

**TUGAS AKHIR – KI141502**

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

**FAIQ**

**NRP. 5115 100 007**

**Dosen Pembimbing 1**

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

**Dosen Pembimbing 2**

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

**DEPARTEMEN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2016**

2016

# Description: its_logo_gifHALAMAN JUDUL

**TUGAS AKHIR – KI141502**

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

**FAIQ**

**NRP. 5115 100 007**

**Dosen Pembimbing 1**

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

**Dosen Pembimbing 2**

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

**DEPARTEMEN TEKNIK 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)***

# 

# D:\IF\SMT 8\Tugas Akhir\Lembar Pengesahan\Lembar Pengesahan0003.jpgLEMBAR PENGESAHAN

**PENCARIAN RELASI ANTAR TOKOH SEJARAH INDONESIA MENGGUNAKAN ONTOLOGI**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

pada

Rumpun Mata Kuliah Manajemen Informasi

Program Studi S-1 Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

**Oleh:**

**MADIS SARALITA**

**NRP. 5112 100 038**

Disetujui oleh Pembimbing Tugas Akhir:

1. Sarwosri, S.Kom., M.T. ........................

NIP. 19760809 200112 2 001 (Pembimbing 1)

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

NIP. 19860722 201504 2 003 (Pembimbing 2)

**SURABAYA**

**JUNI, 2016**

***(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 : 5115100007**

**Jurusan : Teknik Informatika**

**Fakultas Teknologi Informasi dan Komunikasi 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 : Department of Informatics**

**Faculty of Information Technology and Communication 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 Teknik 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](#_Toc11753216)

[3 LEMBAR PENGESAHAN vii](#_Toc11753217)

[4 ABSTRAK ix](#_Toc11753218)

[5 ABSTRACT xi](#_Toc11753219)

[6 KATA PENGANTAR xiii](#_Toc11753220)

[7 DAFTAR ISI xv](#_Toc11753221)

[9 DAFTAR GAMBAR xix](#_Toc11753222)

[DAFTAR TABEL xxi](#_Toc11753223)

[DAFTAR KODE SUMBER xxv](#_Toc11753224)

[1 BAB I PENDAHULUAN 1](#_Toc11753225)

[1.1. Latar Belakang 1](#_Toc11753226)

[1.2. Batasan Masalah 2](#_Toc11753227)

[1.3. Tujuan 3](#_Toc11753228)

[1.4. Metodologi 3](#_Toc11753229)

[1.5. Sistematika Penulisan 4](#_Toc11753230)

[2 BAB II DASAR TEORI 7](#_Toc11753231)

[2.1. Tokoh Bersejarah 7](#_Toc11753232)

[2.2. Ontologi 7](#_Toc11753233)

[2.3. DBpedia 10](#_Toc11753234)

[2.4. Semantic Web Rule Language (SWRL) 11](#_Toc11753235)

[2.5. Family Relationships Ontology 12](#_Toc11753236)

[2.6. SPARQL 14](#_Toc11753237)

[2.7. Apache Jena Fuseki 15](#_Toc11753238)

[2.8. Pellet Reasoner 16](#_Toc11753239)

[2.9. SPARQL Lib 18](#_Toc11753240)

[2.10. Apache Jena 19](#_Toc11753241)

[3 BAB III METODOLOGI PEMECAHAN MASALAH 20](#_Toc11753242)

[3.1. Analisis Data 21](#_Toc11753243)

[3.1.1. Analisis Data dari DBpedia 22](#_Toc11753244)

[3.2. Ekstraksi Data Sebagai Model 23](#_Toc11753245)

[3.3. Pembuatan Ontologi 24](#_Toc11753246)

[3.4. Penggabungan model data dan model ontologi 27](#_Toc11753247)

[3.5. Reasoning pada Model Gabungan 28](#_Toc11753248)

[3.6. Penampilan Data 30](#_Toc11753249)

[4 BAB IV ANALISIS DAN PERANCANGAN SISTEM 32](#_Toc11753250)

[4.1. Analisis 32](#_Toc11753251)

[4.1.1. Cakupan Permasalahan 32](#_Toc11753252)

[4.1.2. Deskripsi Umum Sistem 32](#_Toc11753253)

[4.1.3. Spesifikasi Kebutuhan Perangkat Lunak 33](#_Toc11753254)

[4.1.4. Aktor 34](#_Toc11753255)

[4.1.5. Kasus Penggunaan 34](#_Toc11753256)

[4.2. Perancangan Antarmuka Pengguna 39](#_Toc11753257)

[5 BAB V IMPLEMENTASI 42](#_Toc11753258)

[5.1. Implementasi Fungsi 43](#_Toc11753259)

[5.1.1. Fungsi Dropdown Select 43](#_Toc11753260)

[5.1.2. Fungsi Get Instance Name 44](#_Toc11753261)

[5.1.3. Fungsi Get Description 45](#_Toc11753262)

[5.1.4. Fungsi Get Relations 50](#_Toc11753263)

[5.1. Implementasi Antarmuka Pengguna 68](#_Toc11753264)

[5.1.1. Implementasi Tampilan Halaman Utama 68](#_Toc11753265)

[5.1.2. Implementasi Tampilan Halaman Informasi 69](#_Toc11753266)

[6 BAB VI PENGUJIAN DAN EVALUASI 70](#_Toc11753267)

[6.1. Lingkungan Pengujian 70](#_Toc11753268)

[6.2. Skenario Pengujian 70](#_Toc11753269)

[6.2.1. Pengujian Ontologi 71](#_Toc11753270)

[6.2.2. Pengujian Perbandingan Data 105](#_Toc11753271)

[6.2.3. Pengujian Kompleksitas Ontologi 127](#_Toc11753272)

[6.3. Evaluasi Pengujian 130](#_Toc11753273)

[6.3.1. Evaluasi Pengujian Ontologi 130](#_Toc11753274)

[6.3.2. Evaluasi Pengujian Perbandingan Data 132](#_Toc11753275)

[6.3.1. Evaluasi Pengujian Kompleksitas Ontologi 133](#_Toc11753276)

[7 BAB VII KESIMPULAN DAN SARAN 135](#_Toc11753277)

[**1.** 135](#_Toc11753278)

[7.1. Kesimpulan 135](#_Toc11753279)

[7.2. Saran 136](#_Toc11753280)

[DAFTAR PUSTAKA 137](#_Toc11753281)

[8 LAMPIRAN A. DATA SEBELUM *REASONING* 141](#_Toc11753282)

[9 LAMPIRAN B. DATA SETELAH *REASONING* 153](#_Toc11753283)

[10 BIODATA PENULIS 177](#_Toc11753284)

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

# 

# DAFTAR GAMBAR

[Gambar ‎2.1 Class Hierarchy 13](#_Toc456657349)

[Gambar ‎2.2 Property 14](#_Toc456657350)

[Gambar ‎2.3 Contoh Ontologi 15](file:///D:\IF\SMT%208\Tugas%20Akhir\Buku%20TA%20Madis%204.docx#_Toc456657351)

[Gambar ‎2.4 Family Relationships Map 21](#_Toc456657352)

[Gambar ‎2.5 Ontologi FamilyTree Keluarga Robert Stevens 22](#_Toc456657353)

[Gambar ‎2.6 Arsitektur Pellet Reasoner 30](#_Toc456657354)

[Gambar ‎3.1 Flowchart Pengembangan Ontologi 35](#_Toc456657355)

[Gambar ‎3.2 Halaman Wikipedia Indonesia tentang Sukarno 38](#_Toc456657356)

[Gambar ‎3.3 WikiTaxi HTML tentang Sukarno 40](#_Toc456657357)

[Gambar ‎3.4 Wiki Source tentang Sukarno 41](#_Toc456657358)

[Gambar ‎3.5 Halaman DBpedia tentang Sukarno 42](#_Toc456657359)

[Gambar ‎3.6 Virtuoso SPARQL Query Editor 43](#_Toc456657360)

[Gambar ‎3.7 Hasil SPARQL Query 45](#_Toc456657361)

[Gambar ‎3.8 Google Refine 46](#_Toc456657362)

[Gambar ‎3.9 Data yang redundan dan ambigu 46](#_Toc456657363)

[Gambar ‎3.10 SUMO 58](#_Toc456657364)

[Gambar ‎4.1 Diagram Kasus Penggunaan Sistem 87](#_Toc456657365)

[Gambar ‎4.2 Diagram Aktivitas Memilih Entitas Tokoh 89](#_Toc456657366)

[Gambar ‎4.3 Diagram Aktivitas Melihat Informasi Tokoh 90](#_Toc456657367)

[Gambar ‎4.4 Diagram Aktivitas Memilih Entitas Tautan 91](#_Toc456657368)

[Gambar ‎4.5 Antarmuka Halaman Utama iHerolation 92](#_Toc456657369)

[Gambar ‎4.6 Antarmuka Halaman Informasi iHerolation 93](#_Toc456657370)

[Gambar ‎5.1 Arsitektur Perangkat Lunak 95](#_Toc456657371)

[Gambar ‎5.2 Implementasi Antarmuka Halaman Utama 122](#_Toc456657372)

[Gambar ‎5.3 Implementasi Antarmuka Halaman Informasi 122](#_Toc456657373)

[Gambar ‎6.1 Uji Coba Relasi isSpouseOf 125](#_Toc456657374)

[Gambar ‎6.2 Uji Coba Relasi hasHusband 127](#_Toc456657375)

[Gambar ‎6.3 Uji Coba Relasi hasWife 128](#_Toc456657376)

[Gambar ‎6.4 Uji Coba Relasi hasFather 130](#_Toc456657377)

[Gambar ‎6.5 Uji Coba Relasi hasMother 131](#_Toc456657378)

[Gambar ‎6.6 Uji Coba Relasi hasChild 133](#_Toc456657379)

[Gambar ‎6.7 Uji Coba Relasi hasSon 134](#_Toc456657380)

[Gambar ‎6.8 Uji Coba Relasi hasDaughter 136](#_Toc456657381)

[Gambar ‎6.9 Uji Coba Relasi hasSibling 137](#_Toc456657382)

[Gambar ‎6.10 Uji Coba Relasi hasBrother 139](#_Toc456657383)

[Gambar ‎6.11 Uji Coba Relasi hasSister 140](#_Toc456657384)

[Gambar ‎6.12 Uji Coba Relasi hasStepParent 142](#_Toc456657385)

[Gambar ‎6.13 Uji Coba Relasi hasStepFather 143](#_Toc456657386)

[Gambar ‎6.14 Uji Coba Relasi hasStepMother 145](#_Toc456657387)

[Gambar ‎6.15 Uji Coba Relasi hasStepChild 146](#_Toc456657388)

[Gambar ‎6.16 Uji Coba Relasi hasStepSon 148](#_Toc456657389)

[Gambar ‎6.17 Uji Coba Relasi hasStepDaughter 149](#_Toc456657390)

[Gambar ‎6.18 Uji Coba Relasi hasStepSibling 151](#_Toc456657391)

[Gambar ‎6.19 Uji Coba Relasi hasStepBrother 152](#_Toc456657392)

[Gambar ‎6.20 Uji Coba Relasi hasStepSister 154](#_Toc456657393)

[Gambar ‎6.21 Uji Coba Relasi isAllyOf 155](#_Toc456657394)

[Gambar ‎6.22 Uji Coba Relasi isEnemyOf 157](#_Toc456657395)

# DAFTAR TABEL

[Tabel ‎2.1 Komponen SWRL 19](#_Toc456657480)

[Tabel ‎2.2 FOAF Core 23](#_Toc456657481)

[Tabel ‎2.3 FOAF Social Web 24](#_Toc456657482)

[Tabel ‎2.4 FOAF Classes & Properties 25](#_Toc456657483)

[Tabel ‎2.5 Bio Terms 28](#_Toc456657484)

[Tabel ‎2.6 EasyRDF Methods 32](#_Toc456657485)

[Tabel ‎2.7 Method pada PHP 34](#_Toc456657486)

[Tabel ‎3.1 Data hasil ekstrak Sukarno 38](#_Toc456657487)

[Tabel ‎3.2 Hasil klasifikasi data tunggal 47](#_Toc456657488)

[Tabel ‎3.3 Hasil klasifikasi data lebih dari satu person 48](#_Toc456657489)

[Tabel ‎3.4 Seleksi Class dari BIO 51](#_Toc456657490)

[Tabel ‎3.5 Seleksi Object Properties dari BIO 52](#_Toc456657491)

[Tabel ‎3.6 Import Classes dari FamilyTree 53](#_Toc456657492)

[Tabel ‎3.7 Import Object Properties dari FamilyTree 54](#_Toc456657493)

[Tabel ‎3.8 Import Data Properties dari FamilyTree 55](#_Toc456657494)

[Tabel ‎3.9 Import Classes dari FOAF 56](#_Toc456657495)

[Tabel ‎3.10 Equivalent classes dari BIO 57](#_Toc456657496)

[Tabel ‎3.11 Equivalent properties dari FamilyTree 57](#_Toc456657497)

[Tabel ‎3.12 Daftar properti yang memiliki kemiripan 59](#_Toc456657498)

[Tabel ‎3.13 Pencarian daftar kata No. 1 menggunakan SUMO 59](#_Toc456657499)

[Tabel ‎3.14 Pencarian daftar kata No. 2 menggunakan SUMO 59](#_Toc456657500)

[Tabel ‎3.15 Pencarian daftar kata No. 3 menggunakan SUMO 60](#_Toc456657501)

[Tabel ‎3.16 Pencarian daftar kata No. 4 menggunakan SUMO 60](#_Toc456657502)

[Tabel ‎3.17 Pencarian daftar kata No. 5 menggunakan SUMO 60](#_Toc456657503)

[Tabel ‎3.18 Pencarian daftar kata No. 6 menggunakan SUMO 60](#_Toc456657504)

[Tabel ‎3.19 Pencarian daftar kata No. 7 menggunakan SUMO 61](#_Toc456657505)

[Tabel ‎3.20 Kelas baru di ihero 62](#_Toc456657506)

[Tabel ‎3.21 Properti objek baru di ihero 62](#_Toc456657507)

[Tabel ‎3.22 Properti data baru di ihero 67](#_Toc456657508)

[Tabel ‎4.1 Daftar Kebutuhan Fungsional Perangkat Lunak 86](#_Toc456657509)

[Tabel ‎4.2 Daftar Kode Diagram Kasus Penggunaan 88](#_Toc456657510)

[Tabel ‎4.3 Spesifikasi Usecase Memilih Entitas Tokoh 88](#_Toc456657511)

[Tabel ‎4.4 Spesifikasi Usecase Melihat Informasi Tokoh 89](#_Toc456657512)

[Tabel ‎4.5 Spesifikasi Usecase Memilih Entitas Tautan 91](#_Toc456657513)

[Tabel ‎4.6 Spesifikasi Atribut Halaman iHerolation 93](#_Toc456657514)

[Tabel ‎6.1 Pengujian Kevalidan Relasi isSpouseOf 124](#_Toc456657515)

[Tabel ‎6.2 Pengujian Kevalidan Relasi hasHusband 126](#_Toc456657516)

[Tabel ‎6.3 Pengujian Kevalidan Relasi hasWife 127](#_Toc456657517)

[Tabel ‎6.4 Pengujian Kevalidan Relasi hasFather 129](#_Toc456657518)

[Tabel ‎6.5 Pengujian Kevalidan Relasi hasMother 130](#_Toc456657519)

[Tabel ‎6.6 Pengujian Kevalidan Relasi hasChild 132](#_Toc456657520)

[Tabel ‎6.7 Pengujian Kevalidan Relasi hasSon 133](#_Toc456657521)

[Tabel ‎6.8 Pengujian Kevalidan Relasi hasDaughter 135](#_Toc456657522)

[Tabel ‎6.9 Pengujian Kevalidan Relasi hasSibling 136](#_Toc456657523)

[Tabel ‎6.10 Pengujian Kevalidan Relasi hasBrother 138](#_Toc456657524)

[Tabel ‎6.11 Pengujian Kevalidan Relasi hasSister 139](#_Toc456657525)

[Tabel ‎6.12 Pengujian Kevalidan Relasi hasStepParent 141](#_Toc456657526)

[Tabel ‎6.13 Pengujian Kevalidan Relasi hasStepFather 142](#_Toc456657527)

[Tabel ‎6.14 Pengujian Kevalidan Relasi hasStepMother 144](#_Toc456657528)

[Tabel ‎6.15 Pengujian Kevalidan Relasi hasStepChild 145](#_Toc456657529)

[Tabel ‎6.16 Pengujian Kevalidan Relasi hasStepSon 147](#_Toc456657530)

[Tabel ‎6.17 Pengujian Kevalidan Relasi hasStepDaughter 148](#_Toc456657531)

[Tabel ‎6.18 Pengujian Kevalidan Relasi hasStepSibling 150](#_Toc456657532)

[Tabel ‎6.19 Pengujian Kevalidan Relasi hasStepBrother 151](#_Toc456657533)

[Tabel ‎6.20 Pengujian Kevalidan Relasi hasStepSister 153](#_Toc456657534)

[Tabel ‎6.21 Pengujian Kevalidan Relasi isAllyOf 154](#_Toc456657535)

[Tabel ‎6.22 Pengujian Kevalidan Relasi isEnemyOf 156](#_Toc456657536)

[Tabel ‎6.23 Pengujian Perbandingan Data Sukarno 158](#_Toc456657537)

[Tabel ‎6.24 Pengujian Perbandingan Data Fatmawati 159](#_Toc456657538)

[Tabel ‎6.25 Pengujian Perbandingan Data Mohammad Hatta 160](#_Toc456657539)

[Tabel ‎6.26 Pengujian Perbandingan Data Hasyim Asyari 161](#_Toc456657540)

[Tabel ‎6.27 Pengujian Perbandingan Data Mohammad Natsir 162](#_Toc456657541)

[Tabel ‎6.28 Pengujian Perbandingan Data Cut Nyak Dhien 163](#_Toc456657542)

[Tabel ‎6.29 Pengujian Perbandingan Data Teuku Umar 164](#_Toc456657543)

[Tabel ‎6.30 Pengujian Perbandingan Data Cut Nyak Meutia 165](#_Toc456657544)

[Tabel ‎6.31 Pengujian Perbandingan Data Teuku Nyak Arif 166](#_Toc456657545)

[Tabel ‎6.32 Pengujian Perbandingan Data Pierre Tendean 167](#_Toc456657546)

[Tabel ‎6.33 Pengujian Perbandingan Data A. Haris Nasution 169](#_Toc456657547)

[Tabel ‎6.34 Pengujian Perbandingan Data Ahmad Yani 170](#_Toc456657548)

[Tabel ‎6.35 Pengujian Perbandingan Data Ahmad Dahlan 171](#_Toc456657549)

[Tabel ‎6.36 Pengujian Perbandingan Data Nyai Ahmad Dahlan 172](#_Toc456657550)

[Tabel ‎6.37 Pengujian Perbandingan Data Imam Bonjol 173](#_Toc456657551)

[Tabel ‎6.38 Pengujian Perbandingan Data Wahid Hasyim 174](#_Toc456657552)

[Tabel ‎6.39 Pengujian Perbandingan Data Mas Mansoer 175](#_Toc456657553)

[Tabel ‎6.40 Pengujian Perbandingan Data Diponegoro 176](#_Toc456657554)

[Tabel ‎6.41 Pengujian Perbandingan Data Sudirman 177](#_Toc456657555)

[Tabel ‎6.42 Pengujian Perbandingan Data Slamet Rijadi 179](#_Toc456657556)

[Tabel ‎6.43 Pengujian Kompleksitas Reasoning Ontologi 180](#_Toc456657557)

[Tabel ‎6.44 Pengujian Kompleksitas Export Axiom 181](#_Toc456657558)

[Tabel ‎6.45 Rangkuman Hasil Pengujian 183](#_Toc456657559)

[Tabel ‎6.46 Rangkuman Hasil Pengujian 184](#_Toc456657560)

[Tabel ‎6.47 Rangkuman Hasil Pengujian 185](#_Toc456657561)

[Tabel ‎6.48 Rangkuman Hasil Evaluasi Pengujian 186](#_Toc456657562)

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

# DAFTAR KODE SUMBER

[Kode Sumber ‎3.1 Kode SPARQL 44](#_Toc456658489)

[Kode Sumber ‎5.1 Fungsi Dropdown Select 97](#_Toc456658490)

[Kode Sumber ‎5.2 Fungsi Get Instance Name 97](#_Toc456658491)

[Kode Sumber ‎5.3 Fungsi Get name 98](#_Toc456658492)

[Kode Sumber ‎5.4 Fungsi Get alias 98](#_Toc456658493)

[Kode Sumber ‎5.5 Fungsi Get almamater 99](#_Toc456658494)

[Kode Sumber ‎5.6 Fungsi Get birth date 99](#_Toc456658495)

[Kode Sumber ‎5.7 Fungsi Get birth place 99](#_Toc456658496)

[Kode Sumber ‎5.8 Fungsi Get birth year 100](#_Toc456658497)

[Kode Sumber ‎5.9 Fungsi Get death date 100](#_Toc456658498)

[Kode Sumber ‎5.10 Fungsi Get death place 101](#_Toc456658499)

[Kode Sumber ‎5.11 Fungsi Get death year 101](#_Toc456658500)

[Kode Sumber ‎5.12 Fungsi Get religion 101](#_Toc456658501)

[Kode Sumber ‎5.13 Fungsi Get alternative names 102](#_Toc456658502)

[Kode Sumber ‎5.14 Fungsi Get honorific prefix 102](#_Toc456658503)

[Kode Sumber ‎5.15 Fungsi Get office 103](#_Toc456658504)

[Kode Sumber ‎5.16 Fungsi Get parent 103](#_Toc456658505)

[Kode Sumber ‎5.17 Fungsi Get father 104](#_Toc456658506)

[Kode Sumber ‎5.18 Fungsi Get mother 104](#_Toc456658507)

[Kode Sumber ‎5.19 Fungsi Get stepparent 105](#_Toc456658508)

[Kode Sumber ‎5.20 Fungsi Get stepfather 105](#_Toc456658509)

[Kode Sumber ‎5.21 Fungsi Get stepmother 106](#_Toc456658510)

[Kode Sumber ‎5.22 Fungsi Get child 106](#_Toc456658511)

[Kode Sumber ‎5.23 Fungsi Get son 107](#_Toc456658512)

[Kode Sumber ‎5.24 Fungsi Get daughter 107](#_Toc456658513)

[Kode Sumber ‎5.25 Fungsi Get stepchild 108](#_Toc456658514)

[Kode Sumber ‎5.26 Fungsi Get stepson 108](#_Toc456658515)

[Kode Sumber ‎5.27 Fungsi Get stepdaughter 109](#_Toc456658516)

[Kode Sumber ‎5.28 Fungsi Get sibling 109](#_Toc456658517)

[Kode Sumber ‎5.29 Fungsi Get brother 110](#_Toc456658518)

[Kode Sumber ‎5.30 Fungsi Get sister 110](#_Toc456658519)

[Kode Sumber ‎5.31 Fungsi Get stepsibling 111](#_Toc456658520)

[Kode Sumber ‎5.32 Fungsi Get stepbrother 111](#_Toc456658521)

[Kode Sumber ‎5.33 Fungsi Get stepsister 112](#_Toc456658522)

[Kode Sumber ‎5.34 Fungsi Get party 112](#_Toc456658523)

[Kode Sumber ‎5.35 Fungsi Get prime minister 113](#_Toc456658524)

[Kode Sumber ‎5.36 Fungsi Get spouse 113](#_Toc456658525)

[Kode Sumber ‎5.37 Fungsi Get successor 114](#_Toc456658526)

[Kode Sumber ‎5.38 Fungsi Get vice president 114](#_Toc456658527)

[Kode Sumber ‎5.39 Fungsi Get predecessor 115](#_Toc456658528)

[Kode Sumber ‎5.40 Fungsi Get is child of 115](#_Toc456658529)

[Kode Sumber ‎5.41 Fungsi Get is commander of 116](#_Toc456658530)

[Kode Sumber ‎5.42 Fungsi Get is influenced of 116](#_Toc456658531)

[Kode Sumber ‎5.43 Fungsi Get is person of 117](#_Toc456658532)

[Kode Sumber ‎5.44 Fungsi Get is predecessor of 117](#_Toc456658533)

[Kode Sumber ‎5.45 Fungsi Get is president of 118](#_Toc456658534)

[Kode Sumber ‎5.46 Fungsi Get is spouse of 118](#_Toc456658535)

[Kode Sumber ‎5.47 Fungsi Get is successor of 119](#_Toc456658536)

[Kode Sumber ‎5.48 Fungsi Get isGovernmentHeadOf 119](#_Toc456658537)

[Kode Sumber ‎5.49 Fungsi Get isInauguralOf 120](#_Toc456658538)

[Kode Sumber ‎5.50 Fungsi Get is leader of 120](#_Toc456658539)

[Kode Sumber ‎5.51 Fungsi Get subject 121](#_Toc456658540)

# 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 2.2.1.



Gambar 2.2.1 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 ada di Gambar 2.2.2:



Gambar 2.2.2 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 2.2.3.



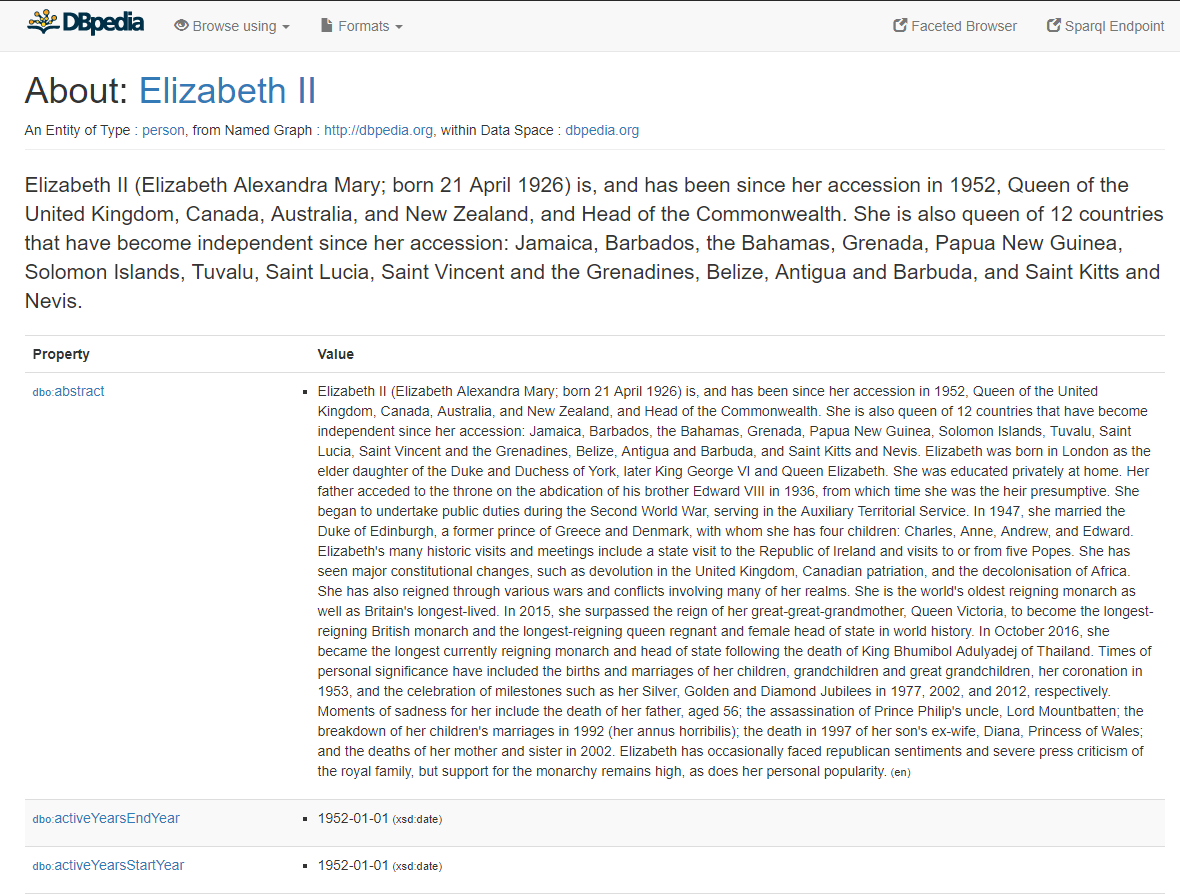
Gambar 2.2.3 Class, Property dan Instance

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* [8].

## 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 2.3.1adalah contoh sebuah halaman DBpedia.



Gambar 2.3.1 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 [13]. 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 [14].

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) [15]. 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 2.5.1.



Gambar 2.5.1 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 2.6.1 :



Gambar 2.6.1 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 Gambar 2.7.1 menunjukkan daftar basis data yang ada di dalam server Apache Jena Fuseki.



Gambar 2.7.1 Database Triple Store Apache Jena Fuseki

Apache Jena Fuseki menyediakan beberapa API untuk digunakan oleh peneliti seperti

## Pellet Reasoner



Gambar 2.8.1 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 [20]. 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 [21].

## 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 2.9.1:



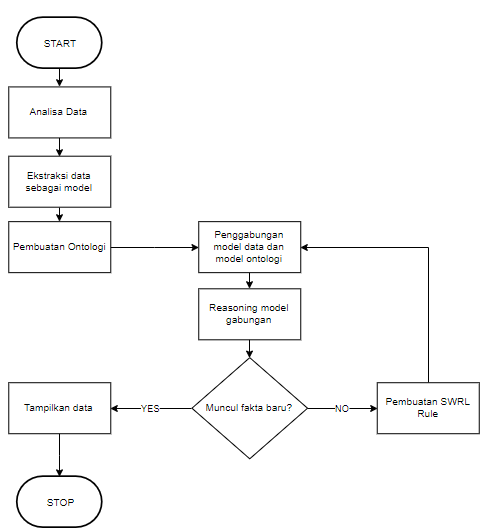
Gambar 2.9.1 Contoh Penggunaan SPARQL Lib

## 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 2.10.1 Flowchart pengembangan

Alur pemecahan masalah dapat dilihat pada Gambar 2.10.1. 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

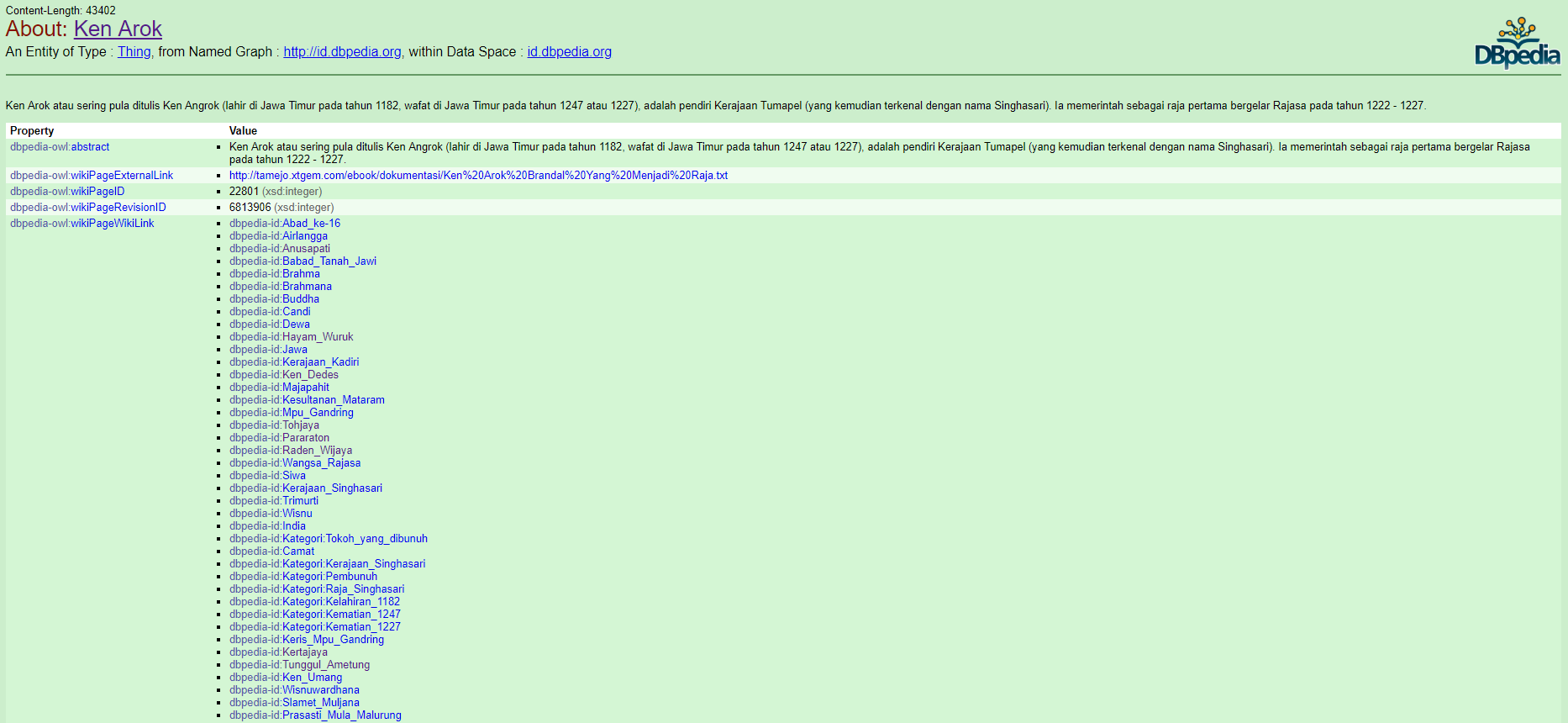
Untuk memecahkan masalah pencarian relasi keluarga tokoh bersejarah, langkah yang pertama kali dilakukan adalah menganalisis dataset yang akan digunakan. Ontologi memiliki beberapa domain, yaitu *actor, place, time, dan event*. Domain yang menjadi topik pada pengerjaan tugas akhir ini adalah *actor*. 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 [28].

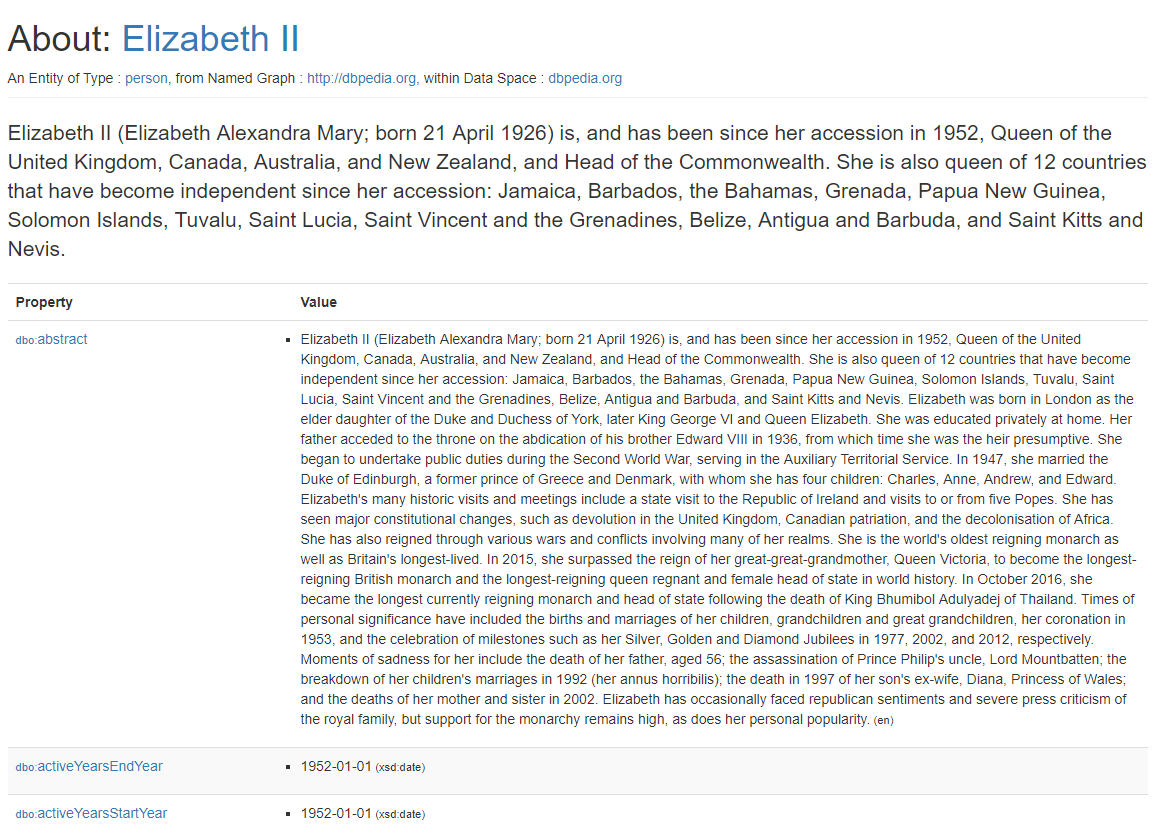
### Analisis Data dari DBpedia

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, Gambar 3.1.2 berikut adalah informasi yang dimiliki DBpedia Indonesia tentang Ken Arok. Informasi tersebut dapat diakses melalui alamat [*https://id.dbpedia.org/wiki/Ken\_Arok*](https://id.dbpedia.org/wiki/Ken_Arok)*.*



Gambar 3.1.1 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 3.1.2.



Gambar 3.1.2 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

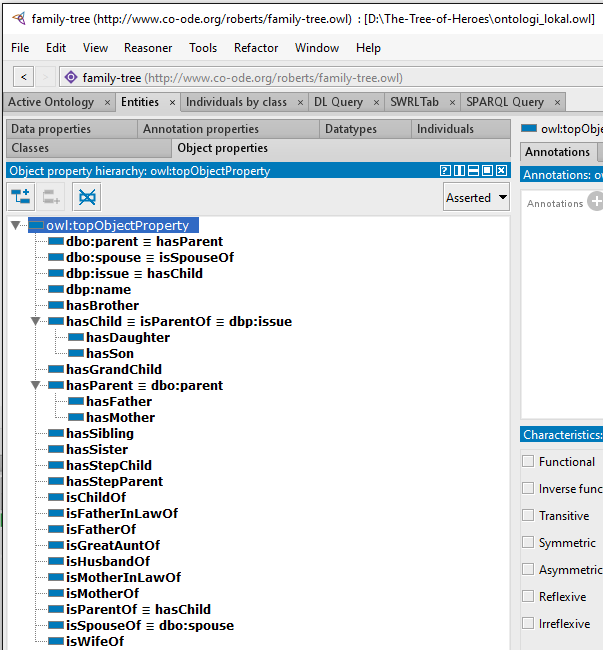
Pada tahap ini, ontologi dibangun dengan menggabungkan beberapa ontologi yang sudah ada. Ontologi yang digunakan adalah *Family Relationships Ontology* milik Robert Stevens. Akan tetapi tidak semua *class, individual, data properties,* ataupun *object properties* akan digunakan, hanya yang benar-benar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



Gambar 3.3.1 Hirarki Class



Gambar 3.3.2 Hirarki Data Property



Gambar 3.3.3 Hirarki Object Property

Berikut pula SWRL Rule yang dibutuhkan untuk menampilkan fakta-fakta baru dengan proses *reasoning*.

hasBiologicalChild:

isSpouseOf(?x, ?y) ^ hasChild(?x, ?z) ^ hasParent(?z, ?x) ^ hasParent(?z, ?y) -> hasChild(?y, ?z)

hasStepChild:

isSpouseOf(?parent, ?firstspouse) ^ isSpouseOf(?parent, ?secondspouse) ^ hasChild(?parent, ?child) ^ hasChild(?firstspouse, ?child) ^ differentFrom(?firstspouse, ?secondspouse) -> hasStepChild(?secondspouse, ?child)

hasStepParent :

isSpouseOf(?parent, ?firstspouse) ^ isSpouseOf(?parent, ?secondspouse) ^ hasChild(?parent, ?child) ^ hasChild(?firstspouse, ?child) ^ differentFrom(?firstspouse, ?secondspouse) -> hasStepParent(?child, ?secondspouse)

## 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. iHerolation 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**.**

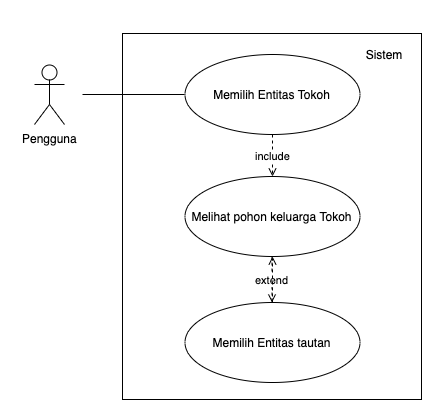
Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak

| **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 4.1.1 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

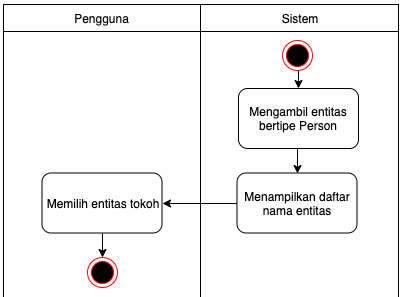
|  |  |
| --- | --- |
| **Kode Kasus Penggunaan** | **Nama** |
| TA-UC0001 | Memilih entitas tokoh |
| TA-UC0002 | Melihat informasi tokoh |
| TA-UC0003 | Memilih entitas tautan |

#### 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 |



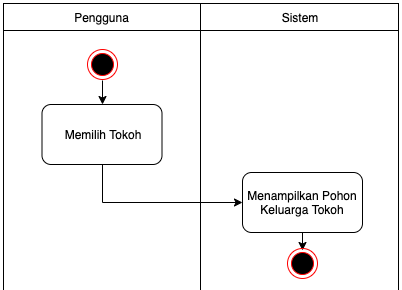
Gambar 4.1.2 Diagram Aktivitas Memilih Entitas Tokoh

#### 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 |



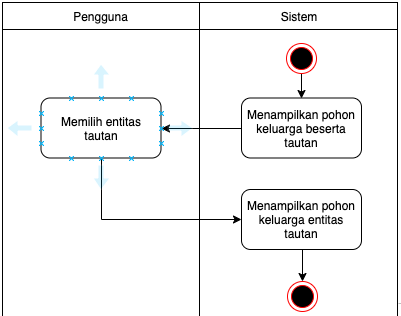
Gambar 4.1.3 Diagram Aktivitas Melihat Pohon Keluarga Tokoh

#### 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 |

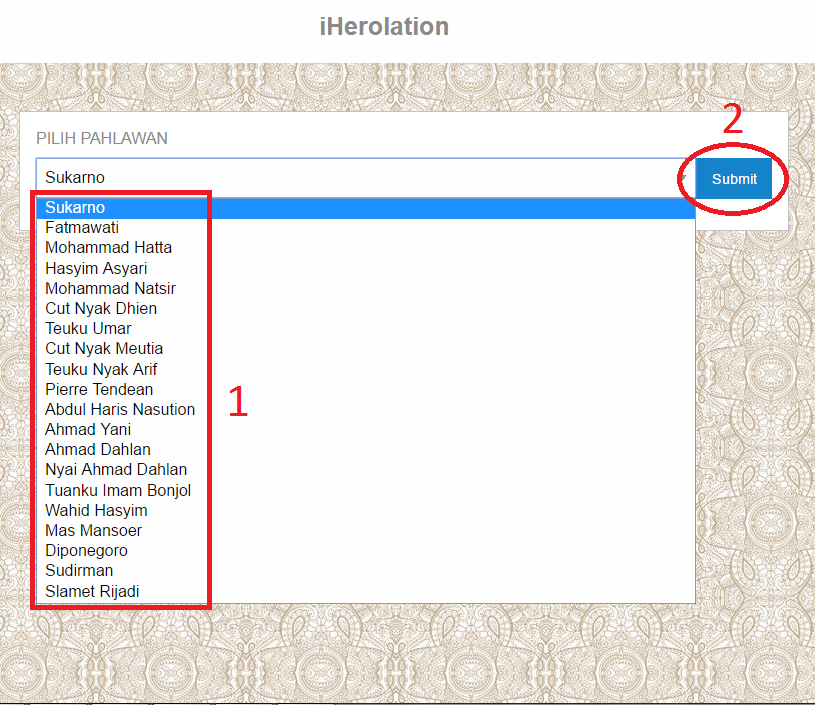


Gambar 4.1.4 Diagram Aktivitas Memilih Entitas Tautan

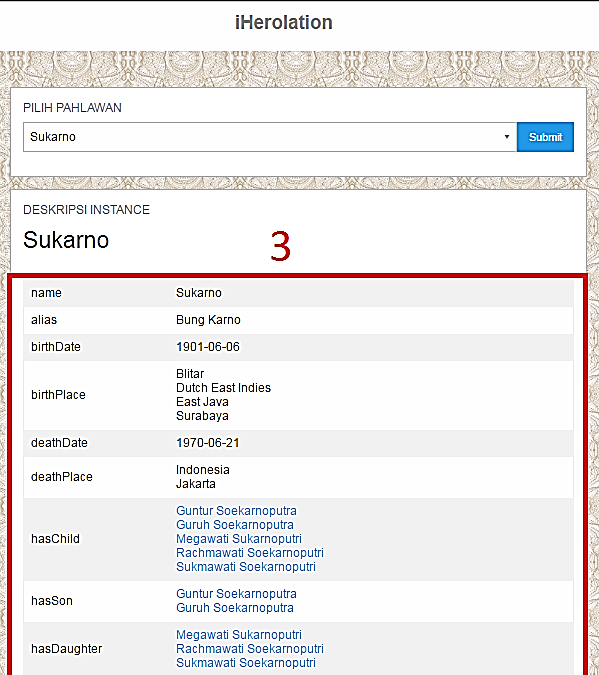
## Perancangan Antarmuka Pengguna

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

Antarmuka iHerolation 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 4.2.1 Antarmuka Halaman Utama Family Tree App



Gambar 4.2.2 Antarmuka Halaman Informasi Family Tree App

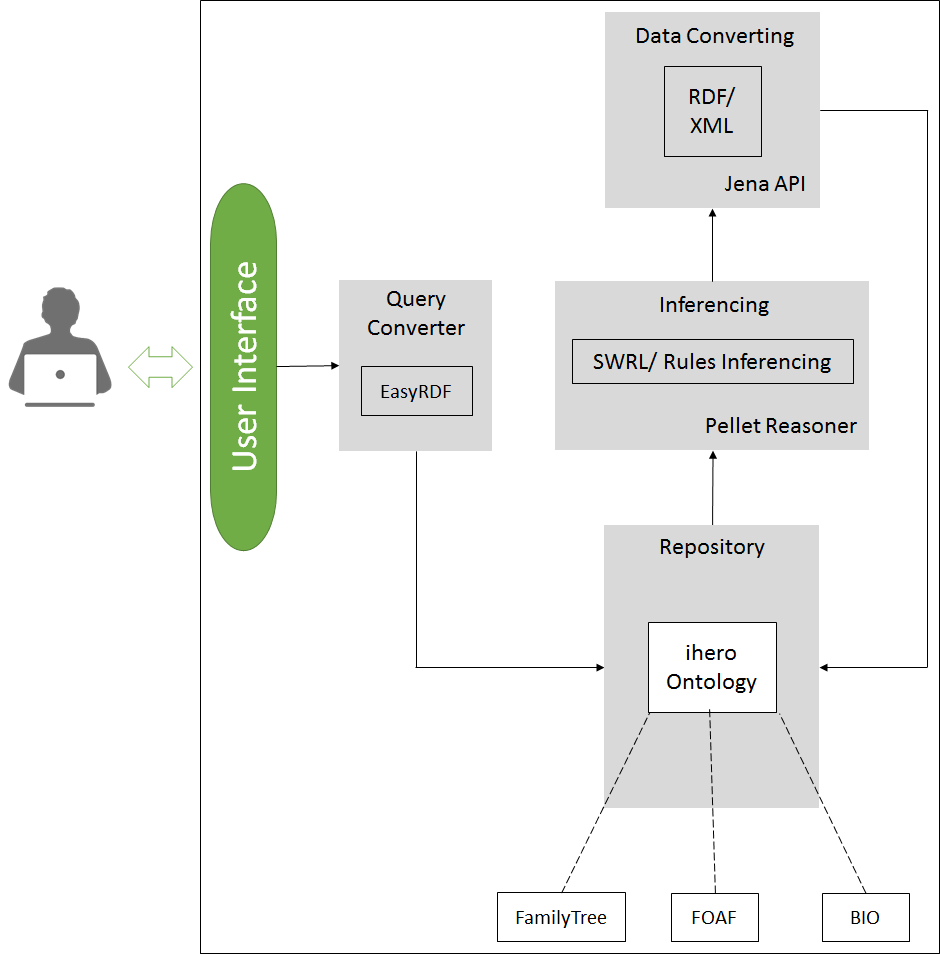
Tabel 4.6 Spesifikasi Atribut Rancangan Antarmuka Halaman Family Tree App

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



Gambar 4.2.1 Arsitektur Perangkat Lunak

Arsitektur perangkat lunak yang dibangun pada tugas akhir ini dapat dilihat pada Gambar 4.2.1 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 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 *file* RDF ontologinya, yaitu memiliki tipe kelas ‘Person. *Query* allOfType digunakan untuk mendapatkan semua tipe ‘Hero’ dari *file* RDF ontologi. allOfType adalah fungsi dari *library* EasyRDF dari kelas graph. Hasil dari *query* disimpan sebagai *value* untuk masing-masing entitas. Kemudian *value* entitas yang dipilih akan dikirim oleh *form* kepada fungsi *get description* ketika tombol *submit* menerima aksi. Implementasi fungsi *dropdown select* dapat dilihat pada Kode Sumber 5.1.

|  |
| --- |
| <form method=**"GET"** action=**"#"**>  <div class=**"input-group"**>  <select class=**"input-group-field"** name=**"entity"**>  <?php  **foreach**($graph->allOfType('ihero:Hero') **as** $name) {  **echo** "<option value='".$name."'>".**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $name->get('familyTree:hasName'))."</option>";  }  ?>  </select>  <div class=**"input-group-button"**>  <input type=**"submit"** class=**"button"** value=**"Submit"**>  </div>  </div>  </form> |

Kode Sumber 5.1 Fungsi Dropdown Select

### Fungsi Get Instance Name

Fungsi *get instance name* digunakan untuk menangkap masukan dari *dropdown select*. *Value* yang ditangkap kemudian berfungsi untuk mengakses nama *instance* yang terdapat dalam *file* RDF ontologi. Implementasi fungsi *get instance name* dapat dilihat pada Kode Sumber 5.2.

|  |
| --- |
| **if**(**isset**($\_GET['entity'])) {  **echo** "  <div class="."callout".">  <h6 class="."subheader".">DESKRIPSI INSTANCE</h6>";    // Nama Instance  $ins = $graph->**resource**($\_GET['entity']);  **echo** "<h3>".**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $ins->get('familyTree:hasName'))."</h3><br/>"; |

Kode Sumber 5.2 Fungsi Get Instance Name

### Fungsi Get Description

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 hasName yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.3.

|  |
| --- |
| **if**($ins->get('familyTree:hasName')){  **echo** "<tr>";  **echo** "<td>name</td><td>".$ins->get('familyTree:hasName')."</td></tr>";  } |

Kode Sumber 5.3 Fungsi Get name

***Get* alias**

Fungsi ini digunakan untuk mendapatkan alias (nama lain) dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti alias yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.4.

|  |
| --- |
| **if**($ins->get('ihero:alias')){  **echo** "<tr>";  **echo** "<td>alias</td><td>".$ins->get('ihero:alias')."</td></tr>";  } |

Kode Sumber 5.4 Fungsi Get alias

***Get* almamater**

Fungsi ini digunakan untuk mendapatkan almamater dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti almamater yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.5.

|  |
| --- |
| **if**($ins->get('ihero:alias')){  **echo** "<tr>";  **echo** "<td>almaMater</td><td>".$ins->get('ihero:almamater')."</td></tr>";} |

Kode Sumber 5.5 Fungsi Get almamater

***Get* birth date**

Fungsi ini digunakan untuk mendapatkan tanggal kelahiran dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasBirthDate yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.6.

|  |
| --- |
| **if**($ins->get('ihero:hasBirthDate')){  **echo** "<tr>";  **echo** "<td>birthDate</td><td>".**str\_replace**('T00:00:00Z', "", $ins->get('ihero:hasBirthDate'))."</td></tr>";  } |

Kode Sumber 5.6 Fungsi Get birth date

***Get* birth place**

Fungsi ini digunakan untuk mendapatkan tempat kelahiran dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasBirthPlace yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.7.

|  |
| --- |
| **if**($ins->get('ihero:hasBirthPlace')){  **echo** "<tr><td>birthPlace</td><td>";  **foreach** ($ins->all('ihero:hasBirthPlace') **as** $subject) {  **echo** **str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject)."<br/>";  }  **echo** "</td></tr>";} |

Kode Sumber 5.7 Fungsi Get birth place

***Get* birth year**

Fungsi ini digunakan untuk mendapatkan tahun kelahiran dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasBirthYear yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.8.

|  |
| --- |
| **if**($ins->get('familyTree:hasBirthYear')){  **echo** "<tr>";  **echo** "<td>birthYear</td><td>".$ins->get('familyTree:hasBirthYear')."</td></tr>";  } |

Kode Sumber 5.8 Fungsi Get birth year

***Get* death date**

Fungsi ini digunakan untuk mendapatkan tanggal kematian dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasDeathDate yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.9.

|  |
| --- |
| **if**($ins->get('ihero:hasDeathDate')){  **echo** "<tr>";  **echo** "<td>deathDate</td><td>".**str\_replace**('T00:00:00Z', "", $ins->get('ihero:hasDeathDate'))."</td></tr>";  } |

Kode Sumber 5.9 Fungsi Get death date

***Get* death place**

Fungsi ini digunakan untuk mendapatkan tempat meninggal dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasDeathPlace yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.10.

|  |
| --- |
| **if**($ins->get('ihero:hasDeathPlace')){  **echo** "<tr><td>deathPlace</td><td>";  **foreach** ($ins->all('ihero:hasDeathPlace') **as** $subject) {  **echo** **str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject)."<br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.10 Fungsi Get death place

***Get* death year**

Fungsi ini digunakan untuk mendapatkan tahun kematian dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasDeathYear yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.11.

|  |
| --- |
| **if**($ins->get('familyTree:hasDeathYear')){  **echo** "<tr>";  **echo** "<td>deathYear</td><td>".$ins->get('familyTree:hasDeathYear')."</td></tr>";  } |

Kode Sumber 5.11 Fungsi Get death year

***Get* religion**

Fungsi ini digunakan untuk mendapatkan agama dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasReligion yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.12.

|  |
| --- |
| **if**($ins->get('ihero:hasReligion')){  **echo** "<tr>";  **echo** "<td>religion</td><td>".$ins->get('ihero:hasReligion')."</td></tr>";  } |

Kode Sumber 5.12 Fungsi Get religion

***Get* alternative names**

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

|  |
| --- |
| **if**($ins->get('ihero:alias')){  **echo** "<tr>";  **echo** "<td>alternativeNames</td><td>".$ins->get('ihero:alias')."</td></tr>";  } |

Kode Sumber 5.13 Fungsi Get alternative names

***Get* honorific prefix**

Fungsi ini digunakan untuk mendapatkan gelar nama dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasHonorificPrefix yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.14.

|  |
| --- |
| **if**($ins->get('ihero:hasHonorificPrefix')){  **echo** "<tr><td>honorificPrefixe</td><td>";  **foreach** ($ins->all('ihero:hasHonorificPrefix') **as** $subject) {  **echo** **str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject)."<br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.14 Fungsi Get honorific prefix

***Get* office**

Fungsi ini digunakan untuk mendapatkan pekerjaan/ jabatan dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasOffice yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.15

|  |
| --- |
| **if**($ins->get('ihero:hasOffice')){  **echo** "<tr><td>office</td><td>";  **foreach** ($ins->all('ihero:hasOffice') **as** $subject) {  **echo** **str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject)."<br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.15 Fungsi Get office

### Fungsi Get Relations

Fungsi ini digunakan untuk mengakses informasi data yang merupakan tautan untuk mendapatkan informasi hubungan antara *person* dengan *person*, grup, organisasi, atau *event* begitu juga sebaliknya. Fungsi ini memiliki beberapa sub fungsi berdasarkan kegunaan informasi yang diambil dari ontologi.

***Get* parent**

Fungsi ini digunakan untuk mendapatkan *value* orang tua dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasParent yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.16.

|  |
| --- |
| **if**($ins->get('familyTree:hasParent')){  **echo** "<tr><td>hasParent</td><td>";  **foreach** ($ins->all('familyTree:hasParent') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.16 Fungsi Get parent

***Get* father**

Fungsi ini digunakan untuk mendapatkan *value* ayah dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasFather yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.17.

|  |
| --- |
| **if**($ins->get('familyTree:hasFather')){  **echo** "<tr><td>hasFather</td><td>";  **foreach** ($ins->all('familyTree:hasFather') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.17 Fungsi Get father

***Get* mother**

Fungsi ini digunakan untuk mendapatkan *value* ibu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasMother yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.18.

|  |
| --- |
| **if**($ins->get('familyTree:hasMother')){  **echo** "<tr><td>hasMother</td><td>";  **foreach** ($ins->all('familyTree:hasMother') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.18 Fungsi Get mother

***Get* stepparent**

Fungsi ini digunakan untuk mendapatkan *value* orang tua tiri dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepParent yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.19.

|  |
| --- |
| **if**($ins->get('ihero:hasStepParent')){  **echo** "<tr><td>hasStepParent</td><td>";  **foreach** ($ins->all('ihero:hasStepParent') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.19 Fungsi Get stepparent

***Get* stepfather**

Fungsi ini digunakan untuk mendapatkan *value* ayah tiri dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepFather yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.20.

|  |
| --- |
| **if**($ins->get('ihero:hasStepFather')){  **echo** "<tr><td>hasStepFather</td><td>";  **foreach** ($ins->all('ihero:hasStepFather') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.20 Fungsi Get stepfather

***Get* stepmother**

Fungsi ini digunakan untuk mendapatkan *value* ibu tiri dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepMother yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.21.

|  |
| --- |
| **if**($ins->get('ihero:hasStepMother')){  **echo** "<tr><td>hasStepMother</td><td>";  **foreach** ($ins->all('ihero:hasStepMother') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.21 Fungsi Get stepmother

***Get* child**

Fungsi ini digunakan untuk mendapatkan *value* anak 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 5.22.

|  |
| --- |
| **if**($ins->get('familyTree:hasChild')){  **echo** "<tr><td>hasChild</td><td>";  **foreach** ($ins->all('familyTree:hasChild') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.22 Fungsi Get child

***Get* son**

Fungsi ini digunakan untuk mendapatkan *value* anak laki-laki dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasSon yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.23

|  |
| --- |
| **if**($ins->get('familyTree:hasSon')){  **echo** "<tr><td>hasSon</td><td>";  **foreach** ($ins->all('familyTree:hasSon') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.23 Fungsi Get son

***Get* daughter**

Fungsi ini digunakan untuk mendapatkan *value* anak perempuan dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasDaughter yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.24.

|  |
| --- |
| **if**($ins->get('familyTree:hasDaughter')){  **echo** "<tr><td>hasDaughter</td><td>";  **foreach** ($ins->all('familyTree:hasDaughter') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.24 Fungsi Get daughter

***Get* stepchild**

Fungsi ini digunakan untuk mendapatkan *value* anak tiri dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepChild yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.25.

|  |
| --- |
| **if**($ins->get('ihero:hasStepChild')){  **echo** "<tr><td>hasStepChild</td><td>";  **foreach** ($ins->all('ihero:hasStepChild') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.25 Fungsi Get stepchild

***Get* stepson**

Fungsi ini digunakan untuk mendapatkan *value* anak tiri laki-laki dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepSon yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.26.

|  |
| --- |
| **if**($ins->get('ihero:hasStepSon')){  **echo** "<tr><td>hasStepSon</td><td>";  **foreach** ($ins->all('ihero:hasStepSon') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.26 Fungsi Get stepson

***Get* step daughter**

Fungsi ini digunakan untuk mendapatkan *value* anak tiri perempuandari entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepDaughter yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.27.

|  |
| --- |
| **if**($ins->get('ihero:hasStepDaughter')){  **echo** "<tr><td>hasStepDaughter</td><td>";  **foreach** ($ins->all('ihero:hasStepDaughter') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.27 Fungsi Get stepdaughter

***Get* sibling**

Fungsi ini digunakan untuk mendapatkan *value* saudara dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasSibling yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.28.

|  |
| --- |
| **if**($ins->get('ihero:hasSibling')){  **echo** "<tr><td>hasSibling</td><td>";  **foreach** ($ins->all('ihero:hasSibling') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.28 Fungsi Get sibling

***Get* brother**

Fungsi ini digunakan untuk mendapatkan *value* saudara laki-laki dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasBrother yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.29.

|  |
| --- |
| **if**($ins->get('familyTree:hasBrother')){  **echo** "<tr><td>hasBrother</td><td>";  **foreach** ($ins->all('familyTree:hasBrother') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.29 Fungsi Get brother

***Get* sister**

Fungsi ini digunakan untuk mendapatkan *value* anak perempuan dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasSister yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.30.

|  |
| --- |
| **if**($ins->get('familyTree:hasSister')){  **echo** "<tr><td>hasSister</td><td>";  **foreach** ($ins->all('familyTree:hasSister') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.30 Fungsi Get sister

***Get* stepsibling**

Fungsi ini digunakan untuk mendapatkan *value* saudara tiri dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepSibling yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.31.

|  |
| --- |
| **if**($ins->get('ihero:hasStepSibling')){  **echo** "<tr><td>hasStepSibling</td><td>";  **foreach** ($ins->all('ihero:hasStepSibling') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>"; } |

Kode Sumber 5.31 Fungsi Get stepsibling

***Get* stepbrother**

Fungsi ini digunakan untuk mendapatkan *value* saudara tiri laki-laki dari entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepBrother yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.32.

|  |
| --- |
| **if**($ins->get('ihero:hasStepBrother')){  **echo** "<tr><td>hasStepBrother</td><td>";  **foreach** ($ins->all('ihero:hasStepBrother') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>"; } |

Kode Sumber 5.32 Fungsi Get stepbrother

***Get* stepsister**

Fungsi ini digunakan untuk mendapatkan *value* saudara tiri perempuandari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasStepSister yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.33.

|  |
| --- |
| **if**($ins->get('ihero:hasStepSister')){  **echo** "<tr><td>hasStepSister</td><td>";  **foreach** ($ins->all('ihero:hasStepSister') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.33 Fungsi Get stepsister

***Get* party**

Fungsi ini digunakan untuk mendapatkan *value* partai dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasParty yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.34.

|  |
| --- |
| **if**($ins->get('ihero:hasParty')){  **echo** "<tr><td>party</td><td>";  **foreach** ($ins->all('ihero:hasParty') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.34 Fungsi Get party

***Get* prime minister**

Fungsi ini digunakan untuk mendapatkan *value* perdana menteri dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasPrimeMinister yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.35.

|  |
| --- |
| **if**($ins->get('ihero:hasPrimeMinister')){  **echo** "<tr><td>primeMinister</td><td>";  **foreach** ($ins->all('ihero:hasPrimeMinister') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.35 Fungsi Get prime minister

***Get* spouse**

Fungsi ini digunakan untuk mendapatkan *value* suami/ istridari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isSpouseOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.36.

|  |
| --- |
| **if**($ins->get('familyTree:isSpouseOf')){  **echo** "<tr><td>spouse</td><td>";  **foreach** ($ins->all('familyTree:isSpouseOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.36 Fungsi Get spouse

***Get* successor**

Fungsi ini digunakan untuk mendapatkan *value* penerus dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasSuccessor yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.37.

|  |
| --- |
| **if**($ins->get('ihero:hasSuccessor')){  **echo** "<tr><td>successor</td><td>";  **foreach** ($ins->all('ihero:hasSuccessor') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.37 Fungsi Get successor

***Get* vice president**

Fungsi ini digunakan untuk mendapatkan *value* wakil presiden dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasVicePresident yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.38.

|  |
| --- |
| **if**($ins->get('ihero:hasVicePresident')){  **echo** "<tr><td>vicePresident</td><td>";  **foreach** ($ins->all('ihero:hasVicePresident') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.38 Fungsi Get vice president

***Get* predecessor**

Fungsi ini digunakan untuk mendapatkan *value* pendahulu dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasPredecessor yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.39.

|  |
| --- |
| **if**($ins->get('ihero:hasPredecessor')){  **echo** "<tr><td>predecessor</td><td>";  **foreach** ($ins->all('ihero:hasPredecessor') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.39 Fungsi Get predecessor

***Get* is child of**

Fungsi ini digunakan untuk mendapatkan *value* adalah anak dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isChildOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.40.

|  |
| --- |
| **if**($ins->get('familyTree:isChildOf')){  **echo** "<tr><td>isChildOf</td><td>";  **foreach** ($ins->all('familyTree:isChildOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.40 Fungsi Get is child of

***Get* is commander of**

Fungsi ini digunakan untuk mendapatkan *value* adalah komandan dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isCommanderOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.41.

|  |
| --- |
| **if**($ins->get('ihero:isCommanderOf')){  **echo** "<tr><td>isCommanderOf</td><td>";  **foreach** ($ins->all('ihero:isCommanderOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.41 Fungsi Get is commander of

***Get* is influenced of**

Fungsi ini digunakan untuk mendapatkan *value* adalah orang berpengaruh dari entitas yang dipilih. *Value* yang diambil berasal dari properti isInfluencedOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.42.

|  |
| --- |
| **if**($ins->get('ihero:isInfluencedOf')){  **echo** "<tr><td>isInfluencedOf</td><td>";  **foreach** ($ins->all('ihero:isInfluencedOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>"; } |

Kode Sumber 5.42 Fungsi Get is influenced of

***Get* is person of**

Fungsi ini digunakan untuk mendapatkan *value* adalah orang yang terlibat dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isPersonOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.43.

|  |
| --- |
| **if**($ins->get('ihero:isPersonOf')){  **echo** "<tr><td>isPersonOf</td><td>";  **foreach** ($ins->all('ihero:isPersonOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.43 Fungsi Get is person of

***Get* is predecessor of**

Fungsi ini digunakan untuk mendapatkan *value* adalah pendahulu dari entitas yang dipilih. *Value* yang diambil berasal dari properti isPredecessorOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.44.

|  |
| --- |
| **if**($ins->get('ihero:isPredecessorOf')){  **echo** "<tr><td>isPredecessorOf</td><td>";  **foreach** ($ins->all('ihero:isPredecessorOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>"; } |

Kode Sumber 5.44 Fungsi Get is predecessor of

***Get* is president of**

Fungsi ini digunakan untuk mendapatkan *value* adalah presiden dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isPresidentOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.45.

|  |
| --- |
| **if**($ins->get('ihero:isPresidentOf')){  **echo** "<tr><td>isPresidentOf</td><td>";  **foreach** ($ins->all('ihero:isPresidentOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.45 Fungsi Get is president of

***Get* is spouse of**

Fungsi ini digunakan untuk mendapatkan *value* adalah suami/ istri 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 5.46.

|  |
| --- |
| **if**($ins->get('familyTree:isSpouseOf')){  **echo** "<tr><td>isSpouseOf</td><td>";  **foreach** ($ins->all('familyTree:isSpouseOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.46 Fungsi Get is spouse of

***Get* is successor of**

Fungsi ini digunakan untuk mendapatkan *value* adalah penerus dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isSuccessorOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.47.

|  |
| --- |
| **if**($ins->get('ihero:isSuccessorOf')){  **echo** "<tr><td>isSuccessorOf</td><td>";  **foreach** ($ins->all('ihero:isSuccessorOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.47 Fungsi Get is successor of

***Get* isGovernmentHeadOf**

Fungsi ini digunakan untuk mendapatkan *value* kepala pemerintahandari entitas yang dipilih. *Value* yang diambil berasal dari properti isGovernmentHeadOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.48.

|  |
| --- |
| **if**($ins->get('ihero:isGovernmentHeadOf')){  **echo** "<tr><td>isGovernmentHeadOf</td><td>";  **foreach** ($ins->all('ihero:isGovernmentHeadOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.48 Fungsi Get isGovernmentHeadOf

***Get* isInauguralOf**

Fungsi ini digunakan untuk mendapatkan *value* adalah pelantikan dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isInaugural yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.49.

|  |
| --- |
| **if**($ins->get('ihero:isInauguralOf')){  **echo** "<tr><td>isInauguralOf</td><td>";  **foreach** ($ins->all('ihero:isInauguralOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.49 Fungsi Get isInauguralOf

***Get* is leader of**

Fungsi ini digunakan untuk mendapatkan *value* adalah pemimpin dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti isLeaderOf yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.50.

|  |
| --- |
| **if**($ins->get('ihero:isLeaderOf')){  **echo** "<tr><td>isLeaderOf</td><td>";  **foreach** ($ins->all('ihero:isLeaderOf') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

Kode Sumber 5.50 Fungsi Get is leader of

***Get* subject**

Fungsi ini digunakan untuk mendapatkan *value* subjek dari sebuah entitas yang dipilih. *Value* yang diambil berasal dari properti hasSubject yang melekat pada entitas. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 5.51.

|  |
| --- |
| **if**($ins->get('ihero:hasSubject')){  **echo** "<tr><td>Subject</td><td>";  **foreach** ($ins->all('ihero:hasSubject') **as** $subject) {  **echo** '<a href="?entity='.urlencode($subject).'">'.**str\_replace**('http://www.semanticweb.org/asus/ontologies/2016/3/ihero/', "", $subject->get('familyTree:hasName'))."</a><br/>";  }  **echo** "</td></tr>";  } |

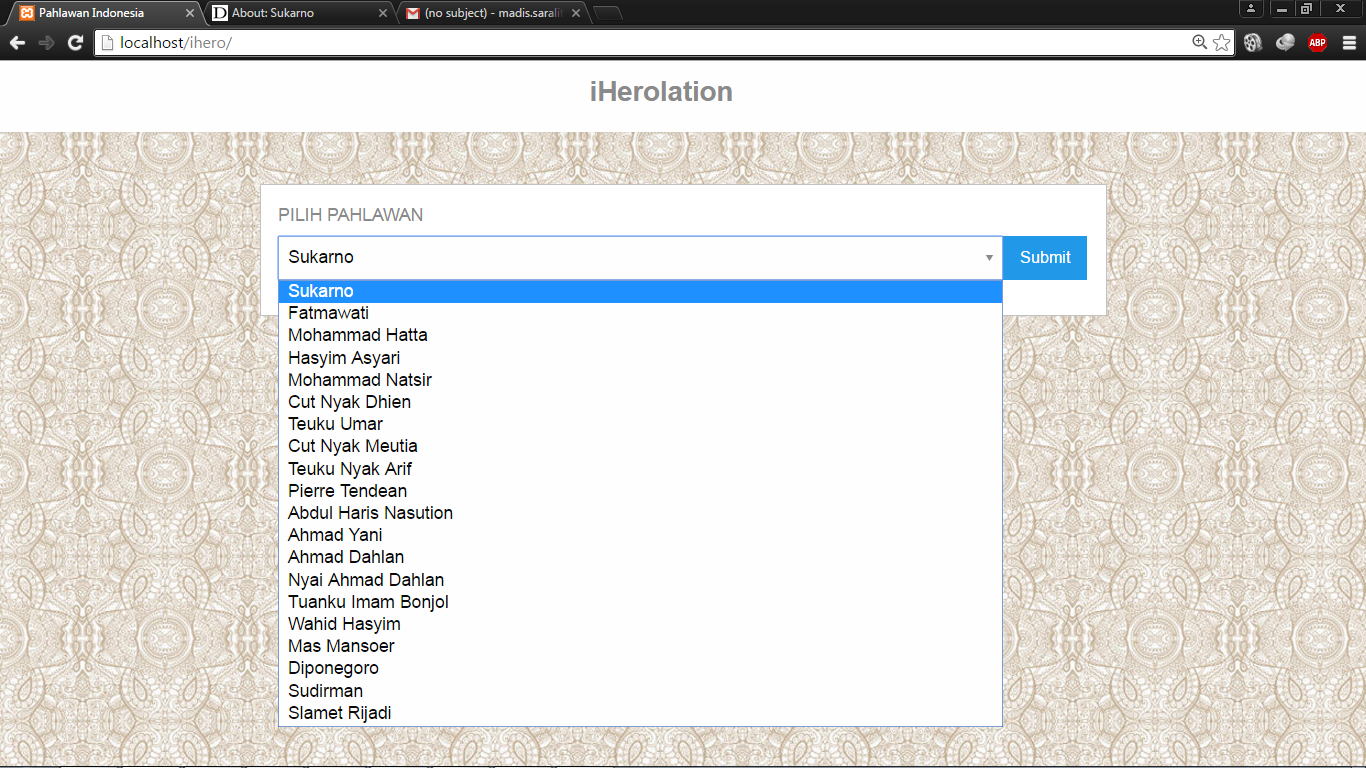
Kode Sumber 5.51 Fungsi Get subject

## Implementasi Antarmuka Pengguna

Implementasi tampilan antarmuka pengguna pada *browser* Google Chrome dilakukan dengan menggunakan dukungan aplikasi XAMPP. XAMPP berfungsi untuk menghubungkan data yang terdapat pada file RDF ontologi dengan *browser*. Berikut ini akan dijelaskan mengenai implementasi tampilan antarmuka pengguna yang terdapat padaiHerolation.

### Implementasi Tampilan Halaman Utama

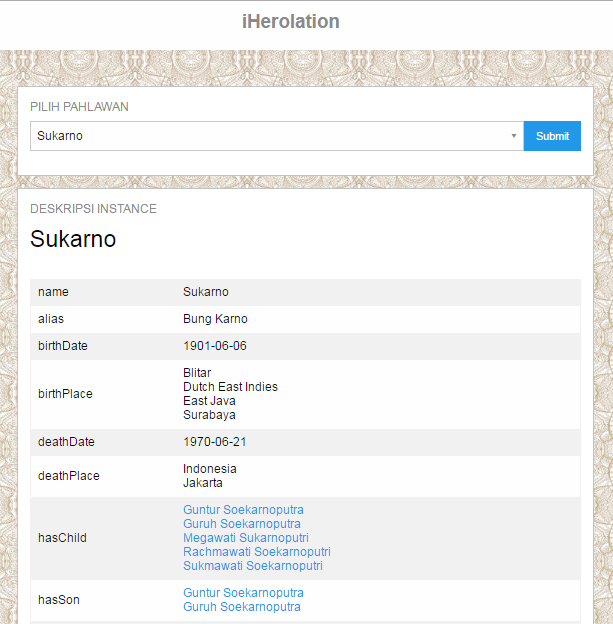
Halaman ini merupakan implementasi halaman utama dari rancangan antarmuka yang telah dijelaskan pada Subbab 4.2. Halaman utama hanya menampilkan kolom *dropdown select* yang dapat digunakan oleh pengguna untuk memilih tokoh. Daftar entitas tokoh yang ditampilkan hanya tokoh utama yang digunakan dalam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



Gambar 5.1.1 Implementasi Antarmuka Halaman Utama

### Implementasi Tampilan Halaman Informasi

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 5.1.2 Implementasi Antarmuka Halaman Informasi

# 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 ontologi dilakukan dengan mengetes fungsionalitas karakteristik dan SWRL yang digunakan. Selanjutnya pengujian perbandingan data dilakukan dengan membandingkan data asli DBpedia dengan data yang dihasilkan dari pohon keluarga Family Tree App. Pengujian Ontologi

Pengujian ontologi merupakan tahap uji kevalidan ontologi yang telah dibangun sebagai dasar proses pencarian relasi antar tokoh sejarah Indonesia. Pengujian dilakukan secara manual dengan mengecek kebenaran fakta baru yang muncul sebagai tolok ukur keberhasilan pengujian.

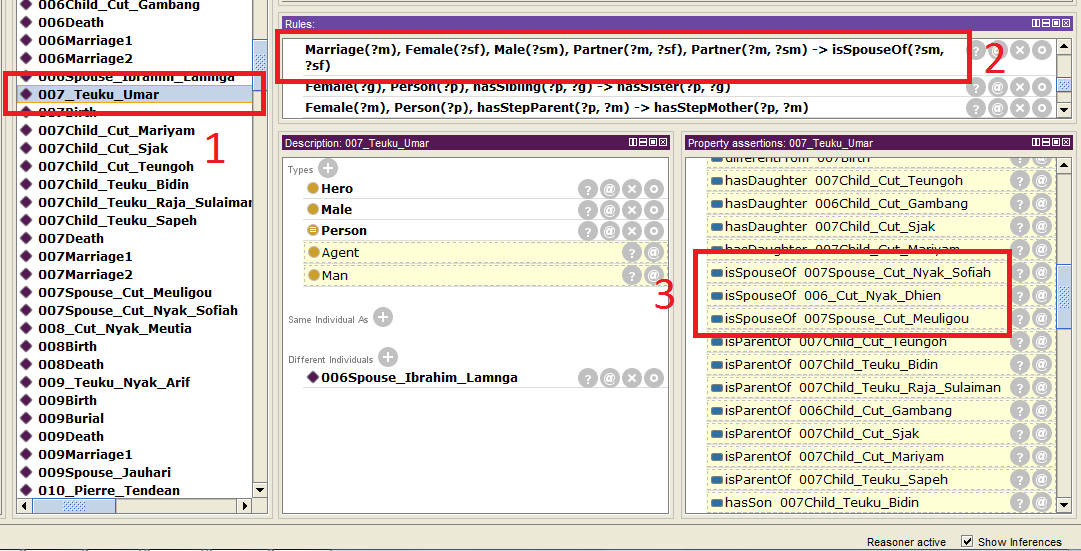
#### Pengujian Kevalidan Relasi isSpouseOf

Pada tahap pengujian relasi isSpouseOf, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian Kevalidan Relasi isSpouseOf

|  |  |
| --- | --- |
| ID | TA-UJ.ON0001 |
| Nama | Pengujian kevalidan relasi isSpouseOf |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi isSpouseOf |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi isSpouseOf dan tidak. |
| Kondisi Awal | Individu yang memiliki suami/ istri dapat dilihat dari properti objek yang dimiliki oleh *instance* bertipe Marriage. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi isSpouseOf didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi isSpouseOf dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi isSpouseOf didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi isSpouseOf dapat dilihat pada Gambar 6.1. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi isSpouseOf dengan individu lain memunculkan data *inference* isSpouseOf sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi isSpouseOf telah berhasil.



Gambar 6.2.1 Uji Coba Relasi isSpouseOf

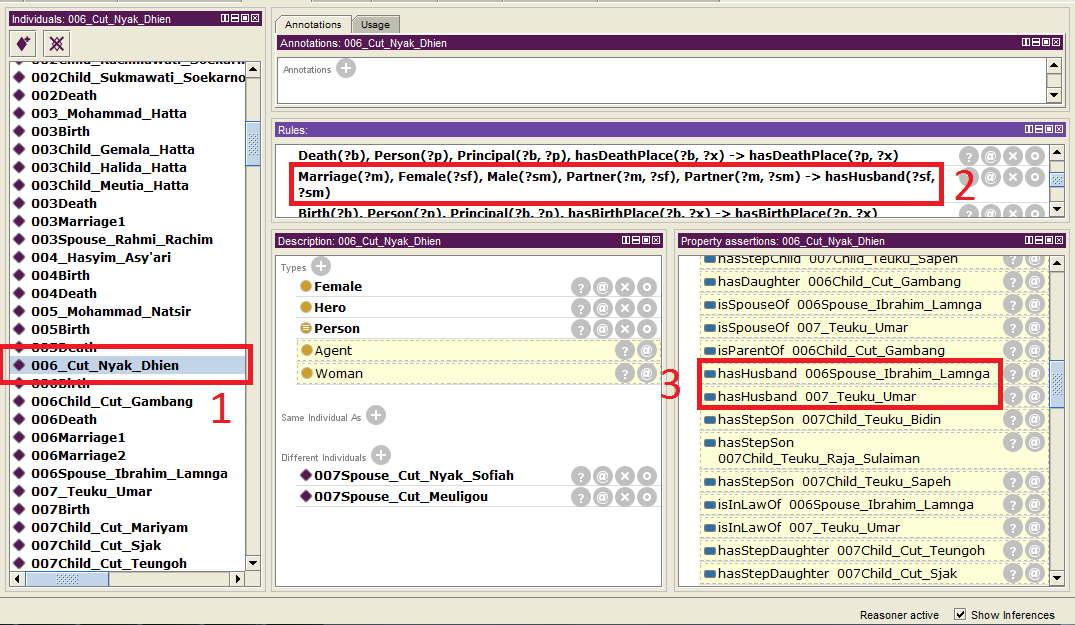
#### Pengujian Kevalidan Relasi hasHusband

Pada tahap pengujian relasi hasHusband, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.2.

Tabel 6.2 Pengujian Kevalidan Relasi hasHusband

|  |  |
| --- | --- |
| ID | TA-UJ.ON0002 |
| Nama | Pengujian kevalidan relasi hasHusband |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasHusband |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasHusband dan tidak. |
| Kondisi Awal | Individu yang memiliki suami/ istri dapat dilihat dari properti objek yang dimiliki oleh *instance* bertipe Marriage. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasHusband didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasHusband dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasHusband didapatkan hasil *inference* data yang benar. |

Hasil pengujian kevalidan relasi hasHusband dapat dilihat pada Gambar 6.2. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasHusband dengan individu lain memunculkan data *inference* hasHusband sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasHusband telah berhasil.



Gambar 6.2.2 Uji Coba Relasi hasHusband

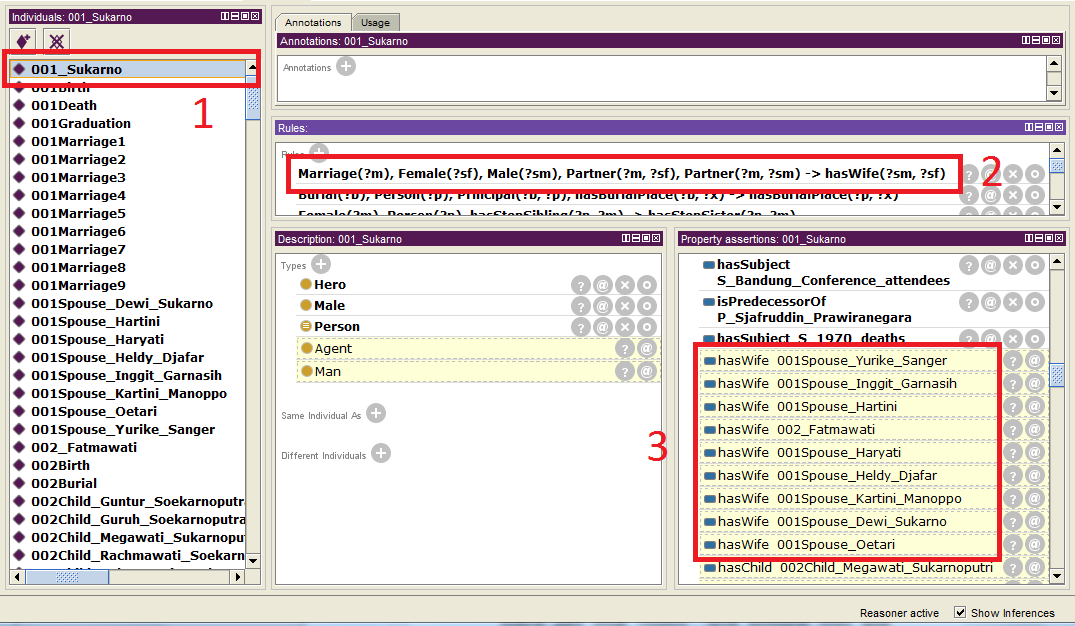
#### Pengujian Kevalidan Relasi hasWife

Pada tahap pengujian relasi hasWife, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.3.

Tabel 6.3 Pengujian Kevalidan Relasi hasWife

|  |  |
| --- | --- |
| ID | TA-UJ.ON0003 |
| Nama | Pengujian kevalidan relasi hasWife |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasWife |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasWife dan tidak. |
| Kondisi Awal | Individu yang memiliki suami/ istri dapat dilihat dari properti objek yang dimiliki oleh *instance* bertipe Marriage. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasWife didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasWife dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasWife didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasWife dapat dilihat pada Gambar 6.3. Pada gambar dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasWife dengan individu lain memunculkan data inference hasWife sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasWife telah berhasil.



Gambar 6.2.3 Uji Coba Relasi hasWife

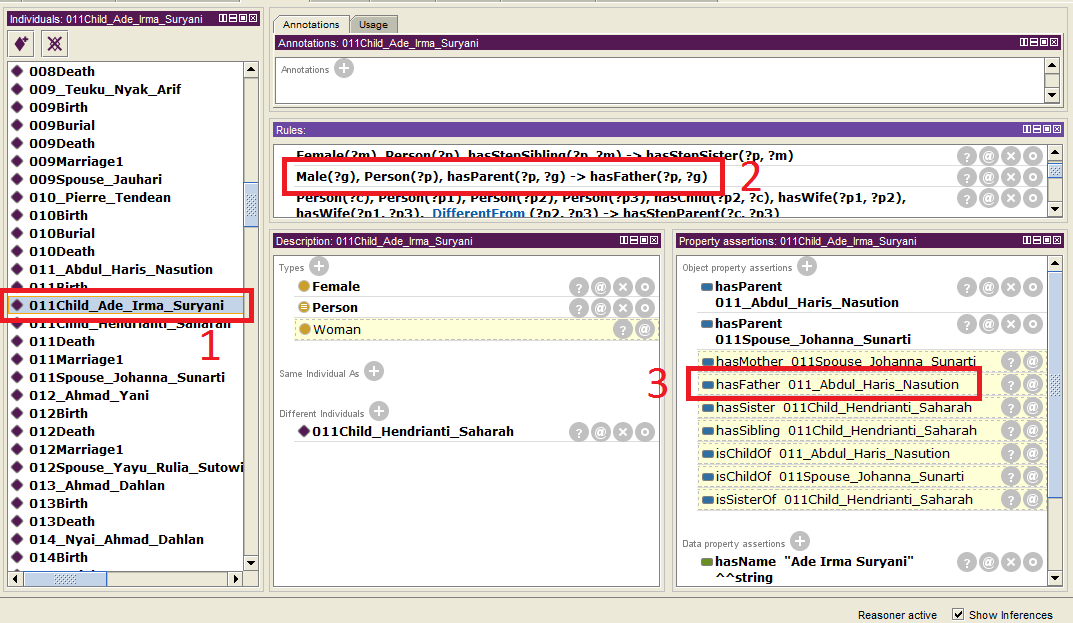
#### Pengujian Kevalidan Relasi hasFather

Pada tahap pengujian relasi hasFather, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.4.

Tabel 6.4 Pengujian Kevalidan Relasi hasFather

|  |  |
| --- | --- |
| ID | TA-UJ.ON0004 |
| Nama | Pengujian kevalidan relasi hasFather |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasFather |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasFather dan tidak. |
| Kondisi Awal | Individu yang bertipe Person dengan kode 0XXChild memiliki properti objek hasParent. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasFather didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasFather dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasFather didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasFather dapat dilihat pada Gambar 6.4. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasFather dengan individu lain memunculkan data *inference* hasFather sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasFather telah berhasil.



Gambar 6.2.4 Uji Coba Relasi hasFather

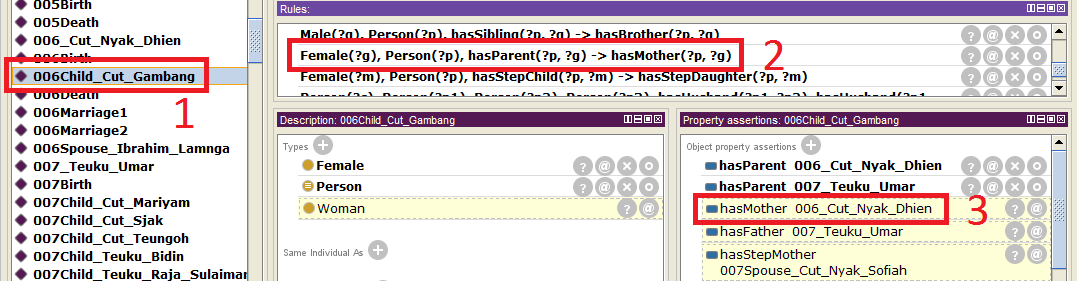
#### Pengujian Kevalidan Relasi hasMother

Pada tahap pengujian relasi hasMother, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.5.

Tabel 6.5 Pengujian Kevalidan Relasi hasMother

|  |  |
| --- | --- |
| ID | TA-UJ.ON0005 |
| Nama | Pengujian kevalidan relasi hasMother |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasMother |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasMother dan tidak. |
| Kondisi Awal | Individu yang bertipe Person dengan kode 0XXChild memiliki properti objek hasParent. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasMother didapatkan hasil *inference* data yang benar |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasMother dengan individu yang bersangkutan.. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasMother didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasMother dapat dilihat pada Gambar 6.5. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasMother dengan individu lain memunculkan data inference hasMother sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasMother telah berhasil.



Gambar 6.2.5 Uji Coba Relasi hasMother

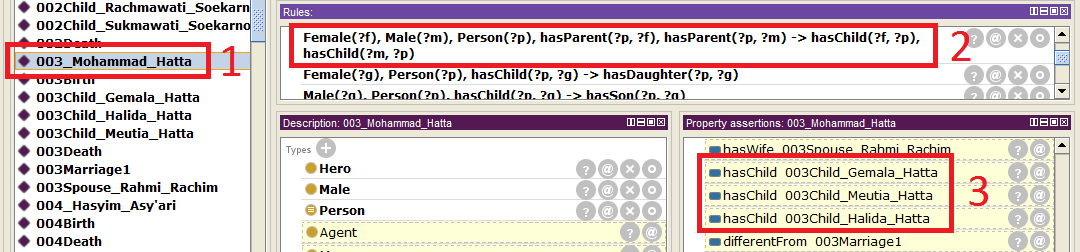
#### Pengujian Kevalidan Relasi hasChild

Pada tahap pengujian relasi hasChild, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.6.

Tabel 6.6 Pengujian Kevalidan Relasi hasChild

|  |  |
| --- | --- |
| ID | TA-UJ.ON0006 |
| Nama | Pengujian kevalidan relasi hasChild |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasChild |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasChild dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki anak belum memiliki properti hasChild. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasChild didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasChild dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasChild didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasChild dapat dilihat pada Gambar 6.6. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasChild dengan individu lain memunculkan data *inference* hasChild sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasChild telah berhasil.



Gambar 6.2.6 Uji Coba Relasi hasChild

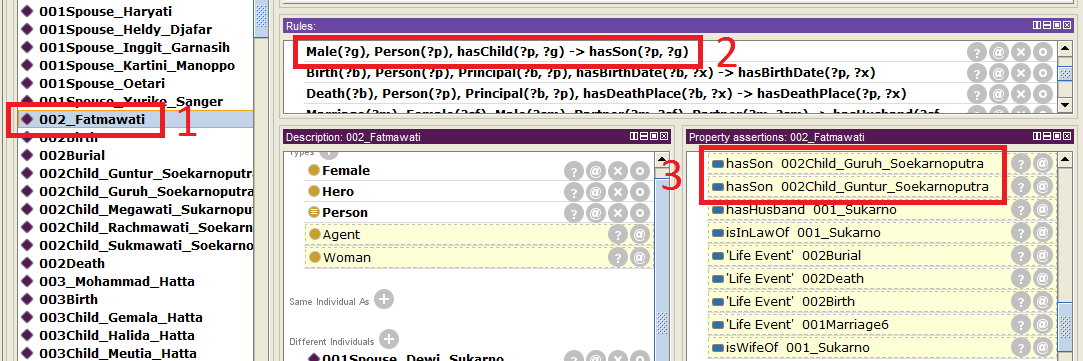
#### Pengujian Kevalidan Relasi hasSon

Pada tahap pengujian relasi hasSon, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.7.

Tabel 6.7 Pengujian Kevalidan Relasi hasSon

|  |  |
| --- | --- |
| ID | TA-UJ.ON0007 |
| Nama | Pengujian kevalidan relasi hasSon |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasSon |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasSon dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki anak laki-laki belum memiliki properti hasSon. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasSon didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasSon dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasSon didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasSon dapat dilihat pada Gambar 6.7. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasSon dengan individu lain memunculkan data *inference* hasSon sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasSon telah berhasil.



Gambar 6.2.7 Uji Coba Relasi hasSon

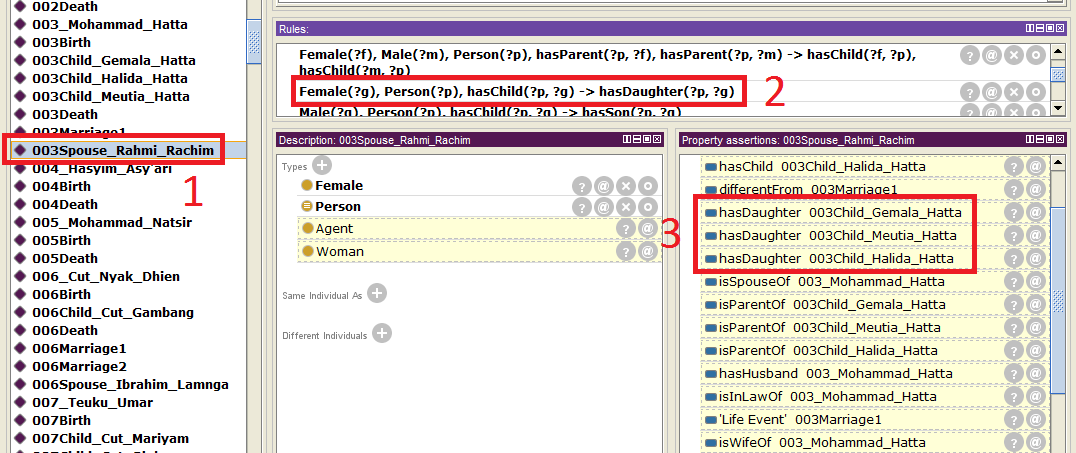
#### Pengujian Kevalidan Relasi hasDaughter

Pada tahap pengujian relasi hasDaughter, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.8.

Tabel 6.8 Pengujian Kevalidan Relasi hasDaughter

|  |  |
| --- | --- |
| ID | TA-UJ.ON0008 |
| Nama | Pengujian kevalidan relasi hasDaughter |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasDaughter |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasDaughter dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki anak perempuan belum memiliki properti hasDaughter. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasDaughter didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasDaughter dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasDaughter didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasDaughter dapat dilihat pada Gambar 6.8. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasDaughter dengan individu lain memunculkan data *inference* hasDaughter sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasDaughter telah berhasil.



Gambar 6.2.8 Uji Coba Relasi hasDaughter

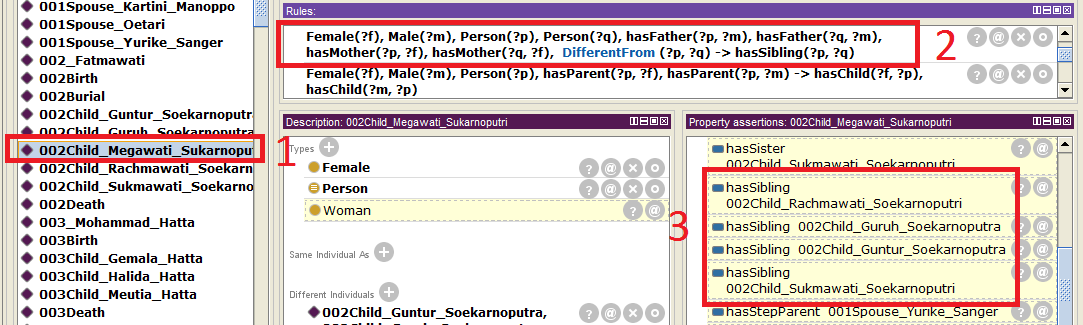
#### Pengujian Kevalidan Relasi hasSibling

Pada tahap pengujian relasi hasSibling, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.9.

Tabel 6.9 Pengujian Kevalidan Relasi hasSibling

|  |  |
| --- | --- |
| ID | TA-UJ.ON0009 |
| Nama | Pengujian kevalidan relasi hasSibling |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasSibling |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasSibling dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara kandung belum memiliki properti hasSibling. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasSibling didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasSibling dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasSibling didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasSibling dapat dilihat pada Gambar 6.9. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasSibling dengan individu lain memunculkan data *inference* hasSibling sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasSibling telah berhasil.



Gambar 6.2.9 Uji Coba Relasi hasSibling

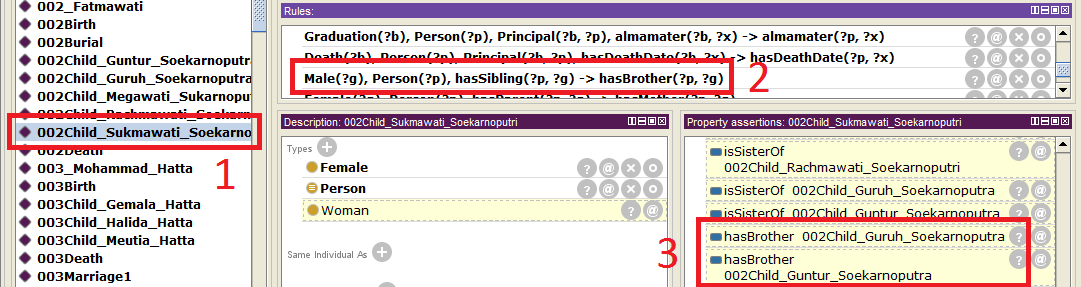
#### Pengujian Kevalidan Relasi hasBrother

Pada tahap pengujian relasi hasBrother, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.10.

Tabel 6.10 Pengujian Kevalidan Relasi hasBrother

|  |  |
| --- | --- |
| ID | TA-UJ.ON0010 |
| Nama | Pengujian kevalidan relasi hasBrother |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasBrother |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasBrother dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara kandung laki-laki belum memiliki properti hasBrother. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasBrother didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasBrother dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasBrother didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasBrother dapat dilihat pada Gambar 6.10. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasBrother dengan individu lain memunculkan data *inference* hasBrother sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasBrother telah berhasil.



Gambar 6.2.10 Uji Coba Relasi hasBrother

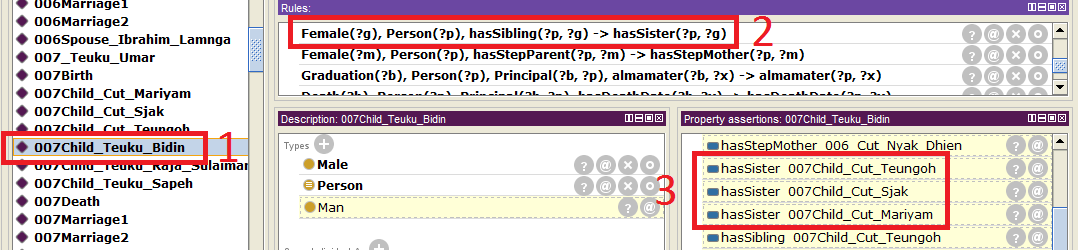
#### Pengujian Kevalidan Relasi hasSister

Pada tahap pengujian relasi hasSister, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.11.

Tabel 6.11 Pengujian Kevalidan Relasi hasSister

|  |  |
| --- | --- |
| ID | TA-UJ.ON0011 |
| Nama | Pengujian kevalidan relasi hasSister |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasSister |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasSister dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara kandung perempuan belum memiliki properti hasSister. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasSister didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasSister dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasSister didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasSister dapat dilihat pada Gambar 6.11. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasSister dengan individu lain memunculkan data *inference* hasSister sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasSister telah berhasil.



Gambar 6.2.11 Uji Coba Relasi hasSister

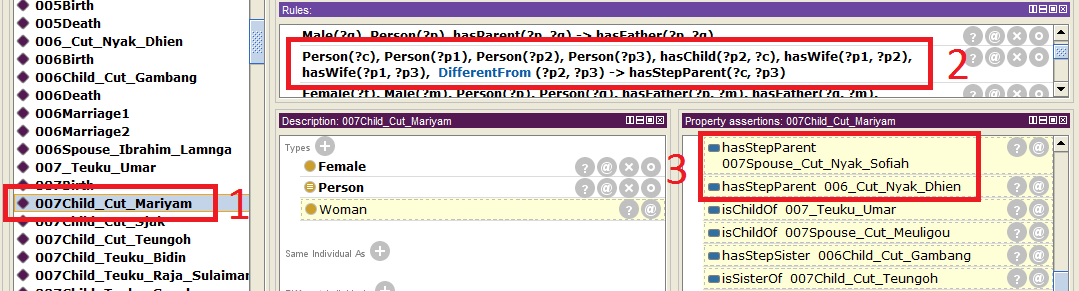
#### Pengujian Kevalidan Relasi hasStepParent

Pada tahap pengujian relasi hasStepParent, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.12.

Tabel 6.12 Pengujian Kevalidan Relasi hasStepParent

|  |  |
| --- | --- |
| ID | TA-UJ.ON0012 |
| Nama | Pengujian kevalidan relasi hasStepParent |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepParent |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepParent dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki orang tua tiri belum memiliki properti hasStepParent. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepParent didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepParent dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepParent didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepParent dapat dilihat pada Gambar 6.12. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepParent dengan individu lain memunculkan data *inference* hasStepParent sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepParent telah berhasil.



Gambar 6.2.12 Uji Coba Relasi hasStepParent

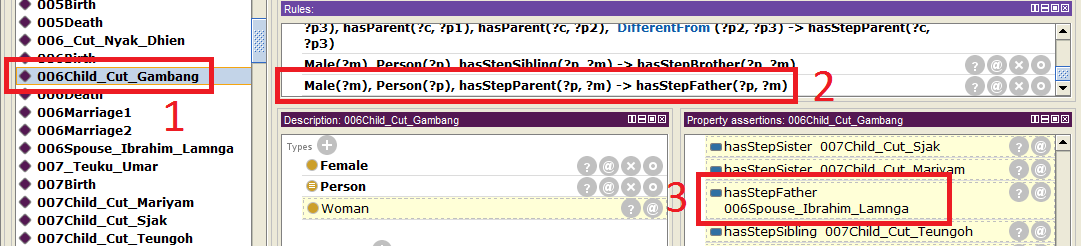
#### Pengujian Kevalidan Relasi hasStepFather

Pada tahap pengujian relasi hasStepFather, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.13.

Tabel 6.13 Pengujian Kevalidan Relasi hasStepFather

|  |  |
| --- | --- |
| ID | TA-UJ.ON0013 |
| Nama | Pengujian kevalidan relasi hasStepFather |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepFather |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepFather dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki ayah tiri belum memiliki properti hasStepFather. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepFather didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepFather dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepFather didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepFather dapat dilihat pada Gambar 6.13. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepFather dengan individu lain memunculkan data *inference* hasStepFather sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepFather telah berhasil.



Gambar 6.2.13 Uji Coba Relasi hasStepFather

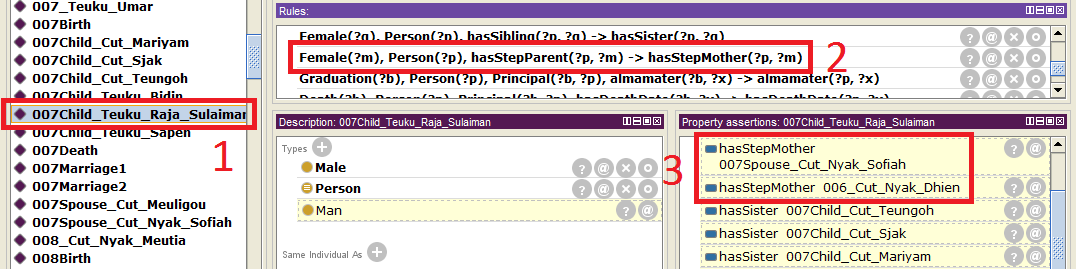
#### Pengujian Kevalidan Relasi hasStepMother

Pada tahap pengujian relasi hasStepMother, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.14.

Tabel 6.14 Pengujian Kevalidan Relasi hasStepMother

|  |  |
| --- | --- |
| ID | TA-UJ.ON0014 |
| Nama | Pengujian kevalidan relasi hasStepMother |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepMother |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepMother dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki ibu tiri belum memiliki properti hasStepMother. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepMother didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepMother dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepMother didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepMother dapat dilihat pada Gambar 6.14. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepMother dengan individu lain memunculkan data *inference* hasStepMother sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepMother telah berhasil.



Gambar 6.2.14 Uji Coba Relasi hasStepMother

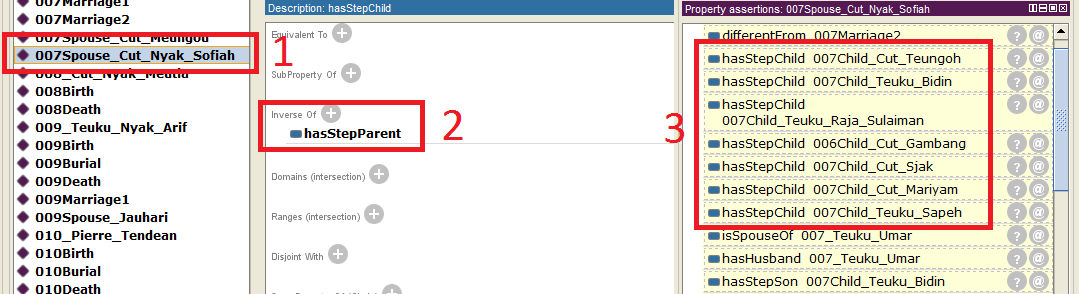
#### Pengujian Kevalidan Relasi hasStepChild

Pada tahap pengujian relasi hasStepChild, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah karakteristik yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.16.

Tabel 6.15 Pengujian Kevalidan Relasi hasStepChild

|  |  |
| --- | --- |
| ID | TA-UJ.ON0015 |
| Nama | Pengujian kevalidan relasi hasStepChild |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepChild |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepChild dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki anak tiri belum memiliki properti hasStepChild. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepChild didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepChild dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepChild didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepChild dapat dilihat pada Gambar 6.15. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepChild dengan individu lain memunculkan data *inference* hasStepChild sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan karakteristik yang dipakai sebagai uji coba. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepChild telah berhasil.



Gambar 6.2.15 Uji Coba Relasi hasStepChild

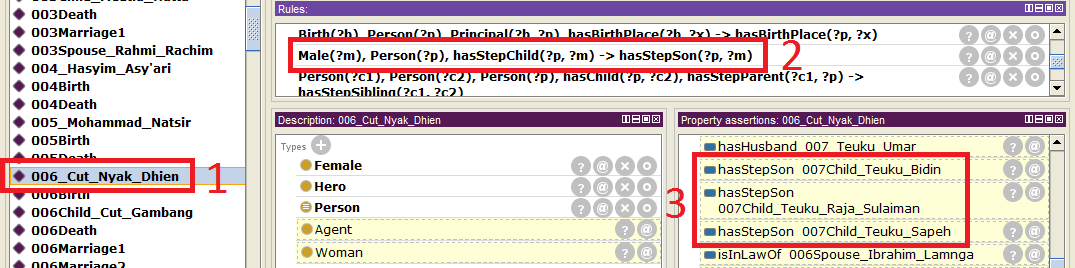
#### Pengujian Kevalidan Relasi hasStepSon

Pada tahap pengujian relasi hasStepSon, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.16.

Tabel 6.16 Pengujian Kevalidan Relasi hasStepSon

|  |  |
| --- | --- |
| ID | TA-UJ.ON0015 |
| Nama | Pengujian kevalidan relasi hasStepSon |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepSon |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepSon dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki anak tiri laki-laki belum memiliki properti hasStepSon. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepSon didapatkan hasil *inference* data yang benar |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepSon dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepSon didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepSon dapat dilihat pada Gambar 6.16. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepSon dengan individu lain memunculkan data *inference* hasStepSon sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepSon telah berhasil.



Gambar 6.2.16 Uji Coba Relasi hasStepSon

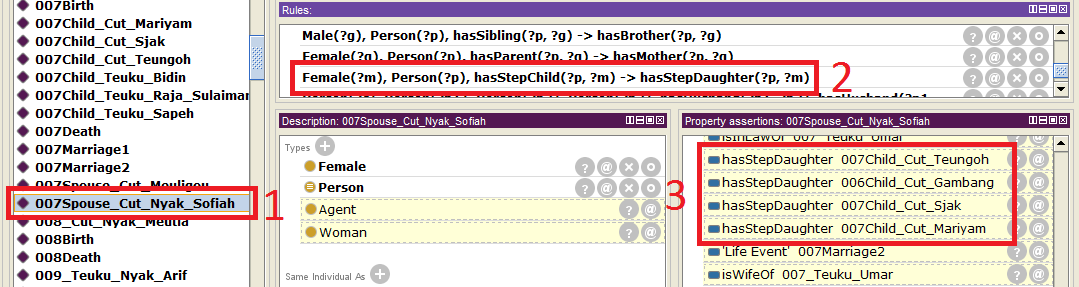
#### Pengujian Kevalidan Relasi hasStepDaughter

Pada tahap pengujian relasi hasStepDaughter, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.17.

Tabel 6.17 Pengujian Kevalidan Relasi hasStepDaughter

|  |  |
| --- | --- |
| ID | TA-UJ.ON0016 |
| Nama | Pengujian kevalidan relasi hasStepDaughter |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepDaughter |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepDaughter dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki anak tiri perempuan belum memiliki properti hasStepDaughter. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepDaughter didapatkan hasil *inference* data yang benar |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepDaughter dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepDaughter didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepDaughter dapat dilihat pada Gambar 6.17. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepDaughter dengan individu lain memunculkan data *inference* hasStepDaughter sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepDaughter telah berhasil.



Gambar 6.2.17 Uji Coba Relasi hasStepDaughter

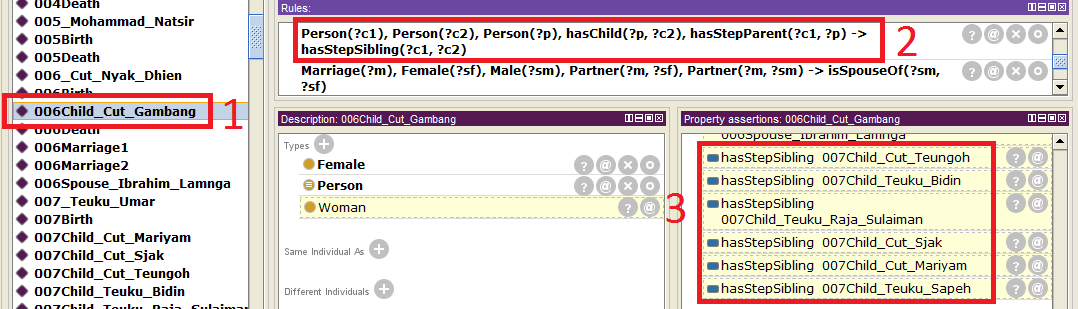
#### Pengujian Kevalidan Relasi hasStepSibling

Pada tahap pengujian relasi hasStepSibling, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.18.

Tabel 6.18 Pengujian Kevalidan Relasi hasStepSibling

|  |  |
| --- | --- |
| ID | TA-UJ.ON0017 |
| Nama | Pengujian kevalidan relasi hasStepSibling |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepSibling |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepSibling dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara tiri belum memiliki properti hasStepSibling. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepSibling didapatkan hasil *inference* data yang benar |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepSibling dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepSibling didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepSibling dapat dilihat pada Gambar 6.18. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepSibling dengan individu lain memunculkan data *inference* hasStepSibling sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepSibling telah berhasil.



Gambar 6.2.18 Uji Coba Relasi hasStepSibling

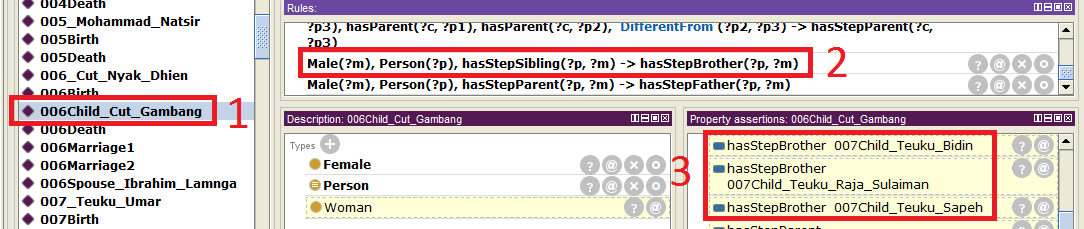
#### Pengujian Kevalidan Relasi hasStepBrother

Pada tahap pengujian relasi hasStepBrother, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.19.

Tabel 6.19 Pengujian Kevalidan Relasi hasStepBrother

|  |  |
| --- | --- |
| ID | TA-UJ.ON0018 |
| Nama | Pengujian kevalidan relasi hasStepBrother |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepBrother |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepBrother dan tidak. |
| Kondisi Awal | Individu tipe Person yang seharusnya memiliki saudara tiri laki-laki belum memiliki properti hasStepBrother. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepBrother didapatkan hasil *inference* data yang benar |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepBrother dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepBrother didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepBrother dapat dilihat pada Gambar 6.19. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepBrother dengan individu lain memunculkan data *inference* hasStepBrother sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepBrother telah berhasil.



Gambar 6.2.19 Uji Coba Relasi hasStepBrother

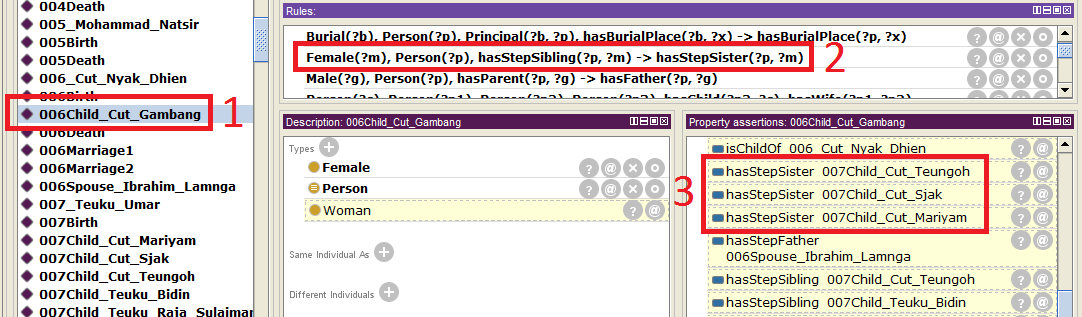
#### Pengujian Kevalidan Relasi hasStepSister

Pada tahap pengujian relasi hasStepSister, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.20.

Tabel 6.20 Pengujian Kevalidan Relasi hasStepSister

|  |  |
| --- | --- |
| ID | TA-UJ.ON0020 |
| Nama | Pengujian kevalidan relasi hasStepSister |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi hasStepSister |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi hasStepSister dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara tiri perempuan belum memiliki properti hasStepSister. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi hasStepSister didapatkan hasil *inference* data yang benar |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi hasStepSister dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi hasStepSister didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi hasStepSister dapat dilihat pada Gambar 6.20. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi hasStepSister dengan individu lain memunculkan data *inference* hasStepSister sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepSister telah berhasil.



Gambar 6.2.20 Uji Coba Relasi hasStepSister

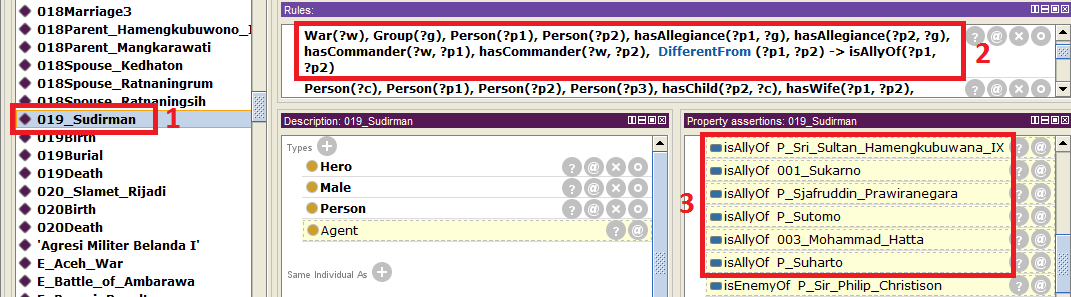
#### Pengujian Kevalidan Relasi isAllyOf

Pada tahap pengujian relasi isAllyOf, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.21.

Tabel 6.21 Pengujian Kevalidan Relasi isAllyOf

|  |  |
| --- | --- |
| ID | TA-UJ.ON0021 |
| Nama | Pengujian kevalidan relasi isAllyOf |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi isAllyOf |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi isAllyOf dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara tiri perempuan belum memiliki properti isAllyOf. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi isAllyOf didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi isAllyOf dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi isAllyOf didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi isAllyOf dapat dilihat pada Gambar 6.21. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi isAllyOf dengan individu lain memunculkan data *inference* isAllyOf sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi isAllyOf telah berhasil.



Gambar 6.2.21 Uji Coba Relasi isAllyOf

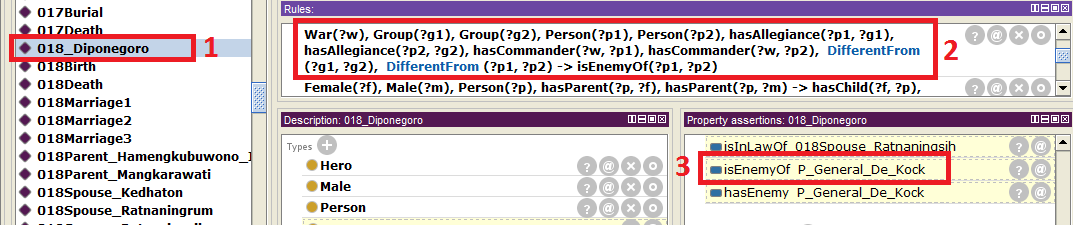
#### Pengujian Kevalidan Relasi isEnemyOf

Pada tahap pengujian relasi isEnemyOf, fakta baru yang muncul pada setiap *instance* dicek kebenaran relasi antar individunya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan sudah benar dan tidak menimbulkan kesalahan pembacaan ketika diberikan *instance* lain yang memiliki properti berbeda. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.22.

Tabel 6.22 Pengujian Kevalidan Relasi isEnemyOf

|  |  |
| --- | --- |
| ID | TA-UJ.ON0022 |
| Nama | Pengujian kevalidan relasi isEnemyOf |
| Tujuan Pengujian | Menguji kevalidan ontologi terhadap relasi isEnemyOf |
| Skenario 1 | Melihat *inference* data individu yang seharusnya memiliki relasi isEnemyOf dan tidak. |
| Kondisi Awal | Individu bertipe Person yang seharusnya memiliki saudara tiri perempuan belum memiliki properti isEnemyOf. |
| Data Uji | Data uji merupakan data individu dari DBpedia dan hasil ekstrak Wikipedia yang sudah disimpan pada ontologi. |
| Langkah Pengujian | Pengguna membuka *file* ontologi melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R |
| Hasil Yang Diharapkan | Pada setiap individu yang seharusnya memiliki relasi isEnemyOf didapatkan hasil *inference* data yang benar. |
| Hasil Yang Didapat | Protégé menampilkan informasi data *inference* individu lain yang memiliki relasi isEnemyOf dengan individu yang bersangkutan. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Pada setiap individu yang seharusnya memiliki relasi isEnemyOf didapatkan hasil *inference* data yang benar. |

Untuk hasil pengujian kevalidan relasi isEnemyOf dapat dilihat pada Gambar 6.22. Pada gambar tersebut dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi isEnemyOf dengan individu lain memunculkan data *inference* isEnemyOf sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub subbab **Error! Reference source not found.**. Sedangkan *pointer* 3 menunjukkan fakta baru berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi isEnemyOf telah berhasil.



Gambar 6.2.22 Uji Coba Relasi isEnemyOf

### Pengujian Perbandingan Data

Pengujian perbandingan data merupakan tahap uji setelah fungsionalitas *rule* telah dilaksanakan. Pengujian ini bertujaun untuk membandingkan kekayaan data. Data yang dibandingkan merupakan data asli DBpedia dengan data yang didapatkan dari ontologi ihero. Pengujian dilakukan secara mandiri dengan membandingkan data tokoh pada halaman DBpedia dengan data tokoh pada ontologi ihero yang disertai dengan fakta-fakta baru.

#### Pengujian Perbandingan Data Sukarno

Pada tahap pengujian tokoh Sukarno, 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.

Tabel 6.23 Pengujian Perbandingan Data Sukarno

|  |  |
| --- | --- |
| ID | TA-UJ.DT0001 |
| Nama | Pengujian perbandingan data Sukarno |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Sukarno pada halaman DBpedia. |
| Kondisi Awal | Individu Sukarno belum memiliki properti hasChild, hasSon, dan hasDaughter. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Sukarno pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia Sukarno yang beralamat *dbpedia.org/page/Sukarno*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Sukarno. |
| Hasil Yang Diharapkan | Data individu Sukarno pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Sukarno, diantaranya yaitu hasChild, hasSon, dan hasDaughter. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Sukarno pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Fatmawati

Pada tahap pengujian tokoh Fatmawati, 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 6.24.

Tabel 6.24 Pengujian Perbandingan Data Fatmawati

|  |  |
| --- | --- |
| ID | TA-UJ.DT0002 |
| Nama | Pengujian perbandingan data Fatmawati |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Fatmawati pada halaman DBpedia. |
| Kondisi Awal | Individu Fatmawati belum memiliki properti hasSon, dan hasDaughter. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Fatmawati pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia Fatmawati yang beralamat *dbpedia.org/page/Fatmawati*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Fatmawati. |
| Hasil Yang Diharapkan | Data individu Fatmawati pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Fatmawati, diantaranya yaitu hasSon, dan hasDaughter. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Fatmawati pada perangkat lunak iHerolation lebih kaya dibandingkan data pada DBpedia. |

#### Pengujian Perbandingan Data Mohammad Hatta

Pada tahap pengujian tokoh Mohammad Hatta, 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 6.25.

Tabel 6.25 Pengujian Perbandingan Data Mohammad Hatta

|  |  |
| --- | --- |
| ID | TA-UJ.DT0003 |
| Nama | Pengujian perbandingan data Mohammad Hatta |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Mohammad Hatta pada halaman DBpedia. |
| Kondisi Awal | Individu Mohammad Hatta belum memiliki properti hasDaughter, hasAlly, dan hasEnemy. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Mohammad Hatta pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia Mohammad Hatta yang beralamat *dbpedia.org/page/Mohammad\_Hatta*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Mohammad Hatta. |
| Hasil Yang Diharapkan | Data individu Mohammad Hatta pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Mohammad Hatta, diantaranya yaitu hasDaughter, hasAlly, dan hasEnemy. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Mohammad Hatta pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Hasyim Asyari

Pada tahap pengujian tokoh Hasyim Asyari, 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 6.26.

Tabel 6.26 Pengujian Perbandingan Data Hasyim Asyari

|  |  |
| --- | --- |
| ID | TA-UJ.DT0004 |
| Nama | Pengujian perbandingan data Hasyim Asyari |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Hasyim Asyari pada halaman DBpedia. |
| Kondisi Awal | Individu Hasyim Asyari belum memiliki properti hasTeacher. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Hasyim Asyari pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Hasyim\_Asy%27ari*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Hasyim Asyari. |
| Hasil Yang Diharapkan | Data individu Hasyim Asyari pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Hasyim Asyari, diantaranya yaitu hasTecaher. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Hasyim Asyari pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Mohammad Natsir

Pada tahap pengujian tokoh Mohammad Natsir, 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 6.27.

Tabel 6.27 Pengujian Perbandingan Data Mohammad Natsir

|  |  |
| --- | --- |
| ID | TA-UJ.DT0004 |
| Nama | Pengujian perbandingan data Mohammad Natsir |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Mohammad Natsir pada halaman DBpedia. |
| Kondisi Awal | Individu Mohammad Natsir memiliki *value* properti successor dan predecessor yang tidak normal. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Mohammad Natsir pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Mohammad\_Natsir*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Mohammad Natsir. |
| Hasil Yang Diharapkan | Data individu Mohammad Natsir pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan *value* properti successor dan predecessor dalam bentuk yang normal dibandingkan dengan yang dimiliki oleh DBpedia. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Mohammad Natsir pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Cut Nyak Dhien

Pada tahap pengujian tokoh Cut Nyak Dhien, 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. Cut Nyak Dhien merupakan tokoh yang difungsikan terkait peran sebagai aktivis kemerdekaan, sehingga dihubungkan dengan properti *isEnemyOf* yang berarti memiliki musuh individu lain. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.28.

Tabel 6.28 Pengujian Perbandingan Data Cut Nyak Dhien

|  |  |
| --- | --- |
| ID | TA-UJ.DT0006 |
| Nama | Pengujian perbandingan data Cut Nyak Dhien |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Cut Nyak Dhien pada halaman DBpedia. |
| Kondisi Awal | Individu Cut Nyak Dhien belum memiliki properti hasStepChild, hasAlly, dan hasEnemy. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Cut Nyak Dhien pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Cut\_Nyak\_Dhien*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Cut Nyak Dhien. |
| Hasil Yang Diharapkan | Data individu Cut Nyak Dhien pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Cut Nyak Dhien, diantaranya yaitu hasStepChild, hasAlly, dan hasEnemy. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Cut Nyak Dhien pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Teuku Umar

Pada tahap pengujian tokoh Teuku Umar, 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 6.29.

Tabel 6.29 Pengujian Perbandingan Data Teuku Umar

|  |  |
| --- | --- |
| ID | TA-UJ.DT0007 |
| Nama | Pengujian perbandingan data Teuku Umar |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Teuku Umar pada halaman DBpedia. |
| Kondisi Awal | Individu Teuku Umar belum memiliki properti hasStepChild, hasAlly, dan hasEnemy. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Teuku Umar pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Teuku\_Umar*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Teuku Umar. |
| Hasil Yang Diharapkan | Data individu Teuku Umar pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Teuku Umar, diantaranya yaitu hasStepChild, hasAlly, dan hasEnemy. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Teuku Umar pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Cut Nyak Meutia

Pada tahap pengujian tokoh Cut Nyak Meutia, 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 6.30.

Tabel 6.30 Pengujian Perbandingan Data Cut Nyak Meutia

|  |  |
| --- | --- |
| ID | TA-UJ.DT0008 |
| Nama | Pengujian perbandingan data Cut Nyak Meutia |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Cut Nyak Meutia pada halaman DBpedia. |
| Kondisi Awal | Individu Cut Nyak Meutia belum memiliki properti hasAlly. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Cut Nyak Meutia pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Cut\_Nyak\_Meutia*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Cut Nyak Meutia. |
| Hasil Yang Diharapkan | Data individu Cut Nyak Meutia pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Cut Nyak Meutia, diantaranya yaitu hasAlly. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Cut Nyak Meutia pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Teuku Nyak Arif

Pada tahap pengujian tokoh Teuku Nyak Arif, 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 6.31.

Tabel 6.31 Pengujian Perbandingan Data Teuku Nyak Arif

|  |  |
| --- | --- |
| ID | TA-UJ.DT0009 |
| Nama | Pengujian perbandingan data Teuku Nyak Arif |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Teuku Nyak Arif pada halaman DBpedia. |
| Kondisi Awal | Individu Teuku Nyak Arif belum memiliki properti hasSpouse dan hasChild. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Teuku Nyak Arif pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Teuku\_Nyak\_Arif*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Teuku Nyak Arif. |
| Hasil Yang Diharapkan | Data individu Teuku Nyak Arif pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Teuku Nyak Arif, diantaranya yaitu hasSpouse dan hasChild. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Teuku Nyak Arif pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Pierre Tendean

Pada tahap pengujian tokoh Pierre Tendean, 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 6.32.

Tabel 6.32 Pengujian Perbandingan Data Pierre Tendean

|  |  |
| --- | --- |
| ID | TA-UJ.DT0010 |
| Nama | Pengujian perbandingan data Pierre Tendean |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Pierre Tendean pada halaman DBpedia. |
| Kondisi Awal | Individu Pierre Tendean belum memiliki properti hasOfficer. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Pierre Tendean pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Pierre\_Tendean*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Pierre Tendean. |
| Hasil Yang Diharapkan | Data individu Pierre Tendean pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Pierre Tendean, diantaranya yaitu hasOfficer. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Pierre Tendean pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Abdul Haris Nasution

Pada tahap pengujian tokoh Abdul Haris Nasution, 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 6.33.

Tabel 6.33 Pengujian Perbandingan Data Abdul Haris Nasution

|  |  |
| --- | --- |
| ID | TA-UJ.DT0011 |
| Nama | Pengujian perbandingan data Abdul Haris Nasution |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Abdul Haris Nasution pada halaman DBpedia. |
| Kondisi Awal | Individu Abdul Haris Nasution belum memiliki properti hasAlly. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Abdul Haris Nasution pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Abdul\_Haris\_Nasution*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Abdul Haris Nasution. |
| Hasil Yang Diharapkan | Data individu Abdul Haris Nasution pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Abdul Haris Nasution, diantaranya yaitu hasAlly. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Abdul Haris Nasution pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Ahmad Yani

Pada tahap pengujian tokoh Ahmad Yani, 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 6.34.

Tabel 6.34 Pengujian Perbandingan Data Ahmad Yani

|  |  |
| --- | --- |
| ID | TA-UJ.DT0012 |
| Nama | Pengujian perbandingan data Ahmad Yani |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Ahmad Yani pada halaman DBpedia. |
| Kondisi Awal | Individu Ahmad Yani belum memiliki properti hasEnemy. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Ahmad Yani pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Ahmad\_Yani*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Ahmad Yani. |
| Hasil Yang Diharapkan | Data individu Ahmad Yani pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Ahmad Yani, diantaranya yaitu hasEnemy. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Ahmad Yani pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Ahmad Dahlan

Pada tahap pengujian tokoh Ahmad Dahlan, 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 6.35.

Tabel 6.35 Pengujian Perbandingan Data Ahmad Dahlan

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

#### Pengujian Perbandingan Data Nyai Ahmad Dahlan

Pada tahap pengujian tokoh Nyai Ahmad Dahlan, 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 6.36.

Tabel 6.36 Pengujian Perbandingan Data Nyai Ahmad Dahlan

|  |  |
| --- | --- |
| ID | TA-UJ.DT0014 |
| Nama | Pengujian perbandingan data Nyai Ahmad Dahlan |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Nyai Ahmad Dahlan pada halaman DBpedia. |
| Kondisi Awal | Individu Nyai Ahmad Dahlan belum memiliki properti hasSpouse. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Nyai Ahmad Dahlan pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Nyai\_Ahmad\_Dahlan*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Nyai Ahmad Dahlan. |
| Hasil Yang Diharapkan | Data individu Nyai Ahmad Dahlan pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Nyai Ahmad Dahlan, diantaranya yaitu hasSpouse. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Nyai Ahmad Dahlan pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Tuanku Imam Bonjol

Pada tahap pengujian tokoh Tuanku Imam Bonjol, 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 6.37.

Tabel 6.37 Pengujian Perbandingan Data Tuanku Imam Bonjol

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

#### Pengujian Perbandingan Data Wahid Hasyim

Pada tahap pengujian tokoh Wahid Hasyim, 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 6.38.

Tabel 6.38 Pengujian Perbandingan Data Wahid Hasyim

|  |  |
| --- | --- |
| ID | TA-UJ.DT0016 |
| Nama | Pengujian perbandingan data Wahid Hasyim |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Wahid Hasyim pada halaman DBpedia. |
| Kondisi Awal | Individu Wahid Hasyim belum memiliki properti hasSon dan hasDaughter. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Wahid Hasyim pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Wahid\_Hasyim*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Wahid Hasyim. |
| Hasil Yang Diharapkan | Data individu Wahid Hasyim pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Wahid Hasyim, diantaranya yaitu hasSon dan hasDaughter. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Wahid Hasyim pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |

#### Pengujian Perbandingan Data Mas Mansoer

Pada tahap pengujian tokoh Mas Mansoer, 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 6.39.

Tabel 6.39 Pengujian Perbandingan Data Mas Mansoer

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

#### Pengujian Perbandingan Data Diponegoro

Pada tahap pengujian tokoh Diponegoro, 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 6.40.

Tabel 6.40 Pengujian Perbandingan Data Diponegoro

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

#### Pengujian Perbandingan Data Sudirman

Pada tahap pengujian tokoh Sudirman, 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 6.41.

Tabel 6.41 Pengujian Perbandingan Data Sudirman

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

#### Pengujian Perbandingan Data Slamet Rijadi

Pada tahap pengujian tokoh Slamet Rijadi, 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. Slamet Rijadi merupakan tokoh yang difungsikan terkait peran sebagai aktivis kemerdekaan, sehingga dihubungkan dengan properti *isEnemeyOf* yang berarti memiliki musuh individu lain. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel 6.42.

Tabel 6.42 Pengujian Perbandingan Data Slamet Rijadi

|  |  |
| --- | --- |
| ID | TA-UJ.DT0018 |
| Nama | Pengujian perbandingan data Slamet Rijadi |
| Tujuan Pengujian | Menguji apakah fakta baru yang diperoleh dari ontologi ihero sudah tersedia di DBpedia. |
| Skenario 1 | Melihat *inference* data yang ditampilkan pada iHerolation dan mencocokkannya dengan data Slamet Rijadi pada halaman DBpedia. |
| Kondisi Awal | Individu Slamet Rijadi belum memiliki properti hasAlly dan hasEnemy. |
| Data Uji | Data uji merupakan data hasil pengembangan ontologi yang ditampilkan pada perangkat lunak iHerolation. |
| Langkah Pengujian | Pengguna memiih entitas Slamet Rijadi pada perangkat lunak iHerolation.  Pengguna membuka halaman DBpedia yang beralamat *dbpedia.org/page/Slamet\_Rijadi*.  Pengguna membandingkan data pada perangkat lunak iHerolation dengan halaman DBpedia Slamet Rijadi. |
| Hasil Yang Diharapkan | Data individu Slamet Rijadi pada perangkat lunak iHerolation lebih kaya dibandingkan dengan data pada DBpedia. |
| Hasil Yang Didapat | Didapatkan properti dengan fakta-fakta baru yang belum dimiliki oleh DBpedia pada data tokoh Slamet Rijadi, diantaranya yaitu hasAlly dan hasEnemy. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Data individu Slamet Rijadi pada perangkat lunak iHerolation 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 ontologi yang telah dijelaskan pada Subbab 6.2.1, evaluasi pengujian perbandingan data yang telah dijelaskan pada Subbab 6.2.2, dan evaluasi pengujian kompleksitas ontologi yang telah dijelaskan pada Subbab 6.2.3.

### Evaluasi Pengujian Ontologi

Rangkuman mengenai hasil pengujian ontologi dapat dilihat pada Tabel 6.45. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil. Sehingga bisa ditarik disimpulkan bahwa ontologi yang dikembangkan telah sesuai dengan yang diharapkan.

Tabel 6.45 Rangkuman Hasil Pengujian

| ID | Nama | Skenario | Hasil |
| --- | --- | --- | --- |
| TA-UJ.ON0001 | Pengujian kevalidan relasi isSpouseOf | Skenario 1 | Berhasil |
| TA-UJ.ON0002 | Pengujian kevalidan relasi hasHusband | Skenario 1 | Berhasil |
| TA-UJ.ON0003 | Pengujian kevalidan relasi hasWife | Skenario 1 | Berhasil |
| TA-UJ.ON0004 | Pengujian kevalidan relasi hasFather | Skenario 1 | Berhasil |
| TA-UJ.ON0005 | Pengujian kevalidan relasi hasMother | Skenario 1 | Berhasil |
| TA-UJ.ON0006 | Pengujian kevalidan relasi hasChild | Skenario 1 | Berhasil |
| TA-UJ.ON0007 | Pengujian kevalidan relasi hasSon | Skenario 1 | Berhasil |
| TA-UJ.ON0008 | Pengujian kevalidan relasi hasDaughter | Skenario 1 | Berhasil |
| TA-UJ.ON0009 | Pengujian kevalidan relasi hasSibling | Skenario 1 | Berhasil |
| TA-UJ.ON0010 | Pengujian kevalidan relasi hasBrother | Skenario 1 | Berhasil |
| TA-UJ.ON0011 | Pengujian kevalidan relasi hasSister | Skenario 1 | Berhasil |
| TA-UJ.ON0012 | Pengujian kevalidan relasi hasStepParent | Skenario 1 | Berhasil |
| TA-UJ.ON0013 | Pengujian kevalidan relasi hasStepFather | Skenario 1 | Berhasil |
| TA-UJ.ON0014 | Pengujian kevalidan relasi hasStepMother | Skenario 1 | Berhasil |
| TA-UJ.ON0015 | Pengujian kevalidan relasi hasStepChild | Skenario 1 | Berhasil |
| TA-UJ.ON0016 | Pengujian kevalidan relasi hasStepSon | Skenario 1 | Berhasil |
| TA-UJ.ON0017 | Pengujian kevalidan relasi hasStepDaughter | Skenario 1 | Berhasil |
| TA-UJ.ON0018 | Pengujian kevalidan relasi hasStepSibling | Skenario 1 | Berhasil |
| TA-UJ.ON0019 | Pengujian kevalidan relasi hasStepBrother | Skenario 1 | Berhasil |
| TA-UJ.ON0020 | Pengujian kevalidan relasi hasStepSister | Skenario 1 | Berhasil |
| TA-UJ.ON0021 | Pengujian kevalidan relasi isAllyOf | Skenario 1 | Berhasil |
| TA-UJ.ON0022 | Pengujian kevalidan relasi isEnemyOf | Skenario 1 | Berhasil |

### 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 6.46 Rangkuman Hasil Pengujian

| ID | Nama | Skenario | Hasil |
| --- | --- | --- | --- |
| TA-UJ.DT0001 | Pengujian perbandingan data Sukarno | Skenario 1 | Berhasil |
| TA-UJ.DT0002 | Pengujian perbandingan data Fatmawati | Skenario 1 | Berhasil |
| TA-UJ.DT0003 | Pengujian perbandingan data Mohammad Hatta | Skenario 1 | Berhasil |
| TA-UJ.DT0004 | Pengujian perbandingan data Hasyim Asyari | Skenario 1 | Berhasil |
| TA-UJ.DT0005 | Pengujian perbandingan data Mohammad Natsir | Skenario 1 | Berhasil |
| TA-UJ.DT0006 | Pengujian perbandingan data Cut Nyak Dhien | Skenario 1 | Berhasil |
| TA-UJ.DT0007 | Pengujian perbandingan data Teuku Umar | Skenario 1 | Berhasil |
| TA-UJ.DT0008 | Pengujian perbandingan data Cut Nyak Meutia | Skenario 1 | Berhasil |
| TA-UJ.DT0009 | Pengujian perbandingan data Teuku Nyak Arif | Skenario 1 | Berhasil |
| TA-UJ.DT0010 | Pengujian perbandingan data Pierre Tendean | Skenario 1 | Berhasil |
| TA-UJ.DT0011 | Pengujian perbandingan data Abdul Haris Nasution | Skenario 1 | Berhasil |
| TA-UJ.DT0012 | Pengujian perbandingan data Ahmad Yani | Skenario 1 | Berhasil |
| TA-UJ.DT0013 | Pengujian perbandingan data Ahmad Dahlan | Skenario 1 | Berhasil |
| TA-UJ.DT0014 | Pengujian perbandingan data Nyai Ahmad Dahlan | Skenario 1 | Berhasil |
| TA-UJ.DT0015 | Pengujian perbandingan data Tuanku Imam Bonjol | Skenario 1 | Berhasil |
| TA-UJ.DT0016 | Pengujian perbandingan data Wahid Hasyim | Skenario 1 | Berhasil |
| TA-UJ.DT0017 | Pengujian perbandingan data Mas Mansoer | Skenario 1 | Berhasil |
| TA-UJ.DT0018 | Pengujian perbandingan data Diponegoro | Skenario 1 | Berhasil |
| TA-UJ.DT0019 | Pengujian perbandingan data Sudirman | Skenario 1 | Berhasil |
| TA-UJ.DT0020 | Pengujian perbandingan data Slamet Rijadi | 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. Penyaringan *person* yang memiliki *alias* lebih dari satu agar tidak terjadi repetisi dalam visualisasi.
2. 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. R. Ramos, "Biography Light Ontology: An Open Vocabulary For Encoding Biographic Texts," University of California, Berkeley, 2009. |
| [2] | 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. |
| [3] | 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. |
| [4] | 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. |
| [5] | 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. |
| [6] | 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. |
| [7] | S. J. Miller, Introduction to Ontology Concepts and Terminology, Lisbon, Portugal: University of Wisconsin-Milwaukee, 2013. |
| [8] | M. A. Ramadhanie, Penerapan Ontologi Objek Pembelajaran Untuk Kebutuhan Personalisasi E-Learning Berbasis Semantic Web, Depok: Universitas Indonesia, 2009. |
| [9] | 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. |
| [10] | 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]. |
| [11] | I. N. Bagus Caturbawa, "Case Tool untuk Pemodelan Data Semantik dalam Web Ontology Language (OWL)," Institut Teknologi Sepuluh Nopember, Surabaya, 2009. |
| [12] | J. Heflin, "OWL Web Ontology Language Use Cases and Requirements," [Online]. Available: https://www.w3.org/TR/webont-req/. [Accessed 06 January 2016]. |
| [13] | S. Nikles, "Expressiveness of Enterprise Modelling Languages," University of Applied Sciences Northwestern Switzerland, Basel, 2010. |
| [14] | 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. |
| [15] | "Professor Robert Stevens," [Online]. Available: http://www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl. [Accessed 06 January 2016]. |
| [16] | D. Brickley and L. Miller, "FOAF Vocabulary Specification 0.98," Creative Commons, 2010. |
| [17] | 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. |
| [18] | I. Davis and D. Galbraith, "BIO: A vocabulary for biographical information," [Online]. Available: http://vocab.org/bio/. [Accessed 06 January 2016]. |
| [19] | R. R. Larson, "Bringing Lives to Light: Biography in Context," University of California, Berkeley, 2010. |
| [20] | 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. |
| [21] | B. Parsia and E. Sirin, "Pellet: An OWL DL Reasoner," University of Maryland, College Park. |
| [22] | 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. |
| [23] | "EasyRdf API Documentation," [Online]. Available: http://www.easyrdf.org/docs/api/EasyRdf\_Graph.html. [Accessed 30 May 2016]. |
| [24] | 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]. |
| [25] | 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. |
| [26] | Z. T. Inc., "An overview on PHP," Zend The PHP Company, 2007. |
| [27] | T. Point, "PHP Hypertext Preprocessor," 2015. [Online]. Available: www.tutorialspoint.com/php/php\_tutorial.pdf. [Accessed 10 June 2016]. |
| [28] | D. Wu and A. Håkansson, "A Method of Identifying Ontology Domain," *Procedia Computer Science,* vol. 35, pp. 504-513, 2014. |
| [29] | "XML," [Online]. Available: https://en.wikipedia.org/wiki/XML. [Accessed 10 June 2016]. |
| [30] | 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. |

# 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 | Teuku Umar | 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 | Teuku Umar | 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.