halaman yang digunakan untuk melakukan pencarian data ontologi dalam model ontologi pada basis data Oracle. Hasil *query* akan ditampilkan pada tabel yang tersedia. Realisasi dari halaman ini dapat ditunjukkan pada Gambar 4.8.



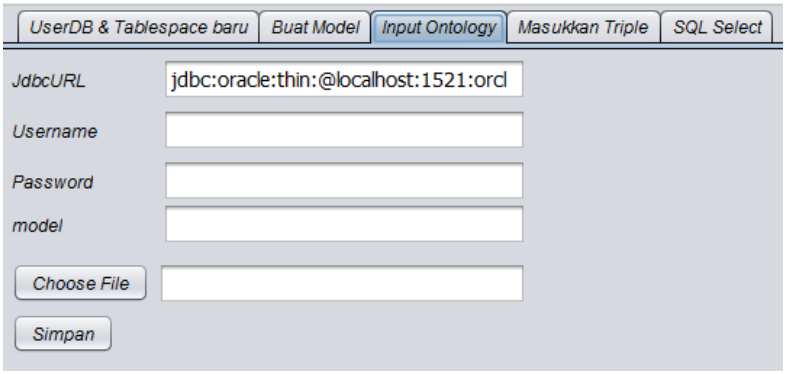
Gambar 4.4 Halaman Menambah User DB dan Tablespace baru



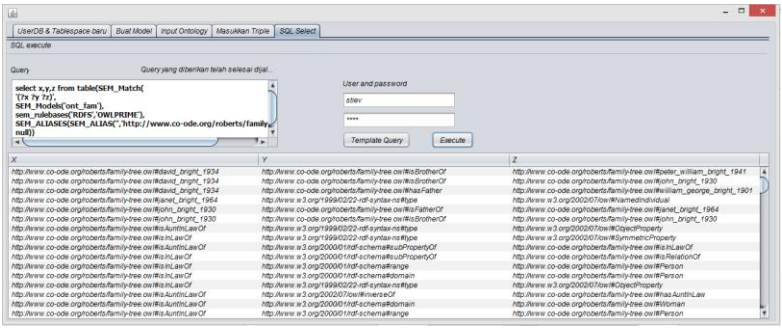
Gambar 4.5 Halaman Menambah Model Ontologi Baru



Gambar 4.6 Halaman Menambah Triple



Gambar 4.7 Halaman Memasukkan Ontologi



Gambar 4.8 Halaman Melakukan *Query*

[halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas mengenai tahap uji coba yang telah dilakukan terhadap sistem beserta evaluasi dari hasil uji coba tersebut. Uji coba dilakukan dengan membandingkan hasil *query* dari pencarian data semantik menggunakan inferencing.. Aspek yang diperhatikan dalam pengujian adalah terpenuhinya fungsionalitas yang sudah dirancang pada kasus pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba adalah kombinasi antara perangkat keras dan perangkat lunak yang digunakan untuk melakukan uji coba. Pengujian dilakukan dengan menggunakan sebuah lingkungan pengujian Adapun rincian dari masing-masing lingkungan pengujian tersebut secara berturut-turut ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian

Spesifikasi	Deskripsi
CPU	Intel ® Core ™ i5-2410M CPU @ 2.30 GHz
RAM	8.00 GB
Sistem Operasi	Windows 8.1 Professional 64 bit
Basis Data	Oracle 11g release 2
RDF Tools	Protégé 4.3
Java	JDK versi 8 update 25

5.2 Aspek Pengujian

Dalam melakukan pengujian terdapat aspek-aspek yang diperhatikan yaitu fungsionalitas, kemampuan inference basis data Oracle, waktu eksekusi *query*, dan *usability* (kemudahan).

5.3 Skenario Uji Coba Fungsionalitas

Pada bagian ini akan dibahas mengenai proses uji coba yang digunakan. Pengujian dilakukan dengan metode *black box* untuk menguji masing-masing fungsionalitas yang sudah dirancang pada aplikasi. Metode *black box* adalah metode pengujian perangkat lunak yang memeriksa fungsionalitas dari suatu perangkat lunak tanpa memandang struktur internalnya.

Pada proses uji coba, setiap peserta uji coba diminta untuk melakukan serangkaian perintah terhadap aplikasi yang selanjutnya akan disebut sebagai kasus pengujian. Kasus pengujian ini berkorelasi dengan kasus-kasus penggunaan dan kebutuhan fungsional yang sebelumnya sudah dirancang dan dijelaskan pada Bab III. Setelah menyelesaikan semua skenario kasus pengujian, peserta uji coba diminta untuk mengisi lembar *feedback* yang berisi pertanyaan-pertanyaan non-fungsional seputar sistem yang diuji.

5.4 Skenario Uji Coba Inference

Pada bagian ini akan dibahas mengenai proses uji coba yang digunakan. Pengujian yang dilakukan yaitu menguji kemampuan *inference* data semantik yang tersimpan pada basis data Oracle. Kasus pengujian yang akan diujikan yaitu uji coba inference menggunakan bentuk bentuk karakteristik properti yang terdapat pada ontologi keluarga.

Pada proses uji coba, setiap karakteristik properti yang akan diuji coba disebut sebagai kasus pengujian. Kasus pengujian ini dilakukan untuk menguji aplikasi berdasarkan kasus penggunaan yang telah dijelaskan pada subbab 3.4 dan membandingkan hasil *query* yang dilakukan oleh Oracle semantik menggunakan fitur Aplikasi melakukan perintah SQL SELECT. Setelah menyelesaikan semua skenario uji coba, akan dilakukan perbandingan kecepatan waktu *query* dari Oracle atau RDF tools seperti protégé. Karakteristik properti yang dapat akan diuji coba dapat dilihat pada Tabel 5.2.

Tabel 5.2 Daftar Karakteristik properti

No.	Karakteristik properti
1	Properti Fungsional
2	Properti Transitif
3	Properti Simetrik
4	Properti *Inverse Of*

5.5 Kasus Pengujian Fungsionalitas

Pengujian fungsionalitas aplikasi dilakukan dengan menguji kasus-kasus uji tiap kasus penggunaan. Berikut adalah kasus uji dari fungsionalitas aplikasi.

5.5.1 Kasus uji coba Menambah user DB dan Tablespace baru

Kasus pengujian ini dibuat untuk menguji fitur aplikasi dalam menambahkan user DB dan *Tablespace* baru pada basis data. Dalam kasus pengujian ini, diperlukan akan hak akses sysdba sehingga dapat membuat user DB dan *Tablespace* baru. Jika user DB dan Tablespace baru dapat dibuat, maka Kasus penggunaan ini dikatakan berhasil. Detil dari kasus pengujian ini dapat ditunjukan pada Tabel 5.3.

Tabel 5.3 Kasus Pengujian Menambah User DB dan Tablespace baru

ID	**UJ-001**
Kasus Penggunaan	Menambah user DB dan Tablespace baru
Nama	Pengujian menambah user DB dan Tablespace baru
Tujuan Pengujian	Menguji apakah aplikasi dapat menambah user DB dan tablespace baru

Kondisi Awal	Nama User DB dan Tablespace belum ada di dalam basis data Oracle
Langkah Pengujian	1. Aktor memasukkan password user sys 2. Aktor memasukkan nama tablespace baru 3. Aktor memasukkan nama user dan password 4. Aktor menekan tombol Buat user DB dan tablespace baru.
Hasil yang Diharapkan	Nama User DB dan tablespace baru telah dibuat
Kondisi Akhir	Nama User DB dan tablespace baru telah dibuat

5.5.2 Kasus Pengujian Menambah Model Ontologi Baru

Kasus pengujian ini dibuat untuk menguji fitur aplikasi dalam menambahkan model ontologi pada basis data Oracle yang akan digunakan untuk menyimpan data ontologi. Pada kasus pengujian ini, diperlukan persiapan yaitu mempersiapkan nama model yang belum ada pada user DB. Jika model ontologi telah tersimpan pada basis data Oracle, maka kasus penggunaan ini dikatakan berhasil. Detil dari kasus pengujian ini dapat ditunjukan pada Tabel 5.4.

Tabel 5.4 Kasus Pengujian Menambah Model Ontologi Baru

ID	**UJ-002**
Kasus Penggunaan	Menambah Model Ontologi Baru
Nama	Pengujian menambah model
Tujuan Pengujian	Menguji apakah aplikasi dapat menambah model ontologi
Kondisi Awal	Model ontologi belum ada
Langkah Pengujian	1. Aktor mengisikan data yang diminta seperti nama user, password dan nama model ontologi yang akan dibuat. 2. Aktor menekan tombol Buat Model.

Hasil yang Diharapkan	Model ontologi dibuat pada basis data Oracle
Kondisi Akhir	Model ontologi telah terbuat pada basis data Oracle

5.5.3 Kasus Pengujian Menambah Data *Triple*

Kasus pengujian menambahkan data *Triple* akan menguji fitur aplikasi memasukkan data *triple* ontologi ke dalam basis data Oracle yang nantinya dapat dicari secara semantik dengan fungsi tabel SEM_MATCH. Penambahan data *triple* ini diharuskan memasukkan URI pada *textfield* yang disediakan. Dalam kasus pengujian ini perlu dipersiapkan data-data triple yang akan dimasukkan ke dalam model sehingga data tripel tidak inkonsisten. Jika data triple yang ditambahkan telah tersimpan maka kasus pengujian dikatakan berhasil. Detil dari kasus pengujian ini dapat ditunjukan pada Tabel 5.5.

Tabel 5.5 Kasus Pengujian Menambah Data *Triple*

ID	**UJ-003**
Kasus Penggunaan	Menambah Data *Triple* ke dalam Model Ontologi
Nama	Pengujian Menambah Data Triple
Tujuan Pengujian	Menguji apakah aplikasi dapat menambah data triple ke dalam model ontologi pada basis data Oracle
Kondisi Awal	Triple tidak ada pada model ontologi
Langkah Pengujian	1. Aktor mengisikan nama model, nama user dan password. 2. Aktor menekan tombol load model. 3. Aktor mengisi data subjek dan objek triple 4. Aktor memilih properti objek pada *combobox* 5. Aktor menekan tombol Insert Triple
Hasil yang Diharapkan	Triple yang ditambahkan tersimpan dalam model ontologi

Kondisi Akhir	Triple telah tersimpan dalam model ontologi

5.5.4 Kasus Pengujian Memasukkan Ontologi ke dalam RDBMS

Kasus pengujian ini dibuat untuk menguji fitur aplikasi yaitu memasukkan file ontologi ke dalam basis data Oracle. Dalam kasus pengujian ini, akan diuji apakah melalui aplikasi, pengguna dapat memasukkan file ontologi yang akan dimasukkan ke dalam model ontologi yang ada pada basis data Oracle. File yang akan dimasukkan adalah file dengan format seperti RDF, OWL, N-TRIPLE dan lain lain. Persiapan untuk pengujian ini adalah mempersiapkan file-file ontologi yang nantinya akan diuji untuk dimasukkan pada model ontologi. Jika data ontologi pada file ontologi dapat dimasukkan pada model ontologi yang diiinginkan, maka kasus pengujian dikatakan berhasil. Detil dari kasus pengujian ini dapat ditunjukan pada Tabel 5.6.

Tabel 5.6 Kasus Pengujian Memasukkan Ontologi

ID	**UJ-004**
Kasus Penggunaan	Memasukkan Ontologi ke dalam RDBMS
Nama	Pengujian Memasukkan Ontologi
Tujuan Pengujian	Menguji apakah aplikasi dapat memasukkan data ontologi ke dalam model ontologi yang ada pada basis data Oracle.
Kondisi Awal	Data ontologi belum dimasukkan
Langkah Pengujian	1. Aktor memasukkan data yang diminta seperti nama user, password, dan model ontologi. 2. Aktor menekan tombol choose file. 3. Aktor memilih file ontologi. 4. Aktor menekan tombol simpan.
Hasil yang Diharapkan	Data ontologi tersimpan pada model ontologi

Kondisi Akhir	Data ontologi telah tersimpan pada model ontologi

5.5.5 Kasus Pengujian Melakukan *Query*

Kasus pengujian melakukan *query* ini dibuat untuk memeriksa apakah fitur aplikasi ini dapat mengirim perintah *query* ke dalam basis data Oracle dan melakukan pengembalian data dalam bentu tabel. Dalam kasus pengujian ini, fitur aplikasi ini akan dicoba untuk melakukan perintah *query* yang telah dipersiapkan. Persiapan yang dimaksudkan di sini adalah mempersiapkan *query-query* yang digunakan untuk mencoba mencari data ontologi pada basis data Oracle. Detil dari kasus pengujian ini dapat ditunjukan pada Tabel 5.7

Tabel 5.7 Kasus Pengujian Melakukan *Query*

ID	**UJ-005**
Kasus Penggunaan	Melakukan *Query*
Nama	Pengujian melakukan *query*
Tujuan Pengujian	Menguji apakah aplikasi dapat melakukan *query* select dan mengembalikan hasil *query*
Kondisi Awal	Tabel masih kosong
Langkah Pengujian	1. Aktor menekan tombol template *query* 2. Aktor mengubah isi template *query* 3. Aktor mengisi nama user dan password basis data. 4. Aktor menekan tombol *execute*.
Hasil yang Diharapkan	Tabel menampilkan hasil *query*
Kondisi Akhir	Tabel menampilkan hasil *query*

5.6 Kasus Pengujian Non-Fungsionalitas

Pengujian non-fungsionalitas dilakukan dengan menguji kasus-kasus uji coba sesuai dengan skenario pada subbab 5.4. Berikut adalah kasus uji dari non-fungsionalitas aplikasi.

5.6.1 Uji Coba inference dengan karakteristik Properti

Pengujian *inference* ini dilakukan dengan menggunakan aplikasi yang telah dibuat dengan tujuan apakah terdapat perbedaan hasil *query* dimana properti objek pada *triple* harus memiliki karakteristik properti yang akan diujikan. Dalam ontologi ini, terdapat 4 karakteristik properti yang akan diuji coba. Berikut penjelasan dan *query* yang akan dilakukan:

5.6.1.1 Kasus Pengujian Karakteristik Properti Fungsional

Kasus pengujian karakteristik properti simetrik akan menguji perbandingan hasil *query* apabila *query* pencarian triple tersebut mengunakan inference dan tidak menggunakan inference. Seperti yang ditunjukan pada Kode Sumber 5.1 dan Kode Sumber 5.2, perbedaan adanya penggunaan inference atau tidak ditentukan dengan adanya sem_rulebase atau tidak.

```
 select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre rdf:type
<http://www.w3.org/2002/07/owl#FunctionalProperty>)',
SEM_Models('ont_fam'), sem_rulebases('OWLPRIME','RDFS'),
SEM_ALIASES(SEM_ALIAS('','http://www.co-
ode.org/roberts/family-tree.owl#')),
  null)) ;
```

Kode Sumber 5.1 *Query* Fungsional dengan menggunakan *inference*

```
Select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre rdf:type
<http://www.w3.org/2002/07/owl#FunctionalProperty>)',
```

```
SEM_Models('ont_fam'),null,
SEM_ALIASES(SEM_ALIAS('','http://www.co-
ode.org/roberts/family-tree.owl#')),
  null)) ;
```

Kode Sumber 5.2 *Query* Fungsional tanpa menggunakan *inference*

5.6.1.2 Kasus Pengujian Karakteristik Properti Transitif

Kasus pengujian properti transitif akan menguji perbandingan hasil *query* apabila *query* pencarian triple tersebut mengunakan inference dan tidak menggunakan inference. Seperti yang ditunjukan pada Kode Sumber 5.3 dan Kode Sumber 5.4, perbedaan adanya penggunaan inference atau tidak ditentukan dengan adanya sem_rulebase atau tidak.

```
select sub,pre ,obj   from table(SEM_Match(
'(?sub ?pre ?obj)(?pre rdf:type
<http://www.w3.org/2002/07/owl#TransitiveProperty>)',
SEM_Models('ont_fam'),sem_rulebases('OWLPRIME','RDFS'),
SEM_ALIASES(
SEM_ALIAS('','http://www.co-ode.org/roberts/family-tree.owl#')),
  null)) ;
```

Kode Sumber 5.3 *Query* Transitif dengan menggunakan *inference*

```
select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre rdf:type
<http://www.w3.org/2002/07/owl#TransitiveProperty>)',
SEM_Models('ont_fam'),Null,SEM_ALIASES(
SEM_ALIAS('','http://www.co-ode.org/roberts/family-tree.owl#')),
  null)) ;
```

Kode Sumber 5.4 *Query* Transitif tanpa menggunakan *inference*

5.6.1.3 Kasus Pengujian Karakteristik Properti Simetrik

Kasus pengujian properti simetrik akan menguji perbandingan hasil *query* apabila *query* pencarian triple tersebut mengunakan inference dan tidak menggunakan inference. Seperti

yang ditunjukan pada Kode Sumber 5.5 dan Kode Sumber 5.6, perbedaan adanya penggunaan inference atau tidak ditentukan dengan adanya sem_rulebase atau tidak.

```
select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre rdf:type
<http://www.w3.org/2002/07/owl#SymmetricProperty>)',
SEM_Models('ont_fam'),
sem_rulebases('OWLPRIME','RDFS'),
SEM_ALIASES(
SEM_ALIAS('','http://www.co-ode.org/roberts/family-tree.owl#')),
  null)) ;
```

Kode Sumber 5.5 *Query* simetrik dengan menggunakan *inference*

```
select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre rdf:type
<http://www.w3.org/2002/07/owl#SymmetricProperty>)',
SEM_Models('ont_fam'),
null,
SEM_ALIASES(SEM_ALIAS('','http://www.co-
ode.org/roberts/family-tree.owl#')),
  null)) ;
```

Kode Sumber 5.6 *Query* simetrik tanpa menggunakan inference

5.6.1.4 Kasus Pengujian Karakteristik Properti *inverseOf*

Kasus pengujian karakteristik properti *inverseOf* akan menguji perbandingan hasil *query* apabila *query* pencarian data *triple* tersebut mengunakan inference dan tidak menggunakan inference. Dengan menggunakan Aplikasi sederhana yang digunakan untuk melakukan perintah select sesuai yang ditunjukan pada Kode Sumber 5.7 dan Kode Sumber 5.8. Perbedaan *query* yang menggunakan inference adalah adanya nilai atribut rulebase yaitu sem_rulebases pada fungsi sem_match.

```
select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre owl:inverseOf  ?pre2)',
SEM_Models('ont_fam'),sem_rulebases('OWLPRIME','RDFS'),
SEM_ALIASES(SEM_ALIAS('','http://www.co-
ode.org/roberts/family-tree.owl#')),null)) ;
```

Kode Sumber 5.7 *Query* inverseOf dengan menggunakan *inference*

```
select sub,pre,obj from table(SEM_Match(
'(?sub ?pre ?obj)(?pre owl:inverseOf  ?pre2)',
SEM_Models('ont_fam'),null,
SEM_ALIASES(SEM_ALIAS('','http://www.co-
ode.org/roberts/family-tree.owl#')),  null)) ;
```

Kode Sumber 5.8 *Query* InverseOf tanpa menggunakan *inference*

5.6.2 Uji Coba *Usability*

Pengujian ini dilakukan dengan menguji kemudahan dalam penggunaan aplikasi. Pada pengujian ini terdapat 2 indikator penilaian yaitu kejelasan tampilan aplikasi dan kemudahan dalam penggunaan aplikasi. Penilaian kejelasan tampilan berdasarkan apakah peserta merasa bahwa tampilan aplikasi sangat jelas sesuai fitur-fitur yang ada, sedangkan penilaian kemudahan dalam penggunaan aplikasi berdasarkan pada mudahnya pengguna menggunakan aplikasi untuk menyimpan ataupun melakukan inference dengan aplikasi. Rentang nilai pada tiap indikator adalah 1-4. Semakin tingginya nilai dari suatu indikator, berarti semakin baik dari suatu indikator tersebut.

5.6.3 Uji Coba Waktu *Query*

Pengujian ini dilakukan dengan mengukur waktu eksekusi tiap *query* yang telah disebutkan pada 5.7.2. Pengujian ini juga menghitung waktu yang diperlukan oleh kakas bantu ontologi dalam hal ini adalah protégé untuk melakukan inference dari data

ontologi yang ada. Pengujian ini menggunakan fitur basis data Orale yaitu *autotrace*. Penjelasan fitur *autotrace* dapat ditunjukkan pada Tabel 5.8. Pengujian ini akan mengaktifkan fitur autotrace dengan pengaturan SET AUTOTRACE TRACEONLY STATISTIC sehingga hasilnya adalah waktu eksekusi dan statistic dari perintah SQL saja.

Tabel 5.8 Pengaturan *autotrace*

Nama Fitur	**Deskripsi**
SET *AUTOTRACE* OFF	Mematikan fitur *autotrace*. Merupakan nilai default pada basis data Oracle
SET *AUTOTRACE* ON EXPLAIN	Laporan *autotrace* hanya menunjukkan *optimizer execution path*
SET *AUTOTRACE* ON STATISTICS	Laporan *autotrace* hanya menunjukkan statistik eksekusi *statement* SQL
SET *AUTOTRACE* ON	Laporan *autotrace* menampilkan *optimizer execution path* dan statistik eksekusi *statement* SQL
SET *AUTOTRACE* TRACEONLY	Memiliki kesamaan dengan SET AUTOTRACE ON, namun tidak menampilkan hasil *query* dari user. Jika fitur Statistic diaktifkan, data *query* tetap diambil namun tidak dicetak atau dimunculkan.

5.7 Evaluasi

Pada subbab ini akan dibahas mengenai evaluasi pengujian yang sudah dilakukan pada setiap kasus uji coba. Sama seperti subbab sebelumnya, dimana pegujian dibagi dua yaitu uji coba fungsionalitas dan uji coba karakteristik properti.

5.7.1 Evaluasi Fungsionalitas

Evaluasi ini adalah hasil dari pengujian kasus uji coba pada subbab 5.2.1. Hasil dinyatakan dalam status terpenuhi atau tidak. Hasil ditunjukan pada Tabel 5.9

Tabel 5.9 Tabel evaluasi kasus uji fungsionalitas

No	Kode Kasus Pengujian	Terpenuhi
1	UJ-001	√
2	UJ-002	√
3	UJ-003	√
4	UJ-004	√
5	UJ-005	√

5.7.2 Evaluasi Non-Fungsionalitas

Evaluasi ini merupakan hasil pengujian dari kasus pengujian yang terdapat pada subbab 5.6. Hasil pengujian akan dijelaskan pada subbab-subbab seperti berikut:

5.7.2.1 Evaluasi inference dengan Karakteristik Properti

Evaluasi ini adalah hasil dari uji coba inference dengan karakteristik properti. Hasil dari uji coba ini merupakan perbandingan hasil *query*, dan menyatakan contoh perbedaan hasil *query* yang menggunakan inference ataupun tidak menggunakan inference. Pengujian dilakukan dengan menggunakan fitur melakukan *query* yang terdapat pada aplikasi.

5.7.2.1.1 Hasil Kasus Pengujian Karakteristik Properti Fungsional

Hasil kasus pengujian karakteristik properti fungsional adalah perbandingan hasil *query* dengan menggunakan *inference* dan yang tidak menggunakan *inference* pada model ontologi yang

ada pada basis data Oracle. Kesamaan dalam kedua *query* yaitu predikatnya berupa properti objek yang memiliki karakteristik properti fungsional. Hasil *query* dari kedua *query* dapat dilihat pada Gambar 5.1 dan Gambar 5.2. Perbedaan hasil *query* adalah data triple yang dihasilkan melalui *inference* dengan predikat seperti pada daftar properti objek yang dapat ditunjukan pada Gambar 3.13. Contoh data triple yang hanya ada pada hasil *query* dengan inference adalah seperti yang ditunjukan pada Gambar 5.2. Berdasarkan kedua hasil *query*, dapat dikatakan Basis data Oracle mampu melakukan inference dengan properti objek yang karakteristik properti fungsional.

SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#alfred_steward_1...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1846
http://www.co-ode.org/roberts/family-tree.owl#john_steward_1...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1833
http://www.co-ode.org/roberts/family-tree.owl#joseph_steward_...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1837
http://www.co-ode.org/roberts/family-tree.owl#harsley_steward_...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1839
http://www.co-ode.org/roberts/family-tree.owl#caroline_steward...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1853
http://www.co-ode.org/roberts/family-tree.owl#john_steward_1...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1802
http://www.co-ode.org/roberts/family-tree.owl#mark_steward_1...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1811
http://www.co-ode.org/roberts/family-tree.owl#ellen_steward_17...	http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear	1797

Gambar 5.1 Hasil *query* tanpa menggunakan inference (fungsional)

SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#edmund_bright_...	http://www.co-ode.org/roberts/family-tree.owl#hasMother	http://www.co-ode.org/roberts/family-tree.owl#sarah_webb
http://www.co-ode.org/roberts/family-tree.owl#mary_eliza_jesso...	http://www.co-ode.org/roberts/family-tree.owl#hasFather	http://www.co-ode.org/roberts/family-tree.owl#james_jessop_18...
http://www.co-ode.org/roberts/family-tree.owl#reginald_tacey_s...	http://www.co-ode.org/roberts/family-tree.owl#hasFather	http://www.co-ode.org/roberts/family-tree.owl#john_tacey_stew...
http://www.co-ode.org/roberts/family-tree.owl#martha_stafford_...	http://www.co-ode.org/roberts/family-tree.owl#hasMother	http://www.co-ode.org/roberts/family-tree.owl#rebecca_wife_of_...
http://www.co-ode.org/roberts/family-tree.owl#violet_heath_1887	http://www.co-ode.org/roberts/family-tree.owl#hasMother	http://www.co-ode.org/roberts/family-tree.owl#harriet_whitefield...
http://www.co-ode.org/roberts/family-tree.owl#mary_ann_whitf...	http://www.co-ode.org/roberts/family-tree.owl#hasFather	http://www.co-ode.org/roberts/family-tree.owl#james_whitfield_...
http://www.co-ode.org/roberts/family-tree.owl#martha_ann_brig...	http://www.co-ode.org/roberts/family-tree.owl#hasFather	http://www.co-ode.org/roberts/family-tree.owl#james_bright_18...
http://www.co-ode.org/roberts/family-tree.owl#leonard_john_bri...	http://www.co-ode.org/roberts/family-tree.owl#hasMother	http://www.co-ode.org/roberts/family-tree.owl#charlotte_hewett...

Gambar 5.2 Hasil *query* Menggunakan inference (fungsional)

5.7.2.1.2 Hasil Kasus Pengujian Karakteristik Properti Transitif

Pengujian ini dilakukan dengan melakukan *query* yang dapat dilihat pada subbab 5.6.1.2. Evaluasi hasil Kasus pengujian yaitu membandingkan hasil *query* yang ditunjukkan pada Gambar 5.3 dan Gambar 5.4. Kesamaan dalam kedua *query* yaitu predikatnya berupa properti objek yang memiliki karakteristik properti transitif. Perbedaan hasil *query* adalah data triple yang dihasilkan melalui *inference* dengan predikat seperti pada daftar properti objek yang dapat ditunjukan pada Gambar 3.11. Contoh data triple yang hanya ada pada hasil *query* dengan inference

adalah seperti yang ditunjukan pada Gambar 5.4. Berdasarkan kedua hasil *query*, dapat dikatakan Basis data Oracle mampu melakukan inference dengan properti objek yang karakteristik properti Transitif.

SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#harry_jam...	http://www.co-ode.org/roberts/family-tree.owl#isForefatherOf	http://www.co-ode.org/roberts/family-tree.owl#robert_nelson_2035
http://www.co-ode.org/roberts/family-tree.owl#steven	http://www.co-ode.org/roberts/family-tree.owl#isForefatherOf	http://www.co-ode.org/roberts/family-tree.owl#louse_templar_1993

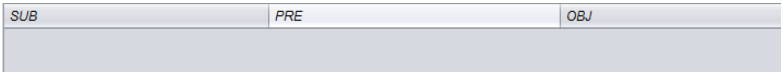
Gambar 5.3 Hasil *query* tanpa menggunakan inference (transitif)

SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#elizabe...	http://www.co-ode.org/roberts/family-tree.owl#isForemotherOf	http://www.co-ode.org/roberts/family-tree.owl#jane_archer
http://www.co-ode.org/roberts/family-tree.owl#elizabe...	http://www.co-ode.org/roberts/family-tree.owl#isForemotherOf	http://www.co-ode.org/roberts/family-tree.owl#john_archer_1804
http://www.co-ode.org/roberts/family-tree.owl#elizabe...	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf	http://www.co-ode.org/roberts/family-tree.owl#william_archer_1801
http://www.co-ode.org/roberts/family-tree.owl#elizabe...	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf	http://www.co-ode.org/roberts/family-tree.owl#jane_archer
http://www.co-ode.org/roberts/family-tree.owl#elizabe...	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf	http://www.co-ode.org/roberts/family-tree.owl#john_archer_1804
http://www.co-ode.org/roberts/family-tree.owl#sarah_...	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#sarah_jane_whitfield
http://www.co-ode.org/roberts/family-tree.owl#sarah_...	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#louisa_whitfield_181
http://www.co-ode.org/roberts/family-tree.owl#sarah_...	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#robert_whitfield_181
http://www.co-ode.org/roberts/family-tree.owl#sarah_...	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#sarah_jane_whitfield

Gambar 5.4 Hasil *query* menggunakan inference (transitif)

5.7.2.1.3 Hasil Kasus Pengujian Karakteristik Properti Simetrik

Hasil kasus pengujian karakteristik properti simetrik adalah perbandingan hasil *query* dengan menggunakan *inference* dan yang tidak menggunakan *inference* pada model ontologi yang ada pada basis data Oracle. Kesamaan dalam kedua *query* yaitu predikatnya berupa properti objek yang memiliki karakteristik properti simetrik. Hasil *query* dari kedua *query* dapat dilihat pada Gambar 5.5 dan Gambar 5.6. Perbedaan hasil *query* adalah data triple yang dihasilkan melalui *inference* dengan predikat seperti pada daftar properti objek yang dapat ditunjukan pada Gambar 3.12. Contoh data triple yang hanya ada pada hasil *query* dengan inference adalah seperti yang ditunjukan pada Gambar 5.6. Berdasarkan kedua hasil *query*, dapat dikatakan basis data Oracle mampu melakukan inference dengan properti objek yang karakteristik properti simetrik.



SUB	PRE	OBJ

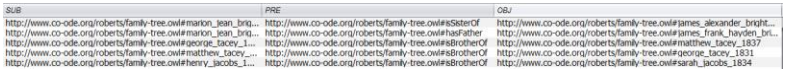
Gambar 5.5 Hasil *query* tanpa menggunakan inference (simetrik)

SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#sarah_jane_whitfield...
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#louisa_whitfield_1817
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#robert_whitfield_1813
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#sarah_jane_whitfield...
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#mary_ann_whitfield...
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#james_whitfield_1821
http://www.co-ode.org/roberts/family-tree.owl#sarah_jewel_1790	http://www.co-ode.org/roberts/family-tree.owl#isRelationOf	http://www.co-ode.org/roberts/family-tree.owl#thomas_william_whit...

Gambar 5.6 Hasil *query* menggunakan inference (simetrik)

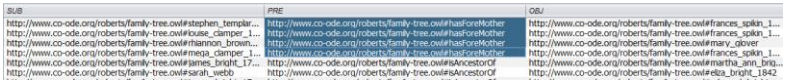
5.7.2.1.4 Hasil Kasus Pengujian Karakteristik Properti *InverseOf*

Hasil kasus pengujian karakteristik properti simetrik adalah perbandingan hasil *query* dengan menggunakan *inference* dan yang tidak menggunakan *inference* pada model ontologi yang ada pada basis data Oracle. Kesamaan dalam kedua *query* yaitu predikatnya berupa properti objek yang memiliki relasi *inverseOf* dengan properti objek yang lain. Hasil *query* dari kedua *query* dapat dilihat pada Gambar 5.7 dan Gambar 5.8. Perbedaan hasil *query* adalah data triple yang dihasilkan melalui *inference* dengan predikat seperti pada daftar properti objek yang dapat ditunjukan pada Gambar 3.14. Contoh data triple yang hanya ada pada hasil *query* dengan inference adalah seperti yang ditunjukan pada Gambar 5.8. Berdasarkan kedua hasil *query*, dapat dikatakan basis data Oracle mampu melakukan inference dengan properti objek yang karakteristik properti simetrik.



SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#marion_jean_briq...	http://www.co-ode.org/roberts/family-tree.owl#isSisterOf	http://www.co-ode.org/roberts/family-tree.owl#james_alexander_bright...
http://www.co-ode.org/roberts/family-tree.owl#marion_jean_briq...	http://www.co-ode.org/roberts/family-tree.owl#hasFather	http://www.co-ode.org/roberts/family-tree.owl#james_frank_hayden_bri...
http://www.co-ode.org/roberts/family-tree.owl#george_tacey_1...	http://www.co-ode.org/roberts/family-tree.owl#isBrotherOf	http://www.co-ode.org/roberts/family-tree.owl#matthew_tacey_1837
http://www.co-ode.org/roberts/family-tree.owl#matthew_tacey_...	http://www.co-ode.org/roberts/family-tree.owl#isBrotherOf	http://www.co-ode.org/roberts/family-tree.owl#george_tacey_1831
http://www.co-ode.org/roberts/family-tree.owl#henry_jacobs_1...	http://www.co-ode.org/roberts/family-tree.owl#isBrotherOf	http://www.co-ode.org/roberts/family-tree.owl#sarah_jacobs_1834

Gambar 5.7 Hasil *query* tanpa menggunakan inference (*inverseOf*)



SUB	PRE	OBJ
http://www.co-ode.org/roberts/family-tree.owl#stephen_templar...	http://www.co-ode.org/roberts/family-tree.owl#hasForeMother	http://www.co-ode.org/roberts/family-tree.owl#frances_spikin_1...
http://www.co-ode.org/roberts/family-tree.owl#louise_clamper_1...	http://www.co-ode.org/roberts/family-tree.owl#hasForeMother	http://www.co-ode.org/roberts/family-tree.owl#frances_spikin_1...
http://www.co-ode.org/roberts/family-tree.owl#rhiannon_brown...	http://www.co-ode.org/roberts/family-tree.owl#hasForeMother	http://www.co-ode.org/roberts/family-tree.owl#mary_glover
http://www.co-ode.org/roberts/family-tree.owl#mega_clamper_1...	http://www.co-ode.org/roberts/family-tree.owl#hasForeMother	http://www.co-ode.org/roberts/family-tree.owl#frances_spikin_1...
http://www.co-ode.org/roberts/family-tree.owl#james_bright_17...	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf	http://www.co-ode.org/roberts/family-tree.owl#martha_ann_briq...
http://www.co-ode.org/roberts/family-tree.owl#sarah_webb	http://www.co-ode.org/roberts/family-tree.owl#isAncestorOf	http://www.co-ode.org/roberts/family-tree.owl#eliza_bright_1842

Gambar 5.8 Hasil *query* menggunakan inference (*inverseOf*)

5.7.2.2 Evaluasi *Usability*

Evaluasi pengujian *Usability* ini dilakukan dengan pengujian aplikasi oleh peserta uji coba yang memahami akan ontologi dan peserta uji coba yang tidak mengerti ontologi.

Pendapat tentang aplikasi ini adalah tampilan aplikasi dapat menjelaskan langkah-langkah yang harus dilakukan untuk menggunakan aplikasi secara tidak langsung urutan tabulasi-tabulasi yang ada pada aplikasi, namun aplikasi akan lebih sulit digunakan jika digunakan bagi pengguna yang kurang paham akan konsep ontologi. Peserta uji coba juga masih kurang mengerti istilah-istilah pada Oracle yang digunakan aplikasi, namun secara keseluruhan, aplikasi ini akan dapat memudahkan pengguna yang memahami konsep ontologi. Adapun saran yang diberikan oleh peserta uji coba yaitu kurangnya informasi tentang Tablespace, model ontologi, format file ontologi, masukkan triple, dan pencarian data ontologi.

5.7.2.3 Evaluasi Waktu *Query*

Evaluasi pengujian waktu eksekusi *query* ini dilakukan untuk menghitung perkiraan waktu eksekuasi yang diperlukan untuk melakukan *inference* pada basis data Oracle. Perhitungan waktu ini menggunakan fitur autotrace yang telah dijelaskan pada subbab 5.6.2. Hasil waktu yang diperoleh pada pengujian ini tidak memungkin hasil waktu sama untuk setiap melakukan *query*. Contoh Laporan autotrace dapat dilihat pada Gambar 5.9, Gambar 5.10, dan Gambar 5.11. Sedangkan waktu yang diperlukan oleh kakas bantu Protégé untuk melakukan inference adalah kurang lebih 130 detik, namun diperlukan waktu sekitar 5 menit untuk melihat data tiap individual. Namun waktu eksekusi *query* protégé tidak dapat dicatat dikarenakan tidak terdapat fitur menghitung waktu eksekusi *query* pada kakas bantu Protégé.

Tabel 5.10 Perbandingan Waktu Eksekusi *Query Oracle dan Protégé*

Query(nama individu)	**Waktu Eksekusi**	
	Protégé	Oracle
Minnie_maud_steward_1909Kode Sumber 5.1	00:20:10.34	00:00:00.34
jean_margaret_archer_1934	00:19:45.12	00:00:00.34

Stieven	00:17:20.56	00:00:00.25

```
SQL> select y,z from table(SEM_Match(
  2  '(:minnie_maud_steward_1909 ?y ?z)',
  3  SEM_Models('ont_fam'),sem_rulebases('RDFS','OWLPRIME'),
  4  SEM_ALIASES(SEM_ALIAS
  5  ('','http://www.co-ode.org/roberts/family-tree.owl#'),
  6  SEM_ALIAS('owl','http://www.w3.org/2002/07/owl#')),null))
  7  ;

616 rows selected.

Elapsed: 00:00:00.34
```

Gambar 5.9 Laporan *autotrace query* individu Minnie_maud_steward_1909

```
SQL> select y,z from table(SEM_Match(
  2  '(:jean_margaret_archer_1934 ?y ?z)',
  3  SEM_Models('ont_fam'),sem_rulebases('RDFS','OWLPRIME'),
  4  SEM_ALIASES(SEM_ALIAS
  5  ('','http://www.co-ode.org/roberts/family-tree.owl#'),
  6  SEM_ALIAS('owl','http://www.w3.org/2002/07/owl#')),null)) ;

602 rows selected.

Elapsed: 00:00:00.34
```

Gambar 5.10 Laporan *autotrace query* individu jean_margaret_archer_1934

```
SQL> select y,z from table(SEM_Match(
  2  '(:stieven ?y ?z)',
  3  SEM_Models('ont_fam'),sem_rulebases('RDFS','OWLPRIME'),
  4  SEM_ALIASES(SEM_ALIAS
  5  ('','http://www.co-ode.org/roberts/family-tree.owl#'),
  6  SEM_ALIAS('owl','http://www.w3.org/2002/07/owl#')),null));

515 rows selected.

Elapsed: 00:00:00.25
```

Gambar 5.11 Laporan *autotrace query* individu stieven

5.8 Evaluasi Pengujian

Berdasarkan hasil pengujian-pengujian tersebut penulis dapat menyimpulkan bahwa aplikasi dapat membantu dalam menggunakan fitur-fitur ontologi pada basis data Oracle dan basis data Oracle dapat digunakan untuk menyimpan data ontologi dan menarik kesimpulan data ontologi dengan menggunakan *inference* secara cepat dan tidak memakan waktu yang lama.

Penulis juga merasa perlunya melakukan pengembangan dan penelitian lebih lanjut dikarenakan basis data yang digunakan adalah Oracle Database 11g release 2, sedangkan produk basis data oracle terakhir pada saat penulisan buku ini adalah Oracle Database 12c. Hal ini dikarenakan perlunya melakukan instalasi Oracle Database 12c dengan edisi *enterprise*. Sedangkan untuk aplikasi yang dibangun, masih belum dapat menjalankan semua prosedur-prosedur yang ada pada basis data Oracle.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari pengerjaan Tugas Akhir ini berdasarkan hasil pengujian dan temuan temuan lainnya. Selain kesimpulan, juga terdapat saran yang penulis ajukan terhadap pengembangan Tugas Akhir ini ke depannya.

6.1 Kesimpulan

Berdasarkan pengerjaan Tugas Akhir ini dan masukan yang didapatan pada tahap uji coba aplikasi, penulis mengambil beberapa kesimpulan sebagai berikut:

1. Untuk menyimpan skema ontologi ke dalam basis data Oracle, skema ontologi dapat disimpan dengan menjalankan *procedure* SEM_APIS.CREATE_SEM_MODEL untuk membuat model ontologi, sesuai dengan nama skema ontologi yang diinginkan, dengan menggunakan *procedure* yang telah dijelaskan pada subbab 2.7.3. x
2. Untuk memodelkan data berbasis ontologi pada basis data Oracle, diperlukan menyimpan data ontologi yang berbentuk *triple* dan menyimpannya dalam tabel yang memiliki kolom dengan tipe data SDO_RDF_TRIPLE_S seperti yang ditunjukkan pada subbab 4.2.
3. Dalam Oracle Semantic dan basis data Oracle, telah disediakan fungsi tabel yang telah teroptimisasi untuk melakukan pencarian data berbasis ontologi pada basis data Oracle sehingga tidak perlu melakukan optimasi untuk pencarian data ontologi pada basis data Oracle.

6.2 Saran

Adapun saran yang penulis berikan untuk pengembangan Tugas Akhir ini lebih lanjut adalah:

1. Dalam merancang perangkat lunak yang sifatnya generik, selain melakukan riset dalam aspek fungsionalitas, seharusnya juga dilakukan riset mengenai aspek antarmuka.
2. Basis data Oracle yang dipakai adalah Oracle Database 11g release 2, sedangkan basis data Oracle yang terbaru adalah Oracle Database 12c, sehingga diperlukan riset lebih lanjut dalam mendalami ontologi pada basis data oracle.

DAFTAR PUSTAKA

[1] Souripriya Das;Eugene Inseok Chong; George Eadon; Jagannathan Srinivasan, "Supporting Ontology-based Semantic Matching in RDBM," in *30 th VLDB Conference*, Toronto, 2004.

[2] C. Candrabiantara, RANCANG BANGUN APLIKASI VISUALISASI SILSILAH KELUARGA BERBASIS ONTOLOGI, Surabaya, 2013.

[3] Azhari and M. Sholichah, "MODEL ONTOLOGI UNTUK INFORMASI JADWAL PENERBANGAN," *JURNAL INFORMATIKA*, vol. 7, pp. 67 - 76, 2006.

[4] J. Gosling, B. Joy, G. Steele, G. Bracha and A. Buckley, "The Java Language Specification," 13 February 2015. [Online]. Available: https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf. [Accessed 21 June 2015].

[5] C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler and M. Smith, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax," [Online]. Available: http://www.w3.org/TR/owl2-syntax/#Data_Property_Axioms. [Accessed 1 April 2015].

[6] "OWL Web Ontology Language," 10 February 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-owl-guide-20040210/#OwlVarieties. [Accessed 12 April 2015].

[7] Oracle, "Oracle Database Semantic Technologies Overview," 2010. [Online]. Available:

http://docs.oracle.com/cd/E18283_01/appdev.112/e11828/sdo_rdf_concepts.htm. [Accessed 20 11 2015].

[8] "INFERENCE," [Online]. Available: http://www.w3.org/standards/semanticweb/inference. [Accessed 9 April 2015].

[9] "Oracle Database Concepts," [Online]. Available: http://docs.oracle.com/database/121/CNCPT/toc.htm. [Accessed 7 April 2015].

[10] "OWL Web Ontology Language Reference," 10 Februari 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/. [Accessed 13 April 2015].

[11] L. Liu and M. T. Özsu, "Ontology," [Online]. Available: http://tomgruber.org/writing/ontology-definition-2007.htm. [Accessed 2 April 2015].

[12] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," [Online]. Available: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html. [Accessed 1 April 2015].

[13] Oracle, "SEM_APIS Package Subprograms," 2010. [Online]. Available: http://docs.oracle.com/cd/E18283_01/appdev.112/e11828/sem_apis_ref.htm. [Accessed 10 11 2015].

[14] Oracle, "Jena Adapter for Oracle," 2010. [Online]. Available: http://docs.oracle.com/cd/E18283_01/appdev.112/e11828/sem_jena.htm. [Accessed 25 12 2015].

BIODATA PENULIS

Stieven Wirakasa, lahir di Surabaya, pada tanggal 18 Juni 1995. Penulis menempuh pendidikan dasar di SD Kristen Petra 13 Sidoarjo (1999-2005), pendidikan menengah pertama di SMP Kristen Petra 3 Surabaya (2005-2008), pendidikan menengah atas di SMA Kristen Petra 2 Surabaya (2008-2011) dan pendidikan tinggi di S1 Teknik Informatika ITS (2011-2016).

Selama masa kuliah, penulis berkesempatan untuk aktif dalam organisasi mahasiswa Himpunan Mahasiswa Teknik Computer (HMTC). Penulis sempat menjadi staff departemen dalam negeri periode 2012-2013. Penulis juga pernah menjadi bagian dalam kegiatan kepanitiaan Schematics sebagai staff divisi keamanan pada Schematics 2012 dan staff divisi kesekretariatan pada Schematics 2013.

Selama kuliah di Teknik Informatika ITS, penulis mendalami bidang minat Rekayasa Perangkat Lunak (RPL). Penulis dapat dihubungi melalui email: **stieven @null.net**.