

BAB 3

TEKNOLOGI SEMANTIC WEB

Bab ini berisi landasan teori yang digunakan penulis untuk mendukung penerapan teknologi dalam penelitian ini. Hal-hal yang akan dibahas meliputi konsep *semantic web*, model ontologi, serta *semantic portal* sebagai salah satu bentuk implementasi *semantic web*.

3.1 Pengenalan Semantic Web

Tim Barners-Lee, penemu *World Wide Web*, menyatakan: “Langkah yang pertama adalah meletakkan data pada *web* dalam suatu bentuk sehingga mesin dapat secara alami memahami, atau mengubahnya menjadi format tertentu. Pembuatan ini yang kita sebut suatu *Semantic Web*—suatu data *web* yang dapat diproses secara langsung atau secara tidak langsung oleh mesin” [16]. Hingga saat ini, *semantic web* masih merupakan visi jangka panjang. Penerapan *semantic web* sendiri belum sampai tahap akhir dan akan terus mengalami perkembangan.

3.1.1 Visi Semantic Web

Tim Berners-Lee mempunyai dua bagian visi untuk masa depan *web*. Bagian pertama adalah untuk membuat *web* sebagai suatu media yang lebih kolaboratif. Bagian yang kedua adalah untuk membuat *web* dapat dimengerti, sehingga bisa diproses, oleh mesin [16].

Berdasarkan visi tersebut, *web* sekarang ini belum sesuai seperti yang diharapkan pertama kali pada tahun 1989 [22]. Walaupun sejak tahun 1994 internet terus berkembang dengan cukup pesat hingga saat ini, namun *web page* saat ini belum dapat diproses secara otomatis oleh mesin komputer. *Web page* yang ada saat ini masih bersifat *human-readable*, sehingga informasi yang ditampilkan hanya dapat dipahami oleh manusia.

Pada intinya, visi dari *semantic web* adalah menjadikan informasi pada *web* bersifat *machine-readable*. Berners-Lee mendefinisikan *semantic web* sebagai

berikut: “*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*” [16].

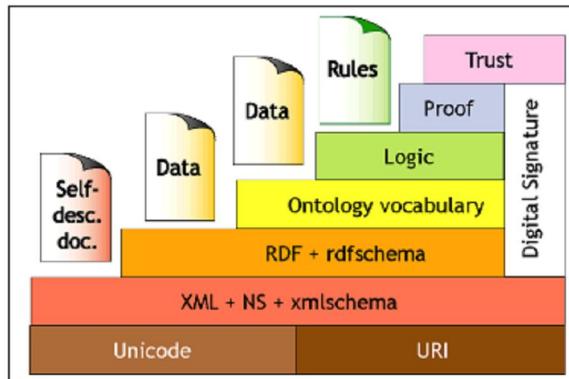
Dari definisi tersebut, dapat disimpulkan bahwa *semantic web* dibuat bukan untuk menggantikan *web* yang ada sekarang, namun untuk memperkaya dalam hal penyajian informasi, sehingga memungkinkan komputer dalam memahami informasi tersebut. Motivasi utamanya yaitu kemampuan mesin komputer untuk melakukan proses secara otomatis, lebih efektif, dan efisien. Di tengah banyaknya informasi yang tersebar pada *web*, kehadiran *software agent* yang dapat memahami informasi laiknya manusia sangat bisa membantu pengguna menemukan informasi yang tepat dan berguna untuk ditampilkan.

Transformasi *web* menuju *semantic web* merupakan suatu pergeseran paradigma dari prinsip *web* sebagai dokumen menjadi data. Informasi dalam *web* dapat diproses karena memiliki representasi yang lebih ‘cerdas’, sehingga keunggulan *software* dinilai bukan dari proses yang dilakukan melainkan dari kualitas datanya. Sedemikian pentingnya data hingga dapat dipandang sebagai raja (*data as the king*) atau sebagai *first class citizen* dalam *semantic web* [16].

Disebutkan dalam beberapa artikel bahwa *semantic web* akan menjadi generasi selanjutnya dari proses perkembangan *web* selama ini, atau disebut dengan Web 3.0. Pada level ini, *web* diharapkan memiliki kemampuan untuk menganalisis dan memproses data serta menemukan *link* di antara data tersebut. Salah satu keunggulan dari penerapan *semantic web* yaitu integrasi data yang dapat dilakukan dari berbagai sumber dan format yang berbeda [52].

3.1.2 Arsitektur *Semantic Web*

Teknologi *semantic web* terbagi dalam beberapa *layer* arsitektur seperti yang ditampilkan pada Gambar 3.1.



Gambar 3.1 *Semantic Web Layer* [4]

Teknologi yang sering disebut sebagai *layer cake technology* ini terdiri atas [4]:

- Unicode dan URI: Unicode adalah standar representasi karakter komputer. URI (*Unified Resources Identifier*) merupakan standar untuk lokasi dan identitas suatu *resource* (misalnya *web page*).
- XML + NS + XML Schema: merupakan aturan sintaks yang berfungsi untuk menyajikan struktur data pada *web*.
- RDF + RDF Schema: RDF merupakan model berbentuk *graph* yang merepresentasikan *resources* dan relasinya. Sedangkan RDF Schema adalah definisi kosakata yang digunakan pada RDF.
- *Ontology vocabulary*: bahasa yang digunakan untuk mendeskripsikan *resources*. W3C merekomendasikan OWL *Web Ontology Language*, suatu bahasa yang lebih kaya dan kompleks, untuk membangun ontologi.
- *Logic* dan *Proof Layer*: berupa *rules* dan sistem untuk melakukan *reasoning* pada ontologi sehingga dapat disimpulkan apakah suatu *resource* memenuhi syarat tertentu.
- *Trust*: *layer* terakhir yang memungkinkan pengguna *web* untuk mempercayai suatu informasi pada *web*.

3.2 Ontologi

Istilah teknologi *semantic web* seringkali disamakan dengan teknologi berbasis ontologi. Hal ini mungkin karena tidak ada batasan jelas yang memisahkan keduanya. Ontologi memang menjadi *backbone* dari teknologi ini, sehingga dapat

disimpulkan bahwa *semantic web* merupakan aplikasi terkini dari ontologi, meskipun sebenarnya penerapan ontologi tidak terbatas pada bentuk aplikasi *web-based*. Ontologi bahkan sering digunakan untuk menggantikan istilah *semantic model* yang menjadi kerangka dalam pengembangan *semantic web*.

3.2.1 Definisi Ontologi

Istilah ontologi awalnya hadir di area ilmu filsafat. Ontologi didefinisikan sebagai studi tentang sesuatu yang ada atau sebuah konsep yang secara sistematis menjelaskan tentang segala sesuatu yang ada atau nyata. Namun pada perkembangan selanjutnya, para peneliti di bidang *Artificial Intelligence* mengadaptasi istilah ontologi dan memaknainya dari sudut pandang yang berbeda. Mereka memberi konsep yang lebih spesifik untuk disiplin ilmu komputer mengenai ontologi, yaitu "*a shared and common understanding of some domain that can be communicated between people and application systems*"[25].

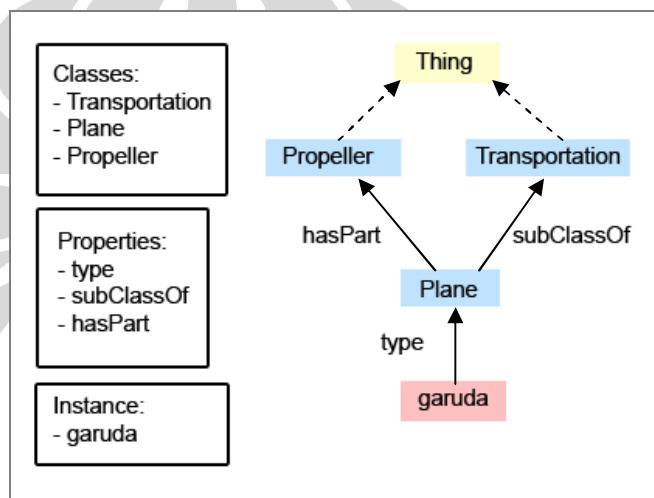
Terdapat berbagai macam pengertian tentang ontologi yang dijelaskan dalam berbagai buku, termasuk yang dikemukakan oleh beberapa ilmuwan. Neches dkk memberikan definisi awal tentang ontologi yaitu "sebuah ontologi merupakan definisi dari pengertian dasar dan relasi vokabulari dari sebuah area sebagaimana aturan dari kombinasi istilah dan relasi untuk mendefinisikan vokabulari" [53]. Sedangkan Barnaras [8] berpendapat bahwa "sebuah ontologi memberikan pengertian untuk penjelasan secara eksplisit dari konsep terhadap representasi pengetahuan pada sebuah *knowledge base*". Ontologi merupakan suatu teori tentang makna dari suatu objek, properti dari suatu objek, serta relasi objek tersebut yang mungkin terjadi pada suatu domain pengetahuan.

Dalam bidang ilmu komputer, umumnya ontologi digunakan pada *Artificial Intelligence* (AI) dan representasi pengetahuan. Segala bidang ilmu yang ada di dunia, dapat menggunakan metode ontologi untuk dapat berhubungan dan saling berkomunikasi dalam hal pertukaran informasi antara sistem yang berbeda.

3.2.2 Representasi Ontologi

Menurut Daconta, secara teknis sebuah ontologi direpresentasikan dengan beberapa komponen [17], yaitu:

- Classes*, atau *concept, general things* pada suatu *domain of interest*.
- Instances*, atau *individual, particular things*.
- Properties* dan nilainya dari *things* tersebut.
- Constraints* dan *rules* untuk *things* tersebut.
- Relationships* di antara *things* tersebut.
- Functions* dan *processes* yang melibatkan *things* tersebut.



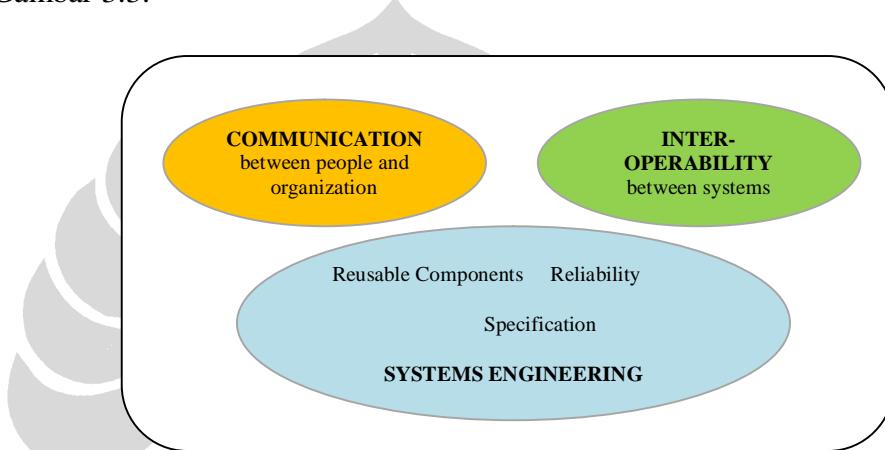
Gambar 3.2 Contoh Ontologi

Gambar 3.2 merupakan contoh ontologi sederhana yang memodelkan konsep mengenai pesawat terbang. Pada gambar tersebut terdapat konsep tentang Thing, Transportation, Plane, dan Propeller. Pada contoh tersebut dapat dilihat bahwa Thing pada domain yang dimaksud terdiri dari Transportation, Plane, dan Propeller. Hubungan hierarkis antara Plane dan Transportation dikenal sebagai relasi *is-a* (Plane *is-a* Transportation). Selain itu, Plane memiliki properti *hasPart* yang merelasikannya dengan Propeller (Plane *hasPart* Propeller). Garuda merupakan contoh *instance* atau objek dari Plane.

Dalam *semantic modeling*, ontologi direpresentasikan dengan bahasa yang terstandardisasi yaitu RDF, RDFS, atau OWL. Penjelasan masing-masing bahasa tersebut akan diberikan pada subbab selanjutnya.

3.2.3 Kegunaan Ontologi

Kegunaan ontologi secara umum antara lain sebagai *controlled vocabulary*, *semantic interoperability*, *knowledge sharing*, dan *reuse*. Uschold [49] memaparkan tiga kategori kegunaan ontologi, seperti yang diperlihatkan pada Gambar 3.3.



Gambar 3.3 Kegunaan Ontologi

Berikut penjelasan dari ketiga kategori tersebut.

- *Communication*

Dalam komunikasi, perlu adanya representasi semantik agar informasi dapat disampaikan secara tepat. Ontologi bermanfaat untuk memfasilitasi komunikasi antar manusia dalam suatu organisasi dengan menyediakan modek konsep sehingga memungkinkan *shared understanding*. Model dibuat dengan menyusun definisi, kosakata, dan terminologi yang secara informal merepresentasikan semantik. Penggunaan bentuk visual seperti diagram *Entity Relationship* (ER) dan *Unified Modeling Language* (UML) juga sangat efektif dalam komunikasi antar manusia untuk memahami suatu sistem. Selain itu, komunikasi antar *software agent* atau *intelligent agent* dalam sistem AI (*Artificial Intelligence*) juga dapat memanfaatkan ontologi, sesuai format pesan yang ditentukan.

- *Interoperability*

Ontologi juga sangat bermanfaat untuk integrasi sistem yang sudah ada, terlebih lagi untuk aplikasi sistem terdistribusi. Hal ini dimungkinkan apabila ontologi yang digunakan sama sehingga dapat terjadi pertukaran informasi antar sistem dengan mudah. Keragaman informasi di level sintaksis dan struktur dapat diatasi dengan ontologi yang berada pada level semantik sebagai format standar data.

- *System Engineering*

Dalam perancangan dan pengembangan sistem, ontologi berguna untuk membuat spesifikasi sistem yang terdiri atas komponen dan relasinya. Ontologi juga dapat meningkatkan realibilitas sistem dengan menjadikannya sebagai standar untuk memeriksa konsistensi *software* terhadap spesifikasi yang telah dibuat.

3.2.4 Proses Pengembangan Ontologi

Terdapat berbagai pendapat mengenai tahapan proses pengembangan sebuah model ontologi. Menurut Cristani, tahapan yang dilakukan dalam proses pengembangan ontologi yaitu [14]:

- a. Tahap Penentuan Domain

Tahap ini merupakan proses awal digitalisasi pengetahuan yang dilakukan dengan cara menjawab beberapa pertanyaan seperti: Apa yang merupakan domain ontologi? Mengapa harus menggunakan ontologi? Apa jenis pertanyaan terhadap ontologi sehingga perlu menyediakan jawaban? Siapa yang akan menggunakan dan memelihara ontologi?

- b. Tahap Penggunaan Ulang

Tahap ini adalah tahap penggunaan kembali dan justifikasi dari ontologi yang telah dibangun. Hal ini bisa terjadi ketika terdapat kebutuhan untuk menghubungkan sistem dengan aplikasi lain yang juga menggunakan ontologi yang sama.

- c. Tahap Penentuan Istilah pada Ontologi

Tahap untuk menentukan istilah-istilah yang digunakan untuk membuat pernyataan atau untuk menjelaskan hal yang sama.

d. Tahap Pendefinisian *Class* dan Hierarki *Class*

Menciptakan beberapa definisi dari konsep dalam hierarki dan kemudian menguraikan properti dari konsep. Hierarki *class* direpresentasikan dengan relasi *is-a*, yang berarti bahwa setiap kelas A adalah *subclass* B jika setiap *instance* dari *class* A juga *instance* dari *class* B.

e. Tahap Pendefinisian *Properties*

Secara umum, ada beberapa jenis *property* objek dalam suatu ontologi: *property* intrinsik (hakiki) seperti rasa dari anggur; *property* ekstrinsik (karena keadaan luar), seperti nama anggur dan area (regional); *property* yang menyatakan hubungan suatu individu dengan individu lain (dalam *class* yang sama atau berbeda).

f. Tahap Pendefinisian *Constraints*

Property dapat memiliki syarat (*constraint*) berupa kardinalitas tunggal (satu nilai) atau kardinalitas banyak (memiliki sejumlah nilai). Kardinalitas dari N berarti suatu *property* mempunyai sedikitnya N nilai. Nilai dari *property* dapat berupa *string*, *boolean*, enumerasi (simbolik), serta *instance*.

g. Tahap Pembuatan *Instance*

Pendefinisian sebuah *instance* dari *class* dapat meliputi pemilihan *class*, pembuatan individu *instance* dari *class*, dan pengisian nilai *property*.

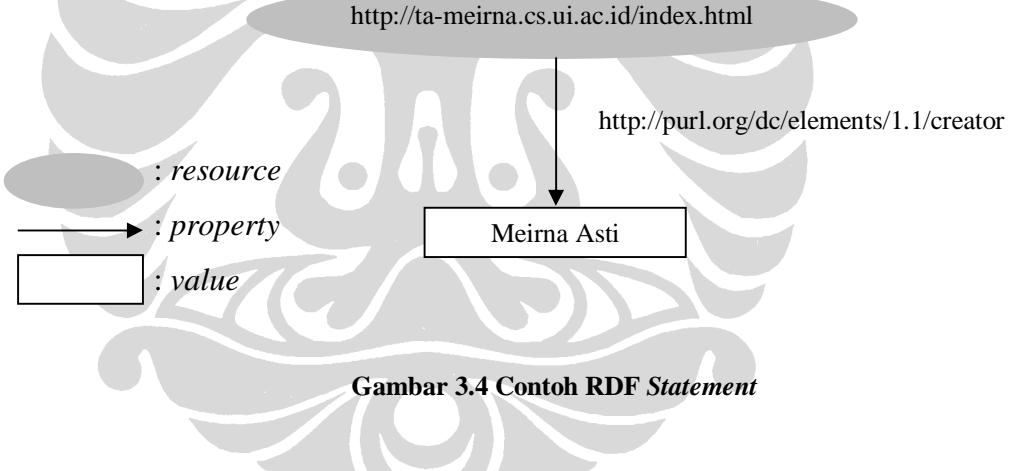
3.3 RDF (*Resources Description Framework*)

RDF (*Resource Description Framework*) merupakan standar yang digunakan untuk mendeskripsikan *resource*. Secara umum, *resource* adalah sesuatu yang ingin dibicarakan, sesuatu yang dapat diidentifikasi, misalnya *web site*, *homepage*, orang, benda, dan sebagainya. RDF berbeda dengan HTML (*Hypertext Markup Language*) yang digunakan untuk menampilkan informasi dan XML (*Extensible Markup Language*) yang berfungsi dalam pertukaran informasi, RDF merupakan model data yang digunakan untuk menjelaskan informasi.

Selama ini penggunaan *metadata* untuk memberikan informasi tambahan pada data sudah banyak digunakan terutama menggunakan XML. Walaupun XML sudah menyajikan bentuk data yang terstruktur, hal tersebut tidak cukup jika

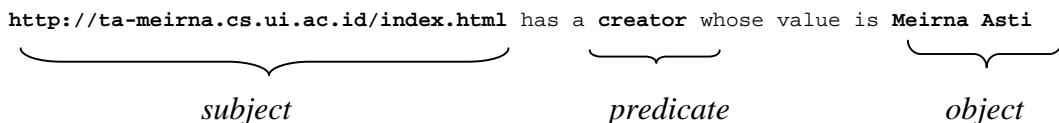
digunakan dalam konteks *semantic web*, karena XML hanya mampu mengakomodasi *syntactic interoperability*. Sebagai contoh, apabila label yang digunakan untuk mendefinisikan penulis dari sebuah buku adalah <author>Shakespeare</author> sedangkan pada dokumen katalog diberi label <writer>Shakespeare</writer>, maka kedua data tersebut bisa dianggap berbeda meskipun sebenarnya menjelaskan buku yang sama. Akibatnya, data tersebut tidak dapat diproses lebih lanjut. RDF mampu mengatasi permasalahan ini dengan menyediakan suatu pondasi dasar dalam menemukan relasi antar *resources*.

RDF adalah model data yang berbentuk *graph* yang terdiri dari *nodes* dan *edges*. Karena tiap *edge* memiliki arah yang dituju maka disebut juga *directed graph*. Tidak seperti pada *relational database (model)*, RDF dinyatakan dalam bentuk *triple*, yang terdiri dari *subject*, *predicate*, dan *object* [16].



Gambar 3.4 Contoh RDF Statement

Subject adalah *resource* yang ingin dijelaskan melalui *property* dan *value of property*. *Predicate* merupakan relasi yang menghubungkan *subject* dan *object*, yang direpresentasikan dengan tanda panah (*edge*). Sedangkan *object* adalah titik akhir dari *edge* yang bisa berupa *resource* atau *literal value*. Berikut kalimat yang berisi pernyataan seperti pada contoh di atas.



Untuk lebih memperjelas perbedaan representasi antara model data relasional dengan RDF, berikut contoh konversi data yang disajikan dalam *relational database* ke dalam bentuk RDF *triples*.

Tabel 3.1 Relational Table

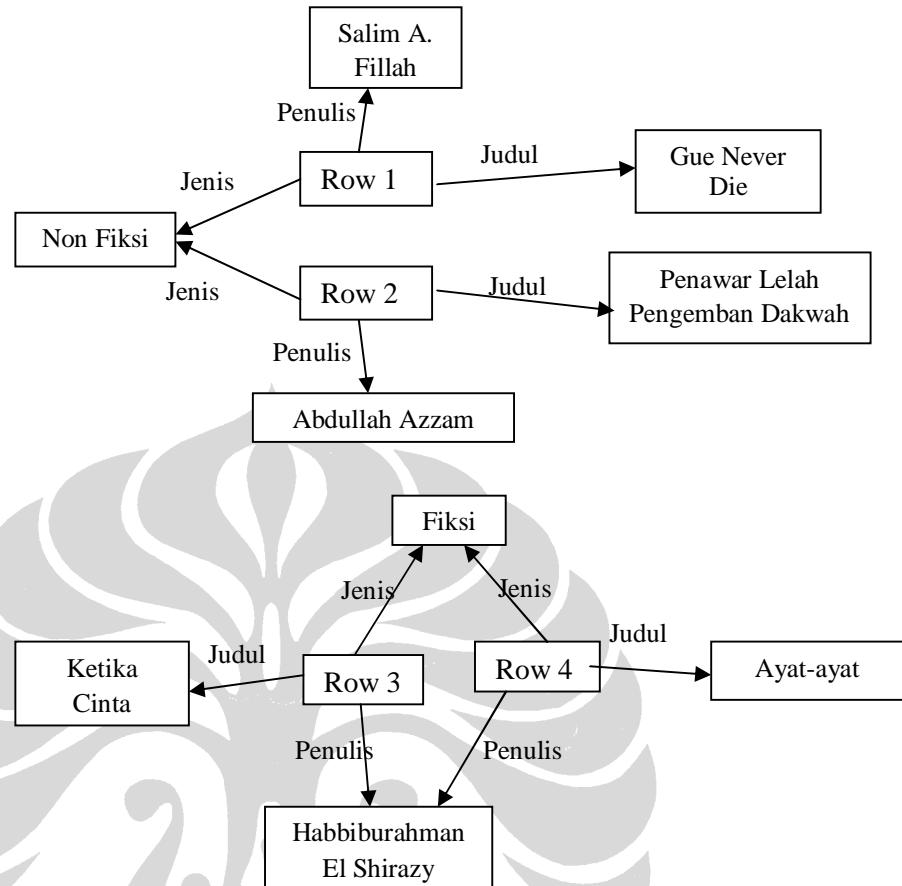
ID	Judul	Penulis	Jenis
1	Gue Never Die	Salim A. Fillah	Non Fiksi
2	Penawar Lelah Pengembangan Dakwah	Abdullah Azzam	Non Fiksi
3	Ketika Cinta Bertasbih	Habbiburahman El Shirazy	Fiksi
4	Ayat-ayat Cinta	Habbiburahman El Shirazy	Fiksi

Pada tabel relasional, setiap *resource* diberikan identitas berupa ID. ID tersebut akan menjadi *subject* pada model RDF *triples*. Kemudian setiap atribut yang dimiliki oleh *resource* tersebut akan diubah menjadi *predicate* yang menghubungkannya dengan *object* yang berasal dari *value* di setiap atribut pada *relational table*.

Tabel 3.2 RDF Triples

Subject	Predicate	Object
Row 1	Judul	Gue Never Die
Row 1	Penulis	Salim A. Fillah
Row 1	Jenis	Non Fiksi
Row 2	Judul	Penawar Lelah Pengembangan Dakwah
Row 2	Penulis	Abdullah Azzam
Row 2	Jenis	Non Fiksi
Row 3	Judul	Ketika Cinta Bertasbih
Row 3	Penulis	Habbiburahman El Shirazy
Row 3	Jenis	Fiksi
Row 4	Judul	Ayat-ayat Cinta
Row 4	Penulis	Habbiburahman El Shirazy
Row 4	Jenis	Fiksi

Jika direpresentasikan dalam bentuk *graph* akan menjadi seperti pada Gambar 3.5.



Gambar 3.5 RDF Graph

3.3.1 RDF Naming

Resource diidentifikasi dengan URI (*Uniform Resource Identifier*), seperti <http://ta-meirna.cs.ui.ac.id/index.html>. Penggunaan URI lebih umum dibandingkan dengan URL (*Uniform Resource Locator*) yang khusus digunakan untuk mengakses *resource* berupa dokumen *web* melalui jaringan komputer. URI tidak hanya dipakai untuk mengakses *resource* tetapi lebih untuk merujuk kepada sesuatu. RDF menggunakan URI sebagai mekanisme dasar untuk identifikasi *resource*. Jadi dalam konteks ini, <http://example.org/index.html> bukan menyatakan suatu alamat yang dapat diakses melalui *web browser* melainkan untuk mengidentifikasi *resource*. Dalam penamaan RDF *statement*, *subject* dan

property harus berupa URI, tetapi *object* dapat berupa URI atau *literal* (nilai konstan).

URI dapat diikuti dengan *fragment identifier*, yaitu setelah tanda '#', misalnya <http://ta-meirna.cs.ui.ac.id/index.html#section1>. Bentuk seperti ini dinamakan **URI reference** (URIref) dan digunakan untuk penamaan sesuatu yang dinyatakan dalam RDF, yaitu sebagai *resource* ID. Karena RDF tidak menggunakan URI untuk mengakses *resource* sebagai suatu dokumen *web*, maka <http://ta-meirna.cs.ui.ac.id/index.html> dianggap tidak memiliki relasi atau kaitan langsung dengan <http://ta-meirna.cs.ui.ac.id/index.html#section1>, sebab relasi antar *resource* pada RDF dinyatakan secara eksplisit melalui *property*.

Untuk menyederhanakan penulisan URI, RDF menggunakan *qualified names* (QNames) yang terdiri atas *prefix* untuk suatu *namespace*, diikuti oleh tanda ':' , dan *local name* (ID). Contohnya ialah 'dc:creator'. QNames merupakan penamaan yang digunakan untuk XML *content* [16], namun dalam konteks RDF tidak terbatas pada penulisan yang berbasis XML. Penggunaan *namespace* dimaksudkan untuk menghindari konflik penamaan pada *tag* XML dengan memakai URI.

3.3.2 RDF Syntax

Terdapat beberapa cara dalam menuliskan RDF, misalnya: RDF/XML, N-Triple, N3, Turtle, dan lain-lain. Format RDF/XML atau disebut juga *serialization* format memiliki *syntax* yang lebih rumit untuk ditulis maupun dibaca oleh manusia. Contoh berikut merupakan format RDF/XML dari *graph* yang ditampilkan pada Gambar 3.4.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://ta-meirna.cs.ui.ac.id/index.html">
    <dc:creator>Meirna Asti</dc:creator>
  </rdf:Description>
</rdf:RDF>
```

Syntax yang lebih sederhana dan mudah dipahami ialah Notation 3 (**N3**). *Statement* RDF dengan jelas dinyatakan dalam bentuk *triples*. Salah satu contoh konvensi penulisan *property* ialah ‘rdf:type’ menjadi ‘a’. *Berikut* contoh penulisannya yang ekuivalen dengan contoh RDF/XML di atas.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://ta-meirna.cs.ui.ac.id/> .
ex:index.html dc:creator "Meirna Asti" .
```

3.3.3 RDF *Query*

Untuk mendapatkan data yang disimpan dalam model RDF, diperlukan bahasa *query*. Ada beberapa macam bahasa *query* RDF, salah satunya adalah SPARQL yang direkomendasikan W3C [21]. SPARQL *query* terdiri atas *triple pattern* yang disebut *basic graph pattern*. *Triple pattern* sama seperti RDF *triple* kecuali masing-masing *subject*, *predicate*, dan *object* dapat berupa variabel. Contoh *Query1* menggunakan *triple pattern* dengan variabel yang ditanyakan adalah *object*. Sedangkan *Query2* menunjukkan variabel yang ditanyakan adalah *subject* dan *object*.

Data:

```
<http://ta-meirna.cs.ui.ac.id/index.html>
<http://purl.org/dc/elements/1.1/creator> "Meirna Asti" .
```

Query1:

```
SELECT ?x
WHERE
{
<http://ta-meirna.cs.ui.ac.id/index.html>
<http://purl.org/dc/elements/1.1/creator> ?x .}
```

Result1:

x
"Meirna Asti"

Query2:

```
SELECT ?x ?y
WHERE
{
?x <http://purl.org/dc/elements/1.1/creator> ?y .}
```

Result2:

x	y
http://ta-meirna.cs.ui.ac.id/index.html	"Meirna Asti"

3.4 RDF Schema

RDF *Schema* adalah suatu skema bahasa untuk RDF. RDFS digunakan untuk mendefinisikan kosakata yang dipakai pada RDF. RDFS mendeskripsikan konstruksi dari suatu tipe objek atau entitas (*Classes*), merelasikan satu tipe objek dengan yang lain (*subClasses*), properti yang mendeskripsikan objek (*Properties*), dan hubungan antara properti tersebut (*subProperty*). Selain itu, RDFS juga dapat mengkonstruksi `rdfs:domain` dan `rdfs:range` untuk menggambarkan hubungan antara *properties* dan *classes*. Semua informasi *schema* ini (*classes*, *subclasses*, *properties*, *subproperties*) direpresentasikan dalam bentuk RDF triples. Tabel-tabel berikut menampilkan beberapa kosakata utama yang didefinisikan dalam RDFS.

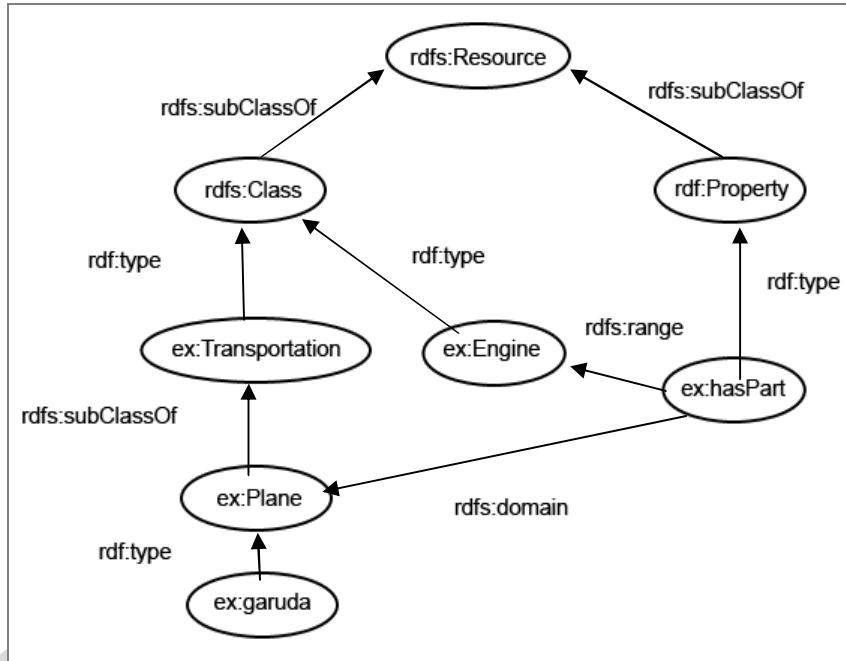
Tabel 3.1 RDF Classes

Class Name	Description
<code>rdfs:Class</code>	Class of classes
<code>rdfs:Resource</code>	Class resources, everything
<code>rdfs:Literal</code>	Class of literal value (strings, integers)
<code>rdfs:Datatype</code>	Class of RDF Datatypes
<code>rdf:Property</code>	Class of RDF Properties

Tabel 3.2 RDF Properties

Property	Description	Domain	Range
<code>rdf:type</code>	Subject is an instance of a class	<code>rdfs:Resource</code>	<code>rdfs:Class</code>
<code>rdfs:subClassOf</code>	Subject is a subclass of a class	<code>rdfs:Class</code>	<code>rdfs:Class</code>
<code>rdfs:subPropertyOf</code>	Subject is a subproperty of a property	<code>rdf:Property</code>	<code>rdf:Property</code>
<code>rdfs:domain</code>	Domain of subject property	<code>rdf:Property</code>	<code>rdfs:Class</code>
<code>rdfs:range</code>	Range of subject property	<code>rdf:Property</code>	<code>rdfs:Class</code>
<code>rdfs:label</code>	Name for the subject	<code>rdfs:Resource</code>	<code>rdfs:Literal</code>

Penggunaan RDF *Schema* pada dasarnya untuk menggambarkan hubungan yang terjadi antar *class*, *properties*, *value*, dan *instances* pada sebuah model semantik. Gambar 3.3 menunjukkan contoh penggunaan RDF *Schema* pada sebuah *graph* yang merepresentasikan ontologi mengenai pesawat. *Graph* tersebut juga dapat direpresentasikan dalam bahasa OWL seperti yang akan dijelaskan pada subbab selanjutnya.



Gambar 3.6 Contoh RDF Schema

3.5 Web Ontology Language (OWL)

Web Ontology Language (OWL) [27] adalah bahasa ontologi untuk *web* yang merupakan ekstensi dari *RDF Schema*. Laiknya *RDF*, *OWL* juga direkomendasikan oleh W3C sebagai bagian dari aktivitas pengembangan *semantic web*. *OWL* merupakan generasi berikutnya dari bahasa yang sebelumnya dikembangkan, yaitu *Defense Advanced Research Projects Agency* (DARPA) *Agent Markup Language* (DAML) + *Ontology Inference Layer* (OIL).

Saat ini *OWL* adalah bahasa ontologi paling ekspresif yang digunakan untuk aplikasi *semantic web*. Karena visi *semantic web* untuk memberikan informasi yang bermakna secara eksplisit sehingga mesin dapat memproses secara otomatis dan mengintegrasikan informasi pada web, maka diperlukan bahasa yang tepat untuk merepresentasikan informasi tersebut. *OWL* digunakan untuk merepresentasikan makna dari kosakata dan relasi antar kata sehingga makna suatu informasi menjadi eksplisit [36].

OWL dapat direpresentasikan sebagai *RDF triples* dan juga memiliki bentuk *graph model* seperti *RDF*. Pemodelan *OWL* secara grafik dapat dilakukan dengan

lebih mudah menggunakan UML (*Unified Modelling Language*) karena notasinya telah banyak digunakan oleh *developer* untuk menggambarkan *class diagram*. Dalam penelitian ini, OWL yang digunakan untuk merepresentasikan ontologi adalah OWL versi 1.0.

3.5.1 Level Bahasa OWL

OWL menyediakan tiga level bahasa (*species*) yang penggunaannya disesuaikan berdasarkan kebutuhan, yaitu OWL Lite, OWL DL, dan OWL Full. OWL DL dapat dipandang sebagai ekstensi dari OWL Lite dan OWL Full sebagai ekstensi OWL DL. Berikut penjelasan masing-masing level bahasa OWL [27].

- OWL Lite

Sub bahasa ini adalah yang paling sederhana dibanding sub bahasa lainnya. OWL Lite memiliki formalitas bahasa yang lebih rendah (secara *logic*) namun lebih ekspresif dibanding RDF(S). OWL Lite digunakan untuk memenuhi kebutuhan klasifikasi secara hierarkis dan batasan (*constraint*) yang sederhana. Batasan kardinalitas yang diperbolehkan pada level ini hanya 0 atau 1. Level ini memberi kemudahan dalam migrasi dari bentuk taksonomi biasa.

- OWL DL (*Description Logic*)

OWL DL berdasarkan *description logic* (subset dari *first-order predicate logic*) yang berkembang dari *semantic network* dan memiliki definisi formal untuk *knowledge representation*. Sub bahasa ini menambahkan beberapa fitur selain dari yang dimiliki oleh OWL Lite, antara lain membuat *class* dengan operasi himpunan (*boolean combinations*) seperti *unionOf*, *intersectionOf*, *complementOf*. Selain tidak membatasi kardinalitas hanya 0 atau 1, OWL DL juga memungkinkan untuk mendefinisikan suatu nilai *property* yang berasal dari *instance* suatu *class* dengan *feature hasValue*. Ada pula tambahan *feature* untuk *class* yaitu *disjointWith* dan *oneOf (enumerated classes)*.

- OWL Full

Level ini merupakan sub bahasa yang paling kompleks dan digunakan oleh pengguna yang menginginkan ekspresi maksimum tanpa adanya jaminan *computational* (pada saat *reasoning* mungkin tidak lengkap atau selesai

dalam waktu yang berhingga). Hal ini berbeda dengan OWL DL yang memberikan jaminan tersebut sehingga bisa dilakukan *automated reasoning* pada sub bahasa tersebut. Pada OWL Full, suatu *class* dapat diperlakukan sebagai *collection of instances* dan *instance* itu sendiri. Selain itu, *datatypeProperty* dapat dispesifikasi sebagai *inverseFunctionalProperty*.

3.5.2 Elemen OWL

Elemen pada OWL terdiri atas *classes*, *properties*, *instances of classes*, dan relasi antar *instances*. Untuk pengembangan ontologi (dengan OWL) sebaiknya menggunakan ontologi *editor* seperti Protégé agar lebih fokus pada representasi yang akan dilakukan, sedang sintaksnya dapat dihasilkan (*generated*) oleh *tool* secara otomatis. Berikut merupakan penjelasan dari masing-masing elemen [27].

- *Classes*

OWL mendefinisikan *root* dari semua yang ada dengan `owl:Thing`. Jadi semua *class* yang dibuat secara implisit merupakan *subclass* `owl:Thing`. Pembuatan *class* menggunakan `owl:Class` dan menyatakan *subclass* dengan `rdfs:subClassOf`. Berikut contoh pendefinisan *class* `Engine` dan *class* `Plane` sebagai *subclass* dari `Transportation`.

```
<owl:Class rdf:ID="Transportation" />
<owl:Class rdf:ID="Engine" />
<owl:Class rdf:ID="Plane">
    <rdfs:subClassOf rdf:resource="#Transportation" />
</owl:Class>
```

- *Individuals*

Individuals atau disebut juga *instances* adalah anggota dari *classes*. *Instances* dapat dipandang sebagai objek yang ada pada domain yang dibahas. Sama seperti `owl:Class` yang menjadi *meta level* untuk *class*, begitu pula *class* yang telah didefinisikan menjadi *meta level* untuk *instance*. Berikut contoh pendefinisan *instance* dari *class* `Plane`.

```
<Plane rdf:ID="garuda" />
```

Sintaks berikut juga memiliki arti yang sama dengan pendefinisian sebelumnya.

```
<owl:Thing rdf:ID="garuda" />
<owl:Thing rdf:about="#garuda" >
    <rdf:type rdf:resource="#Plane" />
</owl:Thing>
```

Dalam format RDF, sintaks tersebut dapat dituliskan seperti berikut.

```
<rdf:Description rdf:about="garuda" >
    <rdf:type rdf:resource="#Plane" />
</rdf:Description>
```

- *Properties*

Property merupakan relasi *binary*. Ada dua jenis *property* pada OWL, yaitu: *ObjectProperty* (relasi antara *instance* dari dua *classes*) dan *DatatypeProperty* (relasi antara *instance* dengan RDF *literal* dan XML *Schema datatypes*). Sama halnya seperti *class* yang dapat dinyatakan secara hierarkis, begitu pula *property* dapat dinyatakan sebagai *subPropertyOf* dengan *rdfs:subPropertyOf*. Untuk memberikan batasan pada suatu *property*, dapat digunakan *rdfs:domain* dan *rdfs:range*, yang disebut juga sebagai *global restriction* karena berlaku untuk umum, tidak terbatas pada *class* tertentu.

```
<owl:ObjectProperty rdf:ID="hasPart" >
    <rdfs:domain rdf:resource="#Plane" />
    <rdfs:range rdf:resource="#Engine" />
</owl:ObjectProperty>
```

3.5.3 Fitur OWL

OWL menawarkan beberapa fitur yang memberikan berbagai keuntungan bagi *developer*, yaitu [26]:

- Mekanisme *solid modularization* yang memungkinkan pendefinisian ontologi yang dapat diperluas dengan mudah.

- Mendukung pendefinisian konsep hierarki, sehingga *reasoner* bisa mengenali adanya hubungan *inheritance* (*is-a*) antara dua buah konsep dengan mudah.
- Cara yang *advance* untuk mendeskripsikan *properties*, seperti: *range* dari sebuah *property* didefinisikan sebagai gabungan dari dua atau lebih *class* lainnya, definisi dari batasan kardinalitas, dan sebagainya.
- Kemampuan untuk mendefinisikan sinonim, sehingga kita bisa membuat ekuivalensi (atau pemetaan) antara dua (atau lebih) konsep kosakata yang berada pada domain yang sama. Sebagai contoh, kita dapat mendefinisikan pemetaan antara terminologi ALOCoM dan SCORM – misalnya, *Content Fragment* pada ALOCoM ekuivalen dengan *Asset* pada SCORM.

3.5.4 Rule dan Reasoning

Reasoning pada OWL DL berdasarkan *open world assumption*, artinya kita tidak dapat mengasumsikan sesuatu tidak ada sampai hal tersebut dinyatakan secara eksplisit tidak ada [27]. Dengan kata lain, karena sesuatu tidak dinyatakan *true*, tidak dapat diasumsikan sesuatu itu *false*. Hal ini berbeda dengan *relational database* yang bersifat *close world assumption*, yaitu sesuatu hanya akan dianggap keberadaannya apabila telah tersimpan pada *database*. Dalam konsep *semantic web*, model data relasional tidak cocok diterapkan karena tidak sesuai dengan domain pengetahuan yang luas serta sifatnya yang terus mengalami perubahan dan perkembangan.

Proses *reasoning* atau *inference* pada OWL DL menggunakan *reasoner* DIG (*Description Logic Implementers Group*) seperti Pellet, Racer or FaCT. *Reasoner* ini digunakan untuk memeriksa konsistensi pada ontologi, melakukan klasifikasi secara otomatis berdasarkan relasi hierarki (*subsumption reasoning*), dan mendapatkan data atau fakta baru berdasarkan *axioms* dan *rules*.

3.5.5 Penggunaan OWL

Berdasarkan dokumen W3C mengenai OWL *Use Cases and Requirements* [27], berikut ini beberapa contoh penggunaan ontologi, khususnya OWL:

- *Web portal*, ontologi sebagai definisi kosakata untuk mendeskripsikan *content* sehingga dapat meningkatkan hasil pencarian misalnya dengan menggunakan *inference*.
- *Multimedia collections*, ontologi untuk *semantic annotation* yang menyediakan deskripsi tentang *image*, *audio*, *video*, dan objek bukan teks lainnya.
- *Corporate web site management*, ontologi untuk mengindeks dokumen perusahaan sehingga dapat dilakukan *knowledge sharing* dalam perusahaan tersebut.
- *Design documentation*, ontologi sebagai model informasi pada dokumen *engineering* untuk suatu domain sehingga dapat dilakukan eksplorasi terhadap domain tersebut.
- *Agents and services*, ontologi sebagai kosakata untuk komunikasi antar *software agent* dan untuk mendefinisikan *service* sehingga dapat menemukan yang sesuai.
- *Ubiquitous computing*, ontologi digunakan untuk deskripsi karakter *devices*, cara untuk mengakses *device*, dan aturan lainnya untuk mendukung penggunaan *device* pada *ubiquitous computing network*.

3.6 Semantic Portal

Semantic web portal atau *semantic portal* yang didefinisikan dalam penelitian ini merupakan aplikasi dari kedua hal yang telah dibahas sebelumnya, yaitu ontologi dan *semantic web*.

3.6.1 Konsep

Berbicara mengenai *semantic portal*, maka tak terlepas dari pemahaman mengenai *web portal*. *Web portal* merupakan *entry point* untuk mengakses informasi melalui internet. *Web portal* memungkinkan pengguna internet untuk melakukan *browsing* pada suatu domain, mulai dari domain yang luas (seperti www.yahoo.com) hingga domain spesifik (seperti www.oracle.com). Tujuan dari *web portal* adalah untuk menyediakan informasi yang terintegrasi dan terstruktur sehingga memudahkan pengguna dalam melakukan *browsing* dan *searching*

melalui internet. *Web portal* juga dikenal sebagai *point of access*, *start page*, atau *anchor*.

Web portal, selain sebagai penyedia informasi satu arah, juga dapat digunakan secara khusus oleh suatu komunitas untuk memfasilitasi anggotanya dalam berbagi informasi. Artinya, pengguna *portal* tidak hanya bisa memperoleh informasi namun juga dapat memberikan kontribusi terhadap *content* pada *web portal*. Dengan demikian, terjadi pertukaran informasi dua arah pada *web portal*. Jenis *portal* seperti ini disebut *community information portal*.

Setelah memahami definisi *web portal*, maka konsep *semantic portal* dapat lebih mudah dimengerti. *Semantic portal* tak lain adalah *web portal* yang menggunakan teknologi *semantic web* [32]. Inti dari *semantic portal* yaitu penyajian informasi secara terstruktur dengan menggunakan ontologi, sesuai dengan konsep pada *semantic web*. Dapat dikatakan bahwa ontologi menjadi *backbone* dari *semantic portal*. Bahkan, melalui *semantic web*, telah disediakan bahasa standar untuk ontologi (OWL) dan format data (RDF) yang dapat meningkatkan proses *sharing* pada *portal*.

3.6.2 Penelitian *Semantic Portal* yang Telah Dikembangkan

AIFB (*Institute for Applied Informatics and Formal Description Method*) [2] dari *University of Karlsruhe* telah membuat sebuah *framework* bernama SEAL (SEmantic portAL) untuk pengembangan *semantic portal* [34]. *Web* yang menggunakan pendekatan ini adalah AIFB *portal*, yang menampilkan informasi mengenai penelitian yang dilakukan oleh AIFB.

Penelitian selanjutnya adalah Esperonto *portal* [21] yang dikembangkan oleh *Ontology Group* dari Facultad de Informática, Universidad Politécnica de Madrid. *Portal* ini berfungsi sebagai *intranet* dan *extranet* yang memfasilitasi proyek Esperonto. Esperonto termasuk salah satu dari kumpulan proyek IST (*Information Society Technologies*) yang dibuat dengan menggunakan ODESeW (*Semantic Web based on WebODE platform*), suatu *knowledge portal generator*.

Selain Esperonto, salah satu proyek IST lainnya adalah OntoWeb [39], yang merupakan sebuah *community portal* yang digunakan oleh pihak akademisi maupun industri yang memiliki ketertarikan terhadap *semantic web*. OntoWeb memanfaatkan ZOPE CMF (*Content Management Framework*) untuk mengelola isi portal.

Contoh penelitian berikutnya yaitu SWED (*Semantic Web Environmental Directory*) [48] dari proyek SWAD-E (*Semantic Web Advanced Development for Europe*) yang dibiayai oleh *European Commission*. *Portal* yang dibangun menggunakan Jena *framework* ini berfungsi sebagai *directory* bagi *cross-community* yang menghimpun beberapa organisasi lingkungan yang bergerak pada ranah *wildlife*, *environmental*, dan *biodiversity*. SWED selanjutnya telah dijadikan sebagai *tool* untuk pengembangan *portal* yang bersifat *open-source* dan dinamakan portalCore.