Bab 3 Analisis dan Perancangan

Bab ini berisi penjelasan mengenai analisa terhadap ontologi, kemudian perancangan visualisasi menggunakan portalCore. Perancangan visualisasi terdiri atas perancangan data dan perancangan ontologi

3.1 Analisis Ontologi Learning Performance

Ontologi *Learning Performance* ini didasarkan bahwa sebuah sistem E-Learning harus mempunyai fungsi untuk melakukan pengukuran *performance* untuk mengukur daya tangkap dari pembelajar tersebut. Hal itu dimaksudkan agar setiap materi pembelajaran yang diberikan sesuai dengan tingkat kemampuan pembelajar, dalam hal ini yang dilihat adalah nilai dari materi pembelajaran sebelumnya, sehingga nantinya pembelajaran akan menjadi lebih efektif. Berdasarkan pemikiran ini materi pembelajaran pada untuk suatu pembelajaran lanjutan akan berbeda antar pembelajar, tergantung pada tingkat performance yang dihasilkan pada pelajaran sebelumnya.

Pada ontologi *Learning Performance* ini, terdapat komponen konsep-konsep yang telah dikuasai (properti *hasPerformance*) oleh pembelajar dan nilai yang diperolehnya. Dalam ontologi ini, nilai atau *performance value* (properti *performanceValue*) dianggap sebagai komponen yang sudah diketahui, entah dimasukan secara langsung ataupun didapat dari ontologi lain. Ontologi ini juga terdapat properti prasyarat dari suatu konsep, sehingga pembelajar tahu bahwa konsep apa yang harusnya sudah dimengeti (properti *hasPrerequisite*) dan konsep yang akan dipelajari selanjutnya (properti *isPrerequisiteOf*), setelah mempelajari konsep tersebut.

3.1.1 Skenario Personalisasi

Penggunaan ontologi *Learning Performance* tidak bisa berdiri sendiri dan harus digabungkan dengan ontologi lain agar memberikan nilai tambahan dalam proses personalisasi.

Perekomendasi Kuliah yang akan diambil

Informasi tentang performa dari pembelajar akan disimpan dalam ontologi Learning Performance, nantinya informasi tersebut akan merepresentasikan Indeks Prestasi (IP). Dalam peraturan yang berlaku di Fasilkom, IP semester sebelumnya akan mencerminkan jumlah kredit (SKS) yang bisa diambil pada semester berikutnya. Sistem akan mengolah informasi tersebut kemudian mengatur jumlah kredit yang bisa diambil dan mata kuliah yang bisa diambil di semester tersebut, karena dalam sistem sudah ada ontologi yang melakukan pengaturan prasyarat (*prerequisite*) untuk suatu kuliah tertentu

Perancangan materi berdasarkan kuliah prasyarat

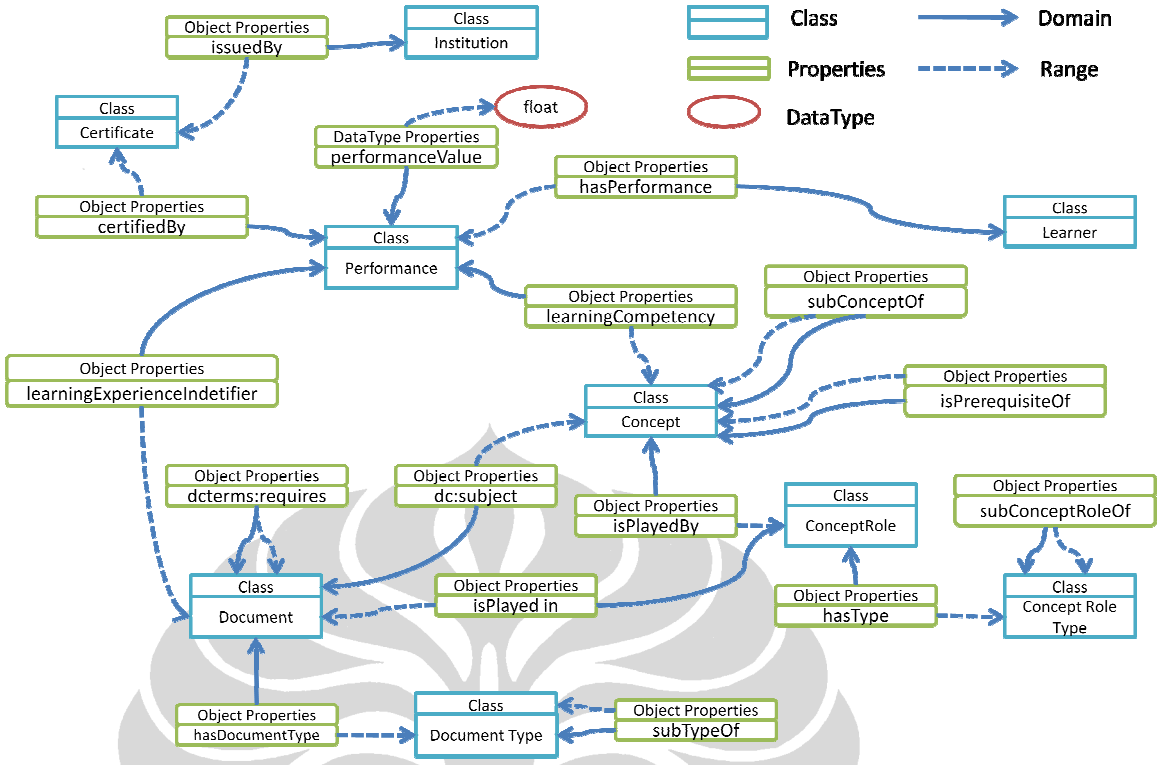
Informasi tentang performa dari pembelajar akan disimpan dalam ontologi Learning Performance. Jika pembelajar mengulang kembali suatu mata kuliah, maka bahan yang diberikan akan disesuaikan dirancang agar lebih mudah dalam pengertian lebih memperhatikan performa pembelajar terhadap kuliah prasyarat. Sistem akan mengetahui bagian mana yang harus diterangkan lebih banyak, karena mungkin saja kegagalan karena kurangnya pemahaman akan kuliah prasyarat.

3.2 Perancangan Data

Pada tahap ini, dilakukan pemilihan ontologi dan persiapan ontologi yang akan dipakai, persiapan tersebut meliputi pemberian instance sebagai data dan pemilihan rules yang akan digunakan.

3.2.1 Pemilihan Ontologi

Ontologi yang digunakan berasal dari paper "*Reasoning and Ontologies for Personalized E-Learning in the Semantic Web*" karya Nicola Henze, Peter Dolog, and Wolfgang Nejdl. Ontologi ini dipilih karena dirasa cukup menggambarkan untuk sebuah ontologi E-Learning, dibandingkan ontologi-ontologi lain. Secara ringkas ontologi dalam paper tersebut dapat dirangkum menjadi:

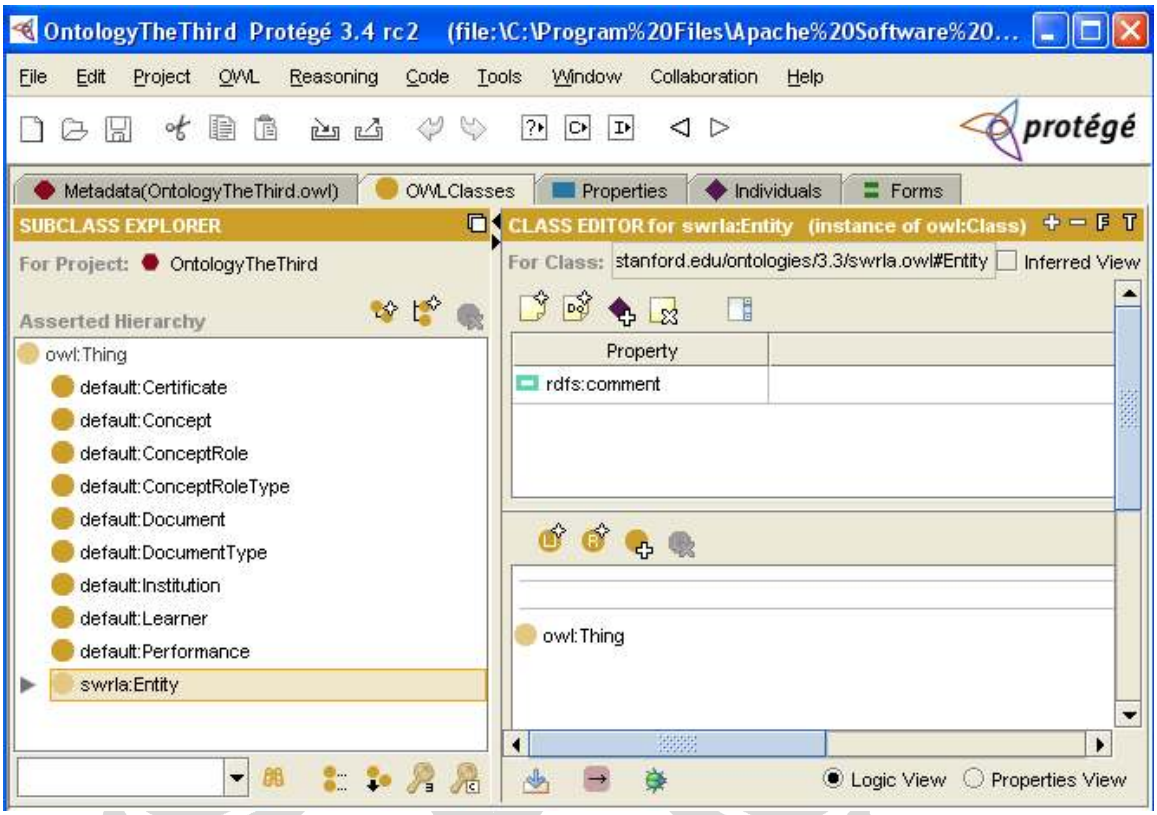


Gambar 3. 1 - Gambar Ontologi *Learning Performance*

Pada gambar diatas, terdapat 9 kelas utama : **Concept**, **ConceptRole**, **ConceptRoleType**, **Learner**, **Performance** , **Certificate**, **Institution**, **Document** dan **DocumentType** dan 12 *object properties* : **isPrerequisiteOf**, **subConceptOf**, **isPlayedBy**, **hasPerformance**, **learningExperienceIdentifier**,**CertifiedBy**, **issuedBy**, **hasType**, **isPlayedIn**, **hasDocumentType**, **dcterms:requires** , **dc:subject**. Adapun 2 object properties terakhir adalah dari ontologi lain yang dipakai disini. 4 buah properties dari gambar tidak dimasukan karena dianggap kurang mendukung ontologi.

Pada Class Performance, terdapat 5 buah properti yang berhubungan, diantaranya **hasPerformance**, **learningCompetency**, **learningExperienceIdentifier**, **learningCompetency**, dan **performanceValue**. Jadi sebuah instance dari class Performance itu mempunyai:

- Dokumen-dokumen yang dipelajari (properti learningExperienceIdentifier).
- Konsep yang dipelajari (properti learningCompetency)
- Pengakuan performa dengan seritikasi (properti certifiedBy)
- Nilai dari performa tersebut (properti performanceValue)
- Orang yang mempunyai performa tersebut (properti hasPerformance)



Gambar 3. 2 - Gambar Ontologi dalam *Ontology Editor* Protege

Untuk menampilkan tampilan yang lebih baik, nantinya akan ada perubahan-perubahan yang dilakukan terhadap ontologi tersebut untuk memberikan tampilan yang mempunyai value yang lebih. Perubahan-perubahan yang terjadi dengan penambahan object properties baru yang merupakan invers objek property lama ataupun juga penambahan rules pendukun untuk menemukan informasi tambahan.

Untuk mengubah/memperbaiki ontologi yang ada, digunakanlah tool bernama Protégé Ontology Editor, yang berfungsi untuk mempermudah pembuatan ontologi. Protégé sendiri adalah tools yang dikembangkan oleh Universitas Stanford khuss untuk mengatur ontologi tentang *biomedical*, yang digunakan pertama kali oleh Stanford Center for Biomedical Informatics Research dibawah naungan Stanford Unversity School of Medicine. Sekarang Protégé adalah *software* yang gratis dan bersifat open source, dapat dipakai siapa saja, pengembangannya menggunakan Java sehingga mendukung sistem *plug-and-play* yang merupakan hal yang penting untuk fleksibilitas pengembangan yang menggunakan metode *rapid prototyping*.

Karena mudah penggunaannya maka penggunaan Protégé ini banyak dipakai untuk merancang ontologi-ontologi lain, contoh yang paling popular dipakai untuk menjelaskan adalah ontologi Pizza dari Universitas Manchester dan ontologi tentang minuman anggur (*wine*). Dalam kaitannya dengan tugas akhir ini, Protégé dipakai untuk merancang ontologi tentang E-Learning dan merubah ontologi.

3.2.2 Pembuatan data

Setelah melakukan perancangan ontologi, maka dilakukan pembuatan data/instance/individu untuk setiap kelas yang ada di ontologi tersebut. Contohnya: di kelas Learner kita akan membuat beberapa orang yang menjadi instance dari kelas Learner seperti Yohanes, Alex, Mierna, Leony dan lain-lain. Jadi ontologi akan terlihat seperti dibawah ini:

```
default:Leony       a       default:Learner
default:Mierna      a       default:Learner
default:Alex        a       default:Learner
default:Yohanes     a       default:Learner
```

Hal ini dilakukan terhadap ke semua kelas dalam ontologi tersebut, namun pembuatan data tidak hanya seperti itu. Individu dibuat tidak hanya sebatas perkenalan tetapi juga properties yang dipunyainya seperti contoh dibawah.

```
default:Yohanes     a       default:Learner ;
                            default:nama "Yohanes Immanuel" ;
                            default:hasLearned default:Decision_Structure.
```

Diatas dijelaskan bahwa **Yohanes** adalah sebuah individu dari kelas **Learner** yang mempunyai property **nama** dan objek dari property tersebut adalah sebuah string **"Yohanes Immanuel"**, juga mempunyai property **hasLearned** dan objek dari property tersebut adalah sebuah objek lain bernama **Decision_Structure**.

```
default:Perf_Yohanes_2 a default:Performance ;
                       rdfs:label "Peformance untuk Konsep Inheritance dalam Objek
Oriented" ;
                       default:isPerformanceOf default:Yohanes ;
                       default:certifiedBy default:K123BF23 ;
                       default:learningCompetency default:OO_Inheritance ;
                       default:learningExperienceIdentifier default:Dokumen_1 ;
```

Diatas dijelaskan bahwa **Perf_Yohanes_2** adalah sebuah individu dari kelas **Performance** yang mempunyai property **rdfs:label** dan objek dari property tersebut adalah sebuah string **"Performance untuk Konsep Inheritance dalam Objek Oriented"**, mempunyai property **hasLearned** dan objek dari property tersebut adalah sebuah objek bernama **Decision_Structure**, mempunyai property **isPerformanceOf** dan objek dari property tersebut adalah sebuah objek bernama **Yohanes**, mempunyai property **certifiedBy** dan objek dari property tersebut adalah sebuah objek bernama **K123BF23**, mempunyai property **learningCompetency** dan objek dari property tersebut adalah sebuah objek bernama **OO_Inheritance**, mempunyai property **learningExperienceIdentifier** dan objek dari property tersebut adalah sebuah objek bernama **Dokumen_1,** , mempunyai property **performanceValue** dan objek dari property tersebut adalah sebuah float bernilai 90.

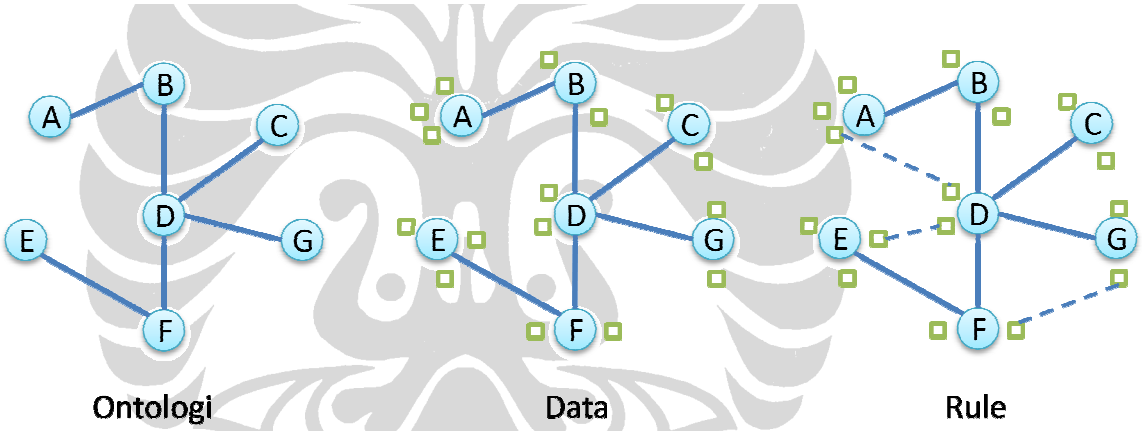
3.3 Perancangan Portal

Untuk memperlihatkan hasil dari ontologi ini, digunakan portalCore sebagai tools untuk dapat menampilkan individu dari ontologi yang dipakai. Seperti Protégé, pada awalnya portalCore merupakan hasil karya untuk sesuatu, baru kemudian dijadikan open-source. Tools ini dibuat oleh The Semantic Web Enviroment Directory (SWED) Project yang merupakan bagian dari SWAD-Europe (Semantic Web Advance Development) yang dibiayai oleh Uni Eropa. Adapun tujuan proyek ini adalah membangun sistem yang Web-based untuk membangun dan mengatur Community Information Resource/portal dan fokus SWED adalah untuk menciptakan direktori tentang organisasi lingkungan

hidup yang ada di Inggris. Hasil dari proyek ini adalah sebuah prototipe direktori yang berisi tentang organisasi lingkungan hidup yang ada di Inggris (http://www.swed.org.uk) , sebuah software yang dapat diunduh agar setiap orang dapat membuat direktori portalnya sendiri, dan laporan detail mengenai sistem[12].

Secara eksekusi program, portalCore dapat dibagi menjadi 4 bagian: ontologi, data, rules dan template. Direktori utama dalam portalCore terdapat dalam folder **\WEB-INF\source.n3** (dari root), dari sinilah kita mengatur semua konfigurasi portal.

Gambar 3. 3 - Gambar Alur Proses dari portalCore



Gambar 3. 4 - Gambar Contoh Proses dalam portalCore

Prinsip kerja portalCore digambarkan pada gambar 3.3 dan 3.4, prinsip kerjanya adalah sebagai berikut:

- Portal mengambil **ontologi** yang berupa Class dan Property[5], sehingga membentuk struktur dari knowledge yang dimiliki. Contoh: terbentuk kelas A sampai G dengan properti. Ontologi ditaruh pada folder **\data** dan memiliki ekstensi .owl.
- Portal mengambil **data** yang ada. Data yang diambil mempunyai format N3. Setelah data diambil maka data akan ditempatkan di kelas data tersebut berada. Contoh: *instance* dari kelas **A** akan digambarkan seperti kotak hijau yang mengelilingi kelas A, begitu juga untuk kelas lain. Karena kelas **A** mempunyai properti (garis) dengan

[5] Merupakan komponen ontologi. Lihat bagian 2.2.4 tentang komponen ontologi

kelas **B**, maka secara langsung *instance* dari kelas **A** bisa saja mempunyai properti dengan instance kelas **B**, namun *instance* dari kelas **A** tidak memiliki properti dengan *instance* dari kelas **D**, karena kelas A tidak berhubungan langsung dengan kelas **D**. Data ditaruh pada folder **\data** dan memiliki ekstensi .n3, boleh memiliki banyak file sebagai input..

- Portal membuat hubungan antar *instance* berdasarkan ***rule*** yang ada. Contoh: Karena kelas **A** mempunyai properti dengan kelas **B** dan kelas **B** mempunyai properti (garis) dengan kelas **D**, maka dengan rule yang tepat bisa saja menemukan informasi baru berupa hubungan antar *instance* yang baru (properti baru) yang digambarkan dengan garis putus-putus. *Rule* ditaruh pada folder **\data**, dan memiliki ekstensi .rules.
- Setelah ontologi dibuat, baru kemudian data ditampilkan dengan menggunakan **template**. *Template* ditaruh pada folder **\template**, dan memiliki ekstensi .vm.

3.3.1 Konfigurasi Ontologi

Konfigurasi ontologi dapat kita kenali dari source.n3 dengan perintah pcv:OntologySourceURL. Perintah seperti dibawah berarti kita akan mengambil ontologi dari folder data yang bernama OntologyTheThird.owl

```
pcv:ontologySourceURL
```

Ontologi diatas merupakan ontologi yang kita rancang dalam Protégé Ontology Editor, ontologi ini disimpan dan dibaca dalam bentuk XML.

3.3.2 Konfigurasi Data

Konfigurasi data dapat kita kenali dari source.n3 dengan perintah pcv:sourceURL. Perintah seperti dibawah kita gunakan untuk mengambil data yang telah disediakan, untuk melakukan panambahan data, cukup dengan menambah link yang ada. Masing-masing data disimpan dalam bentuk n3, yang merupakan standar data dari ontologi dan disimpan di masing-masing tempat, hal itu dimaksudkan agar nantinya penyimpanan data dapat tidak terpencar dan rapih sesuai dengan kelasnya masing-masing

```
pcv:sourceURL <portal://data/Certificate.n3> ;
pcv:sourceURL <portal://data/Concept.n3> ;
pcv:sourceURL <portal://data/ConceptRole.n3> ;
pcv:sourceURL <portal://data/ConceptRoleType.n3> ;
pcv:sourceURL <portal://data/Document.n3> ;
pcv:sourceURL <portal://data/DocumentType.n3> ;
pcv:sourceURL <portal://data/Institution.n3> ;
pcv:sourceURL <portal://data/Learner.n3> ;
pcv:sourceURL <portal://data/Performance.n3> ;
```

Dibawah ini adalah contoh dari kelas Performance (diambil dari data/Performance.n3) disini dijelaskan bahwa **Perf_Yohanes_2** adalah sebuah individu dari kelas **Performance** yang mempunyai *property* **rdfs:label** dan objek dari *property* tersebut adalah sebuah string **"Performance untuk Konsep Inheritance dalam Objek Oriented"**, mempunyai *property* **hasLearned** dan objek dari *property* tersebut adalah sebuah objek bernama **Decision_Structure**, mempunyai *property* **isPerformanceOf** dan objek dari *property* tersebut adalah sebuah objek bernama **Yohanes**, mempunyai *property* **certifiedBy** dan objek dari *property* tersebut adalah sebuah objek bernama **K123BF23**, mempunyai *property* **learningCompetency** dan objek dari *property* tersebut adalah sebuah objek bernama **OO_Inheritance**, mempunyai *property* **learningExperienceIdentifier** dan objek dari *property* tersebut adalah sebuah objek bernama **Dokumen_1,** , mempunyai *property* **performanceValue** dan objek dari *property* tersebut adalah sebuah float bernilai 90. Bentuk seperti ini adalah bentuk N3, yang merupakan salah satu standar dalam representasi ontologi

```
default:Perf_Yohanes_2        a       default:Performance ;
                        rdfs:label "Peformance untuk Konsep Inheritance dalam Objek
Oriented" ;
                        default:isPerformanceOf default:Yohanes ;
                        default:certifiedBy default:K123BF23 ;
                        default:learningCompetency default:OO_Inheritance ;
                        default:learningExperienceIdentifier default:Dokumen_1 ;
```

3.3.3 Konfigurasi Rule

Konfigurasi rule dapat kita kenali dari source.n3 dengan perintah pcv:closureRulesURL. Perintah seperti dibawah kita gunakan untuk mengambil data yang telah disediakan. Pengaturan *rule* dilakukan secara manual. *Rule* dipakai untuk memberikan pengertian tambahan pengertian kepada ontologi.

```
pcv:closureRulesURL   <portal://data/portal.rules> ;
```

Salah satu penggunaan cara penggunaan *rules* adalah *rules* disini dipakai untuk menunjukan sebuah *properties* yang diturunkan dari *properties* lain. *Properties* dibawah menunjukan bahwa jika ada sebuah *instance* (diumpamakan ?A) yang bertipe **Learner**, dan ?A mempunyai *property* **nama** dengan *instance* kedua (diumpamakan ?B), maka ?A juga mempunyai *property* **label** ?B.

```
(?A rdf:type default:Learner), (?A default:nama ?B) -> (?A rdfs:label
```

Rules juga dipakai untuk menyatakan hubungan timbal-balik atau invers, invers dipakai untuk membuat fleksibilitas pengaksesan yang lebih baik lagi. Seperti contoh di bawah dimana sebuah *instance* (diumpamakan ?X) mempunyai *property* **hasProperty** *instance* lain (diumpamakan ?Y) berarti ?Y mempunyai *property* **isPerformanceOf** terhadap ?X, begitu juga sebaliknya.

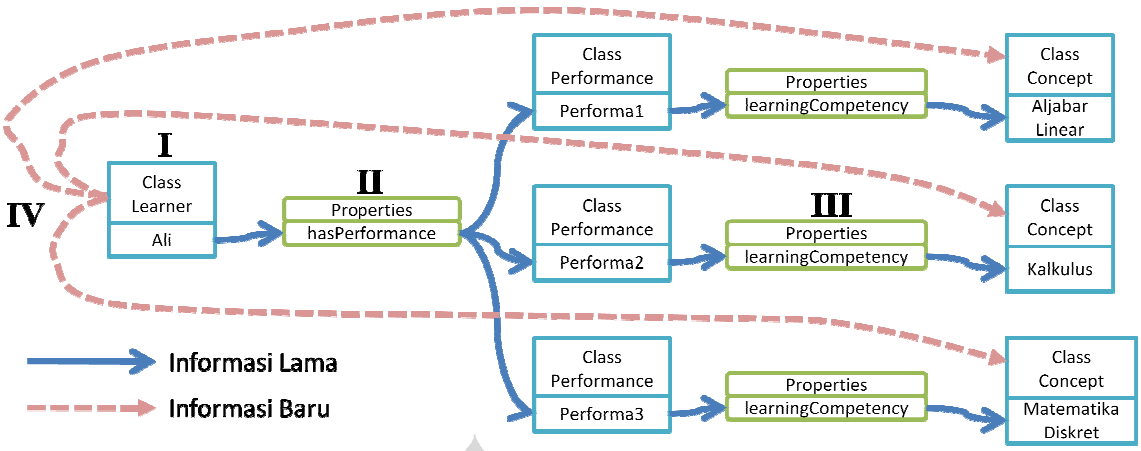
```
(?X default:hasPerformance ?Y) -> (?Y default:isPerformanceOf ?X) .
(?X default:isPerformanceOf ?Y) -> (?Y default:hasPerformance ?X) .
```

Rules juga dipakai untuk melakukan *inferencing*, *inferencing* sendiri adalah proses mendapatkan informasi baru dari informasi yang telah ada sebelumnya. Seperti contoh di bawah dimana sebuah *instance* (diumpamakan ?A) adalah sebuah *instance* dari kelas Learner, ?A mempunyai *property* **hasPerformance** instance lain (diumpamakan ?B), ?B mempunyai *property* **learningCompetency** instance lain (diumpamakan ?C), maka ?A mempunyai *property* **hasLearned** terhadap ?C. Jadi jika ?A sudah mempunyai *performance* terhadap sesuatu konsep, maka adalah hal yang lumrah jika kita mengatakan bahwa ?A sudah mempelajari konsep tersebut, karena tidak mungkin seorang pembelajar diuji jika dia tidak mempelajari konsep itu sebelumnya.

```
(?A rdf:type default:Learner), (?A default:hasPerformance ?B)
,(?Bdefault:learningCompetency ?C)  -> (?A default:hasLearned ?C) .
```

Contoh *inferencing*: Ali (seorang pembelajar/*Learner*) telah mempunyai 3 buah performa (Performa1, Performa2 dan Performa3) dan masing-masing dari performa tersebut memiliki materi/konsep yang diujikan, masing-masing adalah Aljabar Linear, Kalkulus dan Matematika Diskret I. Kita akan coba menjalankan *rule* ini seperti *rule* ini dibaca oleh OWL. Tahapan-tahapannya digambarkan pada gambar 3.4:

- Pertama (**I**), pencarian dilakukan untuk mencari ?A yang merupakan instansiasi dari kelas **Learner**, ditemukan ?A adalah Ali. Hal ini terjadi karena jika dilihat dari sisi *description logic*, Ali adalah jawaban yang muncul karena memenuhi syarat, yaitu: mempunyai **predikat** rdf:type dan mempunyai **objek** rdf:Learner.
- Kedua (**II**), pencarian dilakukan untuk mencari ?B yang merupakan instansiasi dari kelas **Performance**, ditemukan ?B adalah **Performa1**, **Performa2** dan **Performa3**. Hal ini terjadi karena jika dilihat dari sisi *description logic*, **Performa1**, **Performa2** dan **Performa3** adalah jawaban yang muncul karena jawaban tersebut memenuhi syarat, yaitu: mempunyai subjek default:**Ali** dan mempunyai predikat default:**hasPerformance**.
- Ketiga (**III**), pencarian dilakukan untuk mencari ?C yang merupakan instansiasi dari kelas **Concept**, ditemukan ?C adalah **Aljabar Linear** untuk ?B=**Performa1** , **Kalkulus** untuk ?B=**Performa2** dan **Matematika Diskret I** untuk ?B=**Performa3**. Hal ini terjadi karena jika dilihat dari sisi *description logic*, **Aljabar Linear**, **Kalkulus** dan **Matematika Diskret I** adalah jawaban yang muncul karena jawaban tersebut memenuhi syarat, yaitu: mempunyai subjek default:**Performa1**, default:**Performa2** , default:**Performa3** dan mempunyai predikat default: **learningCompetency**.
- Keempat (**IV**), hubungan antara Ali (*Class Learner*) dan **Aljabar Linear**, **Kalkulus** dan **Matematika Diskret I** (*Class Concept*), dihubungkan dengan predikat default:**hasLearned**.



Gambar 3. 5 - Gambar Skenario menemukan Informasi baru dari rule 1

Inferencing juga bisa digunakan adalah untuk mendapatkan rekomendasi konsep yang bisa dipelajari selanjutnya. Seperti contoh di bawah dimana sebuah *instance* (diumpamakan ?A) adalah sebuah *instance* dari kelas Learner, ?A mempunyai *property* **hasLearned** *instance* lain (diumpamakan ?B), ?B mempunyai *property* **isPrerequsiteOf** *instance* lain (diumpamakan ?C), maka ?A mempunyai *property* **ableToLearn** terhadap ?C. Jadi jika ?A sudah mempunyai mempelajari sesuatu konsep, maka adalah hal yang wajar jika kita mengatakan bahwa ?A bisa untuk mempelajari konsep yang merupakan kelanjutan dari konsep yang telah dipelajarinya.

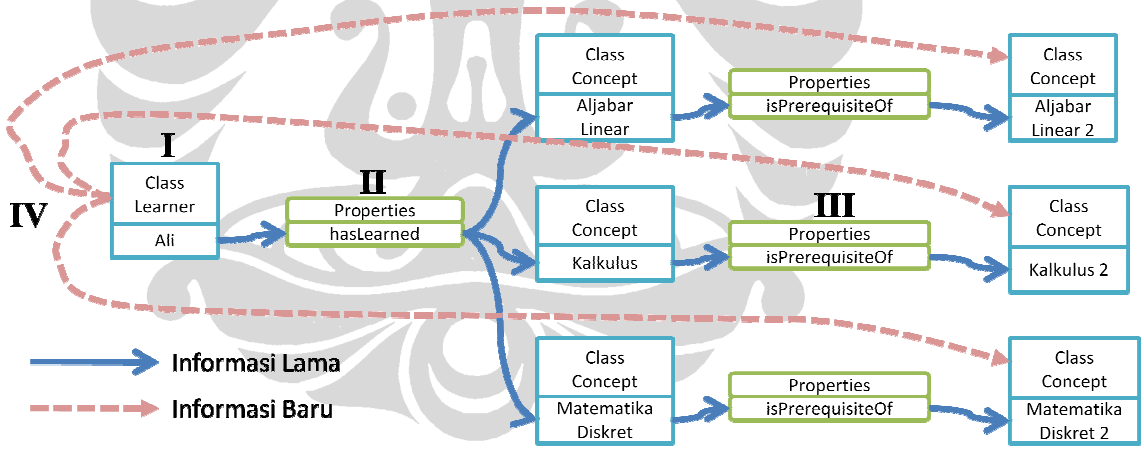
```
(?A rdf:type default:Learner), (?A default:hasLearned ?B), (?B
default:isPrerequisiteOf ?C) -> (?A default:ableToLearn ?C) .
```

Contoh *inferencing*: Ali (seorang pembelajar/*Learner*) telah mempelajari 3 buah konsep (Aljabar Linear, Kalkulus dan Matematika) dan masing-masing dari konsep tersebut memiliki materi/konsep yang menjadi konsep lanjutan dari konsep yang telah dipelajari, masing-masing adalah Aljabar Linear 2, Kalkulus 2 dan Matematika Diskret I 2. Kita akan coba menjalankan *rule* ini seperti *rule* ini dibaca oleh OWL. Tahapan-tahapannya digambarkan pada gambar 3.5:

- Pertama (**I**), pencarian dilakukan untuk mencari ?A yang merupakan instansiasi dari kelas **Learner**, ditemukan ?A adalah Ali. Hal ini terjadi karena jika dilihat dari sisi *description logic*, Ali adalah jawaban yang muncul karena memenuhi syarat, yaitu: mempunyai **predikat** rdf:type dan mempunyai **objek** rdf:Learner.
- Kedua (**II**), pencarian dilakukan untuk mencari ?B yang merupakan instansiasi dari kelas **Concept**, ditemukan ?B adalah **Aljabar Linear**, **Kalkulus** dan **Matematika**

Diskret I. Hal ini terjadi karena jika dilihat dari sisi *description logic*, **Aljabar Linear**, **Kalkulus** dan **Matematika Diskret I** adalah jawaban yang muncul karena jawaban tersebut memenuhi syarat, yaitu: mempunyai subjek default:**Ali** dan mempunyai predikat default:**hasLearned**.

- Ketiga (**III**), pencarian dilakukan untuk mencari ?C yang merupakan instansiasi dari kelas **Concept**, ditemukan ?C adalah **Aljabar Linear 2** untuk ?B=**Aljabar Linear** , **Kalkulus 2** untuk ?B=**Kalkulus** dan **Matematika Diskret I 2** untuk ?B=**Matematika Diskret I**. Hal ini terjadi karena jika dilihat dari sisi *description logic*, **Aljabar Linear 2**, **Kalkulus 2** dan **Matematika Diskret I 2** adalah jawaban yang muncul karena jawaban tersebut memenuhi syarat, yaitu: mempunyai subjek default: **Aljabar_Linear**, default:**Kalkulus** , default: **Matematika_Diskret_I** dan mempunyai predikat default: **isPrerequisiteOf**.
- Keempat (**IV**), hubungan antara Ali (*Class Learner*) dan **Aljabar Linear 2**, **Kalkulus 2** dan **Matematika Diskret I 2** (*Class Concept*), dihubungkan dengan predikat default:**ableToLearn**.



Gambar 3. 6 - Gambar menemukan Informasi baru dari rule 2

Gambar 3.5

3.3.4 Konfigurasi Template

Konfigurasi template dapat kita kenali dari source.n3 dengan perintah pcv:template. Template menggunakan perintah seperti dibawah, dimana terdapat **templatePath** menunjukan alamat template yang yang akan dipakai, dalam hal ini template/pagePerformance.vm dan terdapat **templateClass** untuk menunjukan kelas Performance.

```
pcv:template [ a          pcv:Template;
                  pcv:templateContext "page" ;
                  pcv:templatePath      <portal://templates/pageLEARNER.vm> ;
                  pcv:templateClass     default:Learner;     ];
```

Adapun template ini ditulis dalam format .vm, yaitu format Apache Velocity, format ini mirip dengan format HTML ditambah instance $ dan operasi logic (if-else) seperti PHP. Template diatur sedemikian rupa sehingga mengeluarkan tampilan yang diinginkan, untuk melakukan modifikasi banyak menggunakan API khusus (default dari portalCore) dengan format yang menyerupai PHP. Template digunakan untuk mengatur tampilan yang dikeluarkan oleh portal.

Bab 4 Hasil dan Pembahasan

Bab ini membahas hasil pengembangan prototipe antara lain fungsionalitas semantic portal, contoh skenario penggunaan dari portal yang telah dikembangkan.

4.1 Fungsionalitas

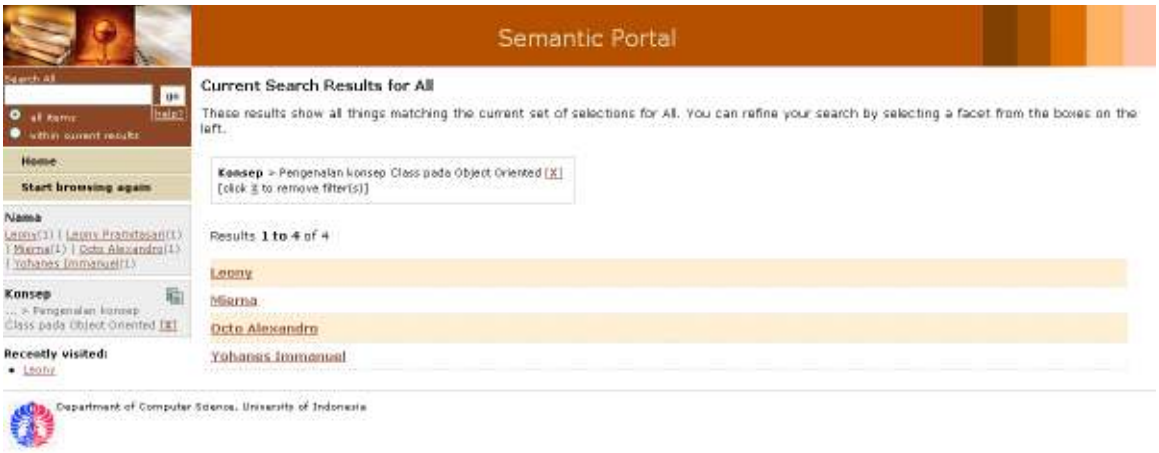
Pencarian menggunakan portalCore dimulai dari halaman utama, yaitu menampilkan facet dimana kategori pencarian ditentukan oleh objek yang ingin kita cari. Dalam portal ini terdapat 2 kategori pencarian : pencarian Pembelajar dan Konsep. Dalam pencarian ini diklasifikasi lagi berdasarkan properties yang digunakan.

Pencarian berdasarkan Konsep yang dipelajari

Sebagai contoh, misalkan kita ingin mencari seseorang pembelajar. Pencarian dapat dilakukan berdasarkan properti-properti yang dimiliki oleh objek bersangkutan. Jika kita ingin melihat pembelajar maka kita akan melihat kembali 2 facet yang ditawarkan. Karena pencarian terhadap pembelajar dapat dilakukan berdasarkan nama sang pembelajar, atau berdasarkan pelajaran yang telah dipelajarinya. Jika kita memilih salah satu kita akan menuju ke halaman browse dimana terdapat pilihan nama pembelajar (hal ini harus dilakukan karena memungkinkan terdapat lebih dari satu pilihan). Setelah itu baru kita masuk ke halaman pembelajar yang kita cari. Gambar dapat dilihat dibawah.

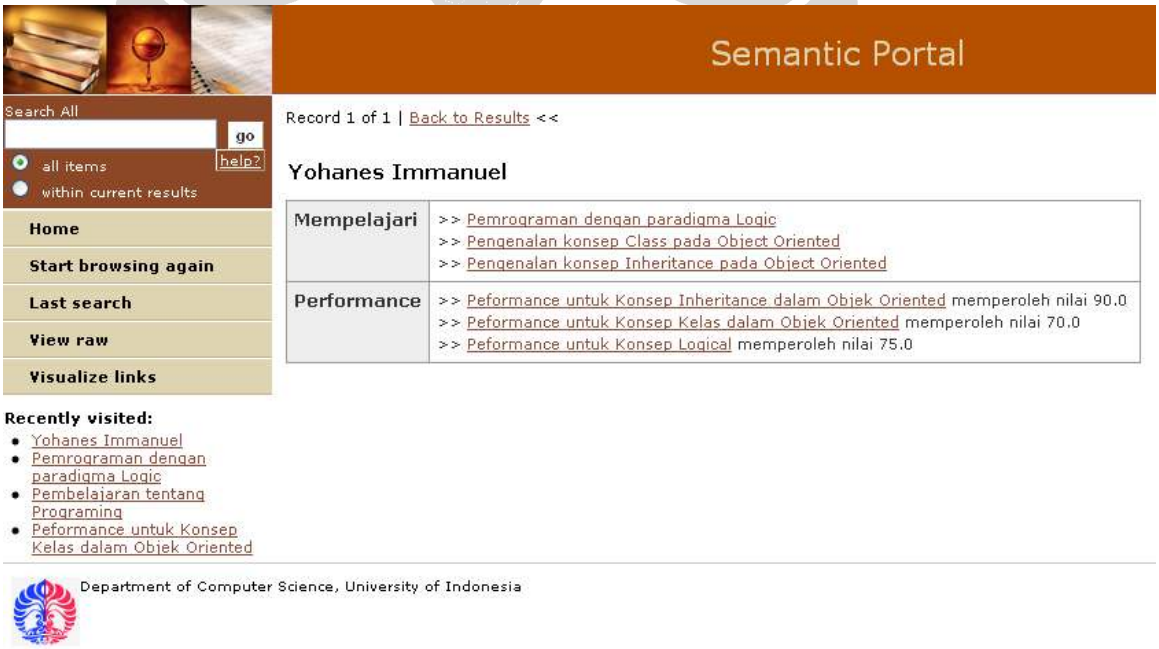
Gambar 4. 1 - Gambar Facet pada portal Core

Hasil dari pencarian kedua facet ini akan menunjukan kepada nama pembelajar. Misalkan kita ingin mencari pembelajar yang telah mempelajari "Pengenalan konsep Class pada Object Oriented" (pada gambar diatas ditunjukan dengan highlight). Nomor disamping pencarian menunjukan jumlah yang didapatkan. Gambar dapat dilihat dibawah.



Gambar 4. 2 - Gambar Data Hasil Query dari portalCore

Pada Gambar 4.2, ditampilkan adanya 4 orang yang mempunyai mempelajari "Pengenalan konsep Class pada Object Oriented", setiap orang yang terdaftar disini berarti telah mempunyai performance untuk konsep yang diujikan. Lalu tinggal memilih orang yang dimaksudkan untuk membuka halaman individu.



Gambar 4. 3 - Gambar Data dari kelas *Learner*

Pada Gambar 4.3, ditampilkan halaman Pembelajar (Class Learner) dimana terdapat 2 kolom, **Mempelajari** menunjukan konsep yang telah dipelajari, sedangkan **Performance** menunjukan performa atas konsep yang telah dipelajarinya beserta dengan nilai yang telah diujikan.

Gambar 4. 4 - Gambar Data dari kelas *Performance*

Pada gambar 4.4,ditampilkan halaman Performance (Class Performance), dimana terdapat 5 kolom, **Nama** menunjukan nama yang mempunyai performa, **Konsep** menunjukan konsep yang telah diujikan, **Nilai** menunjukan hasil pencapaian pembelajar atas konsep tersebut, **Sertifikasi** menunjukan serifikasi yang diberikan atas performa pembelajar, **Dokumen Pendukung** menunjukan materi pengajaran untuk konsep tersebut.

Gambar 4. 5 - Gambar Data dari kelas *Concept*

Pada gambar 4.5,ditampilkan halaman Konsep (Class Concept), dimana terdapat 4 kolom, **Mempunyai Prasyarat** menunjukan Konsep yang harus dipelajari sebelum mengambil konsp tersebut, **Sub Konsep dari** menunjukan konsep besar dari konsep tersebut, **Peran** menunjukan peran dari konsep tersebut secara keseluruhan, **Dipelajari Oleh** menunjukan pembelajar yang telah mengambil konsep tersebut..

Secara garis besar dapat digambarkan dengan flow seperti dibawah

Gambar 4. 6 - Gambar Skenario penggunaan portalCore