ISSN : 2338-4018

PENERAPAN WEB SEMANTIK DALAM PENCARIAN KATALOG BUKU DI PERPUSTAKAAN STMIK SINAR NUSANTARA SURAKARTA

Hendro Wijayanto (indra_mhss@yahoo.co.id)
Wawan Laksito YS (wlaksito@yahoo.com)
Teguh Susyanto (teguhsusyanto@gmail.com)

ABSTRAK

Dalam penelitian ini membahas perancangan model pencarian katalog buku di perpustakaan STMIK Sinar Nusantara Surakarta. Model pencarian tersebut menggunakan teknologi berbasis semantik. Dalam penulisan ini hal terpenting adalah merancang ontologi untuk katalog buku itu sendiri. Perancangan ini tidak meninggalkan kaidah awal yang sudah ada dalam web perpustakaan sendiri, dan ditambahkan metode Dublin Core. Komponen yang digunakan dalam teknologi semantik adalah RDF yang dipergunakan sebagai representasi pengetahuan yang digunakan, kemudian SPARQL yang digunakan sebagai query untuk mengambil informasi yang terdapat dalam Ontologi RDF. Dalam web servis ontologi, digunakan OpenRDF-Sesame. Hasil dari perancangan nantinya di uji dengan menggunakan aplikasi simulasi sederhana dengan pemrograman PHP yang terbagi atas 3 kategori pencarian, keyword, simple dan advanced search. Pengujian yang didapat akan menghasilkan bahwa keyword search memiliki kecenderungan waktu yang lebih lama dibanding simple dan advanced searh. Pemetaan dalam query SPARQL akan mempengaruhi seberapa cepat aplikasi akan menampilkan hasil pencariannya

***Kata kunci** : web semantik, pencarian, sesame, ontologi, RDF*

I. PENDAHULUAN

Latar Belakang

Perkembangan teknologi semakin hari semakin pesat. Diantaranya adalah teknologi dalam pengolahan data. Seperti halnya dalam sebuah perpustakaan yang menyimpan ratusan bahkan ribuan koleksi buku. Hal ini jika direpresentasikan kedalam bentuk digital, yang tujuannya adalah pengguna dapat melihat koleksi buku dalam suatu perpustakaan, akan mengalami kesulitan dalam pencarian katalog buku yang begitu besar. Dibutuhkan banyak waktu untuk melakukan satu langkah pencarian data yang begitu besar. Teknologi semantik merupakan teknologi yang dapat melakukan penyimpanan dan pencarian data yang begitu besar.

Perumusan Masalah

Dari latar belakang dapat diambil perumusan masalah sebagai berikut :

a. Bagaimana merancang ontologi katalog buku di perpustakaan STMIK Sinar Nusantara, mulai dari perancangan dan melakukan query terhadap ontologi tersebut.
b. Bagaimana mengimplementasikan kedalam aplikasi pencarian katalog buku.

Pembatasan Masalah

Adapun beberapa yang akan dibahas didalamnya adalah :

a. Hasil pencarian ditampilkan meliputi judul, penulis, abstrak, ISBN-ISSN, halaman, bahasa dan penerbit.
b. Aplikasi pencarian dibangun dengan pendekatan *Dublin Core*, *query SPARQL*, *PHP*, *cURL* dan *RDF Database*.
c. Aplikasi yang dibangun hanya sebatas simulasi sederhana untuk menunjukkan pengimplementasian ontologi kedalam bahasa pemrograman PHP dan bukan merupakan aplikasi yang utuh.
d. Aplikasi pencarian yang dibangun dilakukan berdasarkan judul, pengarang, penerbit, abstrak, tahun dan rak buku.

Tujuan

a. Dapat merancang ontologi pencarian katalog buku untuk perpustakaan STMIK Sinar Nusantara Surakarta.

b. Dapat melakukan *query SPARQL* terhadap ontologi yang sudah dibangun.
c. Dapat mengimplementasikan hasil perancangan ontologi kedalam aplikasi sederhana.

II. METODE PENELITIAN

Metode penelitian ini adalah menggunakan teknologi semantik dengan kolaborasi dari susunan katalog buku yang sudah ada di perpustakaan STMIK Sinar Nusantara Surakarta dan *Dublin Core*. Adapun metode pengumpulan data dalam penelitian ini menggunakan metode observasi, wawancara dan studi pustaka. Perancangan dan pengujian ontologi menggunakan *Protégé* Editor, dan *OpenRDF-Sesame*.

Untuk mengetahui hasil dari pengujian terhadap ontologi yang sudah dirancang, digunakan pengukuran kecepatan. Pengukuran kecepatan merupakan pengukuran seberapa cepat pencarian menampilkan hasil yang dicari. Dan akan diperoleh prosentase hasil dari setiap model pencarian.

III. TINJAUAN PUSTAKA

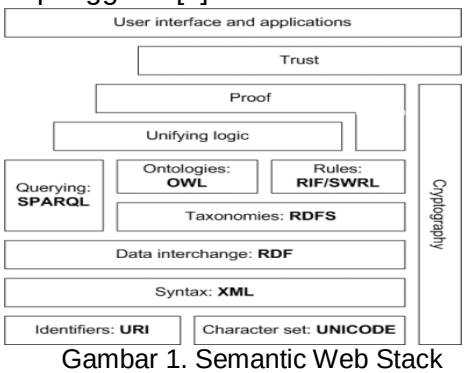
Web Semantik

Semantik berarti arti. Arti dari sumber informasi yang berbasis pengguna atau dapat mengerti intruksi pengguna atau mengerti intruksi dari suatu program yang komplek [1]. Sehingga web semantik adalah informasi dalam jumlah sangat besar yang terhubung secara global dengan suatu cara tertentu dan dimengerti atau dipahami oleh mesin, sehingga dapat diproses secara langsung oleh mesin menjadi pengetahuan untuk ditampilkan kepada pemakai.

Dalam web semantik terdapat berbagai susuna sebagai obyek pokok dalam sebuah aplikasi web semantik. Adapun susunan dari web semantik seperti gambar 1.

Ontology

Istilah ontologi sebenarnya berasal dari istilah filosofi "*ontology*" yang artinya sesuatu yang sesungguhnya ada dan bagaimana menggambarkannya. Dalam dunia komputer ontologi digunakan untuk menspesifikasikan suatu konseptualisasi. Dalam istilah lain ontologi dijelaskan sebagai suatu representasi dari domain pengetahuan tertentu yang berisi istilah-istilah dalam domain tersebut beserta hubungan antara istilah-istilah yang ada yang diarahkan dapat memahami makna suatu kata atau kalimat yang diberikan oleh pengguna [2].

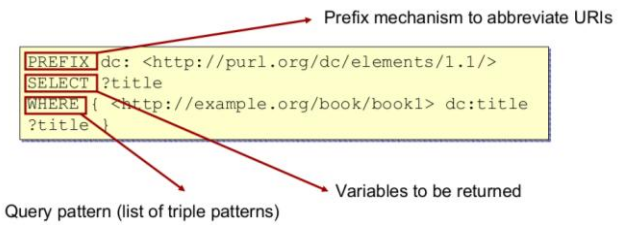


Gambar 1. Semantic Web Stack

SPARQL

SPARQL merupakan bahasa *query* untuk *RDF*. Di dalam lembar rekomendasinya *W3C* menuliskan *SPARQL* menyediakan fasilitas untuk mengekstrak informasi dalam bentuk *URI*, blank nodedan literal, mengekstrak subgraf *RDF*, dan membangun graf *RDF* baru berdasar pada informasi dari graf yang di-*query* [3].

Query SPARQL didasarkan pada pencocokan pola graf. Pola graf yang paling sederhana adalah *triple pattern* yang mirip dengan *RDF triple*, hanya saja pola pada *query* dimungkinkan pemberian nama diluar terminologi *RDF* pada posisi subyek, predikat dan obyek. Model *query SPARQL* serta susunan *query*-nya dapat ditunjukkan dalam gambar 2.



Gambar 2. SPARQL Annatomy

RDF

RDF (*Resource Description Framework*) merupakan bahasa yang digunakan untuk merepresentasikan metadata. *RDF* mendukung interoperabilitas antar aplikasi yang

mempertukarkan informasi yang bersifat machine-understandable di web. *Semantic Web* terdiri dari data yang ditulis dalam bahasa yang dimengerti oleh mesin, seperti *RDF*. *RDF* menggunakan graf untuk merepresentasikan kumpulan pernyataan. Simpul dalam graf mewakili suatu entitas, dan tanda panah mewakili relasi antar entitas [4].

RDF terdiri dari 3 obyek tipe, yaitu Sumber daya (*Resources*), Properti (*Property*), dan Pernyataan (*Statement*). Adapun contoh penulisan *RDF* seperti gambar 3:

```
<rdf:RDF xml:base="http://emeld.org/Hopi_examples">
...
<gold:CompleteLexicalUnit rdf:about="#element(121)">
  <gold:form>
     <gold:LinguisticForm rdf:about="#element(121/1)">
        <gold:symbolicRepresentation>aa'asnatoyni'ywisa
        </gold:symbolicRepresentation>
     </gold:LinguisticForm>
  </gold:form>
  <gold:meaning>
     <gold:CompositionalSense rdf:about="#element(121/2)">
        <gold:translation>perform washing of the
hair</gold:translation>
     </gold:CompositionalSense>
  </gold:meaning>
  <gold:grammar>
     <gold:MorphosyntacticInformation rdf:about="#element(121/3)">
        <gold:features rdf:parseType="Collection">
        <rdf:Description rdf:about="#vt"/>
        <rdf:Description rdf:about="#i"/>
        <rdf:Description rdf:about="#pl"/>
        </gold:features>
     </gold:MorphosyntacticInformation>
  </gold:grammar>
</gold:CompleteLexicalUnit>
...
</rdf:RDF>
```

Gambar 3. Contoh RDF

Dublin Core

Dublin Core adalah salah satu skema metadata yang digunakan untuk *web resource description and discovery.* Dublin Core terdiri atas 15 unsur dasar, yaitu: *Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights* [5].

PHP

PHP merupakan bahasa pemrograman skrip yang diletakkan dalam server yang biasa digunakan untuk membuat aplikasi web yang bersifat dinamis. *PHP* mendukung berbagai database. Dengan demikian database yang dibuat dapat diakses oleh *PHP* dan memungkinkan untuk menampilkan isinya atau bahkan memanipulasi datanya melalui halaman web [6].

OpenRDF-Sesame

Sesame merupakan kerangka open source untuk query dan menganalisa data *RDF*. Itu diciptakan, dan masih dipertahankan, oleh *Dutch software Aduna*. Ini pada awalnya dikembangkan sebagai bagian dari "*On-To-Knowledge*", sebuah proyek web semantik yang bermula dari 1999 sampai 2002.

Sesame mendukung dua bahasa *query*: *SeRQL* dan *SPARQL*. Komponen lain dari Sesame adalah *Alibaba*, sebuah API yang memungkinkan untuk pemetaan kelas *Java* ke ontologi dan untuk menghasilkan file *Java* dari ontologi. Hal ini memungkinkan untuk menggunakan ontologi tertentu seperti *RSS, FOAF dan Dublin Core*, langsung dari Java.

Triplestores lainnya dapat digunakan melalui *Sesame*, termasuk *Mulgara*, dan *AllegroGraph.*

IV. HASIL DAN PEMBAHASAN

Perancangan Ontologi

Untuk merancang aplikasi berteknologi semantik harus telebih dahulu memiliki ontologi. Dalam penelitian ini ontologi diberi nama "*digilibsinus*".

Adapun struktur ontologinya adalah sebagai berikut :

a. *Class Book*, berisi *datatype title, abstract, year, subject, collation, classification, file, image, edition, isbn-issn,* dan *type*.
b. *Class Organization*, berisi *datatype organizationName* dan *place*.
c. *Class Person*, berisi *datatype personName* dan *address*.
d. *Class Bookcase*, berisi *datatype availability*.

Dari kelas yang sudah terdefinisikan serta *datatype* yang sudah didefinisikan pula, kemudian diperlukan suatu relasi yang menghubungkannya. *Class* bertindak sebagai subjek ontologi, dan *datatype* merupakan objek dalam ontologi. Predikat dalam ontologi didefinisikan sebagai penghubung antara subjek dan objek agar memiliki arti kesatuan.

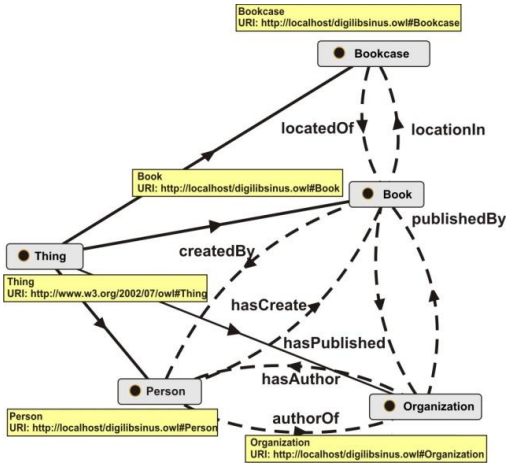
Adapun predikat dari ontologi merupakan *object ontology*, bentuknya seperti tabel 1.

Tabel 1. Domain Range digilibsinus Ontology

Domain	Object Ontology	Range
Book	*createdBy*	Person
Person	*hasCreate*	Book

Book	*publishedBy*	Organization
Organization	*hasPublished*	Book
Book	*locationIn*	Bookcase
Bookcase	*locatedOf*	Book
Person	*authorOf*	Organization
Organization	*hasAuthor*	Person

Skema dari ontologi *digilibsinus* yang dirancang dengan *Protege* ditunjukkan pada gambar 4:



Gambar 4. Kutipan skema ontologi digilibsinus

Perancangan RDF

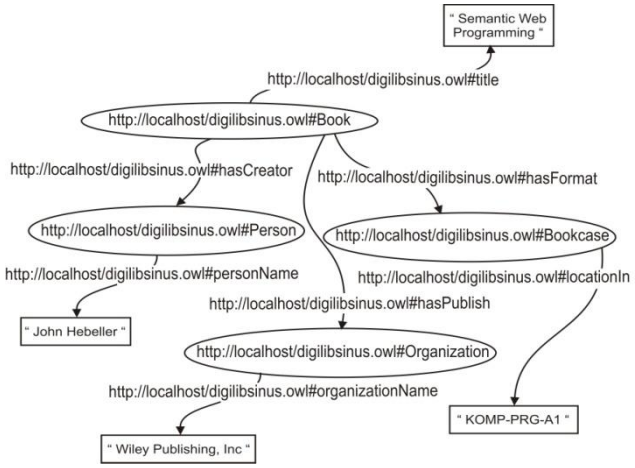
Setelah ontologi dirancang, maka hal selanjutnya adalah melakukan import ontologi *digilibsinus* ke dalam *OpenRDF-sesame*. Dimana natinya didalamnya akan diperoleh bentuk *RDF* database, *prefix*, *namespace*, dan dapat dilakukan simulasi *query SPARQL* didalamnya.

RDF dapat divisualisasikan dalam bentuk *graph*. *Graph* akan menjelaskan alur dari *RDF* itu sendiri. Adapun kilasan *RDF Graph digilibsinus* ditunjukkan pada gambar 5.

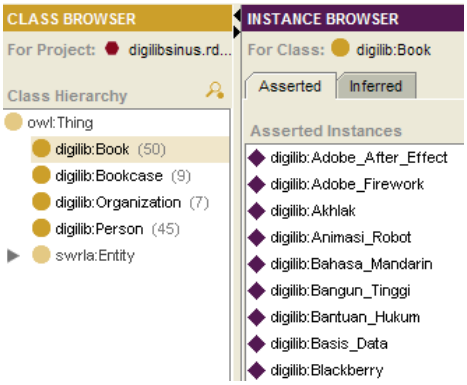
Instance Ontology

Instance adalah memasukkan data-data kedalam sebuah ontologi. Dalam ontologi digilibsinus ini dilakukan *instance* dengan menggunakan editor *Protégé*. Sampel yang digunakan adalah sebanyak 50 katalog buku dari berbagai kategori buku.

Proses *instance* dilakukan seperti halnya memasukkan data biasa sesuai *field*-nya. *Instance* ontologi ditunjukkan pada gambar 6.



Gambar 5. Kutipan RDF Graph



Gambar 6. Instance

Perancangan *Query*

Rancangan *query* dibagi dalam 3 macam, yaitu perancangan *query* untuk pencarian berdasarkan kata kunci, *query* untuk pencarian berdasarkan pencarian sederhana dan *query* untuk *keyword search*.

a. Perancangan *query* pencarian kata kunci

```
PREFIX
digilib:http://localhost/digilibsinus.owl#
SELECT *
WHERE
  {  ?book digilib:title ?title .
     ?book digilib:createdBy ?person .
     ?person digilib:personName
?personName .
     ?book digilib:title ?title.
     ?book digilib:publishedBy
?organization.
     ?organization
digilib:organizationName
?organizationName.
     ?book digilib:collation ?collation.
     ?book digilib:subject ?subject.
     ?book digilib:language ?language.
     ?book digilib:abstract ?abstract.
```

```
    ?book digilib:isbn ?isbn.
    ?book digilib:year ?year.
        FILTER (regex(STR(?title) ,
'".$kunci."', 'i')

  ||regex(STR(?personName) ,
'".$kunci."', 'i')

  ||regex(STR(?abstract) , '".$kunci."', 'i')

  ||regex(STR(?isbn) , '".$kunci."', 'i')

  ||regex(STR(?organizationName) ,
'".$kunci."', 'i')

  ||regex(STR(?subject) , '".$kunci."', 'i')

  ||regex(STR(?language) , '".$kunci."',
'i')

  ||regex(STR(?year) , '".$kunci."', 'i')).
  }
```

Pada *query* diatas, "$*kunci*" merupakan kata kunci yang dimasukkan kedalam *form* pencarian.

b. Perancangan *query* untuk *simple search.*

```
PREFIX
digilib:http://localhost/digilibsinus.owl#
SELECT *
WHERE
  {  ?book digilib:title ?title .
     ?book digilib:createdBy ?person .
     ?person digilib:personName
?personName .
     ?book digilib:title ?title.
     ?book digilib:publishedBy
?organization.
     ?organization
digilib:organizationName
?organizationName.
     ?book digilib:collation ?collation.
     ?book digilib:subject ?subject.
     ?book digilib:language ?language.
     ?book digilib:abstract ?abstract.
     ?book digilib:isbn ?isbn.
     ?book digilib:year ?year.
     ?book digilib:locationIn ?availability.
        FILTER
(regex(STR(?".$kategori.") , '".$kunci."',
'i')).
  }
```

Pada *query* diatas menampilkan seluruh data berdasarkan kategori. Sintak "$*kategori*" merupakan pilihan kategori yang dideklarasikan dalam bentuk *combobox*, dan "$*kunci*" merupakan kata kunci yang dimasukkan dalam *form* pencarian.

c. Perancangan *query* untuk *advanced search.*

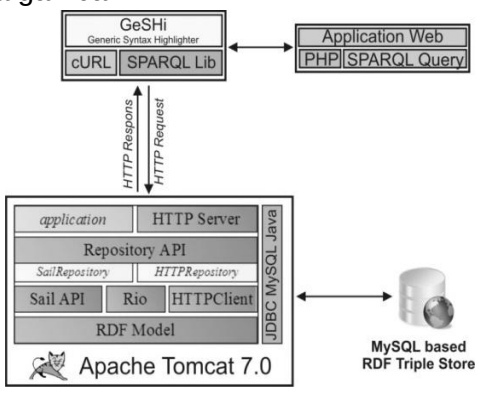
```
PREFIX
digilib:http://localhost/digilibsinus.owl#
SELECT *
WHERE
  {  ?book digilib:title ?title .
     ?book digilib:createdBy ?person .
     ?person digilib:personName
?personName .
     ?book digilib:title ?title.
     ?book digilib:publishedBy
?organization.
     ?organization
digilib:organizationName
?organizationName.
     ?book digilib:collation ?collation.
     ?book digilib:subject ?subject.
     ?book digilib:language ?language.
     ?book digilib:abstract ?abstract.
     ?book digilib:isbn ?isbn.
     ?book digilib:year ?year.
     ?book digilib:locationIn ?availability
        FILTER ("".$bagianWhere.").}
```

Pada *query* "$*bagianWhere*" adalah pemecahan kunci pencarian yang bersifat opsional, dan menggunakan fungsi *AND*, *OR*, dan *NOT*, yang telah dideklarasikan dalam sintak *PHP* sebelumnya.

Perancangan Aplikasi Pencarian

Aplikasi yang akan dirancang menggunakan bahasa pemrograman *PHP*, *cURL* dan *HTTP Request 1.1*.

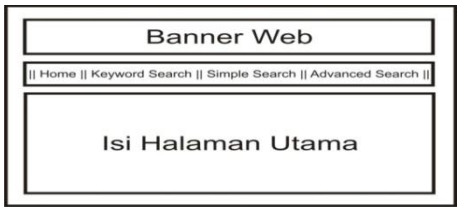
Adapun gambaran umum dari rancangan aplikasi pencarian ditunjukkan pada gambar 7.



Gambar 7. Gambaran Umum Aplikasi

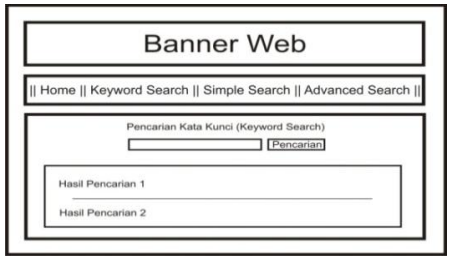
Dalam rancangan aplikasi akan dibagi menjadi 4 bagian, yaitu *Home*, *Keyword Search, Simple Search* dan *Advanced Search*.

Bentuk rancangan halaman utama atau *Home* ditunjukkan pada gambar 8.



Gambar 8. Rancangan Halaman Home

Bentuk rancangan halaman pencarian kata kunci atau *keyword search* ditunjukkan pada gambar 9.



Gambar 9. Rancangan Keyword Search

Bentuk rancangan halaman *simple search* ditunjukkan seperti gambar 10.

Gambar 10. Rancangan Simple Search

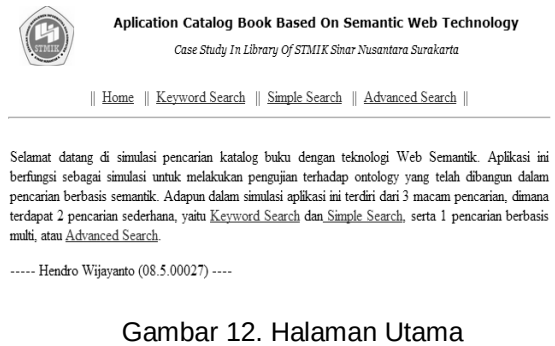
Bentuk rancangan halaman advanced search ditunjukkan pada gambar 11 :

Gambar 11. Rancangan Pencarian Multi

Implementasi

Untuk melakukan implementasi, yang diperispkan adalah *Apache Tomcat* untuk import *OpenRDF-Sesame* ke dalamnya, *Apache XAMPP* sebagai penyimpanan database *MySQL RDF Store*, *cURL*, *HTTP Request 1.1* dan web browser.

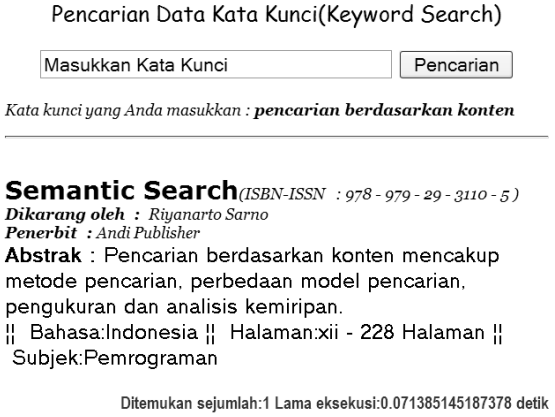
Untuk mengakses halaman aplikasi simulasi pencarian katalog buku, alamat yang dituju dalam web browser adalah *http://localhost/digilibsinus/*. Kemudian akan tampil halaman utama seperti ditunjukkan pada gambar 12.



Gambar 12. Halaman Utama

Untuk malakukan pencarian katalog buku, dapat dilakukan pencarian dengan memilih menu yang berada diatas. Pada aplikasi tersebut, menu yang tersedia adalah *Home, Keyword Search, Simple Search*, dan *Advanced Search*.

Dalam hal ini akan dicontohkan pencarian dengan kalimat potongan abstrak buku "*pencarian berdasarkan konten*" dan akan tampak hasilnya seperti pada gambar 13.

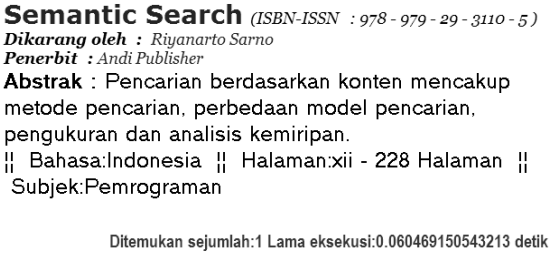


Gambar 13. Pencarian Kata Kunci

Jika dilakukan pencarian dengan *simple search* akan tampak seperti gambar 14.

Gambar 14. Pencarian Sederhana

Jika dilakukan pencarian dengan *advanced search*, hasilnya dapat dilihat seperti gambar 15.



Gambar 15. Pencarian Multi

Hasil Penelitian

Dari ketiga model pencarian yang telah dirancang diatas, maka dapat dilakukan pengujian untuk membandingkan seberapa cepat *query* yang dilakukan oleh masing-masing model pencarian terhadap hasil yang ditampilkan. Sehingga nantinya dapat ditarik sebuah kesimpulan apakah teknologi semantik memang benar cocok dalam penerapan aplikasi yang mempunyai data besar atau tidak.

Proses pengujian dilakukan dengan ketentuan sebagai berikut :

1. Kondisi hardware & software dalam keadaan yang sama saat dilakukan pengujian.
2. Sampel pengujian diberikan sebanyak 50 sampel katalog buku dari segala kategori.
3. Ditetapkan 20 klausa kunci (Q) yang akan menjadi kata kunci sama dalam pencarian.
4. Setiap pencarian diulang sebanyak 10 kali pengulangan untuk satu buah klausa kunci (Q). Sehingga dapat ditarik rata-rata untuk dijadikan pedoman penentuan waktu dalam setiap hasil pencarian dari klausa kunci (Q).

Untuk hasil pengujian yang telah dilakukan dengan perhitungan rata-rata, maka diperoleh hasil yang dapat ditampilkan pada tabel 2.

Tabel 2. Hasil Rata-rata Pengukuran

Q	Rata-rata dalam 10X			F
	Key	Sim	Adv	
Q1	0.04102	0.04194	0.04158	Key
Q2	0.04169	0.04251	0.04119	Adv
Q3	0.04440	0.04445	0.04481	Key
Q4	0.04453	0.04506	0.04421	Adv
Q5	0.04401	0.04584	0.04491	Key
Q6	0.04171	0.04356	0.04254	Key
Q7	0.04849	0.04504	0.04685	Sim
Q8	0.05350	0.05010	0.05440	Sim
Q9	0.06030	0.06000	0.06100	Sim
Q10	0.07490	0.07326	0.07335	Sim
Q11	0.07645	0.07660	0.07626	Adv
Q12	0.07589	0.07503	0.07630	Sim
Q13	0.07947	0.07892	0.07997	Sim
Q14	0.09088	0.09023	0.09042	Sim
Q15	0.09056	0.09053	0.09092	Sim
Q16	0.09177	0.09150	0.09195	Sim
Q17	0.09330	0.09258	0.09285	Adv
Q18	0.09896	0.09735	0.09621	Adv
Q19	0.09894	0.09887	0.09883	Adv
Q20	0.09890	0.09896	0.09889	Adv
QR	**0.06948**	**0.06912**	**0.06937**	

Keterangan :
F = pencarian tercepat
Key = *keyword search*
Sim = *simple search*
Adv = *advanced search*
QR = rata-rata
Q1 = “2009”
Q2 = “hukum”
Q3 = “pemrograman”
Q4 = “john hebeller”
Q5 = “elekmedia komputindo”
Q6 = “pemrograman php”
Q7 = “978-979-29-3110-5”
Q8 = “feng menglong, ling mengehu”
Q9 = “buku robot cerdas”
Q10 = “sistem pendukung keputusan”
Q11 = “metode analisis statistic”
Q12 = “javascript jquery dan ajax”
Q13 = “mobile bisnis dengan J2ME”
Q14 = “perkembangan teknologi CCTV”
Q15 = “belajar javascript dengan jquery”
Q16 = “rancang bangun robot tingkat dasar”
Q17 = “dasar teknik pengujian tegangan tinggi”
Q18 =”menguasai bahasa inggris secara cepat
Q19 = “kisah pilihan dari dinasti song & ming”
Q20 =”tutorial membangun web sekolah dengan cms”

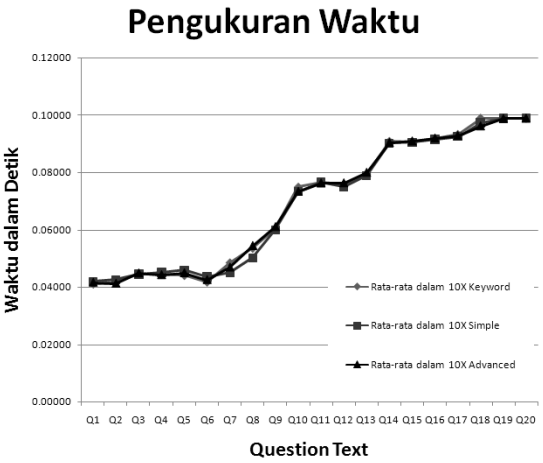
Dari ketiga model dapat dihitung prosentase F dari 20 *Question Test* dengan rata-rata pengulangan 10 kali dihitung dengan perbandingan nilai terendah dari model pencarian yang lain. Prosentase dapat dihitung dengan rumus :

$$P = \frac{\Sigma F}{\Sigma Q} \times 100$$

Keterangan :
P = prosentase Pencarian
ΣF = jumlah pencarian tercepat
ΣQ = jumlah *Question Test*

Sehingga akan didapat prosentase seberapa cepat pencarian dapat melakukan query dan menampilkan hasilnya, dibanding dengan model pencarian lainnya dalam kata kunci yang berbeda-beda (Q).

Hasil rata-rata setiap pencarian pada tabel 2 dapat ditampilkan dalam bentuk grafik seperti terlihat pada gambar 16:



Gambar 16. Grafik Rata-rata Pengukuran

Grafik gambar 16 merupakan hasil pengukuran rata-rata yang dilakukan sebanyak 10 kali untuk setiap *Question Test* (Q) terhadap model pencarian.

Keyword search akan lebih dominan memiliki waktu yang cepat dengan kata kunci tunggal atau pendek. Hal ini disebabkan *query* yang digunakan masih umum, sehingga kata kunci harus dipetakan satu persatu sesuai variabel *query*-nya. Hal ini akan terlihat dari Q1 sampai Q7 yang merupakan kata kunci tunggal & pendek, membuat pencarian *keyword search* menjadi pencarian tercepat dibanding dengan kedua model lainnya. Dengan kesederhanaan kata kunci inilah membuat grafik pencarian memiliki kecenderungan mendatar.

Kata kunci yang beragam akan membuat waktu pencarian menjadi lama. Hal ini tergambar dari grafik gambar 16, bahwa kenaikan terjadi pada Q8 sampai Q20 yang memiliki kata kunci yang bertambah disetiap kenaikan level Q. Dari pengukuran terlihat bahwa *simple & advanced search* mampu meminimalisasi waktu pencarian, karena dalam pencarian tersebut menggunakan pemetaan kata kunci. Sehingga kata kunci dapat dimasukkan ke dalam kelompok-kelompok kata kunci yang dimaksud. Seperti contoh kata kunci tahun "*2009*". Pada *simple & advanced search*, kata kunci tersebut dapat langsung dimasukkan kedalam pencarian berdasarkan tahun.

V. PENUTUP

Kesimpulan

Dari hasil penelitian hal utama yang mendasari perpedaan waktu disini adalah model *query SPARQL*. Model *query SPARQL* yang masih luas (dalam hal ini *keyword search*) sangat mempengaruhi waktu pencarian. Karena setiap kata kunci yang dimasukkan akan dicocokkan sesuai jenis variabel yang sesuai di *query SPARQL*. Ini terlihat pada model *keyword search* yang mempunyai rata-rata waktu 0.06948 detik atau memiliki prosentase pencarian tercepat 20% dari 20 *Question Test* terhadap dua model pencarian yang lain (*simple & advanced search*).

Model *query* yang sudah dipetakan dengan pilihan-pilihan menu seperti *combobox* dan *checkbox* (dalam hal ini *simple & advanced search*), sangat membantu dalam pencarian, karena setiap kata kunci akan langsung dipetakan kedalam jenis variabel di *query SPARQL*. Hal ini terbukti *simple searh* memiliki rata-rata 0.06912 detik atau memiliki prosentase pencarian 45%, dan *advanced search* memiliki rata-rata 0.06937 detik atau memiliki prosentase pencarian 35%.

Dari ketiga model pencarian memiliki kecenderungan waktu yang semakin meningkat sejajar dengan banyaknya kata kunci yang dimasukkan.

Saran

Dari hasil penelitian ini masih banyak yang dapat dikembangkan untuk menjadi

sebuah aplikasi yang baik. Untuk perkembangan kedepannya dapat dilakukan pemodelan pencarian dengan penambahan *vocabulary* kata kunci, sehingga pencarian dapat dilakukan dengan padanan kata dari suatu kata kunci. Hal ini dapat menggunakan metode *Language Based Matching* atau *Vector Space Matching* sebagai salah satu metode untuk melakukan pengukuran kemiripan.

Penambahan data yang lebih banyak dan beragam dapat dilakukan untuk mengetahui hasil kecenderungan dari pengukurannya lebih lanjut, sehingga dapat diketahui perbedaan antara hasil penelitian ini.

DAFTAR PUSTAKA

[1] Hebeler, John., Fisher, Matthew., Blace, Ryan., dan Perez-Lopez, Andrew., 2009, *Semantic Web Programming*, Wiley Publishing, Inc., Indianapolis, Indiana

[2] Zebua, Javier., 2010, *Aplikasi Pencarian Buku Berbasis Web Semantik Untuk Perpustakaan SMK Yadika 7 Bogor*, Universitas Gunadarma

[3] Seaborne, A. dan Prud'hommeaux, E., 2008, *SPARQL Query Language for RDF, Recommendation, World Wide Web Consortium (W3C)*, *http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/* Lastest version available: *http://www.w3.org/TR/rdf-sparql-query/*

[4] Stoermer, Heiko., *RDF Standard and Technologies*, University of Trento, Italy, Dec 06.

[5] Nurkhamid, Mukhamad., 2009, *Aplikasi Bibliografi Perpustakaan Berbasis Teknologi Semantik*, Jurnal Fakultas Teknik, Universitas Maria Kudus

[6] Abdul Kadir, 2008, *Belajar Database Menggunakan MySQL*, Andi Publisher, Yogyakarta, hal 358.