

**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.**

**JURUSAN 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.**

**JURUSAN 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)***

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

**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

*Cultural heritage (Keragaman budaya) dapat terlahir dari tokoh penting seperti tokoh bersejarah dan pahlawan. Indonesia memiliki sejumlah pahlawan nasional. Salah satu open data yang menyediakan daftar Pahlawan Nasional Indonesia adalah Wikipedia. Konten dalam suatu halaman Wikipedia sendiri memiliki keterkaitan dengan DBpedia. DBpedia adalah database hasil ekstrak konten terstruktur dari informasi yang dimuat Wikipedia. Database tersebut berisi berbagai properti. Data yang terdapat dalam properti DBpedia dapat menjelaskan biografi yang dimiliki oleh seorang tokoh. Setiap tokoh memiliki biografi hidup yang berbeda. Biografi mencatat keterkaitan antar seorang tokoh dengan yang lainnya. Keterkaitan tersebut dapat digambarkan dengan ontologi. Actor adalah salah satu domain dalam cultural heritage yang dapat diontologikan. Ruang lingkup actor mencakup person, group, dan organization.*

*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. Tujuan dari pengerjaan tugas akhir ini adalah menghasilkan suatu metode ontologi untuk pencarian relasi antar tokoh dalam domain sejarah Indonesia secara otomatis. Jenis hubungan antar tokoh bisa berupa hubungan dalam keluarga atau hubungan antar teman.*

*Untuk mencari hubungan antar tokoh sejarah Indonesia, beberapa ontologi yang sudah ada seperti Family Relationship Ontology, FOAF (Friend of a Friend), dan BIO Vocabulary digabungkan dan dikembangkan. Penggabungan ontologi tersebut dilakukan dengan Ontology Web Language (OWL) menggunakan tools Protege. Berdasarkan uji coba yang dilakukan, ontologi yang sudah dikembangkan dalam pengerjaan Tugas Akhir ini dapat menampilkan fakta-fakta baru yang belum ada di DBpedia. Hal ini tentunya memperkaya informasi yang dimiliki oleh DBpedia. Selain itu, hasil yang didapatkan juga mempermudah history science dalam menentukan hubungan antar tokoh sejarah Indonesia berdasarkan data kehidupannya. Selanjutnya, hal ini dapat menambah wawasan untuk mengenal tokoh-tokoh lain yang berkaitan dengan seorang tokoh sejarah Indonesia.*

***Kata kunci: BIO Vocabulary, Family Relationship Ontology, Ontologi, Tokoh Sejarah Indonesia, Penggabungan Ontologi.***

**SEARCH RELATIONS AMONG HISTORICAL INDONESIA FIGURES USING ONTOLOGY**

**Name : Madis Saralita**

**NRP : 5112100038**

**Department : Department of Informatics**

**Faculty of Information Technology ITS**

**Supervisor I : Sarwosri, S.Kom., M.T.**

**Supervisor II : Nurul Fajrin A.,S.Kom., M.Sc.**

# ABSTRACT

*Cultural heritage can be born from historical figures and heroes. Indonesia has a lot of national heroes. One of open data provides Indonesia National Heroes list is Wikipedia. Wikipedia page is coupled with DBpedia, structured content database which is extracted from Wikipedia page. DBpedia contains of much property. Data which is contained in DBpedia property can describe biography owned by someone. Each person has different biography. Biography describes relation between person with others. That relation can be described with ontology. One of domains which can be described by ontology is Actor. The scope of actor includes person, group, and organization.*

*Based on life history, relationships between person can be known. In addition, relationships can determine experience between them. Purpose of this final project is resulting ontology method to search relation between Indonesia historical figures. Kind of relationships between person can be relationships in the family or relationships between friends.*

*To find relationships between Indonesia historical figures, some of existing ontology such as Family Relationship Ontology, FOAF (Friend of a Friend), and BIO Vocabulary combined and developed. Ontology is built with Ontology Web Language (OWL) using Protégé tools. Based on testing, ontology presents new facts which did not exist in DBpedia. The results can enrich information performed by DBpedia and simplify history science in determining relationship between Indonesia historical figures based on their biography. Furthermore, it also make people to know other figures associated with Indonesia historical figure.*

***Key words: BIO Vocabulary, Family Relationship Ontology, Indonesia Historical Figure, Merging Ontology, Ontology.***

# KATA PENGANTAR

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

**“Pencarian Relasi Antar Tokoh Sejarah Indonesia Menggunakan Ontologi”**

Tugas akhir ini dilakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika Fakultas Teknologi Informasi 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 Sarwosri, S.Kom., M.T. selaku dosen pembimbing I yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
4. Ibu Nurul Fajrin 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. Ibu Anny Yuniarti, S.Kom, M.Comp.Sc. selaku dosen wali yang telah mengarahkan penulis, serta segenap dosen Teknik Informatika ITS yang telah memberikan ilmunya.
6. Direktorat Jenderal Pendidikan Tinggi (Dikti) yang telah memberikan beasiswa Bidik Misi untuk menunjang keperluan kuliah dan kebutuhan hidup penulis selama 4 tahun.
7. Sahabat-sahabat penulis, Yutika, Afina, Julia, dan Azizha yang selalu memahami, menyemangati dan menghibur penulis serta selalu ada disaat masa-masa tersulit penulis.
8. Teman penulis, Muhammad Baiquni yang selalu memberikan semangat dan informasi terkait pengerjaan Tugas Akhir.
9. Teman disuksi penulis, Alif, Bakhtiar, Kamali, dan Wimba yang bersedia meluangkan waktunya untuk berbagi ilmu dengan penulis.
10. Teman-teman anggota Jam'iyah Sholawat Syifaul Qolbi, Mbak Nisak, Dea, Rahma, Rindi, Nur, Lina, Salsa, Wanda, Ferdi, dan Wigun yang selalu mencairkan suasana saat kejenuhan datang.
11. Seluruh keluarga TC 2012 yang selalu menemani dan memberi semangat selama 4 tahun perkuliahan.
12. 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 2016

Penulis

# DAFTAR ISI

[HALAMAN JUDUL iii](#_Toc456657710)

[LEMBAR PENGESAHAN vii](#_Toc456657711)

[ABSTRAK ix](#_Toc456657712)

[ABSTRACT xi](#_Toc456657713)

[KATA PENGANTAR xiii](#_Toc456657714)

[DAFTAR ISI xv](#_Toc456657715)

[DAFTAR GAMBAR xix](#_Toc456657716)

[DAFTAR TABEL xxi](#_Toc456657717)

[DAFTAR KODE SUMBER xxv](#_Toc456657718)

[BAB I PENDAHULUAN 1](#_Toc456657719)

[1.1. Latar Belakang 1](#_Toc456657720)

[1.2. Rumusan Masalah 3](#_Toc456657721)

[1.3. Batasan Masalah 3](#_Toc456657722)

[1.4. Tujuan 4](#_Toc456657723)

[1.5. Metodologi 5](#_Toc456657724)

[1.6. Sistematika Penulisan 6](#_Toc456657725)

[BAB II DASAR TEORI 9](#_Toc456657726)

[2.1. Penelitian Terkait 9](#_Toc456657727)

[2.2. Tokoh Bersejarah 11](#_Toc456657728)

[2.3. Ontologi 12](#_Toc456657729)

[2.4. OWL 16](#_Toc456657730)

[2.5. Semantic Web Rule Language (SWRL) 19](#_Toc456657731)

[2.6. Family Relationships Ontology 20](#_Toc456657732)

[2.7. FOAF (Friend of a Friend) 23](#_Toc456657733)

[2.8. BIO Vocabulary 27](#_Toc456657734)

[2.9. Pellet Reasoner 30](#_Toc456657735)

[2.10. EasyRDF 31](#_Toc456657736)

[2.11. PHP 33](#_Toc456657737)

[BAB III METODOLOGI PEMECAHAN MASALAH 35](#_Toc456657738)

[3.1. Analisis Data 36](#_Toc456657739)

[3.1.1. Analisis Data dari Wikipedia 38](#_Toc456657740)

[3.1.2. Analisis Data dari WikiTaxi 40](#_Toc456657741)

[3.1.3. Analisis Data dari DBpedia 42](#_Toc456657742)

[3.1.4. Ekstraksi Data Menggunakan Google Refine 43](#_Toc456657743)

[3.1.5. Ekstraksi Data Berdasar Properti Tertentu 47](#_Toc456657744)

[3.2. Perancangan Ontologi 49](#_Toc456657745)

[3.2.1. Penentuan Ontologi yang Digunakan 50](#_Toc456657746)

[3.2.2. Seleksi Kelas dan Properti dari BIO 51](#_Toc456657747)

[3.2.3. Import Kelas dan Properti dari FamilyTree 53](#_Toc456657748)

[3.2.4. Import Kelas dan Properti dari FOAF 56](#_Toc456657749)

[3.2.5. Equivalent Class dan Property 56](#_Toc456657750)

[3.2.6. Pengembangan Ontologi 61](#_Toc456657751)

[3.3. Perancangan Rules 69](#_Toc456657752)

[3.3.1. Relasi Keluarga dalam Hukum 70](#_Toc456657753)

[3.3.2. Relasi Keluarga Sedarah 72](#_Toc456657754)

[3.3.3. Relasi Keluarga Tidak Sedarah 76](#_Toc456657755)

[3.3.4. Relasi Terkait *Data Properties* 80](#_Toc456657756)

[3.3.5. Relasi Terkait Profesi Tokoh 82](#_Toc456657757)

[BAB IV ANALISIS DAN PERANCANGAN SISTEM 85](#_Toc456657758)

[4.1. Analisis 85](#_Toc456657759)

[4.1.1. Cakupan Permasalahan 85](#_Toc456657760)

[4.1.2. Deskripsi Umum Sistem 85](#_Toc456657761)

[4.1.3. Spesifikasi Kebutuhan Perangkat Lunak 86](#_Toc456657762)

[4.1.4. Aktor 87](#_Toc456657763)

[4.1.5. Kasus Penggunaan 87](#_Toc456657764)

[4.2. Perancangan Antarmuka Pengguna 92](#_Toc456657765)

[BAB V IMPLEMENTASI 95](#_Toc456657766)

[5.1. Implementasi Fungsi 96](#_Toc456657767)

[5.1.1. Fungsi Dropdown Select 96](#_Toc456657768)

[5.1.2. Fungsi Get Instance Name 97](#_Toc456657769)

[5.1.3. Fungsi Get Description 98](#_Toc456657770)

[5.1.4. Fungsi Get Relations 103](#_Toc456657771)

[5.1. Implementasi Antarmuka Pengguna 121](#_Toc456657772)

[5.1.1. Implementasi Tampilan Halaman Utama 121](#_Toc456657773)

[5.1.2. Implementasi Tampilan Halaman Informasi 122](#_Toc456657774)

[BAB VI PENGUJIAN DAN EVALUASI 123](#_Toc456657775)

[6.1. Lingkungan Pengujian 123](#_Toc456657776)

[6.2. Skenario Pengujian 123](#_Toc456657777)

[6.2.1. Pengujian Ontologi 124](#_Toc456657778)

[6.2.2. Pengujian Perbandingan Data 157](#_Toc456657779)

[6.2.3. Pengujian Kompleksitas Ontologi 179](#_Toc456657780)

[6.3. Evaluasi Pengujian 182](#_Toc456657781)

[6.3.1. Evaluasi Pengujian Ontologi 182](#_Toc456657782)

[6.3.2. Evaluasi Pengujian Perbandingan Data 184](#_Toc456657783)

[6.3.1. Evaluasi Pengujian Kompleksitas Ontologi 185](#_Toc456657784)

[BAB VII KESIMPULAN DAN SARAN 187](#_Toc456657785)

[7.1. Kesimpulan 187](#_Toc456657787)

[7.2. Saran 188](#_Toc456657788)

[DAFTAR PUSTAKA 189](#_Toc456657789)

[LAMPIRAN A. DATA SEBELUM *REASONING* 193](#_Toc456657790)

[LAMPIRAN B. DATA SETELAH *REASONING* 205](#_Toc456657791)

[BIODATA PENULIS 229](#_Toc456657792)

***[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

*Cultural heritage* (Keragaman budaya) dapat terlahir dari tokoh-tokoh penting yang membawa inspirasi dalam adat maupun kehidupan. Contoh dari tokoh penting tersebut adalah tokoh bersejarah dan pahlawan. Tokoh bersejarah adalah seseorang yang namanya dikenang karena jasa atau kejadian tertentu. 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 lampau. Berbagai macam cerita dan kejadian yang menjadi kenangan sejarah sepanjang masa melibatkan nama tokoh-tokoh penting tersebut. Indonesia memiliki banyak catatan nama tokoh-tokoh bersejarah dan pahlawan nasional. 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*.

Ada beberapa jenis ontologi yang telah dibangun, salah satunya adalah ontologi untuk mengetahui hubungan dalam sebuah keluarga, yaitu *Family Relationships Ontology*. Selain itu, ada model pengetahuan yang dibuat dari hubungan antar teman. FOAF (*Friend of a Friend*) adalah sebuah skema berbasis RDF untuk mendefinisikan seseorang dan jaringan sosialnya secara semantik. FOAF memberikan manfaat dalam *information science* untuk mengetahui hubungan antar *person*. Terdapat sebuah dokumen yang menyediakan *vocabulary* untuk informasi biografi, yaitu BIO. Dengan menggunakan *vocabulary* yang dimiliki oleh BIO, biografi hidup seseorang dapat digambarkan dengan ontologi. Masing-masing ontologi seperti *Family Relationships Ontology*, FOAF, dan BIOmemiliki kelasdan properti berbeda-beda. Untuk mencari hubungan antar tokoh dalam domain sejarah, perlu memodelkan beberapa ontologi yang berkaitan dengan kisah dan rekan hidup seseorang. Ontologi tersebut dapat dibangun dan dikembangkan menggunakan *Ontology Web Language* (OWL). Salah satu aplikasi yang menyediakan fasilitas untuk membangun ontologi adalah Protege. Protege memiliki salah satu ekstensi file OWL sehingga akan mendukung pengerjaan tugas akhir ini.

Berdasarkan ontologi yang telah disebutkan sebelumnya, pengerjaan tugas akhir ini akan menggabungkan tiga ontologi tersebut dengan mengkombinasikan kelasdan properti serta melakukan pengembangan dengan menambahkan beberapa kelasdan properti baru. Hal ini bertujuan untuk menghasilkan suatu metode ontologi yang dapat melakukan pencarian relasi antar tokoh dalam domain sejarah Indonesia. Hasil yang diharapkan dari tugas akhir ini adalah terbentuknya *rule* untuk mengetahui jenis hubungan antar tokoh bersejarah yang diimplementasikan dalam suatu aplikasi sederhana. Jenis hubungan antar tokoh bisa berupa hubungan keluarga atau teman. Contoh hubungan keluarga adalah ayah dari, anak dari, saudara dari, dan lain-lain. Sedangkan contoh hubungan antar teman adalah teman, musuh, *partner* organisasi, dan sebagainya. Hal ini tentunya akan mempermudah *history science* dalam menentukan hubungan antar tokoh sejarah Indonesia berdasarkan data kehidupan yang dimiliki oleh seorang tokoh.

## Rumusan Masalah

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

1. Bagaimana menentukan *class* yang dapat digunakan untuk mendeskripsikan hubungan antar tokoh sejarah Indonesia dengan rekan hidupnya?
2. Bagaimana menentukan *property* yang nantinya dapat digunakan untuk mendefinisikan relasi dalam domain tokoh sejarah Indonesia?
3. Bagaimana menyusun *rule* yang akan digunakan untuk mencari relasi antar tokoh sejarah Indonesia yang satu dengan tokoh yang lain?
4. Bagaimana mengatur kapasitas memori agar tidak membebani perangkat pengujian?
5. Bagaimana membuat aplikasi untuk menampilkan hasil pencarian relasi antar tokoh?

## Batasan Masalah

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

1. Agen *person* yang digunakan adalah Pahlawan Nasional Indonesia yang dimuat oleh Wikipedia pada halaman Daftar Pahlawan Nasional Indonesia [[1]](#footnote-1).
2. Jumlah *person* yang digunakan adalah 20 Pahlawan.
3. *Person* yang digunakan adalah tokoh yang memiliki peran sebagai aktivis perjuangan sebelum kemerdekaan, politisi, atau tokoh agama.
4. Ontologi yang dibangun mengacu pada *Family Relationships Ontology*, FOAF, dan *BIO Vocabulary*.
5. Penamaan kelas dan properti baru yang belum ada pada *Family Relationships Ontology*, FOAF, dan *BIO Vocabulary* mengacu pada nama properti yang digunakan DBpedia.
6. Apabila terdapat kemiripan nama kelas dan properti (nama berbeda tetapi memiliki fungsi sama) maka digunakan *tools* SUMO (*Suggested Upper Merged Ontology*) untuk mencari hubungan antar kata.
7. Data yang digunakan sebagai *value* properti bersumber dari isi propertiDBpedia dan hasil esktraksi manual pada halaman Wikipedia Indonesia *person* terkait.
8. Ekstraksi data secara manual hanya dilakukan jika informasi tersebut dianggap penting untuk memunculkan fakta baru dan belum ada di DBpedia.
9. Pengembangan ontologi dilakukan menggunakan Protege 4.3 yang memiliki ekstensi OWL.
10. *Reasoner* yang digunakan adalah Pellet.
11. Aplikasi yang dibuat tidak menyediakan *form* untuk pengelolaan data (tambah, ubah, hapus).
12. Pengelolaan data yang berupa penambahan, penghapusan, dan pengubahan data hanya dapat dilakukan dengan *tools* Protege.
13. 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 sederhana yang mengimplementasikan pengembangan *Family Relationship Ontology*, FOAF (*Friend of a Friend*), dan *BIO Vocabulary* (*A vocabulary for biographical information*), untuk membantu dan mempermudah pencarian relasi antar tokoh sejarah Indonesia.

## 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, OWL, *Family Relationships Ontology*, FOAF *(Friend of a Friend)*, *BIO* *Vocabulary* (*A vocabulary for biographical information*), SWRL (*Semantic Web Rule Language*), EasyRDF, PHP, dan Pellet *reasoner*.

1. Implementasi

Pada tahap ini, akan dilakukan implementasi berdasarkan rancangan yang dibuat dalam tahap sebelumnya, yaitu *ontology mapping*. Proses *ontology mapping* dilakukan dengan *tools* Protege 4.3 yang mendukung ekstensi *Web Ontology Language* (OWL). Sedangkan aplikasi sederhana untuk menampikan hasil pencarian relasi dibangun dengan bahasa PHP menggunakan *tools* Sublime Text 3.

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.

## Penelitian Terkait

Salah satu ontologi yang dibangun dengan menggabungkan beberapa ontologi adalah *Biography Light Ontology*. Ontologi ini dibangun oleh Michele R. Ramos pada tahun 2009 [1]. *Biography Light Ontology* adalah ontologi yang dibangun dengan mengkombinasikan beberapa ontologi berstruktur RDF dan OWL. Pembangunan ontologi ini dimaksudkan untuk meningkatkan konektivitas dan interoperabilitas antar standar yang ada dan kondusif untuk pengkodean teks biografis. Ontologi yang dimanfaatkan pada penelitian ini meliputi *ABC Ontology, Linking Open Description of Events ontology* (LODE), *Friend of a Friend* (FOAF), *BIO vocabulary, Relationship vocabulary*, dan *Bibliographic Ontology* (Bibo).

LODE menyediakan evaluasi yang komperehensif dari ontologi terkait *event* dan menyajikan *interlingua model* untuk meningkatkan interoperabilitas antara standar yang ada. Ontologi ABC berperan sebagai *core ontology* (ontologi inti). ABC adalah ontologi sederhana yang mendeskripsikan objek multimedia museum dan perpustakaan. ABC dimaksudkan memberikan dasar untuk mempermudah pengembangan domain atau ontologi yang lebih spesifik. Sebagai *core ontology*, pengembang ABC mendesain ontologi ini dengan model *interoperative* yang sederhana dan mendukung prinsip-prinsip pemodelan *metadata*. FOAF menspesifikasikan sumber daya jejaring sosial yang dimaksudkan agar ontologi ini menjadi dasar kamus istilah terkait orang dan hal-hal yang mereka buat dan lakukan. BIO dan *Relationship vocabularies* yang dibuat oleh Ian Davis dibangun menggunakan kelas-kelas dari FOAF. Ontologi tersebut bertujuan untuk pengkodean informasi biografis dan hubungan antar orang. Kosa kata dari FOAF, BIO, dan *Relationship* memberikan keterbatasan untuk pengkodean teks biografis. *Bibliographic Ontology Specification* (Bibo) juga dibangun dari kosa kata FOAF dan beberapa ontologi yang lain. Ontologi ini bertujuan untuk pengkodean kutipan dan referensi bibliografi. Referensi bibliografi merupakan hal yang umum pada teks biografi, khususnya dalam penghargaan tokoh-tokoh akademik atau sastra. Meskipun Bibo memiliki cakupan di luar penelitian ini, Bibo tetap menjadi layak sebagai kebutuhan untuk interoperabilitas antara standar dan spesifikasi. *Biography Light Ontology* menggunakan pendekatan untuk pengkodean teks biografi. Kerangka ontologi ini dibangun untuk mengakomodasi hal-hal umum yang biasanya terdapat pada suatu biografi, seperti perubahan kesehatan, hubungan antar orang, atau penemuan yang pernah dihasilakan oleh seseorang.

Kelebihan dari *Biography Light* adalah model ini menawarkan struktur yang lebih dinamis untuk pengkodean peristiwa biografis daripada model lainnya. *Biography Light* menyediakan kerangka yang dapat digunakan untuk menghubungkan peristiwa biografis dengan sumber daya terkait. Meskipun terdapat berbagai kemiripan tertentu dengan prosoprografi, tujuan dan utilitas model ini tidak spesifik untuk jangka waktu tertentu atau populasi. Model ini dikembangkan dari biografi yang dimiliki oleh beberapa orang. Sedangkan kekurangan dari penelitian ini adalah belum spesifiknya istilah terkait grup, keluarga, dan organisasi. Pengembang dari *Biography Ligh*t berharap bahwa model ini selanjutnya dapat diperluas dengan menyertakan istilah terkait kelompok, biografi keluarga, atau biografi organisasi. Selain itu, diperlukan juga adanya peningkatan interoperabilitas antara spesifikasi yang ada dan standar yang relevan untuk pengkodean teks biografis. *Biography Light* memperkenalkan struktur untuk menggambarkan hubungan antara individu, organisasi, dan artefak yang tidak disediakan oleh model sederhana lainnya. Namun, penelitian lebih lanjut dan kolaborasi diperlukan untuk menguji keterbatasan dan fleksibilitas dari model ini.

## Tokoh Bersejarah

Di Indonesia, tokoh bersejarah identik sebagai pahlawan nasional. Pada dasarnya tokoh bersejarah belum tentu mendapatkan gelar sebagai pahlawan. Tokoh bersejarah adalah seseorang yang namanya dikenang karena jasa atau kejadian tertentu. Pemerintahan Indonesia memberikan gelar penghargaan tingkat tertinggi berupa sebutan Pahlawan Nasional kepada seseorang yang dianggap heroik atas tindakan yang dilakukannya sehingga tindakan tersebut dapat dikenang dan diteladani oleh warga masyarakat lainnya. Selain dianggap heroik, gelar Pahlawan Nasional juga diberikan kepada seseorang yang berjasa sangat luar biasa untuk kepentingan bangsa dan negara.

Tokoh-tokoh sejarah memiliki peran dan fungsi yang berbeda dalam budaya nasional dibandingkan dengan sebuah peristiwa. Tokoh sejarah dapat melambangkan, merealisasikan, dan mewujudkan peradaban budaya nasional. Sedangkan peristiwa penting seperti Perang Dunia II merupakan skema budaya yang dijadikan sebagai pelajaran tentang fakta terjadinya suatu *event*. Dalam budaya, peristiwa berperan untuk menyampaikan pelajaran., sedangkan pahlawan mewujudkan nilai-nilai dan menginspirasi tindakan [2].

Setiap tokoh sejarah memiliki perjalanan hidup masing-masing. Dapat dikatakan bahwa setiap tokoh tersebut memiliki biografi yang berbeda antara yang satu dengan yang lainnya. Tanpa disadari, biografi seorang tokoh terkadang memiliki sangkut paut dengan tokoh lainnya. Biografi adalah cerita atau deskripsi hidup seseorang. Sebuah biografi lebih lengkap daripada identitas yang sekedar berisi tentang daftar tanggal lahir atau mati dan data pekerjaan seseorang. Biografi juga bercerita tentang kejadian-kejadian yang pernah dialami oleh tokoh. Dalam biografi, kehidupan seorang tokoh secara lengkap sejak tokoh lahir sampai meninggal dunia. Semua jasa, tindakan, karya, dan segala hal yang dihasilkan atau dilakukan oleh seorang tokoh juga menjadi bagian dari biografi, terutama biografi tokoh sejarah. Berikut akan diberikan contoh profil dari Cut Nyak Dien yang dikutip dari *website* National Geographic Indonesia.

*Cut Nyak Dien lahir pada tahun 1848 dari keluarga bangsawan Aceh. Dari garis ayahnya, Cut Nyak Dien merupakan keturunan langsung Sultan Aceh. Ia menikah dengan Teuku Ibrahim Lamnga pada usia masih belia tahun 1862 dan memiliki seorang anak laki-laki.*

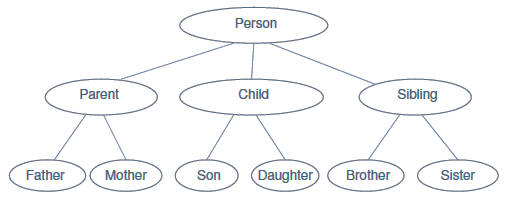
*Ketika Perang Aceh meluas tahun 1873, Cut Nyak Dien memimpin perang di garis depan, melawan Belanda yang mempunyai persenjataan lebih lengkap. Setelah bertahun-tahun bertempur, pasukannya terdesak dan memutuskan untuk mengungsi ke daerah yang lebih terpencil. Dalam pertempuran di Sela Glee Tarun, Teuku Ibrahim gugur.*

*Kendati demikian, Cut Nyak Dien melanjutkan perjuangan dengan semangat berapi-api. Kebetulan saat upacara penguburan suaminya, ia bertemu dengan Teuku Umar yang kemudian menjadi suami sekaligus rekan perjuangan* [[2]](#footnote-2)*.*

## Ontologi

Ontologi adalah spesifikasi eksplisit dari sebuah konseptualisasi. Ketika pengetahuan dari suatu domain direpresentasikan dalam deklarasi formal, satu *set* objek yang terdapat dalam deklarasi tersebut dapat dikatakan sebagai informasi yang *universal*. Objek-objek ini memiliki deskripsi masing-masing dan saling berhubungan satu sama lain yang merefleksikan *knowledge*. Dalam ontologi, informasi yang ditanamkan menggambarkan nama-nama entitas (kelas, relasi, fungsi, dan objek lain) yang dapat dimengerti oleh manusia. Secara formal, ontologi adalah pernyataan dari teori logis [3]. Ontologi digunakan untuk menangkap pengetahuan tentang beberapa domain yang menarik. Konsep dalam suatu domain tertentu dan hubungan antar konsep-konsep tersebut dapat digambarkan dengan ontologi [4]. Menurut Noy dan McGuinness [5] sebuah domain permasalahan dapat dimodelkan dengan banyak cara, dengan kata lain terdapat berbagai cara untuk membangun sebuah ontologi.

Terdapat beberapa komponen dasar dari ontologi, yaitu *classes* (direpresentasikan dalam susunan taksonomi), *relations* (digunakan untuk menghubungkan antar domain), *axioms* (digunakan untuk memodelkan pernyataan yang selalubenar), *properties* (digunakan untuk menjelaskan karakteristik umum *instance* dari sebuah kelas atau untuk menghubungkan antar *class*) dan *instance* (representasi data spesifik). Ontologi mampu memodelkan data dari suatu domain yang spesifik. Ontologi juga mempunyai fitue *inferences* yang dapat menemukan informasi tersembunyi atau fakta baru di dalam model [6].

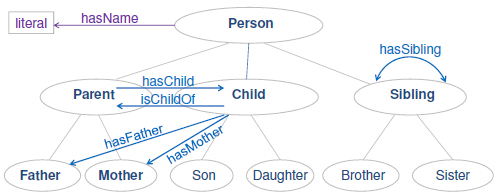


Gambar ‎2.1 Class Hierarchy

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

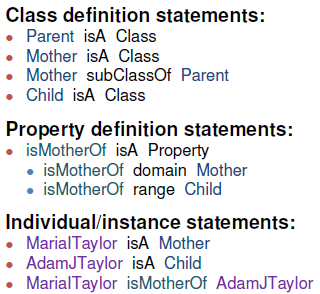
Dari beberapa komponen dasar yang telah disebutkan di paragraf sebelumnya, akan dijelaskan mengenai beberapa komponen yang penting dalam pengerjaan tugas akhir ini secara lebih spesifik. *Class* menspesifikasikan *property* yang sama dari beberapa *instance* dan mendukung klasifikasi hierarkikal. Selain itu, *class* juga mencakup *superclass* dan *subclass. Subclass* merupakan level yang lebih khusus dari *superclass*nya. Setiap *subclass* mewarisi operasi dan atribut dari *superclass*nya. *Subclass* mungkin memiliki operasi dan atributnya sendiri (yang tidak dimiliki oleh *superclass*nya). Hubungan antara *subclass* dan *superclass* digambarkan dengan *class hierarchy* seperti yang dicontohkan oleh Miller [7] pada Gambar ‎2.1.

Ontologi mendefinisikan sekumpulan *property* yang digunakan dalam domain pengetahuan khusus. Dalam konteks ontologi, *property* menghubungankan *member* dari suatu kelas ke *member* kelas lainnya [7]. *Property* mencakup *subproperties* dan *superproperties*. *Property* juga disebut sebagai “slots” dalam terminologi yang lain. Skema *property* dicontohkan pada Gambar ‎2.2.



Gambar ‎2.2 Property

*Instance* merujuk pada sesuatu yang teridentifikasi dalam model konseptual. *Instance* juga disebut sebagai individual. Sehingga dapat dikatakan bahwa *instance* menspesifikasikan *member* dari suatu *class*. *Instance* dapat berwujud (alat, bagian, benda, dkk) atau tidak berwujud (sebuah konsep, konstruksi mental). Contoh relasi antara *class, property*, dan *instance* dapat dilihat pada Gambar ‎2.3.



Gambar ‎2.3 Contoh Ontologi

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]. Menurut Cristami dan Cuel [9], sebuah model ontologi dapat dikembangkan melalui beberapa tahapan proses sebagai berikut:

* Tahap penentuan domain.
* Tahap penggunaan ulang.
* Tahap penentuan istilah pada ontologi.
* Tahap pendefinisian *class* dan *hierarchy class.*
* Tahap pendefinisian *properties.*
* Tahap pendefinisian *constraints.*
* Tahap pembuatan *instance.*

## OWL

OWL (*Web Ontology Language*) adalah bahasa yang digunakan untuk merepresentasikan definisi dari kosakata dan relasi antar kata sehingga makna suatu informasi menjadi eksplisit. OWL merupakan ekstensi dari RDF Schema. W3C *(World Wide Web Consortium)* merekomendasikan OWL dalam penulisan ontologi untuk web semantik [10]. Kemampuan RDF untuk memodelkan informasi dari sekumpulan data diadaptasi oleh OWL. Model informasi tersebut merupakan tata bahasa untuk mengartikan sekumpulan data menjadi suatu informasi [11].

Elemen pada OWL terdiri atas *classes, properties,* dan *instances*. *Classes* pada OWL mendefinisikan *root* dari semua entitas yang berkaitan dengan Thing. Sehingga secara implisit, semua *class* merupakan *subclass* dari Thing. Untuk membuat sebuah *class,* digunakan sintaksis owl:Class. Sintaksis tersebut sudah menyatakan *subclass* dengan rdfs:subClassOf.

*Class* memiliki beberapa deskripsi yaitu *equivalent classes*, *superclasses*, *members*, dan *disjoint classes*. *Equivalent classes* terjadi jika sebuah kelas memiliki definisi yang sama dengan kelas lainnya namun menggunakan nama atau URI yang berbeda. Contoh dari *equivalent classes* adalah pernyataan sebagai berikut, “*Parent it is any Person who has a child”*. *Superclasses* adalah kelas yang berada satu level di atas kelas tersebut. *Members* menjelaskan individual yang merupakan *member* dari *class* yang dirujuk.Sedangkan *disjoint classes* merupakan kelas pengecualian. *Member* dari *class* tersebut tidak bisa menjadi *member* di *class* yang lain. Biasanya tidak perlu didefinisikan, kecuali jika menggunakan *external reasoning* atau aplikasi lain yang membutuhkan *class* definisi yang lebih eksplisit. Contoh dari *disjoint classes* adalah sebagai berikut, *‘Father disjoint class with Mother”.*

*Property* merupakan relasi *binary*. Ada dua jenis *property* pada OWL, yaitu *object property* dan *datatype property*. Sama halnya seperti *class, property* juga dapat dinyatakan sebagai subPropertyOf. Untuk memberikan batasan pada suatu *property*, dapat digunakan *domain* dan *range*, yang disebut juga sebagai *global restriction* karena berlaku untuk umum, tidak terbatas pada *class* tertentu. *Datatype property* pada OWL mendeskripsikan relasi antara *instance* dengan RDF literal atau XML *Schema datatype. Datatype property* menghubungkan objek dengan *datatype values* . Berikut adalah *datatype values* yang dapat digunakan:

|  |
| --- |
| string  normalizedString  boolean |
| decimal  float  double |
| integer  nonNegativeInteger  positiveInteger  nonPositiveInteger  negativeInteger |
| long  int  short  byte |
| unsignedLong  unsignedInt  unsignedShort  unsignedByte |
| hexBinary  base64Binary |
| dateTime  time  date  gYearMonth  gYear  gMonthDay  gDay  gMonth |
| anyURI  token  language |
| NMTOKEN  Name  NCName |

Sedangkan *object properties* berfungsi untuk menghubungkan suatu objek dengan objek lainnya. *Object properties* mendeskripsikan relasi antara *instance* dari dua *classes*. Terdapat beberapa mekanisme yang digunakan untuk penentuan *property*. Tidak menutup kemungkinan bahwa mekanisme tersebut memanfaatkan *object* *property characteristics* yang menggunakan penalaran. *Object* *property characteristics* yang dapat digunakan adalah sebagai berikut:

1. *Functional and Inverse functional*

*Functional property* hanya boleh memiliki satu *member* di *range*nya untuk semua *member domain*.

Contoh: hasGender

1. *Transitive*

|  |
| --- |
| *(x property y) and (y property z) implies (x property z)* |

Contoh: hasSibling

1. *Symmetric*

|  |
| --- |
| *if (x property y) then (y property x)* |

Contoh: hasSpouse

1. *Asymmetric*

|  |
| --- |
| *if (x property y) then (y property x) → false* |

Contoh: hasWife

1. *Reflexive*

Individu bisa berelasi dengan dirinya sendiri.

1. *Irreflexive*

Tidak boleh ada individual yang berelasi dengan dirinya sendiri untuk *property* tersebut.

Contoh: isParentOf

*Individuals* atau disebut juga *instances* adalah anggota dari *classes*. *Instances* dapat dipandang sebagai objek yang ada pada domain tertentu. Sama seperti owl:Class yang menjadi *meta level* untuk *class*, begitu juga *class* yang telah didefinisikan menjadi *mata level* untuk *instance*.

Dalam proses *reasoning* pada OWL, dapat digunakan beberapa *reasoner* seperti Pellet, FACT++, HermiT, dan lain sebagainya. *Reasoner* ini digunakan untuk memeriksa konsistensi pada ontologi, melakukan klarifikasi berdasarkan relasi hierarki, dan mendapatkan fakta baru berdasarkan *axioms* dan *rules* [8]. Terdapat beberapa contoh penggunaan ontologi, khususnya OWL seperti yang didokumentasikan oleh W3C [12] sebagai berikut:

* *Web portal*
* *Multimedia collectionns*
* *Corporate web site management*
* *Design documentation*
* *Agents and services*
* *Ubiquitous computing*

## 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.

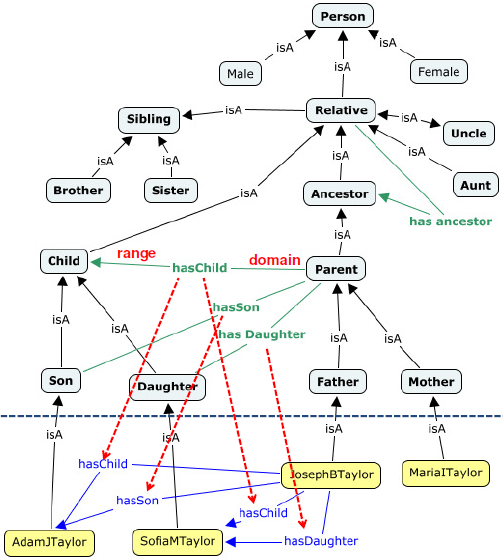
|  |
| --- |
| hasParent(?x1, ?x2), hasWife(?x2, ?x3) -> hasFather(?x1, ?x3) |

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

## Family Relationships Ontology

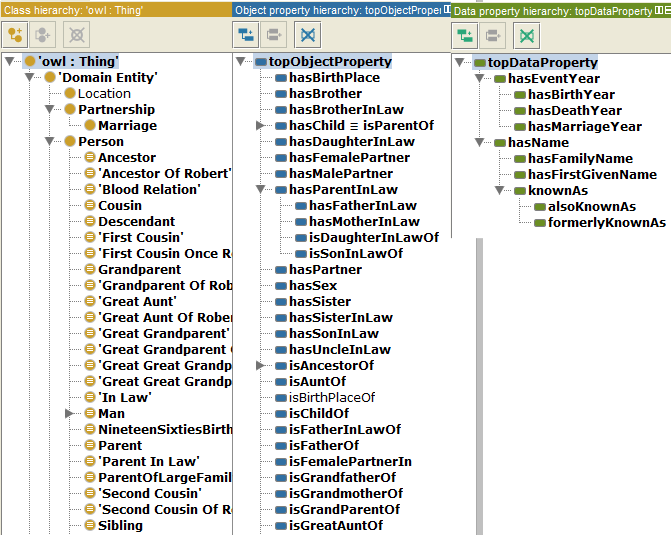
*Family relationship* umumnya digambarkan dengan terstruktur melalui silsilah keluarga. Manusia membutuhkan informasi tentang silsilah keluarganya untuk berbagai hal, diantaranya adalah untuk memperat ikatan batin antar anggota keluarga, mempermudah keturunannya dalam menelusuri asal usul keluarganya, 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. *Family Relationships Ontology* juga mempermudah untuk memahami ontologi lainnya karena hubungan kekeluargaan dapat digeneralisasikan ke domain pengetahuan lainnya dan konsepnya yang mudah dipahami. Contoh ontologi hubungan keluarga dapat dilihat pada Gambar ‎2.4.



Gambar ‎2.4 Family Relationships Map

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 pada Gambar ‎2.5.



Gambar ‎2.5 Ontologi FamilyTree Keluarga Robert Stevens

## FOAF (Friend of a Friend)

FOAF adalah mesin pembaca ontologi yang menspesifikasikan orang, kegiatan, dan hubungan mereka dengan orang atau hal-hal lain. Pada dasarnya, FOAF adalah proyek yang ditujukan untuk menghubungkan orang dan informasi menggunakan web. Tanpa memperhatikan informasi yang terdapat dalam pikiran manusia, informasi dalam bentuk digital atau dalam bentuk data faktual dapat dihubungkan. FOAF mengintegrasikan tiga jenis jaringan: jaringan sosial antar manusia, persahabatan dan asosiasi, dan jaringan representasional yang menggambarkan konsep sederhana dari fakta alam semesta [16]. Sifat terdistribusi pada jaringan FOAF mengindikasikan bahwa FOAF membutuhkan mekanisme untuk menghubungkan informasi individu. Dengan demikian, informasi terkait individu tersebut mungkin ditemukan [17].

FOAF memiliki 3 komponen dasar yang dikelompokkan dalam kategori sebagai berikut:

* *Core*

*Class* dan *property* yang membentuk inti dari FOAF. Dalam hal ini, karakteristik dari orang atau kelompok sosial yang tidak terikat dengan waktu dan teknologi digambarkan. Dengan demikian, didapatkan gambaran informasi tentang sejarah, warisan budaya, dan konteks *digital library* mengenai orang tersebut. Kategori yang terdapat dalam *Core* dapat dilihat pada Tabel ‎2.2.

Tabel ‎2.2 FOAF Core

|  |  |  |
| --- | --- | --- |
| FOAF Core | | |
| Agent  Person  name  title  img  depiction (depicts)  familyName  givenName  knows  based\_near  age  made (maker)  primaryTopic (primaryTopicOf) | Project  Organization  Group  member | Document  Image |

* *Social Web*

Selain bagian inti dari FOAF, terdapat sejumlah istilah yang digunakan untuk mendeskripsikan informasi terkait akun internet, *addressbooks*, dan aktivitas lain yang berkaitan dengan web. Kategori yang terdapat dalam *Social Web* dapat dilihat pada Tabel ‎2.3**.**

Tabel ‎2.3 FOAF Social Web

|  |
| --- |
| FOAF Social Web |
| nick  mbox  homepage  weblog  openid  jabberID  mbox\_sha1sum  interest  topic\_interest  topic (page)  workplaceHomepage  workInfoHomepage  schoolHomepage  publications  currentProject  pastProject  account  OnlineAccount  accountName  accountServiceHomepage  PersonalProfileDocument  tipjar  sha1  thumbnail  logo |

* *Linked Data Utilities*

FOAF dimulai sebagai proyek RDFWeb dan membangun sebuah model yang banyak digunakan untuk mempublikasikan data sederhana melalui keterkaitan antar dokumen RDF. FOAF tetap berperan penting untuk pengembangan “Linked Data”, tetapi juga tetap mempertahankan *scope* untuk informasi yang tidak mudah diringkas sebagai data faktual sederhana.

Awalnya, FOAF adalah web yang digunakan untuk mengintegrasikan informasi faktual dengan informasi dalam bentuk dokumen *human-oriented* (misalnya: video, buku, *spreadsheet*, 3D model), serta informasi yang hanya terekam dalam pikiran manusia. Atas dasar itulah FOAF merupakan 'percobaan' yang digunakan untuk memenuhi berbagai tujuan besar pendidikan (Misalnya: Geekcode) dan juga beberapa istilah utilitas teknis (Misalnya: *focus*, *LabelProperty*) yang mendukung hubungan antar informasi. Istilah kata yang lebih luas memungkinkan FOAF untuk menggunakan pragmatisme tertentu seperti seperangkat istilah yang berguna untuk *Web Community*, sekaligus untuk menekankan ide utama FOAF, yaitu menghubungkan *networks of information* dengan *networks of people*.

Selanjutnya, *class* dan *property* yang dimiliki oleh FOAF dapat dilihat padaTabel ‎2.4**.**

Tabel ‎2.4 FOAF Classes & Properties

| Classes | Properties |
| --- | --- |
| Agent  Document  Group  Image  LabelProperty  OnlineAccount  OnlineChatAccount  OnlineEcommerceAccount  OnlineGamingAccount  Organization  Person  PersonalProfileDocument  Project | account  accountName  accountServiceHomepage  age  aimChatID  based\_near  birthday  currentProject  depiction  depicts  dnaChecksum  familyName  family\_name  firstName  focus  fundedBy  geekcode  gender  givenName  givenname  holdsAccount  homepage  icqChatID  maker  mbox  mbox\_sha1sum  member  membershipClass  msnChatID  myersBriggs  name  nick  openid  page  pastProject  phone  plan  primaryTopic  publications  schoolHomepage  sha1  skypeID  status  surname  theme  thumbnail  tipjar  title  topic  topic\_interest  weblog  workInfoHomepage  workplaceHomepage  yahooChatID  img  interest  isPrimaryTopicOf  jabberID  knows  lastName  logo  made |

## BIO Vocabulary

BIO adalah sebuah dokumen yang memuat kosakata untuk menggambarkan informasi biografis seseorang dan latar belakang mereka serta informasi yang terkait dengan mereka [18]. BIO berisi entitas yang bermanfaat untuk mengetahui lebih banyak tentang latar belakang seseorang secara lengkap berdasarkan kejadian dan waktu. BIO menyediakan kelas untuk *event* dan properti untuk menghubungkan *person* pada *event* dan *event* pada tempat dan waktu [19]. Pendekatan ini dilakukan dengan menggambarkan kehidupan seseorang sebagai rangkaian peristiwa penting yang berhubungan dengan informasi lainnya dan saling berkaitan. BIO mendefinisikan *event framework* dan menyediakan jenis-jenis *event* yang mencakup banyak kasus penggunaan. Namun kedepannya, BIO akan dikembangkan dengan *vocabulary* lain untuk memenuhi berbagai kebutuhan yang ingin dicapai. Tujuan dari dibangunnya BIO ini adalah menentukan jenis-jenis *class* dan *property* untuk mendeskripsikan informasi biografi seseorang.

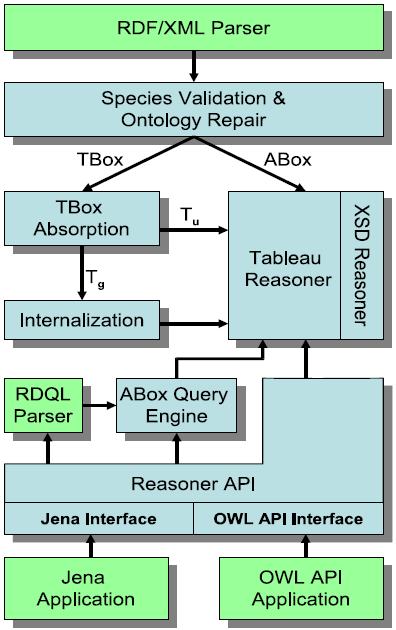
Pada intinya BIO berkaitan dengan orang, hubungan mereka, peristiwa yang terjadi dalam hidup mereka, serta waktu terjadinya sebuah peristiwa. Cakupan inti tersebut dapat digunakan untuk mendeskripsikan cerita kehidupan dan interaksi seseorang dengan orang lain, grup, organisasi, peristiwa, atau waktu kejadian secara kompleks. Interval waktu juga mungkin terikat dengan hubungan khusus antara orang-orang dan kelompok-kelompok atau organisasi.

Berbagai jenis *event* dalam kehidupan didefinisikan melalui *vocabulary* ini, termasuk kelahiran, pernikahan dan kematian. Selain itu, ada juga penobatan, pengangkatan, pengunduran, pekerjaan dan bahkan pembunuhan. Peristiwa ini tidak dimaksudkan untuk menjadi sesuatu yang komprehensif, tetapi merupakan sampel dari jenis *event* yang berkaitan dengan area biografi. Saat ini, jenis hubungan antar kosakata hanya disediakan oleh *relationship class* tertentu. Hal ini terjadi karena pertimbangan jenis hubungan seperti keluarga, kepegawaian, dan kepemilikan akan ditentukan di masa depan. Urutan kejadian dan interval waktu adalah penyusun *timeline* sejarah dimana orang dan hubungan mereka dapat ditempatkan. *Event* yang diurutkan dalam waktu akan mengaitkan satu sama lain dan mengabstraksikan interval waktunya. BIO tersedia dalam format XML atau RDF. Berbagai terminologi kosakata yang dimiliki oleh BIO dapat dilihat pada Tabel ‎2.5**.**

Tabel ‎2.5 Bio Terms

|  |  |
| --- | --- |
| Properties of a person | One-line bio  Biography  Key Words  Father  Mother  Child  Life Event  Birth Event  Death Event |
| Types of event | Event  Individual Event  Group Event  Accession  Adoption  Annulment  Assassination  Baptism  BarMitzvah  BasMitzvah  Birth  Burial  Change of Position  Coronation  Cremation  Death  Demotion  Dismissal  Divorce  migration  Employment  Execution  Enrolment  Funeral  Graduation  Imprisonment  Inauguration  Investiture  Marriage  Murder  Change of Name  Naturalization  Ordination  Promotion  Redundancy  Resignation  Retirement |
| Properties of an event | Date  Place  State  Position |
| Properties that relate an event to an agent | Agent  Parent  Employer  Officiator  Organization  Principal  Partner  Witness  Spectator |
| Properties that relate an event to another event | Concurrent Event  Following Event  Preceding Event  Immediately Following Event  Immediately Preceding Event |

## Pellet Reasoner



Gambar ‎2.6 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 ‎2.6menunjukkan 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].

## EasyRDF

EasyRDF digunakan agar RDF dengan mudah memproduksi dan mengkonsumsi data yang telah disimpan. EasyRDF adalah sebuah *library* algoritma yang ditulis dalam bahasa PHP berbasis *Object Oriented Programming*. *Library* ini memudahkan *kinerja* query SPARQL yang ditulis melalui HTTP menjadi *triplestore* menggunakan *class* EasyRdf Sparql Client untuk mengembalikan objek PHP sebagai hasil [22].

Terdapat bermacam-macam *method* yang dapat dipanggil dari *library* EasyRDF untuk menuliskan *query* sesuai fungsi yang diinginkan. Berdasarkan dokumentasi dari *official website* EasyRDF [23], Tabel ‎2.6 berikut akan menjelaskan beberapa *method* yang digunakan pada pengerjaan tugas akhir ini beserta fungsi yang dimilikinya.

Tabel ‎2.6 EasyRDF Methods

| Method | Fungsi |
| --- | --- |
| EasyRdf\_Graph() | Referensi API yang mendukung penguraian data RDF menjadi *graph* |
| EasyRdf\_Namespace::setDefault() | Mendeskripsikan namespace yang digunakan |
| parseFile() | Menguraikan suatu data RDF menjadi objek *graph* |
| get() | Mendapatkan sebuah nilai dari *property* suatu *resource* |
| all() | Mendapatkan semua nilai dari *property* |
| allOfType() | Mendapatkan semua *Resource* dalam suatu *graph* dengan tipe atau *class* tertentu |
| Resource() | Mendapatkan atau membuat suatu *resource* yang tersimpan dalam *graph* |

Berikut merupakan contoh penggunaan *library* EasyRDF untuk mendapatkan semua *Resource* dengan tipe ‘Hero’ dari *namespace* ihero yang sudah menjadi *graph*.

|  |
| --- |
| $graph->allOfType('ihero:Hero') |

Tanda “:” digunakan sebagai penghubung antara *namespace* dan *Resource* yang berkaitan. Sedangkan, sebuah nilai yang didapatkan ditandai dengan ekspresi “ ‘ ‘ ”.

## PHP

PHP (PHP: *Hypertext Preprocessor*) adalah *open source* bahasa *scripting* yang mempunyai tujuan umum secara khusus untuk pengembangan web [24]. PHP juga dapat ditanamkan ke dalam HTML. PHP tidak seperti bahasa C untuk pengembangan w*eb*. Namun struktur sintaks dasarnya sama, sehingga fleksibel dan mudah untuk diimplementasikan [25].

PHP adalah bahasa pemrograman dengan berbagai fitur yang komperehensif dan juga mendukung orientasi objek. Saat ini, PHP lebih sering disebut sebagai bahasa pemrograman dinamis. Berbeda bahasa pemrograman lainnya yang bersifat tradisional seperti C / C++, kompilasi tidak diperlukan oleh PHP. Salah satu kelebihan yang dimilik oleh PHP adalah kebebasan *platform*. PHP *compatible* untuk berbagai sistem operasi seperti Linux (untuk berbagai arsitektur CPU), Miscrosoft Windows, Mac OS X, Sun Solaris (SPARC dan Intel), IBM AIX, UX-HP, FreeBSD, Novell Netware, SGI, IRIX, IBM AS/400, OS/2 dan RISC OS. Keuntungan lain yang diberikan PHP adalah fleksibilitas. *Bug* yang terjadi dapat dengan mudah dirubah atau diperbaiki dalam beberapa menit karena tidak diperlukan proses kompilasi dan mampu menciptakan versi baru dari program secara bertahap [26].

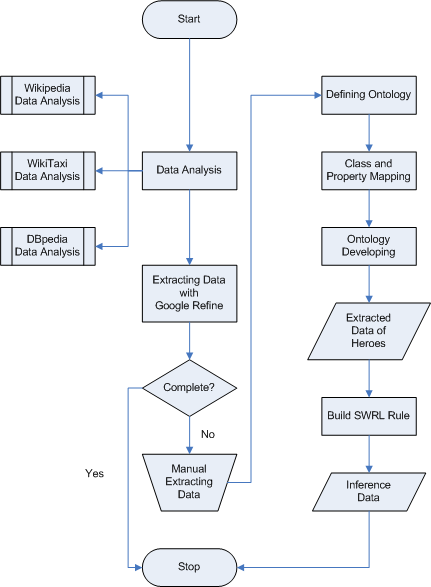
Untuk berhubungan dengan *web browser*, PHP memiliki *method* yang dapat dipanggil sesuai fungsinya. Terdapat dua cara yang digunakan oleh *browser client* untuk mengirimkan informasi pada *web server*, yaitu GET *method* dan POST *method*. Sebelum browser mengirim informasi, *browser* mengkonversi informasi tersebut menggunakan sebuah skema yang disebut URL *encoding*. GET *method* mengirimkan informasi pengguna yang telah dikodekan untuk ditambahkan pada *page request*. Halaman dan kode informasi dipisah oleh karakter "?". Sedang POST method memindahkan informasi melalui header HTTP. Kode informasi yang diberikan oleh GET *method* kemudian dimasukkan ke dalam *header* yang disebut QUERY\_STRING. Selain itu, pengguna juga dapat menyertakan isi dari sebuah berkas PHP ke dalam berkas PHP lain sebelum *server* mengeksekusinya. Salah satunya adalah fungsi require() yang mengambil semua teks dalam *file* tertentu dan menyalinnya dalam *file* yang menggunakan fungsi include() [27]. Tabel ‎2.7 berikut akan menjelaskan beberapa *method* yang digunakan dalam implementasi pengerjaan tugas akhir ini.

Tabel ‎2.7 Method pada PHP

| Method | Fungsi |
| --- | --- |
| isset | Mengecek ada tidaknya suatu variabel |
| echo | Menampilkan string |
| GET | Mengirimkan nilai variabel ke halaman lain |
| if | Mendeskripsikan logika untuk kondisi data |
| foreach as | Perulangan yang digunakan untuk array |
| str\_replace | Mengganti karakter/ substring tertentu dalam suatu string |
| urlencode | Jika ada string yang ingin dikirim atau digabungkan ke penulisan alamat *website* |

# BAB III METODOLOGI PEMECAHAN MASALAH

Pada bab ini dijelaskan mengenai langkah-langkah yang dilakukan untuk mencari relasi antar *person*. Mulai dari metode yang dilakukan untuk menggabungkan beberapa ontologi hingga pengembangan ontologi yang telah digabungkan.



Gambar ‎3.1 Flowchart Pengembangan Ontologi

Penggabungan dan pengembangan ontologi bertujuan untuk menghasilkan suatu ontologi yang dapat digunakan sebagai penentu jenis hubungan antar *person*, apakah terdapat hubungan dalam ikatan keluarga, pertemanan, atau terlibat dalam *event* yang sama. Alur pemecahan masalah dapat dilihat pada Gambar ‎3.1. Pemecahaan masalah dimulai dengan menganalisis data Wikipedia, WikiTaxi, dan DBpedia. Setelah analisis didapatkan, data kemudian diekstrak agar dapat digunakan oleh ontologi dengan mudah. Proses ekstraksi data menggunakan Google Refine gagal karena kekurangan yang dimiliki DBpedia. Oleh karena itu, ekstraksi data dilakukan secara manual. Proses selanjutnya adalah menentukan ontologi yang akan digunakan, selanjutnya diikuti proses pemetaan kelas dan properti dari masing-masing ontologi. Setelah pemetaan selesai, ontologi sudah tergabung menjadi satu ontologi dan kemudian dikembangkan agar dapat memecahkan permasalahan. Data hasil ekstraksi pada proses sebelumnya kemudian diinput ke dalam ontologi. Untuk mengetahui data tersembunyi, diterapkan sejumlah *rule* untuk mendapatkan data *inference* yang merupakan fakta-fakta baru hasil pengetahuan. Deskripsi lebih detail tentang setiap proses akan dijelaskan lebih detail pada subbab bab ini.

## Analisis Data

Untuk memecahkan masalah pencarian relasi antar 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 tokoh bersejarah dan Pahlawan Nasional Indonesia. Daftar Pahlawan Nasional Indonesia diperoleh dari Wikipedia Indonesia yang dapat diakses pada halaman [*https://id.wikipedia.org/wiki/Daftar\_Pahlawan\_Nasional\_Indonesia*](https://id.wikipedia.org/wiki/Daftar_Pahlawan_Nasional_Indonesia). Namun, yang menjadi objek utama untuk dijadikan data adalah Pahlawan Nasional Indonesia. *Person* yang dikatakan sebagai pahlawan pasti mendapatkan gelar karena jasa yang pernah dilakukan semasa hidupnya. *Person* yang digunakan misalnya adalah Sukarno, Cut Nyak Dien, dan Hasyim Asy’ari. Setiap *person* memiliki biografi terkait dengan *place* (misalnya *BirthPlace* dan *DeathPlace*), *time* (misalnya *BirthDate* dan *DeathDate*), dan *event* (misalnya kegiatan politik, peperangan, dan kegiatan belajar mengajar).

*Person* yang digunakan sebagai data adalah *person* yang dinilai memiliki banyak keterkaitan dengan *person* lain. Di sisi lain, ternyata profesi menentukan perbedaan jenis informasi yang dimiliki oleh *person*. Misal, jika *person* berperan aktif dalam aktivitas sebelum kemerdekaan maka *person* tersebut memiliki data tentang peperangan. Sedangkan *person* yang berperan aktif sebagai tokoh agama memiliki data tentang kegiatan belajar mengajar dari mana ilmunya diperoleh dan diajarkan. Berdasarkan daftar Pahlawan Nasional Indonesia yang dimiliki oleh Wikipedia dan perbedaan tersebut, profesi *person* diklasifikasikan menjadi 10 bagian yaitu aktivis kemerdekaan, nasionalis, politisi, pengajar, komposer/ musisi, penyair, pengusaha, tokoh agama, penulis, dan jurnalis. Dikarenakan 10 jenis profesi adalah ruang lingkup yang sangat luas, maka *person* yang digunakan terbatas hanya untuk person yang memiliki profesi atau terkenal sebagai aktivis kemerdekaan, politisi, atau tokoh agama.

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 Wikipedia

Terdapat berbagai macam *open data* yang dapat diakses melalui *internet* tanpa berbayar. Pada awalnya, data diekstrak secara manual dari halaman Wikipedia Indonesia. Sebagai contoh, Gambar ‎3.2 berikut adalah informasi yang dimiliki Wikipedia Indonesia tentang Sukarno. Informasi tersebut dapat diakses melalui alamat [*https://id.wikipedia.org/wiki/Soekarno*](https://id.wikipedia.org/wiki/Soekarno)*.* Melalui data *abstract* dan tabel yang ditampilkan oleh Wikipedia Indonesia, data diekstrak menjadi susunan informasi seperti pada Tabel ‎3.1.



Gambar ‎3.2 Halaman Wikipedia Indonesia tentang Sukarno

Tabel ‎3.1 Data hasil ekstrak Sukarno

|  |  |
| --- | --- |
| Nama | Soekarno |
| Jenis Kelamin | Laki-laki |
| Gelar | Dr.(H.C.) Ir. H. |
| Terkenal Sebagai | Presiden RI |
| Tempat Lahir | Surabaya |
| Tanggal Lahir | 06/06/1901 |
| Tempat Meninggal | Jakarta |
| Tanggal Meninggal | 21/06/1970 |
| Ayah | Raden Soekemi Sosrodihardjo |
| Ibu | Ida Ayu Nyoman Rai |
| Istri | Oetari  Inggit Garnasih  Fatmawati  Hartini  Kartini Manoppo  Ratna Sari Dewi  Haryati  Yurike Sanger  Heldy Djafar |
| Anak | (Dari Fatmawati)  Guntur  Megawati  Rachmawati  Sukmawati  Guruh |
| (Dari Hartini)  Taufan  Bayu |
| (Dari Kartini Manoppo)  Totok |
| (Dari Ratna Sari Dewi)  Kartika |
| (Dari Haryati)  Ayu |
| Agama | Islam |
| Partai Politik | PNI |
| Perang | Agresi Militer Belanda II |

Hasil ekstrak manual data dari halaman Wikipedia Indonesia seperti dijabarkan pada Tabel ‎3.1 ternyata memunculkan permasalahan baru, yaitu penggunaan *label* atau *property* yang tidak baku dan redundan. Padahal *property* menjadi komponen yang penting dalam perancangan ontologi. Selain itu, informasi yang dimuat oleh Wikipedia adalah informasi tidak terstruktur sehingga tidak menjamin kredibilitas data yang akan digunakan.

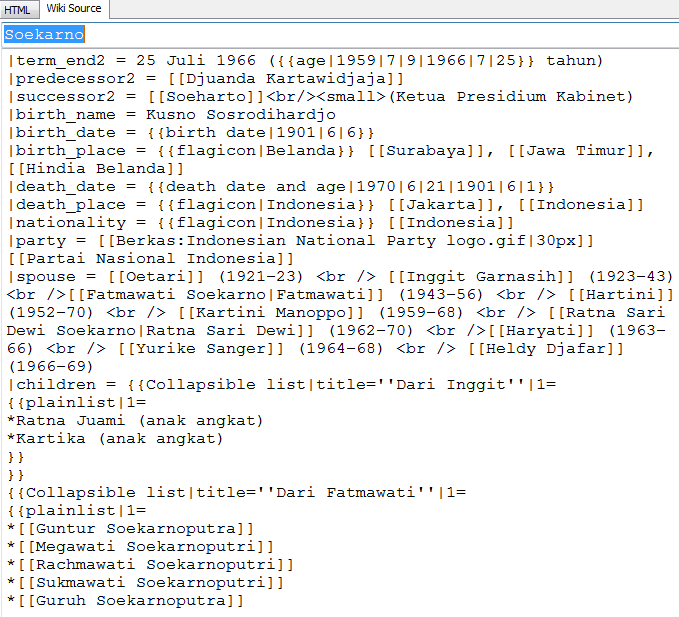
### Analisis Data dari WikiTaxi

Permasalahan baru seperti yang dijelaskan di atas dapat dipecahkan jika halaman Wikipedia dapat diunduh dengan format .XML. XML didesain agar dapat menyimpan data secara ringkas dan mudah diatur. XML mempunyai kata kunci utama berupa data yang jika diolah dapat memberikan informasi. XML dapat digunakan untuk menggambarkan sembarang *view database*, tetapi dengan cara yang sudah terstandarisasi [29]. Data yang disimpan dengan format XML memiliki *property* dengan tata aturan nama yang sudah terstandarisasi. Ada beberapa cara, *tools* dan aplikasi yang menyediakan pengunduhan informasi dari Wikipedia dengan format XML, diantaranya adalah Wikimedia, Kiwix, Aard dictionary, Okawix, dan WikiTaxi. Salah satu *tools* yang mudah digunakan adalah WikiTaxi.



Gambar ‎3.3 WikiTaxi HTML tentang Sukarno

WikiTaxi adalah sebuah aplikasi untuk mengakses halaman HTML Wikipedia secara *offline* dan menyediakan *wiki source* yang memuat *property* dari *file* XML. WikiTaxi yang digunakan adalah versi 1.3.0. Aplikasi WikiTaxi terpisah dengan database WikiTaxi yang memiliki format .taxi sehingga memerlukan *tools* WikiTaxi importer untuk mengubungkannya. Database WikiTaxi yang diunduh adalah semua data yang terdapat pada Wikipedia Indonesia dengan kapasitas *file* sebesar 326 MB. Gambar ‎3.3 di atas adalah contoh tampilan informasi Wikipedia tentang Sukarno yang diakses melalui WikiTaxi HTML. WikiTaxi juga menyediakan fitur bagi pengguna untuk melihat *wiki source* dari halaman yang dikunjungi seperti pada Gambar ‎3.4.



Gambar ‎3.4 Wiki Source tentang Sukarno

Setelah ditinjau lebih lanjut, fitur *wiki source* yang ingin dimanfaatkan dari WikiTaxi dalam pengerjaan tugas akhir ini ternyata menyusahkan pengguna karena susunannya yang tidak rapi. Susunan informasi dari *wiki source* tersebut sulit dibaca dan tidak bisa diekstrak menjadi tabel yang dapat dengan mudah disalin atau di*import* menjadi format lain. Berdasarkan permasalahan tersebut, WikiTaxi tidak dapat menyelesaikan permasalahan sebelumnya.

### Analisis Data dari DBpedia

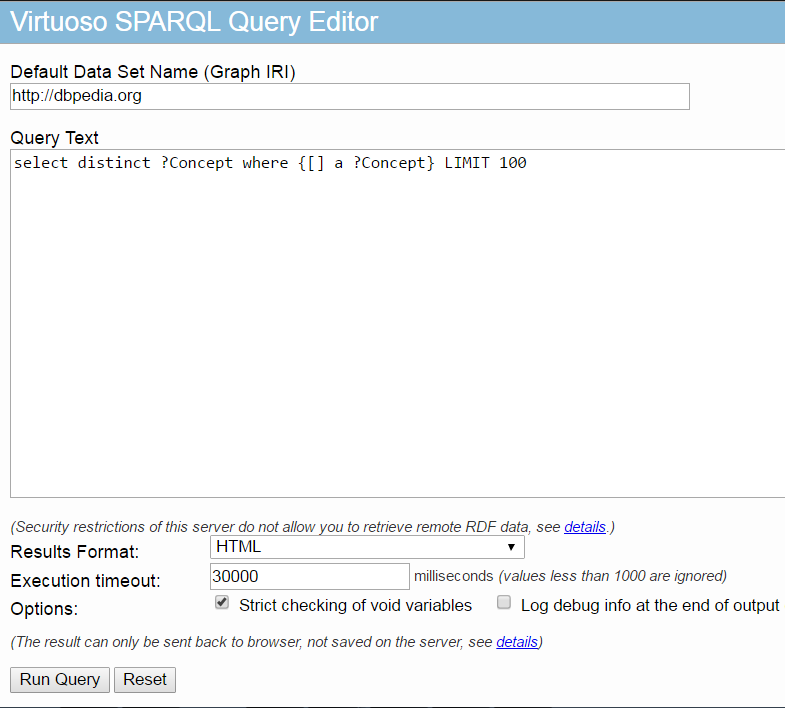
Pada akhirnya, data yang digunakan dalam tugas akhir ini adalah data dari DBpedia. DBpedia memiliki kaitan dengan Wikipedia. Data *abstract* yang dimuat oleh Wikipedia diekstrak oleh DBpedia sehingga menjadi susunan informasi yang terstruktur. Susunan data DBpedia terdiri atas *property* dan *value*. Sebagai contoh, informasi pada *page* DBpedia tentang Sukarno yang diakses melalui alamat [*http://dbpedia.org/page/Sukarno*](http://dbpedia.org/page/Sukarno) dapat dilihat pada Gambar ‎3.5.



Gambar ‎3.5 Halaman DBpedia tentang Sukarno

Berdasarkan semua *property* yang terdapat pada halaman DBpedia, dipilih *property* yang *general* dalam arti mungkin dimiliki oleh semua *person*. Selain itu, dipilih juga *property* yang berkaitan dengan profesi dan *property* yang dapat dijadikan sebagai penentu hubungan antar *person*. Untuk menjadikan informasi tentang *person* sebagai data dalam ontologi, diperlukan *property* dan *value* yang disimpan dalam format khusus agar dapat di*import* ke dalam *tools* perancangan ontologi.

### Ekstraksi Data Menggunakan Google Refine



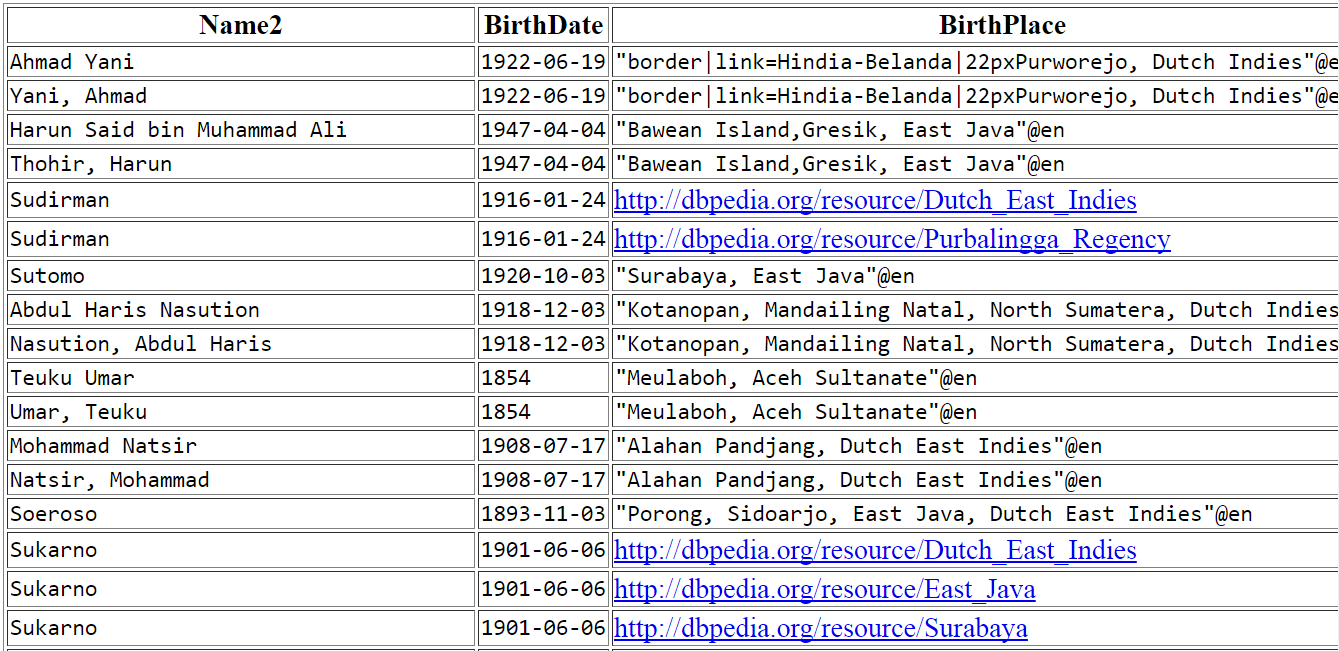
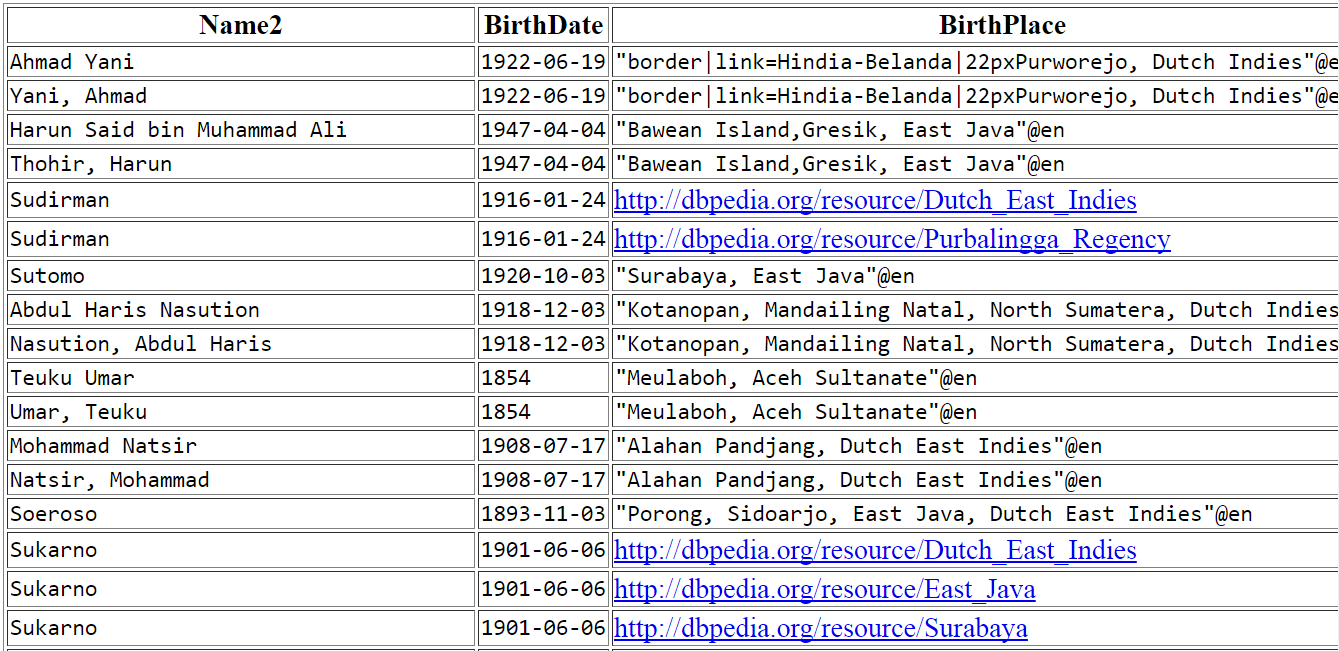
Gambar ‎3.6 Virtuoso SPARQL Query Editor

Awalnya, data yang bersumber dari DBpedia di*export* menjadi format *excel* menggunakan *query* SPARQL. DBpedia sendiri memiliki fitur untuk menjalankan *query* SPARQL yang dapat diakses pada halaman *http://dbpedia.org/sparql*. Gambar ‎3.6 adalah tampilan dari halaman Vituoso SPARQL Query Editor yang dimiliki DBpedia.

Untuk melakukan proses ekstraksi data, diperlukan *query* SPARQL yang berfungsi untuk mengambil *value* dari *property* yang ditentukan. Contoh Kode Sumber ‎3.1 berikut digunakan untuk mengambil *value* dari nama (*Name*), tanggal lahir (*BirthDate*), dan tanggal kematian (*DeathDate*).

|  |
| --- |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX dbpedia: <http://dbpedia.org/resource/>  PREFIX dcterms: <http://purl.org/dc/terms/>  PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>  PREFIX category: <http://dbpedia.org/resource/Category:>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX dbpprop: <http://dbpedia.org/property/>  PREFIX dbprop: <http://dbpedia.org/property/>  PREFIX grs: <http://www.georss.org/georss/>  PREFIX category: <http://dbpedia.org/resource/Category:>  PREFIX dcterms: <http://purl.org/dc/terms/>  PREFIX owl: <http://www.w3.org/2002/07/owl#>  PREFIX freebase: <http://rdf.freebase.com/ns/>  PREFIX db: <http://dbpedia.org/>  PREFIX dbp: <http://dbpedia.org/property/>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX http: <http://www.w3.org/2006/http#>  SELECT DISTINCT ?Name2, $BirthDate, ?BirthPlace  WHERE {  ?president dcterms:subject category:National\_Heroes\_of\_Indonesia.  ?president dbpprop:birthDate ?BirthDate.  ?president dbpprop:birthPlace ?BirthPlace.  ?president dbpprop:name ?Name.  BIND (str(?Name)as ?Name2)  } |

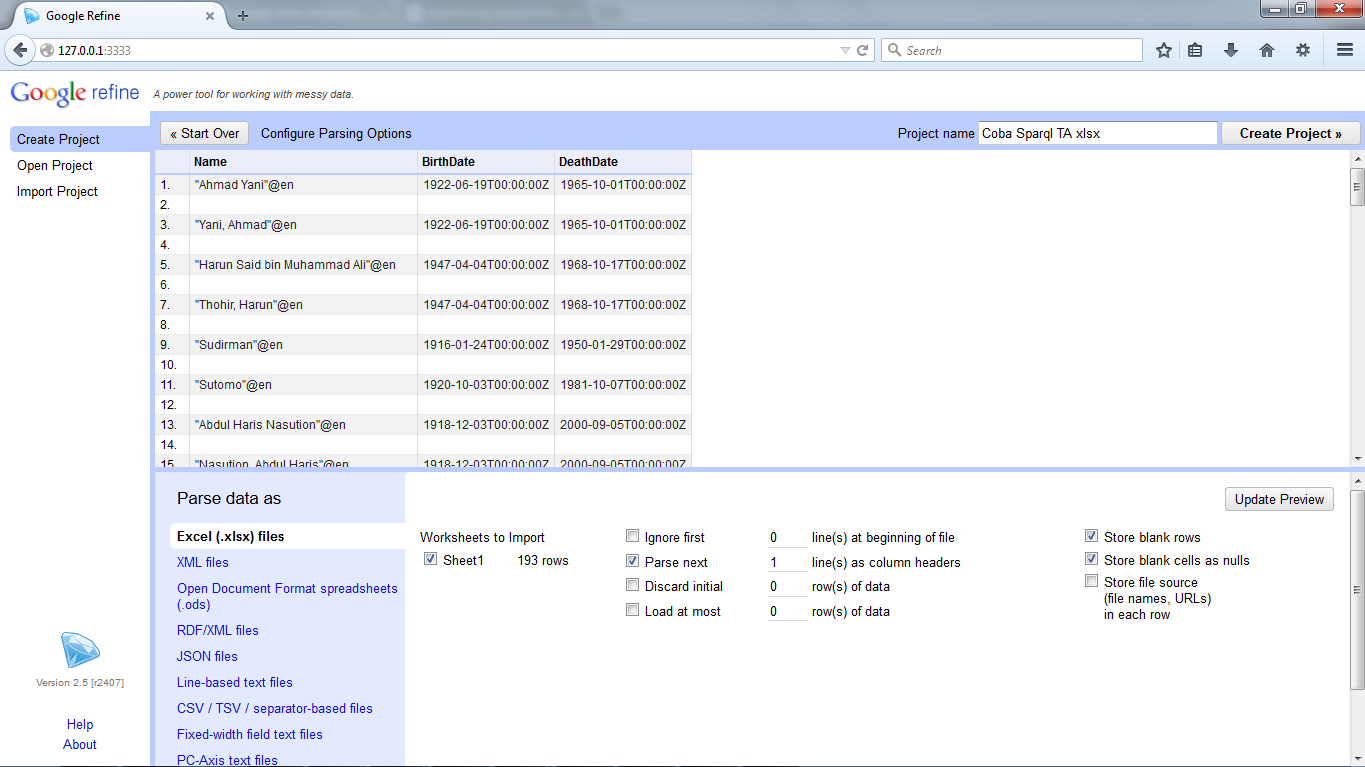
Kode Sumber ‎3.1 Kode SPARQL untuk mendapatkan value Name, BirthDate, dan DeathDate dari DBpedia



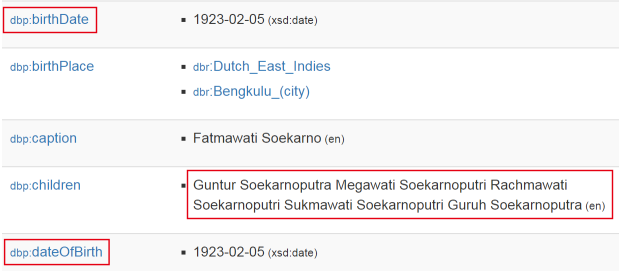
Gambar ‎3.7 Hasil SPARQL Query

Setelah menjalankan kode SPARQL didapatkan hasil seperti pada Gambar ‎3.7 yang kemudian disalin ke dalam Microsoft Excel sehingga data berbentuk file *excel*. Data yang sudah berformat *excel* kemudian di*parsing* menjadi bentuk RDF agar nantinya dapat digunakan ketika merancang ontologi. *Tools* yang digunakan untuk mem*parsing* adalah Google Refine. Google Refine memiliki layanan untuk mengubah data dari berbagai format (TSV, CSV, \*SV, Excel, JSON, XML, RDF sebagai XML, dan Google Data document) menjadi data yang mendukung bahasa ontologi (diantaranya adalah XML, RDF, dan OWL). Gambar ‎3.8 adalah contoh *project* untuk *import* data *excel* yang telah diproses sebelumnya menjadi format RDF. Setelah menjadi RDF, *instance* yang terdapat dalam RDF disalin ke dalam *source* ontologi yang dirancang agar dapat digunakan sebagai data.

Google Refine menjadi salah satu *tools* yang dapat digunakan dengan mudah untuk mem*parsing* data. Namun, DBpedia memiliki beberapa *value* yang redundan. Selain itu, DBpedia juga memiliki beberapa *value* yang *universal* dalam arti beberapa *value* digabung menjadi satu *value* seperti pada Gambar ‎3.9. Hal ini menyulitkan proses pemilihan dan pemanggilan data karena adanya data ganda (yang diblok warna merah). Selain itu, data rangkap (yang diblok warna biru) juga akan dikenali sebagai satu data. Jika memanfaatkan hasil ekstraksi DBpedia menggunakan *tools* di atas, maka akan sulit menemukan fakta baru dari ontologi yang dirancang.



Gambar ‎3.8 Google Refine



Gambar ‎3.9 Data yang redundan dan ambigu

### Ekstraksi Data Berdasar Properti Tertentu

Untuk memecahkan permasalahan di atas, pada akhirnya data diinput secara manual ke dalam *tools* perancangan ontologi. Hal ini bertujuan untuk menghindari adanya data ganda atau rangkap. *Property* yang telah ditentukan akan digunakan untuk membuat klasifikasi data *person*. Contoh klasifikasi data tentang Sukarno dijabarkan pada Tabel ‎3.2.

Tabel ‎3.2 Hasil klasifikasi data tunggal

|  |  |
| --- | --- |
| hasName | **Sukarno** |
| BirthDate | 1901-06-06 |
| BirthPlace | Dutch\_East\_Indies  East\_Java  Blitar  Surabaya |
| DeathDate | 1970-06-21 |
| DeathPlace | Jakarta  Indonesia |
| hasSpouse | Oetari  Kartini\_Manoppo  Hartini  Haryati  Inggit\_Garnasih  Fatmawati  Dewi\_Sukarno  Yurike\_Sanger |
| isParentOf | Megawati\_Sukarnoputri |
| hasVicePresident | Mohammad\_Hatta |
| hasPrimeMinister | Mohammad\_Natsir  Burhanuddin\_Harahap  Wilopo  Soekiman\_Wirjosandjojo  Amir\_Sjarifuddin  Mohammad\_Hatta  Sutan\_Sjahrir  Ali\_Sastroamidjojo  Djuanda\_Kartawidjaja  Abdul\_Halim\_(Indonesia) |

Data yang dijabarkan pada tabel di atas adalah data tunggal *person* Sukarno. Kemudian data diklasifikasikan lebih lanjut agar dapat diketahui hubungan apa yang terjadi antar *person*. Tabel ‎3.3 berikut adalah hasil klasifikasi selanjutnya. Jika dinalar dengan logika, seharusnya kedua *person* tersebut memiliki keterkaitan hubungan keluarga. Apabila dua *person* itu digabungkan, maka akan muncul fakta-fakta baru yang sebelumnya tidak ada. Daftar *person* beserta data hidup yang dimilikinya selengkapnya dapat dilihat pada Lampiran‎8A.

Tabel ‎3.3 Hasil klasifikasi data lebih dari satu person

|  |  |  |
| --- | --- | --- |
| **Sukarno** | name | Sukarno |
| BirthDate | 1901-06-06 |
| BirthPlace | Dutch\_East\_Indies  East\_Java  Blitar  Surabaya |
| DeathDate | 1970-06-21 |
| DeathPlace | Jakarta  Indonesia |
| spouse | Oetari  Kartini\_Manoppo  Hartini  Haryati  Inggit\_Garnasih  Fatmawati  Dewi\_Sukarno  Yurike\_Sanger |
| isParentOf | Megawati\_Sukarnoputri |
| vicePresident | Mohammad\_Hatta |
| primeMinister | Mohammad\_Natsir  Burhanuddin\_Harahap  Wilopo  Soekiman\_Wirjosandjojo  Amir\_Sjarifuddin  Mohammad\_Hatta  Sutan\_Sjahrir  Ali\_Sastroamidjojo  Djuanda\_Kartawidjaja  Abdul\_Halim\_(Indonesia) |
| **Fatmawati** | name | Fatmawati |
| BirthDate | 1923-02-05 |
| BirthPlace | Dutch\_East\_Indies  Bengkulu\_(city) |
| DeathDate | 1980-05-14 |
| DeathPlace | Kuala\_Lumpur |
| spouse | Sukarno |
| children | Guntur Soekarnoputra  Megawati Soekarnoputri  Rachmawati Soekarnoputri  Sukmawati Soekarnoputri  Guruh Soekarnoputra |

*Property* data yang dimiliki DBpedia bermacam-macam. Selain itu, terdapat *property* yang memiliki fungsi sama namun penamaannya berbeda. Untuk itu, pada subbab selanjutnya akan dibahas mengenai penentuan *class* dan *property* dalam perancangan ontologi.

## Perancangan Ontologi

Pada tahap ini, ontologi dibangun dengan menggabungkan beberapa ontologi yang sudah ada. Ontologi yang digunakan adalah *Family Relationships Ontology*, FOAF, dan BIO. *Family Relationships Ontology* adalah ontologi yang mendeskripsikan hubungan dalam ikatan keluarga. FOAF adalah ontologi yang mendeskripsikan hubungan antar *person* dalam ruang lingkup pertemanan. Sedangkan BIOadalah ontologi yang mendeskripsikan hubungan antara *person* dengan *event*. Setelah menggabungkannya, ontologi dikembangkan agar dapat menyimpan informasi yang telah didapat. Jika konsep suatu ontologi dengan ontologi lain berada di domain yang sama mungkin dapat digunakan sebagai kontekstual informasi untuk pengintegrasian ontologi. Oleh karena itu, informasi tambahan di luar ontologi dapat digunakan untuk memperbaiki integrasi ontologi [30].

### Penentuan Ontologi yang Digunakan

Pada tahap awal, terdapat 5 macam ontologi yang akan dimanfaatkan pada pengerjaan tugas akhir ini. Ontologi tersebut adalah *Family Relationships Ontology*, FOAF, BIO*, Biography Light Ontology, dan OWL Time.* Seperti yang telah dijelaskan pada Subbab ‎2.7 *Family Relationships Ontology* berperan sebagai ontologi penentu hubungan dalam ikatan keluarga. *FOAF (Friend of a friend)* beperan sebagai ontologi penentu hubungan antar *person* dalam ruang lingkup pertemanan. BIObereperan sebagai ontologi penentu hubungan antara *person* dengan *event*. Sedangkan ontologi lainnya yaitu *Biography Light Ontology* adalah gabungan *ABC Ontology*, *Linking Open Description of Events ontology* (LODE), *Friend of a Friend* (FOAF), *Bio vocabulary, Relationship vocabulary,* dan *Bibliographic Ontology* (Bibo) yang dibentuk dari ontologi RDF + OWL dalam upaya untuk meningkatkan konektivitas dan interoperabilitas antar standar *encoding* teks biografis yang sudah ada [1]. Satu ontologi lainnya, *OWL Time* adalah ontologi yang mendeskripsikan hubungan dalam ruang lingkup waktu. Waktu dapat berkaitan dengan apa saja termasuk *person* dan *event*.

*Family Relationships Ontology*, *FOAF (Friend of a friend)*, dan BIOmemiliki komponen *class* dan *property* yang sesuai untuk menyelesaikan permasalahan dari topik tugas akhir ini. Sedangkan *Biography Light Ontology* tidak jadi digunakan karena tidak ditemukannya file OWL atau RDF dari ontologi tersebut. Jika mengaitkan *person* dengan *event*, waktu adalah komponen yang menjadi unsur penting di dalamnya. Namun, karena domain utama pengerjaan tugas akhir ini adalah *person*, maka hubungan antar waktu ditiadakan sehingga *OWL Time* tidak jadi digunakan.

Berdasarkan ketiga ontologi yang akan digunakan, BIOdigunakan sebagai landasan pembangunan ontologi yang dikerjakan pada tugas akhir ini. BIOmemiliki komponen paling sesuai untuk pencarian relasi antar *person* yang berhubungan dengan sejarah. Pada dasarnya, BIOsudah memuat komponen yang dimiliki oleh FOAF, namun hanya sebagian kecil komponen FOAF yang digunakan. FOAF tidak dijadikan sebagai landasan pembangunan karena *class* dan *property* yang dimilikinya cocok digunakan untuk *person* yang hidup pada era *digital*. Sedangkan topik tugas akhir ditujukan untuk *person* yang namanya dikenang oleh sejarah. Seperti yang telah diuraikan pada Subbab 2.7. FOAF memiliki *class* dan *property* yang mengarah pada pertemanan di *social media*. Sedangkan *Family Relationships Ontology*  tidak digunakan sebagai landasan pembangunan ontologi karena hanya spesifik menguraikan relasi keluarga.

### Seleksi Kelas dan Properti dari BIO

BIO adalah gabungan beberapa ontologi yang dibangun oleh David Galbraith. Ontologi ini menyediakan *vocabulary* untuk mendeskripsikan informasi biografi manusia, terutama tentang kehidupan dan kematian. Dikarenakan ontologi ini dibuat oleh warga negara Inggris, *class* dan *property* yang digunakan juga mengacu pada adat dan budaya yang berlaku di negara Inggris. Misalnya Baptism, BarMitzvah, dan BasMitzvah.

Budaya dan adat yang ada di Indonesia berbeda dengan budaya dan adat di negara lain. Dikarenakan domain tugas akhir ini adalah *person* yang bersejarah di Indonesia, maka *class* dan *property* BIO yang digunakan adalah yang bersifat umum. *Classes* yang digunakan terurai pada Tabel ‎3.4 dan *object properties* yang digunakan terurai pada Tabel ‎3.5.

Tabel ‎3.4 Seleksi Class dari BIO

| Classes | | |
| --- | --- | --- |
| Agent | URI | *http://xmlns.com/foaf/0.1/Agent* |
| Fungsi | Mendekripsikan *instance* sebagai agen |
| Event | URI | [*http://linkedevents.org/ontology/Event*](http://linkedevents.org/ontology/Event)  [*http://purl.org/vocab/bio/0.1/Event*](http://purl.org/vocab/bio/0.1/Event)  [*http://purl.org/NET/c4dm/event.owl#Event*](http://purl.org/NET/c4dm/event.owl#Event)  [*http://purl.org/dc/dcmitype/Event*](http://purl.org/dc/dcmitype/Event)  [*http://sw.opencyc.org/2009/04/07/concept/*](http://sw.opencyc.org/2009/04/07/concept/) *en/Event* |
| Fungsi | Mendekripsikan *instance* sebagai kejadian/ peristiwa |
| Group Event | URI | *http://purl.org/vocab/bio/0.1/GroupEvent* |
| Fungsi | Mendekripsikan *instance* sebagai kejadian/ peristiwa yang melibatkan lebih dari 1 *person* |
| Marriage | URI | *http://purl.org/vocab/bio/0.1/Marriage* |
| Fungsi | Mendekripsikan *instance* sebagai peristiwa pernikahan |
| Individual Event | URI | [*http://purl.org/vocab/bio/0.1/Individual*](http://purl.org/vocab/bio/0.1/Individual) *Event* |
| Fungsi | Mendekripsikan *instance* sebagai kejadian/ peristiwa yang melibatkan 1 *person* |
| Birth | URI | *http://purl.org/vocab/bio/0.1/Birth* |
| Fungsi | Mendekripsikan *instance* sebagai peristiwa kelahiran |
| Burial | URI | *http://purl.org/vocab/bio/0.1/Burial* |
| Fungsi | Mendekripsikan *instance* sebagai peristiwa pemakaman |
| Death | URI | *http://purl.org/vocab/bio/0.1/Death* |
| Fungsi | Mendekripsikan *instance* sebagai peristiwa kematian |
| Graduation | URI | *http://purl.org/vocab/bio/0.1/Graduation* |
| Fungsi | Mendekripsikan *instance* sebagai peristiwa kelulusan |
| Person | URI | *http://xmlns.com/foaf/0.1/Person* |
| Fungsi | Mendekripsikan *instance* sebagai *person* |
| Wedding  Event\_Generic | URI | [*http://sw.opencyc.org/2009/04/07/concept/*](http://sw.opencyc.org/2009/04/07/concept/) *en/WeddingEvent\_Generic* |
| Fungsi | Mendekripsikan *instance* sebagai peristiwa pernikahan |

Tabel ‎3.5 Seleksi Object Properties dari BIO

| Object Properties | | |
| --- | --- | --- |
| differentFrom | URI | [*http://www.w3.org/2002/07/owl#*](http://www.w3.org/2002/07/owl) *differentFrom* |
| Fungsi | *Node* dari Agent |
| Agent | URI | *http://purl.org/vocab/bio/0.1/agent* |
| Fungsi | Mendeskripsikan *person, group*, atau *organization* yang memiliki peran dalam suatu *event* |
| Partner | URI | *http://purl.org/vocab/bio/0.1/partner* |
| Fungsi | Seseorang yang terlibat dalam sebuah *event* sebagai mitra dalam suatu hubungan. |
| Principal | URI | *http://purl.org/vocab/bio/0.1/principal* |
| Fungsi | Seseorang yang mempunyai peran utama dan paling penting dalam sebuah *event* |

### Import Kelas dan Properti dari FamilyTree

*Family Relationship Ontology* yang digunakan adalah ontologi dengan URI [*http://www.co-ode.org/roberts/family-tree.owl*](http://www.co-ode.org/roberts/family-tree.owl). Tidak semua *class* dan *property* milik ontologi yang selanjutnya diberi label FamilyTree ini digunakan. Informasi yang didapatkan pada Subbab 3.1 sebagian besar hanya melibatkan *property* terkaitorang tua dan anak. FamilyTree memiliki cakupan relasi yang luas, tidak hanya hubungan orang tua dan anak tetapi juga mendeskripsikan hubungan sampai dengan nenek moyang. Dari berbagai *classes* dan *properties* yang terdapat pada FamilyTree, diambil *classes* yang diuraikan pada Tabel ‎3.6, *object properties* yang diuraikan pada Tabel ‎3.7, dan *data properties* yang diuraikan pada Tabel ‎3.8.

Tabel ‎3.6 Import Classes dari FamilyTree

| Classes | | |
| --- | --- | --- |
| Man | URI | [*http://www.co-ode.org/roberts/family-tree.owl#Man*](http://www.co-ode.org/roberts/family-tree.owl#Man) |
| Fungsi | Mendeskripsikan bahwa *person* adalah seorang pria |
| Woman | URI | *http://www.co-ode.org/roberts/family-tree.owl#Woman* |
| Fungsi | Mendeskripsikan bahwa *person* adalah seorang wanita |
| Sex | URI | *http://www.co-ode.org/roberts/family-tree.owl#Sex* |
| Fungsi | Mendeskripsikan jenis kelamin *person* |
| Male | URI | *http://www.co-ode.org/roberts/family-tree.owl#Male* |
| Fungsi | Mendeskripsikan bahwa *person* memiliki jenis kelamin laki-laki |
| Female | URI | *http://www.co-ode.org/roberts/family-tree.owl#Female* |
| Fungsi | Mendeskripsikan bahwa *person* memiliki jenis kelamin perempuan |

Tabel ‎3.7 Import Object Properties dari FamilyTree

| Object Properties | | |
| --- | --- | --- |
| hasChild | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasChild* |
| Fungsi | Mendeskripsikan relasi mempunyai anak |
| hasSon | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasSon* |
| Fungsi | Mendeskripsikan relasi mempunyai anak laki-laki |
| hasDaughter | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasDaughter* |
| Fungsi | Mendeskripsikan relasi mempunyai anak perempuan |
| hasParent | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasParent* |
| Fungsi | Mendeskripsikan relasi mempunyai orangtua |
| hasFather | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasFather* |
| Fungsi | Mendeskripsikan relasi mempunyai ayah |
| hasMother | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasMother* |
| Fungsi | Mendeskripsikan relasi mempunyai ibu |
| hasHusband | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasChild* |
| Fungsi | Mendeskripsikan relasi mempunyai suami |
| hasWife | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasWife* |
| Fungsi | Mendeskripsikan relasi mempunyai istri |
| isChildOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isChildOf* |
| Fungsi | Mendeskripsikan relasi anak dari |
| isSonOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isSonOf* |
| Fungsi | Mendeskripsikan relasi anak laki-laki dari |
| isDaughterOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isDaughterOf* |
| Fungsi | Mendeskripsikan relasi anak perempuan dari |
| isParentOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isParentOf* |
| Fungsi | Mendeskripsikan relasi orang tua dari |
| isFatherOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isFatherOf* |
| Fungsi | Mendeskripsikan relasi ayah dari |
| isMotherOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isMotherOf* |
| Fungsi | Mendeskripsikan relasi ibu dari |
| inLawOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#inLawOf* |
| Fungsi | Mendeskripsikan relasi mempunyai hubungan dalam hukum dari |
| isSpouseOf | URI | [*http://www.co-ode.org/roberts/family-tree.owl#isSpouseOf*](http://www.co-ode.org/roberts/family-tree.owl#isSpouseOf) |
| Fungsi | Mendeskripsikan relasi suami/ istri dari |
| isHusbandOf | URI | [*http://www.co-ode.org/roberts/family-tree.owl#isHusbandOf*](http://www.co-ode.org/roberts/family-tree.owl#isHusbandOf) |
| Fungsi | Mendeskripsikan relasi suami dari |
| isWifeOf | URI | *http://www.co-ode.org/roberts/family-tree.owl#isWifeOf* |
| Fungsi | Mendeskripsikan relasi istri dari |

Tabel ‎3.8 Import Data Properties dari FamilyTree

| Data Properties | | |
| --- | --- | --- |
| hasName | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasName* |
| Fungsi | Mendeskripsikan nama *person* |
| hasFirst GivenName | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasFirstGivenName* |
| Fungsi | Mendeskripsikan nama pertama atau nama pemberian/ lahir *person* |
| hasFamily Name | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasFamilyName* |
| Fungsi | Mendeskripsikan nama belakang/ keluarga *person* |
| knownAs | URI | *http://www.co-ode.org/roberts/family-tree.owl#knownAs* |
| Fungsi | Mendeskripsikan *person* dikenal sebagai |
| also KnownAs | URI | *http://www.co-ode.org/roberts/family-tree.owl#alsoKnownAs* |
| Fungsi | Mendeskripsikan *person* juga dikenal sebagai |
| formerly KnownAs | URI | *http://www.co-ode.org/roberts/family-tree.owl#formerlyKnownAs* |
| Fungsi | Mendeskripsikan *person* lebih akrab dikenal sebagai |
| hasEvent Year | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasEventYear* |
| Fungsi | Mendeskripsikan tahun dari sebuah *event* |
| hasBirthYear | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasBirthYear* |
| Fungsi | Mendeskripsikan tahun kelahiran dari *person* |
| hasDeath Year | URI | *http://www.co-ode.org/roberts/family-tree.owl#hasDeathYear* |
| Fungsi | Mendeskripsikan tahun kematian dari *person* |

### Import Kelas dan Properti dari FOAF

FOAF memiliki URI *http://xmlns.com/foaf/0.1/*. FOAF adalah sebuah ontologi yang mendeskripsikan hubungan pertemanan. Seperti yang telah dijelaskan pada Subbab ‎3.2.2 sebagian *class* dan *property* FOAF sudah digunakan dalam BIO*.*

Untuk mendukung pembangunan ontologi pada tugas akhir ini, beberapa *class* milik FOAF yang belum ada pada BIOditambahkan. *Classes* yang diambil diuraikan pada Tabel ‎3.9.

Tabel ‎3.9 Import Classes dari FOAF

| Classes | | |
| --- | --- | --- |
| Group | URI | *http://xmlns.com/foaf/0.1/Group* |
| Fungsi | Mendeskripsikan individu sebagai grup |
| Organization | URI | *http://xmlns.com/foaf/0.1/Organization* |
| Fungsi | Mendeskripsikan individu sebagai organisasi |

### Equivalent Class dan Property

Terdapat kelas dan properti yang memiliki kesamaan (*equivalent*). Kelas yang dimaksud adalah kelas dari BIO. Sedangkan properti yang dimaksud adalah properti dariFamilyTree. Sifat *equivalent* tersebut merupakan sifat turunan dari ontologi asalnya. Tabel ‎3.10 berikut menguraikan kelas yang memiliki sifat *equivalent* dengan kelas lainnya. Sedangkan Tabel ‎3.11berikut menguraikan properti yang memiliki sifat *equivalent* dengan properti lainnya.

Tabel ‎3.10 Equivalent classes dari BIO

|  |  |
| --- | --- |
| BIO | |
| Marriage | WeddingEvent\_Generic |

Tabel ‎3.11 Equivalent properties dari FamilyTree

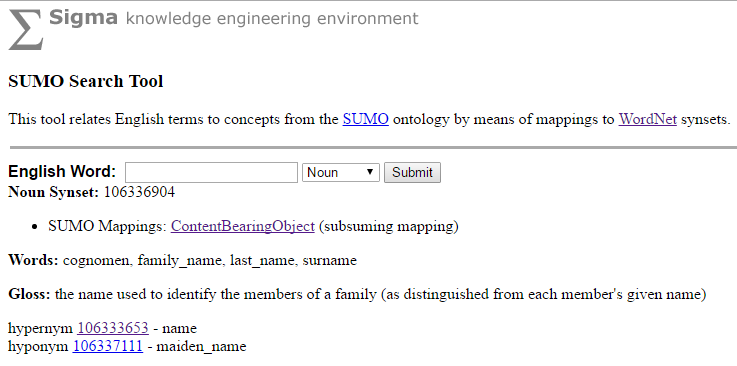
|  |  |
| --- | --- |
| FamilyTree | |
| hasChild | isParentOf |
| hasParent | isChildOf |

Selain sifat *equivalent* yang diturunkan dari ontologi asalnya, analisis dataset dari DBpedia yang dilakukan pada tahap sebelumnya menghasilkan beberapa properti yang memiliki nama berbeda tetapi mempunyai fungsi sama. Properti tersebut nantinya akan digunakan dalam tahap pengembangan ontologi. Properti yang dimiliki DBpedia menggunakan penamaan properti yang sudah terstandarisasi dari berbagai ontologi. Agar properti yang memiliki fungsi sama dapat dijadikan sebagai satu properti, maka susunan nama properti perlu dicari prioritasnya.

Nama properti mengandung satu atau lebih dari satu kata. Sedangkan sebuah kata memiliki susunan kata berdasarkan ruang lingkupnya. WordNet adalah salah satu kamus kata yang menyediakan fitur pencarian hubungan antar kata. Sebuah kata memiliki *hypernim* (susunan kata yang ada di atasnya), *derivationally related* (kata yang saling berhubungan), dan *hyponim* (susunan kata yang ada di bawahnya). Semakin dekat sebuah kata dengan *hypernim*nya, maka kata tersebut semakin memiliki prioritas yang tinggi atau dapat dikatakan sebagai kata yang lebih umum. Namun WorNnet tidak memberikan informasi mengenai *hypernim* dan *hyponim* dari sebuah kata.

SUMO (*Suggested Upper Merged Ontology*) adalah ontologi yang memiliki layanan pencarian hubungan antar kata berbahasa Inggris. SUMO telah memetakan semua kata yang terdapat dalam WordNet. SUMO dapat digunakan secara online melalui alamat [*http://www.adampease.org/OP/*](http://www.adampease.org/OP/). *Hypernim* dan *hyponim* dari sebuah kata ditampilkan secara rinci dalam ontologi ini. Gambar ‎3.10 adalah contoh hasil pencarian kata *nickname*.

Beberapa properti yang memiliki kemiripan dari DBpedia dan ontologi lainnya kemudian dicari hubungannya dengan SUMO. Daftar properti yang memiliki kemiripan dapat dilihat di Tabel ‎3.12. Selanjutnya, properti yang mirip dibandingkan dengan melihat *hyponim* dan *hypernim*nya. Hasil pencarian hubungan antar kata menggunakan SUMO diuraikan pada Tabel ‎3.13 sampai dengan Tabel ‎3.19.



Gambar ‎3.10 SUMO

Tabel ‎3.12 Daftar properti yang memiliki kemiripan

| No | Ontologi | Properti |
| --- | --- | --- |
| 1 | dbpedia | givenName |
| dbpedia | birthName |
| familyTree | firstGivenName |
| 2 | dbpedia | nickName |
| dbpedia | nick |
| 3 | dbpedia | surname |
| familyTree | familyName |
| 4 | dbpedia | alias |
| dbpedia | alternativeNames |
| 5 | dbpedia | placeOfBurial |
| dbpedia | restingPlace |
| 6 | dbpedia | children |
| familyTree | child |
| 7 | dbpedia | occupation |
| dbpedia | profession |

Tabel ‎3.13 Pencarian daftar kata No. 1 menggunakan SUMO

|  |  |  |  |
| --- | --- | --- | --- |
| Words 1 | dbpedia:  **givenName** | dbpedia:  **birthName** | familyTree:  **firstGivenName** |
| hypernim | name | - | name |
| Kesimpulan :  Kata *given name* dan *first given name* berada pada level yang sama. *First given name* merupakan properti yang sudah ada di *Family Relationships Ontology*. Maka, nama properti yang digunakan adalah *firstGivenName*. | | | |

Tabel ‎3.14 Pencarian daftar kata No. 2 menggunakan SUMO

|  |  |  |
| --- | --- | --- |
| Words 2 | dbpedia:  **nickName** | dbpedia:  **nick** |
| derivationally related | dub,  nickname | nick, snick  chip, nick  notch |
| hypernim 1 | appellation,  appellative,  denomination,  designation | cut |
| hypernim 2 | name | Opening |
| Kesimpulan:  Kata *nickname* dan *nick* berada pada ruang lingkup yang berbeda. Karena properti yang dimaksud terkait dengan nama. Maka, nama properti yang digunakan adalah *nickName*. | | |

Tabel ‎3.15 Pencarian daftar kata No. 3 menggunakan SUMO

|  |  |  |
| --- | --- | --- |
| Words 3 | dbpedia:  **surname** | familyTree:  **familyName** |
| hypernim 1 | name | name |
| Kesimpulan:  Kata *surname* dan *familyName* berada pada level yang sama. *FamilyName* merupakan properti yang sudah ada di *Family Relationships Ontology*. Maka, nama properti yang digunakan adalah *familyName*. | | |

Tabel ‎3.16 Pencarian daftar kata No. 4 menggunakan SUMO

|  |  |  |
| --- | --- | --- |
| Words 4 | dbpedia:  **alias** | dbpedia:  **alternativeNames** |
| hypernim 1 | name | - |
| Kesimpulan:  Karena kata *alternative names* tidak ada dalam kamus kata SUMO. Maka, nama properti yang digunakan adalah alias. | | |

Tabel ‎3.17 Pencarian daftar kata No. 5 menggunakan SUMO

|  |  |  |
| --- | --- | --- |
| Words 5 | dbpedia:  **placeOfBurial** | dbpedia:  **restingPlace** |
| hypernim 1 | - | - |
| Kesimpulan:  Karena kata *place of burial* dan *resting place* tidak ada dalam kamus kata SUMO. Sedangkan properti lainnya yang berkaitan dengan *place* menggunakan penamaan *birth place* dan *death place*. Maka, nama properti yang digunakan adalah *burialPlace*. | | |

Tabel ‎3.18 Pencarian daftar kata No. 6 menggunakan SUMO

|  |  |  |
| --- | --- | --- |
| Words 6 | dbpedia:  **children** | familyTree:  **child** |
| hypernim 1 | - | issue,  offspring,  progeny |
| Kesimpulan:  Karena kata *children* tidak ada dalam kamus kata SUMO. Maka, nama properti yang digunakan adalah *child*. | | |

Tabel ‎3.19 Pencarian daftar kata No. 7 menggunakan SUMO

|  |  |  |
| --- | --- | --- |
| Words 7 | dbpedia:  **occupation** | dbpedia:  **profession** |
| hyponim 1 | profession | learned\_profession,  literature,  architecture,  education,  journalism,  politics,  engineering, technology |
| hypernim 1 | activity | business,  job,  line,  line\_of\_work,  occupation |
| Kesimpulan: | | |
| Karena kata *occupation* berada pada level yang lebih tinggi dalam kamus kata SUMO. Maka, nama properti yang digunakan adalah *occupation*. | | |

### Pengembangan Ontologi

Ontologi yang telah digabungkan hanya mendeskripsikan komponen dasar untuk pencarian relasi antar *person*. Jika *person* memiliki aktivitas hidup dan profesi yang berbeda, maka diperlukan kelas dan properti baru untuk memenuhi pencarian relasi antar person dalam domain sejarah.

Ontologi yang diikerjakan dalam tugas akhir ini diberi nama ihero. Ihero adalah ontologi yang mampu melakukan pencarian relasi antar tokoh bersejarah di negara Indonesia. Ihero memanfaatkan beberapa ontologi yang sudah ada yaitu *Family Relationships Ontology*, *FOAF (Friend of a friend)*, dan BIO. Kelas dan properti baru ditambahkan dalam ontologi ini. Kelas baru yang ditambahkan terurai pada Tabel ‎3.20, properti objek baru yang ditambahkan terurai pada Tabel ‎3.21, properti data baru yang ditambahkan terurai pada Tabel ‎3.22. Kelas baru yang ditambahkan pada dasarnya memiliki fungsi untuk memudahkan implementasi perancangan perangkat lunak. Sedangkan properti yang ditambahkan memiliki penamaan dari daftar properti yang dimiliki oleh DBpedia. Properti yang diambil adalah properti terkait analisis dataset yang dilakukan pada tahap sebelumnya. Jika terdapat properti ganda dengan nama yang berbeda namun memiliki fungsi sama maka dilakukan pencarian prioritas kata dari SUMO seperti yang telah dijelaskan pada Subbab ‎3.2.5.

Tabel ‎3.20 Kelas baru di ihero

| Classes | | |
| --- | --- | --- |
| Hero | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/Hero* |
| Fungsi | Mendeskripsikan bahwa *instance* adalah seseorang dengan gelar pahlawan. |
| Subject | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/Subject* |
| Fungsi | Mendeskripsikan *instance* sebagai subjek. |

Tabel ‎3.21 Properti objek baru di ihero

| Object Properties | | |
| --- | --- | --- |
| hasAllegiance | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasAllegiance* |
| Fungsi | Mendeskripsikan relasi tentang kesetiaan pada grup atau organiasi |
| hasBattle | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasBattle* |
| Fungsi | Mendeskripsikan relasi tentang peperangan |
| hasChairman | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasChairman* |
| Fungsi | Mendeskripsikan relasi dengan ketua grup atau organisasi |
| hasCommander | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasCommander* |
| Fungsi | Mendeskripsikan relasi dengan panglima |
| hasEnemy | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasEnemy* |
| Fungsi | Mendeskripsikan relasi dengan musuh |
| hasGovernment Head | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasGovernmentHead* |
| Fungsi | Mendeskripsikan relasi dengan kepala pemerintahan |
| hasKnown People | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasKnownPeople* |
| Fungsi | Mendeskripsikan relasi dengan *person* yang terlibat |
| hasLeader | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasLeader* |
| Fungsi | Mendeskripsikan relasi dengan pemimpin grup, organisasi, atau *event* |
| hasMember | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasMember* |
| Fungsi | Mendeskripsikan relasi dengan anggota |
| hasMilitary Branch | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/*  *hasMilitaryBranch* |
| Fungsi | Mendeskripsikan relasi dengan *branch* militer |
| hasMilitary Command | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/*  *hasMilitaryCommand* |
| Fungsi | Mendeskripsikan relasi dengan komandan militer |
| hasOffice | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasOffice* |
| Fungsi | Mendeskripsikan relasi dengan pekerjaan |
| hasParty | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasParty* |
| Fungsi | Mendeskripsikan relasi dengan partai |
| hasPredecessor | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasPredecessor* |
| Fungsi | Mendeskripsikan relasi dengan pendahulu |
| hasPresident | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasPresident* |
| Fungsi | Mendeskripsikan relasi dengan presiden |
| hasPrime Minister | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasPrimeMinister* |
| Fungsi | Mendeskripsikan relasi perdana menteri |
| hasStudent | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasStudent* |
| Fungsi | Mendeskripsikan relasi dengan murid |
| hasSubject | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasSubject* |
| Fungsi | Mendeskripsikan relasi dengan subjek kejadian yang pernah dialami *person* |
| hasSuccessor | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasSuccessor* |
| Fungsi | Mendeskripsikan relasi dengan penerus |
| hasTeacher | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasTeacher* |
| Fungsi | Mendeskripsikan relasi dengan guru |
| hasVice President | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasVicePresident* |
| Fungsi | Mendeskripsikan relasi dengan wakil presiden |
| isAllyOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isAllyOf* |
| Fungsi | Mendeskripsikan relasi dengan wakil sekutu |
| isAllegianceOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isAllegianceOf* |
| Fungsi | Mendeskripsikan relasi tentang kesetiaan pada grup atau organiasi yang melibatkan *person* |
| isBattleOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isBattleOf* |
| Fungsi | Mendeskripsikan relasi peperangan yang melibatkan *person* |
| isChairmanOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isChairmanOf* |
| Fungsi | Mendeskripsikan relasi dengan ketua grup atau organisasi yang melibatkan *person* |
| isCommanderOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isCommanderOf* |
| Fungsi | Mendeskripsikan relasi dengan panglima yang melibatkan *person* |
| isEnemyOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isEnemyOf* |
| Fungsi | Mendeskripsikan relasi dengan musuh |
| isGovernment HeadOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isGovernmentHeadOf* |
| Fungsi | Mendeskripsikan relasi dengan kepala pemerintahan yang melibatkan *person* |
| isDeputyOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isDeputyOf* |
| Fungsi | Mendeskripsikan relasi dengan wakil grup atau organisasi yang melibatkan *person* |
| isInauguralOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isInauguralOf* |
| Fungsi | Mendeskripsikan relasi dengan pelantikan yang melibatkan *person* |
| isInfluencedOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isInfluencedOf* |
| Fungsi | Mendeskripsikan relasi tentang *person* yang berpengaruh pada grup, organisasi, atau *event* |
| isLeaderOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isLeaderOf* |
| Fungsi | Mendeskripsikan relasi dengan pemimpin grup, organisasi, atau *event* yang melibatkan *person* |
| isMilitary BranchOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isMilitaryBranchOf* |
| Fungsi | Mendeskripsikan relasi dengan branch militer yang melibatkan *person* |
| isMilitary CommandOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isMilitaryCommandOf* |
| Fungsi | Mendeskripsikan relasi dengan komandan militer yang melibatkan *person* |
| isMinisterOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isMinisterOf* |
| Fungsi | Mendeskripsikan relasi dengan menteri yang melibatkan *person* |
| isOfficeOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isOfficeOf* |
| Fungsi | Mendeskripsikan relasi dengan pekerjaan yang melibatkan *person* |
| isPartyOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isPartyOf* |
| Fungsi | Mendeskripsikan relasi dengan partai yang melibatkan *person* |
| isPersonOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isPersonOf* |
| Fungsi | Mendeskripsikan person yang pernah terlibat dalam grup, organisasi, atau *event* |
| isPredecessorOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isPredecessorOf* |
| Fungsi | Mendeskripsikan relasi dengan pendahulu yang melibatkan *person* |
| isPrime MinisterOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ isPrimeMinisterOf* |
| Fungsi | Mendeskripsikan relasi perdana menteri yang melibatkan *person* |
| isSubjectOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isSubjectOf* |
| Fungsi | Mendeskripsikan relasi dengan subjek kejadian yang pernah melibatkan person |
| isSuccessorOf | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/isSuccessorOf* |
| Fungsi | Mendeskripsikan relasi dengan penerus yang melibatkan *person* |
| knownFor | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/knownFor* |
| Fungsi | Mendeskripsikan relasi *person* yang terkenal dalam suatu *event* |

Tabel ‎3.22 Properti data baru di ihero

| Data Properties | | |
| --- | --- | --- |
| almamater | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/almamater* |
| Fungsi | Mendeskripsikan almamater yang dimiliki *person* |
| hasCountChild | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasCountChild* |
| Fungsi | Mendeskripsikan jumlah anak yang dimiliki *person* |
| hasEventDate | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasEventDate* |
| Fungsi | Mendeskripsikan tanggal kejadian/ peristiwa yang pernah dialami *person* |
| hasBirthDate | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasBirthDate* |
| Fungsi | Mendeskripsikan tanggal kelahiran *person* |
| hasDeathDate | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasDeathDate* |
| Fungsi | Mendeskripsikan tanggal kematian *person* |
| hasEventPlace | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasEventPlace* |
| Fungsi | Mendeskripsikan tempat kejadian/ peristiwa yang pernah dialami *person* |
| hasBirthPlace | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasBirthPlace* |
| Fungsi | Mendeskripsikan tempat kelahiran *person* |
| hasBurialPlace | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasBurialPlace* |
| Fungsi | Mendeskripsikan tempat pemakaman *person* |
| hasDeathPlace | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasDeathPlace* |
| Fungsi | Mendeskripsikan tempat kematian *person* |
| hasHonorific Prefix | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/ hasHonorificPrefix* |
| Fungsi | Mendeskripsikan gelar nama yang dimiliki *person* |
| hasNationality | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasNationality* |
| Fungsi | Mendeskripsikan nasionalitas yang dimiliki *person* |
| hasOccupation | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasOccupation* |
| Fungsi | Mendeskripsikan pekerjaan/ jabatan yang dimiliki *person* |
| hasRank | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasRank* |
| Fungsi | Mendeskripsikan pangkat yang dimiliki *person* |
| hasReligion | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasReligion* |
| Fungsi | Mendeskripsikan agama yang dimiliki *person* |
| name | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/name* |
| Fungsi | Mendeskripsikan nama *person* |
| hasBirthName | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasBirthName* |
| Fungsi | Mendeskripsikan nama yang diberikan kepada *person* ketika lahir |
| hasNickName | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/hasNickName* |
| Fungsi | Mendeskripsikan nama pendek *person* |
| alias | URI | [*http://www.semanticweb.org/asus/*](http://www.semanticweb.org/asus/) *ontologies/2016/3/ihero/alias* |
| Fungsi | Mendeskripsikan nama lain *person* |

## Perancangan Rules

Setelah kelas, properti, dan *instances* dibangun, pada tahap ini dilakukan perancangan *rules* menggunakan SWRL (Semantic Web Rule Language). *Rules* dirancang dengan tujuan untuk mengetahui relasi yang terkait antar person. Hasil yang diharapkan dari perancangan *rules* ini adalah munculnya fakta-fakta baru terkait relasi antar *person* yang sebelumnya belum ada di DBpedia. Penulisan *query* pada *rules* mengacu pada kelas, properti objek, dan properti data yang terdapat pada ihero. Aturan penulisan SWRL sudah dijelaskan pada Subbab 3.5.

Relasi yang ingin dicari dalam pengerjaan tugas akhir ini adalah hubungan terkait kekeluargaan, politik, peperangan, dan kegiatan belajar mengajar. Hubungan tersebut adalah *isSpouseOf, hasHusband, hasWife, hasFather, hasMother, hasChild, hasSon, hasDaughter, hasSibling, hasBrother, hasSister, hasStepParent, hasStepFather, hasStepMother, hasStepSon, hasStepDaughter, hasStepSibling, hasStepBrother, hasStepSister, hasBirthDate, hasBirthPlace, hasDeathDate, hasDeathPlace, hasBurialPlace, almamater, isAllyOf,* dan *isEnemyOf.*

### Relasi Keluarga dalam Hukum

Relasi dalam hukum dicari untuk mengetahui apakah antar *person* memiliki hubungan dalam hukum. Hubungan yang dimaksud meliputi *isSpouseOf, hasHusband,* dan *hasWife*.

*Rule* 1 merupakan langkah dasar untuk mencari relasi *isSpouseOf*. Teknik pencarian dilakukan dengan mengidentifikasi kelas *Marriage*. Kelas *Marriage* memiliki *Partner* minimal 2 *Person*. Di dalam *instance* ‘0XX\_MarriageX’ terdapat *Partner* yang berjenis *Male* (Laki-laki) dan *Female* (Perempuan). Arti dari *rule* 1 yaitu apabila kelas Marriage(?m) memiliki *Partner* Male(?sm) dan kelas Marriage(?m) memiliki *Partner* Female(?sf) maka dihasilkan bahwa Male(?sm) adalah *isSpouseOf* Female(?sf).

|  |
| --- |
| ***Rule* 1:**  Marriage(?m), Female(?sf), Male(?sm), Partner(?m, ?sf), Partner(?m, ?sm) -> isSpouseOf(?sm, ?sf) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Marriage6 memiliki *Partner* Sukarno sebagai *Male*.
2. Marriage6 memiliki *Partner* Fatmawati sebagai Fem*ale*.
3. Dapat disimpulkan bahwa Sukarno adalah *spouse* dari Fatmawati.

*Rule* 2 merupakan langkah selanjutnya untuk mencari relasi *hasHusband*. Teknik pencarian dilakukan dengan mengidentifikasi kelas *Marriage*. Kelas *Marriage* memiliki *Partner* minimal 2 *Person*. Di dalam *instance* ‘0XX\_MarriageX’ terdapat *Partner* yang berjenis *Male* (Laki-laki) dan *Female* (Perempuan). Arti dari *rule* 2 yaitu apabila kelas Marriage(?m) memiliki *Partner* Male(?sm) dan kelas Marriage(?m) memiliki *Partner* Female(?sf) maka dihasilkan bahwa Female(?sf) memiliki suami Male(?sm).

|  |
| --- |
| ***Rule* 2:**  Marriage(?m), Female(?sf), Male(?sm), Partner(?m, ?sf), Partner(?m, ?sm) -> hasHusband(?sf, ?sm) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Marriage6 memiliki *Partner* Sukarno sebagai *Male*.
2. Marriage6 memiliki *Partner* Fatmawati sebagai Fem*ale*.
3. Dapat disimpulkan bahwa Fatmawati mempunyai suami Sukarno.

*Rule* 3 merupakan langkah selanjutnya untuk mencari relasi *hasWife*. Teknik pencarian dilakukan dengan mengidentifikasi kelas *Marriage*. Kelas *Marriage* memiliki *Partner* minimal 2 *Person*. Di dalam *instance* ‘0XX\_MarriageX’ terdapat *Partner* yang berjenis *Male* (Laki-laki) dan *Female* (Perempuan). Arti dari *rule* 3 yaitu apabila kelas Marriage(?m) memiliki *Partner* Male(?sm) dan kelas Marriage(?m) memiliki *Partner* Female(?sf) maka dihasilkan bahwa Male(?sm) memiliki istri Female(?sf).

|  |
| --- |
| ***Rule* 3:**  Marriage(?m), Female(?sf), Male(?sm), Partner(?m, ?sf), Partner(?m, ?sm) -> hasWife(?sm, ?sf) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Marriage6 memiliki *Partner* Sukarno sebagai *Male*.
2. Marriage6 memiliki *Partner* Fatmawati sebagai Fem*ale*.
3. Dapat disimpulkan bahwa Sukarno mempunyai istri Fatmawati.

### Relasi Keluarga Sedarah

Relasi sedarah dicari untuk mengetahui apakah antar *person* memiliki relasi terkait hubungan kandung. Hubungan yang dimaksud meliputi *hasFather, hasMother, hasChild, hasSon, hasDaughter, hasSibling, hasBrother,* dan *hasSister.*

*Rule* 4 merupakan langkah untuk mencari relasi *hasFather*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Kelas *Sex* memiliki subkelas *Male* dan *Female*. Masing-masing *instance* yang memiliki kelas *Person* memiliki kelas *Male* (Laki-laki) atau *Female* (Perempuan). Arti dari *rule* 4 yaitu apabila Person(?p) memiliki orangtua bertipe Male(?g) maka dihasilkan bahwa Person(?p) memiliki ayah Male(?g).

|  |
| --- |
| ***Rule* 4:**  Male(?g), Person(?p), hasParent(?p, ?g) -> hasFather(?p, ?g) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Megawati Sukarnoputri mempunyai orang tua yaitu Sukarno yang bertipe *Male*.
2. Dapat disimpulkan bahwa Megawati Sukarnoputri mempunyai ayah yaitu Sukarno.

*Rule* 5 merupakan langkah untuk mencari relasi *hasMother*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 5 yaitu apabila Person(?p) memiliki orangtua bertipe Female(?g) maka dihasilkan bahwa Person(?p) memiliki ibu Female(?g).

|  |
| --- |
| ***Rule* 5:**  Female(?g), Person(?p), hasParent(?p, ?g) -> hasMother(?p, ?g) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Megawati Sukarnoputri mempunyai orang tua yaitu Fatmawati yang bertipe *Female*.
2. Dapat disimpulkan bahwa Megawati Sukarnoputri mempunyai ibu yaitu Fatmawati.

*Rule* 6 merupakan langkah untuk mencari relasi *hasChild*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 6 yaitu apabila Person(?p) memiliki orangtua bertipe Male(?m) dan Person(?p) memiliki orangtua bertipe Female(?f) maka dihasilkan bahwa Male(?m) memiliki anak Person(?p) dan Female(?f) memiliki anak Person(?p).

|  |
| --- |
| ***Rule* 6:**  Female(?f), Male(?m), Person(?p), hasParent(?p, ?f), hasParent(?p, ?m) -> hasChild(?f, ?p), hasChild(?m, ?p) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Megawati Sukarnoputri memiliki orangtua Sukarno yang bertipe *Male*.
2. Megawati Sukarnoputri memiliki orangtua Fatmawati yang bertipe Fem*ale*.
3. Dapat disimpulkan bahwa Sukarno dan Fatmawati mempunyai anak Megawati Sukarnoputri.

*Rule* 7 merupakan langkah untuk mencari relasi *hasSon*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 7 yaitu apabila Person(?p) memiliki anak bertipe Male(?g) maka dihasilkan bahwa Person(?p) memiliki anak laki-laki Male(?g).

|  |
| --- |
| ***Rule* 7:**  Male(?g), Person(?p), hasChild(?p, ?g) -> hasSon(?p, ?g) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Sukarno memiliki anak Guntur Sukarnoputra yang bertipe *Male*.
2. Dapat disimpulkan bahwa Sukarno mempunyai anak laki-laki yaitu Guntur Sukarnoputra.

*Rule* 8 merupakan langkah untuk mencari relasi *hasDaughter*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 8 yaitu apabila Person(?p) memiliki anak bertipe Female(?g) maka dihasilkan bahwa Person(?p) memiliki anak perempuan Female(?g).

|  |
| --- |
| ***Rule* 8:**  Female(?g), Person(?p), hasChild(?p, ?g) -> hasDaughter(?p, ?g) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Sukarno memiliki anak Megawati Sukarnoputri yang bertipe *Female*.
2. Dapat disimpulkan bahwa Sukarno mempunyai anak perempuan yaitu Megawati Sukarnoputri.

*Rule* 9 merupakan langkah untuk mencari relasi *hasSibling*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 9 yaitu apabila Person(?p) memiliki ayah Male(?m) dan Person(?q) memiliki ayah Male(?m) dan Person(?p) memiliki ibu Female(?f) dan Person(?q) memiliki ibu Female(?f) dan Person(?p) berbeda dari Person(?q) maka dihasilkan bahwa Person(?p) memiliki saudara Person(?q).

|  |
| --- |
| ***Rule* 9:**  Female(?f), Male(?m), Person(?p), Person(?q), hasFather(?p, ?m), hasFather(?q, ?m), hasMother(?p, ?f), hasMother(?q, ?f), DifferentFrom (?p, ?q) -> hasSibling(?p, ?q) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Guntur Sukarnoputra memiliki orangtua Sukarno dan Fatmawati.
2. Megawati Sukarnoputri memiliki orangtua Sukarno dan Fatmawati.
3. Guntur Sukarnoputra berbeda dari Megawati Sukarnoputri
4. Dapat disimpulkan bahwa Guntur Sukarnoputra mempunyai saudara Megawati Sukarnoputri.

*Rule* 10 merupakan langkah untuk mencari relasi *hasBrother*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 10 yaitu apabila Person(?p) memiliki saudara Male(?g) maka dihasilkan bahwa Person(?p) memiliki saudara laki-laki Male(?g).

|  |
| --- |
| ***Rule* 10:**  Male(?g), Person(?p), hasSibling(?p, ?g) -> hasBrother(?p, ?g) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Megawati Sukarnoputri memiliki saudara Guntur Sukarnoputra.
2. Dapat disimpulkan bahwa Megawati Sukarnoputri mempunyai saudara laki-laki yaitu Guntur Sukarnoputra.

*Rule* 11 merupakan langkah untuk mencari relasi *hasSister*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 11 yaitu apabila Person(?p) memiliki saudara Female(?g) maka dihasilkan bahwa Person(?p) memiliki saudara perempuan Female(?g).

|  |
| --- |
| ***Rule* 11:**  Female(?g), Person(?p), hasSibling(?p, ?g) -> hasSister(?p, ?g) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Guntur Sukarnoputra memiliki saudara Megawati Sukarnoputri.
2. Dapat disimpulkan bahwa Guntur Sukarnoputra mempunyai saudara perempuan yaitu Megawati Sukarnoputri.

### Relasi Keluarga Tidak Sedarah

Relasi tidak sedarah dicari untuk mengetahui apakah antar *person* memiliki relasi terkait hubungan tiri. Hubungan yang dimaksud meliputi *hasStepParent, hasStepFather, hasStepMother, hasStepSon, hasStepDaughter, hasStepSibling, hasStepBrother,* dan *hasStepSister.*

*Rule* 12 merupakan langkah untuk mencari relasi *hasStepParent*. Teknik pencarian dilakukan dengan membandingkan antar *person*. Terdapat 2 macam *rule* yang dibuat karena adanya *instance* yang memiliki suami dan atau istri lebih dari satu. Jika suami dan atau istri lebih dari satu, maka akan memiliki batasan yang berbeda-beda untuk mendefinisikannya. Arti dari *rule* 12-1 yaitu apabila Person(?p2) memiliki anak Person(?c) dan Person(?p1) memiliki istri Person(?p2) dan Person(?p1) memiliki istri Person (?p3) dan Person(?p2) berbeda dari Person (?p3) maka dihasilkan bahwa Person(?c) memiliki orang tua tiri Person(?3).

|  |
| --- |
| ***Rule* 12-1:**  Person(?c), Person(?p1), Person(?p2), Person(?p3), hasChild(?p2, ?c), hasWife(?p1, ?p2), hasWife(?p1, ?p3), DifferentFrom (?p2, ?p3) -> hasStepParent(?c, ?p3) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Fatmawati mempunyai anak Megawati Sukarnoputri.
2. Sukarno mempunyai istri Fatmawati.
3. Sukarno mempunyai istri Inggit Garnasih.
4. Fatmawati berbeda dari Inggit Garnasih.
5. Dapat disimpulkan bahwa Megawati Sukarnoputri mempunyai orang tua tiri yaitu Inggit Garnasih.

Arti dari *rule* 12-2 yaitu apabila Person(?p1) memiliki suami Person(?p2) dan Person(?p1) memiliki suami Person(?p3) dan Person(?c) memiliki orang tua Person (?p1) dan Person(?c) memiliki orang tua Person (?p3) maka dihasilkan bahwa Person(?c) memiliki orang tua tiri Person(?3).

|  |
| --- |
| ***Rule* 12-2:**  Person(?c), Person(?p1), Person(?p2), Person(?p3), hasHusband(?p1, ?p2), hasHusband(?p1, ?p3), hasParent(?c, ?p1), hasParent(?c, ?p2), DifferentFrom (?p2, ?p3) -> hasStepParent(?c, ?p3) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Nyak Dien memiliki suami Teuku Umar.
2. Cut Nyak Dien memiliki suami Ibrahim Lamnga.
3. Cut Gambang memilki orang tua Cut Nyak Dien.
4. Cut Gambang memilki orang tua Teuku Umar.
5. Teuku Umar berbeda dari Ibrahim Lamnga.
6. Dapat disimpulkan bahwa Cut Gambang mempunyai orang tua tiri yaitu Ibrahim Lamnga.

*Rule* 13 merupakan langkah untuk mencari relasi *hasStepFather*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 13 yaitu apabila Person(?p) memiliki orangtua tiri bertipe Male(?m) maka dihasilkan bahwa Person(?p) memiliki ayah tiri Male(?m).

|  |
| --- |
| ***Rule* 13:**  Male(?m), Person(?p), hasStepParent(?p, ?m) -> hasStepFather(?p, ?m) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Gambang memiliki orang tua tiri Ibrahim Lamnga yang bertipe *Male*.
2. Dapat disimpulkan bahwa Cut Gambang mempunyai ayah tiri Ibrahim Lamnga.

*Rule* 14 merupakan langkah untuk mencari relasi *hasStepMother*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 14 yaitu apabila Person(?p) memiliki orangtua tiri bertipe Female(?m) maka dihasilkan bahwa Person(?p) memiliki ibu tiri Female(?m).

|  |
| --- |
| ***Rule* 14:**  Female(?m), Person(?p), hasStepParent(?p, ?m) -> hasStepMother(?p, ?m) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Megawati Sukarnoputri memiliki orang tua tiri Inggit Garnasih yang bertipe *Female*.
2. Dapat disimpulkan bahwa Megawati Sukarnoputri mempunyai ibu tiri yaitu Inggit Garnasih.

*Rule* 15 merupakan langkah untuk mencari relasi *hasStepSon*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 15 yaitu apabila Person(?p) memiliki anak tiri bertipe Male(?m) maka dihasilkan bahwa Person(?p) memiliki anak tiri laki-laki Male(?m).

|  |
| --- |
| ***Rule* 15:**  Male(?m), Person(?p), hasStepChild(?p, ?m) -> hasStepSon(?p, ?m) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Nyak Dien memiliki anak tiri Teuku Bidin.
2. Dapat disimpulkan bahwa Cut Nyak Dien mempunyai anak tiri laki-laki yaitu Teuku Bidin.

*Rule* 16 merupakan langkah untuk mencari relasi *hasStepDaughter*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 16 yaitu apabila Person(?p) memiliki anak tiri bertipe Female(?m) maka dihasilkan bahwa Person(?p) memiliki anak tiri perempuan Female(?m).

|  |
| --- |
| ***Rule* 16:**  Female(?m), Person(?p), hasStepChild(?p, ?m) -> hasStepDaughter(?p, ?m) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Nyak Dien memiliki anak tiri Cut Sjak.
2. Dapat disimpulkan bahwa Cut Nyak Dien mempunyai anak tiri perempuan yaitu Cut Sjak.

*Rule* 17 merupakan langkah untuk mencari relasi *hasStepSibling*. Teknik pencarian dilakukan dengan membandingkan antar *person*. Arti dari *rule* 17 yaitu apabila Person (?c1) memiliki orang tua tiri Person(?p) dan Person(?p) memiliki anak Person(?c2) maka dihasilkan bahwa Person(?c1) memiliki saudara tiri Person(?c2)

|  |
| --- |
| ***Rule* 17:**  Person(?c1), Person(?c2), Person(?p), hasChild(?p, ?c2), hasStepParent(?c1, ?p) -> hasStepSibling(?c1, ?c2) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Gambang memiliki orang tua tiri Cut Meuligou.
2. Cut Meuligou memiliki anak Teuku Bidin.
3. Dapat disimpulkan bahwa Cut Gambang mempunyai saudara tiri yaitu Teuku Bidin.

*Rule* 18 merupakan langkah untuk mencari relasi *hasStepBrother*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 18 yaitu apabila Person (?p) memiliki saudara tiri Male(?m) maka dihasilkan bahwa Person(?p) memiliki saudara tiri laki-laki Male(?m)

|  |
| --- |
| ***Rule* 18:**  Male(?m), Person(?p), hasStepSibling(?p, ?m) -> hasStepBrother(?p, ?m) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Gambang memiliki saudara tiri Teuku Bidin.
2. Dapat disimpulkan bahwa Cut Gambang mempunyai saudara tiri laki-laki yaitu Teuku Bidin.

*Rule* 19 merupakan langkah untuk mencari relasi *hasStepSister*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Sex* (Jenis Kelamin). Arti dari *rule* 19 yaitu apabila Person (?p) memiliki saudara tiri Female(?m) maka dihasilkan bahwa Person(?p) memiliki saudara tiri perempuan Female(?m).

|  |
| --- |
| ***Rule* 19:**  Female(?m), Person(?p), hasStepSibling(?p, ?m) -> hasStepSister(?p, ?m) |

Agar lebih jelas, contohnya adalah sebagai berikut:

1. Cut Gambang memiliki saudara tiri Cut Sjak.
2. Dapat disimpulkan bahwa Cut Gambang mempunyai saudara tiri perempuan yaitu Cut Sjak.

### Relasi Terkait *Data Properties*

Relasi terkait *data properties* sebenarnya tidak memunculkan fakta baru. Namun *rule* dibuat dengan tujuan untuk memudahkan implementasi perancangan perangkat lunak. Informasi terkait kelahiran, kelulusan, kematian, dan pemakaman disimpan sendiri sebagai *instance* dengan maksud memanfaatkan kelas yang dimiliki oleh BIO. *Instance* yang memiliki tipe *Birth, Graduation, Death*, dan *Burial* memiliki properti objek *Principal* yang mengacu pada *Person* terkait *event* tersebut. Properti data yang dimiliki *instance* dengan kelas *event* kemudian diuraikan agar menjadi properti data pada *instance* dengan kelas *person* yang bersangkutan. Properti data yang dimaksud meliputi *hasBirthDate, hasBirthPlace, hasDeathDate, hasDeathPlace, hasBurialPlace,* dan *almamater.*

*Rule* 20 merupakan langkah untuk menguraikan *value* dari *hasBirthDate*. Teknik penguraian dilakukan dengan memanfaatkan kelas *Birth*. Arti dari *rule* 20 yaitu apabila *Birth* (?b) memiliki *Principal* Person(?p) dan Birth(?b) memiliki tanggal lahir (?x) maka dihasilkan bahwa Person(?p) memiliki tanggal lahir (?x).

|  |
| --- |
| ***Rule* 20:**  Birth(?b), Person(?p), Principal(?b, ?p), hasBirthDate(?b, ?x) -> hasBirthDate(?p, ?x) |

*Rule* 21 merupakan langkah untuk menguraikan *value* dari *hasBirthPlace*. Teknik penguraian dilakukan dengan memanfaatkan kelas *Birth*. Arti dari *rule* 21 yaitu apabila *Birth* (?b) memiliki *Principal* Person(?p) dan Birth(?b) memiliki tempat lahir (?x) maka dihasilkan bahwa Person(?p) memiliki tempat lahir (?x).

|  |
| --- |
| ***Rule* 21:**  Birth(?b), Person(?p), Principal(?b, ?p), hasBirthPlace(?b, ?x) -> hasBirthPlace(?p, ?x) |

*Rule* 22 merupakan langkah untuk menguraikan *value* dari *hasDeathDate*. Teknik penguraian dilakukan dengan memanfaatkan kelas *Death*. Arti dari *rule* 22 yaitu apabila *Death* (?b) memiliki *Principal* Person(?p) dan Death(?b) memiliki tanggal kematian (?x) maka dihasilkan bahwa Person(?p) memiliki tanggal kematian (?x).

|  |
| --- |
| ***Rule* 22:**  Death(?b), Person(?p), Principal(?b, ?p), hasDeathDate(?b, ?x) -> hasDeathDate(?p, ?x) |

*Rule* 23 merupakan langkah untuk menguraikan *value* dari *hasDeathPlace*. Teknik penguraian dilakukan dengan memanfaatkan kelas *Death*. Arti dari *rule* 23 yaitu apabila *Death* (?b) memiliki *Principal* Person(?p) dan Death(?b) memiliki tempat kematian (?x) maka dihasilkan bahwa Person(?p) memiliki tempat kematian (?x).

|  |
| --- |
| ***Rule* 23:**  Death(?b), Person(?p), Principal(?b, ?p), hasDeathPlace(?b, ?x) -> hasDeathPlace(?p, ?x) |

*Rule* 24 merupakan langkah untuk menguraikan *value* dari *hasBurialPlace*. Teknik penguraian dilakukan dengan memanfaatkan kelas *Burial*. Arti dari *rule* 24 yaitu apabila *Burial* (?b) memiliki *Principal* Person(?p) dan Burial(?b) memiliki tempat pemakaman (?x) maka dihasilkan bahwa Person(?p) memiliki tempat pemakaman (?x).

|  |
| --- |
| ***Rule* 24:**  Burial(?b), Person(?p), Principal(?b, ?p), hasBurialPlace(?b, ?x) -> hasBurialPlace(?p, ?x) |

*Rule* 25 merupakan langkah untuk menguraikan *value* dari *almaMater*. Teknik penguraian dilakukan dengan memanfaatkan kelas *Graduation*. Arti dari *rule* 25 yaitu apabila *Graduation* (?b) memiliki *Principal* Person(?p) dan Graduation(?b) memiliki almamater (?x) maka dihasilkan bahwa Person(?p) memiliki almamater (?x).

|  |
| --- |
| ***Rule* 25:**  Graduation(?b), Person(?p), Principal(?b, ?p), almamater(?b, ?x) -> almamater(?p, ?x) |

### Relasi Terkait Profesi Tokoh

Relasi terkait profesi tokoh berkaitan dengan peran yang dimiliki oleh tokoh dalam pengerjaan tugas akhir ini. *Rule* dibuat dengan tujuan untuk mencari relasi keterlibatan *person* (tokoh yang memiliki peran sebagai aktivis kemerdekaan, politisi dan tokoh agama) dalam suatu *event*. Properti data yang dimaksud meliputi *hasStudent, hasTeacher, isAllyOf,* dan *isEnemyOf.*

Properti *hasStudent* dan *hasTeacher* tidak dihasilkan menggunakan rule, namun menggunakan karakteristik *inverseOf*. *inverseOf* merupakan karakeristik yang berfungsi untuk nilai kebalikan. Misal Mas Mansoer *hasTeacher* Kyai Kholil, maka akan didapatkan fakta bahwa Kyai Kholil *hasStudent* Mas Mansoer. Sedangkan properti *hasAlly* dan *hasEnemy* dihasilkan menggunakan rule yang akan dijelaskan pada paragraf selanjutnya.

*Rule* 26 merupakan langkah untuk mencari relasi *isAllyOf*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Event* dan *Group*. Arti dari *rule* 26 yaitu apabila Person (?p1) memiliki kesetiaan kepada Group(?g) dan Person (?p2) memiliki kesetiaan kepada Group(?g) dan Event(?w) memiliki panglima Person (?p1) dan Event(?w) memiliki panglima Person (?p2) dan Person (?p1) berbeda dari Person (?p2) maka dihasilkan bahwa Person(?p1) memiliki sekutu Person(?p2).

|  |
| --- |
| ***Rule* 26:**  Event(?w), Group(?g), Person(?p1), Person(?p2), hasAllegiance(?p1, ?g), hasAllegiance(?p2, ?g), hasCommander(?w, ?p1), hasCommander(?w, ?p2), DifferentFrom (?p1, ?p2) -> isAllyOf(?p1, ?p2) |

*Rule* 27 merupakan langkah untuk mencari relasi *isEnemyOf*. Teknik pencarian dilakukan dengan memanfaatkan kelas *Event* dan *Group*. Arti dari *rule* 26 yaitu apabila Person (?p1) memiliki kesetiaan kepada Group(?g1) dan Person (?p2) memiliki kesetiaan kepada Group(?g2) dan Event(?w) memiliki panglima Person (?p1) dan Event(?w) memiliki panglima Person (?p2) dan Person (?p1) berbeda dari Person (?p2) maka dihasilkan bahwa Person(?p1) memiliki musuh Person(?p2).

|  |
| --- |
| ***Rule* 27:**  Event(?w), Group(?g1), Group(?g2), Person(?p1), Person(?p2), hasAllegiance(?p1, ?g1), hasAllegiance(?p2, ?g2), hasCommander(?w, ?p1), hasCommander(?w, ?p2), DifferentFrom (?g1, ?g2), DifferentFrom (?p1, ?p2) -> isEnemyOf(?p1, ?p2) |

# 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 pencarian relasi antar tokoh sejarah Indonesia. Studi kasus permasalahan tersebut dipecahkan dengan menggabungkan skema dari tiga model ontologi. Pencarian relasi antar *person* dilakukan dengan menggunakan karakteristik 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 hanya bisa dibuka menggunakan aplikasi yang mendukung format berkas RDF. 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 iHerolation. Nama iHerolation diambil dari kata Indonesian Heroes Relation yang merupakan topik dari pengerjaan tugas akhir ini. iHerolation 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* EasyRDF. Perangkat lunak ini memiliki masukan berupa *file* RDF. Berkas RDF yang dimaksud adalah ontologi baru yang dihasilkan setelah proses *reasoning*. Sedangkan keluaran dari perangkat lunak iHerolation adalah halaman HTML dengan tampilan informasi dari seorang pahlawan yang bersumber dari *file* RDF 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**.**

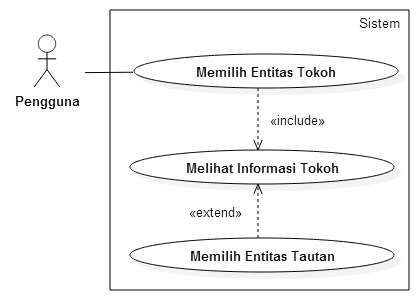
Tabel ‎4.1 Daftar Kebutuhan Fungsional Perangkat Lunak

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

### Aktor

Aktor merupakan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas yang dimaksud dapat berupa manusia, sistem, atau perangkat lunak yang lain. Aktor yang berinteraksi dengan Tugas Akhir ini yaitu pengguna yang diasumsikan tidak memahami bahasa pemrograman. Pengguna dapat memilih entitas melalui *dropdown select* atau memilih tautan yang disediakan oleh sistem untuk melihat informasi dari seorang tokoh sejarah Indonesia.

### Kasus Penggunaan



Gambar ‎4.1 Diagram Kasus Penggunaan Sistem

Kasus penggunaaan dalam Subbab ini akan dijelaskan secara rinci. Kasus penggunaan dijabarkan dalam bentuk spesifikasi kasus penggunaan dan diagram aktivitas. Diagram kasus penggunaan dapat dilihat pada Gambar ‎4.1. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel ‎4.2.

Tabel ‎4.2 Daftar Kode Diagram Kasus Penggunaan

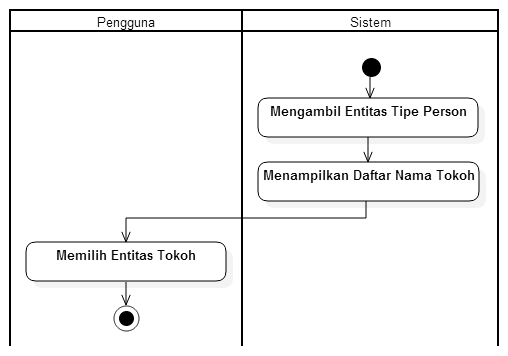
|  |  |
| --- | --- |
| **Kode Kasus Penggunaan** | **Nama** |
| TA-UC0001 | Memilih entitas tokoh |
| TA-UC0002 | Melihat informasi tokoh |
| TA-UC0003 | Memilih entitas tautan |

#### Memilih Entitas Tokoh

Pada kasus penggunaan ini, pengguna dapat memilih entitas tokoh yang ingin dilihat informasinya. Sistem membaca ontologi ihero yang berupa *file* RDF. Entitas bertipe *person* yang terdapat dalam berkas RDF ihero 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 *file* ihero.rdf dalam sistem |
| **Aliran:**   * **Kejadian Normal** |  |
| 1. Sistem mengambil entitas dalam ontologi ihero 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 *dropdwon select* |
| **Kebutuhan Khusus** | Tidak ada |



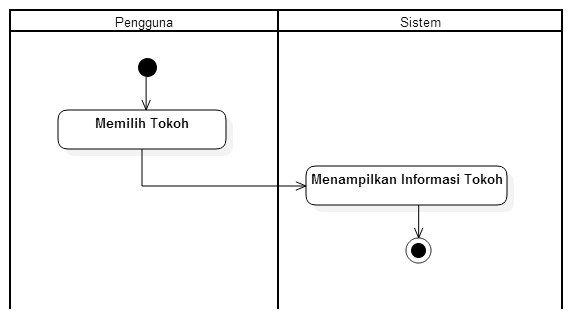
Gambar ‎4.2 Diagram Aktivitas Memilih Entitas Tokoh

#### Melihat Informasi Tokoh

Pada kasus penggunaan ini, sistem membaca ontologi ihero yang berupa *file* RDF. Informasi yang terdapat dalam berkas RDF ihero 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 informasi 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 *file* ihero.rdf dalam sistem |
| **Aliran:**   * **Kejadian Normal** |  |
| 1. Pengguna memilih tokoh melalui *dropdown select*. 2. Pengguna menekan tombol *submit.* 3. Sistem menampilkan halaman informasi dari tokoh yang dipilih. |
| **Kondisi Akhir** | Sistem menampilkan informasi dari tokoh yang dipilih dalam bentuk tabel dan *list* |
| **Kebutuhan Khusus** | Tidak ada |



Gambar ‎4.3 Diagram Aktivitas Melihat Informasi 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 berkas RDF ihero 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 informasi tokoh yang dipilih sebelumnya. |
| **Aliran:**   * **Kejadian Normal** |  |
| 1. Sistem menampilkan halaman informasi dari tokoh yang dipilih sebelumnya. 2. Pengguna memilih tautan pada halaman informasi. 3. Sistem menampilkan halaman informasi 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 tabel dan *list* |
| **Kebutuhan Khusus** | Tidak ada |

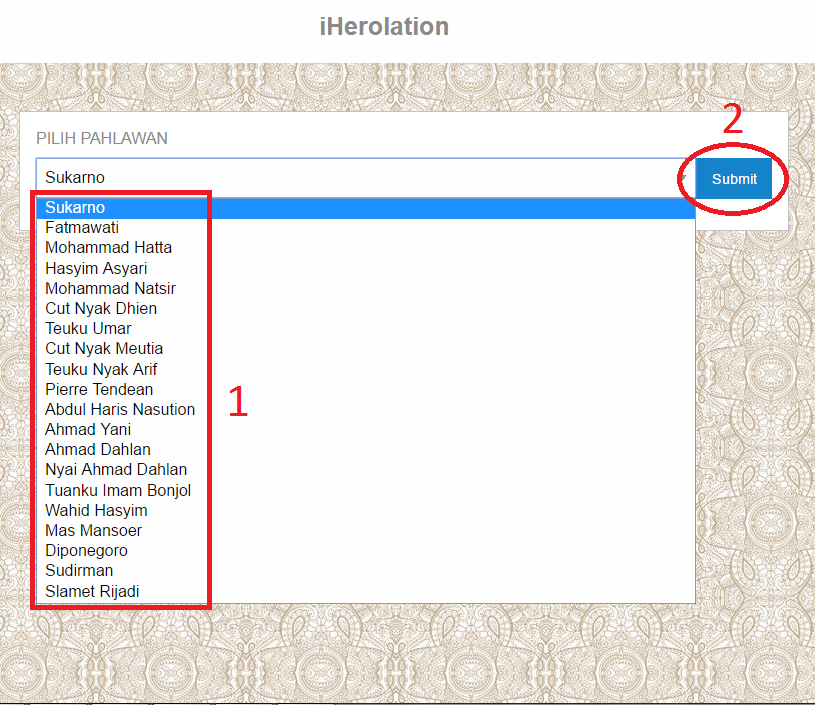


Gambar ‎4.4 Diagram Aktivitas Memilih Entitas Tautan

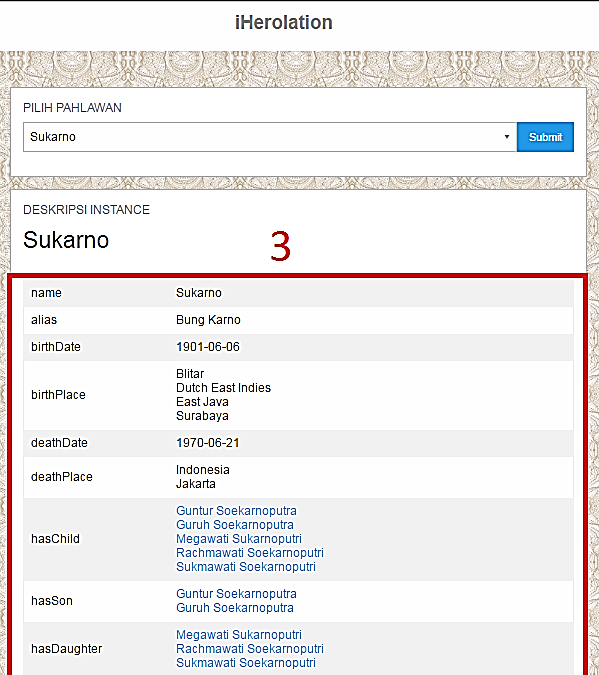
## Perancangan Antarmuka Pengguna

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

Antarmuka 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 ‎4.5 Antarmuka Halaman Utama iHerolation



Gambar ‎4.6 Antarmuka Halaman Informasi iHerolation

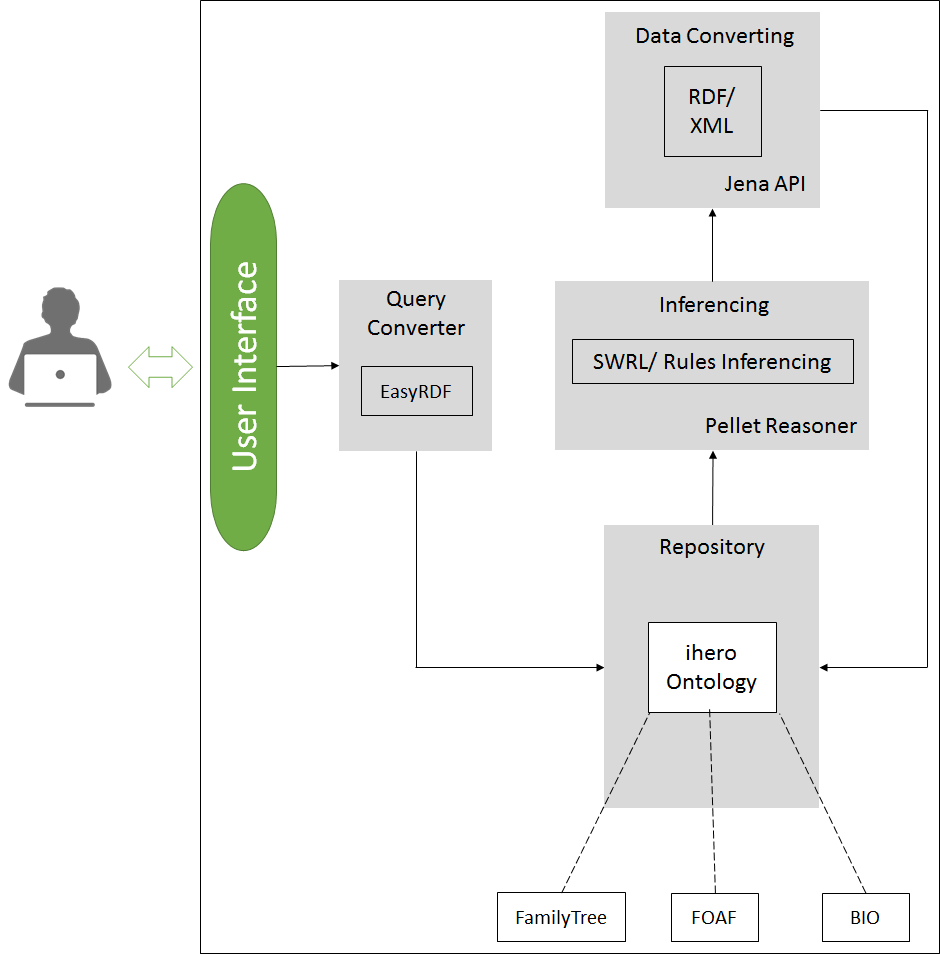
Tabel ‎4.6 Spesifikasi Atribut Rancangan Antarmuka Halaman iHerolation

| **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 Description Table* | *Table* | Menampilkan deskripsi 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 iHerolation menggunakan bahasa pemrograman PHP dengan *library* EasyRDF.



Gambar ‎5.1 Arsitektur Perangkat Lunak

Arsitektur perangkat lunak yang dibangun pada tugas akhir ini dapat dilihat pada Gambar ‎5.1 di halaman sebelumnya. Tiga ontologi yang digabungkan, yaitu FamilyTree, FOAF, dan BIO selanjutnya dikembangkan dan diberi nama sebagai ontologi ihero. Agar dapat menampilkan informasi yang tersembunyi, pada ontologi ini diterapkan sejumlah *rule* yang telah dijelaskan pada Sub subbab ‎3.3. Setelah itu, dilakukan proses *reasoning* ontologi menggunakan Pellet *Reasoner*. Data *inference* yang didapatkan dari proses *inferencing* kemudian dikonversi menjadi data RDF agar dapat dibaca oleh *library* EasyRDF. EasyRDF berperan sebagai *query converter* yang dapat mengambil data dari ontologi ihero 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 ‘Hero’. *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>";  } |

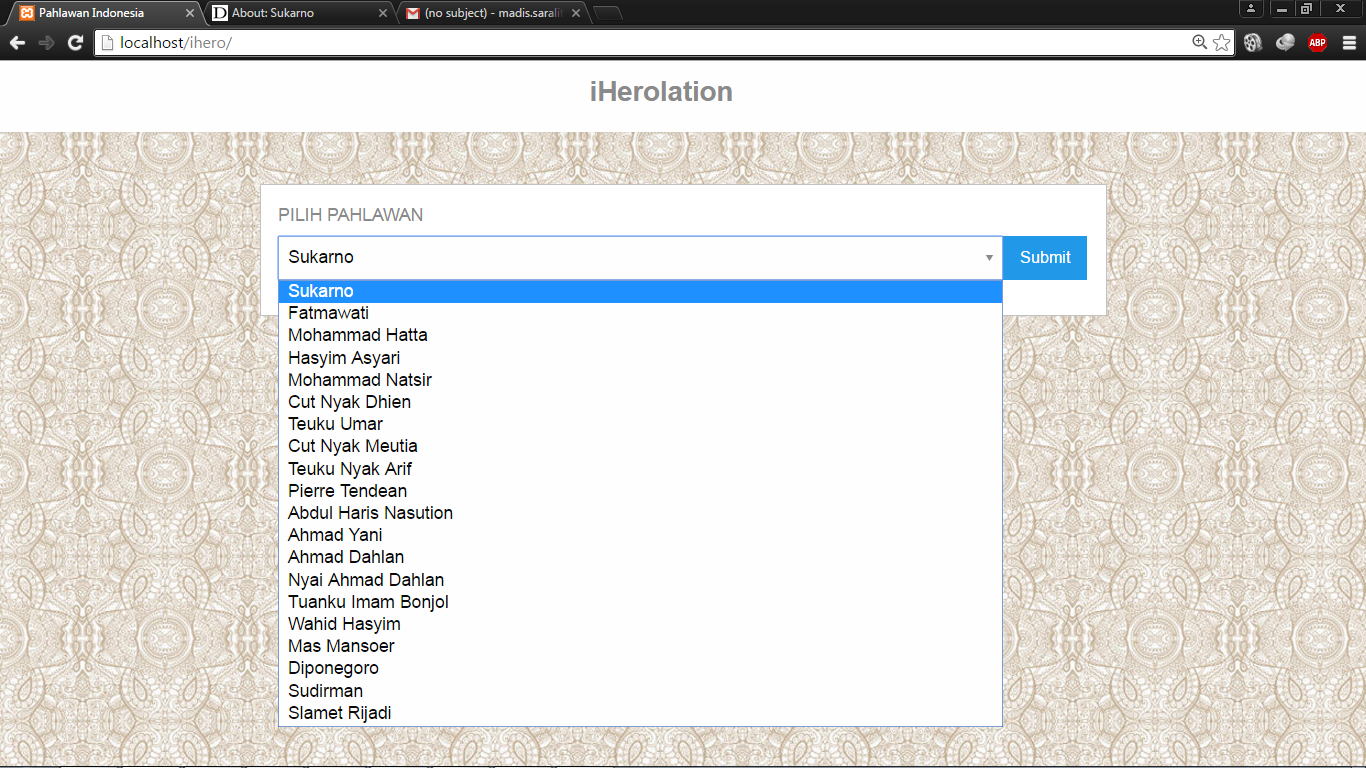
Kode Sumber ‎5.51 Fungsi Get subject

## 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

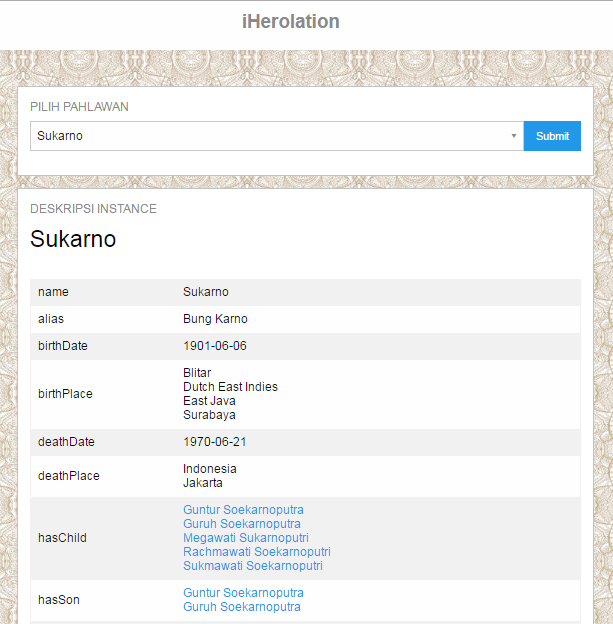
Halaman ini merupakan implementasi halaman utama dari rancangan antarmuka yang telah dijelaskan pada Subbab ‎4.2. Halaman utama hanya menampilkan kolom *dropdown select* yang dapat digunakan oleh pengguna untuk memilih tokoh. Daftar entitas tokoh yang ditampilkan hanya tokoh utama yang digunakan dalam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



Gambar ‎5.2 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 ‎5.3 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 ‎3.3. 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 i3-4030U

CPU @ 1.90GHz

Memori : 2.00 GB

Jenis *Device* : Laptop

Sistem Operasi : Microsoft Windows 7 Ultimate 64-bit

*Protege* : Protege 4.3

*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 ontologi ihero. Sedangkan pengujian kompleksitas ontologi dilakukan dengan menghitung waktu yang dibutuhkan ontologi untuk melakukan *inference* *axiom* dan *export inferenced* *axiom* berdasarkan kapasitas memori yang digunakan. *Inference axiom* juga disebut sebagai fakta baru yang dihasilkan ontologi.

### 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.

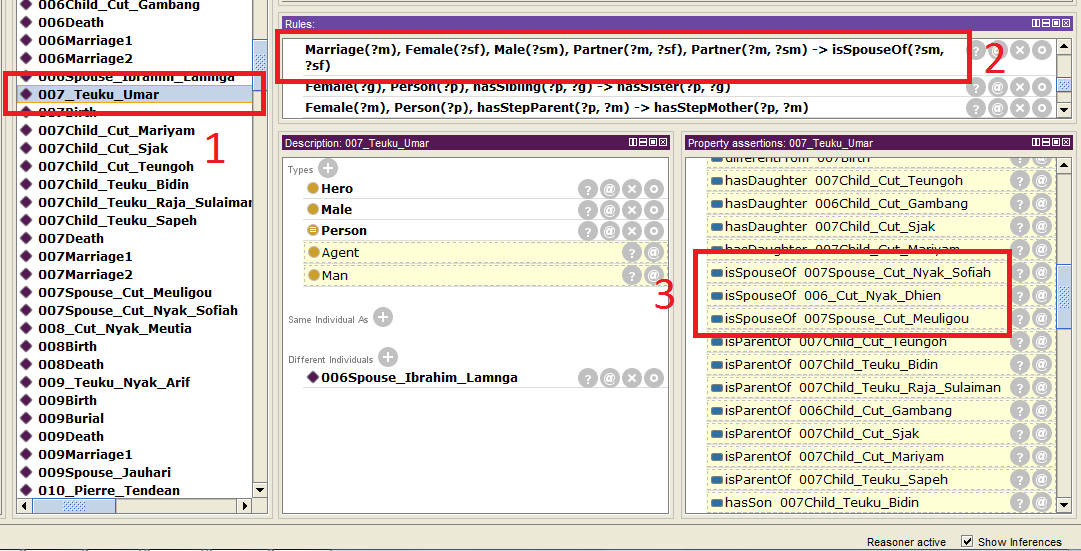
#### Pengujian Kevalidan Relasi isSpouseOf

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 ‎3.3. 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 ‎6.1 Uji Coba Relasi isSpouseOf

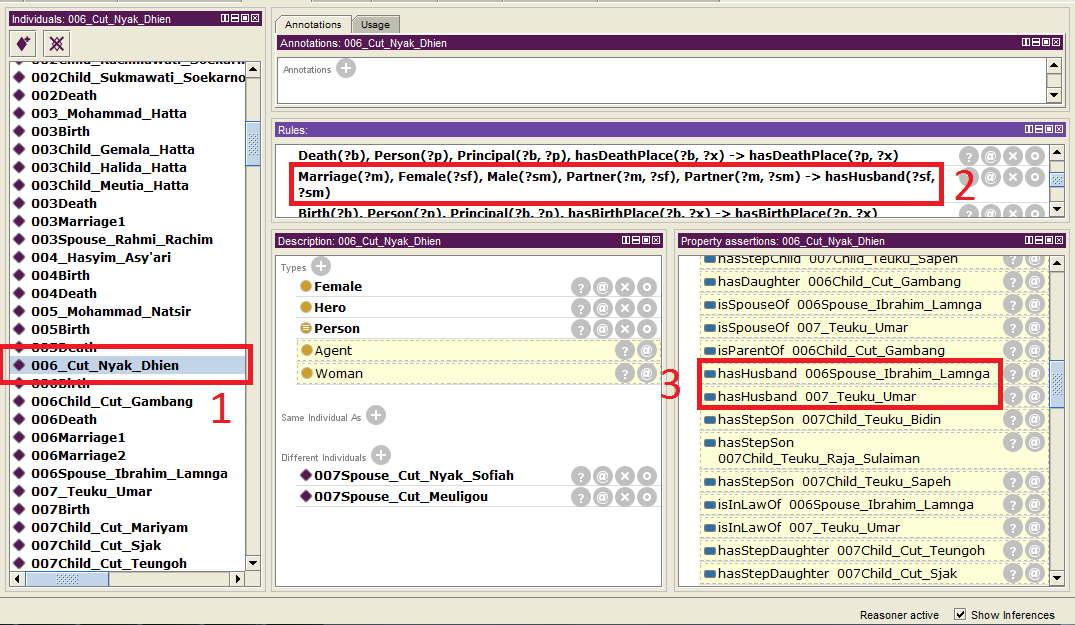
#### Pengujian Kevalidan Relasi hasHusband

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 ‎3.3. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasHusband telah berhasil.



Gambar ‎6.2 Uji Coba Relasi hasHusband

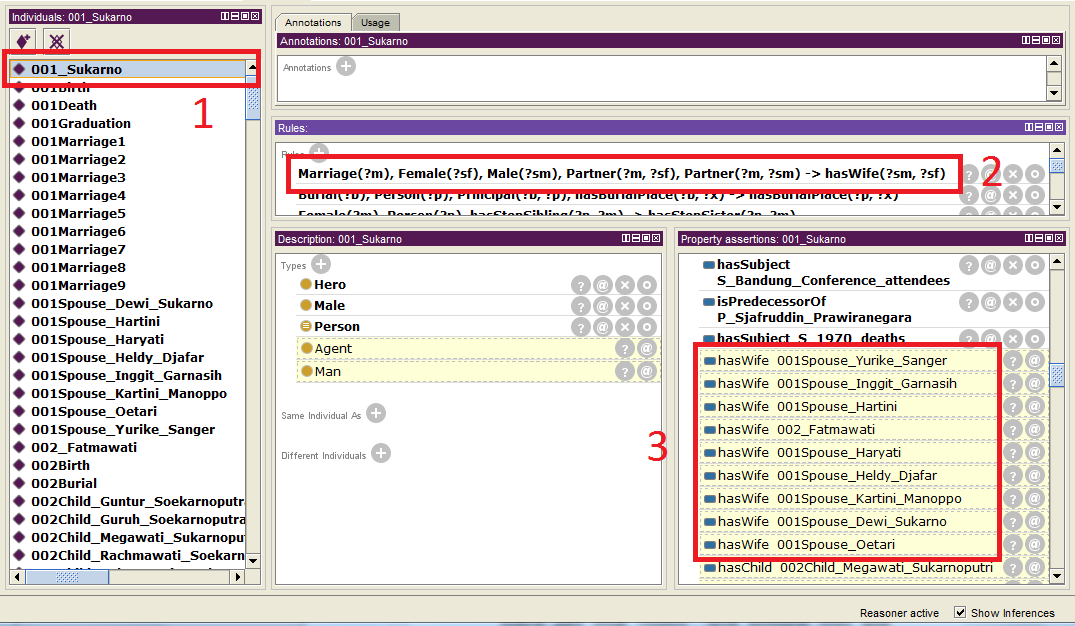
#### Pengujian Kevalidan Relasi hasWife

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 ‎3.3. 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 ‎6.3 Uji Coba Relasi hasWife

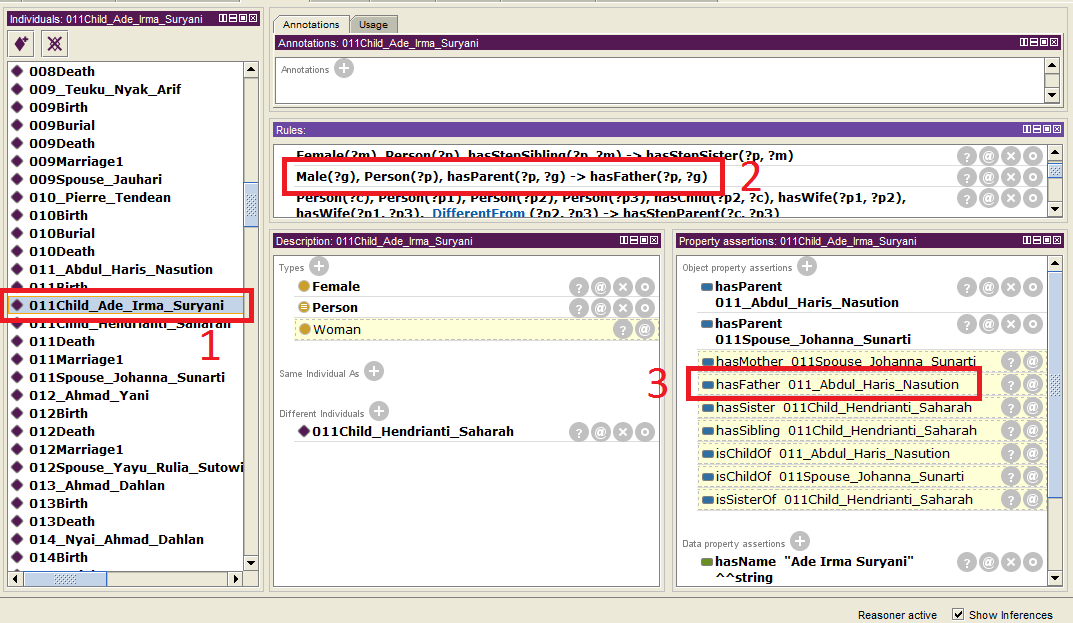
#### Pengujian Kevalidan Relasi hasFather

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 ‎3.3. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasFather telah berhasil.



Gambar ‎6.4 Uji Coba Relasi hasFather

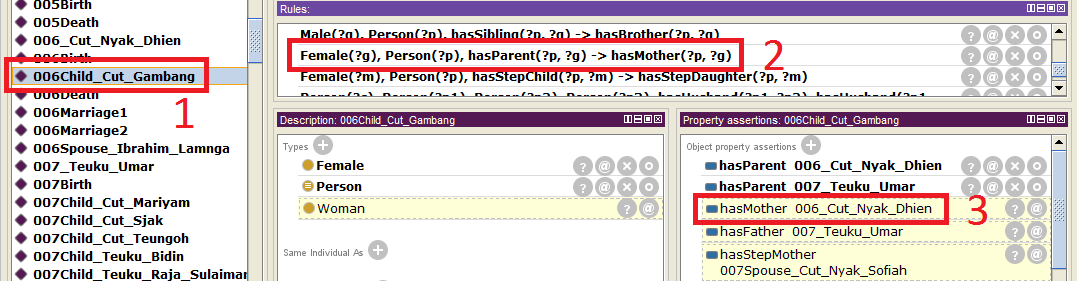
#### Pengujian Kevalidan Relasi hasMother

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 ‎3.3. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasMother telah berhasil.



Gambar ‎6.5 Uji Coba Relasi hasMother

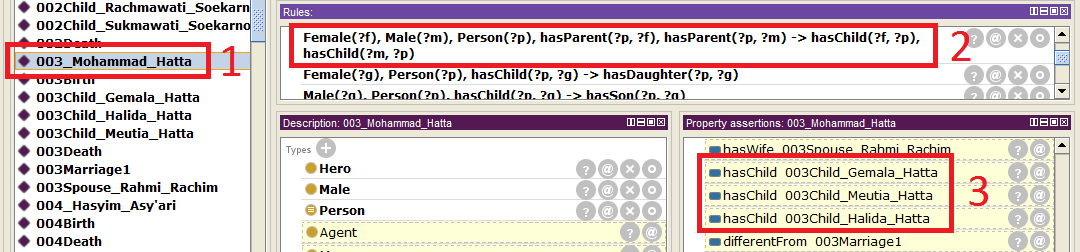
#### Pengujian Kevalidan Relasi hasChild

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 ‎3.3. 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 ‎6.6 Uji Coba Relasi hasChild

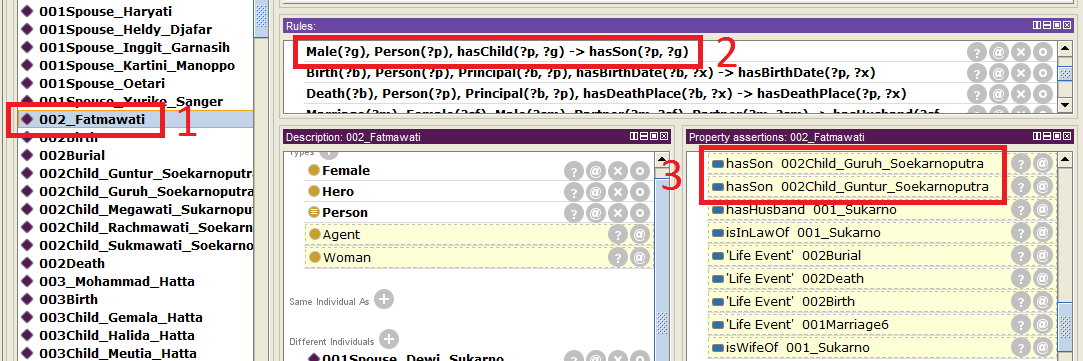
#### Pengujian Kevalidan Relasi hasSon

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 ‎3.3. 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 ‎6.7 Uji Coba Relasi hasSon

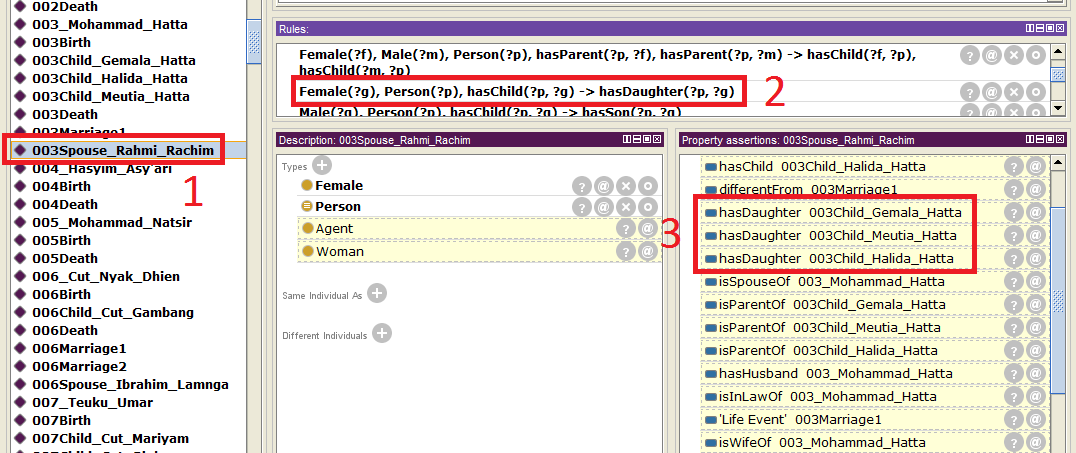
#### Pengujian Kevalidan Relasi hasDaughter

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 ‎3.3. 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 ‎6.8 Uji Coba Relasi hasDaughter

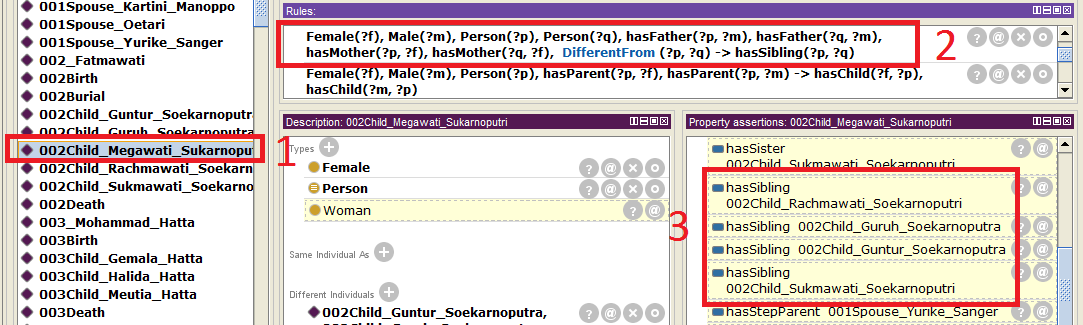
#### Pengujian Kevalidan Relasi hasSibling

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 ‎3.3. 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 ‎6.9 Uji Coba Relasi hasSibling

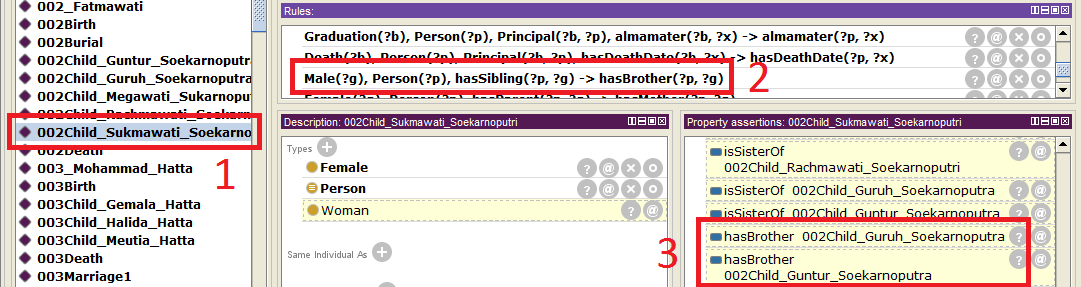
#### Pengujian Kevalidan Relasi hasBrother

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 ‎3.3. 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 ‎6.10 Uji Coba Relasi hasBrother

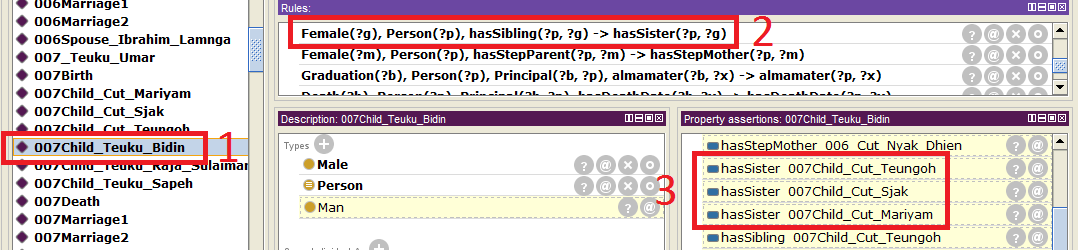
#### Pengujian Kevalidan Relasi hasSister

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 ‎3.3. 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 ‎6.11 Uji Coba Relasi hasSister

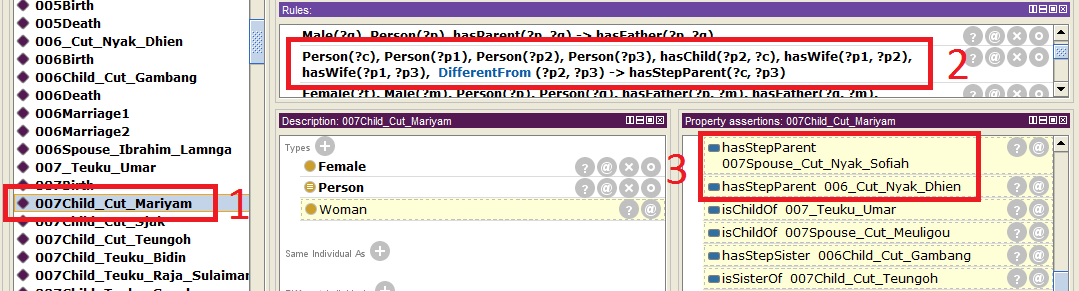
#### Pengujian Kevalidan Relasi hasStepParent

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 ‎3.3. 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 ‎6.12 Uji Coba Relasi hasStepParent

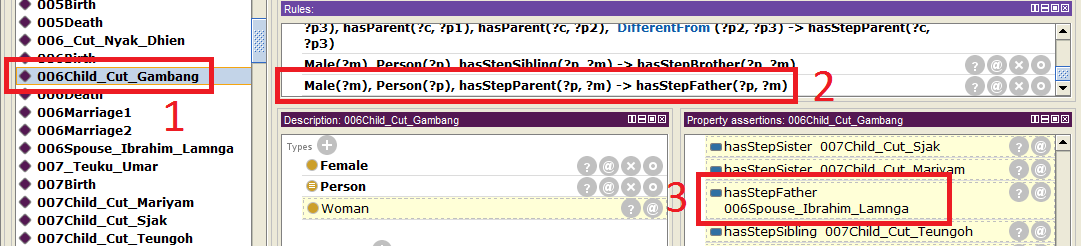
#### Pengujian Kevalidan Relasi hasStepFather

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 ‎3.3. 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 ‎6.13 Uji Coba Relasi hasStepFather

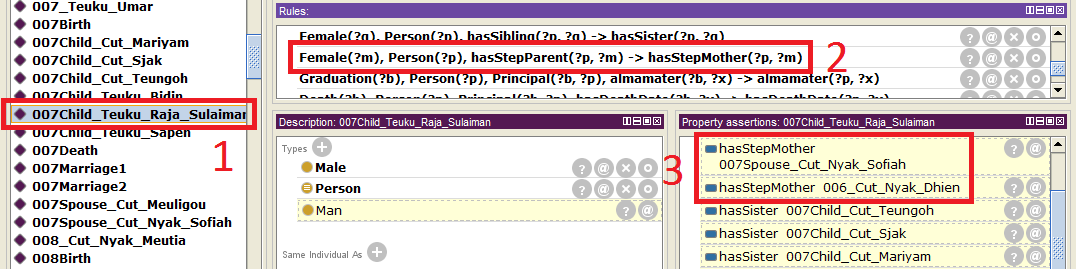
#### Pengujian Kevalidan Relasi hasStepMother

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 ‎3.3. 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 ‎6.14 Uji Coba Relasi hasStepMother

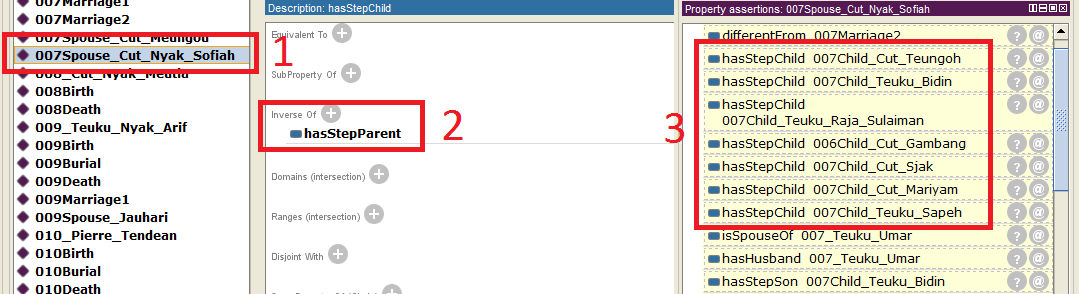
#### Pengujian Kevalidan Relasi hasStepChild

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 ‎6.15 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 ‎3.3. 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 ‎6.16 Uji Coba Relasi hasStepSon

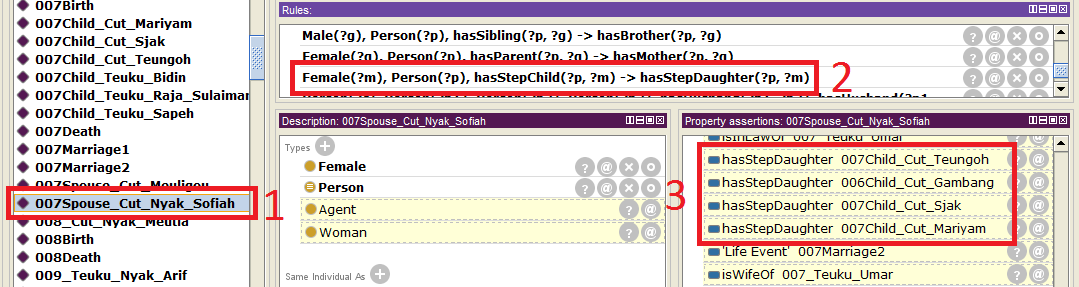
#### Pengujian Kevalidan Relasi hasStepDaughter

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 ‎3.3. 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 ‎6.17 Uji Coba Relasi hasStepDaughter

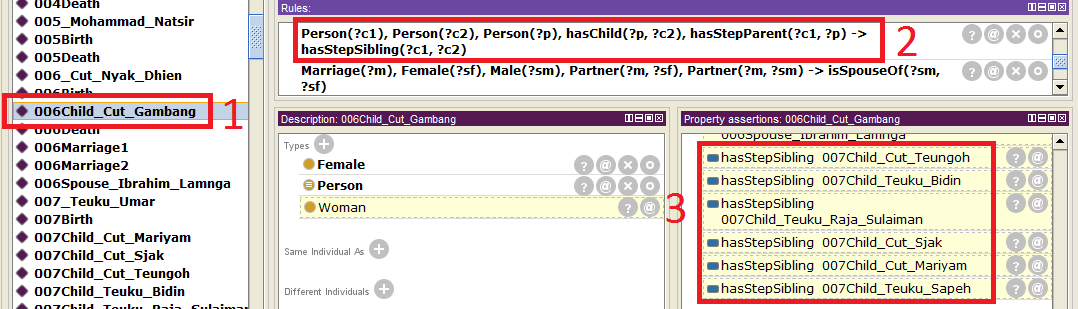
#### Pengujian Kevalidan Relasi hasStepSibling

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 ‎3.3. Sedangkan *pointer* 3 menunjukkan fakta baru setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi hasStepSibling telah berhasil.



Gambar ‎6.18 Uji Coba Relasi hasStepSibling

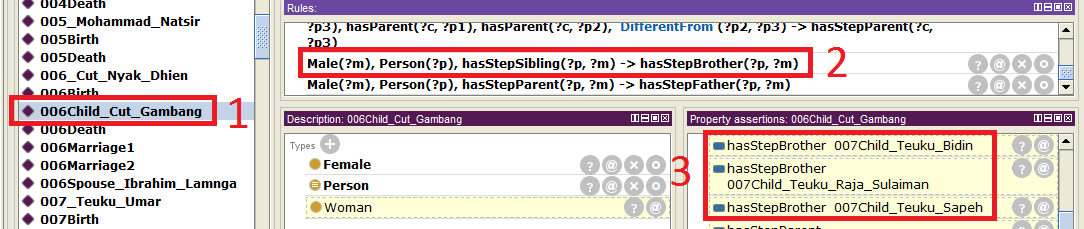
#### Pengujian Kevalidan Relasi hasStepBrother

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 ‎3.3. 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 ‎6.19 Uji Coba Relasi hasStepBrother

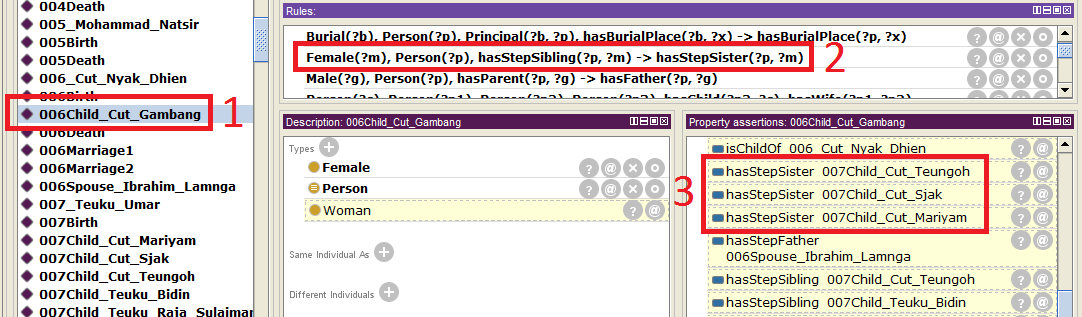
#### Pengujian Kevalidan Relasi hasStepSister

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 ‎3.3. 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 ‎6.20 Uji Coba Relasi hasStepSister

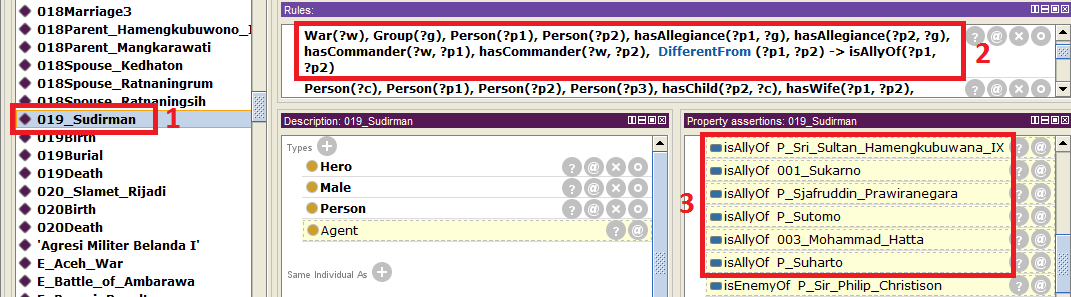
#### Pengujian Kevalidan Relasi isAllyOf

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 ‎3.3. 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 ‎6.21 Uji Coba Relasi isAllyOf

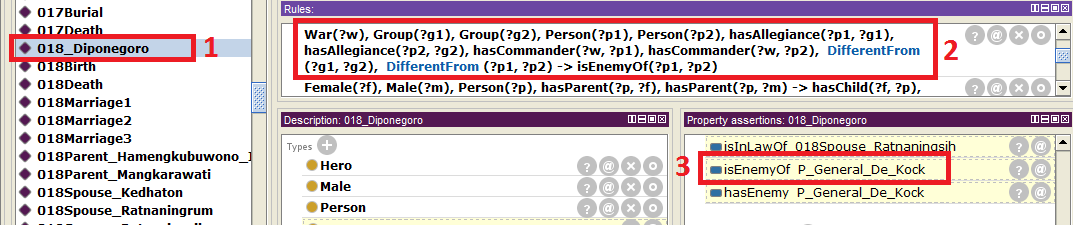
#### Pengujian Kevalidan Relasi isEnemyOf

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 ‎3.3. Sedangkan *pointer* 3 menunjukkan fakta baru berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi isEnemyOf telah berhasil.



Gambar ‎6.22 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. |

### Pengujian Kompleksitas Ontologi

Pengujian kompleksitas ontologi merupakan tahap uji yang bertujuan untuk mengetahui berapa lama waktu yang dibutuhkan oleh ontologi ihero untuk melakukan *reasoning* dan *export inferenced axiom* sebagai fakta baru. Ontologi ihero memiliki ukuran 504 KB. Pengujian pada tahap ini bergantung pada kapasitas memori yang dialokasikan untuk menjalankan tools Protégé. Alokasi kapasitas memori dapat diperkecil atau diperbesar dengan memodifikasi Protege.lax pada bagian *maximum heap size*. Protege.lax dapat ditemukan pada folder pemasangan Protégé. *Maximum heap size* adalah fungsi yang mengatur pengalokasian kapasitas maksimal jalannya tools Protégé. Pengujian dilakukan dengan mengukur waktu proses *reasoning* dan *export inferenced axiom* dengan kapasitas memori yang ditetapkan oleh Protege.lax pada pengaturan awal sebagai tolok ukur keberhasilan pengujian.

#### Pengujian Kompleksitas Reasoning Ontologi

Pada tahap pengujian ini dilakukan pengukuran waktu yang dibutuhkan ontologi ihero untuk menarik kesimpulan dari *axiom* yang ditanamkan. Tujuannya adalah untuk mengetahui apakah waktu yang ditempuh membebani perangkat pengujian atau tidak. Pengujian dilakukan dengan mengukur waktu proses *reasoning* menggunakan kapasitas memori yang ditetapkan oleh Protege.lax pada pengaturan awal sebagai tolok ukur keberhasilan pengujian. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel ‎6.43.

Tabel ‎6.43 Pengujian Kompleksitas Reasoning Ontologi

|  |  |
| --- | --- |
| ID | TA-UJ.KM0001 |
| Nama | Pengujian kompleksitas *reasoning* ontologi |
| Tujuan Pengujian | Menguji berapa lama waktu yang dibutuhkan oleh ontologi untuk melakukan proses *reasoning*. |
| Skenario 1 | Melihat waktu yang dibutuhkan oleh ontologi untuk melakukan *reasoning* pada jendela *command prompt.* |
| Kondisi Awal | Ontologi ihero belum menampilkan hasil *inferenced axiom* dan belum diketahuinya waktu proses *reasoning.* |
| Data Uji | Data uji merupakan *file* ontologi ihero yang memiliki format OWL dengan ukuran 504 KB. Kapasitas maksimal memori *(maximum heap size)* yang dialokasikan adalah 524288000 *bytes*. Kapasitas memori yang digunakan oleh Protégé adalah 506 MB. |
| Langkah Pengujian | Pengguna membuka *file* ontologi ihero melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R  Pengguna melihat jendela *command prompt* Protégé yang menyatakan status *‘Pellet classified in ... ms’* |
| Hasil Yang Diharapkan | Waktu yang dibutuhkan untuk melakukan proses *reasoning* kurang dari 1 menit. |
| Hasil Yang Didapat | Didapatkan status *‘Pellet classified in 20170 ms’* yang menyatakan bahwa proses *reasoning* membutuhkan waktu 20,17 detik. |
| Hasil Pengujian | Berhasil. |
| Kondisi Akhir | Waktu yang dibutuhkan untuk melakukan proses *reasoning* kurang dari 1 menit. |

#### Pengujian Kompleksitas Export Axiom

Pada tahap pengujian ini dilakukan pengukuran waktu yang dibutuhkan ontologi ihero untuk meng*export* hasil penarikan kesimpulan dari *axiom* yang ditanamkan. Tujuannya adalah untuk mengetahui apakah waktu yang ditempuh membebani perangkat pengujian atau tidak. Pengujian dilakukan dengan mengukur waktu proses *export inferenced axiom* menggunakan kapasitas memori yang ditetapkan oleh Protege.lax pada pengaturan awal sebagai tolok ukur keberhasilan pengujian. Rincian pengujian perbandingan data ini dapat dilihat pada Tabel ‎6.44.

Tabel ‎6.44 Pengujian Kompleksitas Export Axiom

|  |  |
| --- | --- |
| ID | TA-UJ.KM0002 |
| Nama | Pengujian kompleksitas *export axiom* |
| Tujuan Pengujian | Menguji berapa lama waktu yang dibutuhkan oleh ontologi untuk melakukan proses *export inferenced axiom*. |
| Skenario 1 | Menghitung waktu untuk melakukan *export inferenced axiom* dengan alat pencatat waktu.. |
| Kondisi Awal | Ontologi ihero sudah menampilkan hasil *inferenced axiom* setelah proses *reasoning* dan belum diketahuinya waktu proses *export inferenced axiom.* |
| Data Uji | Data uji merupakan *file* ontologi ihero yang memiliki format OWL dengan ukuran 504 KB. Kapasitas maksimal memori *(maximum heap size)* yang dialokasikan adalah 524288000 *bytes*. Kapasitas memori yang digunakan oleh Protégé adalah 506 MB. |
| Langkah Pengujian | Pengguna membuka *file* ontologi ihero melalui aplikasi Protégé  Pengguna menjalankan *reasoner* Pellet di Protégé dengan menekan tombol CTRL+R  Pengguna melakukan proses *export* dengan memilih menu *file* kemudian *export inferenced axiom.*  Pengguna mengukur waktu proses *export inferenced axiom* menggunakan alat pencatat waktu. |
| Hasil Yang Diharapkan | Waktu yang dibutuhkan untuk melakukan proses *export inferenced axiom* kurang dari 5 menit. |
| Hasil Yang Didapat | Didapatkan hasil pengukuran bahwa proses *export inferenced axiom* membutuhkan waktu lebih dari 20 menit. |
| Hasil Pengujian | Tidak berhasil. |
| Kondisi Akhir | Waktu yang dibutuhkan untuk melakukan proses *export inferenced axiom* lebih dari 5 menit. |

## 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 |

### Evaluasi Pengujian Kompleksitas Ontologi

Rangkuman mengenai hasil pengujian kompleksitas ontologi dapat dilihat pada Tabel ‎6.47. Berdasarkan data pada tabel tersebut, terdapat skenario pengujian yang tidak berhasil. Sehingga dapat ditarik disimpulkan bahwa ontologi yang dikembangkan belum sesuai dengan yang diharapkan.

Tabel ‎6.47 Rangkuman Hasil Pengujian

| ID | Nama | Skenario | Hasil |
| --- | --- | --- | --- |
| TA-UJ.KM0001 | Pengujian kompleksitas *reasoning* ontologi | Skenario 1 | Berhasil |
| TA-UJ.KM0002 | Pengujian kompleksitas *export axiom* | Skenario 1 | Tidak Berhasil |

Agar pengujian kompleksitas ontologi sesuai dengan yang diharapkan, maka dilakukan evaluasi pengujian dengan memodifikasi kapasitas maksimal memori pada Protege.lax dengan harapan bahwa pengujian dengan ID TA-UJ.KM0002 berhasil. Rangkuman hasil evaluasi pengujian kompleksitas ontologi dapat dilihat pada Tabel ‎6.48.

Tabel ‎6.48 Rangkuman Hasil Evaluasi Pengujian

| ID | Max  Heap Size | Memory | Reasoning Time | Export Time | Hasil |
| --- | --- | --- | --- | --- | --- |
| TA-E0001 | 1024288000 Bytes | 991 MB | 19595 ms | > 20 min | Tidak Berhasil |
| TA-E0002 | 1424288000 Bytes | 1378 MB | 19351 ms | > 20 min | Tidak Berhasil |

Pada tabel di atas, *max heap size* adalah pengalokasian kapasitas maksimal untuk menjalankan *tools* Protégé. *Memory* adalah kapasitas memori yang digunakan oleh *tools* Protégé ketika dijalankan. *Reasoning time* adalah waktu yang dibutuhkan untuk melakukan proses *reasoning* dan memiliki satuan ms *(miliseconds).* Sedangkan *export time* adalah waktu yang dibutuhkan untuk melakukan proses *export inferenced axiom* dan memiliki satuan min *(minutes)*.

Berdasarkan evaluasi pengujian yang telah dilakukan, semakin besar alokasi *max heap size* maka waktu yang dibutuhkan proses *reasoning* lebih cepat. Namun, evaluasi pengujian ini tidak berdampak pada waktu yang dibutuhkan proses *export inferenced axiom* sehingga pengujian kompleksitas ontologi masih belum bisa sesuai dengan yang diharapkan. *File* ontologi ihero memiliki ukuran 504 KB, cukup besar untuk ukuran suatu berkas ontologi karena data yang ditanamkan luas dan kompleks. Ukuran *file* yang besar dapat membebani perangkat pengujian. Hal ini dikarenakan terbatasanya memori perangkat *device* yang digunakan. Memori yang dimiliki oleh perangkat *device* adalah 2.00 GB. Sedangkan *max heap size* yang dapat dialokasikan terbatas hingga 1.3 GB. Jika dialokasikan lebih dari 1.3 GB maka *tools* Protégé tidak dapat dijalankan. Sehingga bisa ditarik disimpulkan bahwa kompleksitas ontologi belum sesuai dengan yang diharapkan dan perlu ditingkatkan pada pengembangan selanjutnya.

# 

# 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 pencarian relasi antar tokoh sejarah Indonesia mampu dimodelkan dengan menggabungkan serta mengembangkan kelas yang terdapat pada tiga skema ontologi yaitu *Family Relationship Ontology*, FOAF, dan *BIO Vocabulary*.
2. Properti yang dimiliki oleh *Family Relationship Ontology* dan *BIO Vocabulary* dapat digunakan pada domain tokoh sejarah Indonesia. Sedangkan properti yang dimiliki oleh FOAF tidak efektif untuk memecahkan studi kasus pencarian relasi antar *person* dalam domain sejarah karena FOAF memiliki properti untuk domain *person* yang hidup di era digital. Untuk memenuhi pendefinisian properti pada domain sejarah lebih lengkap, maka ditambahkan properti baru yang mengacu pada standar penamaan properti DBpedia.
3. *Rule* dikembangkan menggunakan SWRL untuk mencari relasi hubungan keluarga dan keterlibatan *person* (tokoh yang memiliki peran sebagai aktivis kemerdekaan, politisi dan tokoh agama) dalam suatu *event.* Fungsionalitas *rule* 100% dapat diterapkan pada 20 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 20 data uji yang digunakan.
4. Kapasitas memori diatur dengan memodifikasi *maximum heap size* pada Protege.lax. Keterbatasan kapasitas memori sebesar 2.00 GB pada perangkat *device* yang digunakan mempengaruhi kompleksitas ontologi sehingga proses *reasoning* ontologi berhasil dieksekusi kurang dari satu menit, namun proses *export axiom* membutuhkan waktu lebih dari 20 menit.
5. Library EasyRDF mampu menghubungkan berkas RDF ontologi ihero 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. Penambahan atau penggunaan kelas dan properti dari ontologi lain terkait profesi yang tidak digunakan dalam pengerjaan tugas akhir ini seperti pengajar, komposer/ musisi, penyair, pengusaha, penulis, dan jurnalis.
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

Madis Saralita, lahir pada tanggal 23 Januari 1995 di Surabaya. Penulis pernah menempuh pendidikan di MI Roudlotul Mu’allimin Laban (2000-2006), MTs Negeri 2 Surabaya (2006-2009), dan SMA Negeri 15 Surabaya (2009-2012).

Saat ini penulis sedang menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya di jurusan Teknik Informatika Fakultas Teknologi Informasi angkatan tahun 2012. Dalam menyelesaikan pendidikan S1 penulis mengambil bidang minat Manajemen Informasi (MI). Pernah menjadi asisten mata kuliah Rekayasa Pengetahuan. Terlibat aktif dalam organisasi kemahasiswaan dan kerohanian serta kepanitiaan selama perkuliahan, antara lain staff Departemen Media dan Informasi di Himpunan Mahasiswa Teknik Computer-Informatika ITS, staff Departemen Informasi dan Komunikasi di UKM Cinta Rebana ITS, staff Kementrian Komunikasi dan Informasi di Badan Eksekutif Mahasiswa ITS, Sekretaris Departemen Hubungan Masyarakat di Keluarga Muslim Informatika ITS, anggota di Pergerakan Mahasiswa Islam Indonesia, Sekretaris Schematics 2013, staff ahli Departemen Hubungan Luar di UKM Cinta Rebana ITS, staff Kesekretariatan di Schematics 2014, serta Ketua Vivat Media Award 2014. Penulis dapat dihubungi melalui alamat *email* madis.saralita@gmail.com.

1. *https://id.wikipedia.org/wiki/Daftar\_Pahlawan\_Nasional\_Indonesia* [↑](#footnote-ref-1)
2. *http://nationalgeographic.co.id/berita/2013/09/cut-nyak-dien-pejuang-perempuan-bernyali-singa* [↑](#footnote-ref-2)