

**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**

2016

# Description: its_logo_gifHALAMAN JUDUL

**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)***

Halaman Judul



**FINAL PROJECT – KI141502**

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

**FAIQ**

**NRP. 5115 100 007**

**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)

1. 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 bersejarah dan pahlawan nasional, keduanya 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.*

*Dengan adanya data keluarga dari suatu halaman DBpedia, hubungan antar tokoh-tokoh tersebut dapat diketahui. Keterkaitan atau relasi tokoh bersejarah dapat digambarkan dengan ontologi. Tujuan dari pengerjaan tugas akhir ini adalah untuk melengkapi data tokoh dengan proses reasoning lalu menyimpan data tersebut dalam suatu basis data sehingga bisa ditampilkan secara grafis hubungan keluarga tokoh bersejarah 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: Visualisasi, Family Tree, Tokoh Sejarah Indonesia, Ontologi.***

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

**Name : Faiq**

**NRP : 5115100007**

**Department : Informatics FTIK-ITS**

**First Advisor : Nurul Fajrin A.,S.Kom., M.Sc.**

**Second Advisor : Adhatus Solichah A.,S.Kom., M.Sc.**

# ABSTRACT

*Historical figures and national heroes, both are 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.*

*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 historical figures of Indonesia 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: Visualization, Family Tree, Indonesia’s National Figures, Ontology.***

# 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 dan Arya.
8. Teman-teman HMTC 2016/2017 dan BEM FTIK 2016/2017 - 2017/2018.
9. Seluruh keluarga TC 2015 yang selalu menemani dan memberi semangat selama 4 tahun perkuliahan.
10. Serta semua pihak yang 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

[1 HALAMAN JUDUL iii](#_Toc12146697)

[3 LEMBAR PENGESAHAN vii](#_Toc12146698)

[4 ABSTRAK ix](#_Toc12146699)

[5 ABSTRACT xi](#_Toc12146700)

[6 KATA PENGANTAR xiii](#_Toc12146701)

[7 DAFTAR ISI xv](#_Toc12146702)

[9 DAFTAR GAMBAR xviii](#_Toc12146703)

[DAFTAR TABEL xx](#_Toc12146704)

[DAFTAR KODE SUMBER xxiii](#_Toc12146705)

[1 BAB I PENDAHULUAN 1](#_Toc12146706)

[1.1. Latar Belakang 1](#_Toc12146707)

[1.2. Batasan Masalah 2](#_Toc12146708)

[1.3. Tujuan 3](#_Toc12146709)

[1.4. Metodologi 3](#_Toc12146710)

[1.5. Sistematika Penulisan 4](#_Toc12146711)

[2 BAB II DASAR TEORI 7](#_Toc12146712)

[2.1. Tokoh Bersejarah 7](#_Toc12146713)

[2.2. Ontologi 7](#_Toc12146714)

[2.3. DBpedia 10](#_Toc12146715)

[2.4. Semantic Web Rule Language (SWRL) 10](#_Toc12146716)

[2.5. Family Relationships Ontology 11](#_Toc12146717)

[2.6. SPARQL 13](#_Toc12146718)

[2.7. Apache Jena Fuseki 14](#_Toc12146719)

[2.8. Pellet Reasoner 15](#_Toc12146720)

[2.9. SPARQL Lib 17](#_Toc12146721)

[2.10. Apache Jena 18](#_Toc12146722)

[3 BAB III METODOLOGI PEMECAHAN MASALAH 19](#_Toc12146723)

[3.1. Analisis Data 20](#_Toc12146724)

[3.1.1. Analisis Data dari DBpedia 21](#_Toc12146725)

[3.2. Ekstraksi Data Sebagai Model 23](#_Toc12146726)

[3.3. Pembuatan Ontologi 23](#_Toc12146727)

[3.4. Penggabungan model data dan model ontologi 25](#_Toc12146728)

[3.5. Reasoning pada Model Gabungan 26](#_Toc12146729)

[3.6. Penampilan Data 26](#_Toc12146730)

[4 BAB IV ANALISIS DAN PERANCANGAN SISTEM 27](#_Toc12146731)

[4.1. Analisis 27](#_Toc12146732)

[4.1.1. Cakupan Permasalahan 27](#_Toc12146733)

[4.1.2. Deskripsi Umum Sistem 27](#_Toc12146734)

[4.1.3. Spesifikasi Kebutuhan Perangkat Lunak 28](#_Toc12146735)

[4.1.4. Aktor 28](#_Toc12146736)

[4.1.5. Kasus Penggunaan 29](#_Toc12146737)

[4.2. Perancangan Antarmuka Pengguna 34](#_Toc12146738)

[5 BAB V IMPLEMENTASI 37](#_Toc12146739)

[5.1. Implementasi Preprocessor 38](#_Toc12146740)

[5.2. Implementasi Fungsi 42](#_Toc12146741)

[5.2.1. Fungsi Dropdown Select 42](#_Toc12146742)

[5.2.2. Fungsi Get Family 43](#_Toc12146743)

[5.3. Implementasi Antarmuka Pengguna 59](#_Toc12146744)

[5.3.1. Implementasi Tampilan Halaman Utama 59](#_Toc12146745)

[5.3.2. Implementasi Tampilan Halaman Pohon Keluarga 59](#_Toc12146746)

[6 BAB VI PENGUJIAN DAN EVALUASI 61](#_Toc12146747)

[6.1. Lingkungan Pengujian 61](#_Toc12146748)

[6.2. Skenario Pengujian 61](#_Toc12146749)

[6.2.1. Pengujian Perbandingan Data 62](#_Toc12146750)

[6.3. Evaluasi Pengujian 92](#_Toc12146751)

[6.3.1. Evaluasi Pengujian Perbandingan Data 92](#_Toc12146752)

[7 BAB VII KESIMPULAN DAN SARAN 94](#_Toc12146753)

[**1.** 94](#_Toc12146754)

[7.1. Kesimpulan 94](#_Toc12146755)

[7.2. Saran 94](#_Toc12146756)

[DAFTAR PUSTAKA 96](#_Toc12146757)

[8 LAMPIRAN A. DATA SEBELUM *REASONING* 100](#_Toc12146758)

[9 LAMPIRAN B. DATA SETELAH *REASONING* 112](#_Toc12146759)

[10 BIODATA PENULIS 136](#_Toc12146760)

***[Halaman ini sengaja dikosongkan]***

# 

# DAFTAR GAMBAR

[Gambar 2.1 Class Hierarchy 8](#_Toc12146796)

[Gambar 2.2 Property 9](#_Toc12146797)

[Gambar 2.3 Class, Property dan Instance 9](#_Toc12146798)

[Gambar 2.4 Contoh Halaman DBpedia 10](#_Toc12146799)

[Gambar 2.5 Ontologi FamilyTree Keluarga Robert Stevens 13](#_Toc12146800)

[Gambar 2.6 Contoh SPARQL Query 14](#_Toc12146801)

[Gambar 2.7 Database Triple Store Apache Jena Fuseki 14](#_Toc12146802)

[Gambar 2.8 Daftar API Apache Jena Fuseki 15](#_Toc12146803)

[Gambar 2.9 Arsitektur Pellet Reasoner 15](#_Toc12146804)

[Gambar 2.10 Contoh Penggunaan SPARQL Lib 18](#_Toc12146805)

[Gambar 3.1 Flowchart pengembangan 19](#_Toc12146806)

[Gambar 3.2 Halaman DBpedia Indonesia tentang Ken Arok 21](#_Toc12146807)

[Gambar 3.3 Halaman DBpedia tentang Ratu Elizabeth II 22](#_Toc12146808)

[Gambar 3.4 Hirarki Class 24](#_Toc12146809)

[Gambar 3.5 Hirarki Data Property 24](#_Toc12146810)

[Gambar 3.6 Hirarki Object Property 25](#_Toc12146811)

[Gambar 4.1 Diagram Kasus Penggunaan Sistem 29](#_Toc12146812)

[Gambar 4.2 Diagram Aktivitas Memilih Entitas Tokoh 31](#_Toc12146813)

[Gambar 4.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh 32](#_Toc12146814)

[Gambar 4.4 Diagram Aktivitas Memilih Entitas Tautan 34](#_Toc12146815)

[Gambar 4.5 Antarmuka Halaman Utama Family Tree App 35](#_Toc12146816)

[Gambar 4.6 Antarmuka Halaman Pohon Keluarga Family Tree App 35](#_Toc12146817)

[Gambar 5.1 Arsitektur Perangkat Lunak 37](#_Toc12146818)

[Gambar 5.2 Implementasi Antarmuka Halaman Utama 59](#_Toc12146819)

[Gambar 5.3 Implementasi Antarmuka Halaman Pohon Keluarga 60](#_Toc12146820)

# DAFTAR TABEL

[Tabel 2.1 Komponen SWRL 11](#_Toc12146762)

[Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak 28](#_Toc12146763)

[Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan 30](#_Toc12146764)

[Tabel 4.3 Spesifikasi Kasus Penggunaan Memilih Entitas Tokoh 30](#_Toc12146765)

[Tabel 4.4 Spesifikasi Kasus Penggunaan Melihat Informasi Tokoh 31](#_Toc12146766)

[Tabel 4.5 Spesifikasi Kasus Penggunaan Memilih Entitas Tautan 33](#_Toc12146767)

[Tabel 4.6 Spesifikasi Atribut Rancangan Antarmuka Halaman Family Tree App 35](#_Toc12146768)

[Tabel 6.1 Pengujian Perbandingan Data Mark Phillips 62](#_Toc12146769)

[Tabel 6.2 Pengujian Perbandingan Data Prince Andrew, Duke of York 63](#_Toc12146770)

[Tabel 6.3 Pengujian Perbandingan Data Meghan Markle 64](#_Toc12146771)

[Tabel 6.4 Pengujian Perbandingan Data Diana, Princess of Wales 66](#_Toc12146772)

[Tabel 6.5 Pengujian Perbandingan Data Sophie, Countess of Wessex 67](#_Toc12146773)

[Tabel 6.6 Pengujian Perbandingan Data Prince Philip, Duke of Edinburgh 68](#_Toc12146774)

[Tabel 6.7 Pengujian Perbandingan Data Sarah, Duchess of York 69](#_Toc12146775)

[Tabel 6.8 Pengujian Perbandingan Data Prince William, Duke of Cambridge 70](#_Toc12146776)

[Tabel 6.9 Pengujian Perbandingan Data Anne, Princess Royal 71](#_Toc12146777)

[Tabel 6.10 Pengujian Perbandingan Data Autumn Phillips 73](#_Toc12146778)

[Tabel 6.11 Pengujian Perbandingan Data Charles, Prince of Wales 74](#_Toc12146779)

[Tabel 6.12 Pengujian Perbandingan Data Camilla, Duchess of Cornwall 75](#_Toc12146780)

[Tabel 6.13 Pengujian Perbandingan Data Princess Charlotte of Cambridge 76](#_Toc12146781)

[Tabel 6.14 Pengujian Perbandingan Data Elizabeth II 77](#_Toc12146782)

[Tabel 6.15 Pengujian Perbandingan Data Prince Harry 78](#_Toc12146783)

[Tabel 6.16 Pengujian Perbandingan Data Catherine, Duchess of Cambridge 80](#_Toc12146784)

[Tabel 6.17 Pengujian Perbandingan Data Prince Edward, Earl of Wessex 81](#_Toc12146785)

[Tabel 6.18 Pengujian Perbandingan Data Zara Phillips 82](#_Toc12146786)

[Tabel 6.19 Pengujian Perbandingan Data Prince George of Cambridge 83](#_Toc12146787)

[Tabel 6.20 Pengujian Perbandingan Data Mike Tindall 84](#_Toc12146788)

[Tabel 6.21 Pengujian Perbandingan Data Princess Eugenie 85](#_Toc12146789)

[Tabel 6.22 Pengujian Perbandingan Data Lady Louise 86](#_Toc12146790)

[Tabel 6.23 Pengujian Perbandingan Data Peter Phillips 88](#_Toc12146791)

[Tabel 6.24 Pengujian Perbandingan Data Timothy Laurence 89](#_Toc12146792)

[Tabel 6.25 Pengujian Perbandingan Data Princess Beatrice 90](#_Toc12146793)

[Tabel 6.26 Pengujian Perbandingan Data James, Viscount Severn 91](#_Toc12146794)

[Tabel 6.27 Rangkuman Hasil Pengujian 92](#_Toc12146795)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR KODE SUMBER

[Kode Sumber 3.1 Kode Java untuk memodelkan data Ratu Elizabeth II ke dalam modelActor 23](#_Toc12146821)

[Kode Sumber 5.1 Implementasi proses ekstraksi, penggabungan dan reasoning 42](#_Toc12146822)

[Kode Sumber 5.2 Kode Sumber SPARQL untuk mengambil value bertipe Person dan Fungsi Dropdown Select 43](#_Toc12146823)

[Kode Sumber 5.3 Fungsi Get name 44](#_Toc12146824)

[Kode Sumber 5.4 Fungsi Get father 46](#_Toc12146825)

[Kode Sumber 5.5 Fungsi Get mother 47](#_Toc12146826)

[Kode Sumber 5.6 Fungsi Get sibling 49](#_Toc12146827)

[Kode Sumber 5.7 Fungsi Get spouse 51](#_Toc12146828)

[Kode Sumber 5.8 Fungsi Get child 52](#_Toc12146829)

[Kode Sumber 5.9 Fungsi Get child in law 54](#_Toc12146830)

[Kode Sumber 5.10 Fungsi Get grand child 55](#_Toc12146831)

[Kode Sumber 5.11 get grand child in law 57](#_Toc12146832)

[Kode Sumber 5.12 Get great grand child 58](#_Toc12146833)

# 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.

## Latar Belakang

Tokoh bersejarah adalah seseorang yang namanya dikenang karena jasanya. Sedangkan pahlawan adalah gelar penghargaan yang diberikan kepada seseorang atas tindakan heroiknya. Tokoh bersejarah dan pahlawan, 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, sedangkan otak manusia dapat memproses informasi visual 60.000 kali lebih cepat daripada informasi teks (Burmark, L., 2008). Berdasarkan ontologi yang telah disebutkan sebelumnya, pengerjaan tugas akhir ini akan mengembangkan ontologi yang sudah ada dengan mengkombinasikan *class* dan *property* yang dimilikinya dan ditampilkan dalam sebuah situs web untuk memudahkan pemahaman terkait tokoh bersejarah dan relasinya. 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 family tree tokoh?

## Batasan Masalah

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

1. Data yang digunakan adalah tokoh bersejarah dari DBpedia.
2. Data bersumber dari artikel Wikipedia mengenai kerajaan Inggris.
3. Platform pengembangan aplikasi adalah situs web.
4. Data yang digunakan sebagai *value* properti bersumber dari isi propertiDBpedia dan hasil esktraksi manual pada halaman Wikipedia Inggris *person* terkait.
5. Aplikasi tidak dapat menangani *person* yang memiliki *alias* yang banyak.
6. Batas relasi adalah ayah, ibu, saudara, istri, anak, dan cucu
7. Proses pelengkapan data dilakukan menggunakan Protege 5.2.0 yang memiliki ekstensi OWL dan Pellet *Reasoner*.
8. *Reasoner* yang digunakan adalah Pellet.
9. Aplikasi yang dibuat tidak menyediakan *form* untuk pengelolaan data (tambah, ubah, hapus).
10. Pengelolaan data yang berupa penambahan, penghapusan, dan pengubahan data hanya dapat dilakukan dengan *tools* Protege.
11. Aplikasi yang dibuat hanya untuk menampilkan deskripsi *person* yang merupakan hasil dari ontologi yang dibangun.

## Tujuan

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

## 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*.

1. 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 menampikan hasil pencarian relasi dibangun dengan bahasa PHP menggunakan *tools* PhpStorm.

1. 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.

1. 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.

## 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.

1. **Pendahuluan**

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

1. **Dasar Teori**

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

1. **Metode Pemecahan Masalah**

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

1. **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.

1. **Implementasi**

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

1. **Pengujian dan Evaluasi**

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*)dari perangkat lunak dan pengujian fungsionalitas yang dibuat dengan memperhatikan keluaran yang dihasilkan serta evaluasi terhadap fitur-fitur perangkat lunak.

1. **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.

## Tokoh Bersejarah

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 jasa atau posisinya. Setiap tokoh bersejarah memiliki pengalaman hidup yang berbeda-beda. Dalam biografi tentang tokoh sejarah sering kita temui nama-nama keluarga dari seorang tokoh. Biografi adalah deskripsi detail dari kehidupan seseorang dari lahir sampai meninggal dunia. Setiap jasa atau karya yang dihasilkan setiap toko sejarah dicatat dalam biografinya. Setiap tokoh sejarah memiliki perjalanan hidup masing-masing dan menjalani hidup dengan orang yang berbeda-beda.

## Ontologi

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

- *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

(Sumber : *Introduction to Ontology Concepts and Terminology, Steven J. Miller, 2013)*

- *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 **Error! Reference source not found.**



Gambar 2.2 Property

*- 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



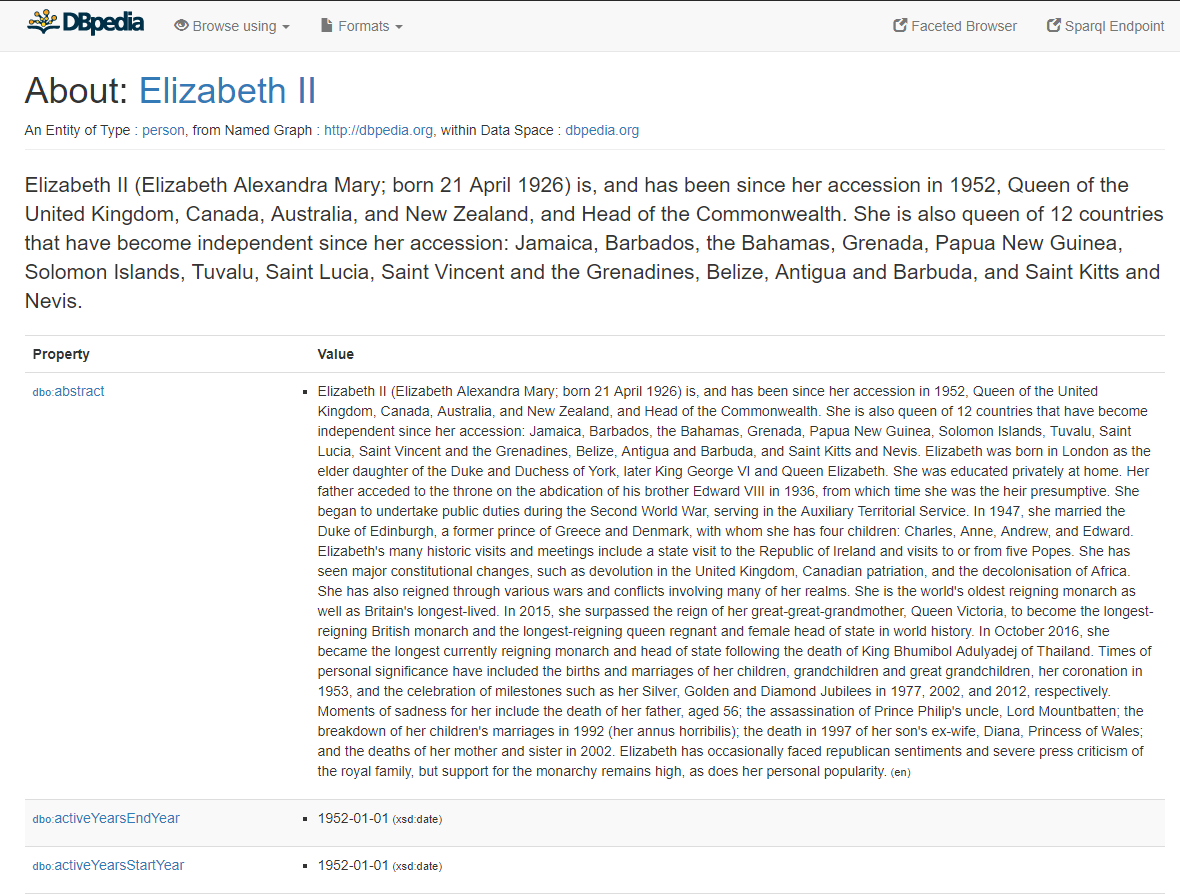
Gambar 2.3 Class, Property dan Instance

Selain 3 komponen penting yang telah dijelaskan di atas, terdapat beberapa istilah lain yang perlu dipahami dalam konteks ontologi antara lain d*omain* (*member* dari suatu kelas yang dapat menjadi subjek dari *property* yang diberikan), r*ange* (member dari suatu kelas yang dapat menjadi objek dari *property* yang diberikan), c*onstraint* 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 standarseperti RDF, RDFS, atau OWL. Secara umum, kegunaan ontologi adalah sebagai *controlled vocabulary, semantic interoperability, knowledge sharing,* dan *reuse* [1].

## 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 2.4adalah contoh sebuah halaman DBpedia.



Gambar 2.4 Contoh Halaman DBpedia

## 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 [2]. Pada Tabel 2.1 berikut akan dijabarkan bentuk-bentuk atom yang didefinisikan.

Tabel 2.1 Komponen SWRL

| Atom | Deskripsi |
| --- | --- |
| C(x) | C adalah deklarasi *class* (nama *class*) dan x adalah nama individual atau variabel |
| D(y) | D adalah deklarasi *data range* dan y adalah variabel atau *data value* |
| P(x, y) | P adalah data atau *object property*, x dan y adalah variabel atau OWL individual. y adalah sebuah individual jika P adalah *object property*, sedangkan y adalah sebuah *data value* jika P adalah *data property*. |
| 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 “?”.

## Family Relationships Ontology

*Family relationship* umumnya digambarkan dengan terstruktur melalui silsilah keluarga. Manusia membutuhkan informasi tentang silsilah keluarganya untuk berbagai hal, diantaranya adalah untuk memperat 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 [3].

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 umunya hanya satu atau dua orang yang mengetahui detail keluarga. Semakin bertambahnya pengetahuan membuat hubungan dalam sebuah keluarga dapat diketahui dengan mudah melalui *Family Relationships Ontology*. Ontologi ini memiliki beberapa kelebihan, diantaranya adalah dapat diketahuinya keakraban, relasi, pewarisan, *domain, range, constraint,* dan kesimpulan logis dalam sebuah keluarga secara praktis.

Terdapat banyak ontologi yang telah dibangun menggunakan domain keluarga, salah satunya adalah ontologi yang digunakan pada pengerjaan tugas akhir ini, yaitu FamilyTree. Ontologi tersebut didapatkan dari portal The University of Manchester. Ontologi FamilyTree memiliki URI [*http://www.co-ode.org/roberts/family-tree.owl*](http://www.co-ode.org/roberts/family-tree.owl) [4]. 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 padaGambar 2.5.



Gambar 2.5 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 kerajaan Inggris, maka *instance* di ontologi ini dihapus.

## SPARQL

SPARQL (dibaca “sparkle) adalah protokol RDF Query Language yang berfungsi untuk mengambil dan memanipulasi data dari sebuah basis data triple-store. Protokol SPARQL umumnya digunakan oleh peneliti Semantic Web. Contoh *syntax* SPARQL seperti yang ditunjukkan oleh Gambar 2.6:



Gambar 2.6 Contoh SPARQL Query

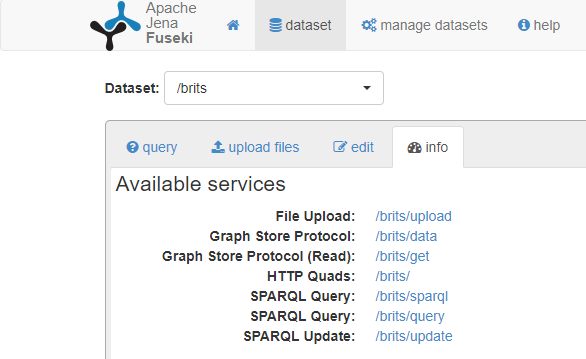
## Apache Jena Fuseki

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



Gambar 2.7 Database Triple Store Apache Jena Fuseki

Apache Jena Fuseki menyediakan beberapa API untuk digunakan oleh peneliti seperti pada Gambar 2.8.



Gambar 2.8 Daftar API Apache Jena Fuseki

## Pellet Reasoner



Gambar 2.9 Arsitektur Pellet Reasoner

*(Sumber: Pellet An OWL DL Reasoner, Bijan Parsia & Evren Sirin)*

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 [5]. 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 2.4menunjukkan komponen utama Pellet *reasoner*.

Ontologi OWL di*parsing* 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 tipepola *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 [6].

## SPARQL Lib

SPARQL Lib adalah sebuah library PHP yang dikembangkan oleh departemen Computer Science dari University of Southampton, United Kingdom yang berfungsi untuk mengolah data bertipe RDF dalam aplikasi berbasis PHP. Dalam konteks ini, SPARQL Lib digunakan untuk mengambil data RDF dari basis data triple store melalui panggilan API. Contoh dasar penggunaan SPARQL Lib dalam proyek berbasis PHP adalah seperti Gambar 2.9



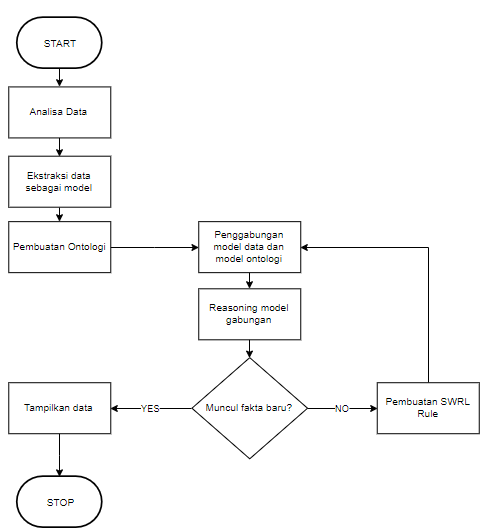
Gambar 2.10 Contoh Penggunaan SPARQL Lib

## Apache Jena

Apache Jena adalah *framework 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 *framework* 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.

# 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

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 kerajaan Inggris dikarenakan kurangnya relasi tokoh 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.

## Analisis Data

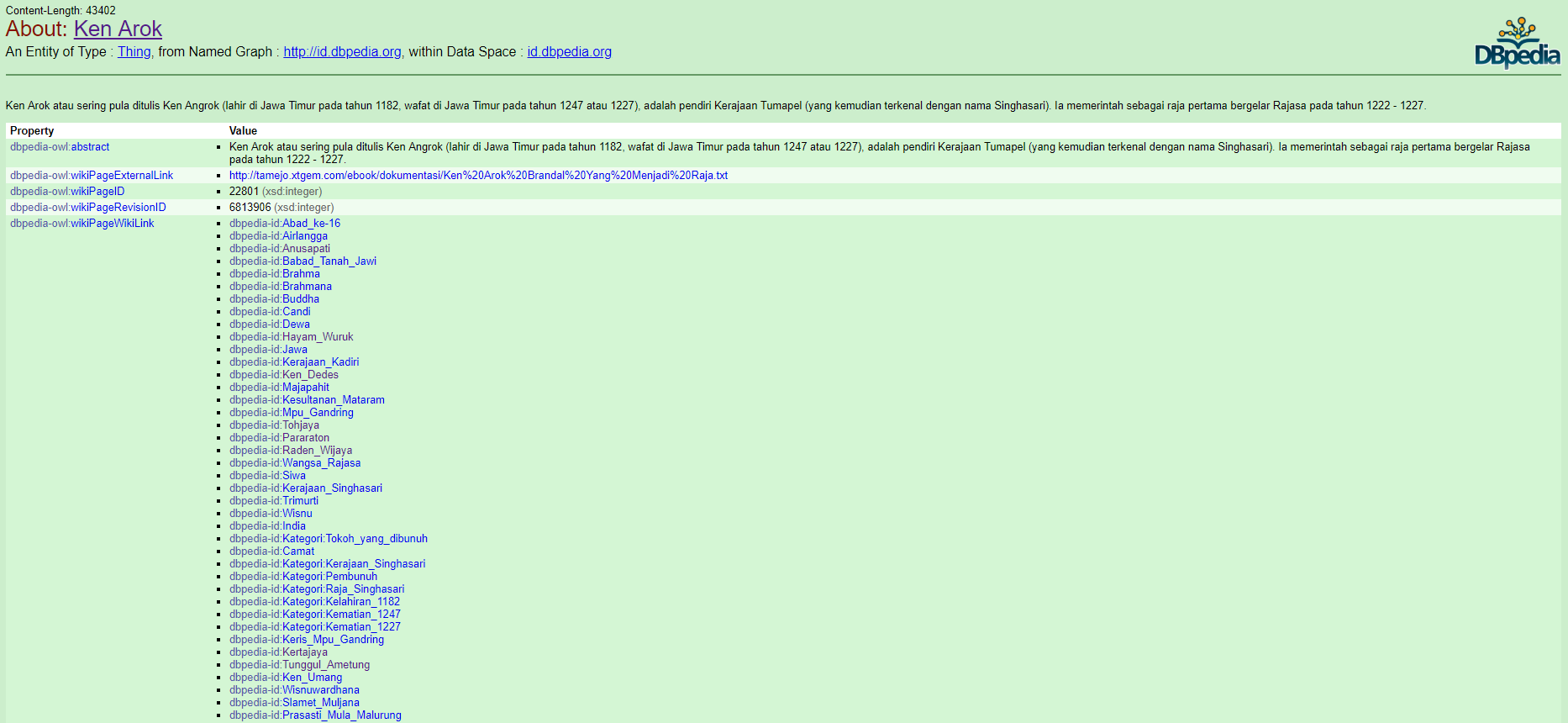
Untuk memecahkan masalah pencarian relasi keluarga tokoh bersejarah, langkah yang pertama kali dilakukan adalah menganalisis dataset yang akan digunakan. Ontologi memiliki beberapa domain, yaitu *actor, place, time, dan event*. Domain yang menjadi topik pada pengerjaan tugas akhir ini adalah *actor*. 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 kerajaan Inggris Raya. Daftar keluarga kerajaan Inggris raya dapat diperoleh dari Wikipedia Indonesia yang dapat diakses pada halaman *https://en.wikipedia.org/wiki/Family\_tree\_of\_the\_British\_royal\_family*. Terdapat tiga *house* (keluarga besar) dalam sejarah kerajaan Inggris, namun yang menjadi objek utama untuk dijadikan data adalah House Windsor yang saat ini masih memimpin Inggris Raya. *Person* yang digunakan misalnya adalah Elizabeth II, Prince Charles, Catherine Middleton. Setiap *person* memiliki biografi terkait dengan *place* (misalnya *BirthPlace* dan *DeathPlace*), *time* (misalnya *BirthDate* dan *DeathDate*), dan *event* (misalnya kegiatan politik atau peperangan).

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 [7].

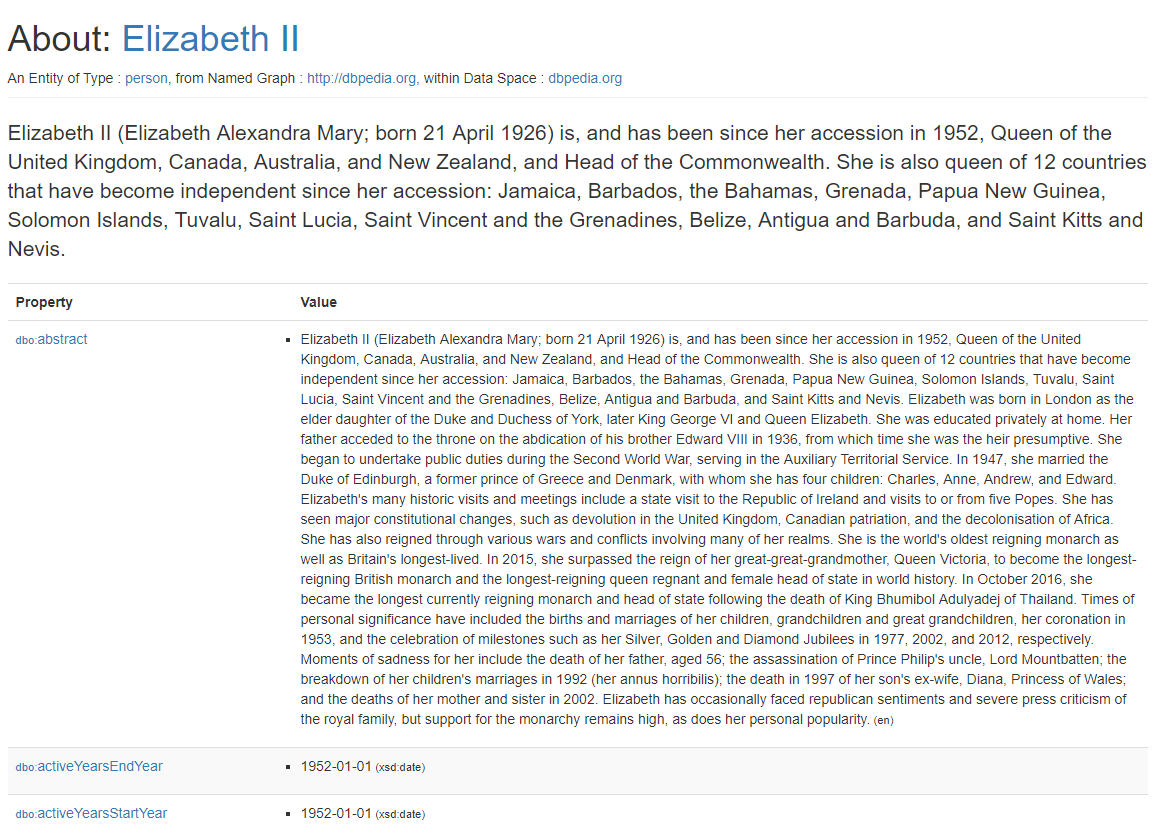
### Analisis Data dari DBpedia

Terdapat berbagai macam *open data* yang dapat diakses melalui *internet* tanpa berbayar. Pada awalnya, data yang diekstrak dari halaman DBpedia Indonesia adalah data kerajaan Indonesia. Sebagai contoh, pada Gambar 3.2 terdapat daftar seluruh informasi yang dimiliki DBpedia Indonesia tentang Ken Arok. Contoh informasi tersebut dapat diakses melalui alamat [*https://id.dbpedia.org/wiki/Ken\_Arok*](https://id.dbpedia.org/wiki/Ken_Arok)*.*



Gambar 3.2 Halaman DBpedia Indonesia tentang Ken Arok

Hasil ekstrak manual data dari halaman DBpedia Indonesia seperti dijabarkan pada ternyata memunculkan permasalahan baru, yaitu tidak tercatatnya Ken Dedes sebagai property *spouse* dari Ken Arok yang tidak memungkinkan untuk mencari fakta baru, padahal menurut halaman Wikipedia Indonesia Ken Arok (*https://id.wikipedia.org/wiki/Ken\_Arok*) tercatat bahwa Ken Dedes adalah istri dari Ken Arok. Hal ini disebabkan karena dalam DBpedia Indonesia properti yang dipakai terlalu umum seperti *dbpedia-owl:wikiPageWikiLink* padahal seharusnya relasi antara Ken Arok dan Ken Dedes direpresentasikan sebagai *dbpedia-ontology:spouse*. Pada akhirnya data yang akan dipakai sebagai model adalah data kerajaan Inggris karena kelengkapan properti-properti utama yang diperlukan untuk mengetahui silsilah keluarga seorang *person*. Contoh yang akan dipakai adalah data dari Ratu Elizabeth II yang bisa dilihat di Gambar 3.3.



Gambar 3.3 Halaman DBpedia tentang Ratu Elizabeth II

Berdasarkan semua *property* yang terdapat pada halaman DBpedia, dipilih *property* dalam batasan masalah seperti *name, parent, spouse,* dan *issue*(istilah resmi untuk keturunan biologis).

## 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.

|  |
| --- |
| Model modelActor = fManager.loadModel("http://dbpedia.org/data/Elizabeth\_II"+ ".ttl"); |

Kode Sumber 3.1 Kode Java untuk memodelkan data Ratu Elizabeth II ke dalam modelActor

Kode pada Kode Sumber 3.1 akan membaca data properti Ratu Elizabeth serta relasinya. Tentuya tidak hanya Ratu Elizabeth saja yang dijadikan model, tetapi 25 *person* lainnya juga dimodelkan.

## Pembuatan Ontologi

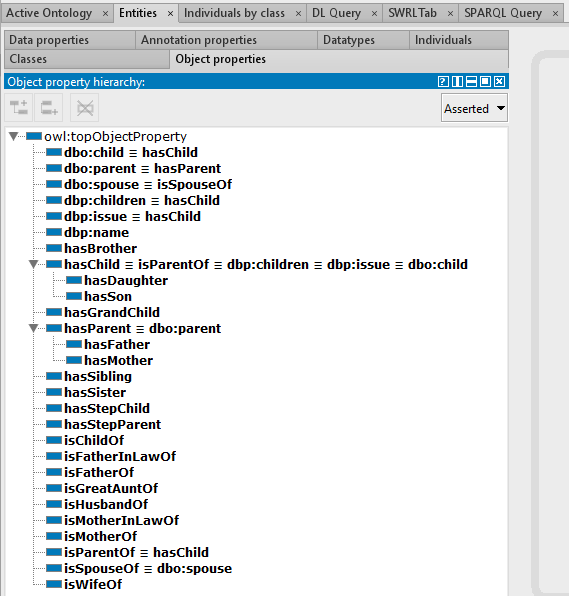
Pada tahap ini, ontologi dibangun dengan menggabungkan beberapa ontologi yang sudah ada. Ontologi yang digunakan adalah *Family Relationships Ontology* milik Robert Stevens. Akan tetapi tidak semua *class, individual, data properties,* ataupun *object properties* akan digunakan, hanya yang benar-benar dibutuhkan saja. Daftar *property* yang akan digunakan adalah :



Gambar 3.4 Hirarki Class



Gambar 3.5 Hirarki Data Property



Gambar 3.6 Hirarki Object Property

## Penggabungan model data dan model ontologi

Penggabungan model data DBpedia yang diperoleh dengan ekstraksi via Apache Jena dan model ontologi *Family Relationship Ontology* yang dibuat menggunakan tools Protege. Penggabungan ini dilakukan dengan menggunakan fungsi createUnion dari class ModelFactory.

## Reasoning pada Model Gabungan

Untuk proses *reasoning*, yang digunakan adalah Pellet Reasoner. Proses ini terdiri dari tiga fase, yaitu *reading*, *classifying*, dan *realizing*. Setelah diperoleh hasilnya, maka hasil tersebut akan diprint menjadi file RDF yang selanjutnya diunggah ke basis data Apache Jena Fuseki.

## Penampilan Data

Untuk penampilan data, ada tiga bagian, yaitu

# 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.

## Analisis

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

### Cakupan Permasalahan

Permasalahan yang diangkat dalam tugas akhir ini adalah visualisasi pohon keluarga tokoh kerajaan Inggris. 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.

### 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. 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.

### 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.

#### 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**.**

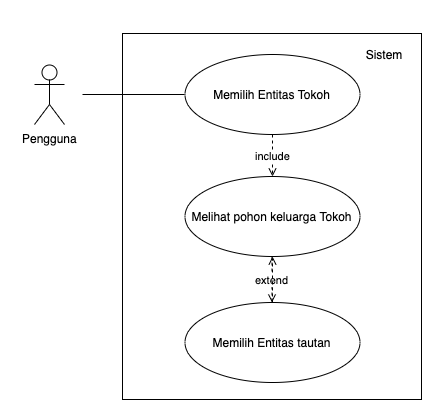
Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak

| **Kode Kebutuhan** | **Kebutuhan Fungsional** | **Deskripsi** |
| --- | --- | --- |
| TA-F0001 | Memilih entitas tokoh | Pengguna dapat memilih entitas tokoh yang ingin dilihat informasinya |
| TA-F0002 | Menampilkan pohon keluarga tokoh | Pengguna dapat melihat informasi pohon keluarga dari entitas yang dipilih |
| TA-F0003 | Memilih entitas tautan | Pengguna dapat memilih entitas tautan yang ingin dilihat informasinya |

### 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.

### Kasus Penggunaan



Gambar 4.1 Diagram Kasus Penggunaan Sistem

Kasus penggunaaan 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 4.1. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel 4.2.

Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan

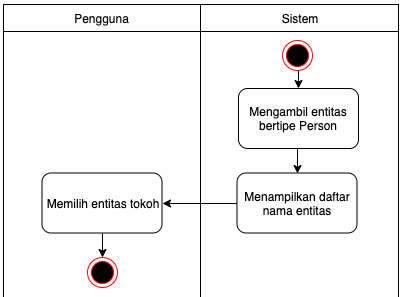
|  |  |
| --- | --- |
| **Kode Kasus Penggunaan** | **Nama** |
| TA-UC0001 | Memilih entitas tokoh |
| TA-UC0002 | Melihat informasi tokoh |
| TA-UC0003 | Memilih entitas tautan |

#### Memilih Entitas Tokoh

Pada kasus penggunaan ini, pengguna dapat memilih entitas tokoh yang ingin dilihat informasinya. Sistem melakukan *request* ke API Apache Jena Fuseki. Entitas bertipe *person* yang terdapat dalam Apache Jena Fuseki selanjutnya ditampilkan dalam *dropdown select*. Spesifikasi kasus penggunaannya dapat dilihat pada Tabel 4.3. Diagram aktivitasnya dapat dilihat pada Gambar 4.2.

Tabel 4.3 Spesifikasi Kasus Penggunaan Memilih Entitas Tokoh

|  |  |
| --- | --- |
| **Nama** | Memilih entitas tokoh |
| **Kode** | TA-UC0001 |
| **Deskripsi** | Memilih entitas tokoh sejarah Indonesia yang ingin dilihat informasinya oleh pengguna |
| **Tipe** | Fungsional |
| **Pemicu** | Pengguna menekan dan memilih *dropdown select* |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Sudah terdapat data para tokoh dalam Apache Jena Fuseki |
| **Aliran:**   * **Kejadian Normal** |  |
| 1. Sistem mengambil entitas dalam basis data Apache Jena Fuseki yang bertipe *person*. 2. Sistem menampilkan daftar nama tokoh yang dapat dipilih oleh pengguna. 3. Pengguna memilih pahlawan melalui *dropdown select*. |
| **Kondisi Akhir** | Sistem menampilkan daftar nama tokoh yang dapat dipilih dalam bentuk *dropdown select* |
| **Kebutuhan Khusus** | Tidak ada |



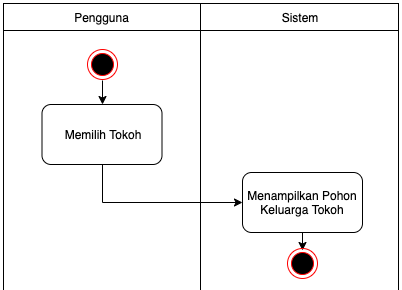
Gambar 4.2 Diagram Aktivitas Memilih Entitas Tokoh

#### Melihat Informasi Tokoh

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 4.4Tabel 4.3. Diagram aktivitasnya dapat dilihat pada Gambar 4.4.

Tabel 4.4 Spesifikasi Kasus Penggunaan Melihat Informasi Tokoh

|  |  |
| --- | --- |
| **Nama** | Melihat pohon keluarga tokoh |
| **Kode** | TA-UC0002 |
| **Deskripsi** | Memilih entitas tautan yang ingin dilihat informasinya oleh pengguna |
| **Tipe** | Fungsional |
| **Pemicu** | Pengguna menekan tombol *submit* |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Sudah terdapat data para tokoh dalam Apache Jena Fuseki |
| **Aliran:**   * **Kejadian Normal** |  |
| 1. Pengguna memilih tokoh melalui *dropdown select*. 2. Pengguna menekan tombol *submit.* 3. Sistem menampilkan halaman pohon keluarga dari tokoh yang dipilih. |
| **Kondisi Akhir** | Sistem menampilkan informasi dari tokoh yang dipilih dalam bentuk pohon keluarga |
| **Kebutuhan Khusus** | Tidak ada |



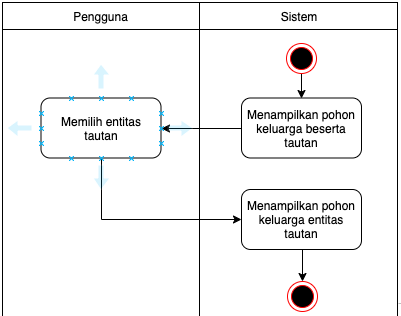
Gambar 4.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh

#### Memilih Entitas Tautan

Pada kasus penggunaan ini, pengguna hanya bisa memilih tautan setelah halaman informasi dari tokoh yang dipilih pada *dropdown select* ditampilkan. Tampilan informasi pada halaman bersumber dari basis data Apache Jena Fuseki yang sudah dikonversi menjadi sebuah halaman HTML. Spesifikasi kasus penggunaannya dapat dilihat pada Tabel 4.5. Diagram aktivitasnya dapat dilihat pada Gambar 4.4.

Tabel 4.5 Spesifikasi Kasus Penggunaan Memilih Entitas Tautan

|  |  |
| --- | --- |
| **Nama** | Memilih entitas tautan |
| **Kode** | TA-UC0003 |
| **Deskripsi** | Menampilkan informasi entitas dari tautan terpilih yang disajikan dalam sebuah halaman HTML |
| **Tipe** | Fungsional |
| **Pemicu** | Pengguna memilih tautan |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Sudah berada pada halaman pohon keluarga tokoh yang dipilih sebelumnya. |
| **Aliran:**   * **Kejadian Normal** |  |
| 1. Sistem menampilkan halaman pohon keluarga dari tokoh yang dipilih sebelumnya. 2. Pengguna memilih tautan pada halaman pohon keluarga. 3. Sistem menampilkan halaman pohon keluarga dari tautan yang dipilih. |
| * **Kejadian Alternatif** | 2.1 Pengguna memilih tokoh melalui *dropdown select*.  Pengguna menekan tombol *submit.*  Kembali ke langkah 1. |
| **Kondisi Akhir** | Sistem menampilkan informasi dari tautan yang dipilih dalam bentuk pohon keluarga |
| **Kebutuhan Khusus** | Tidak ada |

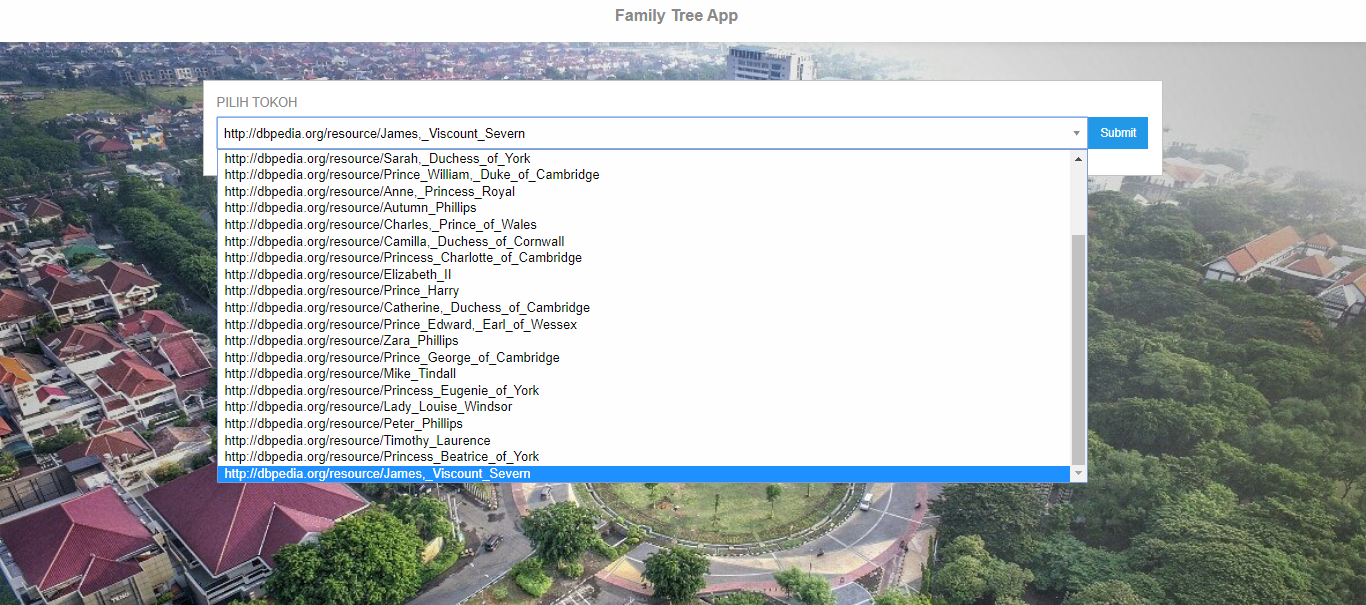


Gambar 4.4 Diagram Aktivitas Memilih Entitas Tautan

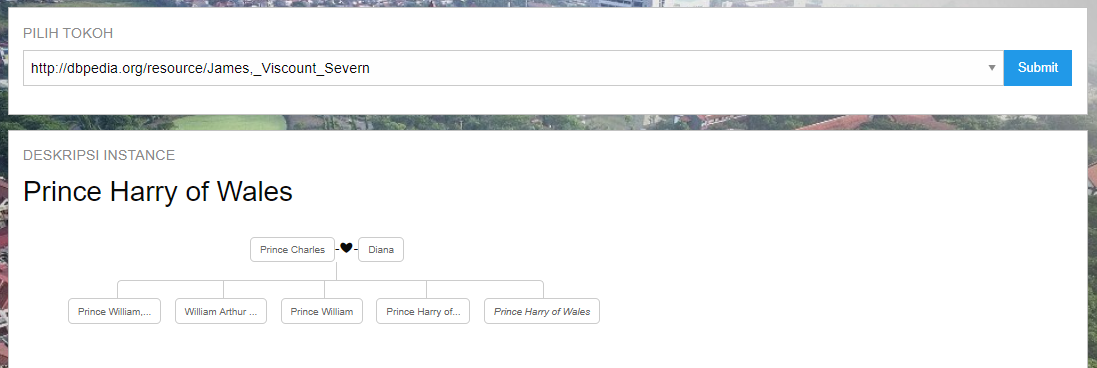
## Perancangan Antarmuka Pengguna

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 4.5. Sedangkan rancangan antarmuka halaman informasi data tokoh dapat dilihat pada Gambar 4.6. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini bisa dilihat pada Tabel 4.6.



Gambar 4.5 Antarmuka Halaman Utama Family Tree App



Gambar 4.6 Antarmuka Halaman Pohon Keluarga Family Tree App

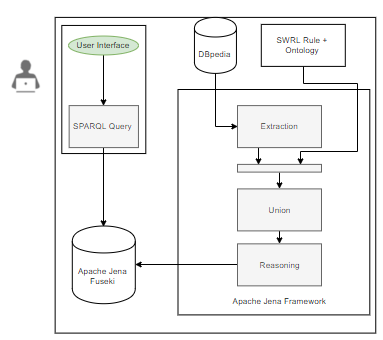
Tabel 4.6 Spesifikasi Atribut Rancangan Antarmuka Halaman Family Tree App

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

***[Halaman ini sengaja dikosongkan]***

# 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 selangkapnya pada bab ini. Implementasi perangkat lunak Family Tree App menggunakan bahasa pemrograman PHP dengan *library* SPARQL Lib.



Gambar 5.1 Arsitektur Perangkat Lunak

Arsitektur perangkat lunak yang dibangun pada tugas akhir ini dapat dilihat pada Gambar 5.1 di halaman sebelumnya. Agar dapat menampilkan fakta yang belum ada, pada ontologi ini diterapkan sejumlah *rule* yang telah dijelaskan pada Sub subbab. 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*.

## Implementasi Preprocessor

Pada bagian ini dijelaskan secara terperinci mengenai implementasi prose 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.

|  |
| --- |
| Import com.hp.hpl.jena.ontology.OntModel; import com.hp.hpl.jena.ontology.OntModelSpec; import com.hp.hpl.jena.rdf.model.\*; import com.hp.hpl.jena.reasoner.Reasoner; import com.hp.hpl.jena.util.FileManager; import org.apache.log4j.BasicConfigurator; import org.apache.log4j.LogManager; import org.apache.log4j.Logger; import org.apache.log4j.varia.NullAppender; import org.mindswap.pellet.jena.PelletReasonerFactory;  import java.io.\*;  */\*\*  \* Simple Ontology and DBpedia learning service preprocessor for  \* family tree web app.  \*  \** ***@author*** *FAIQ, karyoutomoo@Gmail.com  \*/* public class Main {   private static final Logger *logger* = LogManager.*getLogger*(Main.class);  public static void main(String[] args) throws FileNotFoundException {   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");   BasicConfigurator.*configure*(new NullAppender());  final OntModel ontModel = ModelFactory.*createOntologyModel*( OntModelSpec.*OWL\_DL\_MEM* );  System.*out*.println("Apache Jena Modelling, Reasoning and Inferring Tool");   // MEMODELKAN FILE ACTOR DARI OWL ONTOLOGI FAMILY dan JENA-FUSEKI  FileManager.*get*().addLocatorClassLoader(Main.class.getClassLoader());  Model Instances = FileManager.*get*().loadModel(READ\_FUSEKI);  Instances.read(READ\_FUSEKI,"RDF/XML");   Model famonto = FileManager.*get*().loadModel(OWL\_FILE\_LOCATION);   //ADD ACTOR  FileManager fManager = FileManager.*get*();  fManager.addLocatorURL();    String[] royalFamilies = {  //British Royal Family  "Elizabeth\_II",  "Prince\_Philip,\_Duke\_of\_Edinburgh",  "Anne,\_Princess\_Royal",  "Charles,\_Prince\_of\_Wales",  "Prince\_Edward,\_Earl\_of\_Wessex",  "Prince\_Andrew,\_Duke\_of\_York",  "Diana,\_Princess\_of\_Wales",  "Camilla,\_Duchess\_of\_Cornwall",  "Mark\_Phillips",  "Timothy\_Laurence",  "Sarah,\_Duchess\_of\_York",  "Sophie,\_Countess\_of\_Wessex",  "Prince\_William,\_Duke\_of\_Cambridge",  "Catherine,\_Duchess\_of\_Cambridge",  "Prince\_Harry",  "Meghan\_Markle",  "Peter\_Phillips",  "Autumn\_Phillips",  "Zara\_Phillips",  "Mike\_Tindall",  "Princess\_Beatrice\_of\_York",  "Princess\_Eugenie\_of\_York",  "Lady\_Louise\_Windsor",  "James,\_Viscount\_Severn",  "Prince\_George\_of\_Cambridge",  "Princess\_Charlotte\_of\_Cambridge",  "Savannah\_Phillips",  "Isla\_Phillips"   };    for (Integer counter = 0; counter < royalFamilies.length; counter++) {  Model modelActor = fManager.loadModel("http://dbpedia.org/data/" + royalFamilies[counter] + ".ttl");   Instances.add(modelActor);  System.*out*.println(royalFamilies[counter]);  }   // MERGING MODEL DARI JENA-FUSEKI DAN MODEL ONTOLOGI FAMILY  final Model union = ModelFactory.*createUnion*(Instances,famonto);   // REASONING MODEL UNION  Reasoner reasoner = PelletReasonerFactory.*theInstance*().create();  InfModel reasonedModel = ModelFactory.*createInfModel*(reasoner,union); // Model reasonedModel = union;   // KONVERSI KE FILE .RDF   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 5.1 Implementasi proses ekstraksi, penggabungan dan reasoning

## Implementasi Fungsi

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

### 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* allOfType digunakan untuk mendapatkan semua tipe ‘Person’ dari basis data *triple store*. Implementasi fungsi *dropdown select* dapat dilihat pada Kode Sumber 5.1.

|  |
| --- |
| $data = sparql\_get(**"localhost:3030/brits/query"**,  **"PREFIX fam: <http://www.co-ode.org/roberts/family-tree.owl#>  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>    SELECT DISTINCT ?s  WHERE {  ?s rdf:type foaf:Person.  ?s foaf:name ?name  }"**);  **if** (!**isset**($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**> |

Kode Sumber 5.2 Kode Sumber SPARQL untuk mengambil value bertipe Person dan Fungsi Dropdown Select

### 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 5.2.

|  |
| --- |
| $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 rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  SELECT ?name  WHERE {  <'** . $selected\_val . **'> foaf:name ?name  }  LIMIT 1'**); **if** (!**isset**($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 5.3 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 5.3.

|  |
| --- |
| $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 rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  SELECT ?name  WHERE {  <'** . $selected\_val . **'>fam:hasParent ?fatherIRI.  ?fatherIRI foaf:name ?name. ?fatherIRI foaf:gender "male"@en  }  LIMIT 1'**);  $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 rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  SELECT ?fatherIRI  WHERE {  <'** . $selected\_val . **'> fam:hasParent ?fatherIRI.  ?fatherIRI foaf:name ?name. ?fatherIRI foaf:gender "male"@en  }  LIMIT 1'**);  **foreach** ($data\_fatherIRI **as** $row) {  **foreach** ($data\_fatherIRI->fields() **as** $field) {  $fatherIRI = $row[$field];  }  }  **if** (!**isset**($data\_father) || $data\_father == **''**) {  **echo "<ul>"**;  **echo "<li>"**;  **echo "<a>Father Unknown</a>"**;  **echo "-❤-"**;  } **else** {  **foreach** ($data\_father **as** $row) {  **foreach** ($data\_father->fields() **as** $field) {  **echo "<ul>"**;  **echo "<li>"**;  **echo '<a href="?entity='**.*urlencode*($fatherIRI).**'">'**.*str\_replace*(**'http://www.dbpedia.org/resource/'**, **""**,$row[$field]).**'</a>'**;  **echo "-❤-"**;  }  }  } |

Kode Sumber 5.4 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 5.4.

|  |
| --- |
| $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/>  SELECT ?name  WHERE {  <'** . $selected\_val . **'> fam:hasParent ?motherIRI.  ?motherIRI foaf:name ?name. ?motherIRI foaf:gender "female"@en  }  LIMIT 1'**);  $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/>  SELECT ?motherIRI  WHERE {  <'** . $selected\_val . **'> fam:hasParent ?motherIRI.  ?motherIRI foaf:gender "female"@en  }'**);  **foreach** ($data\_motherIRI **as** $row) {  **foreach** ($data\_motherIRI->fields() **as** $field) {  $motherIRI = $row[$field];  }  }  **if** (!**isset**($data\_mother) || $data\_mother == **''**) {  **echo "<a>Mother Unknown</a>"**;  } **else** {  **foreach** ($data\_mother **as** $row) {  **foreach** ($data\_mother->fields() **as** $field) {  **echo '<a href="?entity='**.*urlencode*($motherIRI).**'">'**.*str\_replace*(**'http://www.dbpedia.org/resource/'**, **""**,$row[$field]).**'</a>'**;  }  }  } |

Kode Sumber 5.5 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 5.5.

|  |
| --- |
| $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/>  SELECT DISTINCT ?siblingIRI  WHERE {   <'** . $selected\_val . **'> fam:hasParent ?parentIRI.  ?parentIRI foaf:name ?name.  ?parentIRI fam:hasChild ?siblingIRI.  ?siblingIRI foaf:name ?siblingname  FILTER(?siblingIRI != <'** . $selected\_val . **'>)  }LIMIT 3'**);  $i=0; **foreach** ($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>"**;  $data\_sibling = sparql\_get(**"localhost:3030/brits/query"**, **'PREFIX fam: <http://www.co-ode.org/roberts/family-tree.owl#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  SELECT ?siblingname  WHERE {   <'** .$siblingIRI[$i] . **'> foaf:name ?siblingname  }LIMIT 1'**);   **foreach** ($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>'**;  }  }  $i++;  **echo "</li>"**;  }  } } |

Kode Sumber 5.6 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 5.6.

|  |
| --- |
| $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/>  SELECT DISTINCT ?spouseIRI  WHERE {  <'** . $selected\_val . **'> fam:isSpouseOf ?spouseIRI.  ?spouseIRI foaf:name ?name  } LIMIT 1'**); $i=0; **foreach** ($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) {  **echo "-❤-"**;  $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/>  SELECT ?name  WHERE {  <'** . $spouseIRI[$i] . **'> foaf:name ?name  } LIMIT 1  '**);  **foreach** ($data\_spouse **as** $row) {  **foreach** ($data\_spouse->fields() **as** $field) {  **echo '<a href="?entity='** . *urlencode*($spouseIRI[$i]) . **'">'** . *str\_replace*(**'http://www.dbpedia.org/resource/'**, **""**, $row[$field]) . **'</a>'**;  }  } |

Kode Sumber 5.7 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 5.7.

|  |
| --- |
| $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/>  SELECT DISTINCT ?childIRI  WHERE {   <'** . $selected\_val . **'> fam:hasChild ?childIRI.  <'** .$spouseIRI[$i] . **'> fam:hasChild ?childIRI.  ?childIRI foaf:name ?childName  }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>"**;  $data\_child = 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/>  SELECT ?childName  WHERE {  <'** . $childIRI[$cc] . **'> foaf:name ?childName  }LIMIT 1'**);  **foreach** ($data\_child **as** $rowChild) {  **foreach** ($data\_child->fields() **as** $field) {  **if** (*strlen*($rowChild[$field]) > 20)  $rowChild[$field] = *substr*($rowChild[$field], 0, 15) . **'...'**;  **echo '<a href="?entity='**.*urlencode*($childIRI[$cc]).**'">'**.*str\_replace*(**'http://www.dbpedia.org/resource/'**, **""**,$rowChild[$field]).**'</a>'**;  }  } |

Kode Sumber 5.8 Fungsi Get child

***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 5.8.

|  |
| --- |
| $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/> SELECT ?name WHERE {  <'** . $childIRI[$cc] . **'> fam:isSpouseOf ?sbj.  ?sbj foaf:name ?name }LIMIT 1'**); $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/> SELECT ?sbj WHERE {  <'** . $childIRI[$cc] . **'> fam:isSpouseOf ?sbj.  ?sbj foaf:name ?name }LIMIT 1'**); $cc++; } **foreach** ($data\_ChildInLawIRI **as** $rowChildInLawIRI) {  **foreach** ($data\_ChildInLawIRI->fields() **as** $field) {  $childInLawIRI = $rowChildInLawIRI[$field];  } } **if** (!**isset**($data\_ChildInLaw) || $data\_ChildInLaw == **''**) {  **echo "-❤-<a>?</a>"**; }**else if**(**isset**($data\_ChildInLaw)){  **foreach** ($data\_ChildInLaw **as** $rowChildInLaw) {  **foreach** ($data\_ChildInLaw->fields() **as** $field) {  **echo "-❤-"**;  **echo '<a href="?entity='** . *urlencode*($childInLawIRI) . **'">'** . *str\_replace*(**'http://www.dbpedia.org/resource/'**, **""**, $rowChildInLaw[$field]) . **'</a>'**; |

Kode Sumber 5.9 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 5.9.

|  |
| --- |
| $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 dbp: <http://dbpedia.org/property/> SELECT DISTINCT ?grandchildIRI WHERE {   <'** . $childInLawIRI . **'> dbp:issue ?grandchildIRI.  ?grandchildIRI foaf:name ?name }'**); $m=0; **foreach** ($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>"**;  $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/>  SELECT ?grandChildName  WHERE {  <'** . $grandchildIRI[$n] . **'> foaf:name ?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>'**;  }  } |

Kode Sumber 5.10 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 5.10.

|  |
| --- |
| $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/>  SELECT ?name  WHERE {  <'** . $grandchildIRI[$n] . **'> fam:isSpouseOf ?sbj.  ?sbj foaf:name ?name  }LIMIT 1'**); $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/>  SELECT ?sbj  WHERE {  <'** . $grandchildIRI[$n] . **'> fam:isSpouseOf ?sbj.  ?sbj foaf:name ?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\_GrandChildInLaw **as** $rowGrandChildInLaw) {  **foreach** ($data\_GrandChildInLaw->fields() **as** $field) {  **echo "-❤-"**;  **echo '<a href="?entity='** . *urlencode*($grandChildInLawIRI) . **'">'** . *str\_replace*(**'http://www.dbpedia.org/resource/'**, **""**, $rowGrandChildInLaw[$field]) . **'</a>'**; |

Kode Sumber 5.11 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 padaKode Sumber 5.11.

|  |
| --- |
| $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 dbp: <http://dbpedia.org/property/>  SELECT DISTINCT ?greatgrandchildIRI  WHERE {   <'** . $grandChildInLawIRI . **'> fam:hasChild ?greatgrandchildIRI.  <'** . $grandchildIRI[$n] . **'> fam:hasChild ?greatgrandchildIRI.  ?greatgrandchildIRI foaf:name ?name  }'**); $m=0; **foreach** ($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>"**;  $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/>  SELECT ?name  WHERE {   <'** . $greatGrandChildIRI[$p] . **'> foaf:name ?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>'**;  }  }  $p++;  **echo "</li>"**;  }  }  **echo "</ul>"**; |

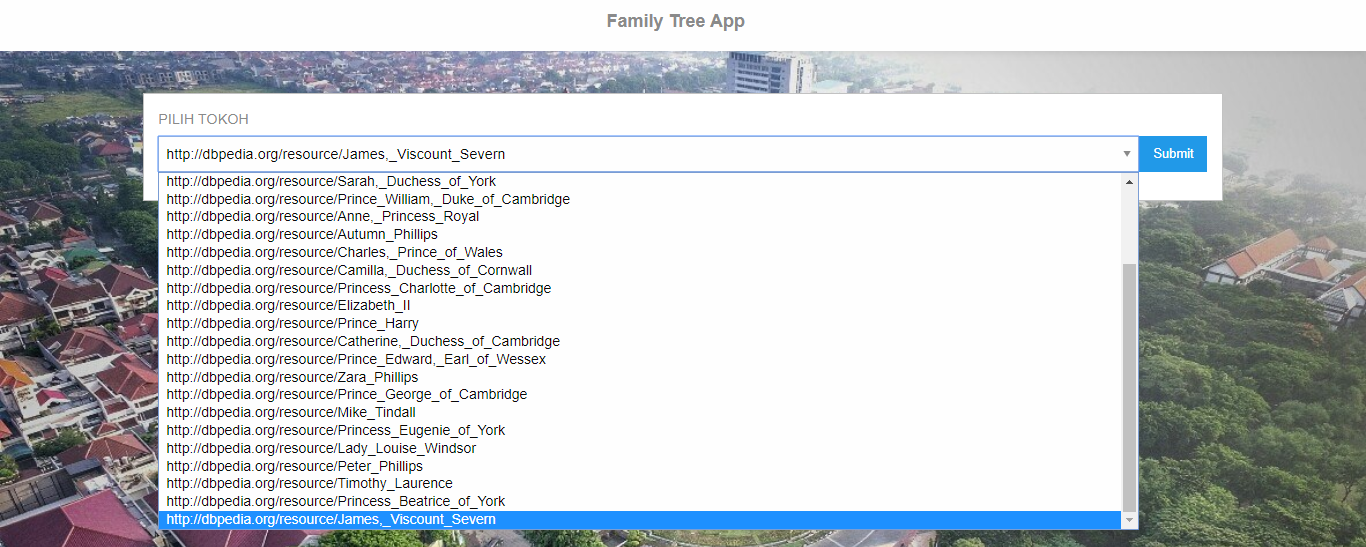
Kode Sumber 5.12 Get great grand child

## 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 padaFamily Tree App.

### Implementasi Tampilan Halaman Utama

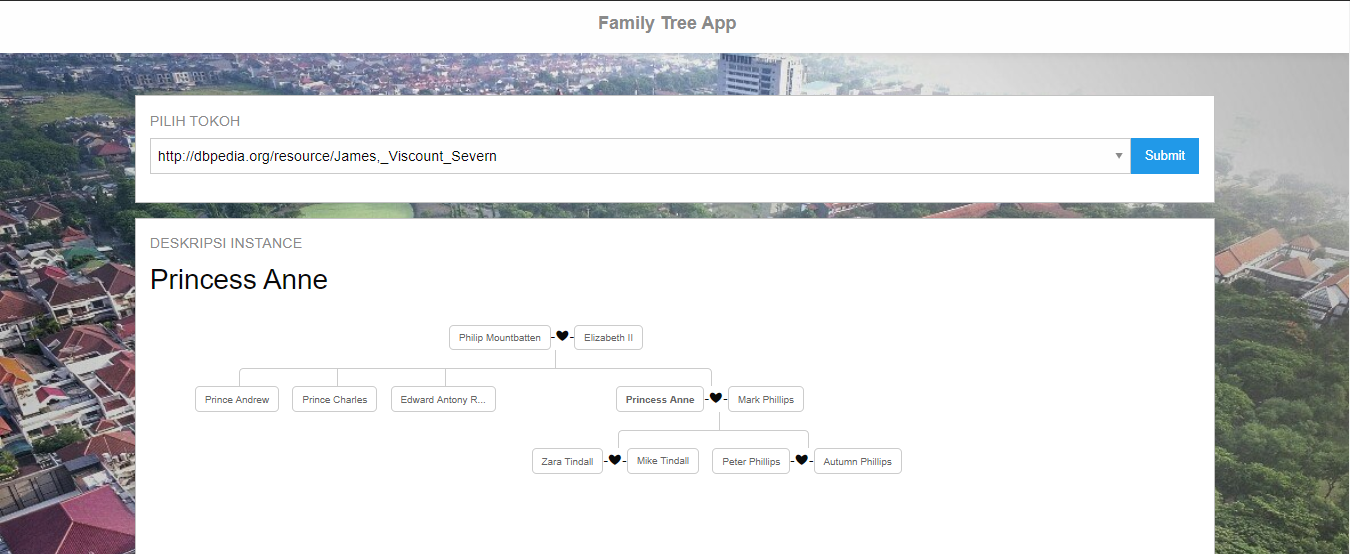
Halaman ini merupakan implementasi halaman utama dari rancangan antarmuka yang telah dijelaskan pada Subbab 4.2. Halaman utama hanya menampilkan kolom *dropdown select* yang dapat digunakan oleh pengguna untuk memilih tokoh. Daftar entitas tokoh yang ditampilkan hanya tokoh utama yang digunakan dalam pengerjaan tugas akhir ini. Tampilan antarmuka halaman utama ini dapat dilihat pada Gambar 5.2.



Gambar 5.2 Implementasi Antarmuka Halaman Utama

### 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 5.3.



Gambar 5.3 Implementasi Antarmuka Halaman Pohon Keluarga

# 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.

## 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 *Device* : Laptop

Sistem Operasi : Microsoft Windows 10 64-bit

*Protege* : Protege 5.2

*Reasoner* : Pellet

*Browser* : Google Chrome

## Skenario Pengujian

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

Pengujian data merupakan tahap uji kevalidan ontologi yang telah dibangun sebagai dasar proses pencarian relasi pohon keluarga tokoh kerajaan Inggris. Pengujian dilakukan secara manual dengan mengecek situs DBpedia.

### Pengujian Perbandingan Data

Pengujian perbandingan data merupakan tahap uji setelah fungsionalitas *rule* atau SPARQL *query* telah dilaksanakan. Pengujian ini bertujaun untuk membandingkan kekayaan data. Data yang dibandingkan merupakan data asli DBpedia dengan data yang didapatkan dari hasil proses *reasoning*. Pengujian dilakukan secara mandiri dengan membandingkan data tokoh pada halaman DBpedia dengan data tokoh pada aplikasi web yang disertai dengan fakta-fakta baru.

#### Pengujian Perbandingan Data Mark Phillips

Pada tahap pengujian tokoh Mark Phillips, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian Perbandingan Data Mark Phillips

|  |  |
| --- | --- |
| ID | TA-UJ.DT0001 |
| Nama | Pengujian perbandingan data Mark Phillips |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Mark Phillips pada halaman DBpedia. |
| Kondisi Awal | Individu Mark Phillips belum memiliki properti hasChild, hasChildInLaw, atau hasGrandChild |
| Data Uji | Data uji merupakan data yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Mark Phillips pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia Mark Phillips yang beralamat *dbpedia.org/page/Mark\_Phillips*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Mark Philips. |
| Hasil Yang Diharapkan | Data individu Mark Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Mark Phillips, diantaranya yaitu hasChild, hasChildInLaw dan hasGrandChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Mark Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Prince Andrew, Duke of York

Pada tahap pengujian tokoh Prince Andrew, fakta baru yang muncul dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.2.

Tabel 6.2 Pengujian Perbandingan Data Prince Andrew, Duke of York

|  |  |
| --- | --- |
| ID | TA-UJ.DT0002 |
| Nama | Pengujian perbandingan data Prince Andrew |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince Andrew pada halaman DBpedia. |
| Kondisi Awal | Individu Prince Andrew belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince Andrew pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia Prince Andrew yang beralamat *dbpedia.org/page/* *Prince\_Andrew,\_Duke\_of\_York*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince Andrew. |
| Hasil Yang Diharapkan | Data individu Prince Andrew pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince Andrew, yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince Andrew pada perangkat lunak Family Tree App lebih kaya dibandingkan data pada DBpedia. |

#### Pengujian Perbandingan Data Meghan Markle

Pada tahap pengujian tokoh Meghan Markle, fakta baru yang muncul dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.3.

Tabel 6.3 Pengujian Perbandingan Data Meghan Markle

|  |  |
| --- | --- |
| ID | TA-UJ.DT0003 |
| Nama | Pengujian perbandingan data Meghan Markle |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Meghan Markle pada halaman DBpedia. |
| Kondisi Awal | Individu Meghan Markle belum memiliki properti relasi keluarga. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Meghan Markle pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia Meghan Markle yang beralamat *dbpedia.org/page/Meghan\_Markle*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Meghan Markle. |
| Hasil Yang Diharapkan | Data individu Meghan Markle pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Meghan Markle, dikarenakan data DBpedia yang belum diperbarui. |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Meghan Markle pada perangkat lunak Family Tree App sama kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Diana, Princess of Wales

Pada tahap pengujian tokoh Princess Diana, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Hasyim Asyari merupakan tokoh yang difungsikan terkait peran sebagai tokoh agama, sehingga dihubungkan dengan properti *hasTeacher* yang berarti memiliki guru individu lain. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.4.

Tabel 6.4 Pengujian Perbandingan Data Diana, Princess of Wales

|  |  |
| --- | --- |
| ID | TA-UJ.DT0004 |
| Nama | Pengujian perbandingan data Princess Diana |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Princess Diana pada halaman DBpedia. |
| Kondisi Awal | Individu Princess Diana belum memiliki properti hasGrandChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Princess Diana pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Diana,\_Princess\_of\_Wales*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Princess Diana. |
| Hasil Yang Diharapkan | Data individu Princess Diana pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Princess Diana, diantaranya yaitu hasGrandChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Princess Diana pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Sophie, Countess of Wessex

Pada tahap pengujian tokoh Sophie, Countess of Wessex, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.5.

Tabel 6.5 Pengujian Perbandingan Data Sophie, Countess of Wessex

|  |  |
| --- | --- |
| ID | TA-UJ.DT0005 |
| Nama | Pengujian perbandingan data Sophie, Countess of Wessex |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Sophie, Countess of Wessex pada halaman DBpedia. |
| Kondisi Awal | Individu Sophie, Countess of Wessex memiliki *value* properti *spouse* dan *issue*. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Sophie, Countess of Wessex pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Sophie,\_Countess\_of\_Wessex*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Sophie, Countess of Wessex. |
| Hasil Yang Diharapkan | Data individu Sophie, Countess of Wessex pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan *value* atau *property* baru. |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Sophie, Countess of Wessex pada perangkat lunak Family Tree App sama kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Prince Philip, Duke of Edinburgh

Pada tahap pengujian tokoh Prince Philip, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.6.

Tabel 6.6 Pengujian Perbandingan Data Prince Philip, Duke of Edinburgh

|  |  |
| --- | --- |
| ID | TA-UJ.DT0006 |
| Nama | Pengujian perbandingan data Prince Philip |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince Philip pada halaman DBpedia. |
| Kondisi Awal | Individu Prince Philip belum memiliki properti hasChildInLaw, hasGrandChild, hasGrandChildInLaw dan hasGreatGrandChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince Philip pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Prince\_Philip,\_Duke\_of\_Edinburgh*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince Philip. |
| Hasil Yang Diharapkan | Data individu Prince Philip pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince Philip, diantaranya yaitu hasChildInLaw, hasGrandChild, hasGrandChildInLaw dan hasGreatGrandChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince Philip pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Sarah, Duchess of York

Pada tahap pengujian tokoh Sarah, Duchess of York, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.7.

Tabel 6.7 Pengujian Perbandingan Data Sarah, Duchess of York

|  |  |
| --- | --- |
| ID | TA-UJ.DT0007 |
| Nama | Pengujian perbandingan data Sarah, Duchess of York |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Mark Philips pada halaman DBpedia. |
| Kondisi Awal | Individu Sarah, Duchess of York memiliki *property* spouse dan issue. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Sarah, Duchess of York pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Sarah,\_Duchess\_of\_York.*  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Sarah, Duchess of York. |
| Hasil Yang Diharapkan | Data individu Sarah, Duchess of York pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Sarah, Duchess of York. |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Sarah, Duchess of York pada perangkat lunak Family Tree App sama dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Prince William, Duke of Cambridge

Pada tahap pengujian tokoh Prince William, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.8.

Tabel 6.8 Pengujian Perbandingan Data Prince William, Duke of Cambridge

|  |  |
| --- | --- |
| ID | TA-UJ.DT0008 |
| Nama | Pengujian perbandingan data Prince William. |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince William pada halaman DBpedia. |
| Kondisi Awal | Individu Prince William belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince William pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Prince\_William,\_Duke\_of\_Cambridge*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince William. |
| Hasil Yang Diharapkan | Data individu Prince William pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince William, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince William pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Anne, Princess Royal

Pada tahap pengujian tokoh Princess Anne, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.9.

Tabel 6.9 Pengujian Perbandingan Data Anne, Princess Royal

|  |  |
| --- | --- |
| ID | TA-UJ.DT0009 |
| Nama | Pengujian perbandingan data Princess Anne |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Princess Anne pada halaman DBpedia. |
| Kondisi Awal | Individu Princess Anne belum memiliki properti hasSibling dan hasGrandChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Princess Anne pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page*/*Anne,\_Princess\_Royal*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Princess Anne. |
| Hasil Yang Diharapkan | Data individu Princess Anne pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Princess Anne, diantaranya yaitu hasSibling dan hasGrandChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Princess Anne pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Autumn Phillips

Pada tahap pengujian tokoh Autumn Philips, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.10.

Tabel 6.10 Pengujian Perbandingan Data Autumn Phillips

|  |  |
| --- | --- |
| ID | TA-UJ.DT0010 |
| Nama | Pengujian perbandingan data Autumn Phillips |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Autumn Phillips pada halaman DBpedia. |
| Kondisi Awal | Individu Autumn Philips memiliki properti hasChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Autumn Phillips pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Autumn\_Phillips*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Autumn Philips. |
| Hasil Yang Diharapkan | Data individu Autumn Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Autumn Phillips, dan properti hasChild tidak ada dikarenakan data spouse dari Autumn Phillips tidak mencantumkan properti hasChild dimana syarat hasChild adalah ketika dua orang memiliki *value* properti hasChild yang sama. |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Autumn Phillips pada perangkat lunak Family Tree App tidak lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Charles, Prince of Wales

Pada tahap pengujian tokoh Prince Charles, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.11.

Tabel 6.11 Pengujian Perbandingan Data Charles, Prince of Wales

|  |  |
| --- | --- |
| ID | TA-UJ.DT0011 |
| Nama | Pengujian perbandingan data Prince Charles |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince Charles pada halaman DBpedia. |
| Kondisi Awal | Individu Prince Charles belum memiliki properti hasSibling dan hasGrandChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince Charles pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Charles,\_Prince\_of\_Wales*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince Charles. |
| Hasil Yang Diharapkan | Data individu Prince Charles pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince Charles, diantaranya yaitu hasSibling dan hasGrandChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince Charles pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Camilla, Duchess of Cornwall

Pada tahap pengujian tokoh Camilla Rosemary, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.12.

Tabel 6.12 Pengujian Perbandingan Data Camilla, Duchess of Cornwall

|  |  |
| --- | --- |
| ID | TA-UJ.DT0012 |
| Nama | Pengujian perbandingan data Camilla Rosemary |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Mark Philips pada halaman DBpedia. |
| Kondisi Awal | Individu Camilla Rosemary memiliki properti *spouse*, *parent* dan *issue*. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Camilla Rosemary pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Camilla,\_Duchess\_of\_Cornwall*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Camilla Rosemary. |
| Hasil Yang Diharapkan | Data individu Camilla Rosemary pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Camilla Rosemary. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Camilla Rosemary pada perangkat lunak Family Tree App tidak lebih kaya dibandingkan dengan data pada DBpedia dikarenakan *issue* dan *parent* dari Camilla bukanlah bagian dari *Royal Family*. |

#### Pengujian Perbandingan Data Princess Charlotte of Cambridge

Pada tahap pengujian tokoh Princess Charlotte, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Ahmad Dahlan merupakan tokoh yang difungsikan terkait peran sebagai tokoh agama, sehingga dihubungkan dengan properti *hasTeacher* yang berarti memiliki guru individu lain. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.13.

Tabel 6.13 Pengujian Perbandingan Data Princess Charlotte of Cambridge

|  |  |
| --- | --- |
| ID | TA-UJ.DT0013 |
| Nama | Pengujian perbandingan data Princess Charlotte |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Princess Charlotte pada halaman DBpedia. |
| Kondisi Awal | Individu Princess Charlotte belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Princess Charlotte pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Princess\_Charlotte\_of\_Cambridge*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Princess Charlotte. |
| Hasil Yang Diharapkan | Data individu Princess Charlotte pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Princess Charlotte, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Princess Charlotte pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Elizabeth II

Pada tahap pengujian tokoh Ratu Elizabeth, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.14.

Tabel 6.14 Pengujian Perbandingan Data Elizabeth II

|  |  |
| --- | --- |
| ID | TA-UJ.DT0014 |
| Nama | Pengujian perbandingan data Ratu Elizabeth |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Ratu Elizabeth pada halaman DBpedia. |
| Kondisi Awal | Individu Ratu Elizabeth belum memiliki properti hasChildInLaw, hasGrandChild, hasGrandChildInLaw dan hasGreatGrandChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Ratu Elizabeth pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Elizabeth\_II*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Ratu Elizabeth. |
| Hasil Yang Diharapkan | Data individu Ratu Elizabeth pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Ratu Elizabeth, diantaranya yaitu hasChildInLaw, hasGrandChild, hasGrandChildInLaw dan hasGreatGrandChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Ratu Elizabeth pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Prince Harry

Pada tahap pengujian tokoh Prince Harry, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.15.

Tabel 6.15 Pengujian Perbandingan Data Prince Harry

|  |  |
| --- | --- |
| ID | TA-UJ.DT0015 |
| Nama | Pengujian perbandingan data Prince Harry |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince Harry pada halaman DBpedia. |
| Kondisi Awal | Individu Prince Harry belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince Harry pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Prince\_Harry*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince Harry. |
| Hasil Yang Diharapkan | Data individu Prince Harry pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince Harry, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince Harry pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Catherine, Duchess of Cambridge

Pada tahap pengujian tokoh Catherine Middleton, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.16.

Tabel 6.16 Pengujian Perbandingan Data Catherine, Duchess of Cambridge

|  |  |
| --- | --- |
| ID | TA-UJ.DT0016 |
| Nama | Pengujian perbandingan data Catherine Middleton |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Catherine Middleton pada halaman DBpedia. |
| Kondisi Awal | Individu Catherine Middleton memiliki properti hasChild, hasSpouse dan hasParent. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Catherine Middleton pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Catherine,\_Duchess\_of\_Cambridge*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Catherine Middleton. |
| Hasil Yang Diharapkan | Data individu Catherine Middleton pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Catherine Middleton |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Catherine Middleton pada perangkat lunak Family Tree App tidak lebih kaya dibandingkan dengan data pada DBpedia dikarenakan properti hasParent milik Catherine Middleton bukanlah bagian dari Royal Family. |

#### Pengujian Perbandingan Data Prince Edward, Earl of Wessex

Pada tahap pengujian tokoh Prince Edward, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.17.

Tabel 6.17 Pengujian Perbandingan Data Prince Edward, Earl of Wessex

|  |  |
| --- | --- |
| ID | TA-UJ.DT0017 |
| Nama | Pengujian perbandingan data Prince Edward |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince Edward pada halaman DBpedia. |
| Kondisi Awal | Individu Prince Edward belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince Edward pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Prince\_Edward,\_Earl\_of\_Wessex*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince Edward. |
| Hasil Yang Diharapkan | Data individu Prince Edward pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince Edward, diantaranya yaitu hasSon dan hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince Edward pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Zara Phillips

Pada tahap pengujian tokoh Zara Phillips, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.18.

Tabel 6.18 Pengujian Perbandingan Data Zara Phillips

|  |  |
| --- | --- |
| ID | TA-UJ.DT0018 |
| Nama | Pengujian perbandingan data Zara Phillips |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Zara Phillips pada halaman DBpedia. |
| Kondisi Awal | Individu Zara Phillips belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Zara Phillips pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Zara\_Tindall*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Zara Phillips. |
| Hasil Yang Diharapkan | Data individu Zara Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Zara Phillips, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Zara Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Prince George of Cambridge

Pada tahap pengujian tokoh Prince George, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.19.

Tabel 6.19 Pengujian Perbandingan Data Prince George of Cambridge

|  |  |
| --- | --- |
| ID | TA-UJ.DT0019 |
| Nama | Pengujian perbandingan data Prince George |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Prince George pada halaman DBpedia. |
| Kondisi Awal | Individu Prince George belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Prince George pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Prince\_George\_of\_Cambridge*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Prince George. |
| Hasil Yang Diharapkan | Data individu Prince George pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Prince George, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Prince George pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Mike Tindall

Pada tahap pengujian tokoh Mike Tindall, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.20.

Tabel 6.20 Pengujian Perbandingan Data Mike Tindall

|  |  |
| --- | --- |
| ID | TA-UJ.DT0020 |
| Nama | Pengujian perbandingan data Mike Tindall |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Mark Philips pada halaman DBpedia. |
| Kondisi Awal | Individu Mike Tindall memiliki properti hasSpouse dan hasChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Mike Tindall pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Mike\_Tindall*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Mike Tindall. |
| Hasil Yang Diharapkan | Data individu Mike Tindall pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Mike Tindall. |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Mike Tindall pada perangkat lunak Family Tree App tidak lebih kaya dibandingkan dengan data pada DBpedia dikarenakan properti hasChild belum ada halaman DBpedianya. |

#### Pengujian Perbandingan Data Princess Eugenie of York

Pada tahap pengujian tokoh Princess Eugenie, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.21

Tabel 6.21 Pengujian Perbandingan Data Princess Eugenie

|  |  |
| --- | --- |
| ID | TA-UJ.DT0021 |
| Nama | Pengujian perbandingan data Princess Eugenie |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Princess Eugenie pada halaman DBpedia. |
| Kondisi Awal | Individu Princess Eugenie belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Princess Eugenie pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Princess\_Eugenie\_of\_York*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Princess Eugenie. |
| Hasil Yang Diharapkan | Data individu Princess Eugenie pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Princess Eugenie, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Princess Eugenie pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Lady Louise Windsor

Pada tahap pengujian tokoh Lady Louise, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada

Tabel 6.22 Pengujian Perbandingan Data Lady Louise

|  |  |
| --- | --- |
| ID | TA-UJ.DT0022 |
| Nama | Pengujian perbandingan data Lady Louise |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Lady Louise pada halaman DBpedia. |
| Kondisi Awal | Individu Lady Louise belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Lady Louise pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Lady\_Louise\_Windsor*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Lady Louise. |
| Hasil Yang Diharapkan | Data individu Lady Louise pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Lady Louise, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Lady Louise pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Peter Phillips

Pada tahap pengujian tokoh Peter Phillips, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada

Tabel 6.23 Pengujian Perbandingan Data Peter Phillips

|  |  |
| --- | --- |
| ID | TA-UJ.DT0023 |
| Nama | Pengujian perbandingan data Peter Phillips. |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Peter Phillips pada halaman DBpedia. |
| Kondisi Awal | Individu Peter Phillips belum memiliki properti hasSibling.  Individu Peter Phillips memiliki properti *dbp:children* tetapi dalam bentuk String. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Peter Phillips pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Peter\_Phillips*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Peter Phillips. |
| Hasil Yang Diharapkan | Data individu Peter Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Peter Phillips, yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Peter Phillips pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. Data hasChild tidak ada dikarenakan data pada Peter Phillips tidak direpresentasikan sebagai DBpedia *resource*. |

#### Pengujian Perbandingan Data Timothy Laurence

Pada tahap pengujian tokoh Timothy Laurence, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada

Tabel 6.24 Pengujian Perbandingan Data Timothy Laurence

|  |  |
| --- | --- |
| ID | TA-UJ.DT0024 |
| Nama | Pengujian perbandingan data Timothy Laurence |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Timothy Laurence pada halaman DBpedia. |
| Kondisi Awal | Individu Timothy Laurence hanya memiliki properti hasSpouse. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Timothy Laurence pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Timothy\_Laurence*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Timothy Laurence. |
| Hasil Yang Diharapkan | Data individu Timothy Laurence pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Tidak didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Timothy Laurence. |
| Hasil Pengujian | Gagal. |
| Kondisi Akhir | Data individu Timothy Laurence pada perangkat lunak Family Tree App tidak lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Princess Beatrice of York

Pada tahap pengujian tokoh Princess Beatrice, fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada

Tabel 6.25 Pengujian Perbandingan Data Princess Beatrice

|  |  |
| --- | --- |
| ID | TA-UJ.DT0025 |
| Nama | Pengujian perbandingan data Princess Beatrice |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data Princess Beatrice pada halaman DBpedia. |
| Kondisi Awal | Individu Princess Beatrice belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas Princess Beatrice pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Princess\_Beatrice\_of\_York*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia Princess Beatrice. |
| Hasil Yang Diharapkan | Data individu Princess Beatrice pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Princess Beatrice, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Princess Beatrice pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data James, Viscount Severn

Pada tahap pengujian tokoh James, Viscount Severn fakta baru yang muncul setelah diterapkannya karakteristik dan sejumlah *rule* dibandingkan dengan data aslinya. Tujuannya adalah untuk mengetahui apakah fakta baru yang didapatkan sudah ada pada DBpedia atau belum. Jika fakta baru belum ada pada DBpedia, maka fakta tersebut akan melengkapi data yang dimiliki oleh DBpedia. Rincian pengujian perbandingan data ini dapat dilihat pada

Tabel 6.26 Pengujian Perbandingan Data James, Viscount Severn

|  |  |
| --- | --- |
| ID | TA-UJ.DT0026 |
| Nama | Pengujian perbandingan data James, Viscount Severn |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari Family Tree App sudah tersedia di DBpedia. |
| Skenario 1 | Melihat data pohon keluarga yang ditampilkan pada Family Tree App dan mencocokkannya dengan data James, Viscount Severn pada halaman DBpedia. |
| Kondisi Awal | Individu James, Viscount Severn belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak Family Tree App. |
| Langkah Pengujian | Pengguna memiih entitas James, Viscount Severn pada perangkat lunak Family Tree App.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/James,\_Viscount\_Severn*.  Pengguna membandingkan data pada perangkat lunak Family Tree App dengan halaman DBpedia James, Viscount Severn. |
| Hasil Yang Diharapkan | Data individu James, Viscount Severn pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh James, Viscount Severn, diantaranya yaitu hasSibling. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu James, Viscount Severn pada perangkat lunak Family Tree App lebih kaya dibandingkan dengan data pada DBpedia. |

## Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian perbandingan data yang telah dijelaskan pada Subbab 6.2.2.

### Evaluasi Pengujian Perbandingan Data

Rangkuman mengenai hasil pengujian perbandingan data dapat dilihat pada Tabel 6.46. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil. Sehingga bisa ditarik disimpulkan bahwa ontologi yang dikembangkan telah sesuai dengan yang diharapkan.

Tabel 6.27 Rangkuman Hasil Pengujian

| ID | Nama | Skenario | Hasil |
| --- | --- | --- | --- |
| TA-UJ.DT0001 | Pengujian perbandingan data Mark Phillips | Skenario 1 | Berhasil |
| TA-UJ.DT0002 | Pengujian perbandingan data Prince Andrew, Duke of York | Skenario 1 | Berhasil |
| TA-UJ.DT0003 | Pengujian perbandingan data Meghan Markle | Skenario 1 | Gagal |
| TA-UJ.DT0004 | Pengujian perbandingan data Diana, Princess of Wales | Skenario 1 | Berhasil |
| TA-UJ.DT0005 | Pengujian perbandingan data Sophie, Countess of Wessex | Skenario 1 | Gagal |
| TA-UJ.DT0006 | Pengujian perbandingan data Prince Philip, Duke of Edinburgh | Skenario 1 | Berhasil |
| TA-UJ.DT0007 | Pengujian perbandingan data Sarah, Duchess of York | Skenario 1 | Gagal |
| TA-UJ.DT0008 | Pengujian perbandingan data Prince William | Skenario 1 | Berhasil |
| TA-UJ.DT0009 | Pengujian perbandingan data Princess Anne | Skenario 1 | Berhasil |
| TA-UJ.DT0010 | Pengujian perbandingan data Autumn Philips | Skenario 1 | Gagal |
| TA-UJ.DT0011 | Pengujian perbandingan data Prince Charles | Skenario 1 | Berhasil |
| TA-UJ.DT0012 | Pengujian perbandingan data Camilla Rosemary | Skenario 1 | Berhasil |
| TA-UJ.DT0013 | Pengujian perbandingan data Princess Charlotte | Skenario 1 | Berhasil |
| TA-UJ.DT0014 | Pengujian perbandingan data Ratu Elizabeth | Skenario 1 | Berhasil |
| TA-UJ.DT0015 | Pengujian perbandingan data Prince Harry | Skenario 1 | Berhasil |
| TA-UJ.DT0016 | Pengujian perbandingan data Catherine Middleton | Skenario 1 | Gagal |
| TA-UJ.DT0017 | Pengujian perbandingan data Prince Edward | Skenario 1 | Berhasil |
| TA-UJ.DT0018 | Pengujian perbandingan data Zara Phillips | Skenario 1 | Berhasil |
| TA-UJ.DT0019 | Pengujian perbandingan data Prince George | Skenario 1 | Berhasil |
| TA-UJ.DT0020 | Pengujian perbandingan data Mike Tindall | Skenario 1 | Gagal |
| TA-UJ.DT0021 | Pengujian perbandingan data Princess Eugenie | Skenario 1 | Berhasil |
| TA-UJ.DT0022 | Pengujian perbandingan data Lady Louise Windsor | Skenario 1 | Berhasil |
| TA-UJ.DT0023 | Pengujian perbandingan data Peter Phillips | Skenario 1 | Berhasil |
| TA-UJ.DT0024 | Pengujian perbandingan data Timothy Laurence | Skenario 1 | Gagal |
| TA-UJ.DT0025 | Pengujian perbandingan data Princess Beatrice | Skenario 1 | Berhasil |
| TA-UJ.DT0026 | Pengujian perbandingan data James, Viscount Severn | Skenario 1 | 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.



## Kesimpulan

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

1. Studi kasus visualisasi pohon keluarga tokoh kerajaan Inggris mampu dimodelkan dan digabungkan dengan model ontologi lokal dengan Apache Jena.
2. Properti yang dimiliki oleh *Family Relationship Ontology* dapat digunakan pada domain tokoh kerajaan Inggris.
3. *Rule* dikembangkan menggunakan SWRL untuk mencari relasi hubungan keluarga*.* Fungsionalitas *rule* 100% dapat diterapkan pada <jumlah> data uji coba dengan memperoleh hasil data *inference* yang tepat. Selain itu, data *inference* yang dihasilkan dari penerapan *rule* juga 100% dapat memperkaya informasi DBpedia pada <jumlah> data uji yang digunakan.
4. Library SPARQL Lib mampu menghubungkan basis data Apache Jena Fuseki dengan perangkat lunak berbasis *web* menggunakan bahasa pemrograman PHP.

## 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 spesfikasi kapasitas memori yang lebih besar agar waktu yang dibutuhkan untuk proses *export inferenced axiom* lebih cepat.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | M. A. Ramadhanie, Penerapan Ontologi Objek Pembelajaran Untuk Kebutuhan Personalisasi E-Learning Berbasis Semantic Web, Depok: Universitas Indonesia, 2009. |
| [2] | S. Nikles, "Expressiveness of Enterprise Modelling Languages," University of Applied Sciences Northwestern Switzerland, Basel, 2010. |
| [3] | 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. |
| [4] | "Professor Robert Stevens," [Online]. Available: http://www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl. [Accessed 06 January 2016]. |
| [5] | 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. |
| [6] | B. Parsia and E. Sirin, "Pellet: An OWL DL Reasoner," University of Maryland, College Park. |
| [7] | D. Wu and A. Håkansson, "A Method of Identifying Ontology Domain," *Procedia Computer Science,* vol. 35, pp. 504-513, 2014. |
| [8] | C. H. d. Nascimento, F. S. Ferraz, R. E. Assad, D. L. e. Silva and V. H. d. Rocha, "OntoLog: Using Web Semantic and Ontology for Security Log Analysis," in *ICSEA 2011 : The Sixth International Conference on Software Engineering Advances*, Brazil, 2011. |
| [9] | N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 ,* 2001. |
| [10] | K. Hanke, J. H. Liu, C. G. Sibley, D. Paez, S. O. Gaines Jr., G. Moloney, C.-H. Leong, W. Wagner, L. Licata, O. Klein, I. Garber, G. Böhm, D. J. Hilton, V. Valchev, S. S. Khan and R. Cabecinhas, "“Heroes” and “Villains” of World History across Cultures," *PLOS ONE,* vol. 10, no. 2, 2015. |
| [11] | M. Horridge, A Practical Guide To Building OWL Ontologies Using Prot´eg´e 4 and CO-ODE Tools Edition 1.2, The University Of Manchester, 2009. |
| [12] | T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human-Computer Studies,* vol. 43, no. 5-6, p. 907–928, 1995. |
| [13] | S. J. Miller, Introduction to Ontology Concepts and Terminology, Lisbon, Portugal: University of Wisconsin-Milwaukee, 2013. |
| [14] | M. Cristani and R. Cuel, "A Survey on Ontology Creation Methodologies," *Int’l Journal on Semantic Web & Information Systems,* vol. 1, no. 2, pp. 48-68, April-June 2005. |
| [15] | 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]. |
| [16] | I. N. Bagus Caturbawa, "Case Tool untuk Pemodelan Data Semantik dalam Web Ontology Language (OWL)," Institut Teknologi Sepuluh Nopember, Surabaya, 2009. |
| [17] | J. Heflin, "OWL Web Ontology Language Use Cases and Requirements," [Online]. Available: https://www.w3.org/TR/webont-req/. [Accessed 06 January 2016]. |
| [18] | D. Brickley and L. Miller, "FOAF Vocabulary Specification 0.98," Creative Commons, 2010. |
| [19] | M. M. A. Al-Mukhtar and A. T. A. Al-Assafy, "The Implementation of FOAF Ontology for an Academic Social Network," *International Journal of Computer Science Engineering and Technology( IJCSET),* vol. 4, no. 1, pp. 10-14, January 2014. |
| [20] | R. R. Larson, "Bringing Lives to Light: Biography in Context," University of California, Berkeley, 2010. |
| [21] | C. H. Cordovi-Garcia, C. H. Rizo, Y. H. Delgado and L. R. Alvarez, Using Search Paradigms and Architecture Information Components to Consume Linked Data journal, Havana, Cuba: University of Informatics Sciences. |
| [22] | "EasyRdf API Documentation," [Online]. Available: http://www.easyrdf.org/docs/api/EasyRdf\_Graph.html. [Accessed 30 May 2016]. |
| [23] | I. Davis and D. Galbraith, "BIO: A vocabulary for biographical information," [Online]. Available: http://vocab.org/bio/. [Accessed 06 January 2016]. |
| [24] | Z. T. Inc., "An overview on PHP," Zend The PHP Company, 2007. |
| [25] | M. Achour, F. Betz, A. Dovgal, N. Lopes, H. Magnusson, G. Richter, D. Seguy and J. Vrana, "PHP Manual," The PHP Documentation Group, [Online]. Available: http://php.net/manual/en/. [Accessed 10 June 2016]. |
| [26] | A. Mishra, "Critical Comparison Of PHP And ASP.NET For Web Development," *International Journal of Scientific & Technology Research,* vol. 3, no. 7, pp. 331-333, July 2014. |
| [27] | T. Point, "PHP Hypertext Preprocessor," 2015. [Online]. Available: www.tutorialspoint.com/php/php\_tutorial.pdf. [Accessed 10 June 2016]. |
| [28] | "XML," [Online]. Available: https://en.wikipedia.org/wiki/XML. [Accessed 10 June 2016]. |
| [29] | E. Simperl, S. C, R. Ungrangsi and T. Bürger, "Ontology metadata for ontology reuse," *International Journal Metadata, Semantics and Ontologies,* vol. 6, no. 2, 2011. |
| [30] | M. R. Ramos, "Biography Light Ontology: An Open Vocabulary For Encoding Biographic Texts," University of California, Berkeley, 2009. |

# LAMPIRAN A. DATA SEBELUM *REASONING*

**Tabel A.1 Sampel Data Sebelum *Reasoning* Individu 006\_Cut\_Nyak\_Dhien**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006\_Cut\_Nyak\_Dhien | | |
| Types | Hero  Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasSubject | S\_National\_Heroes\_of\_Indonesia | Subject |
| hasSubject | S\_1850\_births | Subject |
| hasSubject | S\_Indonesian\_Muslims | Subject |
| hasSubject | S\_Women\_in\_war\_1900-1945 | Subject |
| hasSubject | S\_Indonesian\_revolutionaries | Subject |
| hasSubject | S\_Women\_in\_war\_in\_Indonesia | Subject |
| hasSubject | S\_Islam\_and\_women | Subject |
| hasSubject | S\_1908\_deaths | Subject |
| hasSubject | S\_Acehnese\_people | Subject |
| hasSubject | S\_Women\_in\_19th\_century\_warfare | Subject |
| isCommanderOf | E\_Aceh\_War | Event |
| knownFor | G\_National\_Heroine | Group |
| Data property assertions | hasName | Cut Nyak Dhien | String |
| alias | Cut Nyak Dien | String |
| hasReligion | Islam | String |
| alsoKnownAs | National Heroine | String |
| alias | Tjoet Nja Dhien | String |

**Tabel A.2 Sampel Data Sebelum *Reasoning* Individu 006Birth**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Birth | | |
| Types | Birth | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Principal | 006\_Cut\_Nyak\_Dhien | Person |
| Data property assertions | hasBirthDate | 1850-01-01T00:00:00Z | dateTime |
| hasBirthPlace | Aceh Besar Regency | String |
| hasBirthPlace | Indonesia | String |
| hasBirthPlace | Aceh\_Sultanate | String |
| hasBirthYear | 1850 | Integer |
| hasBirthPlace | Aceh | String |

**Tabel A.3 Sampel Data Sebelum *Reasoning* Individu 006Child\_Cut\_Gambang**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Child\_Cut\_Gambang | | |
| Types | Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Cut Gambang | String |

**Tabel A.4 Sampel Data Sebelum *Reasoning* Individu 006Death**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Death | | |
| Types | Death | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Principal | 006\_Cut\_Nyak\_Dhien | Person |
| Data property assertions | hasDeathPlace | West Java | String |
| hasDeathPlace | Sumedang | String |
| hasBirthDate | 1908-11-06T00:00:00Z | dateTime |
| hasDeathYear | 1908 | Integer |
| hasDeathPlace | Dutch East Indies | String |
| hasDeathPlace | Indonesia | String |

**Tabel A.5 Sampel Data Sebelum *Reasoning* Individu 006Marriage1**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Marriage1 | | |
| Types | Marriage | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Partner | 007\_Teuku\_Umar | Person |
| Partner | 006\_Cut\_Nyak\_Dhien | Person |
| Data property assertions |  |  |  |

**Tabel A.6 Sampel Data Sebelum *Reasoning* Individu 006Marriage1**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Marriage2 | | |
| Types | Marriage | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Partner | 006\_Cut\_Nyak\_Dhien | Person |
| Partner | 006Spouse\_Ibrahim\_Lamnga | Person |
| Data property assertions |  |  |  |

**Tabel A.7 Sampel Data Sebelum *Reasoning* Individu 006Spouse\_Ibrahim\_Lamnga**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Spouse\_Ibrahim\_Lamnga | | |
| Types | Male  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions |  |  |  |
| Data property assertions | hasName | Ibrahim Lamnga | String |

**Tabel A.8 Sampel Data Sebelum *Reasoning* Individu 007\_Teuku\_Umar**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007\_Teuku\_Umar | | |
| Types | Hero  Male  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasSubject | S\_Acehnese\_people | Subject |
| hasSubject | S\_History\_of\_Aceh | Subject |
| hasSubject | S\_National\_Heroes\_of\_Indoneisa | Subject |
| hasSubject | S\_1899\_deaths | Subject |
| hasSubject | S\_1854\_births | Subject |
| isCommanderOf | E\_Aceh\_War | Event |
| Data property assertions | hasFamilyName | Umar | String |
| hasOccupation | Uleebalang | String |
| hasFirstGivenName | Teuku | String |
| hasName | Sarah, Duchess of York | String |
| hasReligion | Islam | String |

**Tabel A.9 Sampel Data Sebelum *Reasoning* Individu 007Birth**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Birth | | |
| Types | Birth | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Principal | 007\_Teuku\_Umar | Person |
| Data property assertions | hasBirthDate | 1854-01-01T00:00:00Z | dateTime |
| hasBirthPlace | Meulaboh | String |
| hasBirthPlace | Aceh Sultanate | String |
| hasBirthPlace | Aceh\_Sultanate | String |
| hasBirthYear | 1854 | Integer |

**Tabel A.10 Sampel Data Sebelum *Reasoning* Individu 007Child\_Cut\_Mariyam**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Cut\_Mariyam | | |
| Types | Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Cut Mariyam | String |

**Tabel A.11 Sampel Data Sebelum *Reasoning* Individu 007Child\_Cut\_Sjak**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Cut\_Sjak | | |
| Types | Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Cut Sjak | String |

**Tabel A.12 Sampel Data Sebelum *Reasoning* Individu 007Child\_Cut\_Teungoh**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Cut\_Teungoh | | |
| Types | Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Cut Teungoh | String |

**Tabel A.13 Sampel Data Sebelum *Reasoning* Individu 007Child\_Teuku\_Bidin**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Teuku\_Bidin | | |
| Types | Male  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Teuku Bidin | String |

**Tabel A.14 Sampel Data Sebelum *Reasoning* Individu 007Child\_Teuku\_Raja\_Sulaiman**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Teuku\_Raja\_Sulaiman | | |
| Types | Male  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Teuku Raja Sulaiman | String |

**Tabel A.15 Sampel Data Sebelum *Reasoning* Individu 007Child\_Teuku\_Sapeh**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Teuku\_Sapeh | | |
| Types | Male  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| Data property assertions | hasName | Teuku Sapeh | String |

**Tabel A.16 Sampel Data Sebelum *Reasoning* Individu 007Death**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Death | | |
| Types | Death | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Principal | 007\_Teuku\_Umar | Person |
| Data property assertions | hasDeathPlace | Meulaboh | String |
| hasDeathPlace | Aceh Sultanate | String |
| hasBirthDate | 1899-02-11T00:00:00Z | dateTime |
| hasDeathYear | 1899 | integer |

**Tabel A.17 Sampel Data Sebelum *Reasoning* Individu 007Marriage1**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Marriage1 | | |
| Types | Marriage | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Partner | 007Spouse\_Cut\_Meuligou | Person |
| Partner | 007\_Teuku\_Umar | Person |
| Data property assertions |  |  |  |

**Tabel A.18 Sampel Data Sebelum *Reasoning* Individu 007Marriage2**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Marriage2 | | |
| Types | Marriage | | |
|  | Property Name | Value | Property Types |
| Object property assertions | Partner | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| Partner | 007\_Teuku\_Umar | Person |
| Data property assertions |  |  |  |

**Tabel A.19 Sampel Data Sebelum *Reasoning* Individu 007Spouse\_Cut\_Meuligou**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Spouse\_Cut\_Meuligou | | |
| Types | Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions |  |  |  |
| Data property assertions | hasName | Cut Meuligou | String |
| alias | Nyak Malighai | String |

**Tabel A.20 Sampel Data Sebelum *Reasoning* Individu 007Spouse\_Cut\_Nyak\_Sofiah**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Spouse\_Cut\_Nyak\_Sofiah | | |
| Types | Female  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions |  |  |  |
| Data property assertions | hasName | Cut Nyak Sofiah | String |

***[Halaman ini sengaja dikosongkan]***

# LAMPIRAN B. DATA SETELAH *REASONING*

**Tabel B.1 Sampel Data Setelah *Reasoning* Individu 007\_Cut\_Nyak\_Dhien**

| Individuals | 006\_Cut\_Nyak\_Dhien | | |
| --- | --- | --- | --- |
| Types | Hero  Female  Person  Agent  Woman | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasSubject | S\_National\_Heroes\_of\_Indonesia | Subject |
| hasSubject | S\_1850\_births | Subject |
| hasSubject | S\_Indonesian\_Muslims | Subject |
| hasSubject | S\_Women\_in\_war\_1900-1945 | Subject |
| hasSubject | S\_Indonesian\_revolutionaries | Subject |
| hasSubject | S\_Women\_in\_war\_in\_Indonesia | Subject |
| hasSubject | S\_Islam\_and\_women | Subject |
| hasSubject | S\_1908\_deaths | Subject |
| hasSubject | S\_Acehnese\_people | Subject |
| hasSubject | S\_Women\_in\_19th\_century\_warfare | Subject |
| isCommanderOf | E\_Aceh\_War | Event |
| knownFor | G\_National\_Heroine | Group |
| hasChild | 006Child\_Cut\_Gambang | Person |
| hasStepChild | 007Child\_Cut\_Teungoh | Person |
| hasStepChild | 007Child\_Teuku\_Bidin | Person |
| hasStepChild | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasStepChild | 007Child\_Cut\_Sjak | Person |
| hasStepChild | 007Child\_Cut\_Mariyam | Person |
| hasStepChild | 007Child\_Teuku\_Sapeh | Person |
| hasHusband | 006Spouse\_Ibrahim\_Lamnga | Person |
| hasHusband | 007\_Teuku\_Umar | Person |
| hasStepSon | 007Child\_Teuku\_Bidin | Person |
| hasStepSon | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasStepSon | 007Child\_Teuku\_Sapeh | Person |
| hasStepDaughter | 007Child\_Cut\_Teungoh | Person |
| hasStepDaughter | 007Child\_Cut\_Sjak | Person |
| hasStepDaughter | 007Child\_Cut\_Mariyam | Person |
| isSpouseOf | 006Spouse\_Ibrahim\_Lamnga | Person |
| isSpouseOf | 007\_Teuku\_Umar | Person |
| isParentOf | 006Child\_Cut\_Gambang | Person |
| isInLawOf | 006Spouse\_Ibrahim\_Lamnga | Person |
| isInLawOf | 007\_Teuku\_Umar | Person |
| isWifeOf | 006Spouse\_Ibrahim\_Lamnga | Person |
| isWifeOf | 007\_Teuku\_Umar | Person |
| differentFrom | 006Marriage1 | Marriage |
| differentFrom | 006Marriage2 | Marriage |
| differentFrom | 006Death | Death |
| differentFrom | 006Birth | Birth |
| ‘Life Event’ | 006Marriage1 | Marriage |
| ‘Life Event’ | 006Marriage2 | Marriage |
| ‘Life Event’ | 006Death | Death |
| ‘Life Event’ | 006Birth | Birth |
| Data property assertions | hasName | Cut Nyak Dhien | String |
| alias | Cut Nyak Dien | String |
| hasReligion | Islam | String |
| alsoKnownAs | National Heroine | String |
| alias | Tjoet Nja Dhien | String |

**Tabel B.2 Sampel Data Setelah *Reasoning* Individu 006Child\_Cut\_Gambang**

| Individuals | 006Child\_Cut\_Gambang | | |
| --- | --- | --- | --- |
| Types | Female  Person  Women | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 007Spouse\_Cut\_Meuligou | Person |
| hasStepBrother | 007Child\_Teuku\_Bidin | Person |
| hasStepBrother | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasStepBrother | 007Child\_Teuku\_Sapeh | Person |
| hasStepParent | 006Spouse\_Ibrahim\_Lamnga | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 007Spouse\_Cut\_Meuligou | Person |
| hasStepSister | 007Child\_Cut\_Teungoh | Person |
| hasStepSister | 007Child\_Cut\_Sjak | Person |
| hasStepSister | 007Child\_Cut\_Mariyam\_ | Person |
| hasStepFather | 006Spouse\_Ibrahim\_Lamnga | Person |
| hasStepSibling | 007Child\_Cut\_Teungoh | Person |
| hasStepSibling | 007Child\_Teuku\_Bidin | Person |
| hasStepSibling | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasStepSibling | 007Child\_Cut\_Sjak | Person |
| hasStepSibling | 007Child\_Cut\_Mariyam\_ | Person |
| hasStepSibling | 007Child\_Teuku\_Sapeh | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 006\_Cut\_Nyak\_Dhien | Person |
| Data property assertions | hasName | Cut Gambang | String |

**Tabel B.3 Sampel Data Setelah *Reasoning* Individu 006Spouse\_Ibrahim\_Lamnga**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 006Spouse\_Ibrahim\_Lamnga | | |
| Types | Male  Person  Agent  Man | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasWife | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepChild | 006Child\_Cut\_Gambang | Person |
| hasStepDaughter | 006Child\_Cut\_Gambang | Person |
| isSpouseOf | 006\_Cut\_Nyak\_Dhien | Person |
| isHusbandOf | 006\_Cut\_Nyak\_Dhien | Person |
| isInLawOf | 006\_Cut\_Nyak\_Dhien | Person |
| differentFrom | 006Marriage2 | Marriage |
| ‘Life Efent’ | 006Marriage2 | Marriage |
| Data property assertions | hasName | Ibrahim Lamnga | String |

**Tabel B.4 Sampel Data Setelah *Reasoning* Individu 007\_Teuku\_Umar**

| Individuals | 007\_Teuku\_Umar | | |
| --- | --- | --- | --- |
| Types | Hero  Male  Person  Agent  Man | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasSubject | S\_Acehnese\_people | Subject |
| hasSubject | S\_History\_of\_Aceh | Subject |
| hasSubject | S\_National\_Heroes\_of\_Indoneisa | Subject |
| hasSubject | S\_1899\_deaths | Subject |
| hasSubject | S\_1854\_births | Subject |
| isCommanderOf | E\_Aceh\_War | Event |
| hasWife | 007\_Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasWife | 006\_Cut\_Nyak\_Dhien | Person |
| hasWife | 007Spouse\_Cut\_Meuligou | Person |
| hasChild | 007Child\_Cut\_Teungoh | Person |
| hasChild | 007Child\_Teuku\_Bidin | Person |
| hasChild | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasChild | 006Child\_Cut\_Gambang | Person |
| hasChild | 007Child\_Cut\_Sjak | Person |
| hasChild | 007Child\_Cut\_Mariyam | Person |
| hasChild | 007Child\_Teuku\_Sapeh | Person |
| hasDaughter | 007Child\_Cut\_Teungoh | Person |
| hasDaughter | 006Child\_Cut\_Gambang | Person |
| hasDaughter | 007Child\_Cut\_Sjak | Person |
| hasDaughter | 007Child\_Cut\_Mariyam | Person |
| hasSon | 007Child\_Teuku\_Bidin | Person |
| hasSon | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSon | 007Child\_Teuku\_Sapeh | Person |
| isSpouseOf | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| isSpouseOf | 006\_Cut\_Nyak\_Dhien | Person |
| isSpouseOf | 007Spouse\_Cut\_Meuligou | Person |
| isSpouseOf | 007Child\_Cut\_Teungoh | Person |
| isSpouseOf | 007Child\_Teuku\_Bidin | Person |
| isSpouseOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isSpouseOf | 006Child\_Cut\_Gambang | Person |
| isSpouseOf | 007Child\_Cut\_Sjak | Person |
| isSpouseOf | 007Child\_Cut\_Mariyam | Person |
| isSpouseOf | 007Child\_Teuku\_Sapeh | Person |
| isHusbandOf | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| isHusbandOf | 006\_Cut\_Nyak\_Dhien | Person |
| isHusbandOf | 007Spouse\_Cut\_Meuligou | Person |
| isInLawOf | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| isInLawOf | 006\_Cut\_Nyak\_Dhien | Person |
| isInLawOf | 007Spouse\_Cut\_Meuligou | Person |
| differentFrom | 007Marriage2 | Marriage |
| differentFrom | 006Marriage1 | Marriage |
| differentFrom | 007Death | Death |
| differentFrom | 007Marriage1 | Marriage |
| differentFrom | 007Birth | Birth |
| ‘Life Event’ | 007Marriage2 | Marriage |
| ‘Life Event’ | 006Marriage1 | Marriage |
| ‘Life Event’ | 007Death | Death |
| ‘Life Event’ | 007Marriage1 | Marriage |
| ‘Life Event’ | 007Birth | Birth |
| Data property assertions | hasFamilyName | Umar | String |
| hasOccupation | Uleebalang | String |
| hasFirstGivenName | Teuku | String |
| hasName | Sarah, Duchess of York | String |
| hasReligion | Islam | String |

**Tabel B.5 Sampel Data Setelah *Reasoning* Individu 007Child\_Cut\_Mariyam**

| Individuals | 007Child\_Cut\_Mariyam | | |
| --- | --- | --- | --- |
| Types | Female  Person  Woman | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 007Spouse\_Cut\_Meuligou | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasSister | 007Child\_Cut\_Teungoh | Person |
| hasSister | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Cut\_Teungoh | Person |
| hasSibling | 007Child\_Teuku\_Bidin | Person |
| hasSibling | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSibling | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Teuku\_Sapeh | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepSister | 006Child\_Cut\_Gambang | Person |
| hasStepSibling | 006Child\_Cut\_Gambang | Person |
| hasBrother | 007Child\_Teuku\_Bidin | Person |
| hasBrother | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasBrother | 007Child\_Teuku\_Sapeh | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 007Spouse\_Cut\_Meuligou | Person |
| isSisterOf | 007Child\_Cut\_Teungoh | Person |
| isSisterOf | 007Child\_Teuku\_Bidin | Person |
| isSisterOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isSisterOf | 007Child\_Cut\_Sjak | Person |
| isSisterOf | 007Child\_Teuku\_Sapeh | Person |
| Data property assertions | hasName | Cut Mariyam | String |

**Tabel B.6 Sampel Data Setelah *Reasoning* Individu 007Child\_Cut\_Sjak**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Cut\_Sjak | | |
| Types | Female  Person  Woman | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 007Spouse\_Cut\_Meuligou | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasSister | 007Child\_Cut\_Teungoh | Person |
| hasSister | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Cut\_Teungoh | Person |
| hasSibling | 007Child\_Teuku\_Bidin | Person |
| hasSibling | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSibling | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Teuku\_Sapeh | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepSister | 006Child\_Cut\_Gambang | Person |
| hasStepSibling | 006Child\_Cut\_Gambang | Person |
| hasBrother | 007Child\_Teuku\_Bidin | Person |
| hasBrother | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasBrother | 007Child\_Teuku\_Sapeh | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 007Spouse\_Cut\_Meuligou | Person |
| isSisterOf | 007Child\_Cut\_Teungoh | Person |
| isSisterOf | 007Child\_Teuku\_Bidin | Person |
| isSisterOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isSisterOf | 007Child\_Cut\_Mariyam | Person |
| isSisterOf | 007Child\_Teuku\_Sapeh | Person |
| Data property assertions | hasName | Cut Sjak | String |

**Tabel B.7 Sampel Data Setelah *Reasoning* Individu 007Child\_Cut\_Teungoh**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Cut\_Teungoh | | |
| Types | Female  Person  Woman | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 007Spouse\_Cut\_Meuligou | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasSister | 007Child\_Cut\_Sjak | Person |
| hasSister | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Teuku\_Bidin | Person |
| hasSibling | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSibling | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Teuku\_Sapeh | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepSister | 006Child\_Cut\_Gambang | Person |
| hasStepSibling | 006Child\_Cut\_Gambang | Person |
| hasBrother | 007Child\_Teuku\_Bidin | Person |
| hasBrother | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasBrother | 007Child\_Teuku\_Sapeh | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 007Spouse\_Cut\_Meuligou | Person |
| isSisterOf | 007Child\_Cut\_Sjak | Person |
| isSisterOf | 007Child\_Teuku\_Bidin | Person |
| isSisterOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isSisterOf | 007Child\_Cut\_Mariyam | Person |
| isSisterOf | 007Child\_Teuku\_Sapeh | Person |
| Data property assertions | hasName | Cut Teungoh | String |

**Tabel B.8 Sampel Data Setelah *Reasoning* Individu 007Child\_Teuku\_Bidin**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Teuku\_Bidin | | |
| Types | Male  Person  Man | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 007Spouse\_Cut\_Meuligou | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasSister | 007Child\_Cut\_Teungoh | Person |
| hasSister | 007Child\_Cut\_Mariyam | Person |
| hasSister | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Cut\_Teungoh | Person |
| hasSibling | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSibling | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Teuku\_Sapeh | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepSister | 006Child\_Cut\_Gambang | Person |
| hasStepSibling | 006Child\_Cut\_Gambang | Person |
| hasBrother | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasBrother | 007Child\_Teuku\_Sapeh | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 007Spouse\_Cut\_Meuligou | Person |
| isBrotherOf | 007Child\_Cut\_Teungoh | Person |
| isBrotherOf | 007Child\_Teuku\_Sjak | Person |
| isBrotherOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isBrotherOf | 007Child\_Cut\_Mariyam | Person |
| isBrotherOf | 007Child\_Teuku\_Sapeh | Person |
| Data property assertions | hasName | Teuku Bidin | String |

**Tabel B.9 Sampel Data Setelah *Reasoning* Individu 007Child\_Teuku\_Raja\_Sulaiman**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Teuku\_Raja\_Sulaiman | | |
| Types | Male  Person | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 007Spouse\_Cut\_Meuligou | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasSister | 007Child\_Cut\_Teungoh | Person |
| hasSister | 007Child\_Cut\_Mariyam | Person |
| hasSister | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Cut\_Teungoh | Person |
| hasSibling | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Teuku\_Bidin | Person |
| hasSibling | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Teuku\_Sapeh | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepSister | 006Child\_Cut\_Gambang | Person |
| hasStepSibling | 006Child\_Cut\_Gambang | Person |
| hasBrother | 007Child\_Teuku\_Bidin | Person |
| hasBrother | 007Child\_Teuku\_Sapeh | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 007Spouse\_Cut\_Meuligou | Person |
| isBrotherOf | 007Child\_Cut\_Teungoh | Person |
| isBrotherOf | 007Child\_Teuku\_Sjak | Person |
| isBrotherOf | 007Child\_Teuku\_Bidin | Person |
| isBrotherOf | 007Child\_Cut\_Mariyam | Person |
| isBrotherOf | 007Child\_Teuku\_Sapeh | Person |
| Data property assertions | hasName | Teuku Raja Sulaiman | String |

**Tabel B.10 Sampel Data Setelah *Reasoning* Individu 007Child\_Teuku\_Sapeh**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Child\_Teuku\_Sapeh | | |
| Types | Male  Person  Man | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasParent | 007Spouse\_Cut\_Meuligou | Person |
| hasParent | 007\_Teuku\_Umar | Person |
| hasMother | 007Spouse\_Cut\_Meuligou | Person |
| hasFather | 007\_Teuku\_Umar | Person |
| hasStepMother | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepMother | 006\_Cut\_Nyak\_Dhien | Person |
| hasSister | 007Child\_Cut\_Teungoh | Person |
| hasSister | 007Child\_Cut\_Mariyam | Person |
| hasSister | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Cut\_Teungoh | Person |
| hasSibling | 007Child\_Cut\_Sjak | Person |
| hasSibling | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSibling | 007Child\_Cut\_Mariyam | Person |
| hasSibling | 007Child\_Teuku\_Bidin | Person |
| hasStepParent | 007Spouse\_Cut\_Nyak\_Sofiah | Person |
| hasStepParent | 006\_Cut\_Nyak\_Dhien | Person |
| hasStepSister | 006Child\_Cut\_Gambang | Person |
| hasStepSibling | 006Child\_Cut\_Gambang | Person |
| hasBrother | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasBrother | 007Child\_Teuku\_Bidin | Person |
| isChildOf | 007\_Teuku\_Umar | Person |
| isChildOf | 007Spouse\_Cut\_Meuligou | Person |
| isBrotherOf | 007Child\_Cut\_Teungoh | Person |
| isBrotherOf | 007Child\_Teuku\_Sjak | Person |
| isBrotherOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isBrotherOf | 007Child\_Cut\_Mariyam | Person |
| isBrotherOf | 007Child\_Teuku\_Bidin | Person |
| Data property assertions | hasName | Teuku Sapeh | String |

**Tabel B.11 Sampel Data Setelah *Reasoning* Individu 007Spouse\_Cut\_Meuligou**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Spouse\_Cut\_Meuligou | | |
| Types | Female  Person  Agent  Woman | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasChild | 007Child\_Cut\_Teungoh | Person |
| hasChild | 007Child\_Teuku\_Bidin | Person |
| hasChild | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasChild | 007Child\_Cut\_Sjak | Person |
| hasChild | 007Child\_Cut\_Mariyam | Person |
| hasChild | 007Child\_Teuku\_Sapeh | Person |
| hasStepChild | 006Child\_Cut\_Gambang | Person |
| hasDaughter | 007Child\_Cut\_Teungoh | Person |
| hasDaughter | 007Child\_Cut\_Sjak | Person |
| hasDaughter | 007Child\_Cut\_Mariyam | Person |
| hasSon | 007Child\_Teuku\_Bidin | Person |
| hasSon | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasSon | 007Child\_Teuku\_Sapeh | Person |
| hasHusband | 007\_Teuku\_Umar | Person |
| hasStepDaughter | 006Child\_Cut\_Gambang | Person |
| isSpouseOf | 007\_Teuku\_Umar | Person |
| isParentOf | 007Child\_Cut\_Teungoh | Person |
| isParentOf | 007Child\_Teuku\_Bidin | Person |
| isParentOf | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| isParentOf | 007Child\_Cut\_Sjak | Person |
| isParentOf | 007Child\_Cut\_Mariyam | Person |
| isParentOf | 007Child\_Teuku\_Sapeh | Person |
| isInLawOf | 007\_Teuku\_Umar | Person |
| isWifeOf | 007\_Teuku\_Umar | Person |
| differentFrom | 007Marriage1 | Marriage |
| ‘Life Event’ | 007Marriage1 | Marriage |
| Data property assertions | hasName | Cut Meuligou | String |
| alias | Nyak Malighai | String |

**Tabel B.12 Sampel Data Setelah *Reasoning* Individu 007Spouse\_Cut\_Nyak\_Sofiah**

|  |  |  |  |
| --- | --- | --- | --- |
| Individuals | 007Spouse\_Cut\_Nyak\_Sofiah | | |
| Types | Female  Person  Agent  Woman | | |
|  | Property Name | Value | Property Types |
| Object property assertions | hasStepChild | 007Child\_Cut\_Teungoh | Person |
| hasStepChild | 007Child\_Teuku\_Bidin | Person |
| hasStepChild | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasStepChild | 006Child\_Cut\_Gambang | Person |
| hasStepChild | 007Child\_Cut\_Sjak | Person |
| hasStepChild | 007Child\_Cut\_Mariyam | Person |
| hasStepChild | 007Child\_Teuku\_Sapeh | Person |
| hasHusband | 007\_Teuku\_Umar | Person |
| hasStepSon | 007Child\_Teuku\_Bidin | Person |
| hasStepSon | 007Child\_Teuku\_Raja\_Sulaiman | Person |
| hasStepSon | 007Child\_Teuku\_Sapeh | Person |
| hasStepDaughter | 007Child\_Cut\_Teungoh | Person |
| hasStepDaughter | 006Child\_Cut\_Gambang | Person |
| hasStepDaughter | 007Child\_Cut\_Sjak | Person |
| hasStepDaughter | 007Child\_Cut\_Mariyam | Person |
| isSpouseOf | 007\_Teuku\_Umar | Person |
| isInLawOf | 007\_Teuku\_Umar | Person |
| isWifeOF | 007\_Teuku\_Umar | Person |
| differentFrom | 007Marriage2 | Marriage |
| ‘Life Event’ | 007Marriage2 | Marriage |
| Data property assertions | hasName | Cut Nyak Sofiah | String |

# 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.