Bab 2 Landasan Teori

Bab ini akan menjelaskan landasan teori dan teknologi yang digunakan oleh penulis dalam tugas akhir. Hal yang dibahas diantaranya adalah E-Learning, Semantic Web (termasuk didalamnya RDF, OWL dan ontologi).

2.1 E-Learning

E-Learning merupakan tipe pembelajaran yang menggunakan teknologi komputer sebagai media pembelajaran, biasanya terhubung dengan jaringan sehingga proses pembelajaran dapat dilakukan kapan saja dan dimana saja[3]. Adanya perkembangan teknologi internet yang pesat menjadi katalisator dari perkembangan E-Learning, dimana batas ruang dan waktu tidak lagi menjadi permasalahan, namun walaupun E-Learning lebih akrab dikenal dalam bentuk online, tidak jarang terdapat E-Learning dalam bentuk offline misalnya dengan menggunakan keping CD/DVD.

2.1.1 Tujuan E-Learning

Pembelajaran menggunakan E-Learning biasanya dirancang untuk membimbing pembelajar dalam mempelajari suatu pelajaran atau membantu pembelajar untuk melakukan suatu pekerjaan yang spesifik. Pembelajaran sendiri diperoleh dari content E-Learning itu sendiri, yang mengandung informasi bagi pembelajar.

Pembelajaran Berbasis Komputer / Computer Based Learning (CBL) merupakan terminologi yang digunakan untuk pembelajaran yang menggunakan komputer sebagai media utama dalam pembelajaran, namun dilihat secara garis besar CBL merupakan sebuah *environment* pembelajaran dimana komputer dipakai sebagai media pengajar. Konsep ini dibedakan dengan pembelajaran menggunakan komputer yang bersifat pengalaman, seperti games atau browsing.

2.1.2 Tipe E-Learning

Pembelajaran menggunakan E-Learning dapat dipisahkan berdasakan penggunaannya, setiap pembelajaran pasti mempunyai dari masing-masing komponen. Komponen-komponen tersebut adalah:

- Purely Online/ Blended Learning.

  Purely Online adalah pembelajaran tanpa adanya pertemuan tatap muka, jadi pembelajaran dilakukan langsung via media elektronik. Blended Learning adalah pembelajaran melalui media elektronik namun diselingi dengan pertemuan tatap muka.
- Synchronous / Asynchronous. Sychronous adalah komponen pembelajaran yang dilakukan pada waktu yang bersamaan (*real time*), antara pengajar dan pembelajar, contohnya seperti chatting. Asynchronous adalah komponen pembelajaran yang dapat dilakukannya secara tidak bersamaan, misalnya forum.
- Group disertai pembimbing / Self Study.

  Group disertai pembimbing adalah komponen dimana ada pembimbing yang ahli (*domain expert*) untuk membimbing anggota group. *Self Study* adalah komponen pembelajaran dimana tidak disertai pembimbing.
- Web Based / Computer Based (menggunakan CD) / Video Tape. Web Based / Computer Based / Video Tape adalah media yang digunakan.

Jadi misalnya, pembelajaran di tempat X adalah Blended Learning, Asynchronous, Group disertai pembimbing dan Web Based.

2.1.3 Keuntungan E-Learning

E-Learning mempunyai keuntungan yang jelas dibandingkan pengajaran tradisional di dalam kelas. Hal yang paling jelas adalah fleksibilitas dan penghematan biaya transportasi atau pengeluaran dari produktifitas kerja yang hilang. Ada juga keuntungan yang tidak terlalu terlihat dengan jelas seperti:

- Tidak terlalu mahal untuk diproduksi, karena dengan menggunakan *authoring tools*, kita dapat membuat program yang bersifat asynchronous, ketika penggunaannya sudah mencapai break even point maka secara kasar bisa dibilang program itu gratis, sedangkan program yang bersifat synchronous mempunyai pengeluaran tetap seperti

untuk membayar pembimbing, walaupun lebih murah dari pengajaran tatap muka tradisional.

- Kecepatan pembelajaran bisa dikontrol sendiri. Kebanyakan E-Learning program dapat digunakan kapan saja dibutuhkan.
- Lebih cepat, berdasarkan sebuah artikel karangan Jennifer Salopek yang berjudul "Training and Development Magazine", pengajaran menggunakan E-Learning menghasilkan waktu yang 50% lebih cepat dari pengajaran tatap muka. Salah satu penyebabnya adalah adanya pendekatan secara individu yang memungkinkan seseorang untuk melewati materi yang sudah dipelajarinya dan dikuasai, lalu melanjutkan ke materi yang belum dikuasai.
- Mempunyai pesan yang konsisten, E-Learning meniadakan masalah dimana berbeda pembimbing akan mengajarkan hal yang berbeda, walaupun pelajarannya sama.
- Dapat dilakukan dimanapun dan kapanpun, pembelajaran dapat dilakukan dimana saja, kapan saja.
- Dapat diupdate secara cepat dan mudah, sesi pembelajaran online dapat dengan mudah diperbaharui. Karena materi sudah dimasukan ke *server*.
- Dapat meningkatkan retensi dan daya tangkap yang lebih baik, hal ini disebabkan oleh banyaknya kombinasi penggunaan E-Learning untuk mendukung dengan kuis, video ataupun interaksi.
- Dapat dengan mudah diatur apabila terdiri dari grup besar, untuk melakukan perekrutan, *Human Resource* manager tinggal mencari rekam jejak dari tawaran.

2.1.4 Personalisasi pada E-Learning

Personalisasi pada E-Learning adalah fitur yang bisa memberikan nilai tambah dalam penggunaan E-Learning. Dengan menerapkan prinsip-prinsip pembelajaran yang telah diketahui, pemberian materi dapat dirancang sedemikian rupa sehingga memudahkan pembelajar untuk menyerap pengetahuan yang diberikan.

Proses personalisasi ini tidak lepas dari prinsip-prinsip psikologi pembelajaran, interaksi antara rangsangan berupa gambar, suara ataupun video dengan jaringan saraf yang ada di dalam otak akan memberikan hasil yang berbeda, jika ditempatkan dalam

keadaan yang berbeda. Tugas dari personalisasi inilih yang digunakan untuk mencari jenis interaksi yang paling efektif sehingga menghasilkan hasil yang paling maksimal

2.2 Semantic Web

Jaringan Internet pada masa sekarang kebanyakan berbentuk dokumen yang ditulis dalam bentuk HTML (Hypertext Markup Language), bentuk terstandarisasi yang digunakan untuk menuliskan code yang dapat mewakili objek multimedia. Dengan menggunakan HTML dan tool untuk menterjemahkan code (misalnya Web Browser), tool dapat menampilkan suatu halaman Website.

Misalnya kita mengambil contoh sebuah situs penjual buku, code HTML dari situs tersebut akan memberi tahu kita bahwa "judul dari dokumen ini adalah 'Toko Buku A'", tetapi HTML tidak mempunyai kemampuan untuk menilai bahwa barang dengan kode 123456 adalah 'buku Biologi untuk kelas 5 SD yang diterbitkan oleh penerbit Gajah Tidur yang berharga Rp 30.000,-'. HTML hanya mengetahui bahwa kode 123456 adalah serangkaian teks "123456" yang secara posisi berada dekat dengan teks "Buku Biologi", "5 SD", "Gajah Tidur", "Rp 30.000,-", jadi tidak mungkin diketahui bahwa "Buku Biologi" sebagai nama buku, "Gajah Tidur" sebagai penerbit buku ataupun "RP 30.000,-" sebagai harga buku, karena HTML tidak dapat mengungkapkan bahwa informasi-informasi tadi berkaitan. Keterbatasan HTML seperti inilah yang mendorong pengembangan Web yang lebih "pintar".

Semantic Web adalah hasil pengembangan dari World Wide Web yang memungkinkan orang untuk bertukar *content* di luar batas website dan aplikasi [1], dibuat untuk menjadikan suatu Web menjadi lebih "pintar". Dalam hal ini "pintar" karena Semantic Web mampu membaca data secara semantic, sebuah terobosan baru dalam Web yang selama ini mengandalkan nilai sintaksis untuk melakukan *query*. Akibatnya adalah kesulitan karena batasan terminologi dan struktur bahasa, diminimalisir karena data dibaca secara semantic. Perlu diingat bahwa walaupun teknologi ini sudah merupakan improvisasi dari *Web* 2.0, teknologi Semantic Web ini belum sepenuhnya *mature*. Sehingga untuk beberapa hal terdapat kekurangan dibandingkan dengan teknologi yang sudah *mature*, salah satunya adalah *database*, namun pengembangan kedepannya[2].

[2] Pembahasan lebih lanjut pada bagian 2.2.2 RDF – Perbandingan dengan Database

Semantic Web adalah teknologi yang menjanjikan dan walaupun belum namun sudah mendekati apa yang menjadi cita-cita dari pendiri WWW untuk Web.

Semantic Web meliputi penanganan bahasa yang secara menangani data, seperti *Resource Description Framework* (RDF), *Ontology Web Language* (OWL) dan *Extensible Markup Language* (XML). HTML tetap digunakan untuk menjelaskan dokumen dan hubungan diantara dokumen tersebut, sedangkan RDF, OWL dan XML mengatur data-data yang ditampilkan.

Semantic Web seringkali digambarkan sebagai jaringan-jaringan web yang saling terhubung, dimana masing-masing isi dari web (data dan service) didefinisikan sehingga memungkinkan setiap Web mengerti dan menggunakan *content* yang ada di Web tersebut. Untuk dapat seperti itu, maka Semantic Web menggunakan ontologi (dibahas dalam bagian 2.2 dan 2.3) sebagai teknologi backbone-nya. Pada intinya, Semantic Web adalah jaringan yang saling terhubung dimana masing masing node memahami data yang *machine-readable* yang diwujudkan dalam bahasa yang telah distandarisasi dan dibakukan.

Elemen yang digunakan oleh Semantic Web diwujudkan dalam bentuk yang sudah dibakukan, antara lain Resource Description Framework (RDF) yang merupakan *framework* baku yang digunakan, format dari pertukaran data (N3, TURTLE, RDF/XML, N-Triples), notasi dari RDF (RDFS) dan bahasa Web Ontology Language (OWL). Elemen-elemen tersebut digunakan dengan tujuan menyediakan penjelasan dalam bentuk baku tentang konsep, makna dan hubungan dari domain yang ada[2]. Selain itu, Semantic Web juga menyiapkan elemen untuk teknologi yang diprediksi mungkin akan digunakan pada masa depan, walaupun sampai sekarang belum diimplementasikan atau bahkan belum dipikirkan.

2.2.1 Visi

> I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize.
>
> – Tim Berners-Lee, 1999

Berdasarkan World Web Consortium (W3C), web akan mencapai pemanfaatan secara penuh hanya jika suatu web dapat berbagi, memproses dan mengerti secara otomatisasi ataupun secara manual oleh manusia. Untuk sampai kepada tahap itu, program masa mendatang harus dapat berbagi, memproses dan mengerti data walaupun program itu diciptakan berbeda dari program yang dibacanya.

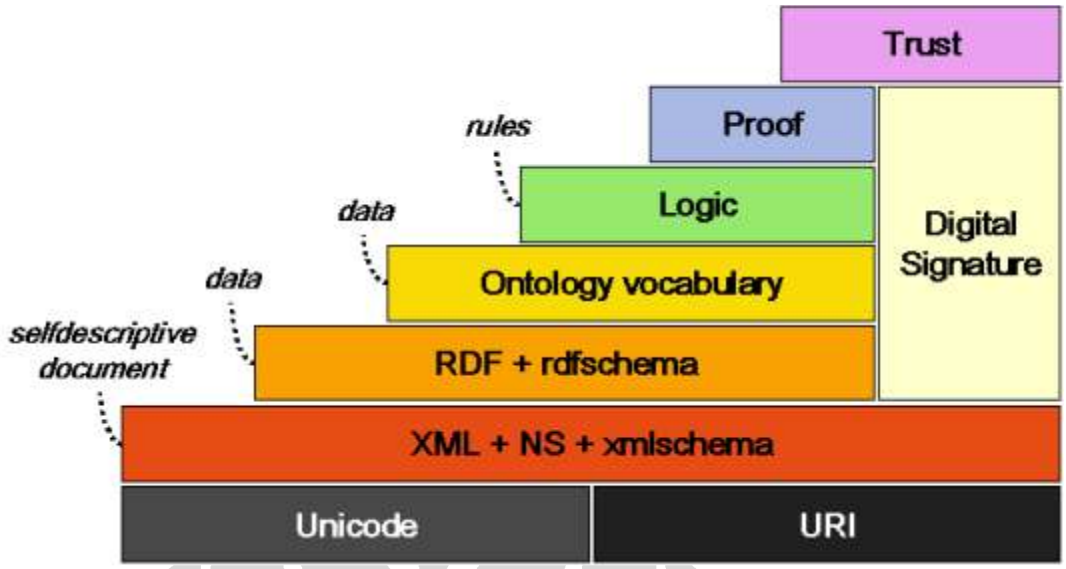
Untuk mencai kemampuan seperti itu, rancangannya pembuatannya sudah dikemukakan, dilihat secara konsep Semantic Web terbagi menjadi beberapa *layer* teknologi(gambar 2.1), Teknologi ini terdiri atas:

- Unicode dan URI : Unicode adalah standard representasi karakter komputer. URI (Uniform Resources Identifier) merupakan standard untuk lokasi dan identitas suatu resource (misalnya web page). Universitas Indonesia
- XML + NS + XML(S) : XML(Extensible Markup Language) dan Namespace, merupakan aturan sintaks yang berfungsi untuk menyajikan struktur data pada web, namun tidak menekankan batasan pada makna dari dokumen tersebut. XML Schema merupakan bahasa untuk membatasi struktur XML dan memperkaya XML dengan datatype.
- RDF + RDF(S) : RDF(Resource Description Framework) merupakan model dalam format triple yang dapat direpresentasikan dalam bentuk graph untuk menjelaskan resource dan relasinya. Sedangkan RDF Schema(RDFS) adalah kata kerja yang menjelaskan property dan kelas dari resource yang terdapat dalam RDF .

- Ontology vocabulary : Bahasa ontologi yang direkomendasikan oleh W3C pada 10 Februari 2004 adalah Web Ontology Language(OWL), merupakan bahasa yang lebih kaya(perluasan dari RDF) dan kompleks untuk mendeskripsikan resource. OWL menggunakan format triple, sama seperti RDF
- Logic dan Proof : Layer ini berupa rule dan sistem untuk melakukan reasoning pada ontologi sehingga dapat disimpulkan apakah suatu resource memenuhi syarat tertentu.
- Trust : Layer terakhir dari semantic web yang memungkinkan pengguna web untuk mempercayai suatu informasi pada web.

Terminologi Semantic Web merupakan hal baru bagi kebanyakan orang, bahkan termasuk orang IT sekalipun, namun problem yang muncul dari itu sebenarnya sudah kita hadapi dari beberapa dekade yang lalu antara lain: *information overload* dan proses penggabungan konten yang buruk. Permasalahan utama dari semuanya itu adalah kurangnya definisi semantic dalam masing-masing sistem, kurangnya integrasi antar semantic diantara data dan kurangnya inter-operabilitas semantic yang ada pada berbagai sistem. Cakupan Semantic Web yang melebihi kemampuan dari Web yang ada sekarang dan juga teknologi informasi yang ada memungkinkan untuk terjadinya kolaborasi yang efektif dan kemampuan pengambilan keputusan yang lebih pintar. Semantic Web merupakan aggregasi dari *intelligent website* dan *data stores* yang dapat diakses oleh teknologi semantic, framework yang konseptual dan aturan-aturan interaksi yang memungkinkan mesin bekerja tidak sekedar melakukan respon atas permintaan.

Untuk mencapai sampai ke tahap seperti ini, walaupun sudah dibuat rancangan dasar dan *layer-layer* yang harus dikerjakan, sangat tidak memungkinkan untuk dapat dicapai dalam jangka waktu beberapa tahun yang akan datang, yang pasti perkembangan dalam dunia teknologi informasi yang cepat akan mencapai suatu hasil, namun progress yang akan dicapai belum dapat diramalkan. Bisnis model yang mulai dimengerti dan kesadaran akan teknologi mendatang sudah meningkat, namun tetap saja seperti teknologi informasi yang lain, akan ada jeda waktu yang cukup lama sampai serpihan-serpihan visi ini tersusun [14].



Gambar 2. 1 – Gambar Layer-layer teknologi dalam Semantic Web

Dibawah akan dijelaskan tentang RDF dan OWL sebagai salah satu layer dari Semantic Web dengan lebih jelas lagi.

2.2.2 RDF

Ketika kita membicarakan definisi dari *semantic* dalam Semantic Web, kita akan menemukan banyak sekali definisi, namun definisi yang cukup sederhana dari *semantic* adalah representasi dari keterhubungan antara terminologi yang dipakai dengan entitas di dunia nyata yang dimaksud dengan terminology tersebut. Hal inilah yang menjadi salah satu pencetus Semantic Web, dimana semantic dipakai sebagai referensi terhadap entitas di dunia nyata.

Web yang ada sekarang, terdiri atas dokumen-dokumen yang saling terhubung satu sama lain, dan setiap koneksi (referensi) antara dokumen dan benda yang dijelaskan di dunia nyata dibuat di dalam orang yang membaca dokumen tersebut. Jadi dapat saja ada hubungan antara dokumen tentang Yohanes dan dokumen tentang komputer, tapi tidak ada notasi yang menghubungkan dokumen tentang Yohanes dengan Yohanes Immanuel yang ada di dunia nyata.

Dalam Semantic Web, kita menyebut benda yang ada sebagai *resources*, *resource* dalam hal ini merupakan sesuatu yang ingin kita deskripsikan, bentuknya dapat apa saja

baik yang abstrak maupun yang real, sebagai contoh dapat orang, rumah, pohon, email, URL, semua sapi yang ada di pulau jawa dan lain-lain. Mungkin menggunakan kata *resource* terdengar aneh, namun kata-kata lain yang lebih akurat seperti *thing* atau entitas mempunyai permasalahan sendiri, sehingga dalam hal ini kata *resource* dipakai untuk merepresentasikan hal itu.

Distribusi data dalam jaringan

Data paling sering diwujudkan dalam bentuk tabel dimana setiap baris menunjukan benda yang kita ingin terangkan, kolomnya menunjukan properti yang ada dalam benda tersebut dan isi dari kotak tersebut terdapat nilai. Salah satu prinsip yang menyimpan data dalam bentuk tabular adalah aplikasi pada relational database. Misalkan terdapat daftar tabel seni dari jaman dahulu[3] yang ingin direpresentasikan dalam bentuk tabel. Dimana ID **1** menujukan karya seni bernama **As You Like It** yang merupakan **Drama** karangan **Shakespeare** pada tahun **1599**.

Table 1 - Data Seni Klasik dalam bentuk tabel

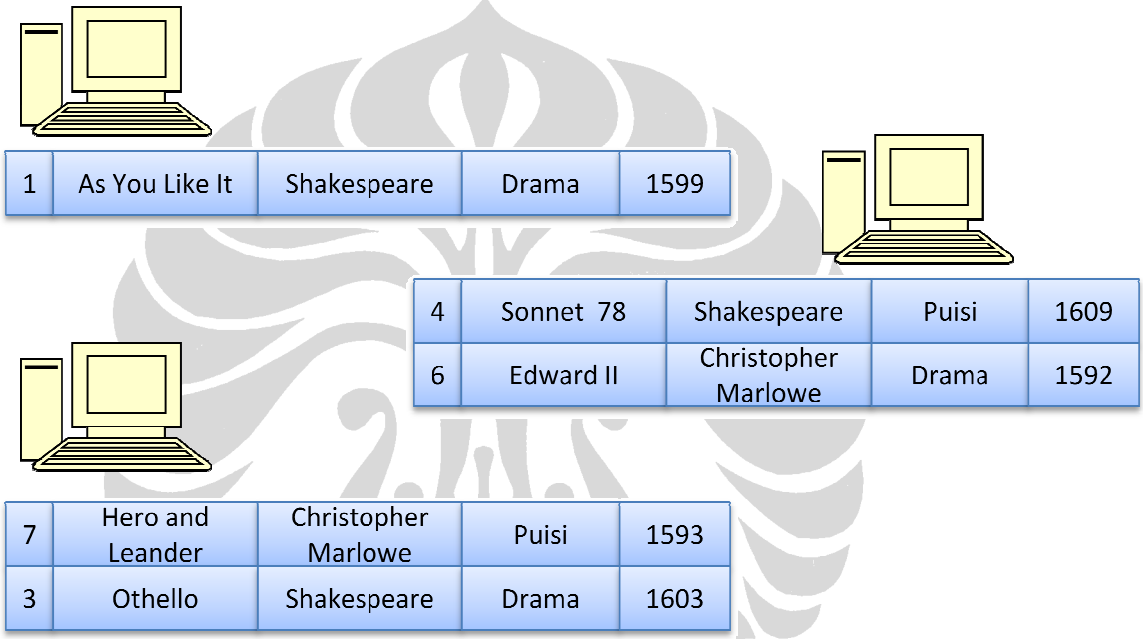
ID	Judul	Pengarang	Media	Tahun
1	As you Like It	Shakespeare	Drama	1599
2	Hamlet	Shakespeare	Drama	1604
3	Ohello	Shakespeare	Drama	1603
4	“Sonnet 78”	Shakespeare	Puisi	1609
5	Astrophil and Stella	Sir Phillip Sidney	Puisi	1590
6	Edward II	Christopher Marlowe	Drama	1592
7	Hero and Leander	Christopher Marlowe	Puisi	1593
8	Greensleeves	Henry VII Rex	Lagu	1525

Namun dalam kasus data yang terdistribusi, data harus diletakan terpencar di beberapa server komputer. Pembagian penyebaran data ini dapat dilakukan dengan beberapa cara, antara lain :

[3] Contoh data diambil dari [18].

1 Pembagian berdasarkan baris.

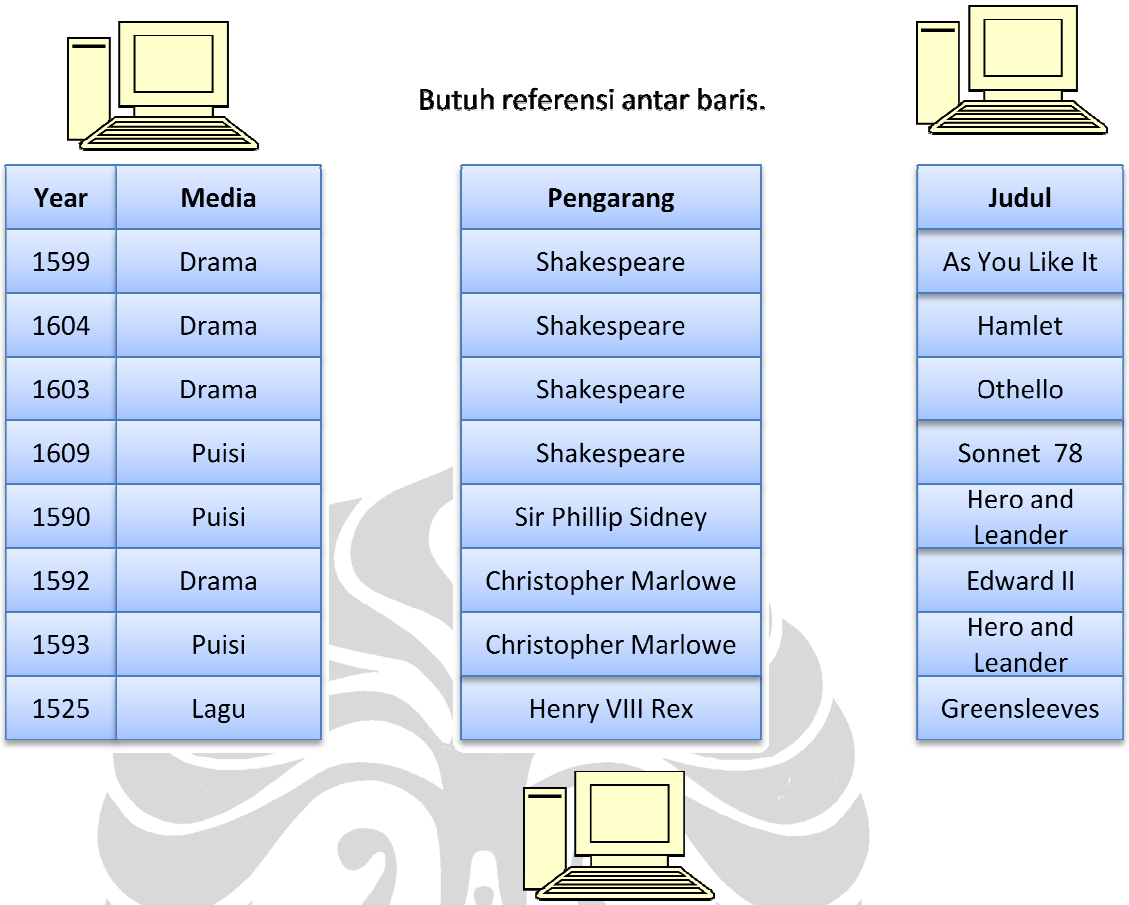
Pembagian ini dilakukan agar setiap komputer memiliki masing-masing data, sehingga nantinya untuk memperoleh data yang utuh tinggal digabungkan saja dengan koordinasi antara masing-masing server. Cara ini memiliki masalah, karena kita tidak mungkin mempunyai skema/kolom yang sama dalam setiap server untuk menampung data, coba bayangkan setiap komputer mempunyai kolom untuk menampung seluruh data di dunia, tentunya akan *redundant* dan tidak efisien. Gambar pembagiannya dapat dilihat dibawah.



Gambar 2. 2 - Gambar Pembagian Data Tabular per baris

2 Pembagian berdasarkan kolom.

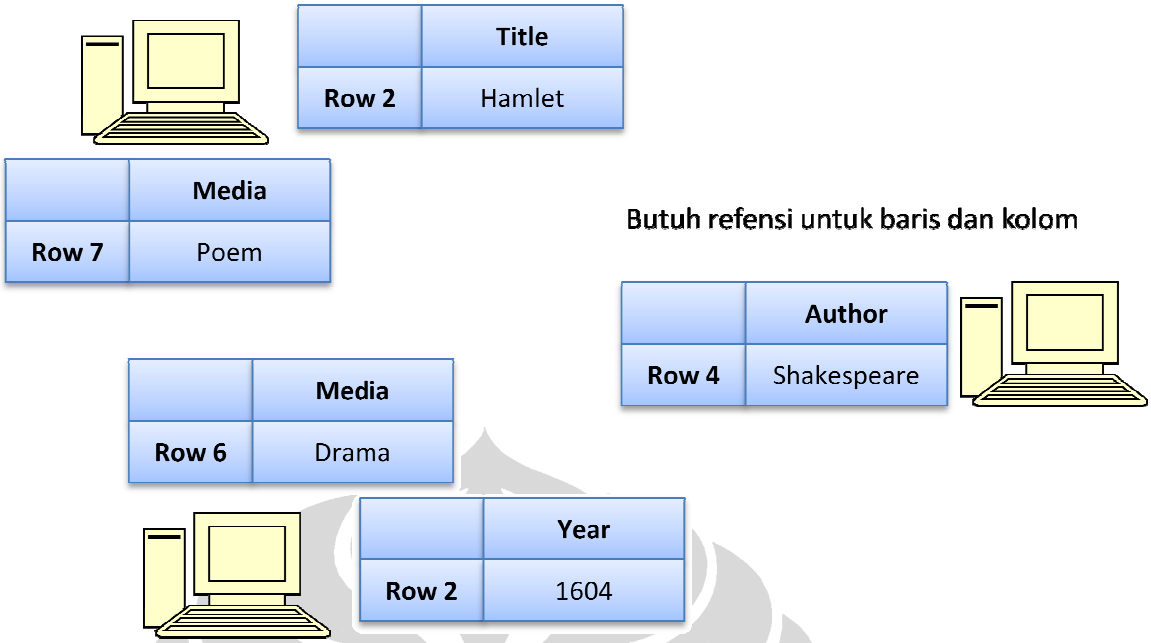
Pembagian ini dilakukan agar setiap komputer memiliki masing-masing kolom, sehingga tidak perlu lagi dipusingkan dengan format kolom dan dapat mengambil kolom yang kita mau saja. Cara ini tidak akan mengalami masalah seperti nomor 1, sehingga nantinya untuk memperoleh data yang utuh tinggal digabungkan saja dengan koordinasi antara masing-masing server. Cara ini memiliki masalah, bagaimana kita tahu bahwa kolom no 3 di server ke-1 berhubungan dengan dengan kolom no 3 di server ke-2, karena itu cara ini susah untuk diterapkan. Gambar pembagiannya dapat dilihat dibawah.



Gambar 2. 3 - Gambar Pembagian Data Tabular per kolom

3 Pembagian secara sel

Pemecahan jenis yang ketiga adalah pemecahan secara sel, sehingga yang masing-masing server miliki adalah data tingkat terkecil atau datanya langsung. Berbeda dengan kedua jenis pemecahan diatas. Pemecahan tingkat ini dapat dipecah dengan mudah, asalkan terdapat referensi yang benar, yang merupakan problema pada jenis ke 1 dan 2 namun cara inilah yang dipakai dalam distribusi data Semantic Web. Gambar pembagiannya dapat dilihat dibawah.



Gambar 2. 4 - Gambar Pembagian data Tabular per sel

Dengan terdapatnya data pada masing-masing server, maka untuk mengakses suatu data kita membutuhkan 3 buah referensi untuk merujuk ke suatu nilai: referensi global ke kolom, referensi global ke baris, dan nilai dari sel itu sendiri. Ketiga nilai inilah yang menjadi dasar RDF yang disebut *triple*. RDF (*Resource Description Framework*) sesuai dengan namanya, digunakan untuk mendeskripsikan *resource*, hal ini berarti memecah knowledge kedalam bentuk yang lebih kecil (triple), yang menunjukan makna dari *triple* tersebut.

Triple

Penggunaan RDF adalah standar yang ditetapkan oleh W3C untuk pertukaran knowledge, sebuah fakta atau statement yang ingin kita representasikan ditulis dalan bentuk *triple*, dimana terdapat Subjek-Predikat-Objek sama seperti bahasa manusia. Triple ini dapat dipandang sebagai 3 referensi yang telah dijelaskan diatas. Contoh ada statement: "**Shakerpeare write Hamlet**", dimana **Shakespeare** adalah subjek, **write** adalah predikat dan **hamlet** adalah objek. Jadi disini terdapat 3 kata, ketiga kata inilah yang kita anggap sebagai entitas yang disebut juga *resource*. Jadi **Shakespeare** adalah *resource*, **write** adalah *resource* dan **Hamlet** adalah *resource*. Penamaan ini bersifat global, jadi jika misalnya ada statement tambahan "**Shakespeare married Anna_Hathaway**", maka kita akan tahu bahwa Shakespeare dalam "**Shakerpeare write**

Hamlet" adalah Shakespeare dalam "**Shakespeare married Anna_Hathaway**", karena **Shakespeare** merujuk ke referensi yang sama. Karena itu untuk membedakannya digunakan URI(Uniform Resource Identifier), sebuah sistem penamaan yang globally unique, yang memakai format seperti URL dalam *website*. Sekarang kita akan coba melihat kedua statement diatas dalam bentuk URL, berikut adalah contoh:

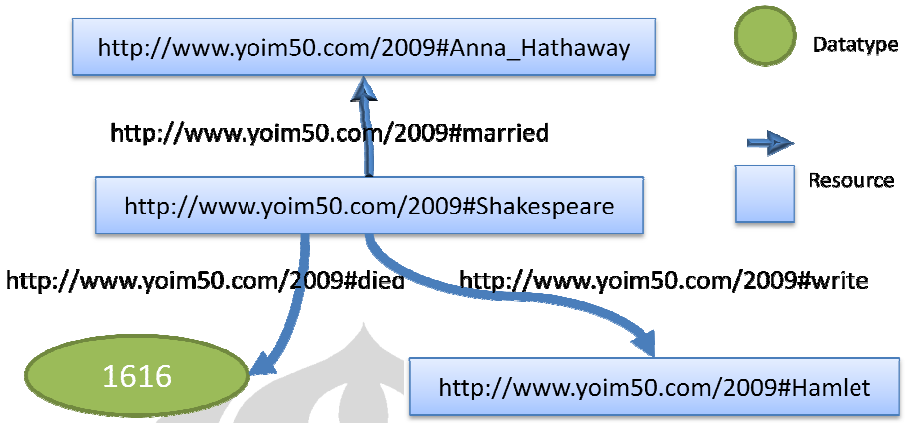
- Shakespeare => http://www.yoim50.com/2009#Shakerpeare
- write => http://www.yoim50.com/2009#write
- Hamlet => http://www.yoim50.com/2009#Hamlet
- Shakespeare => http://www.yoim50.com/2009#Shakerpeare
- married => http://www.yoim50.com/2009#married
- Anna_Hathaway => http://www.yoim50.com/2009#Anna_Hathaway

Jadi untuk merepresentasikan "**Shakerpeare write Hamlet**" dalam *triple* bentuk URI, kita dapat menuliskannya dengan bentuk <http://www.yoim50.com/2009#Shakerpeare http://www.yoim50.com/2009#write http://www.yoim50.com/2009#Hamlet>. Bentuk representasi diatas memang terlihat aneh namun bentuk diatas bukanlah satu-satunya format standar yang baku.

Triple seperti diatas dapat saja direpresentasikan dalam bentuk 3 buah URI seperti diatas, namun juga dapat direpresentasikan dalam bentuk RDF XML yang merupakan standar resmi format untuk pertukaran RDF dalam semantic Web, jadi sebagai representasi RDF dalam syntax XML, ada juga bentuk Notation3 (N3) yang dipakai secara umum (untuk percobaan kita akan menggunakan N3 sebagai notasi untuk data). Jika sebuah informasi sudah direpresentasikan dalam bentuk RDF maka sebuah informasi tersebut akan mudah untuk diproses karena RDF merupakan format umum yang mempunyai banyak syarat kalimat, yang memudahkan mesin memprosesnya.

Objek dari sebuah *triple* belum tentu *resource*, jika objek dari sebuah *triple* adalah sebuah *resource* maka predikat/properties-nya disebut Object Properties, sedangkan apabila objek dari sebuah *triple* adalah standar data, maka predikat/properties-nya disebut Datatype Properties. Misalkan kita punya "**Shakerpeare write Hamlet**", karena **Hamlet** (objek) adalah resource, maka **write** adalah Object Properties, sedangkan jika ada statement "**Shakerspeare died** 1616" dimana 1616 (objek) bukan merupakan resource, maka died adalah Datatype Properties

Triple dapat juga digambarkan sebagai graph, misalkan dari ketiga data tersebut, kita ingin representasikan sebagai gambar maka kita dapat representasikan sebagai.

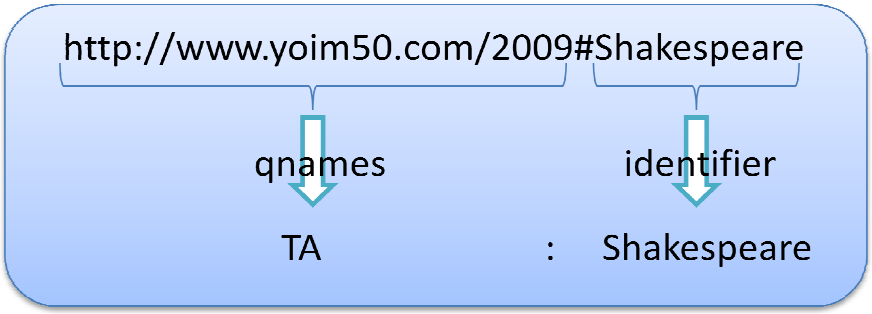


Gambar 2. 5 - Gambar Data dalam bentuk Triple

Namespace

Untuk membedakan semua *resource* yang ada, kita menggunakan URI (Uniform Resource Identifier), karena dengan URI kita dapat mengidentifikasi semua *resource*. Penggunaan ini diharapkan agar mempermudah pengenalan suatu item, misalnya contoh diatas, dengan memakai <http://www.yoim50.com/2009#Shakerpeare> kita dapat tahu bahwa *resource* ini punya 3 *properties*, namun dapat saja 3 orang menuliskan URI yang lain walaupun *resource* yang dimaksud adalah *resource* <http://www.yoim50.com/2009#Shakerpeare> hal ini tidak salah namun tidak diajurkan karena lebih baik diseragamkan.

Penjelasan dengan URI dapat menjelaskan resource dengan baik, namun akan merepotkan jika kita menggunakan setiap detail untuk mengekspresikan sebuah model, karena itu kita menggunakan versi simpel dari singkatan URI yang disebut *qnames*. Dalam bentuk terkecilnya qnames terdiri atas 2 bagian: namespace dan identifier[18].



Gambar 2. 6 - Gambar Representasi URI dalam bentuk Namespace

Jadi alamat URI seperti <http://www.yoim50.com/2009#Shakerpeare> dapat kita singkat menjadi TA:Shakerpeare. Namespace berlaku untuk semua alamat URI yang sama, jadi misalnya terdapat alamat URI http://www.yoim50.com/2009#married dapat disingkat menjadi TA:married. Perlu diingat bahwa qnames bukanlah identitas global, yang merupakan identitas global hanyalah URI yang memenuhi syarat. Jadi yang perlu diingat adalah penggunaan qnames harus diikuti dengan deklarasi namespace yang bersangkutan.

@prefix TA: < http:// www.yoim50.com/2009#>

Identifier pada namespace RDF sudah beberapa yang menjadi *pre-defined namespace*. Berikut adalah beberapa dari namespace tersebut:

- rdf:type. Properti ini dipakai untuk membuat instansiasi dari suatu kelas, sehingga kita tahu bahwa subjek dari rdf:type merupakan individu dari kelas objek.
- rdf:property. Properti ini dipakai untuk membuat instansiasi dari suatu properties, sehingga kita tahu bahwa **subjek** dari rdf:Properties merupakan individu dari kelas **objek.**
- rdf:subject. Properti ini sendiri merupakan instance dari rdf:property. Jadi bisa dituliskan dalam bentuk triplet seperti ini (rdf:subject rdf:type rdf:Property). Properti ini dipakai untuk menunjukan **subjek** dari suatu *statement*.
- rdf:predicate. Properti ini sendiri merupakan instance dari rdf:property. Jadi bisa dituliskan dalam bentuk triplet seperti ini (rdf: predicate rdf:type rdf:Property). Properti ini dipakai untuk menunjukan predikat dari suatu *statement*.
- rdf:object. Properti ini sendiri merupakan instance dari rdf:property. Jadi bisa dituliskan dalam bentuk triplet seperti ini (rdf:object rdf:type rdf:Property). Properti ini dipakai untuk menunjukan **objek** dari suatu *statement*.

Perbandingan dengan Database

Seperti yang telah dijelaskan pada bab sebelumnya, bahwa Semantic Web mempunyai kelebihan dari Web biasa, dimana Semantic Web mengambil keuntungan dari kemampuan untuk menelaah nilai semantic dari data sedangkan Web biasa hanya melihat data dari nilai sintaksis saja. Nilai semantic ini diwujudkan dalam bentuk *triple*, bentuk yang telah distandarisasi untuk merepresentasikan *knowledge*. *Triple* disimpan

dalam tempat yang bernama RDF *Store* dalam format N3, salah satu format untuk RDF. Untuk penyimpanan nilai (*string*/*integer*/*float*), akan diletakan direpresentasikan sebagai objek dalam *triple*.

Sifat RDF Store sama seperti database, yaitu menyimpan data dan memakainya suatu saat nanti. Namun, RDF *Store* mampunyai keunggulan karena mempunyai kemampuan untuk menggabungkan dua kumpulan data, hal ini disebabkan oleh sifat dari data RDF yang fleksibel. Setiap kumpulan data terdiri atas sekumpulan *triple*, sehingga penggabungan dua atau lebih kumpulan data tetap akan menghasilkan kumpulan *triple* juga. Dalam hal ini, *relational database* mempunyai kekurangan karena ragam format yang digunakan sebagai representasi dari kolom akan mempersulit penggabungan kedua kumpulan data.

Implementasi dari RDF *Store* bervariasi, mulai dari program database yang dikostumisasi sampai produk siap pakai dari vendor. Cara mudah merepresentasikan *triple* dalam *relational database* adalah dengan menggunakan tabel dengan tiga kolom, satu untuk *subject*, satu untuk *predicate* dan satu untuk *object* [18]. Untuk memperlengkapi RDF *Store*, biasanya RDF *Store* juga diperlengkapi dengan *parser*, *serializer*, dan *query engine*. *Query engine* digunakan agar dapat melakukan *query* sama seperti *database* biasa.

Walaupun banyak kesamaan, namun perlu diingat bahwa teknologi RDF ini merupakan teknologi baru yang belum *mature*. Untuk masalah penyimpanan data, teknologi RDF belum mempunyai kemampuan seperti *database*. Hal yang harusnya bisa dimaklumi, karena teknologi Semantic Web masih belum mature dan masih dalam pengembangan. Dalam penelitian ini, menggunakan Semantic Web karena kemampuan melihat nilai semantic dari data sehingga memungkinkan *machine-readable*.

RDFS

RDF Schema (RDFS) adalah *schema* untuk bahasa RDF yang berfungsi untuk membantu menyediakan kemampuan pengertian pada data. *Schema* ini sendiri karena dari awalnya didefinisikan dalam RDF, maka semua *schema informasi* dari RDFS mempunyai bentuk RDF *triple*. Karena kedua bahasa tersebut diwujudkan dalam bentuk *triple*, maka untuk menyediakan informasi tambahan (*schema* untuk informasi yang kita

punyai) tetap menggunakan *triple,* informasi baru hasil *inference* dari data yang ada juga dalam bentuk *triple*. Dalam RDF, semua direpresentasikan dalam bentuk *triple*.[11]

2.2.3 OWL

Web Ontology Language adalah bahasa untuk mendefinisikan dan menginstansiasi Ontology. Ontology sediri adalah terminology yang dipinjam dari bidang filosofi untuk menjelaskan entitas yang ada di dunia dan hubungan antara entitas tersebut. OWL dituliskan dalam format *triple*, sama seperti RDF dan RDFS.

OWL dan RDF

OWL (Web Ontologi Language) dirancang khusus untuk digunakan pada aplikasi yang butuh untuk memproses informasi daripada hanya untuk sekedar menampilkan informasi kepada manusia. OWL memfasilitas kemampuan pengolahan *Web Content* yang lebih baik dibandingkan yang didukung oleh XML, RDF, dan RDFS dengan menyediakan *vocabulary* tambahan yang berhubungan dengan standar semantic yang formal. Hubungan OWL dengan lapisan Semantic Web lainya dapat digambarkan seperti dibawah [5]:

- **XML** menyediakan *syntax* untuk mengatur struktur dokumen, namun tidak memiliki batasan yang memberikan makna terhadap dokumen itu sendiri.
- **XML Schema** adalah bahasa yang membatasi struktur XML dan memperkaya XML dengan menggunakan tipe data.
- **RDF** adalah datamodel untuk *resource*, dan memberikan relasi diantara *resource* tersebut beserta dengan nilai *semantic* sederhana untuk datamodel tersebut. Pemodelan data dalam bentuk RDF dapat direpresentasikan dalam bentuk XML yaitu bentuk RDF/XML
- **RDF Schema (RDFS)**, *vocabulary* (dalam hal ini seperti *default properties*) yang menjelaskan properti dan kelas dari data RDF, menyediakan tambahan kemampuan semantuk untuk generalisasi hirarki properti ataupun hirarki kelas. Misalkan terdapat rdfs:label dan rdfs:comment untuk menerangkan *resource*.
- **OWL** menyediakan lebih banyak lagi *vocabulary* (dalam hal ini seperti *default properties*) yang properti dan kelas, misalnya menjelaskan tentang *disjoint*, cardinalitas, *equality*, *intersection*, karakteristik property (simerti, *transitive*).

Jenis-jenis OWL

OWL sendiri dibagi lagi berdasarkan tingkat kemampuan ekspresif yang dapat ditanganinya atau tingkat keformalan bahasanya, terdapat 3 level bahasa OWL: OWL Lite, OWL DL dan OWL Full. OWL Lite merupakan OWL DL yang diperkecil, OWL DL merupakan OWL Full yang diperkecil. Berikut adalah penjelasannya:

- **OWL Lite**: Merupakan OWL yang paling sederhana dibandingkan OWL yang lain, tingkat formalitasnya rendah, walaupun tetap memberikan ekspresifitas yang lebih tinggi dari RDF karena dapat dibilang OWL adalah ekstensi dari RDFS.
- **OWL DL**: Merupakan OWL yang lebih kompleks dari OWL Lite namun lebih seserhana dari OWL Full. Bahasa ini didasarkan pada *description logic*, sehingga lebih ekspresif daripada OWL Lite, selain itu juga mempunyai tambahan fitur dan property. Mempunyai tingkat fleksibilitas yang masih menjamin yang kita dapat melakukan *reasoning* dengan *reasoning software* (*automated resoning)*.
- **OWL Full**: Merupakan OWL yang paling kompleks, dengan kata lain batasan-batasan pada syntax sebegitu kecil, sehingga memungkinkan fleksibilitas yang sangat tinggi. Dengan menggunakan OWL Full, kita bahkan bisa menambah makna dari *pre-defined vocabulary* sehingga tidak menjamin kita dapat melakukan *reasoning* dengan *reasoning software* (*automated resoning)*.

Perbedaan pada ketiga jenis diatas ada untuk mempermudah developer untuk mengembangkan ontologi sesuai dengan kebutuhan, jadi pembuatan ontologi sederhana yang tidak membutuhkan automated, tidak perlu sampai menggunakan yang OWL Full, cukup OWL Lite atau OWL DL yang sudah cukup *powerful*.

Pengunaan OWL

Berdasarkan dokumen W3C mengenai OWL Use Cases and Requirements [4], 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 *ubiquitos computing network*.

2.2.4 Ontologi

> An ontology is an explicit specification of a conceptualization
>
> - Tom Gruber , 1993[7]

Salah satu definisi tentang ontologi yang paling singkat adalah dari Tom Gruber yang memberikan definisi ontologi sebagai "*explicit specification of a conceptualization*", walaupun sebenarnya definisi tersebut menimbulkan banyak perdebatan , terutama penggunaan kata "*specification*" dan "*conceptualization*"[4], namun terdapat poin penting cari definisi ontologi tersebut yaitu: ontologi menjelaskan tentang konsep, relasi dan pembedaan lain yang perlu digunakan untuk memodelkan sebuah domain dan spesifikasi tersebut harus mempunyai makna dalam penggunaannya. Dengan definisi seperti diatas, maka kita dapat katakan bahwa ontologi juga adalah model karena keduanya menjelaskan tentang sesuatu.

Komponen Ontologi

Secara garis besar ontologi terdiri dari beberapa komponen antara lain:

[4] Sanggahan tentang ambiguitas penggunaan kata "specification" dan "conceptualization" dapat dilihat dalam paper "Understanding, Building and Using Ontologies" yang merupakan karya Nicola Guarini[19]. Disini dijelaskan beberapa definisi ontologi dan masalah pada pengertian dari definisi tersebut.

- Class atau Concept. Suatu konsep yang ingin dibahas.
- Property atau Properties. Karakteristik dari suatu konsep.
- Individual. Instance dari class, sebuah objek.
- Relationship. Hubungan antar class dan individu yang saling berelasi
- Rules. Statement dalam bentuk jika-maka untuk memberikan informasi tentang relasi
- Function. Struktur kompleks yang terbentuk dari berbagai relasi.

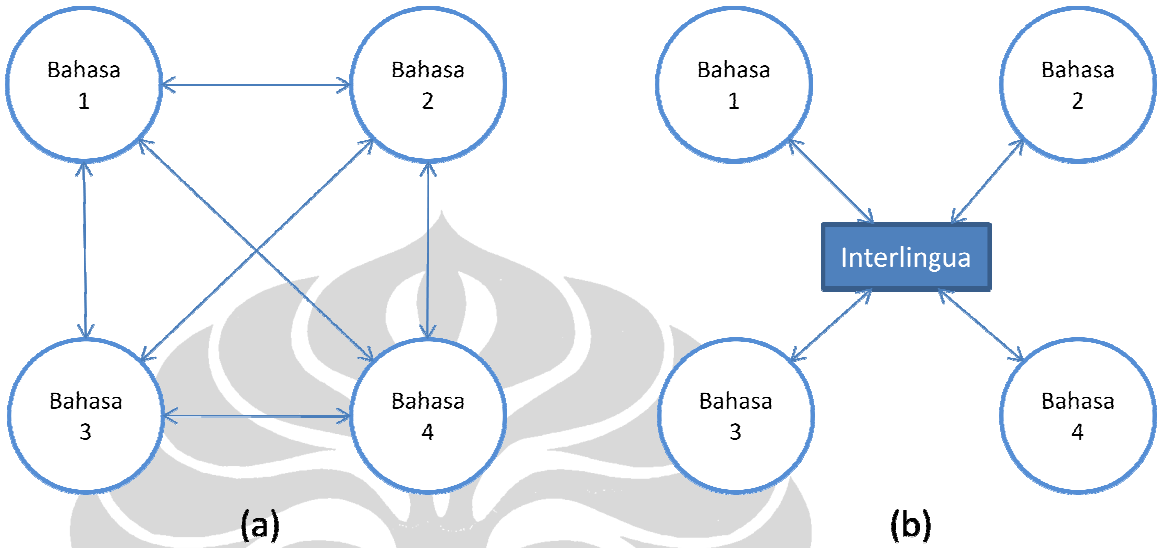
Untuk formatnya sendiri telah ada standar yang baku dipakai untuk merepresentasikan semua ontologi. Sebuah ontologi mengandung penjelasan baku tentang **konsep** (diwujudkan dalam bentuk class seperti pada pemrograman Object Oriented) dalam domain tertentu, ***property*** dari setiap konsep yang menjelaskan beragam fitur dan atribut dari konsep, ***restriction*** terhadap sebuah property seringkali disebut facet (membuat sifat khusus property yang menjelaskan konsep tertentu, berfungsi untuk membatasi). Sebuah ontologi bersama dengan sejumlah individual(instance dari suatu class) membentuk sebuah knowledge base, karena dalam sebuah knowledge base kita hanya perduli pada data yang real, konsep hanya sekedar bayang-bayang bahwa suatu kelas ada sehingga *instance* dari kelas itu juga ada. Berbeda dengan knowledge base, ontologi berpusat kepada konsep/class. Jadi ketika kita berbicara tentang ontologi, maka kita dapat langsung membayangkan bahwa ontologi seperti konsep class yang merepresentasikan dunia pada paradigma Object Oriented.

Kegunaan Ontologi

Setelah sedikit penjelasan tentang ontologi, kita akan melihat kegunaan ontologi. Menurut Mike Ushold dan Michael Gruninger [9], ontologi sendiri muncul sebagai solusi dari beberapa permasalahan yang timbul, diantaranya:

- Permasalahan komunikasi antar orang (COMMUNICATION). Seringkali orang dapat salah mengungkapkan maksud dan tujuan karena permasalahan bahasa. Namun, dengan perancangan ontologi yang baik kita dapat mengidentifikasi konsep yang penting, mendefinisikan konsep dan membuat relasi diantara mereka, proses ini dikenal dengan nama *unifying framework*. Dengan cara ini, ontologi dapat membagikan pengertian dan komunikasi antara orang dengan berbagai kebutuhan dan sudut pandang yang muncul dari konteks masing-masing.

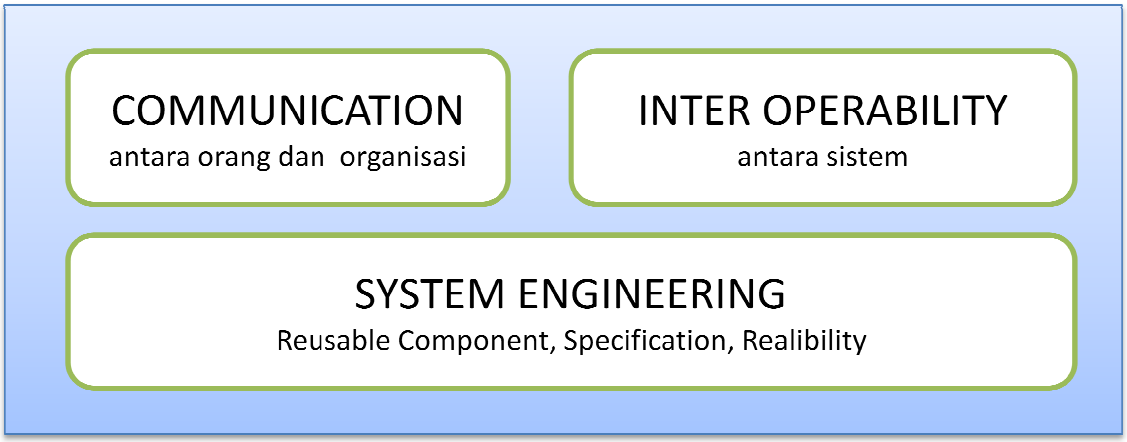
- Kebutuhan akan inter-operabilitas sistem (INTEROPERABILITY). Hal ini seringkali diakibatkan oleh keterbatasan metode, bahasa, paradigma ataupun software tools. Ontologi dapat digunakan untuk mendukung translasi antar bahasa dan representasi. Hal ini dikarenakan ontologi dapat berfungsi sebagai *interlingua*.



Gambar 2. 7 - Ontologi sebagai Pengantara antar Bahasa

Gambar a: Translasi antar bahasa harus dilakukan masing-masing.
Gambar b: Translasi antar bahasa dilakukan ontologi sebagai interlingua.

- Kesusahan mengidentifikasi kebutuhan karena susah mendefinisi spesifikasi sistem (SYSTEM ENGINEERING) dalam konteks permasalahan komunikasi untuk membangun sistem IT. Ontologi dapat digunakan untuk memfasilitasi proses identifikasi kebutuhan sistem dan relasi diantara komponen-komponen sistem. Selain itu, karena ontologi menyediakan spesifikasi sistem secara declarative, sehingga memungkinkan kita terfokus pada tujuan sistem itu dibangun dibandingkan dengan bagaimana fungsionalitas sistem dibangun.



Gambar 2. 8 - Gambar Tiga Kegunaan Ontologi