BAB II
LANDASAN TEORI

2.1 Web

Web merupakan suatu ruang informasi di mana sumber-sumber daya yang berguna diidentifikasi oleh pengenal global yang disebut *Uniform Resource Identifier* (URI). URL dapat diibaratkan suatu alamat, di mana alamat tersebut terdiri atas [1] :

1. Protokol yang digunakan oleh suatu *browser* untuk mengambil informmasi.
2. Nama komputer (*server*) di mana informasi tersebut berada.
3. Jalur atau *path* serta nama *file* dari suatu informasi.

Format umum dari URL adalah sebagai berikut:

"protokol_transfer://nama_host/path/nama_file"

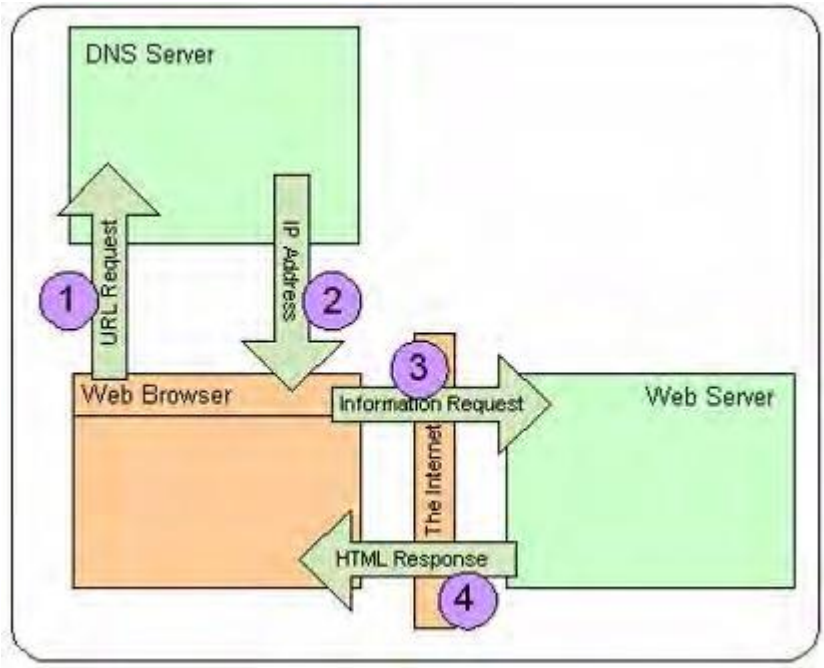
Contoh: http://www.mine.com/e-jurnal/index.html

Dari contoh tersebut dapat disimpulkan bahwa:

1. http adalah protokol yang digunakan.
2. www.mine.com adalah nama *host* atau *server* komputer di mana informasi yang dicari berada.
3. e-jurnal adalah jalur atau *path* dari informasi yang dicari.
4. index.html adalah nama file di mana informasi tersebut berada.

Sebuah halaman web diakses dengan menggunakan *web browser* dengan menuliskan URL-nya atau mengikuti *link* yang menuju kepadanya. *Uniform Resource Locator* (URL) akan menunjukkan lokasi dokumen yang dikelola oleh *web server*. URL diubah menjadi alamat IP *web server* yang bersangkutan. *Browser* kemudian mengirimkan request *Hypertext Transfer Protocol* (HTTP) ke *web server* dan akan menjawab dokumen yang diminta dalam format *Hypertext Markup Language* (HTML). HTTP adalah suatu protokol yang menentukan aturan yang perlu diikuti oleh *web browser* dalam meminta atau menggambil suatu dokumen dan oleh *web server* dalam menyediakan dokumen yang diminta *web browser*. Protokol ini merupakan komunikasi jaringan komputer yang diatur dengan protokol yang

memungkinkan beragam jaringan komputer untuk berkomunikasi. Protokol ini secara resmi dikenal dengan *Transmission Control Protocol* (TCP/IP) yang merupakan cara untuk mempaketkan sinyal elektronik sehingga data tersebut dapat dikirim ke komputer lain. Gambar 2.1 menggambarkan cara kerja web yang diakses.



Gambar 2.1 Cara Kerja Web yang Diakses

Web dapat dikategorikan menjadi dua yaitu web statis dan web dinamis atau interaktif. Web statis adalah web yang menampilkan informasi-informasi yang sifatnya statis atau tetap, sedangkan web dinamis adalah web yang menampilkan informasi serta dapat berinteraksi dengan user yang sifatnya dinamis. Untuk membuat web dinamis dibutuhkan pemrograman web yang mempunyai dua kategori, yaitu [1] :

1. Server – side Programming

   Perintah-perintah program atau script dijalankan di web server, kemudian hasilnya dikirimkan ke browser dalam bentuk HTML.

2. Client – side Programming

   Perintah program dijalankan di web browser sehingga ketika client meminta dokumen yang mengandung script, maka script tersebut akan didownload dari servernya kemudian dijalankan di browser yang bersangkutan.

Program web yang tergolong dalam server side seperti CGI/Perl. Active Server Pages (ASP), Java Server Pages (JSP), PHP dan ColdFussion (CFM). Sedangkan yang tergolong client side seperti Javascript, VBScript dan HTML. Teknologi server side yang dilakukan dalam aplikasi ini adalah PHP. PHP termasuk dalam produk open source sehingga source code dapat diubah dan didistribusikan secara bebas. PHP dapat berjalan diberbagai web server misalnya IIS, Apache dan PWS. Adapun kelebihan-kelebihan PHP adalah sebagai berikut:

1. PHP mudah dibuat dengan kecepatan akses tinggi.
2. PHP dapat berjalan dalam web server yang berbeda dan dalam sistem operasi yang berbeda pula.
3. PHP diterbitkan secara gratis.
4. PHP adalah termasuk bahasa yang embedded (bisa diletakkan dalam tag HTML).

2.1.1 Perkembangan Web

Dalam perkembangan teknologi web, banyak praktisi yang memberi label perkembangan web dengan Web 1.0, Web 2.0 dan Web 3.0. Sebenarnya tidak ada kesepakatan adanya versi dalam aplikasi web tetapi hanya untuk memudahkan perkembangannya saja.

1. Web 1.0

   Web 1.0 memiliki sifat yang sedikit interaktif dan dikembangkan untuk pengaksesan informasi. Sifat dari Web 1.0 adalah *read* karena *user* hanya akan membaca informasi yang ditampilkan web.

2. Web 2.0

   Menurut Tim O'Really, Web 2.0 dapat didefinisikan sebagai berikut : "Web 2.0 adalah revolusi bisnis di industri komputer yang disebabkan oleh penggunaan internet sebagai *platform* dan merupakan suatu percobaan untuk memahami berbagai aturan untuk mencapai keberhasilan pada *platform* baru tersebut. Salah satu aturan utama adalah membangun aplikasi yang mengeksploitasi efek jaringan untuk mendapatkan lebih banyak lagi pengguna aplikasi tersebut" [2].

   Kemudahan interaksi antara *user* dengan sistem merupakan tujuan dibangunnya teknologi web 2.0. Sifat dari web 2.0 adalah *read* dan *write*. Perkembangan web 2.0 lebih menekankan pada perubahan cara berpikir dalam menyajikan konten dan tampilan di dalam sebuah web. Web 2.0 diaplikasikan sebagai bentuk penyajian halaman web yang bersifat sebagai program dekstop pada umumnya seperti windows. Implementasi dapat dilihat pada aplikasi *spreadsheet* pada Google yang merupakan aplikasi untuk operasi mengolah angka seperti MS. Excel. Aplikasi tersebut dapat diakses secara *online* tanpa *user* harus menginstalnya terlebih dahulu. Web 2.0 pada umumnya adalah suatu teknologi yang gratis atau lebih dikenal dengan sebutan *Open Source* dan sangat memudahkan untuk *share, upload* dan *download* data.

3. Web 3.0

   Web 3.0 adalah generasi ketiga dari layanan Internet berbasis web. Menurut Tim Berners Lee, Web 3.0 sebagai sebuah sarana bagi mesin untuk membaca halaman-halaman web [2]. Hal ini berarti bahwa mesin akan memiliki kemampuan yang sama dengan manusia dalam membaca web. Web 3.0 berhubungan dengan konsep semantik web yang memungkinkan manusia dapat berkomunikasi dengan mesin pencari yang juga mampu

menyedikan keterangan – keterangan yang relevan tentang informasi yang dicari. Web 3.0 memiliki beberapa standar operasional agar dapat menjalankan fungsinya dalam menampung metadata, yaitu *Resource Description Framework* (RDF), dan *Ontology Web language* (OWL).

Dalam perkembangan web harus diimbangi dengan kecepatan untuk mengakses karena faktor yang menentukan kinerja aplikasi adalah kecepatan akses jaringan dan Internet. Oleh karena itu diperlukan *bandwidth* yang cukup dalam menjalankan suatu aplikasi.

2.2 Semantik Web

2.2.1 Pengertian Semantik

Semantik (Bahasa Yunani : semantikos, memberikan tanda, penting, dari kata sema, tanda) adalah cabang linguistik yang mempelajari makna yang terkandung pada suatu bahasa, kode atau jenis representasi lain. Semantik biasanya dikontraskan dengan ekspresi makna sintaksis yaitu pembentukan simbol kompleks dari simbol yang lebih sederhana.

Web Semantik merupakan pengembangan dari *world wide web* dimana content web yang ditampilkan tidak hanya dalam bahasa format manusia yang umum (*natural language*) tetapi juga dalam format yang dapat dibaca dan digunakan oleh mesin (*software*). Web Semantik memiliki informasi yang dimiliki oleh mesin yang memiliki kecerdasan buatan sehingga mampu menemukan dan mengintegrasikan informasi dengan mudah. Dengan demikian fungsi web menjadi wadah bagi pertukaran data, informasi dan pengetahuan melalui kecerdasan buatan sehingga mengerti keinginan user dimana dapat diinstruksikan untuk mengambil informasi sesuai kriteria tertentu. Tujuan dari web semantik adalah mengatur informasi dan prosedur. Fundamental dalam pembangunan semantic web adalah kreasi dan semantic metadata. Metadata terdiri dari dua bagian, yaitu :

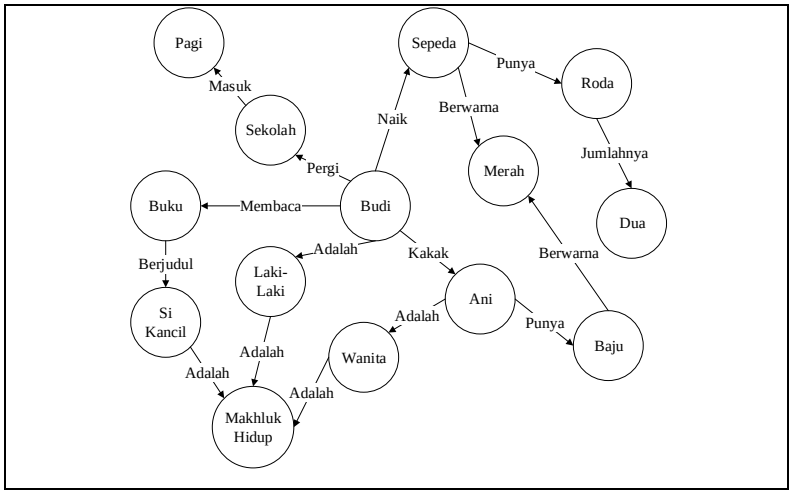
1. Penggambaran sebuah dokumen. Contohnya adalah halaman web atau bagian dari suatu dokumen seperti sebuah paragraf.
2. Penggambaran entitas didalam suatu dokumen. Contohnya adalah seseorang atau sebuah perusahaan.

Pada semua kasus, yang terpenting bahwa metadata adalah semantik, yang menggambarkan semua isi dari dokumen tersebut.

Saat membuat aplikasi semantik, sebenarnya terdapat dua variabel yang dibangun secara sekaligus. Variabel pertama adalah web terdiri dari protocol komunikasi dan format web. Terdapat standar web semantik yang direkomendasikan W3C seperti RDF, OWL dan SPARQL. Variabel lainnya adalah semantik merepresentasikan makna dari web data.

2.2.2 Jaringan Semantik

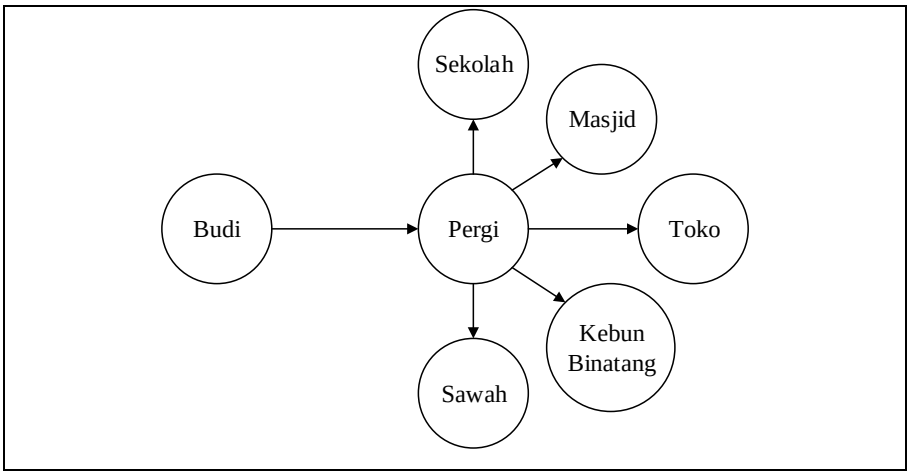
Jaringan semantik merupakan gambaran pengetahuan grafis yang menunjukkan hubungan antar berbagai obyek. Jaringan semantik terdiri dari lingkaran-lingkaran yang menunjukkan obyek dan informasi tentang obyek-obyek tersebut. Obyek di sini bisa berupa benda atau peristiwa. Antara 2 obyek dihubungkan oleh *arc* yang menunjukkan hubungan antar obyek. Gambar 2.2 merupakan contoh representasi pengetahuan dengan menggunakan jaringan semantik.



Gambar 2.2 Contoh Jaringan Semantik

Salah satu kelebihan dari jaringan semantik adalah ‘bisa mewariskan’. Sebagai contoh, pada gambar 2.2 ada garis yang menghubungkan antara Budi dengan laki-laki, dan laki-laki ke makhluk hidup. Sehingga apabila ada pertanyaan: Apakah Budi makhluk hidup? Maka kita bisa merunut garis dari makhluk hidup, kemudiian ke laki-laki, dan akhirnya ke Budi. Sehingga terbukti bahwa Budi adalah makluk hidup.

Sistem jaringan semantik ini selalu tergantung pada jenis masalah yang akan dipecahkan. Jika masalah itu bersifat umum, makan hanya memerlukan sedikit rincian. Jika ternyata masalah itu banyak melibatkan hal-hal lain, maka di dalam jaringan awalnya diperlukan rincian dari node awal Budi, apabila Budi hendak pergi ke berbagai tempat. Node Budi dihubungkan dengan node baru, yaitu pergi [7].



Gambar 2.3 Perluasan Jaringan Semantik

2.2.3 Komponen-Komponen dalam *Semantic Web*

Pembuatan semantic web dimungkinkan dengan adanya sekumpulan standar yang dikoordinasi oleh *World Wide Web Consortium* (W3C). Standar yang paling penting dalam pembangunan semantic web adalah XML, XML Schema, RDF, RDF Schema dan OWL. Komponen – komponen dalam semantic web ini yang memungkinkan komunikasi dan interaksi pada level mesin [2].

1. XML dan XML Schema

   Extensible Markup Language (XML) merupakan bahasa markup yang didesain untuk menjadi sarana yang mudah dalam mengirimkan dokumen melalui web. Berbeda dengan *Hypertext Markup Language* (HTML), XML memungkinkan penggunanya untuk mendefinisikan custom tag. XML Schema merupakan bahasa yang digunakan untuk mendefinisikan sekumpulan aturan (*schema*) yang harus dipatuhi oleh dokumen XML. Struktur dari dokumen XML yang dibuat harus sesuai dengan schema yang telah didefinisikan tersebut.

2. RDF dan RDF Schema

   Resource Description Framework (RDF) adalah spesifikasi yang dibuat oleh W3C sebagai metode umum untuk memodelkan informasi dengan menggunakan sekumpulan format sintaks. Ide dasar dari RDF adalah bagaimana kita dapat membuat pertanyaan mengenai sebuah resource web dalam bentuk ekspresi subjek (S), predikat (P), objek (O). Dalam terminology RDF, SPO ini seringkali disebut dengan istilah N-triple. Subjek mengacu pada resource yang ingin dideskripsikan. Predikat merupakan komposisi yang menerangkan sudut pandang dari subjek yang dijelaskan objek, sementara subjek dan objek merupakan entitas. Objek di dalam RDF dapat menjadi subjek yang diterangkan oleh objek lainnya. Dengan inilah objek dapat berupa masukan yang dapat diterangkan secara jelas dan detail, sesuai dengan keinginan pengguna yang memberikan masukan. Dengan menggunakan RDF, website dapat menyimpan dan melakukan pertukaran informasi antar web. RDF telah digunakan pada aplikasi-aplikasi berikut:

1. RDF Site Summary (RSS)
   RSS memberikan informasi update sebuah website tanpa pengunjung perlu mengunjungi website tersebut.
2. Friend of a Friend (FOAO)
   Didesain untuk mendeskripsikan orang-orang, ketertarikan dan hubungan mereka.
3. Semantically-Interlinked Online Communities (SIOC)
   Menerangkan komunitas online dan menciptakan koneksi antara diskusi berbasis intenet seperti message board, blog maupun mailing list. RDF schema dapat dipandang sebagai kamus data atau vocabulary untuk mendeskripsikan properties dan classes dari resources RDF.

1. Ontology Web Language (OWL)
   OWL adalah suatu bahasa yang dapat digunakan oleh aplikasi-aplikasi yang bukan sekedar menampilkan informasi tersebut pada manusia melainkan juga yang perlu memproses isi informasi. Ontologi sendiri dapat didefinisikan sebagai suatu cara untuk mendeskripsikan arti dan relasi dari istilah-istilah tersebut dengan cara yang lebih mudah atau dengan pengertian lain adalah representasi istilah beserta hubungannya. Ketika informasi yang ada dalam dokumen perlu untuk diproses oleh aplikasi atau mesin, OWL dapat digunakan untuk merepresentasikan makna suatu istilah secara eksplisit sekaligus hubungan antara istilah-istilah tersebut. Dengan mengunakan OWL, kita dapat menambah *vocabulary* tambahan disamping semantic formal yang telah dibuat sebelumnya menggunakan XML, RDF dan RDF Schema. Hal ini sangat membantu penginterpretasian mesin yang lebih baik

terhadap isi web. Untuk mendeskripsikan *properties* dan *classes*, OWL menambahkan *vocabulary* seperti:

1. "*among others*"
2. Relasi antar *classes* (misalnya : "*disjointness*")
3. Kardinalitas
4. Kesamaan (*equality*)
5. *Characteristic property*
6. *Enumerated classes*

OWL memiliki tiga *sub-language* yaitu:

1. OWL Lite

   Mendukung pengguna yang memerlukan klasifikasi hirarki dan dalam batasan yang sederhana.

2. OWL DL

   Mendukung konstruksi seluruh OWL, tetapi hanya dapat digunakan pada batasan tertentu.

3. OWL Full

   Diperuntukkan bagi pengguna yang menginginkan maksimum penggunaan dan kebebasan sintaksis.

2.2.4 Keuntungan Semantik

Keuntungan yang dimiliki oleh *semantic web* adalah sebagai berikut:

1. Waktu yang diperlukan untuk mendapatkan informasi yang dicari lebih singkat.
2. Pekerjaan pencarian yang dilakukan manusia dapat digantikan oleh mesin.

2.3 Ontologi

2.3.1 Pengertian Ontologi

Dalam literatur kecerdasan buatan terdapat beberapa pengertian ontologi. Ontologi adalah istilah yang dipinjam dari filosofi yang mengacu kepada ilmu untuk menggambarkan jenis-jenis entitas di

dunia dan bagaimana mereka berhubungan. Menurut Barnaras pada proyek Kactus [3] memberikan definisi ontologi yaitu : "Penjelasan secara eksplisit dari konsep terhadap representasi pengetahuan pada *knowledge base*". Proyek Sensus [3] juga memberikan definisi : "Sebuah ontologi adalah sebuah struktur hirarki dari istilah untuk menjelaskan sebuah domain yang dapat digunakan sebagai landasan untuk sebuah *knowledge base*" [4]. Pengertian lain mengemukakan ontologi adalah sebuah uraian formal yang menjelaskan tentang sebuah konsep dalam suatu domain tertentu (*classes*, terkadang disebut *concepts*), properti dari setiap konsep yang menjelaskan bermacam-macam fitur dan atribut sebuah *concepts* (*slots*, terkadang disebut *roles* atau *properties*) dan batasan pada *slots* (*facets*, terkadang disebut *role restriction*). Sebuah ontologi bersama dengan seperangkat *instances* (menyatakan objek pada suatu domain) dari *class* membentuk sebuah *knowledge base*. Pendapat lain mengatakan bahwa ontologi terbentuk oleh 4 tuple (C,R,I,A), C adalah *concept*, R adalah *relation*, I adalah *instance*, dan A adalah *axiom*. *Axiom* digunakan untuk menyediakan informasi mengenai *class* dan *properties*, sebagai contoh batasan pada *properties*.

2.3.2 Alasan Menggunakan Ontologi

Terdapat beberapa alasan untuk menggunakan ontologi, yaitu [4]:

1. Menjelaskan suatu domain secara eksplisit.
   Memberikan struktur hirarki dari istilah untuk menjelaskan sebuah domain dan bagaimana mereka berhubungan.
2. Berbagi pemahaman dari informasi yang terstruktur.
   Sebagai contoh beberapa website yang berbeda mempunyai informasi medis atau menyediakan servis medis e-commerce. Jika website tersebut dipakai bersama dan dipublikasikan dengan dasar ontologi yang sama maka perangkat lunak dapat mengekstrak dan mengumpulkan informasi dari *site* yang

berbeda. Perangkat lunak tersebut dapat menggunakan kumpulan informasi tersebut untuk menjawab permintaan *user* atau sebagai data input untuk aplikasi lainnya.

3. Penggunaan ulang domain pengetahuan.
   Apabila ingin membangun ontologi yang luas dapat mengintegrasikan dengan beberapa ontologi yang sudah ada
   .

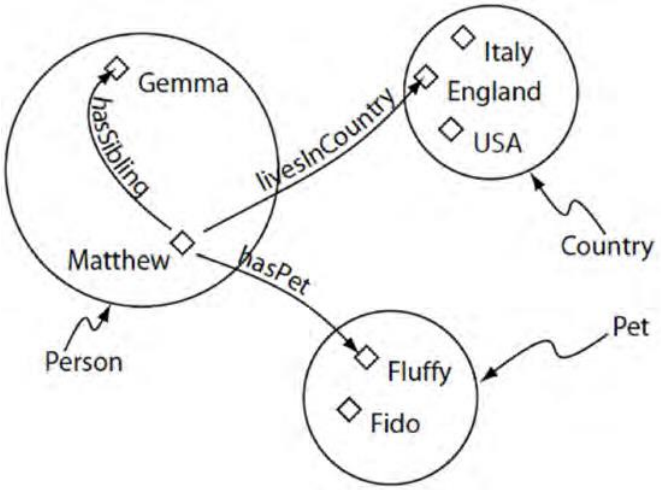
2.3.3 Komponen-Komponen dalam Ontologi

Ontologi memiliki beberapa komponen yang dapat menjelaskan ontologi tersebut, yaitu:

1. *Instance* atau *individual* digunakan untuk merepresentasikan elemen pada suatu domain. Contoh *instance* dapat dilihat pada Gambar 2.4.

Gambar 2.4 Representasi *instance* atau *individual* [5]

2. *Class* merupakan titik pusat ontologi. *Class* menjelaskan sebuah konsep dalam suatu domain yang terdiri dari beberapa *instance* atau *individual*. *Class* juga dikenal sebagai *concept, object* dan *categories*. Sebuah *class* memiliki *subclasses* yang menyatakan *concept* yang lebih spesifik dari *superclass*. Contoh *class* dapat dilihat pada Gambar 2.5.

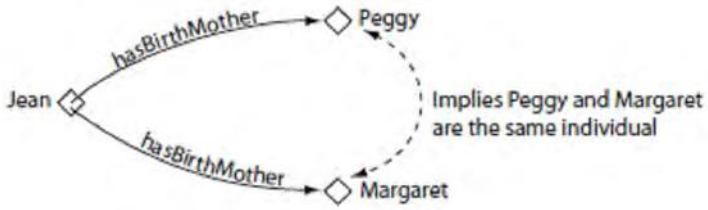


Gambar 2.5 Representasi *class* yang terdiri dari *instance* [5]

Dari gambar tersebut dapat dilihat bahwa *class* Person memiliki *instance* atau *individual* Gemma dan Matthew.

3. *Properties* atau *slot*. *Properties* atau *slot* terdiri dari dua jenis, yaitu *object properties* dan *datatype properties*. *Object properties* akan menghubungkan *instance* dengan *instance* sedangkan *datatype properties* akan menghubungkan *instance* dengan *datatype value* seperti *text, string,* atsu *number*. Pada *object properties*, terdapat beberapa jenis *properties* [5], yaitu:
    1. *Functional Properties*.

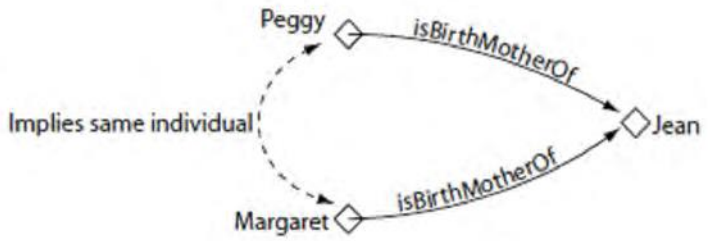
       Functional properties adalah sebuah individu yang berhubungan hanya dengan satu individu. *Functional property* disebut juga sebagai *single valued property* atau *feature*. Sebagai contoh pada Gambar 2.6 menunjukkan individu Jean *hasBirthMother* Peggy dan individu Jean *hasBirthMother* Margaret, *hasBirthMother* merupakan *fuctional property*, dapat disimpulkan bahwa Peggy dan Margaret adalah individu yang sama karena Jean hanya memiliki satu *BirthMother*.



Gambar 2.6 Contoh *Functional Properties* [5].

2. *Inverse Functional Properties.*

   Jika sebuah properti adalah *inverse functional*, maka *inverse property* tersebut adalah *functional*, sebuah individu berhubungan hanya dengan satu individu. Sebagai contoh pada Gambar 2.7 menunjukkan *inverse functional property isBirthMotherOf* yang merupakan *inverse property* dari *hasBirthMother*, *hasBirthMother* adalah *functional*, *isBirthMother* adalah *inverse functional*. Jika dikatakan bahwa Peggy merupakan *birthmother* dari Jean, dapat dikatakan juga bahwa Margaret merupakan *birthmother* dari Jean. Sehingga dapat disimpulkan bahwa Peggy dan Margaret adalah individu yang sama.

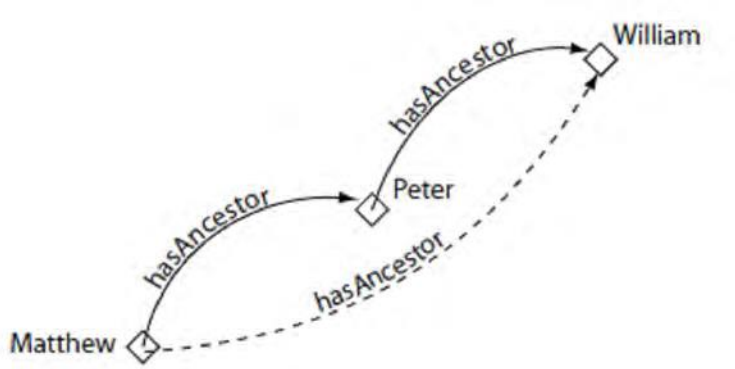


Gambar 2.7 Contoh *Inverse Functional Properties* [5].

3. *Transitive Properties*.

   Jika sebuah properti *transitive*, properti menghubungkan individu A dengan individu B serta menghubungkan individu B dengan individu C, maka

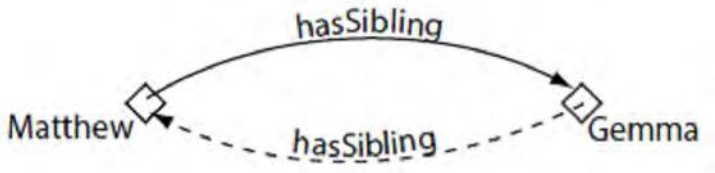
dapat disimpulkan bahwa individu A berhubungan dengan individu C melalui properti P. Sebagai contoh pada Gambar 2.8 menunjukkan *transitive property hasAncestor*. Jika individu Matthew mempunyai *ancestor* Peter, Peter mempunyai *ancestor* William, maka dapat disimpulkan bahwa Mathew mempunyai *ancestor* William.



Gambar 2.8 Contoh *Transitive Properties* [5].

4. *Symetric Properties*.

   Properti P adalah *symetric* jika properti tersebut menghubungkan A ke individu B kemudian menghubungkan B ke individu A. Sebagai contoh pada Gambar 2.9, individu Mathew berhubungan dengan individu Gema melalui *hasSibling property*. Dengan kata lain, jika Mathew bersaudara dengan Gema maka dapat disimpulkan bahwa Gema bersaudara dengan Mathew.



Gambar 2.9 Contoh *Symetric Properties* [5].

2.3.4 Pembangunan Ontologi

Terdapat beberapa tahapan yang dilakukan dalam pengembangan ontologi yaitu [6]:

1. Menentukan domain dan batasan ontologi.

   Dalam mengembangkan ontologi dimulai dengan mendefinisikan domain dan batasan dengan menjawab pertanyaan berikut:

   1. Domain apa yang akan melingkupi ontologi?
   2. Mengapa ontologi digunakan?
   3. Apa jenis pertanyaan terhadap informasi dalam ontologi sehingga perlu menyediakan jawaban?

   Salah satu cara dalam menetukan batasan dalam ontologi adalah dengan membuat daftar pertanyaan yang harus dapat dijawab oleh *knowledge base* atau yang biasa disebut *competency questions* [6].

2. Mempertimbangkan penggunaan ontologi yang sudah ada.

   Ontologi yang sudah ada dapat diperhalus dan diperluas untuk domain dan *task* yang akan dibuat. Penggunaan ontologi yang sudah ada merupakan persyaratan apabila sistem yang akan dibuat akan berinteraksi dengan aplikasi lainnya yang telah dilakukan pada suatu ontologi atau perbendaharaan kata yang dikontrol. Banyak ontologi yang telah tersedia dan dapat dimasukkan dalam penngembangan ontologi yang dilakukan.

3. Menentukan istilah penting dalam ontologi Menulis daftar istilah yang akan dijelaskan ke user. Terdapat beberapa pertanyaan yang dapat membantu penentuan istilah, yaitu:

   1. Istilah apa saja yang akan diperbincangkan?
   2. Apa yang akan menjadi jawaban mengenai istilah–istilah tersebut?
   3. Properti apa saja yang dimiliki istilah tersebut?

Sebagai contoh pada ontologi minuman anggur, istilah yang penting meliputi minuman anggur, anggur, tempat membuat anggur, warna minuman anggur, bentuk, cita rasa dan kadar gula. Pada dasarnya, hal ini dibutuhkan untuk mendapat daftar istilah yang menyeluruh tanpa khawatir tumpang tindih antara *concept*, hubungan antara istilah dan properti dari *concept* atau *concept* tersebut termasuk *class* atau *slot*.

4. Mendefinisikan *class* ontologi dan menyusun *class* dalam hirarki taksonomi (*subclass–superclass*).

   Terdapat beberapa pendekatan dalam pengembangan hirarki *class* [5], yaitu :

   1. Proses pengembangan ***top-down*** dimulai dengan mendefinisikan *concept* umum dalam domain dilanjutkan dengan *concept* yang lebih spesifik.
   2. Proses pengembangan ***bottom-up*** dimulai dengan mendefinisikan *class* yang paling spesifik kemudian dikelompokkan menjadi *class* dengan konsep yang lebih umum.
   3. Proses pengembangan ***combination*** adalah sebuah kombinasi antara pendekatan *top-down* dan *bottom-up*. Mendefinisikan konsep yang menonjol terlebih dahulu kemudian menggeneralisasi dan mengkhususkan konsep tersebut.

5. Mendefinisikan *slot* atau *properties* dan menjabarkan nilai dari *slot* tersebut.
6. Mendefinisikan *facets* pada *slots*.

   Slot dapat memiliki *facet* yang berbeda dalam menggambarkan tipe nilai, nilai tersebut dapat berupa *cardinality* (sejumlah nilai) dan fitur lainnya. Berikut merupakan beberapa *facet* yang umum digunakan:

1. *Slot cardinality*.

   Slot cardinality mendefinisikan sejumlah nilai yang dimiliki oleh *slot*. Beberapa sistem hanya membedakan *single cardinality* (mempunyai satu nilai) dengan *multiple cardinality* (mempunyai beberapa nilai).

2. *Slot value type.*

   Sebuah tipe *facet* menggambarkan tipe dari nilai yang dapat mengisi *slot*. Berikut merupakan tipe nilai yang umum digunakan:

   1. *String*, tipe nilai yang paling sederhana yang biasa digunakan untuk *slot*.
   2. *Number*, terkadang tipe nilai yang lebih spesifik seperti *Float* dan *Integer* yang digunakan dalam penggambaran *slot* dengan nilai numerik.
   3. Boolean slot mempunyai *yes-no flag*.
   4. *Enumerated slot* merinci daftar dari spesifik nilai yang diperbolehkan untuk *slot*.
   5. *Instance* membuat hubungan antara individual.

7. Membuat *instances*.

   Langkah terakhir adalah membuat *individual* atau *instances* pada *class* dalam suatu hirarki. Dalam mendefinisikan *individual* atau *instances* pada *class* harus diperhatikan persyaratan berikut:

   1. Pemilihan *class*.
   2. Pembuatan *individual* atau *instances* pada *class* tersebut. *Knowledge base* selanjutnya dapat dibuat dengan mendefinisikan *individual* atau *instances* dari *class* yang terisi pada nilai spesifik *slot* dan *slot* batasan tambahan (*facets*).
   3. Mengisi nilai *slot*.

2.4 Tool Ontologi

2.4.1 Protage

Protege merupakan tool ontologi dengan *platform open source* untuk membangun domain model dan aplikasi *knowledge based*. Protege mengimplementasikan struktur pemodelan *knowledge* dan dapat memvisualisasikan hasil ontologi dalam berbagai format. Sebuah ontologi menggambarkan *concept* dan hubungan-hubungan yang penting dalam domain yang khusus, yang menyediakan kosa kata dalam domain tersebut. Dalam beberapa tahun terakhir, ontologi telah diadopsi dalam bisnis dan komunitas ilmiah seperti *scientific knowledge portal*, manajemen informasi, integrasi informasi, *electronic commerce* dan *semantic web service*. Pada Protege terdapat dua cara dalam pemodelan ontologi, yaitu:

1. Protege Frame editor User dapat membangun ontologi dalam *frame-based* dengan kesepakatan dengan *Open Knowledge Base Connectivity Protocol* (OKBC). Pada model ini, sebuah ontologi terdiri dari seperangkat *class* yang terorganisir pada suatu hirarki yang merepresentasikan sebuah domain, seperangkat slot yang berhubungan dengan *class* serta *instance* dari tiap *class* tersebut.
2. Protege OWL Protege-OWL editor merupakan kelanjutan dari Protege yang mendukung *Ontology Web Language* (OWL). OWL merupakan pengembangan mutakhir standar bahasa ontologi yang disahkan oleh *World Wide Web Consortium* (W3C) untuk mempopulerkan *semantic web vision*. Protege-OWL editor memungkinkan user untuk :
    1. Mengambil dan menyimpan OWL dan RDF ontologi.
    2. Mengubah dan memvisualisaikan *class*, *properties* dan *Semantic Web Rule Language* (SWRL).
    3. Menjabarkan karakteristik *class* secara logis sebagai ekspresi OWL.

4. Mengeksekusi penalaran seperti *description logic classifier*.
5. Mengubah OWL individual untuk web semantik.

Protege-OWL berhubungan secara erat dengan Jena dan mempunyai *open source Java API* untuk pengembangan *Semantic Web Service*. Jena merupakan *framework* berbasis Java untuk mengkonstruksi aplikasi semantik web. *Framework* ini menyediakan lingkungan pemrograman RDF, RDF Schema OWL dan SPARQL.

Protege-OWL *Application Programming Interface* (API) adalah sebuah *Java Library open source* untuk *Ontology Web Language* (OWL) dan RDF. API menyediakan metode untuk mengambil dan menyimpan *file* OWL, menanyakan OWL data model, menjalankan penalaran berdasarkan *Description Logic engines* dan sebagai *graphical user interface*. API didesain untuk dapat digunakan dalam dua konteks, yaitu:

1. Pengembangan komponen yang dieksekusi dalam Protege-OWL *editor user interface*.
2. Pengembangan aplikasi *stand-alone* (contoh : aplikasi Swing, Servlet dan plug-in Eclipse).

2.4.2 Jena Fuseki

Jena adalah sebuah *framework* Java untuk membangun aplikasi web semantik. Jena menyediakan perpustakaan Java yang luas untuk membantu *developer* mengembangkan kode yang menangani RDF, RDFS, RDFa, OWL dan SPARQL sesuai dengan rekomendasi dari W3C yang diterbitkan. Jena juga menyertakan *rule-based inference engine* untuk melakukan penalaran berdasarkan ontologi OWL dan RDFS, serta berbagai strategi penyimpanan dalam penyimpanan tiga kali lipat RDF pada memori atau *disk*.

Jena awalnya dikembangkan oleh para peneliti di HP Labs, dimulai di Bristol, Inggris, pada tahun 2000. Jena selalu menjadi proyek *open-source*, dan telah banyak digunakan dalam berbagai aplikasi web semantik dan aplikasi demo. Pada tahun 2009, HP memutuskan untuk memfokuskan kembali aktivitas pengembangan dari dukungan langsung pengembang Jena, meskipun tetap mendukung tujuan proyek. Akhirnya, tim proyek berhasil mengajikan permohonan agar Jena diakusisi oleh Apache Software Foundation pada November 2010 [8].