

TUGAS AKHIR - IF184802

**RANCANG BANGUN APLIKASI BERBASIS WEB
UNTUK VISUALISASI POHON KELUARGA
TOKOH SEJARAH INDONESIA MENGGUNAKAN
ONTOLOGI DBPEDIA DAN PELLET REASONER**

FAIQ
NRP. 05111540000007

Dosen Pembimbing 1
Nurul Fajrin A., S.Kom., M.Sc.

Dosen Pembimbing 2
Adhatus Solichah A., S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019



TUGAS AKHIR - IF184802

**RANCANG BANGUN APLIKASI BERBASIS WEB
UNTUK VISUALISASI POHON KELUARGA
TOKOH SEJARAH INDONESIA MENGGUNAKAN
ONTOLOGI DBPEDIA DAN PELLET REASONER**

**FAIQ
NRP. 05111540000007**

**Dosen Pembimbing 1
Nurul Fajrin A.,S.Kom., M.Sc.**

**Dosen Pembimbing 2
Adhatus Solichah A.,S.Kom., M.Sc.**

**DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019**

(Halaman ini sengaja dikosongkan)



UNDERGRADUATE THESIS - IF184802

**FAMILY TREE VISUALIZATION OF INDONESIAN
HISTORY ACTORS USING DBPEDIA ONTOLOGY
AND PELLET REASONER**

**FAIQ
NRP. 05111540000007**

**Supervisor 1
Nurul Fajrin A.,S.Kom., M.Sc.**

**Supervisor 2
Adhatus Solichah A.,S.Kom., M.Sc.**

**DEPARTMENT OF INFORMATICS
Faculty of Information Technology and Communication
Sepuluh Nopember Institute of Technology
Surabaya 2019**

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI BERBASIS WEB UNTUK VISUALISASI POHON KELUARGA TOKOH SEJARAH INDONESIA MENGGUNAKAN ONTOLOGI DBPEDIA DAN PELLET REASONER

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Manajemen Informasi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

FAIQ

NRP. 05111540000007

Disetujui oleh Pembimbing Tugas Akhir:

1. Nurul Fajrin A., S.Kom., M.Sc.
NIP. 19860722 201504 2 003 (Pembimbing 1)
2. Adhatus Solichah A., S.Kom., M.Sc.
NIP. 19850826 201504 2 002 (Pembimbing 2)

**SURABAYA
JULI, 2019**

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN APLIKASI BERBASIS WEB UNTUK VISUALISASI POHON KELUARGA TOKOH SEJARAH INDONESIA MENGGUNAKAN ONTOLOGI DBPEDIA DAN PELLET REASONER

Nama : Faiq
NRP : 0511540000007
Departemen : Informatika FTIK-ITS
Dosen Pembimbing I : Nurul Fajrin A.,S.Kom., M.Sc.
Dosen Pembimbing II : Adhatus Solichah A.,S.Kom., M.Sc.

ABSTRAK

Tokoh sejarah di Indonesia, menjadi bukti dari adanya suatu kejadian penting di masa lalu. Setiap tokoh memiliki rekan hidup dan keluarga yang berbeda. Salah satu platform ensiklopedia online yang menyediakan daftar pahlawan nasional Indonesia adalah Wikipedia. Konten dari sebuah halaman Wikipedia memiliki keterkaitan dengan DBpedia dimana DBpedia menyediakan daftar hyperlink yang memiliki keterkaitan dengan halaman Wikipedia tersebut, seperti orang tua, pasangan dan anak cucu.. Namun, seringkali halaman Wikipedia merepresentasikan data tersebut sebagai paragraf, dan halaman DBpedia sebagai tabel.

Dengan adanya data keluarga dari suatu halaman DBpedia, hubungan kekeluargaan tokoh sejarah Indonesia dapat diketahui. Keterkaitan atau relasi tokoh bersejarah dapat digambarkan dengan ontologi. Tujuan dari pengerjaan tugas akhir ini adalah untuk melengkapi data dan relasi keluarga tokoh sejarah di Indonesia dan merepresentasikannya dalam bentuk pohon keluarga.

Langkah-langkah dari pengerjaan tugas akhir ini, pertama-tama melengkapi data tokoh dengan proses reasoning lalu menyimpan data tersebut dalam suatu basis data sehingga bisa ditampilkan secara grafis hubungan keluarga tokoh sejarah dalam bentuk pohon keluarga. Untuk melengkapi data keluarga,

menggabungkan dan menjalankan proses reasoning pada model ontologi dengan data DBpedia sudah terbukti dapat menghasilkan fakta-fakta baru yang belum tercatat dalam DBpedia. Untuk penyimpanan data, Apache Jena-Fuseki dapat menjadi server basis data triple store. Berdasarkan uji coba yang dilakukan, aplikasi berbasis web ini dapat menampilkan pohon keluarga suatu tokoh dan lebih lengkap relasinya dibandingkan dengan DBpedia. Tugas Akhir ini dapat membantu penelitian sejarah dalam menentukan hubungan keluarga dari suatu tokoh sejarah. Hal ini dapat menambah wawasan sejarah bangsa Indonesia terhadap para pelaku sejarah beserta keluarganya.

Kata kunci: Keluarga tokoh sejarah Indonesia, Ontologi, Pohon keluarga, Tokoh Sejarah, Visualisasi.

FAMILY TREE VISUALIZATION DESIGN OF INDONESIAN HISTORY ACTORS USING DBPEDIA ONTOLOGY AND PELLET REASONER

Name : Faiq
NRP : 05111540000007
Department : Informatics FTIK-ITS
First Advisor : Nurul Fajrin A.,S.Kom., M.Sc.
Second Advisor : Adhatus Solichah A.,S.Kom., M.Sc.

ABSTRACT

Historical and figures of Indonesia, are proof of important events in our history. Every figure has different partners and relatives. One of the open encyclopedia platform is Wikipedia. The pages or subjects of a Wikipedia page has a direct association with DBpedia page, whereas DBpedia provides list of hyperlinks of related things of a Wikipedia subject as table rows, such as parents, partners, and children. But sometimes a Wikipedia page represents the data as paragraphs and DBpedia as table rows.

From a DBpedia page, we can get information of a person's family and relations. This Wikipedia hyperlink relation can be modelled as an ontology. The purpose of this thesis is to complete the family data of the Indonesia's historical figures and to represent them as a family tree.

The steps required to complete this thesis is first completing the figure' data using reasoning process, store the data on a triple store database, and to display the information in a family tree graph. To complete the family data of a person, Family Relationship Ontology by Robert Stevens is used and combined with the DBpedia page and reasoned using Pellet Reasoner. It is proven that this method generates facts that are unknown to DBpedia page. To store the data, Apache Jena-Fuseki can act as a triple store database. According to test results, this web

application is able to display family tree of a DBpedia subject and the relations are more complete than its DBpedia page. This thesis can help history scientist to determine the family tree of a historical figure. This thesis is also capable to educate people about Indonesia's historical figures and their relations.

Key words: Family tree, Indonesian history,, Ontology, Visualization

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

“Rancang Bangun Aplikasi Berbasis Web untuk Visualisasi Family Tree Tokoh Sejarah Indonesia Menggunakan Ontologi DBpedia dan Pellet Reasoning”

Tugas akhir ini dilakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Allah SWT atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Orang tua, saudara serta keluarga penulis yang tiada henti-hentinya memberikan semangat, perhatian dan doa selama perkuliahan penulis di Jurusan Teknik Informatika ini.
3. Ibu Nurul Fajrin A., S.Kom., M.Sc. selaku dosen pembimbing I yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
4. Ibu Adhatus Sholichah A., S.Kom., M.Sc. selaku dosen pembimbing II yang telah banyak memberikan arahan dan bantuan, waktu untuk berdiskusi serta ilmu-ilmu baru sehingga penulis dapat menyelesaikan tugas akhir ini.
5. Segenap dosen Departemen Informatika ITS yang telah memberikan ilmu dalam kuliah-kuliah saya.
6. Firda Rheinalia, S.Kom yang selalu memberikan semangat dan informasi terkait pengerjaan Tugas Akhir.

7. Sahabat-sahabat Rumah Perjuangan, Fatur, Illham, Ichsan, Huda, Bimo, Dias, Azka, Adam, Rio dan Djohan, serta Tegar, Naufal, Arya dan Adib.
8. Teman-teman admin laboratorium Manajemen Informasi yang memfasilitasi dan memberi semangat saat mengerjakan tugas akhir ini.
9. Teman-teman HMTTC 2016/2017 dan BEM FTIK 2016/2017 - 2017/2018.
10. Seluruh keluarga TC 2015 yang selalu menemani dan memberi semangat selama 4 tahun perkuliahan.
11. Serta semua pihak yang telah memberikan dukungan selama penulis menyelesaikan tugas akhir ini.

Saya mohon maaf apabila terdapat kekurangan dalam penulisan buku tugas akhir ini. Kritik dan saran saya harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Juni 2019

Penulis

DAFTAR ISI

| | |
|---|-------|
| HALAMAN JUDUL | iii |
| LEMBAR PENGESAHAN..... | vii |
| ABSTRAK | ix |
| ABSTRACT | xi |
| KATA PENGANTAR..... | xiii |
| DAFTAR ISI | xv |
| DAFTAR GAMBAR | xix |
| DAFTAR TABEL | xxi |
| DAFTAR KODE SUMBER | xxiii |
| 1 BAB I PENDAHULUAN | 1 |
| 1.1. Latar Belakang | 1 |
| 1.2. Rumusan Masalah | 2 |
| 1.3. Batasan Masalah..... | 2 |
| 1.4. Tujuan..... | 3 |
| 1.5. Metodologi | 3 |
| 1.6. Sistematika Penulisan..... | 4 |
| 2 BAB II DASAR TEORI..... | 7 |
| 2.1. Tokoh Sejarah Indonesia | 7 |
| 2.2. DBpedia..... | 7 |
| 2.3. Ontologi..... | 9 |
| 2.4. Semantic Web Rule Language (SWRL)..... | 14 |
| 2.5. Family Relationships Ontology | 15 |
| 2.6. Java..... | 17 |
| 2.7. Reasoning | 18 |
| 2.8. Pellet Reasoner | 19 |
| 2.9. Apache Jena..... | 21 |
| 2.10. SPARQL..... | 22 |
| 2.11. SPARQL Lib | 22 |
| 2.12. Apache Jena Fuseki | 24 |
| 2.13. PHP..... | 25 |
| 2.14. Visualisasi | 27 |
| 3 BAB III METODOLOGI PEMECAHAN MASALAH..... | 29 |

| | | |
|--------|--|----|
| 3.1. | Analisis Data | 30 |
| 3.1.1. | Analisis Data dari DBpedia..... | 31 |
| 3.2. | Ekstraksi Data Sebagai Model..... | 34 |
| 3.3. | Pembuatan Ontologi | 36 |
| 3.4. | Penggabungan Model Data dan Model Family Relationship Ontology | 41 |
| 3.5. | Reasoning pada Model Gabungan | 45 |
| 3.6. | Penampilan Data..... | 46 |
| 4 | BAB IV ANALISIS DAN PERANCANGAN SISTEM..... | 47 |
| 4.1. | Analisis | 47 |
| 4.1.1. | Cakupan Permasalahan..... | 47 |
| 4.1.2. | Deskripsi Umum Sistem..... | 47 |
| 4.1.3. | Spesifikasi Kebutuhan Perangkat Lunak..... | 50 |
| 4.1.4. | Aktor..... | 50 |
| 4.1.5. | Kasus Penggunaan..... | 51 |
| 4.2. | Perancangan Antarmuka Pengguna | 53 |
| 5 | BAB V IMPLEMENTASI | 55 |
| 5.1. | Implementasi Proses Ekstraksi, Penggabungan, dan Reasoning | 55 |
| 5.2. | Implementasi Antarmuka Pohon Keluarga..... | 59 |
| 5.2.1. | Fungsi Dropdown Select | 59 |
| 5.2.2. | Fungsi Get Family | 61 |
| 5.3. | Implementasi Antarmuka Pengguna..... | 81 |
| 5.3.1. | Implementasi Halaman Utama | 81 |
| 5.3.2. | Implementasi Halaman Pohon Keluarga | 82 |
| 6 | BAB VI PENGUJIAN DAN EVALUASI | 83 |
| 6.1. | Lingkungan Pengujian..... | 83 |
| 6.2. | Skenario Pengujian | 83 |
| 6.2.1. | Pengujian Reasoning | 84 |
| 6.2.2. | Pengujian Visualisasi | 84 |
| 6.3. | Hasil Pengujian..... | 85 |
| 6.3.1. | Pengujian Reasoning | 85 |
| 6.3.2. | Pengujian Visualisasi | 86 |
| 6.4. | Evaluasi Hasil Pengujian | 88 |
| 7 | BAB VII KESIMPULAN DAN SARAN | 91 |

| | | |
|----------------------|-----------------|-----|
| 7.1. | Kesimpulan..... | 91 |
| 7.2. | Saran..... | 91 |
| DAFTAR PUSTAKA..... | | 93 |
| LAMPIRAN..... | | 96 |
| BIODATA PENULIS..... | | 112 |

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 2.1 Contoh Halaman DBpedia..... | 8 |
| Gambar 2.2 Data Elizabeth II dalam berbagai versi DBpedia | 9 |
| Gambar 2.3 <i>Class Hierarchy</i> | 10 |
| Gambar 2.4 <i>Property</i> | 10 |
| Gambar 2.5 Class, Property dan Instance..... | 13 |
| Gambar 2.6 Ontologi FamilyTree Keluarga Robert Stevens | 17 |
| Gambar 2.7 Arsitektur Pellet Reasoner | 20 |
| Gambar 2.8 Contoh SPARQL Query | 22 |
| Gambar 2.9 Contoh Penggunaan SPARQL Lib..... | 23 |
| Gambar 2.10 Database Triple Store Apache Jena Fuseki | 24 |
| Gambar 2.11 Daftar API Apache Jena Fuseki..... | 25 |
| Gambar 3.1 Flowchart pengembangan sistem..... | 29 |
| Gambar 3.2 Halaman DBpedia tentang property keluarga Raden Wijaya | 31 |
| Gambar 3.3 Data DBpedia Haryati | 33 |
| Gambar 3.4 Data DBpedia Prince Philip..... | 34 |
| Gambar 3.5 Halaman DBpedia Fatmawati..... | 35 |
| Gambar 3.6 Representasi data keluarga Fatmawati | 36 |
| Gambar 3.7 Hirarki Class | 37 |
| Gambar 3.8 Hirarki Data property | 37 |
| Gambar 3.9 Hirarki Object property | 38 |
| Gambar 3.10 Ontologi Hayam Wuruk (Individuals)..... | 42 |
| Gambar 3.11 Ontologi Hayam Wuruk (Object Properties) | 42 |
| Gambar 3.12 Ontologi Family Tree (Individuals)..... | 43 |
| Gambar 3.13 Ontologi Family Tree (Object Properties) | 43 |
| Gambar 3.14 Ontologi union (Individuals) | 44 |
| Gambar 3.15 Ontologi union (Object Properties) | 44 |
| Gambar 3.16 Individu Susilo Bambang Yudhoyono sebelum reasoning | 45 |
| Gambar 3.17 Individu Susilo Bambang Yudhoyono setelah reasoning | 45 |

Gambar 3.18 Silsilah keluarga kerajaan Singasari dan Majapahit [11]46

Gambar 4.1 Arsitektur Sistem48

Gambar 4.2 Diagram Kasus Penggunaan Sistem51

Gambar 4.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh 53

Gambar 4.4 Antarmuka Halaman Utama Family Tree App.....54

Gambar 4.5 Antarmuka Halaman Pohon Keluarga Family Tree App54

Gambar 4.1 Implementasi Antarmuka Halaman Utama.....81

Gambar 4.2 Implementasi Antarmuka Halaman Pohon Keluarga82

DAFTAR TABEL

| | |
|--|----|
| Tabel 2.1 Karakteristik <i>Property</i> | 11 |
| Tabel 2.2 Deskripsi <i>Property</i> | 12 |
| Tabel 2.3 Komponen SWRL | 14 |
| Tabel 2.4 Method PHP | 26 |
| Tabel 3.1 Daftar property yang dibutuhkan | 32 |
| Tabel 3.2 Daftar property yang dihasilkan | 32 |
| Tabel 3.3 Namespace DBpedia | 32 |
| Tabel 3.4 Pemetaan property equivalent | 38 |
| Tabel 3.5 Daftar Class | 39 |
| Tabel 3.6 Daftar Object Property | 39 |
| Tabel 3.7 Daftar Data Property | 41 |
| Tabel 3.8 Daftar Kebutuhan Fungsional Perangkat Lunak | 50 |
| Tabel 3.9 Daftar Kode Diagram Kasus Penggunaan | 51 |
| Tabel 3.10 Spesifikasi Kasus Penggunaan Melihat Informasi Tokoh | 52 |
| Tabel 3.11 Spesifikasi Atribut Rancangan Antarmuka Halaman Family Tree App | 54 |
| Tabel 5.1 Pengujian Reasoning property hasSpouse | 85 |
| Tabel 5.2 Pengujian Reasoning property hasChild | 85 |
| Tabel 5.3 Pengujian Reasoning property hasParent | 86 |
| Tabel 5.4 Pengujian Visualisasi tanpa anak | 86 |
| Tabel 5.5 Pengujian Visualisasi memiliki anak | 87 |
| Tabel 5.6 Pengujian Visualisasi memiliki cucu | 87 |
| Tabel 5.7 Pengujian Visualisasi memiliki cicit | 87 |
| Tabel 5.8 Pengujian Visualisasi memiliki banyak pasangan | 88 |
| Tabel 5.9 Pengujian Visualisasi tidak memiliki identitas | 88 |
| Tabel 5.10 Evaluasi Pengujian | 88 |

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

| | |
|---|----|
| Kode Sumber 3.1 Kode Java untuk memodelkan data Fatmawati ke dalam modelActor | 35 |
| Kode Sumber 3.2 Kode sumber untuk menggabungkan dua model | 41 |
| Kode Sumber 5.1 Implementasi proses inisialisasi variabel statis | 56 |
| Kode Sumber 5.2 Implementasi inisialisasi model Instance dan famonto..... | 56 |
| Kode Sumber 5.3 Implementasi ekstraksi file RDF tokoh..... | 58 |
| Kode Sumber 5.4 Implementasi penggabungan model..... | 58 |
| Kode Sumber 5.5 Implementasi proses reasoning | 58 |
| Kode Sumber 5.6 Implementasi print hasil reasoning sebagai file RDF | 59 |
| Kode Sumber 5.7 Kode Sumber SPARQL untuk mengambil value bertipe Person dan Fungsi Dropdown Select..... | 61 |
| Kode Sumber 5.8 Fungsi Get name..... | 62 |
| Kode Sumber 5.9 Fungsi Get father..... | 64 |
| Kode Sumber 5.10 Fungsi Get mother..... | 66 |
| Kode Sumber 5.11 Fungsi Get sibling | 68 |
| Kode Sumber 5.12 Fungsi Get spouse | 70 |
| Kode Sumber 5.13 Fungsi Get child | 72 |
| Kode Sumber 5.14 Fungsi Get child in law | 74 |
| Kode Sumber 5.15 Fungsi Get grand child | 76 |
| Kode Sumber 5.16 get grand child in law | 78 |
| Kode Sumber 5.17 Get great grand child | 81 |

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan, metodologi dan sistematika penulisan yang digunakan dalam pembuatan buku tugas akhir ini.

1.1. Latar Belakang

Tokoh bersejarah adalah seseorang yang namanya dikenang karena jasanya. Tokoh bersejarah menjadi bukti dari adanya suatu kejadian penting di masa lalu. Setiap tokoh memiliki kisah serta rekan hidup yang berbeda. Rekan hidup dapat berarti keluarga, sahabat, teman, dan sebagainya. Berdasarkan pada *history* rekan hidup, tokoh yang satu dengan tokoh yang lain memiliki hubungan terkait sehingga relasi antar tokoh tersebut dapat diketahui. Selain itu, hubungan tersebut juga dapat menentukan kejadian apa yang pernah terlibat di antara mereka.

Keterkaitan antar satu tokoh dengan tokoh yang lain dapat digambarkan dengan ontologi. Ontologi adalah spesifikasi formal dari konsep-konsep yang saling berhubungan. Ontologi mendefinisikan *class*, *property*, *instance*, dan hubungan sebuah individu dengan individu lain untuk domain tertentu. Dengan ontologi, uraian dari seorang tokoh dapat didefinisikan. Pendefinisian tersebut berguna untuk mencari hubungan antar tokoh. Dalam *cultural heritage*, *actor* adalah salah satu domain yang dapat diontologikan. Ruang lingkup *actor* mencakup *person*, *group*, dan *organization*. Sedangkan tokoh bersejarah dan pahlawan termasuk dalam agen *person*.

Dalam perkembangan teknologi, pengetahuan tentang tokoh bersejarah dan pahlawan nasional tidak hanya terhimpun di dalam buku-buku sejarah. Banyak situs daring yang menyediakan informasi tentang tokoh bersejarah dan pahlawan nasional, seperti Wikipedia, DBpedia, Everything2, Quora, dan lain-lain. Akan tetapi dalam situs-situs tersebut, mayoritas informasi yang

diberikan masih berupa paragraf-paragraf teks atau tabel, sedangkan otak manusia dapat memproses informasi visual 60.000 kali lebih cepat daripada informasi teks [1]. Pengerjaan tugas akhir ini akan mengembangkan ontologi data keluarga tokoh sejarah Indonesia yang sudah ada dan melengkapinya dengan mengkombinasikan *class* dan *property* yang dimilikinya dan ditampilkan dalam sebuah situs web untuk memudahkan pemahaman terkait tokoh sejarah Indonesia dan relasinya.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana menentukan data property yang nantinya dapat digunakan untuk mendefinisikan relasi dalam domain tokoh sejarah Indonesia?
2. Bagaimana memodelkan proses reasoning untuk melengkapi relasi tokoh sejarah pada DBpedia?
3. Bagaimana membuat aplikasi untuk menampilkan visualisasi pohon keluarga tokoh sejarah Indonesia?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data yang digunakan adalah tokoh sejarah Indonesia dari DBpedia.
2. Aplikasi tidak dapat menangani *person* yang tidak memiliki halaman DBpedia.
3. *Person* yang tidak memiliki atribut nama atau label tidak akan ditampilkan
4. Batas relasi adalah ayah, ibu, saudara, istri, anak, menantu, cucu, pasangan cucu, dan cicit.
5. Aplikasi sangat bergantung pada kelengkapan atribut data DBpedia.
6. *Reasoner* yang digunakan adalah Pellet.

7. Aplikasi yang dibuat tidak menyediakan *form* untuk pengelolaan data (tambah, ubah, hapus).

1.4. Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah membuat aplikasi web yang dapat menampilkan pohon keluarga dari tokoh sejarah Indonesia secara visual untuk membantu dan mempermudah pencarian relasi dari tokoh sejarah Indonesia.

1.5. Metodologi

Ada beberapa tahapan dalam pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. Studi Literatur

Pada tahap ini, akan dilakukan studi mengenai sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai informasi yang melekat pada tokoh bersejarah, DBpedia, ontologi, *reasoning*, visualisasi, *Family Relationships Ontology*, SPARQL, SPARQL Lib, Apache Jena, Apache Jena Fuseki, SWRL (*Semantic Web Rule Language*), Java, PHP, dan Pellet Reasoner.

2. Implementasi

Pada tahap ini, akan dilakukan implementasi berdasarkan rancangan yang dibuat dalam tahap sebelumnya, yaitu pelengkapan data yang dilakukan dengan *tools* Protege 5.2.0 dengan ekstensi *Web Ontology Language* (OWL). Sedangkan aplikasi sederhana untuk menampilkan hasil pencarian relasi dibangun dengan bahasa PHP menggunakan *tools* PhpStorm.

3. Pengujian dan evaluasi

Tahap ini dilakukan dengan uji coba aplikasi untuk mencari dan mengetahui relasi keterkaitan antar tokoh serta mengadakan perbaikan jika ada kekurangan. Pengujian ontologi akan dilakukan dengan menggunakan Pellet reasoner. Selain itu, pengujian juga dilakukan dengan membandingkan data hasil uji coba yang ditampilkan pada aplikasi dengan data aslinya yang bersumber dari DBpedia.

Evaluasi dilakukan untuk mengetahui karakteristik dan kecenderungan jalannya sebuah program atas sebuah rangkaian *rule* yang diberikan.

4. Penyusunan buku tugas akhir

Tahap ini merupakan tahap penyusunan laporan berupa buku sebagai dokumentasi pengerjaan tugas akhir yang mencakup seluruh dasar teori, desain, implementasi serta hasil pengujian yang telah dilakukan.

1.6. Sistematika Penulisan

Sistematika penulisan dibuat bertujuan untuk mendapatkan gambaran umum dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan rancang bangun aplikasi berbasis web untuk visualisasi pohon keluarga tokoh sejarah Indonesia ini.

Bab III Metode Pemecahan Masalah

Bab ini membahas mengenai metode yang digunakan untuk memecahkan masalah yang dipaparkan pada rumusan permasalahan.

Bab IV Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada perangkat lunak.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak dan implementasi fitur-fitur penunjang.

Bab VI Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian objektif untuk mengetahui kecocokan data dan kekayaan data.

Bab VII Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

(Halaman ini sengaja dikosongkan)

BAB II

DASAR TEORI

Bab ini akan membahas mengenai dasar teori dan literatur yang menjadi dasar pengerjaan tugas akhir ini.

2.1. Tokoh Sejarah Indonesia

Pahlawan adalah gelar tertinggi di Indonesia. Gelar ini diberikan oleh pemerintah Republik Indonesia untuk seseorang yang menunjukkan perilaku atau tindakan yang dianggap ‘heroik’, yang didefinisikan sebagai “perbuatan nyata yang dapat diingat dan dicontoh oleh masyarakat untuk selamanya” atau “pelayanan luar biasa untuk memajukan kepentingan masyarakat atau negara”. Tokoh sejarah seringkali dikaitkan dengan gelar pahlawan nasional. Padahal belum tentu tokoh sejarah adalah pahlawan nasional.

Tokoh sejarah adalah seseorang yang diingat namanya atas jasanya. Setiap tokoh bersejarah memiliki pengalaman hidup yang berbeda-beda. Sering kita temui dalam biografi seorang tokoh bersejarah bahwa mereka masih memiliki relasi dengan tokoh sejarah yang lain. Biografi adalah deskripsi detail dari kehidupan seseorang dari lahir sampai meninggal dunia. Setiap jasa atau karya yang dihasilkan setiap tokoh sejarah dicatat dalam biografinya.

Setiap tokoh sejarah memiliki perjalanan hidup dan teman hidup masing-masing. Untuk tugas akhir ini, data yang digunakan adalah data tokoh sejarah Indonesia yang diambil dari laman ensiklopedia bebas seperti Wikipedia. Data tokoh ini akan diunduh dan dimodelkan dalam aplikasi Jena untuk menjalani proses *reasoning* agar tercipta fakta-fakta baru.

2.2. DBpedia

DBpedia adalah situs web yang bergerak untuk mengekstrak data-data dari halaman Wikipedia dan menampilkannya sebagai informasi yang sudah terstruktur. Data

dari sebuah halaman DBpedia dapat kita ambil dengan format yang kita inginkan seperti CSV, RDF, N-Triples, JSON, dan lain-lain.. Data di DBpedia masih berupa tabel property dan value. Gambar 2.1 adalah contoh sebuah halaman DBpedia.

Content-Length: 67902

About: [Raden Wijaya](#)

An Entity of Type : [Thing](#), from Named Graph : <http://id.dbpedia.org>, within Data Space : id.dbpedia.org

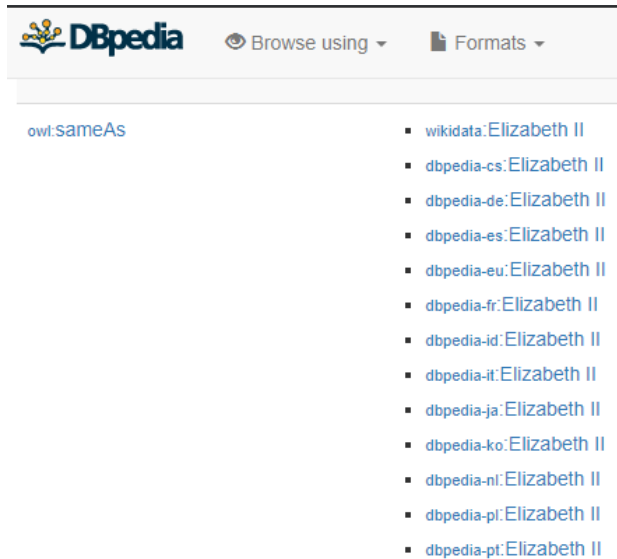


Kertarajasa Jayawardhana atau disebut juga Raden Wijaya adalah pendiri Kerajaan Majapahit sekaligus raja pertama Majapahit yang memerintah pada tahun 1293-1309, bergelar Prabu Kertarajasa Jayawardana, atau lengkapnya Nararya Sanggramawijaya Sri Maharaja Kertarajasa Jayawardhana.

| Property | Value |
|--|---|
| dbpedia-owl:abstract | <ul style="list-style-type: none">Kertarajasa Jayawardhana atau disebut juga Raden Wijaya adalah pendiri Kerajaan Majapahit sekaligus raja pertama Majapahit yang memerintah pada tahun 1293-1309, bergelar Prabu Kertarajasa Jayawardana, atau lengkapnya Nararya Sanggramawijaya Sri Maharaja Kertarajasa Jayawardhana. |
| dbpedia-owl:thumbnail | <ul style="list-style-type: none">http://upload.wikimedia.org/wikipedia/commons/thumb/b/b6/Harihara_Majapahit_1.JPG/200px-Harihara_Majapahit_1.JPG |
| dbpedia-owl:wikiPageID | <ul style="list-style-type: none">11640 (xsd:integer) |
| dbpedia-owl:wikiPageRevisionID | <ul style="list-style-type: none">6634882 (xsd:integer) |
| dbpedia-owl:wikiPageWikiLink | <ul style="list-style-type: none">dbpedia-id: Babad_Tanah_Jawidbpedia-id: Ciung_Manaradbpedia-id: Jawadbpedia-id: Suku_Jawadbpedia-id: Jayakatwangdbpedia-id: Jayanagaradbpedia-id: Kerajaan_Kadiridbpedia-id: Ken_Arokdbpedia-id: Kartanagaradbpedia-id: Kublai_Khandbpedia-id: Pulau_Maduradbpedia-id: Majapahitdbpedia-id: Penguasa_monarkidbpedia-id: 12_Novemberdbpedia-id: Elispedia_Pamayudbpedia-id: Pararatondbpedia-id: Pathidbpedia-id: Pranaparamitadbpedia-id: Wangsa_Rajasadbpedia-id: Siwadbpedia-id: Kerajaan_Singhasaridbpedia-id: Suku_Sunda |

Gambar 2.1 Contoh Halaman DBpedia

Data pada gambar diatas akan diunduh oleh aplikasi Jena dan dimodelkan untuk menjalani proses *reasoning*. Selain halaman DBpedia diatas (DBpedia Indonesia), terdapat banyak versi DBpedia yang menyediakan data dan property yang bermacam-macam dan belum tentu versi yang lain memiliki property yang sama. Contohnya adalah data Ratu Elizabeth II pada Gambar 2.2 yang tidak hanya ada di DBpedia berbahasa Inggris saja, namun juga ada dalam situs DBpedia Jerman, Spanyol, Italia, Prancis, dan Indonesia.



Gambar 2.2 Data Elizabeth II dalam berbagai versi DBpedia

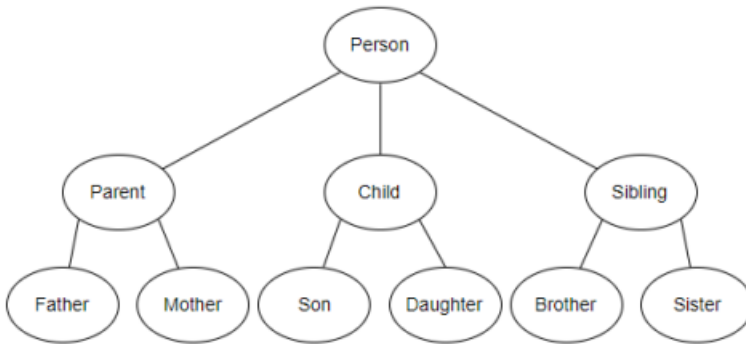
2.3. Ontologi

Istilah ontologi berasal dari kajian ilmu filsafat yang kemudian diresap oleh ilmu komputer. Definisi ontologi adalah sebagai studi tentang konsep yang secara sistematis menjelaskan tentang keberadaan segala sesuatu yang konkret. Terdapat tiga komponen utama dari ontologi, yaitu *class*, *property*, dan *instance* [2]. Berikut adalah penjelasan mengenai komponen-komponen tersebut:

a) *Class*

Class menspesifikasikan *property* yang sama dari beberapa *instance* dan berbentuk hierarki. Selain itu, *class* juga mencakup *superclass* dan *subclass*. *Subclass* merupakan turunan dari *superclass*-nya yang lebih detail. Setiap *subclass* mewarisi fungsi dan atribut dari leluhurnya. *Subclass* mungkin memiliki fungsi dan atribut tambahan sendiri (yang tidak dimiliki oleh

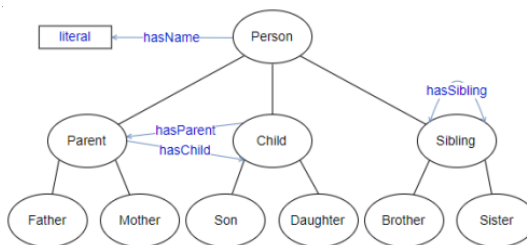
leluhurnya). Contohnya adalah *class* Child memiliki *subclass* Son dan Daughter, serta memiliki *superclass* Person. Hubungan antara *subclass* dan *superclass* digambarkan dengan class hierarki yang dicontohkan pada Gambar 2.3.



Gambar 2.3 Class Hierarchy

b) Property

Property adalah atribut-atribut yang dimiliki oleh suatu *Class*. *Property* juga menghubungkan member dari suatu kelas ke member kelas lainnya. Contoh *property* adalah seperti yang terdapat pada Gambar 2.4.



Gambar 2.4 Property

Property memiliki atribut tersendiri yang menjadikan suatu *property* mempunyai karakteristik tersendiri.

Tabel 2.1 Karakteristik *Property*

| Karakteristik <i>Property</i> | Keterangan |
|-------------------------------|---|
| Functional | Hanya memiliki satu <i>value range</i> . Contoh : Megawati memiliki satu <i>range</i> ibu kandung |
| Inverse functional | Hanya memiliki satu <i>value domain</i> . Contoh : <i>Domain</i> Ir. Soekarno memiliki label “Soekarno”. |
| Transitive | Memiliki hubungan berantai. Misalnya, Soekarno memiliki relasi dengan Fatmawati, Fatmawati memiliki relasi dengan Megawati, maka Soekarno memiliki relasi dengan Megawati. |
| Symmetric | Memiliki hubungan dua arah. Contohnya adalah Soekarno memiliki pasangan Fatmawati, maka Fatmawati memiliki pasangan Soekarno. |
| Asymmetric | Memiliki hubungan satu arah. Contohnya adalah Soekarno memiliki anak Megawati, tetapi Megawati tidak memiliki anak Soekarno. |
| Reflexive | Menegaskan bahwa suatu individu bisa mempunyai <i>property</i> yang merujuk pada dirinya sendiri. Misalnya individu Soekarno memiliki Presiden, yaitu dirinya sendiri. |
| Irreflexive | Menegaskan bahwa suatu individu mempunyai <i>property</i> yang tidak bisa merujuk pada dirinya sendiri. Misalnya individu Soekarno memiliki anak, maka <i>propertinya</i> adalah <i>irreflexive</i> . |

Selain itu, ada juga deskripsi *property* yang menjelaskan hubungan antara property satu dengan yang lain.

Tabel 2.2 Deskripsi *Property*

| Deskripsi <i>Property</i> | Keterangan |
|---------------------------|---|
| Equivalent to | Satu property dengan yang lain memiliki identitas yang sama dengan karakteristik yang sama. Misal, property dbp:issue dan dbo:child adalah equivalent karena sama-sama menjelaskan hubungan kepemilikan keturunan. |
| SubProperty of | Property yang dipilih adalah bagian dari property lain. Semisal property hasSon dan hasDaughter adalah SubProperty of hasChild |
| Inverse of | Property yang dipilih bersifat berlawanan. Contohnya adalah property hasChild dan hasParent. |
| Domain (Intersection) | Property memiliki domain tertentu, misalnya property hasChild hanya dimiliki oleh domain Parent. |
| Range (Intersection) | Property memiliki range tertentu, misalnya property hasWife hanya memiliki range Female. |
| Disjoint with | Property yang terpilih tidak akan berhubungan dengan property lain. Jika property hasParent bersifat Disjoint with hasChild, maka jika Megawati memiliki Parent Soekarno, maka Soekarno tidak memiliki Parent Megawati. |
| SuperProperty of (chain) | Property terpilih adalah SuperProperty dari dua atau lebih property yang lain dengan ditandai “o”. Misalnya property hasGrandChild adalah SuperProperty dari ‘hasChild o hasChild’. |

Karakteristik dan deskripsi property sangat penting dalam pengerjaan tugas akhir ini karena diperlukan beberapa deskripsi seperti *SuperProperty of* untuk mempermudah mencari relasi dengan data yang ada, dan *Equivalent to* untuk menyatukan properti-property yang berbeda namun fungsinya sama.

c) *Instance*

Instance merupakan individual dari sebuah class atau biasa disebut dengan member dari *class*. Contoh hubungan dari *Class*, *Property* dan *Instance* ditunjukkan oleh Gambar 2.5

Class definition statements :

- **Parent** isA Class
- **Father** isA Class
- **Mother** subClassOf **Parent**
- **Child** isA Class

Property definition statements :

- **isParentOf** isA Property
 - isParentOf domain **Parent**
 - isParentOf range **Child**

Instance statements :

- **DaveSmith** isA **Father**
- **AnnSmith** isA **Child**
- **AnnSmith** isChildOf **DaveSmith**

Gambar 2.5 Class, Property dan Instance

Selain 3 komponen penting yang telah dijelaskan di atas, terdapat beberapa istilah lain yang perlu dipahami dalam konteks ontologi antara lain *domain* (member dari suatu kelas yang dapat menjadi subjek dari *property* yang diberikan), *range* (member dari suatu kelas yang dapat menjadi objek dari *property* yang diberikan), *constraint* dan *rule* (menentukan batasan dan istilah-istilah teknis untuk mendukung *reasoning*), dan *relationship* (mekanisme inferensi untuk menggenerasi pengetahuan baru).

Dalam *semantic modelling*, ontologi dapat direpresentasikan dengan berbagai bahasa yang sudah memiliki standar seperti RDF, RDFS, atau OWL. Secara umum, kegunaan ontologi adalah sebagai *controlled vocabulary*, *semantic interoperability*, *knowledge sharing*, dan *reuse* [3].

Kelebihan *knowledge base* dari basis data relasional adalah level abstraksinya. Basis data relasional digunakan untuk membangun struktur penyimpanan data, sedangkan ontologi digunakan untuk menganalisa banyak *resource* dan menyimpulkan fakta dan pengetahuan baru dari relasi dan *rules* yang ada melalui proses *reasoning*. Jika basis data relasional tidak ada data dalam *record*, maka tidak ada hasil yang didapatkan, namun dengan melakukan proses *reasoning* pada *knowledge base*, dapat diketahui hasilnya (*inferred value*).

Dalam tugas akhir ini, ontologi digunakan untuk pemodelan dan penyimpanan pengetahuan tentang data-data yang diunduh dari DBpedia, serta dalam pengaturan karakteristik dan deskripsi properti.

2.4. Semantic Web Rule Language (SWRL)

SWRL merupakan bahasa berbentuk *unary* dan *binary rule statement* yang menjadi bagian dari OWL. Pada dasarnya, *rule* terdiri dari *antecedent* dan *consequent*, keduanya terdiri dari pasangan-pasangan atom. Jika *antecedent* bernilai benar, maka *consequent* juga akan bernilai benar [4]. Pada Tabel 2.3 berikut akan dijabarkan bentuk-bentuk atom yang didefinisikan.

Tabel 2.3 Komponen SWRL

| Atom | Deskripsi |
|------|---|
| C(x) | C adalah deklarasi <i>class</i> (nama <i>class</i>) dan x adalah nama individual atau variabel |
| D(y) | D adalah deklarasi <i>data range</i> dan y adalah variabel atau <i>data value</i> |

| Atom | Deskripsi |
|------------------------------|---|
| $P(x, y)$ | P adalah data atau <i>object property</i> , x dan y adalah variabel atau OWL individual. y adalah sebuah individual jika P adalah <i>object property</i> , sedangkan y adalah sebuah <i>data value</i> jika P adalah <i>data property</i> . |
| $\text{sameAs}(x, y)$ | x dan y adalah variabel atau individual yang menyatakan bahwa keduanya merupakan individu yang sama |
| $\text{differentFrom}(x, y)$ | x dan y adalah variabel atau individual yang menyatakan bahwa keduanya merupakan individu yang berbeda |

2.5. Family Relationships Ontology

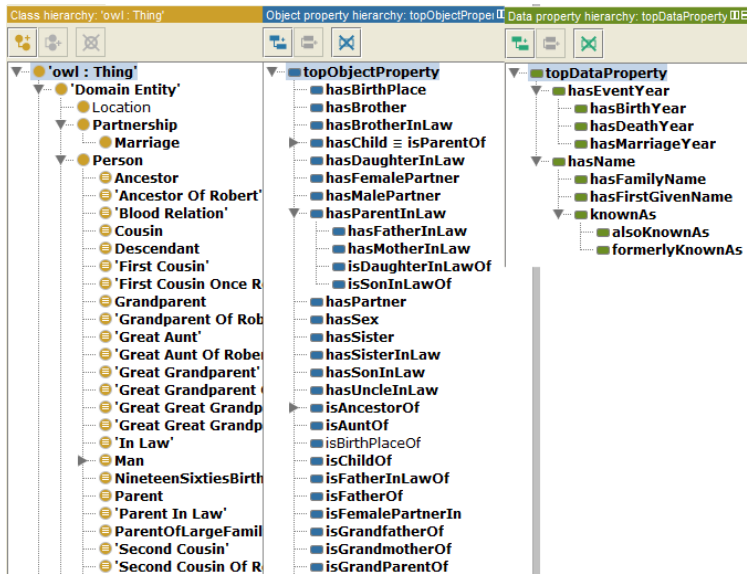
Family relationship umumnya digambarkan dengan terstruktur melalui silsilah keluarga. Manusia membutuhkan informasi tentang silsilah keluarganya untuk berbagai hal, diantaranya adalah untuk mempererat ikatan batin antar anggota keluarga, mempermudah keturunannya dalam menelusuri asal usul keluarganya, menentukan pewarisan, perkawinan, dan lain sebagainya. Silsilah keluarga adalah bagan yang menampilkan struktur keluarga dalam bentuk pohon. Silsilah keluarga menyimpan informasi yang mendeskripsikan relasi antar anggota keluarga secara kompleks [5].

Keluarga memiliki struktur garis keturunan yang panjang. Jika relasi keturunan dicari secara manual, maka dibutuhkan waktu dan analisis yang lama. Belum tentu setiap anggota keluarga mengenal kerabatnya, karena pada umumnya hanya satu atau dua orang yang mengetahui detail keluarga. Semakin bertambahnya pengetahuan membuat hubungan dalam sebuah keluarga dapat diketahui dengan mudah melalui *Family Relationships Ontology*. Ontologi ini memiliki beberapa kelebihan, diantaranya adalah dapat diketahuinya keakraban, relasi, pewarisan, *domain*, *range*,

constraint, dan kesimpulan logis dalam sebuah keluarga secara praktis.

Terdapat banyak ontologi yang telah dibangun menggunakan domain keluarga, salah satunya adalah ontologi yang digunakan pada pengerjaan tugas akhir ini, yaitu FamilyTree. Ontologi tersebut didapatkan dari portal The University of Manchester. Ontologi FamilyTree memiliki URI <http://www.co-ode.org/roberts/family-tree.owl> [6]. Ontologi tersebut adalah sebuah ontologi sederhana dengan domain hubungan keluarga yang mendeskripsikan keluarga Robert Stevens. FamilyTree merupakan ontologi yang kompleks dan lengkap. Pembangunan ontologi tersebut dimaksudkan untuk menghasilkan suatu ontologi yang meminimalkan *relationships* dan memaksimalkan *inference*. Oleh karena itu, ontologi ini banyak menggunakan *role chain*, *nominal*, dan *properties hierarchy*.

Cuplikan kelas, properti, dan individu yang terdapat dalam ontologi tersebut dapat dilihat pada Gambar 2.6. Dari Gambar 2.6 dapat disimpulkan bahwa ontologi milik Robert Stevens adalah salah satu yang paling lengkap. Akan tetapi dalam pengerjaan tugas akhir ini, tidak semua *property* dan *class* dari ontologi tersebut hanya akan dipakai relasi yang umum, seperti *hasChild*, *hasParent*, *hasGrandchild*, *hasSpouse*, selain itu akan dihapus. Dan karena *instance* atau *individual* di tugas akhir ini adalah keluarga tokoh sejarah Indonesia, maka *instance* di ontologi ini dihapus.



Gambar 2.6 Ontologi FamilyTree Keluarga Robert Stevens

2.6. Java

Java adalah bahasa pemrograman yang dikembangkan oleh James Gosling di Sun Microsystem yang sekarang sudah dibeli oleh Oracle. Java adalah salah satu bahasa pemrograman yang berbasis objek dan didesain untuk bisa bekerja dalam semua platform, yang artinya ketika program Java dikompilasi, maka program tersebut bisa berjalan di semua platform yang mendukung aplikasi Java [7], termasuk perangkat ponsel pintar Android. Penulisan kode atau *syntax* Java hampir mirip seperti C dan C++. Menurut situs *Version Control* terkenal GitHub, bahasa Java adalah bahasa yang paling populer dengan sembilan juta pengguna [8], khususnya untuk aplikasi web berbasis *client-server*.

Selain itu, Java menyediakan *virtual machine* yang memungkinkan komputer untuk menjalankan program Java dan program-program yang ditulis dengan bahasa lain yang

terkompilasi dalam Java *bytecode*. Contoh bahasa JVM adalah Scala dan Kotlin.

Dalam kasus tugas akhir ini, program Java digunakan untuk mempermudah proses ekstraksi data berbasis RDF dari situs DBpedia, memodelkan data tersebut dan mengeluarkan output file RDF yang sudah dilakukan *reasoning*.

2.7. Reasoning

Reasoning atau proses penalaran adalah kemampuan untuk menjadikan suatu hal dapat diterima oleh akal, membangun dan menerima fakta, serta menerapkan logika berdasarkan informasi yang ada [9]. Proses penalaran kerap dipakai dalam aktivitas kehidupan manusia dan menghasilkan konsep matematika, ilmu pengetahuan alam, seni dan bahasa, dan menjadi salah satu kemampuan unik yang hanya dimiliki oleh manusia.

Terdapat beberapa macam jenis proses *logical reasoning*, seperti *inductive reasoning*, *deductive reasoning*, dan *abductive reasoning*.

a) Inductive Reasoning

Inductive reasoning adalah salah satu bentuk penalaran yang menghasilkan statemen berdasarkan pengamatan sebelumnya. Contoh statemen hasil dari *inductive reasoning* adalah :

Premis: Matahari selalu terbit dari timur.

Kesimpulan: Matahari akan terbit dari timur besok.

b) Deductive Reasoning

Deductive reasoning adalah bentuk penalaran logis yang mengambil kesimpulan dari premis-premis yang ada [10]. Beberapa konsep *deductive reasoning* adalah modus *Ponens*, modus *Tollens*, dan hukum *Syllogism*. Contoh *syllogism* sebagai berikut :

Premis 1: Soekarno memiliki anak Megawati.

Premis 2: Megawati memiliki anak Puan Maharani.

Kesimpulan: Soekarno memiliki cucu Puan Maharani.

c) *Abductive Reasoning*

Abductive reasoning adalah bentuk penalaran yang dimulai dengan pengamatan atau pengalaman dan menghasilkan kesimpulan yang paling sederhana dan paling mungkin kebenarannya. *Abductive reasoning* seringkali dipakai dalam kehidupan sehari-hari. Contoh *abductive reasoning* adalah :

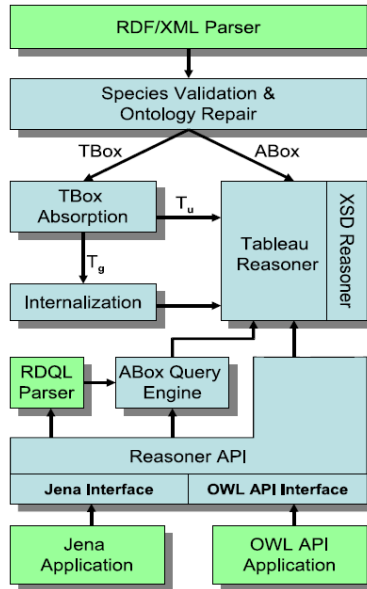
Ibu mendengar bayi menangis, dan bau tidak enak. Lalu ibu tersebut menyimpulkan secara *abductive* bahwa bayi tersebut menangis dan perlu mengganti popok.

Dalam konteks tugas akhir ini, jenis *reasoning* yang digunakan adalah *deductive reasoning* dikarenakan proses penalaran *deductive* adalah satu-satunya penalaran yang menggunakan fakta sebagai argumen untuk menghasilkan fakta baru. *Reasoning* juga menjadi kelebihan tersendiri dalam suatu *knowledge base* dibanding basis data relasional. Pada basis data relasional, jika tidak ada *record* atau data, maka saat dilakukan *query*, tidak akan ada hasilnya. Namun pada *knowledge base*, tidak masalah jika *record* atau data belum ada, setelah *knowledge base* tersebut melakukan proses *reasoning*, data dapat ditemukan dengan *query*.

2.8. Pellet Reasoner

Pellet Reasoner adalah salah satu perangkat lunak *reasoner* berbasis Java yang bersifat *open source* dan menggunakan *library* OWL API. Implementasi OWL *reasoner* yang sudah ada didasarkan pada beberapa pendekatan. Reasoner deskripsi logika (seperti Pellet dan RacerPro) menggunakan implementasi algoritma tableaux. Penggunaan algoritma tersebut memanfaatkan penelitian yang telah dilakukan untuk kasus algoritma deskripsi logika pengetahuan berdasar pada formalitas OWL [11]. Pellet didasarkan pada algoritma tableaux yang dikembangkan untuk

mengekspresikan *Description Logics*. Pellet mendukung semua konstruksi OWL DL termasuk `owl:oneOf` dan `owl:hasValue`. Saat ini, belum ada algoritma lengkap yang *decidable* dan efektif untuk semua OWL DL (khususnya, penanganan *inverse properties* dan *cardinality restrictions*).



Gambar 2.7 Arsitektur Pellet Reasoner

Pellet mengkombinasikan algoritma yang lengkap sebagai reasoner, yaitu OWL DL tanpa *nominals* (SHIN (D)) dan OWL DL tanpa *inverse properties* (SHON (D)). Algoritma ini dikombinasikan untuk mendapatkan penalaran yang lengkap dan berkaitan dengan semua DL. Pellet telah terbukti praktis berguna dalam berbagai pekerjaan saat ini. Gambar 2.7 menunjukkan komponen utama Pellet *reasoner*.

Ontologi OWL diparsing ke dalam RDF dengan pola *triple* (Sintaksis RDF / XML, N3 dan N-Triple yang mendukung). Pellet memvalidasi jenis dari ontologi dimana *triple RDF* dikonversi

menjadi pernyataan dan *axiom* berbasis pengetahuan. Jika level ontologi adalah OWL Full karena hilangnya tipe pola *triple*, maka Pellet menggunakan beberapa heuristik untuk memperbaiki ontologi. Misalnya *untyped resource* yang telah digunakan dalam predikat *position* dalam sebuah pola *triple* akan disimpulkan menjadi *datatype property* jika *triple* literal dalam posisi objek.

Pellet menyimpan *axiom* tentang kelas-kelas dalam komponen TBox dan menyimpan pernyataan tentang individu dalam komponen abox. Partisi TBox, adalah tempat penyerapan dan optimasi berlangsung. Tableau reasoner menggunakan *rule* tableau standar dan mencakup berbagai optimasi standar seperti keterkaitan yang diarahkan pada *backjumping*, percabangan semantik dan strategi pemblokiran awal. *Datatype reasoning* untuk *built-in* dan pengambilan XML Schema *datatypes* primitif didukung dalam *reasoner* ini. Pellet diimplementasikan dalam Java dan berada di bawah lisensi MIT [12].

Dalam tugas akhir ini, Pellet Reasoner dimasukkan sebagai plugin Java dalam aplikasi Jena, dan bertugas sebagai *reasoner*. *Reasoner* ini akan menerima input model RDF, dan akan mengeluarkan model RDF baru dengan fakta-fakta baru.

2.9. Apache Jena

Apache Jena adalah *plugin open source* berbasis Java yang digunakan untuk membangun aplikasi *Linked Data* dan *Semantic Web Framework* ini terdiri dari beberapa API yang berinteraksi secara bersamaan untuk memproses data dengan format RDF. Aplikasi yang memiliki *plugin* Apache Jena sanggup membuat model, memodelkan data dari API triple store, menggabungkan model, hingga *reasoning*. Kode sumber Apache Jena bisa diunduh di <https://jena.apache.org/download/index.cgi>. Dalam konteks ini, Apache Jena bertindak sebagai *plugin* dalam program Java sehingga memungkinkan aplikasi Java bisa memodelkan dan mengolah data RDF.

2.10. SPARQL

SPARQL (dibaca “sparkle”) adalah protokol RDF *Query Language* yang berfungsi untuk mengambil dan memanipulasi data dari sebuah basis data *triple-store*. RDF (*Resource Description Framework*) adalah tipe file yang terdiri dari statemen-statemen yang memiliki tiga variabel sebagai subjek, predikat, dan objek. Protokol SPARQL umumnya digunakan oleh peneliti Semantic Web. Contoh *syntax* SPARQL seperti yang ditunjukkan oleh Gambar 2.8:

```
PREFIX fam: <http://www.co-ode.org/roberts/family-tree.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?s
WHERE {
  ?s rdf:type foaf:Person.
  ?s foaf:name ?name
}
```

Gambar 2.8 Contoh SPARQL Query

Perbedaan *query* SPARQL dari *query* SQL adalah SPARQL digunakan untuk melakukan *query* data berbentuk RDF sedangkan SQL digunakan untuk melakukan *query* data relasional. Kelebihan dari *query* SPARQL adalah *syntax* yang tidak harus mengacu pada struktur basis data dan adanya atribut relasi yang memudahkan pengaksesan data (tidak harus memakai konsep *primary key* dan *foreign key*). SPARQL juga bisa melakukan *query* terhadap hirarki *class ontology*, sedangkan dalam SQL semua tabel dianggap sama. Pada tugas akhir ini, SPARQL Query dilakukan untuk mencari nama-nama relasi pada basis data Apache Jena Fuseki.

2.11. SPARQL Lib

SPARQL Lib adalah sebuah library PHP yang dikembangkan oleh departemen Computer Science dari University of Southampton, United Kingdom yang berfungsi untuk mengolah

data bertipe RDF dalam aplikasi berbasis PHP. Dalam konteks ini, SPARQL Lib digunakan untuk mengambil data RDF dari basis data triple store melalui panggilan API. Contoh dasar penggunaan SPARQL Lib dalam proyek berbasis PHP adalah seperti Gambar 2.9. Hasil dari SPARQL Query dengan SPARQL Lib adalah array *row* dan *field*. Gambar 2.9 mencontohkan cara mengambil data ruangan dan labelnya dari situs basis data yang menyediakan SPARQL API yaitu <http://sparql.data.southampton.ac.uk/>.

Code

```
<?php
require_once( "sparqllib.php" );

$data = sparql_get(
    "http://sparql.data.southampton.ac.uk/",

    PREFIX rooms: <http://vocab.der.i.e/rooms#>
    SELECT DISTINCT * WHERE { ?room a rooms:Building . ?room rdfs:label ?label } LIMIT 5
    "
);
if( !isset($data) )
{
    print "<p>Error: ".sparql_errno().": ".sparql_error()."</p>";
}

print "<table class='example_table'>";
print "<tr>";
foreach( $data->fields() as $field )
{
    print "<th>$field</th>";
}
print "</tr>";
foreach( $data as $row )
{
    print "<tr>";
    foreach( $data->fields() as $field )
    {
        print "<td>$row[$field]</td>";
    }
    print "</tr>";
}
print "</table>";
```

Output

| room | label |
|---|---------------------------|
| http://id.southampton.ac.uk/building/60 | Gower |
| http://id.southampton.ac.uk/building/1101 | 31 University Road |
| http://id.southampton.ac.uk/building/177 | Building 177 |
| http://id.southampton.ac.uk/building/70D | Chamberlain Dining Room |
| http://id.southampton.ac.uk/building/42 | Students' Union/Refectory |

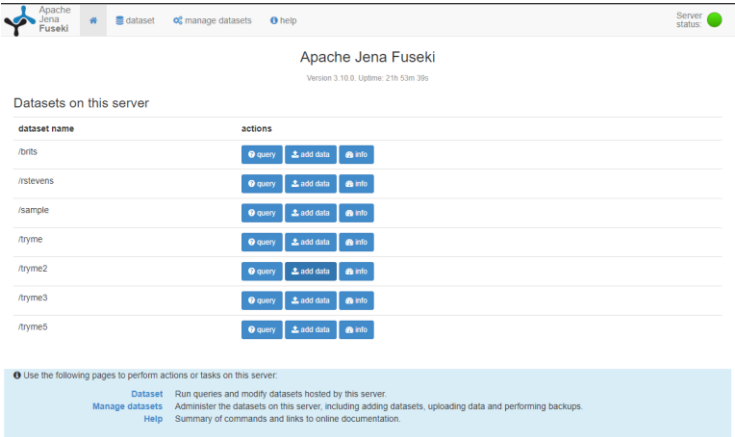
Gambar 2.9 Contoh Penggunaan SPARQL Lib

Dalam tugas akhir ini, SPARQL Lib digunakan oleh aplikasi web untuk berkomunikasi dengan basis data Apache Jena

Fuseki untuk membaca data dan mengambil informasi relasi dan tokoh yang dibutuhkan.

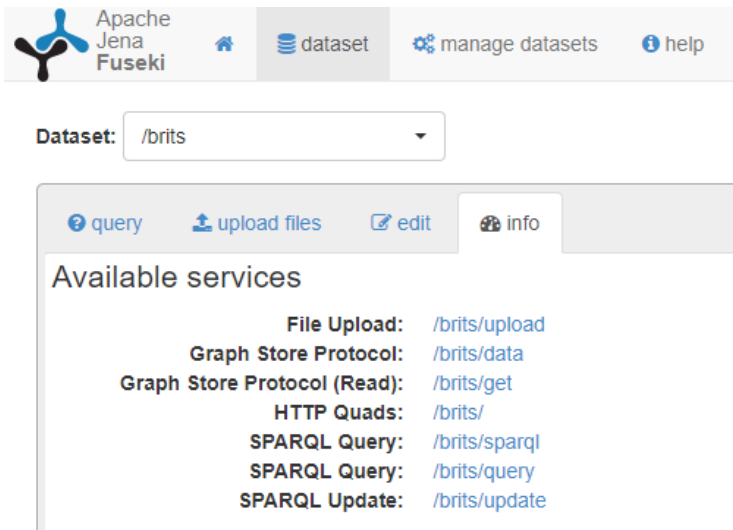
2.12. Apache Jena Fuseki

Apache Jena Fuseki adalah server SPARQL yang juga bisa bertindak sebagai *service* sistem operasi dan aplikasi web berbasis java. Dalam konteks ini, Apache Jena-Fuseki bertindak sebagai basis data *triple-store* yang bisa diakses melalui request HTTP. Gambar 2.10. menunjukkan daftar basis data yang ada di dalam *server* Apache Jena Fuseki.



Gambar 2.10 Database Triple Store Apache Jena Fuseki

Apache Jena Fuseki menyediakan beberapa API untuk digunakan oleh peneliti seperti pada Gambar 2.11.



Gambar 2.11 Daftar API Apache Jena Fuseki

Beberapa fungsi API tersebut antara lain untuk *endpoint* pengunggahan file RDF (/upload), membaca data (/get), *query* SPARQL (/query), dan memperbarui data (/update). Untuk implementasi pengaksesan basis data, API yang digunakan adalah SPARQL *Query*. *Return value* dari SPARQL *query* menggunakan Apache Jena Fuseki bisa berupa JSON (JavaScript Object Notation), XML (eXtensible Markup Language), atau CSV (Comma Separated Value).

Dalam tugas akhir ini, Apache Jena Fuseki berperan sebagai penyimpanan data RDF dan melayani aplikasi web saat aplikasi web melakukan SPARQL query.

2.13. PHP

PHP adalah bahasa pemrograman berbasis *open source* yang dikembangkan untuk mengembangkan perangkat lunak berbasis web dan bisa digabungkan dengan HTML. PHP juga dapat ditanamkan ke dalam HTML. PHP tidak seperti bahasa C untuk

pengembangan *web*. Namun struktur sintaks dasarnya sama, sehingga fleksibel dan mudah untuk diimplementasikan [13].

PHP sendiri sudah memiliki berbagai fitur yang komprehensif dan juga mendukung pemrograman berorientasi objek. Saat ini, PHP lebih sering disebut sebagai bahasa pemrograman dinamis. Berbeda bahasa pemrograman lainnya yang bersifat tradisional seperti C/ C++, kompilasi tidak diperlukan oleh PHP. Keuntungan lain yang diberikan PHP adalah fleksibilitas. *Bug* yang terjadi dapat dengan mudah dirubah atau diperbaiki dalam beberapa menit karena tidak diperlukan proses kompilasi dan mampu menciptakan versi baru dari program secara bertahap [14].

Untuk berhubungan dengan *web browser*, PHP memiliki *method* yang dapat dipanggil sesuai fungsinya. Terdapat dua cara yang digunakan oleh *browser client* untuk mengirimkan informasi pada *web server*, yaitu *GET method* dan *POST method*. Sebelum browser mengirim informasi, *browser* mengkonversi informasi tersebut menggunakan sebuah skema yang disebut *URL encoding*. *GET method* mengirimkan informasi pengguna yang telah dikodekan untuk ditambahkan pada *page request*. Halaman dan kode informasi dipisah oleh karakter "?". Sedangkan *POST method* memindahkan informasi melalui header HTTP. Kode informasi yang diberikan oleh *GET method* kemudian dimasukkan ke dalam *header* yang disebut *QUERY_STRING*. Selain itu, pengguna juga dapat menyertakan isi dari sebuah berkas PHP ke dalam berkas PHP lain sebelum *server* mengeksekusinya. Salah satunya adalah fungsi `require()` yang mengambil semua teks dalam *file* tertentu dan menyalinnya dalam *file* yang menggunakan fungsi `include()` [15]. Tabel berikut akan menerangkan beberapa *method* yang digunakan dalam implementasi pengerjaan tugas akhir ini.

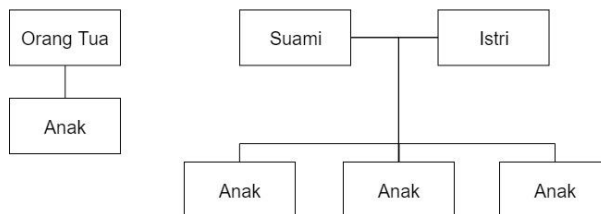
Tabel 2.4 Method PHP

| Method | Fungsi |
|--------|--------------------------------------|
| isset | Mengecek ada tidaknya suatu variabel |

| Method | Fungsi |
|-------------|--|
| echo | Menampilkan string |
| GET | Mengirimkan nilai variabel ke halaman lain |
| if | Mendesripsikan logika untuk kondisi data |
| foreach | Perulangan yang digunakan untuk array |
| str_replace | Mengganti karakter/ substring tertentu dalam suatu string |
| urlencode | Jika ada string yang ingin dikirim atau digabungkan ke penulisan alamat <i>website</i> |

2.14. Visualisasi

Visualisasi adalah teknik untuk menampilkan sesuatu secara grafis, seperti gambar, foto, bagan, diagram, dan lain-lain. Sekarang, visualisasi sudah diaplikasikan di bidang sains, pendidikan, kesehatan, industri, dan periklanan. Hal tersebut dikarenakan informasi yang divisualisasikan lebih mudah dicerna oleh manusia daripada informasi yang berbentuk paragraf atau teks panjang. Dalam konteks tugas akhir ini, visualisasi yang dimaksud adalah bagan pohon keluarga. Dalam bagan pohon keluarga, terdapat informasi sekumpulan orang beserta relasinya dalam keluarga tersebut dalam beberapa generasi. Relasi dalam suatu pohon keluarga dimodelkan dengan bentuk garis vertikal dan horizontal seperti pada Gambar 2.12. Jika terdapat garis vertikal di antara dua orang, maka hubungan kedua orang tersebut adalah orang tua dan anaknya, dan jika terdapat garis horizontal di antara dua orang atau lebih, maka hubungan tersebut adalah saudara atau pasangan.



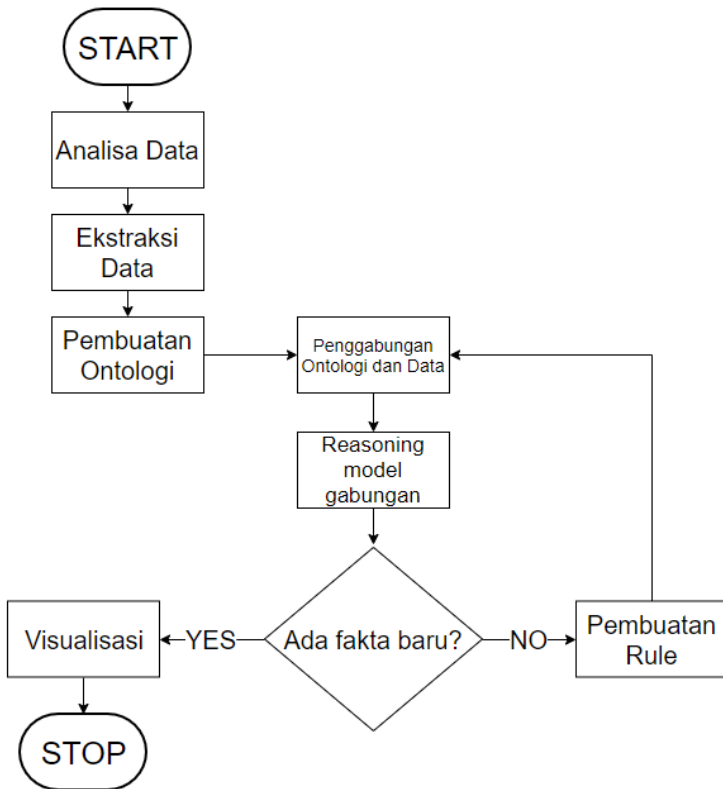
Gambar 2.12 Contoh pohon keluarga

(Halaman ini sengaja dikosongkan)

BAB III

METODOLOGI PEMECAHAN MASALAH

Pada bab ini dijelaskan mengenai langkah-langkah yang dilakukan untuk mencari relasi dari suatu *person*. Mulai dari metode yang dilakukan untuk mengambil data *person* sampai menampilkan grafik pohon keluarga.



Gambar 3.1 Flowchart pengembangan sistem

Alur pemecahan masalah dapat dilihat pada Gambar 3.1. Pemecahaan masalah dimulai dengan menganalisis data DBpedia. Setelah analisis dilakukan, maka diputuskan untuk menggunakan data keluarga tokoh sejarah Indonesia yang disediakan oleh DBpedia Indonesia. Kemudian, data diekstrak dengan aplikasi berbasis Jena agar dapat digabungkan dengan ontologi secara mudah. Proses ekstraksi data dan penggabungan data dilakukan menggunakan Apache Jena. Proses selanjutnya adalah melakukan *reasoning* dengan Pellet Reasoner di dalam aplikasi Jena. Setelah *reasoning* selesai, maka fakta-fakta baru akan dihasilkan, serta ontologi yang baru akan diupload ke basis data *triple-store*. Data di *triple-store* lalu ditampilkan secara grafis sebagai pohon keluarga. Deskripsi lebih detail tentang setiap proses akan dijelaskan lebih detail pada subbab bab ini.

3.1. Analisis Data

Untuk memecahkan masalah pencarian relasi keluarga tokoh bersejarah, langkah yang pertama kali dilakukan adalah menganalisis dataset yang akan digunakan. Ontologi memiliki beberapa domain, yaitu *actor*, *place*, *time*, dan *event*. Domain yang menjadi topik pada pengerjaan tugas akhir ini adalah *actor*. Ruang lingkup *actor* meliputi *person*, *group*, dan *organization*. *Person* tidak dapat berdiri sendiri tanpa adanya keterkaitan dengan *place*, *time*, dan *event*.

Data yang digunakan dalam perancangan ontologi ini adalah data biografi keluarga tokoh sejarah Indonesia. Daftar tokoh sejarah Indonesia dapat diperoleh dari ensiklopedia gratis Wikipedia Indonesia. Data yang diambil adalah data Presiden dan keluarganya. Data daftar Presiden Indonesia bisa dilihat di laman https://id.wikipedia.org/wiki/Daftar_Presiden_Indonesia.

Person yang digunakan sebagai data adalah *person* yang dinilai memiliki banyak keterkaitan dengan *person* lain. Selain memiliki atribut *property* keluarga, *Person* juga harus memiliki atribut identitas seperti *foaf:name* atau *rdfs:label*. Maka dari itu,

untuk pembandingan, akan diolah pula data DBpedia Inggris tentang kerajaan Inggris.

Domain inti dari sebuah ontologi menangkap konsep utama (*classes*) dan hubungan (*properties*) yang mencakup ruang lingkup domain tersebut. Bahkan ontologi dengan *domain* yang sama bisa heterogen karena berbagai kepentingan, perspektif pengembang, tujuan yang berbeda, dan konteks aplikasi. Untuk membuat ontologi yang lengkap dan mencakup semua inti domain akan membutuhkan *cost* yang tinggi karena ekonomi, waktu, sumber daya lainnya, serta kondisi dunia yang selalu berubah [16].

3.1.1. Analisis Data dari DBpedia

Terdapat berbagai macam *open data* yang dapat diakses melalui *internet* tanpa membayar seperti DBpedia. Data keluarga yang akan dipakai sebagai model adalah data dengan kelengkapan properti-property utama yang diperlukan untuk mengetahui silsilah keluarga seorang *person* seperti *child*, *spouse*, dan *parent*. Contoh property yang akan dipakai adalah *data property* dari Raden Wijaya yang bisa dilihat di Gambar 3.2.

| | |
|------------------------|---|
| dbpprop-id:consort | ▪ dbpedia-id:Gayatri |
| dbpprop-id:coronation | ▪ 15 (xsd:integer) |
| dbpprop-id:deathDate | ▪ 1309 (xsd:integer) |
| dbpprop-id:deathPlace | ▪ Majapahit |
| dbpprop-id:dynasty | ▪ dbpedia-id:Wangsa_Rajasa |
| dbpprop-id:fullName | ▪ Nararya Sanggramawijaya |
| dbpprop-id:heir | ▪ dbpedia-id:Jayanegara |
| dbpprop-id:image | ▪ 180 (xsd:integer) |
| dbpprop-id:jabatan | ▪ Raja Majapahit |
| dbpprop-id:name | ▪ Raden Wijaya |
| dbpprop-id:otherTitles | ▪ Kertarajasa Jayawardhana |
| dbpprop-id:pendahulu | ▪ - |
| dbpprop-id:pengganti | ▪ dbpedia-id:Jayanagara |
| dbpprop-id:queen | ▪ dbpedia-id:Tribhuwaneswari |
| dbpprop-id:reign | ▪ Majapahit: 1293 - 1309 |
| dbpprop-id:spouse | ▪ dbpedia-id:Prajnaparamita ▪ dbpedia-id:Narendraduhita ▪ dbpedia-id:Indreswari |

Gambar 3.2 Halaman DBpedia tentang property keluarga Raden Wijaya

Berdasarkan semua *property* yang terdapat pada halaman DBpedia, dipilih *property* dalam batasan masalah seperti *name*, *parent*, *spouse*, dan *child* atau *issue*(istilah resmi untuk keturunan biologis). Untuk melengkapi data, akan dibuat visualisasi menantu, cucu, pasangan cucu, dan cicit seperti pada subbab 1.3, Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Daftar property yang dibutuhkan untuk visualisasi

| Property yang dibutuhkan |
|--------------------------|
| Name |
| Spouse |
| Parent |
| Issue |

Tabel 3.2 Daftar property tambahan yang divisualisasikan

| Property yang akan dihasilkan |
|-------------------------------|
| Child in Law |
| Grandchild |
| Grandchild in Law |
| Great grand child |

Selain itu, penting juga untuk mempertimbangkan *namespace* dalam DBpedia, dikarenakan DBpedia memiliki banyak situs alternatif selain DBpedia (*dbpedia.org*), yaitu DBpedia Indonesia (*id.dbpedia.org*), DBpedia Italia (*it.dbpedia.org*), DBpedia Perancis (*fr.dbpedia.org*), dan lain-lain dimana tiap situs memiliki dan menyediakan properti-property yang berbeda-beda. Beberapa *namespace* dan keterangannya bisa dilihat di Tabel 3.3.

Tabel 3.3 Namespace DBpedia

| Namespace | Keterangan |
|------------|---|
| dbpedia | < http://dbpedia.org/ > |
| id.dbpedia | < http://id.dbpedia.org/ > |

| | |
|-------------|---|
| foaf | <http://xmlns.com/foaf/0.1/> |
| rdf | <http://www.w3.org/1999/02/22-rdf-syntax-ns#> |
| rdfs | <http://www.w3.org/2000/01/rdf-schema#> |
| dbo | <http://dbpedia.org/ontology/> |
| dbp | <http://dbpedia.org/property/> |
| dbpprop-id | <http://id.dbpedia.org/property/> |
| dbpedia-owl | <http://id.dbpedia.org/ontology/> |
| owl | <http://www.w3.org/2002/07/owl#> |
| yago | <http://yago-knowledge.org/resource/> |
| schema | <http://schema.org/> |
| dct | <http://purl.org/dc/terms/> |

Perlu diketahui pula pengertian properti-property yang akan sering digunakan dalam pengerjaan tugas akhir ini dikarenakan setiap URL tokoh yang diambil tidak memiliki jumlah property maupun jenis property yang sama. Tidak semua URL memiliki property dasar seperti nama (foaf:name, dbpprop-id:name, dbp:name), sehingga data yang dibutuhkan untuk visualisasi tidak lengkap. Contoh data dasar *person* yang tidak lengkap seperti Gambar 3.3 yaitu salah satu istri Ir. Soekarno, Ibu Haryati yang tidak memiliki property nama dan *entity type*-nya adalah *Thing* padahal seharusnya *Person* seperti yang telah dijelaskan di subbab 2.2 poin a. Dapat dilihat juga data tersebut hanya memiliki property keluarga *spouse* saja. Meskipun beberapa property dapat dihasilkan dari proses *reasoning*, akan tetapi jika data dasar tidak lengkap, tidak semua data bisa diperkaya.

Content-Length: 6908

About: <http://id.dbpedia.org/resource/Haryati>

An Entity of Type : [Thing](#), from Named Graph : <http://id.dbpedia.org>, within Data Space : id.dbpedia.org

DBpedia

| Property | Value |
|------------------------------------|---|
| is dbpedia-owl:spouse of | <ul style="list-style-type: none"> dbpedia-id:Soekarno |
| is dbpedia-owl:wikiPageWikiLink of | <ul style="list-style-type: none"> dbpedia-id:Soekarno |
| is dbpprop-id:spouse of | <ul style="list-style-type: none"> dbpedia-id:Soekarno |

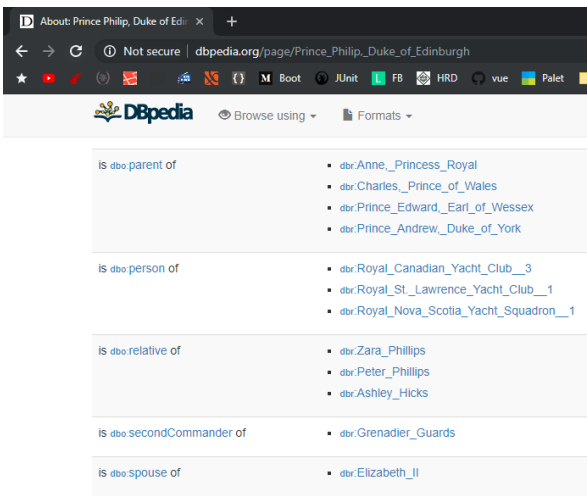
Browse using: [OpenLink Data Explorer](#) | [Zotist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) | Raw Data in: [CSV](#) | [RDF](#) | [N-Triples](#) | [N3](#) | [Turtle](#) | [JSON](#) | [XML](#) | [JSON-LD](#) | [Microdata](#) | [JSON](#) | [JSON-LD](#) | [JSON-LD](#) | [About](#)

[Creative Commons](#) | [Virtuoso](#) | [Linking Open Data](#) | [W3C](#) | [DBpedia](#) | [Open Data](#) | [W3C](#) | [Creative Commons](#)

This content was extracted from [Wikidata](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

Gambar 3.3 Data DBpedia Haryati

Sebaliknya, data yang memiliki property dasar dan property keluarga seperti spouse, parent, child akan lebih lengkap hasilnya ketika dilakukan proses *reasoning*, contohnya adalah data keluarga kerajaan Inggris (*Royal Family*) dari DBpedia berbahasa Inggris (*dbpedia.org*) seperti contoh data keluarga Prince Philip pada Gambar 3.4.



Gambar 3.4 Data DBpedia Prince Philip

3.2. Ekstraksi Data Sebagai Model

Untuk melakukan proses ekstraksi data, diperlukan aplikasi berbasis Java yang memiliki *plugin* Apache Jena. Kode Sumber 3.1 berikut digunakan untuk ekstraksi dan pemodelan tokoh dari DBpedia. Adapun daftar tokoh yang datanya digunakan dalam tugas akhir ini terdapat pada Kode Sumber 3.1


Error! eference source not found.

```
Model modelActor =
fManager.loadModel ("http://dbpedia.org/data/Fatmawati");
```

Kode Sumber 3.1 Kode Java untuk memodelkan data Fatmawati ke dalam modelActor

Kode pada Kode Sumber 3.1 akan membaca semua data property Fatmawati serta relasinya yang tercantum dalam halaman DBpedia pada Gambar 3.5. Tentu saja tidak hanya Fatmawati saja yang dijadikan model, tokoh sejarah lainnya juga dimodelkan untuk dijadikan data pembanding. Jika model Fatmawati saja yang direpresentasikan sebagai *graph*, maka representasinya adalah seperti pada Gambar 3.6:

Content-Length: 50597

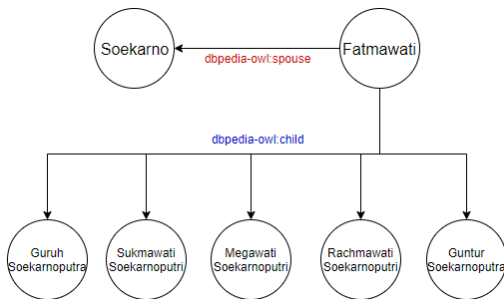


About: Fatmawati
An Entity of Type : [Person](#), from Named Graph : <http://id.dbpedia.org>, within Data Space : id.dbpedia.org

Fatmawati yang bernama asli Fatimah adalah istri dari Presiden Indonesia pertama Soekarno. Ia menjadi ibu Negara Indonesia pertama dari tahun 1945 hingga tahun 1967 dan merupakan istri ke-3 dari Presiden Pertama Indonesia, Soekarno. Ia juga dikenal akan jasanya dalam menjahit Bendera Pusaka Sang Saka Merah Putih yang turut dikibarkan pada upacara Proklamasi Kemerdekaan Indonesia di Jakarta pada tanggal 17 Agustus 1945.

| Property | Value |
|--|--|
| dbpedia-owl:abstract | <ul style="list-style-type: none"> Fatmawati yang bernama asli Fatimah adalah istri dari Presiden Indonesia pertama Soekarno. Ia menjadi ibu Negara Indonesia pertama dari tahun 1945 hingga tahun 1967 dan merupakan istri ke-3 dari Presiden Pertama Indonesia, Soekarno. Ia juga dikenal akan jasanya dalam menjahit Bendera Pusaka Sang Saka Merah Putih yang turut dikibarkan pada upacara Proklamasi Kemerdekaan Indonesia di Jakarta pada tanggal 17 Agustus 1945. |
| dbpedia-owl:activeYearsEndDate | <ul style="list-style-type: none"> 1967-03-12 (xsd:date) |
| dbpedia-owl:activeYearsStartDate | <ul style="list-style-type: none"> 1945-08-17 (xsd:date) |
| dbpedia-owl:birthDate | <ul style="list-style-type: none"> 1923-02-05 (xsd:date) |
| dbpedia-owl:birthPlace | <ul style="list-style-type: none"> dbpedia-id:Bengkulu dbpedia-id:Hindia_Belanda |
| dbpedia-owl:child | <ul style="list-style-type: none"> dbpedia-id:Guruh_Soekarnoputra dbpedia-id:Megawati_Soekarnoputri dbpedia-id:Sukmawati_Soekarnoputri dbpedia-id:Rachmawati_Soekarnoputri dbpedia-id:Guntur_Soekarnoputra |
| dbpedia-owl:deathPlace | <ul style="list-style-type: none"> dbpedia-id:Kuala_Lumpur dbpedia-id:Malaysia |
| dbpedia-owl:nationality | <ul style="list-style-type: none"> dbpedia-id:Indonesia |
| dbpedia-owl:office | <ul style="list-style-type: none"> Ibu Negara Indonesia |
| dbpedia-owl:orderInOffice | <ul style="list-style-type: none"> 1 |
| dbpedia-owl:president | <ul style="list-style-type: none"> dbpedia-id:Soekarno |
| dbpedia-owl:religion | <ul style="list-style-type: none"> dbpedia-id:Islam |
| dbpedia-owl:spouse | <ul style="list-style-type: none"> dbpedia-id:Soekarno |

Gambar 3.5 Halaman DBpedia Fatmawati

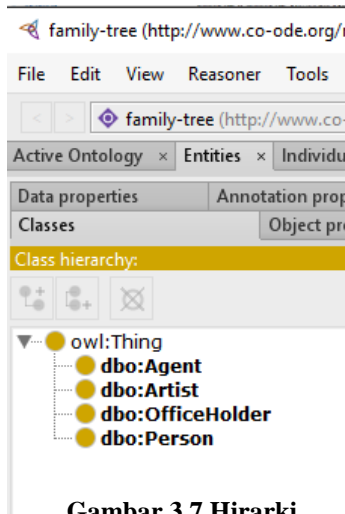


Gambar 3.6 Representasi data keluarga Fatmawati

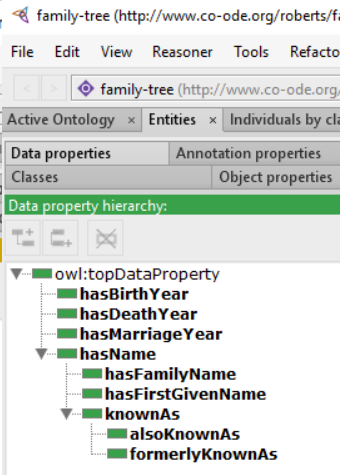
3.3. Pembuatan Ontologi

Pada tahap ini, ontologi dibangun dengan mengubah ontologi yang sudah ada. Ontologi yang digunakan adalah *Family Relationships Ontology* milik Robert Stevens. Akan tetapi tidak semua *class*, *individual*, *data properties*, ataupun *object properties* akan digunakan, hanya yang benar-benar dibutuhkan saja. Beberapa *property* yang digunakan adalah *hasChild*, *hasParent*, *isSpouseOf*, *hasChildInLaw*, *hasGrandChild*, *hasGrandChildInLaw*, *hasGreatGrandChild*. Beberapa *property* yang memiliki arti yang sama akan diatur sebagai *equivalent class*, seperti *property* *hasChild*, *isParentOf*, *dbp:children*, *dbp:issue*, dan *dbo:child*. Daftar *class* dan *property* yang akan digunakan ditunjukkan di Gambar 3.7, Gambar 3.8 dan Gambar 3.9. Dan pemetaan *property* DBpedia dan Family Tree App dapat dilihat di

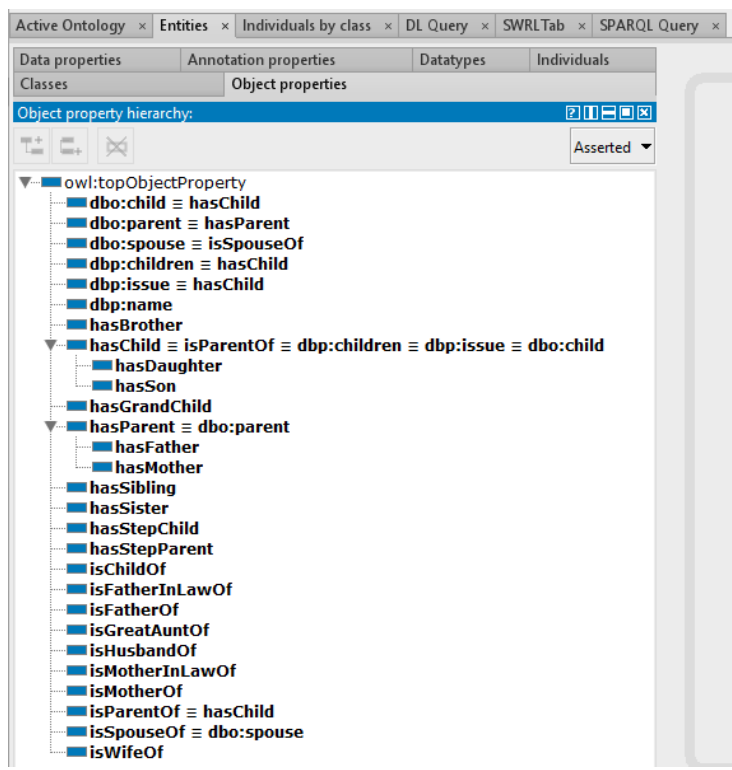
Tabel 3.4. *Object property* dan *data property* yang digunakan dapat dilihat di Tabel 3.6 dan Tabel 3.7.



Gambar 3.7 Hirarki Class



Gambar 3.8 Hirarki Data property



Gambar 3.9 Hirarki Object property

Tabel 3.4 Pemetaan property equivalent

| Property DBpedia | Property Family Tree App |
|------------------|--|
| foaf:name | foaf:name |
| dbo:child | hasChild, isParentOf, dbp:children, dbp:issue, dbo:child |
| dbo:spouse | isSpouseOf, dbo:spouse |
| dbo:parent | hasParent, dbo:parent |

Tabel 3.5 Daftar Class

| Class | URL | Karakteristik | Keterangan |
|------------|---|---------------|--------------------------------------|
| Owl:Thing | http://www.w3.org/2002/07/owl#thing | - | Class untuk menjelaskan suatu hal |
| Dbo:agent | http://dbpedia.org/ontology/agent | - | Class untuk menjelaskan <i>agent</i> |
| Dbo:Person | http://dbpedia.org/ontology/person | - | Class untuk menjelaskan seseorang |

Tabel 3.6 Daftar Object Property

| Object property | URL | Karakteristik | Keterangan |
|-----------------|--|---------------|-----------------------------|
| dbo:child | dbpedia.org/ontology/child | - | ekuivalen dengan hasChild |
| dbo:parent | dbpedia.org/ontology/parent | - | ekuivalen dengan hasParent |
| dbo:spouse | dbpedia.org/ontology/spouse | symmetric | ekuivalen dengan isSpouseOf |
| dbp:children | dbpedia.org/property/children | - | ekuivalen dengan hasChild |

| | | | |
|---------------|--|-----------|---|
| dbp:issue | dbpedia.org/property/issue | - | ekuivalen dengan hasChild |
| hasChild | co-ode.org/roberts/family-tree.owl#haschild | - | ekuivalen dengan dbo:child, dbp:children, dbp:issue, dan isParentof |
| hasGrandChild | co-ode.org/roberts/family-tree.owl#hasgrandchild | - | SuperProperty dari 'hasChild o hasChild' |
| hasParent | co-ode.org/roberts/family-tree.owl#hasparent | - | ekuivalen dengan dbo:parent |
| hasSibling | co-ode.org/roberts/family-tree.owl#hassibling | symmetric | property ini memberlakukan kebalikan |
| isChildOf | co-ode.org/roberts/family-tree.owl#ischildof | - | inverse dari hasChild |
| isParentOf | co-ode.org/roberts/family-tree.owl#isparentof | - | ekuivalen dengan hasChild |
| isSpouseOf | co-ode.org/roberts/family-tree.owl#isspouseof | - | ekuivalen dengan dbo:spouse |

Tabel 3.7 Daftar Data Property

| Data property | URL | Karakteristik | Keterangan |
|---------------|--|---------------|------------------------|
| Dbp:name | dbpedia.org/property/name | - | Menerangkan data nama |
| hasName | code.org/roberts/family-tree.owl#hasname | - | Menerangkan data nama |
| Label | rdfs:label | - | Menerangkan data label |

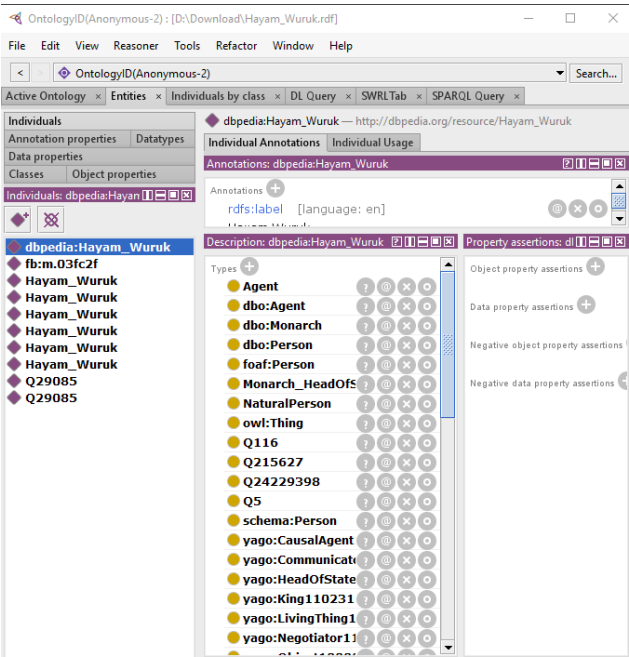
3.4. Penggabungan Model Data dan Model Family Relationship Ontology

Model yang digabungkan adalah model data DBpedia yang diperoleh dengan ekstraksi dan model ontologi *Family Relationship Ontology* yang telah dibuat pada langkah pembuatan ontologi di subbab 3.3. Penggabungan ini dilakukan dengan menggunakan fungsi *createUnion* dari class ModelFactory seperti pada Kode Sumber 3.2. Input dari fungsi ini berupa parameter dua model yang ingin digabungkan, dan outputnya adalah model yang sudah digabungkan.

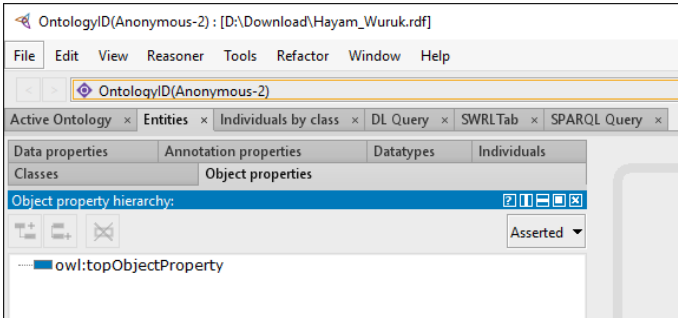
```
final Model union =
ModelFactory.createUnion(modelHayamWuruk,modelFamilyTree
);
```

Kode Sumber 3.2 Kode sumber untuk menggabungkan dua model

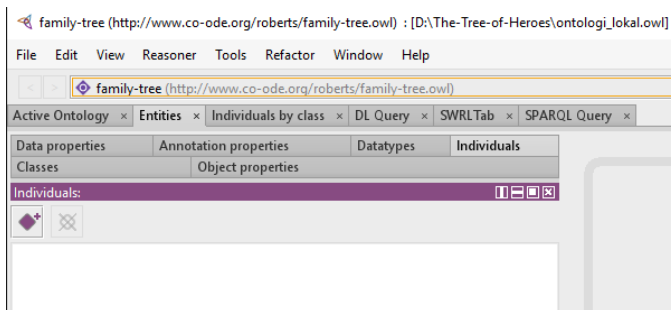
Sebagai input, adalah model ontologi Hayam Wuruk (Gambar 3.10 dan Gambar 3.11) dan model ontologi Family Tree (Gambar 3.12 dan Gambar 3.13). Sebagai output, adalah ontologi union yang baru, yang menyimpan data dan property dari kedua ontologi pada Gambar 3.14 dan Gambar 3.15.



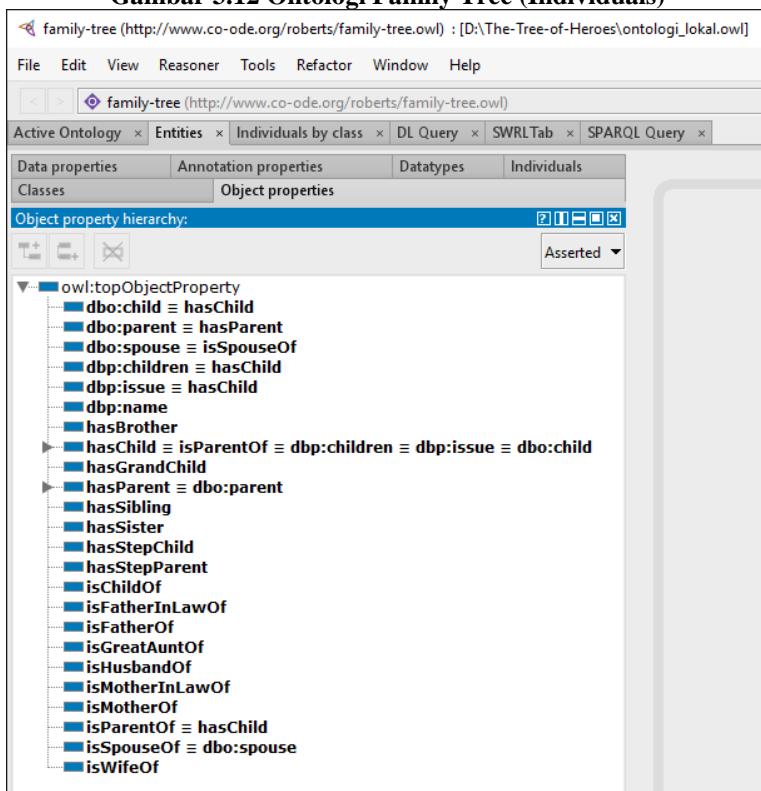
Gambar 3.10 Ontologi Hayam Wuruk (Individuals)



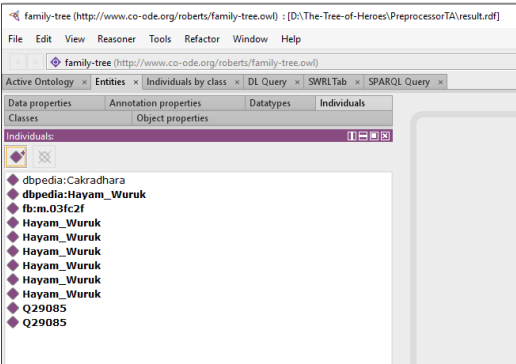
Gambar 3.11 Ontologi Hayam Wuruk (Object Properties)



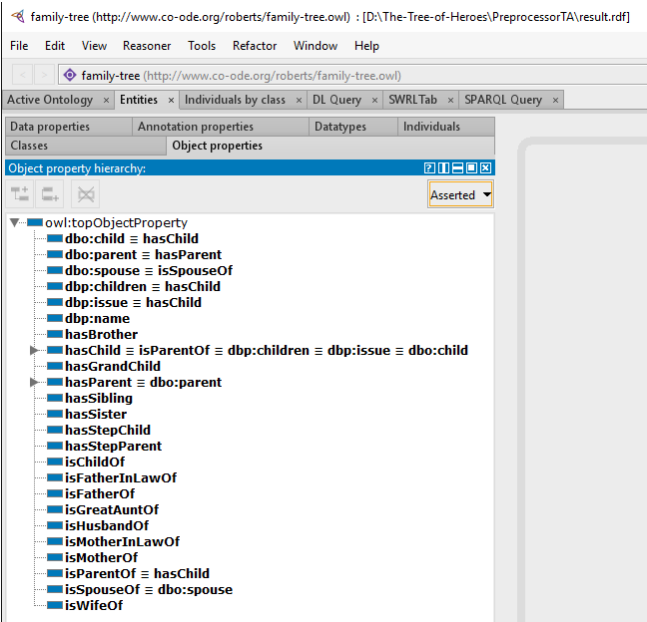
Gambar 3.12 Ontologi Family Tree (Individuals)



Gambar 3.13 Ontologi Family Tree (Object Properties)



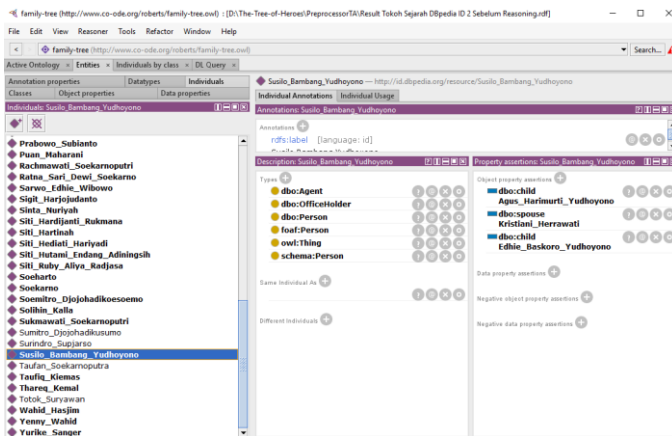
Gambar 3.14 Ontologi union (Individuals)



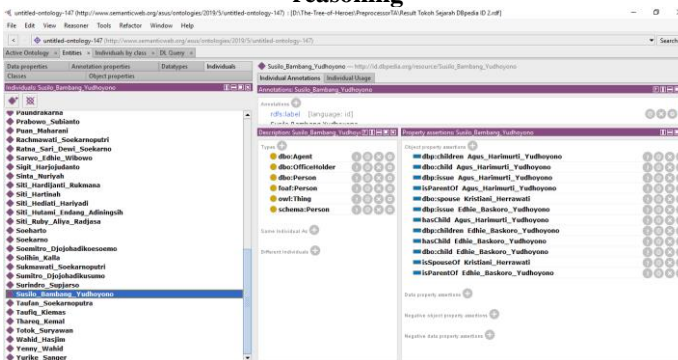
Gambar 3.15 Ontologi union (Object Properties)

3.5. Reasoning pada Model Gabungan

Untuk proses *reasoning*, yang digunakan adalah Pellet Reasoner. Proses ini terdiri dari tiga fase, yaitu *model reading* yaitu membaca model RDF, *classifying* atau pengklasifikasi, dan *realizing*. Setelah diperoleh hasilnya, maka hasil tersebut akan diprint menjadi file RDF yang selanjutnya diunggah ke basis data Apache Jena Fuseki. Contoh proses reasoning bisa dilihat di Gambar 3.16 sebagai input dan Gambar 3.17 sebagai output.



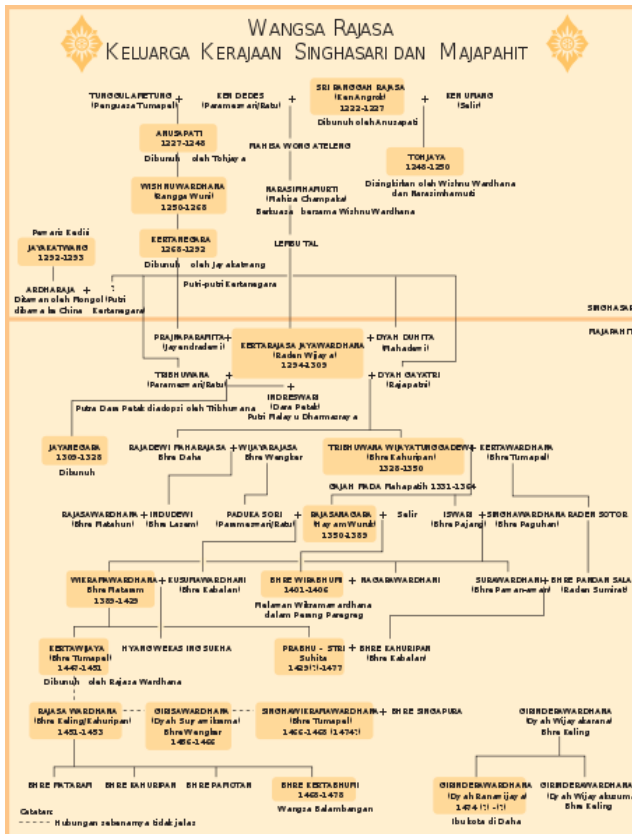
Gambar 3.16 Individu Susilo Bambang Yudhoyono sebelum reasoning



Gambar 3.17 Individu Susilo Bambang Yudhoyono setelah reasoning

3.6. Penampilan Data

Untuk penampilan data dalam *platform* web, ada tiga bagian, yaitu proses *query* SPARQL menggunakan SPARQL Lib, pemilihan data individu dan visualisasi sebagai pohon keluarga seperti yang dicontohkan pada Gambar 3.18 yang akan diimplementasi dengan bahasa pemrograman PHP.



Gambar 3.18 Silsilah keluarga kerajaan Singasari dan Majapahit [17]

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan tentang analisis permasalahan dan perancangan Tugas Akhir. Analisis permasalahan membahas tentang permasalahan yang diangkat dalam Tugas Akhir ini beserta solusi yang ditawarkan. Selanjutnya dibahas juga tentang perancangan sistem yang dibuat.

4.1. Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem dan kebutuhan perangkat lunak.

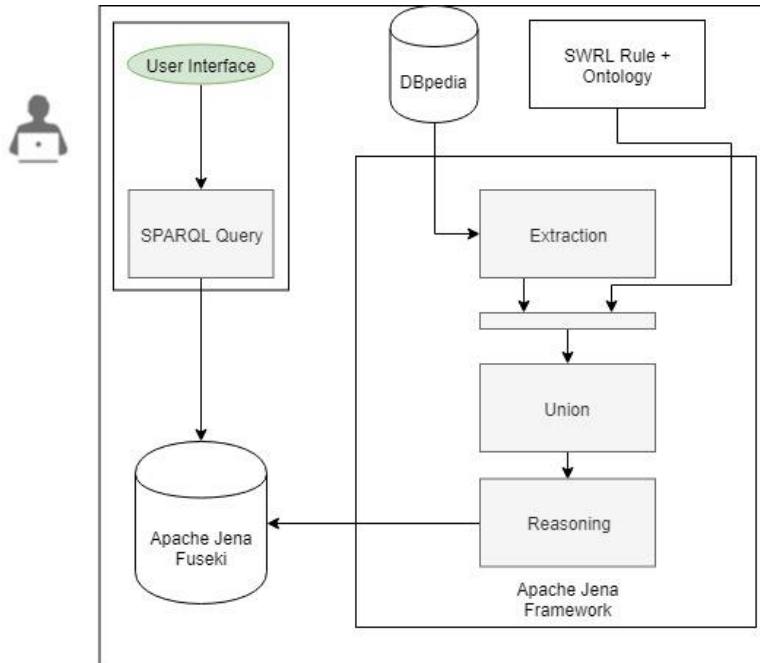
4.1.1. Cakupan Permasalahan

Permasalahan yang diangkat dalam tugas akhir ini adalah visualisasi pohon keluarga tokoh sejarah Indonesia. Studi kasus permasalahan tersebut dipecahkan dengan ekstraksi data, penggabungan data, *reasoning* dan visualisasi. Pencarian relasi antar *person* dilakukan dengan menggunakan *property* dan *SWRL rule*. Untuk mendapatkan fakta-fakta baru, dilakukan proses *reasoning* menggunakan *Pellet reasoner*. Setelah proses *reasoning* selesai, akan didapatlam fakta-fakta baru yang kemudian disimpan sebagai ontologi baru dalam bentuk RDF. Ontologi baru tersebut lalu disimpan di dalam basis data triple store. Tentu saja hal tersebut akan menyulitkan pengguna yang ingin mengetahui fakta-fakta baru yang muncul setelah ontologi diberikan *rule*. Oleh karena itu, agar dapat dimanfaatkan secara aplikatif maka dibutuhkan sebuah sistem sederhana yang dapat menampilkan hasil *reasoning* dari ontologi yang dibangun. Untuk memudahkan pengguna, sistem sederhana tersebut akan dirancang dengan tampilan yang mudah dipahami.

4.1.2. Deskripsi Umum Sistem

Perangkat lunak yang dibangun dalam pengerjaan tugas akhir ini diberi nama Family Tree App. Family Tree App dibangun

dengan tujuan untuk membantu ontologi dalam menampilkan hasil-hasil yang didapatkannya.



Gambar 4.1 Arsitektur Sistem

Arsitektur sistem bisa dilihat di Gambar 4.1 Arsitektur Sistem. Untuk menampilkan fakta-fakta yang didapatkan dari ontologi tersebut, perangkat lunak harus bisa membaca berkas ontologi yang telah dibangun. Family Tree App dirancang sebagai perangkat lunak berbasis *web* yang menggunakan bahasa pemrograman PHP dan *library* SPARQL Lib. Perangkat lunak ini bisa mengakses data dari basis data triple store. Sedangkan keluaran dari perangkat lunak Family Tree App adalah halaman HTML dengan tampilan pohon keluarga dari seorang tokoh yang bersumber dari basis data triple store tersebut. Berikut detail tiap komponennya.

- a) User Interface
User Interface ini memungkinkan pengguna untuk berinteraksi dengan sistem dengan cara memilih URL *person* dalam dropdown. Sistem lalu menampilkan skema pohon keluarga tokoh *person* yang dipilih.
- b) SPARQL Query
Setelah pengguna memilih *person*, sistem akan melakukan SPARQL *query* menggunakan *plugin* SPARQL Lib untuk mengambil data keluarga *person* yang dipilih dari basis data Apache Jena Fuseki.
- c) Apache Jena Fuseki
Apache Jena Fuseki berfungsi untuk menyimpan data berbentuk *triple-store*. Basis data ini menyediakan API untuk membaca data dan perintah *query*.
- d) DBpedia
Data dari DBpedia akan diunduh oleh aplikasi Jena, dan dimodelkan untuk digabungkan dengan ontologi *Family Relationship*.
- e) Ontology
Family Relationship Ontology yang telah direvisi seperti pada subbab 3.3 dibaca oleh aplikasi Jena dan dimodelkan, lalu digabungkan dengan model data DBpedia.
- f) Extraction
Merupakan proses ekstraksi data *person* dari DBpedia dengan cara mengunduh file RDF. Input untuk proses ini adalah URL DBpedia, sedangkan outputnya adalah file RDF seperti yang disampaikan pada subbab 3.2.

g) Union

Merupakan proses penggabungan model *Family Relationship* dan model data DBpedia. Input dari proses ini adalah dua model RDF dan outputnya adalah kombinasi dari keduanya. Contoh dari penggabungan model Hayam Wuruk dan model *Family Relationship Ontology* pada subbab 3.4.

h) Reasoning

Merupakan proses untuk mencari fakta baru dari suatu model di aplikasi Jena. Input dan output dari proses ini adalah model seperti yang dijelaskan pada subbab 0.

4.1.3. Spesifikasi Kebutuhan Perangkat Lunak

Bab ini menjelaskan kebutuhan perangkat lunak dalam bentuk diagram kasus dan diagram aktivitas. Masing-masing diagram menjelaskan perilaku atau sifat dari sistem ini.

4.1.3.1. Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan pokok yang harus dipenuhi agar sistem dapat berjalan dengan baik. Daftar kebutuhan fungsional dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak

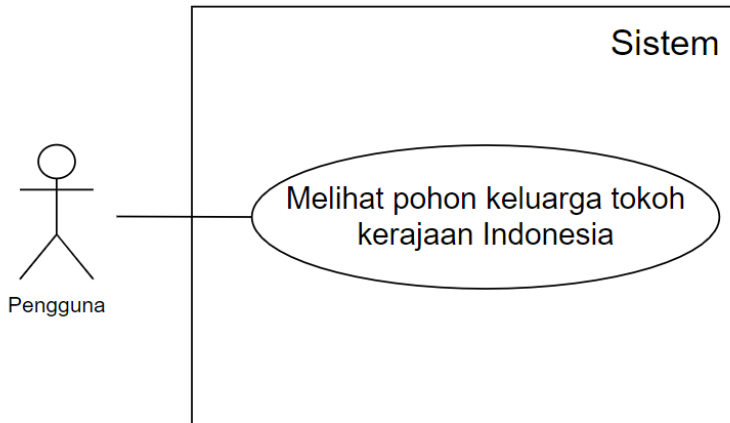
| Kode Kebutuhan | Kebutuhan Fungsional | Deskripsi |
|----------------|----------------------------------|---|
| TA-F0001 | Menampilkan pohon keluarga tokoh | Pengguna dapat melihat pohon keluarga tokoh sejarah Indonesia |

4.1.4. Aktor

Aktor merupakan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas yang dimaksud dapat berupa manusia, sistem, atau perangkat lunak yang lain. Aktor yang berinteraksi dengan Tugas Akhir ini yaitu pengguna yang diasumsikan tidak memahami bahasa pemrograman. Pengguna

dapat memilih entitas melalui *dropdown select* atau memilih tautan yang disediakan oleh sistem untuk melihat informasi dari seorang tokoh sejarah Indonesia.

4.1.5. Kasus Penggunaan



Gambar 4.2 Diagram Kasus Penggunaan Sistem

Kasus penggunaan dalam Subbab ini akan dijelaskan secara rinci. Kasus penggunaan dijabarkan dalam bentuk spesifikasi kasus penggunaan dan diagram aktivitas. Diagram kasus penggunaan dapat dilihat pada Gambar 4.2. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel 4.2.

Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan

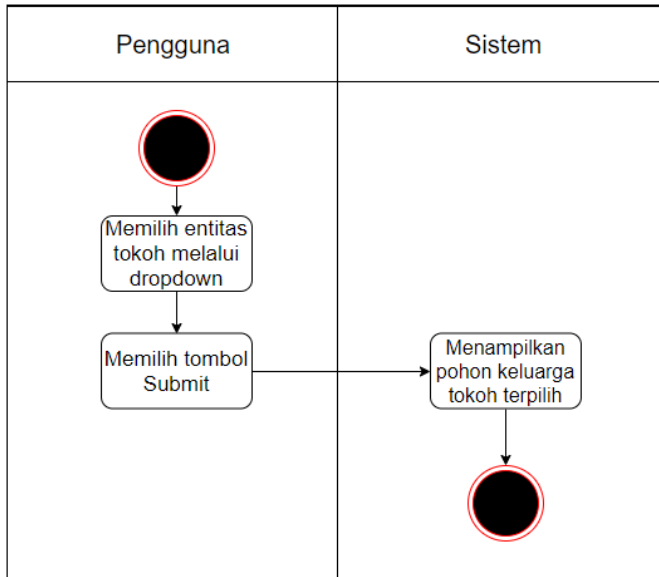
| Kode Kasus Penggunaan | Nama |
|-----------------------|--|
| TA-UC0001 | Melihat pohon keluarga tokoh sejarah Indonesia |

4.1.5.1. Melihat Pohon Keluarga Tokoh sejarah Indonesia

Pada kasus penggunaan ini, sistem membaca data yang ada di basis data Apache Jena Fuseki. Informasi yang terdapat dalam basis data tersebut selanjutnya dikonversi menjadi sebuah halaman HTML. Spesifikasi kasus penggunaannya dapat dilihat pada Tabel 4.3. Diagram aktivitasnya dapat dilihat pada Gambar 4.3.

Tabel 4.3 Spesifikasi Kasus Penggunaan Melihat Informasi Tokoh

| | |
|--|---|
| Nama | Melihat pohon keluarga tokoh |
| Kode | TA-UC0001 |
| Deskripsi | Pengguna bisa melihat pohon keluarga tokoh yang dipilih |
| Tipe | Fungsional |
| Pemicu | Pengguna menekan tombol <i>submit</i> |
| Aktor | Pengguna |
| Kondisi Awal | Pohon keluarga belum ditampilkan |
| Alur: - Kejadian Normal | <ol style="list-style-type: none"> 1. Pengguna memilih tokoh melalui <i>dropdown select</i>. 2. Pengguna menekan tombol <i>submit</i>. 3. Sistem menampilkan halaman pohon keluarga dari tokoh yang dipilih. |
| Kondisi Akhir | Sistem menampilkan informasi dari tokoh yang dipilih dalam bentuk pohon keluarga |
| Alur alternatif | <ol style="list-style-type: none"> 1. Pengguna memilih tautan. 2. Sistem menampilkan pohon keluarga tokoh tautan. |
| Kebutuhan Khusus | Tidak ada |

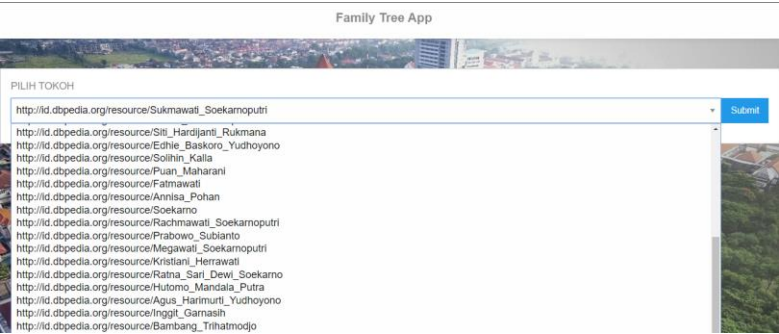


Gambar 4.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh

4.2. Perancangan Antarmuka Pengguna

Bagian ini membahas mengenai perancangan antarmuka yang akan dibuat. Rancangan antarmuka dibuat agar semudah mungkin dapat dipahami dan digunakan oleh pengguna.

Antarmuka Family Tree App terdiri dari satu halaman. Di halaman tersebut, terdapat satu panel *dropdown select* dan satu panel sebagai tempat deskripsi entitas tokoh atau tautan yang dipilih. Deskripsi entitas tokoh terdiri dari satu tabel dengan sejumlah baris informasi terkait entitas tokoh yang dipilih. Rancangan antarmuka halaman utama ini dapat dilihat pada Gambar 4.4. Sedangkan rancangan antarmuka halaman informasi data tokoh dapat dilihat pada Gambar 4.5. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini bisa dilihat pada Tabel 4.4.



Gambar 4.4 Antarmuka Halaman Utama Family Tree App

Sukmawati Soekarnoputri



Gambar 4.5 Antarmuka Halaman Pohon Keluarga Family Tree App

Tabel 4.4 Spesifikasi Atribut Rancangan Antarmuka Halaman Family Tree App

| No. | Nama Atribut Antarmuka | Jenis Atribut | Kegunaan |
|-----|------------------------|---------------|--|
| 1 | Entity Dropdown Select | Form | Menampilkan daftar entitas tokoh |
| 2 | Submit Button | Button | Mengeksekusi request form |
| 3 | Entity Family Tree | Tree | Menampilkan pohon keluarga dari entitas tokoh yang dipilih |

BAB V IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem yang telah dibuat. Proses implementasi dari setiap fungsi pada perangkat lunak Family Tree App akan diuraikan selengkapnya pada bab ini. Implementasi perangkat lunak Family Tree App menggunakan bahasa pemrograman PHP dengan *library* SPARQL Lib.

Agar dapat menampilkan fakta yang belum ada, pada ontologi ini diterapkan sejumlah *rule* yang telah dijelaskan pada Sub subbab 3.3. Setelah itu, dilakukan proses *reasoning* ontologi menggunakan Pellet *Reasoner*. Data model yang didapatkan dari proses *reasoning* kemudian dikonversi menjadi data RDF agar dapat dibaca oleh Apache Jena Fuseki. Apache Jena Fuseki berperan sebagai basis data untuk menyimpan data RDF dalam bentuk triple store. Lalu SPARQL Lib sebagai *query converter* yang dapat mengambil data dari Apache Jena Fuseki untuk ditampilkan di *user interface*.

5.1. Implementasi Proses Ekstraksi, Penggabungan, dan Reasoning

Pada bagian ini dijelaskan secara terperinci mengenai implementasi proses ekstraksi, penggabungan dan *reasoning* yang digunakan untuk menghasilkan data yang akan dipakai. Implementasi dilakukan di dalam kerangka kerja Apache Jena yang ditunjukkan pada Kode Sumber 5.1 sampai dengan Kode Sumber 5.6.

```
String dbJenaFuseki="brits";
String READ_FUSEKI =
"http://localhost:3030/"+dbJenaFuseki;
String OWL_FILE_LOCATION = "D:/The-Tree-of-
Heroes/ontologi_lokal.owl";
File fileRDF = new File("D:\\The-Tree-of-
Heroes\\PreprocessorTA\\result.rdf");
```

Kode Sumber 5.1 Implementasi proses inisialisasi variabel statis

Fungsi kode sumber diatas adalah menginisialisasi variabel-variabel yang akan digunakan di fungsi lainnya. Variabel dbJenaFuseki bernilai nama basis data Apache Jena Fuseki yang digunakan. READ_FUSEKI adalah alamat URL basis data yang digunakan. OWL_FILE_LOCATION adalah ontologi yang dibuat pada subbab 3.3. fileRDF adalah alamat file hasil proses reasoning.

```
FileManager.get().addLocatorClassLoader(Main.class.
getClassLoader());
Model Instances =
FileManager.get().loadModel(READ_FUSEKI);
Instances.read(READ_FUSEKI, "RDF/XML");

Model famonto =
FileManager.get().loadModel(OWL_FILE_LOCATION);
```

Kode Sumber 5.2 Implementasi inisialisasi model Instance dan famonto

Fungsi diatas digunakan untuk memungkinkan aplikasi Jena membaca file RDF yang akan diekstraksi. Fungsi ini juga menginisialisasi model *Instance* dan model family ontologi.

```
FileManager fManager = FileManager.get();
fManager.addLocatorURL();

String[] persons = {
    "Soekarno",
    "Oetari",
    "Inggit Garnasih",
    "Ratna Sari Dewi Soekarno",
    "Haryati",
    "Hartini",
    "Kartini Manoppo",
    "Yurike Sanger",
    "Heldy Djafar",
    "Fatmawati",
    "Megawati Soekarnoputri",
```



```

"Taufiq_Kiemas",
"Puan_Maharani",
"Guruh_Soekarnoputra",
"Guntur_Soekarnoputra",
"Kartika_Sari_Dewi_Soekarno",
"Sukmawati_Soekarnoputri",
"Rachmawati_Soekarnoputri",
"Soeharto",
"Siti_Hartinah",
"Siti_Hardijanti_Rukmana",
"Indra_Rukmana",
"Hutomo_Mandala_Putra",
"Siti_Hediati_Hariyadi",
"Prabowo_Subianto",
"Bambang_Trihatmodjo",
"Sigit_Harjojudanto",
"Siti_Hutami_Endang_Adiningsih",
"Bacharuddin_Jusuf_Habibie",
"Hasri_Ainun_Habibie",
"Ilham_Akbar",
"Thareq_Kemal",
"Abdurrahman_Wahid",
"Yenny_Wahid",
"Inayah_Wulandari",
"Sinta_Nuriyah",
"Susilo_Bambang_Yudhoyono",
"Kristiani_Herrawati",
"Agus_Harimurti_Yudhoyono",
"Annisa_Pohan",
"Edhie_Baskoro_Yudhoyono",
"Siti_Ruby_Aliya_Radjasa",
"Jusuf_Kalla",
"Mufidah_Jusuf_Kalla",
"Solihin_Kalla",
"Joko_Widodo",
"Indreswari",
"Narendraduhita",
"Prajnaparamita",
"Raden_Wijaya",

}

```

```

for (Integer counter = 0; counter <
royalFamilies.length; counter++) {
    Model modelActor =
fManager.loadModel("http://dbpedia.org/data/" +
persons[counter] + ".ttl");

    Instances.add(modelActor);
    System.out.println(persons[counter]);
}

```

Kode Sumber 5.3 Implementasi ekstraksi file RDF tokoh

Fungsi diatas digunakan untuk membaca file RDF yang didownload, memodelkan file tersebut, menambahkan model tersebut ke *modelActor* dan mencetak URL *person* pada setiap iterasinya.

```

final Model union =
ModelFactory.createUnion(Instances,famonto);

```

Kode Sumber 5.4 Implementasi penggabungan model

Fungsi diatas digunakan untuk menggabungkan dua model menjadi satu model. Model-model yang digabungkan adalah *Instances* dan *famonto*, hasil penggabungan modelnya adalah *union*.

```

Reasoner reasoner =
PelletReasonerFactory.theInstance().create();
InfModel reasonedModel =
ModelFactory.createInfModel(reasoner,union);

```

Kode Sumber 5.5 Implementasi proses reasoning

Fungsi diatas digunakan untuk melakukan proses *reasoning* pada suatu model. Model yang direasoning adalah model *union*.

```

if(fileRDF.delete())
{
    System.out.println("The old result.rdf file
deleted successfully");
}
else
{
    System.out.println("Creating new result as RDF
File");
}
PrintStream fileStream = new
PrintStream("result.rdf");
System.setOut(fileStream);

reasonedModel.write( System.out, "RDF/XML" );

```

Kode Sumber 5.6 Implementasi print hasil reasoning sebagai file RDF

Fungsi diatas digunakan untuk mengecek apakah file RDF hasil *reasoning* sudah ada atau belum. Jika sudah ada, maka akan dihapus, dan akan membuat file RDF baru lagi.

5.2. Implementasi Antarmuka Pohon Keluarga

Pada bagian ini dijelaskan secara terperinci mengenai implementasi fungsi-fungsi yang digunakan dalam membangun sistem.

5.2.1. Fungsi Dropdown Select

Fungsi *Dropdown Select* digunakan untuk menampilkan daftar entitas tokoh. Daftar nama tokoh ditampilkan dalam bentuk *form dropdown select*. Untuk menampilkannya, digunakan *method* *get*. Daftar entitas tokoh yang ditampilkan memiliki ciri khusus di basis data triple storenya, yaitu memiliki tipe kelas 'Person'. *Query* digunakan untuk mendapatkan semua tipe 'Person' dari basis data *triple store*. Implementasi fungsi *dropdown select* dapat dilihat pada Kode Sumber 5.7.

```

$data = sparql_get("localhost:3030/brits/query",
                  "PREFIX fam: <http://www.co-
ode.org/roberts/family-tree.owl#>
                  PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                  PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                  PREFIX foaf:
<http://xmlns.com/foaf/0.1/>

                  SELECT DISTINCT ?s
                  WHERE {
                    ?s rdf:type foaf:Person.
                    ?s foaf:name ?name
                  }");

if (!isset($data)) {
    print "<p>Error: " . sparql_errno() . ": " .
sparql_error() . "</p>";
}

?>
<div class="row content">
    <div class="large-up-8">
        <div class="callout">
            <h6 class="subheader">PILIH TOKOH</h6>
            <form method="GET" action="#">
                <div class="input-group">
                    <select class="input-group-
field" name="entity">

                        <?php
                            foreach ($data as $row) {
                                foreach ($data-
>fields() as $name) {
                                    ?>
                                    <option selected
value="<?= $row[$name] ?>"><?= $row[$name]
?></option>

                                    <?php
                                        }
                                    } ?>
                                </select>
                                <div class="input-group-
button">

```

```

                                <input type="submit"
class="button" name="submit" value="Submit">
                                </div>
                                </div>
                                </form>
                                </div>
                                </div>
                                </div>
                                </div>

```

Kode Sumber 5.7 Kode Sumber SPARQL untuk mengambil value bertipe Person dan Fungsi Dropdown Select

5.2.2. Fungsi Get Family

Fungsi *Get Description* digunakan untuk menangkap masukan dari *dropdown select*. *Value* yang ditangkap kemudian berfungsi untuk mengakses informasi data yang bukan merupakan tautan. Fungsi ini memiliki beberapa sub fungsi berdasarkan kegunaan informasi yang diambil dari ontologi.

Get name

Fungsi ini digunakan untuk mendapatkan nama dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari property `foaf:name` yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.8.

```

$data_name =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
                                PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                                PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
                                PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                                PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                                SELECT ?name
                                WHERE {

```

```

                                <' . $selected_val
. '> rdfs:label ?name
                                }
                                LIMIT 1');
if (!isset($data_name)) {
    print "<p>Error: " . sparql_errno() . ": " .
sparql_error() . "</p>";
}
foreach ($data_name as $row) {
    foreach ($data_name->fields() as $field) {
        print "<h3>$row[$field]</h3>";
    }
}

```

Kode Sumber 5.8 Fungsi Get name

Get father

Fungsi ini digunakan untuk mendapatkan ayah dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari property `hasParent` yang memiliki atribut `foaf:gender male`. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.9.

```

$data_fatherIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
                                PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                                PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
                                PREFIX dbpedia-owl:
<http://id.dbpedia.org/ontology/>
                                PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                                PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                                SELECT DISTINCT ?fatherIRI
                                WHERE {
                                    <' . $selected_val .
'> fam:hasParent|dbpedia-owl:parent ?fatherIRI
                                }

```

```

LIMIT 1');
$fatherIRI = "";
foreach ($data_fatherIRI as $row) {
    foreach ($data_fatherIRI->fields() as $field) {
        $fatherIRI = $row[$field];
    }
}
$father = -1;
if (!isset($fatherIRI) || $fatherIRI == '') {
    $father = 0;
} else {
    foreach ($data_fatherIRI as $row) {
        foreach ($data_fatherIRI->fields() as
$field) {
            echo "<ul>";
            echo "<li>";
            $hasName = 0;
            $data_father =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>

PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
PREFIX dbpedia-owl:
<http://id.dbpedia.org/ontology/>
PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?name
WHERE {
    <' . $fatherIRI . '>
rdfs:label ?name
}
LIMIT 1');
        foreach ($data_father as $row) {
            foreach ($data_father->fields() as
$field) {
                echo '<a href="?entity=' .
urlencode($fatherIRI) . "'>' .
str_replace('http://www.dbpedia.org/resource/', "",

```

```

$row[$field]) . '</a>';
        $father = 1;
        $hasName=1;
    }
    }
    if ($hasName==0){
        if (strlen($fatherIRI) > 20)
            $fatherIRIShort=
substr($fatherIRI, 31, 15) . '...';
        echo '<a ' .
urlencode($fatherIRIShort) . '>' . $fatherIRIShort .
'</a>';
    }
    }
}
}

```

Kode Sumber 5.9 Fungsi Get father

Get mother

Fungsi ini digunakan untuk mendapatkan ibu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari property `hasParent` yang memiliki atribut `foaf:gender female`. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.10.

```

$data_motherIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
    PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
    PREFIX dbpedia-owl:
<http://id.dbpedia.org/ontology/>
    SELECT ?motherIRI
    WHERE {
        <' . $selected_val
. '> fam:hasParent|dbpedia-owl:parent ?motherIRI
        FILTER(?motherIRI

```



```

!= <' . $fatherIRI . '>)
    }');
foreach ($data_motherIRI as $row) {
    foreach ($data_motherIRI->fields() as $field) {
        $motherIRI = $row[$field];
    }
}
if (!isset($data_motherIRI) || $data_motherIRI ==
'') {
    echo "<a>Mother Unknown</a>";
} else {
    foreach ($data_motherIRI as $row) {
        foreach ($data_motherIRI->fields() as
$field) {
            if ($father == 0){
                echo "<ul>";
                echo "<li>";
            }
            $hasName =0;
            $data_mother =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
                PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
                PREFIX dbpedia-owl:
<http://id.dbpedia.org/ontology/>
                SELECT ?name
                WHERE {
                    <' . $motherIRI .
'> rdfs:label ?name
                }
                LIMIT 1');

            foreach ($data_mother as $row) {
                foreach ($data_mother->fields() as
$field) {
                    echo "-♥-";
                    echo '<a href="?entity=' .
urlencode($motherIRI) . '>' .
str_replace('http://www.dbpedia.org/resource/', "",

```

```

$row[$field]) . '</a>';
        $hasName=1;
    }
    }
    if ($hasName==0){
        if (strlen($motherIRI) > 20)
            $motherIRI= substr($motherIRI,
31, 15) . '...';
        echo '<a ' . urlencode($motherIRI)
. '>' . $motherIRI . '</a>';
    }
    }
}
}

```

Kode Sumber 5.10 Fungsi Get mother

Get sibling

Fungsi ini digunakan untuk mendapatkan saudara dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari murni *query* SPARQL. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.11.

```

$data_siblingIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
        PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
        PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
        SELECT DISTINCT ?siblingIRI
        WHERE {
            <' . $selected_val .
'> fam:hasParent ?parent1IRI.
            <' . $selected_val .
'> fam:hasParent ?parent2IRI.
                ?parent1IRI
fam:hasChild ?siblingIRI.
                ?parent2IRI
fam:hasChild ?siblingIRI.

```

```

                                FILTER(?siblingIRI
!= <' . $selected_val . '>)
                                FILTER(?parent1IRI
!= ?parent2IRI)
                                }');

$i=0;
foreach ($data_siblingIRI as $rowSiblingIRI) {
    foreach ($data_siblingIRI->fields() as $field)
    {
        $siblingIRI[$i] = $rowSiblingIRI[$field];
        $i++;
    }
}

if (!isset($data_siblingIRI) || $data_siblingIRI ==
'') {
    echo "<ul>";
} else {
    echo "<ul>";
    $i=0;
    foreach ($data_siblingIRI as $rowSiblingIRI) {
        foreach ($data_siblingIRI->fields() as
$field) {
            echo "<li>";
            $hasName = 0;
            $data_sibling =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
                                PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
                                PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
                                SELECT ?siblingname
                                WHERE {
                                    <' . $siblingIRI[$i] .
'> rdfs:label ?siblingname
                                }LIMIT 1');

            foreach ($data_sibling as $row) {
                foreach ($data_sibling->fields() as
$field) {

```

```

        if (strlen($row[$field]) > 20)
            $row[$field] =
substr($row[$field], 0, 15) . '...';
        echo '<a href="?entity=' .
urlencode($siblingIRI[$i]) . '>' .
str_replace('http://www.dbpedia.org/resource/', "",
$row[$field]) . '</a>';
        $hasName = 1;
    }
}
if ($hasName==0){
    if (strlen($siblingIRI[$i]) > 20)
        $siblingIRIShort[$i] =
substr($siblingIRI[$i], 31, 15) . '...';
    echo '<a ' .
urlencode($siblingIRI[$i]) . '>' .
$siblingIRIShort[$i] . '</a>';
}
    $i++;
    echo "</li>";
}
}
}

```

Kode Sumber 5.11 Fungsi Get sibling

Get spouse

Fungsi ini digunakan untuk mendapatkan pasangan suami atau dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari property `isSpouseOf` yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.12.

```

$data_spouseIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>

        PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
        PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>

```

```

SELECT DISTINCT ?spouseIRI
WHERE {
    <' . $selected_val
. '> fam:isSpouseOf ?spouseIRI
    } '};

$i=0;
foreach ($data_spouseIRI as $rowSpouseIRI) {
    foreach ($data_spouseIRI->fields() as $field) {
        $spouseIRI[$i] = $rowSpouseIRI[$field];
        $i++;
    }
}

if (!isset($data_spouseIRI) || $data_spouseIRI ==
'') {
    echo "-♥-<a>Spouse Unknown</a>";
} else {
    $i=0;
    foreach ($data_spouseIRI as $rowSpouseIRI) {
        foreach ($data_spouseIRI->fields() as
$field) {
            if($i>0){ //jika istri lebih dari
satu
                echo "<br>";echo "<br>";echo
"<br>";echo "<br>";echo "<br>";echo "<br>";echo
"<br>";echo "<br>";
                foreach ($data_name as $row) {
                    foreach ($data_name->fields()
as $field) {
                        print "<a style='font-
weight: bold'>$row[$field]</a>";
                    }
                }
            }
            echo "-♥-";
            $hasName = 0;
            $data_spouse =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdofs:
<http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpprop-id:

```

Get child

```
$data_childIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
```

```

tree.owl#>
    PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
    PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
    SELECT DISTINCT ?childIRI
        WHERE {
            <' . $selected_val . '>
fam:hasChild ?childIRI.
            <' . $spouseIRI[$i] . '>
fam:hasChild ?childIRI
        }LIMIT 10');
$i++;
$j=0;
foreach ($data_childIRI as $rowChildIRI) {
    foreach ($data_childIRI->fields() as $field) {
        $childIRI[$j] = $rowChildIRI[$field];
        $j++;
    }
}
$flagChild = 0;
if (isset($data_childIRI)) {
    foreach ($data_childIRI as $rowChild) {
        foreach ($data_childIRI->fields() as $field)
        {
            if($rowChild[$field] == ''){
                $flagChild = 0; //tidak punya anak
            }else $flagChild = 1;
        }
    }
    if($flagChild == 1){
        $cc=0;
        echo "<ul>";
        foreach ($data_childIRI as $rowChildIRI) {
            foreach ($data_childIRI->fields() as
$field) {
                if(isset($childIRI[$cc])) {
                    echo "<li>";
                    $hasName = 0;
                    $data_child =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>

```


Fungsi ini digunakan untuk mendapatkan menantu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari property `isSpouseOf` yang melekat pada anak entitas terpilih. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.14.

```
$data_ChildInLawIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
SELECT ?sbj
WHERE {
    <' . $childIRI[$cc] . '> fam:isSpouseOf ?sbj
}LIMIT 1');
$cc++;
}
foreach ($data_ChildInLawIRI as $rowChildInLawIRI)
{
    foreach ($data_ChildInLawIRI->fields() as
$field) {
        $childInLawIRI = $rowChildInLawIRI[$field];
    }
}
if (!isset($data_ChildInLawIRI) ||
$data_ChildInLawIRI == '') {
    echo "-♥-<a>?</a>";
}else if(isset($data_ChildInLawIRI)){
    foreach ($data_ChildInLawIRI as
$rowChildInLawIRI) {
        foreach ($data_ChildInLawIRI->fields() as
$field) {
            echo "-♥-";
            $hasName = 0;
            $data_ChildInLaw =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
PREFIX foaf:
```

```

<http://xmlns.com/foaf/0.1/>
    PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
    PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
    SELECT ?name
    WHERE {
        <' . $childInLawIRI . '> rdfs:label
?name
        }LIMIT 1');
    foreach ($data_ChildInLaw as
$rowChildInLaw) {
        foreach ($data_ChildInLaw->fields()
as $field) {
            echo '<a href="?entity=' .
urlencode($childInLawIRI) . '>' .
str_replace('http://www.dbpedia.org/resource/', "",
$rowChildInLaw[$field]) . '</a>';
            $hasName =1;
        }
    }
    if ($hasName==0){
        if (strlen($childInLawIRI) > 20)
            $childInLawIRI =
substr($childInLawIRI, 31, 15) . '...';
        echo '<a ' .
urlencode($childInLawIRI) . '>' . $childInLawIRI .
'</a>';
    }
}
}
}

```

Kode Sumber 5.14 Fungsi Get child in law

Get grand child

Fungsi ini digunakan untuk mendapatkan cucu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari anak menantu entitas dengan cara *query* SPARQL. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.15.

```

$data_grandchildIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX

```

```

fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT DISTINCT ?grandchildIRI
WHERE {
  <' . $childInLawIRI . '> dbp:issue ?grandchildIRI
}');
$m=0;
foreach ($data_grandchildIRI as $rowgrandChildIRI)
{
    foreach ($data_grandchildIRI->fields() as
$field) {
        $grandchildIRI[$m] =
$rowgrandChildIRI[$field];
        $m++;
    }
}
$flagGrandChild=0;
if (isset($data_grandchildIRI)) {
    foreach ($data_grandchildIRI as $rowGC) {
        foreach ($data_grandchildIRI->fields() as
$field) {
            if ($rowGC[$field] == '') {
                $flagGrandChild = 0; //tidak punya
                cucu
            } else $flagGrandChild = 1;
        }
    }
    if($flagGrandChild==1){
        $n=0;
        echo "<ul>"; //garis vertikal cucu
        foreach ($data_grandchildIRI as
$rowgrandChildIRI) {
            foreach ($data_grandchildIRI->fields()
as $field) {
                echo "<li>";
                $hasName = 0;
                $data_grandchild =
sparql_get("localhost:3030/brits/query", 'PREFIX

```

```

fam: <http://www.co-ode.org/roberts/family-
tree.owl#>

        PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
        PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
        SELECT ?grandChildName
        WHERE {
                <' . $grandchildIRI[$n] .
'> rdfs:label ?grandChildName
                }LIMIT 1');
        foreach ($data_grandchild as
$rowGC) {
                foreach ($data_grandchild-
>fields() as $field) {
                        if (strlen($rowGC[$field])
> 20)
                                $rowGC[$field] =
substr($rowGC[$field], 0, 15) . '...';
                                echo '<a href="?entity=' .
urlencode($grandchildIRI[$n]) . '>' .
str_replace('http://www.dbpedia.org/resource/', "",
$rowGC[$field]) . '</a>';
                                $hasName = 1;
                        }
                }
                if ($hasName==0){
                        if (strlen($grandchildIRI[$n])
> 20)
                                $grandchildIRI[$n] =
substr($grandchildIRI[$n], 31, 15) . '...';
                                echo '<a ' .
urlencode($grandchildIRI[$n]) . '>' .
$grandchildIRI[$n] . '</a>';
                        }
                }
        }
}

```

Kode Sumber 5.15 Fungsi Get grand child

Get grand child in law

Fungsi ini digunakan untuk mendapatkan pasangan cucu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari anak menantu entitas dengan cara *query* SPARQL. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.16.

```
$data_GrandChildInLawIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
SELECT ?sbj
WHERE {
    <' . $grandchildIRI[$n] . '>
fam:isSpouseOf ?sbj.
    ?sbj rdfs:label ?name
}LIMIT 1');
foreach ($data_GrandChildInLawIRI as
$rowGrandChildInLawIRI) {
    foreach ($data_GrandChildInLawIRI->fields() as
$field) {
        $grandChildInLawIRI =
$rowGrandChildInLawIRI[$field];
    }
}
if (!isset($data_GrandChildInLawIRI) ||
$data_GrandChildInLawIRI == '') {
    echo "-♥-<a>?</a>";
}else if(isset($data_GrandChildInLawIRI)){
    foreach ($data_GrandChildInLawIRI as
$rowGrandChildInLawIRI) {
        foreach ($data_GrandChildInLawIRI->fields()
as $field) {
            echo "-♥-";
            $hasName = 0;
            $data_GrandChildInLaw =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
```

```

        PREFIX foaf:
<http://xmlns.com/foaf/0.1/> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
        PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
        SELECT ?name
        WHERE {
            <' . $grandchildIRI[$n] . '>
fam:isSpouseOf ?sbj.
            ?sbj rdfs:label ?name
        }LIMIT 1');

        foreach ($data_GrandChildInLaw as
$rowGrandChildInLaw) {
            foreach ($data_GrandChildInLaw-
>fields() as $field) {
                echo '<a href="?entity=' .
urlencode($grandChildInLawIRI) . '>' .
str_replace('http://www.dbpedia.org/resource/', '',
$rowGrandChildInLaw[$field]) . '</a>';
                $hasName = 1;
            }
        }
        if($hasName==0) {
            if
(strlen($rowGrandChildInLaw[$field]) > 20)
                $rowGrandChildInLaw[$field] =
substr($rowGrandChildInLaw[$field], 31, 15) . '...';
            echo '<a ' .
urlencode($rowGrandChildInLaw[$field]) . '>' .
$rowGrandChildInLaw[$field] . '</a>';
        }
    }
}

```

Kode Sumber 5.16 get grand child in law

Get great grand child

Fungsi ini digunakan untuk mendapatkan cicit dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari anak menantu

entitas dengan cara *query* SPARQL. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.17.

```
$data_greatGrandChildIRI =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/> PREFIX
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
SELECT DISTINCT ?greatgrandchildIRI
WHERE {
    <' . $grandChildInLawIRI . '> fam:hasChild
?greatgrandchildIRI.
    <' . $grandChildIRI[$n] . '> fam:hasChild
?greatgrandchildIRI
    }');
$m=0;
foreach ($data_greatGrandChildIRI as
$rowgreatGrandChildIRI) {
    foreach ($data_greatGrandChildIRI->fields() as
$field) {
        $greatGrandChildIRI[$m] =
$rowgreatGrandChildIRI[$field];
        $m++;
    }
}
$flagGreatGrandChild = 0;
if(isset($data_greatGrandChildIRI)){
    foreach ($data_greatGrandChildIRI as $rowGGC) {
        foreach ($data_greatGrandChildIRI->fields()
as $field) {
            if ($rowGGC[$field] == '') {
                $flagGreatGrandChild = 0; //tidak
punya cicit
            } else $flagGreatGrandChild = 1;
        }
    }
    if($flagGreatGrandChild == 1){
        $p=0;
        echo "<ul>";
        foreach ($data_greatGrandChildIRI as
```

```

$rowGreatGrandChildIRI) {
    foreach ($data_greatGrandChildIRI-
>fields() as $field) {
        echo "<li>";
        $hasName = 0;
        $data_greatGrandChild =
sparql_get("localhost:3030/brits/query", 'PREFIX
fam: <http://www.co-ode.org/roberts/family-
tree.owl#>
        PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
        PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
        PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>SELECT ?name
        WHERE {
            <' . $greatGrandChildIRI[$p] .
'> rdfs:label ?name
        }LIMIT 1');
        foreach ($data_greatGrandChild as
$rowGGC) {
            foreach ($data_greatGrandChild-
>fields() as $field) {
                if (strlen($rowGGC[$field])
> 20)
                    $rowGGC[$field] =
substr($rowGGC[$field], 0, 15) . '...';
                echo '<a href="?entity=' .
urlencode($greatGrandChildIRI[$p]) . '>' .
str_replace('http://www.dbpedia.org/resource/', "",
$rowGGC[$field]) . '</a>';
                $hasName = 1;
            }
        }
        if ($hasName==0){
            if
(strlen($greatGrandChildIRI[$p]) > 20)
                $greatGrandChildIRI[$p] =
substr($greatGrandChildIRI[$p], 31, 15) . '...';
            echo '<a ' .
urlencode($greatGrandChildIRI[$p]) . '>' .
$greatGrandChildIRI[$p] . '</a>';
        }
        $p++;
    }
}

```



```

        echo "</li>";
    }
}
echo "</ul>";
}
}

```

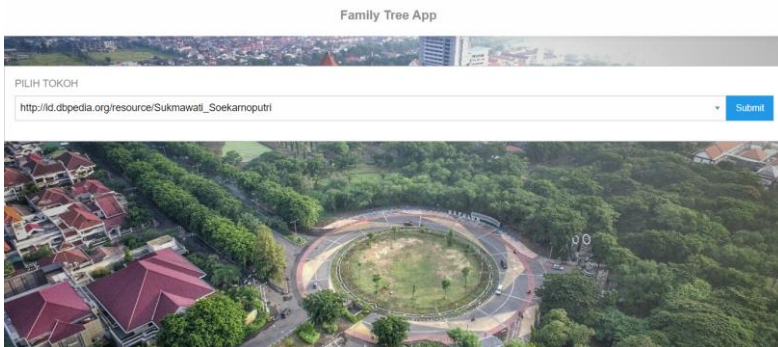
Kode Sumber 5.17 Get great grand child

5.3. Implementasi Antarmuka Pengguna

Implementasi tampilan antarmuka pengguna pada *browser* Google Chrome dilakukan dengan menggunakan dukungan aplikasi XAMPP. XAMPP berfungsi untuk menjalankan aplikasi web dengan server Apache. Berikut ini akan dijelaskan mengenai implementasi tampilan antarmuka pengguna yang terdapat pada Family Tree App.

5.3.1. Implementasi Halaman Utama

Halaman ini merupakan implementasi halaman utama dari rancangan antarmuka yang telah dijelaskan pada Subbab 4.2. Halaman utama hanya menampilkan kolom *dropdown select* yang dapat digunakan oleh pengguna untuk memilih tokoh. Daftar entitas tokoh yang ditampilkan hanya tokoh utama yang digunakan dalam pengerjaan tugas akhir ini. Tampilan antarmuka halaman utama ini dapat dilihat pada Gambar 5.1.

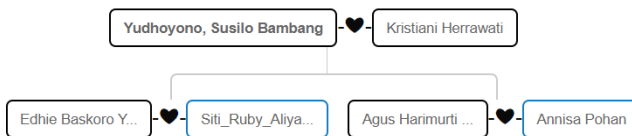


Gambar 5.1 Implementasi Antarmuka Halaman Utama

5.3.2. Implementasi Halaman Pohon Keluarga

Halaman ini merupakan implementasi halaman informasi untuk menampilkan data entitas yang dipilih dari rancangan antarmuka yang telah dijelaskan pada Subbab 4.2.

Yudhoyono, Susilo Bambang



Gambar 5.2 Implementasi Antarmuka Halaman Pohon Keluarga

Tampilan antarmuka halaman informasi ini dapat dilihat pada Gambar 5.2. Dan seperti yang dijelaskan di Batasan Masalah, batas generasi pendahulu adalah orang tua, sedangkan batas keturunan adalah cicit.

BAB VI

PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada ontologi yang dikembangkan. Pengujian yang dilakukan adalah pengujian ontologi, pengujian perbandingan data, dan pengujian kompleksitas ontologi. Pengujian ontologi mengacu pada perancangan *rule* pada Sub subbab Semantic Web Rule Language (SWRL). Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

6.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor : Intel Core i7-6700
CPU @ 3.90GHz
Memori : 16.00 GB
Jenis Perangkat : Laptop
Sistem Operasi : Microsoft Windows 10 64-bit
Protege : Protege 5.2
Reasoner : Pellet
Browser : Google Chrome
Web Server : Apache 2
Database Server: Apache Jena Fuseki

6.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian perbandingan data dilakukan dengan membandingkan data asli DBpedia sebelum dilakukan proses *reasoning* dengan data yang dihasilkan dari proses *reasoning* yang ditampilkan di pohon keluarga Family Tree App.

Pengujian *reasoning* meliputi tiga sub pengujian, yaitu pengujian fitur *hasSpouse*, *hasChild*, dan *hasParent*. Pengujian

visualisasi meliputi tiga sub pengujian, yaitu visualisasi hubungan orang tua-anak, suami-istri, dan saudara kandung.

6.2.1. Pengujian Reasoning

| Kode Uji | Nama Pengujian | Keterangan |
|----------|----------------------------|--|
| R-01 | Pengujian relasi hasSpouse | Untuk mengujikan ketepatan axiom hasSibling |
| R-02 | Pengujian relasi hasChild | Untuk mengujikan ketepatan axiom hasChildInLaw |
| R-03 | Pengujian relasi hasParent | Untuk mengujikan ketepatan axiom hasGrandChild |

6.2.2. Pengujian Visualisasi

| Kode Uji | Nama Pengujian | Keterangan |
|----------|--|--|
| V-01 | Kondisi tidak punya anak | Untuk mengujikan tampilan saat URL tokoh tidak punya anak |
| V-02 | Kondisi memiliki anak | Untuk mengujikan tampilan saat URL tokoh memiliki anak |
| V-03 | Kondisi memiliki cucu | Untuk mengujikan tampilan saat URL tokoh memiliki cucu |
| V-04 | Kondisi memiliki cicit | Untuk mengujikan tampilan saat URL tokoh memiliki cicit |
| V-05 | Kondisi memiliki pasangan lebih dari satu | Untuk mengujikan tampilan saat URL tokoh memiliki pasangan lebih dari satu |
| V-06 | Kondisi relasi tidak memiliki property identitas | Untuk mengujikan tampilan saat URL tokoh tidak memiliki nama atau label |

6.3. Hasil Pengujian

Pada subbab ini akan dipaparkan hasil dari pengujian-pengujian yang telah dilakukan. Hasil yang diberikan meliputi hasil pengujian *reasoning* dan hasil pengujian visualisasi yang telah dijelaskan pada Subbab 6.2.

6.3.1. Pengujian Reasoning

Tabel 6.1 Pengujian Reasoning property hasSpouse

| | |
|---------------------------|--|
| Kode Uji | R-01 |
| Nama Pengujian | Pengujian axiom hasSpouse |
| Kondisi sebelum reasoning | Lampiran Gambar 1 |
| Kondisi sesudah reasoning | Lampiran Gambar 2 |
| Keterangan | Property hasSpouse bersifat <i>symmetrical</i> . Property ini berlaku kepada kedua subjek dan objek. Jika x hasSpouse y, maka y hasSpouse x. |
| Nilai | Berhasil |

Tabel 6.2 Pengujian Reasoning property hasChild

| | |
|---------------------------|--------------------------|
| Kode Uji | R-02 |
| Nama Pengujian | Pengujian axiom hasChild |
| Kondisi sebelum reasoning | Lampiran Gambar 3 |
| Kondisi sesudah reasoning | Lampiran Gambar 4 |

| | |
|------------|--|
| Keterangan | Ekuivalensi property <i>dbo:child</i> , <i>dbp:children</i> , dan <i>dbp:issue</i> menjadi <i>hasChild</i> |
| Nilai | Berhasil |

Tabel 6.3 Pengujian Reasoning property *hasParent*

| | |
|---------------------------|--|
| Kode Uji | R-03 |
| Nama Pengujian | Pengujian axiom <i>hasParent</i> |
| Kondisi sebelum reasoning | Lampiran Gambar 5 |
| Kondisi sesudah reasoning | Lampiran Gambar 6 |
| Keterangan | Property <i>hasParent</i> bersifat <i>inverse</i> dari property <i>hasChild</i> . Jika <i>x hasChild y</i> , maka <i>y hasParent x</i> . |
| Nilai | Berhasil |

6.3.2. Pengujian Visualisasi

Tabel 6.4 Pengujian Visualisasi tanpa anak

| | |
|-------------------|--|
| Kode Uji | V-01 |
| Nama Pengujian | Kondisi tidak punya anak |
| Hasil Visualisasi | Lampiran Gambar 7 |
| Keterangan | Contoh pohon keluarga Oetari dan Soekarno yang tidak memiliki property anak. |
| Nilai | Berhasil |

Tabel 6.5 Pengujian Visualisasi memiliki anak

| | |
|----------------|---|
| Kode Uji | V-02 |
| Nama Pengujian | Kondisi memiliki anak |
| Hasil | Lampiran Gambar 8 |
| Keterangan | Contoh pohon keluarga Susilo Bambang Yudhoyono yang memiliki dua anak (Agus Harimurti Yudhoyono dan Edhie Baskoro Yudhoyono). |
| Nilai | Berhasil |

Tabel 6.6 Pengujian Visualisasi memiliki cucu

| | |
|----------------|---|
| Kode Uji | V-03 |
| Nama Pengujian | Kondisi memiliki cucu |
| Hasil | Lampiran Gambar 9 |
| Keterangan | Contoh pohon keluarga Fatmawati yang memiliki relasi hingga ke cucu (Puan Maharani) |
| Nilai | Berhasil |

Tabel 6.7 Pengujian Visualisasi memiliki cicit

| | |
|----------------|---|
| Kode Uji | V-04 |
| Nama Pengujian | Kondisi memiliki cicit |
| Hasil | Lampiran Gambar 10 |
| Keterangan | Contoh pohon keluarga Ratu Elizabeth II dari <i>dataset</i> Royal Family yang memiliki relasi cicit (Prince George dan Princess Charlotte). |
| Nilai | Berhasil |

Tabel 6.8 Pengujian Visualisasi memiliki banyak pasangan

| | |
|----------------|--|
| Kode Uji | V-05 |
| Nama Pengujian | Kondisi memiliki pasangan lebih dari satu |
| Hasil | Lampiran Gambar 11 |
| Keterangan | Contoh pohon keluarga Raden Wijaya yang memiliki relasi <i>spouse</i> lebih dari satu. |
| Nilai | Berhasil |

Tabel 6.9 Pengujian Visualisasi tidak memiliki identitas

| | |
|----------------|--|
| Kode Uji | V-06 |
| Nama Pengujian | Kondisi relasi tidak memiliki property identitas |
| Hasil | Lampiran Gambar 12 |
| Keterangan | Contoh pohon keluarga Anisa Pohan yang memiliki relasi ke URL yang tidak memiliki property <i>label</i> atau <i>name</i> . |
| Nilai | Berhasil |

6.4. Evaluasi Hasil Pengujian

Berdasarkan pengujian *reasoning* dan visualisasi pada subbab 6.3.1 dan 6.3.2, semua pengujian memberikan hasil yang sesuai dengan skenario yang direncanakan. Rangkuman mengenai hasil uji *reasoning* dan visualisasi dapat dilihat di Tabel 6.10.

Tabel 6.10 Evaluasi Pengujian

| Kode Uji | Hasil Uji |
|----------|-----------|
| R-01 | Berhasil |
| R-02 | Berhasil |
| R-03 | Berhasil |

| | |
|------|----------|
| V-01 | Berhasil |
| V-02 | Berhasil |
| V-03 | Berhasil |
| V-04 | Berhasil |
| V-05 | Berhasil |
| V-06 | Berhasil |

Berdasarkan data pada Tabel 6.10, dapat disimpulkan bahwa sistem yang dibuat bekerja sesuai dengan yang diharapkan.

(Halaman ini sengaja dikosongkan)

BAB VII

KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan dan saran mengenai hal-hal yang masih bisa untuk dikembangkan dari tugas akhir ini.

7.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Data property yang dimiliki oleh *Family Relationship Ontology* dapat digunakan pada domain tokoh sejarah Indonesia.
2. Studi kasus visualisasi pohon keluarga tokoh sejarah Indonesia mampu dimodelkan dan digabungkan dengan model ontologi dengan Apache Jena serta bisa melakukan proses *reasoning* dengan Pellet Reasoner.
3. Aplikasi untuk visualisasi pohon keluarga tokoh sejarah Indonesia dapat dikembangkan dengan library SPARQL Lib yang mampu menghubungkan basis data Apache Jena Fuseki dengan perangkat lunak yang menggunakan bahasa pemrograman PHP.

7.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

- 1) Penggunaan perangkat uji coba dengan spesifikasi kapasitas memori yang lebih besar agar waktu yang

dibutuhkan untuk proses *export inferenced axiom* lebih cepat.

- 2) Penambahan visualisasi generasi pendahulu dan penerus.
- 3) Fitur penambahan data secara dinamis.

DAFTAR PUSTAKA

- [1] L. Burmark, "Visual literacy: What you get is what you see," 2008.
- [2] S. J. Miller, Introduction to Ontology Concepts and Terminology, Lisbon, Portugal: University of Wisconsin-Milwaukee, 2013.
- [3] M. A. Ramadhanie, Penerapan Ontologi Objek Pembelajaran Untuk Kebutuhan Personalisasi E-Learning Berbasis Semantic Web, Depok: Universitas Indonesia, 2009.
- [4] S. Nikles, "Expressiveness of Enterprise Modelling Languages," University of Applied Sciences Northwestern Switzerland, Basel, 2010.
- [5] C. Candrabiantara, D. O. Siahaan and U. L. Yuhana, "Rancang Bangun Aplikasi Visualisasi Silsilah Keluarga Berbasis Ontologi," *Jurnal Teknik POMITS*, vol. 2, no. 1, 2013.
- [6] "Professor Robert Stevens," [Online]. Available: <http://www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl> . [Accessed 06 January 2016].
- [7] "Design Goals of the Java™ Programming Language," Oracle, 1 January 1999. [Online]. Available: <https://www.oracle.com/technetwork/java/intro-141325.html>. [Accessed 30 6 2019].
- [8] A. U. D. Shafiq, "JavaOne 2013 Review: Java Takes on the Internet of Things," Oracle, 19 April 2016. [Online]. Available: <https://www.imarslan.com/javaone-2013-review-java-takes-on-the-internet-of-things>. [Accessed 30 June 2019].
- [9] N. Kompridis, "So We Need Something Else for Reason to Mean," *International Journal of Philosophical Studies*, no. 8, p. 271–295, 2000.

- [10] R. J. Sternberg, *Cognitive Psychology*, Belmont, CA: Wadsworth, 2009.
- [11] G. Meditskos and N. Bassiliades, "A Rule-Based Object-Oriented OWL Reasoner," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 3, pp. 397-410, 2008.
- [12] B. Parsia and E. Sirin, "Pellet: An OWL DL Reasoner," University of Maryland, College Park.
- [13] A. Mishra, "Critical Comparison Of PHP And ASP.NET For Web Development," *International Journal of Scientific & Technology Research*, vol. 3, no. 7, pp. 331-333, July 2014.
- [14] Z. T. Inc., "An overview on PHP," Zend The PHP Company, 2007.
- [15] T. Point, "PHP Hypertext Preprocessor," 2015. [Online]. Available: www.tutorialspoint.com/php/php_tutorial.pdf. [Accessed 10 June 2016].
- [16] D. Wu and A. Håkansson, "A Method of Identifying Ontology Domain," *Procedia Computer Science*, vol. 35, pp. 504-513, 2014.
- [17] N. Bullough, "Wangsa Rajasa Keluarga Kerajaan Singasari dan Majapahit," in *Historic East Java: Remains in Stone*, Jakarta, ADLine Communications, 1995, p. 116–117.
- [18] D. L. McGuinness and F. v. Harmelen, "OWL Web Ontology Language Overview," [Online]. Available: <https://www.w3.org/TR/owl-features/>. [Accessed 06 January 2016].
- [19] "XML," [Online]. Available: <https://en.wikipedia.org/wiki/XML>. [Accessed 10 June 2016].
- [20] M. Saralita, "Pencarian Relasi Antar Tokoh Sejarah Indonesia Menggunakan Ontologi," 2016.
- [21] Kate Samuelson And Raisa Bruner, "Royal Family Tree," TIME, 06 05 2019. [Online]. Available:

<https://time.com/5238004/royal-family-tree/>. [Accessed 23 06 2019].

- [22] Wikipedia, "Daftar Raja di Jawa," 09 06 2019. [Online]. Available:
https://id.wikipedia.org/wiki/Daftar_raja_di_Jawa.
[Accessed 26 06 2019].

(Halaman ini sengaja dikosongkan)

LAMPIRAN

Lampiran Tabel 1 Daftar URL data tokoh

| Nama | URL |
|----------------------------|---|
| Soekarno | http://id.dbpedia.org/data/Soekarno |
| Oetari | http://id.dbpedia.org/data/Oetari |
| Inggit Garnasih | http://id.dbpedia.org/data/Inggit_Garnasih |
| Ratna Sari Dewi Soekarno | http://id.dbpedia.org/data/Ratna_Sari_Dewi_Soekarno |
| Haryati | http://id.dbpedia.org/data/Haryati |
| Hartini | http://id.dbpedia.org/data/Hartini |
| Kartini Manoppo | http://id.dbpedia.org/data/Kartini_Manoppo |
| Yurike Sanger | http://id.dbpedia.org/data/Yurike_Sanger |
| Heldy Djafar | http://id.dbpedia.org/data/Heldy_Djafar |
| Fatmawati | http://id.dbpedia.org/data/Fatmawati |
| Megawati Soekarnoputri | http://id.dbpedia.org/data/Megawati_Soekarnoputri |
| Taufiq Kiemas | http://id.dbpedia.org/data/Taufiq_Kiemas |
| Puan Maharani | http://id.dbpedia.org/data/Puan_Maharani |
| Guruh Soekarnoputra | http://id.dbpedia.org/data/Guruh_Soekarnoputra |
| Guntur Soekarnoputra | http://id.dbpedia.org/data/Guntur_Soekarnoputra |
| Kartika Sari Dewi Soekarno | http://id.dbpedia.org/data/Kartika_Sari_Dewi_Soekarno |
| Sukmawati Soekarnoputri | http://id.dbpedia.org/data/Sukmawati_Soekarnoputri |
| Rachmawati Soekarnoputri | http://id.dbpedia.org/data/Rachmawati_Soekarnoputri |
| Soeharto | http://id.dbpedia.org/data/Soeharto |
| Siti Hartinah | http://id.dbpedia.org/data/Siti_Hartinah |
| Siti Hardijanti Rukmana | http://id.dbpedia.org/data/Siti_Hardijanti_Rukmana |
| Indra Rukmana | http://id.dbpedia.org/data/Indra_Rukmana |

| | |
|-------------------------------|---|
| Hutomo Mandala Putra | http://id.dbpedia.org/data/Hutomo_Mandala_Putra |
| Siti Hediati Hariyadi | http://id.dbpedia.org/data/Siti_Hediati_Hariyadi |
| Prabowo Subianto | http://id.dbpedia.org/data/Prabowo_Subianto |
| Bambang Trihatmodjo | http://id.dbpedia.org/data/Bambang_Trihatmodjo |
| Sigit Harjojudanto | http://id.dbpedia.org/data/Sigit_Harjojudanto |
| Siti Hutami Endang Adiningsih | http://id.dbpedia.org/data/Siti_Hutami_Endang_Adiningsih |
| Bacharuddin Jusuf Habibie | http://id.dbpedia.org/data/Bacharuddin_Jusuf_Habibie |
| Hasri Ainun Habibie | http://id.dbpedia.org/data/Hasri_Ainun_Habibie |
| Ilham Akbar | http://id.dbpedia.org/data/Ilham_Akbar |
| Thareq Kemal | http://id.dbpedia.org/data/Thareq_Kemal |
| Abdurrahman Wahid | http://id.dbpedia.org/data/Abdurrahman_Wahid |
| Yenny Wahid | http://id.dbpedia.org/data/Yenny_Wahid |
| Inayah Wulandari | http://id.dbpedia.org/data/Inayah_Wulandari |
| Sinta Nuriyah | http://id.dbpedia.org/data/Sinta_Nuriyah |
| Susilo Bambang Yudhoyono | http://id.dbpedia.org/data/Susilo_Bambang_Yudhoyono |
| Kristiani Herrawati | http://id.dbpedia.org/data/Kristiani_Herrawati |
| Agus Harimurti Yudhoyono | http://id.dbpedia.org/data/Agus_Harimurti_Yudhoyono |
| Annisa Pohan | http://id.dbpedia.org/data/Annisa_Pohan |
| Edhie Baskoro Yudhoyono | http://id.dbpedia.org/data/Edhie_Baskoro_Yudhoyono |

| | |
|-------------------------|---|
| Siti Ruby Aliya Radjasa | http://id.dbpedia.org/data/Siti_Ruby_Aliya_Radjasa |
| Jusuf Kalla | http://id.dbpedia.org/data/Jusuf_Kalla |
| Mufidah Jusuf Kalla | http://id.dbpedia.org/data/Mufidah_Jusuf_Kalla |
| Solihin Kalla | http://id.dbpedia.org/data/Solihin_Kalla |
| Joko Widodo | http://id.dbpedia.org/data/Joko_Widodo |

Lampiran Gambar 1 Individu Fatmawati sebelum reasoning

family-tree (http://www.co-ode.org/robots/family-tree.owl) : [D:\The-Tree-of-Heroes\Preprocessor\A\Result Tokoh Sejarah Dbpedia ID 2 sebelum Reasoning]

File Edit View Reasoner Tools Refactor Window Help

family-tree (http://www.co-ode.org/robots/family-tree.owl) DL Query

Active Ontology

Annotation properties Datatypes Individuals Classes Object properties Data properties

Individuals: Fatmawati

Abdurrahman_Wahid Agus_Harimurti_Yudhoyono Annisa_Pohan Aulia_Pohan Bacharuddin_Jusuf_Habibie Bambang_Trihatmodjo Bayu_Soekarnoputra Edhie_Baskoro_Yudhoyono Fatmawati Soekarno Guntur_Soekarnoputra Guruh_Soekarnoputra Gusriyenoza_Sabina_Padmawati Hartini Haryati Hasri_Annun_Habibie Hassan_Gamal_Ahmad_Hasan Heldy_Djafar Hutomo_Mandala_Putra Ilham_Akbar Inayah_Wulandari Indra_Rukmana Inggit_Garnasih Joko_Widodo Jusuf_Kalla Kartika_Sari_Dewi_Soekarno Kartini_Manoppo

Annotations +

rdf:type [language: id]

Annotations: Fatmawati

Individual Annotations Individual Usage

Types +

dbo:Agent dbo:OfficeHolder dbo:Person owl:Thing schema:Person

Same individual As +

Different individuals +

Property assertions: Fatmawati

Object property assertions +

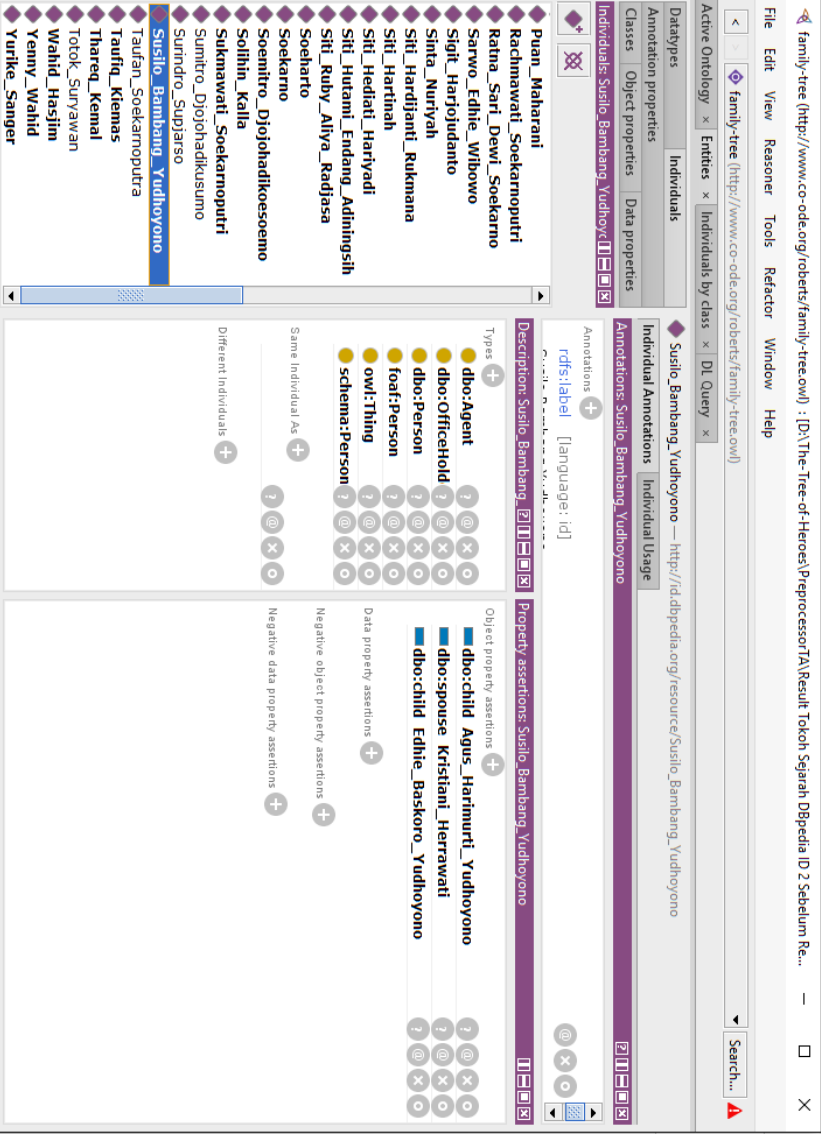
dbo:child Megawati_Soekarnoput Guntur_Soekarnoputra Guruh_Soekarnoputra dbo:child Sukmawati_Soekarnopi dbo:child Rachmawati_Soekarnop

Data property assertions +

Negative object property assertions +

Negative data property assertions +

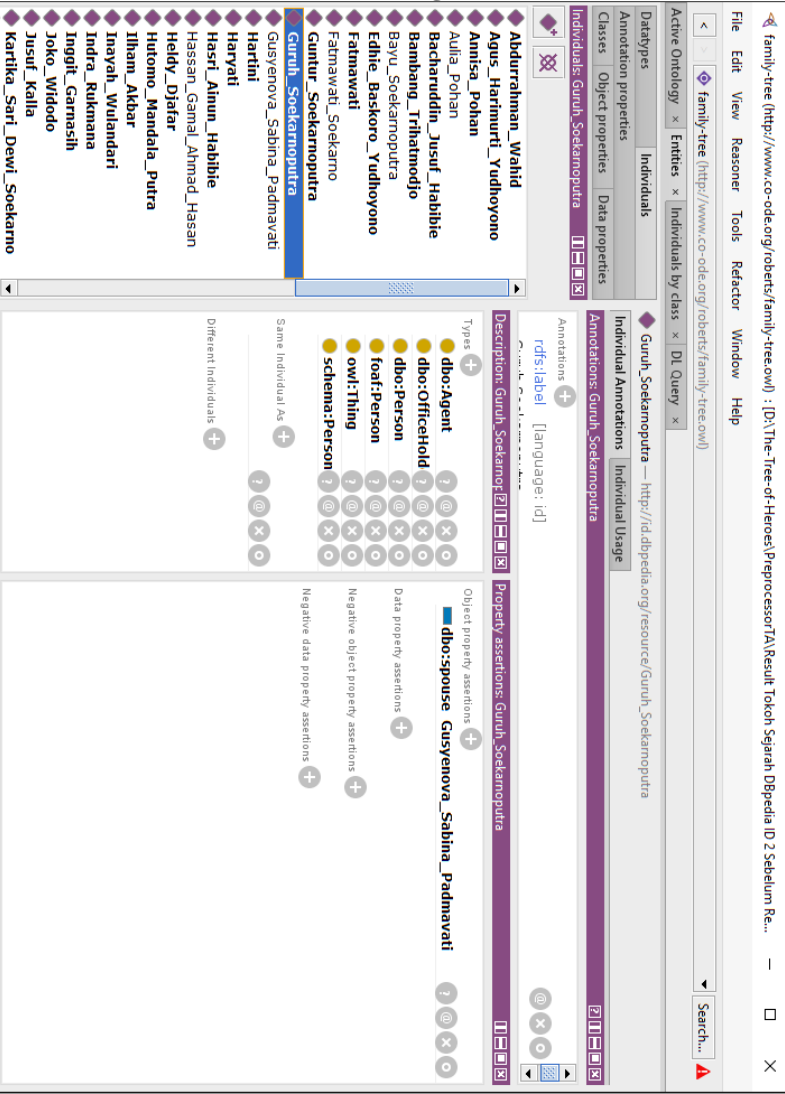
Lampiran Gambar 3 Individu Susilo Bambang Yudhoyono sebelum reasoning



Lampiran Gambar 4 Individu Susilo Bambang Yudhoyono setelah reasoning

[illegible]

Lampiran Gambar 5 Individu Guruh Soekarnoputra sebelum reasoning



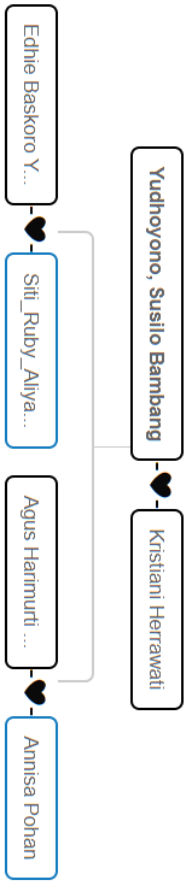
Lampiran Gambar 7 Kasus tidak memiliki anak

Oetari

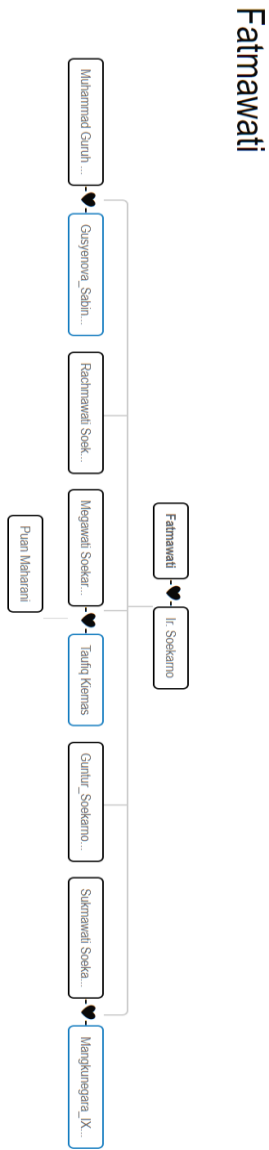


Lampiran Gambar 8 Kasus memiliki anak

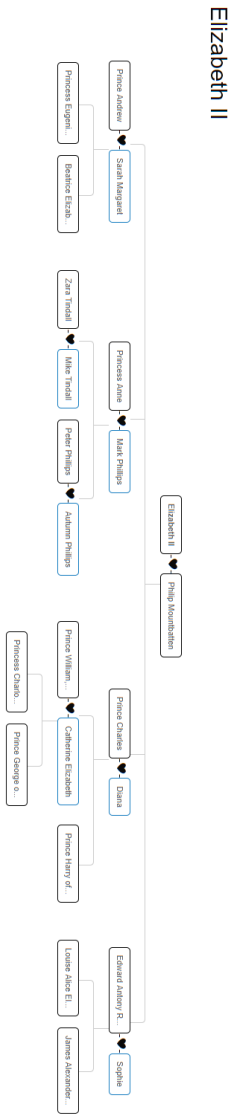
Yudhoyono, Susilo Bambang



Lampiran Gambar 9 Kasus memiliki cucu



Lampiran Gambar 10 Kasus memiliki cicit



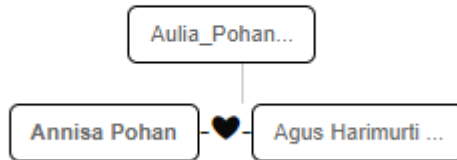
Lampiran Gambar 11 Kasus memiliki pasangan lebih dari satu

Raden Wijaya



Lampiran Gambar 12 Kasus memiliki relasi tanpa nama

Annisa Pohan



(Halaman ini sengaja dikosongkan)

BIODATA PENULIS



Faiq, lahir pada tanggal 8 Juli 1997 di Kediri. Penulis pernah menempuh pendidikan di SDIT Nurul Islam Pare (2003-2007) SD Islam Ar-Robithoh (2007-2009), MTs Negeri 1 Pare (2009-2012), dan SMA Negeri 2 Kediri (2013-2015).

Saat ini penulis sedang menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya

di departemen Informatika Fakultas Teknologi Informasi dan Komunikasi angkatan tahun 2015. Dalam menyelesaikan pendidikan S1 penulis mengambil bidang minat Manajemen Informasi (MI). Penulis juga pernah terlibat aktif dalam organisasi kemahasiswaan serta kepanitiaan selama perkuliahan, antara lain staff Departemen Hubungan Luar di Himpunan Mahasiswa Teknik Computer-Informatika ITS, dan menjadi kabinet dalam organisasi BEM FTIK ITS. Di sisi profesional, penulis pernah melakukan kerja praktek di Blibli.com, Direktorat Pengembangan Teknologi dan Sistem Informasi (DPTSI) – ITS, dan PT. Aku Pintar Indonesia. Penulis dapat dihubungi melalui alamat *email* karyoutomoo@gmail.com.