BAB II

LANDASAN TEORI

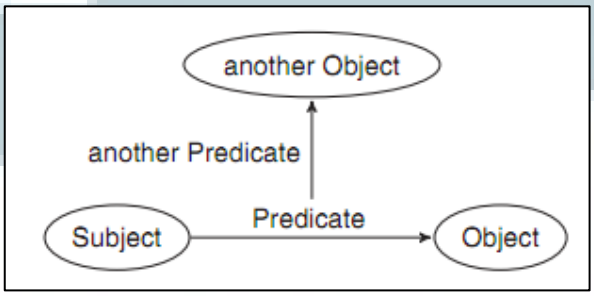
2.1 Resource Description Framework

Resource Description Framework (RDF) merupakan bahasa yang digunakan untuk menyatakan *statements* tentang suatu sumber daya (Oren & Schenk, 2011). RDF menjadi bahasa dasar dari *semantic web* (Pollock, 2009). RDF dapat digunakan sebagai bahasa untuk mendeskripsikan kata, *metadata*, dan bahkan bahasa data lainnya (Pollock, 2009). Model data atau bahasa data apapun yang menggunakan RDF menjadi bagian dari *Semantic Web* (Pollock, 2009).

RDF menggunakan format data berbentuk grafik, berbeda dengan format data relasional, seperti pada kebanyakan *database*, dan format data hierarkis, seperti pada *Extensible Markup Language* (XML) (Pollock, 2009). Grafik tidak memiliki akar, sedangkan XML berbasis pohon sehingga terdapat elemen akar (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Menggabungkan dua XML yang berbasis pohon, akan sulit menentukan *node* akar yang sebaiknya digunakan (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Hal ini dikarenakan struktur pohon sangat penting bagi keseluruhan makna kata (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Sebaliknya, menggabungkan grafik RDF lebih mudah karena secara konsep sama dengan meletakkan grafik yang digabung di bawah grafik yang satunya (Hebeler, Fisher, Blace, & Perez-Lopez, 2009).

Sekumpulan pernyataan (*statements*) pada RDF merupakan representasi dari informasi yang dibentuk dari tiga bagian, yaitu subjek, predikat, dan objek

(Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Oleh karena terdapat tiga bagian, pernyataan ini terkadang juga disebut sebagai *triples* (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Gambar 2.1 menunjukkan grafik RDF dasar yang berbentuk *triples*. Nama bagian subjek dan predikat harus berasal dari *namespaces* spesifik sehingga tidak ada orang atau proses yang merancukan nama-nama tersebut dengan nama yang sama, terutama jika suatu data ingin digabungkan dengan data lainnya (DuCharme, 2011). Seperti XML, RDF memperbolehkan penggunaan *prefix* untuk merepresentasikan sebuah *namespace* URI (*Uniform Resource Identifier*) sehingga data tidak mengandung terlalu banyak kata (DuCharme, 2011). Akan tetapi, tidak seperti XML, RDF memperbolehkan penggunaan URIs lengkap dengan nama sebagai ganti *prefixes* (DuCharme, 2011).



Gambar 2.1 Grafik RDF Dasar
(Oren & Schenk, 2011)

Selain itu, RDF dapat dilihat sebagai sebuah bahasa untuk *statements* tanpa menspesifikasikan makna dari *statements* tersebut (Oren & Schenk, 2011). Dengan demikian, RDF dapat diibaratkan sebagai sebuah alfabet yang dapat digunakan untuk membangun kata-kata dan kalimat-kalimat, tetapi belum dapat dikatakan sebagai sebuah bahasa karena kata-kata dan kalimat-kalimat tersebut belum diberikan maknanya (Oren & Schenk, 2011).

Statements RDF dapat diibaratkan sebagai titik-titik pada ruang tiga dimensi yang abstrak (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Setiap garis merepresentasikan subjek, predikat atau objek (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Representasi *statements* RDF seperti ini mengilustrasikan kelebihan *statements* RDF untuk berbagi data, yaitu sebagai berikut (Hebeler, Fisher, Blace, & Perez-Lopez, 2009).

1. Mudah digabungkan: ibarat titik-titik pada ruang dimensi yang dapat saling *overlay* untuk membentuk gambar yang lebih kaya, dua grafik dari *statements* dapat digabungkan untuk membentuk grafik yang lebih kaya.
2. Tidak ada urutan: jika sebuah gambar tersusun dari titik-titik, semua titik sudah ditambahkan dan gambar sudah lengkap, tidak perlu menentukan urutan ketika titik yang berbeda ditambahkan.
3. Tidak ada duplikasi: jika dua *statements* memiliki subjek, predikat, dan objek yang sama, kedua *statements* ini identik sehingga tidak terjadi penambahan informasi ketika duplikasi dari suatu *statements* ditambahkan kembali.

Istilah teknis dari menyimpan RDF pada sebuah *disk* sebagai sebuah *string of bytes* adalah *serialization* (DuCharme, 2011). Istilah ini digunakan sebagai ganti "*files*" karena terdapat sistem operasi yang tidak menggunakan istilah "*file*" untuk koleksi data yang disimpan dan memiliki nama (DuCharme, 2011). Pada praktiknya, semua serialisasi RDF sejauh ini berbentuk *files* teks dengan berbagai *syntax* digunakan untuk merepresentasikan *triples* (DuCharme, 2011).

Jika ingin menyimpan *triples* dalam jumlah yang sangat besar, menyimpannya sebagai Turtle atau RDF/XML dalam satu *file* teks yang besar bukanlah opsi terbaik karena sistemlah yang mengindeks data dan menentukan data mana yang dimuat ke memori, sedangkan sistem pengelolaan *database* dapat melakukannya dengan lebih efisien (DuCharme, 2011). Terdapat cara tertentu untuk menyimpan RDF dalam *database* relasional, seperti MySQL atau Oracle, tetapi cara terbaik adalah *database* relasional yang dioptimasi untuk RDF *triples* yang disebut dengan *triplestore* (DuCharme, 2011).

RDF dapat divalidasi seperti memvalidasi XML, HTTP (*Hypertext Transfer Protocol*), atau bahasa pemrograman lainnya (Pollock, 2009). Validator secara sederhana memeriksa untuk melihat apakah ada masalah dengan kode RDF yang dibuat (Pollock, 2009). Layanan validasi RDF dapat dilakukan pada www.w3.org/RDF/Validator (Pollock, 2009).

2.2 RDF Schema

RDF *Schema* merupakan sebuah *vocabulary* untuk RDF yang mendefinisikan istilah, tujuan penggunaan, dan semantiknya (Oren & Schenk, 2011). RDF *Schema* digunakan untuk mendefinisikan RDF *vocabularies*, seperti *vocabulary* FOAF (*Friend of a Friend*) atau *Simple Knowledge Organization System* (Oren & Schenk, 2011). Dalam pengembangan *semantic web*, sebuah *vocabulary* merupakan sekumpulan *terms* yang disimpan menggunakan suatu format standar yang dapat digunakan kembali oleh banyak orang (DuCharme, 2011). RDF *Schema* ini sebenarnya sudah merupakan bahasa ontologi yang

sederhana (Oren & Schenk, 2011). RDF *Schema* sering disingkat menjadi RDFS, sedangkan kombinasi antara RDF dan RDF *Schema* disingkat menjadi RDF(S) (Oren & Schenk, 2011).

RDF *Schema* merupakan bahasa untuk mendeskripsi RDF *vocabulary* (RDF *Vocabulary Description Language*) (DuCharme, 2011). Dengan demikian, *metadata* ini hanya dapat diakses oleh *queries* SPARQL (DuCharme, 2011). RDF *Schema* menyediakan bahasa yang dapat digunakan untuk membuat *shared vocabulary* sendiri (Hebeler, Fisher, Blace, & Perez-Lopez, 2009).

2.3 The Web Ontology Language

Web Ontology Language disingkat menjadi OWL karena lebih mudah diucapkan dibanding WOL (DuCharme, 2011). OWL ini dibuat menggunakan RDFS untuk mendefinisikan ontologi (DuCharme, 2011). OWL didefinisikan sebagai sebuah pengembangan RDFS dengan tujuan menyediakan *vocabulary* untuk membuat data RDF (Isaac, Schenk, & Scherp, 2011). Oleh karena itu, OWL ini sangat dipengaruhi dengan model data RDF yang berbasis grafik tersusun dari *nodes* dan panah berarah (Isaac, Schenk, & Scherp, 2011).

OWL menyediakan bahasa ekspresif untuk mendefinisikan ontologi dengan unsur semantik dari *domain knowledge* (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Dengan demikian, OWL ini stabil dan sangat bermanfaat untuk mendefinisikan data agar tidak ambigu (Pollock, 2009).

2.4 Simple Knowledge Organization System

Simple Knowledge Organization System (SKOS) merupakan model data umum untuk berbagi (*sharing*) dan menghubungkan (*linking*) sistem-sistem pengorganisasian *knowledge* melalui *web* (Semantic Web Deployment Working Group, 2009). Banyak sistem pengorganisasian *knowledge*, seperti tesaurus, taksonomi, skema klasifikasi, dan sistem *subject heading*, yang memiliki struktur yang serupa dan digunakan pada aplikasi yang serupa (Semantic Web Deployment Working Group, 2009). SKOS menangkap kemiripan ini dan membuatnya menjadi eksplisit, yang memperbolehkan berbagi data dan teknologi lintas aplikasi yang beragam (Semantic Web Deployment Working Group, 2009). Hal ini dikarenakan dengan menggunakan SKOS, suatu sistem pengorganisasian *knowledge* dapat diekspresikan sebagai data yang *machine-readable* sehingga dapat dipertukarkan antara aplikasi komputer dan dipublikasi pada suatu format yang *machine-readable* pada *web* (Semantic Web Deployment Working Group, 2009). Tujuan utama SKOS dibangun adalah memperbolehkan publikasi *controlled structured vocabulary* untuk *semantic web* yang mudah (PoolParty, 2013b).

Model Data SKOS memperlihatkan sebuah sistem pengorganisasian *knowledge* sebagai *concept scheme* yang terdiri dari sekumpulan konsep (Semantic Web Deployment Working Group, 2009). Hal ini berarti SKOS mengusulkan model berbasis konsep untuk merepresentasikan *controlled vocabularies* (Isaac, Schenk, & Scherp, 2011). Sebagai lawan dari pendekatan berbasis *term*, di mana *terms* dari bahasa natural merupakan elemen urutan

pertama dari *Knowledge Organization System* (KOS), SKOS menggunakan *concepts* yang lebih abstrak yang dapat memiliki *lexicalizations* yang berbeda (Isaac, Schenk, & Scherp, 2011). SKOS memperkenalkan *class* spesial (skos:Concept) untuk digunakan dalam data RDF untuk merepresentasikan tipe dari entitas bersangkutan (Isaac, Schenk, & Scherp, 2011).

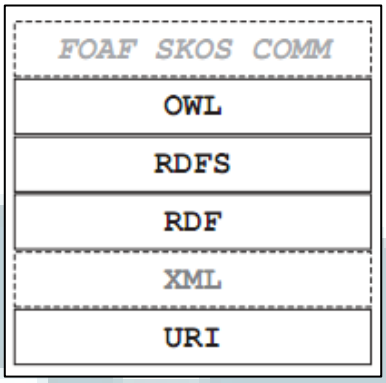
Skema konsep SKOS dan konsep SKOS diidentifikasi menggunakan URIs (*Uniform Resource Identifiers*) (Semantic Web Deployment Working Group, 2009). URIs ini pada dasarnya digunakan untuk menyediakan nama unik di *Resource Description Framework* (RDF) (Pollock, 2009). URIs ini membuat *resources* dapat diidentifikasi secara unik (Hebeler, Fisher, Blace, & Perez-Lopez, 2009).

URIs terlihat seperti *Universal Resource Locators* (URLs), tetapi memiliki tujuan yang berbeda (Pollock, 2009). URLs merupakan lokasi primer, misalnya halaman *web*, yang dapat dituju melalui *web*, sedangkan URIs mungkin juga dapat dituju melalui *web*, tetapi tidak harus *addressable* sebagaimana URLs (Pollock, 2009). URLs, yang juga dikenal sebagai alamat *web*, merupakan salah satu URI (DuCharme, 2011). Sebuah *locator* membantu menemukan sesuatu, seperti halaman *web* dan *identifier* mengidentifikasi sesuatu (DuCharme, 2011). Sebagai contoh, *identifier* unik untuk Richard di dalam *database* buku alamat adalah http://www/learningsparql.com/ns/addressbook#richard (DuCharme, 2011). Sebuah URI boleh terlihat seperti sebuah URL dan dapat saja sebuah halaman *web* ada pada alamat tersebut, tetapi bisa saja tidak ada karena tugas utamanya adalah menyediakan nama unik untuk sesuatu (DuCharme, 2011).

Oleh karena kebanyakan URIs adalah URLs, banyak orang tertukar dalam menggunakan istiliah ini (DuCharme, 2011). Selain URLs, URN (*Uniform Resource Name*) juga merupakan URI yang mendeskripsikan sebuah buku melalui *International Standard Book Number* (ISBN) (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Selain itu, juga terdapat IRIs (*Internationalized Resource Identifiers*) yang merupakan URIs yang memperbolehkan penggunaan jangkauan karakter-karakter yang lebih luas untuk mengakomodasi sistem penulisan lainnya (DuCharme, 2011). Pada pemakaian umum, IRI merupakan hal yang sama dengan URI (DuCharme, 2011). Akan tetapi, spesifikasi SPARQL *Query Language* menggunakan IRIs ketika berbicara tentang penamaan sumber daya, bukan URIs atau URLs, karena IRIs merupakan istilah yang paling luas (DuCharme, 2011).

Data SKOS diekspresikan sebagai *Resource Description Framework* (RDF) *triples*, seperti RDF/XML atau Turtle (Semantic Web Deployment Working Group, 2009). RDF/XML merupakan sebuah XML (*Extensible Markup Language*) *syntax* untuk merepresentasikan RDF *triples* dan merupakan *syntax* standar untuk pertukaran informasi (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). XML masih menjadi bagian karena standar-standar lainnya menggunakan kembali bagian dari teknologi XML (Oren & Schenk, 2011). Terlebih lagi, RDF dapat diserialisasi dalam dokumen XML (Oren & Schenk, 2011). Akan tetapi, RDF dapat diekspresikan dengan baik tanpa menggunakan XML (Oren & Schenk, 2011).

SKOS *namespace* URI adalah http://www.w3.org/2004/02/skos/core# (Semantic Web Deployment Working Group, 2009).



Gambar 2.2 Lapisan Standar Penting *Semantic Web*
(Oren & Schenk, 2011)

SKOS *vocabulary* adalah kumpulan dari URIs, misalnya skos:Concept, skos:prefLabel, skos:altLabel, skos:hiddenLabel, skos:notation, dan sebagainya (Semantic Web Deployment Working Group, 2009). *Vocabulary* SKOS yang akan dibahas lebih lanjut adalah skos:Concept, skos:prefLabel, skos:altLabel, skos:hiddenLabel, dan skos:notation. Untuk daftar *vocabulary* yang lebih lengkap dapat melihat dokumentasi SKOS Reference (Semantic Web Deployment Working Group, 2009).

Sebuah konsep SKOS dapat dilihat sebagai sebuah ide atau gagasan; suatu unit pemikiran (Semantic Web Deployment Working Group, 2009). Walaupun demikian, hal yang membenarkan suatu unit pemikiran bersifat subjektif dan definisi ini dimaksudkan untuk menjadi saran daripada membatasi (Semantic Web Deployment Working Group, 2009).

skos:Concept merupakan *class* dari konsep SKOS dan juga merupakan *vocabulary* SKOS yang terkait dengan konsep SKOS (Semantic Web Deployment Working Group, 2009). skos:Concept ini merupakan *instance* dari owl:Class (Semantic Web Deployment Working Group, 2009).

Sebuah label leksikal merupakan sebuah *string* dari karakter UNICODE (Semantic Web Deployment Working Group, 2009). SKOS menyediakan beberapa *vocabulary* dasar untuk menghubungkan label-label leksikal dengan sumber daya dari berbagai tipe (Semantic Web Deployment Working Group, 2009). Pada khususnya, SKOS memperbolehkan perbedaan dibuat antara leksikal label yang *preferred,* alternatif, dan *hidden* untuk berbagai sumber daya (Semantic Web Deployment Working Group, 2009). *Vocabulary* SKOS terkait label leksikal ini adalah skos:prefLabel, skos:altLabel, dan skos:hiddenLabel dengan definisi *class* dan properti sebagai berikut (Semantic Web Deployment Working Group, 2009).

1. skos:prefLabel, skos:altLabel, dan skos:hiddenLabel merupakan *instance* dari owl:AnnotationProperty.
2. skos:prefLabel, skos:altLabel, dan skos:hiddenLabel merupakan subproperti dari rdfs:label.
3. rdfs:range untuk setiap skos:prefLabel, skos:altLabel, dan skos:hiddenLabel merupakan *class* dari RDF *plain literal*. Secara konvensi, RDF *plain literals* selalu digunakan pada posisi objek dari sebuah *triple* dengan predikat salah satu dari skos:prefLabel, skos:altLabel, atau skos:hiddenLabel.

Label-label *preferred* dan alternatif berguna ketika membuat representasi sistem pengorganisasian pengetahuan yang dapat dibaca oleh manusia (*human-readable*) (Semantic Web Deployment Working Group, 2009). Label-label ini

menyediakan petunjuk yang paling kuat ke pengertian dari konsep SKOS (Semantic Web Deployment Working Group, 2009).

Nilai prefLabel haruslah istilah yang tidak ambigu yang mengidentifikasi suatu konsep secara unik dan dapat digunakan sebagai *descriptor* dalam sebuah sistem *indexing* (Isaac, Schenk, & Scherp, 2011). Sedangkan, nilai altLabel digunakan untuk memperkenalkan istilah alternatif, seperti sinonim, singkatan, dan variasi, yang dapat digunakan untuk mendapatkan konsep melalui mesin pencari yang tepat atau untuk memperkaya indeks suatu dokumen dengan kata kunci tambahan (Isaac, Schenk, & Scherp, 2011).

SKOS memperbolehkan konsep-konsep dihubungkan dengan label *preferred* dan alternatif dalam bahasa yang berbeda (Isaac, Schenk, & Scherp, 2011). Dengan demikian, *Knowledge Organization Systems* (KOSs) dapat digunakan dengan baik pada lingkungan multibahasa (Isaac, Schenk, & Scherp, 2011).

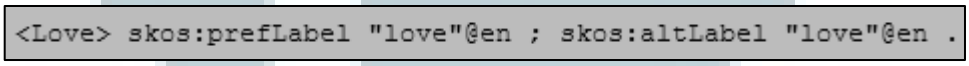
Label-label *hidden* berguna ketika pengguna berinteraksi dengan sistem pengorganisasian pengetahuan melalui fungsi pencarian berbasis teks (Semantic Web Deployment Working Group, 2009). Pengguna dapat saja melakukan kesalahan ejaan saat memasukkan kata untuk mencari konsep yang relevan (Semantic Web Deployment Working Group, 2009). Jika *query* yang salah ejaan dapat dicocokkan dengan label *hidden*, pengguna dapat menemukan konsep yang relevan walau terjadi kesalahan ejaan (Semantic Web Deployment Working Group, 2009).

Secara formal, label leksikal merupakan RDF *plain liteteral* (Semantic Web Deployment Working Group, 2009). Sebuah RDF *plain literal* dapat tersusun dari suatu bentuk leksikal, yaitu sebuah *string* dari karakter UNICODE dan sebuah *tag* bahasa yang bersifat opsional (Semantic Web Deployment Working Group, 2009).

Kondisi integritas terkait label leksikal yang harus dipenuhi pada grafik yang dibangun agar konsisten dengan model data SKOS adalah sebagai berikut (Semantic Web Deployment Working Group, 2009).

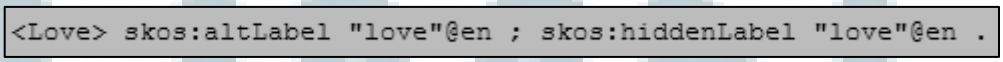
1. skos:prefLabel, skos:altLabel, dan skos:hiddenLabel merupakan pasangan properti yang *disjoint*.

   Sebagai contoh, grafik berikut tidak konsisten dengan model data SKOS karena terjadi ketidakserasian antara label leksikal *preferred* dan alternatif.



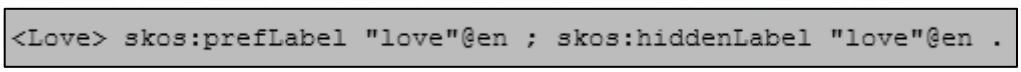
Gambar 2.3 Contoh Grafik Tidak Konsisten (1)
(Semantic Web Deployment Working Group, 2009)

   Grafik berikut juga tidak konsisten dengan model data SKOS karena terjadi ketidakserasian antara label leksikal alternatif dan *hidden.*



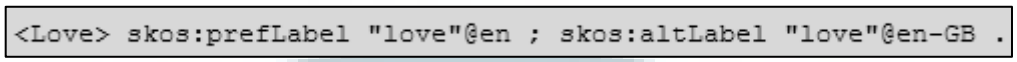
Gambar 2.4 Contoh Grafik Tidak Konsisten (2)
(Semantic Web Deployment Working Group, 2009)

   Grafik berikut juga tidak konsisten dengan model data SKOS karena terjadi ketidakserasian antara label leksikal *preferred* dan *hidden*.



Gambar 2.5 Contoh Grafik Tidak Konsisten (3)
(Semantic Web Deployment Working Group, 2009)

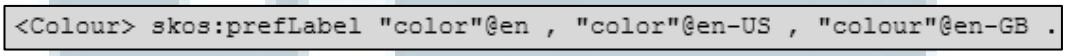
Berikut ini contoh grafik yang konsisten dengan model data SKOS karena "en" dan "en-GB" merupakan tag bahasa yang berbeda.



Gambar 2.6 Contoh Grafik Konsisten (1)
(Semantic Web Deployment Working Group, 2009)

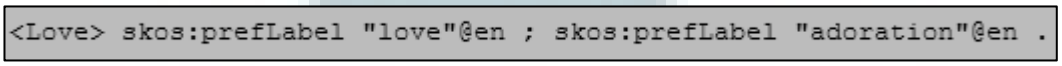
2. Sebuah sumber daya tidak boleh memiliki lebih dari satu nilai skos:prefLabel untuk satu *tag* bahasa.

   Berikut ini contoh grafik yang konsisten dengan model data SKOS karena "en", "en-US", dan "en-GB" merupakan *tag* bahasa yang berbeda.



Gambar 2.7 Contoh Grafik Konsisten (2)
(Semantic Web Deployment Working Group, 2009)

Grafik berikut merupakan contoh grafik yang tidak konsisten dengan model data SKOS karena terdapat dua label leksikal *preferred* untuk *tag* bahasa yang sama.

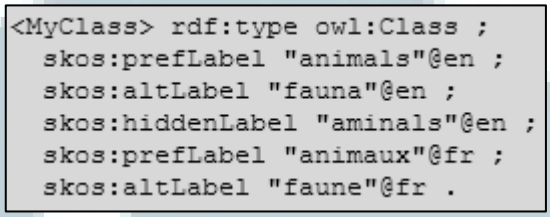


Gambar 2.8 Contoh Grafik Tidak Konsisten (4)
(Semantic Web Deployment Working Group, 2009)

Selain kondisi integrasi di atas, terdapat beberapa catatan tambahan terkait label leksikal, yaitu sebagai berikut (Semantic Web Deployment Working Group, 2009).

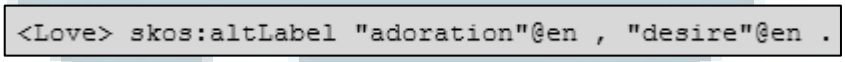
1. Menggunakan properti skos:prefLabel, skos:altLabel, dan skos:hiddenLabel untuk memberi label ke tipe sumber daya apapun konsisten dengan model data SKOS karena tidak ada *domain* diberikan untuk properti ini sehingga *domain* yang efektif untuk properti ini adalah *class* untuk semua sumber daya

(rdfs:Resource). Misalnya seperti pada grafik berikut, skos:prefLabel, skos:altlabel, dan skos:hiddenLabel digunakan untuk memberi label pada sumber daya dengan tipe owl:Class.



Gambar 2.9 Contoh Grafik Konsisten (3)
(Semantic Web Deployment Working Group, 2009)

2. Hanya terdapat label alternatif pada grafik, tidak terdapat label *preferred* merupakan model data SKOS yang konsisten. Akan tetapi, yang menjadi catatan adalah banyak aplikasi yang membutuhkan label *preferred* untuk menghasilkan tampilan yang dapat dibaca oleh manusia secara optimum.



Gambar 2.10 Contoh Grafik Konsisten (4)
(Semantic Web Deployment Working Group, 2009)

Notasi merupakan sebuah *string* yang digunakan untuk mengidentifikasi sebuah konsep secara unik dalam cakupan suatu skema konsep (Semantic Web Deployment Working Group, 2009). Notasi ini berbeda dengan label leksikal karena notasi tidak dikenali sebagai satu kata atau barisan kata dalam bahasa natural (Semantic Web Deployment Working Group, 2009).

Vocabulary SKOS terkait notasi ini adalah skos:notation, yang merupakan *instance* dari owl:DatatypeProperty (Semantic Web Deployment Working Group, 2009). Secara konvensi, properti skos:notation hanya digunakan dengan *typed*

literal, yaitu *string* UNICODE yang dikombinasi dengan URI tipe data, pada posisi objek dari *triple* (Semantic Web Deployment Working Group, 2009).

Sebuah konsep dapat memiliki nol, satu, atau lebih notasi, tidak ada batasan *cardinality* pada property skos:notation (Semantic Web Deployment Working Group, 2009). Penggunaan skos:notation juga dapat digabungkan dengan skos:prefLabel (Semantic Web Deployment Working Group, 2009). Grafik berikut konsisten dengan model data SKOS walau *string* yang sama digunakan sebagai notasi dan label *preferred* (Semantic Web Deployment Working Group, 2009).

```
<Potassium>
  skos:prefLabel "K"@en ;
  skos:notation "K"^^<ChemicalSymbolNotation> .
```

Gambar 2.11 Contoh Grafik Konsisten (5)
(Semantic Web Deployment Working Group, 2009)

Secara konvensi, skos:notation hanya digunakan dengan *typed literals* pada posisi objek pada *triple* dan skos:prefLabel, skos:altLabel, skos:hiddenLabel hanya digunakan dengan *plain literals* pada posisi objek dari *triple* (Semantic Web Deployment Working Group, 2009). Tidak ada literal RDF yang dilengkapi dengan *tag* bahasa dan URI tipe data secara bersamaan (Semantic Web Deployment Working Group, 2009). *Typed literal* tidak memiliki *tag* bahasa dan *plain literal* tidak memiliki URI tipe data (Semantic Web Deployment Working Group, 2009).

Tidak ada *domain* diberikan untuk properti skos:notation sehingga *domain* yang efektif untuk properti ini adalah *class* untuk semua sumber daya (rdfs:Resource) (Semantic Web Deployment Working Group, 2009). Oleh karena

itu, menggunakan skos:notation dengan semua tipe sumber daya konsisten dengan model data SKOS (Semantic Web Deployment Working Group, 2009).

Kelebihan menggunakan SKOS untuk merepresentasikan tesaurus dibandingkan dengan representasi lainnya dapat dilihat pada tabel berikut (Pastor-Sanchez, Mendez, & Rodriguez-Munoz, 2009).

Tabel 2.1 Kelebihan SKOS untuk Representasi Tesaurus Dibandingkan Alternatifnya (Pastor-Sanchez, Mendez, & Rodriguez-Munoz, 2009)

Alternatif SKOS untuk Representasi SKOS	**Kelebihan SKOS**
XML *vocabularies* untuk tesaurus (ZTHES, MESH)	Integrasi ke dalam *Semantic Web* pada level deskriptif menggunakan RDF
Pemetaan Konseptual dan *Topic Maps* (XTM)	Integrasi ke dalam *Semantic Web* pada level logical dengan OWL
Vocabularies RDF lainnya (LIMBER, CERES, ILRT)	Fleksibel, pengembangan terstandarisasi berbasis pada paradigma konseptual
Ontologi OWL	Representasi dan pengelolaan yang lebih sederhana

2.5 SPARQL

Nama SPARQL merupakan akronim rekursif dari SPARQL *Protocol and RDF Query Language* (DuCharme, 2011). Bagian nama "RQL" menunjukkan bahwa SPARQL dirancang untuk melakukan *query* RDF (DuCharme, 2011). Akan tetapi, tidak terbatas pada melakukan *query* data yang disimpan dalam satu dari format RDF (DuCharme, 2011). Bagian "*Protocol*" dari nama SPARQL menunjukkan aturan bagaimana sebuah program *client* dan *server* pemroses SPARQL bertukar SPARQL *queries* dan hasil (DuCharme, 2011). Berbagai macam prosesor SPARQL tersedia untuk menjalankan *queries* terhadap data, baik secara lokal maupun *remote* (DuCharme, 2011). Istilah prosesor SPARQL dan

mesin SPARQL memiliki makna yang sama, yaitu sebuah program yang dapat melakukan *query* SPARQL terhadap sekumpulan data dan mengembalikan hasilnya (DuCharme, 2011).

SPARQL dapat digunakan untuk *query* sumber data publik (DuCharme, 2011). Tidak diperlukan *software* khusus karena koleksi data ini seringkali dibuat tersedia secara publik melalui sebuah SPARQL *endpoint* (DuCharme, 2011). SPARQL *endpoint* merupakan sebuah layanan *web* yang menerima *queries* SPARQL (DuCharme, 2011).

2.6 ARC2

ARC merupakan sebuah sistem RDF yang fleksibel untuk *semantic web* dan praktisi PHP (ARC2, 2012b). Salah satu fitur yang tersedia adalah dapat digunakan untuk berbagai macam *parser*, seperti RDF/XML, N-Triples, Turtle, SPARQL + SPOG, Legacy XML, HTML tag soup, RSS 2.0, Google Social Graph API JSON (ARC2, 2012b). ARC ini tidak berbayar, *open-source*, mudah digunakan, dan berjalan pada kebanyakan lingkungan *web server*, yaitu kompatibel dengan PHP 5.3 E_STRICT (ARC2, 2012b).

ARC dimulai pada tahun 2004 sebagai sistem RDF yang ringan untuk *parsing* dan menserialisasi RDF/XML *files* (ARC2, 2012b). Kemudian, ARC berevolusi ke *framework* yang lebih lengkap dengan fungsionalitas penyimpanan dan *query* (ARC2, 2012b). Pada tahun 2011, ARC2 telah menjadi salah satu dari *libraries* RDF yang harus di-*install* (ARC2, 2012b).

ARC2 merupakan hasil penulisan ulang secara lengkap dari ARC1 (ARC2, 2012a). ARC2 ini tersedia di bawah dua lisensi, yaitu W3C Software License atau GPL (GNU *General Public Lincense*) (ARC2, 2012b). Persyaratan dasar untuk menginstalasi ARC2 adalah sebagai berikut (McIntyre & Durham, 2011).

1. Sebuah *web server* dengan PHP5 atau PHP 4.3 atau lebih tinggi.
2. MySQL 5.0 atau lebih tinggi.
3. Juga bekerja pada lingkungan LAMP.

ARC2 menggunakan sebuah kelas statik yang merupakan satu-satunya yang diperlukan untuk di-*include* (ARC2, 2012a). Komponen yang lainnya akan di-*load* melalui ARC2 tanpa perlu mengetahui jalur secara tepat ke *file* kelas (ARC2, 2012a).

Beberapa komponen dari ARC2 memerlukan *database* MySQL, misalnya RDF *store* dan mesin SPARQL (ARC2, 2012a). *Database* perlu disiapkan terlebih dahulu, tetapi ARC akan secara otomatis membuat tabel-tabel yang diperlukan (ARC2, 2012a). Sebuah *database* dapat digunakan untuk multiple ARC *stores*, setiap *store* dapat dibedakan dengan memberikan nama *custom* yang kemudian digunakan sebagai awalan nama tabel (ARC2, 2012a).

ARC menggunakan kode berorientasi objek untuk komponen-komponennya dan metode-metodenya, tetapi struktur data yang diproses terdiri dari *arrays* berasosiasi yang membuat operasi menjadi lebih cepat dan lebih sedikit mengonsumsi memori (ARC2, 2011). ARC dibangun dengan dua struktur

ini, yaitu *triple sets* dan *resource indexes* (ARC2, 2011). Sebuah *triple set* merupakan *flat array* yang mengandung *triple array* asosiatif (ARC2, 2011).

Sebuah *triple array* tunggal dapat mengandung kunci-kunci sebagai berikut (ARC2, 2011).

1. s: nilai subjek (sebuah URI, Bnode ID, atau Variabel).
2. p: properti URI (atau sebuah Variabel).
3. o: nilai subjek (sebuah URI, Bnode ID, atau Variabel).
4. s_type: “uri”, “bnode”, atau “var”.
5. o_type: “uri”, “bnode”, “literal”, atau “var”.
6. o_datatype: sebuah tipe data URI.
7. o_lang: pengidentifikasi bahasa, misalnya (“en-us”).

Variabel-variabel dibuat oleh ARC Turtle *parser* yang diimplementasi untuk prosesor SPARQL sehingga memperluas spesifikasi Turtle dengan mendukung fitur seperti petik tunggal yang mengapit literal-literal, dan variabel (ARC2, 2011).

Sebuah *resource index* merupakan sebuah *array* asosiatif dari *triple* yang diindeks menggunakan subjek → predikat → objek (ARC2, 2011). ARC mendukung dua bentuk indeks sebagai berikut (ARC2, 2011).

1. *Flat objects*, yang lebih mudah dipakai untuk operasi akses yang disederhanakan, tetapi dapat membuat hilangnya informasi, misalnya ketika tipe objek tidak jelas atau ketika tipe data ada pada *triples* original.
2. Struktur indeks sedikit diperbanyak untuk menyimpan detail objek.

2.7 WordPress Sebagai Content Management System

Content Management System (CMS) merupakan sebuah *sophisticated tool* untuk membangun *websites* dan mengelola konten web (University of Bristol IT Services, 2011). CMS menyediakan sekumpulan *tools* yang terintegrasi untuk membantu *technical* dan *non-technical web publishers* untuk mengelola, membuat, dan mengelola *websites* (University of Bristol IT Services, 2011). CMS ini dapat digunakan untuk menulis, mengedit, dan mempublikasi suatu hasil kerja secara daring (Hedengren, 2010).

WordPress adalah sebuah *Content Management System* (CMS), walau pada mulanya hanyalah untuk publikasi *blog* (Hedengren, 2010). Saat ini WordPress lebih dari sekadar *blog* (Hedengren, 2010). Pada dasarnya, WordPress sebagai CMS dapat digunakan untuk mengelola konten tertulis, gambar, suara, dan video (Hedengren, 2010). WordPress juga dapat digunakan untuk mendukung sistem yang bukan blog, sebagaimana CMS secara tradisional (Hedengren, 2010).

WordPress menjadi pilihan jika memerlukan CMS yang:

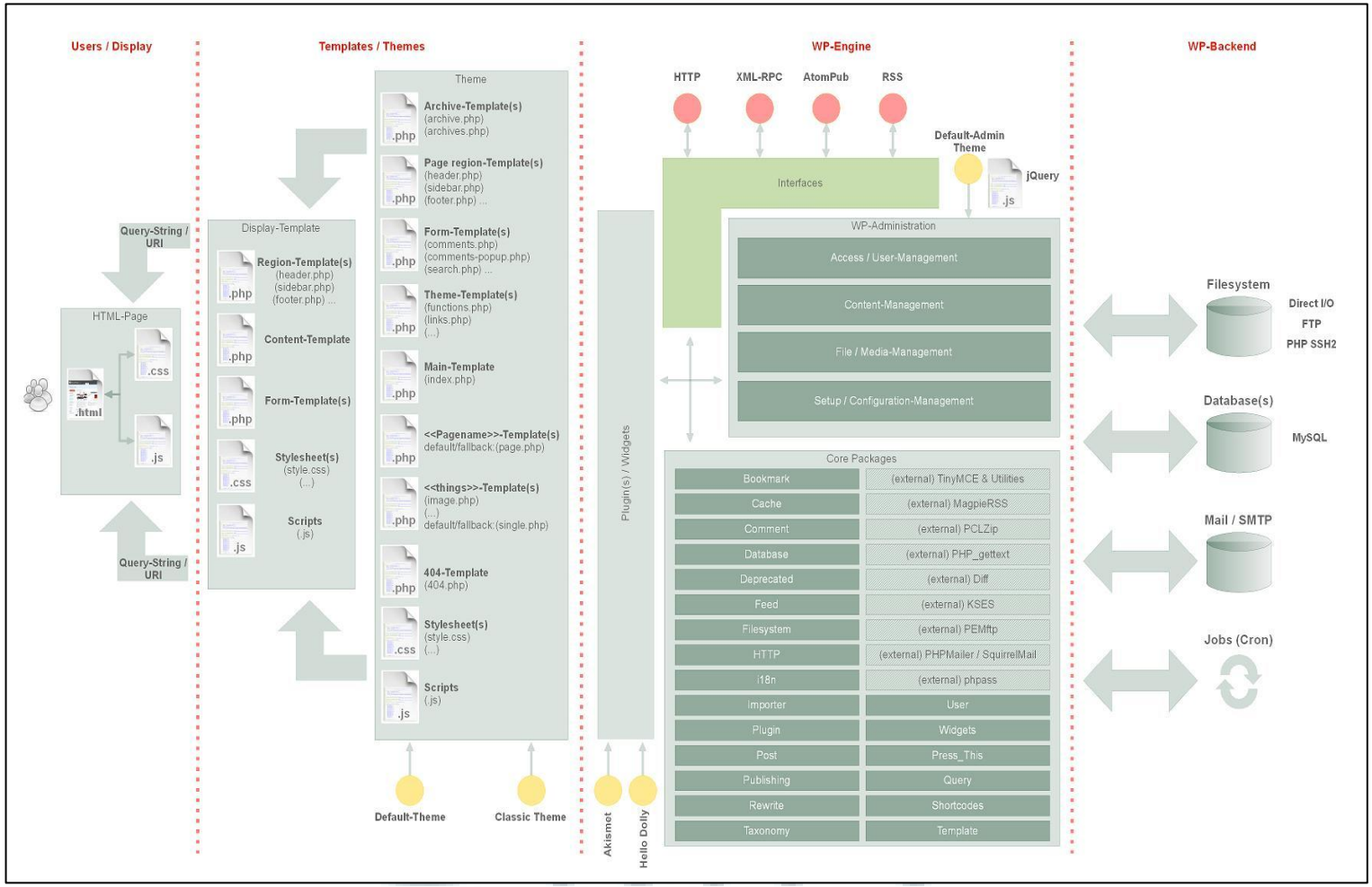
1. *open source* dan tidak berbayar,
2. cepat dan mudah digunakan,
3. mudah dikembangkan,
4. mudah dirancang dan dibuat *plugin*-nya,
5. bekerja secara baik dengan teks, dan
6. cukup baik dengan gambar-gambar (Hedengren, 2010).

WordPress merupakan pilihan CMS yang baik, terutama jika membangun sebuah *editorial site* (Hedengren, 2010).

Terdapat beberapa bagian yang bekerja sama agar *blog* dapat bekerja dengan benar (Bell, 2013). Jika menggunakan WordPress, bagian-bagian ini telah ditangani oleh WordPress (Bell, 2013). Arsitektur WordPress secara sederhana yang menjelaskan hubungan antara *software* dan *hardware* yang berbeda terbagi menjadi tiga bagian utama sebagai berikut (Bell, 2013).

1. WordPress *software*: *software* yang membuat dan menjalankan *blog*, termasuk halaman-halaman *websites*, *themes*, dan *add-ons*.
2. PHP *software* dan *scripts*: PHP (*Hypertext Preprocessor*), merupakan bahasa *scripting*, digunakan untuk membuat halaman *web* dinamis melalui WordPress *software* dan berkomunikasi dengan MySQL *database* untuk mengumpulkan data dan membangun halaman dinamis.
3. MySQL *database*: menyimpan semua informasi yang membangun *blog*, yaitu semua *entries* dan pengaturan konfigurasi.

Arsitektur WordPress ini, secara lebih detail dapat digambarkan dengan gambar 2.12 berikut.



Gambar 2.12 Arsitektur WordPress
(WordPress.org, 2009)

2.8 Plugin WordPress

Plugin WordPress digunakan ketika ingin mengembangkan fungsionalitas WordPress dengan berbagai fitur-fitur tambahan (Hedengren, 2010). *Plugin* merupakan salah satu cara untuk menambahkan fitur apapun yang diinginkan (Hedengren, 2010). *Plugins* WordPress memperbolehkan modifikasi yang mudah, kustomisasi, dan meningkatkan sebuah *blog* WordPress (WordPress.org, 2013). Daripada mengubah pemrograman inti dari WordPress, fungsionalitas dapat ditambah dengan *plugins* WordPress (WordPress.org, 2013).

Definisi sederhana dari *plugin* WordPress adalah sebuah program, atau kumpulan dari satu atau lebih fungsi, ditulis menggunakan bahasa *scripting* PHP (*Hypertext Preprocessor*), yang menambah sekumpulan fitur-fitur atau layanan spesifik ke WordPress *weblog*, yang dapat dengan mudah diintegrasikan dengan *weblog* menggunakan *access points* dan metode-metode yang disediakan oleh WordPress *Plugin Application Program Interface* (API) (WordPress.org, 2013).

Hal yang perlu dilakukan agar WordPress dapat menemukan suatu *plugin* adalah satu *file* dengan pembuka yang mengidentifikasi *plugin* (Hedengren, 2010). Pembuka yang mengidentifikasi *plugin* ini diletakkan pada bagian paling atas *file* PHP utama *plugin* dan berisi informasi *plugin* standar (WordPress.org, 2013). Meletakkan *file plugin* ini pada wp-content/plugins akan membuat *plugin* yang dibuat terdaftar di bagian Plugins pada WordPress (Hedengren, 2010). Aktifkan *plugin* dan *plugin* WordPress ini menjadi dapat digunakan (Hedengren, 2010). Gambar berikut menunjukkan format pembuka yang mengidentifikasi *plugin* WordPress.

```
<?php
/*
Plugin Name: Name Of The Plugin
Plugin URI: http://URI_Of_Page_Describing_Plugin_and_Updates
Description: A brief description of the Plugin.
Version: The Plugin's Version Number, e.g.: 1.0
Author: Name Of The Plugin Author
Author URI: http://URI_Of_The_Plugin_Author
License: A "Slug" license name e.g. GPL2
*/
?>
```

Gambar 2.13 Pembuka yang Mengidentifikasi *Plugin* WordPress (WordPress.org, 2013)

Informasi minimum yang dibutuhkan WordPress untuk mengenal *plugin* adalah baris *Plugin Name* (WordPress.org, 2013). Setelah pembuka standar yang mengidentifikasi *plugin*, informasi mengenai lisensi *plugin* dapat diberikan, tetapi hal ini bersifat opsional (WordPress.org, 2013).

2.9 Extensible Markup Language

The eXtensible Markup Language (XML) merupakan sebuah bahasa untuk menandai dokumen-dokumen dan pesan-pesan dengan *tags* yang dapat membuat mesin lebih mudah untuk melakukan *parse* data dari *files* (Pollock, 2009).

XML saat ini merupakan cara yang terkenal dan efektif untuk bertukar informasi (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). XML *languages* sesuai untuk *well-defined syntax* dan kompatibel dengan banyak *parsers* yang tersedia (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). XML juga menyediakan solusi yang efektif bagi masalah *syntax* pada berbagi data (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). *Syntax* mengacu pada cara informasi ditransfer (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Walau terdapat batasan yang keras pada stuktur XML, kurangnya *reserved keywords* atau kosakata istilah yang sudah

ditetapkan membuat XML menjadi cukup fleksibel untuk digunakan secara luas pada berbagai macam aplikasi (Hebeler, Fisher, Blace, & Perez-Lopez, 2009).

Di luar dari fleksibilitasnya, XML tidak membahas masalah berbagi data semantik (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Elemen-elemen dan atribut-atribut XML tidak memiliki makna sendiri (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). *Tag* yang bersarang pada XML secara sederhana merupakan relasi yang tidak terdefinisi (Pollock, 2009). Oleh karena semantik yang lemah ini, skema *tag* XML ini dapat berarti segala sesuatu (Pollock, 2009). Dengan demikian, tugas untuk menggabungkan dua dokumen XML memiliki beban kerja yang signifikan (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Hal ini dikarenakan fokus dari usaha *interoperability* pada XML belum pada masalah berbagi data semantik (Hebeler, Fisher, Blace, & Perez-Lopez, 2009). Akan tetapi, hal ini dapat dimengerti karena masalah berbagi data yang *syntactic* harus diatasi sebelum masalah berbagi data semantik menjadi isu (Hebeler, Fisher, Blace, & Perez-Lopez, 2009).

Kebanyakan orang membuat kesalahan dengan berpikir bahwa XML dapat menjadi sebuah model data atau XML dapat menjadi sebuah *general-purpose tagging* dan *metadata framework* untuk *software applications* (Pollock, 2009). Akan tetapi, secara fundamental, XML merupakan sebuah *document markup language*, bukan sebuah bahasa untuk memodelkan data (Pollock, 2009).

2.10 XML Web Service

Transfer dokumen XML dasar antara sistem-sistem yang tidak kompatibel, misalnya antara dua *databases* yang berbeda, sangatlah sederhana (Powell, 2007). Akan tetapi, pada faktanya, kebutuhan untuk menulis kode *custom* pada kedua sistem yang melakukan transfer agar dapat dikonversi itu sangat memakan waktu (Powell, 2007). Selain itu, pemetaan terus-menerus dokumen-dokumen XML antar dua poin yang berbeda dalam sebuah transfer, di mana poin transfer tersebut dapat saja merupakan dua pihak yang sangat berbeda, dapat menimbulkan banyak masalah (Powell, 2007). Walaupun misalnya pihak sumber dan target membuat persetujuan untuk menggunakan tipe data yang sama, struktur data yang digunakan dapat sangat berbeda secara logik (Powell, 2007). Selain itu, masalah juga dapat muncul oleh karena perbedaan sistem operasi yang digunakan, perbedaan *platform* perangkat keras, atau bahkan mesin *database* yang berbeda (Powell, 2007). Oleh karena itu, dibutuhkan suatu cara untuk mengorganisasi transfer data ini agar dapat dimengerti secara universal sehingga dapat membatasi bahkan menghilangkan kebutuhan *custom coding* yang mahal dan memakan waktu (Powell, 2007). Jawaban terhadap masalah ini adalah berbagi data XML yang tidak hanya sekadar XML, tetapi menggunakan suatu standar yang umum untuk semua *platforms* (Powell, 2007).

Sebuah *web service* pada dasarnya adalah penyedia layanan informasi melalui *local area network* (LAN), *wide area network* (WAN), atau bahkan internet pada skala besar (Powell, 2007). Menurut definisi, *web service* adalah suatu layanan yang menyediakan informasi melalui internet (Powell, 2007).

Transfer informasi dapat ditangani dengan menggunakan data XML (Powell, 2007).

2.11 Tesaurus

Encyclopedia of Library and Information Science (1980) mengatakan bahwa istilah tesaurus berasal dari bahasa Yunani dan Latin, *thesauros*, yang artinya tempat penyimpanan harta benda, kekayaan atau gudang (Dewi, 2006). *The American Dictionary Webster's* mendefinisikan istilah tesaurus sebagai buku berisi kata atau informasi mengenai bidang tertentu atau sekumpulan konsep, khususnya kamus sinonim (Prakasa, 2008). Pada Kamus Besar Bahasa Indonesia Daring, definisi tesaurus juga dituliskan sebagai buku referensi berupa daftar kata dengan sinonimnya (Pusat Bahasa Departemen Pendidikan Nasional, 2008). Tesaurus dapat digunakan untuk memperluas kosakata ketika menulis dan meragamkan representasi dari konsep yang sama (Prakasa, 2008).

2.12 Kateglo

Kateglo merupakan kamus, tesaurus, dan glosarium daring untuk bahasa Indonesia (Programmableweb, 2013). Nama Kateglo diambil dari akronim unsur layanan yang disediakan, yaitu ka(mus), te(saurus), dan glo(sarium) (Kateglo, 2009b).

Kateglo *Application Program Interface* (API) dibuat untuk memungkinkan para pengembang memanfaatkan data yang disediakan oleh Kateglo (Kateglo, 2009a). Kateglo API ini, untuk tahap awal, baru dapat

mengakses modul kamus (Kateglo, 2009a). Kateglo API menggunakan RESTful untuk mekanisme pemanggilan dan respon diformat dalam *eXtensible Markup Language* (XML) atau *JavaScript Object Notation* (JSON) (Programmableweb, 2013).

REST, yang merupakan singkatan dari *Representational State Transfer*, merupakan pendekatan yang lebih sederhana dibandingkan XML-RPC atau SOAP, menggunakan metode HTTP (*Hypertext Transfer Protocol*) standar seperti GET, POST, dan PUT untuk mengirim dan mengambil data XML (David, 2004). REST tidak menjadi standar, baik formal maupun informal (David, 2004). Kelebihan dari REST adalah tidak dibutuhkan *extensions* atau *tool* khusus untuk membuat *web service* (David, 2004). Spesifikasi protokol HTTP mengandung semua yang dibutuhkan untuk mentransmit dan menerima pesan XML (David, 2004).

2.13 PoolParty Thesaurus

Plugin WordPress PoolParty Thesaurus membantu membuat WordPress *blogs* dan *websites* menjadi lebih mudah dimengerti (PoolParty Thesaurus, 2011). *Plugin* ini akan mengimpor *controlled vocabulary*, yaitu tesaurus berbasis *Simple Knowledge Organization System* (SKOS), atau mengambil sebuah tesaurus dari SPARQL *endpoint* (publik) melalui *web* (PoolParty Thesaurus, 2011). Tesaurus diimpor ke sistem dan di-*query* melalui ARC2 (PoolParty Thesaurus, 2011). Berbasis pada tesaurus, *terms* dalam artikel akan secara otomatis di-*highlight*, definisi dari *terms* tersebut juga disediakan ketika melakukan *mouseover*, dan

hubungan ke tesaurus *terms* juga dihasilkan *on-the-fly* (PoolParty Thesaurus, 2011). Tesaurus ini tersedia sebagai sumber daya tambahan pada *blog* dan dapat dinavigasi untuk mempelajari lebih tentang *knowledge domain* (PoolParty Thesaurus, 2011). *Plugin* ini juga bekerja pada *blog* yang multibahasa (PoolParty Thesaurus, 2011). Definisi akan diambil otomatis menggunakan Dbpedia (*linked data*) ketika tesaurus yang diimpor kekurangan definisi (PoolParty Thesaurus, 2011).

Ketika menggunakan *plugin* PoolParty Thesaurus ini, akan dihasilkan dua halaman tambahan pada *blog* yang dihasilkan secara otomatis (PoolParty Thesaurus, 2011). Pada dua halaman ini, tesaurus akan dihasilkan dan digunakan sebagai media pencarian glosarium pada *blog* (PoolParty Thesaurus, 2011). Pada halaman glosarium utama, akan ditampilkan keseluruhan konsep secara urut alfabetis dan dapat difilter dengan huruf pertama (PoolParty Thesaurus, 2011). Halaman glosarium kedua merepresentasikan tampilan detail dari setiap konsep (PoolParty Thesaurus, 2011).

Setiap *post* akan dianalisis secara otomatis untuk menemukan kata-kata dan frasa-frasa yang sesuai dengan label suatu konsep dalam tesaurus (PoolParty Thesaurus, 2011). Yang paling pertama sesuai akan di-*highlight* secara otomatis (PoolParty Thesaurus, 2011). Sebuah *mouseover tooltip* akan ditampilkan untuk menunjukkan deskripsi singkat dari kata atau frasa dan hubungan yang mengarah ke deskripsi lebih detail pada halaman glosarium (PoolParty Thesaurus, 2011).

2.14 Scenario-Based Modeling

Scenario-based modeling merupakan pemodelan terhadap analisis dan perancangan kebutuhan *software* yang menggunakan skenario untuk mendapatkan fungsionalitas sistem dari sudut pandang pengguna (Bai, Tsai, & Paul, 2002). Metode ini efektif ketika sistem yang dibangun belum memiliki *requirement* yang jelas sejak awal (Hsia & Asur, 1991). Narasi awal, *paper models*, dan representasi grafis dibangun untuk merepresentasikan fungsi sistem final (Hsia & Asur, 1991).

Teknik *use-oriented* banyak digunakan dalam analisis dan perancangan kebutuhan *software* (Bai, Tsai, & Paul, 2002). Pada *scenario-based modeling*, skenario digunakan untuk mendapatkan fungsionalitas sistem (Bai, Tsai, & Paul, 2002). Sebuah skenario mendeskripsikan suatu fungsi dari sudut pandang pengguna (Bai, Tsai, & Paul, 2002). *Use cases* dan skenario penggunaan memfasilitasi pemahaman tentang sistem dan menyediakan bahasa yang lebih umum untuk komunikasi antara berbagai pihak, termasuk pelanggan, *software analysts*, dan *software developers* (Bai, Tsai, & Paul, 2002). Hal ini dikarenakan *use case* secara relatif mudah untuk dimengerti (Pressman, 2005). Model skenario ini dapat dideskripsikan dengan perancangan *software* yang direpresentasikan dengan *Unified Modeling Language* (UML) (Bai, Tsai, & Paul, 2002). Model analisis dengan UML dimulai dengan membuat skenario-skenario dalam bentuk *use cases*, *activity diagrams*, dan *swimlane diagram* (Pressman, 2005).

Use cases saat ini banyak digunakan terlepas dari pendekatan yang digunakan pada pengembangan *software* (Kendall & Kendall, 2008). Suatu *use case* menjelaskan interaksi yang terjadi antara produser dan konsumer informasi

dan sistem tersebut (Pressman, 2005). Pada banyak kasus, representasi grafis dari suatu skenario penggunaan tidak dibutuhkan (Pressman, 2005). Akan tetapi, representasi dalam bentuk diagram dapat memfasilitasi pemahaman, terutama ketika skenario tersebut kompleks (Pressman, 2005). Skenario penggunaan dalam bentuk diagram dapat direpresentasikan menggunakan *use case diagram* (Pressman, 2005). UML *activity diagram* pada *scenario-based modeling* ini akan melengkapi *use case* dengan menyediakan representasi grafis dari aliran interaksi dalam suatu skenario spesifik (Pressman, 2005). Selain itu, UML *swimlane diagram*, yang merupakan variasi dari *activity diagram*, memperbolehkan pemodel untuk merepresentasikan aliran aktivitas yang dideskripsikan melalui *use case* sekaligus mengindikasikan aktor yang bertanggung jawab terhadap aksi tersebut (Pressman, 2005).