

**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)***



**FINAL PROJECT - KI141502**

**FAMILY TREE VISUALIZATION DESIGN 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 Sholicah A., S.Kom., M.Sc. ....  
NIP. 19850826 201504 2 002 (Pembimbing 2)

**SURABAYA**  
**JUNI, 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 monarch figures of Indonesia, are both proofs 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 and monarch figures and to represent them as a family tree.*

*The steps required to complete this thesis is first completing the figure's 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, Indonesian kingdoms, 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 dan Djohan, serta Tegar, Naufal dan Arya.
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 .....	1
ABSTRACT .....	3
KATA PENGANTAR.....	5
DAFTAR ISI .....	7
DAFTAR GAMBAR .....	xv
DAFTAR TABEL .....	xvii
DAFTAR KODE SUMBER .....	xviii
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. Ontologi.....	7
2.3. DBpedia.....	12
2.4. Semantic Web Rule Language (SWRL).....	14
2.5. Family Relationships Ontology .....	16
2.6. SPARQL.....	17
2.7. Apache Jena Fuseki .....	18
2.8. Pellet Reasoner .....	20
2.9. PHP.....	22
2.10. Java.....	23
2.11. SPARQL Lib .....	24
2.12. Apache Jena.....	25
3 BAB III METODOLOGI PEMECAHAN MASALAH.....	27
3.1. Analisis Data .....	28
3.1.1. Analisis Data dari DBpedia.....	29

3.2.	Ekstraksi Data Sebagai Model.....	32
3.3.	Pembuatan Ontologi .....	34
3.4.	Penggabungan Model Data dan Model Family Relationship Ontology .....	38
3.5.	Reasoning pada Model Gabungan .....	42
3.6.	Penampilan Data.....	43
4	BAB IV ANALISIS DAN PERANCANGAN SISTEM.....	45
4.1.	Analisis .....	45
4.1.1.	Cakupan Permasalahan.....	45
4.1.2.	Deskripsi Umum Sistem.....	45
4.1.3.	Spesifikasi Kebutuhan Perangkat Lunak.....	48
4.1.4.	Aktor.....	48
4.1.5.	Kasus Penggunaan.....	49
4.2.	Perancangan Antarmuka Pengguna .....	51
5	BAB V IMPLEMENTASI .....	53
5.1.	Implementasi Proses Ekstraksi, Penggabungan, dan Reasoning .....	53
5.2.	Implementasi Antarmuka Pohon Keluarga.....	56
5.2.1.	Fungsi Dropdown Select .....	56
5.2.2.	Fungsi Get Family .....	58
5.3.	Implementasi Antarmuka Pengguna.....	78
5.3.1.	Implementasi Tampilan Halaman Utama .....	78
5.3.2.	Implementasi Tampilan Halaman Pohon Keluarga .....	79
6	BAB VI PENGUJIAN DAN EVALUASI .....	80
6.1.	Lingkungan Pengujian.....	81
6.2.	Skenario Pengujian .....	81
6.2.1.	Pengujian Reasoning .....	82
6.2.2.	Pengujian Visualisasi .....	82
6.3.	Hasil Pengujian.....	82
6.3.1.	Pengujian Reasoning .....	83
6.3.2.	Pengujian Visualisasi .....	84
7	BAB VII KESIMPULAN DAN SARAN .....	87
1.	87	
7.1.	Kesimpulan.....	87




7.2.   Saran.....87

DAFTAR PUSTAKA.....89

LAMPIRAN.....91

8     93



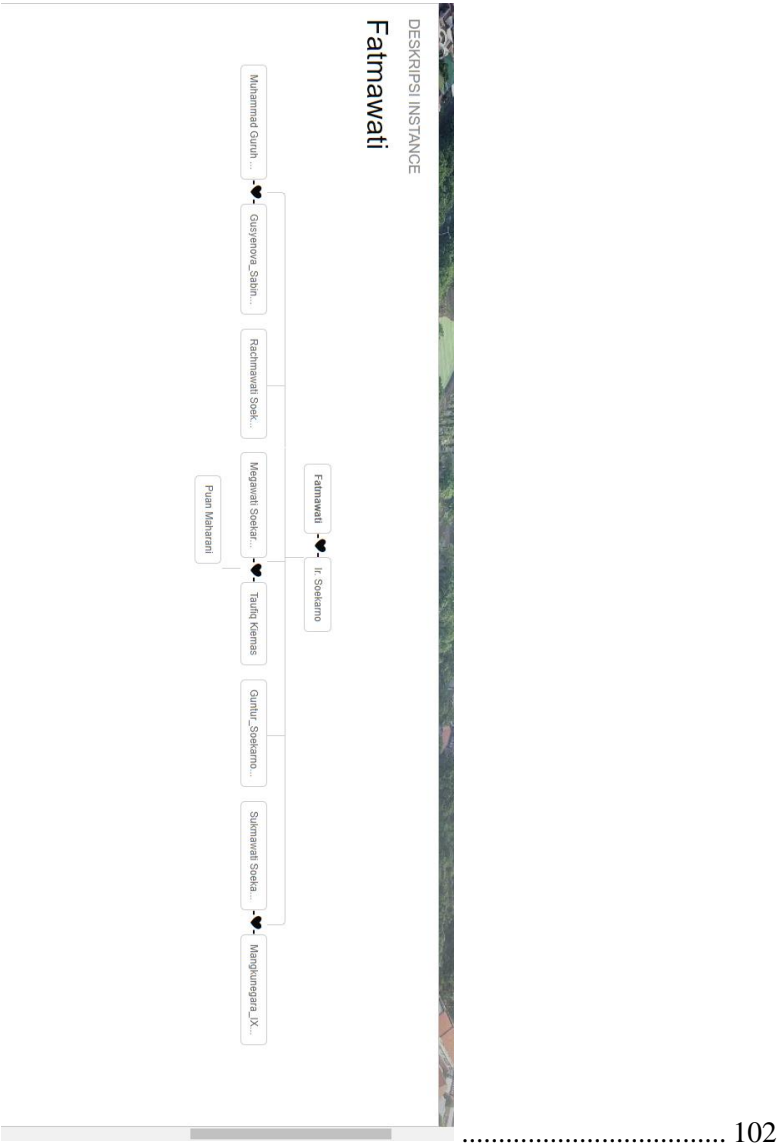
DESKRIPSI INSTANCE

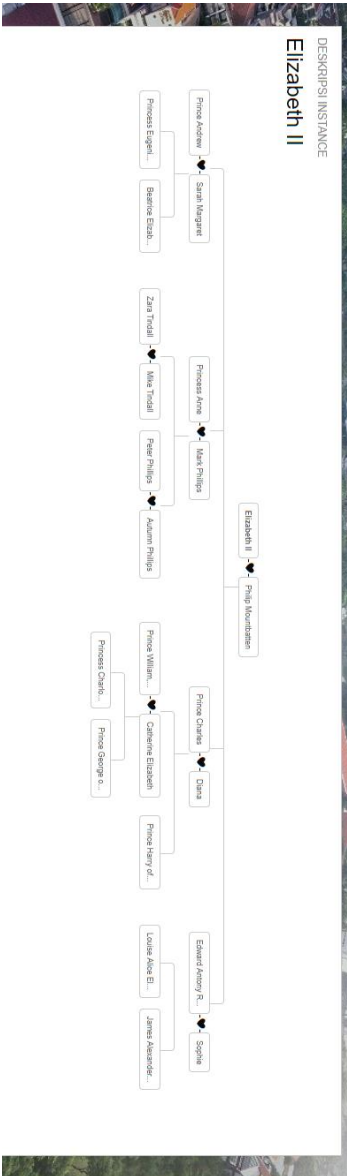
Oetari

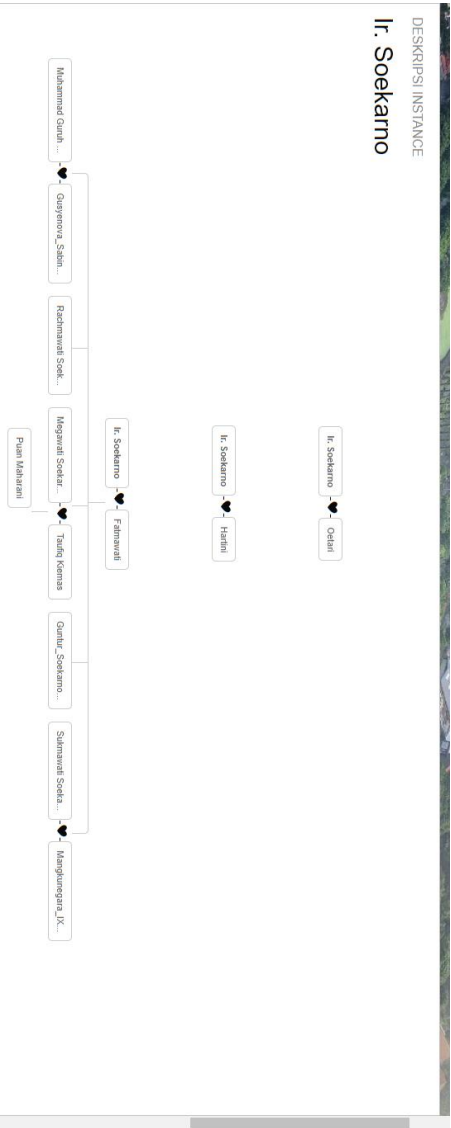
Oetari

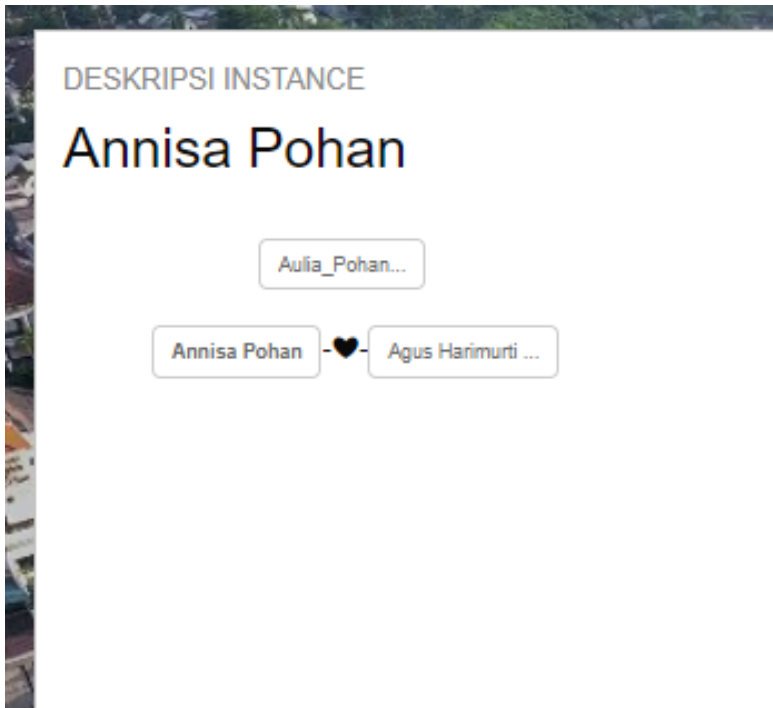
-♥-

Ir. Soekarno









105	
BIODATA PENULIS.....	106

*[Halaman ini sengaja dikosongkan]*

## DAFTAR GAMBAR

Gambar 2.1 Class Hierarchy .....	8
Gambar 2.2 Property .....	9
Gambar 2.3 Class, Property dan Instance.....	12
Gambar 2.4 Contoh Halaman DBpedia.....	13
Gambar 2.5 Data Elizabeth II dalam berbagai versi DBpedia ....	14
Gambar 2.6 Ontologi FamilyTree Keluarga Robert Stevens .....	17
Gambar 2.7 Contoh SPARQL Query .....	18
Gambar 2.8 Database Triple Store Apache Jena Fuseki .....	19
Gambar 2.9 Daftar API Apache Jena Fuseki.....	19
Gambar 2.10 Arsitektur Pellet Reasoner.....	20
Gambar 2.11 Contoh Penggunaan SPARQL Lib .....	25
Gambar 3.1 Flowchart pengembangan sistem.....	27
Gambar 3.2 Halaman DBpedia tentang properti keluarga Raden Wijaya .....	29
Gambar 3.3 Data DBpedia Haryati .....	31
Gambar 3.4 Data DBpedia Prince Philip.....	32
Gambar 3.5 Halaman DBpedia Fatmawati.....	33
Gambar 3.6 Representasi data keluarga Fatmawati .....	33
Gambar 3.7 Hirarki Class.....	34
Gambar 3.8 Hirarki Data property .....	34
Gambar 3.9 Hirarki Object property .....	35
Gambar 3.10 Ontologi Hayam Wuruk (Individuals).....	39
Gambar 3.11 Ontologi Hayam Wuruk (Object Properties).....	39
Gambar 3.12 Ontologi Family Tree (Individuals).....	40
Gambar 3.13 Ontologi Family Tree (Object Properties).....	40
Gambar 3.14 Ontologi union (Individuals) .....	41
Gambar 3.15 Ontologi union (Object Properties) .....	41
Gambar 3.16 Individu Susilo Bambang Yudhoyono sebelum reasoning .....	42
Gambar 3.17 Individu Susilo Bambang Yudhoyono setelah reasoning .....	43
Gambar 3.18 Silsilah keluarga kerajaan Singasari dan Majapahit [11] .....	44

Gambar 4.1 Arsitektur Sistem .....46

Gambar 4.2 Diagram Kasus Penggunaan Sistem .....49

Gambar 4.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh 51

Gambar 4.4 Antarmuka Halaman Utama Family Tree App.....52

Gambar 4.5 Antarmuka Halaman Pohon Keluarga Family Tree App.....52

Gambar 5.1 Implementasi Antarmuka Halaman Utama.....78

Gambar 5.2 Implementasi Antarmuka Halaman Pohon Keluarga .....79



## DAFTAR TABEL

Tabel 2.1 Karakteristik Properti .....	9
Tabel 2.2 Deskripsi Property .....	10
Tabel 2.3 Komponen SWRL .....	14
Tabel 2.4 Method PHP .....	23
Tabel 3.1 Daftar properti yang dibutuhkan dan yang akan dihasilkan .....	30
Tabel 3.2 Namespace DBpedia .....	30
Tabel 3.3 Pemetaan properti equivalent .....	35
Tabel 3.4 Daftar Class .....	35
Tabel 3.5 Daftar Object Property .....	36
Tabel 3.6 Daftar Data Property .....	38
Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak .....	48
Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan .....	49
Tabel 4.3 Spesifikasi Kasus Penggunaan Melihat Informasi Tokoh .....	50
Tabel 4.4 Spesifikasi Atribut Rancangan Antarmuka Halaman Family Tree App .....	52

## DAFTAR KODE SUMBER

Kode Sumber 3.1 Kode Java untuk memodelkan data Fatmawati ke dalam modelActor .....	32
Kode Sumber 3.2 Kode sumber untuk menggabungkan dua model .....	38
Kode Sumber 5.1 Implementasi proses inialisasi variabel statis .....	54
Kode Sumber 5.2 Implementasi inialisasi model Instance dan famonto.....	54
Kode Sumber 5.3 Implementasi ekstraksi file RDF tokoh .....	55
Kode Sumber 5.4 Implementasi penggabungan model .....	55
Kode Sumber 5.5 Implementasi proses reasoning.....	55
Kode Sumber 5.6 Implementasi print hasil reasoning sebagai file RDF .....	56
Kode Sumber 5.7 Kode Sumber SPARQL untuk mengambil value bertipe Person dan Fungsi Dropdown Select.....	58
Kode Sumber 5.8 Fungsi Get name.....	59
Kode Sumber 5.9 Fungsi Get father .....	61
Kode Sumber 5.10 Fungsi Get mother .....	63
Kode Sumber 5.11 Fungsi Get sibling.....	65
Kode Sumber 5.12 Fungsi Get spouse.....	67
Kode Sumber 5.13 Fungsi Get child .....	69
Kode Sumber 5.14 Fungsi Get child in law.....	71
Kode Sumber 5.15 Fungsi Get grand child .....	73
Kode Sumber 5.16 get grand child in law .....	75
Kode Sumber 5.17 Get great grand child .....	78

# **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. Sedangkan raja adalah gelar yang diberikan kepada anggota kerajaan secara turun-temurun. Tokoh bersejarah, keduanya 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. Data bersumber dari artikel Wikipedia mengenai tokoh sejarah Indonesia.
3. Platform pengembangan aplikasi adalah situs web.
4. Data yang digunakan sebagai *value* properti bersumber dari isi properti DBpedia dan hasil ekstraksi manual pada halaman Wikipedia Indonesia *person* terkait.
5. Aplikasi tidak dapat menangani *person* yang tidak memiliki halaman DBpedia.

6. Batas relasi adalah ayah, ibu, saudara, istri, anak, menantu, cucu, pasangan cucu, dan cicit.
7. Aplikasi sangat bergantung pada kelengkapan atribut data DBpedia.
8. *Reasoner* yang digunakan adalah Pellet.
9. Aplikasi yang dibuat tidak menyediakan *form* untuk pengelolaan data (tambah, ubah, hapus).
10. Aplikasi yang dibuat hanya untuk menampilkan deskripsi *person* yang merupakan hasil dari ontologi yang dibangun.

#### 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, ontologi, DBpedia, *Family Relationships Ontology*, SPARQL, Apache Jena Fuseki, SWRL (*Semantic Web Rule Language*), 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.

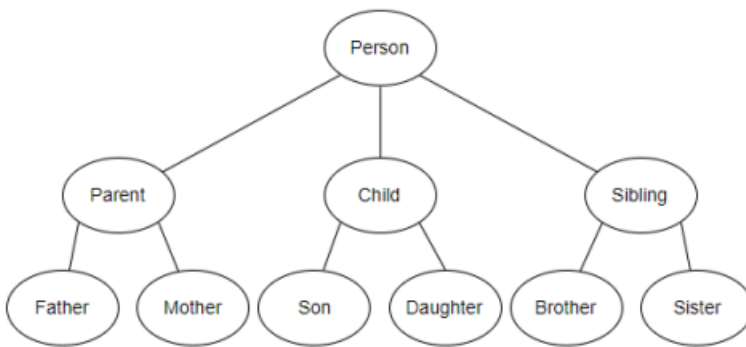
#### **2.2. Ontologi**

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 [3]. 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 superclassnya 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 hierarchy yang dicontohkan pada Gambar 2.1.

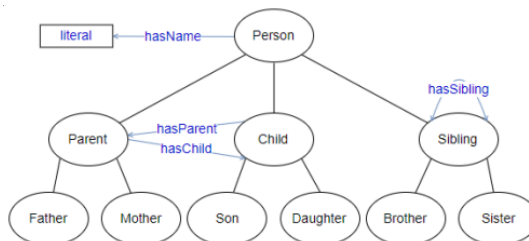


**Gambar 2.1 Class Hierarchy**

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 2.2.



**Gambar 2.2 Property**

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

**Tabel 2.1 Karakteristik Properti**

Karakteristik Property	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 properti yang merujuk pada dirinya sendiri. Misalnya individu Soekarno memiliki Presiden, yaitu dirinya sendiri.
Irreflexive	Menegaskan bahwa suatu individu mempunyai properti yang tidak bisa merujuk pada dirinya sendiri. Misalnya individu Soekarno memiliki anak, maka propertinya adalah <i>irreflexive</i> .

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

**Tabel 2.2 Deskripsi Property**

Deskripsi Property	Keterangan
Equivalent to	Satu properti dengan yang lain memiliki identitas yang sama dengan karakteristik yang sama. Misal, properti dbp:issue dan dbo:child adalah equivalent karena sama-sama menjelaskan hubungan kepemilikan keturunan.
SubProperty of	Properti yang dipilih adalah bagian dari properti lain. Semisal properti hasSon dan hasDaughter adalah SubProperty of hasChild
Inverse of	Properti yang dipilih berbalik juga ke dirinya. Contohnya adalah properti hasSpouse.
Domain (Intersection)	Properti memiliki domain tertentu, misalnya properti hasChild hanya dimiliki oleh domain Parent.

Range (Intersection)	Properti memiliki range tertentu, misalnya properti hasWife hanya memiliki range Female.
Disjoint with	Properti yang terpilih tidak akan berhubungan dengan properti lain. Jika properti hasParent bersifat Disjoint with hasChild, maka jika Megawati memiliki Parent Soekarno, maka Soekarno tidak memiliki Parent Megawati.
SuperProperty of (chain)	Properti terpilih adalah SuperProperty dari dua atau lebih properti yang lain dengan ditandai “o”. Misalnya properti hasGrandChild adalah SuperProperty dari ‘hasChild o hasChild’.

Karakteristik dan deskripsi properti 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-properti 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 2.3

```

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 2.3 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* [4].

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.

### 2.3. 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 2.4 adalah contoh sebuah halaman DBpedia.

Content-Length: 67802	
About: <a href="#">Raden Wijaya</a>	
An Entity of Type : <a href="#">Thing</a> , from Named Graph : <a href="http://id.dbpedia.org">http://id.dbpedia.org</a> , within Data Space : <a href="http://id.dbpedia.org">id.dbpedia.org</a>	
	
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
<a href="#">dbpedia-owl:abstract</a>	<ul style="list-style-type: none"> <li>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 Jayawardana.</li> </ul>
<a href="#">dbpedia-owl:thumbnail</a>	<ul style="list-style-type: none"> <li><a href="http://upload.wikimedia.org/wikipedia/commons/thumb/b/b6/Harihara_Majapahit_1.JPG/200px-Harihara_Majapahit_1.JPG">http://upload.wikimedia.org/wikipedia/commons/thumb/b/b6/Harihara_Majapahit_1.JPG/200px-Harihara_Majapahit_1.JPG</a></li> </ul>
<a href="#">dbpedia-owl:wikiPageID</a>	<ul style="list-style-type: none"> <li>11640 (xsd:integer)</li> </ul>
<a href="#">dbpedia-owl:wikiPageRevisionID</a>	<ul style="list-style-type: none"> <li>6634882 (xsd:integer)</li> </ul>
<a href="#">dbpedia-owl:wikiPageWikiLink</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Babad_Tanah_Jawi</a></li> <li><a href="#">dbpedia-id:Clung_Wanara</a></li> <li><a href="#">dbpedia-id:Java</a></li> <li><a href="#">dbpedia-id:Suku_Jawa</a></li> <li><a href="#">dbpedia-id:Jayalatriwang</a></li> <li><a href="#">dbpedia-id:Jayanagara</a></li> <li><a href="#">dbpedia-id:Kerajaan_Kadiri</a></li> <li><a href="#">dbpedia-id:Ken_Anot</a></li> <li><a href="#">dbpedia-id:Kertanagara</a></li> <li><a href="#">dbpedia-id:Kubilai_Khan</a></li> <li><a href="#">dbpedia-id:Pakuan_Madura</a></li> <li><a href="#">dbpedia-id:Majapahit</a></li> <li><a href="#">dbpedia-id:Pengucatan_monarki</a></li> <li><a href="#">dbpedia-id:12_November</a></li> <li><a href="#">dbpedia-id:Ekspedisi_Pamalayu</a></li> <li><a href="#">dbpedia-id:Paranaton</a></li> <li><a href="#">dbpedia-id:Patih</a></li> <li><a href="#">dbpedia-id:Pranaparamita</a></li> <li><a href="#">dbpedia-id:Wangsa_Rajasa</a></li> <li><a href="#">dbpedia-id:Siwa</a></li> <li><a href="#">dbpedia-id:Kerajaan_Singhasari</a></li> <li><a href="#">dbpedia-id:Suku_Sunda</a></li> </ul>

**Gambar 2.4 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 properti yang bermacam-macam dan belum tentu versi yang lain memiliki properti yang sama. Contohnya adalah data Ratu Elizabeth II yang tidak hanya ada di DBpedia berbahasa Inggris saja, namun juga ada dalam situs DBpedia Jerman, Spanyol, Italia, Prancis, dan Indonesia.



Gambar 2.5 Data Elizabeth II dalam berbagai versi DBpedia

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 [5]. 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>
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> .



Atom	Deskripsi
sameAs(x, y)	x dan y adalah variabel atau individual yang menyatakan bahwa keduanya merupakan individu yang sama
differentFrom(x, y)	x dan y adalah variabel atau individual yang menyatakan bahwa keduanya merupakan individu yang berbeda

Berikut merupakan contoh SWRL *rule* yang menyatakan bahwa x3 adalah ayah (*father*) dari x1 jika x2 adalah orang tua (*parent*) dari x1 dan x3 adalah istri (*wife*) dari x2.

hasParent(?x1, ?x2), hasWife(?x2, ?x3) -> hasFather(?x1, ?x3)
---

Tanda “->” digunakan sebagai penghubung antara *antecedent* dan *consequent* atom. Sedangkan “,” berfungsi sebagai penghubung antar atom. Sebuah variabel ditandai dengan ekspresi “?”.

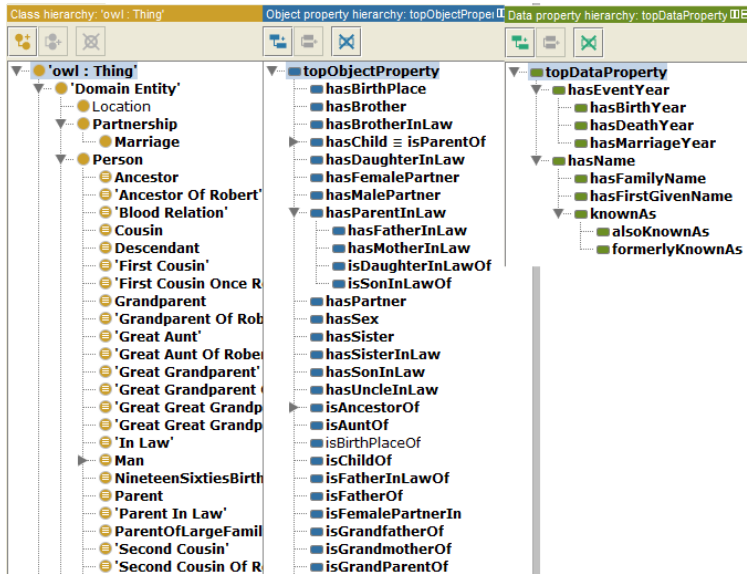
## 2.5. Family Relationships Ontology

*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 [6].

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.ode.org/roberts/family-tree.owl> [7]. 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.



**Gambar 2.6 Ontologi FamilyTree Keluarga Robert Stevens**

Bisa 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.

## 2.6. 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-statementen 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 2.7:

```
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/>

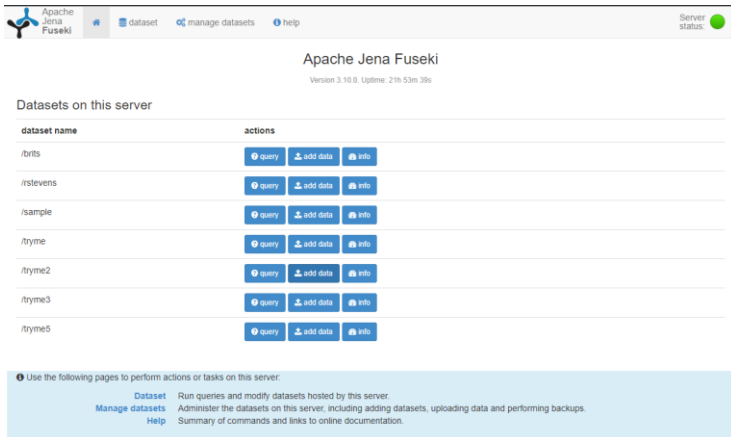
SELECT DISTINCT ?s
WHERE {
    ?s rdf:type foaf:Person.
    ?s foaf:name ?name
}
```

**Gambar 2.7 Contoh SPARQL Query**

Pada tugas akhir ini, SPARQL Query dilakukan untuk mencari relasi pada basis data Apache Jena Fuseki.

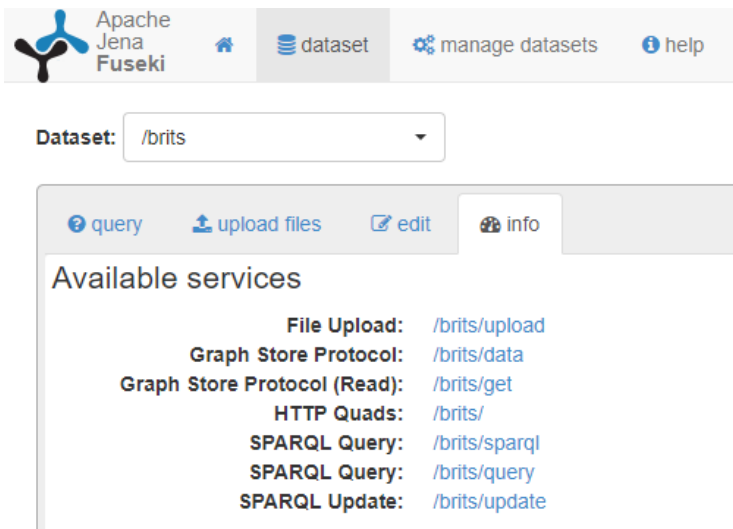
## **2.7. Apache Jena Fuseki**

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 2.8. menunjukkan daftar basis data yang ada di dalam server Apache Jena Fuseki.



**Gambar 2.8 Database Triple Store Apache Jena Fuseki**

Apache Jena Fuseki menyediakan beberapa API untuk digunakan oleh peneliti seperti pada Gambar 2.9.

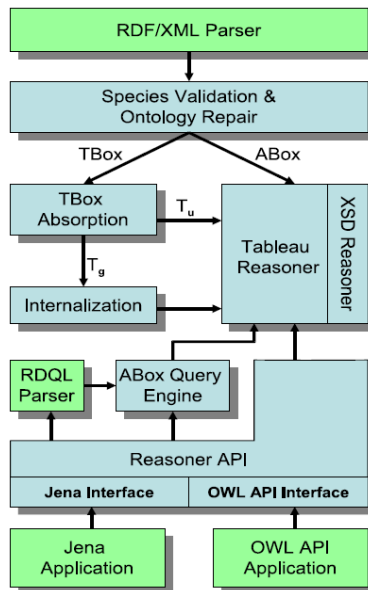


**Gambar 2.9 Daftar API Apache Jena Fuseki**

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.

## 2.8. Pellet Reasoner



Gambar 2.10 Arsitektur Pellet Reasoner

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 [8]. 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*). 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 2.10 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 [9].

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. PHP

PHP adalah bahasa pemrograman berbasis *open source* yang dikhususkan 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 [9].

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 [10].

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 "?". Sedang *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()` [11]. Tabel berikut akan menerangkan beberapa *method* yang digunakan dalam implementasi pengerjaan tugas akhir ini.

**Tabel 2.4 Method PHP**

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

## 2.10. 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 [12], 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 [13], 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*.

### **2.11. 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 2.11. Hasil dari SPARQL Query dengan SPARQL Lib adalah array *row* dan *field*. Gambar 2.11 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 2.11 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

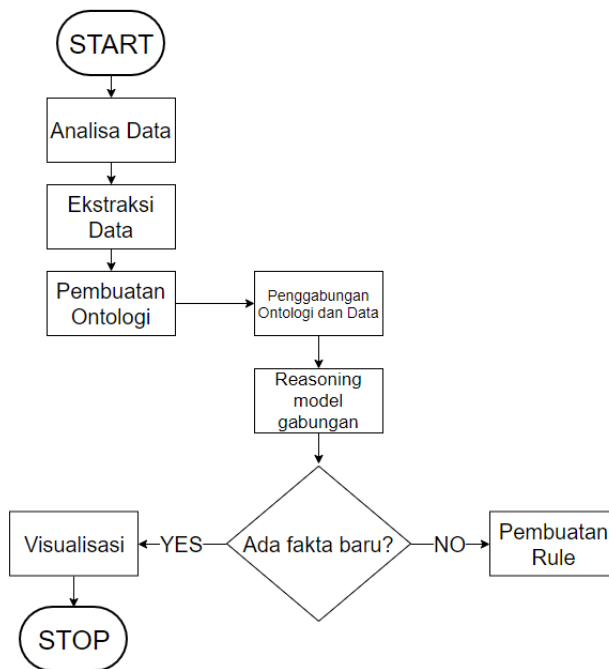
Apache Jena adalah *plugin open source* berbasis Java yang digunakan untuk membangun aplikasi *Linked Data* dan *Semantic Web. Framework* ini terdiri dari beberapa API yang berinteraksi secara bersamaan untuk memproses data dengan format RDF. Aplikasi yang memiliki *plugin* Apache Jena sanggup membuat 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.

### 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](https://id.wikipedia.org/wiki/Daftar_Presiden_Indonesia).

*Person* yang digunakan sebagai data adalah *person* yang dinilai memiliki banyak keterkaitan dengan *person* lain. *Person* harus memiliki atribut `foaf:name`. Maka dari itu, untuk pembandingan, akan diolah pula data dari Presiden Indonesia, data pahlawan nasional, dan 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 [10].

### 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-properti utama yang diperlukan untuk mengetahui silsilah keluarga seorang *person* seperti *child*, *spouse*, dan *parent*. Contoh properti yang akan dipakai adalah data properti 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 properti keluarga Raden Wijaya

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

**Tabel 3.1 Daftar properti yang dibutuhkan dan yang akan dihasilkan**

Properti yang dibutuhkan	Properti yang akan dihasilkan
Name	Child in Law
Spouse	Grandchild
Parent	Grandchild in Law
Issue	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 Prancis (*fr.dbpedia.org*), dan lain-lain dimana tiap situs memiliki dan menyediakan properti-properti yang berbeda-beda. Beberapa *namespace* dan keterangannya bisa dilihat di Tabel 3.2.

**Tabel 3.2 Namespace DBpedia**


Namespace	Keterangan
dbpedia	< <a href="http://dbpedia.org/">http://dbpedia.org/</a> >
id.dbpedia	< <a href="http://id.dbpedia.org/">http://id.dbpedia.org/</a> >
foaf	< <a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a> >
rdf	< <a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a> >
rdfs	< <a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a> >
dbo	< <a href="http://dbpedia.org/ontology/">http://dbpedia.org/ontology/</a> >
dbp	< <a href="http://dbpedia.org/property/">http://dbpedia.org/property/</a> >
dbpprop-id	< <a href="http://id.dbpedia.org/property/">http://id.dbpedia.org/property/</a> >
dbpedia-owl	< <a href="http://id.dbpedia.org/ontology/">http://id.dbpedia.org/ontology/</a> >
owl	< <a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a> >



yago	< <a href="http://yago-knowledge.org/resource/">http://yago-knowledge.org/resource/</a> >
schema	< <a href="http://schema.org/">http://schema.org/</a> >
dct	< <a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a> >

Perlu diketahui pula pengertian properti-properti yang akan sering digunakan dalam pengerjaan tugas akhir ini dikarenakan setiap URL tokoh yang diambil tidak memiliki jumlah properti maupun jenis properti yang sama. Tidak semua URL memiliki properti 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 properti 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 properti keluarga *spouse* saja. Meskipun beberapa properti dapat dihasilkan dari proses *reasoning*, akan tetapi jika data dasar tidak lengkap, tidak semua data bisa dipercaya.


Content-Length: 8908





**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](http://id.dbpedia.org)


Property	Value
is <i>dbpedia-owl:spouse</i> of	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Soekarno</a></li> </ul>
is <i>dbpedia-owl:wikiPageWikiLink</i> of	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Soekarno</a></li> </ul>
is <i>dbpprop-id:spouse</i> of	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Soekarno</a></li> </ul>


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



POWERED BY  
VIRTUOSO



LINKING OPENDATA


SPARQL


OPEN DATA


W3C


XHTML

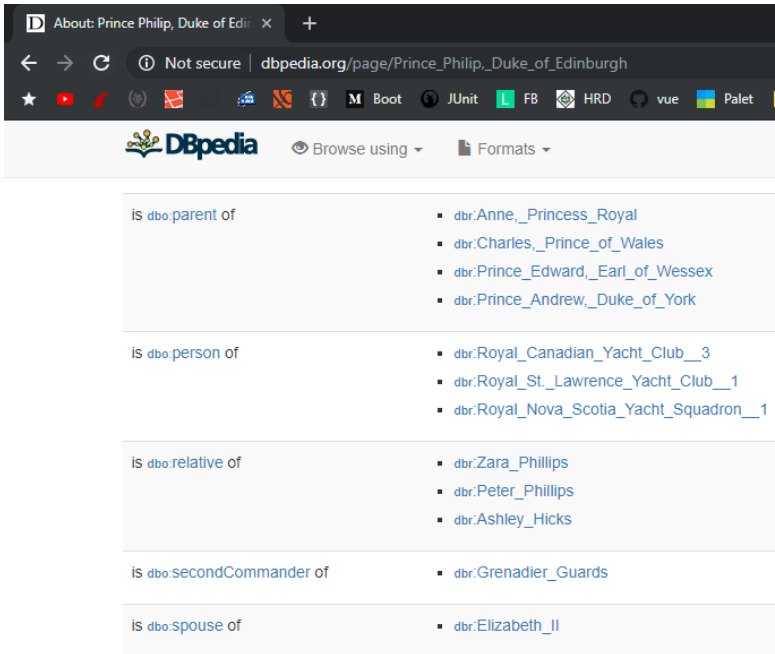

RDFa

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

**Gambar 3.3 Data DBpedia Haryati**

Sebaliknya, data yang memiliki properti dasar dan properti 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 lampiran subbab **Error! Reference 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 properti Fatmawati serta relasinya yang tercantum dalam halaman DBpedia pada Gambar 3.5. Tentunya tidak hanya Fatmawati saja yang dijadikan model, tetapi raja-raja, tokoh sejarah dan pahlawan nasional lainnya juga dimodelkan. Jika model Fatmawati saja yang direpresentasikan sebagai *graph*, maka representasinya adalah seperti pada Gambar 3.6:

Content-Length: 50597

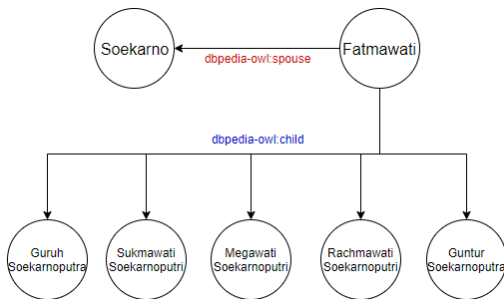


**About: Fatmawati**  
An Entity of Type : [Person](#), from Named Graph : <http://id.dbpedia.org>, within Data Space : [id.dbpedia.org](http://id.dbpedia.org)

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
<a href="#">dbpedia-owl:abstract</a>	<ul style="list-style-type: none"> <li>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.</li> </ul>
<a href="#">dbpedia-owl:activeYearsEndDate</a>	<ul style="list-style-type: none"> <li>1967-03-12 (xsd:date)</li> </ul>
<a href="#">dbpedia-owl:activeYearsStartDate</a>	<ul style="list-style-type: none"> <li>1945-08-17 (xsd:date)</li> </ul>
<a href="#">dbpedia-owl:birthDate</a>	<ul style="list-style-type: none"> <li>1923-02-05 (xsd:date)</li> </ul>
<a href="#">dbpedia-owl:birthPlace</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Bengkulu</a></li> <li><a href="#">dbpedia-id:Hindia_Belanda</a></li> </ul>
<a href="#">dbpedia-owl:child</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Guruh_Soekarnoputra</a></li> <li><a href="#">dbpedia-id:Megewati_Soekarnoputri</a></li> <li><a href="#">dbpedia-id:Sukmawati_Soekarnoputri</a></li> <li><a href="#">dbpedia-id:Rachmawati_Soekarnoputri</a></li> <li><a href="#">dbpedia-id:Guntur_Soekarnoputra</a></li> </ul>
<a href="#">dbpedia-owl:deathPlace</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Kuala_Lumpur</a></li> <li><a href="#">dbpedia-id:Malaysia</a></li> </ul>
<a href="#">dbpedia-owl:nationality</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Indonesia</a></li> </ul>
<a href="#">dbpedia-owl:office</a>	<ul style="list-style-type: none"> <li>Ibu Negara Indonesia</li> </ul>
<a href="#">dbpedia-owl:orderInOffice</a>	<ul style="list-style-type: none"> <li>1</li> </ul>
<a href="#">dbpedia-owl:president</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Soekarno</a></li> </ul>
<a href="#">dbpedia-owl:religion</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Islam</a></li> </ul>
<a href="#">dbpedia-owl:spouse</a>	<ul style="list-style-type: none"> <li><a href="#">dbpedia-id:Soekarno</a></li> </ul>

Gambar 3.5 Halaman DBpedia Fatmawati

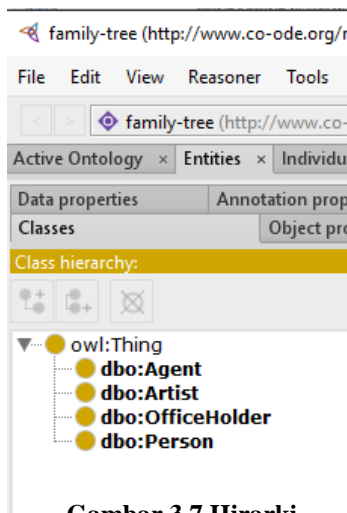


**Gambar 3.6 Representasi data keluarga Fatmawati**

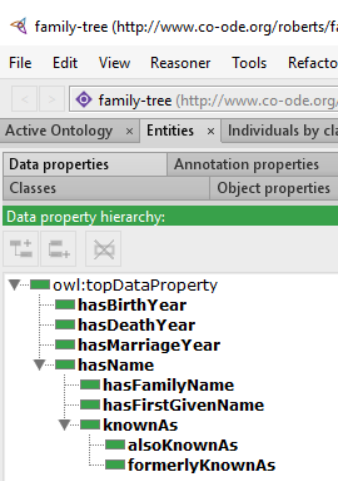
### 3.3. Pembuatan Ontologi

Pada tahap ini, ontologi dibangun dengan menggabungkan 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 properti yang memiliki arti yang sama akan diatur sebagai *equivalent class*, seperti properti *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 properti DBpedia dan Family Tree App dapat dilihat di

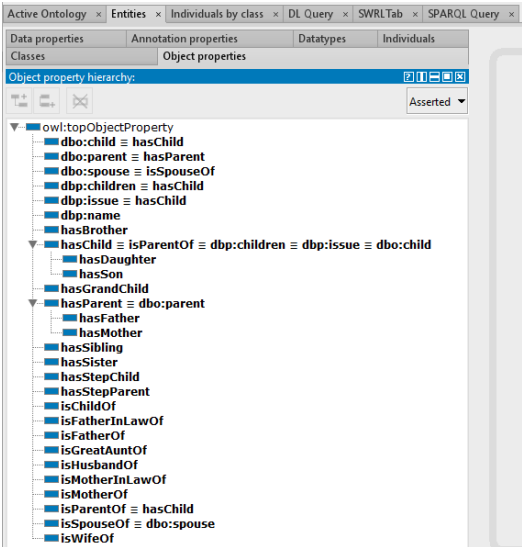
Tabel 3.3. Object property dan data property yang digunakan dapat dilihat di Tabel 3.5 dan Tabel 3.6.



**Gambar 3.7 Hirarki Class**



**Gambar 3.8 Hirarki Data property**



Gambar 3.9 Hirarki Object property

Tabel 3.3 Pemetaan properti equivalent

Properti DBpedia	Properti 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 3.4 Daftar Class

Class	URL	Karakteristik	Keterangan
Owl:Thing	http://www.w3.org/2002/07/owl#thing	-	Class untuk menjelaskan

			kan suatu hal
Dbo:agent	<a href="http://dbpedia.org/ontology/agent">http://dbpedia.org/ontology/agent</a>	-	Class untuk menjelas kan <i>agent</i>
Dbo:Person	<a href="http://dbpedia.org/ontology/person">http://dbpedia.org/ontology/person</a>	-	Class untuk menjelas kan seseoran g

**Tabel 3.5 Daftar Object Property**

Object property	URL	Karakteristik	Keterangan
dbo:child	<a href="http://dbpedia.org/ontology/child">dbpedia.org/ontology/child</a>	-	ekuivalen dengan hasChild
dbo:parent	<a href="http://dbpedia.org/ontology/parent">dbpedia.org/ontology/parent</a>	-	ekuivalen dengan hasParent
dbo:spouse	<a href="http://dbpedia.org/ontology/spouse">dbpedia.org/ontology/spouse</a>	symmetric	ekuivalen dengan isSpouseOf
dbp:children	<a href="http://dbpedia.org/property/children">dbpedia.org/property/children</a>	-	ekuivalen dengan hasChild
dbp:issue	<a href="http://dbpedia.org/property/issue">dbpedia.org/property/issue</a>	-	ekuivalen dengan hasChild
hasChild	<a href="http://co-ode.org/roberts/fa">co-ode.org/roberts/fa</a>	-	ekuivalen dengan

	mily-tree.owl#haschild		dbo:child, dbp:children, dbp:issue, dan isParentof
hasGrandChild	code.org/roberts/family-tree.owl#hasgrandchild	-	SuperProperty dari 'hasChild o hasChild'
hasParent	code.org/roberts/family-tree.owl#hasparent	-	ekuivalen dengan dbo:parent
hasSibling	code.org/roberts/family-tree.owl#hassibling	symmetric	satu properti ini dipakai oleh dua URL
isChildOf	code.org/roberts/family-tree.owl#ischildof	-	inverse dari hasChild
isParentOf	code.org/roberts/family-tree.owl#isparentof	-	ekuivalen dengan hasChild
isSpouseOf	code.org/roberts/family-tree.owl#isspouseof	-	ekuivalen dengan dbo:spouse



**Tabel 3.6 Daftar Data Property**

Data property	URL	Karakteristik	Keterangan
Dbp:name	Dbpedia.org/property/name	-	-
hasName	Code.org/roberts/family-tree.owl#hasName	-	-
Name	Id.dbpedia.org/property/name	-	-

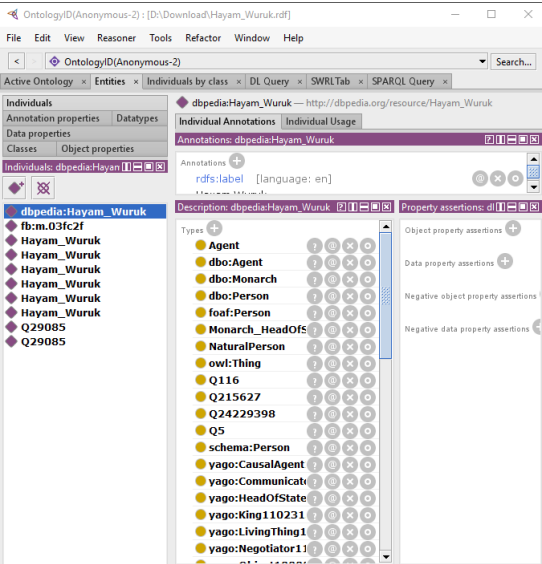
### 3.4. Penggabungan Model Data dan Model Family Relationship Ontology

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.

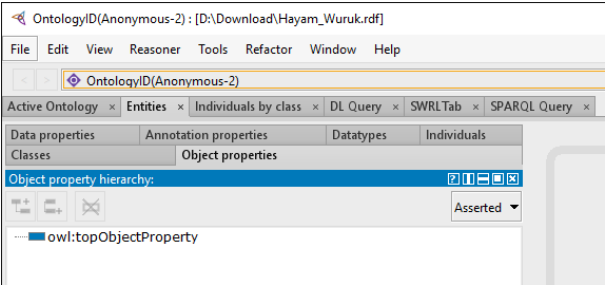
```
final Model union =
ModelFactory.createUnion(modelHayamWuruk,modelFamilyTree
);
```

#### **Kode Sumber 3.2 Kode sumber untuk menggabungkan dua model**

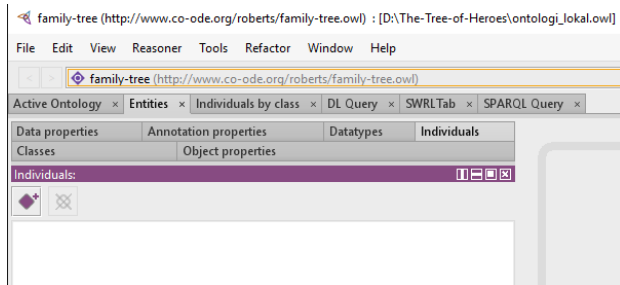
Sebagai input, adalah model ontologi Hayam Wuruk (Gambar 3.10 dan Gambar 3.11) dan model ontologi Family Tree (Gambar 3.12 dan Gambar 3.13). Sebagai output, adalah ontologi union yang baru, yang menyimpan data dan properti dari kedua ontologi pada Gambar 3.14 dan Gambar 3.15.



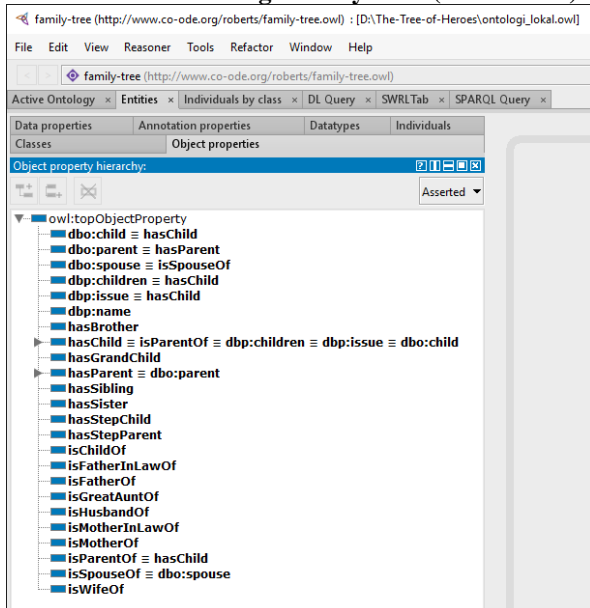
Gambar 3.10 Ontologi Hayam Wuruk (Individuals)



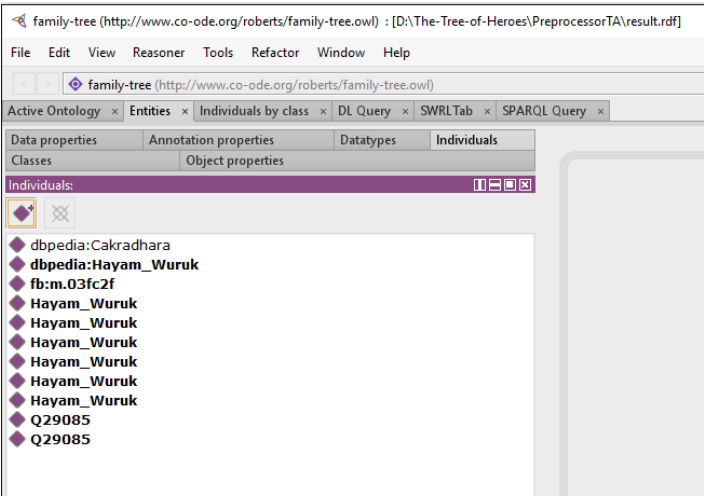
Gambar 3.11 Ontologi Hayam Wuruk (Object Properties)



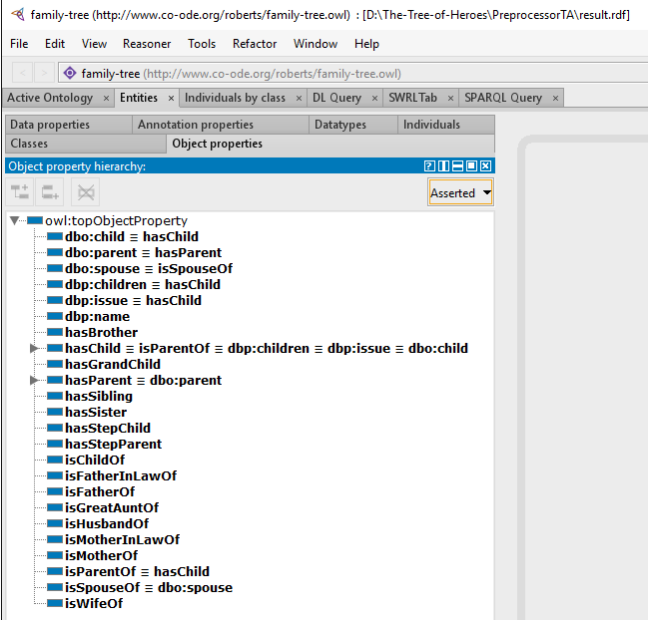
**Gambar 3.12 Ontologi Family Tree (Individuals)**



**Gambar 3.13 Ontologi Family Tree (Object Properties)**



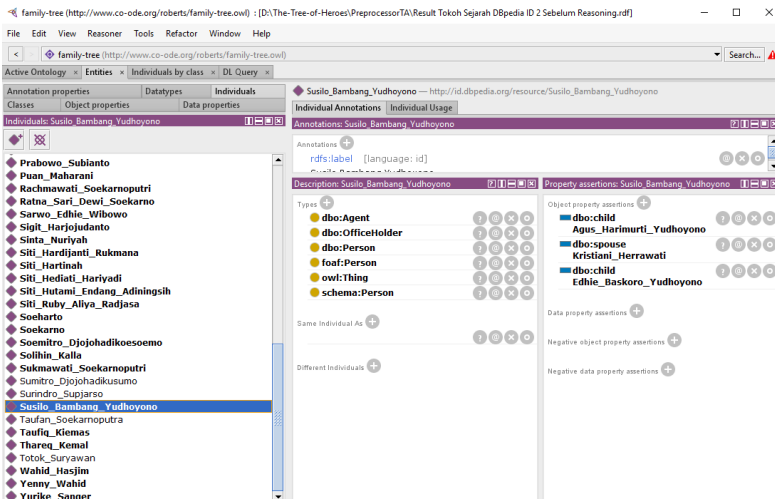
**Gambar 3.14 Ontologi union (Individuals)**



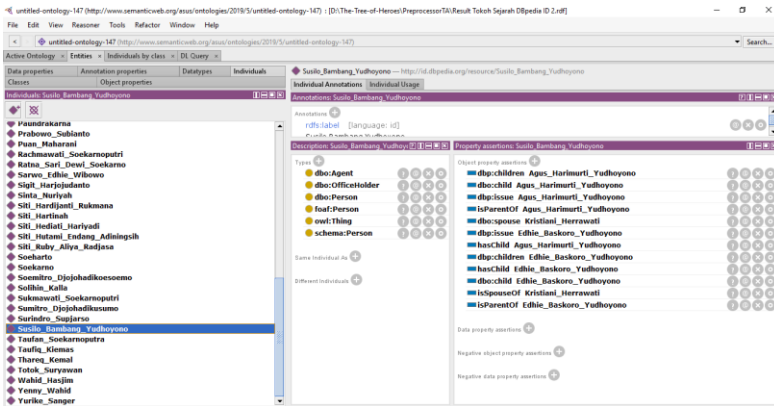
**Gambar 3.15 Ontologi union (Object Properties)**

### 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.



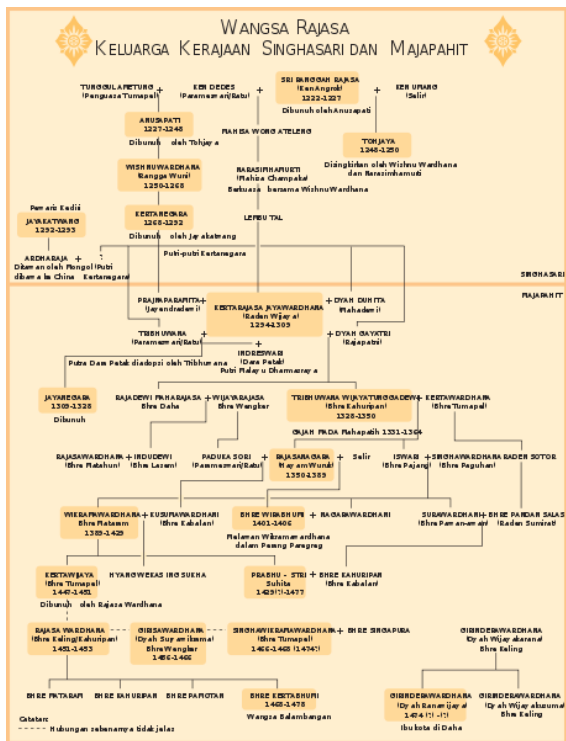
**Gambar 3.16 Individu Susilo Bambang Yudhoyono sebelum reasoning**



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**  
[11]

## **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**

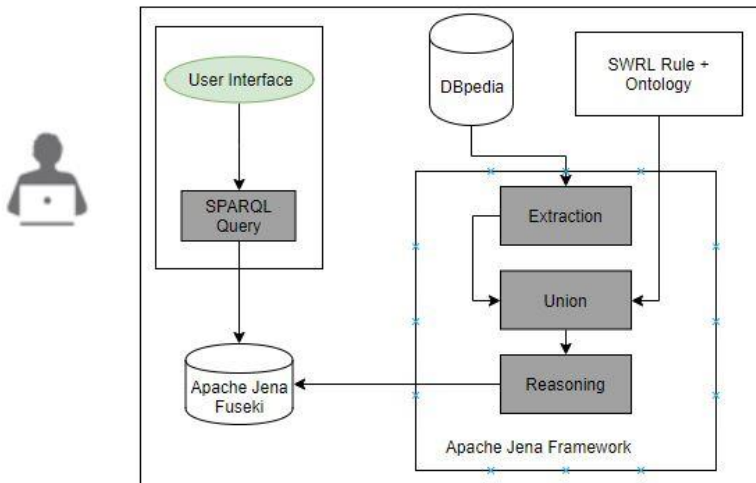
Permasalahan yang diangkat dalam tugas akhir ini adalah visualisasi pohon keluarga tokoh sejarah Indonesia. Studi kasus permasalahan tersebut dipecahkan dengan ekstraksi data, penggabungan data, *reasoning* dan visualisasi. Pencarian relasi antar *person* dilakukan dengan menggunakan *property* dan *SWRL 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 3.5.

### 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**

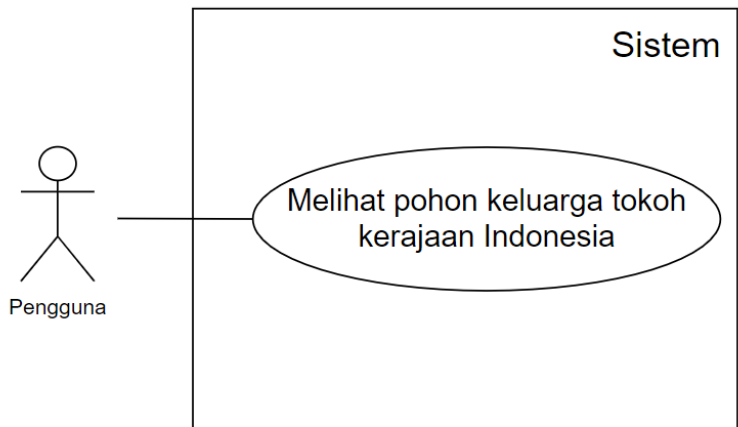
Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
TA-F0001	Menampilkan pohon keluarga tokoh	Pengguna dapat melihat pohon keluarga tokoh sejarah Indonesia

#### 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

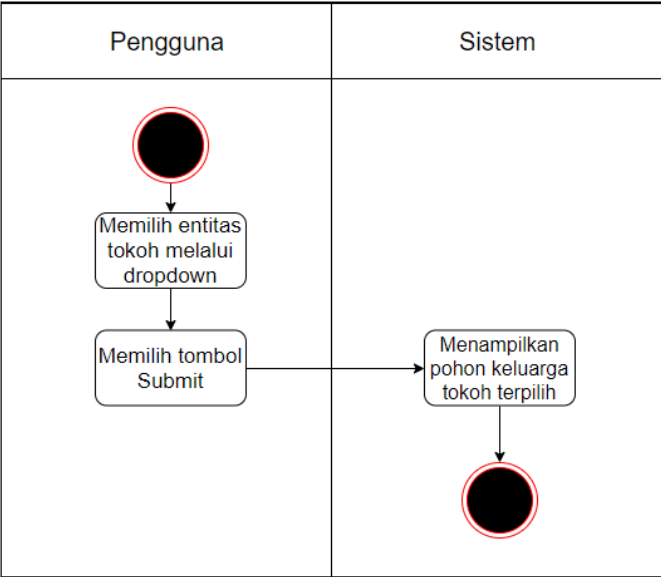
Kode Kasus Penggunaan	Nama
TA-UC0001	Melihat pohon keluarga tokoh sejarah Indonesia

#### 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**

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



Gambar 4.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh

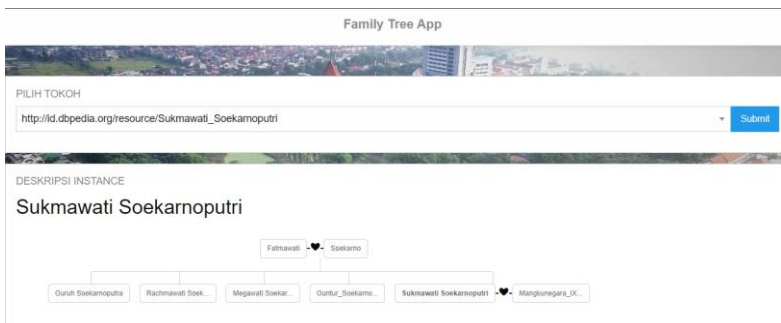
4.2. Perancangan Antarmuka Pengguna

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**



**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	<i>Entity Dropdown Select</i>	<i>Form</i>	Menampilkan daftar entitas tokoh
2	<i>Submit Button</i>	<i>Button</i>	Mengeksekusi <i>request form</i>
3	<i>Entity Family Tree</i>	<i>Tree</i>	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 inialisasi 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.g
etClassLoader());
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 inialisasi 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 = {
    //Indonesian Emperors
    "Soekarno",
    "Fatmawati",
    "Megawati_Soekarnoputri"
}

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 properti `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 properti `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 properti `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 properti `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 rdfs:
<http://www.w3.org/2000/01/rdf-schema#>

```

```

PREFIX dbpprop-id:
<http://id.dbpedia.org/property/>
SELECT ?name
WHERE {
    <' . $spouseIRI[$i]
. '> rdfs:label ?name
    } LIMIT 1
    ');
    foreach ($data_spouse as $row) {
        foreach ($data_spouse->fields() as
$field) {
            if (strlen($row[$field]) > 20)
                $row[$field] = substr($row[$field], 0, 15)
. '...';
                echo '<a href="?entity=' .
urlencode($spouseIRI[$i]) . '>' .
str_replace('http://www.dbpedia.org/resource/', "",
$row[$field]) . '</a>';
                $hasName = 1;
            }
        }
        if ($hasName==0){
            if (strlen($spouseIRI[$i]) > 20)
                $spouseIRI[$i] =
substr($spouseIRI[$i], 31, 15) . '...';
            echo '<a ' .
urlencode($spouseIRI[$i]) . '>' . $spouseIRI[$i] .
'</a>';
        }
    }
}
}

```

**Kode Sumber 5.12 Fungsi Get spouse**

### **Get child**

Fungsi ini digunakan untuk mendapatkan keturunan dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti `hasChild` yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.13.

```

$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-

```





### Get child in law

Fungsi ini digunakan untuk mendapatkan menantu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti `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>";
}
}

```

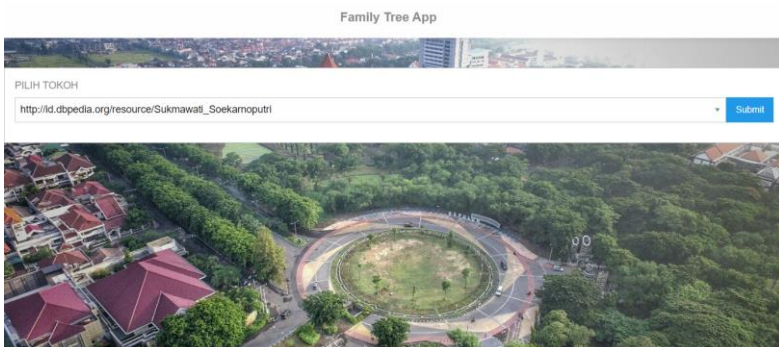
Kode Sumber 5.17 Get great grand child

### 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 Tampilan 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.

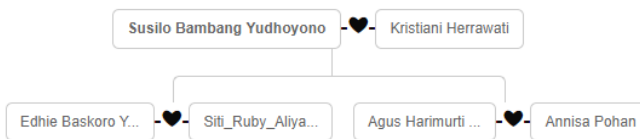


Gambar 5.1 Implementasi Antarmuka Halaman Utama

### 5.3.2. Implementasi Tampilan 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. 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.

## Susilo Bambang Yudhoyono



**Gambar 5.2 Implementasi Antarmuka Halaman Pohon Keluarga**

*(Halaman ini sengaja dikosongkan)*

## 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 <i>Device</i>	: Laptop
Sistem Operasi	: Microsoft Windows 10 64-bit
<i>Protege</i>	: Protege 5.2
<i>Reasoner</i>	: Pellet
<i>Browser</i>	: Google Chrome

### 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 properti 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

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

Kode Uji	R-02
Nama Pengujian	Pengujian axiom hasChild
Kondisi sebelum reasoning	Lampiran Gambar 3
Kondisi sesudah reasoning	Lampiran Gambar 4
Keterangan	Ekuivalensi properti <i>dbo:child</i> , <i>dbp:children</i> , dan <i>dbp:issue</i> menjadi hasChild
Nilai	Berhasil

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

### 6.3.2. Pengujian Visualisasi

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 properti anak
Nilai	Berhasil

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

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

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

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

Kode Uji	V-06
----------	------



Nama Pengujian	Kondisi relasi tidak memiliki properti identitas
Hasil	Lampiran Gambar 12
Keterangan	Contoh pohon keluarga Anisa Pohan yang memiliki relasi ke URL yang tidak memiliki properti <i>label</i> atau <i>name</i> .
Nilai	Berhasil

## **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 properti 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 lokal 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] Wikipedia, "Daftar Raja di Jawa," 09 06 2019. [Online]. Available:  
[https://id.wikipedia.org/wiki/Daftar\\_raja\\_di\\_Jawa](https://id.wikipedia.org/wiki/Daftar_raja_di_Jawa).  
[Accessed 26 06 2019].
- [3] S. J. Miller, Introduction to Ontology Concepts and Terminology, Lisbon, Portugal: University of Wisconsin-Milwaukee, 2013.
- [4] M. A. Ramadhanie, Penerapan Ontologi Objek Pembelajaran Untuk Kebutuhan Personalisasi E-Learning Berbasis Semantic Web, Depok: Universitas Indonesia, 2009.
- [5] S. Nikles, "Expressiveness of Enterprise Modelling Languages," University of Applied Sciences Northwestern Switzerland, Basel, 2010.
- [6] 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.
- [7] "Professor Robert Stevens," [Online]. Available:  
<http://www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl>  
. [Accessed 06 January 2016].
- [8] 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.
- [9] B. Parsia and E. Sirin, "Pellet: An OWL DL Reasoner," University of Maryland, College Park.
- [10] D. Wu and A. Håkansson, "A Method of Identifying Ontology Domain," *Procedia Computer Science*, vol. 35, pp. 504-513, 2014.

- [11] 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].
- [12] 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].
- [13] Z. T. Inc., "An overview on PHP," Zend The PHP Company, 2007.
- [14] "XML," [Online]. Available: <https://en.wikipedia.org/wiki/XML>. [Accessed 10 June 2016].
- [15] M. Saralita, "Pencarian Relasi Antar Tokoh Sejarah Indonesia Menggunakan Ontologi," 2016.

## LAMPIRAN

**Lampiran Tabel 1 Daftar URL data tokoh**

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

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

Siti Ruby Aliya Radjasa	<a href="http://id.dbpedia.org/data/Siti_Ruby_Aliya_Radjasa">http://id.dbpedia.org/data/Siti_Ruby_Aliya_Radjasa</a>
Jusuf Kalla	<a href="http://id.dbpedia.org/data/Jusuf_Kalla">http://id.dbpedia.org/data/Jusuf_Kalla</a>
Mufidah Jusuf Kalla	<a href="http://id.dbpedia.org/data/Mufidah_Jusuf_Kalla">http://id.dbpedia.org/data/Mufidah_Jusuf_Kalla</a>
Solihin Kalla	<a href="http://id.dbpedia.org/data/Solihin_Kalla">http://id.dbpedia.org/data/Solihin_Kalla</a>
Joko Widodo	<a href="http://id.dbpedia.org/data/Joko_Widodo">http://id.dbpedia.org/data/Joko_Widodo</a>





Lampiran Gambar 2 Individu Fatmawati setelah reasoning

untitled-ontology-147 (http://www.semanticweb.org/asus/ontologies/2019/5/untitled-ontology-147) : [D:\The Tree of Heroes\Preprocessor\1A.Result...]

File Edit View Reasoner Tools Refactor Window Help

Active Ontology x Entities x Individuals by class x DL Query x

Annotation properties Individuals

Classes Object properties Data properties

Individuals: Fatmawati

Annotations +

rdf:type [language: id]

Description: Fatmawati

Property assertions: Fatmawati

Types +

- dbo:Agent
- dbo:OfficeHol
- dbo:Person
- foaf:Person
- owl:Thing
- schema:Persc

Same individual As +

Different individuals +

Individual Annotations Individual Usage

Annotations: Fatmawati

Property assertions: Fatmawati

dbp:issue Guntur\_Soekarnoputra

isParentOf Guntur\_Soekarnoputra

isSpouseOf Soekarno

hasChild Guntur\_Soekarnoputra

hasGrandChild Puan\_Maharani

isParentOf Megawati\_Soekarnoputri

hasChild Sukmawati\_Soekarnoputri

dbp:child Sukmawati\_Soekarnoputri

hasChild Megawati\_Soekarnoputri

isParentOf Guntur\_Soekarnoputra

hasGrandChild Surindro\_Supjarso

dbp:child Guntur\_Soekarnoputra

dbp:issue Guntur\_Soekarnoputra

dbo:spouse Soekarno

dbp:issue Rachmawati\_Soekarnoputri

hasChild Rachmawati\_Soekarnoputri

isParentOf Sukmawati\_Soekarnoputri

dbp:children Sukmawati\_Soekarnoputri

dbp:children Guntur\_Soekarnoputra

Abdurrahman Wahid

Agus\_Harimurti\_Yudhoyono

Amnisa\_Polhan

Aulia\_Pohan

Bacharuddin\_Jusuf\_Habibie

Bambang\_Trihatmodjo

Bayu\_Soekarnoputra

Edhie\_Baskoro\_Yudhoyono

Fatmawati

Fatmawati\_Soekarno

Guntur\_Soekarnoputra

Guruh\_Soekarnoputra

Gusyenova Sabina Padmawati

Halimah\_Agustina\_Kanili

Hartini

Harati

Hasri\_Almin\_Habibie

Hassan\_Gamal\_Ahmad\_Hasan

Heldi Djafar

Hutomo\_Mandala\_Putra

Iltam\_Akbar

Inayati\_Wulandari

Indra\_Rukmana

Inggit\_Garnasih

Joko\_Widodo

Jusuf\_Kalla

**Lampiran Gambar 3 Individu Susilo Bambang Yudhoyono sebelum reasoning**

[illegible]

Lampiran Gambar 4 Individu Susilo Bambang Yudhoyono setelah reasoning

untitled-ontology-147 (http://www.semanticweb.org/asus/ontologies/2019/5/untitled-ontology-147) : ID: The Tree of Heroes! Preprocessor TA Result...

File Edit View Reasoner Tools Refactor Window Help

Active Ontology x Entities x Individuals by class x DL Query x

Datatypes Annotation properties Classes Object properties Data properties Individuals Susilo Bambang Yudhoyono

Susilo Bambang Yudhoyono — http://rd.dbpedia.org/resource/Susilo\_Bambang\_Yudhoyono

Individual Annotations Individual Usage Annotations Susilo Bambang Yudhoyono

Annotations + rdfs:label (language: id)

Description: Susilo Bambang Yudhoyono

Types + dbo:Agent dbo:OfficeHolder dbo:Person foaf:Person owl:Thing schema:Person

Same individual as + Different individuals +

Property assertions: Susilo Bambang Yudhoyono

Object property assertions +

dbp:children Agus\_Harimurti\_Yudhoyono

dbo:child Agus\_Harimurti\_Yudhoyono

dbp:issue Agus\_Harimurti\_Yudhoyono

isParentOf Agus\_Harimurti\_Yudhoyono

dbo:spouse Kristiani\_Herrawati

dbp:issue Edhie\_Baskoro\_Yudhoyono

hasChild Agus\_Harimurti\_Yudhoyono

dbp:children Edhie\_Baskoro\_Yudhoyono

hasChild Edhie\_Baskoro\_Yudhoyono

dbo:child Edhie\_Baskoro\_Yudhoyono

isSpouseOf Kristiani\_Herrawati

isParentOf Edhie\_Baskoro\_Yudhoyono

Data property assertions +

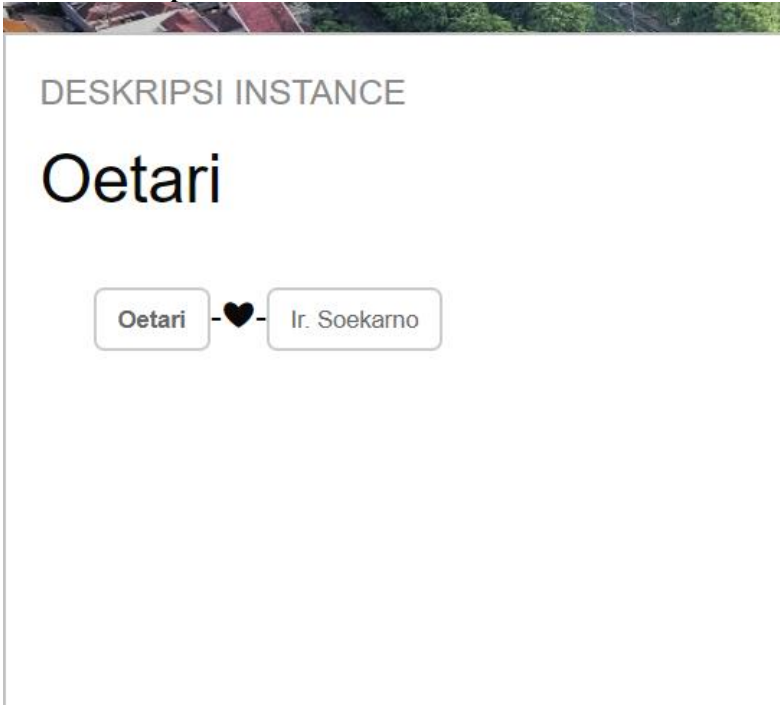
Negative object property assertions +

Negative data property assertions +

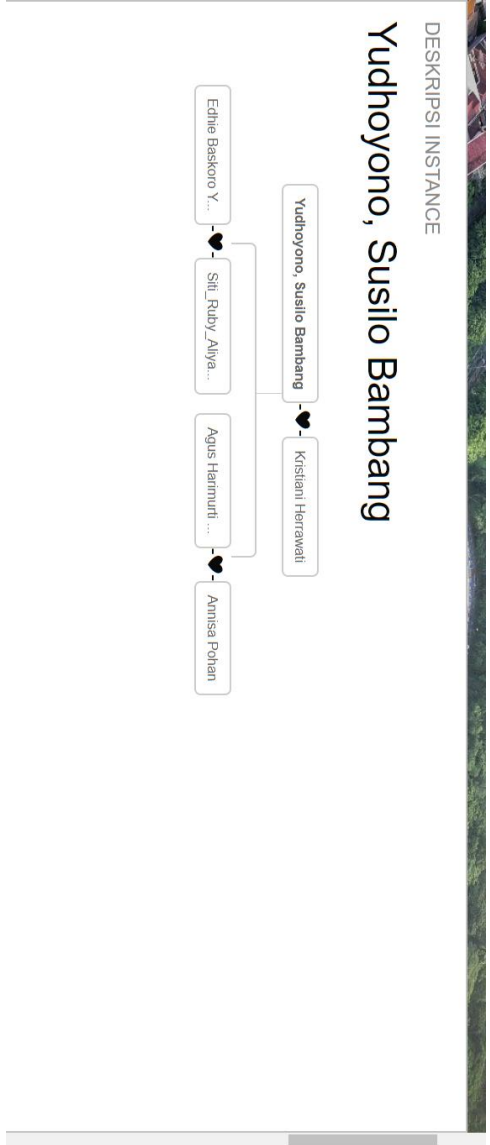
File < >

---



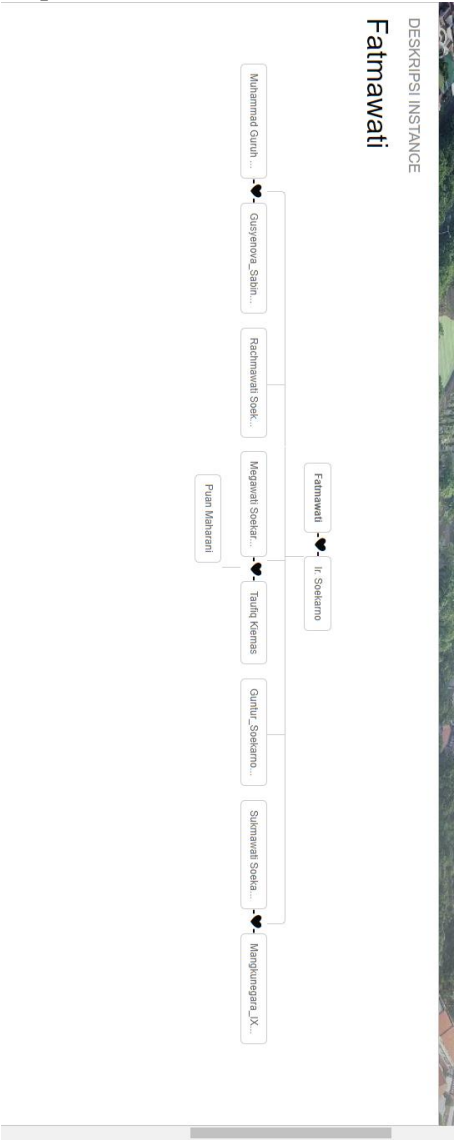
**Lampiran Gambar 7 Kasus tidak memiliki anak**

Lampiran Gambar 8 Kasus memiliki anak

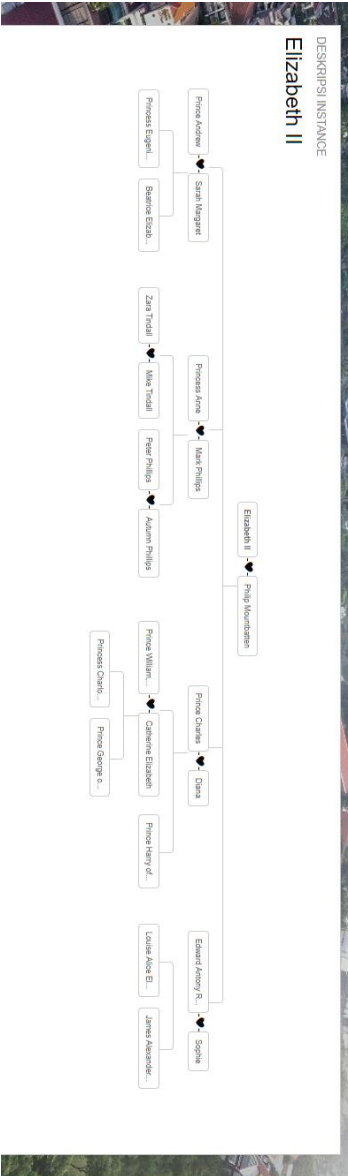




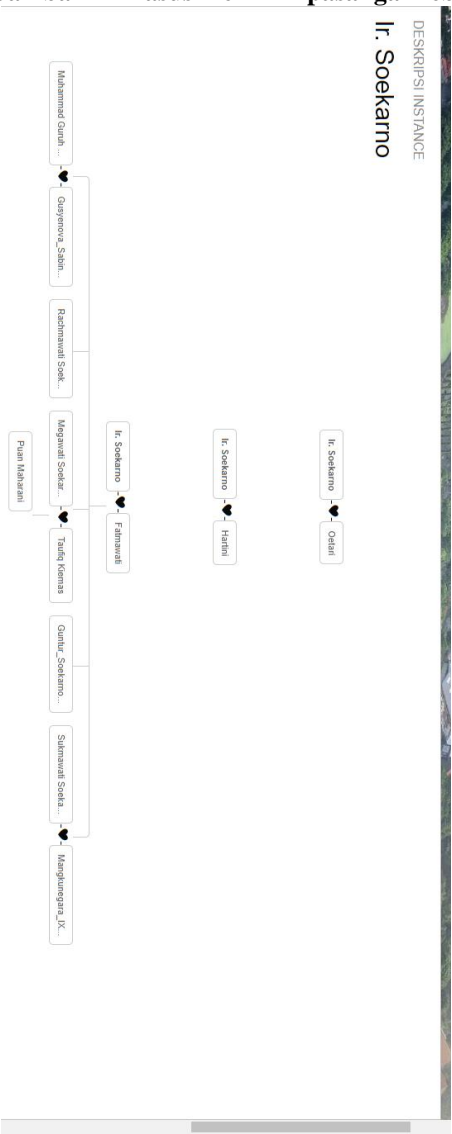
Lampiran Gambar 9 Kasus memiliki cucu

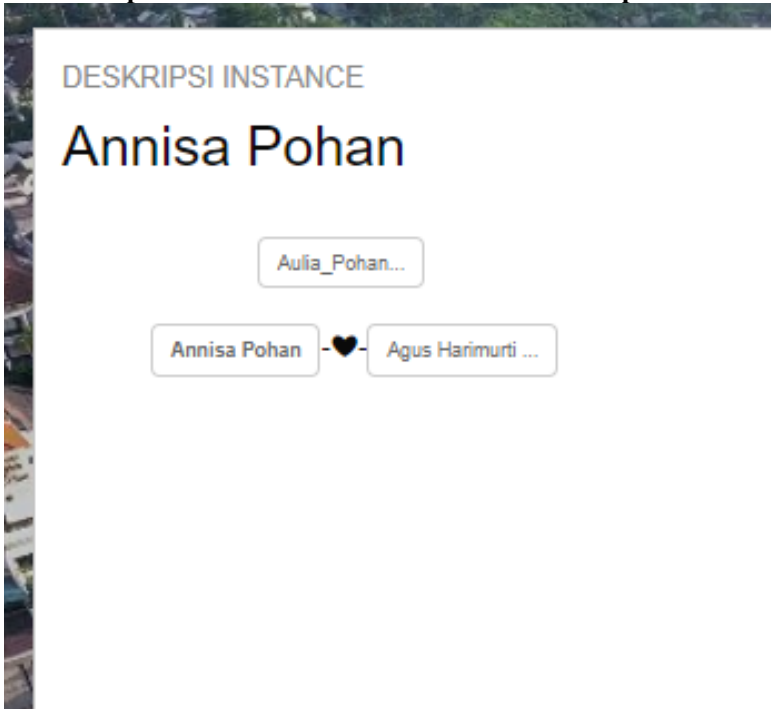


Lampiran Gambar 10 Kasus memiliki cicit



Lampiran Gambar 11 Kasus memiliki pasangan lebih dari satu



**Lampiran Gambar 12 Kasus memiliki relasi tanpa nam**

*(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](mailto:karyoutomoo@gmail.com).

