

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

**GİZLİLİK FARKINDA ANLAMSAZ
KÖKEN MODELİNİN GELİŞTİRİLMESİ:
HASTA KAYITLARI ÖRNEĞİ**

Dilek YILMAZER

Tez Danışmanı : Yrd. Doç. Dr. Özgü Can

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00.

Sunuş Tarihi : 04.06.2015

Bornova-İZMİR

2015

Dilek Yılmaz tarafından Yüksek Lisans tezi olarak sunulan “Gizlilik Farkında Anlamsal Köken Modelinin Geliştirilmesi: Hasta Kayıtları Örneği” başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 04.06.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:

İmza

Jüri Başkanı	:	
Raportör Üye	:	
Üye	:	

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum “Gizlilik Farkında Anlamsal Köken Modelinin Geliştirilmesi: Hasta Kayıtları Örneği” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

.... / / 20..

İmzası

Adı-Soyadı

ÖZET**GİZLİLİK FARKINDA ANLAMSA
KÖKEN MODELİNİN GELİŞTİRİLMESİ:
HASTA KAYITLARI ÖRNEĞİ**

YILMAZER, Dilek

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Yrd. Doç. Dr. Özgü Can

Haziran 2015, 159 sayfa

Bilim adamları, veriyi ilgili veritabanından alıp doğrudan kullanmaktadır. Dolayısıyla, veritabanında saklanan verilerin çoğu manuel müdahalelerle dönüştürülmüş ve türetilmiştir. Veritabanını kullanan araştırmacılar, veriye uygulanan dönüşümler ve kaynaklar hakkında bilgi sahibi olmak istemektedir. Verinin kaynağı ve son haline gelene kadar geçirdiği süreçler hakkındaki bilgi önemli bir konu haline gelmektedir. Köken bilgisi; verinin orijinali ve veri üzerinde yapılan işlemlerin geçmişi hakkındaki detaylı bilgilerden oluşmaktadır. Bozulmuş veri parçalarını onarmak, verileri istenmeyen erişimlerden korumak ve güvenlik ihlallerini tespit etmek için köken ve güvenlik kavramları birlikte kullanılmalıdır. Bir kişiye ait bulaşıcı hastalık kayıtları gibi bilgiler oldukça hassastır ve istenmeyen erişimlerden korunması gerekmektedir. Bu da verilerin gizliliğinin korunması ile gerçekleştirilebilmektedir. Tez kapsamında, güvenlik ve gizliliği destekleyen bir köken modeli etki alanından bağımsız olarak geliştirilmiştir. Bu modelin uygulanması kapsamında, Bulaşıcı Hastalıklar Ontolojisi (BHO) ve Ulusal Aşı Ontolojisi (UAO) bu model ile entegre edilerek bulaşıcı hastalık etki alanında kişisel bilgilerin gizliliği ve güvenliğinin sağlanması, geçmişe yönelik verilerin izlenmesi, aşılama geçmişinin saklanması ve güvenlik ihlallerinin tespit edilmesi hedeflenmektedir. Geliştirilen bu köken modeli, Sağlık Köken Bilgi Sistemi (SKBS) altyapısında güvenlik ve gizliliğin desteklenmesi amacı ile kullanılabilir.

Anahtar sözcükler: Köken Yönetimi, Güvenlik, Gizlilik, Anlamsal Web, Sağlık Bilgi Sistemi, Bulaşıcı Hastalıklar Ontolojisi, Ulusal Aşı Ontolojisi.

ABSTRACT**DEVELOPING A PRIVACY-AWARE SEMANTIC
MODEL: PATIENT RECORDS SAMPLE**

YILMAZER, Dilek

MSc in Computer Engineering

Supervisor: Asst. Prof. Dr. Özgü Can

June 2015, 159 pages

Scientists use data directly through pulling it from related database. Thus, majority of the data which are stored in database were manually manipulated and derived. Researchers who use database are willing to get information about transformations applied to data and resources. The information about the resources of data and processes of getting it to the latest state have become a significant issue. Provenance information consists of detailed information about the original of data and history of operations made on data. Provenance and security terms should be used together in order to fix corrupted data parts, protect data from unwanted accesses and detect security violations. The information such as infectious disease records belongs to a person is quite sensitive and should be protected from unwanted accesses. This also can fulfill through protecting privacy of the data. In the scope of thesis, a provenance model which supports security and privacy is developed as independent from domain. In the scope of applying this model, it is aimed to provide privacy and security of personal information in the infectious disease domain, to monitor retrospective data, to preserve vaccination history and to detect security violations by integrating Infectious Disease Ontology (BHO) and National Vaccine Ontology (UAO) with this model. This provenance model developed can be used in the infrastructure of Healthcare Provenance Information System (SKBS) aiming to be supported security and privacy.

Keywords: Provenance Management, Security, Privacy, Semantic Web, Healthcare Information System, Infectious Disease Ontology, National Vaccine Ontology.

TEŞEKKÜR

Bu tez çalışması boyunca, tecrübe ve önerilerini benden esirgemeyen, satır satır yaptıklarımı inceleyip sabırla düzeltmeler yapan danışmanım Ege Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Sayın Yrd. Doç. Dr. Özgü Can'a desteği ve hoşgörüsü için sonsuz teşekkürlerimi sunarım. Tezin her aşamasında, deneyim ve önerilerinden yararlandığım, takıldığım noktalarda sorularımı yanıtsız bırakmayan Ege Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Sayın Doç. Dr. Murat Osman Ünalır'a destekleri için teşekkür ederim. Tezin hayata geçirilmesi sırasında, karşılaştığım problemlerin çözümü için deneyimlerini ve bilgilerini içtenlikle benimle paylaşan Ege Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Sayın Dr. Emine Sezer'e önerileri ve destekleri için teşekkürü bir borç bilirim. Son olarak, okul hayatım boyunca arkamda duran ve desteklerini bir an olsun esirgemeyen sevgili aileme teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET	vii
ABSTRACT	ix
TEŞEKKÜR	xi
ŞEKİLLER DİZİNİ	xvi
ÇİZELGELER DİZİNİ	xxv
SİMGELER VE KISALTMALAR DİZİNİ	xxvi
1. GİRİŞ	1
1.1 Tezden Çıkan Yayınlar	4
2. TEMEL KAVRAMLAR VE TEKNOLOJİLER	5
2.1 Anlamsal Web Kavramı ve Teknolojiler	5
2.2 Açık Köken Modeli ve Dublin Core Terimleri ile Hazırlanan Açık Köken Model Profili	7
2.3 ICD-10	11
2.4 ALCQ DL Tanımlama Dili	13
2.5 Literatürde Yer Alan Çalışmalar	14
2.6 Kullanılan Araçlar	19
2.6.1 Eclipse	19

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
2.6.2 Apache Jena Adaptor	19
2.6.3 Protégé	20
2.6.4 SPARQL	20
2.6.5 Oracle 12c	21
3. GİZLİLİK FARKINDA ANLAMSAK KÖKEN MODELİ	21
4. HASTA KAYITLARINDA GÜVENLİK VE GİZLİLİĞİ DESTEKLEYEN SAĞLIK KÖKEN BİLGİ SİSTEMİ.....	43
4.1 Bulaşıcı Hastalıklar Ontolojisi	53
4.1.1 Bulaşıcı Hastalıklar Ontolojisi Üzerinde Hastalık Gösterim Örneği	61
4.1.2 Gizlilik Farkında Anlamsal Köken Modeli ve BHO	63
4.1.2.1 Köken Modeli ile hasta kayıtlarının erişim geçmişinin kontrol edilmesi örneği	63
4.1.2.2 Köken Modeli ile hasta kayıtları üzerinde güvenlik ve gizlilik ihlallerinin tespit edilmesi örneği	70
4.2 Ulusal Aşı Ontolojisi.....	77
4.2.1 Ulusal Aşı Ontolojisi üzerinde aşı gösterim örneği	87
4.2.2 Ulusal Aşı Ontolojisi üzerinde aşı takvimi gösterim örneği	91
4.2.3 Köken Modeli ile hastanın aşı geçmişinin izlenmesi örneği.....	95

İÇİNDEKİLER (devam)Sayfa

4.3 Sağlık Köken Bilgi Sistemi'nin Gerçekleştirimi	100
4.4 SKBS Kullanıcı Arayüzleri	133
4.5 SKBS Mobil Uygulaması	153
5. SONUÇLAR VE GELECEK ÇALIŞMALAR	157
KAYNAKLAR DİZİNİ.....	160
ÖZGEÇMİŞ	170

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1.1 Anlamsal Web Mimarisi	7
2.2.1 AKM olmadan köken bilgileri ve uygulamalar	8
2.2.2 AKM kullanıldığı sistemlerde köken bilgileri ve uygulamalar	9
2.2.3 DC Profil versiyon ilişkisi örneği	11
2.3.1 ICD-10 kodları ve tanıları	12
3.1 Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan atomik kavram ve roller	37
3.2 Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan sınıfların çizge yapısı	41
4.1 Sağlık Köken Bilgi Sistemi için aktörler ve kullanım durumları.....	43
4.2 Sağlık Köken Bilgi Sistemi Katmanlı Yapısı	45
4.3 Variables sınıfının Eclipse üzerinde gösterimi	46
4.4 SqlQueries sınıfının Eclipse üzerinde gösterimi.....	46
4.5 SparqlQueries sınıfının Eclipse üzerinde gösterimi	47
4.6 DataObjectLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi.....	48
4.7 DataTransferObjectLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi.....	48

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.8 DataAccessLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi	49
4.9 BusinessLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi	50
4.10 PresentationLayer katmanında yer alan ekranların Eclipse üzerinde gösterimi	50
4.11 SKBS’ne hasta giriş işleminin sıralı diagram olarak gösterimi.....	52
4.1.1 Bulaşıcı Hastalık Ontolojisi’nde yer alan atomik kavram ve roller	57
4.1.2 Bulaşıcı Hastalık Ontolojisi’nde yer alan sınıfların çizge yapısı.....	60
4.1.1.1 Bulaşıcı Hastalık Ontolojisi’nde yer alan “Sıtma” tanımı	62
4.1.2.1.1 Hepatit C teşhisi ve hasta kayıtları üzerinde erişim kontrolü	65
4.1.2.1.2 Erişim geçmiş için çalıştırılan SPARQL sorgusunun sonucu	69
4.1.2.2.1 HIV teşhisi ve hasta kayıtları üzerinde erişim kontrolü	74
4.1.2.2.2 Hasta kayıtları üzerinde güvenlik ihlallerinin kontrol edilmesi için çalıştırılan SPARQL sorgusunun sonucu	76
4.2.1 UAO üzerinde aşı ve bulaşıcı hastalıkları ifade eden kavramlar.....	79

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.2.2 UAO'ya eklenen PersonalVaccinationCalender ve VaccineApplicationDose sınıflarının diğer sınıflarla ilişkileri.....	85
4.2.1.1 Hepatit B aşısının Protege ontoloji editörü üzerinde gösterimi	90
4.2.2.1 Hepatit B aşı takvimi için çalıştırılan SPARQL sorgusunun sonucu.....	95
4.2.3.1 Tetanoz aşısının uygulanması ve verilerin saklanması	97
4.2.3.2 Tetanoz aşısı geçmişi için çalıştırılan SPARQL sorgusunun sonucu	99
4.2.3.3 Tetanoz aşısı takvimi için çalıştırılan SPARQL sorgusunun sonucu	100
4.3.1 ConnectionProvider sınıfındaki loadOntology() metodu.....	100
4.3.2 ConnectionProvider sınıfındaki executeSparqlQuery (queryString) ve executeSparqlUpdateQuery(query-String) metotları	101
4.3.3 MD5HashCodeProvider sınıfı ve generateMD5HashCode metodu.....	102
4.3.4 UserPage sınıfında kişilerin isim kriterine göre sorgulanması için çalıştırılan kod parçası	103
4.3.5 İsim kriterine göre kişilerin listelenmesi ilişkin sıralı diagram.....	104
4.3.6 SubjectBusinessProvider sınıfı ve getPersonInfoBy-FullName(fullname) metodu	105

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.3.7 SubjectDataAccessProvider sınıfı ve getPersonInfoBy- FullName(fullname) metodu	106
4.3.8 SubjectBusinessProvider sınıfında yer alan isPermitFor- Access metodu.....	107
4.3.9 SubjectDataAccessProvider sınıfında yer alan getPermissionForPersonalInfo metodu	109
4.3.10 DateTimeBusinessProvider sınıfı, convertDateToString ve convertStringToDate metotları	110
4.3.11 Yeni bir Person nesnesinin yaratılma aşamalarına ilişkin sıralı diagram	112
4.3.12 Yeni bir Person nesnesinin yaratılma aşamasında UserPage sınıfında çalıştırılan kod parçası	113
4.3.13 SubjectDataAccessProvider sınıfındaki insertPerson- Info(newPerson) metodu	113
4.3.14 SparqlQueries sınıfında yer alan getQueryInsert- Person(newPerson) metodu	114
4.3.15 SubjectDataAccessProvider sınıfı altındaki getPerson- PathByIdentityNumber(identityNumber) metodu.....	115
4.3.16 ArtifactBusinessProvider sınıfında yer alan create- ArtifactFromPersonInfo(person, gender, marital- Status, hasOwner, creator) metodu.....	116

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.3.17 ArtifactDataAccessProvider sınıfında yer alan insert- Artifact(newArtifact) metodu.....	116
4.3.18 SparqlQueries sınıfında yer alan getQueryInsertArtifact (artifact) metodu.....	116
4.3.19 SparqlQueries sınıfında yer alan getQueryUpdatePerson- ByIdentityNumber(identityNumber) metodu.....	117
4.3.20 ArtifactDataAccessProvider sınıfında yer alan insert- HasVersionAndIsVersionOfProperties(newArtifact.get- ArtifactName(), artifact.getArtifactName()) metodu	118
4.3.21 SparqlQueries sınıfında yer alan getQueryUpdate- ArtifactHasVersion ve getQueryUpdateArtifactIs- VersionOf metotları.....	118
4.3.22 Sağlık çalışanı tarafından Artifact nesnesinin TC Kimlik numarası kullanılarak sorgulanmasına ilişkin sıralı diagram	120
4.3.23 UserPage sınıfında Artifact sahibinin TC Kimlik numarasına göre Artifact nesnelerinin listelenmesi için çalıştırılan kod parçası.....	121
4.3.24 SparqlQueries sınıfında yer alan getQuerySelect- ArtifactByOwner(identityNumber) ve getQuerySelect- ArtifactByOwnerName(owner) metotları.....	122
4.3.25 ArtifactBusinessProvider sınıfı altında yer alan isPermitForAccess(loginPersonInfo, identityNumber, purpose, artifactName	125

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.3.26 UserPage sınıfında yer alan Operation nesnesinin eklenmesi için çalıştırılan kod parçası	126
4.3.27 OperationDataAccessProvider sınıfındaki insert-OperationInfo(newOperation) metodu	127
4.3.28 SparqlQueries sınıfındaki getQueryInsertOperation(newOperation)	127
4.3.29 VaccineDataAccessProvider sınıfında yer alan insert-PersonalVaccinationCalenderList(identityNumber, vaccineName, performedDate) metodu	129
4.3.30 UserPage sınıfında Artifact detayı görüntülenirken işletilen kod parçası	130
4.3.31 PermissionDataAccessProvider sınıfında yer alan insertPermissionInfo(permission, purposeList, rightList, personList, roleList) metodu	131
4.3.32 AccessHistoryBusinessProvider sınıfında yer alan searchForSecurityViolationsByArtifactName(artifact-Name)	132
4.4.1 Kullanıcı giriş ekranı	133
4.4.2 Kullanıcı giriş ekranı hatalı girişin gösterilmesi.....	133
4.4.3 Kullanıcı şifre güncelleme ekranı ve eksik bilgi uyarısı	134
4.4.4 Kullanıcı şifre güncelleme ekranı ve hatalı e-posta/TC Kimlik numarası uyarısı	134

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.4.5 Kullanıcı şifre güncelleme ekranı ve yeni şifrenin e-posta ile gönderildiğine ilişkin mesaj	135
4.4.6 Kullanıcıya gönderilen ve yeni şifreyi de içeren e-posta	135
4.4.7 Kullanıcının sisteme giriş yapmak istediği rolü seçtiği ekran	136
4.4.8 Sağlık çalışanı ekranı	136
4.4.9 Person sekmesi ve kişi arama	137
4.4.10 Person sekmesi ve kişisel bilgilerin okuma yetkisi ile görüntülenmesi	137
4.4.11 Person sekmesi ve kişisel bilgilerin yazma yetkisi ile görüntülenmesi	138
4.4.12 Person sekmesi ve kişisel bilgilerin güncellenmesi	138
4.4.13 Person sekmesi ve yeni kişi eklenmesi	139
4.4.14 Yeni Artifact nesnesi için gönderilen e-posta	139
4.4.15 Prohibition sekmesi ve Prohibition nesnesinin yaratılması ...	140
4.4.16 Prohibition sekmesi ve Prohibition nesnelerinin sorgulanması	141
4.4.17 Permission sekmesi ve Permission nesnesinin yaratılması	141
4.4.18 Permission sekmesi ve Permission nesnelerinin sorgulanması ..	142

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.4.19 Operation sekmesi ve Operation nesnelerinin sorgulanması.....	143
4.4.20 Operation sekmesi ve bir Operation nesnesinin detaylarının görüntülenmesi	144
4.4.21 Artifact sekmesi ve Artifact nesnelerinin sorgulanması	144
4.4.22 Artifact sekmesi ve Artifact nesnesinin okuma yetkisi ile görüntülenmesi	145
4.4.23 Artifact sekmesi ve erişim için gerekli izin bulunmadığına ilişkin mesaj.....	145
4.4.24 Object sekmesi ve Object nesnelerin sorgulanması	146
4.4.25 Object sekmesi ve Object nesnesinin detaylarının görüntülenmesi.	146
4.4.26 AccessHistory sekmesi ve AccessHistory nesnelerinin sorgulanması.....	147
4.4.27 AccessHistory sekmesi ve AccessHistory detaylarının görüntülenmesi	148
4.4.28 Artifact sekmesi ve hasta tarafından Artifact detaylarının görüntülenmesi	148
4.4.29 Provenance sekmesi ve Artifact nesnesinin kökeninin sorgulanması.....	149
4.4.30 Provenance sekmesi ve Artifact nesnesinin bir versiyonunun görüntülenmesi	149

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.4.31 AccessHistory sekmesi ve hasta tarafından AccessHistory nesnesinin detaylarının görüntülenmesi.....	150
4.4.32 PersonalVaccinationCalender sekmesi ve kişisel aşı takviminin görüntülenmesi	151
4.4.33 Administration sekmesi ve sistem yöneticisi tarafından güvenlik ihlallerinin tespit edilmesi.....	152
4.4.34 Profile sekmesi ve kullanıcılar tarafından kendi profillerinin görüntülenmesi.....	152
4.5.1 Aşı hatırlatma uygulaması aşılarda listelenmesi.....	153
4.5.2 Aşılama tarihinin görüntülenmesi.....	154
4.5.3 Aşılama tarihi hatırlatıcısının ayarlanması	154
4.5.4 “Days before reminder” ekranı	155
4.5.5 Bildirim saati ayarlama ekranı	156
4.5.6 Aşı hatırlatma bildiriminin görüntülenmesi	156

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
2.2.1 Dublin Core ve AKM terimlerinin AKM kavramları üzerindeki işlevleri ..	10
2.4.1 ALCQ DL dilinde kullanılan semboller ve anlamları	14
3.1 Güvenlik ve gizliliği destekleyen köken modelinde yer alan sınıflar.....	24
3.2 Subject sınıfının özellikler	26
3.3 Person sınıfının özellikleri	26
3.4 Organization sınıfının özellikleri.....	27
3.5 Service sınıfının özellikleri.....	28
3.6 Artifact sınıfının özellikleri.....	28
3.7 Operation sınıfının özellikleri	29
3.8 Permission sınıfının özellikleri.....	31
3.9 Prohibition sınıfının özellikleri	32
3.10 Purpose sınıfının özellikleri.....	32
3.11 Role sınıfının özellikleri	33
3.12 Right sınıfının özellikleri.....	34
3.13 OperationType sınıfının özellikleri.....	34
3.14 Gender ve MaritalStatus sınıflarının özellikleri	35

ÇİZELGELER DİZİNİ (devam)

<u>Çizelge</u>	<u>Sayfa</u>
3.15 FOAF kişi profili özellikleri	35
3.16 AccessHistory sınıfının özellikleri	36
3.17 Güvenlik ve gizliliği destekleyen köken ontolojisi ölçütleri	40
4.1.1 InfectiousDisease sınıfının özellikleri	54
4.1.2 Symptom sınıfının özellikleri	55
4.1.3 BodyPart, RiskGroup, PreventionMethod ve DiagnosticCriteria sınıflarının özellikleri.....	56
4.1.4 Cause sınıfının özellikleri	57
4.1.5 ICD-10 sınıfının özellikleri.....	57
4.1.6 BHO için ontoloji ölçütleri	60
4.2.1 VaccineShotPeriod sınıfının özellikleri	84
4.2.2 PersonalVaccineCalender sınıfının özellikleri.....	84
4.2.3 UAO'nun kullanım kolaylığını arttırmak için eklenen özellikler.....	85

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

BHO	Bulaşıcı Hastalıklar Ontolojisi
UAO	Ulusal Aşı Ontolojisi
XML	eXtensible Markup Language
RDF	Resource Description Framework
OPM	Open Provenance Model
DC	Dublin Core
DL	Description Logic
BFO	Basic Formal Ontology
OBO	Open Biomedical Ontology
OTMA	Organ Transplant Management Application
UNAIDS	Birleşmiş Milletler HIV/AIDS Ortak Programı
AKM	Açık Köken Modeli
SKBS	Sağlık Köken Bilgi Sistemi
WHO	World Health Organization
VO	Vaccine Ontology
FOAF	Friend of a Friend
SOA	Service Oriented Architecture

1. GİRİŞ

İnternet kaynaklı bilgi akışının hızlanması ve erişilebilir depolama alanlarının artması ile birlikte verinin kaynağı ve yaratılış süreci hakkındaki bilgiye ilgi artmıştır. Biyoloji, kimya, fizik gibi alanlardaki bilim adamları ilgili veritabanından veriyi alıp kullanmaktadır. Veritabanında saklanan verilerin çoğu manuel müdahalelerle dönüştürülmüş ve türetilmiştir. Veritabanını kullanan araştırmacılar, veriye uygulanan dönüşümler ve kaynaklarla ilgili bilgi sahibi olmak istemektedir. Bu bilgi; verinin kalitesini değerlendirmek, verinin türetilme sürecini incelemek ya da yeniden gözden geçirmek için kullanılabilir. Veri ambarları farklı veri gösterimlerini ve farklı kaynakları kullanarak veriyi entegre etmek ve entegre edilmiş veriyi analiz etmek için kullanılmaktadır. Veri ambarındaki verinin yaratılmasında kullanılan orijinal veri kaynağı ve dönüşümler hakkındaki bilgiden yararlanılarak analizler çıkarılmaktadır. İş akışı yönetim sistemleri, Servis Odaklı Mimari (Service Oriented Architecture, SOA) tabanlı sistemlerde, bazı dağıtık servisler büyük olasılıkla karmaşık matematiksel görevleri gerçekleştirmek için kullanılmaktadır. Kullanıcı için bir hesaplama bir sonucun nasıl üretildiğini anlamak önem taşımaktadır. İş uygulamalarında, doküman yönetim sistemleri şirket dokümanlarını yönetmek ve birden fazla kullanıcıya aynı doküman üzerinde çalışma hakkı vermek için kullanılmaktadır. Bir süpervizör, köken (provenance) bilgisini şirketteki iş akışlarını daha iyi anlamak için ve hatalı doküman ya da doküman parçalarının orijinallerini bulmak için kullanılmaktadır. Köken bilgisi, verinin orijinali ve verinin yaşam döngüsü boyunca gerçekleştirilen süreçler sonucunda veri üzerinde yapılan değişiklikleri kapsayan kayıtlardır. Bu bilgi, veriler üzerinde bilimsel analiz ve süreçlerin tekrarlanabilirliğini sağlamaktadır.

Hastane yönetim sistemlerinde ise hastaya ait veriler dağınık halde bulunmakta ve veri sağlık kuruluşları tarafından saklanmaktadır. Hastane personeli, verileri girebilmekte, değiştirebilmekte ve verilere erişebilmektedir. Hastanın mahremiyet ve özel hayat hakkı kapsamında Dünya Sağlık Örgütü (World Health Organization, WHO) (1994) : "Hastanın sağlık durumu, tıbbi durumu, tanısı, prognozu, tedavisi hakkındaki ve kişiye özel diğer tüm bilgiler, ölümden sonra bile gizli olarak korunmalıdır. Hastalar, tanıları, tedavileri ve bakımları ile ilgili kayıtlara, diğer dosyalara, teknik kayıtlara ve tıbbi dosyalarına bakabilme ve kendi dosyalarının ve kayıtlarının kopyasını alabilme hakkına sahiptir. Bu hak üçüncü kişilerin bilgilerine bakabilmeyi içermez. Hastalar, kendileriyle ilgili tıbbi ve kişisel bilgilerin uygunsuz, eksik, çift anlamlı, eski olması veya tanı, tedavi ve bakım amacıyla ilgili olmaması durumunda

bu bilgileri yenileme, daha açık hale getirme, bazı kısımlarını çıkarma, tamamlama, düzeltme hakkına sahiptir. Hastanın tanı, tedavi ve bakımı için gerekli olmadıkça ve ek olarak hasta izin vermedikçe hastanın özel ve aile hayatına girilemez.” maddelerini yayınlamıştır. Gizlilik hakkı kapsamında ise Dünya Tabipler Birliği (1995) : “Hastanın sağlık durumu, tıbbi durumu, tanısı, prognozu, tedavisi ve kişiye özel diğer tüm bilgiler ölümden sonra bile gizli olarak korunmalıdır. İstisna olarak hasta yakınlarının kendileri ilgili sağlık risklerini öğrenmeleri açısından bu bilgilere ulaşabilme hakkı olabilir. Gizli bilgiler sadece hastanın açık izni veya mahkemenin kesin isteği üzerine açıklanabilir. Hastanın açık olarak izin vermediği durumlarda bu bilgiler sadece bilgilendirilmesi gereken diğer sağlık personeline verilebilir. Hastanın kimliğine ait tüm bilgiler korunmalıdır. Bu bilgilerin korunması usulüne uygun yapılmalıdır. Bu tür verilerin alındığı insan ürünleri de aynı şekilde korunmalıdır.” maddelerini yayınlamıştır. Bu maddelerin uygulanabilmesi için hasta tarafından bilgilerine erişim izninin belirlenmesi ve bu izinlere uyulması gerekmektedir.

Tez çalışması kapsamında, gizlilik farkında Anlamsal Web tabanlı bir köken modelinin geliştirilmesi ve bu modelin bulaşıcı hastalıklar etki alanı üzerinde uygulanması hedeflenmektedir. Gizlilik farkında köken modeli kullanılarak, veriler ilk oluşturulduğu andan itibaren yapılan tüm değişikliklerin izlenmesi, belirlenen erişim izinlerine ve kısıtlarına uygun olarak verilere erişim sağlanması ve güvenlik ihlallerinin tespit edilmesi sağlanmaktadır. Bu model, bulaşıcı hastalıklar etki alanına uygulandığında, hasta bilgilerinin saklanması ile ilgili güvenliğin ve hasta mahremiyetinin korunması sağlanmaktadır.

Anlamsal Web, web üzerindeki bilgilerin iyi şekilde tanımlanması ve bu bilgilerin arasında anlamsal bir bağ kurulması için geliştirilmiş bir teknolojidir. Farklı etki alanları ve servislerle birlikte çalışabilirliği, veriler arasında anlamsal bağlar oluşturabilmeyi ve ilişkisel veritabanından farklı olarak sabit bir veri modeli değil de zaman içinde farklı gereksinimlere uygun olarak geliştirilebilecek, uygulama bağımsız veri modelleri yaratmayı desteklemesi nedeniyle tez çalışmasında Anlamsal Web tabanlı bir model geliştirilmiştir. Anlamsal Web teknolojileri kullanılarak yaratılan veri modelleri ontoloji olarak adlandırılmaktadır. Tez kapsamında üç farklı ontoloji entegre olarak kullanılmaktadır. Bunlardan ilki, gizlilik farkında Anlamsal Web tabanlı köken modeli için ve ikincisi, bulaşıcı hastalıklar etki alanı için Protege ontoloji editörü kullanılarak oluşturulan ontolojilerdir. Diğerisi ise, aşı detaylarını saklamak ve kullanmak için daha önce doktora tezi olarak geliştirilmiş olan bir aşı ontolojisidir. Anlamsal Web verileri için sorgulama dili olarak SPARQL ve

uygulama geliřtirmi iin ise Eclipse ortamı ve Apache Jena atısı kullanılmaktadır.

Tez alıřması kapsamında, etki alanından bağımsız olarak geliřtirilen model ile ařağıda yer alan maddelerin gerekleřtirilmesi hedeflenmektedir:

- Veriler zerinde yapılan deėiřiklik kayıtlarının tutulması.
- Yapılan deėiřikliklerle oluřturulan kayıtların aralarında versiyon iliřkilerinin kurulmasının saėlanması.
- Verilerin eriřim gemiřlerinin saklanması.
- Veriler ve verilerin eriřim gemiřleri zerinde eriřim izin ve yasaklarının tanımlanması ile veri gizliliėinin korunması.
- Verilere ait eriřim gemiřleri ile tanımlı eriřim izin ve yasaklar kullanılarak gvenlik aıklarının tespit edilmesinin saėlanması.

Tez alıřması kapsamında, geliřtirilen modelin saėlık etki alanına uygulanması ile ařağıda yer alan maddelerin gerekleřtirilmesi hedeflenmektedir:

- Bulařıcı hastalıklar etki alanında tutulan hasta kayıtları ile ilgili gemiře ynelik verilerin saėlanması.
- Hastaya ait bulařıcı hastalık kayıtlarının gvenliėinin ve gizliliėinin saėlanması.
- Hasta kayıtlarına eriřim gemiři kontrol edilerek gvenlik ve gizlilik ihlallerinin tespit edilmesi.
- Bulařıcı hastalıėın seyrinin izlenebilmesi.
- Kiřinin yakın akrabalarına ait bulařıcı hastalık gemiři incelenerek nlem alınmasının saėlanması.
- Bulařıcı hastalıkların seyri incelenerek diėer hastalarda benzer kořullarda sonuların ngrlmesi.

- Uygulanacak tedavi ve tetkikler için geçmiş bulaşıcı hastalık köken bilgileri kullanılarak kritik kararlarda destek sağlanması.
- Kişinin aşı geçmişinin takip edilebilmesi ve bir sonraki aşı tarihinin hatırlatılması.

1.1 Tezden Çıkan Yayınlar

Can, Ö., Yilmazer, D., 2014, A Privacy-Aware Semantic Model for Provenance Management, Metadata and Semantics Research: 8th Research Conference, CCIS 478, 162-169 pp.

2. TEMEL KAVRAMLAR VE TEKNOLOJİLER

Bu bölüm altında, tezde kullanılan temel kavramlara ve bu kavramlar kullanılarak geçmişte yapılan çalışmaların detaylarına yer verilmektedir. Ayrıca modelin gerçekleştiriminde kullanılan teknik araçlarla ilgili bilgiler verilmektedir.

2.1 Anlamsal Web Kavramı ve Teknolojiler

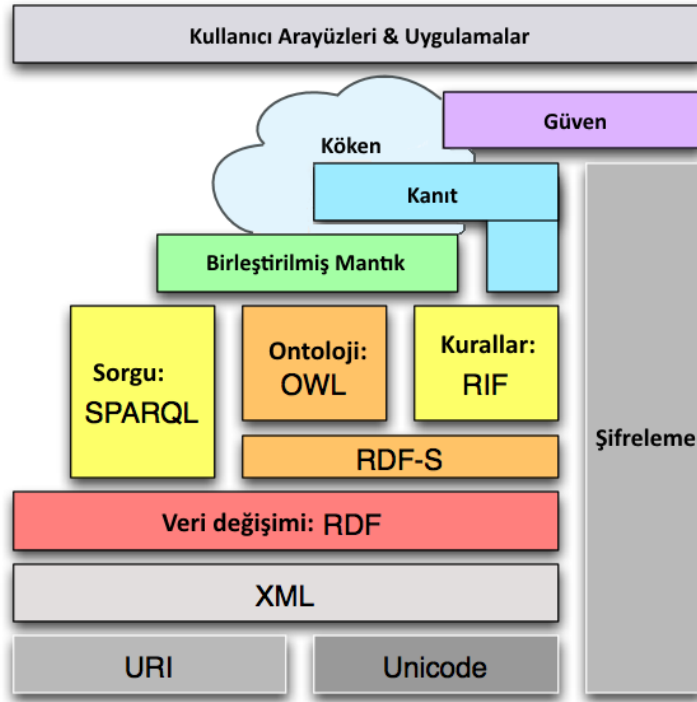
Günümüzde web sayfalarının içerikleri insanların okumasına yönelik olarak tasarlanmıştır, ancak bilgisayar programları bu sayfaları anlamsal olarak değiştirememekte ya da işleyememektedir. Bu amaçla, bilgisayarlar web sayfalarının içeriklerini çözümleyecek şekilde çalıştırılabilir, fakat bilgisayarlar web sayfalarının içeriğinde yer alan anlamları işlemek için yeterince güvenilir değildir. Çünkü her kelimenin farklı kelimelerle kombinasyonundan farklı anlamlar çıkarılabilmektedir.

Makineler tarafından işlenebilen verilerin yer aldığı bir web ortamı için bugünkü web'in temeli olan URI, HTML ve HTTP gibi yapıların da yaratıcısı olan Tim Berners-Lee tarafından Anlamsal Web terimi ortaya atılmıştır.

Web sayfalarının içeriğinin anlamsal bir yapıya sahip hale getiren Anlamsal Web'den ayrı olarak değil de, Web'in bir eklentisi olarak düşünülmelidir. Bilgisayar ve insanların birlikte çalışabilirliğini desteklemekte ve veriye iyi tanımlanmış anlamlar verilmesini sağlamaktadır. Bunun için hem veri hem de veriler arasındaki anlamların tanımlı olduğu kuralları ifade eden bir dil sunmaktadır.

Anlamsal Web üzerinde geliştirme yapmak için iki önemli teknoloji bulunmaktadır: XML (eXtensible Markup Language) (Quin, 2015) ve RDF (Resource Description Framework) (RDF Working Group, 2015). XML, kullanıcıların kendi etiketlerini yaratabilmesine ve dokümanların yapılarını kendi isteklerine göre oluşturmalarına izin veren bir dildir. RDF ise; özne, yüklem, nesne gibi üçlü ifade kümelerini XML tabanlı olarak kodlama imkanı sunmaktadır. Özne ve nesnelerin her biri URI (Universal Resource Identifier) kullanılarak tanımlanmaktadır (Berners-Lee et al., 2001). Şekil 2.1.1'de (Berners-Lee, 2006; Sizov, 2007) yer alan mimari uygulamalar arasında verilerin birlikte çalışabilirliğini destekleyen bir yapı sunulmaktadır. Bu yapıya göre, en alt

katmande Unicode ve URI alanları bulunur. Üstte yer alan katmanlar ise XML ve isim alanları (Namespaces) tabanlıdır. XML, sadece URI kullanırken; isim alanları sadece Unicode kullanmaktadır (Berners-Lee et al., 2001). Sonraki katmanlar, XML ve RDF'tir. Bu katmanların üstünde ise, RDF-S (RDF Schema) (W3C Recommendation, 2014) yer almaktadır. RDF-S, RDF'ten farklı olarak özne, nesne gibi ifadelerin yer aldığı sınıfların tanımlanmasına olanak sağlamaktadır. Ontoloji (Ontology: OWL) ise, çok daha geniş bir tanım kümesine izin vermektedir. OWL(Web Ontology Language) dili kullanılarak, modelde yer alan kavramların, gerçek dünya nesneleri ile eşleştirilebilmesini sağlamaktadır. Yaratılan model üzerinde SPARQL kullanılarak sorgu yapılabilir. Kurallar (Rules: RIF) katmanı ise; OWL, RDF-S ve SPARQL yardımıyla model üzerinden kurallar tanımlanabilmesine olanak vermektedir (Gerber et al., 2008). Birleştirilmiş Mantık (Unifying Logic) katmanı, modelde yer alan bilgiler ve RIF(Rule Interchange Format) (Kifer et al., 2013) yardımıyla tanımlanan kurallardan yararlanılarak mantıksal çıkarsamalar elde etmek için kullanılmaktadır (Sezer, 2014). Birleştirilmiş mantık ve kurallar yardımıyla, elde edilen sonucun nasıl ortaya çıktığını anlamaya yönelik ispatlar yapılabilir. Bu nedenle, köken bilgisi; Kanıt, Birleştirilmiş Mantık ve Güven katmanları etrafında büyüyen bir bulut gibidir ve Şekil 2.1.1'de gösterildiği gibi entegre edilmektedir (Sizov, 2007). Şifreleme(Cryptography), Anlamsal Web verilerinin güvenilir bir kaynaktan geldiğini doğrulamak için önem taşımaktadır. Güvenilir bir kaynaktan gelen veriler ve verilerin doğruluğunun ispatlanması ile verilere güven duyulacaktır. En üstte ise, kullanıcı arayüzü ve uygulamalar katmanı yer almaktadır.

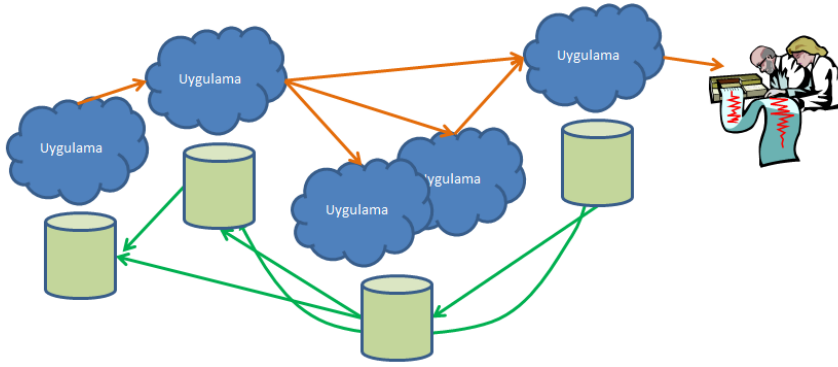


Şekil 2.1.1 Anlamsal Web Mimarisi

2.2 Açık Köken Modeli ve Dublin Core Terimleri ile Hazırlanan Açık Köken Model Profili

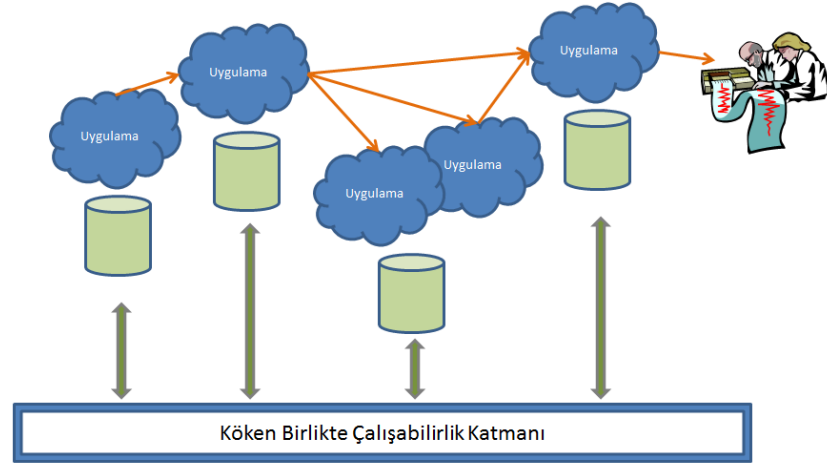
Köken (Provenance) kavramı, dijital bir nesnenin yaşam döngüsü boyunca, dokümantasyon süreçleri ya da bir eserin son haline getirilene kadar üzerinde yapılan değişiklik kayıtlarının belgelenmesi ile elde edilen dijital kütüphaneler olarak anlaşılmaktadır. Köken kavramı, bilimsel analiz ve süreçlerin tekrarlanabilirliğini sağlayarak bilim adamlarına yardımcı olan iş akışı sistemlerinin önemli bir parçası olduğundan beri, e-bilim topluluğunun köken kavramına ilgisi büyümektedir. Bunun üzerine, veri kökeni, süreç dokümantasyonu, veri türetme ve veri bilgi notları ile ilgilenen 50 katılımcının dahil olduğu International Provenance and Annotation Workshop (IPAW'06), 2006 yılında Chicago'da düzenlenmiştir. Köken araştırma topluluğunun, köken standartları oturumu boyunca tasarımların oluşmasında rol oynayan gerekçeler, benzerlikler, farklılıklar ve köken için kullanılan gösterimler ile farklı sistemlerin kapasitelerinin daha anlaşılmasına ihtiyaç duymasıyla birlikte bir fikir birliği ortaya çıkmaya başlamıştır. Bununla beraber, ilk köken zorluğu, en başından itibaren kökenin rekabetçi olmaktan ziyade bilgilendirici olması olarak belirlenmiştir. Farklı köken sistemlerinin kapasiteleri ve bu sistemlerin köken

gösterimlerinin anlamlı olmasını sağlamak bu zorluğun içinde yer almaktadır. Katılımcılar, Fonksiyonel Manyetik Rezonans Görüntüleme (Functional Magnetic Resonance Imaging) iş akışı simülasyonu yapmışlar ve bu simülasyon üzerinde önceden tanımlanmış olan köken sorgularını yürütmüşlerdir. Ardından ikinci köken zorluğu ortaya çıkmıştır. İkinci köken zorluğu, köken bilgi alışverişi yapılırken sistemlerin birlikte çalışılabilirliğini sağlamaktadır. Tartışmalar sonunda, çekirdek köken gösterimi konusunda önemli bir anlaşmaya varılmıştır. Sonuç olarak, 2007 yılında Salt Lake City’de düzenlenen çalıştayda, ilgili bir veri modeli Açık Köken Modeli (Open Provenance Model) olarak yayınlanmıştır. (Moreau et al., 2008, Moreau et al., 2010)



Şekil 2.2.1 AKM olmadan köken bilgileri ve uygulamalar

Şekil 2.2.1’de Açık Köken Modeli (AKM) ortaya çıkmadan önce doğrudan köken bilgileri için kullanılan sistemler temsil edilirken, Şekil 2.2.2’de AKM kullanan sistemlerin köken bilgilerinin nasıl saklandığı temsil edilmektedir. AKM kullanılmadan önce, köken bilgileri farklı sistemlerde dağınık halde tutulmaktadır. Köken bilgileri, ortak bir gösterim ve altyapıya sahip olmadıkları için anlaşılabilirliği ve veri alışverişi oldukça zordur. AKM ile birlikte farklı uygulama ve köken kayıtları için birlikte çalışabilirlik katmanı eklenmiş, farklı sistemlerde yer alan köken bilgi transferlerine olanak sağlanmıştır. Farklı sistemlerde yer alan köken gösterimleri anlamlı hale gelmiştir.



Şekil 2.2.2 AKM kullanıldığı sistemlerde köken bilgileri ve uygulamalar

AKM aşağıda yer alan gereksinimleri karşılamaya yönelik oluşturulmuş bir modeldir:

- Paylaşılan bir köken modeli tabanlı uyumluluk katmanı aracılığı ile sistemler arasında köken bilgi alışverişine izin verilmesi,
- Geliştiricilerin köken modeli üzerinde çalışabilir araçların inşaa etmesine ve paylaşabilmesine olanak tanınması,
- Modelin teknoloji bağımsız bir şekilde tam olarak tanımlanması,
- Bilgisayar sistemleri tarafından üretilip üretilmemesi önemli olmadan herşeyin dijital köken gösteriminin desteklenmesi,
- Köken çizgeleri üzerinde yapılabilecek geçerli çıkarımları belirleyen çekirdek kural kümesi tanımlanması. (Moreau et al., 2008, Moreau et al., 2010)

Açık Köken Modeli'ni, etki alanı ya da bir problemin çözümüne yönelik özelleştirilmek amacı ile farklı profil tanımları ortaya çıkmıştır. Dublin Core terimleri kullanılarak hazırlanan AKM profili de bunlardan biridir. Köken öncelikli olarak atıf, alıntı, bibliyografik bilgiler hakkındadır. Dublin Core (DC)

(Miles et al., 2009), bu bilgilerle ilişkili terimler sağlamaktadır. DC profili, DC terimlerini AKM kavramları ve çizgeleri üzerinde kullanmayı amaçlamaktadır.

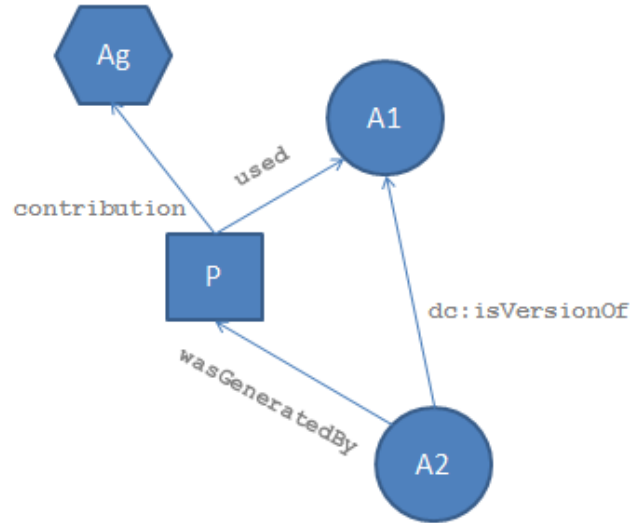
DC terimleri, kaynaklar hakkında üst veri çeşitleridir. AKM üzerinde kaynaklar, doğrudan iletişim kurmamaktadır. Kaynaklar arasındaki ilişkileri belirleme süreci, kaynakları çalıştıran fonksiyonelliğin ve kaynakların tanımlanmasıyla başlamalıdır, dolayısıyla AKM üzerinde ilişkilendirilmelidir. AKM’de üç temel kavram yer almaktadır: Artifact, Process ve Agent. Process bir eylemi ifade eder, girdi olarak Artifact nesneleri alır ve çıktı olarak yeni Artifact nesneleri oluşturur. Agent kavramı, eylemi yani Process’i gerçekleştiren özneyi temsil etmektedir. AKM’de yer alan Artifact kavramı ise, çalışma zamanı içinde değişen durumların herhangi birindeki fiziksel kaynağı ya da veriyi temsil etmektedir. Bu nedenle, aynı kaynağın farklı durumlardaki gösterimlerini bir Artifact’ı bir diğer Artifact’ın versiyonu olduğunu göstermek için bir notasyona ihtiyaç duyulmaktadır. Çizelge 2.2.1’de Dublin Core ve AKM terimleri ve bu terimler kullanılarak AKM kavramlarının nasıl ilişkilendirildiği detaylandırılmaktadır.

Çizelge 2.2.1 Dublin Core ve AKM terimlerinin AKM kavramları üzerindeki işlevleri

Dublin Core /AKM Terimleri	AKM Kavramları Üzerindeki İşlevleri
dc:isVersionOf	Kaynaklar arasındaki versiyon ilişkisinin göstermektedir.
wasDerivedFrom	Birbirinden türetilmiş kaynakları ifade etmek için kullanılmaktadır.
wasGeneratedBy	Kaynağın bir süreç ile ilişkisini göstermek için kullanılmaktadır.
wasControlledBy	Bir süreç ile etkili olan sistem ya da kişi arasındaki ilişkiyi temsil etmektedir.
dc:accrualMethod	Koleksiyona yeni öge eklemek için kullanılan metodu göstermek için kullanılmaktadır.
dc:available	Kaynağın müsait olduğu zaman aralığının tanımlanmasına yardımcı olmaktadır.
dc:contributor	Kaynağın bir sonraki versiyonunu oluması için kaynağa katkıda bulunan kişi, organizasyon veya sistemi temsil etmektedir.
dc:creator	Bir kaynağın yaratıcısı olan kişi organizasyon ya da sistemi temsil etmektedir.
dc:date	Kaynağın yaşam döngüsü içindeki bir olay ile ilişkili bir tarih veya zaman periyodunu

	göstermektedir.
dc:hasVersion	Bir kaynağın ilişkili olduğu versiyonu göstermektedir.

Şekil 2.2.3'te DC Profil kullanılarak iki Artifact nesnesi arasında versiyon ilişkisi kurulması örneklendirilmektedir. Ag isimli Agent, A1 isimli Artifact'ı kullanarak, P Process'ini gerçekleştirmektedir. P Process'i tarafından A2 Artifact'ı oluşturulmaktadır. A2 Artifact'ı, P Process'i ile A1 Artifact'ına katkı yapılarak oluşturulduğundan, A1 Artifact'ı ile A2 Artifact'ı arasında versiyon ilişkisi bulunmaktadır. Bu ilişki, dc:isVersionOf bağlantısı ile gösterilmektedir.



Şekil 2.2.3 DC Profil üzerinde versiyon ilişkisi örneği

AKM ve DC terimleri ile tez kapsamında yaratılan gizlilik farkında köken modelinde tanımlanan terimlerin eşleştirmeleri Bölüm 3'te detaylı olarak açıklanmaktadır.

2.3 ICD-10

ICD (International Statistical Classification of Diseases and Related Health Problems), Dünya Sağlık Örgütü (WHO) tarafından hastalıkların uluslararası sınıflandırılmasının yapıldığı tıbbi sınıflandırma listesidir. ICD-10 ise, bu sınıflandırma listesinin onuncu revizyonunu ifade etmektedir. Bu liste; hastalıklar,

belirti ve bulgular, anormal bulgular, şikayetler, sosyal koşullar ve yaralanma ya da hastalıkların dış nedenleri için tanımlanmış kodları içermektedir.

Ölüm bildirileri ve sağlık kayıtlarını içeren, sağlıkla ilgili tüm hayati/hayati olmayan kayıtların, hastalıklar ve diğer sağlık problemleri olarak sınıflandırılması için doktorlar, hemşireler, diğer sağlayıcılar, araştırmacılar, sağlık bilgileri yöneticileri ve kodlayıcıları, sağlık bilgi teknolojisi çalışanları, politika tanımlayıcılar, sigortacılar ve hasta örgütleri tarafından ICD kullanılmaktadır. ICD; klinik, epidemiyolojik ve kalite amaçları için tanı bilgilerinin depolanması ve sorgulanmasına olanak verir, bu kayıtlar ölüm ve hastalık istatistiklerinin derlenmesi için temel oluşturmaktadır.

ICD kodları, 43 dile tercüme edilmiştir ve 117 ülke tarafından kullanılmaktadır. Çok sayıda ülke, ölüm verileri ve sağlık durumunun birincil göstergesi olarak bu sistemi kullanmaktadır.

ICD-10, Mayıs 1990'da 43. Dünya Sağlık Asamblesi tarafından onaylanmış ve 1994'ten sonra Dünya Sağlık Örgütü üyesi devletler tarafından kullanılmaya başlanmıştır. ICD revizyon süreci hala devam etmektedir. Bir sonraki sürüm olan ICD-11'in, 2017 yılında yayınlanması planlanmaktadır.

ICD KODU	TANI
A00	Kolera
A00.0	Kolera, <i>Vibrio cholerae</i> 01, biovar kolera'ya bağlı
A00.1	Kolera, <i>Vibrio cholerae</i> 01, biovar eltor'a bağlı
A00.9	Kolera, tanımlanmamış
A01	Tifo ve paratifo
A01.0	Tifo
A01.1	Paratifo A
A01.2	Paratifo B
A01.3	Paratifo C
A01.4	Paratifo, tanımlanmamış
A02	Salmonella enfeksiyonları, diğer
A02.0	Salmonella enteritisi
A02.1	Salmonella septisemisi
A02.2	Salmonella enfeksiyonları, lokalize
A02.8	Salmonella enfeksiyonları, diğer tanımlanmış
A02.9	Salmonella enfeksiyonu, tanımlanmamış
A03	Şigelloz
A03.0	Şigelloz, <i>Shigella dysenteriae</i> 'ye bağlı
A03.1	Şigelloz, <i>Shigella flexneri</i> 'ye bağlı
A03.2	Şigelloz, <i>Shigella boydii</i> 'ye bağlı
A03.3	Şigelloz, <i>Shigella sonnei</i> 'ye bağlı
A03.8	Şigellozlar, diğer
A03.9	Şigelloz, tanımlanmamış
A04	Bakteriyel barsak enfeksiyonları, diğer
A04.0	Enteropatojenik <i>Escherichia coli</i> enfeksiyonu

Şekil 2.3.1 ICD-10 kodları ve tanıları

Şekil 2.3.1’de ICD-10 kodları listesinin bir kısmı örnek olarak gösterilmektedir. Tez kapsamında, bulaşıcı hastalıklar etki alanında çalışmalar yapıldığından, Bölüm 4’te detaylarına yer verilen, Bulaşıcı Hastalıklar Ontolojisi içinde tanımlanan hastalıklar için ICD-10 kodlarına ihtiyaç duyulmaktadır.

2.4 ALCQ DL Tanımlama Dili

DL (Description Logic), yapısal, resmi ve iyi anlaşılabilir bir yolla uygulama etki alanındaki bilgilerin gösterimi için kullanılan resmi bilgi gösterim dilleri ailesidir. Kavram açıklamaları tarafından tanımlanan etki alanındaki önemli notasyonların gösterilmesine yardımcı olmaktadır. İfadeler, atomik roller ve atomik kavramlar kullanılarak oluşturulmaktadır (Baader et al., 2007).

DL ifadesinin yanında yer alan eklerin her biri farklı, anlamlar ifade etmektedir. Tezin ilerleyen bölümlerinde kullanım senaryoları için ALCQ DL dili kullanılmıştır.

AL (Attributive Language) eklentisi; atomik kavramların olumsuzlamasını, kavram kesişmesini, evrensel kısıtlamalar ve sınırlı varoluşsal nicelleştirmeye izin verdiği anlamına gelmektedir (Rudolph, 2011).

Q eklentisi, nitelikli önem düzeyi kısıtlamalarına izin verdiğini anlamına gelmektedir. OWL2’den sonra kullanılabilir olmaktadır (Rudolph, 2011).

C, karmaşık kavram olumsuzlamasına izin verildiği anlamına gelmektedir (Rudolph, 2011).

ALCQ DL dilinde kullanılan sembollerden bazılarının açıklamalarına Çizelge 2.4.1’de yer verilmiştir (Rudolph, 2011).

Çizelge 2.4.1 ALCQ DL dilinde kullanılan semboller ve anlamları

Semboller	Açıklamalar
$\top$	Üst kavramı
$\perp$	Alt kavramı
$\sqcap$	Kesişim kavramı
$\sqcup$	Birleşim kavramı
$\neg$	Olumsuzlama
$\forall$	Genel kısıtlama kavramı, tüm öğeler için kısıtlama geçerli
$\exists$	Varolan öğeler için kısıtlama, bazı öğeler için geçerli
$\sqsubseteq$	İçerme kavramı
$\equiv$	Denklik

2.5 Literatürde Yer Alan Çalışmalar

Bu bölüm altında; Açık Köken Modeli, köken ve Anlamsal Web kavramları kullanılarak yapılan çalışmalara yer verilmektedir. Ayrıca, bulaşıcı hastalıklar üzerine tanımlanan modeller için literatürde yer alan çalışmalardan bahsedilmektedir.

Köken kavramı; iş akışı sistemleri, e-bilim, Anlamsal Web ve veritabanları gibi pek çok alanda araştırmanın bir parçası olarak yer almaktadır. Köken, iş akışlarında sonucun nasıl ortaya çıktığının anlaşılması ve gerçekleştirilen işlemlerin tekrarlanabilirliği açısından önem taşımaktadır. Bilimsel iş akışlarında tüm gerçekleştirilmiş işlemler üzerinde köken sorgularına açık şekilde cevaplar getirmek gizli bilgileri açığa çıkarabilir. Sınırlı/sınırsız sayıda köken sorgusu için kanıtlanabilir gizlilik desteği sağlanarak veri, modül yapısal gizliliği analiz etme ve iş akışı özellikleri depoları üzerinde yapısal sorguların cevaplanması hakkında bir çalışma önerilmiştir (Davidson, 2010). Tez çalışmasında, iş akışlarına odaklanılmadığından bu öneri kullanılmamaktadır.

E-bilim alanında ise, sorgu anında kullanıcı kararlarının kökenini türetmek için bir yaklaşım geliştirilmiştir. Bu yaklaşım, insan kararlarını simule etmek için geliştirildiğinden, doğrudan insan psikolojisi üzerine modellenmiştir. Karar

kökeni, Açık Köken Modeli üzerinde modellenmiş, geniş bir köken grafiği üzerinde sorgulamayı desteklemiş ve kullanıcı kararlarına karşı köken sorgularının tutarlı olmasını sağlamıştır (Nunes et al., 2012). Tez çalışmasında, karar kökeni konusuna odaklanılmadığından bu öneri kullanılmamaktadır.

Her sistem için yeni köken yaklaşımları oluşturuldukça, birbirinden oldukça farklı köken modelleri ortaya çıkmış ve sistemler arasında köken bilgi alışverişi problem olmuş, ayrıca köken bilgisi gösterimlerinde farklılıklar nedeniyle köken bilgi gösterimleri farklı yorumlara açık hale gelmiştir. Farklı köken sistemlerinin yeteneklerini, köken gösterimlerinin açıklayıcılığını anlamlandırma, farklı sistemler arasında köken bilgilerinin alışverişi ve birlikte çalışabilirliğinin zorluğu nedeniyle, tüm sistemler tarafından kullanılabilecek ortak bir model (Açık Köken Modeli) geliştirilmiştir (Moreau et al., 2008). Bu model, tez çalışmasında veri kökeninin tutulması için geliştirilen modelin altyapısında kullanılmaktadır.

Köken kavramı, yukarıda bahsediliği gibi verilerin türetilme süreçlerinin izlenmesine ve analiz edilmesine olanak sağladığı için veri gizliliğini korumada tercih edilmektedir. Bu sebeple, detaylı, özel tanımlanmış politikalar, kişisel tercihler ve farklı uygulanabilen politikalardan karar toplanması gibi özel istekleri destekleyen köken kontrol dilleri önerilmiştir. Bu öneriler verilerin güvenliği üzerine yoğunlaşmıştır (Ni et al., 2009). Verilerin güvenliğini sağlamak üzere politikalardan yararlanılarak oluşturulmuş bir köken kontrol yaklaşımı önerilmiştir (Ni et al., 2009). Tez çalışması kapsamında, biz de veri güvenliğinin korunmasına yönelik bir köken modeli geliştirdik. Ancak bu model üzerinde, verileri izinsiz erişimlerden korumak için gerekli izin ve yasak tanımlarını, bu köken kontrolü yaklaşımında (Ni et al., 2009) olduğu gibi politikalar kullanılarak değil de; roller, izin ve yasaklar için tanımlanmış sınıflar ile kayıtlı veriler ilişkilendirilerek sağlanmaktadır. Bu yöntem ile; erişim amacı, tarih aralığı ve rol dikkate alınarak veriler erişilebilir ya da erişilemez hale gelmektedir. Bu yaklaşımda (Ni et al., 2009) ise; erişim amacı, koşullar altında verilmiş, doğrudan sorgulanabilir durumda değildir ve erişim izni veya erişim kısıtlaması için tarih aralığı alanı bulunmamaktadır. Ek olarak, kişiler ve roller ile ilişkilendirme koşullar ya da kısıtlamalar altında ifade edilmekte ve doğrudan sorgulanabilir

durumda değildir, önce bu alandaki ifadenin çözümlenmesi gerekmektedir. Bu nedenle, bu yaklaşım (Ni et al., 2009) tez çalışmasında kullanılmamaktadır.

Sağlık etki alanında, veri güvenliği oldukça önemli bir konu olduğu için, köken kavramı sağlık uygulamalarında da kullanılmaktadır. Örneğin, Organ Nakli Yönetimi etki alanında uygulamalar geliştirilmiştir. Gerçekleştirilen bağışların kayıtları, bağışın özgün bilgileri Organ Bağış Yönetimi Uygulaması(Organ Transplant Management Application - OTMA) tarafından vakalara göre saklanmaktadır. Bir OTMA, farklı kurumlarda dağınık halde tutulan hasta kayıtlarının, nakil verilerinin yönetilmesini ve bu verilere erişilmesini sağlamaktadır. Bir organ nakli senaryosu gerekli laboratuvar testleri ve hasta kayıtlarından sorumlu olan bir ya da birden fazla hastane ünitesi içerebilmektedir. Bu durumda organ nakli için bekleyen hastalar için bir liste, nakil yapılabilecek donörlerin bilgileri, gerçekleştirilen organ nakli kayıtları vaka bazlı olarak tek bir sistemde tutulması gerekmektedir. Bir OTMA, nakil öncesi donör tespiti ve değerlendirilmesi, naklin gerçekleştirilmesi, nakil sonrası alıcının sağlık durumunun izlenmesi aşamalarını desteklemektedir. Naklin neden vücut tarafından reddedildiği, kurallara uygun olarak gerçekleştirilip gerçekleştirilmediği, hastanın uygun donör ya da alıcı olup olmadığı gibi konular analiz edilebilmektedir (Kifor et al., 2007; Álvarez et al., 2006; Álvarez et al., 2005). Tez çalışmasında, organ nakli etki alanına odaklanılmadığından, bu yaklaşımdan yararlanılmamaktadır.

Bir başka yaklaşım ise, elektronik sağlık sistemlerinden toplanmış klinik verilerin dönüşümlerinin kökeninin araştırılması için önerilmiştir. Verinin orijinal formundan mevcut haline nasıl geldiği, verinin nereden geldiği gibi çıkarımlar elde edilebilmektedir. Klinik veriler, klinik denemeler için hastaların belirlenmesi ve takibe alınması için yeniden tasarlanabilmektedir. Bu çalışma aynı sonucu yeniden elde edebilme için klinik verilerinin tutarlı ve iyi formüle edilmiş olmasını sağladığından önemlidir, böylece verinin dönüşüm süreci daha şeffaf ve izlenebilir olmaktadır (Bache et al., 2013; Patel et al., 2007). Bu yaklaşım, verilerin dönüşüm süreçlerine odaklanmakta ve belli bir hastane sistemindeki kayıtlardan yararlanılarak veriler elde edilmektedir. Tez çalışmasında ise, verilerin dönüşüm süreçleri için varolan veriler üzerinden çıkarsama yapılmamakta,

süreçler işletildikçe erişim geçmişleri ve detaylar kaydedilip, bu kayıtlar üzerinden veri kökenine ulaşılmaktadır.

Geleneksel yaklaşımlarda çalıştırılan kuralların sırası kontrol edilerek uygulamanın gerçekleştirimi esnasında kural etkileşimleri tutulmaktadır. Bu şekilde bir yaklaşım yerine, karar destek sistemleri için klinik veriyi göstermeyi sağlayan kavramsal bir şema geliştirilmiştir. Bu şema kuralın kendi mantığını, klinik içeriği ve bağlamını ele almaktadır. Bu kavramsal şema yaklaşımı ile, sağlık hizmetlerinin kalite, güvenlik ve etkinliğini arttırmaya yönelik yetkili bilgi havuzlarının geliştirilmesi, kuralların tutarlı bir şekilde gerçekleştirimi ve bakımının yapılması amaçlanmaktadır (Sordo et al., 2012). Tez çalışmasında, klinik veriler üzerinde karar destek çalışmaları yer almadığından, bu yaklaşımdan yararlanılmamaktadır.

Köken yönetimi sistemleri, sahibine özel verilerin bir kısmı saklanırken taraflar arasında köken veri paylaşımı için oluşturulmuş güçlü modellerden yoksundur. Köken verilerinin ortaklar arasında sınırlı ve güvenilir bir şekilde paylaşılabilir olması kritiktir. Sağlık etki alanı üzerinde köken verisinin paylaşılmasını sağlamak için köken çizgeleri üzerinde soyutlamalar yaratarak gizliliği korumak için bir metot önerilmiştir (Missier et al., 2013). Bu metot, taraflar arasında köken bilgisinin gizliliğinin korunmasına odaklanırken, tez çalışmasında geliştirilen köken modelinde ise, kullanıcılar arasında köken bilgisinin gizliliği korunmaktadır, farklı taraflar işin içine dahil olmamaktadır.

Tez kapsamında geliştirilen köken modelinin, bulaşıcı hastalıklar etki alanı üzerinde kullanımını göstermek için bulaşıcı hastalıklar ontolojisine ihtiyaç duyulmaktadır. Bu nedenle, literatürde yer alan bulaşıcı hastalıklar etki alanı için oluşturulmuş olan ontolojiler incelenmiştir. Örneğin, hesaplama uygulamaları üzerinde ontolojinin yararlarını vurgulamak için bulaşıcı hastalıklar alanında ontoloji tabanlı hesaplama üzerinde bir çalışma yapılmıştır. Bulaşıcı hastalıklar araştırmaları ile ilişkili kelime kaynakları ve tanımlar belirlenmiştir. Bu çalışma altında bulaşıcı hastalıklar ontolojisi (Infectious Disease Ontology, IDO) oluşturulmuştur (Cowell et al., 2010). Bir diğeri ise, Open Biomedical Ontology (OBO) prensipleri baz alınarak bulaşıcı hastalıklar ontolojisi geliştirilmiştir.

Temel üst ontoloji olarak Basic Formal Ontology (BFO) kullanılmıştır (Goldfain et al., 2010). Tez kapsamında, literatürde yer alan bulaşıcı hastalık ontolojileri, bulaşıcı hastalıklar etki alanında çalışmak için kullanmayı planladığımız korunma yöntemleri, risk grupları, vücut bölümleri, belirtiler, teşhis kriterleri ve kamu sağlığı sorumluluğu kavramlarını karşılayan tüm terimleri içermediğinden ve her hastalık belli özellikler altında kategorize edildiğinden kullanılmayıp, hastalıkları kategorize edilmeden ve ihtiyaç duyduğumuz terimlerin yer aldığı bir bulaşıcı hastalıklar ontolojisi oluşturulmuştur.

Bulaşıcı hastalıklar ontolojisi, geliştirilen köken modeli ile entegre edildikten sonra, bulaşıcı hastalıklardan korunma yöntemi olarak aşılarda da tez çalışmasında geliştirilen modele dahil edilmesi gerekmektedir. Bu nedenle literatürde yer alan aşı ontolojileri incelenmiştir. Aşı bilgisinin tutarlı olarak temsil edilmesini sağlamak ve otomatik tümevarımı desteklemek için bir aşı ontolojisi (Vaccine Ontology, VO) geliştirilmektedir. Bu aşı ontolojisi geliştirimi, uluslararası işbirliği içinde gerçekleştirilmektedir (He et al., 2008). Bir diğer aşı ontolojisi ise, doktora tezi olarak geliştirilmiş olan Ulusal Aşı Ontolojisi (UAO)'dır. Bu ontolojide, aşı ile ilgili kavramların yanı sıra, aşı takvimi kavramı da yer almaktadır. UAO'da aşılarda, bulaşıcı hastalıklar ile eşleştirilirken; aşılama takvimleri de, kişiler ile eşleştirilmektedir (Sezer, 2014). Bu nedenle, VO aşılarda ile ilgili çok detaylı kavramlar içermesine rağmen, tez çalışmasında UAO kullanılmaktadır.

Ek olarak, tez kapsamında geliştirilen gizlilik farkında köken modeli, farklı etki alanlarına uygulandığında kişisel bilgilere de ihtiyaç duyulmaktadır. Bu model içinde, kişisel bilgileri saklamak yerine Friend of a Friend (FOAF) projesinden yararlanılmaktadır. FOAF, RDF dilinde temel profil gösterimidir ve bir kişinin demografik özelliklerinin gösterilmesini sağlayan en yaygın dokümandır. FOAF, farklı etki alanlarında kişisel bilgileri tanımlamak için yaygın olarak kullanılmaktadır (Bursa, 2014). Tez çalışmasında; hasta, doktor, laboratuvar asistanı, hemşire profilleri ve kişilerin demografik ve dinamik özellikleri ile oluşturulmuş kişi profillerini içeren bir FOAF çalışmasından yararlanılmaktadır (Bursa, 2014).

2.6 Kullanılan Araçlar

Bu bölüm altında tezde geliştirilen modelin gerçekleştirimi için kullanılan araçlardan bahsedilmektedir.

2.6.1 Eclipse

Eclipse, açık kaynak kodlu ve özgür bir tümleşik geliştirme ortamıdır (Integrated Development Environment, IDE). Ana odak noktası Java ve Java ile ilişkili teknolojiler olsa da, esnek yapısı sayesinde C ve Python gibi farklı diller için de kullanılmaktadır (The Eclipse Foundation, 2014).

Eclipse projeleri Eclipse Vakfı tarafından yönetilmektedir. Eclipse Vakfı kar amacı gütmeyen, üyeleri tarafından desteklenen, Eclipse açık kaynak kodlarını sağlayan ve hem açık kaynak topluluğunu ilerleten hem de tamamlayıcı ürünler ve servislerini içinde barındıran bir kuruluştur (The Eclipse Foundation, 2014).

Eclipse IDE ek yazılım bileşenleriyle genişletilebilmektedir. Bu yazılım bileşenleri plug-in olarak adlandırılmaktadır. Ayrıca genel amaçlar için oluşturulacak uygulamalar için taban olarak da kullanılabilir. Bu uygulamalar Eclipse Zengin İstemci Platformu (Eclipse Rich Client Platform, Eclipse RCP) uygulamaları olarak da bilinmektedir (The Eclipse Foundation, 2014).

Genel kullanıma açık Eclipse lisansı (Eclipse Public Licence, EPL); Eclipse Vakfı tarafından kendi yazılımları için kullandığı açık kaynaklı lisans yazılımıdır. EPL lisanslı programlar kullanılabilir, değiştirilebilir, kopyalanabilir, ücretsiz dağıtılabilir ve EPL yazılım lisansına sahip olanlar bunu kapalı kaynak kodlu programlarda kullanabilirler (The Eclipse Foundation, 2014).

Tez projesinde, Eclipse IDE sürümü olan Eclipse Juno kullanılmıştır.

2.6.2 Apache Jena Adaptor

Apache Jena, Anlamsal Web ve Bağlı Veri (Linked Data) uygulamaları inşa etmek için kullanılan ücretsiz ve açık kaynak kodlu Java çatısıdır. Bu çatı, RDF verilerini işlemek için farklı uygulama eklentileri (API) ile birlikte kullanılmaktadır (The Apache Software Foundation, 2013).

Eclipse üzerinde proje geliştirilirken, Jena kütüphane dosyaları proje içine eklenmektedir. Tez kapsamında, geliştirilen Anlamsal Web tabanlı modellere, yeni nesneler eklemek ve varolan nesneleri sorgulayıp güncellemek için kurulmuş olan SPARQL sorgularını çalıştırmak amacıyla kullanılmaktadır.

2.6.3 Protégé

Protégé, ücretsiz açık kaynak kodlu bir ontoloji editörüdür. Aynı zamanda bir bilgi edinme sistemidir. Protégé, ontoloji tanımlamak için kullanıcı arayüzü sağlamaktadır. Ek olarak, bir ontolojinin analizine dayalı yeni bilgilerin anlaşılması ve modellerin tutarlı olduğunun doğrulanması için tümdengelimli sınıflandırıcılar içermektedir. Protégé, Stanford Üniversitesi ile Manchester Üniversitesi'nin işbirliği ile Java tabanlı olarak geliştirilmiştir (Stanford University, 2014).

Tez kapsamında, geliştirilen Anlamsal Web tabanlı gizlilik farkında köken modeli ontolojisi ve bulaşıcı hastalıklar ontolojisinin oluşturulması için kullanılmaktadır.

2.6.4 SPARQL

SPARQL, RDF erişimini sağlamak için kullanılan bir protokol ve sorgu dilidir. Bir sorgu dili olarak, modelde tutulan bilgilerin sorgulanmasında veri odaklı çalışmaktadır. SPARQL sorguları çalıştırıldığında, kurulan sorgunun sonucu olarak veri kümesi ya da RDF grafiği elde edilmektedir. SPARQL, gerekli veya opsiyonel çizge desenlerini çizgede yer alan bağlantıları kullanarak sorgulamaya olanak vermektedir. Ayrıca bu dil; toplama, olumsuzlama, alt sorgular, yeni veriler yaratma, eski verileri güncelleme ve silme işlemlerinin yapılabilmesini sağlamaktadır (The Apache Software Foundation, 2015).

Tez kapsamında geliştirilen gizlilik farkında köken modeli ontolojisi, bulaşıcı hastalıklar ontolojisi ve aşı ontolojisi üzerinde yer alan verileri sorgulamak, yeni veriler yaratmak ve varolan verileri güncellemek amacıyla SPARQL sorguları kullanılmaktadır.

2.6.5 Oracle 12c

Oracle veritabanı, bir nesne ilişkisel veritabanı yönetim sistemidir (Oracle Corporation, 2015). Oracle veritabanı, Anlamsal Web desteği bulunduğundan tezde kullanılacak modellerin saklanması için tercih edilmiştir (Stephens et al., 2006).

Oracle 10g R2 sürümüyle beraber, Oracle veritabanına RDF ve RDFS desteği gelmiştir. Bu destek ile ontolojiler, veritabanında saklanmaya başlanmıştır ve Anlamsal Web uygulamaları ile sorgu yapma olanağı sağlanmıştır. Veritabanında, RDF ve RDFS ile oluşturulmuş ontolojiler içerisinde yer alan üçlemeler kullanılarak saklanmaktadır. Oracle 11g sürümünde, OWL desteği de gelmiştir (Oracle Corporation, 2015).

Tez kapsamında, Anlamsal Web desteğinin devam ettiği, 2013 yılında yayınlanan Oracle 12c sürümü kullanılmıştır. OWL ontoloji dili kullanılarak geliştirilen anlamsal veri modellerinin, veritabanında saklanıp, geliştirilen uygulama ile sorgulanmasına olanak sağlamaktadır (Oracle Corporation, 2015).

3. GİZLİLİK FARKINDA ANLAMSAL KÖKEN MODELİ

Tez kapsamında, Dublin Core terimleri ile hazırlanan Açık Köken Model Profili kullanılarak Gizlilik Farkında Anlamsal Köken Modeli geliştirilmektedir. Ardından, bu modelin bulaşıcı hastalıklar etki alanı üzerinde yer alan hasta kayıtlarının güvenlik ve gizliliğini sağlamak üzere kullanımına yer verilmektedir. Geliştirilen bu model, istenilen herhangi bir etki alanı üzerinde güvenlik ve gizliliği sağlamak, güvenlik açıklarını tespit etmek için kullanılabilecek şekilde tasarlanmaktadır. Bu model, aşağıda yer alan soruları yanıtlamak için kullanılmaktadır:

- Verinin ilk versiyonu ne şekildedir?
- Veri üzerinde değişiklik yapanlar kimlerdir?
- Veri üzerinde yapılan işlemler nelerdir?
- Verinin versiyonları nelerdir?
- Veri için tanımlanmış olan erişim izin ve yasakları nelerdir?
- Veriye kimler tarafından erişim gerçekleştirilmiştir?
- Veri erişimi sırasında herhangi bir güvenlik ihlali yapılmış mıdır?

Geliştirilen modelin güvenlik ve gizliliği desteklemesi için kişisel bilgilerin paylaşımı kontrol edilmeli, açığa çıkması engellenmeli ve istenmeyen erişimlerden korunmalıdır. Köken kavramı, verinin orijinalinin tanımlanmasına ve yetkilerinin izlenmesine olanak sağladığından gizlilik ihtiyaçlarına yönelik olarak güvenlik kontrolü için kullanılabilmektedir. Bu nedenle, model geliştirilirken AKM ve Dublin Core terimleri ile hazırlanan Açık Köken Model Profili(Dublin Core Profili)’nden yararlanılmaktadır. AKM ve Dublin Core Profili ile ilgili detaylar Bölüm 2.2’de verilmektedir.

Dublin Core Profili’nde yer alan Dublin Core terimleri, kaynaklarla ilgili üst veri(metadata) çeşitleridir. AKM’de bir kaynak doğrudan AKM modele bağlanmaz, bu nedenle kaynaklar ve kaynakları değiştiren işlemlerin tanımlanması için bir eşleştirme süreci vardır. Farklı durumlardaki kayıtlar ya da fiziksel kaynaklar birer AKM nesnesi olarak gösterilmektedir. Dublin Core

kaynaklar arasındaki bu ilişkiyi ve diğer bir kaynaktan türemiş yeni kaynakları gösteren terimler sunmaktadır. Böylece, model üzerinde versiyon bazlı olarak nesneler tutulabilmektedir. Gizlilik Farkında Anlamsal Köken Modeli'nde, bu versiyonlama, kaynaklar arasındaki ilişkileri ve kaynaklar üzerinde yapılan değişiklikleri takip edebilme özelliklerinden faydalanmak için Dublin Core terimlerine ihtiyaç duyulmaktadır. Model geliştirilirken, Dublin Core terimlerinin bazıları doğrudan kullanılırken, bazıları ise dolaylı olarak kullanılmıştır. Aşağıda yer alan çizelgelerde doğrudan ve dolaylı olarak kullanılan Dublin Core terimlerine yer verilmektedir.

Model geliştiriminde öncelikle sınıflar tanımlanmaktadır: Subject, Person, Organization, Service, Artifact, Operation, Authorization, Prohibition, Permission, Right, Purpose, Role, AccessHistory, Object, Gender, MaritalStatus. Modelde yer alan sınıfların bulaşıcı hastalıklar etki alanı üzerindeki işlevleri Çizelge 3.1'de yer almaktadır. Ayrıca modelde yer alan sınıfların herbiri için de detaylı tablolar oluşturulmaktadır.

Subject sınıfı, yeni bir veri yaratma, varolan bir kayıt üzerine ekleme ya da değişiklik yapma gibi işlemleri gerçekleştiren aktörü ifade etmektedir. Bir kişi, servis ya da organizasyon aktör olabilir. Bu nedenle Subject sınıfının alt sınıfı olarak Person, Organization ve Service sınıfları tanımlanmaktadır. Person sınıfı bir kişiyi temsil ederken, Organization sınıfı bir kuruluşu ve Service sınıfı ise kayıtlar üzerinde çeşitli işlemler yürüten servisleri temsil etmektedir. Artifact, bir işleme girdi olarak verilen ya da bir işlem gerçekleştikten sonra çıktı olarak alınan veriyi ifade etmek için oluşturulmaktadır. Bu veri XML doküman ya da sadece düz metin gibi farklı formatlarda olabilir. Operation sınıfı ise, bir aktör tarafından gerçekleştirilen güncelleme, ekleme gibi bir işlemi ifade etmek için yaratılmaktadır. Operation sınıfı, bir Artifact nesnesini girdi olarak almakta ve çıktı olarak yine bir Artifact nesnesi yaratmaktadır. Authorization sınıfı, bir Artifact nesnesi için yetkileri ifade etmektedir. İzin ve yasakların tanımlanmasını sağlamaktadır; izinler için Permission, yasaklar için ise Prohibition alt sınıflarına sahiptir. Right sınıfı izin ve yasakların hangi haklar kapsamında olacağını belirtmektedir. Right sınıfından yaratılan öğeler, yazma, okuma, çalıştırma gibi hakları tanımlamaktadır. Purpose sınıfı, veri erişim izninin ya da erişim yasaklarının hangi erişim amacı koşulunda geçerli olacağını belirtmek için yaratılmaktadır. Role sınıfı, yaratılan Subject nesnesinin hangi role sahip olacağını ifade

etmektedir. İzin ve yasaklar; Subject ve Role sınıfları kullanılarak, belirli bir kişiye ya da belirli bir role yönelik olarak tanımlanmaktadır. AccessHistory sınıfı, bir Artifact nesnesine yaratıldığı andan itibaren tüm erişim detaylarının saklanması amacıyla yaratılmaktadır. AccessHistory nesneleri üzerinde sorgu yapılarak, izin ve yasaklara uygun olarak erişimin gerçekleşip gerçekleşmediği kontrol edilmektedir. Object sınıfı ise, bir işlemin ne için gerçekleştirildiğini ifade etmektedir. Bu sınıf bulaşıcı hastalıklar ontolojisi ile güvenlik ve gizlilik destekleyen köken modelin bağlanmasını sağlayan bir sınıftır. Örneğin; Hepatit B aşısı bir Operation nesnesidir. Hepatit B aşısı, Hepatit B hastalığı için yapılmaktadır. Hepatit B hastalığı ise, bulaşıcı hastalıklar ontolojisi üzerinde yer alan bir ögedir.

Çizelge 3.1 Gizlilik Farkında Anlamsal Köken Modeli’nde yer alan sınıflar

Sınıf	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
Subject	dct: Agent dct: creator1 dct: contributor1 dct: publisher1	Sağlık personeli, hasta, bir sağlık organizasyonu veya bir sağlık servisi.
Person	dct: Agent dct: creator1 dct: contributor1 dct: publisher1	Subject sınıfının alt sınıfı olarak tanımlanmıştır. Sağlık personeli veya hasta.
Organization	dct: Agent dct: creator1 dct: contributor1 dct: publisher1	Subject sınıfının alt sınıfı olarak tanımlanmıştır.
Service	dct: Agent dct: creator1 dct: contributor1 dct: publisher1	Subject sınıfının alt sınıfı olarak tanımlanmıştır.
Artifact	dct:Collection dct:BibliographicResource	Kişisel bilgiler, klinik test sonuçları, teşhis ve uygulanan tedaviler.
Operation	dct:instructionalMethod	Klinik testler ve tedavi süreçleri.
Authorization	Karşılığı yok.	Kişisel bilgiler, klinik test sonuçları gibi Artifact nesnelere erişip erişilemeyeceğine ilişkin

¹ Değer aralığı(Range): Subject, Person, Organization, Service.

		yetkiler.
Prohibition	Karşılığı yok.	Authorization alt sınıfı olarak tanımlanmış olan erişim yasakları.
Permission	Karşılığı yok.	Authorization alt sınıfı olarak tanımlanmış olan erişim izinleri.
Right	dct:rights ²	dct:rights ile belirlenebilecek hakların kümesi.
Purpose	Karşılığı yok.	Permission ve Prohibition nesneleri tanımlanırken hangi durumlarda geçerli olacaklarını tutan sınıf.
Role	Karşılığı yok.	Tanımlı her nesnenin bulaşıcı hastalıklar çalışma alanındaki rolü. (hasta, doktor, hemşire vb.)
AccessHistory	Karşılığı yok.	Artifact nesnelere yapılan erişimlere ilişkin detay bilgilerin tutulduğu nesne.(Kim tarafından, ne zaman, ne amaçla erişildiği tutulur.)
Object	Karşılığı yok.	Bulaşıcı hastalıklar, aşılar gibi farklı bir çalışma alanında yer alan terimlerden yaratılan nesnelerin tutulacağı genel sınıf.
Gender	Karşılığı yok.	Cinsiyet bilgisi için yaratılan nesnelerin (Female, Male, Other) tutulacağı sınıf.
MaritalStatus	Karşılığı yok.	Medeni hal bilgisi için yaratılan nesnelerin (Married, Single) tutulacağı sınıf.

Subject sınıfı, yeni bir veri yaratma, varolan bir kayıt üzerine ekleme ya da değişiklik yapma gibi işlemleri gerçekleştiren aktörü ifade etmektedir. Subject sınıfı için hasRole, isOwnerOf, requests ve performs nesne özellikleri tanımlanmıştır. Bu özellikler, Çizelge 3.2’de detaylandırılmaktadır.

² Değer aralığı(Range): Right.

Çizelge 3.2 Subject sınıfının özellikleri

Subject Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
hasRole	Karşılığı yok.	Subject'in hangi role sahip olduğunu tanımlar.
isOwnerOf	Karşılığı yok.	Bir hastanın sağlık sistemine girilmiş olan hangi verinin sahibi olduğunu gösterir.
requests	Karşılığı yok.	Subject'in hangi işlemin gerçekleşmesini istediğini gösterir.
performs	Karşılığı yok.	Bir Subject tarafından hangi işlemin gerçekleştirildiğini gösterir.

Person sınıfı, yeni bir veri yaratma, varolan bir kayıt üzerine ekleme ya da değişiklik yapma gibi işlemleri gerçekleştiren kişiyi ifade etmektedir. Person sınıfı için personFirstName, personSecondName, personSurname, personFullName, personAge, personBirthdate, personIdentityNumber, personPhoneNumber, personEmail, personAddress veri özellikleri ile personMaritalStatus, personGender, hasFather, hasMother ve hasChild nesne özellikleri tanımlanmıştır. Bu özellikler, Çizelge 3.3'te detaylandırılmaktadır.

Çizelge 3.3 Person sınıfının özellikleri

Person Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
personFirstName	Karşılığı yok.	Kişinin adını tutar.
personSecondName	Karşılığı yok.	Kişinin ikinci adını tutar.
personSurname	Karşılığı yok.	Kişinin soyadını tutar.
personFullName ³	Karşılığı yok.	Kişinin tam adını tutar.
personAge	Karşılığı yok.	Kişinin yaşını tutar.
personBirthdate	Karşılığı yok.	Kişinin doğum tarihini tutar.
personIdentityNumber	Karşılığı yok.	Kişinin kimlik numarasını tutar.
personPhoneNumber	Karşılığı yok.	Kişinin telefon numarasını tutar.
personEmail	Karşılığı yok.	Kişinin elektronik posta adresini tutar.

³ personFullName=personFirstName+personSecondName+personSurname

personAddress	Karşılığı yok.	Kişinin adres bilgisini tutar.
personMaritalStatus ⁴	Karşılığı yok.	Kişinin medeni halini tutar.
personGender ⁵	Karşılığı yok.	Kişinin cinsiyet bilgisini tutar.
hasFather	Karşılığı yok.	Kişinin babasına ait bilgilerin tutulduğu Person nesnesi ile bağlar.
hasMother	Karşılığı yok.	Kişinin annesine ait bilgilerin tutulduğu Person nesnesi ile bağlar.
hasChild	Karşılığı yok.	Kişinin çocuğuna ait bilgilerin tutulduğu Person nesnesi ile bağlar.

Organization sınıfı, yeni bir veri yaratma, varolan bir kayıt üzerine ekleme ya da değişiklik yapma gibi işlemleri gerçekleştiren organizasyonu ifade etmektedir. Organization sınıfı için organizationName ve organizationDescription veri özellikleri tanımlanmıştır. Bu özellikler, Çizelge 3.4’te detaylandırılmaktadır.

Çizelge 3.4 Organization sınıfının özellikleri

Organization Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
organizationName	dct:title	Organizasyon adını tutar.
organizationDescription	Karşılığı yok.	Organizasyon tanımını tutar.

Service sınıfı, yeni bir veri yaratma, varolan bir kayıt üzerine ekleme ya da değişiklik yapma gibi işlemleri gerçekleştiren servisi ifade etmektedir. Service sınıfı için serviceName ve serviceDescription veri özellikleri tanımlanmıştır. Bu özellikler, Çizelge 3.5’te detaylandırılmaktadır.

⁴ Değer aralığı(Range): MaritalStatus

⁵ Değer aralığı(Range): Gender

Çizelge 3.5 Service sınıfının özellikleri

Service Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
serviceName	dct:title	Servis adını tutar.
serviceDescription	Karşılığı yok.	Servis tanımını tutar.

Artifact, bir işleme girdi olarak verilen ya da bir işlem gerçekleşikten sonra çıktı olarak alınan veriyi ifade etmektedir. Artifact sınıfı için artifactValue, hasFormat, isVersionOf, hasVersion, priorVersion, Format, versionInfo, Creator, Created, Available, dateCopyrighted, hasAccessHistory, hasPermission, hasProhibition, hasOwner, isInputOf, isOutputOf veri/nesne özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.6’da detaylandırılmaktadır.

Çizelge 3.6 Artifact sınıfının özellikleri

Artifact Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
artifactName	dct:title	Artifact nesnesinin adını tutar.
artifactValue	Karşılığı yok.	Kayıtlı verinin değerini tutar.
hasFormat	dct:hasFormat	Kayıtlı verinin formatı (text, xml dosyası, resim vb.)
isVersionOf	dct:isVersionOf	İki Artifact nesnesi arasındaki versiyon ilişkisini gösterir.
hasVersion	dct:hasVersion	İki Artifact nesnesi arasındaki versiyon ilişkisini gösterir.
priorVersion	dct:priorVersion	Bir Artifact nesnesinin ilk versiyonunu gösterir.
Format	dct:format	Kayıtlı verinin formatı (text, xml dosyası, resim vb.)
versionInfo	dct:versionInfo	Bir Artifact nesnesinin sayısal olarak versiyon bilgisini tutar.
Creator	dct:creator	Bir Artifact nesnesini kimin yarattığını gösterir.
Created	dct:created	Bir Artifact nesnesinin ne zaman yaratıldığını gösterir.

Available	dct:available	Bir Artifact nesnesinin geçerli olduğu tarihi gösterir.
dateCopyrighted	dct:dateCopyrighted	Telik hakkı tarihi.
hasAccessHistory	Karşılığı yok.	Bir Artifact nesnesine Bir Subject tarafından erişildiğinde yaratılan AccessHistory nesnesini bu Artifact'a bağlar.
hasPermission	Karşılığı yok.	Bir Artifact nesnesine erişilebilmesi için tanımlı olan yetkileri tutan Permission nesnesine bağlar. Erişimin izinlerinin belirlenmesini sağlar. Örneğin; bir test sonucuna iki ay boyunca sadece doktor rolüne sahip kişiler tarafından erişilebilir.
hasProhibition	Karşılığı yok.	Bir Artifact nesnesine erişilmemesi için tanımlı olan yasakları tutan Prohibition nesnesine bağlar. Erişimi sınırlandırmasını sağlar.
hasOwner	Karşılığı yok.	Artifact nesnesinin içinde yer alan bilgilerin sahiplerine bağlar. Çünkü sağlık sisteminde bulunan kişisel bilgiler ve sağlık kayıtlarının herbiri birer hastaya aittir.
isInputOf	Karşılığı yok.	Kişisel bilgiler, sağlık işlemleri için yapılan istekler ve klinik test sonuçlarının bir sağlık sürecinin girdisi olabilmesini sağlar.
isOutputOf	Karşılığı yok.	Tedavi süreci veya klinik testler sonunda ortaya çıkan sonuçlar veya değişikliklerin bir sağlık sürecinin çıktısı olmasını sağlar.

Operation sınıfı ise, bir aktör tarafından gerçekleştirilen güncelleme, ekleme gibi bir işlemi ifade etmektedir. Operation sınıfı için operationName, date, operationPerformedDate, creator, operationMinApplicableAge, operationMaxApplicableAge, created, operationPeriodForLowRisk, operationPeriodForHighRisk, type veri özellikleri ile hasInput, hasOutput, isPerformedBy, isPerformedFor, isRequestedBy, nesne özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.7'de detaylandırılmaktadır.

Çizelge 3.7 Operation sınıfının özellikleri

Operation Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
operationName	dct:title	Bir sağlık operasyonunun adını tutar.
hasInput	Karşılığı yok.	Kişisel bilgiler, operasyon istekleri ve test sonuçlarının bir sağlık işleminin girdisi olmasını sağlar.
hasOutput	Karşılığı yok.	Klinik test sonuçları, tedavi sonuçları ve sonuç değişiklikleri gibi verilerin bir işlemin çıktısı olmasını sağlar.
isPerformedBy	Karşılığı yok.	Bir sağlık işleminin kim tarafından gerçekleştirildiğini gösterir.
isPerformedFor	Karşılığı yok.	Hangi bulaşıcı hastalık için işlemin gerçekleştirildiğini gösterir.
date	dct:date	İşlemin gerçekleştirilme tarihi.
isRequestedBy	Karşılığı yok.	Bir sağlık operasyonun kim tarafından istendiğini gösterir.
creator	dct:creator	Nesnenin kim tarafından yaratıldığı bilgisi tutulur.
created	dct:created	Sağlık operasyonunun yaratıldığı tarih.
operationPerformedDate	Karşılığı yok.	Sağlık operasyonunun gerçekleştirildiği tarih.
operationMinApplicableAge	Karşılığı yok.	Sağlık operasyonunun uygulanabilmesi için asgari yaş sınırı.
operationMaxApplicableAge	Karşılığı yok.	Sağlık operasyonunun uygulanabilmesi için azami yaş sınırı.
operationPeriodForLowRisk	Karşılığı yok.	Sağlık operasyonunun düşük risk grupları için uygulanması gereken zaman aralığı.
operationPeriodForHighRisk	Karşılığı yok.	Sağlık operasyonunun yüksek risk grupları için uygulanması gereken zaman aralığı.
type	dct:type	Sağlık operasyonun çeşidi. (Clinical Test, Treatment ya da Precaution)

Permission sınıfı; erişim amacı, rol ya da kişi, erişim hakkı ve erişim izninin geçerli olduğu tarih aralığı dikkate alınarak veriye erişim izinlerinin tanımlanması amacıyla oluşturulmaktadır. Permission sınıfı için hasPurpose, hasRelatedSubject, hasRight nesne özellikleri ile

startedDate, expiredDate, creator, created, modified, contributor veri özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.8’de detaylandırılmaktadır.

Çizelge 3.8 Permission sınıfının özellikleri

Permission Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
permissionName	dct:title	Permission nesnesinin adını tutar.
hasPurpose	Karşılığı yok.	Tanımlanan Permission nesnesinin hangi koşullarda geçerli olacağı bilgisini tutar.
hasRelatedSubject	Karşılığı yok.	Tanımlanan Permission nesnesinin ilişkili olduğu kişi ya da rol.
hasRight	Karşılığı yok.	Sağlık kaydına hangi haklar kapsamında erişilebileceğini tutar. (Read, Write vb.)
startedDate	Karşılığı yok.	Permission nesnesinin geçerliliğinin başladığı tarihi tutar.
expiredDate	Karşılığı yok.	Permission nesnesinin geçerliliğinin sona erdiği tarihi tutar.
creator	dct:creator	Nesnenin kim tarafından yaratıldığı bilgisi tutulur.
created	dct:created	Permission nesnesinin yaratıldığı tarihi tutar.
modified	dct:modified	Permission nesnesinin güncellendiği tarihi tutar.
contributor	dct:contributor	Kim tarafından güncellendiği bilgisini tutar.

Prohibition sınıfı; erişim amacı, rol ya da kişi, erişim hakkı ve erişim yasağının geçerli olduğu tarih aralığı dikkate alınarak veriye erişim yasaklarının tanımlanması amacıyla oluşturulmaktadır. Prohibition sınıfı için Permission nesnesinde olduğu gibi hasPurpose, hasRelatedSubject, hasRight nesne özellikleri ile startedDate, expiredDate, creator, created, modified, contributor veri özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.9’de detaylandırılmaktadır.

Çizelge 3.9 Prohibition sınıfının özellikleri

Prohibition Sınıfları		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
prohibitionName	dct:title	Prohibition nesnesinin adını tutar.
hasPurpose	Karşılığı yok.	Tanımlanan Prohibition nesnesinin hangi koşullarda geçerli olacağı bilgisini tutar.
hasRelatedSubject	Karşılığı yok.	Tanımlanan Prohibition nesnesinin ilişkili olduğu kişi ya da rol.
hasRight	Karşılığı yok.	Sağlık kaydına hangi haklar kapsamında erişilemeyeceğini tutar. (Read, Write vb.)
startedDate	Karşılığı yok.	Prohibition nesnesinin geçerliliğinin başladığı tarihi tutar.
expiredDate	Karşılığı yok.	Prohibition nesnesinin geçerliliğinin sona erdiği tarihi tutar.
creator	dct:creator	Nesnenin kim tarafından yaratıldığı bilgisi tutulur.
created	dct:created	Prohibition nesnesinin yaratıldığı tarihi tutar.
modified	dct:modified	Prohibition nesnesinin güncellendiği tarihi tutar.
contributor	dct:contributor	Kim tarafından güncellendiği bilgisini tutar.

Purpose sınıfı, bir veriye erişim amacını ifade etmektedir. Veri için, erişim izin ve yasakları tanımlanırken, erişim amacı koşulu için bu sınıftan yararlanılmaktadır. Purpose sınıfı için purposeName, purposeDescription, creator, created, modified, contributor veri özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.10’da detaylandırılmaktadır.

Çizelge 3.10 Purpose sınıfının özellikleri

Purpose Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
purposeName	dct:title	Purpose nesnesinin adını tutar.
purposeDescription	Karşılığı yok.	Purpose nesnesine ilişkin açıklama tutulur.
creator	dct:creator	Nesnenin kim tarafından yaratıldığı bilgisi

		tutulur.
created	dct:created	Purpose nesnesinin yaratıldığı tarihi tutar.
modified	dct:modified	Purpose nesnesinin güncellendiği tarihi tutar.
contributor	dct:contributor	Kim tarafından güncellendiği bilgisini tutar.

Role sınıfı, bir Subject nesnesinin ilişkili etki alanındaki pozisyonunu ifade etmektedir. Role sınıfı için roleName, creator, created, modified, contributor veri özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.11’de detaylandırılmaktadır.

Çizelge 3.11 Role sınıfının özellikleri

Role Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
roleName	dct:title	Rolün adını/tanımını tutar. (Doctor, nurse vb.)
roleDescription	Karşılığı yok.	Role nesnesine ilişkin açıklama tutulur.
created	dct:created	Rolün ne zaman yaratıldığı bilgisini tutar.
creator	dct:creator	Rolün kim tarafından yaratıldığını tutar.
modified	dct:modified	Rolün güncellendiği tarihi tutar.
contributor	dct:contributor	Kim tarafından güncellendiği bilgisini tutar.

Right sınıfı, veri erişimi için kullanılacak olan okuma, yazma gibi hakları ifade etmektedir. Right sınıfı için rightName, rightDescription, creator, created, modified, contributor veri özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.12’de detaylandırılmaktadır.

Çizelge 3.12 Right sınıfının özellikleri

Right Sınıfı		
Özellikler	Dublin Core / AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
rightName	dct:title	Right nesnesinin adını/tanımını tutar. (Read, Write vb.)
rightDescription	Karşılığı yok.	Right nesnesine ilişkin açıklama tutulur.
created	dct:created	Right nesnesinin ne zaman yaratıldığı bilgisini tutar.
creator	dct:creator	Right nesnesinin kim tarafından yaratıldığını tutar.
modified	dct:modified	Right nesnesinin güncellendiği tarihi tutar.
contributor	dct:contributor	Kim tarafından güncellendiği bilgisini tutar.

OperationType sınıfı, tanımlanan işlemlerin türünü ifade etmektedir. OperationType sınıfı için operationTypeName, operationTypeDescription, creator, created, modified, contributor veri özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.13'te detaylandırılmaktadır.

Çizelge 3.13 OperationType sınıfının özellikleri

OperationType Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
operationTypeName	dct:title	OperationType nesnesinin adını/tanımını tutar. (Precaution, Treatment vb.)
operationTypeDescription	Karşılığı yok.	OperationType nesnesine ilişkin açıklama tutulur.
created	dct:created	OperationType nesnesinin ne zaman yaratıldığı bilgisini tutar.
creator	dct:creator	OperationType nesnesinin kim tarafından yaratıldığını tutar.
modified	dct:modified	OperationType nesnesinin güncellendiği tarihi tutar.
contributor	dct:contributor	Kim tarafından güncellendiği bilgisini tutar.

Gender ve MaritalStatus sınıfları için genderName ve maritalStatus veri özellikleri tanımlanmıştır. Bu özellikler, Çizelge 3.14'te detaylandırılmaktadır. Ancak bu özellikler doğrudan Person sınıfı ile ilişkili olduğundan, güvenlik ve gizliliği destekleyen model yapısının kapsamı dışında kalmaktadır. Bu nedenle bu özellikler yerine, kişisel bilgiler için FOAF ontolojisinden (Bursa, 2014) yararlanılmaktadır.

Çizelge 3.14 Gender ve MaritalStatus sınıflarının özellikleri

Sınıf	Özellikler	Dublin Core / AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
Gender	genderName	dct:title	Gender nesnesinin adını/tanımını tutar. (Male, Female vb.)
MaritalStatus	maritalStatusName	dct:title	MaritalStatus nesnesinin adını/tanımını tutar. (Married, Single vb.)

FOAF ontolojisi (Bursa, 2014) yardımıyla, her kişi ve her rol için bir profil yapısı tanımlanmaktadır. Kişi için tanımlanan profil yapısı ile, temel kişisel bilgilerin, Gizlilik Farkında Anlamsal Köken Modeli'nden bağımsız tutulması amaçlanmaktadır. Böylece çok fazla değişmeyen kişisel bilgiler, farklı ontolojiler tarafından ortak olarak kullanılabilmekte ve ontolojilerde gereksiz yeniden tanımlamalardan kaçınılmaktadır. Ayrıca, Role sınıfı kullanılarak tanımlanan roller, FOAF ontolojisinde yer alan profiller ile eşleştirilmektedir. Böylece tanımlı roller, belli özellikleri barındıran profil yapısı olarak kullanılabilir. Çizelge 3.15'te FOAF üzerinde kişi profili oluştururken kullanılan özellikler ve işlevlerine yer verilmektedir.

Çizelge 3.15 FOAF kişi profili özellikleri

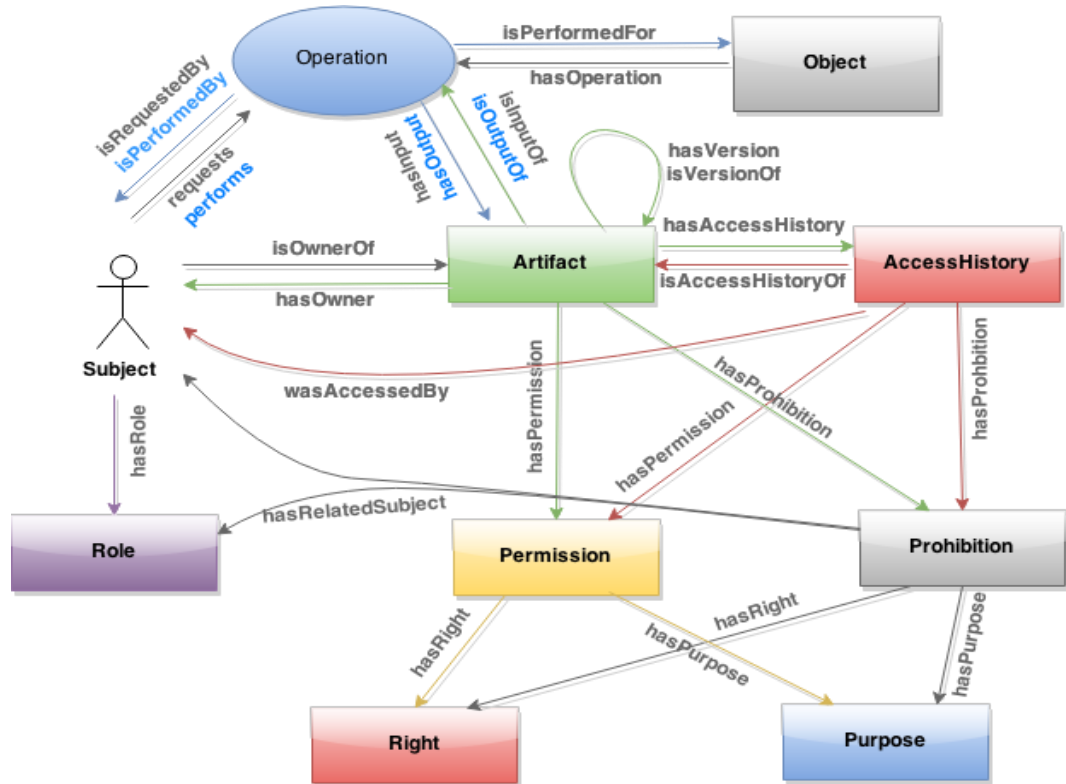
FOAF Kişi Profili Özellikleri	
Özellikler	İşlevleri
foaf:name	Kişinin tam adını tutar.
foaf:firstname	Kişinin ilk adını tutar.
foaf:givenname	Kişinin soyadı dışındaki tüm adlarını tutar.
foaf:family_name	Kişinin soyadını tutar.
foaf:age	Kişinin yaşını tutar.
foaf:birthday	Kişinin doğum tarihini tutar.

foaf:identityNumber	Kişinin kimlik numarasını tutar.
foaf:phone	Kişinin telefon numarasını tutar.
foaf:mbox	Kişinin elektronik posta adresini tutar.
foaf:hasAddress	Kişinin adres bilgisini tutar.
foaf:marital_status	Kişinin medeni halini tutar.
foaf:gender	Kişinin cinsiyet bilgisini tutar.
foaf:hasProfiles	Kişiyi, Profile ile ilişkilendirir.

AccessHistory sınıfı, bir veriye yapılan tüm erişimlerin erişim tarihi, erişim amacı, rol ve erişimi gerçekleştiren kişi bazlı saklanması amacıyla oluşturulmaktadır. AccessHistory sınıfı için accessedDate, accessDetail veri özellikleri ile isAccessHistoryOf, wasAccessedBy, created, hasPermission, hasProhibition nesne özellikleri tanımlanmaktadır. Bu özellikler, Çizelge 3.16’da detaylandırılmaktadır.

Çizelge 3.16 AccessHistory sınıfının özellikleri

AccessHistory Sınıfı		
Özellikler	Dublin Core /AKM Terimleri	Bulaşıcı Hastalıklar Çalışma Alanı
accessHistoryName	dct:title	AccessHistory nesnesinin adını tutar.
isAccessHistoryOf	Karşılığı yok.	Bir Artifact nesnesi ile ArtifactHistory nesnesi arasındaki bağlantıyı sağlar. Bir Artifact nesnesine sağlık personeli tarafından erişildiğinde bu Artifact için bir AccessHistory nesnesi yaratılır ve erişim detayları bu AccessHistory nesnesinde tutulur.
accessedDate	Karşılığı yok.	Erişimin ne zaman gerçekleştiği bilgisi tutulur.
wasAccessedBy	Karşılığı yok.	Kim tarafından erişildiği bilgisi tutulur.
accessDetail	Karşılığı yok.	Hangi amaçla, hangi role sahip olan kişinin eriştiği ve hangi değişikliği yaptığı bilgisi tutulur.
created	dct:created	Nesnenin yaratılma tarihi tutulur.
hasPermission	Karşılığı yok.	Hangi izinler dahilinde AccessHistory nesnesine erişim sağlanabileceği bilgisi tutulur.
hasProhibition	Karşılığı yok.	Hangi durumlarda AccessHistory nesnesine erişimin engelleneceği bilgisi tutulur.



Şekil 3.1 Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan atomik kavram ve roller

Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan atomik kavram ve roller Şekil 3.1'de gösterilmektedir. Bu kavram ve rollere ilişkin detaylar aşağıda yer almaktadır:

- Atomik kavramlar; Subject, Person, Organization, Service, Artifact, Operation, Authorization, Prohibition, Permission, Right, Purpose, Role, AccessHistory, Object, Gender ve MaritalStatus kavramlarıdır.
- Atomik rol olan hasRole; Subject nesnesi ile Role nesnesi arasında, hasPermission; Artifact ya da AccessHistory nesnesi ile Permission nesnesi arasında, hasProhibition; Artifact ya da AccessHistory nesnesi ile Prohibition nesnesi arasında, wasAccessedBy; AccessHistory nesnesi ile Person nesnesi arasında bir bağlantı kurulmasını sağlamaktadır.

- Atomik rol olan `hasOwner`, `Artifact` nesnesi ile `Person` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isOwnerOf` atomik rolü ile ters ilişkilidir, dolayısıyla `Person` nesnesi ile `Artifact` nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan `hasInput`, `Operation` nesnesi ile `Artifact` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isInputOf` atomik rolü ile ters ilişkilidir, dolayısıyla `Artifact` nesnesi ile `Operation` nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan `hasOutput`, `Operation` nesnesi ile `Artifact` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isOutputOf` atomik rolü ile ters ilişkilidir, dolayısıyla `Artifact` nesnesi ile `Operation` nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan `hasOperation`, `Object` nesnesi ile `Operation` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isPerformedFor` atomik rolü ile ters ilişkilidir, dolayısıyla `Operation` nesnesi ile `Object` nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan `requests`, `Person` nesnesi ile `Operation` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isRequestedBy` atomik rolü ile ters ilişkilidir, dolayısıyla `Operation` nesnesi ile `Person` nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan `performs`, `Person` nesnesi ile `Operation` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isPerformedBy` atomik rolü ile ters ilişkilidir, dolayısıyla `Operation` nesnesi ile `Person` nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan `hasVersion`, iki `Artifact` nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, `isVersionOf` atomik rolü ile ters ilişkilidir.

- Atomik rol olan *hasAccessHistory*, *Artifact* nesnesi ile *AccessHistory* nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, *isAccessHistoryOf* atomik rolü ile ters ilişkilidir, dolayısıyla *AccessHistory* nesnesi ile *Artifact* nesnesi arasında bir bağlantı oluşturmaktadır.

Gizlilik Farkında Anlamsal Köken Modeli için tanımlanan kuralların ALCQ DL dili ile ifadeleri aşağıda yer almaktadır:

- ❖ Her aktör, bir rol ile ilişkilendirilmektedir.

$$\forall \text{Subject}(\text{hasRole}(\text{Subject}, \text{Role})), \text{Role} \subseteq \text{hasRole}.\text{Subject}$$

- ❖ Her erişim geçmişi, erişimi gerçekleştiren aktör ile ilişkilendirilmektedir.

$$\forall \text{AccessHistory}(\text{wasAccessedBy}(\text{AccessHistory}, \text{Subject})), \\ \text{Subject} \subseteq \text{wasAccessedBy}.\text{AccessHistory}$$

- ❖ Her kayıt için, bir aktör ile sahiplik ilişkisi tanımlanmaktadır.

$$\forall \text{Artifact}(\text{hasOwner}(\text{Artifact}, \text{Subject})) \leftrightarrow \\ \exists \text{Subject}(\text{isOwnerOf}(\text{Subject}, \text{Artifact}))$$

- ❖ Her işlem için girdi olarak bir kayıt tanımlanmaktadır.

$$\forall \text{Operation}(\text{hasInput}(\text{Operation}, \text{Artifact})) \leftrightarrow \\ \exists \text{Artifact}(\text{isInputOf}(\text{Artifact}, \text{Operation}))$$

- ❖ Her işlem çıktı olarak bir kayıt tanımlanmaktadır.

$$\forall \text{Operation}(\text{hasOutput}(\text{Operation}, \text{Artifact})) \leftrightarrow \\ \exists \text{Artifact}(\text{isOutputOf}(\text{Artifact}, \text{Operation}))$$

- ❖ Her işlem, tanımlı olan bir nesne kavramı için gerçekleştirilmektedir.

$$\forall \text{Object}(\text{hasOperation}(\text{Object}, \text{Operation})) \leftrightarrow \\ \exists \text{Operation}(\text{isPerformedFor}(\text{Operation}, \text{Object}))$$

- ❖ Her işlem bir kişi tarafından talep edilmektedir.

$$\forall \text{Operation}(\text{isRequestedBy}(\text{Operation}, \text{Person})) \leftrightarrow \\ \exists \text{Person}(\text{requests}(\text{Person}, \text{Operation}))$$

- ❖ Her işlem bir kişi tarafından gerçekleştirilmektedir.

$$\forall Operation(isPerformedBy(Operation, Person)) \leftrightarrow \exists Person(performs(Person, Operation))$$

- ❖ Bazı kayıtlar, diğer bir kaydın versiyonu olarak tanımlanmaktadır.

$$\exists Artifact(hasVersion(Artifact1, Artifact2)) \leftrightarrow \exists Artifact(isVersionOf(Artifact2, Artifact1))$$

- ❖ Bir aktör tarafından erişilen kayıtlar, erişim geçmişleri ile ilişkilendirilmektedir.

$$\exists Artifact(hasAccessHistory(Artifact, AccessHistory)) \leftrightarrow \forall AccessHistory(isAccessHistoryOf(AccessHistory, Artifact))$$

- ❖ Kayıtlar ve erişim geçmişleri için izin ve yasaklar tanımlanabilmektedir.

$$\begin{aligned} Subject \times Permission \times Artifact &\rightarrow hasPermission.Subject \\ Subject \times Prohibition \times Artifact &\rightarrow hasProhibition.Subject \\ Subject \times Permission \times AccessHistory &\rightarrow hasPermission.Subject \\ Subject \times Prohibition \times AccessHistory &\rightarrow hasProhibition.Subject \end{aligned}$$

- ❖ Bir aktör tarafından erişilen kayıtlar için erişim geçmişleri oluşturulmaktadır.

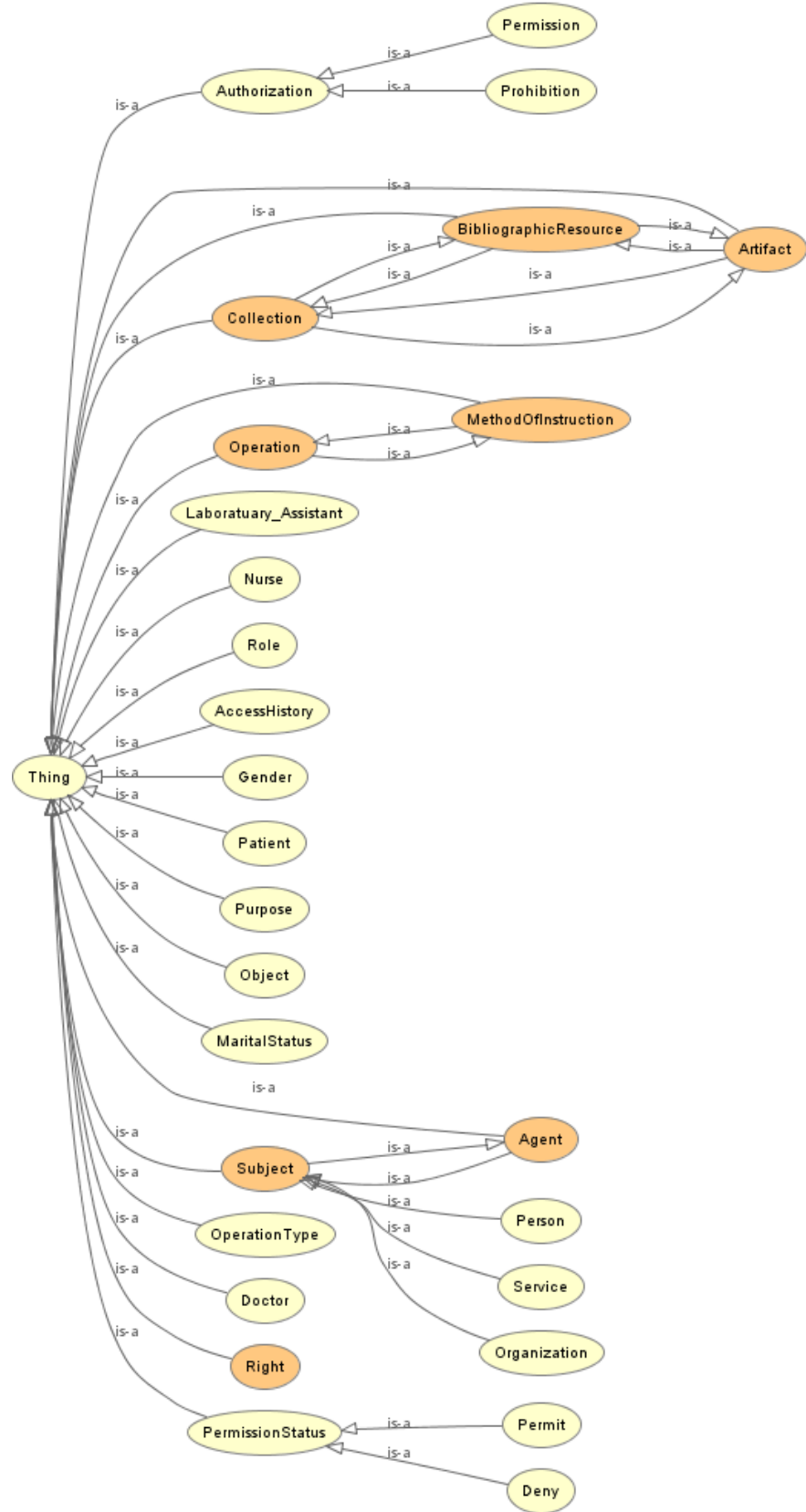
$$Subject \times Artifact \times AccessHistory \rightarrow hasAccessHistory.Artifact$$

Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan sınıfların, veri özelliklerinin, nesne özelliklerinin ve örneklerin sayısal oranları Çizelge 3.17'de yer almaktadır.

Çizelge 3.17 Gizlilik Farkında Anlamsal Köken Modeli Ontolojisi ölçütleri

Ölçüt Adı	Değer
Sınıf Sayısı	50
Nesne Özelliği Sayısı	56
Veri Özelliği Sayısı	54
Örnek Sayısı	169

Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan sınıfların, çizge yapısı Şekil 3.2'de yer almaktadır.



Şekil 3.2 Gizlilik Farkında Anlamsal Köken Modeli'nde yer alan sınıfların çizge yapısı

Geliştirilen köken modelinin, DC Profile katkıları aşağıda yer almaktadır:

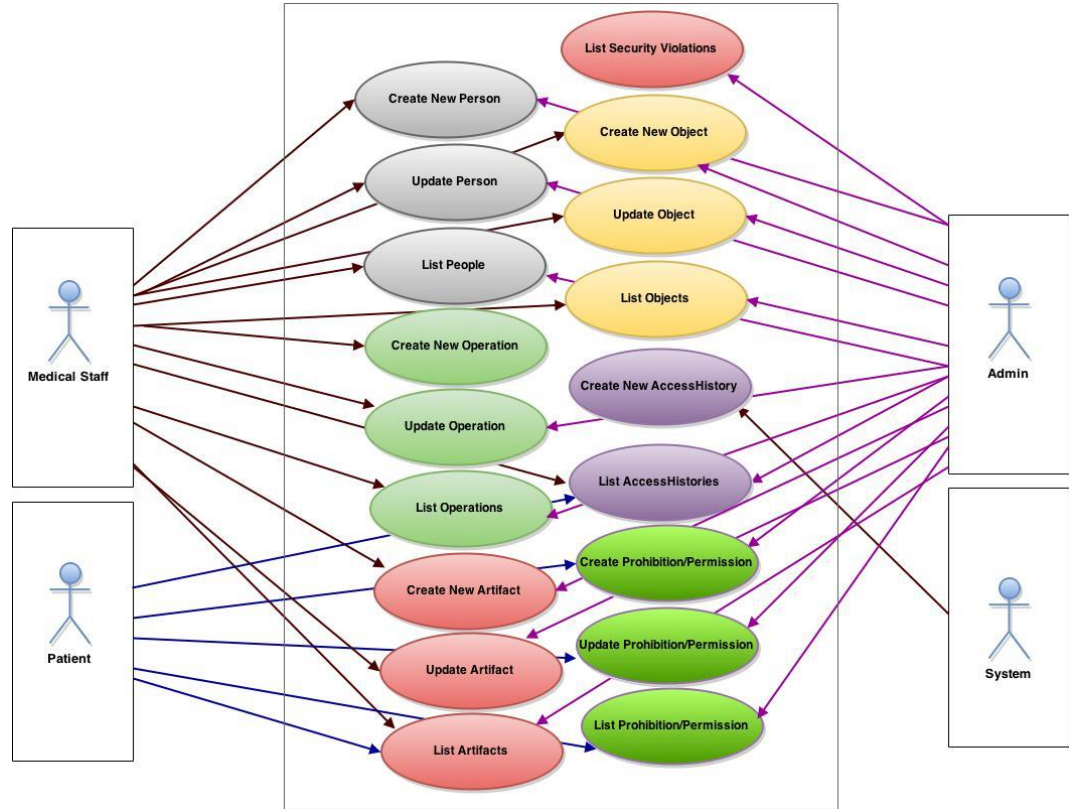
- İşlemler (Operation) ile kişilerin (Person) ilişkilendirilmesi,
- Kişiler (Person) için rollerin (Role) tanımlanması,
- Erişim amacı (Purpose), erişimi gerçekleştiren kişi ve bu kişinin hangi hakkı kullanarak eriştiği gibi bilgiler ile erişim geçmişlerinin (AccessHistory) tanımlanması,
- İzin ve kısıtlamaların; haklar (Right), erişim amacı(Purpose), kişiler (Person) veya roller (Role) kullanılarak oluşturulması,
- Kaynaklar (Artifact) ile kişilerin (Person) ilişkilendirilmesi,
- Kaynaklar ve erişim geçmişleri (Artifact ve AccessHistory) için izin ve kısıtlamaların tanımlanması,
- Erişim geçmişleri (AccessHistory) ve kaynaklar (Artifact) arasındaki ilişkiler kullanılarak kaynağın kökenine ulaşılabilmesi,
- Erişim geçmişleri ile erişim izin ve yasakları kullanılarak güvenlik ihlallerinin saptanabilmesi sağlanmaktadır.

Bir sonraki bölümde, altyapısında Gizlilik Farkında Anlamsal Köken Modeli'ni kullanarak, hasta kayıtlarında güvenlik ve gizliliğin desteklenmesini sağlayan Sağlık Köken Bilgi Sistemi'ne yer verilmektedir.

4. HASTA KAYITLARINDA GÜVENLİK VE GİZLİLİĞİ DESTEKLEYEN SAĞLIK KÖKEN BİLGİ SİSTEMİ

Tez kapsamında, güvenlik ve gizliliği desteklenmesi için geliştirilen köken modelinin sağlık personeli ve hastalar tarafından kullanılabilmesine yönelik bir Sağlık Köken Bilgi Sistemi (SKBS) geliştirilmektedir. Bu bölümde geliştirilen Sağlık Köken Bilgi Sistemi'nin yapısına ve sistemin gerçekleştirilmesine yer verilmektedir.

SKBS, altyapıda Anlamsal Web teknolojilerinin kullanıldığı, sağlık personeli, hastalar ve sistem yöneticisi tarafından kullanılan, bulaşıcı hastalıklar etki alanı üzerine kurulmuş bir sistemdir.



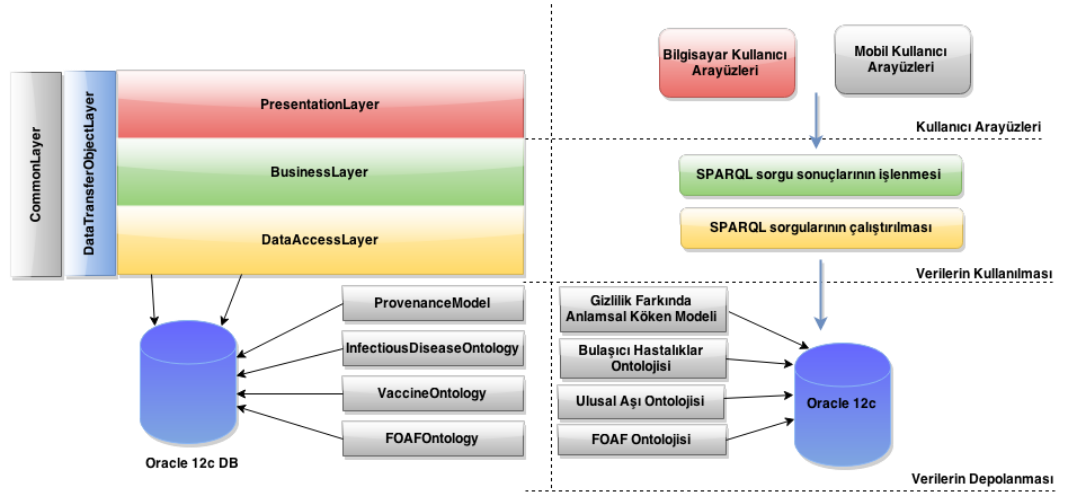
Şekil 4.1 Sağlık Köken Bilgi Sistemi için aktörler ve kullanım durumları

SKBS'nin; sağlık personeli, hasta, yönetici ve sistem olmak üzere 4 adet aktörü bulunmaktadır. Bu aktörler, Şekil 4.1'de gösterilen ve aşağıda belirtilen maddelerde yer alan işlemleri gerçekleştirebilmektedir:

- Sağlık personeli, yeni Person, Operation, Object, Artifact nesnesi yaratabilmekte, varolan Person, Operation, Object ve Artifact nesnelerini güncelleyip, listeleyebilmektedir. Ayrıca AccessHistory nesnelerini de listeleyebilmektedir. Şekil 4.1’de yer alan *Medical Staff* kavramı, hastane personelini temsil etmektedir. *Medical Staff* üzerinden çıkan oklar, hastane personeli tarafından gerçekleştirilebilen işlemleri göstermektedir.
- Hasta, Permission ve Prohibition nesneleri yaratabilmekte, varolan Permission ve Prohibition nesnelerini güncelleyebilmektedir. Artifact, AccessHistory, Prohibition ve Permission nesnelerini listeleyebilmektedir. Şekil 4.1’de yer alan *Patient* kavramı, hastayı temsil etmektedir. *Patient* üzerinden çıkan oklar, hasta tarafından gerçekleştirilebilen işlemleri göstermektedir.
- Sistem yöneticisi ise, yeni Person, Operation, Object, Artifact nesnesi yaratabilmekte, varolan Person, Operation, Object ve Artifact nesnelerini güncelleyip, listeleyebilmektedir. Ek olarak AccessHistory nesnelerini ve güvenlik ihlallerini listeleyebilmektedir. Şekil 4.1’de yer alan *Admin* kavramı, sistem yöneticisini temsil etmektedir. *Admin* üzerinden çıkan oklar, sistem yöneticisi tarafından gerçekleştirilebilen işlemleri göstermektedir.
- Sistem, AccessHistory nesnelerini yaratabilmektedir. Sistem dışında hiçbir aktör AccessHistory nesnesi yaratamamaktadır. Şekil 4.1’de yer alan *System* kavramı, sistemi temsil etmektedir. *System* üzerinden çıkan oklar, sistem tarafından gerçekleştirilebilen işlemleri göstermektedir.

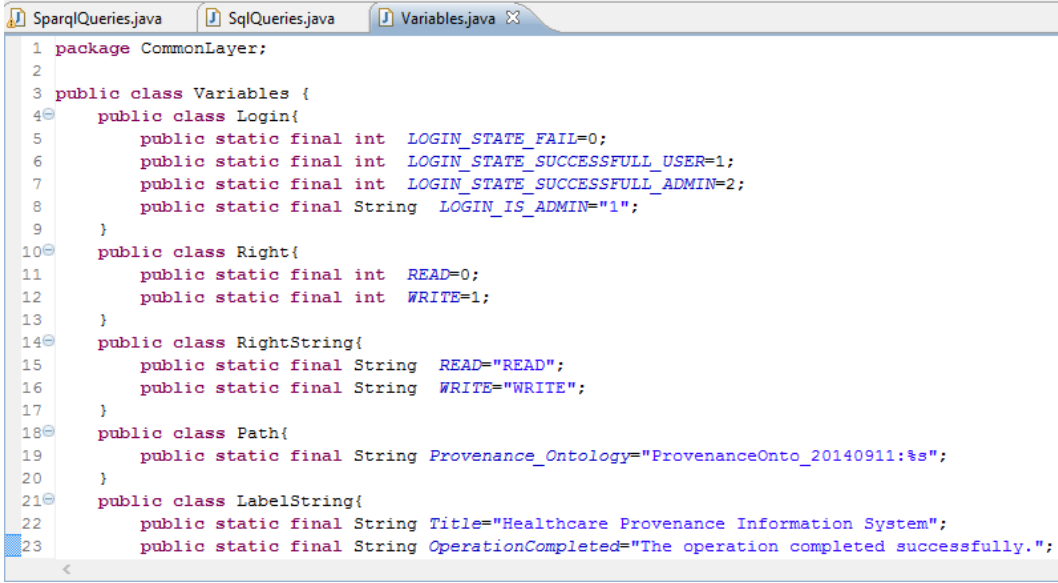
Sistemin gerçekleştirimi için, öncelikle Bulaşıcı Hastalıklar Ontolojisi, Ulusal Aşı Ontolojisi ve Gizlilik Farkında Anlamsal Köken Modeli (Köken Ontolojisi) Eclipse Java Editör’ü yardımıyla Anlamsal Web desteği olan Oracle 12c veritabanına aktarılmaktadır. Aktarımın ardından, bu ontolojilerde yer alan bilgilerin Oracle 12c veritabanı üzerinden kullanılabilmesi için katmanlı bir yapı tasarlanmaktadır. SKBS için Şekil 4.2’de görüldüğü gibi CommonLayer, DataAccessLayer, DataObjectLayer, DataTransferObjectLayer, BusinessLayer ve PresentationLayer olmak üzere altı katman

tanımlanmaktadır. CommonLayer, tüm katmanlar tarafından erişilebilen ortak sınıflardan oluşmaktadır. DataAccessLayer, veritabanı ile bağlantı kurulmasını sağlayan sınıfların yer aldığı katmandır. DataObjectLayer, Bölüm 3'te anlatılan köken modelinde yer alan nesneler için oluşturulan sınıflardan meydana gelmektedir. DataTransferObjectLayer ise, DataObjectLayer katmanında yer alan sınıfların yeterli olmaması nedeniyle, katmanlar arasında veri alışverişini sağlamak için oluşturulmuş sınıfların yer aldığı katmandır. BusinessLayer, DataAccessLayer ile PresentationLayer arasında yer alan katmandır. Bu katman, DataAccessLayer katmanında yer alan metotlar aracılığıyla çalıştırılan sorguların sonuçları kullanılarak yapılan işlemlerin ve kullanıcı ihtiyaçlarına yönelik geliştirilen algoritmalara ait kodlamaların yer aldığı sınıflardan oluşmaktadır. PresentationLayer katmanı ise, geliştirilen algoritmaların kullanıcılar tarafından çalıştırılabilmesi için tasarlanmış kullanıcı arayüzlerini içermektedir. Bu katmanların herbiri aşağıda ayrıntılı olarak açıklanmaktadır.



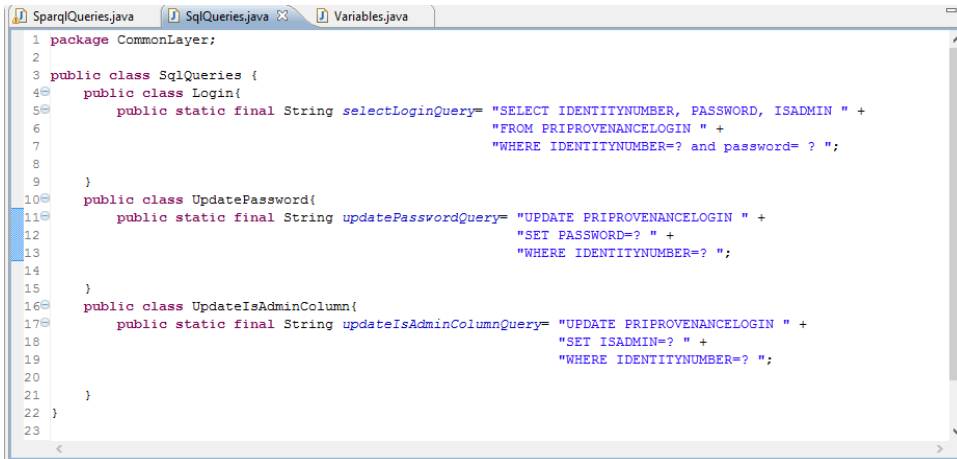
Şekil 4.2 Sağlık Köken Bilgi Sistemi Katmanlı Yapısı

CommonLayer, tüm katmanlardan erişilebilecek ortak değişkenlerin saklandığı sınıfları kapsamaktadır. Bu katman altında; Variables, SparqlQueries ve SqlQueries sınıfları yer almaktadır. Variables sınıfı, SKBS'nin farklı bölümlerinde kullanılan sabit değişkenlerin saklandığı sınıftır. Bu sınıf içinde değişkenlerin kullanıldığı bölümleri kategorize etmek için alt sınıflar yaratılmaktadır. Örneğin, sisteme giriş yapmak için programda kullanılan değişkenler, Şekil 4.3'te görüldüğü gibi Login sınıfının altında yer almaktadır.



Şekil 4.3 Variables sınıfının Eclipse üzerinde gösterimi

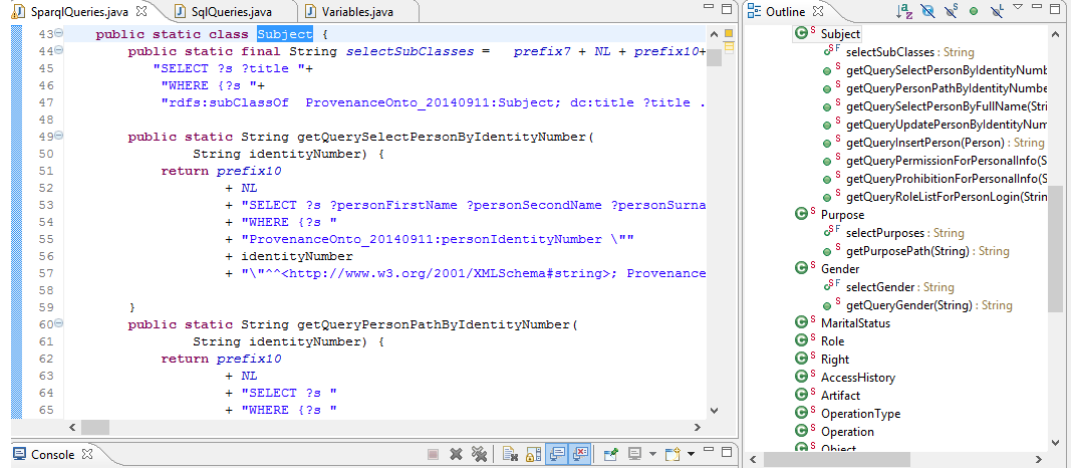
Sisteme giriş yapılabilmesi için her kullanıcının TC Kimlik numarası, şifresi ve yönetici olup olmadığı bilgisinin saklandığı birer sütun içeren PRIPOVENANCELOGIN adında bir tablo oluşturulmaktadır. Sisteme giriş için bu tablo üzerinden sorgulama yapılmaktadır. Şifre değişikliği ve sisteme yönetici atanma durumu PRIPOVENANCELOGIN tablosu üzerinde yapılan güncellemeler ile sağlanmaktadır. SqlQueries sınıfı, Şekil 4.4'te görüldüğü gibi PRIPOVENANCELOGIN tablosu üzerinde yapılacak SQL sorgularının tutulduğu sınıftır.



Şekil 4.4 SqlQueries sınıfının Eclipse üzerinde gösterimi

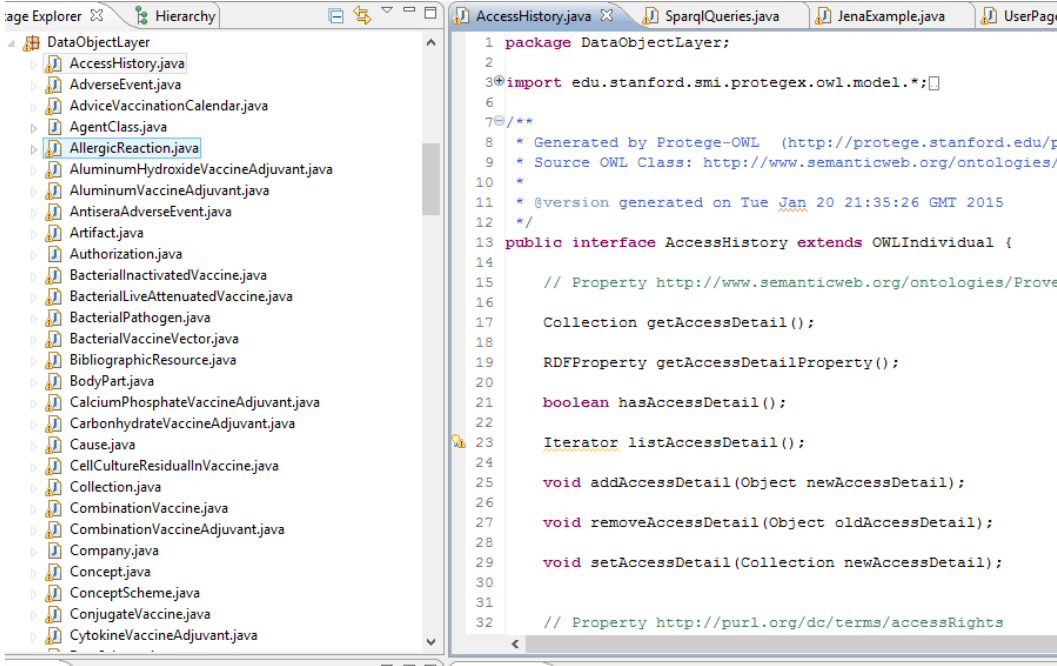
Bölüm 3'te detaylarına yer verilmiş olan, hasta kayıtlarında güvenlik ve gizliliğin desteklenmesi için geliştirilen köken modeli, Bölüm 4.1'de yer alan

bulaşıcı hastalıklar modeli ve Bölüm 4.2’de yer alan aşı ontolojisi üzerinde yer alan verileri sorgulamak için SPARQL sorgularına ihtiyaç duyulmaktadır. SPARQL sorguları ise, SparqlQueries sınıfında tutulmaktadır. SPARQL sorguları, modelde yer alan sınıflara göre kategorize edilmek üzere alt sınıflar altında tutulmaktadır. Şekil 4.5’te SparqlQueries sınıfının Eclipse üzerinde gösterimine yer verilmektedir.



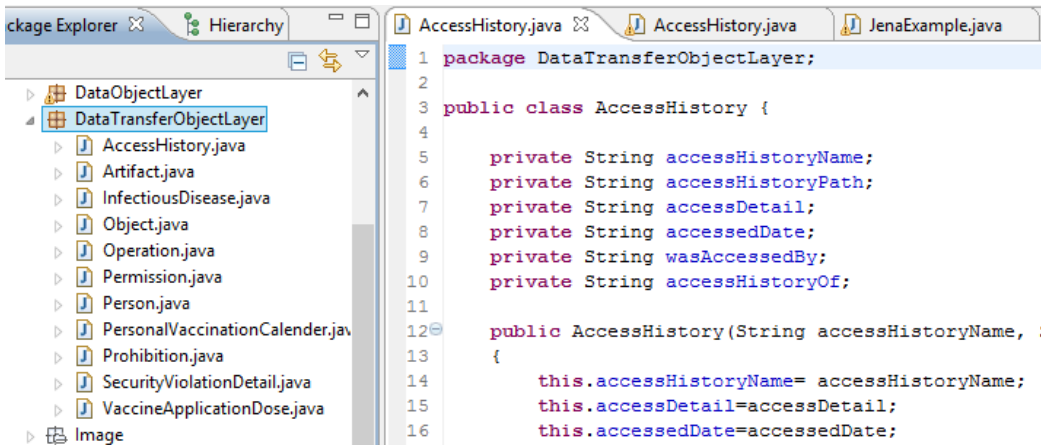
Şekil 4.5 SparqlQueries sınıfının Eclipse üzerinde gösterimi

DataObjectLayer, Bölüm 3’te anlatılan modelde yer alan nesneler için Protege Ontoloji Editörü yardımıyla otomatik olarak oluşturulmuş sınıfların yer aldığı katmandır. Bu sınıflarda, özellikler Collection ve RDFProperty tipinde oluşturulmakta, erişim metotları tanımlanmaktadır. Ancak bu sınıf içinde yer alan değişkenlerin tipleri ve bu değişkenlere erişim için oluşturulmuş metodlar, SKBS üzerinde veri alışverişi için ihtiyaç duyulan sınıf tanımını karşılamamaktadır. Bu nedenle projede DataObjectLayer katmanı altında yer alan sınıflar kullanılmamaktadır. DataObjectLayer katmanı altında yer alan sınıflar, Şekil 4.6’da gösterilmektedir.



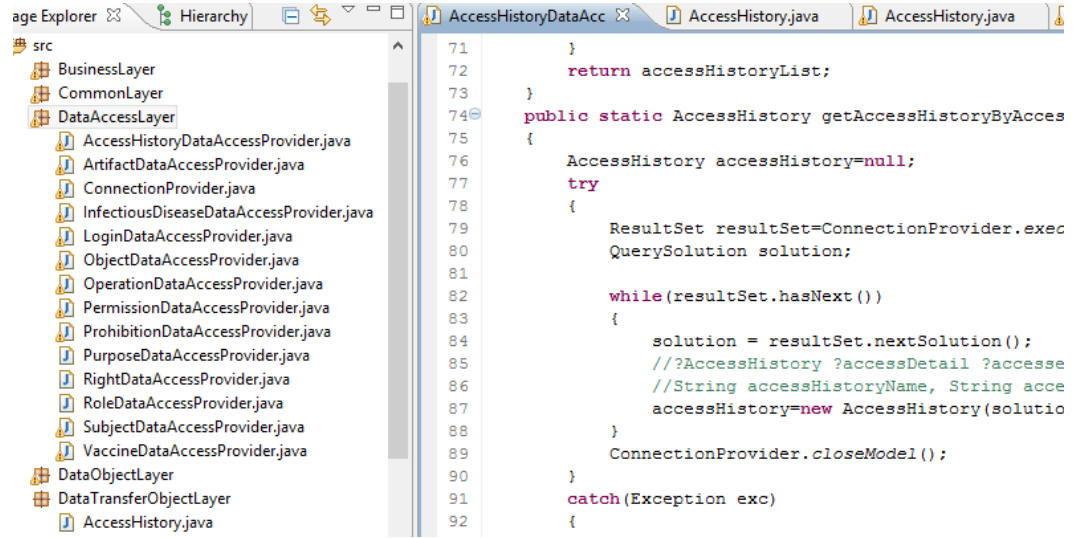
Şekil 4.6 DataObjectLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi

DataObjectLayer katmanında yer alan sınıfların yetersiz olması nedeniyle, katmanlar arasında veri alışverişini sağlamak amacıyla DataTransferObjectLayer katmanı yaratılmakta, Bölüm 3'te detaylandırılan modelde yer alan her nesneyi temsil eden sınıflar manuel olarak bu katman altında oluşturulmaktadır. Nesnelerde yer alan özellikler için yaratılan sınıflarda birer değişken tanımlanmakta ve bu değişkenlere erişim metotları yaratılmaktadır. Şekil 4.7'de DataTransferObjectLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimine yer verilmektedir.



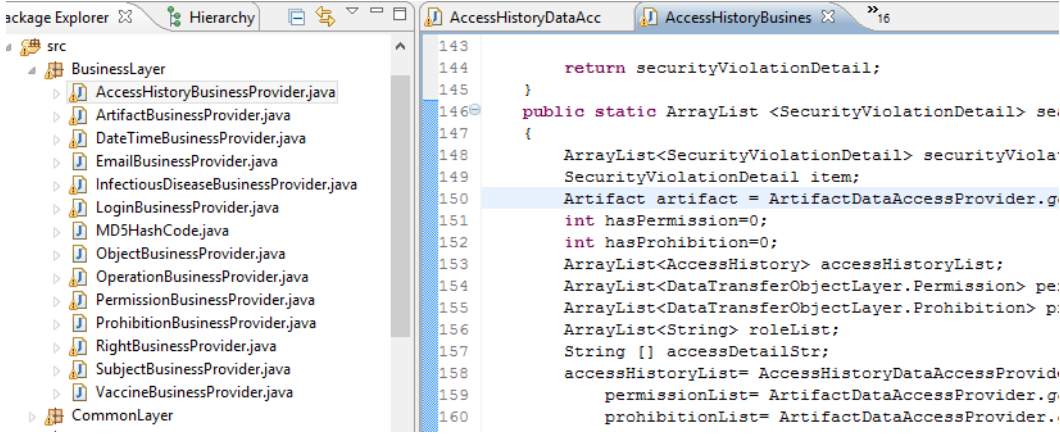
Şekil 4.7 DataTransferObjectLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi

DataAccessLayer katmanı, Bölüm 3, 4.1 ve 4.2’de detaylandırılan modellere erişim sağlamak amacıyla oluşturulmaktadır. Bu katman altında yer alan ConnectionProvider sınıfında, aşı ontolojisi, bulaşıcı hastalıklar ontolojisi ve güvenlik ve gizliliğe destekleyen köken ontolojisi için erişim bilgileri kullanılarak veri modellerine erişim sağlanmaktadır. Bu katmanda yer alan diğer sınıflar, SparqlQueries sınıfında yer alan sorguları ve ConnectionProvider sınıfında yer alan veri modellerine erişim için yazılmış metodları kullanarak modeller üzerinde gerekli işlemleri gerçekleştirmektedir. Şekil 4.8’de DataAccessLayer katmanında yer alan sınıflar gösterilmektedir.



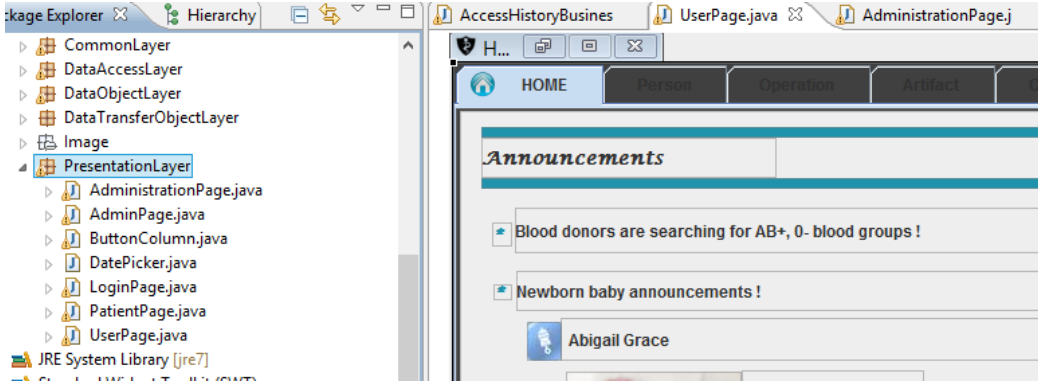
Şekil 4.8 DataAccessLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi

BusinessLayer katmanı, DataAccessLayer katmanının üzerinde yer almaktadır. Bu katmanda, DataAccessLayer katmanında yer alan metodlar aracılığıyla çalıştırılan sorgu sonuçları kullanılarak yapılan işlemlere ve güvenlik açıklarının tespit etmek gibi kullanıcı ihtiyaçlarına yönelik geliştirilen algoritmaların kodlamaları yer almaktadır. Şekil 4.9’da BusinessLayer katmanında yer alan sınıflar gösterilmektedir. Bölüm 4.3’te bu katmanda yer alan metodlar detaylandırılmaktadır.



Şekil 4.9 BusinessLayer katmanında yer alan sınıfların Eclipse üzerinde gösterimi

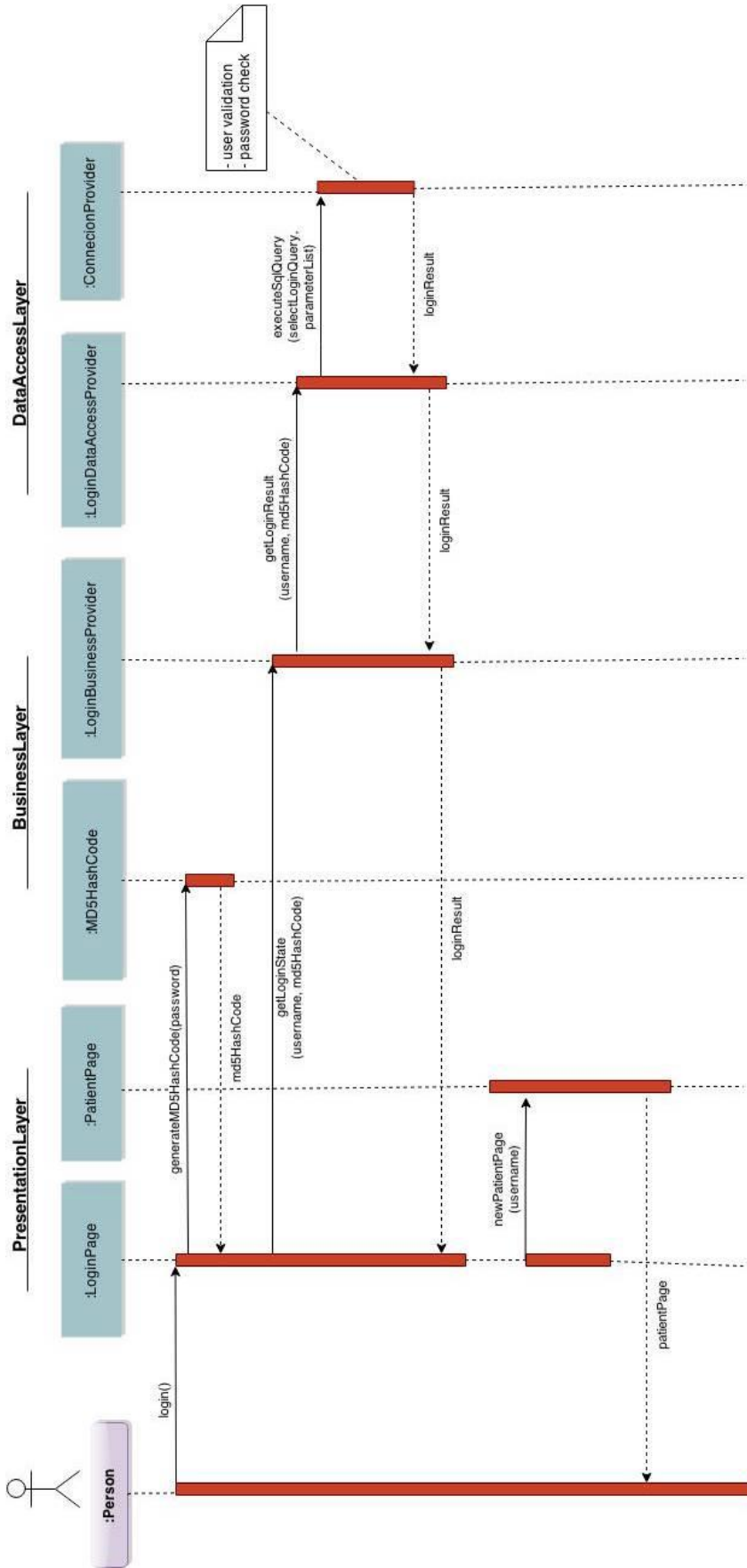
PresentationLayer katmanı, arka planda çalıştırılan algoritmaların, kullanıcılar ile buluşmasını sağlamaktadır. Kullanıcı arayüzlerinin yer aldığı bu katman sayesinde, kullanıcılar sisteme giriş yapmakta ve SKBS üzerinde işlemlerinin gerçekleştirebilmektedir. Bu katman altında sağlık personeli, hastalar ve sistem yöneticileri için tasarlanmış farklı ekranlar yer almaktadır. Bu ekranlar yardımıyla kullanıcıların, Bölüm 3, 4.1 ve 4.2’de detaylandırılmış olan veri modelleri üzerinde veri yaratma, sorgulama ve güncelleme yapılabilmesine olanak sağlanmaktadır. Şekil 4.10’da PresentationLayer katmanında yer alan sınıflar gösterilmektedir.



Şekil 4.10 PresentationLayer katmanında yer alan ekranların Eclipse üzerinde gösterimi

Katmanlar arasında veri akışını göstermek için Şekil 4.11’de SKBS’ne giriş yapan bir kullanıcı için yürütülen işlemlerin yer aldığı bir sıralı diagram verilmektedir. Kullanıcı, PresentationLayer katmanında yer alan LoginPage sayfası ile sisteme giriş yapma isteği göndermektedir. LoginPage, BusinessLayer katmanında yer alan MD5HashCode sınıfındaki,

generateMD5HashCode(password) metodunu çağırarak, dönen md5HashCode nesnesini kullanarak LoginBusinessProvider sınıfında yer alan getLoginState(username, md5HashCode) metodunu çağırılmaktadır. BusinessLayer katmanından sonra, DataAccessLayer katmanında yer alan LoginDataAccessProvider ve ConnectionProvider sınıfları kullanılarak kullanıcı adı ve şifreyi kontrol etmek için yazılmış olan SQL sorgusu veritabanı üzerinde çalıştırılmaktadır. Yapılan sorgu sonucu, önce BusinessLayer katmanına, ardından PresentationLayer katmanına ulaşmaktadır. Bu katmanda, sorgu sonucuna göre kullanıcının rolünün hasta olduğuna karar verilmekte ve PatientPage ekranı açılmaktadır. Böylece bir hastanın sisteme giriş işlemi tamamlanmış olmaktadır.



Şekil 4.11 SKBS’ne hasta giriş işleminin sıralı diagram olarak gösterimi

Bir sonraki bölümde, gizlilik farkında köken modelinin bulaşıcı hastalıklar etki alanına uygulanması için kullanılan Bulaşıcı Hastalıklar Ontolojisi detaylandırılmaktadır.

4.1 Bulaşıcı Hastalıklar Ontolojisi

Bulaşıcı hastalıklar; bakteri, virüs, parazitler gibi patojenik mikroorganizmaların neden olduğu hastalıklardır. Bulaşıcı hastalıklar, doğrudan ya da dolaylı olarak insandan insana, hayvandan insana bulaşabilmektedir. Genelde öksürük, doğrudan deri teması, vücut sıvısı teması, hasta olan kişinin kullandığı eşyaların kullanılması, hamile anneden bebeğine geçmesi, hastalığa neden olan mikrobu taşıyan yiyecek ve içeceklerin tüketilmesi gibi sebeplerle hastalık sağlıklı insanlara bulaşabilmektedir.

Her bulaşıcı hastalığın bulaştıktan sonra bir kuluçka süresi bulunmaktadır. Bu süre boyunca hastalığa ilişkin herhangi bir belirti görülmeyebilmektedir. Kuluçka dönemi ve sonrasında her bulaşıcı hastalık için farklı belirtiler gözlemlenmektedir. Bu belirtilerden başlıcaları; mide bulantısı, kusma, ani ateş, deri üzerinde döküntüler, terleme, halsizlik, baş ağrısı, kas ve eklem ağrılarıdır.

Hastalığa ilişkin belirtilerin gözlemlendiği hastalarda, hastalığın teşhisi için bazı teşhis yöntemlerinden yararlanılmaktadır. Bunlar; kan testi, ultrason ile izleme, EMAR, röntgen gibi yöntemler olabilmekte ve bu yöntemler ile elde edilen sonuçların biri ya da birkaçı incelenerek kesin teşhis konmaktadır. Teşhis konulduktan sonra bir tedavi yöntemi seçilmekte ve tedavi boyunca hasta izlenmektedir.

Hastalık teşhis edildikten sonra etrafında bulunan risk altındaki kişiler de hastalık bulaşmaması için uyarılmalıdır. Genelde hamile bayanlar, çok küçük bebekler ve yaşlılar risk altında olmaktadır. Risk altında olan kişilere hastalık bulaşmaması için çeşitli korunma yöntemleri önerilmektedir. Bunlardan başlıcası aşıdır. Pek çok bulaşıcı hastalık için aşılama yöntemi ile vücudun bağışıklık kazanması sağlanmaktadır. Ayrıca hasta ile temas edilmemesi, hastane personalinin maske ya da özel giysi giymesi, hastanın kullandığı eşyaların kullanılmaması gibi farklı korunma yöntemleri de bulunmaktadır.

Bulaşıcı Hastalıklar Ontolojisi oluşturulurken; hastalığın belirtileri, teşhisi, tedavisi, hastalıktan korunma yöntemleri ve risk altındaki kişiler gözönünde bulundurulmaktadır.

Bu ontoloji ile aşağıdaki soruların yanıtlarının elde edilmesi beklenmektedir:

- Bulaşıcı hastalığın ICD-10 kodu nedir?
- Bulaşıcı hastalığın belirtileri nelerdir?
- Bulaşıcı hastalığa sebep olan patojen nedir?
- Bulaşıcı hastalık belirtileri vücudun hangi bölümlerinde görülmektedir?
- Bulaşıcı hastalık nasıl teşhis edilmektedir?
- Bulaşıcı hastalıktan korunmak için neler yapılması gerekmektedir?
- Bulaşıcı hastalığın risk oluşturduğu kişiler kimlerdir?

Bulaşıcı Hastalıklar Ontolojisi'nin yukarıdaki sorulara yanıt verebilmesi için, InfectiousDisease, DiagnosticCriteria, Prevention-Method, RiskGroup, Symptom, ICD-10, Cause ve BodyPart sınıfları tanımlanmaktadır. Çizelge 4.1.1'de özellikleri gösterilen InfectiousDisease sınıfı ile bulaşıcı hastalığa ilişkin tanım, belirtiler, risk grubu, hastalığın yayılma şekli, hastalığın kuluçka süresi, hastalıktan korunma yöntemleri ve detayları tutulmaktadır.

Çizelge 4.1.1 InfectiousDisease sınıfının özellikleri

InfectiousDisease Sınıfı	
Özellik	İşlevi
InfectiousDiseaseName	Bulaşıcı hastalığın adı tutulur.
InfectiousDiseaseDescription	Bulaşıcı hastalığın tanımı ve tarihi geçmişine ilişkin bilgiler tutulur.
InfectiousDiseaseDiagnosis	Bulaşıcı hastalığın ne şekilde teşhis edilebileceğine ilişkin bilgi tutulur.

InfectiousDiseaseIncubationPeriod	Bulaşıcı hastalığın kuluçka süresi tutulur.
InfectiousDiseasePrevention	Bulaşıcı hastalıktan nasıl korunabileceğine ilişkin bilgiler tutulur.
InfectiousDiseasePublicHealthResponse	Bulaşıcı hastalığın teşhis edilmesinden sonra bu durumun kayıt altına alınabilmesi için nasıl bir bildirim yapılması gerektiği bilgisi tutulur.
InfectiousDiseaseSpread	Bulaşıcı hastalığın nasıl yayıldığına ilişkin bilgi tutulur.
InfectiousDiseaseTreatment	Bulaşıcı hastalığın ne şekilde tedavi edilmesinin gerektiğine ilişkin detaylar tutulur.
causedBy	Bulaşıcı hastalığın, Cause sınıfından oluşturulmuş patojen ile ilişkilendirir.
hasDiagnosticCriteria	Bulaşıcı hastalığın, DiagnosticCriteria sınıfından yaratılan teşhis yöntemi ile ilişkilendirir.
hasPreventionMethod	Bulaşıcı hastalığın, PreventionMethod sınıfından yaratılan hastalıktan korunma yöntemi ile ilişkilendirir.
hasRiskGroup	Bulaşıcı hastalığı, RiskGroup sınıfından yaratılan risk altında olan kişiler ile ilişkilendirir.
hasSymptom	Bulaşıcı hastalığı, Symptom sınıfından yaratılan hastalık belirtileri ile ilişkilendirir.
hasICD-10	Bulaşıcı hastalığın ICD-10 kodu ile ilişkilendirilmesini sağlar.

Symptom sınıfı hastalığa ait belirtilerin tutulması için yaratılmaktadır. Symptom sınıfından oluşturulan bir öge farklı bulaşıcı hastalıklarda kullanılabilmektedir. Bu sınıfın özellikleri Çizelge 4.1.2’de gösterilmektedir.

Çizelge 4.1.2 Symptom sınıfının özellikleri

Symptom Sınıfı	
Özellik	İşlevi
SymptomName	Bulaşıcı hastalık belirtisinin adını tutar.
isIn	Bulaşıcı hastalık belirtisinin vücudun hangi bölgesinde görüldüğünü belirtmek için BodyPart ile ilişkilendirilmesini sağlar.

Hastalık sonucu görülen belirtilerin vücudun hangi bölgesinde olduğunu belirtmek ve vücut bölgelerinin tanımlanması için bir başka sınıfa ihtiyaç duyulmaktadır. Bu gereksinime yönelik olarak BodyPart sınıfı yaratılmakta, Symptom sınıfı ile ilişkilendirilmektedir. Çizelge 4.1.3'te BodyPart sınıfının özelliklerine yer verilmektedir.

Çizelge 4.1.3 BodyPart, RiskGroup, PreventionMethod ve DiagnosticCriteria sınıflarının özellikleri

Sınıf	Özellik	İşlevi
BodyPart	BodyPartName	Bulaşıcı hastalık belirtilerinin görüldüğü vücut bölgesinin adını tutar.
RiskGroup	RiskGroupName	Bulaşıcı hastalık için risk altında olan grupların adını tutar.
PreventionMethod	PreventionMethodName	Bulaşıcı hastalıktan korunma metodunun adını tutar.
DiagnosticCriteria	DiagnosticCriteriaName	Bulaşıcı hastalığın teşhisi için hangi kriterin baz alınacağını bilgisini tutar.

Hastalığın daha fazla kişiye yayılmasını engellemek için bilinmesi gereken hastalıktan korunma yöntemleri için PreventionMethod sınıfı kullanılmaktadır. Bunlara ek olarak, risk altında olan kişilerin belirtilebilmesi için RiskGroup sınıfı yaratılmaktadır. Bu sınıfların özellikleri, Çizelge 4.1.3'te detaylandırılmaktadır.

Cause sınıfı ise bulaşıcı hastalığa neden olan patojenlerin tutulmasını sağlamaktadır. Cause sınıfı kullanılarak yaratılan bakteri, virüs ve parazit gibi mikroorganizmalar ile hastalık ilişkilendirilmektedir. Hastalığın teşhisi için kan testi gibi kullanılması gereken metotların tutulması için ise DiagnosticCriteria sınıfı yaratılmaktadır. Çizelge 4.1.4'te Cause sınıfının özelliklerine yer verilmektedir.

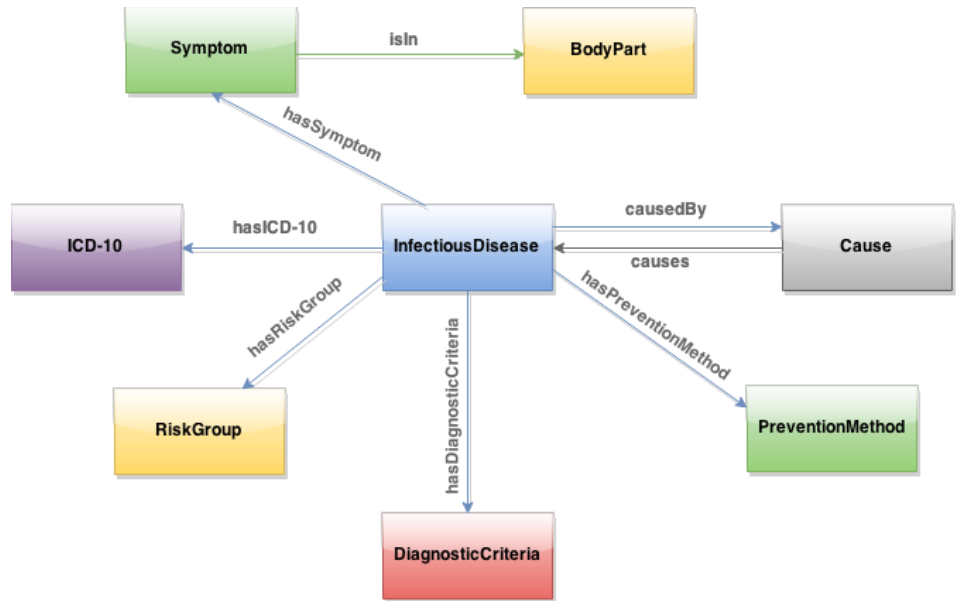
Çizelge 4.1.4 Cause sınıfının özellikleri

Cause Sınıfı	
Özellik	İşlevi
CauseName	Bulaşıcı hastalığa neden olan nesnenin adı tutulur.
causes	Patojenin hangi bulaşıcı hastalığa neden olduğunu belirtmek için InfectiousDisease ile ilişkilendirilmesini sağlar.

Bulaşıcı hastalıkların uluslararası standartlar uygun olarak sınıflandırılmasını sağlamak için ICD-10 sınıfı yaratılmaktadır. ICD-10 sınıfı, ICD-10 kodu ile bu kodun tanımını tutmak için kullanılmaktadır. Çizelge 4.1.5'te ICD-10 sınıfının özelliklerine yer verilmektedir.

Çizelge 4.1.5 ICD-10 sınıfının özellikleri

ICD-10 Sınıfı	
Özellik	İşlevi
ICD-10Code	ICD-10 kodunu tutar.
ICD-10Description	ICD-10 koduna ilişkin açıklamayı tutar.



Şekil 4.1.1 Bulaşıcı Hastalık Ontolojisi'nde yer alan atomik kavram ve roller

Bulaşıcı Hastalıklar Ontolojisi'nde yer alan atomik kavram ve roller Şekil 4.1.1'de gösterilmektedir. Bu kavram ve rollerin detayları aşağıdaki maddelerde yer almaktadır:

- Atomik kavramlar; InfectiousDisease, Cause, Symptom, BodyPart, RiskGroup, PreventionMethod ve ICD-10 kavramlarıdır.
- Atomik rol olan causedBy, InfectiousDisease nesnesi ile Cause nesnesi arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, causes atomik rolü ile ters ilişkilidir, dolayısıyla Cause nesnesi ile InfectiousDisease nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan hasDiagnosticCriteria; InfectiousDisease ile DiagnosticCriteria arasında, hasPreventionMethod; InfectiousDisease ile PreventionMethod arasında, hasRiskGroup; InfectiousDisease ile RiskGroup arasında, hasSymptom; InfectiousDisease ile Symptom arasında ve hasICD-10 ise; InfectiousDisease ile ICD-10 arasında bir bağlantı kurulmasını sağlamaktadır.
- Atomik rol olan isIn, Symptom ile BodyPart arasında bir ilişki kurulmasını sağlamaktadır.

Bulaşıcı Hastalıklar Ontolojisi için tanımlanan kuralların, *ALCQ DL* ifadeleri ile gösterimine aşağıda yer verilmektedir:

- ❖ Her bulaşıcı hastalık; bakteri, virüs ve parazit gibi bir patojen nedeniyle oluşmaktadır.

$\forall \text{InfectiousDisease}(\text{causedBy}(\text{InfectiousDisease}, \text{Cause})) \leftrightarrow$

$\exists \text{Cause}(\text{causes}(\text{Cause}, \text{InfectiousDisease}))$

- ❖ Her bulaşıcı hastalığın; kan testi ve idrar testi gibi en az bir teşhis kriteri bulunmaktadır.

$\forall \text{InfectiousDisease}(\text{hasDiagnosticCriteria}(\text{InfectiousDisease}, \text{DiagnosticCriteria})), \text{DiagnosticCriteria} \subseteq \text{hasDiagnosticCriteria.InfectiousDisease}$

- ❖ Her bulaşıcı hastalığın; aşılama gibi en az bir korunma yöntemi bulunmaktadır.

$\forall \text{InfectiousDisease}(\text{hasPreventionMethod}(\text{InfectiousDisease}, \text{PreventionMethod})), \text{PreventionMethod} \subseteq \text{hasPreventionMethod.InfectiousDisease}$

- ❖ Her bulaşıcı hastalığın; yeni doğan bebekler, hamileler ve yaşlılar gibi üzerinde risk oluşturduğu en az bir risk grubu bulunmaktadır.

$\forall \text{InfectiousDisease}(\text{hasRiskGroup}(\text{InfectiousDisease}, \text{RiskGroup})), \text{RiskGroup} \subseteq \text{hasRiskGroup.InfectiousDisease}$

- ❖ Her bulaşıcı hastalığın; yüksek ateş, bulantı ve deride döküntüler gibi en az bir belirtisi bulunmaktadır.

$\forall \text{InfectiousDisease}(\text{hasSymptom}(\text{InfectiousDisease}, \text{Symptom})), \text{Symptom} \subseteq \text{hasSymptom.InfectiousDisease}$

- ❖ Her bulaşıcı hastalığın, bir ICD kodu bulunmaktadır.

$\forall \text{InfectiousDisease}(\text{hasICD-10}(\text{InfectiousDisease}, \text{ICD-10})), \text{ICD-10} \subseteq \text{hasICD-10.InfectiousDisease}$

- ❖ Bazı bulaşıcı hastalık belirtileri, vücudun belli bir bölümünde görülmektedir.

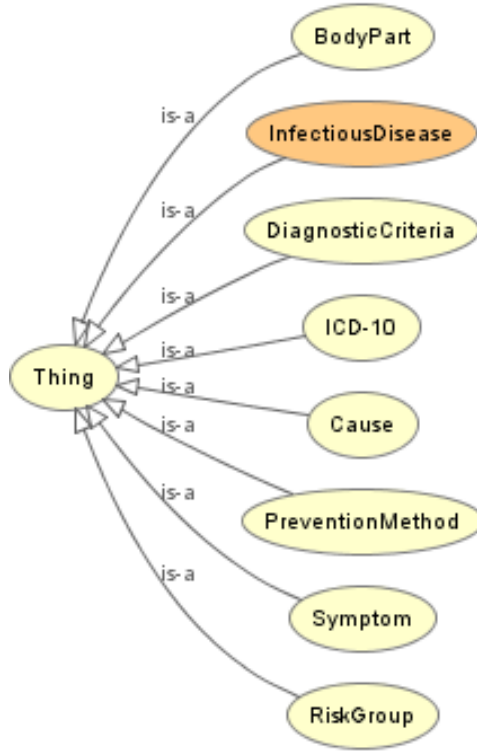
$\exists \text{Symptom}(\text{isIn}(\text{Symptom}, \text{BodyPart})), \text{Symptom} \subseteq \text{isIn.BodyPart}$

BHO’da yer alan sınıfların, veri özelliklerinin, nesne özelliklerinin ve örneklerin sayısal oranları Çizelge 4.1.6’da yer almaktadır.

Çizelge 4.1.6 BHO için ontoloji ölçütleri

Ölçüt Adı	Değer
Sınıf Sayısı	9
Nesne Özelliği Sayısı	8
Veri Özelliği Sayısı	16
Örnek Sayısı	64
Uluslararası Standartlar	ICD-10

BHO’da yer alan sınıfların, çizge yapısı Şekil 4.1.2’de gösterilmektedir.



Şekil 4.1.2 Bulaşıcı Hastalık Ontolojisi’nde yer alan sınıfların çizge yapısı

Bir sonraki bölümde, BHO üzerinde hastalık gösterimine ilişkin bir örneğe yer verilmektedir.

4.1.1 Bulaşıcı Hastalıklar Ontolojisi üzerinde hastalık gösterim örneği

Bu bölümde Sıtma (Malaria) hastalığının oluşturulan Bulaşıcı Hastalıklar Ontolojisi üzerinde gösterimi örneklendirilmektedir.

Sıtma hastalığı ile ilgili bilgiler ve *ALCQ DL* ifadeleri aşağıda yer almaktadır:

- ❖ Sıtma, tropikal ve yarı tropikal bölgelerdeki sivrisineklerin ısırmasıyla insan ve hayvanlara geçen, ciddi ve bazı durumlarda hayatı tehlikeye atan bir bulaşıcı hastalıktır. Beş farklı parazit bu hastalığa neden olmaktadır: Plasmodium ovale, P. malariae, P. knowlesi, P. vivax ve P. falciparum (WHO, 2015).

causedBy(Malaria) $\equiv$ Parasite

- ❖ Hastalığın başlıca belirtileri; ani ateş, deride döküntüler, baş ağrısı, terleme, mide bulantısı, kusma, kas ve eklem ağrılarıdır (WHO, 2015).

hasSymptom(Malaria) $\equiv$ SuddenFever $\cap$ Vomiting $\cap$ Chills $\cap$ Headache $\cap$ Sweating $\cap$ Nausea $\cap$ Pain

isIn(Pain) $\equiv$ Joints $\cap$ Muscles

- ❖ Hastalığın kuluçka süresi 9-14 gün kadar olmaktadır (WHO, 2015).

InfectiousDiseaseIncubationPeriod(Malaria) $\equiv$ 9-14 days

- ❖ Sıtma, dişi anafel sivrisineğinin ısırması ile yayılmaktadır. Hastalığa neden olan parazitler, sivrisineğin bağırsak ve tükürük bezlerinde yaşamaktadır. Kişi sivrisinek tarafından ısırıldığında bu parazitler, kana geçmektedir. Ardından karaciğer ve kan hücrelerinin enfekte olmasına neden olmaktadır. Hastalık kan nakli, enjeksiyon gibi gereçlerin ortak kullanılması yoluyla ya da anneden fetüse geçmektedir. Bu hastalık, kan testi sonuçları ve seyahat geçmişi dikkate alınarak teşhis edilmektedir (WHO, 2015).

$$hasDiagnosticCriteria(Malaria) \equiv BloodTest \cap TravelHistory$$

- ❖ Sıtma görülen bölgelerde yaşayan ve seyahat edenler, hamile bayanlar, küçük çocuklar, denizaşırı akraba ve arkadaş ziyareti yapanlar risk altında olmaktadır (WHO, 2015).

$$hasRiskGroup(Malaria) \equiv PregnantWomen \cup YoungChildren \cup PeopleVisitingFriendsAndRelativesOverseas$$

- ❖ Hastalıktan korunmak için sivrisinek ısırıklarından kaçınmak ya da parazitleri öldüren ilaçlar kullanılabilir (WHO, 2015).

$$hasPreventionMethod(Malaria) \equiv Drug$$

- ❖ Bu hastalığın ICD-10 kodu, B52 olarak belirtilmektedir (WHO, 2015).

$$hasICD-10(Malaria) \equiv B52$$

Bahsedilen bilgilerin Protege ontoloji editörü üzerinde gösterimine Şekil 4.1.1.1’de yer verilmektedir.

Property assertions: Malaria	
hasSymptom	Headache
hasRiskGroup	PeopleVisitingFriendsAndRelativesOverseas
hasSymptom	Nausea
hasSymptom	Sweating
hasSymptom	Chills
hasPreventionMethod	Drug
hasSymptom	Pain
hasICD-10	B52
causedBy	Parasite
hasDiagnosticCriteria	TravelHistory
hasSymptom	SuddenFever
hasRiskGroup	YoungChildren
hasDiagnosticCriteria	BloodTest
hasRiskGroup	PregnantWomen
hasSymptom	Vomiting
Data property assertions +	
InfectiousDiseaseDiagnosis	"Malaria is diagnosed by special blood tests that detect malaria parasites. It is important that your doctor knows about your travel history and symptoms so the right blood test can be ordered."
InfectiousDiseasePublicHealthResponse	"Laboratories are required to notify cases of malaria on diagnosis to the local public health unit."
InfectiousDiseaseIncubationPeriod	"9-14 days"
InfectiousDiseaseName	"Malaria"
InfectiousDiseasePrevention	"Overseas travellers can prevent malaria by avoiding mosquito bites and taking antimalarial drugs that kill the parasite."
InfectiousDiseaseDescription	"Malaria is an infection of the liver and red blood cells caused by microscopic parasites. There are five types of parasites that cause malaria: Plasmodium ovale, P. malariae, P. knowlesi, P. vivax and P. falciparum. Malaria parasites are spread through the bite of certain mosquitoes. Mainland Australia is free of malaria, but malaria is occasionally present in the Torres Strait Islands. Australians can contract malaria while travelling in tropical and subtropical areas of Asia, Africa, Central and South America, the Pacific Islands and parts of the Middle East. Currently, approximately 500 cases of malaria are diagnosed in Australia each year - almost all are in people who have travelled to malaria-affected countries and didn't take anti-malarial medications."

Şekil 4.1.1.1 Bulaşıcı Hastalık Ontolojisi’nde yer alan “Sıtma” tanımı

4.1.2 Gizlilik Farkında Anlamsal Köken Modeli ve BHO

Gizlilik Farkında Anlamsal Köken Modeli, BHO kullanılarak bulaşıcı hastalıklar etki alanı üzerine uygulandığında, aşağıdaki soruların yanıtlanmasını sağlamaktadır:

- Hastanın bulaşıcı hastalık kayıtları nelerdir?
- Hastaya ait bulaşıcı hastalık kayıtlarında kimler değişiklik yapmıştır?
- Bulaşıcı hastalık kayıtları üzerinde yapılan işlemler nelerdir?
- Bulaşıcı hastalığın seyri nasıl ilerlemektedir?
- Bulaşıcı hastalık kayıtları için hasta tarafından tanımlanan izin ve yasaklar nelerdir?
- Hastanın bulaşıcı hastalık kayıtlarına kimler erişim gerçekleştirmiştir?
- Bulaşıcı hastalık kayıtları erişimi sırasında herhangi bir güvenlik ihlali yapılmış mıdır?

Sonraki iki bölümde, köken modelinin bulaşıcı hastalıklar etki alanı üzerine uygulanmasına ilişkin örneklere yer verilmektedir.

4.1.2.1 Köken Modeli ile hasta kayıtlarının erişim geçmişinin kontrol edilmesi örneği

Bu bölümde, model üzerinde Hepatit C hastalığının teşhis süreci gözönünde bulundurularak bir örnek gösterim gerçekleştirilmektedir. *Person*, hastane personeli ve hastaları göstermektedir. *Operation* kavramı, klinik testler ve tedavi sürecini temsil etmektedir. *Artifact* ise, hastanın kişisel bilgileri, klinik test sonuçları, teşhis ve hastaya uygulanan tedavi yöntemlerini ifade etmek için kullanılmaktadır. Modelde yer alan tüm kavramlar ve bulaşıcı hastalıklar etki alanı ile eşleştirilmesi Bölüm 3'te gösterilmektedir.

Bu örnek gösterimde, doktor hastanın şikayetlerini dinledikten sonra hastanın Hepatit C taşıdığını düşünmektedir. Hepatit C teşhisi için; *HCV antibody enzyme immunoassay (ELISA)*, *recombinant immunoblot assay*, ve *quantitative HCV RNA polymerase chain reaction (PCR)* gibi testler uygulanmaktadır. Doktor, hastalığın teşhisi için *ELISA* testi talep etmekte, hasta ise sağlık kayıtlarını yetkisiz erişimlerden korumak istemektedir. Öncelikle hemşire Lucy Law, hasta Diana Jodie Black'in kişisel bilgilerini ve doktor John Smith'in *ELISA* testi talebini sisteme girmektedir. Sisteme kaydedilen bu veriler için hasta tarafından erişim yetkileri belirlenmektedir. Ardından *ELISA* testi yapılmakta ve test sonucu çıktı olarak verilmektedir. Test sonuçları için yine hasta tarafından erişim yetkilendirmesi yapılmaktadır. Laboratuvar asistanı Mark Terry, eksikleri tamamlamak için test sonucu güncellemek istemektedir. Test sonucunu güncellemek için yeni bir işlem talebi girilmektedir. Test sonucu, bu işleme girdi olarak verilmektedir. İşlem laboratuvar asistanı tarafından gerçekleştirildikten sonra yeni bir çıktı oluşmaktadır. Bu çıktı, bir önceki test sonucunun yeni versiyonudur. Yine bir önceki test sonucunda olduğu gibi hasta tarafından erişim yetkilendirmesi yapılmaktadır. Girdi ve çıktı olarak yaratılan her veri üzerinde yapılan her değişiklik ve erişim için sisteme bir erişim geçmişi kaydı eklenmektedir. Ayrıca, her erişim geçmişi kaydı için hasta tarafından erişim yetkilendirmesi yapılmaktadır. Şekil 4.1.2.1.1'de örneğin şematik gösterimi verilmiştir.

Şekil 4.1.2.1.1 Hepatit C teşhisi ve hasta kayıtları üzerinde erişim kontrolü

Bir önceki bölümde belirtilen kurallara göre örneğin *ALCQ DL* dilinde gösterimine aşağıda yer verilmektedir:

- ❖ John Smith, doktor rolünde; Diana Jodie Black, hasta rolünde; Lucy Law, hemşire rolünde; Mark Terry, laboratuvar asistanı rolünde tanımlanmaktadır.

hasRole(John Smith) $\equiv$ Doctor

hasRole(Diana Jodie Black) $\equiv$ Patient

hasRole(Lucy Law) $\equiv$ Nurse

hasRole(Mark Terry) $\equiv$ LaboratoryAssistant

- ❖ Hepatit C hastalığı için, ELISA test gerçekleştirilmektedir.

isPerformedFor(ELISA, HepatitisC) $\equiv$ hasOperation(HepatitisC, ELISA)

- ❖ John, hastaya ELISA test yapılmasını talep etmektedir.

requests(John, ELISA) $\equiv$ isRequestedBy(ELISA, John)

- ❖ Lucy, hastanın kişisel bilgilerinin de yer aldığı ilk kayıdı yaratmaktadır.

creator(Artifact1) = Lucy

- ❖ Yaratılan kayıt, hastaya ait olduğundan sahiplik ilişkisi oluşturulmaktadır.

Artifact1 = Diana's PersonalInformation $\cup$ Diana's ELISATestRequest
isOwnerOf(Diana, Artifact1) $\equiv$ hasOwnerOf(Artifact1, Diana)

- ❖ Kayıt için yetki tanımlanmaktadır.

hasPermission(Artifact1) = (Write, Nurse, Clinical Test,
2014-10-01, 2014-10-31)

- ❖ Kayıt, ELISA test işlemine girdi olarak verilmektedir.

$isInputOf(Artifact1, ELISA) \equiv hasInput(ELISA, Artifact1)$

- ❖ ELISA test sonucunda yeni bir kayıt oluşmaktadır. Bu kayıt için de Diana için sahiplik ilişkisi yaratılmakta, ardından izinler tanımlanmaktadır.

$Artifact2 = ELISA_TestResult$

$isOutputOf(Artifact2, ELISA) \equiv hasOutput(ELISA, Artifact2)$

$isOwnerOf(Diana Jodie Black, Artifact2) \equiv hasOwner(Artifact2, Diana Jodie Black)$

$hasPermission2(Artifact2) = (Write, Doctor, Treatment, 2014-10-01, 2014-10-31)$

$hasPermission3(Artifact2) = (Read, Doctor, Treatment, 2014-10-01, 2014-10-31)$

$hasPermission6(Artifact2) = (Write, LaboratoryAssistant, Clinical Test, 2014-10-01, 2014-10-31)$

- ❖ John, test sonucunu okuduğunda, test sonucu için bir erişim geçmiş kaydı oluşturulmaktadır. Erişim geçmişi için, erişim izin ve yasakları tanımlanmaktadır.

$hasAccessHistory1(Artifact2) = (Clinical Test, John Smith, Read, 2014-10-01 15:30)$

$wasAccessedBy(AccessHistory1) \equiv John Smith$

$isAccessHistoryOf(AccessHistory1) \equiv Artifact2$

$hasPermission4(AccessHistory1) = (Diana Jodie Black, Read)$

$hasProhibition1(AccessHistory1) = (Read, Write, Nurse, Doctor, LaboratoryAssistant, Clinical Test, Treatment, Precaution)$

- ❖ Mark, test sonucunu güncelleme işlemi talep etmektedir.

$requests(Mark Terry, Update Artifact2) \equiv isRequestedBy(Update Artifact2, Mark Terry)$

- ❖ Test sonucu güncelleme işlemi için girdi olmaktadır.

$$isInputOf(Artifact2, Update Artifact2) \equiv hasInput(Update Artifact2, Artifact2)$$

- ❖ Mark güncelleme işlemini gerçekleştirmektedir. Bu işlemin ardından bir erişim geçmişi kaydı oluşmaktadır. Erişim geçmişi için, erişim izin ve yasakları tanımlanmaktadır.

$$performs(Mark Terry, Update Artifact2) \equiv isPerformedBy(Update Artifact2, Mark Terry)$$

$$hasAccessHistory2(Artifact2) = (Clinical Test, Mark Terry, Write, 2014-10-01 16:00)$$

$$wasAccessedBy(AccessHistory2) \equiv Mark Terry$$

$$isAccessHistoryOf(AccessHistory2) \equiv Artifact2$$

$$hasPermission4(AccessHistory2) = (Diana Jodie Black, Read)$$

$$hasProhibition1(AccessHistory2) = (Read, Write, Nurse, Doctor, LaboratoryAssistant, Clinical Test, Treatment, Precaution)$$

- ❖ Güncelleme işleminin ardından çıktı olarak yeni bir kayıt oluşmaktadır. Yeni kayıt için izinler tanımlanmaktadır.

$$isOutputOf(Artifact3, Update Artifact2) \equiv hasOutput(Update Artifact2, Artifact3)$$

$$hasPermission2(Artifact3) = (Write, Doctor, Treatment, 2014-10-01, 2014-10-31)$$

$$hasPermission3(Artifact3) = (Read, Doctor, Treatment, 2014-10-01, 2014-10-31)$$

- ❖ Yeni kayıt için, eski kayıt ile versiyon ilişkisi yaratılmaktadır.

$$priorVersion(Artifact3) = Artifact2$$

$$isVersionOf(Artifact3, Artifact2) \equiv hasVersion(Artifact2, Artifact3)$$

- ❖ John, güncelleme işleminin ardından oluşan yeni kaydı okumaktadır. Ardından bu kayıt için de erişim geçmişi oluşmakta ve erişim geçmişi için izin ve yasaklar tanımlanmaktadır.

hasAccessHistory3(Artifact3) = (Clinical Test, John Smith, Read, 2014-10-02 11:00)

wasAccessedBy(AccessHistory3) = John Smith

isAccessHistoryOf(AccessHistory3) = Artifact3

hasPermission4(AccessHistory3) = (Diana Jodie Black, Read)

hasProhibition1(AccessHistory3) = (Read, Write, Nurse, Doctor, LaboratoryAssistant, Clinical Test, Treatment, Precaution)

Hasta, yaratılan model üzerinde bulaşıcı hastalık kayıtlarına ait erişim geçmişlerini sorgulayabilmektedir. Erişim geçmişi için, önce TC Kimlik numarası ile model üzerinde kişi sorgulanmaktadır. Eğer kişi var ise, sahibi olduğu kayıtların ilişkilendirildiği erişim geçmişi kayıtları listelenmektedir. Şekil 4.1.2.1.2’de aşağıda yer alan sorgunun sonucu gösterilmiştir.

```
PREFIX ProvenanceOnto_20140911:
<http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#>
SELECT ?person ?accessHistory ?accessDetail ?accessedDate
WHERE{
  ?person ProvenanceOnto_20140911:personIdentityNumber
    "12345678901"^^<http://www.w3.org/2001/XMLSchema#string> ;
  ProvenanceOnto_20140911:isOwnerOf ?artifact .
  ?artifact ProvenanceOnto_20140911:hasAccessHistory ?accessHistory.
  ?accessHistory ProvenanceOnto_20140911:accessedDate ?accessedDate;
  ProvenanceOnto_20140911:accessDetail ?accessDetail . }
```

accessHistory	accessDetail	accessedDate
ProvenanceOnto_20140911:AccessHistory_3	"Clinical Test, John Smith, Read"	"2014-10-02 11:00"
ProvenanceOnto_20140911:AccessHistory_2	"Clinical Tests, Mark Terry, Write"	"2014-10-01 16:00"
ProvenanceOnto_20140911:AccessHistory_1	"Clinical Tests, John Smith, Read"	"2014-10-01 15:30"

Şekil 4.1.2.1.2 Erişim geçmiş için çalıştırılan SPARQL sorgusunun sonucu

Bu şekilde, erişim geçmişi sorgulanarak güvenlik ihlalleri tespit edilebilmektedir. Hasta bulaşıcı hastalık kayıtlarına kimin hangi amaçla, hangi tarihte eriştiğini görebilmekte, erişim izin ve yasaklarını kullanarak erişim denetimi yapabilmektedir. Böylece, hastanın bulaşıcı hastalık kayıtları yetkisiz erişimden korunmaktadır.

4.1.2.2 Köken Modeli ile hasta kayıtları üzerinde güvenlik ve gizlilik ihlallerinin tespit edilmesi örneği

Bu bölümde, Gizlilik Farkında Anlamsal Köken Modeli üzerinde hasta kayıtlarına erişim geçmişi incelenerek güvenlik ve gizlilik ihlallerinin tespit edilmesine yönelik bir örnek anlatılmaktadır. Bulaşıcı hastalıklar etki alanı ile Gizlilik Farkında Anlamsal Köken Modeli'nin eşleştirmesine Bölüm 3'te ve Çizelge 3.2'de değinilmektedir. Bu eşleştirme dikkate alınarak HIV hastalığı için örnek gerçekleştirimi yapılmaktadır.

HIV, bağışıklık sistemine zarar veren bir virüstür. Bu virüs, vücut sıvıları aracılığıyla bulaşmaktadır. HIV için tedavi yöntemleri mevcuttur, ancak virüsten korunmak için herhangi bir aşı bulunmamaktadır. AIDS ise, HIV enfeksiyonunun son aşamasıdır (WHO, 2014).

Birleşmiş Milletler HIV/AIDS Ortak Programı (UNAIDS), raporuna göre Dünya'da 35 milyon yetişkin HIV virüsü taşımaktadır. Türkiye'de ise 6 bin 802 HIV/AIDS hastası kayıtlıdır. Ancak bu sayı gerçek değeri yansıtmamaktadır, çok daha fazla sayıda HIV/AIDS hastasının bulunduğu tahmin edilmektedir. Türkiye'de bu hastalığın yayılma nedenleri arasında; nüfusun genç olması, cinsel yolla bulaşan hastalıklar konusunda bilgilendirmenin az olması, yurtdışından gelen turistler ve damar içi madde kullanımları sayılmaktadır. HIV virüsü; korunmasız cinsel ilişki, anneden bebeğine geçmekte ve kan yoluyla bulaşmaktadır (Yeditepe Üniversitesi, 2015).

Tüm bulaşıcı hastalıklarda olduğu gibi HIV virüsünün de erken teşhisi hayati önem taşımaktadır. HIV virüsünün teşhisi için, virüsün vücuda girdiği zamana göre farklı testler uygulanmaktadır. Virüs vücuda girdikten 9-11 gün sonra, HIV PCR (Viral DNA,RNA) testi; 3 hafta ile 3 ay arasında, p24 antijen testi; 3-6 ay sonra, ELISA(Anti-HIV) testi yapılmaktadır (WHO, 2014).

Bu örnekte, doktor hastanın şikayetlerini dinledikten sonra hastanın HIV virüsü taşıyıp taşımadığının kontrol edilmesi gerektiğine karar vermektedir. Doktor, hastalığın teşhisi için HIV PCR testi talep etmekte, hasta ise sağlık kayıtlarını yetkisiz erişimlerden korumak istemektedir.

Bu örneğin *ALCQ DL* dilinde gösterimine aşağıda yer verilmektedir:

- John Smith, doktor rolü ile; Susi Anderson, hasta rolü ile; Pamela Rose Rowe ise hem laboratuvar asistanı hem de hemşire rolü ile ilişkilendirilmektedir.

$hasRole(John\ Smith) \equiv Doctor,$ $hasRole(Susi\ Anderson) \equiv Patient$
 $hasRole(Pamela\ Rose\ Rowe) \equiv LaboratoryAssistant \ \& \ Nurse$

- HIV PCR testi, HIV teşhisi için yapılmaktadır.

$isPerformedFor(HIV\ PCR, HIV) \equiv hasOperation(HIV, HIV\ PCR)$

- Hemşire, hasta Susi Anderson'ın kişisel bilgilerini ve doktor John Smith'in HIV PCR testi talebini sisteme girmektedir.

$requests(John, HIV\ PCR) \equiv isRequestedBy(HIV\ PCR, John)$

$creator(Artifact5) = Pamela$

$Artifact5 = Susi's\ PersonalInformation \cup Susi's\ HIV\ PCR\ Test\ Request$

- Sisteme kaydedilen bu veriler, Susi'ye ait olduğundan bir sahiplik ilişkisi yaratılmaktadır.

$isOwnerOf(Susi, Artifact5) \equiv hasOwnerOf(Artifact5, Susi)$

- Hasta Susi, kayıtlı verileri için erişim yetkilerini tanımlamaktadır.

$hasPermission8(Artifact5) = (Write, Nurse, Clinical\ Test)$

- Ardından HIV PCR testi yapılmakta ve test sonucu çıktı olarak verilmektedir.

$isInputOf(Artifact5, HIV\ PCR) \equiv hasInput(HIV\ PCR, Artifact5)$

$Artifact6 = HIVPCR_TestResult$

$isOutputOf(Artifact6, HIV\ PCR) \equiv hasOutput(HIV\ PCR, Artifact6)$

- Test sonuçları için yine hasta tarafından erişim yetkilendirmesi yapılmaktadır.

$isOwnerOf(Susi Anderson, Artifact6) \equiv hasOwner(Artifact6, Susi Anderson)$

$hasPermission9(Artifact6) = (Read, John Smith, Clinical Test)$

$hasPermission10(Artifact6) = (Write, LaboratoryAssistant, Clinical Test)$

$hasProhibition3(Artifact6) = (Read, Write, Nurse, Clinical Test \& Treatment \& Precaution)$

- ❖ Doktor John, test sonuçlarını görüntülemektedir. Test sonuçları görüntülediğinde, erişim geçmişi kaydı oluşturulmaktadır.

$hasAccessHistory4(Artifact6) = (Clinical Tests, John Smith, Read, 2015-03-18 13:30)$

$wasAccessedBy(AccessHistory4) \equiv John Smith$

$isAccessHistoryOf(AccessHistory4) \equiv Artifact6$

- Erişim kayıtları için hasta tarafından erişim yetkilendirilmesi yapılmaktadır.

$hasPermission4(AccessHistory4) = (Pamela Rose Rowe, Read)$

$hasProhibition1(AccessHistory4) = (Read, Write, Nurse, Doctor, LaboratoryAssistant, Clinical Test, Treatment, Precaution)$

- Laboratuvar asistanı Pamela Rose Rowe, eksikleri tamamlamak için test sonucu güncellemek istemektedir. Test sonucunu güncellemek için yeni bir işlem talebi girilmektedir.

$requests(Pamela Rose Rowe, Update Artifact6) \equiv isRequestedBy(Update Artifact6, Pamela Rose Rowe)$

- Test sonucu, bu işleme girdi olarak verilmektedir. İşlem laboratuvar asistanı tarafından gerçekleştirildikten sonra yeni bir çıktı oluşmaktadır. Bu çıktı, bir önceki test sonucunun yeni bir versiyonu olmaktadır. Girdi ve çıktı olarak yaratılan her veri üzerinde yapılan her değişiklik ve

erişim için sisteme bir erişim geçmişi kaydı olarak eklenmektedir. Yine bir önceki test sonucunda olduğu gibi hasta tarafından erişim yetkilendirmesi yapılmaktadır.

$$\begin{aligned}
 &isInputOf(Artifact6, Update Artifact6) \equiv hasInput(Update Artifact6, \\
 &\hspace{15em}Artifact6) \\
 &performs(Pamela Rose Rowe, Update Artifact6) \equiv \\
 &\hspace{4em}isPerformedBy(Update Artifact6, Pamela Rose Rowe) \\
 &hasAccessHistory5(Artifact6) = (Clinical Tests, Pamela Rose Rowe, \\
 &\hspace{4em}Write, 2015-03-18 13:35) \\
 &wasAccessedBy(AccessHistory5) \equiv Pamela Rose Rowe \\
 &isAccessHistoryOf(AccessHistory5) \equiv Artifact6 \\
 &hasPermission4(AccessHistory5) = (Pamela Rose Rowe, Read) \\
 &hasProhibition1(AccessHistory6) = (Read, Write, Nurse, Doctor, \\
 &\hspace{15em}LaboratuaryAssistant, Clinical \\
 &\hspace{15em}Test, Treatment, Precaution) \\
 &isOutputOf(Artifact7, Update Artifact6) \equiv hasOutput(Update Artifact6, \\
 &\hspace{15em}Artifact7) \\
 &hasPermission9(Artifact7) = (Read, John Smith, Clinical Test) \\
 &hasPermission10(Artifact7) = (Write, LaboratuaryAssistant, Clinical \\
 &\hspace{10em}Test) \\
 &hasProhibition3(Artifact7) = (Read, Write, Nurse, Clinical \\
 &\hspace{10em}Test&Treatmet &Precaution) \\
 &priorVersion(Artifact7) = Artifact6 \\
 &isVersionOf(Artifact7, Artifact6) \equiv hasVersion(Artifact6, Artifact7)
 \end{aligned}$$

Şekil 4.1.2.2.1’de örneğin şematik gösterimi verilmektedir.

Şekil 4.1.2.2.1 HIV teşhisi ve hasta kayıtları üzerinde erişim kontrolü

Bu sađlık kuruluřunda laboratuvar asistanı Pamela, aynı zamanda hemřire olarak görev almaktadır. Hasta Susi, HIV PCR testi sonularına sadece doktoru iin klinik test amaciđıyla okuma yetkisi verirken, laboratuvar asistanlarına test sonucu iin yazma yetkisi vermektedir. Hemřire rolünde olan sađlık kurumu alıřanlarının ise test sonucuna eriřimini ise engellemektedir.

Yaratılan model üzerinde bulařıcı hastalık kayıtlarına ait eriřim gemiřleri, bulařıcı hastalık kayıtları iin hasta tarafından tanımlanan yetki ve kısıtlamalar taranarak gvenlik ihlalleri tespit edilebilmektedir. Gvenlik ihlallerini tespit etmek iin ařađıdaki adımlar gerekleřtirilmektedir:

- Belirli bir kayıt ya da tm kayıtlar iin nce eriřim gemiř getirilmektedir.
- Eriřim gemiřini yapan kiřiye ait rol bilgisi, eriřim tarihi ve eriřim amacı ile kayıt iin tanımlanan izinlerde yer alan rol veya kiři bilgisi, eriřim amacı ve izin bařlangı ve bitiř tarihleri kıyaslanıp iznin olup olmadığı kontrol edilmektedir.
- Aynı řekilde eriřim gemiřini yapan kiřiye ait rol bilgisi, eriřim tarihi ve eriřim amacı ile kayıt iin tanımlanan kısıtlarda yer alan rol veya kiři bilgisi, eriřim amacı ve kısıtlamanın bařlangı ve bitiř tarihleri kıyaslanıp kısıtlamanın olup olmadığı kontrol edilmektedir.
- Bu iki sonuca gre tanımlı izni olan ve herhangi bir kısıtlama kriterine takılmayan eriřim gemiřleri dıřında kalanlar gvenlik ihlallerini gstermektedir.

rnekte yer alan `Artifact` nesnesinin izin, yasak ve eriřim gemiřlerini listelemek iin alıřtırılan SPARQL sorgusuna ařađıda yer verilmektedir. řekil 4.1.2.2.2’de alıřtırılan bu sorgunun sonucu gsterilmektedir.

```
PREFIX ProvenanceOnto_20140911:
<http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX terms: <http://purl.org/dc/terms/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?s ?artifactName ?permissionName ?prohibitionName
        ?accessHistoryName
WHERE { ?s ProvenanceOnto_20140911:artifactName
        "Artifact_6"^^<http://www.w3.org/2001/XMLSchema#string>;
        ProvenanceOnto_20140911:artifactName ?artifactName;
        OPTIONAL { ?s ProvenanceOnto_20140911:hasPermission
```

```

?permission .
?permission ProvenanceOnto_20140911:permissionName
?permissionName . }
OPTIONAL { ?s ProvenanceOnto_20140911:hasProhibition
?prohibition .
?prohibition
ProvenanceOnto_20140911:prohibitionName
?prohibitionName .}
OPTIONAL { ?s ProvenanceOnto_20140911:hasAccessHistory
?accessHistory .
?accessHistory
ProvenanceOnto_20140911:accessHistoryName
?accessHistoryName . } }

```

artifactName	permissionName	prohibitionName	accessHistoryName
"Artifact_6"	"Permission_10"	"Prohibition_3"	"AccessHistory_4"
"Artifact_6"	"Permission_10"	"Prohibition_3"	"AccessHistory_5"
"Artifact_6"	"Permission_9"	"Prohibition_3"	"AccessHistory_4"
"Artifact_6"	"Permission_9"	"Prohibition_3"	"AccessHistory_5"

Şekil 4.1.2.2.2 Hasta kayıtları üzerinde güvenlik ihlallerinin kontrol edilmesi için çalıştırılan SPARQL sorgusunun sonucu

Güvenlik ihlalleri tespiti için, Artifact nesnesi ile ilişkili olan Permission, Prohibition ve AccessHistory nesnelerinde yer alan özellikler programatik olarak karşılaştırılmakta, izin ve yasaklara uygun olmayan AccessHistory nesneleri saptanmaktadır.

Bu şekilde, erişim geçmişi, izin ve kısıtlamalar kullanılarak gerçekleştirilen sorgular yardımıyla güvenlik ihlalleri tespit edilebilmektedir. Hasta, bu örnekte kısıtlamış olduğu halde hemşire rolüne sahip bir sağlık çalışanının HIV PNR test sonucuna eriştiğini görmektedir. Bu sayede, kişiye verilen rollerin, hasta tarafından tanımlanan izin ve kısıtların tekrar düzenlenip, gizlilik ihlallerinin önüne geçilmesi sağlanmaktadır.

Bir sonraki bölümde, bulaşıcı hastalıklardan korunma yöntemi olarak aşıyı sisteme dahil etmek için yararlanılan Ulusal Aşı Ontolojisi detaylandırılmaktadır.

4.2 Ulusal Aşı Ontolojisi

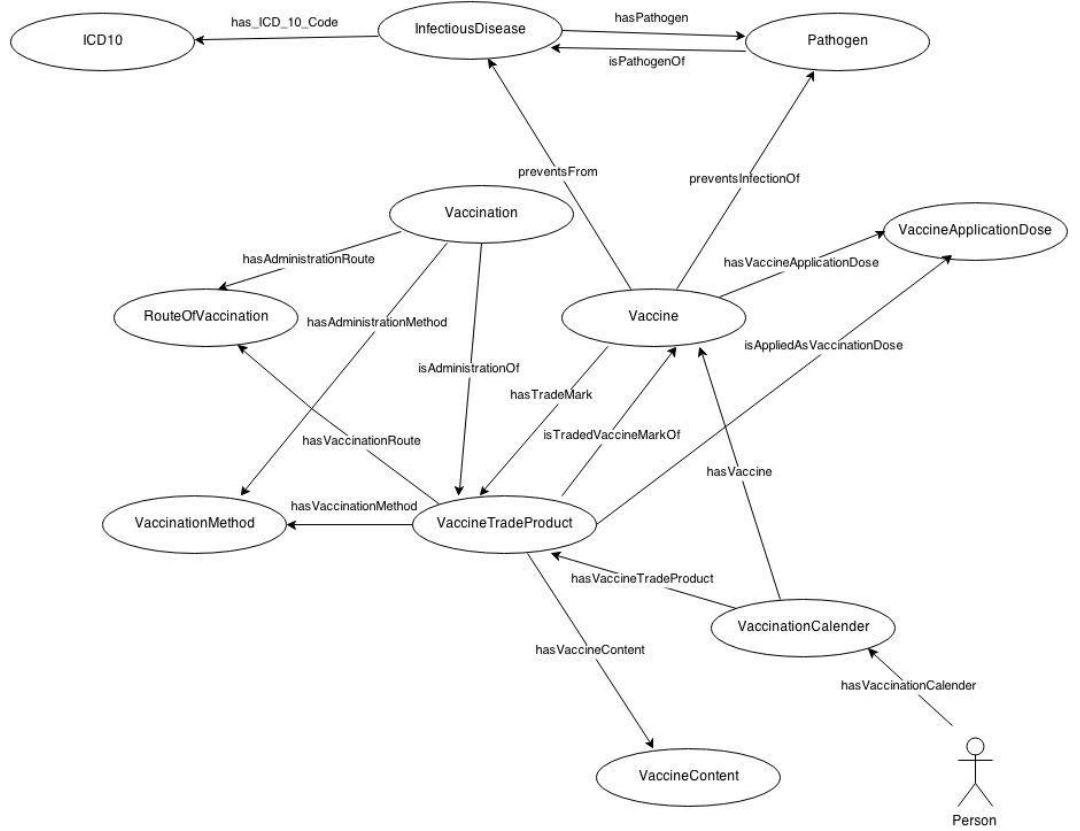
Aşı, hastalık etkeni vücuda girmeden önce, vücuda zayıtlatılmış hastalık etkeninin enjekte edilmesi yoluyla, hayvan ve insanların vücudu tarafından hastalığın önceden tanınıp, bağışıklık kazanılması için kullanılan bir çözüldür.

Bulaşıcı hastalıkların büyük kitlelere yayılmasını önlemek için kullanılan en ucuz ve etkin yoldur. Ayrıca, kızamık gibi bazı bulaşıcı hastalıklar bir kez bile geçirildiğinde zeka geriliği, gelişim bozukluğu gibi kalıcı hasarlar bırakabilmektedir. Bu nedenle bağışıklık ekstra önem kazanmaktadır. Yeni doğan ve çocuklar için aşılama yapılan tüm yatırımlar çok büyük bir geri dönüş sağlamaktadır. Her yıl çocuklar üzerinde %97'ye varan oranlarda aşılama yapılmaktadır (İstanbul İl Sağlık Müdürlüğü, 2015). Bu sayede, binlerce yaşam kurtulmakta ve milyonlarca insan bulaşıcı hastalıklardan korunmaktadır. Sağlık sektöründe ise milyonlarca lira harcamanın önüne geçilmektedir. Ancak her birey için yapılan aşılarla ilişkin kişisel bir kayıt bulunmamaktadır. Kişilerin ellerindeki aşı kartları dışında, aşı geçmişlerine erişebilecekleri bir sistem yoktur.

Tez kapsamında, bulaşıcı hastalıklar için tedavi ve teşhis yöntemlerinin yanısıra hastalıktan korunma yöntemleri de önerilmektedir ve geçmişte uygulanan tedavi, teşhis ve korunma yöntemlerine ait kayıtlar da sorgulanacaktır. Kişiler, aşı geçmişlerine erişebilecek ve bir sonraki aşı tarihlerinden önceden haberdar olabileceklerdir. Bu nedenle Ulusal Aşı Ontolojisi (UAO)'den yararlanılacaktır (Sezer, 2014).

UAO; aşılama, aşı takvimi, aşı uygulama dozu, aşılanma yolu, aşı içeriği gibi temel kavramları tek bir çatı altında toplamaktadır. Tez çalışmasında, UAO içinde yer alan tüm kavramlar kullanılmayacak, sadece Şekil 4.1'de gösterilmekte olan kavramlar ve ilişkilerden yararlanılacaktır. Bunlar; Vaccination, RouteOfVaccination, Vaccine, VaccinationMethod, VaccineTradeProduct, Pathogen, VaccinationCalender, VaccineApplicationDose ve InfectiousDisease kavramlarıdır. Vaccination kavramı, bireyin bulaşıcı hastalıklardan korunması için aşı uygulaması olayını temsil etmektedir. Vaccine, uygulanan aşıya ilişkin bilgileri tutmak için yaratılmış bir kavramdır. VaccineTradeProduct ise, aşığı ürün olarak insanların satışına sunan markayı ifade etmektedir. Örneğin; HepavaxGene markası Hepatit B aşısını satan ticari bir markadır. Hepatit B, Vaccine kavramı ile temsil edilirken; HepavaxGene ticari ürünü, VaccineTradeProduct ile

temsil edilmektedir. `RouteOfVaccination`, aşılama yolunu belirtmek için kullanılmaktadır. Aşılama yolu; deri içi (`IntradermalRoute`), kas içi (`IntramuscularRoute`), burun (`IntranasalRoute`), periton içi (`IntraperitonealRoute`), damar içi (`IntravenousRoute`), ağız (`OralRoute`) ya da deri altı (`SubcutaneousRoute`) olabilmektedir. Aşılama yollarının her biri için UAO'de birer tanımlama yapılmıştır. `VaccinationMethod` kavramı, aşının uygulanma yöntemini temsil etmek için kullanılmaktadır. Aşı uygulama yöntemi; damla, enjeksiyon ya da sprey olabilmektedir. `VaccineApplicationDose`, aşının uygulanma dozunu ifade etmek için kullanılmaktadır. Uygulanma dozu; dozu arttırarak aşılama (`BoostVaccination`), temel aşılama (`PrimaryVaccination`), dörtlü aşılama (`QuaternaryVaccination`), ikincil aşılama (`SecondaryVaccination`) ve üçüncül aşılama (`TertiaryVaccination`) olarak kavramlara ayrılmaktadır. `VaccineContent`, aşının içeriğinde yer alan maddelerin belirtilmesi için kullanılmaktadır. `VaccinationCalender` ise, her bireyin aşı takvimin tutmak için yaratılmış olan bir kavramdır. Bu kavram altında, cinsiyete göre önerilen ve zorunlu aşı takvimleri tanımlanabilmesine olanak sağlanmaktadır. Cinsiyete göre önerilen aşı takvimi, devlet tarafından karşılanmayan aşıları tutmaktadır. Zorunlu aşı takviminde yer alan aşılar ise, ücretsiz olarak yapılmaktadır. Bunlara ek olarak, her aşının koruduğu hastalık bilgisinin ve patojenin belirtilmesi gerekmektedir. Bulaşıcı hastalıklar etki alanında çalışma yapıldığı için aşının sadece bulaşıcı hastalıkla ve patojen ile ilişkilendirildiği kısımlar kullanılacaktır. UAO'de yer alan `Pathogen`, aşının koruyacağı patojeni temsil ederken; `InfectiousDisease` kavramı ise, aşının koruma sağlayacağı bulaşıcı hastalığı temsil etmektedir (Sezer, 2014). Tez çalışmasında, bulaşıcı hastalık kavramlarının tutulması için BHO kullanılmaktadır. UAO'de ve BHO'de yer alan bulaşıcı hastalık kavramını eşleştirmek için ICD 10 kodundan yararlanılmaktadır. UAO üzerinde, ICD 10 kodu kullanılarak, önce bulaşıcı hastalığa, ardından da ilişkili aşıya erişilmektedir.



Şekil 4.2.1 UAO üzerinde aşı ve bulaşıcı hastalıkları ifade eden kavramlar

Tez çalışmasında, Ulusal Aşı Ontolojisi'nin Şekil 4.2.1'de gösterilen atomik kavram ve rolleri kullanılmaktadır. Bu kavram ve rollere ilişkin detaylar aşağıda yer almaktadır:

- Atomik kavramlar; Vaccination, Vaccine, RouteOfVaccination, VaccineTradeProduct, VaccinationMethod, VaccinationCalender, VaccineContent, VaccineApplicationDose, Pathogen ve InfectiousDisease kavramlarıdır.
- Atomik rol olan isTradedVaccineMarkOf; VaccineTradeProduct ile Vaccine arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, hasTradeMark atomik rolü ile ters ilişkilidir, dolayısıyla Vaccine nesnesi ile VaccineTradeProduct nesnesi arasında bir bağlantı oluşturmaktadır.

- Atomik rol olan hasAdministrationRoute; Vaccination ile RouteOfVaccination arasında, hasAdministrationMethod; Vaccination ile VaccinationMethod arasında ve isAdministrationOf ise, Vaccination ile VaccineTradeProduct arasında bir bağlantı kurulmasını sağlamaktadır.
- Atomik rol olan preventsFrom; Vaccine ile InfectiousDisease arasında, preventsInfectionOf; Vaccine ile Pathogen arasında ve hasVaccineApplicationDose; Vaccine ile VaccineApplicationDose arasında bir bağlantı kurulmasını sağlamaktadır.
- Atomik rol olan hasVaccinationRoute; VaccineTradeProduct ile RouteOfVaccination arasında, isAppliedAsVaccinationDose; VaccineTradeProduct ile VaccineApplicationDose arasında, hasVaccinationMethod; VaccineTradeProduct ile VaccinationMethod arasında ve hasVaccineContent ise, VaccineTradeProduct ile VaccineContent arasında bir bağlantı kurulmasını sağlamaktadır.
- Atomik rol olan hasVaccineTradeProduct; VaccinationCalender ile VaccineTradeProduct arasında, hasVaccine; VaccinationCalender ile Vaccine arasında ve hasVaccinationCalender; Person ile VaccinationCalender arasında bir bağlantı kurulmasını sağlamaktadır.
- Atomik rol olan hasPathogen; InfectiousDisease ile Pathogen arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, isPathogenOf atomik rolü ile ters ilişkilidir, dolayısıyla Pathogen nesnesi ile InfectiousDisease nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan has_ICD_10_Code, InfectiousDisease ile ICD10 arasında bir ilişki kurulmasını sağlamaktadır.

Ulusal Aşı Ontolojisi için tanımlanan kuralların *ALCQ DL* ifadeleri aşağıda yer almaktadır:

- ❖ Her ticari aşı, ticari bir marka tarafından satılmaktadır.

$$\forall VaccineTradeProduct(isTradedVaccineMarkOf(VaccineTradeProduct, Vaccine)) \leftrightarrow \exists Vaccine(hasTradeMark(Vaccine, VaccineTradeProduct))$$

- ❖ Her aşılama için damar içi, ağız ve kas içi gibi bir uygulama yolu bulunmaktadır.

$$\forall Vaccination(hasAdministrationRoute(Vaccination, RouteOfVaccination)), RouteOfVaccination \subseteq hasAdministrationRoute. Vaccination$$

- ❖ Her aşılama için enjeksiyon ve damla gibi bir aşılama yöntemi bulunmaktadır.

$$\forall Vaccination(hasAdministrationMethod(Vaccination, VaccinationMethod)), VaccinationMethod \subseteq hasAdministrationMethod. Vaccination$$

- ❖ Her aşılama, bir ticari aşı ürününün uygulanmasıdır.

$$\forall Vaccination(isAdministrationOf(Vaccination, VaccineTradeProduct)), VaccineTradeProduct \subseteq isAdministrationOf. Vaccination$$

- ❖ Her aşı, bir bulaşıcı hastalıktan korunmayı sağlamaktadır.

$$\forall Vaccine(preventsFrom(Vaccine, InfectiousDisease)), InfectiousDisease \subseteq preventsFrom. Vaccine$$

- ❖ Her aşı, en az bir patojenden hastalık bulaşmasını önlemektedir.

$$\forall Vaccine(preventsInfectionOf(Vaccine, Pathogen)), Pathogen \subseteq preventsInfectionOf. Vaccine$$

- ❖ Her aşı için bir uygulama dozu bulunmaktadır.

$\forall Vaccine(hasVaccineApplicationDose (Vaccine, VaccineApplicationDose)),$
 $VaccineApplicationDose \subseteq hasVaccineApplicationDose. Vaccine$

- ❖ Her ticari aşı ürünü için bir aşılama yolu bulunmaktadır.

$\forall VaccineTradeProduct(hasVaccinationRoute(VaccineTradeProduct, RouteOfVaccination)),$
 $RouteOfVaccination \subseteq hasVaccinationRoute.VaccineTradeProduct$

- ❖ Her ticari aşı ürünü bir aşılama dozu olarak uygulanmaktadır.

$\forall VaccineTradeProduct(isAppliedAsVaccinationDose (VaccineTradeProduct, VaccineApplicationDose)),$
 $VaccineApplicationDose \subseteq isAppliedAsVaccinationDose.VaccineTradeProduct$

- ❖ Her ticari aşı ürünü için bir aşılama metodu bulunmaktadır.

$\forall VaccineTradeProduct (hasVaccinationMethod (VaccineTradeProduct, VaccinationMethod)),$
 $VaccinationMethod \subseteq hasVaccinationMethod. VaccineTradeProduct$

- ❖ Her ticari aşı ürünü için aşı içeriği bulunmaktadır.

$\forall VaccineTradeProduct(hasVaccineContent (VaccineTradeProduct, VaccineContent)),$
 $VaccineContent \subseteq hasVaccineContent.VaccineTradeProduct$

- ❖ Her ticari aşı ürünü için bir aşılama takvimi bulunmaktadır.

$\forall VaccinationCalender(hasVaccineTradeProduct(VaccinationCalender, VaccineTradeProduct)),$

$VaccineTradeProduct \subseteq$
 $hasVaccineTradeProduct.VaccinationCalender$

- ❖ Her aşılama takvimi bir aşı ile ilişkilendirilmektedir.

$\forall VaccinationCalender(hasVaccine(VaccinationCalender, Vaccine)),$
 $Vaccine \subseteq hasVaccine.VaccinationCalender$

- ❖ Her birey bir aşılama takvimi ile ilişkilendirilmektedir.

$\forall Person(hasVaccinationCalender(Person, VaccinationCalender)),$
 $VaccinationCalender \subseteq hasVaccinationCalender.Person$

- ❖ Her bulaşıcı hastalık bir patojen nedeniyle oluşmaktadır. Bu nedenle, her bulaşıcı hastalık bir patojen ile ilişkilendirilmektedir.

$\forall InfectiousDisease(hasPathogen(InfectiousDisease,$
 $Pathogen)) \leftrightarrow \exists Pathogen(isPathogenOf(Pathogen,$
 $InfectiousDisease))$

- ❖ Her bulaşıcı hastalık bir ICD-10 kodu ile ilişkilendirilmektedir.

$\forall InfectiousDisease(has_ICD_10_Code(InfectiousDisease, ICD10)),$
 $ICD10 \subseteq has_ICD_10_Code.InfectiousDisease$

Tez kapsamında, aşı takvimi için yukarıda bahsedilen modele yeni özellikler eklenmiştir. Aşı dozlarının uygulanma periyotlarını tutmak için VaccineShotPeriod sınıfı eklenmiştir. Bu sınıf, Vaccine sınıfı ile hasVaccineShotPeriod tanımı kullanılarak ilişkilendirilmektedir. İlk aşılama yapıldıktan sonra yapılacak aşılamaların ne kadar süre sonra yapılması gerektiğini belirtmek için kullanılacaktır. VaccineShotPeriod sınıfı, VaccineApplicationDose sınıfı ile isRelatedWith özelliği ile bağlanmaktadır. Çizelge 4.2.1’de VaccineShotPeriod sınıfın özelliklerine yer verilmektedir.

Çizelge 4.2.1 VaccineShotPeriod sınıfının özellikleri

VaccineShotPeriod Sınıfı	
Özellik	İşlevi
periodDay	Bir sonraki aşılanmanın kaç gün sonra yapılması gerektiğini ifade eder.
periodMonth	Bir sonraki aşılanmanın kaç ay sonra yapılması gerektiğini ifade eder.
periodYear	Bir sonraki aşılanmanın kaç yıl sonra yapılması gerektiğini ifade eder.
isRelatedWith	VaccineShotPeriod nesnesinin, hangi aşılama dozu(SecondaryVaccination, BoostVaccination vb.) ile ilişkili olduğunu tutan özelliktir.

Aşıların uygulandığı ve önerilen tarihleri tutmak için PersonalVaccineCalender sınıfı VaccinationCalender sınıfının alt sınıf olarak eklenmiştir. Person sınıfı ile PersonalVaccineCalender sınıfı arasındaki bağlantı, hasPersonalVaccineCalender özelliği ile kurulmaktadır. İlk aşılama yapıldıktan sonra PersonalVaccineCalender nesnesi yaratılır, ilişkili olduğu aşıda yer alan bilgiler kullanılarak, sonraki aşılamaların yapılacağı tarihleri içeren bir aşılama takvimi sunulmasını sağlamaktadır. Daha sonraki aşılamalarda ise bu takvim güncellenmektedir. Çizelge 4.2.2’de PersonalVaccineCalender sınıfının özellikleri gösterilmektedir.

Çizelge 4.2.2 PersonalVaccineCalender sınıfının özellikleri

PersonalVaccineCalender Sınıfı	
Özellik	İşlevi
calendarName	Aşı takviminin adını tutar.
vaccineName	Aşının adını tutar.
shotName	Hangi aşılama dozunun (SecondaryVaccination, BoostVaccination vb.) yapıldığı bilgisini tutar.
administratedDate	Aşılanmanın yapıldığı tarihi tutar.
proposedDate	Önerilen aşılama tarihi bilgisini tutar.
hasVaccine	PersonalVaccineCalender nesnesinin hangi aşı ile ilişkili olduğunu tutar.
isRelatedWith	PersonalVaccineCalender nesnesinin, hangi aşılama dozu(SecondaryVaccination, BoostVaccination vb.) ile ilişkili olduğunu tutan özelliktir.

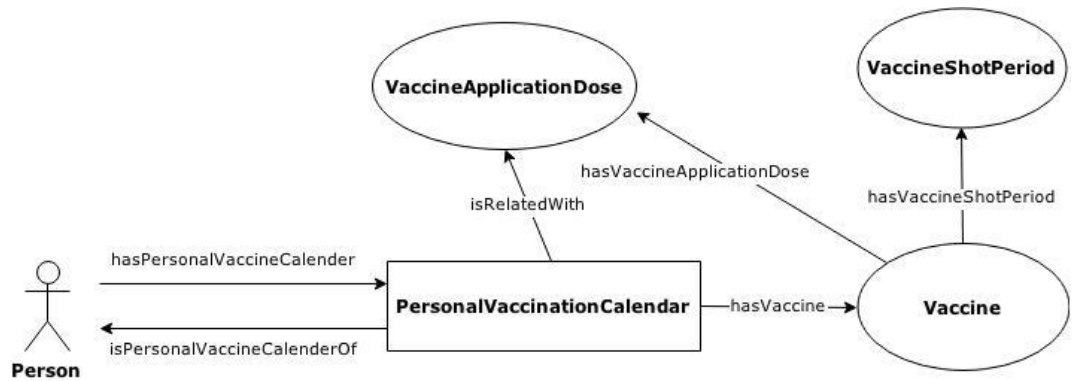
isPersonalVaccineCalenderOf	Aşı takvimin, kime ait olduğunu göstermek için, PersonalVaccineCalender nesnesinin, Person nesnesi ile ilişkisini tutar. hasPersonalVaccineCalender özelliği ile ters ilişkilidir, Person ile PersonalVaccineCalender arasındaki ilişkiyi tutmaktadır.
-----------------------------	--

UAO'nun kullanım kolaylığını arttırmak için aşı isimleri, aşı uygulama dozları gibi bazı sınıflara isimlerini tutmak için nesne özellikleri eklenmiştir. Çizelge 4.2.3'te eklenen diğer özelliklere yer verilmektedir.

Çizelge 4.2.3 UAO'nun kullanım kolaylığını arttırmak için eklenen özellikler

Sınıf Adı	Özellik	İşlevi
Vaccine	vaccineName	Aşının adını tutar.
VaccineApplicationDose	vaccineApplicationDoseName	Aşılama dozunun adını tutar.

UAO'ya eklenen PersonalVaccinationCalender ve VaccineShotPeriod sınıflarının ontolojide yer alan Vaccine, VaccineApplicationDose ve Person ile ilişkileri Şekil 4.2.2'de gösterilmektedir.



Şekil 4.2.2 UAO'ya eklenen PersonalVaccinationCalender ve VaccineApplicationDose sınıflarının diğer sınıflarla ilişkileri

Şekil 4.2.2’de yer alan atomik kavram ve roller aşağıda yer almaktadır:

- Atomik kavramlar; Vaccine, VaccineShotPeriod, Person, PersonalVaccinationCalendar, VaccineApplicationDose kavramlarıdır.
- Atomik rol olan hasPersonalVaccineCalender; Person ile PersonalVaccinationCalendar arasında bir ilişki kurulmasını sağlamaktadır. Bu atomik rol, isPersonalVaccineCalenderOf atomik rolü ile ters ilişkilidir, dolayısıyla PersonalVaccinationCalendar nesnesi ile Person nesnesi arasında bir bağlantı oluşturmaktadır.
- Atomik rol olan isRelatedWith; PersonalVaccinationCalendar ile VaccineApplicationDose arasında, hasVaccine; PersonalVaccinationCalendar ile Vaccine bir bağlantı kurulmasını sağlamaktadır.
- Atomik rol olan hasVaccineApplicationDose; Vaccine ile VaccineApplicationDose arasında, hasVaccineShotPeriod; Vaccine ile VaccineShotPeriod arasında bir bağlantı kurulmasını sağlamaktadır.

Şekil 4.2.2’de yer alan kavramlar için tanımlanan kurallar aşağıdaki şekildedir:

- ❖ Bazı bireyler, aşılama takvimlerinin tutulması için kişisel aşılama takvimi ile ilişkilendirilmektedir.

$$\exists Person(hasPersonalVaccineCalender(Person, PersonalVaccinationCalendar)) \leftrightarrow \forall PersonalVaccinationCalendar(isPersonalVaccineCalenderOf(PersonalVaccinationCalendar, Person))$$

- ❖ Her kişisel aşı takvimi, bir aşı uygulama dozu ile ilişkilendirilmektedir.

$$\forall PersonalVaccinationCalendar(isRelatedWith(PersonalVaccinationCalendar, VaccineApplicationDose)),$$

$VaccineApplicationDose \subseteq$
 $isRelatedWith.PersonalVaccinationCalendar$

- ❖ Her kişisel aşı takvimi, bir aşı ile ilişkilendirilmektedir.

$\forall PersonalVaccinationCalendar(hasVaccine$
 $(PersonalVaccinationCalendar, Vaccine)),$
 $Vaccine \subseteq hasVaccine.PersonalVaccinationCalendar$

- ❖ Her aşı, bir aşı uygulama dozu ile ilişkilendirilmektedir.

$\forall Vaccine(hasVaccineApplicationDose(Vaccine,$
 $VaccineApplicationDose)), VaccineApplicationDose \subseteq$
 $hasVaccineApplicationDose.Vaccine$

- ❖ Her aşı, bir aşı uygulama periyodu ile ilişkilendirilmektedir.

$\forall Vaccine(hasVaccineShotPeriod(Vaccine,$
 $VaccineShotPeriod)), VaccineShotPeriod \subseteq$
 $hasVaccineShotPeriod.Vaccine$

Sonraki iki bölümde, Ulusal Aşı Ontolojisi üzerinde aşı ve aşı takvimi gösterimi örneklendirilmektedir.

4.2.1 Ulusal Aşı Ontolojisi üzerinde aşı gösterim örneği

Bu bölümde, Hepatit B virüsünden korunmak için kullanılan Hepatit B aşısının UAO üzerinde gösterimine değinilmektedir.

Hepatit B virüsü, sarılık olarak da bilinen bir çeşit karaciğer iltihaplanması olan Hepatit hastalığına yol açmaktadır. AIDS etkeni olan HIV'den 100 kat daha bulaşıcı olan bu hastalıktan, aşılama yolu ile korunma sağlanabilmektedir (WHO, 2015).

Hepatit B aşısı; kan ürünleriyle yakın temas halinde bulunanlara, Hepatit B virüsü taşıyıcıları ile beraber yaşayanlara ve yurt gibi toplu yaşam alanlarında yaşayanlara önerilmektedir (WHO, 2015). Hepatit B aşısı, üç ayrı doz halinde kas içine enjekte edilmektedir. İlk doz yapıldıktan bir ay sonra ikinci doz, ikinci doz uygulandıktan altı ay sonra da üçüncü doz yapılmaktadır. Birkaç yılda bir kez de etkinlik artırıcı rapel doz uygulanabilmektedir. Uygulanacak dozlar kullanılarak,

aşı takvimi önerisi oluşturulmaktadır (WHO, 2015). Kişinin cinsiyetine bağlı olarak bir aşı takvimi önerisi sunulmaktadır. UAO’de zorunlu aşı takvimi devlet tarafından karşılanan, önerilen aşı takvimi ise devlet tarafından ücreti karşılanmayan aşıları temsil etmektedir.

Sağlık sektöründe; Engerix-B, HBvaxPRO, Genhevac B, HBvaxPRO ve Hepavax-Gene gibi farklı ticari markalar Hepatit B aşısı sunmaktadır. Örnekte, ticari aşı olarak Hepavax-Gene kullanılmıştır. Hepavax-Gene; HbsAg ve koruyucu olarak Timerosal içermektedir.

Hepatit B aşı gösterimi için tanımlanan kuralların ALCQ DL dili ile ifadesi aşağıda yer almaktadır:

- ❖ Hepatit B için patojen olarak Hepatit B virüsü tanımlanmaktadır.

$$\begin{aligned} hasPathogen(HepatitisB, HepatitisB_Virus) \equiv \\ isPathogenOf(HepatitisB_Virus, HepatitisB) \end{aligned}$$

- ❖ Hepatit B için ICD-10 kodu B16 olarak tanımlanmaktadır.

$$has_ICD_10_Code(HepatitisB) \equiv B16$$

- ❖ Hepatit B hastalığından Hepatit B aşısı ile korunulmaktadır.

$$preventsFrom(HepatitisB_Vaccine) \equiv HepatitisB$$

- ❖ Hepatit B virüsünden Hepatit B aşısı ile korunulmaktadır.

$$preventsInjectionOf(HepatitisB_Vaccine) \equiv HepatitisB_Virus$$

- ❖ Hepatit B aşısı; ilk doz, ikinci doz, üçüncü doz ve rapel doz olarak yapılmaktadır.

$$\begin{aligned} hasVaccineApplicationDose(HepatitisB_Vaccine) \equiv \\ PrimaryVaccination \cup SecondaryVaccination \cup \\ TertiaryVaccination \cup BoostVaccination \end{aligned}$$

- ❖ Hepatit B aşısı, bir ticari marka olarak Hepavax-Gene ile ilişkilendirilmektedir.

$hasTradeMark(HepatitisB_Vaccine, Hepavax-Gene) \equiv$
 $isTradedVaccineMarkOf(Hepavax-Gene, HepatitisB_Vaccine)$

- ❖ Hepavax-Gene ticari aşısı, kas içine enjekte edilerek yapılmaktadır.

$hasVaccinationRoute(Hepavax-Gene) \equiv IntramuscularRoute$
 $hasVaccinationMethod(Hepavax-Gene) \equiv Injection$

- ❖ Hepavax-Gene ticari aşısı, içeriğinde HbsAg ve ThimerosalVaccinePreservative maddelerini barındırmaktadır.

$hasVaccinationContent(Hepavax-Gene) \equiv HbsAg \cup$
 $ThimerosalVaccinePreservative$

- ❖ Hepatit B aşısı, Hepavax-Gene ticari aşısının uygulanması olarak tanımlanmaktadır.

$isAdministrationOf(HepatitisB_Vaccination) \equiv Hepavax-Gene$

- ❖ Hepatit B aşılması için aşı uygulama yolu kas içine olarak tanımlanmaktadır.

$hasAdministrationRoute(HepatitisB_Vaccination) \equiv$
 $IntramuscularRoute$

- ❖ Hepatit B aşılması için aşı uygulama metodu enjeksiyon olarak tanımlanmaktadır.

$hasAdministrationMethod(HepatitisB_Vaccination) \equiv Injection$

- ❖ Hepatit B aşısı, bayanlar için önerilen aşı takvimi içinde yer almaktadır.

hasVaccine(FemaleAdviceVaccinationCalender_1) $\equiv$
HepatitisB_Vaccine

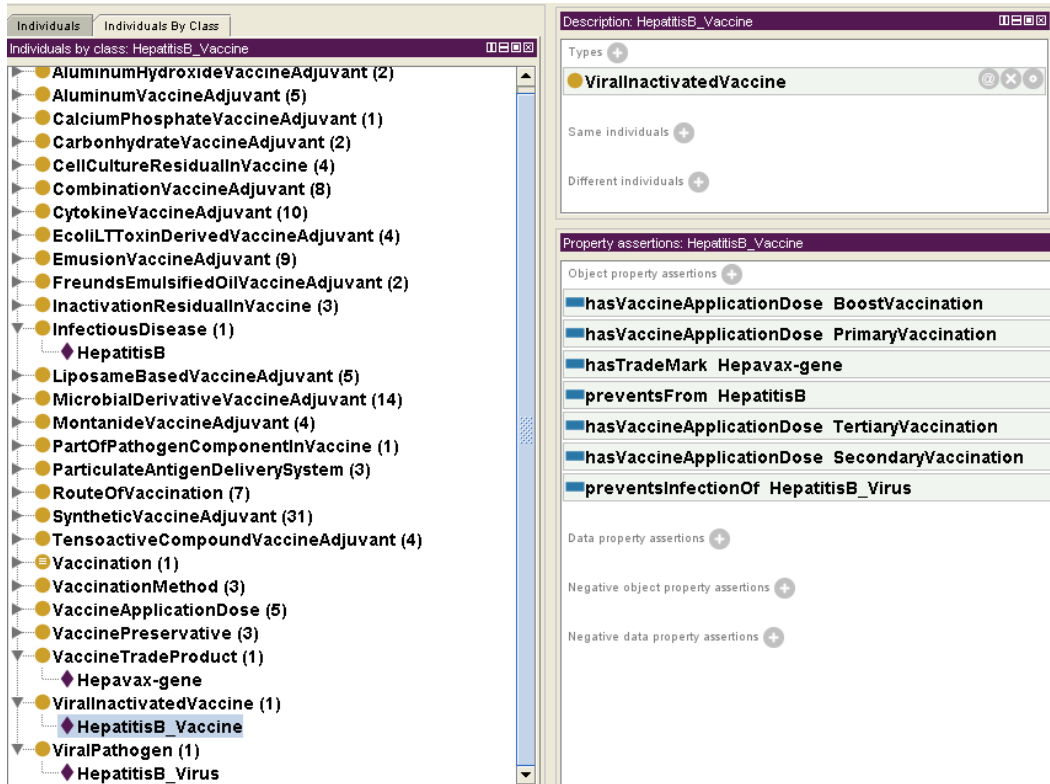
- ❖ Hepavax-Gene ticari aşısı, bayanlar için önerilen aşı takvimi içinde yer almaktadır.

hasVaccineTradeProduct(FemaleAdviceVaccinationCalender_1) $\equiv$
Hepavax-Gene

- ❖ Bir birey, bayanlar için önerilen aşı takvimi ile ilişkilendirilmektedir.

hasVaccinationCalender(FemaleAdviceVaccinationCalender_1) $\equiv$
Person_1

Hepatit B aşısı için hazırlanan bu örneğin, Protege ontoloji editörü üzerinde gösterimine Şekil 4.2.1.1’de yer verilmektedir.



Şekil 4.2.1.1 Hepatit B aşısının Protege ontoloji editörü üzerinde gösterimi

4.2.2 Ulusal Aşı Ontolojisi üzerinde aşı takvimi gösterim örneği

Bu bölümde, bir bireyin Hepatit B aşı takviminin UAO üzerinde gösterimine yer verilmektedir.

Bölüm 4.2.1’de bahsedildiği üzere, Hepatit B aşısı; kan ürünleriyle yakın temas halinde bulunanlara, Hepatit B virüsü taşıyıcıları ile beraber yaşayanlara ve yurt gibi toplu yaşam alanlarında yaşayanlara önerilmektedir. Hepatit B aşısı, üç ayrı doz halinde kas içine enjekte edilmektedir. İlk doz yapıldıktan bir ay sonra ikinci doz, ikinci doz uygulandıktan altı ay sonra da üçüncü doz yapılmaktadır. Birkaç yılda bir kez de etkinlik artırıcı rapel doz uygulanabilmektedir. UAO’de zorunlu ve önerilen aşı takvimi kavramlarına yer verilmektedir. Ancak, bu kavramlar aşılarda uygulanan doz, önerilen aşılama tarihi ve aşılanmanın yapıldığı tarihi içeren kişisel aşı takvimi ihtiyacını karşılamamaktadır. Bu nedenle kişisel aşı takvimini temsil eden *PersonalVaccinationCalendar* nesnesi oluşturulmaktadır. Bu örnekte, 12345678901 TC Kimlik numarasına sahip bir bireyin, aşı takvimi gösterilmektedir. İlk doz uygulanmış, ikinci, üçüncü ve etki artırıcı doz için aşılama tarihleri önerilmektedir.

Yukarıda bahsedilen bilgilerin, *ALCQ DL* dili ile gösterimine aşağıda yer verilmektedir:

- ❖ Bir birey, 12345678901 TC kimlik numarası ile tanımlanmaktadır.

hasTCKimlik (Person_1) ≡ 12345678901

- ❖ Birey, ilk doz için yaratılan kişisel aşı takvimi ile ilişkilendirilmektedir.

*hasPersonalVaccinationCalendar(Person1,
PersonalVaccinationCalendar1) ≡
isPersonalVaccinationCalendarOf
(PersonalVaccinationCalendar1, Person1)*

- ❖ İlk doz için yaratılan kişisel aşı takvimi, Hepatit B aşısı ile ilişkilendirilmektedir.

hasVaccine(PersonalVaccinationCalendar1) ≡ HepatitsB_Vaccine

- ❖ Kişisel aşı takvimi, ilk doz aşılama ile ilişkilendirilmektedir.

isRelatedWith(PersonalVaccinationCalendar1) ≡ PrimaryVaccination

- ❖ Kişisel aşı takvimi için aşı adı, takvim adı ve aşılama dozu tanımlanmaktadır.

vaccineName(PersonalVaccinationCalendar1) ≡ “Hepatits B Vaccine”

calendarName(PersonalVaccinationCalendar1) ≡

“Calendar1ForPerson1”

shotName(PersonalVaccinationCalendar1) ≡ “First Shot”

- ❖ Kişisel aşı takviminde, Hepatit B için önerilen aşı tarihi ve aşılamının yapıldığı tarih tanımlanmaktadır.

proposedDate(PersonalVaccinationCalendar1) ≡ “2015-01-03”

administratedDate(PersonalVaccinationCalendar1) ≡ “2015-01-03”

- ❖ Birey, ikinci doz için yaratılan kişisel aşılama takvimi ile ilişkilendirilmektedir.

hasPersonalVaccinationCalendar(Person1,

PersonalVaccinationCalendar2)≡

isPersonalVaccinationCalendarOf

(PersonalVaccinationCalendar2, Person1)

- ❖ İkinci doz için yaratılan kişisel aşı takvimi, Hepatit B aşısı ile ilişkilendirilmektedir.

hasVaccine(PersonalVaccinationCalendar2) ≡ HepatitsB_Vaccine

- ❖ Kişisel aşı takvimi, ikinci doz aşılama ile ilişkilendirilmektedir.

isRelatedWith(PersonalVaccinationCalendar2) ≡
SecondaryVaccination

- ❖ Kişisel aşı takvimi için aşı adı, takvim adı ve aşılama dozu tanımlanmaktadır.

vaccineName(PersonalVaccinationCalendar2) ≡ “Hepatits B Vaccine”
calendarName(PersonalVaccinationCalendar2) ≡
“Calendar2ForPerson1”
shotName(PersonalVaccinationCalendar2) ≡ “Second Shot”

- ❖ Kişisel aşı takviminde, Hepatit B için önerilen aşı tarihi tanımlanmaktadır.

proposedDate(PersonalVaccinationCalendar2) ≡ “2015-02-03”

- ❖ Birey, üçüncü doz için yaratılan kişisel aşılama takvimi ile ilişkilendirilmektedir.

hasPersonalVaccinationCalendar
(Person1,PersonalVaccinationCalendar3)≡
isPersonalVaccinationCalendarOf
(PersonalVaccinationCalendar3, Person1)

- ❖ Üçüncü doz için yaratılan kişisel aşı takvimi, Hepatit B aşısı ile ilişkilendirilmektedir.

hasVaccine(PersonalVaccinationCalendar3) ≡ HepatitsB_Vaccine

- ❖ Kişisel aşı takvimi, üçüncü doz aşılama ile ilişkilendirilmektedir.

isRelatedWith(PersonalVaccinationCalendar3) ≡ TertiaryVaccination

- ❖ Kişisel aşı takvimi için aşı adı, takvim adı ve aşılama dozu tanımlanmaktadır.

$vaccineName(PersonalVaccinationCalendar3) \equiv \text{“Hepatits B Vaccine”}$

$calendarName(PersonalVaccinationCalendar3) \equiv$

$\text{“Calendar3ForPerson1”}$

$shotName(PersonalVaccinationCalendar3) \equiv \text{“Third Shot”}$

- ❖ Kişisel aşı takviminde, üçüncü doz için önerilen aşı tarihi tanımlanmaktadır.

$proposedDate(PersonalVaccinationCalendar3) \equiv \text{“2015-08-03”}$

- ❖ Birey, rapel doz için yaratılan kişisel aşılama takvimi ile ilişkilendirilmektedir.

$hasPersonalVaccinationCalendar(Person1,$

$PersonalVaccinationCalendar4) \equiv$

$isPersonalVaccinationCalendarOf$

$(PersonalVaccinationCalendar4, Person1)$

- ❖ Rapel doz için yaratılan kişisel aşı takvimi, Hepatit B aşısı ile ilişkilendirilmektedir.

$hasVaccine(PersonalVaccinationCalendar4) \equiv HepatitsB_Vaccine$

- ❖ Kişisel aşı takvimi, rapel doz aşılama ile ilişkilendirilmektedir.

$isRelatedWith(PersonalVaccinationCalendar4) \equiv BoostVaccination$

- ❖ Kişisel aşı takvimi için aşı adı, takvim adı ve aşılama dozu tanımlanmaktadır.

$vaccineName(PersonalVaccinationCalendar4) \equiv \text{“Hepatits B Vaccine”}$

$calendarName(PersonalVaccinationCalendar4) \equiv$

$\text{“Calendar4ForPerson1”}$

$shotName(PersonalVaccinationCalendar4) \equiv \text{“Rapel Shot”}$

- ❖ Kişisel aşı takviminde, rapel doz için önerilen aşı tarihi tanımlanmaktadır.

proposedDate(PersonalVaccinationCalendar4) ≡ "2016-08-03"

Birey, UAO üzerinde bir bulaşıcı hastalık için yaratılan aşı takimindeki aşılama dozlarını, aşılama dozlarının uygulandığı tarihi ve aşılama dozları için önerilen tarihleri sorgulayabilmektedir. Aşağıda verilen sorgunun sonucu Şekil 4.2.2.1’de gösterilmektedir.

```
PREFIX VaccineOntology:
<http://www.vaccineontology.com/VaccineOntology.owl#>
SELECT ?vaccineName ?shotName ?administratedDate ?proposedDate
WHERE{ ?calender VaccineOntology:hasVaccine
VaccineOntology:HepatitisB_Vaccine .
?calender VaccineOntology:vaccineName ?vaccineName;
VaccineOntology:shotName ?shotName .
?calender VaccineOntology:isPersonalVaccineCalenderOf
?person .
?person VaccineOntology:hasTCKimlikNo
"12345678901"^^<http://www.w3.org/2001/XMLSchema#string> .
OPTIONAL{?calender VaccineOntology:administratedDate
?administratedDate . }
OPTIONAL{?calender VaccineOntology:proposedDate ?proposedDate . }}
ORDER BY ASC(?proposedDate)
```

vaccineName	shotName	administratedDate	proposedDate
"Hepatitis B Vaccine"	"First Shot"	"2015-01-03"	"2015-01-03"
"Hepatitis B Vaccine"	"Second Shot"		"2015-02-03"
"Hepatitis B Vaccine"	"Third Shot"		"2015-08-03"
"Hepatitis B Vaccine"	"Rapel Shot"		"2016-08-03"

Şekil 4.2.2.1 Hepatit B aşı takvimi için çalıştırılan SPARQL sorgusunun sonucu

Bir sonraki bölümde, hastaya uygulanmış aşılar ile ilgili geçmiş bilgilerin Köken Modeli’nden yararlanılarak elde edilmesi örneklendirilmektedir.

4.2.3 Köken Modeli ile hastanın aşı geçmişinin izlenmesi örneği

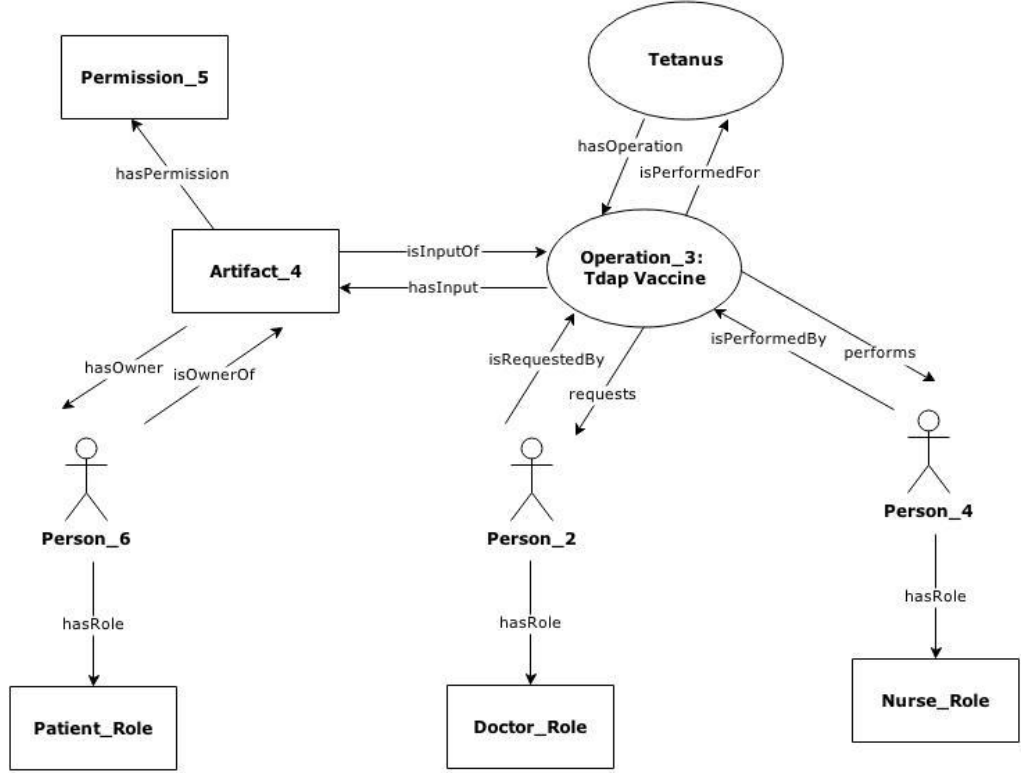
Bu bölümde Gizlilik Farkında Anlamsal Köken Modeli üzerinden aşı geçmişinin izlenmesine yönelik bir örnek anlatılmaktadır. Bulaşıcı hastalıklar etki alanı ile Gizlilik Farkında Anlamsal Köken Modeli’nin eşleştirmesi Bölüm 3.1’de anlatılmakta ve Çizelge 3.1’de gösterilmektedir. Bu eşleştirme dikkate alınarak Tetanoz hastalığı için aşı örneği gerçekleştirimi yapılmaktadır.

Tetanoz hastalığı, Clostridium Tetani adında bir bakterinin neden olduğu bir bulaşıcı hastalıktır. Toprakta, insan ve hayvanların gastrointestinal sistemlerinde bulunan bu bakteri, kirli ve açık yaralar üzerinden insanlara bulaşmaktadır. Hastalığın kuluçka süresi 3-21 gün arasında değişmektedir. Sinirleri felç edip, kas spazmlarına yol açarak ölümcül bir hastalık tablosu çizmektedir. Her yıl dünya genelinde bu hastalık nedeniyle 213.000 ölüm olmaktadır (WHO, 2015).

Tetanoz, aşı ile korunabilir bir hastalıktır. 1968 yılından beri Tetanoz aşısı ülkemizde de uygulanmaktadır. Doğumdan itibaren ilköğretim 1. sınıfa kadar 5 doz tetanoz aşısı yapılmaktadır. Sonraki dönemlerde ise, bağışıklığın devam etmesi için yeniden tekrarlanması gerekmektedir (WHO, 2015).

Tetanoz aşısı; 11 ile 65 yaş arasında yer alan bireylere, birey düşük risk altındaysa 10 yılda bir kez, yüksek risk altında ise 5 yılda bir kez uygulanmalıdır. Tetanoz aşısı, üç ayrı doz halinde kas içine enjekte edilmektedir. İlk doz yapıldıktan bir ay sonra ikinci doz, ikinci doz uygulandıktan altı ay sonra da üçüncü doz yapılmaktadır. Birkaç yılda bir kez de etkinlik artırıcı rapel doz uygulanabilmektedir (WHO, 2015).

Örnekte; hasta, doktorun önerisiyle Tetanoz aşısı olacaktır. Hemşire Lucy, hasta Tom Watson'ın kişisel bilgileri ve işlemin minimum uygulanma yaşı, maksimum uygulanma yaşı gibi bilgileri göz önünde bulundurarak işlem talebi girmektedir. Ardından hemşire tarafından aşı yapılmaktadır. Örneğin şematik gösterimi Şekil 4.2.3.1'de gösterilmektedir. Örnek sadece ilk dozun işlem sıralamasını göstermektedir, hasta ikinci ve üçüncü doz için sağlık kurumuna geldiğinde de yeniden aynı işlemler gerçekleştirilmiştir.



Şekil 4.2.3.1 Tetanoz aşısının uygulanması ve verilerin saklanması

Tetanoz aşısı yapılan hastanın aşı sürecinin *ALCQ DL* dilinde gösterimine aşağıda yer verilmektedir:

- ❖ John Smith için, doktor rolü; Alex Green için, doktor rolü; Lucy Law için, hemşire rolü ve Tom Watson için, hasta rolü tanımlanmaktadır.

$$\begin{aligned} \text{hasRole}(\text{John Smith}) &\equiv \text{Doctor}, & \text{hasRole}(\text{Alex Green}) &\equiv \text{Doctor} \\ \text{hasRole}(\text{Lucy Law}) &\equiv \text{Nurse}, & \text{hasRole}(\text{Tom Watson}) &\equiv \text{Patient} \\ \text{age}(\text{Tom Watson}) &= 36 \end{aligned}$$

- ❖ Tetanoz için Tdap aşısı yapılmaktadır.

$$\text{isPerformedFor}(\text{Tdap Vaccine}, \text{Tetanus}) \equiv \text{hasOperation}(\text{Tetanus}, \text{Tdap})$$

- ❖ Tdap aşısı için işlem tipi korunma olarak belirtilmektedir.

$$\text{type}(\text{Tdap Vaccine}) = \text{precaution}$$

- ❖ Tdap aşısı, 11 ve 65 yaş arasındaki bireylere uygulanmaktadır. Düşük risk grupları için 10 yılda bir kez, yüksek risk grupları için ise 5 yılda bir kez bu aşının uygulanması gerekmektedir.

$minApplicableAge(Tdap Vaccine) = 11$
 $maxApplicableAge(Tdap Vaccine) = 65$
 $periodForLowRisk(Tdap Vaccine) = 10 \text{ years}$
 $periodForHighRisk(Tdap Vaccine) = 5 \text{ years}$

- ❖ John, Tom için Tdap aşısı talebinde bulunmaktadır.

$requests(John Smith, Tdap Vaccine) \equiv isRequestedBy(Tdap Vaccine, John Smith)$

- ❖ Hemşire Lucy tarafındai kişisel bilgiler ve aşı talebi için kayıt oluşturulmaktadır. Oluşturulan kayıt ile Tom arasında sahiplik ilişkisi oluşturulmaktadır. Yeni oluşan bu kayıt için izinler tanımlanmaktadır.

$creator(Artifact4) = Lucy Law$
 $Artifact4 = Tom's PersonalInformation \cup Toms's TdapVaccineRequest$
 $isOwnerOf(Tom Watson, Artifact4) \equiv hasOwner(Artifact4, Tom Watson)$
 $hasPermission5(Artifact4) \equiv (Read, Doctor, Emergencies, 2012-06-25, no \text{ expired date})$

- ❖ Oluşturulan kayıt, Tdap aşı işlemine girdi olarak verilmekte ve Lucy tarafından aşılama işlemi tamamlanmaktadır.

$isInputOf(Artifact4, Tdap Vaccine) \equiv hasInput(Tdap Vaccine, Artifact4)$
 $isPerformedBy(Tdap Vaccine, Lucy Law) \equiv performs(Lucy Law, Tdap Vaccine)$

Önceden aşı yaptırmış olan hasta Tom Watson, iki yıl sonra kolunda büyük bir yara ile hastaneye getirilmiştir. Doktor Alex Green, hastanın Tetanoz aşı geçmişini öğrenmek istemektedir, fakat hasta baygın olduğundan bu bilgiyi kendisinden öğrenememektedir. Doktor, güvenlik ve gizliliği destekleyen model

üzerinde aşağıdaki sorguları çalıştırarak hastanın Tetanoz aşısı köken bilgisine erişilecektir. Aşağıda yer alan sorgunun sonucu Şekil 4.2.3.2’de gösterilmektedir.

```
PREFIX ProvenanceOnto_20140911:
<http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#>
SELECT ?operation ?operationName ?operationPerformedDate ?isPerformedBy
WHERE {?operation ProvenanceOnto_20140911:operationName ?operationName .
      ?operation ProvenanceOnto_20140911:hasInput ?hasInput .
      ?hasInput ProvenanceOnto_20140911:hasOwner ?owner .
      ?owner ProvenanceOnto_20140911:personIdentityNumber
        "345678901212"^^<http://www.w3.org/2001/XMLSchema#string> .
      OPTIONAL{ ?operation ProvenanceOnto_20140911:hasOutput ?hasOutput.
        ?hasOutput ProvenanceOnto_20140911:artifactName
          ?outputName . }
      OPTIONAL{ ?operation ProvenanceOnto_20140911:isPerformedBy
        ?isPerformedBy . }
      ?operation ProvenanceOnto_20140911:isPerformedFor
        ?isPerformedFor .
        ?isPerformedFor ProvenanceOnto_20140911:objectName
          "Tetanus"^^<http://www.w3.org/2001/XMLSchema#string> .
      OPTIONAL{ ?operation
        ProvenanceOnto_20140911:operationPerformedDate
          ?operationPerformedDate . } }
```

operation	operationName	operationPerformedDate
ProvenanceOnto_20140911:Operation_7	"Tdap Vaccine 3"	"2013-01-25"
ProvenanceOnto_20140911:Operation_6	"Tdap Vaccine 2"	"2014-07-25"
ProvenanceOnto_20140911:Operation_3	"Tdap Vaccine"	"2014-06-25"

Şekil 4.2.3.2 Tetanoz aşısı geçmişi için çalıştırılan SPARQL sorgusunun sonucu

Aşı köken bilgisine ek olarak kişinin aşılama takvimine de erişilerek, bir sonraki aşılama tarihleri ile ilgili bilgi sahibi olmak mümkündür. Kişinin aşılama takvimi için aşağıdaki sorgu çalıştırılır. Aşağıda yer alan sorgunun sonucu Şekil 4.2.3.3’te gösterilmektedir.

```
PREFIX VaccineOntology:
<http://www.vaccineontology.com/VaccineOntology.owl#>
SELECT ?person ?vaccineName ?shotName ?administratedDate ?proposedDate
WHERE{?person VaccineOntology:hasTCKimlikNo
      "345678901212"^^<http://www.w3.org/2001/XMLSchema#string> .
      ?person VaccineOntology:hasPersonalVaccineCalender ?calender .
      ?calender VaccineOntology:vaccineName ?vaccineName;
      VaccineOntology:shotName ?shotName .
      OPTIONAL{?calender VaccineOntology:administratedDate
        ?administratedDate . }

      OPTIONAL{?calender VaccineOntology:proposedDate ?proposedDate .} }
```

person	vaccineName	shotName	administratedDate	proposedDate
VaccineOntology:Person_6	"Tetanus Vaccine"	"Third Shot"	"2013-01-25"	"2013-01-25"
VaccineOntology:Person_6	"Tetanus Vaccine"	"Rapel Shot"	"2018-01-25"	"2018-01-25"
VaccineOntology:Person_6	"Tetanus Vaccine"	"Second Shot"	"2012-07-25"	"2012-07-25"
VaccineOntology:Person_6	"Tetanus Vaccine"	"First Shot"	"2012-06-25"	"2012-06-25"

Şekil 4.2.3.3 Tetanoz aşısı takvimi için çalıştırılan SPARQL sorgusunun sonucu

Tetanoz aşısı uygulanma tarihinin üzerinden en az 5 yıl geçmeden tekrar uygulanırsa hasta üzerinde kalıcı hasar bırakabileceği gibi ölümlere de yol açmaktadır. Tez kapsamında geliştirilen bu model üzerinde, aşı kökeni ve aşı takvimi sorgulanabilmektedir. Bu yöntemle, aşı geçmişleri sorgulanarak bilinçsizce aşılama sonucu ortaya çıkabilecek hayati tehlikeler önlenmektedir.

Bir sonraki bölümde, Bölüm 4'ten itibaren detayları verilen SKBS'nin gerçekleştirimi anlatılmaktadır.

4.3 Sağlık Köken Bilgi Sistemi'nin Gerçekleştirimi

Sağlık Köken Bilgi Sistemi üzerinde, kullanıcıların giriş yaptığı andan itibaren arka planda birçok algoritma işletilmektedir. Bu bölüm altında, SKBS'nin farklı katmanlarında çalıştırılan algoritmalar için yazılmış kodların detaylarına yer verilmektedir.

SKBS, verileri Oracle 12c veritabanı üzerinden işlemektedir. Bu nedenle, geliştirilen Bulaşıcı Hastalıklar Ontolojisi, Ulusal Aşı Ontolojisi ve Köken Ontolojisi; Eclipse Java Editör'ü kullanılarak Anlamsal Web desteği olan Oracle 12c veritabanına aktarılmaktadır. Her bir model için `ConnectionProvider` sınıfında yer alan `loadOntology()` metodu (Bkz. Şekil 4.3.1) çalıştırılmaktadır.

```
public void loadOntology () throws SQLException, IOException
{
    InputStream in = FileManager.get().open(ontologyPath);
    Oracle oracle = new Oracle(dbURL, userId, password);
    Model model = ModelOracleSem.createInstance(oracle, modelName);
    model.read(in, null);
    OutputStream os = new FileOutputStream(outputPath);
    model.write(os, "N-TRIPLE");
    os.close();

    model.close();
    oracle.dispose();
}
```

Şekil 4.3.1 `ConnectionProvider` sınıfındaki `loadOntology()` metodu

Modellerin veritabanına aktarılmasından sonra, her bir veri modeli için, listeleme sorguları, ConnectionProvider sınıfında yer alan executeSparqlQuery(queryString) metodu ile çalıştırılırken, yeni nesne yaratma, silme ve güncelleme işlemlerini gerçekleştiren sorgular updateSparqlUpdateQuery(queryString) metodu ile çalıştırılmaktadır. Bu metotların detayları Şekil 4.3.2’de verilmektedir.

```
public static ResultSet executeSparqlQuery(String queryString) throws Exception
{
    Query query = QueryFactory.create(queryString) ;
    QueryExecution qexec = QueryExecutionFactory.create(query, getModel()) ;
    ResultSet resultSet = qexec.execSelect();
    return resultSet;
}

public static void executeSparqlUpdateQuery(String queryString) throws Exception
{
    ModelOracleSem semModel= getModel();
    UpdateAction.parseExecute(queryString, semModel);
    semModel.notifyEvent(GraphEvents.finishRead);
    semModel.commit();
}
```

Şekil 4.3.2 ConnectionProvider sınıfındaki executeSparqlQuery(queryString) ve executeSparqlUpdateQuery(queryString) metotları

Kullanıcı, SKBS’ye giriş yapmak istediğinde girdiği şifre; MD5 Hash koduna çevrilerek, veritabanında MD5 Hash kod ile şifrelenmiş hali tutulan kullanıcı şifreleri ile eşleştirilmektedir (Şekil 4.11). Kullanıcı şifrelerinin, MD5 Hash kod çevirisi MD5HashCodeProvider sınıfında yer alan generateMD5HashCode metodu yardımıyla gerçekleştirilmektedir. Şekil 4.3.3’te generateMD5HashCode metodu gösterilmektedir.

```

public class MD5HashCode {

    public MD5HashCode()
    {

    }

    public static String generateMD5HashCode(String password)
    {
        String generatedPassword = null;
        try {
            // Create MessageDigest instance for MD5
            MessageDigest md = MessageDigest.getInstance("MD5");

            //Update input string in message digest
            md.update(password.getBytes(), 0, password.length());

            //Converts message digest value in base 16 (hex)
            generatedPassword = new BigInteger(1, md.digest()).toString(16);

        }
        catch (NoSuchAlgorithmException e)
        {
            e.printStackTrace();
        }
        System.out.println(generatedPassword);
        return generatedPassword;
    }
}

```

Şekil 4.3.3 MD5HashCodeProvider sınıfı ve generateMD5HashCode metodu

Sağlık personeli veya sistem yöneticisi sisteme giriş yaptıktan sonra kişilerin TC Kimlik numaraları veya isimlerini parametre olarak girerek aratabilmektedir. Sisteme giriş yapan kullanıcının erişim yetkisi varsa ve erişim kısıtlaması yoksa, arama kriterlerine uygun kişi bilgileri, Person listesi halinde UserPage ya da AdministrationPage ekranına dönmektedir.

Sağlık personeli tarafından, kişilerin ismine göre yapılan aramaya ilişkin sıralı diagram Şekil 4.3.5'te verilmektedir. Sağlık personeli, UserPage üzerinden arama butonuna bastıktan sonra, çalıştırılan kod parçası Şekil 4.3.4'te gösterilmektedir. Önce arama kriteri olan isme göre sonuçların döndürülmesi için SubjectBusinessProvider sınıfında yer alan getPersonInfoByFullName(fullname) metodu çalıştırılmaktadır.

```

//Kişi girilen isim ile sorgulanmaktadır.
ArrayList<Person> personList = SubjectBusinessProvider
    .getPersonInfoByFullName(jTextFieldName
        .getText());
if (personList.size() > 0) { //Sorgu sonucunda dönen kişi
    //listesi boş değilse.
    int isPermit = 0, isProhibit = 0;

    for (int i = 0; i < personList.size(); i++) {

        //Kişi listesinde yer alan kişilerin,
        //kişisel bilgilerine erişilebilmesi için
        //sisteme giriş yapmış olan kullanıcının
        //erişim izni olup olmadığı sorgulanmaktadır.
        isPermit = SubjectBusinessProvider
            .isPermitForAccess(
                loginPersonInfo,
                personList
                    .get(i)
                    .getPersonIdentityNumber(),
                jComboBoxPurpose
                    .getSelectedItem()
                    .toString(), loginRoleName);

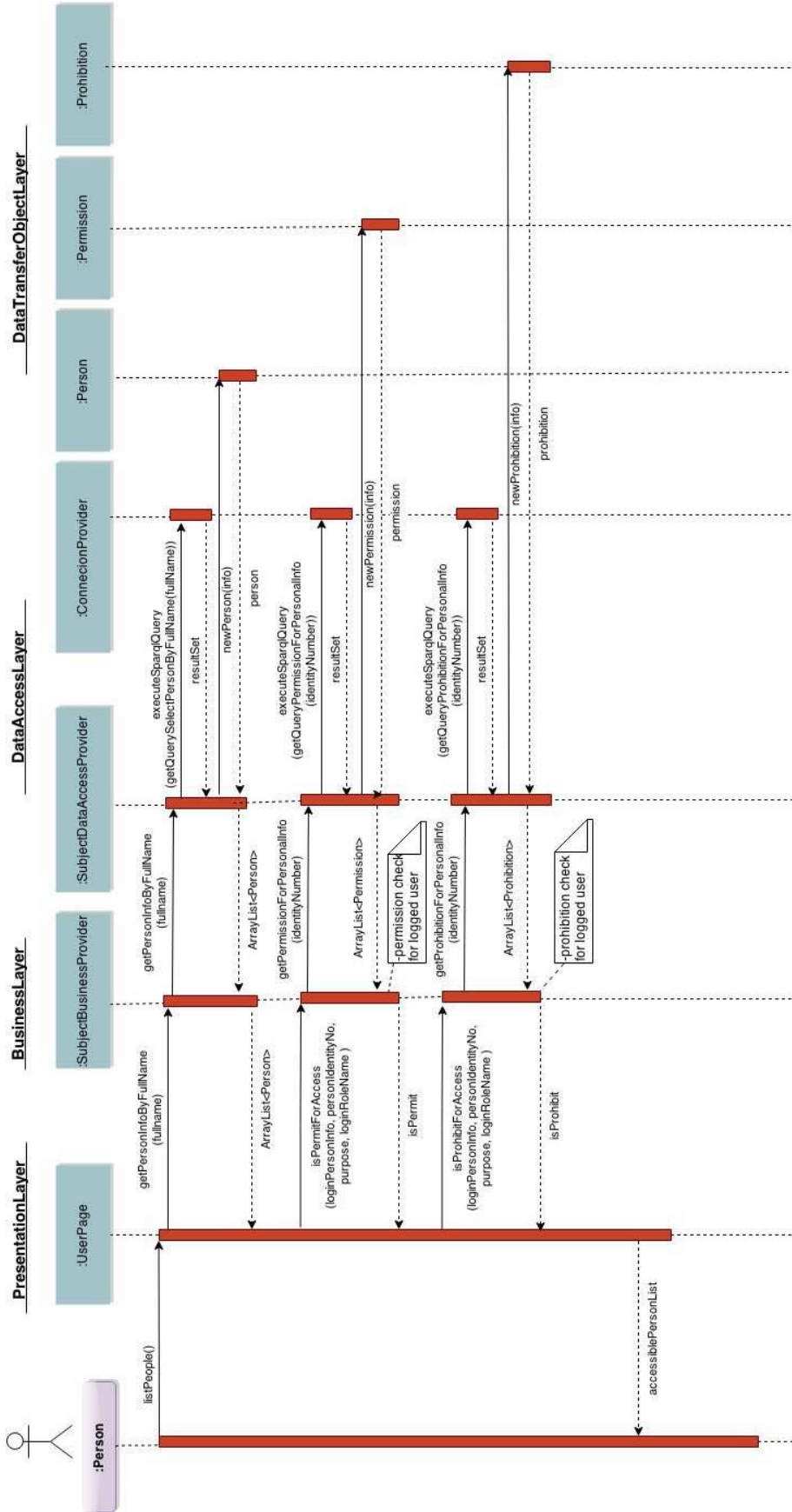
        //Kişi listesinde yer alan kişilerin,
        //kişisel bilgilerine erişimin sınırlandırılması için
        //sisteme giriş yapmış olan kullanıcının
        //erişim yasağı olup olmadığı sorgulanmaktadır.
        isProhibit = SubjectBusinessProvider
            .isProhibitForAccess(
                loginPersonInfo,
                personList
                    .get(i)
                    .getPersonIdentityNumber(),
                jComboBoxPurpose
                    .getSelectedItem()
                    .toString(), loginRoleName);

        //Sisteme giriş yapmış olan kullanıcının
        //erişim izni varsa ve erişim yasağı yoksa
        //sorgu sonucundaki kişisel bilgiler listelenmektedir.
        if (isPermit != -1
            && isProhibit == -1) {
            jScrollPanePersonInfo
                .setVisible(true);
            jTablePersonInfo
                .setVisible(true);
        } else {
            //Sisteme giriş yapmış olan kullanıcının
            //erişim izni yoksa veya erişim yasağı varsa
            //sorgu sonucundaki kişisel bilgiler listeden çıkarılmaktadır.
            personList.remove(i);
        }
    }
    //Sonuç listesinde kalan veriler kullanıcı tarafından görülmektedir.
    if (personList.size() > 0) {
        Object[][] data = new Object[personList
            .size()][4];
        for (int i = 0; i < personList
            .size(); i++) {
            data[i][0] = personList
                .get(i)
                .getPersonIdentityNumber();
            data[i][1] = personList.get(i)
                .getPersonFullName();
            data[i][2] = personList.get(i)
                .getPersonGender();
            data[i][3] = "Edit";
        }

        showData(data);
    } else {
        //Eğer sonuç listesinde herhangi bir veri kalmazsa,
        //ekranda uyarı mesajı gösterilmektedir.
        jLabelWarning
            .setText("You don't have permission to access. " +
                "Please contact your administrator!");
    }
} else {
    //Sorgu sonucunda herhangi bir veri elde edilmezse,
    //ekranda uyarı mesajı gösterilmektedir.
    jLabelWarning.setText("No data found!");
}
}

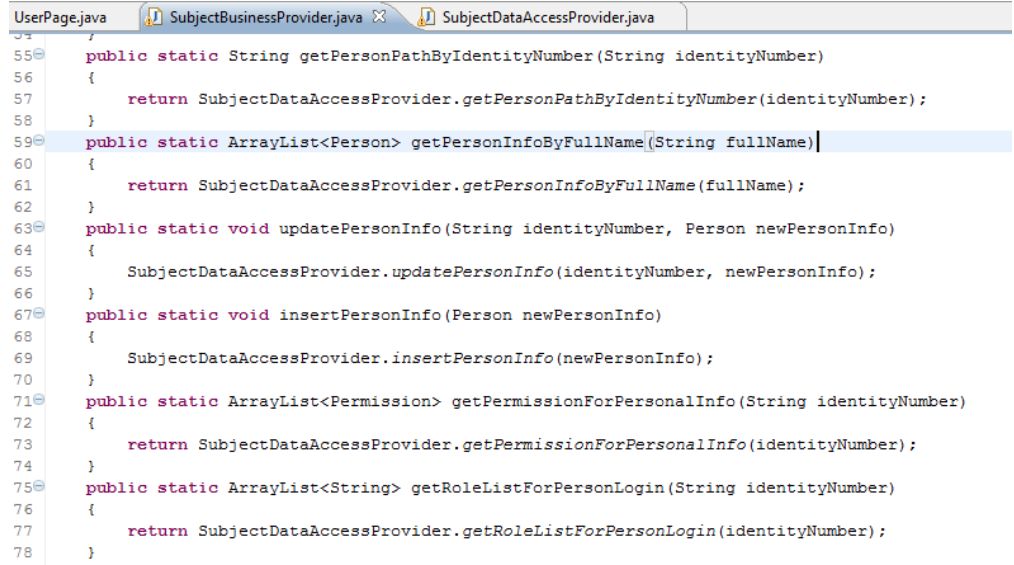
```

Şekil 4.3.4 UserPage sınıfında kişilerin isim kriterine göre sorgulanması için çalıştırılan kod parçası



Şekil 4.3.5 İsim kriterine göre kişilerin listelenmesi ilişkin sıralı diagram

SubjectBusinessProvider sınıfında yer alan getPersonInfoByFullName metodu Şekil 4.3.6’da gösterilmektedir.



```

55 public static String getPersonPathByIdentityNumber(String identityNumber)
56 {
57     return SubjectDataAccessProvider.getPersonPathByIdentityNumber(identityNumber);
58 }
59 public static ArrayList<Person> getPersonInfoByFullName(String fullName)
60 {
61     return SubjectDataAccessProvider.getPersonInfoByFullName(fullName);
62 }
63 public static void updatePersonInfo(String identityNumber, Person newPersonInfo)
64 {
65     SubjectDataAccessProvider.updatePersonInfo(identityNumber, newPersonInfo);
66 }
67 public static void insertPersonInfo(Person newPersonInfo)
68 {
69     SubjectDataAccessProvider.insertPersonInfo(newPersonInfo);
70 }
71 public static ArrayList<Permission> getPermissionForPersonalInfo(String identityNumber)
72 {
73     return SubjectDataAccessProvider.getPermissionForPersonalInfo(identityNumber);
74 }
75 public static ArrayList<String> getRoleListForPersonLogin(String identityNumber)
76 {
77     return SubjectDataAccessProvider.getRoleListForPersonLogin(identityNumber);
78 }

```

Şekil 4.3.6 SubjectBusinessProvider sınıfı ve getPersonInfoByFullName (fullname) metodu

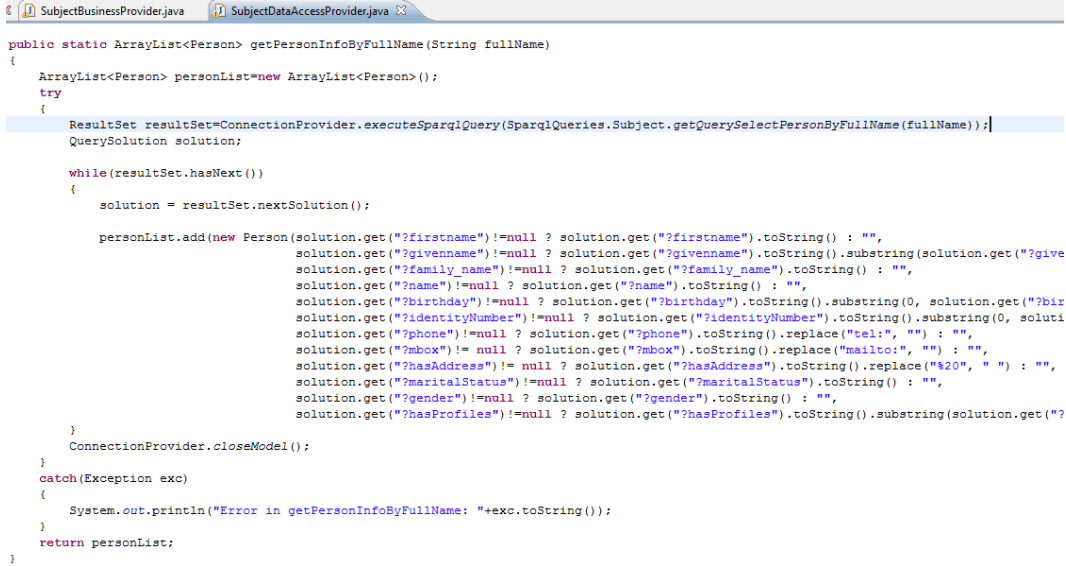
Aşağıda yer alan SPARQL sorgusu, sorguya parametre olarak verilen isim değerine göre Person nesnesine ilişkin detayların getirilmesini sağlamaktadır. SubjectBusinessProvider sınıfındaki getPersonInfoByFullName(fullname) metodu çalıştırıldığında, SubjectDataAccessProvider sınıfındaki getPersonInfoByFullName(fullname) metodu çağırılarak, aşağıdaki SPARQL sorgusu çalıştırılmaktadır:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf:
<http://semantic.ege.edu.tr/ontology/foaf_proje.owl#>
SELECT * WHERE {
    ?person foaf:name ?name;
    foaf:firstname ?firstname;
    foaf:givenname ?givenname;
    foaf:family_name ?family_name;
    foaf:gender ?gender;
    foaf:maritalStatus ?maritalStatus;
    foaf:birthday ?birthday;
    foaf:age ?age;
    foaf:phone ?phone;
    foaf:hasAddress ?hasAddress;
    foaf:hasProfiles ?hasProfiles;
    foaf:mbox ?mbox;
    foaf:identityNumber ?identityNumber;
    FILTER (REGEX (STR(?name), "\"" + name + "\"")) . };

```

Sorgu sonucu olarak, dönen sonuç kümesi çözümlenerek, DataTransferObjectLayer katmanı altında yer alan Person nesnesi yaratılmaktadır. Sonuç kümesindeki tüm kişiler ise, bu Person nesnesi yardımıyla bir liste haline getirilmektedir. Şekil 4.3.7’de sorgu sonuçlarının SubjectDataAccessProvider sınıfı altındaki getPersonInfoByFullName(fullname) metodu içinde nasıl çözümlendiği gösterilmektedir.



```

public static ArrayList<Person> getPersonInfoByFullName(String fullname)
{
    ArrayList<Person> personList=new ArrayList<Person>();
    try
    {
        ResultSet resultSet=ConnectionProvider.executeSparqlQuery(SparqlQueries.Subject.getQuerySelectPersonByFullName(fullname));
        QuerySolution solution;

        while(resultSet.hasNext())
        {
            solution = resultSet.nextSolution();

            personList.add(new Person(solution.get("?firstname")!=null ? solution.get("?firstname").toString() : "",
                solution.get("?givenname")!=null ? solution.get("?givenname").toString().substring(solution.get("?give
                solution.get("?family_name")!=null ? solution.get("?family_name").toString() : "",
                solution.get("?name")!=null ? solution.get("?name").toString() : "",
                solution.get("?birthday")!=null ? solution.get("?birthday").toString().substring(0, solution.get("?bir
                solution.get("?identityNumber")!=null ? solution.get("?identityNumber").toString().substring(0, soluti
                solution.get("?phone")!=null ? solution.get("?phone").toString().replace("tel:", "") : "",
                solution.get("?mailbox")!= null ? solution.get("?mailbox").toString().replace("mailto:", "") : "",
                solution.get("?hasAddress")!= null ? solution.get("?hasAddress").toString().replace("%20", " ") : "",
                solution.get("?maritalStatus")!=null ? solution.get("?maritalStatus").toString() : "",
                solution.get("?gender")!=null ? solution.get("?gender").toString() : "",
                solution.get("?hasProfiles")!=null ? solution.get("?hasProfiles").toString().substring(solution.get("?
            }
        }
        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in getPersonInfoByFullName: "+exc.toString());
    }
    return personList;
}

```

Şekil 4.3.7 SubjectDataAccessProvider sınıfı ve getPersonInfoByFullName(fullname) metodu

TC Kimlik numarası ile ilişkili bir kişinin kişisel bilgilerine, sisteme giriş yapmış olan kullanıcının erişim yetkisi olup olmadığını kontrol etmek için SubjectBusinessProvider sınıfındaki isPermitForAccess(loginPersonInfo, personIdentityNumber, purpose, loginRoleName) metodu çalıştırılmaktadır. Şekil 4.3.8’de isPermitForAccess(loginPersonInfo, personIdentityNumber, purpose, loginRoleName) metodu gösterilmektedir.

```

public static int isPermitForAccess(Person loginPersonInfo, String identityNumber, String purpose, String loginRoleName)
{
    //TC Kimlik numarası ile ilişkili kişinin kişisel bilgileri için tanımlanmış izinler sorgulanmaktadır.
    ArrayList<Permission> permissionList = SubjectDataAccessProvider.getPermissionForPersonalInfo(identityNumber);
    Date date=Calendar.getInstance().getTime();
    //Sorgu sonucu olarak dönen izin listesi üzerinde gezilerek,
    //sisteme giriş yapmış kullanıcının erişim izni olup olmadığı kontrol edilmektedir.
    for(int i=0; i<permissionList.size(); i++)
    {
        //Kullanıcı, izin tanımında yer alan özne ile eşleşiyorsa.
        if(permissionList.get(i).getRelatedSubject().equals(loginPersonInfo.getPersonFullName()))
            //İzin tanımında yer alan erişim hakkı YAZMA ise.
            && permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.WRITE))
        {
            //İzinin geçerli olduğu tarih aralığı yoksa.
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.WRITE;
            //İzin zaman aşımına uğramadıysa.
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyükse ve
            //izin zaman aşımına uğrayacağı tarih yoksa.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.WRITE;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyük ve
            //zaman aşımı tarihinden küçükse.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
        }
        //Kullanıcı, izin tanımında yer alan özne ile eşleşiyorsa.
        else if (permissionList.get(i).getRelatedSubject().equals(loginPersonInfo.getPersonFullName()))
            //İzin tanımında yer alan erişim hakkı OKUMA ise.
            && permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.READ))
        {
            //İzinin geçerli olduğu tarih aralığı yoksa.
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.READ;
            //İzin zaman aşımına uğramadıysa.
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyükse ve
            //izin zaman aşımına uğrayacağı tarih yoksa.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.READ;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyük ve
            //zaman aşımı tarihinden küçükse.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
        }
        //Kullanıcının rolü, izin tanımında yer alan özne ile eşleşiyorsa
        else if (permissionList.get(i).getRelatedSubject().equals(loginRoleName))
            //İzin tanımında yer alan erişim hakkı YAZMA ise.
            && permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.WRITE))
        {
            //İzinin geçerli olduğu tarih aralığı yoksa.
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.WRITE;
            //İzin zaman aşımına uğramadıysa.
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyükse ve
            //izin zaman aşımına uğrayacağı tarih yoksa.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.WRITE;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyük ve
            //zaman aşımı tarihinden küçükse.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
        }
        //Kullanıcının rolü, izin tanımında yer alan özne ile eşleşiyorsa
        else if (permissionList.get(i).getRelatedSubject().equals(loginRoleName))
            //İzin tanımında yer alan erişim hakkı OKUMA ise.
            && permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.READ))
        {
            //İzinin geçerli olduğu tarih aralığı yoksa.
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.READ;
            //İzin zaman aşımına uğramadıysa.
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyükse ve
            //izin zaman aşımına uğrayacağı tarih yoksa.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.READ;
            //Tarih, iznin geçerli olduğu başlangıç tarihinden büyük ve
            //zaman aşımı tarihinden küçükse.
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
        }
    }
    return -1;
}

```

Şekil 4.3.8 SubjectBusinessProvider sınıfında yer alan isPermitForAccess metodu

SubjectBusinessProvider sınıfında yer alan isPermitForAccess(loginPersonInfo, personIdentityNumber, purpose, loginRoleName) metodu içinde, SubjectDataAccessProvider sınıfındaki getPermissionForPersonalInfo(identityNumber) metodu çağırılmaktadır. Bu metot, Şekil 4.3.9'da gösterilmektedir. Metodun amacı, kişisel bilgilere erişim için tanımlanan izinlerin listesini veri modeli üzerinden getirmektir. Aşağıda yer alan sorgu, TC Kimlik numarası ile beraber kişisel bilgilerinin kayıtlı olduğu Artifact nesnesini ve bu Artifact nesnesi için tanımlanmış Permission listesini sonuç olarak döndürmektedir. Bu nedenle, Permission listesini elde etmek için aşağıdaki sorgu çalıştırılmaktadır:

```
PREFIX ProvenanceOnto_20140911:
<http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?s ?Artifact ?purposeName ?relatedSubject ?relatedSubject2 ?right
       ?startedDate ?expiredDate "+"
WHERE {?s ProvenanceOnto_20140911:personIdentityNumber "\"+identityNumber
       +\"\"^<http://www.w3.org/2001/XMLSchema#string>;
       ProvenanceOnto_20140911:isOwnerOf ?Artifact. ?Artifact
       ProvenanceOnto_20140911:hasPermission ?Permission;
       ProvenanceOnto_20140911:artifactValue ?value . FILTER
       (REGEX(STR(?value), "\"+identityNumber+\"\")) .
       ?Permission ProvenanceOnto_20140911:hasRight ?rightObj;
       ProvenanceOnto_20140911:hasRelatedSubject ?relatedSubject;
       ProvenanceOnto_20140911:startedDate ?startedDate;
       ProvenanceOnto_20140911:expiredDate ?expiredDate;
       ProvenanceOnto_20140911:hasPurpose ?Purpose . ?Purpose
       ProvenanceOnto_20140911:purposeName ?purposeName . ?rightObj
       ProvenanceOnto_20140911:rightName ?right . OPTIONAL {
       ?relatedSubject rdf:type ProvenanceOnto_20140911:Role .
       ?relatedSubject ProvenanceOnto_20140911:roleName
       ?relatedSubject2 . } }";
```



```

public static ArrayList<Permission> getPermissionForPersonalInfo(String identityNumber)
{
    ArrayList<Permission> permissionList = new ArrayList<Permission>();
    try
    {
        ResultSet resultSet=ConnectionProvider.executeSparqlQuery(SparqlQueries.Subject.
                                                                    getQueryPermissionForPersonalInfo(identityNumber));

        QuerySolution solution;

        Permission permissionItem;
        String purposeName, relatedSubject, relatedSubject2, right, startedDate, expiredDate;

        while(resultSet.hasNext())
        {
            solution = resultSet.nextSolution();

            purposeName=solution.get("?purposeName").toString();
            relatedSubject = solution.get("?relatedSubject").toString();
            relatedSubject2 = solution.get("?relatedSubject2").toString();
            if(!relatedSubject2.equals(""))
                relatedSubject =relatedSubject2;
            right = solution.get("?right").toString();
            startedDate = solution.get("?startedDate")==null? "" : solution.get("?startedDate").toString();
            expiredDate = solution.get("?expiredDate")==null? "" :solution.get("?expiredDate").toString();
            permissionItem = new Permission(purposeName, relatedSubject, right,
                                           DateTimeBusinessProvider.convertStringToDate(startedDate),
                                           DateTimeBusinessProvider.convertStringToDate(expiredDate));
            permissionList.add(permissionItem);
        }

        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in getPermissionForPersonalInfo: "+exc.toString());
    }
    return permissionList;
}

```

Şekil 4.3.9 SubjectDataAccessProvider sınıfında yer alan getPermissionForPersonalInfo metodu

Sorgu çalıştırıldıktan sonra dönen sonuç kümesinde yer alan değişkenler DataTransferObjectLayer katmanı altında yer alan Permission nesnesi olarak çözümlenmektedir (Şekil 4.3.9). Oluşturulan Permission nesneleri kullanılarak bir liste oluşturulmaktadır. Bu liste sonuç olarak döndürülmektedir. Permission ögeleri oluşturulurken, veri dönüşümlerine ihtiyaç duyulmaktadır. Permission nesnesinde yer alan, startedDate ve expiredDate veri özellikleri String tipinde gelmektedir. Bunların, Date tipine dönüşümü yapılması gerekmektedir. Bu şekilde String'ten Date tipine ve Date veri tipinden String tipine yapılacak dönüşümler için Şekil 4.3.10'da gösterilen DateTimeBusinessProvider sınıfında yer alan metotlar kullanılmaktadır.

```

rPage.java  DateTimeBusinessProvider.java X SubjectBusinessProvider.java SubjectDataAccessProvide
package BusinessLayer;

import java.util.Date;

public class DateTimeBusinessProvider {
    public static String convertDateToString(Date date)
    {
        Format formatter;
        if(date.getHours()==0 && date.getMinutes()==0)
            formatter= new SimpleDateFormat("yyyy-MM-dd");
        else
            formatter= new SimpleDateFormat("yyyy-MM-dd HH:mm");

        return formatter.format(date);
    }
    public static Date convertStringToDate(String date)
    {
        Date result =null;
        try{
            if(date.equals(""))
                return null;
            if(date.contains(":"))
                result= new SimpleDateFormat("yyyy-MM-dd HH:mm").parse(date);
            else
                result= new SimpleDateFormat("yyyy-MM-dd").parse(date);
        }catch(Exception exc)
        {
            System.out.println("Error in convertStringToDate: "+exc.getMessage());
            return null;
        }
        return result;
    }
}

```

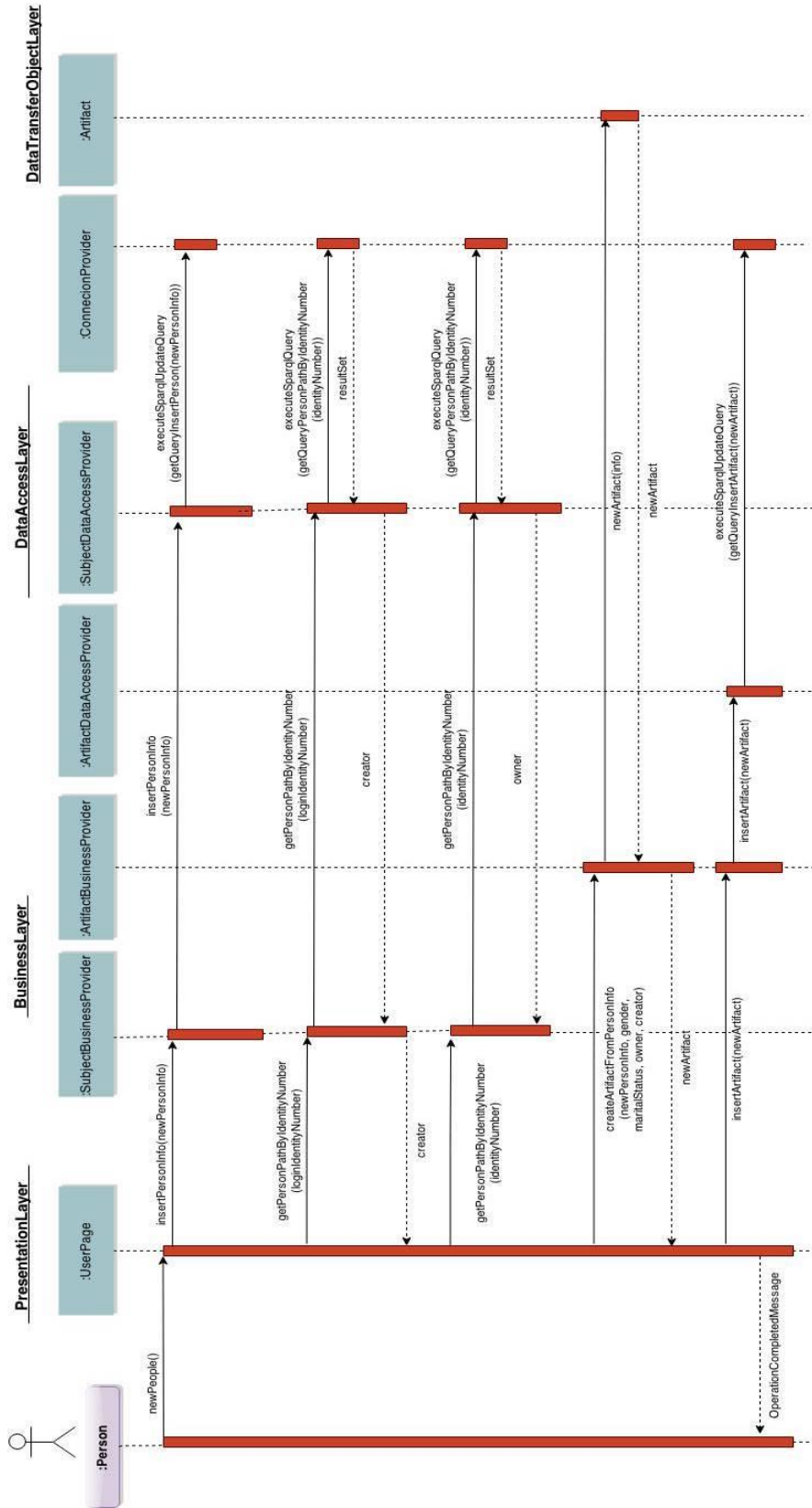
Şekil 4.3.10 DateTimeBusinessProvider sınıfı, convertDateToString ve convertStringToDate metotları

SubjectDataAcccesProvider sınıfında yer alan getPermissionForPersonalInfo(identityNumber) metodu sonucu olarak Permission listesi, SubjectBusinessProvider sınıfında yer alan isPermitForAccess metodu içine dönmektedir. Bu metotta, Şekil 4.3.8’de görüldüğü gibi, sisteme giriş yapmış kullanıcı bilgileri kullanılarak, Permission nesnesinde yer alan RelatedSubject, Right, startedDate ve expiredDate veri özellikleri ile karşılaştırmalar yapılmaktadır. Permission nesnesinde yer alan RelatedSubject özelliğinin, kişinin adı veya rolü ile eşleşip eşleşmediği, startedDate ve expiredDate tarihleri ile tanımlı olan iznin geçerlilik tarih aralığına uygun olup olmadığı kontrol edilmektedir. Bunun sonucu olarak isPermit değeri UserPage ekranına döndürülmektedir. Eğer bu değer -1 ise, kişinin yetkisi bulunmadığı anlamına gelmektedir. isPermit değerinin geri döndürülmesi için yürütülen adımların tümü aynı şekilde isProhibit değerinin geri döndürülmesi için yürütülmektedir. Aynı Permission listesi gibi Prohibition listesi de sorgulanıp, sisteme giriş yapmış kullanıcı için tanımlanmış herhangi bir yasak olup olmadığı kontrol edilmektedir. Bunun için

SubjectBusinessProvider sınıfı altında yer alan isProhibitForAccess(loginPersonInfo, identityNumber, purpose, loginRoleName) metodu çağırılmaktadır. Sonuç olarak dönen isProhibit değeri -1 ise, sisteme giriş yapmış kişi için erişim yasağı tanımlanmamış anlamına gelmektedir. Erişim yasağı tanımlanmamış ve izin tanımlanmış ise bilgi listelenecek kişisel bilgiler listesine eklenmektedir. Eğer liste boş değil ise, ilgili ekranda kullanıcıya gösterilmektedir.

Sağlık personeli veya sistem yöneticisi sisteme giriş yaptıktan sonra kişilerin yeni kişi oluşturmak için Person nesneleri yaratabilmektedir. Bunun için kişinin tüm bilgileri ekrandan girilmelidir. Ardından yeni kişi için yeni bir Person nesnesi, kişisel bilgiler için ise yeni bir Artifact nesnesi yaratılmaktadır.

Bir sağlık çalışanının, sistemde yeni bir Person kişisi yaratmasına ilişkin sıralı diagram Şekil 4.3.11’de gösterilmektedir. Kişi, UserPage üzerinden yeni bir Person eklemek istediğinde, creator olarak kendi bilgileri kullanılmaktadır. Yeni Person için kişisel bilgilerin tutulduğu yeni bir Artifact nesnesi oluşturulmakta ve owner olarak yeni yaratılan Person nesnesi tanımlanmaktadır.



Şekil 4.3.11 Yeni bir Person nesnesinin yaratılma aşamalarına ilişkin sıralı diagram

Şekil 4.3.12’de görüldüğü gibi, UserPage sınıfında, sırayla BusinessLayer katmanında bulunan SubjectBusinessProvider sınıfının içindeki insertPersonInfo(newPersonInfo) ve getPersonPathByIdentityNumber(identityNumber) metotları ile ArtifactBusinessProvider sınıfındaki insertArtifact(newArtifact) metodu çalıştırılmaktadır.

```
SubjectBusinessProvider
    .insertPersonInfo(newPersonInfo);
String creator = SubjectBusinessProvider
    .getPersonPathByIdentityNumber(loginIdentityNumber);
String hasOwner = SubjectBusinessProvider
    .getPersonPathByIdentityNumber(jTextFieldPersonIdentityNumber
        .getText());
Artifact newArtifact = ArtifactBusinessProvider
    .createArtifactFromPersonInfo(
        newPersonInfo, jComboBoxGender
            .getSelectedItem()
            .toString(),
        jComboBoxMaritalStatus
            .getSelectedItem()
            .toString(), hasOwner,
        creator);
ArtifactBusinessProvider
    .insertArtifact(newArtifact);
```

Şekil 4.3.12 Yeni bir Person nesnesinin yaratılma aşamasında UserPage sınıfında çalıştırılan kod parçası

SubjectBusinessProvider sınıfındaki insertPersonInfo(newPersonInfo) metodu, SubjectDataAccessProvider sınıfında yer alan insertPersonInfo(newPersonInfo) metodunu çağırılmaktadır. Bu metot Şekil 4.3.13’te gösterilmektedir.

```
public static void insertPersonInfo(Person newPersonInfo)
{
    try
    {
        ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Subject.getQueryInsertPerson(newPersonInfo));
        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in insertPersonInfo: "+exc.toString());
    }
}
```

Şekil 4.3.13 SubjectDataAccessProvider sınıfındaki insertPersonInfo(newPersonInfo) metodu

SubjectDataAccessProvider sınıfında yer alan insertPersonInfo(newPerson) metodu ise, SparqlQueries sınıfında oluşturulan ve Şekil 4.3.14’te gösterilmiş olan sorguyu çalıştırmaktadır. Herhangi bir hata almadan, işlem tamamlandığında yeni kişi yaratılmış olmaktadır.

```
public static String getQueryInsertPerson(Person newPersonInfo)
{
    String givenName=newPersonInfo.getPersonSecondName().equals("")? newPersonInfo.getPersonFirstName(): newPersonInfo.getPersonFirstName()+"
    return prefix12+NL
    +"INSERT DATA { <http://semantic.ege.edu.tr/ontology/foaf_proje.owl#"+ newPersonInfo.getPersonFullName().replace(" ", "_")+ "_Profile"> " +
        "foaf:name "+ newPersonInfo.getPersonFullName()+" "; " +
        "foaf:firstName "+ newPersonInfo.getPersonFirstName()+" "; " +
        "foaf:givenName "+ givenName+" "; " +
        "foaf:family_name "+ newPersonInfo.getPersonSecondName()+" "; " +
        "foaf:gender "+ newPersonInfo.getPersonGender()+" "; " +
        "foaf:maritalStatus "+ newPersonInfo.getPersonMaritalStatus()+" "; " +
        "foaf:birthday "+newPersonInfo.getPersonBirthdate()+" "; " +
        "foaf:phone "+newPersonInfo.getPersonPhoneNumber()+" "; " +
        "foaf:hasAddress "+newPersonInfo.getPersonAddress()+" "; " +
        "foaf:hasProfiles "+newPersonInfo.getPersonRole()+" "; " +
        "foaf:mbox "+newPersonInfo.getPersonEmail()+" "; " +
        "foaf:identityNumber "+identityNumber+" . } ";
}
```

Şekil 4.3.14 SparqlQueries sınıfında yer alan getQueryInsertPerson

(newPerson) metodu

Kişi yaratıldıktan sonra kişisel bilgileri saklamak ve gizliliğini korumak için Artifact nesnesi yaratılmaktadır. Artifact nesnesi yaratmadan önce nesnenin yaratıcısı ve sahibinin kim olduğu bilgilerinin getirilmesi gerekmektedir. Bunun için SubjectBusinessProvider sınıfındaki getPersonPathBy-IdentityNumber(identityNumber) metodu hem, sisteme giriş yapmış olan kullanıcının TC Kimlik numarası kullanılarak hem de yaratılan yeni kişiye ait TC Kimlik numarası kullanılarak çağırılmaktadır. İlki creator değişkenine atanırken, ikinci sonuç owner değişkenine atanmaktadır. SubjectBusinessProvider altındaki getPersonPathByIdentity-Number(identityNumber) metodu bir sonraki aşama olarak, SubjectDataAccessProvider sınıfı altındaki getPersonPathBy-IdentityNumber(identityNumber) metodunu çağırılmaktadır. Bu metod Şekil 4.3.15’te gösterilmektedir.

```

public static String getPersonPathByIdentityNumber(String identityNumber)
{
    String path=null;
    try
    {
        ResultSet resultSet=ConnectionProvider.executeSparqlQuery(
            SparqlQueries.Subject.getQueryPersonPathByIdentityNumber(identityNumber));
        QuerySolution solution;

        while(resultSet.hasNext())
        {
            solution = resultSet.nextSolution();
            path = solution.get("?s").toString();
            path = path.substring(path.indexOf("#") + 1,
                path.length());
            path = "ProvenanceOnto_20140911:" + path;
        }
        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in getPersonInfoByIdentityNumber: "+exc.toString());
    }
    return path;
}

```

Şekil 4.3.15 SubjectDataAccessProvider sınıfı altındaki getPersonPathBy-IdentityNumber(identityNumber) metodu

Şekil 4.3.15'te gösterilen metot, kişinin TC Kimlik numarasına göre modeldeki yolunu geri döndüren ve aşağıda yer alan SPARQL sorgusunu çalıştırmaktadır:

```

PREFIX ProvenanceOnto_20140911:
<http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#>
SELECT ?s
WHERE {?s ProvenanceOnto_20140911:personIdentityNumber \"
    + identityNumber
    + \"\"^^<http://www.w3.org/2001/XMLSchema#string> . }\"

```

Sorgu çalıştırıldıktan sonra elde edilen sonuç kümesi Şekil 4.3.14'te görüldüğü gibi çözümlenip, TC Kimlik numarası gönderilen kişiye ait modelde yer alan Person nesnesinin yolunu verecek şekilde bir değişken elde edilmektedir. Bu yöntemle elde edilen creator ve owner nesneleri yeni Artifact nesnesi yaratmak için parametre olarak kullanılmaktadır. Bir sonraki aşama olarak ArtifactBusinessProvider sınıfında yer alan createArtifactFromPersonInfo(person, gender, marital-Status, hasOwner, creator) metodu çağırılarak bir Artifact nesnesi yaratılmaktadır. Bu metot Şekil 4.3.16'da gösterilmektedir.

```

public static Artifact createArtifactFromPersonInfo(Person person, String gender, String maritalStatus, String hasOwner, String creator)
{
    String value="personFullName: " + person.getPersonFullName() + NL +
        "personPhoneNumber: "+person.getPersonPhoneNumber()+NL +
        "personBirthdate: " + person.getPersonBirthdate() + NL +
        "personEmail: " + person.getPersonEmail() + NL +
        "personAddress: " + person.getPersonAddress() + NL +
        "personIdentityNumber: "+person.getPersonPhoneNuber() + NL +
        "personMaritalStatus: "+maritalStatus + NL +
        "personGender: "+ gender + NL;
    Random generator = new Random();
    int i = generator.nextInt(10000000); // String.valueOf(i)
    String name = "Artifact_"+String.valueOf(i);

    Artifact artifact= new Artifact(name, value, hasOwner, Calendar.getInstance().getTime(), creator);

    return artifact;
}

```

Şekil 4.3.16 ArtifactBusinessProvider sınıfında yer alan createArtifactFromPersonInfo(person, gender, maritalStatus, hasOwner, creator) metodu

Yeni Artifact nesnesi yaratıldıktan sonra, ArtifactBusinessProvider sınıfında yer alan insertArtifact(newArtifact) metodu çağırılmaktadır. Bu metot ise, ArtifactDataProvider'da yer alan Şekil 4.3.17'de gösterilmiş olan insertArtifact(newArtifact) metodunu çağırılmaktadır.

```

public static void insertArtifact(Artifact artifact)
{
    try
    {
        ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Artifact.getQueryInsertArtifact(artifact));
        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in insertArtifact: "+exc.toString());
    }
}

```

Şekil 4.3.17 ArtifactDataAccessProvider sınıfında yer alan insertArtifact(newArtifact) metodu

Yeni Artifact nesnesinin modele eklenmesi için SparqlQueries sınıfında yer alan getQueryInsertArtifact(artifact) metodu ile elde edilen sorgu çalıştırılmaktadır.

```

public static String getQueryInsertArtifact(DataTransferObjectLayer.Artifact artifact)
{
    Random generator = new Random();
    int i = generator.nextInt(10000000);
    String createdDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(artifact.getCreated());
    return prefix10
    + NL + prefix8 + NL + prefix1 + NL + prefix4 + NL
    + "INSERT DATA { <http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#Artifact_"+i+"> " +
        "rdf:type ProvenanceOnto_20140911:Artifact; " +
        "ProvenanceOnto_20140911:artifactName \"\"+ artifact.getArtifactName() + "\"\"<http://www.w3.org/2001/XMLSchema#string>; " +
        "ProvenanceOnto_20140911:artifactValue \"\"+artifact.getArtifactValue()+\"\"<http://www.w3.org/2001/XMLSchema#string>; " +
        "terms:created \"\"+createdDate+\"\"<http://www.w3.org/2001/XMLSchema#string>; " +
        "dc:creator \"\"+artifact.getCreator()+\"\" ; " +
        "ProvenanceOnto_20140911:hasOwner \"\"+ artifact.getHasOwner()+\"\" . } ";
}

```

Şekil 4.3.18 SparqlQueries sınıfında yer alan getQueryInsertArtifact(artifact) metodu

Yeni Person ve Artifact nesneleri başarılı bir şekilde modele eklendiyse, kullanıcıya işlemin başarılı bir şekilde tamamlandığını belirten bir mesaj gösterilmektedir.

Varolan bir Person nesnesinin sağlık çalışanı ya da yönetici tarafından güncellenmesi durumunda ise, Person nesnesi önce SubjectBusinessLayer sonra SubjectDataAccessLayer sınıfında yer alan updatePersonInfo(identityNumber, newPersonInfo) metotları çağrılmaktadır. DataAccessLayer katmanında daha sonra Şekil 4.3.19’da gösterilmiş olan SPARQL sorgusu çalıştırılmaktadır.

```
public static String getQueryUpdatePersonByIdentityNumber(String identityNumber, Person newPersonInfo)
{
    String givenName=newPersonInfo.getPersonSecondName().equals("")? newPersonInfo.getPersonFirstName(): newPersonInfo.getP
    return prefix12 + NL
    + "DELETE { ?s foaf:name ?name ; " + NL +
        "foaf:firstName ?firstName ; " + NL +
        "foaf:givenName ?givenName ; " + NL +
        "foaf:family_name ?family_name ; " + NL +
        "foaf:gender ?gender ; " + NL +
        "foaf:maritalStatus ?maritalStatus ; " + NL +
        "foaf:birthday ?birthday ; " + NL +
        "foaf:phone ?phone ; " + NL +
        "foaf:hasAddress ?hasAddress ; " + NL +
        "foaf:mbox ?mbox ; " + NL +
        "foaf:hasProfiles ?hasProfiles . } " + NL
    + "INSERT { ?s foaf:name "+ newPersonInfo.getPersonFirstName()+ " ; " + NL +
        "foaf:firstName "+ newPersonInfo.getPersonFirstName()+ " ; " +
        "foaf:givenName "+ givenName+ " ; " +
        "foaf:family_name "+ newPersonInfo.getPersonSecondName()+ " ; " +
        "foaf:gender "+ newPersonInfo.getPersonGender()+ " ; " +
        "foaf:maritalStatus "+ newPersonInfo.getPersonMaritalStatus()+ " ; " +
        "foaf:birthday "+newPersonInfo.getPersonBirthdate()+ " ; " +
        "foaf:phone 'tel:'+newPersonInfo.getPersonPhoneNumber()+ " ; " +
        "foaf:hasAddress "+newPersonInfo.getPersonAddress()+ " ; " +
        "foaf:mbox "+newPersonInfo.getPersonEmail()+ " ; " +
        "foaf:hasProfiles "+newPersonInfo.getPersonRole()+ " . } " + NL
    + "WHERE { ?s foaf:identityNumber \""+identityNumber+"\"^^<http://www.w3.org/2001/XMLSchema#string> ; " +
        "foaf:name ?name ; " + NL +
        "foaf:firstName ?firstName ; " + NL +
        "foaf:givenName ?givenName ; " + NL +
        "foaf:family_name ?family_name ; " + NL +
        "foaf:gender ?gender ; " + NL +
        "foaf:maritalStatus ?maritalStatus ; " + NL +
        "foaf:birthday ?birthday ; " + NL +
        "foaf:phone ?phone ; " + NL +
        "foaf:hasAddress ?hasAddress ; " + NL +
        "foaf:mbox ?mbox ; " + NL +
        "foaf:hasProfiles ?hasProfiles . } " + NL +
        " ; "
}
```

Şekil 4.3.19 SparqlQueries sınıfında yer alan getQueryUpdatePersonBy-IdentityNumber(identityNumber) metodu

Eski Person nesnesi güncellendikten sonra yeni kişisel bilgiler kullanılarak, yine yeni bir Artifact nesnesi yaratılmaktadır. Yeni kişi ekleme aşamalarından farklı olarak kişisel bilgiler için yaratılmış olan bir önceki Artifact nesnesi ile versiyon ilişkisi kurması sağlanmaktadır. Bunun için, UserPage sınıfında ArtifactBusinessProvider sınıfındaki insertHasVersionAndIsVersionOfProperties (newArtifact.getArtifactName(), artifact.getArtifactName()) metodu

(Bkz. Şekil 4.3.20), ardından `ArtifactDataAccessProvider` sınıfındaki aynı adı taşıyan metot çağrılmaktadır.

```
public static void insertHasVersionAndIsVersionOfProperties(String artifactName, String artifactNameForPriVersion)
{
    try
    {
        ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Artifact.
                                                    getQueryUpdateArtifactHasVersion(artifactName, artifactNameForPriVersion));
        ConnectionProvider.closeModel();

        ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Artifact.
                                                    getQueryUpdateArtifactIsVersionOf(artifactNameForPriVersion, artifactName));
        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in insertHasVersionAndIsVersionOfProperties: "+exc.toString());
    }
}
```

Şekil 4.3.20 `ArtifactDataAccessProvider` sınıfında yer alan `insertHasVersionAndIsVersionOfProperties(newArtifact.getArtifactName(), artifact.getArtifactName())` metodu

`ArtifactDataAccessProvider` sınıfındaki `insertHasVersionAndIsVersionOfProperties` metodunda çalıştırılan SPARQL sorguları Şekil 4.3.21’de yer almaktadır.

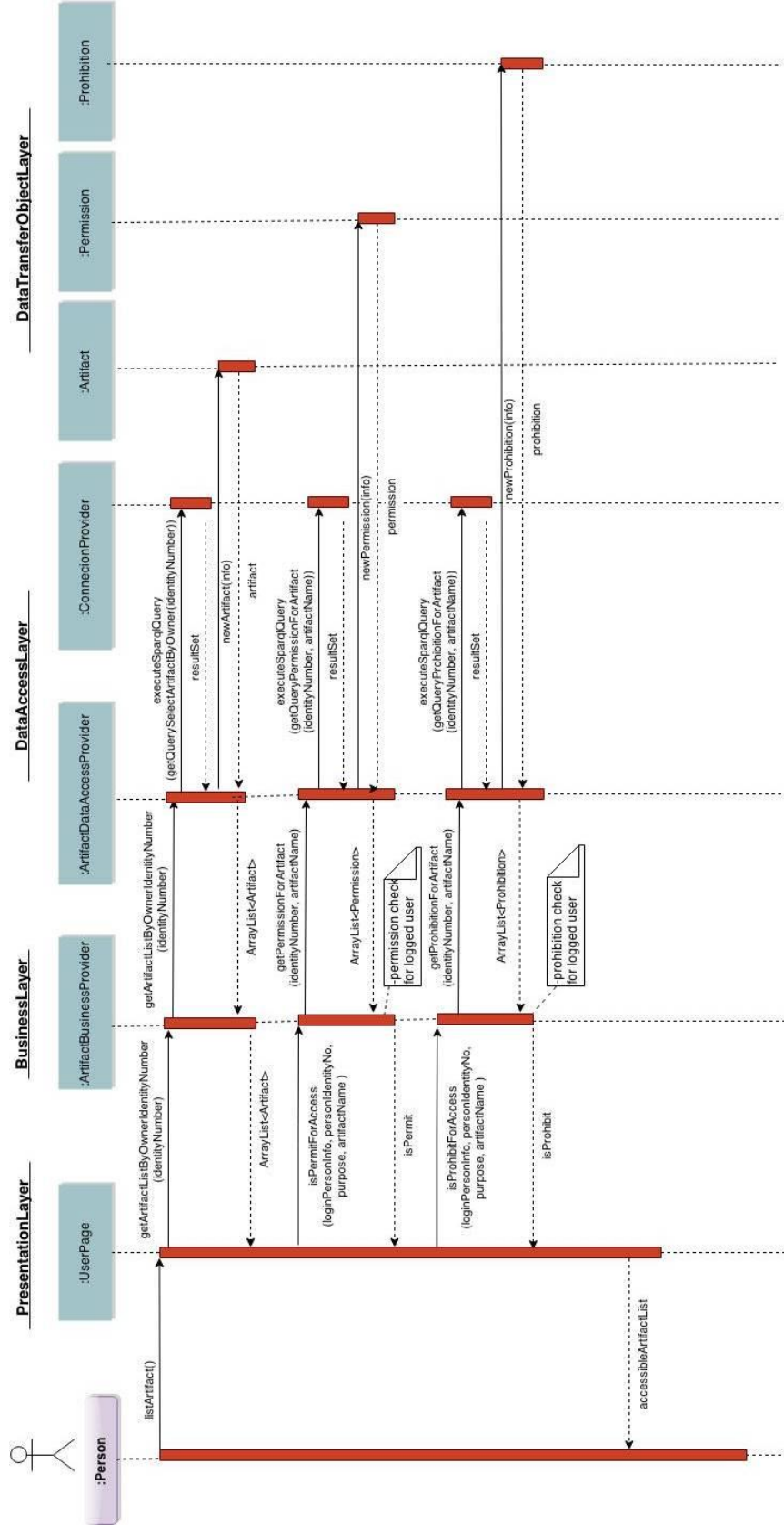
```
public static String getQueryUpdateArtifactHasVersion(String artifactName, String artifactNameForPriVersion)
{
    return prefix10
    + NL + prefix1 + NL + prefix3 + NL + prefix2 + NL
    + "DELETE {?s terms:hasVersion ?hasVersion;" +
    "      terms:priorVersion ?priorVersion . } " + NL
    + "INSERT {?s terms:hasVersion ProvenanceOnto_20140911:"+ artifactName + ";" +
    "      owl:priorVersion ProvenanceOnto_20140911:"+ artifactNameForPriVersion + " . } " + NL
    + "WHERE {?s "
    + "ProvenanceOnto_20140911:artifactName \""
    + artifactName
    + "\"^^<http://www.w3.org/2001/XMLSchema#string> . }";
}

public static String getQueryUpdateArtifactIsVersionOf(String artifactName, String artifactNameForNewVersion)
{
    return prefix10
    + NL + prefix1 + NL
    + "DELETE {?s terms:isVersionOf ?isVersionOf . } " + NL
    + "INSERT {?s terms:isVersionOf ProvenanceOnto_20140911:"+ artifactNameForNewVersion + " . } " + NL
    + "WHERE {?s "
    + "ProvenanceOnto_20140911:artifactName \""
    + artifactName
    + "\"^^<http://www.w3.org/2001/XMLSchema#string> . }";
}
```

Şekil 4.3.21 `SparqlQueries` sınıfında yer alan `getQueryUpdateArtifactHasVersion` ve `getQueryUpdateArtifactIsVersionOf` metotları

Eski `Person` nesnesi güncellenme işlemi; yeni `Artifact` nesnesi yaratılıp, eski `Artifact` nesnesi ile versiyon olarak ilişkilendirildikten sonra tamamlanmış olmaktadır. İşlemler yürütülürken herhangi bir hata alınmadıysa, kullanıcıya işlemin başarılı bir şekilde tamamlandığını belirten bir mesaj gösterilmektedir.

Sağlık çalışanı ve yönetici, Artifact nesnesi yaratabilmekte ve varolan Artifact nesnelerini sorgulayabilmektedir. Artifact nesnesinin yaratma aşamaları yukarıda anlatıldığından tekrar anlatılmayacaktır. Artifact nesneleri, nesnenin sahibinin ismi ya da TC Kimlik numarasına göre listelenebilmektedir.



Şekil 4.3.22 Sağlık çalışanı tarafından Artifact nesnesinin TC Kimlik numarası kullanılarak sorgulanmasına ilişkin sıralı diagram

Sağlık personeli tarafından, Artifact sahibinin TC Kimlik numarası kullanılarak yapılan Artifact sorgulama işlemine ilişkin sıralı diagram Şekil 4.3.22’de verilmektedir. Sağlık personeli, UserPage üzerinden arama butonuna bastıktan sonra, çalıştırılan kod parçası Şekil 4.3.23’te gösterilmektedir. Önce arama kriteri olan isme göre sonuçların döndürülmesi için ArtifactBusinessProvider sınıfındaki getArtifactListByOwner-IdentityNumber (identityNumber) metodu çalıştırılmaktadır. Ardından ArtifactDataAccessProvider sınıfında yer alan aynı isimdeki metod çalıştırılmaktadır.

```

ArrayList<Artifact> artifactList = ArtifactBusinessProvider
    .getArtifactListByOwnerIdentityNumber(getJTextFieldIdentityNumberArtifact()
        .getText());
if (artifactList.size() > 0) {
    int isPermit = 0, isProhibit = 0;
    for (int i = 0; i < artifactList.size(); i++) {

        isPermit = ArtifactBusinessProvider
            .isPermitForAccess(
                loginPersonInfo,
                getJTextFieldIdentityNumberArtifact()
                    .getText(),
                jComboBoxPurposeArtifact1
                    .getSelectedItem()
                    .toString(), artifactList.get(i).getArtifactName());
        isProhibit = ArtifactBusinessProvider
            .isProhibitForAccess(
                loginPersonInfo,
                getJTextFieldIdentityNumberArtifact()
                    .getText(),
                jComboBoxPurposeArtifact1
                    .getSelectedItem()
                    .toString(), artifactList.get(i).getArtifactName());

        if (isPermit != -1
            && isProhibit == -1) {
            jScrollPaneArtifactInfo
                .setVisible(true);
            jTableArtifactInfo
                .setVisible(true);
            artifactList.get(i).setPermissionValue(isPermit);
            artifactList.get(i).setProhibitionValue(isProhibit);
        } else {
            artifactList.remove(i);
            i=i-1;
        }
    }
    if (artifactList.size() > 0) {
        Object[][] data = new Object[artifactList
            .size()][4];
        for (int i = 0; i < artifactList
            .size(); i++) {
            data[i][0] = artifactList
                .get(i)
                .getArtifactName();
            data[i][1] = artifactList
                .get(i).getHasOwner();
            data[i][2] = "Edit";
            data[i][3] = String.valueOf(artifactList.get(i).getPermissionValue());
        }

        showDataArtifact(data);
    } else {
        jLabelWarningArtifact
            .setText("You don't have permission to access. Please contact your administrator!");
    }
} else {
    jLabelWarningArtifact
        .setText("No data found! ");
}

```

Şekil 4.3.23 UserPage sınıfında Artifact sahibinin TC Kimlik numarasına göre Artifact nesnelerinin listelenmesi için çalıştırılan kod parçası

ArtifactDataAccessProvider sınıfında yer alan getArtifactListByOwnerIdentityNumber(identityNumber) metodu SparqlQueries sınıfında yer alan getQuerySelectArtifactByOwner(identityNumber) metodu ile elde ettiği sorguyu çalıştırmaktadır. Eğer Artifact listeleme işlemi Artifact sahibinin ismine göre yapılırsa, SparqlQueries sınıfında yer alan getQuerySelectArtifactByOwnerName(owner) metodu ile elde edilen sorgu çalıştırılmaktadır. SparqlQueries sınıfında yer alan bu metotlar Şekil 4.3.24'te gösterilmektedir.

```
public static String getQuerySelectArtifactByOwner(String ownerIdentityNumber)
{
    return prefix10
    + NL + prefix8 + NL + prefix1 + NL + prefix4 + NL
    + "SELECT ?s ?artifactName ?hasOwner " +
    "WHERE {?s " +
    "    ProvenanceOnto_20140911:artifactName ?artifactName; " +
    "    ProvenanceOnto_20140911:hasOwner ?hasOwner . " +
    "?hasOwner ProvenanceOnto_20140911:personIdentityNumber \"\"+ ownerIdentityNumber+
    "\"\"^<http://www.w3.org/2001/XMLSchema#string> . }";
}

public static String getQuerySelectArtifactByOwnerName(String ownerName)
{
    return prefix10
    + NL + prefix8 + NL + prefix1 + NL + prefix4 + NL
    + "SELECT ?s ?artifactName ?hasOwner ?ownerIdentityNumber ?personFullName " +
    "WHERE {?s " +
    "    ProvenanceOnto_20140911:artifactName ?artifactName; " +
    "    ProvenanceOnto_20140911:hasOwner ?hasOwner . " +
    "?hasOwner ProvenanceOnto_20140911:personFullName ?personFullName; " +
    "    ProvenanceOnto_20140911:personIdentityNumber ?ownerIdentityNumber . " +
    "FILTER (REGEX(STR(?personFullName),\"\"+ownerName+\"\")) .}";
}
```

Şekil 4.3.24 SparqlQueries sınıfında yer alan getQuerySelectArtifactBy-Owner(identityNumber) ve getQuerySelectArtifactByOwnerName(owner) metotları

Sorgular çalıştırdıktan sonra sonuç kümesi çözümlenerek Artifact nesneleri yardımıyla, bir Artifact listesi oluşturulur. Bu liste önce ArtifactBusinessProvider sınıfına, sonra da UserPage sınıfına ulaşır. UserPage sınıfında dönen Artifact listesinde yer alan her Artifact için sisteme giriş yapmış kullanıcının erişim izinleri ve kısıtlamaları kontrol edilmektedir.

Sisteme giriş yapmış olan kullanıcının Artifact nesnesi üzerinde yetkisi olup olmadığı kontrol etmek için ArtifactBusinessProvider sınıfında yer alan isPermitForAccess(loginPersonInfo, identityNumber, purpose, artifactName) metodu çağrılmaktadır. Şekil 4.3.25'te ArtifactBusinessProvider sınıfında yer alan isPermitForAccess(loginPersonInfo, identityNumber,

purpose, artifactName) metodu gösterilmektedir. Bu metot içinde ArtifactDataAccessProvider sınıfındaki getPermissionForArtifact(identityNumber, artifactName) metodu ile bir Artifact için tanımlanmış olan Permission listesi getirilmektedir. Permission listesi için getPermissionForArtifact(identityNumber, artifactName) metodu içinde aşağıdaki sorgu çalıştırılmaktadır:

```
PREFIX ProvenanceOnto_20140911:
< http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#>
PREFIX rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?s ?Artifact ?purposeName ?relatedSubject ?relatedSubject2 ?right
       ?startedDate ?expiredDate ?personIdentityNumber
WHERE { ?s ProvenanceOnto_20140911:personIdentityNumber \"\"+identityNumber
        +\"\"^<http://www.w3.org/2001/XMLSchema#string>;
        ProvenanceOnto_20140911:isOwnerOf ?Artifact.
        ?Artifact ProvenanceOnto_20140911:artifactName\"\"+
        artifactName+\"\"^<http://www.w3.org/2001/XMLSchema#string>;
        ProvenanceOnto_20140911:hasPermission ?Permission;
        ProvenanceOnto_20140911:artifactValue ?value .
        ?Permission ProvenanceOnto_20140911:hasRight ?rightObj;
        ProvenanceOnto_20140911:hasRelatedSubject ?relatedSubject;
        ProvenanceOnto_20140911:hasPurpose ?Purpose .
        ?Purpose ProvenanceOnto_20140911:purposeName ?purposeName .
        ?rightObj ProvenanceOnto_20140911:rightName ?right .
        OPTIONAL { ?relatedSubject rdf:type ProvenanceOnto_20140911:Role .
        ?relatedSubject ProvenanceOnto_20140911:roleName
        ?relatedSubject2 . }
        OPTIONAL{?Permission ProvenanceOnto_20140911:startedDate
        ?startedDate . }
        OPTIONAL{?Permission ProvenanceOnto_20140911:expiredDate
        ?expiredDate . }
        OPTIONAL{?relatedSubject rdf:type ProvenanceOnto_20140911:Person .
        ?relatedSubject ProvenanceOnto_20140911:personIdentityNumber
        ?personIdentityNumber . } }
```

Sorgu sonucunda elde edilen Permission listesi, Artifact-BusinessProvider sınıfında yer alan isPermitForAccess metodu içine dönmeaktadır. Bu metotta, Şekil 4.3.25'te görüldüğü gibi, sisteme giriş yapmış kullanıcı bilgileri kullanılarak, Permission nesnesinde yer alan RelatedSubject, Right, startedDate ve expiredDate veri özellikleri ile karşılaştırmalar yapılmaktadır. Permission nesnesinde yer alan RelatedSubject özelliğinin, kişinin adı veya rolü ile eşleşip eşleşmediği, startedDate ve expiredDate tarihleri ile tanımlı olan iznin geçerlilik tarih aralığına uygun olup olmadığı kontrol edilmektedir. Bunun sonucu olarak isPermit değeri UserPage ekranına döndürülür. Eğer bu değer -1 ise, kişinin

yetkisi bulunmadığı anlamına gelmektedir. `isPermit` değerinin geri döndürülmesi için yürütülen adımların tümü aynı şekilde `isProhibit` değerinin geri döndürülmesi için yürütülmektedir. Aynı `Permission` listesi gibi `Prohibition` listesi de sorgulanıp, sisteme giriş yapmış kullanıcı için tanımlanmış herhangi bir yasak olup olmadığı kontrol edilmektedir. Bunun için `ArtifactBusinessProvider` sınıfı altında yer alan `isProhibitForAccess(loginPersonInfo, identityNumber, purpose, artifactName)` metodu çağırılmaktadır. Sonuç olarak dönen `isProhibit` değeri -1 ise, sisteme giriş yapmış kişi için erişim yasağı tanımlanmamış anlamına gelir. Erişim yasağı tanımlanmamış ve izin tanımlanmış ise bilgi listelenecek `Artifact` nesneleri listesine eklenmektedir. Eğer liste boş değil ise, ilgili ekranda kullanıcıya gösterilmektedir.

```

public static int isPermitForAccess(Person loginPersonInfo, String identityNumber, String purpose, String artifactName)
{
    ArrayList<Permission> permissionList = ArtifactDataAccessProvider.getPermissionForArtifact(identityNumber, artifactName);
    Date date=Calendar.getInstance().getTime();
    for(int i=0; i<permissionList.size(); i++)
    {
        if(permissionList.get(i).getRelatedSubject().equals(loginPersonInfo.getPersonFullName()) &&
            permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.WRITE))
        {
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.WRITE;
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.WRITE;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
        }
        else if (permissionList.get(i).getRelatedSubject().equals(loginPersonInfo.getPersonFullName()) &&
            permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.READ))
        {
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.READ;
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.READ;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
        }
        else if (permissionList.get(i).getRelatedSubject().equals(loginPersonInfo.getPersonRole()) &&
            permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.WRITE))
        {
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.WRITE;
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.WRITE;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.WRITE;
        }
        else if (permissionList.get(i).getRelatedSubject().equals(loginPersonInfo.getPersonRole()) &&
            permissionList.get(i).getRight().toUpperCase().equals(Variables.RightString.READ))
        {
            if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()==null)
                return Variables.Right.READ;
            else if(permissionList.get(i).getStartDate()==null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()==null
                && permissionList.get(i).getStartDate().compareTo(date)<=0)
                return Variables.Right.READ;
            else if(permissionList.get(i).getStartDate()!=null && permissionList.get(i).getExpiredDate()!=null
                && permissionList.get(i).getStartDate().compareTo(date)<=0
                && permissionList.get(i).getExpiredDate().compareTo(date)>=0)
                return Variables.Right.READ;
        }
    }
    return -1;
}

```

Şekil 4.3.25 ArtifactBusinessProvider sınıfı altında yer alan isPermitForAccess(loginPersonInfo, identityNumber, purpose, artifactName)

Sağlık çalışanları ve yönetici Operation nesnelerini sorgulayabilmekte, yeni Operation nesneleri yaratabilmekte ve varolan Operation nesneleri üzerinde operationName, performedFor ve operationType veri özelliklerini kullanarak sorgulama yapabilmektedir.

Sağlık personeli, UserPage üzerinden Operation nesnesi için tüm gerekli verileri girdikten sonra butona bastığında, yeni bir Operation nesnesi yaratmak için çalıştırılan kod parçası Şekil 4.3.26’da gösterilmektedir. Önce ekranda yer alan bilgiler biraraya getirilerek Operation nesnesi yaratılmaktadır. Ardından Operation nesnesinin creator sahasını doldurmak için sisteme giriş yapmış kullanıcının TC Kimlik bilgisi yardımıyla Person nesnesinin yolu elde edilmektedir. setCreator(creator) metodu ile Operation nesnesine kaydedilir. Yaratılma tarihi ise, o anki sistem zamanı alınarak, DateTimeBusinessProvider sınıfında yer alan convertDateTo-String(created) yardımıyla String tipine çevirilmekte, Operation nesnesine kaydedilmektedir.

```
Operation newOperation=getOperationInfoFromScreen();
Date created=Calendar.getInstance().getTime();
String createdDate=DateTimeBusinessProvider.convertDateToString(created);
getJTextFieldOperationCreatedDate().setText(createdDate);
newOperation.setCreated(createdDate);
String creatorPath=SubjectBusinessProvider.getPersonPathByIdentityNumber(loginIdentityNumber);
newOperation.setCreator(creatorPath);
OperationBusinessProvider.insertOperationInfo(newOperation);
if(getJComboBoxVaccineList().getSelectedIndex()>0)
{
    String vaccinePath=VaccineBusinessProvider.getVaccinePathByName
        (getJComboBoxVaccineList().getSelectedItem().toString());
    OperationBusinessProvider.insertOperationSameAsVaccine
        (getJTextFieldOperationName1().getText(), vaccinePath);

    String identityNumber= OperationBusinessProvider.
        getOperationInputOwnerIdentityNumberByOperationName
        (getJTextFieldOperationName1().getText());
    if(!getJTextFieldOperationPerformedDate().getText().equals("") && !identityNumber.equals(""))
    {
        VaccineBusinessProvider.insertPersonalVaccinationCalenderList
            (identityNumber, getJComboBoxVaccineList().getSelectedItem().toString(),
            getJTextFieldOperationPerformedDate().getText());
    }
}
```

Şekil 4.3.26 UserPage sınıfında yer alan Operation nesnesinin eklenmesi için çalıştırılan kod parçası

Yeni Operation nesnesini modele eklemek için, OperationBusinessProvider sınıfında yer alan, insertOperationInfo(newOperation) sınıfı çağırılmaktadır. Bu sınıfta OperationDataAccessProvider sınıfındaki aynı ismi taşıyan metodu çağırılmaktadır. Bu metot Şekil 4.3.27’de gösterilmektedir.

```

public static void insertOperationInfo(Operation newOperationInfo)
{
    try
    {
        String query=SparqlQueries.Operation.getQueryInsertOperation(newOperationInfo);
        ConnectionProvider.executeSparqlUpdateQuery(query);
        ConnectionProvider.closeModel();
    }
    catch(Exception exc)
    {
        System.out.println("Error in insertOperationInfo: "+exc.toString());
    }
}

```

Şekil 4.3.27 OperationDataAccessProvider sınıfındaki insertOperationInfo(newOperation) metodu

Şekil 4.3.27’de gösterilmiş olan OperationDataAccessProvider sınıfındaki insertOperationInfo(newOperation) metodu içinde SparqlQueries sınıfında yer alan getQueryInsertOperation(newOperation) sınıfı ile elde edilen SPARQL sorgusu getirilmektedir. Bu sorguyu oluşturmak için Operation sınıfındaki değişkenlere herhangi veri atanıp atanmadığı kontrol edilip sorgu değişkenine ekleme yapılmaktadır. Bu metot Şekil 4.3.28’de detaylı olarak gösterilmektedir.

```

public static String getQueryInsertOperation(DataTransferObjectLayer.Operation newOperationInfo)
{
    String queryParameters="";
    if(!newOperationInfo.getHasInputPath().equals(""))
        queryParameters+="ProvenanceOnto_20140911:hasInput "+ newOperationInfo.getHasInputPath()+ " ";
    if(!newOperationInfo.getHasOutputPath().equals(""))
        queryParameters+="ProvenanceOnto_20140911:hasOutput "+ newOperationInfo.getHasOutputPath()+ " ";
    if(!newOperationInfo.getIsPerformedBy().equals(""))
        queryParameters+="ProvenanceOnto_20140911:isPerformedBy "+ newOperationInfo.getIsPerformedBy()+ " ";
    if(!newOperationInfo.getIsRequestedBy().equals(""))
        queryParameters+="ProvenanceOnto_20140911:isRequestedBy "+ newOperationInfo.getIsRequestedBy()+ " ";
    if(!newOperationInfo.getIsPerformedFor().equals(""))
        queryParameters+="ProvenanceOnto_20140911:isPerformedFor "+ newOperationInfo.getIsPerformedFor()+ " ";
    if(newOperationInfo.getOperationPerformedDate() != null)
    {
        String performedDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(newOperationInfo.getOperationPerformedDate());
        queryParameters+="ProvenanceOnto_20140911:operationPerformedDate \""+performedDate + "\"^^<http://www.w3.org/2001/XMLSchema#string>";
    }
    if(newOperationInfo.getOperationMinApplicableAge() != null && !newOperationInfo.getOperationMinApplicableAge().equals(""))
        queryParameters+="ProvenanceOnto_20140911:operationMinApplicableAge \""+ newOperationInfo.getOperationMinApplicableAge()+ "\"^^<http://www.w3.org/2001/XMLSchema#string>";
    if(newOperationInfo.getOperationMaxApplicableAge() != null && !newOperationInfo.getOperationMaxApplicableAge().equals(""))
        queryParameters+="ProvenanceOnto_20140911:operationMaxApplicableAge \""+ newOperationInfo.getOperationMaxApplicableAge()+ "\"^^<http://www.w3.org/2001/XMLSchema#string>";
    if(newOperationInfo.getOperationPeriodForLowRisk() != null && !newOperationInfo.getOperationPeriodForLowRisk().equals(""))
        queryParameters+="ProvenanceOnto_20140911:operationPeriodForLowRisk \""+ newOperationInfo.getOperationPeriodForLowRisk()+ "\"^^<http://www.w3.org/2001/XMLSchema#string>";
    if(newOperationInfo.getOperationPeriodForHighRisk() != null && !newOperationInfo.getOperationPeriodForHighRisk().equals(""))
        queryParameters+="ProvenanceOnto_20140911:operationPeriodForHighRisk \""+ newOperationInfo.getOperationPeriodForHighRisk()+ "\"^^<http://www.w3.org/2001/XMLSchema#string>";

    return prefix10
    + NL + prefix4 + NL + prefix1 + NL + prefix8+NL
    + "INSERT DATA { <http://www.semanticweb.org/ontologies/ProvenanceOnto_20140911.owl#"+ newOperationInfo.getOperationName().replaceAll(" ", "_") + "> type ProvenanceOnto_20140911:Operation; "+
    "ProvenanceOnto_20140911:operationName \""+ newOperationInfo.getOperationName()+ "\"^^<http://www.w3.org/2001/XMLSchema#string>"+
    queryParameters+
    "dc:type "+newOperationInfo.getType()+ " ";
    "dc:creator "+ newOperationInfo.getCreator()+ " ";
    "terms:created "+ newOperationInfo.getCreated()+ " . } ";
}

```

Şekil 4.3.28 SparqlQueries sınıfındaki getQueryInsertOperation(newOperation)

Yeni Operation nesnesi eklendikten sonra ekranda Operation nesnesinin bir aşıya eş olup olmadığını belirten alanın dolu olup olmadığı kontrol edilmektedir. Eğer bu alanda bir aşı seçilmişse, önce aşı için Vaccine nesnesinin UAO üzerinde yer alan yolu getirilmektedir. Vaccine nesnesi modelde kayıtlı olan Operation nesnesinin sameAs özelliği olarak güncellenmektedir. Eğer Operation nesnesinin PerformedDate özelliği doldurulduysa yani işlem gerçekleştirildiyse, kişi için aşı takvimi eklenir ya da aşı takvimi varsa güncellenmektedir. Bunun için önce VaccineBusinessProvider sınıfında yer alan insertPersonalVaccinationCalenderList (identity-Number, vaccineName, performedDate) metodu çağırılmaktadır. Bu metot ise, VaccineDataAccessProvider sınıfında yer alan aynı isimdeki metodu çağırılmaktadır. Bu metot Şekil 4.3.29’da gösterilmiştir. Bu metotta önce, metoda girdi olarak verilen aşı yolu ve kişinin TC Kimlik numarası kullanılarak, kişinin aşı takvimi sorgulanmaktadır. Eğer aşı takvimi bulunuyorsa, bu takvimde yer alan aşı güncellenmektedir. Güncellenecek aşıyı bulmak için, aşı takvimi listesi üzerinde dolaşmaktadır.

Aşı takvimi ögesinde yer alan aşı uygulama tarihi boş ise ve

- önerilen aşı tarihi ile aynı ise ya da
- önerilen aşı tarihini geçmiş ise ve bir sonraki aşının önerilen tarihinden önce ise ya da
- aşı takvimindeki son aşı ise ve önerilen tarihi geçmiş ise güncellenir.

```

public static void insertPersonalVaccinationCalenderList(String identityNumber, String vaccineName, String performedDate)
{
    try
    {
        String vaccinePath= VaccineBusinessProvider.getVaccinePathByName(vaccineName);
        ArrayList<PersonalVaccinationCalender> calenderList=getPersonalVaccinationCalenderList(vaccinePath, identityNumber);
        int control=0;

        for(int i=0; i< calenderList.size(); i++)
        {
            if(calenderList.get(i).getProposedDate().equals(performedDate) && calenderList.get(i).getAdministratedDate().equals(""))
            {
                //update administratedDate
                updatePersonalVaccinationCalender(calenderList.get(i).getCalendarName(), performedDate);
                control=1;
                break;
            }
            if(calenderList.get(i).getProposedDate().equals(performedDate) && !calenderList.get(i).getAdministratedDate().equals(""))
            {
                control=1;
                break;
            }
            if(calenderList.get(i).getProposedDate().compareTo(performedDate)<0 && calenderList.get(i).getAdministratedDate().equals(""))
            {
                if((i+1<calenderList.size() && calenderList.get(i+1).getProposedDate().compareTo(performedDate) > 0) ||
                (i+1==calenderList.size() && calenderList.get(i).getAdministratedDate().equals(""))))
                {
                    //update administratedDate
                    updatePersonalVaccinationCalender(calenderList.get(i).getCalendarName(), performedDate);
                }
                control=1;
                break;
            }
        }

        if(control==0)
        {
            Date performed=DateTimeBusinessProvider.convertStringToDate(performedDate);
            Date secondShotDate, thirdShotDate, rapelShotDate;
            Calendar c;
            ArrayList<VaccineApplicationDose> vaccineApplicationDoseList=getVaccineApplicationDoseListByVaccineName(vaccineName);
            //(String vaccineName, String shotName, String administratedDate, String proposedDate)
            PersonalVaccinationCalender calender=new PersonalVaccinationCalender("",vaccineName, "First Shot", performedDate, performedDate);
            calender.setIsRelatedWith("VaccineOntology:PrimaryVaccination");
            calender.setHasVaccine(vaccineApplicationDoseList.get(0).getVaccinePath());
            insertPersonalVaccinationCalender(calender, identityNumber);

            secondShotDate=performed;
            thirdShotDate=performed;
            rapelShotDate=performed;
            for(int i=0; i<vaccineApplicationDoseList.size(); i++)
            {
                if(vaccineApplicationDoseList.get(i).getDoseName().equals("Second Shot"))
                {
                    if(!vaccineApplicationDoseList.get(i).getPeriodYear().equals(""))
                    {
                        c = Calendar.getInstance();
                        c.setTime(secondShotDate);
                        c.add(Calendar.YEAR,Integer.valueOf(vaccineApplicationDoseList.get(i).getPeriodYear()));
                        secondShotDate = c.getTime();
                    }
                    if(!vaccineApplicationDoseList.get(i).getPeriodMonth().equals(""))
                    {
                        c = Calendar.getInstance();
                        c.setTime(secondShotDate);
                        c.add(Calendar.MONTH,Integer.valueOf(vaccineApplicationDoseList.get(i).getPeriodMonth()));
                        secondShotDate = c.getTime();
                    }
                    if(!vaccineApplicationDoseList.get(i).getPeriodDay().equals(""))
                    {
                        c = Calendar.getInstance();
                        c.setTime(secondShotDate);
                        c.add(Calendar.DATE,Integer.valueOf(vaccineApplicationDoseList.get(i).getPeriodDay()));
                        secondShotDate = c.getTime();
                    }
                    calender=new PersonalVaccinationCalender("", vaccineName, vaccineApplicationDoseList.get(i).getDoseName(), "",
                    DateTimeBusinessProvider.convertDateToString(secondShotDate));
                    calender.setIsRelatedWith("VaccineOntology:SecondaryVaccination");
                    calender.setHasVaccine(vaccineApplicationDoseList.get(i).getVaccinePath());
                    insertPersonalVaccinationCalender(calender, identityNumber);
                    break;
                }
            }
        }
    }
}

```

Şekil 4.3.29 VaccineDataAccessProvider sınıfında yer alan insertPersonalVaccinationCalenderList(identityNumber, vaccineName, performedDate) metodu

Eğer kişinin herhangi bir aşı takvimi kaydı yoksa, aşı ismi yardımıyla aşının uygulanma dozları bir liste halinde alınır ve bu listede yer alan her bir uygulama dozu için bir PersonalVaccinationCalender ögesi eklenmektedir.

Sağlık çalışanı ve yönetici, Object nesnesi yaratabilmekte ve varolan Object nesnelerini sorgulayabilmektedir. Object nesnesi, nesnenin ismi ve BHO'nde yer alan bulaşıcı hastalığın yolu kullanılarak, yaratılmakta ve güncellenmektedir. Diğer nesnelerde olduğu gibi, bu işlemler içinde BussinessLayer ve DataAccessLayer katmanlarındaki nesneye özel yaratılmış sınıflar kullanılmaktadır. ObjectBusinessProvider ve ObjectDataAccessProvider sınıfları yardımıyla, sorgulama ve güncelleme işlemleri gerçekleştirilmektedir.

Sistem, AccessHistory nesnesini herhangi bir Artifact nesnesi sorgulanıp görüntülediğinde otomatik olarak yaratmaktadır. Artifact listesinde yer alan herhangi bir öğenin detayı görüntülenmek istediğinde, Artifact nesnesinin hangi Right nesnesi ile görüntülediği kontrol edilerek, Artifact nesnesi için bir AccessHistory nesnesi eklenir. Şekil 4.3.30'da bu işlem için çalıştırılan kod parçası verilmektedir.

```
Artifact artifact = ArtifactBusinessProvider
    .getArtifactFromPersonInfo(identityNumber, person
        .getGraphPath());

int isPermit = SubjectBusinessProvider.isPermitForAccess(
    loginPersonInfo, identityNumber, jComboBoxPurpose
        .getSelectedItem().toString(), loginRoleName);
String permission = "";
if (isPermit == Variables.Right.WRITE) {
    jButtonSubmit.setVisible(true);
    jTextFieldFirstName.setEnabled(true);
    jTextFieldSecondName.setEnabled(true);
    jTextFieldSurname.setEnabled(true);
    jTextFieldPhoneNumber.setEnabled(true);
    jTextFieldAddress.setEnabled(true);
    jTextFieldEmail.setEnabled(true);
    jTextFieldBirthDate.setEnabled(true);
    jComboBoxGender.setEnabled(true);
    jComboBoxMaritalStatus.setEnabled(true);
    jComboBoxRole.setEnabled(true);
    jTextFieldPersonIdentityNumber.setEnabled(true);
    permission = "WRITE";
    // AccessHistory is inserted
    AccessHistoryBusinessProvider.insertAccessHistory(loginPersonInfo,
        permission, getJComboBoxPurpose().getSelectedItem()
            .toString(), artifact.getArtifactName());
}
```

Şekil 4.3.30 UserPage sınıfında Artifact detayı görüntülenirken işletilen kod parçası

Hasta, kendine ait Artifact nesnelerini sorgulayabilmekte ve Permission, Prohibition nesnelerini kullanarak izin ve kısıtlamaları

belirlemektedir. Permission ve Prohibition nesnelerini yaratıp, varolan nesnelerini güncelleyebilmektedir. Permission nesnesi yaratmak için PermissionDataAccessProvider sınıfında yer alan insertPermissionInfo (permission, purposeList, rightList, personList, roleList) metodu Şekil 4.3.31’de gösterilmektedir. Bu metotta Permission nesnesi eklendikten sonra Purpose listesi, Right listesi, Person listesi ve Role listesi de eklenen Permission nesnesine kaydedilmektedir.

```
public static void insertPermissionInfo(DataTransferObjectLayer.Permission permission, ArrayList<String> purposeList,
                                     ArrayList<String> rightList, ArrayList<String> personList, ArrayList<String> roleList)
{
    try
    {
        String query=SparqlQueries.Permission.getQueryInsertPermission(permission);
        ConnectionProvider.executeSparqlUpdateQuery(query);
        ConnectionProvider.closeModel();

        String purposePath="";
        for(int i=0; i<purposeList.size(); i++)
        {
            purposePath=PurposeDataAccessProvider.getPurposePath(purposeList.get(i));
            ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Permission.getQueryInsertPermissionPurpose(permission.getPermissionName(),
                                                                                                     purposePath));
            ConnectionProvider.closeModel();
        }

        String rightPath="";
        for(int i=0; i<rightList.size(); i++)
        {
            rightPath=RightDataAccessProvider.getRightPath(rightList.get(i));
            ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Permission.getQueryInsertPermissionRight(permission.getPermissionName(),
                                                                                                     rightPath));
            ConnectionProvider.closeModel();
        }

        String personPath="";
        for(int i=0; i<personList.size(); i++)
        {
            personPath=SubjectDataAccessProvider.getPersonPathByIdentityNumber(personList.get(i));
            ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Permission.getQueryInsertPermissionRelatedSubjectPerson(permission.getPermissionName(),
                                                                                                     personPath));
            ConnectionProvider.closeModel();
        }

        String rolePath="";
        for(int i=0; i<roleList.size(); i++)
        {
            rolePath=RoleDataAccessProvider.getRolePath(roleList.get(i));
            ConnectionProvider.executeSparqlUpdateQuery(SparqlQueries.Permission.getQueryInsertPermissionRelatedSubjectRole(permission.getPermissionName(),
                                                                                                     rolePath));
            ConnectionProvider.closeModel();
        }
    }
    catch(Exception exc)
    {
        System.out.println("Error in insertPermissionInfo: "+exc.toString());
    }
}
```

Şekil 4.3.31 PermissionDataAccessProvider sınıfında yer alan insertPermissionInfo(permission, purposeList, rightList, personList, roleList) metodu

Sistem yöneticisi, Artifact nesnelerinin ismini parametre olarak AdministrationPage sayfasından girerek ya da herhangi bir girdi olmadan güvenlik açıklarını sorgulayabilmektedir. Güvenlik açıklarının saptanması için kayıtlı olan AccessHistory nesnelerinden yararlanılmaktadır. AdministrationPage sınıfından, AccessHistoryBusinessProvider sınıfındaki Şekil 4.3.32’de gösterilmekte olan searchForSecurityViolationsByArtifactName(artifactName) metodu çağırılmaktadır.

Bu metotta Artifact nesnesinin Permission ve Prohibition nesneleri ile tanımlanmış izin ve yasaklarına uygun olup olmadığını kontrol etmek için, AccessHistory nesnesindeki bilgiler Permission ve Prohibition nesneleri ile uyumluluğu kontrol edilmektedir. Eğer tutarsızlık bulunuyorsa, saptanan güvenlik ihlalleri bir listeye eklenir ve metot çıktısı olarak AdministrationPage sınıfına döndürülmektedir.

```
public static ArrayList<SecurityViolationDetail> searchForSecurityViolationsByArtifactName(String artifactName)
{
    ArrayList<SecurityViolationDetail> securityViolationDetail=new ArrayList<SecurityViolationDetail>();
    SecurityViolationDetail item;
    Artifact artifact = ArtifactDataAccessProvider.getArtifactInfoByArtifactName(artifactName);
    int hasPermission=0;
    int hasProhibition=0;
    ArrayList<AccessHistory> accessHistoryList;
    ArrayList<DataTransferObjectLayer.Permission> permissionList;
    ArrayList<DataTransferObjectLayer.Prohibition> prohibitionList;
    ArrayList<String> roleList;
    String [] accessDetailStr;
    accessHistoryList= AccessHistoryDataAccessProvider.getAccessHistoryByArtifactNameAndIdentityNumber(artifact.getArtifactName(), artifact.getHasOwner());
    permissionList= ArtifactDataAccessProvider.getPermissionForArtifact(artifact.getHasOwner(), artifact.getArtifactName());
    prohibitionList= ArtifactDataAccessProvider.getProhibitionForArtifact(artifact.getHasOwner(), artifact.getArtifactName());
    roleList= new ArrayList<String> ();
    for(int j=0; j<accessHistoryList.size(); j++)
    {
        hasPermission=0;
        hasProhibition=0;
        roleList= SubjectBusinessProvider.getRoleListForPersonLogin(accessHistoryList.get(j).getWasAccessedBy());
        accessDetailStr=accessHistoryList.get(j).getAccessDetail().split(",");
        for(int x=0; x<permissionList.size(); x++)
        {
            if(((permissionList.get(x).getRelatedSubjectIdentityNo()!=null && accessHistoryList.get(j).getWasAccessedBy().equals(permissionList.get(x).getRelatedSubjectIdentityNo()) &&
            roleList.contains(permissionList.get(x).getRelatedSubjectIdentityNo()) &&
            (accessDetailStr[0].trim().equals(permissionList.get(x).getPurpose())) &&
            (accessDetailStr[2].trim().equals(permissionList.get(x).getRight())) &&
            (permissionList.get(x).getStartDate()!=null && permissionList.get(x).getStartDate().compareTo(DateTimeBusinessProvider.getCurrentDate())>0) &&
            (permissionList.get(x).getExpiredDate()!=null && permissionList.get(x).getExpiredDate().compareTo(DateTimeBusinessProvider.getCurrentDate())<0))
            {
                hasPermission=1;
                break;
            }
        }
        for(int x=0; x<prohibitionList.size(); x++)
        {
            if(((prohibitionList.get(x).getRelatedSubjectIdentityNo()!=null && accessHistoryList.get(j).getWasAccessedBy().equals(prohibitionList.get(x).getRelatedSubjectIdentityNo()) &&
            roleList.contains(prohibitionList.get(x).getRelatedSubjectIdentityNo()) &&
            (accessDetailStr[0].trim().equals(prohibitionList.get(x).getPurpose())) &&
            (accessDetailStr[2].trim().equals(prohibitionList.get(x).getRight())) &&
            (prohibitionList.get(x).getStartDate()!=null && prohibitionList.get(x).getStartDate().compareTo(DateTimeBusinessProvider.getCurrentDate())>0) &&
            (prohibitionList.get(x).getExpiredDate()!=null && prohibitionList.get(x).getExpiredDate().compareTo(DateTimeBusinessProvider.getCurrentDate())<0))
            {
                hasProhibition=1;
                break;
            }
        }
        if(hasPermission==0 || hasProhibition==1)
        {
            item= new SecurityViolationDetail(artifact, accessHistoryList.get(j));
            securityViolationDetail.add(item);
        }
    }
}

return securityViolationDetail;
}
```

Şekil 4.3.32 AccessHistoryBusinessProvider sınıfında yer alan searchForSecurityViolationsByArtifactName(artifactName)

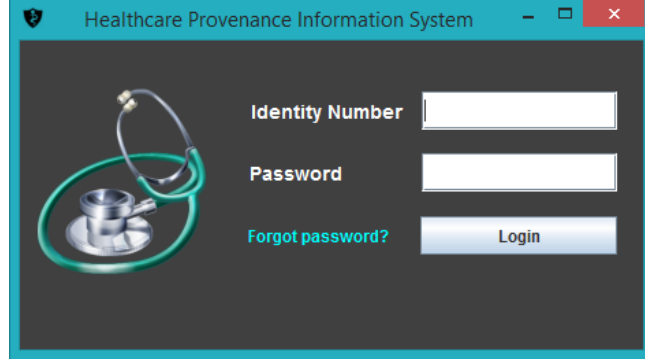
Kullanıcılar tarafından yapılan diğer işlemler yukarıda bahsedilen işlemlerle benzer olduğundan detaylandırılmamaktadır.

Sonraki iki bölümde, Sağlık Köken Bilgi Sistemi'nin bilgisayar ve mobil kullanıcı arayüzlerine ilişkin detaylara yer verilmektedir.

4.4 SKBS Kullanıcı Arayüzleri

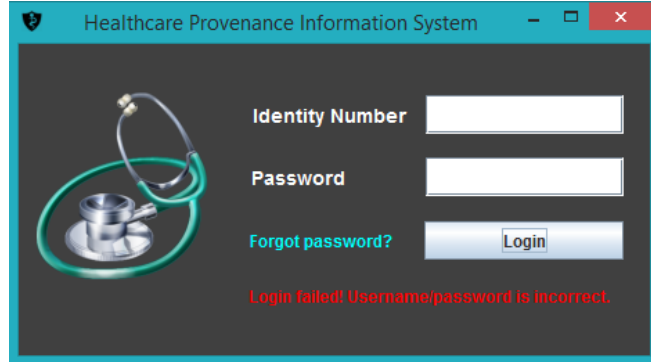
Bu bölüm altında, tez kapsamında geliştirilen Sağlık Köken Bilgi Sistemi kullanıcı arayüzlerine yer verilmektedir.

Kullanıcı, sisteme giriş yapmak için Şekil 4.4.1’de gösterilen giriş ekranına TC Kimlik numarasını ve şifresini girmelidir.



Şekil 4.4.1 Kullanıcı giriş ekranı

Eğer TC Kimlik numarası veya şifre hatalıysa giriş ekranında Kullanıcı adı ya da şifrenin hatalı olduğuna ilişkin hata mesajı gösterilmektedir.



Şekil 4.4.2 Kullanıcı giriş ekranı hatalı girişin gösterilmesi

Kullanıcı şifresini unuttuysa “*Forgot password?*” yazılı bölüme basıp, şifre güncelleme ekranına erişebilmektedir. Şifre güncelleme ekranı, Şekil 6.4.3’te gösterilmektedir. Bu ekranda, kullanıcının e-posta adresi ile TC Kimlik numarasının girilmesi gerekmektedir. Bu bilgilerden herhangi biri eksikse ekranda uyarı gösterilmektedir.

Şekil 4.4.3 Kullanıcı şifre güncelleme ekranı ve eksik bilgi uyarısı

Kullanıcı e-posta ve TC Kimlik bilgileri doğru şekilde girmemiş ise, e-posta ya da TC Kimlik numarasının sistemde kayıtlı olmadığına ilişkin bir mesaj ekranda gösterilmektedir. Bu mesaj Şekil 4.4.4’te gösterilmektedir.

Şekil 4.4.4 Kullanıcı şifre güncelleme ekranı ve hatalı e-posta/TC Kimlik numarası uyarısı

Kullanıcı, TC Kimlik numarası ile e-posta adresi doğru şekilde girip, “Update” butonuna bastığında yeni bir şifre yaratılır ve kullanıcıya e-posta olarak gönderilmektedir. Ekranda ise yeni şifrenin e-posta olarak gönderildiğine ilişkin mesaj kullanıcıya gösterilmektedir. Bu mesaj Şekil 4.4.5’te gösterilmektedir.

Şekil 4.4.5 Kullanıcı şifre güncelleme ekranı ve yeni şifrenin e-posta ile gönderildiğine ilişkin mesaj

Kullanıcının şifresi güncellendikten sonra kullanıcıya gönderilen e-posta Şekil 4.4.6’da gösterilmektedir.

Password successfully changed

Dear Jodie Diana Black,

Your password has been changed successfully.

Your new password has been set. New password:NqLlguKt

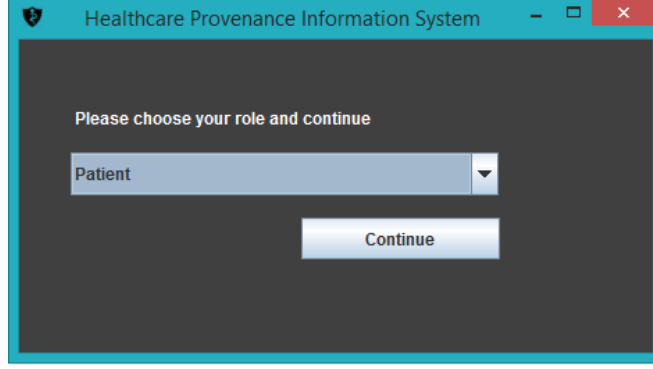
This is an automated email, please don't reply.

Best Regards,

Healthcare Provenance System Administration

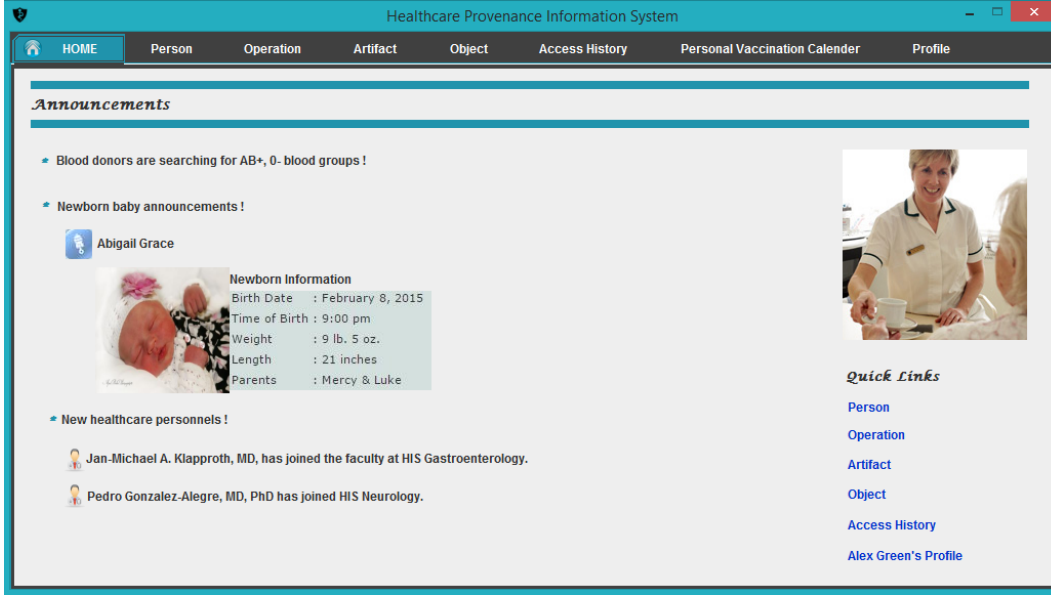
Şekil 4.4.6 Kullanıcıya gönderilen ve yeni şifreyi de içeren e-posta

Kullanıcı, hem hasta hem de sağlık çalışanı gibi birden fazla role sahip ise sisteme giriş yapmak istediğinde yeni bir ekrandan hangi rol ile sisteme giriş yapacağını seçmektedir. Bu ekran Şekil 4.4.7’de gösterilmektedir. Kullanıcı, rolü seçtikten sonra, “Continue” butonuna basıp, sisteme giriş işlemini tamamlamaktadır.



Şekil 4.4.7 Kullanıcının sisteme giriş yapmak istediği rolü seçtiği ekran

Kullanıcı sisteme sağlık çalışanı olarak giriş yaptığında karşılaştığı ekran Şekil 4.4.8’de gösterilmektedir.



Şekil 4.4.8 Sağlık çalışanı ekranı

Sağlık çalışanı kişileri sorgulamak için **Person** sekmesine geçmekte, önce **Purpose** seçmekte ve TC Kimlik numarası/isim bilgisi girmektedir. Aksi halde ekranda uyarı gösterilmekte ve kullanıcının bu bilgileri girmesi beklenmektedir. Bilgiler eksiksiz ise kullanıcı arama butonuna basıp, kişisel bilgiler için tanımlı izin ve yasaklar dikkate alınarak listeleme işlemi gerçekleştirilmektedir. Şekil 4.4.9’da **Person** sekmesi gösterilmektedir. Bölüm 4.2.2’de bahsedilen senaryodaki gibi doktor (Alex Green), hastanın (Tom Watson) bilgilerini sorgulamaktadır.

Identity Number	Fullname	Gender	Edit
34567890122	Tom Watson	Male	Edit

Şekil 4.4.9 Person sekmesi ve kişi arama

Doktorun kişisel bilgileri sadece okuma yetkisi olduğundan bilgiler, güncellemeye kapalı olarak gelmektedir. Şekil 4.4.10'da sadece okuma izni olması halinde kişisel bilgilerin görüntülenmesi gösterilmektedir.

First Name: Tom Birthdate: 1980-07-07 Role: Patient

Second Name: Email: tom.watson@sample.com Identity Number: 34567890122

Surname: Gender: Male

Phone Number: (555)3334444 Marital Status: Married

Address: 1430 North West Blvd. PO Box 688, Vineland, NJ

Şekil 4.4.10 Person sekmesi ve kişisel bilgilerin okuma yetkisi ile görüntülenmesi

Kullanıcının, kişisel bilgiler için yazma yetkisi varsa bu bilgiler güncellenebilir bir şekilde görüntülenmektedir. Şekil 4.4.11'de yazma yetkisi ile kişisel bilgilerin nasıl görüntülendiği gösterilmektedir.

Healthcare Provenance Information System

HOME Person Operation Artifact Object Access History Administration

Name Di Identity Number Purpose Clinical Tests New Item

First Name Jodie Birthdate 1986-01-02 Role Patient

Second Name Diana Email diana.black@sample.com Identity Number

Surname Black Gender Female 12345678901

Phone Number (533)1122345 Marital Status Single

Address 234 Elm St. San Francisco, CA 94115 Submit

Şekil 4.4.11 Person sekmesi ve kişisel bilgilerin yazma yetkisi ile görüntülenmesi

Kişisel bilgiler, yazma yetkisi ile görüntüledikten sonra, ekranda herhangi bir değişiklik yapıp “*Submit*” butonuna basıldığında değişiklikler kaydedilmekte ve işlemin tamamlandığına ilişkin bir kullanıcı mesajı gösterilmektedir. Şekil 4.4.12’de adres alanında güncelleme yapılmakta ve sisteme kaydedilmektedir.

Healthcare Provenance Information System

HOME Person Operation Artifact Object Access History Administration

Name Dia Identity Number Purpose Clinical Tests New Item

The operation completed successfully.

First Name Jodie Birthdate 1986-01-02 Role Patient

Second Name Diana Email diana.black@sample.com Identity Number

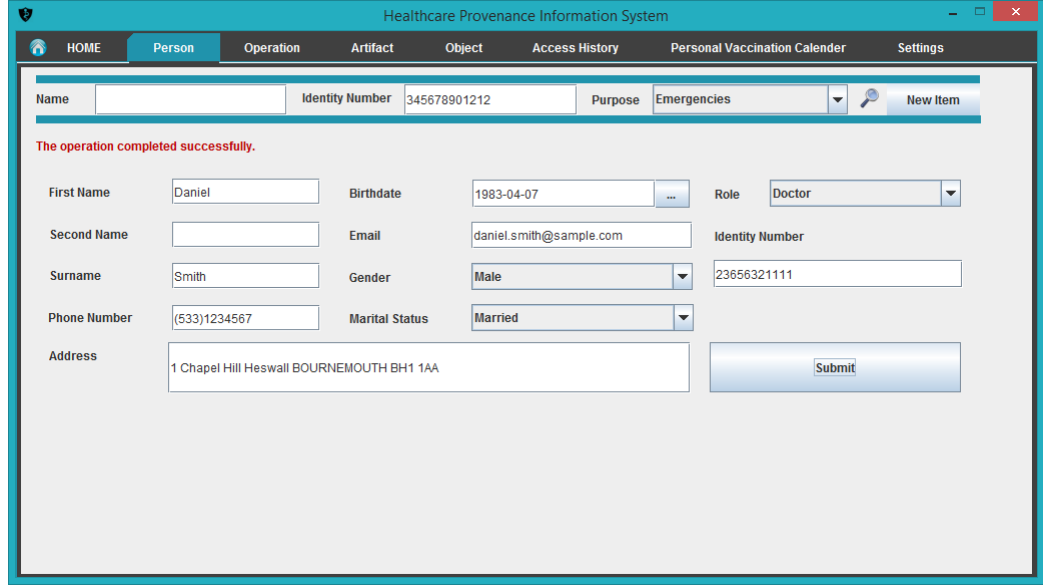
Surname Black Gender Female 12345678901

Phone Number (533)1122345 Marital Status Single

Address 234 Elm St. San Francisco, CA 99999 Submit

Şekil 4.4.12 Person sekmesi ve kişisel bilgilerin güncellenmesi

Yeni kişi eklemek için ise “*New Item*” butonuna basılır ve bilgiler eksiksiz şekilde girildikten sonra “*Submit*” butonuna basılır. Şekil 4.4.13’te yeni bir Person nesnesinin eklenmesi gösterilmektedir.



Healthcare Provenance Information System

HOME Person Operation Artifact Object Access History Personal Vaccination Calender Settings

Name Identity Number 345678901212 Purpose Emergencies

The operation completed successfully.

First Name Daniel Birthdate 1983-04-07 Role Doctor

Second Name Email daniel.smith@sample.com Identity Number

Surname Smith Gender Male 23656321111

Phone Number (533)1234567 Marital Status Married

Address 1 Chapel Hill Heswall BOURNEMOUTH BH1 1AA

Şekil 4.4.13 Person sekmesi ve yeni kişi eklenmesi

Yeni kişi sisteme eklendiğinde yeni bir Artifact nesnesi de sisteme eklenmektedir. Bu Artifact nesnesi için tanımlanan izin ve yasaklar dikkate alınarak kişisel bilgilere erişim kontrol edilmektedir. Yaratılan her bir Artifact nesnesi için Artifact nesnesinin sahibine izin ve yasakları tanımlaması için bir e-posta gönderilmektedir. Bu e-posta Şekil 4.4.14’te gösterilmektedir.

New Artifact Item

Dear Daniel Smith,

An artifact item has been added to Healthcare Provenance Information System.

Please define permissions and prohibitions of your new artifact item.

This is an automated email, please don't reply.

Best Regards,

Healthcare Provenance System Administration

Şekil 4.4.14 Yeni Artifact nesnesi için gönderilen e-posta

Yeni Artifact yaratıldığına ilişkin e-postayı alan kullanıcı, Prohibition sekmesinden yeni bir Prohibition nesnesi yaratıp, Artifact nesnesi ile ilişkilendirebilmektedir. Prohibition sekmesinde yeni bir Prohibition nesnesinin yaratılması için kullanıcı önce “New Item”

butonuna basmakta ve çıkan ekranda ilgili alanları doldurarak “*Submit*” butonuna basarak yeni öğeyi kaydetmektedir. Şekil 4.4.15’te yeni Prohibition nesnesinin detayları gösterilmektedir. Ekranda yer alan Person ve Role alanları ile kısıtlamanın ilişkili olduğu özne belirlenirken, Right alanı ile erişim hakkı ve Purpose alanı ile ise erişim amacı belirlenmektedir. Prohibition nesneleri, Artifact nesnelerinde olduğu gibi, AccessHistory nesnelerinin erişimlerini kısıtlamak için kullanılmaktadır.

Şekil 4.4.15 Prohibition sekmesi ve Prohibition nesnesinin yaratılması

Kullanıcı, yine Prohibition sekmesinden kendi oluşturduğu Prohibition nesnelerini sorgulayıp listeleyebilmektedir. Şekil 4.4.16’da kullanıcı (Susi Anderson) tarafından Prohibition nesnesinin ismine göre yapılan listeleme işlemi gösterilmektedir.

Prohibition Name	Created Date	
Prohibition_3	5015-03-18	Edit

Şekil 4.4.16 Prohibition sekmesi ve Prohibition nesnelerinin sorgulanması

Yeni Artifact yaratıldığına ilişkin e-postayı alan kullanıcı, Permission sekmesinden yeni bir Permission nesnesi yaratıp, Artifact nesnesi ile ilişkilendirebilmektedir. Permission sekmesinde yeni bir Permission nesnesinin yaratılması için kullanıcı önce “New Item” butonuna basmakta ve çıkan ekranda ilgili alanları doldurarak “Submit” butonuna basarak yeni öğeyi kaydetmektedir. Şekil 4.4.17’de yeni Permission nesnesinin detayları gösterilmektedir. Ekranda yer alan Person ve Role alanları ile iznin ilişkili olduğu özne belirlenirken, Right alanı ile erişim hakkı ve Purpose alanı ile ise erişim amacı belirlenmektedir.

Permission Name: Permission_9

Started Date: [] Created Date: 2015-03-18 10:00

Expired Date: []

Related Subjects (People or Roles)

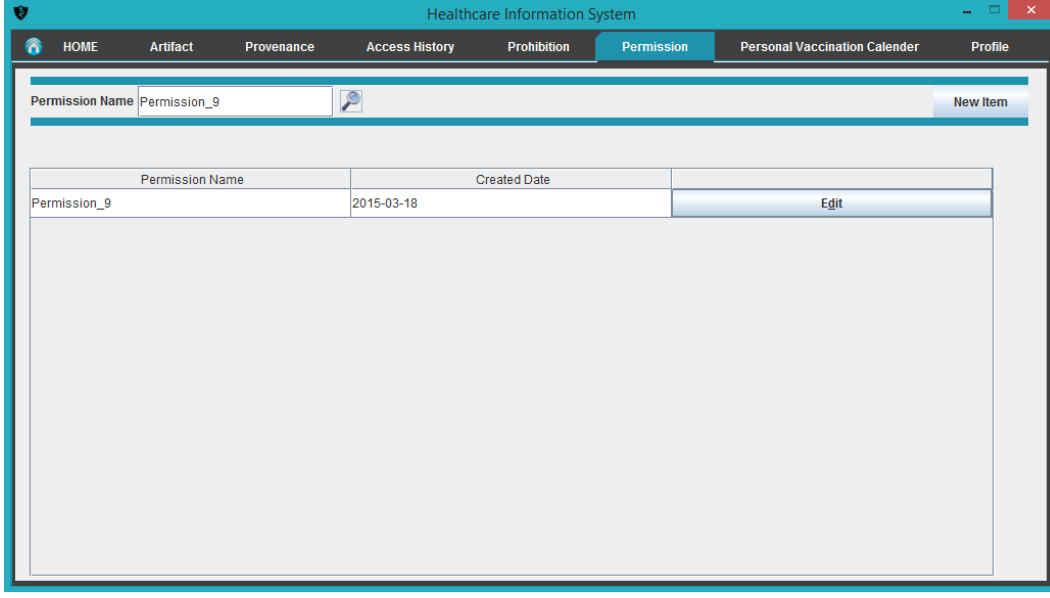
Person	Role	Right	Purpose
12345677701		Read	Clinical Tests

Enter an identity number + - Choose a role + - Choose a right + - Choose a purpose + -

Submit

Şekil 4.4.17 Permission sekmesi ve Permission nesnesinin yaratılması

Kullanıcı, yine Permission sekmesinden kendi oluşturduğu Permission nesnelerini sorgulayıp listeleyebilmektedir. Şekil 4.4.18’de kullanıcı (Susi Anderson) tarafından Permission nesnesinin ismine göre yapılan listeleme işlemi gösterilmektedir. Permission nesneleri, Artifact ve AccessHistory nesnelerinin erişim izinlerini tanımlamak için kullanılmaktadır.



Şekil 4.4.18 Permission sekmesi ve Permission nesnelerinin sorgulanması

Operation nesneleri üzerinde yapılacak işlemler için Operation sekmesi kullanılmaktadır. Bu sekmede, Operation ismine, hangi Object nesnesi için gerçekleştirildiğine (Performed for) ve tipine (Clinical Test/ Precaution/Treatment) göre sağlık personeli ve sistem yöneticisi tarafından sorgulama yapılabilir. Şekil 4.4.19’da Operation tipi Precaution seçilerek yapılan sorgulama gösterilmektedir.

Operation Name	Performed For	Operation Type	
Tdap Vaccine	Tetanus	Precaution	Edit
Tdap Vaccine 2	Tetanus	Precaution	Edit
Tdap Vaccine 3	Tetanus	Precaution	Edit
Hep B Vac Shot 3	Hepatitis B	Precaution	Edit
Operation Hep B Vac 2		Precaution	Edit
Hepatitis B Vaccine	Hepatitis B	Precaution	Edit

Şekil 4.4.19 Operation sekmesi ve Operation nesnelerinin sorgulanması

Operation nesneleri sorgulandıktan sonra Şekil 4.4.20’de gösterildiği gibi detaylar görüntülenmektedir. Bölüm 4.2.2’de örneklenen Tetanoz aşısı için oluşturulmuş aşılama takvimi, Şekil 4.4.20’de gösterilmektedir. Operation nesnesinin, eşleştirildiği aşıya ilişkin varsa aşılama takvimi nesnenin detaylarının alt kısmında gösterilmektedir. Ekran üzerinde aşılama takvimi hariç tüm sahalar, güncellenebilmektedir. Aşılama takvimi, Operation nesnesi ilk kez bir aşı ile eşleştirildiği zaman oluşturulmaktadır. Ardından yeniden başka bir Operation nesnesi aynı aşı ile eşleştirilirse ve PerformedDate alanı doldurulursa daha önce kaydedilmiş olan aşı takvimindeki AdministratedDate alanı güncellenmektedir. Böylece aşı takviminin Operation nesneleri ile ilişkili olarak yaratılması ve güncellenmesi sağlanmaktadır. Geriye dönük Operation nesneleri sorgulandığında gerçekleştirilen işlemler görüntülenebilmektedir. Aşı takvimlerinde ise, hem aşılamanın gerçekleştiği tarih hem de gerçekleştirilmesi için önerilen tarih gösterilmektedir. Böylece, kişinin aşılama geçmişi ve aşılama yapılacak diğer tarihler takip edilebilmekte ve yanlış zamanda yapılabilecek aşılamanın yol açacağı sağlık problemlerinin önüne geçilmektedir.

Healthcare Provenance Information System

HOME Person **Operation** Artifact Object Access History Personal Vaccination Calender Settings

Operation Name: [] Performed for: [] Type: [Precaution] [New Item]

Operation Name: Tdap Vaccine 3 Performed Date: 2013-01-25 00:00:00 Max Applicable Age: 65

Input: Artifact_4 Created Date: [] Min Applicable Age: 11

Output: [] Creator: [] Same as: []

Performed by: 1234566601 Type: [Precaution] Tetanus Vaccine

Requested by: 12345677701 Period-Low Risk Group: 5 years Period-Low High Group: 10 years [Submit]

☒ View vaccination calender

Vaccine Name	Shot Name	Administrated Date	Proposed Date
Tetanus Vaccine	First Shot	2012-06-25	2012-06-25
Tetanus Vaccine	Second Shot	2012-07-25	2012-07-25
Tetanus Vaccine	Third Shot	2013-01-25	2013-01-25
Tetanus Vaccine	Rapel Shot		2018-01-25

Şekil 4.4.20 Operation sekmesi ve bir Operation nesnesinin detaylarının görüntülenmesi

Artifact nesneleri üzerinde yapılacak işlemler için Artifact sekmesi kullanılmaktadır. Artifact nesnesinin sahibinin adı veya TC Kimlik numarasına göre sorgulama yapılabilir. Şekil 4.4.21’de doktor (Alex Green) tarafından, hastanın (Tom Watson) TC Kimlik numarası kullanılarak yapılan sorgulama gösterilmektedir.

Healthcare Provenance Information System

HOME Person Operation **Artifact** Object Access History Personal Vaccination Calender Profile

Name: [] Identity Number: 345678901212 Purpose: [Emergencies] [New Item]

Artifact Name	Owner Identity Number	
Artifact_4	345678901212	[Edit]

Şekil 4.4.21 Artifact sekmesi ve Artifact nesnelerinin sorgulanması

Artifact nesnelerinin detayları okuma yetkisi ya da yazma yetkisi ile görüntülenebilmektedir. Şekil 4.4.22’de doktor (Alex Green) tarafından hasta (Tom Watson) bilgilerinin okuma yetkisi ile görüntülenmesi gösterilmektedir.

The screenshot shows the 'Artifact' tab in the Healthcare Provenance Information System. The top navigation bar includes HOME, Person, Operation, Artifact (selected), Object, Access History, Personal Vaccination Calendar, and Profile. Below the navigation bar, there is a search bar with fields for Name, Identity Number (345678901212), Purpose (Emergencies), and a New Item button. The main content area displays details for 'Artifact_4':

Artifact Name	Artifact_4	Created Date	2012-06-25 00:00:00	Owner Identity Number	
Input Of	Tdap Vaccine	Creator	12345666601		345678901212
Output Of		Previous Version			
Artifact Value	personFullName: Tom Watson personPhoneNumber: (555)3334444 personBirthdate: 1980-07-07 personEmail: tom.watson@sample.com personAddress: 1430 North West Blvd, PO Box 688, Vineland, NJ personIdentityNumber: 345678901212 personAge: 34 personMaritalStatus: Married personGender: Male TDAP Vaccine is required.				

Şekil 4.4.22 Artifact sekmesi ve Artifact nesnesinin okuma yetkisi ile görüntülenmesi

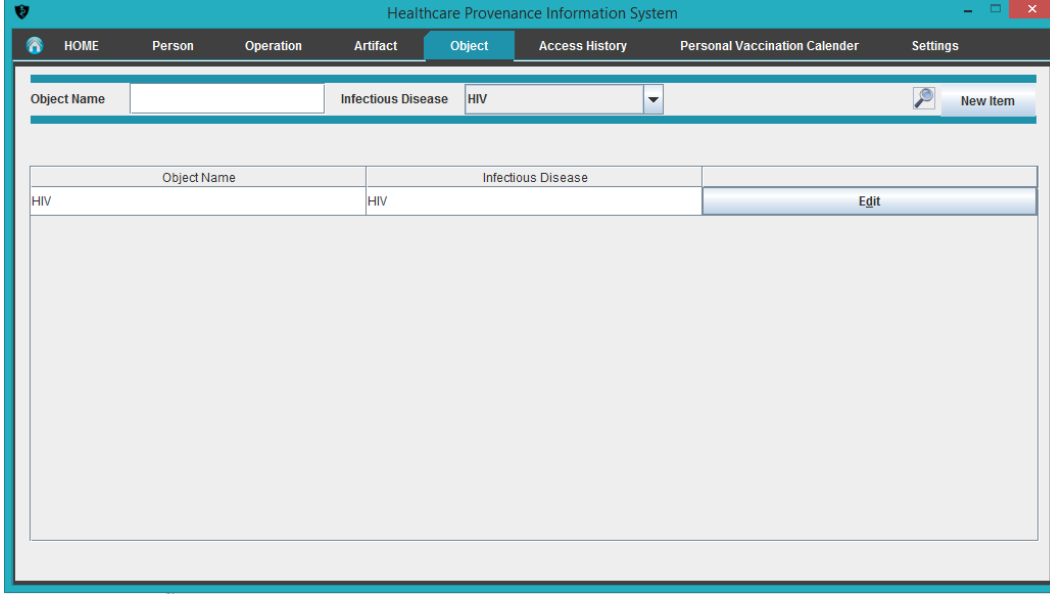
Kullanıcı yetkisi olmadığı halde bir kişiye ait kayıtları sorgulamak isterse ekranda izninin olmadığına ilişkin bir uyarı ile karşılaşmaktadır. Uyarı mesajı, Şekil 4.4.23’de gösterilmektedir.

The screenshot shows the 'Artifact' tab in the Healthcare Provenance Information System. The top navigation bar includes HOME, Person, Operation, Artifact (selected), Object, Access History, Personal Vaccination Calendar, and Settings. Below the navigation bar, there is a search bar with fields for Name, Identity Number (12345678901), Purpose (Clinical Tests), and a New Item button. A red error message is displayed in the main content area:

You don't have permission to access. Please contact your administrator!

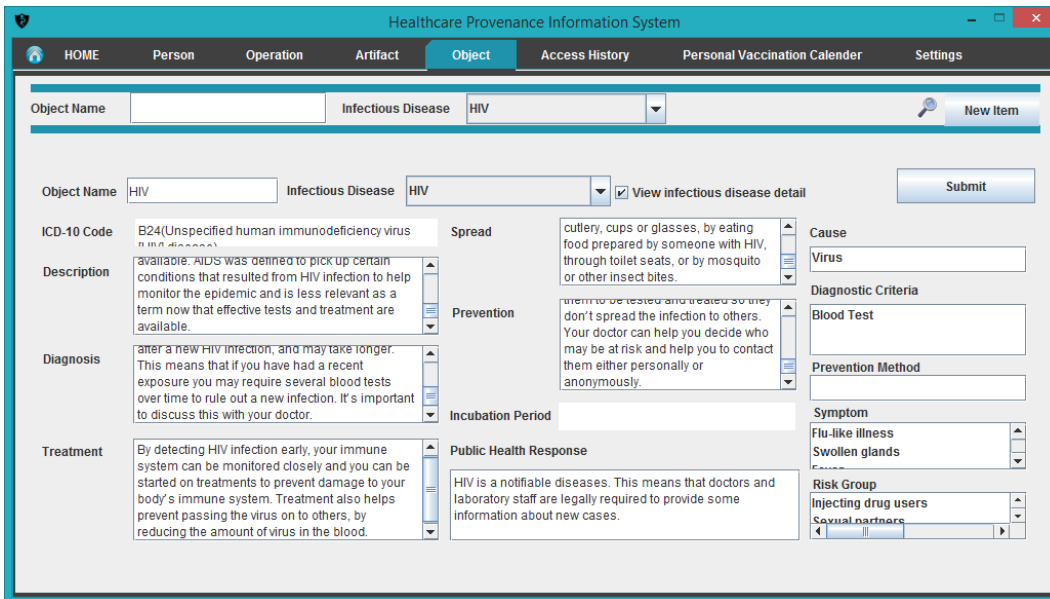
Şekil 4.4.23 Artifact sekmesi ve erişim için gerekli iznin bulunmadığına ilişkin mesaj

Object nesnesi için ise, işlemler Object sekmesi üzerinden yapılmaktadır. Bu sekmede isim ve bulaşıcı hastalık kriterlerine göre arama yapılabilir. Şekil 4.4.24’de bulaşıcı hastalığa göre yapılan aramanın sonuçları gösterilmektedir.



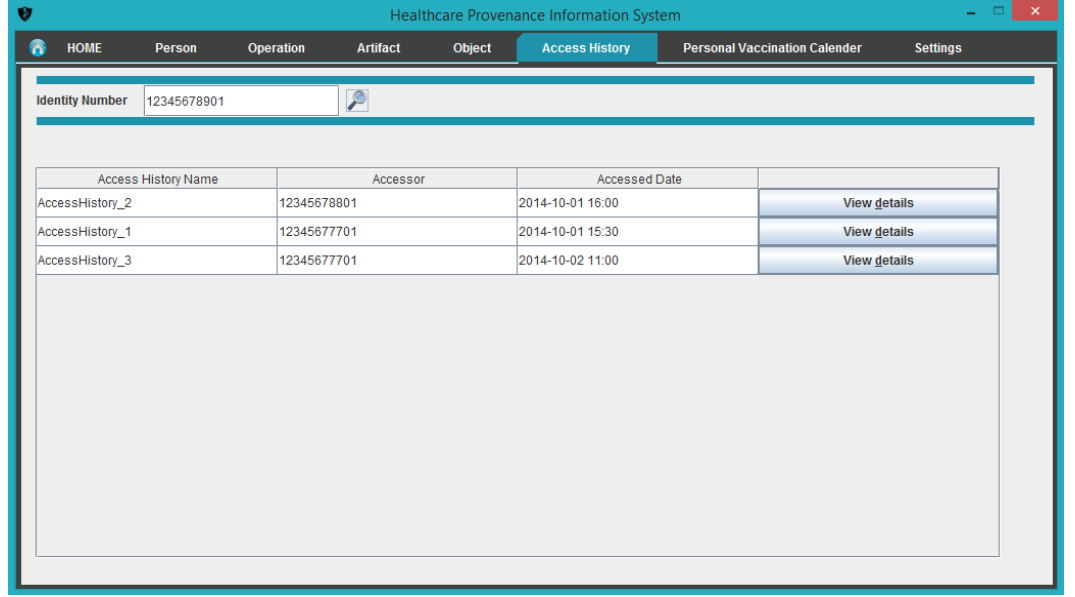
Şekil 4.4.24 Object sekmesi ve Object nesnelerin sorgulanması

Object nesneleri sorgulandıktan sonra, “Edit” butonu yardımıyla detaylar görüntülenebilmektedir. Şekil 4.4.25’de HIV için yaratılmış olan Object nesnesinin detayların yer verilmektedir.



Şekil 4.4.25 Object sekmesi ve Object nesnesinin detaylarının görüntülenmesi

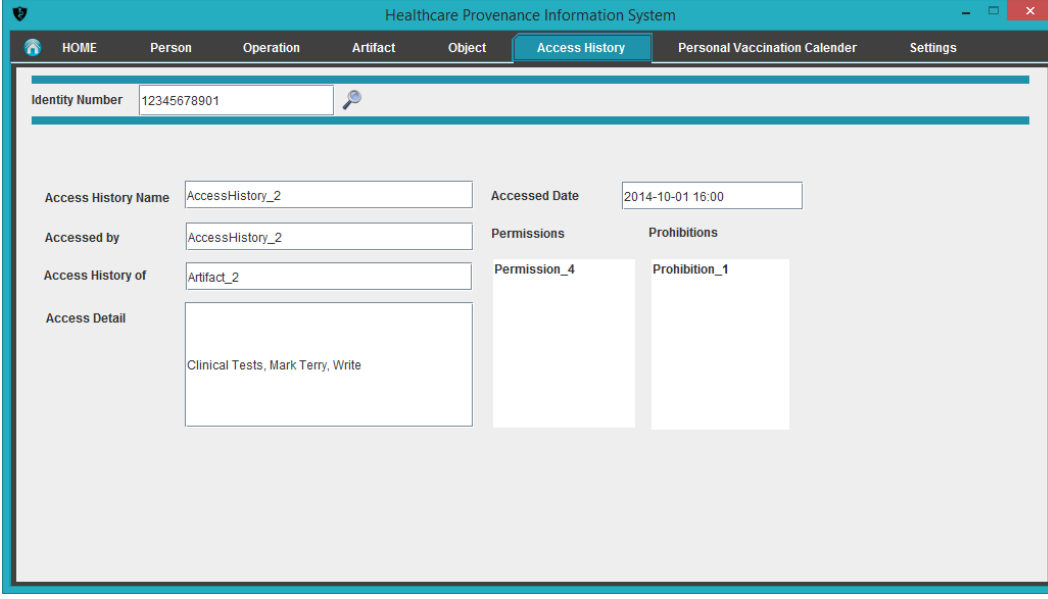
AccessHistory nesneleri, AccessHistory sekmesinde TC Kimlik numarası kullanılarak sorgulanabilmektedir. İzin ve kısıtlamalara göre listeleme yapılmaktadır. Bölüm 4.2.1’de verilen kullanım senaryosunda yer alan “12345678901” TC Kimlik numarası için AccessHistory listesi Şekil 4.4.26’da gösterilmektedir.



Access History Name	Accessor	Accessed Date	
AccessHistory_2	12345678801	2014-10-01 16:00	View details
AccessHistory_1	12345677701	2014-10-01 15:30	View details
AccessHistory_3	12345677701	2014-10-02 11:00	View details

Şekil 4.4.26 AccessHistory sekmesi ve AccessHistory nesnelerinin sorgulanması

Sorgulama yapıldıktan sonra AccessHistory nesnesi için, “*View details*” butonuna basılarak Şekil 4.4.27’de gösterildiği gibi detaylar görüntülenmektedir.



Healthcare Provenance Information System

HOME Person Operation Artifact Object Access History Personal Vaccination Calender Settings

Identity Number: 12345678901

Access History Name: AccessHistory_2 Accessed Date: 2014-10-01 16:00

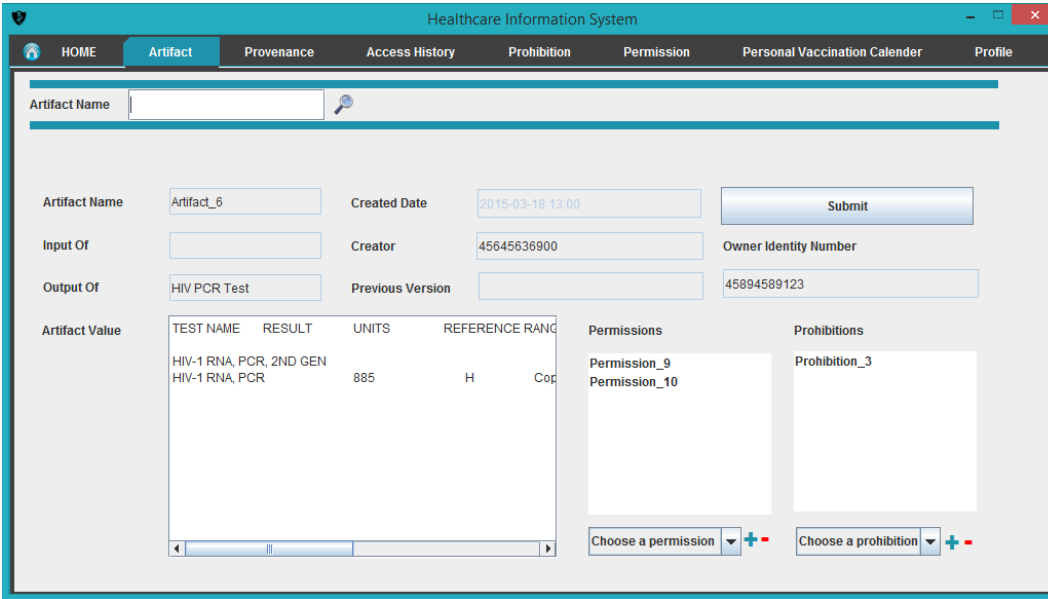
Accessed by: AccessHistory_2 Permissions: Prohibitions

Access History of: Artifact_2 Permission_4 Prohibition_1

Access Detail: Clinical Tests, Mark Terry, Write

Şekil 4.4.27 AccessHistory sekmesi ve AccessHistory detaylarının görüntülenmesi

Hasta, sisteme giriş yaptığında kendine ait Artifact nesnelerini isimlerine göre sorgulayabilmektedir. Sorguladığı Artifact nesnelerinin detaylarını görüntüleyebilmektedir. Artifact nesnelerinin, izin ve yasaklarını belirlemek için Permission ve Prohibition nesneleri ile ilişkilendirilmesini sağlayabilmektedir. Şekil 4.4.28’de hasta (Susi Anderson) tarafından görüntülenen bir Artifact nesnesinin detaylarına yer verilmektedir.



Healthcare Information System

HOME Artifact Provenance Access History Prohibition Permission Personal Vaccination Calender Profile

Artifact Name:

Artifact Name: Artifact_6 Created Date: 2015-03-18 13:00 Submit

Input Of: Creator: 45645636900 Owner Identity Number: 45894589123

Output Of: HIV PCR Test Previous Version:

Artifact Value:

TEST NAME	RESULT	UNITS	REFERENCE RANGE
HIV-1 RNA, PCR, 2ND GEN		885	H Cop
HIV-1 RNA, PCR			

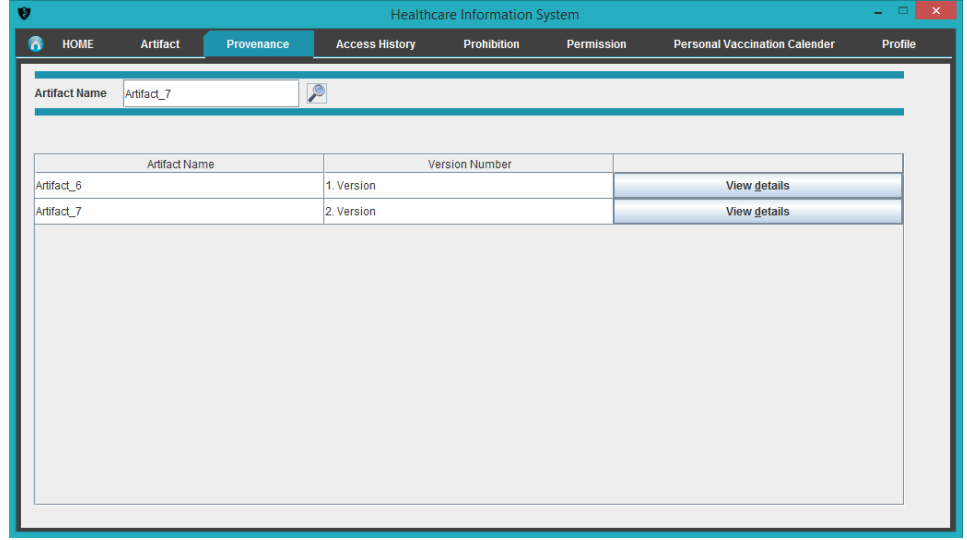
Permissions: Permission_9, Permission_10

Prohibitions: Prohibition_3

Choose a permission: + - Choose a prohibition: + -

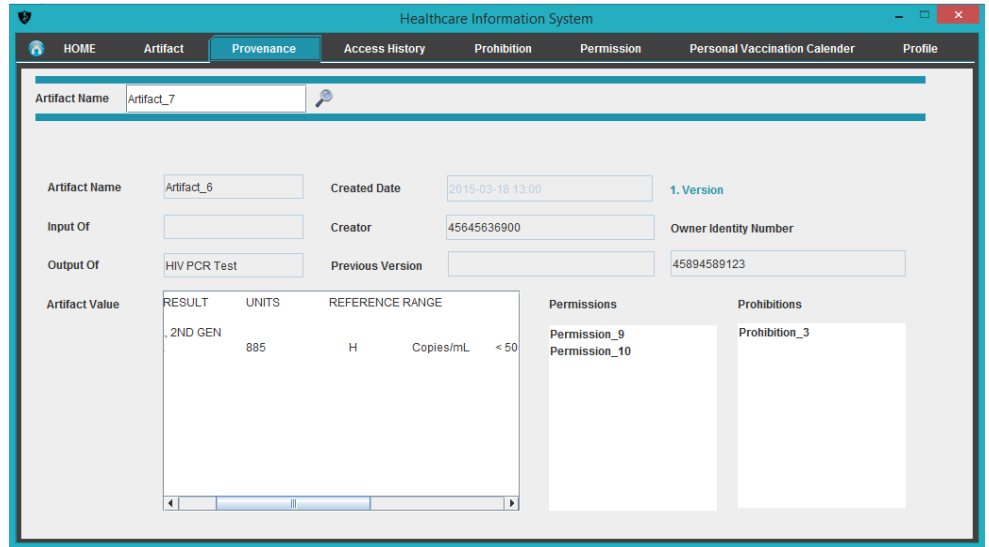
Şekil 4.4.28 Artifact sekmesi ve hasta tarafından Artifact detaylarının görüntülenmesi

Artifact nesnelerinin kökeni için Provenance sekmesi kullanılabilir. Şekil 4.4.29’da hasta (Susi Anderson) tarafından Artifact_7 nesnesinin kökeninin sorgulanması gösterilmektedir. Sorgu sonucunda, ilgili Artifact nesnesinin tüm versiyonları listelenmektedir.



Şekil 4.4.29 Provenance sekmesi ve Artifact nesnesinin kökeninin sorgulanması

Bir Artifact nesnesinin kökeni sorgulandıktan sonra, “*View details*” butonu ile istenilen versiyonun detayları görüntülenebilmektedir. Şekil 4.4.30’da Artifact_7’nin birinci versiyonu olan Artifact_6 nesnesinin detayları gösterilmektedir.



Şekil 4.4.30 Provenance sekmesi ve Artifact nesnesinin bir versiyonunun görüntülenmesi

Hasta, AccessHistory nesnelerini Artifact nesnelerinin isimlerine göre sorgulayabilmektedir. Sorguladığı AccessHistory nesnelerinin detaylarını görüntüleyebilmektedir. AccessHistory nesnelerinin, izin ve yasaklarını belirlemek için Permission ve Prohibition nesneleri ile ilişkilendirilmesini sağlayabilmektedir. Şekil 4.4.31’de hasta (Susi Anderson) tarafından görüntülenen bir AccessHistory nesnesinin detaylarına yer verilmektedir.

Healthcare Information System

HOME Artifact Provenance **Access History** Prohibition Permission Personal Vaccination Calender Profile

Artifact Name:

Access History Name: Accessed Date:

Accessed by: Permissions: Prohibitions:

Access History of:

Access Detail:

Choose a permission Choose a prohibition

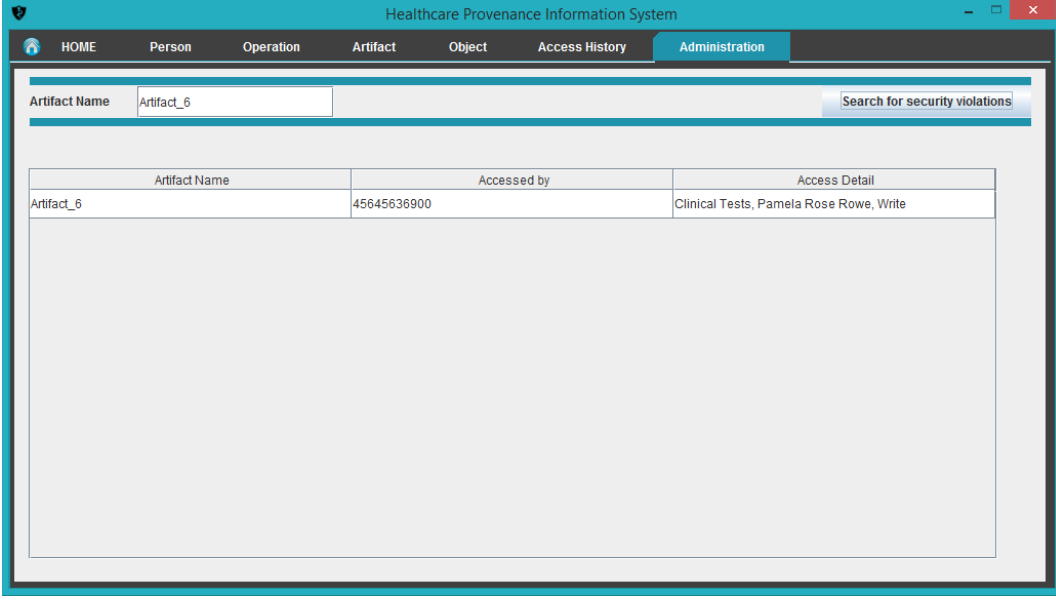
Şekil 4.4.31 AccessHistory sekmesi ve hasta tarafından AccessHistory nesnesinin detaylarının görüntülenmesi

Sağlık çalışanı ve hasta, PersonalVaccinationCalender sekmesini kullanarak kişisel aşı takvimini sorgulayabilmektedir. Şekil 4.4.32’de doktor (Alex Green) tarafından, “12345678901” TC Kimlik numarasına sahip hastaya ait kişisel aşı takviminin görüntülenmesi gösterilmektedir.

Vaccine Name	Shot Name	Administrated Date	Proposed Date
Hepatitis B Vaccine	First Shot	2015-01-03	2015-01-03
Hepatitis B Vaccine	Second Shot	2015-02-03	2015-02-03
Hepatitis B Vaccine	Third Shot	2015-08-03	2015-08-03
Hepatitis B Vaccine	Rapel Shot		2016-08-03

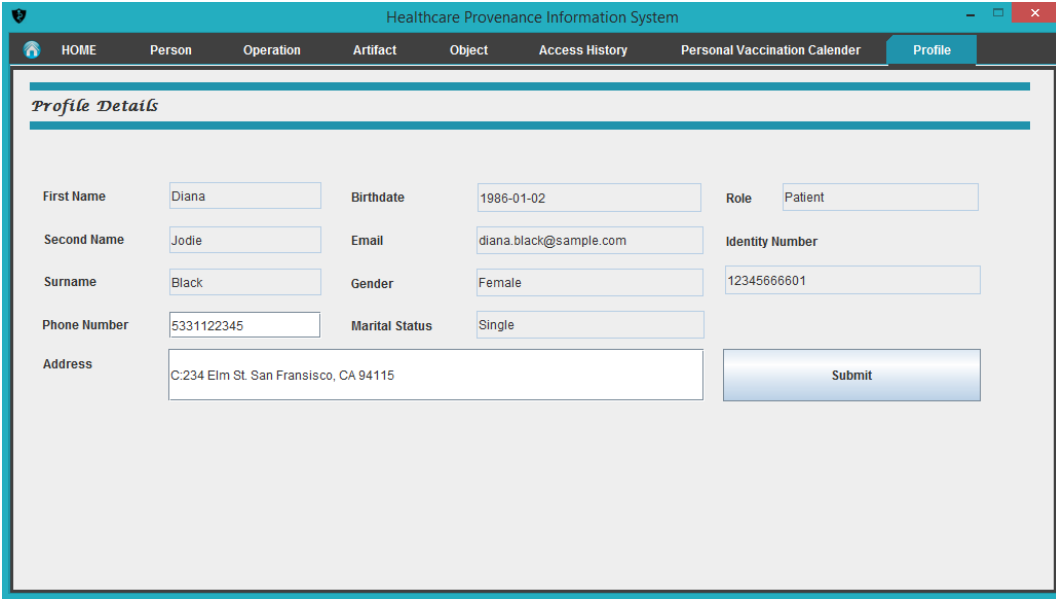
Şekil 4.4.32 PersonalVaccinationCalender sekmesi ve kişisel aşı takviminin görüntülenmesi

Sistem yöneticisi, sağlık çalışanı ve hastadan farklı olarak güvenlik açıklarını sorgulayabilmektedir. Güvenlik açıklarını Artifact nesnesine göre sorgulamak için Artifact nesnesinin ismini girmesi gerekmektedir. Bölüm 4.1.2.2’te bahsedilen güvenlik açığının saptanması için Artifact_6 için yapılan güvenlik açıkları arama sonuçları Şekil 4.4.33’de gösterilmektedir. Güvenlik açığı, Artifact_6 nesnesinin erişim geçmişlerinin, Artifact_6 nesnesi için tanımlanmış izin (Permission_9, Permission_10) ve yasaklara (Prohibition_3) uygunluğu karşılaştırılarak saptanmaktadır. Administration ekranında ise, güvenlik açığının saptandığı erişim geçmişinin (AccessHistory_5) detayları gösterilmektedir. Buna göre, hemşire rolüne sahip olan Pamela Rose Rowe isimli sağlık çalışanının, Artifact_6 nesnesi için erişim kısıtlaması olmasına rağmen erişim yaptığı tespit edilmiştir.



Şekil 4.4.33 Administration sekmesi ve sistem yöneticisi tarafından güvenlik ihlallerinin tespit edilmesi

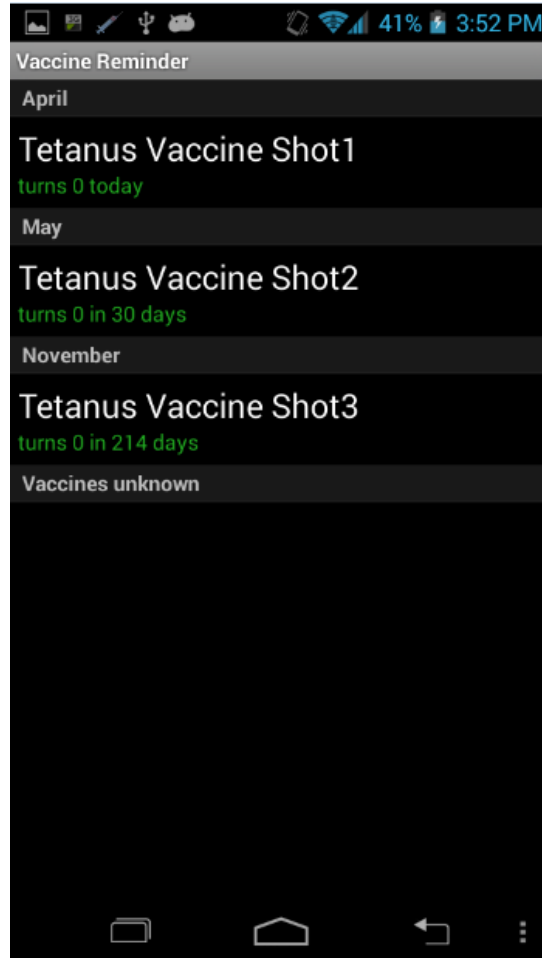
Tüm kullanıcılar, Profile sekmesi üzerinden FOAF ontolojisi üzerinde tanımlı olan kişisel profillerini ve bu profillerinde kayıtlı olan bilgileri görüntüleyebilmektedir. Kullanıcılar, FOAF profillerinde kayıtlı olan, telefon numarası ve adres bilgilerini bu ekrandan güncelleyebilmektedir. Diğer bilgiler, bu ekrandan güncellenememektedir. Şekil 4.4.34’de Profile sekmesi gösterilmektedir.



Şekil 4.4.34 Profile sekmesi ve kullanıcılar tarafından kendi profillerinin görüntülenmesi

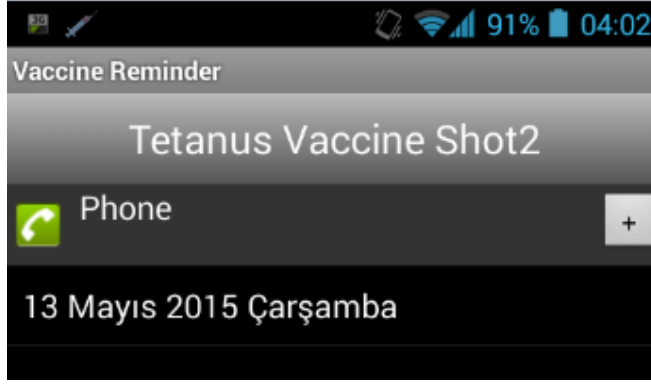
4.5 SKBS Mobil Uygulaması

Tez kapsamında, kullanıcıların mobil olarak aşılama tarihlerini takip edebilmesi ve aşılama tarihi öncesinde hatırlatma bildirimi alabilmesi için mobil uygulama geliştirilmiştir. Uygulamaya TC Kimlik numarası ile giriş yapıldıktan sonra, kullanıcının aşıları listelenmektedir. Aşı adının alt kısmında, aşılama için beklenmesi gereken zaman belirtilmektedir. Eğer aşılama tarihi bugün ise, bu zaman alanındaki değer sıfır olmaktadır.



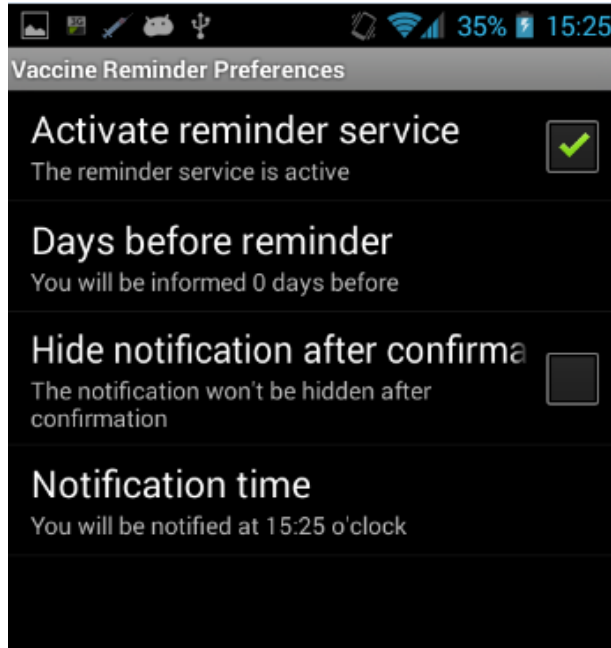
Şekil 4.5.1 Aşı hatırlatma uygulaması aşıların listelenmesi

Aşıların yer aldığı listede herhangi bir aşı üzerine gidildiğinde aşılama tarihi görüntülenmektedir. Şekil 4.5.2’de aşılama tarihinin görüntülenmesi gösterilmektedir.



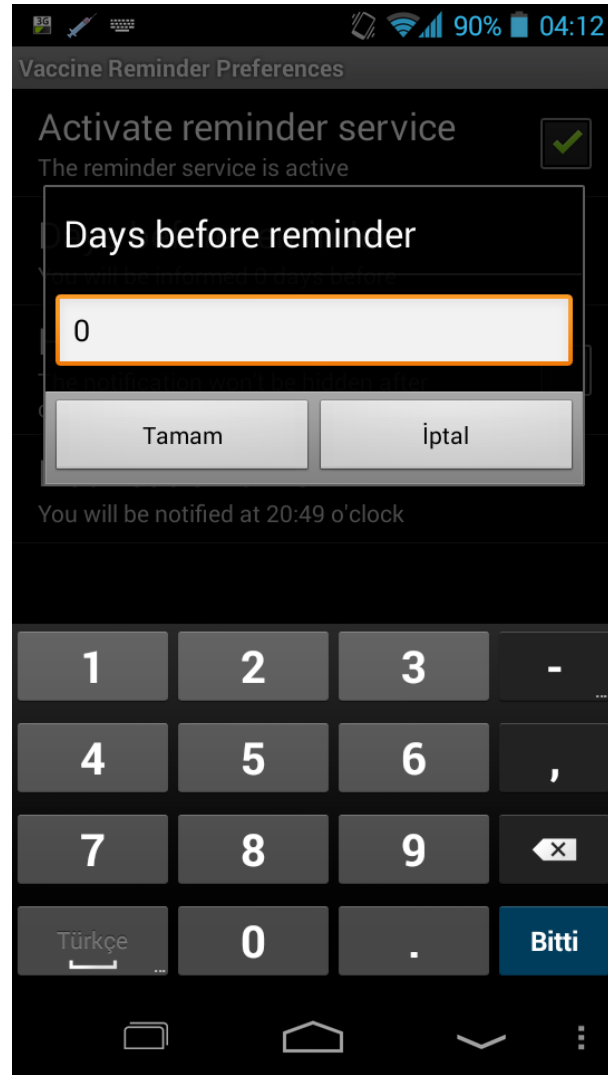
Şekil 4.5.2 Aşılama tarihinin görüntülenmesi

Aşılama hatırlatması için ayarlar altından zamanlama ayarı yapılabilmektedir. Bu hatırlatma zamanı dikkate alınarak bildirimler gösterilmektedir. Şekil 4.5.3’de aşılama tarihi hatırlatıcısının ayarları gösterilmektedir.



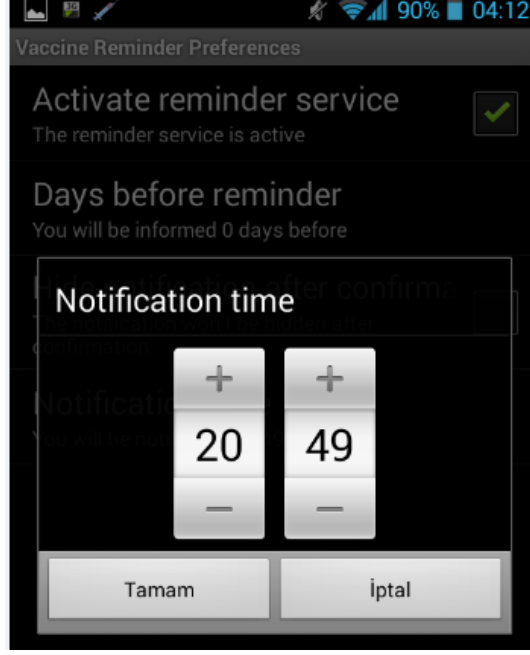
Şekil 4.5.3 Aşılama tarihi hatırlatıcısının ayarlanması

“*Days before reminder*” bölümüne basıldığında, hatırlatmanın kaç gün önceden yapılması isteniyorsa, ona göre gün sayısı ayarlanmaktadır. Şekil 4.5.4’te aşılama hatırlatmasının gün ayarının yapıldığı ekran gösterilmektedir.



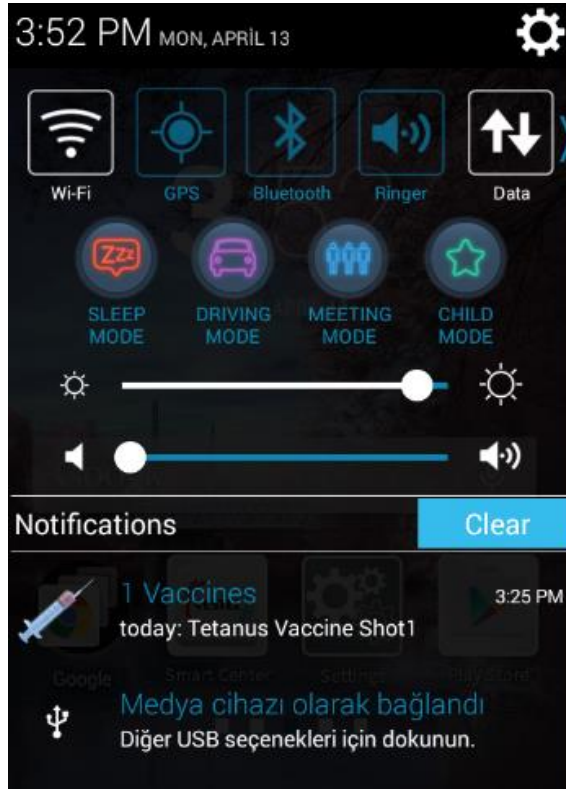
Şekil 4.5.4 “Days before reminder” ekranı

“Notification Time” bölümü, aşı hatırlatmasının, gün içinde hangi saatte yapılacağını belirten bildirim saatini ayarlamak için kullanılmaktadır. Şekil 4.5.5’te bildirim saati ayarlama ekranı gösterilmektedir.



Şekil 4.5.5 Bildirim saati ayarlama ekranı

Aşı hatırlatma ayarları yapıldıktan sonra, hatırlatma tarihi geldiğinde bildirim ekranında aşı adının yer aldığı bir bildirim gösterilmektedir. Aşı hatırlatma bildirimi, Şekil 4.5.6’da gösterilmektedir.



Şekil 4.5.6 Aşı hatırlatma bildirimin görüntülenmesi

5. SONUÇLAR VE GELECEK ÇALIŞMALAR

Tez kapsamında, hasta kayıtlarında güvenlik ve gizliliğin sağlanması için Anlamsal Web tabanlı Sağlık Köken Bilgi Sistemi geliştirilmiştir. Bu sistemin geliştirilmesi sırasında, farklı kullandırım senaryoları sunulmakta ve sistem üzerinde çeşitli örneklerle yer verilmektedir.

Köken kavramı, verinin orijinali ve veri üzerinde yapılan değişiklik geçmişi hakkındaki bilgiler olarak tanımlanmaktadır. Dolayısıyla, köken bilgisi; hatalı veri parçalarının tespit edilip onarılması, verinin izinsiz erişimlerden korunması ve güvenlik ihlallerinin tespit edilmesi için güvenlik kavramları ile birlikte kullanılmaktadır. Tez çalışmasında ise, AKM ve DC Profil kullanılarak gizlilik farkında Anlamsal Web tabanlı bir köken modeli geliştirilmektedir. Bu köken modeli ile; veri ilk yaratıldığı andan itibaren izlenebilmekte, veriye yapılan tüm erişimlerin detayları kaydedilmekte ve veriler için erişim izin ve yasakları tanımlanabilmektedir. Bu model üzerinden, bir veri için erişim geçmişleri taranıp, erişim izin ve yasakları ile kontrol edilerek güvenlik ihlalleri tespit edilebilmektedir. Bu model farklı etki alanlarına uygulandığında, temel kişisel bilgilere de ihtiyaç duyulmaktadır, ancak bu bilgiler model kapsamında yer almamaktadır. Bu nedenle, temel kişisel bilgilerin saklanması ve farklı uygulamalar tarafından kullanılabilmesi için FOAF ontolojisinden (Bursa, 2014) yararlanılmaktadır. Kişiler ve tanımlı roller için yaratılan profiller yardımıyla kişisel bilgiler ve rol detayları tutulmaktadır. Bu ontoloji, kişilerin demografik ve dinamik bilgilerinin profil yapısı altında tutulmasına olanak sağlamaktadır.

Kişisel verilerin güvenliği, bulaşıcı hastalık alanında daha hassas ve önemli bir konudur. Geliştirilen bu model, kişilerin bulaşıcı hastalık geçmişlerinin yetkisiz erişimlerden korunması için, tez kapsamında geliştirilen Bulaşıcı Hastalıklar Ontolojisi (BHO) ile entegre edilmektedir. Bulaşıcı hastalıklardan korunma yöntemi olarak aşılama bilgisine ve aşılama takvimine de ihtiyaç duyulduğundan, Ulusal Aşı Ontolojisi de model ile entegre edilmektedir. Böylece, bulaşıcı hastalık etki alanında yer alan kullanıcılar için aşağıdaki gereksinimler karşılanmaktadır:

- Hastaların kayıtlı verileri ile ilgili geçmişe yönelik detayların erişilebilir olması.
- Hastaya ait bulaşıcı hastalık kayıtlarının güvenliği ve gizliliğinin sağlanması.
- Hastanın aşı geçmişinin erişilebilir olması.
- Aşılama tarihlerinin önerilmesi ve hatırlatılması.
- Hastalığın seyrinin takip edilebilmesi.

Geliştirilen köken modelinin, UAO ve BHO ile entegrasyonundan sonra kullanıcılar tarafından kullanılabilmesi için SKBS geliştirilmektedir. SKBS, bulaşıcı hastalıklar etki alanında yer alan kullanıcıların isteklerini kullanıcı arayüzleri yardımıyla karşılamaktadır.

Tez çalışması süresince elde edilen deneyimler sonucunda geliştirilen bu köken modelinin;

1. Anlamsal Web tabanlı güvenlik ve gizliliği destekleyen erişim kontrol sistemleri için örnek bir temel oluşturacağı,
2. Birbiri ile entegre şekilde çalışan farklı Anlamsal Web modellerinin, Anlamsal Web destekli veritabanında saklanarak, sorgulanması ve kayıtların bu veritabanı üzerinde yaratılması; veritabanı üzerinde Anlamsal Web modellerinin işletilmesi için yapılan çalışmalara bir örnek oluşturacağı,
3. Köken yönetimi alanında yapılacak çalışmalar için bir temel oluşturacağı

düşünülmektedir.

Gelecek çalışmalarda, Gizlilik Farkında Anlamsal Köken Modeli ile entegre edilmek üzere politikaları içeren ontolojiler yazılarak, model üzerinde politikalardan yararlanılarak verilerin güvenlik ve gizlilik ihtiyacı karşılanabilecektir. Ayrıca, geliştirilen model Ontoloji Tabanlı Erişim Denetim (Ontology-Based Access Control) (Can, 2009) düzeneği ile entegre edilebilecek ve iyi tanımlanmış politikalar yardımıyla modeldeki uyumsuzlukların çözümü gerçekleştirilebilecektir (Can et al., 2010). Bu model, tüm kişiler tarafından ortak bir şekilde kullanılır hale getirilecek, kişisel bilgiler için tanımlı izin ve yasaklar tek bir sistem altından yapılandırılabilir, erişim denetimi sağlanabilir, ayrıca güvenlik ve gizliliği destekleyen köken erişim kontrol aracı haline getirilebilecektir. BHO ve UAO; yeni örnekler, sınıflar ve kavramlar eklenerek genişletilecektir. SKBS farklı teknoloji ortamlarına ve farklı etki alanlarına taşınacaktır. Örneğin; .Net için hazırlanmış Jena çatısı kullanılarak mevcut Java altyapısı ile oluşturulmuş durumda olan SKBS, C# altyapısı ile geliştirilebilecektir. Bu işlem için DotNetRDF (DotNetRDF, 2015) ya da Jena .Net (LinkedDataTools, 2015) gibi hazır araçlar kullanılabilir. SKBS için manuel olarak yaratılan sınıflar Kazuki (Rager, 2005) gibi Anlamsal Web tabanlı modellerden otomatik olarak sınıf oluşturan araçlar kullanılarak yaratılacaktır. Güvenlik ve gizliliği destekleyen Anlamsal Web tabanlı köken modeli sadece bulaşıcı hastalıklar için değil, diğer etki alanlarında yer alan modellerle entegre edilecek, sistemlerde kayıtlı kullanıcı bilgilerinin güvenliği sağlanabilir ve güvenlik ihlallerinin tespiti için kullanılabilir. Bulaşıcı hastalıklar etki alanı için ise, bulaşıcı hastalıkların seyri incelenerek diğer hastalarda benzer koşullarda sonuçların öngörülmesi, uygulanacak tedavi ve tetkikler için geçmiş bulaşıcı hastalık köken bilgileri kullanılarak kritik kararlarda destek sağlanması için yeni algoritmalar geliştirilecek ve köken modeli yeni kavramlarla genişletilecektir. Ayrıca, bulaşıcı hastalıklar etki alanında risk gruplarının belirlenmesi ile gerekli önlemler (aşı vb.) alınarak hastalıkların görülme oranlarının azaltılmasına katkıda bulunulacaktır.

KAYNAKLAR DİZİNİ

- Ni1, Q., Xu, S., Bertino, E., Sandhu, Ravi. and Han, W.,** 2009, An Access Control Language for a General Provenance Model, Published in: Proceeding SDM '09 Proceedings of the 6th VLDB Workshop on Secure Data Management, 68–88 pp.
- Tozer, G.,** 1999, Metadata Management For Information Control and Business Success, 217-147 pp.
- Berners-Lee, T., James H. and Ora L.,** 2001, The Semantic Web, Scientific American Magazine, 4p.
- Berners-Lee, T.,** 2006, Artificial Intelligence and the Semantic Web, slide 10, [http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html#\(14\)](http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html#(14)) (Erişim tarihi: 10.02.2006).
- Berners-Lee, T. and Jaffe, J.,** The World Wide Web Consortium, <http://www.w3.org/Consortium/> (Erişim tarihi: 01.06.2015).
- Sezer, E.,** 2014, Tıbbi Bilişim Standartları ve Anlamsal Web Teknolojileri Temelli Ulusal Aşı Bilgi Sistemi, Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, 130s.
- Gerber , A., van der Merwe, A. and Barnard, A.,** 2008, A Functional Semantic Web Architecture, The Semantic Web: Research and Applications Lecture Notes in Computer Science Volume 5021, 273-287 pp.
- Sizov, S.,** 2007, What Makes You Think That? The Semantic Web's Proof Layer, Published by: IEEE Intelligent Systems, 22(5).
- Lima, C., Lu, S., Chebotkob, A. and Fotouhia, F.,** 2011, Storing, Reasoning, and Querying OPM-Compliant Scientific Workflow Provenance Using Relational Databases, Published in: Journal Future Generation Computer Systems archive Volume 27 Issue 6, 781-789 pp.

KAYNAKLAR DİZİNİ (devam)

- Foster, I., T., Vockler, J., Wilde, M. and Zhao, Y.,** 2002, Chimera: A virtual data system for representing, querying, and automating data derivation. In SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management, 37–46 pp, Washington, DC, USA. IEEE Computer Society.
- Zhao, J., Goble, C., A., Stevens, R. and Bechhofer, S.,** 2004, Semantically linking and browsing provenance logs for e-science. In ICSNW, 158–176 pp.
- Myers, J., D., Allison, T., C., Bittner, S., Didier, B., Frenklach, M., William, J., Green, H., Ho, Y., L., Hewson, J., Koegler, W., Lansing, C., Leahy, D., Lee, M., McCoy, R., Minkoff, M., Nijssure, S., von Laszewski, G., Montoya, D., Pancerella, C., Pinzon, R., Pitz, W., Rahn, L., A., Ruscic, B., Schuchardt, K., Stephan, E., Wagner, A., Windus, T. and Yang, C.,** 2004, A collaborative informatics infrastructure for multi-scale science. Clade, Cluster Computing 8(4): 243-253 pp.
- Frew, J. and Bose, R.,** 2001, Earth system science workbench: A data management infrastructure for earth science products. In SSDBM '01: Proceedings of the Thirteenth International Conference on Scientific and Statistical Database Management, Washington, DC, USA, IEEE Computer Society, 180p.
- Widom., J.,** 2005, Trio: A system for integrated management of data, accuracy, and lineage. In Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR'05).
- Barga, R., S. and Digiampietri, L., A.,** 2006, Automatic generation of workflow provenance. In Proceedings of the International Provenance and Annotation Workshop (IPAW), 1–9 pp.

KAYNAKLAR DİZİNİ (devam)

- Szomszor, M. and Moreau, L.,** 2003, Recording and reasoning over data provenance in web and grid services. In International Conference on Ontologies, Databases and Applications of Semantics (ODBASE), volume 2888 of Lecture Notes in Computer Science, Catania, Italy, 603–620 pp.
- Vijayakumar, N., N. and Plale, B.,** 2006, Towards low overhead provenance tracking in near real-time stream filtering. In Proceedings of the International Provenance and Annotation Workshop (IPAW), 46–54 pp.
- Golbeck, J.,** 2006, Combining provenance with trust in social networks for semantic web content filtering. In Proceedings of the International Provenance and Annotation Workshop (IPAW), 101–108 pp.
- Tannenbaum, A.,** 2001, Metadata Solutions: Using Metamodels, Repositories, Xml, and Enterprise Portals to Generate Information on Demand, 85–173 pp.
- Glavic, B. and Dittrich, K.,** 2007, Data Provenance: A Categorization of Existing Approaches, In Proceedings of the 12th GI Conference on Datenbanksysteme in Business, Technologie und Web (BTW), 15p.
- Kifor, T., Varga, L., Z., Vázquez-Salceda, J., Álvarez, S., Willmott, S., Miles, S. and Moreau, L.,** 2007, Provenance in Agent-mediated Healthcare Systems, Published in: Intelligent Systems, IEEE (Volume:21 , Issue: 6), 38-46 pp.
- Moreau, L., Freire, J., Futrelle, J., McGrath, R., E., Myers, J. and Paulson, P.,** 2008, The Open Provenance Model: An Overview, Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, 323-326 pp.

KAYNAKLAR DİZİNİ (devam)

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E. and den Bussche, J., V., 2010, The Open Provenance Model: Abstract Model Latest OPM specification. The open provenance model core specification (v1.1). Future Generation Computer Systems, 30p.

Moehrke, J., 2012, Healthcare Metadata, Online at: <http://healthcaresecprivacy.blogspot.com.tr/2012/05/healthcare-metadata.html> (Erişim tarihi: 13/04/2015)

Kluge, W., 2004, Informed consent and the security of the electronic health record (EHR): some policy considerations, International Journal of Medical Informatics, Volume 73, Issue 3, 229–234 pp.

Dünya Sağlık Örgütü, 1994, Mahremiyet ve Özel Hayat, Avrupa Hasta Haklarının Geliştirilmesi Bildirgesi Olarak Amsterdam Bildirgesi.

Dünya Tabipler Birliği, 1995, Bali Bildirgesi, Endonezya.

Rapoza, J., 2006, SPARQL Will Make the Web Shine, eWeek, Retrieved 2007-01-17, <http://www.eweek.com/c/a/Application-Development/SPARQL-Will-Make-the-Web-Shine> (Erişim tarihi: 13/04/2015)

Herman, I., 2008, SPARQL is a Recommendation, http://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation/ (Erişim tarihi: 13/04/2015)

The Apache Software Foundation, 2013, Jena tutorials, <http://jena.apache.org/tutorials/index.html> (Erişim tarihi: 17/03/2015)

KAYNAKLAR DİZİNİ (devam)

Stanford University, 2009-2014, Protégé, <http://protege.stanford.edu/> (Erişim tarihi: 13/04/2015)

The Eclipse Foundation, 2001-2014, Eclipse, Online at: <https://www.eclipse.org/org/> (Erişim tarihi: 13/04/2015)

Moreau, L., Groth, P. and Zhao, J., OPM Tutorial, <http://openprovenance.org/tutorial/> (Erişim tarihi: 20/09/2010)

Dünya Sağlık Örgütü, International Classification of Diseases (ICD), <http://www.who.int/classifications/icd/en/> (Erişim tarihi: 13/04/2015)

Baader, F., Horrocks, I. and Sattler, U., 2007, Chapter 3 Description Logics, In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, Handbook of Knowledge Representation, Elsevier, 47p.

Rudolph, S., 2011, Foundations of Description Logics, 7th International Summer School 2011, Galway, Ireland, Tutorial Lectures, 76-136 pp.

Miles, S., Moreauy, L. And Futrellez, J., 2009, OPM Profile for Dublin Core Terms (Version 0.3), <http://twiki.ipaw.info/pub/OPM/ChangeProposal-DublinCoreMapping/dcprofile.pdf> (Erişim tarihi: 30/07/2009)

LinkedDataTools, Jena .Net, <http://www.linkeddatatools.com/downloads/jena-net> (Erişim tarihi: 13/04/2015)

DotNetRDF, An Open Source C# .Net Library for RDF, <http://dotnetrdf.org/tagview.asp?tagID=231> (Erişim tarihi: 13/04/2015)

KAYNAKLAR DİZİNİ (devam)

- Rager, D.**, Using the Kazuki Generated Java Interfaces, <http://owl-eclipse.projects.semwebcentral.org/html/userguide/tasks-5.2.html> (Erişim tarihi: 07/09/2005)
- Davidson, S., B.**, 2010, On Provenance and Privacy, Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, 8p.
- Bache, R., Miles, S., Coker, B. and Taweel, A.**, 2013, Informative Provenance for Repurposed Data: A Case Study Using Clinical Research Data, The International Journal of Digital Curation (Volume 8, Issue 2), 27-46 pp.
- Patel, C., Cimino, J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E. and Srinivas, K.**, 2007, Matching Patient Records to Clinical Trials Using Ontologies, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, 816-829 pp.
- Sordo, M., Rocha, B. H., Morales, A. A., Maviglia, S. M., Dell'Oglio, E., Fairbanks, A., Aroy, T., Dubois, D., Bouyer-Ferullo, S. and Rochaab, R. A.**, 2012, Modeling Decision Support Rule Interactions in a Clinical Setting, BMC Medical Informatics and Decision Making 2012, 5p.
- Missier, P., Bryans, J., Dangern, R. and Curcin, V.**, 2013, Preserving privacy in shared provenance data, Newcastle University, Computing Science, Technical Report Series, CS-TR-1366, 15p.
- Nunes, I., Chen, Y., Miles, S., Luck, M. and Lucena, C.**, 2012, Transparent Provenance Derivation for User Decisions, 4th International Provenance and Annotation Workshop, IPAW 2012, Santa Barbara, CA, USA, 111-125 pp.

KAYNAKLAR DİZİNİ (devam)

Cowell, L. G. and Smith, B., 2010, Infectious Disease Ontology, Infectious Disease Informatics 2010, 373-395 pp.

Goldfain, A., Smith, B. and Cowell, L. G., 2010, Dispositions and the Infectious Disease Ontology. In: Proc. of the FOIS 2010, 400–413 pp.

RDF Working Group, Resource Description Framework (RDF), <http://www.w3.org/RDF/> (2014) (Erişim tarihi: 23/03/2015)

W3C Recommendation, RDF Schema 1.1 , <http://www.w3.org/TR/rdf-schema/> (Erişim tarihi: 18/02/2014)

Quin, L., Extensible Markup Language (XML), <http://www.w3.org/XML/> (Erişim tarihi: 20/04/2015)

Kifer, M. and Boley, H., RIF Overview (Second Edition), <http://www.w3.org/TR/rif-overview/> (Erişim tarihi: 02/04/2013)

Stephens, S., Morales, A. and Quinlan, M., 2006, Applying Semantic Web Technologies to Drug Safety Determination, Intelligent Systems IEEE, 21(1), 82-88 pp.

Oracle Corporation, Oracle Database, <https://www.oracle.com/database> (Erişim tarihi: 12/05/2015)

KAYNAKLAR DİZİNİ (devam)

Oracle Corporation, Oracle Database 10g, Technical Information,
<http://www.oracle.com/technetwork/database/database10g/overview/index-095623.html> (Erişim tarihi: 12/05/2015)

Oracle Corporation, Oracle Database Online Documentation 11g Release 2 (11.2), http://docs.oracle.com/cd/E11882_01/index.htm (Erişim tarihi: 27/04/2015)

Oracle Corporation, Oracle Database 12c Release 1 (12.1),
<http://docs.oracle.com/database/121/index.htm> (Erişim tarihi: 08/05/2015)

The Apache Software Foundation, SPARQL Tutorial,
<https://jena.apache.org/tutorials/sparql.html> (Erişim tarihi: 17/03/2015)

World Health Organization (WHO), Nisan 2015, Malaria,
<http://www.who.int/mediacentre/factsheets/fs094/en/> (Erişim tarihi: 12/05/2015)

İstanbul İl Sağlık Müdürlüğü, Nisan Ayının son haftası: Aşı Haftası
http://www.istanbulsaglik.gov.tr/w/onemli_gun/asi_haft.asp (Erişim tarihi: 12/05/2015)

World Health Organization (WHO), Mart 2015, Hepatit B,
<http://www.who.int/mediacentre/factsheets/fs204/en/> (Erişim tarihi: 13/05/2015)

KAYNAKLAR DİZİNİ (devam)

World Health Organization (WHO), Mart 2012, Tetanoz, http://www.wpro.who.int/mediacentre/factsheets/fs_20120307_tetanus/en/ (Erişim tarihi: 13/05/2015)

Yeditepe Üniversitesi, HIV'den korunma ve tedavi aralığını kapatma, <http://www.yeditepe.edu.tr/home/haberler.dot?storyId=331412> (Erişim tarihi: 13/05/2015)

World Health Organization (WHO), Kasım 2014, HIV/AIDS, <http://www.who.int/features/qa/71/en/> (Erişim tarihi: 13/05/2015)

Bursa, O., Sezer, E., Can, O. and Unalır, M. O., 2014, Using FOAF for Interoperable and Privacy Protected Healthcare Information Systems, Metadata and Semantics Research, 8th Research Conference, MTSR 2014, 154-161 pp.

He, Y. and Xiang, A., 2008, Vaccine Ontology, University of Michigan Medical School, <http://www.ontobee.org/browser/index.php?o=VO> (Erişim tarihi: 14/05/2015)

Simmhan, Y., L., Plale, B. and Gannon, D., 2005, A survey of data provenance in e-science. SIGMOD Rec., 34(3):31–36 pp.

Can, Ö., Bursa, O. and Ünalır, M., O., 2010, Personalizable Ontology-Based Access Control, Gazi University Journal of Science, 23(4):465-474 pp.

KAYNAKLAR DİZİNİ (devam)

Can, Ö., 2009, Anlamsal Web İçin Kişiselleştirilebilir Ontoloji Tabanlı Erişim Denetimi ve Politika Yönetimi, Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, 260s.

ÖZGEÇMİŞ

Genel

Adı Soyadı:	Dilek Yılmaz
Yazışma Adresi:	Bilgisayar Müh. Bölümü, Ege Üniversitesi, Bornova, İzmir
Doğum Tarihi ve Yeri	13.07.1990, Konak
Telefon	0533 771 21 90
E-Posta	dilekyilmazer@windowlive.com

Eğitim

Öğrenim Dönemi	Derece (*)	Kurum	Öğrenim Alanı
2013-Devam ediyor	Yüksek Lisans	Ege Üniversitesi	Bilgisayar Mühendisliği
2008-2012	Lisans (Onur Öğr.)	Ege Üniversitesi	Bilgisayar Mühendisliği
2004-2008	Lise	Bornova Anadolu Lisesi	Sayısal Bölüm

Mesleki Deneyim

Öğrenim Dönemi	Unvan	Kurum
2014-Devam ediyor	Kıdemli Yazılım Tasarım Mühendisi	Vestel Dijital A.Ş., Manisa
2012-2014	Yazılım Geliştirme Uzmanı	Veripark, İzmir
08.2011-09.2011	Stajyer	Üni-Pa A.Ş.
07.2011-08.2011	Stajyer	Vestel Elektronik A.Ş.

Yayınlar:

1. Can, Ö., **Yılmaz, D.**, 2014, A Privacy-Aware Semantic Model for Provenance Management, Metadata and Semantics Research: 8th Research Conference, CCIS 478, 162-169 pp.