

ONTOLOGY : BAHASA DAN TOOLS PROTEGE

2.1 Ontology

2.1.1 Definisi Ontology

Pengertian *ontology* sangat beragam dan berubah sesuai dengan berjalannya waktu, ada beberapa definisi *ontology*. Neches dan rekannya [9] memberikan definisi awal tentang *ontology* yaitu: “Sebuah *ontology* merupakan definisi dari pengertian dasar dan relasi vocabulari dari sebuah area sebagaimana aturan dari kombinasi istilah dan relasi untuk mendefinisikan vokabulari”.

Kemudian Gruber [5] memberikan definisi yang sering digunakan oleh beberapa orang, definisi tersebut adalah “*Ontology* merupakan sebuah spesifikasi eksplisit dari konseptualisme”. Berdasarkan definisi Gruber tersebut banyak orang yang mengemukakan definisi tentang *ontology* diantaranya Guarino dan Giaretta [6] yang pada tahun 1995 mengumpulkan hingga tujuh definisi yang berkoresponden dengan *syntactic* dan *semantic* interpretasi. Sedangkan pada tahun 1997, Borst [3] melakukan penambahan dari definisi Gruber dengan mengatakan “Sebuah *ontology* adalah spesifikasi formal dari sebuah konseptual yang diterima (*share*)”.

Kemudian oleh Studer [10] mencoba mengemukakan definisi tentang *ontology* yang mengambil acuan dari definisi yang dikemukakan oleh Gruber dan

Borst, definisi tersebut adalah : “Konseptualisasi mengacu kepada sebuah model abstrak dari beberapa fenomena di dunia dengan memiliki identifikasi konsep yang relevan dari fenomena tersebut. Yang dimaksud dengan eksplisit adalah tipe dari konsep yang digunakan, dan batasan dari eksplisit yang digunakan. *Shared* adalah merefleksikan bahwa sebuah *ontology* mencoba menangkap pengetahuan secara konsensus yang tidak merupakan hal yang hanya terkait pada individu tetapi diterima oleh sebuah group/domain”.

Barnaras [1] pada proyek KACTUS memberikan definisi *ontology* yang berdasarkan pada pengembangan *ontology*. Definisi yang diberikan adalah : “Sebuah *ontology* memberikan pengertian untuk penjelasan secara eksplisit dari konsep terhadap representasi pengetahuan pada sebuah *knowledge base*”. Proyek SENSUS [12] juga memberikan definisi : “Sebuah *ontology* adalah sebuah struktur hirarki dari istilah untuk menjelaskan sebuah domain yang dapat digunakan sebagai landasan untuk sebuah *knowledge base*”.

Ada buku yang memberikan definisi tentang *ontology*, salah satunya adalah “The Semantic Web” [4], definisi dari *ontology* adalah :

1. Salah satu cabang metafisika yang terfokus pada alam dan hubungan antara makhluk hidup;
2. Teori tentang sifat alami makhluk hidup.

Ontology merupakan suatu teori tentang makna dari suatu objek, *property* dari suatu objek, serta relasi objek tersebut yang mungkin terjadi pada suatu domain pengetahuan. Pada tinjauan filsafat, *ontology* adalah studi tentang sesuatu yang ada.

Selain itu ontology adalah sebuah konsep yang secara sistematis menjelaskan tentang segala sesuatu yang ada atau nyata. Dalam bidang *Artificial Intelligence* (AI) *ontology* memiliki dua pengertian yang berkaitan. Pertama *ontology* merupakan kosakata representasi yang sering dikhususkan untuk *domain* atau subjek pembahasan tertentu. Kedua, sebagai suatu *body of knowledge* untuk menjelaskan suatu bahasan tertentu.

Literature yang berisi tentang *Artificial Intelligence* banyak menjelaskan tentang definisi *ontology*, banyak yang bertentangan satu dengan yang lainnya. Tetapi dapat diambil kesimpulan, *ontology* adalah sebuah uraian formal yang menjelaskan tentang sebuah konsep dalam sebuah *domain* tertentu (*Classes*, terkadang disebut *concepts*), *properties* dari masing-masing konsep menjelaskan bermacam-macam corak dan atribut dari sebuah *concept* (*Slots*, terkadang disebut *roles* atau *properties*), dan batasan-batasan (*facets*, terkadang disebut *role restrictions*). Sebuah *ontology* bersama dengan beberapa set *instances* dari *class* membentuk sebuah *knowledge base*.

Secara umum, *ontology* digunakan pada *Artificial Intelligence* (AI) dan persentasi pengetahuan. Segala bidang ilmu yang ada di dunia, dapat menggunakan metode *ontology* untuk dapat berhubungan dan saling berkomunikasi dalam hal pertukaran informasi antara sistem-sistem yang berbeda.

2.1.2 Komponen Ontology

Ontology menggunakan banyak variasi struktur, tergantung dari penggunaan bahasa *ontology* termasuk sintaksis yang digunakan. Perlu diingat adalah *ontology*

tidak melakukan apapun, fungsi perhitungan dan lainnya yang memproses *ontology* tidak hanya tergantung dari data yang terdapat dalam *ontology* tersebut, tetapi juga tergantung kepada aplikasi yang digunakan.

Ontology memiliki beberapa komponen yang dapat menjelaskan *ontology* tersebut, diantaranya [14]:

- Konsep (*Concept*)

Digunakan dalam pemahaman yang luas. Sebuah konsep dapat sesuatu yang dikatakan, sehingga dapat pula merupakan penjelasan dari tugas, fungsi, aksi, strategi, dan sebagainya.

Concept juga dikenal sebagai *classes*, *object* dan *categories*.

- Relasi (*relation*)

Merupakan representasi sebuah tipe dari interaksi antara konsep dari sebuah domain. Secara formal dapat didefinisikan sebagai subset dari sebuah produk dari n set, $R:C_1 \times C_2 \times \dots \times C_n$. Sebagai contoh dari relasi binary termasuk *subclass-of* dan *connected-to*.

- Fungsi (*functions*)

Adalah sebuah relasi khusus dimana elemen ke- n dari relasi adalah unik untuk elemen ke $n-1$. $F:C_1 \times C_2 \times \dots \times C_{n-1} \rightarrow C_n$, contohnya adalah *Mother-of*.

- Aksiom (*axioms*)

Digunakan untuk memodelkan sebuah *sentence* yang selalu benar.

- Instances

Digunakan untuk merepresentasikan elemen.

2.2 Bahasa Ontology

Untuk dapat digunakan, sebuah *ontology* harus diekspresikan dalam notasi yang nyata. Sebuah bahasa *ontology* adalah sebuah bahasa formal dari sebuah pengembangan *ontology*. Beberapa komponen yang menjadi struktur *ontology*, antara lain:

- XML (*Extensible Markup Language*)

Menyediakan sintaksis untuk *output* dokumen terstruktur, tetapi belum dipaksakan untuk dokumen XML menggunakan *semantic constrains*.

- XML *Schema*

Bahasa untuk pembatasan struktur dari dokumen XML.

- RDF (*Resource Description Framework*)

Model data untuk objek (*'resources'*) dan relasi diantaranya, menyediakan *semantic* yang sederhana untuk model data tersebut, dan data model ini dapat disajikan dalam sintaksis XML.

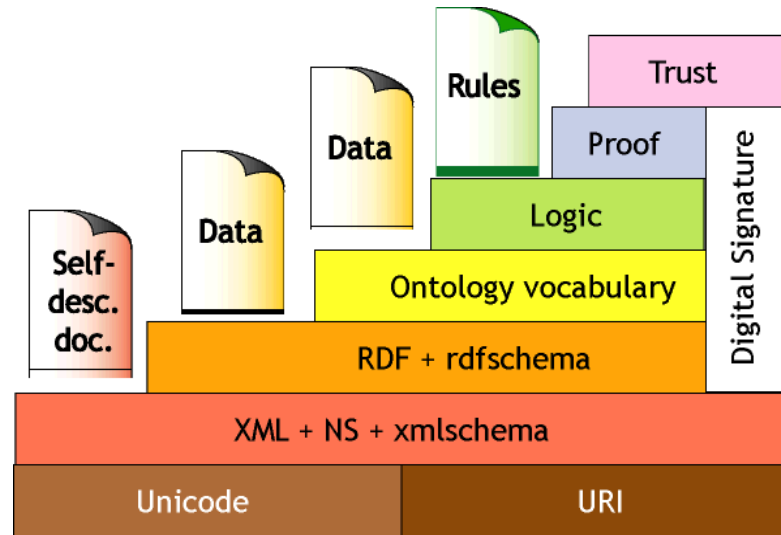
- RDF *Schema*

Adalah kosakata untuk menjelaskan *properties* dan *classes* dari sumber RDF, dengan sebuah *semantics* untuk hirarki penyamarataan dari *properties* dan *classes*.

- OWL (*Ontology Web Language*)

Menambahkan beberapa kosakata untuk menjelaskan *properties* dan *classes*, antara lain : relasi antara *classes* (misalkan *disjointness*), kardinalitas (misalkan ‘tepat satu’), *equality*, berbagai tipe dari *properties*, karakteristik dari *properties* (misalkan *symmetry*), menyebutkan satu persatu *classes*.

Berbagai bahasa yang menyusun *ontology*, seperti yang telah dijelaskan di atas memiliki kedudukan tertentu dalam struktur *ontology*. Struktur layer ontology ditunjukkan seperti gambar 2.1. Setiap layer akan memiliki fungsi tambahan dan kompleksitas tambahan dari layer sebelumnya. Pengguna atau user yang memiliki fungsi pemrosesan layer paling rendah dapat memahami walaupun tidak seluruh *ontology* yang terletak di layer atasnya.



Gambar 2.1 Ontology layer, bersumber dari [11]

Dalam setiap layer tersebut, masing-masing bagian memiliki fungsi masing-masing [7]:

- XML memiliki fungsi menyimpan isi halaman web.
- RDF adalah layer untuk merepresentasikan semantik dari isi halaman tersebut.
- *Ontology layer* untuk menjelaskan *vocabulary* dari domain.
- *Logic Layer* memungkinkan untuk mengambil data yang diinginkan.

2.2.1 XML

Extensible Markup Language (XML) adalah sebuah format teks yang sederhana yang berdasarkan SGML(ISO 8879), yang didesain untuk mempertemukan berbagai macam sumber informasi dalam dunia web.

2.2.1.1 Fungsi dan Tujuan XML

XML sudah banyak dikenal oleh banyak orang, dan adalah dasar untuk pengembangan Software yang meningkat dengan pesat. XML adalah dokumen yang menyimpan data dalam struktur-struktur yang dapat berubah-ubah, hal ini berbeda dengan html yang didesain untuk dokumen hypertext dengan struktur yang baku. Struktur XML yang baik menciptakan struktur yang berbentuk hirarki terstruktur yang memiliki pasangan tags awal dan akhir, yang dapat terdiri dari beberapa atribut yang berpasangan. Tidak ada aturan kosakata tags yang baku atau pasangan tags yang diperbolehkan, jadi hal ini diatur di setiap aplikasi.

2.2.1.2 Sintaksis dan Elemen XML

Sintak dokumen XML yang sederhana terdiri dari deklarasi XML dan elemen puncak. Deklarasi XML merupakan tempat untuk menyatakan Versi dari XML dan encoding untuk dokumen tersebut. Untuk dokumen XML standar versi yang tersedia adalah versi “1.0” dan menggunakan ISO-8859-1 (Latin-1/West European) sebagai encodingnya.

Bagian yang selanjutnya adalah elemen-elemen yang menyusun dokumen XML tersebut. Setiap elemen tersebut memiliki *tags* penutup. *Tags* dalam XML memperhatikan penggunaan huruf atau dalam arti *Case Sensitive*.

Dalam dokumen XML setiap elemen harus tersusun dengan benar. Dalam arti setiap elemen harus benar-benar terkurung. Sebuah dokumen XML harus memiliki

elemen utama. Sedangkan setiap nilai atribut dari elemen tersebut harus menggunakan tanda kutip dua (“ ”). Berikut ini adalah contoh sintak XML yang sederhana :

```
<?xml version="1.0"?> <note date=12/11/99>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

2.2.2 RDF

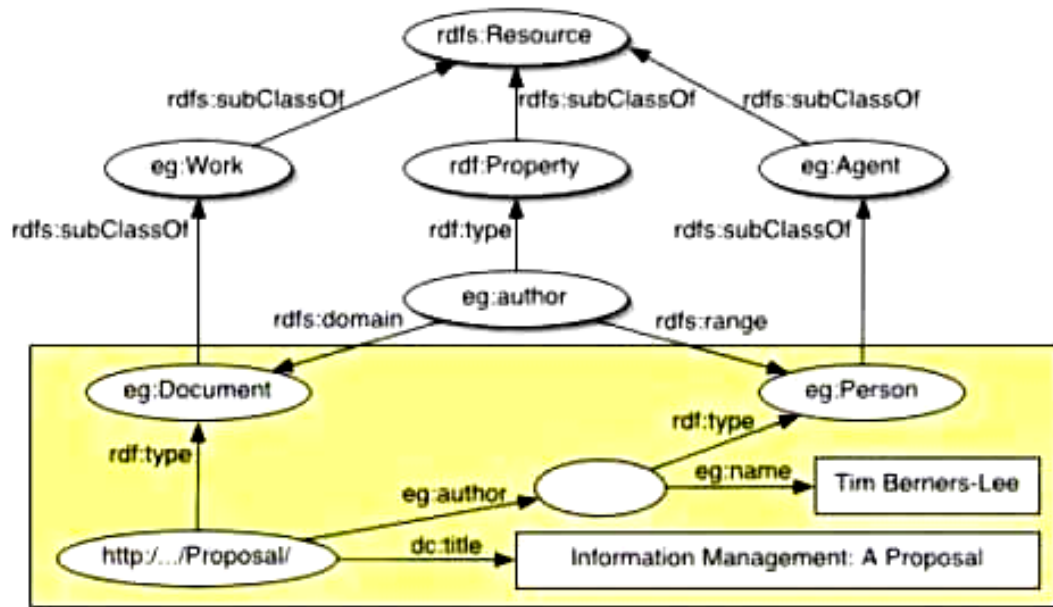
Resources Description Framework (RDF) merupakan sebuah model sederhana untuk mendeskripsikan hubungan antara sumber-sumber daya yang merupakan *properties* dan *values*. *RDF properties* dapat sebagai atribut dari sebuah sumber daya. *RDF properties* dapat merepresentasikan hubungan antara sumber daya. *RDF data model* dapat disusun dari sebuah diagram *entity-relationship*, tetapi ‘tidak’ menyediakan mekanisme untuk mendeskripsikan *propertiesnya* dan ‘tidak’ dapat menyediakan mekanisme untuk menjelaskan hubungan antara *properties* tersebut dengan sumber lain. *RDF vocabulary* menyediakan bahasa untuk mendeskripsikan *classes* dan *properties* yang dapat digunakan untuk menjelaskan *classes* dan *properties* lain.

RDF tidak memaksakan pembatasan *logic* pada *domain* dan *range* dari *properties*. Dalam praktiknya sebuah *properties* dapat diaplikasikan ke dalam dirinya

sendiri. Sebagai contoh, diperbolehkan sebuah *class* ‘*universal*’ untuk menampung *class*-nya sendiri sebagai anggota, secara umum terdapat pada klasifikasi teratas.

Pendeklarasian *properties* (atribut) dan semantic yang berhubungan di definisikan dalam konteks RDF adalah dalam skema RDF. Sebuah skema tidak hanya mendefinisikan *properties* dari sebuah sumber (contohnya: ‘judul’, ‘pengarang’, ‘subjek’, ‘ukuran’) tetapi juga menjelaskan jenis sumber daya yang sedang dijelaskan.

Kosakata RDF yang digunakan sebagai bahasa penggambaran sesuatu, skema RDF, secara khusus digunakan pada model dasar informasi pada RDF atau sebagai sebuah struktur grafik yang menjelaskan sumber daya dan *properties*. Semua kosakata RDF membagi struktur dasar yang sering digunakan, mereka menjelaskan *classes* dari sumber daya dan tipe hubungan antara sumber daya tersebut. Skema RDF yang merupakan kosakata penjelasan memperbolehkan perancang kosakata untuk merepresentasikan *classes* dan *properties* dalam *World Wide Web*. Contoh di bawah ini menggambarkan penggunaan kosakata skema RDF untuk menjelaskan *classes* dan *properties*, dan hubungan ke data pada tingkat aplikasi.



Gambar 2.2 contoh skema RDFS, bersumber dari [13]

Contoh di atas menggambarkan RDF dapat digunakan untuk mendeskripsikan sesuatu yang nyata termasuk *classes* mana mereka, dan *properties* yang digunakan untuk, menghubungkan anggota dari *classes* tersebut.

Bahasa yang digunakan merupakan koleksi dari sumber RDF yang dapat digunakan untuk mendeskripsikan properties dari sumber RDF yang lain yang mendefinisikan kosakata RDF untuk spesifikasi aplikasi. Kosakata inti yang didefinisikan dalam namespace dikenal sebagai '*rdfs*', dan diidentifikasi dengan referensi URI '*http://www.w3.org/2000/01/rdf-schema#*'. Spesifikasi ini juga menggunakan prefik '*rdf*' untuk merujuk ke namespace inti dari RDF '*http://www.w3.org/1999/02/22-rdf-syntax-ns#*'

Sistim *class* dan *properties* pada skema RDF hampir sama dengan bahasa pemrograman tipe *object oriented programming* seperti Java. Tetapi ada beberapa perbedaan antara RDF dengan bahasa pemrograman tersebut yaitu dalam mendefinisikan *class* dalam sebuah *property*. Sebuah skema RDF akan mendefinisikan *properties* dalam *class* mana dia diaplikasikan. Hal tersebut adalah tugas dari *rdfs:domain* dan *rdfs:range*. Sebagai contoh, kita akan mendefinisikan *property* “*author*” akan memiliki *domain* “*Document*” dan sebuah *range* “*Person*”, dimana dalam sistim *Object Oriented* akan didefinisikan sebuah *class* “*book*” dengan sebuah atribut yang dinamakan “*author*” dari tipe “*Person*”. Jika kita menggunakan pendekatan RDF, sangatlah mudah bagi kita untuk menambahkan *property* tambahan dengan *domain* dari dokumen atau *range* dari “*Person*”. Hal tersebut dapat dilakukan tanpa mendefinisikan ulang deskripsi *original* dari *class* tersebut.

Skema RDF dapat mendeskripsikan sebuah hubungan antara kosakata dari skema yang tidak saling berhubungan. Sejak referensi URI digunakan untuk mendefinisikan *class* dan *properties* pada web, sangatlah mungkin bagi kita untuk menciptakan *properties* baru yang mempunyai nilai dari *domain* dan *range* merupakan sebuah *class* yang didefinisikan dari *namespace* lain.

Tabel di bawah ini adalah sebagian dari kosakata dasar dalam RDF, digunakan bersama kosakata-kosakata tambahan dari model RDF dan sintaksis khusus untuk *class* dan *properties*.

Tabel 2.1 Property

Nama property	Pengertian
rdfs:Resource	Class sumber daya
rdfs:Class	Konsep Class
rdf:Property	Konsep dari property
rdfs:Literal	Merepresentasikan bagian dari nilai literal
rdf:Statement	Class dari statement RDF
rdfs:Container	Merepresentasikan penyimpanan
Rdf:Bag	Koleksi yang tidak terstruktur
Rdf:Seq	Koleksi yang terstruktur
Rdf:Alt	Koleksi alternative

Tabel 2.2 Classes

Nama property	Pengertian	Domain	Range
rdfs:isDefinedBy	Nama space dari sumber	rdfs:Resource	rdfs:Resource
rdf:subject	Subjek dari statement RDF	rdf:Statement	rdfs:Resource
rdf:predicate	Predikat dari statement RDF	rdf:Statement	rdf:Property

2.2.3 OWL

OWL adalah bahasa ontology yang baru untuk sebuah semantic web, 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 *Semantic Web W3C*, OWL harus dapat cocok dengan visi *Semantic Web* yaitu bahasa yang dikelompokkan bersama-sama dengan XML dan RDF. OWL yang diharapkan menjadi salah satu bahasa *ontology*, harus dapat merepresentasikan bagian-bagian yang berguna dalam sebuah *ontology*.

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 menyisipkan mereka 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 kedalam 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*, *asymmetric*, fungsional atau bertolak belakang dengan *property* lainnya.

OWL dapat mengekspresikan objek (dapat juga disebut '*individu*') mana yang dimiliki oleh *class* yang mana, dan apa nilai *property* untuk sebuah individu. Pernyataan yang sama dapat dibuat pada *class* dan *property*, pernyataan *disjoint* dapat dibuat pada *class*, dan *equality* dan *inequality* dapat juga disisipkan antara individu.

Kemampuan OWL yang lebih dari RDF, adalah kemampuannya untuk menyediakan pembatasan pada bagaimana posisi *property* terhadap *class*. OWL dapat mendefinisikan *class* mana yang mempunyai *property* terbatas yang membuat semua nilai untuk *property* tersebut, sehingga semua nilai untuk *property* dalam *instances* harus dimiliki oleh *class* tertentu (atau tipe data); setidaknya satu nilai harus datang dari *class* tertentu (atau tipe data); setidaknya terdapat nilai tertentu; dan harus setidaknya satu atau lebih angka tertentu yang berbeda dari sebuah nilai, sebagai contoh dengan menggunakan skema RDF, kita dapat:

- Mendeklarasikan *class*, seperti 'negara', 'orang', 'mobil';
- Menyatakan bahwa 'murid' adalah *sub-class* dari 'orang';
- Menyatakan bahwa 'Indonesia' dan 'Jerman' adalah anggota *class* 'negara';
- Mendeklarasikan 'bangsa' sebagai *property* yang menghubungkan *class* 'orang' (sebagai domain) dan *class* 'negara' (sebagai range);
- Menyatakan bahwa 'umur' adalah *property*, dengan 'orang' sebagai *domain* dan *integer* sebagai *range*;
- Menyatakan bahwa 'Budi' sebagai anggota dari *class* 'Indonesia', dan 'umur' yang dimilikinya memiliki nilai '48'.

Dengan OWL kita juga dapat:

- Menyatakan bahwa ‘negara’ dan ‘orang’ adalah *class* yang *disjoint*;
- Menyatakan bahwa ‘Canada’ dan ‘Jerman’ adalah *individu* yang berbeda;
- Mendeklarasikan ‘memiliki warga’ sebagai kebalikan dari *property* ‘bangsa’;
- Menyatakan bahwa *class* ‘tidak ada negara’ dibuat untuk semua anggota dari *class* ‘orang’ yang tidak memiliki nilai untuk *property* ‘bangsa’.

OWL mempunyai suatu penukaran sintaksis RDF/XML dan suatu abstrak sintaksis seperti frame, dan OWL telah memiliki tiga nama *sub languages*. Keanekaragaman ini adalah hasil berusaha langsung untuk mencukupi sejumlah besar kebutuhan yang berlawanan dan pengaruh.

2.2.3.1 Sub Bahasa OWL

OWL menyediakan tiga sub bahasa yang berbeda tingkatan bahasanya yang dirancang untuk berbagai kebutuhan tertentu dari pengguna, antara lain :

- *OWL Lite*

Mendukung pengguna yang memerlukan hirarki klasifikasi dan batasan yang sederhana. Sebagai contoh, hanya mendukung batasan *Cardinality*, dengan nilai untuk *Cardinality* sebesar 0 atau 1. Nilainya haruslah lebih sederhana untuk menyediakan alat pendukung untuk *OWL Lite* dibanding yang lebih ekspresif, dan *OWL Lite* menyediakan suatu alur migrasi cepat untuk thesauri dan taksonomi lain. *OWL Lite* juga mempunyai suatu kompleksitas formal

yang lebih rendah dibanding *OWL DL*. *OWL Lite* didesain untuk kemudahan penerapan dan untuk memberikan user dengan fungsi subset yang akan membawa mereka ke permulaan dalam penggunaan OWL.

- *OWL DL (Description Logic)*

Mendukung pengguna yang menginginkan ekspresi maksimum tanpa kehilangan perhitungan yang lengkap (semua kesimpulan dijamin menjadi dapat diperhitungkan) dan ketepatan (semua perhitungan akan diselesaikan dalam waktu tertentu). *OWL DL* meliputi semua bahasa konstruksi dalam OWL, tetapi mereka dapat digunakan hanya dibawah batasan tertentu (sebagai contoh, selama suatu *class* menjadi suatu subclass dari banyak kelas, suatu kelas tidak bisa menjadi suatu *instance* dari kelas yang lain). *OWL DL* diberi nama dalam kaitan dengan *Description Logic*, suatu bidang riset yang telah belajar logika yang membentuk pondasi bagi yang formal pada OWL. *OWL DL* memang dirancang untuk mendukung deskripsi logical dari segmen bisnis dan untuk menyediakan bahasa subset yang memberikan *property* untuk perhitungan dalam sebuah system.

- *OWL Full*

Sangat berguna untuk pengguna yang menginginkan ekspresi maksimum dan kebebasan sintaksis dari RDF tanpa ada jaminan perhitungan. Sebagai contoh, dalam *OWL Full* sebuah class dapat diperlakukan berturut-turut sebagai kumpulan individu dan sebagai individu dengan haknya sendiri. *OWL Full*

memperbolehkan *ontology* untuk meningkatkan arti dari kosakata yang belum digambarkan (*RDF* atau *OWL*). Tidaklah mungkin untuk semua pembuat software untuk mendukung secara penuh kemampuan yang ada pada *OWL Full*.

OWL Full dan *OWL DL* menyediakan konstruksi bahasa OWL yang sama. Perbedaan mereka hanya terletak pada pembatasan dalam penggunaan fungsi dan dalam penggunaan fungsi dalam RDF. *OWL Full* memperbolehkan pencampuran dari OWL dengan skema RDF dan, seperti skema RDF, tidak memaksakan peraturan yang memisahkan *classes*, *properties*, *individuals* dan nilai data. *OWL DL* memberikan pembatasan dalam penggabungan dengan RDF dan membutuhkan *disjoint* dari *classes*, *properties*, *individuals*, dan nilai data. *OWL Lite* adalah sub bahasa dari *OWL DL* yang hanya mendukung subset dari konstruksi bahasa OWL. *OWL Lite* secara umum ditujukan sebagai alat untuk membangun sebuah *ontology* bagi orang-orang yang ingin menggunakan OWL, tetapi ingin memulainya dengan struktur atau fungsi bahasa yang sederhana.

Pengembangan *ontology* yang mengadopsi OWL harus menentukan sub bahasa mana yang tepat untuk kebutuhannya. Pemilihan antara *OWL Lite* dan *OWL DL* tergantung kebutuhan dalam penggunaannya oleh user yang membutuhkan konstruksi yang lebih ekspresif di sediakan oleh *OWL DL*. Pemilihan antara *OWL DL* dan *OWL Full* tergantung juga kepada kebutuhan dalam penggunaannya yang user membutuhkan fasilitas *meta-modeling* dari skema RDF (contohnya, mendefinisikan *class* dari *class*, atau mengaitkan *properties* dengan *class*).

OWL Full dapat dipandang sebagai ekstensi dari *RDF*, *OWL Lite* dan *OWL DL* dapat dipandang sebagai ekstensi dari tampilan yang terbatas dari *RDF*. Setiap dokumentasi *OWL (Lite, DL Full)* adalah dokumen *RDF*, dan setiap dokumen *RDF* adalah dokumen *OWL Full*, tetapi hanya sebagian dokumen *RDF* akan menjadi dokumen *OWL Lite* atau *OWL DL*. Karena hal tersebut, perhatian yang harus lebih diberikan user yang ingin bermigrasi dari dokumen *RDF* ke *OWL*.

2.2.3.2 Rancangan OWL

Untuk berbagai alasan, pendiskripsian dalam bagian proses, terdapat dua tipe penggunaan *OWL*. Tipe yang pertama, bagian dari *OWL DL* dan *OWL Lite*, hanya konstruksi tertentu yang diperbolehkan untuk dideskripsikan, dan konstruksi tersebut hanya dapat dikombinasikan dengan cara-cara tertentu. Keunggulan dalam pembatasan ini meliputi menentukan kesimpulan dan kemungkinan berfikir dari *OWL* di dalam suatu lingkungan yang standar, utamanya sebagai suatu ekspresi *Description Logic*. Didalam tipe yang kedua, yang terdapat dalam *OWL Full*, semua grafik *RDF* diijinkan. Kelebihan tipe ini meliputi total peningkatan kecocokan dengan *RDF* dan suatu ekspresif yang lebih besar.

Walaupun versi terbatas dari *OWL* memiliki bebrapa perbedaan dari standar *Description Logic*. Perbedaan ini memindahkan versi *OWL* tersebut dari dunia *Description Logic* ke dunia *Semantic Web*. Beberapa penjelasan tentang *OWL*:

- OWL menggunakan referensi URI sebagai nama, dan membangun URI tersebut menggunakan kondisi yang sama yang digunakan dalam RDF. Alasan inilah dalam OWL untuk menggunakan nama yang singkat untuk referensi URI, contohnya menggunakan '*owl:Thing*' untuk referensi URI;
- OWL mengumpulkan semua informasi ke dalam *ontology*, yang secara umum disimpan sebagai dokumen Web yang ditulis dalam *RDF/XML*. *Ontology* dapat memasukkan *ontology* lain, menambahkan informasi dari *ontology* lain ke dalam *ontology* sendiri.
- Walaupun tipe *DL/Lite* dalam menggunakan OWL memperbolehkan catatan *property* RDF digunakan untuk meyisipkan informasi ke dalam *class*, *properties*, dan *ontology* seperti '*owl:DeprecatedClass*'. Penyisipan ini adalah *RDF triples*, dan hal tersebut digunakan dalam *semantic* yang lebih besar. Mereka tidak dapat diperlakukan sebagai komentar tanpa ada arti sesungguhnya. Hal ini memecahkan perbedaan dalam *Description Logic* antara *individual* dan *classes* dan *properties*;
- OWL menggunakan fasilitas dari tipe data RDF dan skema tipe data XML untuk menyediakan tipe data dan nilai data;
- Versi DL dan Lite dari OWL memiliki frame seperti sintaksis abstrak, walaupun RDF/XML adalah sintaksis pengganti yang resmi untuk semua OWL.

2.2.3.3 Sintaksis OWL

Sebuah OWL ontology adalah sebuah grafik RDF, yang berbentuk set dari RDF Triples. Seperti grafik RDF, grafik OWL ontology dapat ditulis dalam berbagai macam bentuk sintaksis yang berbeda. Arti dari OWL ontology dapat digambarkan dengan menggunakan grafik RDF. Tetapi dapat dimungkinkan untuk menggunakan berbagai bentuk sintaksis RDF/XML yang berbeda, selama hasil yang dikeluarkan memiliki kecocokan dengan set dari RDF Triples. Berikut ini adalah sedikit contoh sintaksis dalam OWL yang memiliki arti sama dengan yang dibuat dengan menggunakan RDF Triples:

```
<owl:class rdf:ID="Continent"/>
```

Sintak dalam RDF/XML:

```
<rdf:Description rdf:about="#Continent">
```

```
<rdf:type
```

```
rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
```

```
<rdf:Description >
```

Jika kedua coding tersebut di encode, maka akan dibangkitkan arti yang sama. Contoh sederhana dokumen OWL adalah :

```
<owl:Class>
```

```
<owl:oneOf rdf:parseType ="Collection">
```

```
<owl:Thing rdf:about ="#Eurasia"/>
```

```
<owl:Thing rdf:about ="#Africa"/>
```

```
<owl:Thing rdf:about ="#NorthAmerica"/>
```

```

    <owl:Thing rdf:about="#SouthAmerica"/>
    <owl:Thing rdf:about="#Australia"/>
    <owl:Thing rdf:about="#Antartica"/>
  </owl:oneOf>
</owl:Class>

```

2.3 Protégé

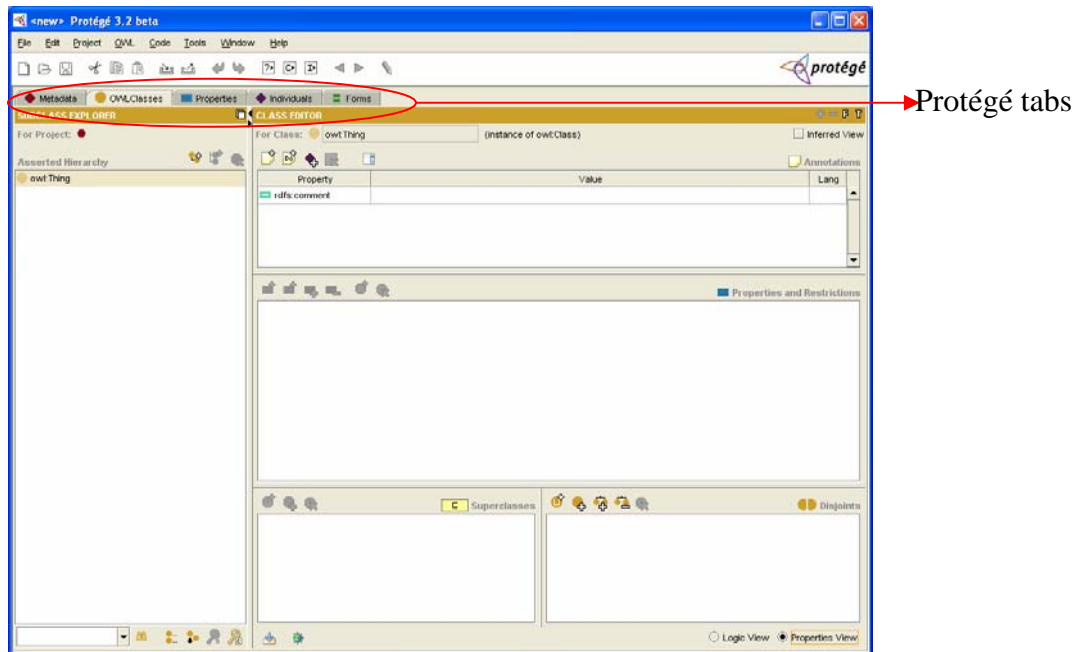
Protégé adalah perangkat lunak bantu yang digunakan untuk pengembangan sistem berikut *Knowledge Base System*. Aplikasi yang dikembangkan oleh *protégé* digunakan dalam pemecahan masalah dan pembuatan keputusan dalam sebuah domain. *Protégé* dikembangkan oleh sebuah organisasi yang bernaung di bawah Stanford, yang mengambil spesialisasi dibidang *ontology*.

Protégé merupakan sebuah alat yang digunakan untuk membuat sebuah domain *ontology*, menyesuaikan form untuk entry 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 [8].

Perangkat lunak *protégé* menyediakan konsepsi dasar pengetahuan yang terintegrasi, serta mengubah tampilan visual lingkungan dengan memperluas arsitektur sistem untuk membuat pemodelan dasar pengetahuan secara lebih sederhana dan mudah. *Protégé* dapat juga digunakan dengan tujuan berikut: membangun *ontology*, memodelkan tampilan pengetahuan akuisisi dan memasukkan domain pengetahuan. *Protégé* memvisualisasikan hubungan subclass dalam tree, mendukung berbagai penurunan (*multiple inheritance*) dan root pada hirarki *class* yang terbentuk adalah *class* “THING”.

Untuk mendapatkan *protégé* dapat dilakukan dengan cara men-Download dari web penyedia tool, alamat web tersebut adalah <http://Protege.stanford.edu/>. Ukuran file instalasi untuk *protégé* tergantung pada versi yang diinginkan dan juga tergantung termasuk atau tidaknya SDK Java. *Protégé* membutuhkan SDK Java, terdapat dua pilihan yaitu apakah SDK Java termasuk ke dalam file instalasi atau tidak. *Protégé* dapat berjalan di berbagai platform operating system, antara lain Windows, Mac OS, Solaris, Linux, HP-UX, Unix, AIX. *Protégé* dapat membuka berbagai macam format file, ada tiga format file umum yang dapat dibuka dengan *protégé*, yaitu XML, RDF dan OWL. Untuk dapat membuka format file tersebut hal yang perlu dilakukan adalah dengan membuat sebuah project baru pada *protégé*, project tersebut akan memiliki format file .pprj.



Gambar 2.3 Protégé 3.2 beta

DAFTAR PUSTAKA

- [1]. A Barnaras, L Laresgoiti, and J Corera. Building and Reusing Ontologies for Electrical Network Application. In *12th European Conference on Artificial Intelligence*, pages 298-302, 1996.
- [2]. V Richard Benjamins and Assunción Gómez-Pérez. Knowledge System Technology: Ontologies and Problem-Solving Methods. 5 2004. <www.swi.psy.uva.nl/usr/richard/pdf/kais.pdf>.

- [3]. Willem Nico Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, University of Twente, Netherland, 5 September 1997. SIKS The Dutch Graduate School.

- [4]. Michael C Daconta, Leo J Obrst, and Kevin T Smith. *A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing, Indianapolis, Indiana, 2003.

- [5]. T Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *Int. Journal of Human-Computer Studies*, 43:907-928, 1995.

- [6]. N. Guarino and P. Giaretta. *Ontologies and Knowledge Bases: Towards a Terminological Clarification*, chapter Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, pages 25-32. IOS Press, Amsterdam, 1995.

- [7]. Vladimir Kolovski and John Galletly. Towards E-Learning via the Semantic Web. In *International Conference on Computer Systems and Technologies-CompSysTech'2003*, page 2, 2003.

- [8]. Protege. <http://protege.stanford.edu/>, 2005.

- [9]. T. Finin T. R. Gruber T. Senator R. Neches, R. E. Fikes and W. R. Swartout. Enabling Technology for Knowledge Sharing. 1991. AI Magazine.
- [10]. V. R. Benjamins R. Studer and D. Fensel. *Knowledge Engineering, Principles and Methods.*, chapter Data and Knowledge Engineering, pages 25(1-2):161-197. 1998.
- [11]. York Sure and Rudi Studer. Towards the Semantic Web: Ontology driven Knowledge Management, 2003.
- [12]. K. Knight W. Swartout, R. Patil and T. Russ. *Toward Distributed use of large-scale Ontologies.*, chapter Spring Symposium Series on Ontological Engineeringg, pages 33_40. AAAI Press, 1997.
- [13]. W3C. <http://www.w3.org/tr/2002/wd-rdf-schema-20020430/>, 2 2006.
- [14]. I Wayan Simri Wicaksana. Survei dan Evaluasi Metode Pengembangan Ontologi (Survey and Evaluation of Methodology of Ontology Development). In *Proc. of KOMMIT 2004*, Jakarta&Depok, 24 2004. University Gunadarma.