EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

DAĞITIK BAĞLI VERİLERDE SORGULAMA YÖNETİMİ

Ziya AKAR

Tez Danışmanı: Prof. Dr. Oğuz DİKENELLİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu: 619.01.00
Sunuş Tarihi: 05.07.2012

Bornova-İZMİR
2012

Ziya AKAR tarafından YÜKSEK LİSANS tezi olarak sunulan "**DAĞITIK BAĞLI VERİLERDE SORGULAMA YÖNETİMİ**" 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 **05/07/2012** tarihinde yapılan tez savunma sınavında aday oy birliği/oy çokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:	**İmza**
Jüri Başkanı: Prof. Dr. Oğuz DİKENELLİ	.....
Raportör Üye: Yrd. Doç. Dr. R. Cenk ERDUR	.....
Üye: Yrd. Doç. Dr. Tuğkan TUĞLULAR	.....

ÖZET

DAĞITIK BAĞLI VERİLERDE SORGULAMA YÖNETİMİ

AKAR, Ziya

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Prof. Dr. Oğuz DİKENELLİ
Temmuz 2012, 85 sayfa

Sorgu işletimi, anlamsal veb üzerindeki verilere erişim için önemli bir yoldur. Günümüzde anlamsal veb birbirine bağlı verisetleri vebi olarak görülmektedir ve bu yüzden vebi sorgulamak tüm bu verisetlerini bir bütün olarak dikkate alıp sorgulama yapmak olarak görülebilir. Ayrıca vebi sorgularken dikkate alınacak olan verisetlerinin belirlenmesi sorgulayıcılardan saydam bir şekilde gerçekleştirilmelidir. Sorgulanacak verisetlerinin seçilmesi ve sorgulanmak üzere bir araya getirilmesi bu verisetlerinin üst verilerinin olmasını gerektirmektedir. Bu tez kapsamında VOID denilen veriseti üstverisini kullanarak sorgular ile ilişkili veriseti seçilimini kullanıcıların yerine gerçekleştiren WoDQA (Veri Vebi Sorgu Analizcisi) birleştirilmiş sorgu motoru tanıtılacaktır. WoDQA'nın analiz sürecinde bir SPARQL sorgusu için VOID belgelerinde yer alan veriseti ve bağseti tanımları analiz edilerek birleştirilmiş sorgular yaratılır. VOID'in bağseti kavramı sayesinde verisetleri arasındaki bağlar da ilişkili verisetlerinin belirlenmesinde kullanılmış olur. Ayrıca performanslı sorgu işletimi için bazı eniyileştirme adımları da gerçekleştirilmiştir. Ancak bu tez kapsamında WoDQA, veriseti üstverilerini dikkate alıp sorguları analiz ederek etkili bir şekilde veriseti eleme görevine odaklanmıştır. WoDQA sorgu işletimleri de farklı açılardan ele alınıp değerlendirilmiştir.

Anahtar sözcükler: Dağıtık Sorgulama, Birleştirilmiş Sorgular, Bağlı Veri, SPARQL, Anlamsal Veb.

ABSTRACT

QUERYING MANAGEMENT OF DISTRIBUTED LINKED DATA

AKAR, Ziya

MSc in Computer Engineering.
Supervisor: Prof. Dr. Oğuz DİKENELLİ
July 2012, 85 pages

Query processing is an important way of accessing data on the Semantic Web. Today, the Semantic Web is characterized as a web of interlinked datasets, and thus querying the web can be seen as dataset integration on the web. Also, this dataset integration must be transparent from the data consumer as if she is querying the whole web. To decide which datasets should be selected and integrated for a query, one requires a metadata of the web of data. In this paper, to enable this transparency, we introduce a federated query engine called WoDQA (Web of Data Query Analyzer) which discovers datasets relevant with a query in an automated manner using VOID documents as metadata. While WoDQA analyzing queries, dataset and linkset descriptions in VOID documents are analyzed for a SPARQL query and a federated query is constructed. By means of linkset concept of VOID, links between datasets are incorporated into selection of federated data sources. Furthermore some query optimization steps were implemented to execute queries with high performance. But in this thesis, WoDQA focuses on powerful dataset elimination by analyzing query structure with respect to the metadata of datasets. Some query executions on WoDQA were evaluated with different perspectives.

Keywords: Distributed Querying, Federated Queries, Linked Data, SPARQL, Semantic Web.

TEŞEKKÜR

Eğitim ve tez çalışmam sürecinde bana verdikleri destekten dolayı Prof. Dr. Oğuz Dikenelli ve Erdem Eser Ekinci'ye, her durumda eğitimimi destekleyen ve manevi desteğini esirgemeyen aileme, bana sağladıkları çalışma ortamı için Tayfun Gökmen Halaç, Zehra Gül Çabuk, Esin Gül Karabacakoğlu, Burak Yönyül ve Pınar Göçebe'ye ve yüksek lisans eğitimim için bana burs sağlayan Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) Bilim İnsanı Destekleme Daire Başkanlığı'na (BİDEB) ve 1001 Araştırma Projeleri kapsamında bana sağladığı destekten dolayı Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) Araştırma Destek Programları Başkanılığı'na (ARDEB) teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama

Kısaltmalar

RDF	Resource Description Framework (Kaynak Tanımlama Çerçevesi)
BGP	Basic Graph Pattern (Temel Çizge Deseni)
VOID	Vocabulary of Interlinked Dataset
IRI	Internationalized Resource Identifiers
URI	Uniform Resource Indentifier

1 GİRİŞ

Günümüzde veb, yapısal bir veri uzayı üzerinde gelişmeye devam ederken, pek çok uygulama da verilerini yayımlamaya ve bağlamaya devam etmektedir. Gün geçtikçe de bu bağlı veri bulutu devasa bir hale ulaşacaktır. Bu bağlı ve yapısal veri uzayı içerisinde sorgu çalıştırabilmek önemli bir araştırma alanı haline gelmiştir. Literatürde de farklı sorgu çalıştırma yaklaşımları ve araçları görülmektedir (Hartig et al., 2009; Görlitz and Staab, 2011). Veri vebi üzerinde sorgu çalıştırımı, temel olarak ihtiyaçlarımızı karşılayacak olan kaynakların bulunması demektir. Fakat asıl sorun bu büyük veri uzayının neresinde ihtiyacımız olan kaynakları arayacağımızdır. Kaynakları nerede arayacağımızı etkin olarak belirleyebilmek için hangi kaynakların hangi verisetlerinde kullanıldığının ve hangi verisetlerinin hangilerine bağlı olduğunun dikkate alınması gerekmektedir. Eğer veriseti yayımlayıcıları verisetlerinin içeriklerini yansıtan üstverileri sağlarlarsa sorgular ile ilişkili olan verisetleri etkin bir şekilde seçilebilir. Veriseti ile ilgili bu üstverileri sağlamak için *Vocabulary of Interlinked Datasets (VOID)* (Alexander et al., 2009) denilen bir sözlük W3C tarafından Semantic Web Interest Group notu olarak yayınlanmıştır[1]. VOID, RDF verisetlerinin üstverilerinin tanımlanması için tasarlanmış bir RDF sözlüğüdür. Bir bakıma veri vebinin üstverisi de sayılabilir. Verisetlerinin birer düğüm ve aralarındaki bağların da birer köprü olarak kabul edildiği durumda bağlı açık veri bulutu, bir verisetleri çizgesi olarak temsil edilir. VOID, veri vebinin çizge tabanlı ruhuna uygun olduğundan dolayı verisetlerinin üstverilerini tanımlamak için çok güçlü bir yöntemdir. Bu yöntemle tanımlanan verisetleri, sorguların verisetleri üzerinde dağıtık olarak çalıştırılması amacıyla kolayca keşfedilebilir.

Bu tez kapsamında, verisetlerinin tanımlarını oluşturan VOID üstverileri kullanılarak bağlı açık veri bulutundaki verisetleri üzerinde dağıtık olarak sorgular çalıştırmak amacıyla geliştirilmiş olan birleştirilmiş sorgu motoru WoDQA (Web of Data Query Analyzer) tanıtılacaktır. WoDQA sorgulanacak verisetlerinin etkin bir biçimde seçilmesine ve sorgu yapısının analizine odaklanır ve böylece sorgu ile ilişkisiz verisetleri sorgulanmadan elenirler. Hangi verisetinin bir sorgu için sonuçlar içerebileceği ve verisetlerinin birbiri ile ilişkili olma durumları VOID belgelerinin analizi ile mümkün olmaktadır.

[1] http://www.w3.org/TR/void/

Sorguların hangi verisetleri üzerinde çalıştırılacaklarının seçimini kullanıcıların yerine yaparak sorguları çalıştırabilen iki temel yaklaşım vardır. Birinci yaklaşım *bağ dolanımı* (Hartig et al., 2009) olarak bilinir. Bu yaklaşım veriler arasındaki bağları takip ederek olası tüm ilişkili verileri bulmaya dayalıdır. Diğer yaklaşım ise *sorgu federasyonu* (Görlitz and Staab, 2011) yaklaşımı olarak bilinir. Bu yaklaşım da verisetlerinin üstverilerinden faydalanarak bir sorguyu alt sorgulara bölüp alt sorguları ilişkili verisetlerine dağıtarak çalıştırmayı amaçlamaktadır.

Bağ dolanımı yaklaşımı, vebi verisetleri olarak değil de çözümlenebilir URI'ler içeren RDF belgeleri çizgesi olarak kavramsallaştırır. Bu yaklaşım sorgu içerisindeki kaynaklardan yola çıkarak esas olarak kaynakların bulunduğu RDF belgeleri içerisindeki kaynaklar arasındaki bağları kullanır. Fakat bu yöntem tamlık[2] ve performans sorunlarını arttırmaktadır. Farklı sorgu tiplerini cevaplayabilmek için bazen sezgisel sorgu planlama yöntemleri kullanılabilmesine rağmen (Hartig, 2011), bu yaklaşım sorgu için tüm sonuçların bulunmasını garanti etmez. Çünkü sorgulanacak RDF belgeleri sorgulamanın başlatılacağı kaynağa göre farklılık gösterebilirler. Ayrıca sorgulanacak dökümanlar çok büyük olabilir ve büyüklüğünden dolayı sorgulanmak için getirilmesi sorun olabilir.

Sorgu federasyonu yaklaşımı ise veritabanı literatüründeki farklı veritabanları üzerinde çalıştırılmak üzere oluşturulan birleştirilmiş sorgular fikrine dayanan bir yaklaşımdır (Sheth and Larson, 1990). Bu yaklaşım, bir sorgu işletilmeden önce iki ana işlem gerçekleştirir. İlk olarak verisetlerinin içeriklerini yansıtan üstveriler kullanılarak sorgu ilişkili verisetlerinden sorgulanmak üzere alt sorgulara parçalanır. Daha sonra da değerlendirme planı, veriseti ile ilgili bazı istatistikler ve bazı yöntemler kullanılarak eniyileştirilmeye çalışılır. Alt sorguların dağıtık veri kaynakları üzerinde çalıştırılması için veri kaynaklarının SPARQL uçnoktalarının kullanılması gereklidir. Sorgu federasyonu yaklaşımı tüm veriseti üstverilerinin tam ve doğru olduğu varsayımı altında, bağ dolanımı yaklaşımının aksine, sorguları tüm ilişkili verisetleri üzerinde çalıştırarak tam bir sonuç alınmasını sağlar.

Bu fikirlerin ışığında, WoDQA, veri vebinin adeta bir iz düşümü gibi olan VOID belgelerini analiz ederek sorguları çalıştıran bir sorgu federasyonu aracıdır. Ayrıca üstverilerde tanımlı olan verisetleri arasındaki bağları da kullanarak bağ dolanımı yaklaşımını sorgu federasyonu yaklaşımına bir nevi dahil etmiş olur. Bu

[2]Tamlık, sorgulama sonuçlarının tümünün bulunması anlamına gelmektedir.

tez kapsamında sorgular için bazı eniyileştirme adımlarının da gerçekleştirilmiş olmasıyla birlikte asıl ortaya konan tüm ilişkisiz verisetlerinin elenerek sadece ilişkili verisetlerinin belirlenmesi fikridir. Şu anda var olan sorgu federasyonu yaklaşımları ilişkili verisetlerini belirlerken sadece verisetinin içerisinde kullanılan kaynaklar ile ilgili bilgilerden faydalanmaktadır. WoDQA'nın sorgu federasyonu yaklaşımına en önemli katkısı, sorgulanacak ilişkili verisetlerini belirlerken VOID belgeleri içerisindeki *bağseti* tanımlamaları sayesinde veriler arasındaki bağları ve üçlü desenleri arasındaki ilişkileri de dikkate almasıdır. WoDQA hem bir SPARQL uçnoktası, hem indirilebilir bir API, hem de basit bir veb sayfası[3] olarak kullanıma sunulmuştur. VOID depolarındaki veriseti üstverileri kullanılarak ham sorgular dağıtık bir şekilde çalıştırılabilir.

Tezin ilerleyen bölümleri şu şekilde düzenlenmiştir: Bölüm 2'de WoDQA mimarisini anlatmak için gerekli bazı temel bilgiler verilmiştir. Bölüm 3'te WoDQA sorgulama motoru ve modülleri ayrıntılı olarak açıklanmıştır. Ardından Bölüm 4'te ise WoDQA'nın bazı sorgular üzerinde çalıştırılarak değerlendirilmesi yapılmıştır. Bölüm 5'te literatürde yer alan bazı sorgulama araçlarından söz edilmiştir. Son olarak Bölüm 6'da tez sonuçlandırılmaktadır.

[3]WoDQA veb sayfası, güncel SPARQL uçnoktası adresi ve WoDQA API'sine erişmek için güncel bilgiler http://seagent.ege.edu.tr/etmen/wodqa.html adresinde bulunabilir.

2 ALTYAPI

Bu bölüm tezin altyapısını oluşturan bazı konular hakkında tezin daha iyi anlaşılabilmesi için gerekli olan bazı genel bilgiler ve tanımlamalar vermeyi amaçlamaktadır. RDF kavramı, bağlı veri kavramı, veriseti tanımlama belgeleri olan VOID ontolojileri ve içerikleri, anlamsal verilerin sorgulama dili olan SPARQL sorgularının öğelerinin genel tanımı ve veriseti gibi temel kavramların biçimsel tanımları bu bölümde belirtilecektir.

2.1 RDF

Bu tez kapsamında sunulan WoDQA sorgulama motorunun çalıştığı veriler anlamsal veb üzerindeki veriler olduğu için öncelikle bu verilerin nasıl tanımlandığını belirtmeliyiz. *RDF (Resource DescriptionFramework),* veb üzerinde kaynak tanımlama dilidir[1]. Kaynak denilen kavram ise veb üzerinde hakkında bazı bilgiler tanımlanabilen her şey olabilir. Bir kaynak tanımlanırken *özne, yüklem* ve *nesne* olarak adlandırılan üç öğe kullanılır ve bu üç öğe. *RDF üçlüsü* denilen kavrama denk gelmektedir. Bir RDF üçlüsü kavram biçimsel olarak $< s, p, o > \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ şeklinde gösterilir. Burada $\mathcal{I}$ IRI'ler (*Internationalized Resource Identifiers*)(Duerst and Suignard, 2005) kümesi, $\mathcal{B}$ boş düğümler (blank nodes) kümesi, $\mathcal{L}$ ise literaller kümesidir. Bütün RDF kavramları da $\mathcal{T} = \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ ile temsil edilir. Bu kaynak tanımlama yapısı sayesinde bir kaynak ile ilişkili olan herşey yeni bağlar yolu ile tanımlanabilir ve birbirine bağlı verilerden oluşan büyük bir çizge oluşturulur. Bölüm 2.2'de RDF ile oluşturulan veb üzerindeki bağlı veriler daha detaylı olarak anlatılmaktadır. Örneğin <http://dbpedia.org/resource/Bursa> <http://dbpedia.org/ontology/name> "Turkey"@en. RDF üçlüsünün öznesi olan <http://dbpedia.org/resource/Bursa> kaynağının isim özelliği <http://dbpedia.org/ontology/name> yüklemi kullanılarak tanımlanmıştır.

2.2 Bağlı Veri Vebi

Veb, bilgi yayımlanmasının ve paylaşımının en etkili şekilde gerçekleştirilmesine olanak vermektedir. Bu paylaşım klasik vebde veb sayfaları ile gerçekleştirilirken

[1] http://www.w3.org/RDF/

sadece insanların bu veb sayfalarını okuması ile mümkün olmaktadır. Ayrıca veb sayfaları arasında gezinebilmek sadece bir sayfadan diğerine bir veb sayfası bağı ile olmaktadır. İki veb sayfası arasındaki bağın da bir anlamı yoktur. Aslında kullanıcılar bir sayfa içerisindeki kaynaklar ile ilgili bilgilere erişmek isterler. Bu nedenle bağların sayfalar arasında değil kaynaklar arasında olması daha anlamlıdır. Bu bağın bir anlamının olması yani veri ile ilişkisinin ne olduğunun belirtilmesi veb üzerinde yeni kaynakların keşfinin daha kolay olmasını sağlayacaktır. Bu amaçla Tim Berners-Lee ilk defa *bağlı veri* kavramını ortaya atmış ve anlamsal vebin temelini oluşturan bağlı veri ilkelerini(Berners-Lee, 2006) yayımlamıştır. Bu ilkelere göre veb üzerinde yayımlanan bağlı verinin özellikleri şunlardır:

1. Her kaynağı eşsiz olarak tanımlamak için *URI (Berners-Lee et al., 1998)* denilen standart tanımlayıcılar kullanılmalıdır (Kaynaklar için bu olmadığı durumda anlamsal vebin varlığından söz edilemez).

2. Hem insanların hem de makinelerin de bu kaynak tanımlayıcıları okuyabilmesi için *HTTP URI*'leri (Fielding et al., 1999) kullanılmalıdır (Çünkü HTTP URI'leri kaynakların sunucular tarafından sağlandığı anlamına gelir ve makineler tarafından da erişilebilirler).

3. URI'lere erişilmek istendiğinde RDF/XML biçiminde kaynak ile ilgili bilgilere de erişilmelidir.

4. Kaynakların veb üzerindeki diğer kaynaklara bağlı olarak bulunmalıdır.

Bu ilkelerin gerçekleştirilmemesi hiç birşeyi yok etmez ancak var olmaları tam anlamıyla bağlı veri yaratılmış olmasını sağlar ve veri erişimini ve keşfini kolaylaştırır(Berners-Lee, 2006).

Bağlı veri, yapısal veriyi veb üzerinde yayımlamanın ve verileri birbirine bağlamanın en iyi yöntemidir(Bizer et al., 2009). Veb üzerinde birbirine bağlanmış bu veriler ise dev bir RDF çizgesi olan *veri vebi*ni oluşturur(Heath and Bizer, 2011). Temelde bir RDF çizgesi ($\mathcal{G}$) biçimsel olarak RDF üçlüleri kümesi ile temsil edilir: $\mathcal{G} = \{\langle s, p, o\rangle \mid < s, p, o > \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})\}$. Bu doğrultuda veri vebi de veb üzerindeki tüm IRI'ler, boş düğümler, ve literallerden oluşan geniş çaplı bir çizgedir($\mathcal{G}_{wod}$). $\mathcal{G}_{wod}$, veri vebi için matematiksel bir çizge yapısı modelidir.

Bir diğer açıdan, veri vebi birbirine bağlı verisetleri vebidir(Bizer et al., 2007). Bir veriseti (δ), anlamlı RDF üçlüleri(Alexander et al., 2009) kümesinden oluşur ve

vebin daha büyük parçacıklı olmasını sağlar. Verisetleri, kaynakları tek başına yayımlayıp onları birbirine bağlamak yerine $\mathcal{G}_{wod}$'un alt çizgeler olarak inşa edilmesini ve bu alt çizgelerin birbirine bağlanması ile yayımlanmasını sağlarlar. Bu alt çizgeler, yani verisetleri, farklı verisetlerindeki kaynakları birbirine bağlayan RDF üçlüleri sayesinde birbirine bağlanırlar(Heath and Bizer, 2011). Veriseti yayımlayıcıları kaynaklarını yaratırlar ve kullanıcılar da kendi verisetlerini yaratırken bu kaynakları kullanırlar. Buna göre, bir verisetini biçimsel olarak tanımlamadan önce bir veriseti içerisinde yer alan üçlülerin öznelerini oluşturan kaynak kümesini temsil eden $subj\left(\mathcal{G}\right)$'nin biçimsel olarak tanımlanması gerekir. Verisetinin biçimsel olarak tanımlanması Bölüm 2.3'te VOID tanımları ile birlikte verilecektir.

Tanım 1 *Bir veriseti, veri vebinin bir alt çizgesidir, $\delta_x \subset \mathcal{G}_{wod}$, ve δ_x tarafından içerilen kaynaklar şu şekilde belirtilir:* $\forall r, \delta_i \left(r \in subj\left(\delta_x\right) \rightarrow owner\left(\delta_x, r\right)\right)$.

Tanım 1'de kaynakların veb üzerindeki verisetleri üzerinde nasıl tanımlandığı belirtilmiştir. Şimdi de bağlı verinin asıl yapısını oluşturan veriler arasındaki bağlar biçimsel olarak tanımlanacaktır. Nesneleri başka bir verisetinde tanımlı bir kaynak olan RDF üçlüleri, veri vebini birbirine bağlı verisetlerinden oluşan bir çizge haline getirir. Böyle üçlülere *bağ üçlüleri* denir ve $\mathcal{LT} = \{(s,p,o) \,|owner\left(s\right) \neq owner\left(o\right)\}$, $s, o \in \mathcal{I}$ *iken* şeklinde biçimselleştirilirler. Bu biçimsel tanımda yer alan yüklem özel bir yüklemdir ve *bağ yüklem* $\left(p^{link}\right)$ olarak adlandırdığımız bu yüklemler farklı verisetleri içerisindeki verileri birbirine bağlarlar. Tüm bağ üçlülerinde kullanılan bağ yüklemler kümesi $\left(\mathcal{P}^{link}\right)$ şu şekilde biçimselleştirilir: $\mathcal{P}^{link} = \{p^{link} | \exists \left\langle s, p^{link}, o \right\rangle \in \mathcal{LT}\}$.

2.3 VOID

VOID (Alexander et al., 2009), RDF verisetlerinin tanımlanması için geliştirilmiş, verisetlerinin içeriklerini ve verisetleri arasındaki bağları tanımlamaya yarayan bir üstveri modelidir.

VOID belgelerinde bir veriseti tanımlanırken bazı özellikler[2] kullanılır. Bu tez kapsamında dikkate aldığımız veriseti özellikleri ile bir veriseti şu şekilde biçimsel olarak tanımlanır: $\left\langle \mathcal{L}^{space}, \mathcal{I}^{voc} \right\rangle \in \mathcal{L} \times \mathcal{I}$. Verisetinin ilk özelliği, bir verisetindeki tanımlı olarak bulunan kaynakları yani üçlülerindeki özneleri temsil eden IRI'lerin

[2]Bu tez kapsamında sadelik açısından yalnızca void:sparqlEndpoint, void:uriSpace ve void:vocabulary özellikleri dikkate alınmıştır. İleriki çalışmalarda istatistikler gibi bilgiler de kullanılacaktır.

sahip oldukları ortak isim uzayını temsil etmek için kullanılan void:uriSpace özelliğine karşılık gelen $\mathcal{L}^{space}$'dir. Diğeri ise veriseti tarafından kullanılan sözlükleri temsil etmek için kullanılan void:vocabulary özelliğine karşılık gelen $\mathcal{I}^{voc}$'dur[3]. Örneğin DBpedia verisetindeki dbpedia:Turkey dpedia-owl:name "Turkey"@en. üçlüsü var olduu için DBpedia verisetinin VOID belgesinde dbpedia-owl, void:vocabulary özelliği olarak dbpedia, void:uriSpace özelliği olarak tanımlı olmalıdır.

Farklı verisetleri içerisindeki veriler arasındaki bağları da VOID'in en önemli kavramlarından biri olan *bağseti* kavramı ile tanımlanmaktadır. Bağseti, bir verisetinde tanımlı olan kaynakları farklı bir verisetinde tanımlı olan kaynaklara aynı bağ yüklem ile bağlayan bağ üçlüleri kümesini ifade eder ve şu şekilde biçimselleştirilir: $\lambda = \left\langle \delta_\lambda^{from}, \delta_\lambda^{to}, p_\lambda^{link} \right\rangle \in \Delta \times \Delta \times \mathcal{P}^{link}$. Burada Δ veb üzerindeki tüm verisetlerini temsil eder. δ_λ^{from} bağ üçlülerinin öznelerinin tanımlı olduğu *kaynak veriseti*, δ_λ^{to} ise bağ üçlülerinin nesnelerinin tanımlı olduğu *hedef veriseti*ni temsil eder. p_λ^{link} ise bu bağ üçlüleri için tanımlı olan bağ yüklemdir ve void:linkPredicate kavramına karşılık gelir.

Veriseti yayımlayıcıları kendi verisetlerinin VOID belgelerini oluştururlar ve anlamsal web programcıları bu belgelere voiD Browser[4], CKAN[5], voiDStore[6], MONDECA[7] ve LOD-CLOUD[8] gibi VOID depoları üzerinden erişirler. Bir VOID deposu bağlı veri bulutunun bir izdüşümü gibidir. Bundan dolayı veriseti yayımlayıcıları verisetlerinin, sorgular analiz edildiğinde keşfedilip dikkate alınması için VOID belgelerini gerçekten verisetinin içeriğini yansıtacak bir biçimde oluşturmalıdırlar ve verisetleri içerisindeki verilerin diğer verisetlerindeki veriler ile bağlı olmasına dikkat etmelidirler.

2.4 SPARQL

SPARQL, anlamsal vebi oluşturan RDF verilerinin sorgulanması için kullanılan bir sorgulama dilidir[9]. Bu tezde veri vebinin sorgulanması amacıyla *temel çizge desen*ine denk gelen bazı SPARQL sorguları dikkate alınmıştır(Buil et al., 2011). Bir temel

[3]RDFS, OWL gibi Anlamsal Veb dilinin şemaları $\mathcal{I}^{voc}$ kümesine dahil edilmez.
[4]http://kwijibo.talis.com/voiD/
[5]http://ckan.net/
[6]http://void.rkbexplorer.com/
[7]http://labs.mondeca.com/endpoint/ends
[8]http://dsi.lod-cloud.net/sparql
[9]http://www.w3.org/TR/rdf-sparql-query/

çizge deseni üçlü desenlerinden oluşur, $BGP = \{tp_i | \langle s_{tp_i}, p_{tp_i}, o_{tp_i} \rangle \in (\mathcal{I} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})\}$. Bir üçlü deseni bir RDF üçlüsünden farklıdır çünkü içerisinde en az bir tane *değişken* içerir. Değişkenler sonsuz $\mathcal{V}$ kümesinin elemanlarıdır[10]. Sorgu işletilirken bir değişkenin yerine ona karşılık gelen RDF kavramları gelmektedir. Bu tanımlara göre bir sorgunun sonucu, çözüm eşleşmeleri kümesidir: $\{\mu | \mu : \mathcal{V} \rightarrow \mathcal{T}\}$, burada μ değişkenlerden RDF kaynaklarına kısmi bir fonksiyondur.

Bu tezin amacı ham sorguların analiz edilerek ilişkili verisetlerini sorgulayacak olan birleştirilmiş sorgular oluşturmaktır. Birleştirilmiş sorgular birden fazla temel çizge deseninin ilişkili verisetlerinin sorgulanması amacıyla bir araya getirilmesi ile oluşturulur. SPARQL, SERVICE[11] anahtar kelimesi sayesinde birleştirilmiş sorgular yazılmasına imkan verir. Bir birleştirilmiş sorgu içerisindeki her temel çizge deseni bazı verisetlerinin sorgulanması amacıyla *servis çizge deseni* denilen yapılar altında yer alırlar. Servis çizge desenleri detaylı olarak Bölüm 3.3'te açıklanacaktır.

[10] Boş düğümler yok sayılmıştır.

[11] http://www.w3.org/TR/sparql11-federated-query/

3 WODQA SORGULAMA MOTORU

Bu bölümde WoDQA sorgu işletim mimarisi detaylı bir şekilde ele alınacaktır. Bir sorguyu web üzerinde yayınlanmış olan tüm verisetlerinden sorgulamak pratik olarak mümkün olmadığından dolayı WoDQA, sorguyu sadece ilişkili verisetleri üzerinde çalışacak birleştirilmiş bir sorguya dönüştürmeyi amaçlar. Bu doğrultuda, bir sorguyu bağlı veri bulutunda çalıştırabilmek için WoDQA Şekil 3.1'de görüldüğü gibi dört ana modül içerir: Veriseti Analizcisi, Sorgu Eniyileştirici, Sorgu Düzenleyici ve Jena ARQ sorgu motoru[1].

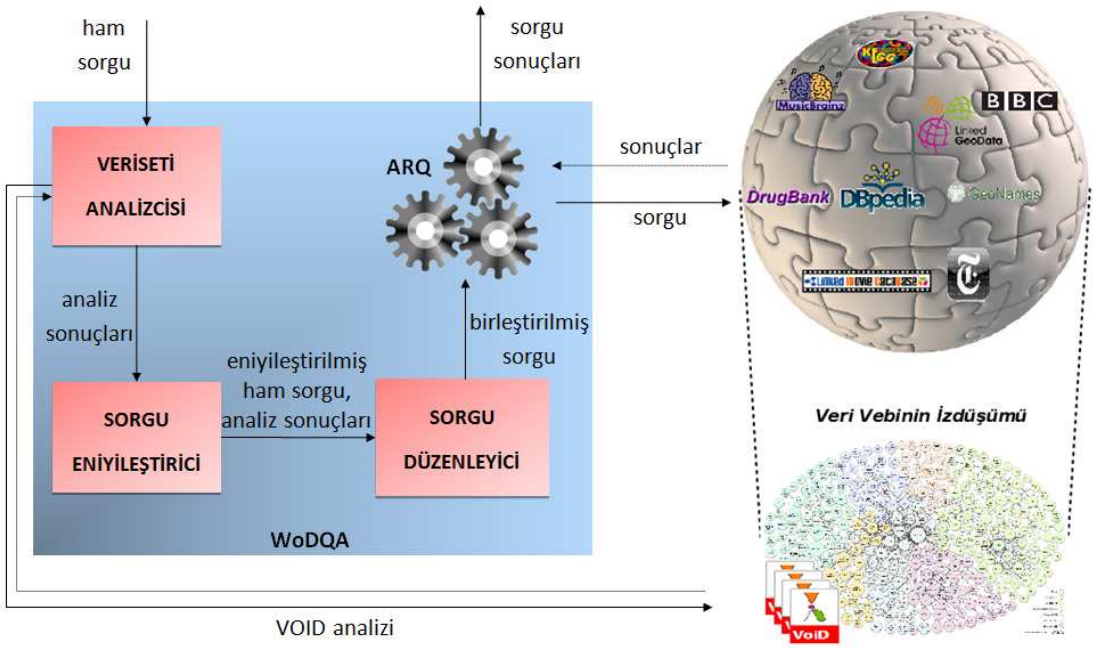
Veriseti Analizcisi, VOID depolarındaki verisetlerinin VOID belgelerini kullanarak ilişkili verisetlerinin keşfedilmesinden ve ilişkisiz olanların elenmesinden sorumlu olan modüldür. Bu tez kapsamında VOID depolarındaki VOID belgelerinin veriseti yayımlayıcıları tarafından güncellendiği varsayılmıştır. Verisetlerini tanımlayan bu VOID belgelerindeki veriseti ve bağseti tanımları her üçlü deseni için analiz edilerek ilişkili verisetleri bulunmaya çalışılmıştır. VOID belgelerinin verisetleri için tam ve doğru olduğu varsayımı altında, veriseti analizi, sorgulamaya kesinlikle katkıda bulunmayacak olan bütün verisetlerini eler. Veriseti analizi kural tabanlı bir yaklaşım ile gerçekleştirilir. İlişkili verisetlerini bulan bu kurallar bölüm 3.1.1'de açıklanacaktır. Ayrıca VOID belgelerinden ziyade ASK sorguları ile de verisetlerinin ilişkili veriseti olup olmadıklarının kontrolü analiz aşamasında gerçekleştirilmektedir. ASK sorguları analizi Bölüm 3.1.1.2'de anlatılmaktadır.

Bu tez kapsamında sorgu analizi aşamasından sonra bir takım sorgu eniyileştirmeleri yapılmaktadır. Sezgisel yöntemlerle ve bazı istatistiksel verilere dayanarak yapılan bu eniyileştirme adımları WoDQA'nın ikinci modülü olan Sorgu Eniyileştiricisi tarafından gerçekleştirilmektedir ve Bölüm 3.2'de anlatılmaktadır.

Üçüncü modül, veriseti analiz sonuçlarına göre sorguları yeniden yazan Sorgu Düzenleyici'dir. Bu yeniden yazım sürecinde, içerisinde SERVICE ifadeleri bulunduran birleştirilmiş SPARQL sorguları yaratılır. Sorgu Düzenleyici Bölüm 3.3'te detaylı olarak açıklanacaktır.

Son modül, Sorgu Düzenleyici tarafından içerisine SERVICE ifadeleri eklenmiş SPARQL sorgularını çalıştırmak için doğrudan Jena ARQ sorgu motorunu kullanan Sorgu Çalıştırıcı'dır. Sorgular çalıştırıldıktan sonra sonuçları sorgulayıcıya

[1] http://jena.sourceforge.net/ARQ/



Şekil 3.1: WoDQA içsel mimarisi

döndürülür. Bundan sonraki alt bölümlerde Veriseti Analizcisi, Sorgu Eniyileştirici ve Sorgu Düzenleyici modülleri detaylı olarak anlatılmıştır.

3.1 Veriseti Analizcisi

Bu bölümde WoDQA'nın özgün yanını oluşturan Veriseti Analizcisi modülünden bahsedilecektir. Diğer sorgu federasyonu yaklaşımlarından farklı olarak, WoDQA sorgulanacak verisetlerini belirlerken sorgu içerisindeki üçlü desenleri arasındaki ilişkileri ve verisetleri arasındaki bağları dikkate almaktadır. WoDQA'nın veriseti analizi sayesinde pek çok ilişkisiz veriseti elenirken ilişkili olan verisetleri belirlenmektedir. Veriseti analizi sonucunun çıktısı veri vebi üzerinde yayımlanmış olan tüm verisetlerinin bir alt kümesidir ve sorgu sadece ilişkili verisetlerini içeren bu alt küme üzerinde çalıştırılır.

Veriseti Analizcisi, ilişkili verisetlerinin seçilmesi için diğer bir deyişle sorgularla alakalı sonuç içerebilecek alt çizgelerin seçilmesi için VOID belgelerindeki hem veriseti tanımlamalarını hem de bağseti tanımlamalarını kullanır. VOID belgeleri içerisindeki veriseti ve bağseti tanımlarınının analiz edilmesinin yanısıra sorgudaki üçlü desenlerinin kendi aralarındaki ilişkiler de dikkate alınır.

Bir sorgunun, hiçbir analizin yapılmadığı durumda veri vebi üzerinde çalıştırabilmesi için sorgu içerisindeki her üçlü deseninin veb üzerinde yayımlanmış tüm

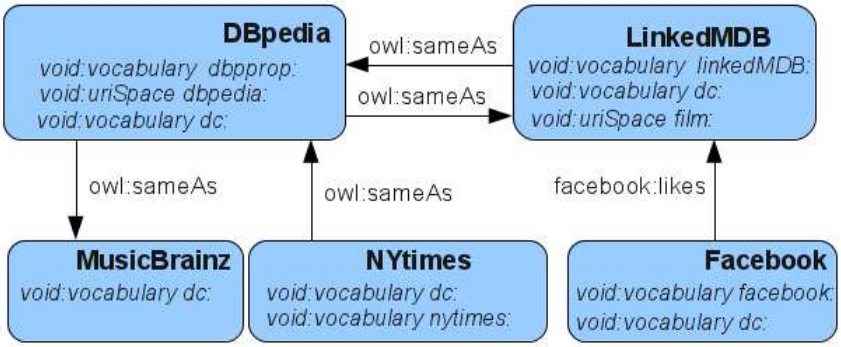
verisetleri üzerinde sorgulanması gerekir. Ancak bu veb üzerindeki verisetlerinin çokluğundan dolayı mümkün olmayacağı için WoDQA'nın analiz sürecinde ilişkisiz verisetlerinin elenmesi amaçlanmaktadır. Böylece oluşturulan birleştirilmiş sorgudaki alt sorgular sadece ilgili verisetlerine dağıtılır ve sorgu çalıştırılır. İlişkisiz verisetlerinin elenebilmesi için bu tezde ilişkili verisetlerini ortaya çıkaran *ilişkili veriseti keşif kuralları* olarak adlandırılan bazı kurallar tanımlanmıştır. Bir üçlü deseni için ilişkili olan bir veriseti $\rho(\delta_x, tp_i)$ ilişkili veriseti belirtimi ile gösterilir ve burada δ_x verisetinin tp_i üçlü deseni için bir sorgulama sonucu içerebileceği anlamına gelmektedir. Veriseti keşif kuralları da bu ilişkili veriseti belirtimlerinin bulunmasını sağlarlar.

Bu noktada veriseti keşif kuralları sayesinde ilişkisiz verisetlerinin nasıl elendiği açıklanacaktır. $\mathcal{Q}_{tp_i}$, tp_i ile ilişkili olan tüm verisetlerinin tutulduğu kümeyi temsil etmektedir ve bu kümenin herhangi bir eleme yapılmadan önce veb üzerinde yayımlanan tüm verisetlerini içerdiği varsayılır, yani ilk durumda $Q_{tp_i}^{init} \equiv \Delta$ olarak gösterilir. Her bir veriseti keşif kuralı Q_{tp_i} içerisindeki verisetlerini dikkate alarak işletilir. Her bir kural uygulandıktan sonra ilişkili veriseti belirtimleri çıkarsanır. Çıkarsanan bu belirtimlerdeki verisetleri dışındaki verisetleri kesinlikle ilişkisizdirler ve Q_{tp_i} kümesinden uygulanan kural için çıkarılarak elenmiş olurlar. Fakat her zaman bir kuralın uygulanması ile ilişkili veriseti çıkarsanmayabilir. Bu durumda ilişkili olan bir veriseti bulunmadığı için Q_{tp_i} kümesindeki verisetleri için ilişkisiz olma durumu da çıkarsanmaz ve bir eleme yapılmaz. Bir kural uygulandıktan sonra Q_{tp_i}'nin *ilişkisiz verisetlerini eleme metodu*yla güncellenerek ilişkisiz verisetlerinden arındırılması şu şekilde biçimselleştirilmiştir:

Kural 1 *Bir keşif kuralı uygulandıktan sonra en az bir ilişkili veriseti bulunduysa, bulunan bu ilişkili verisetlerinden $\mathcal{Q}_{tp_i}$ kümesinde olanlar $\mathcal{Q}_{tp_i}^{new}$ kümesinin elemanıdır, $\mathcal{Q}_{tp_i}$ kümesindeki ilişkisiz verisetleri elenmiş olurlar. Ancak uygulanan keşif kuralı ile hiç bir ilişkili veriseti belirtimi bulanamıyorsa, $\mathcal{Q}_{tp_i}^{new}$ kümesi aynen $\mathcal{Q}_{tp_i}$ olarak kalır çünkü elenecek ilişkisiz veriseti de bulunmamış olur.*

$$\exists \delta_x \left(\rho\left(\delta_x, tp_i\right)\right) \Rightarrow \mathcal{Q}_{tp_i}^{new} = \{\delta_x | \left(\delta_x \in \mathcal{Q}_{tp_i}\right) \wedge \rho(\delta_x, tp_i)\}$$
$$\forall \delta_x \left(\neg\rho\left(\delta_x, tp_i\right)\right) \Rightarrow \mathcal{Q}_{tp_i}^{new} = \mathcal{Q}_{tp_i}.$$

Bundan sonraki bölümlerde veriseti keşif kuralları ayrıntılarıyla ele alınacaktır ve kullanımları bazı örnek sorgular üzerinde gösterilecektir. Şekil 3.2, veriseti keşif kurallarının bazı örnek sorgular üzerinde uygulanmaları için kullanılacak olan verisetlerinin VOID modellerini göstermektedir. Buradaki VOID belgeleri beş veriseti için



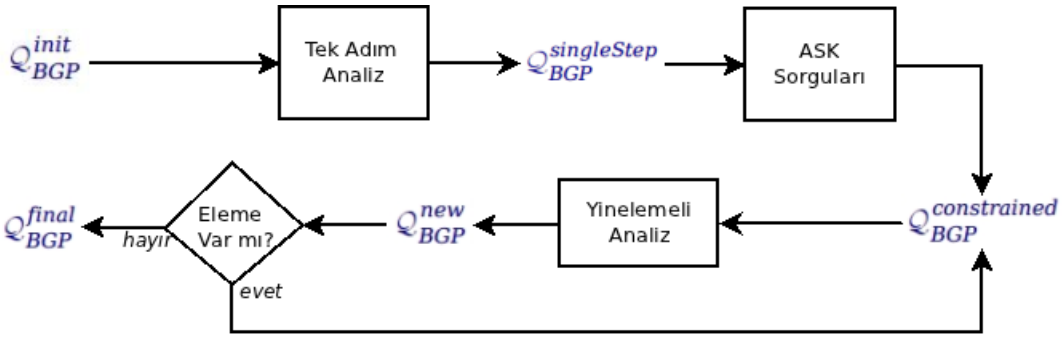
Şekil 3.2: Örnek VOID modelleri

tanımlanmıştır ve sadece WoDQA tarafından kullanılan özellikler gösterilmiştir. Şekil 3.2'de verisetleri dörtgenler ile, bağsetleri ise üzerinde bağ yüklemleri belirtilmiş oklar ile temsil edilmiştir. Örnek VOID modelinde bir de bizim yarattığımız Facebook kullanıcılarının verilerini tutan Facebook veriseti için de bir VOID belgesi vardır. Bu kullanıcı verileri, hangi Facebook kullanıcılarının LinkedMDB'de kayıtlı olan hangi filmleri sevdiği ile ilgilidir. Gerçekte verisetleri arasında pek çok bağseti tanımlı olmasına rağmen kurallarımızı basitçe açıklayabilmemiz için sadece Şekil 3.2'de gösterilenlerin olduğu varsayılacaktır.

3.1.1 İlişkili veriseti keşif kuralları

Bu bölümde, ilişkili verisetlerinin etkin olarak belirlenmesinde kullanılan keşif kurallarından söz edilecektir. Her bir keşif kuralı, verisetlerini farklı bakış açılarından analiz edip üçlü desenlerinin bu verisetlerinden sorgulanmak üzere ilişkili olup olmadıklarına karar vermekte ve ilişkili veriseti belirtimlerini (ρ) çıkarsamaktadır. Keşif kurallarının ele alındığı bakış açıları üç grupta toplanabilir. İlk bakış açısı olan *IRI Tabanlı Analiz*de, üçlü desenleri içerisindeki IRI'ler analiz edilir. Sorgulardaki özne ve nesne konumundaki IRI'lerin isim uzayları ve üçlü desenlerinde IRI olarak bulunan yüklemler ile VOID belgelerindeki void:uriSpace ve void:vocabulary tanımlamaları analiz edilip verisetlerinin üçlü desenleri için ilişkili olup olmadıklarına karar verilir. İkinci bakış açısında birbirine bağlı kaynaklar dikkate alınmaktadır. *Bağ Analizi* denilen bu bakış açısında birbirine bağlı iki kaynağın aynı verisetinde ya da farklı verisetlerinde olmalarına göre farklı değerlendirmeler yapılır. Son bakış açısı *Paylaşılan Değişken Analizi'*dir. Üçlü desenleri aynı değişkenleri içerebilirler ve bu durum birbirlerinin ilişkili verisetlerini etkileyebilir. Bu son analiz bakış açısı üçlü desenlerinin ortak değişkenlerine bakarak ilişkisiz verisetlerini elemeyi amaçlar.

Keşif kurallarının daha iyi anlaşılması için kuralların detaylı tanımları analiz süreci ile beraber anlatılacaktır. Şekil 3.3 WoDQA analiz sürecini göstermektedir. Burada bazı analiz aşamaları, her aşamadan sonra üretilen çıktılar ve girdiler görülmektedir. Tek tek her aşama ve her aşama içerisindeki kurallar ve uygulananlar açıklanacaktır. Böylece analiz süreci içerisinde kuralların işletilmesinin veriseti seçilimine nasıl bir etkisi olduğu daha iyi görülebilecektir.



Şekil 3.3: WoDQA analiz süreci

3.1.1.1 Tek adım analiz aşaması

Bu bölümde, ilişkili verisetlerini bulmak için ilk kez uygulanan kuralların uygulandığı *Tek Adım Analiz* aşaması anlatılacaktır. $\mathcal{Q}_{BGP} = \{\mathcal{Q}_{tp_i} | tp_i \in BGP\}$ bir sorgu içerisindeki üçlü desenlerinin sorgulanacakları ilişkili verisetleri kümelerini temsil eder ve analiz aşamasından önce hiçbir kural uygulanmadığı durumda her üçlü deseni için ilişkili verisetleri var olan tüm verisetleridir, $\forall \mathcal{Q}_{tp_i} \in \mathcal{Q}^{init}_{BGP} \, (\mathcal{Q}_{tp_i} \equiv \Delta)$. Bu aşamadaki kuralların ortak özelliği analiz için sorgudaki her üçlü desenini tek tek kullanmaları ve işletim sıralarından bağımsız olarak her bir üçlü deseni için tek bir kez çalıştırılmalarıdır. Bu aşamadan sonra üçlü desenleri için bazı ilişkisiz verisetleri elenmiş olur.

Şimdi tanıtılacak olan ilk iki keşif kural IRI Tabanlı Analiz bakış açısı altında sınıflandırılmaktadır. İkisi de VOID belgelerindeki void:vocabulary $(\mathcal{I}^{voc})$ ile belirtilen veriseti tarafından kullanılan sözlükleri dikkate alır. İlk keşif kuralı, üçlü desenlerindeki IRI olarak bulunan yüklemlerin VOID belgelerindeki tanımlı sözlüklere $(\mathcal{I}^{voc})$ ait bir özellik olup olmadıklarını kontrol eder. *Sözlük Eşleşmesi Kuralı*'nı biçimsel olarak tanımlayabilmek için, bir kaynağın $(r \in \mathcal{I})$, bir verisetinin VOID belgesinde tanımlı herhangi bir sözlüğe $(\mathcal{I}^{voc}_{\delta_x})$ ait olup olmadığını gösteren $has(\mathcal{I}^{voc}_{\delta_x}, r)$ fonksiyonunu kullanacağız. Böylece verisetinde kullanılan sözlük tanımlarına göre bir kaynağın hangi verisetleri ile ilişkili olup olmadığı Tanım 2'deki gibi gösterilir.

Tanım 2 *Eğer bir kaynak bir verisetinin tanımlı sözlüklerinden herhangi birine aitse o kaynak verisetinin sözlükleri için eşleşiyordur.*

$\forall \delta_x(has(\mathcal{I}^{voc}_{\delta_x}, r) \rightarrow VocMatch(\delta_x, r))$, $r \in \mathcal{I}$ iken.

?s dbpprop:name “Nikola Tesla”@en.

Yukarıdaki gibi bir üçlü deseni için, yüklem konumunda bulunan dbpprop:name RDFS özelliği, üçlü deseninin dbpprop[2] sözlüğünü kullanan bir verisetinden sorgulanmasını zorunlu kılmıştır. Çünkü bu sözlükte kullanılan bir kaynak üçlü deseni içerisinde yüklem olarak kullanılmıştır ve sadece dbpprop sözlüğünü kullanan verisetlerinin bu üçlü deseni için bazı üçlüler içerme olasılığı vardır. Bu nedenle dbpprop sözlüğünü kullanmayan verisetleri elenmelidir. Kural 2, yani Sözlük Eşleşmesi Kuralı ile keşfedilen ilişkili verisetleri dışındaki verisetleri incelenen üçlü deseni için elenir.

Kural 2 *Eğer bir üçlü deseninin yüklemi bir verisetinin kullandığı sözlüklerden herhangi birisi ile eşleşiyorsa, veriseti üçlü deseni için ilişkilidir.*

$\forall tp_i, \delta_x \left(VocMatch\left(\delta_x, p_{tp_i}\right) \rightarrow \rho\left(\delta_x, tp_i\right)\right)$, $p_{tp_i} \in < s_{tp_i}, p_{tp_i}, o_{tp_i} >$ *iken.*

Şekil 3.2'ye göre DBpedia verisetinin dbpprop sözlüğünü kullandığı görülmektedir ve DBpedia'nın bu üçlü deseni için ilişkili olduğu sonucu çıkarılır. Veri vebi üzerinde herhangi bir sözlüğü kullanan çok fazla veriseti bulunabilir. Onların içerisinde bulunan ilişkisiz olan verisetlerini de elemek için ileride anlatılacak olan keşif kuralları kullanılacaktır.

IRI Tabanlı Analiz altındaki ikinci kural RDF Tip Eşleşmesi kuralıdır. Bu kuralda içerisinde rdf:type yüklemini içeren üçlü desenlerinin nesnelerindeki IRI'lerin hangi sözlükler ile eşleştikleri incelenir. Bunun için

?producer rdf:type linkedMDB:producer.

gibi bir üçlü deseni örneğini ele alacağız. Bu örnekteki üçlü deseninin nesnesi linkedMDB sözlüğünü kullanan verisetleri dışındaki verisetlerinin elenmesini sağlar. Şekil 3.2'deki VOID modellerine göre LinkedMDB veriseti bu sözlüğü kullanmaktadır ve ilişkili bir veri setidir. Kural 3, RDF Tip Eşleşmesi Kuralı'nı göstermektedir.

Kural 3 *Eğer bir üçlü deseninin yüklemi* rdf:type *ise ve o üçlü deseninin nesnesi bir verisetinin tanımlı sözlüklerinden herhangi biri ile eşleşiyorsa, o veriseti o üçlü deseni için ilişkilidir.*

[2]Bu tezde kullanılan tüm önekler Ek3'deki tabloda gösterilmiştir.

$\forall tp_i, \delta_x(VocMatch(\delta_x, o_{tp_i}) \wedge (p_{tp_i} = rdf : type) \rightarrow \rho(\delta_x, tp_i))$, $o_{tp_i} \in <$ $s_{tp_i}, p_{tp_i}, o_{tp_i} >$ *iken..*

İlişkili verisetlerini keşfetmede kullanılan bir diğer bakış açısı Bağ Analizi'dir. Bir üçlünün öznesi ve nesnesi bir IRI olduğunda üçlü iki kaynağı birbirine bağlıyor demektir. Nesne konumundaki kaynak özne ile aynı verisetine ait bir kaynak olabilecei gibi farklı verisetine ait bir kaynak da olabilir. Bu yüzden bu gibi üçlü desenlerini iki aşamada analiz etmek gerekir. İlk aşamada *İçsel Bağ Analizi* gerçekleştirilir. Bu analiz, üçlülerin aynı veriseti içerisindeki kaynakları birbirine bağladığını varsayarak ilişkili verisetlerini keşfeder. Bu analiz ile bulunan ilişkili verisetleri içsel ilişkili veriseti olarak adlandırılır ve $\rho^{int}(\delta_x, tp_i)$ belirtimi ile gösterilir. Diğer aşamada *Dışsal Bağ Analizi* yapılır. Üçlülerin bağladığı kaynakların farklı verisetinde oldukları varsayılarak analiz yapılır. Bu durumda VOID belgelerindeki bağseti tanımlamaları dikkate alınır. Bu analiz ile bulunan ilişkili verisetleri dışsal ilişkili veriseti olarak adlandırılır ve $\rho^{ext}(\delta_x, tp_i)$ belirtimi ile gösterilir. Bir üçlü deseni için Bağ Analizi bakış açısı uygulanırken hem İçsel hem de Dışsal Bağ Analizi yapılır ve üretilen içsel ve dışsal ilişkili verisetleri üçlü deseni için ilişkili veriseti olarak kabul edilirler. Her iki yöntemle bulunan çözüm kümelerinin ilişkili veriseti olduğu Kural 4 ile gösterilmiştir.

Kural 4 *Bir üçlü deseni için üretilen içsel ve dışsal ilişkili verisetlerinin tümü o üçlü deseni için ilişkili verisetleridir.*

$\forall \delta_x, \delta_y, \delta_z, tp_i \, (\rho^{int} \, (\delta_x, tp_i) \vee \rho^{ext} \, (\delta_y, tp_i) \rightarrow \rho \, (\delta_z, tp_i))$.

Kural 4 ile bulunan ilişkili verisetleri kullanılarak üçlü deseninin sorgulanacak verisetleri $(\mathcal{Q}_{tp_i})$ içerisindeki ilişkisiz verisetleri elenir. İçsel ve Dışsal ilişkili verisetleri, üçlü desenleri için ilişkili verisetlerinin çıkarsanması için ara sonuçlardır.

Bağ Analizi bakış açısının kullanıldığı ilk iki kural *IRI'ye Bağlanma Kuralları* olarak adlandırılır. Bunun için bir üçlüde özne ya da nesne konumunda olan bir kaynağın bir verisetine ait olma durumunu biçimsel olarak tanımlamamız gerekmektedir. Eğer bir kaynağın IRI'si bir veriseti için tanımlanan isim uzayı özelliklerinden $(\mathcal{L}_{\delta_x}^{space})$ herhangi biri ile başlıyorsa, verisetinin kaynağın sahibi olduğu çıkarsaması yapılabilir. Bu durum Tanım 3'te gösterilmiştir.

Tanım 3 $\forall \delta_x(startsWith(r, \mathcal{L}_{\delta_x}^{space}) \rightarrow owner(\delta_x, r))$.

?film owl:sameAs dbpedia:A_Fistful_of_Dollars.

üçlü deseni örneğine göre ?film değişkenine karşılık gelebilecek kaynaklar dbpedia:A_Fistful_of_Dollars kaynağına owl:sameAs yüklemi ile bağlı olan kaynaklardır ve bu kaynakların sahibi olan verisetleri de ilişkili verisetleridir.

Kural 5, İçsel Bağ Analizi bakış açısıyla analizlerini gerçekleştiren *IRI'ye Bağlanma İçsel Kuralı*'nı göstermektedir. Yukarıdaki örnek sorguya göre ?film değişkenine karşılık gelen kaynaklar dbpedia:A_Fistful_of_Dollars kaynağı ile aynı verisetinde olmalıdır. Bu nedenle bu üçlü deseni, dbpedia:A_Fistful_of_Dollars kaynağının da sahibi olan DBpedia verisetinden sorgulanmalıdır.

Kural 5 *Bir üçlü deseninde nesne konumunda bulunan IRI'nin sahibi olan veriseti, o üçlü deseni için içsel ilişkili verisetidir.*

$$\forall tp_i, \delta_x \left((\delta_x \in \mathcal{Q}_{tp_i}) \wedge owner\left(\delta_x, o_{tp_i}\right) \rightarrow \rho^{int}\left(\delta_x, tp_i\right) \right) \ , \ o_{tp_i} \in \mathcal{I}, s_{tp_i} \in \mathcal{V}$$

iken.

Diğer taraftan Dışsal Bağ Analizi bakış açısına göre uygun üçlüler, nesne IRI'sinin sahibi olan verisetine bağlanan verisetlerinde bulunurlar. Örneğe göre ?film değişkenine karşılık gelen kaynaklar, nesneleri DBpedia'da tanımlı olan bağ üçlülerini içeren DBpedia'dan farklı verisetlerinde bulunabilirler. Bu analizi yapabilmek için VOID belgelerindeki bağseti tanımlamaları kullanılır. IRI'ye Bağlanma analizini dışsal açıdan uygulayabilmemiz için üçlü deseni içerisindeki yüklemin bir IRI ve aynı zamanda bir bağ yüklem olması gerekmektedir. Üçlü desenleri için hangi bağseti tanımlamalarının kullanılacağını Tanım 4'deki *Compatible* bildirimi ile biçimsel olarak gösterilmektedir.

Tanım 4 *Eğer bir üçlünün o anki ilişkili veriseti kümesi* $(\mathcal{Q}_{tp_i})$ *bağseti tanımlamasında yer alan kaynak verisetini içeriyorsa ve bağseti tanımlamasında yer alan bağ yüklem üçlü desenindeki yüklem ile aynıysa bağseti üçlü deseni için uygundur.*

$$\forall \lambda_m, tp_i((\delta^{from}_{\lambda_m} \in \mathcal{Q}_{tp_i}) \wedge (p^{link}_{\lambda_m} = p_{tp_i}) \rightarrow Compatible(\lambda_m, tp_i)).$$

?film owl:sameAs dbpedia:A_Fistful_of_Dollars.

üçlü deseni örneğine ve Şekil 3.2'deki VOID modeli örneğine göre dbpedia:A_Fistful_of_Dollars kaynağına sahip olan DBpedia verisetine owl:sameAs bağ yüklemi ile sadece LinkedMDB ve NYtimes verisetleri bağlıdır. Buna göre Kural 6'daki *IRI'ye Bağlanma Dışsal Kuralı* bu iki verisetini dışsal ilişkili verisetleri olarak çıkarsamaktadır.

Kural 6 *Bir üçlü deseni ile uyumlu bir bağseti tanımı varsa ve bu bağseti tanımının hedef veriseti üçlü desendeki nesne olan IRI'nin sahibi ise o bağseti tanımının kaynak veriseti üçlü deseni için dışsal ilişkili verisetidir.*

$\forall tp_i, \lambda_m, \delta_x(Compatible(\lambda_m, tp_i) \wedge owner(\delta_x, o_{tp_i}) \wedge (\delta_x = \delta^{to}_{\lambda_m}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i))$, $o_{tp_i} \in \mathcal{I}, s_{tp_i} \in \mathcal{V}$ *iken.*

İçsel ve dışsal ilişkili verisetleri bulunduktan sonra üçlü deseni için ilişkili verisetinin oluşturmak amacıyla Kural 4 ile birleştirilmelidirler. Böylece IRI'ye Bağlanma Kuralları'ne göre ilişkili verisetleri DBpedia, LinkedMDB ve NYtimes verisetleridir.

Bağ Analizi bakış açısı altındaki diğer bir çift kural ise *IRI'nin Bağları Kuralları* olarak adlandırılır ve içsel ve dışsal açıdan ilişkili verisetlerini bulmaya yöneliktir. Bu kurallar öznesi bir IRI ve nesnesi bir değişken olan bir üçlü desenini analiz etmek için kullanılır. Kural 7, *IRI'nin Bağları İçsel Kuralı*'dır ve aynı verisetinde kaynakları birbirine bağlayan üçlüleri bulmayı hedefler. Bu kurala göre üçlü desenindeki öznenin sahibi olan veriseti, o üçlü deseni için içsel ilişkili verisetidir.

Kural 7 *Bir üçlü deseninin öznesinin sahibi olan veriseti, üçlü deseni için içsel ilişkili verisetidir.*

$\forall tp_i, \delta_x ((\delta_x \in \mathcal{Q}_{tp_i}) \wedge owner(\delta_x, s_{tp_i}) \rightarrow \rho^{int}(\delta_x, tp_i))$, $o_{tp_i} \in \mathcal{V}, s_{tp_i} \in \mathcal{I}$ *iken.*

dbpedia:Ennio_Morricone owl:sameAs ?person.

örneğini ele alalım. Üçlü deseninin öznesinin isim uzayı, DBpedia VOID belgesinde tanımlı olan bir isim uzayı özelliğidir. Bu yüzden DBpedia, bu üçlü deseni örneği için içsel ilişkili verisetidir.

IRI'nin Bağları Dışsal Kuralı bağseti tanımlamalarını dikkate alarak farklı verisetlerinde birbirine bağlı olarak bulunan kaynaklar hakkında bilgi vermeyi amaçlamaktadır. Ancak dikkate alınan üçlü deseninin öznesi konumundaki kaynağın veriseti, yani üçlü deseninin sorgulanacağı veriseti, isim uzayı tanımlaması IRI'nin Bağları İçsel Kuralı ile zaten belirlenmiştir. Bağseti tanımlamaları ile üçlü desenindeki nesneye karşılık gelen kaynakların bulundukları veriseti ile bilgiler edinilir ama bu bilgi de incelenen üçlü deseni için bir işe yaramaz. Bağ Analizi bakış açısı altındaki kurallardan IRI'nin Bağları Kuralları bir istisnadır ve sadece içsel olarak ele alınır. İçsel ilişkili olarak bulunan verisetleri de üçlü deseni için doğrudan ilişkili verisetleri olurlar.

Tek Adım Analiz aşamasında yer alan kurallar bu kadardır. Bu aşama sonunda bazı üçlüler için ilişkisiz verisetleri elenerek ilişkili verisetleri belirlenmiş olabilir ve ilişkili verisetleri kümeleri $\mathcal{Q}_{BGP}^{singleStep}$ olarak temsil edilir. Bu çıktı analiz süreci içerisinde bir ara sonuçtur.

Analiz sürecinin ve verisetlerinin elenmesinin daha iyi anlaşılması için

SELECT ?actor ?news WHERE
{ film:TarzanMovie linkedMDB:actor ?actor .
?actor owl:sameAs ?x.
?y owl:sameAs ?x .
?y nytimes:topicPage ?news }

gibi bir analiz süreci örnek sorgusunun ilişkili verisetlerinin her analiz aşamasının sonunda nasıl elde edildiğini ve ilişkisizlerin nasıl elendiğini göstereceğiz.

Analiz süreci öncesinde Şekil 3.2'deki VOID belgeleri dikkate alındığında her üçlü deseni için sorgulanacak verisetleri kümesi $\mathcal{Q}_{BGP}^{init} = \{\delta_{DBpedia}, \delta_{LinkedMDB}, \delta_{NYtimes}, \delta_{MusicBrainz}, \delta_{Facebook}\}$ şeklindedir.

- Birinci üçlü deseni ele alındığında yüklem olarak linkedMDB:actor bulunduğundan Sözlük Eşleşmesi Kuralı'na göre sadece LinkedMDB ilişkili verisetidir. Diğer tüm verisetleri elenir. Ayrıca IRI'nin Bağları İçsel Kuralı'na göre de film isim uzayı sadece LinkedMDB'de bulunduğundan sadece LinkedMDB ilişkili veriseti belirtimi olarak bulunur. Ancak ilişkili verisetlerinde de sadece LinkedMDB olduğundan bir başka bir eleme yapılmaz. Birinci üçlü deseni için Tek Adım Analiz aşamasında başka bir kural uygulanmaz. Sonuçta $\mathcal{Q}_{tp_1} = \{\delta_{LinkedMDB}\}$ olur.

- İkinci ve üçüncü üçlü desenleri de Tek Adım Analiz aşamasındaki kurallar için uygun değildir ve herhangi bir eleme yapılmaz. Sonuçta $\mathcal{Q}_{tp_2} = \mathcal{Q}_{tp_3} = \{\delta_{DBpedia}, \delta_{LinkedMDB}, \delta_{NYtimes}, \delta_{MusicBrainz}, \delta_{Facebook}\}$ olur.

- Dördüncü üçlü deseni için Sözlük Eşleşmesi Kuralı işletildikten sonra NYtimes veriseti ilişkili olarak bulunur ve diğer DBpedia, LinkedMDB, MusicBrainz ve Facebook verisetleri elenirler. Bu durumda $\mathcal{Q}_{tp_4} = \{\delta_{NYtimes}\}$ olur..

Algoritma 1 ASK sorguları ile analiz uygulama algoritması

FUNCTION $analyzeWithAskQueries()$
INPUT $bgp = \{tp_1, ..., tp_n\}$; //$n$ tane üçlü deseni içeren sorgu deseni
FOR EACH tp_i IN bgp DO
  IF ($\mathcal{Q}_{tp_i}$ EQUALS $\mathcal{Q}_{bgp}^{init}$)
    continue;
  ELSE
    FOR EACH δ_x IN $\mathcal{Q}_{tp_i}$DO
      LET $isExist$:=*ask* tp_i *to endpoint of* δ_x*;*
      IF ($\neg isExist$)
        $\mathcal{Q}_{tp_i}$:=$\mathcal{Q}_{tp_i} \setminus \delta_x$;

3.1.1.2 ASK sorgular ile analiz

Bu aşamada Tek Adım Analiz aşaması ile sorgulanacak verisetleri kümesi elenerek azaltılmış olan üçlü desenleri dikkate alınır. İlişkili verisetleri hiç elenmemiş diğer üçlü desenleri için bu aşamada *ASK Sorguları ile Analiz* gerçekleştirilmez. Çünkü veb üzerindeki tüm verisetlerine ASK sorguları atılması çok büyük bir maliyet getirir. Dikkate alınan üçlü desenlerinin ilişkili verisetlerine ASK sorguları atılarak üçlü deseninin ilişkili verisetinde gerçekten var olup olmadığına dair bir bilgi alınması sağlanır. Bu bilgiye göre olumsuz yanıt alınan verisetleri ilişkisiz olarak kabul edilir ve elenirler. Algoritma 1'de bu ASK sorguları ile analizin nasıl gerçekleştirildiği gösterilmiştir.

Tek Adım Analiz aşamasında bir üçlü deseni için elenmiş olan verisetleri kesin olarak üçlü deseni için sonuç içermemekle beraber ilişkili olarak bulunan verisetleri de içerisinde herhangi bir sonuç içermeyen bazı verisetlerini bulundurabilir. Bu durumun nedenlerini kuralları ele alarak şu şekilde açıklayabiliriz. Sözlük Eşleşmesi ve RDF Tip Eşleşmesi Kuralları ile kullanılan veriseti tarafından kullanılan sözlükler dikkate alınırken sözlük içerisindeki hangi özelliğin veya sınıfın kullanıldığı dikkate alınmamaktadır. Örneğin

?s rdfs:label "Bursa".

gibi bir üçlü deseni için rdfs sözlüğünü kullanan bir veriseti rdfs:label yüklemini bulunduran bir üçlüyü içermeyebilir. Ayrıca içerse bile üçlünün nesnesinin de "Bursa" olması gerektiğinden sadece bir yüklemin kullanıldığından emin olmak bile üçlü deseninin kesin olarak kullanıldığını göstermez. Benzer şekilde

?s rdf:type foaf:Person.

gibi bir üçlü deseni için RDF Tip Eşleşmesi Kuralı foaf sözlüğünü kullanan verisetlerinin bu üçlü deseni için sonuç içerdiğini belirtir. Ancak içerisinde foaf sö-

zlüğünü kullanıp nesnesi foaf:Person olmayan üçlüleri içeren verisetleri de olabileceğinden bazı ilişkisiz verisetleri de ilişkili olarak kabul edilir. IRI'nin Bağları ve IRI'ye Bağlanma Kuralları'nda da bir kaynağın sadece isim uzayına bakılarak yorum yapıldığından kaynağın verisetinde bulunmama ihtimali sözkonusudur. Örneğin

<http://dbpedia.org/resource/Bursa> ?p ?o.

örneğinde özne olarak bulunan kaynağının isim uzayı *http://dbpedia.org/resource/* olduğu için bu isim uzayını bulunduran her veriseti IRI'nin Bağları kuralına göre üçlü deseni için ilişkili kabul edilir. Ancak bazı verisetleri bu isim uzayını kaynaklarında bulundurup sözkonusu kaynağı üçlülerinin içerisinde bulundurmayabilir. Aynı durum IRI'ye Bağlanma Kuralı için de geçerlidir. Bu nedenle Tek Adım Analiz aşamasından sonra ilişkili verisetleri içerisinde bulunan ilişkisiz verisetlerini eleyebilmek için üçlü desenleri (?s ?p ?o gibi üç elemanı da değişken olan üçlü desenleri hariç) ASK sorguları ile ilişkili verisetlerinden tek tek sorgulanırlar. Böylece kesin olarak sonuç içeren verisetleri seçilirler. Daha sonra bu sonuçlar $\mathcal{Q}_{BGP}^{constrained}$ olarak *Yinelemeli Analiz* aşamasına girdi olarak verilirler.

Analiz süreci sorgusu örneğini analiz etmeye devam edecek olursak;

- Birinci üçlü deseni için sadece sorgulanacak verisetleri kümesinde Tek Adım Analiz aşamasından sonra sadece LinkedMDB veriseti kaldığı için sadece LinkedMDB verisetine ASK sorgusu atılır ve olumlu yanıt alındığı için elenmeden kalır

- İkinci ve üçüncü üçlü desenlerinin sorgulanacak verisetleri kümeleri daha önce hiç elenmedikleri için bu üçlü desenleri ASK sorguları ile analiz edilmezler.

- Aynı şekilde dördüncü üçlü deseni için de sadece tek ilişkili verisetleri olan NYtimes verisetine ASK sorgusu atılır ve olumlu sonuç alınarak elenmeden kalır.

3.1.1.3 Yinelemeli analiz aşaması

Buraya kadar incelenen kurallar tek bir üçlü deseni için işletilmekteydi. Ancak Yinelemeli Analiz aşamasında bulunan kurallar iki üçlü desenini birlikte ele almayı gerektirir. Bu nedenle sorgu içerisindeki tüm üçlü desenlerinin ikili kombinasyonları için bu kurallar işletilir. Bu aşamadaki kurallar farklı işletim sıraları ile farklı ilişkili veriseti belirtimleri üretirler. Her farklı belirtim de bazı ilişkili verisetlerinin ilişkisiz olduğunun çıkarsanmasını sağlar. Sonuçta Yinelemeli Analiz aşaması altındaki tüm

Algoritma 2 Yinelemeli Analiz aşamasındaki kuralların işletim algoritması

FUNCTION $applyRepetitiveAnalyzeRules()$
INPUT $bgp = \{tp_1, ..., tp_n\}$; //$n$ tane üçlü deseni içeren sorgu deseni
LET $\mathcal{Q}_{bgp}^{constrained} := \{\mathcal{Q}_{tp_1}, ..., \mathcal{Q}_{tp_n}\}$;
FOR *i:=1; i<=n; i++* DO
  FOR EACH tp_j IN bgp DO
    IF (tp_i NOT EQUALS tp_j)
      apply all repetitive rules for tp_i *and* tp_j*;*
  //kurallar uygulandıktan sonra elenmeler olmuş olabilir.
  //elemeden sonra üretilen yeni sorgulanacak verisetleri kümesi $\mathcal{Q}_{bgp}^{new}$'dir.
  IF($\mathcal{Q}_{bgp}^{new}$ NOT EQUALS $\mathcal{Q}_{bgp}^{constrained}$)
    LET i:=1;
  LET $\mathcal{Q}_{bgp}^{constrained} := \mathcal{Q}_{bgp}^{new}$;

kurallar üçlü desenlerinin tüm ikili kombinasyonları için birer kere işletildikten sonra girdi olarak verilen $\mathcal{Q}_{BGP}^{constrained}$ kümesi $\mathcal{Q}_{BGP}^{new}$ olarak çıkar. $\mathcal{Q}_{BGP}^{constrained}$ kümesi elenmeye uğrayarak analizden geçtiyse ilişkili verisetlerinin değişmesi kurallar tarafından yine farklı belirtimler üretilmesine neden olabilir. Bu nedenle üretilen $\mathcal{Q}_{BGP}^{new}$ kümesi bu aşamadaki kurallar ile tekrar çalıştırılmak üzere girdi olarak Yinelemeli Analiz aşamasına tekrar verilir. Bu döngü Yinelemeli Analiz aşamasına girdi olarak verilen $\mathcal{Q}_{BGP}^{constrained}$ kümesinin elemeye uğramadan $\mathcal{Q}_{BGP}^{new}$ olarak çıkmasına kadar tekrarlanır. Bu durum Algoritma 2 ile gösterilmektedir. Şimdi bu aşama altındaki kurallar detaylı bir şekilde tanımlanacaktır.

Yinelemeli Analiz aşamasındaki tüm kurallar Paylaşılan Değişken Analizi bakış açısı altındadır. Ayrıca bu bakış açısı altındaki kurallar uygulanırken IRI Tabanlı Analiz'de ve Bağ Analizi'nde belirtilen yöntemler de kullanılmaktadır. Paylaşılan Değişken Analizi altındaki ilk iki kural *Zincirleme Üçlü Desenleri Kuralları* olarak adlandırılır. Bu kurallar ilişkili verisetlerini keşfedebilmek için birinin nesnesi diğerinin öznesi ile aynı değişkeni içeren iki üçlü desenini birlikte ele alır.

```
?s owl:sameAs ?film.
?film linkedMDB:producer_name “Sergio Leone”.
```

üçlü desenleri ?film değişkenini hem özne hem de nesne olarak içerdikleri için birlikte ele alınabilirler. İçsel Bağ Analizi ile bakıldığında ?film değişkenine karşlık gelen kaynakların bulunduğu veriseti aynı zamanda ?s değişkenine kaşılık gelen kaynakların da bulunduğu verisetidir. Yani örneğe göre üçlü desenleri ?film değişkenine karşılık gelen kaynakların bulunduğu verisetinden sorgulanmalıdır. Bu yaklaşıma göre üçlü de-

senlerinin içsel ilişkili verisetlerini bulabilmek için *Zincirleme Üçlü Desenleri İçsel Kuralı*, Kural 8 tanımlanmıştır.

Kural 8 *Eğer bir üçlü deseninin nesnesi başka bir üçlü deseninin öznesi ile aynıysa bu üçlü desenlerinin sorgulanacakları veriseti kümelerindeki ortak elemanlar kesişimi bu üçlü desenleri için içsel ilişkili verisetleridir.*

$\forall \delta_x, tp_i, tp_j((o_{tp_i} = s_{tp_j}) \wedge (\delta_x \in \mathcal{Q}_{tp_i}) \wedge (\delta_x \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{int}(\delta_x, tp_i) \wedge \rho^{int}(\delta_x, tp_j))$, $o_{tp_i}, s_{tp_i} \in \mathcal{V}$ *iken.*

?s owl:sameAs ?film.
?film linkedMDB:producer_name “Sergio Leone”.

örneğine göre, Sözlük Eşleşmesi Kuralı uygulandıktan sonra ikinci üçlü deseni için ilişkisiz olan verisetleri Şekil 3.2'de LinkedMDB verisetinin linkedMDB sözlüğünü kullanmasından dolayı elenir ve sonuçta $Q_{tp_2} = \{\delta_{LinkedMDB}\}$ olur ve birinci üçlü desenininki de $Q_{tp_1} \equiv \Delta$'dir. Daha sonra Zincirleme Üçlü Desenleri İçsel Kuralı şu belirtimleri çıkarsar: $\rho^{int}(\delta_{LinkedMDB}, tp_1)$, $\rho^{int}(\delta_{LinkedMDB}, tp_2)$. Yani birinci üçlü deseni için ilişkisiz verisetleri de elendikten sonra içsel ilişkili verisetleri sadece LinkedMDB olur.

Diğer yandan aynı örneğe Dışsal Bağ Analizi ile yaklaştığımızda ise üçlü desenleri incelenirken bağseti tanımlamaları kullanılır. İçsel Bağ Analizi ile üçlü desenlerinde bağ yüklem olmasına bakılmazken, burada paylaşılan değişkeni nesne olarak içeren üçlü deseninin bağ yüklem içermesi beklenir.

Kural 9 *Eğer bir üçlü deseninin* (tp_i) *nesnesi bir diğer üçlü deseninin* (tp_j) *öznesi ile aynıysa ve* (tp_i) *üçlü deseni ile uyumlu bir bağseti tanımı var ise ve bağseti tanımının hedef veriseti* tp_j *üçlü deseninin o anki sorgulanacak ilişkili verisetlerinden biri ise bağseti tanımının kaynak veriseti* tp_i *üçlü deseni için, hedef veriseti de* tp_j *üçlü deseni için dışsal ilişkili verisetidir.*

$\forall \lambda_m, tp_i, tp_j((o_{tp_i} = s_{tp_j}) \wedge Compatible(\lambda_m, tp_i) \wedge (\delta^{to}_{\lambda_m} \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i) \wedge \rho^{ext}(\delta^{to}_{\lambda_m}, tp_j))$, $o_{tp_i} \in \mathcal{V}$ *iken.*

Zincirleme Üçlü Desenleri Dışsal Kuralı'nın örnek sorguya uygulanmadan önce üçlü desenleri için sorgulanacak ilişkili verisetlerinin $Q_{tp_1} \equiv \Delta$ ve $Q_{tp_2} = \{\delta_{LinkedMDB}\}$ olduğunu varsayalım. Kural 9'a göre birinci üçlü deseni için dışsal

ilişkili veriseti sadece DBpedia'dır. Çünkü ?film değişkenine karşılık gelen kaynaklar sadece LinkedMDB'dedir ve LinkedMDB verisetine owl:sameAs ile bağlı olan tek veriseti de DBpedia'dır. Sonuçta sorgulanacak dışsal ilişkili verisetleri $\rho^{ext}(\delta_{DBpedia}, tp_1)$ ve $\rho^{ext}(\delta_{LinkedMDB}, tp_2)$ olur. Zincirleme Üçlü Desenleri İçsel ve Dışsal Kuralları uygulanıp içsel ve dışsal ilişkili verisetleri çıkarsandıktan sonra Kural 4 ile birleştirilirler ve ilişkili verisetleri oluşturulur.

Zincirleme Üçlü Desenleri Kuralları'na benzer şekilde Paylaşılan Değişken Analizi bakış açısı altında incelenen diğer kural çifti ise *Nesne Paylaşan Üçlü Desenleri Kuralları*'dır. Bu kurallar aynı nesne değişkenini paylaşan üçlü desenlerini ele alıp bu üçlü desenleri için ilişkili verisetlerini bulmaya çalışır. Bu kuralların açıklanması için

```
?person facebook:likes ?movie.
?film owl:sameAs ?movie.
```

sorgu örneği kullanılacaktır.

Örnekteki üçlü desenlerini İçsel Bağ Analizi ile ele aldığımızda ?person ve ?film değişkenlerine karşılık gelecek olan kaynaklar ?movie değişkenine karşılık gelen kaynakların bulunduğu veriseti ile aynı verisetinde olmalıdır. Dolayısıyla iki üçlü deseninin de içsel ilişkili verisetleri aynı olmalıdır. Üçlü desenlerinin o anki sorgulanacakları ilişkili verisetlerinin $\mathcal{Q}_{tp_1} = \{\delta_{Facebook}\}$ ve $\mathcal{Q}_{tp_2} \equiv \Delta$ olduğunu varsayalım. Buna göre Kural 10'da biçimselleştirilen *Nesne Paylaşan Üçlü Desenleri İçsel Kuralı*'nın uygulanması ile iki üçlü deseni için de içsel ilişkili verisetlerinin sadece $\rho^{int}(\delta_{Facebook}, tp_1)$ ve $\rho^{int}(\delta_{Facebook}, tp_2)$olduğu çıkarsanır. Çünkü sadece Facebook veriseti her iki üçlü deseni için de ortak çözümler içerir.

Kural 10 *Eğer bir üçlü deseninin nesnesi ile bir diğer üçlü deseninin nesnesi aynı ise, sorgulanacakları ilişkili verisetleri kümelerindeki ortak elemanlar her ikisi için de içsel ilişkili verisetleridir.*

$\forall \delta_x, tp_i, tp_j((o_{tp_i} = o_{tp_j}) \wedge (\delta_x \in \mathcal{Q}_{tp_i}) \wedge (\delta_x \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{int}(\delta_x, tp_i) \wedge \rho^{int}(\delta_x, tp_j))$, $o_{tp_i} \in \mathcal{V}$ *iken.*

Aynı sorgu örneğini Dışsal Bağ Analizi ile ele alabilmek için bağseti tanımlamaları kullanılır. Bu bakış açısına göre üçlü desenlerinin en az birisinde bağ yüklem olmalıdır. Her iki üçlü deseninin de bağ yüklem içererek ortak bir nesne değişkeni içerdiği durum ele alındığında üçlü desenlerinin öznelerine karşılık gelen kaynaklar

paylaşılan nesne değişkenine karşılık gelen kaynağın bulunduğu verisetinden farklı bir verisetinde bulunmalıdır. Sadece bir üçlü deseninin bağ yüklem içererek ortak bir nesne değişkeni içerdiği durumda bağ yüklem içeren üçlü deseni öznesine karşılık gelen kaynaklar paylaşılan nesneye karşılık gelen kaynaktan farklı bir verisetinde diğer üçlü deseninin öznesine karşılık gelen kaynaklar ise paylaşılan nesneye karşılık gelen kaynaklar ile aynı verisetinde bulunmalıdır. Kural 11 ile biçimselleştirilen *Nesne Paylaşan Üçlü Desenleri Dışsal Kuralları* aynı nesneyi paylaşan üçlü desenleri için dışsal ilişkili verisetlerinin bulunmasını sağlamaktadır. Bu kural en az bir üçlü deseninin bağseti tanımlamalarına uymasını gerektirmektedir.

Kural 11 *Eğer iki üçlü deseni aynı değişkeni nesne olarak bulunduruyorsa ve her ikisi için de hedef verisetleri aynı olan ve üçlü desenleri için uygun olan bağseti tanımlamaları varsa üçlü desenlerinin kendilerine uygun olan bağseti tanımlamalarının kaynak verisetleri üçlü desenleri için dışsal ilişkili verisetleridir.*

$\forall \lambda_m, \lambda_n, tp_i, tp_j((o_{tp_i} = o_{tp_j}) \wedge Compatible(\lambda_m, tp_i) \wedge Compatible(\lambda_n, tp_j) \wedge (\delta^{to}_{\lambda_m} = \delta^{to}_{\lambda_n}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i) \wedge \rho^{ext}(\delta^{from}_{\lambda_n}, tp_j))$, $o_{tp_i} \in \mathcal{V}$ *iken.*

Eğer iki üçlü deseni aynı değişkeni nesne olarak bulunduruyorsa ve sadece birisi tp_i bağ yüklem içeriyorsa ve tp_i için uygun olan bağseti tanımlamalarının hedef verisetleri bağ yüklem içermeyen üçlü deseninin, tp_j, o anki ilişkili verisetleri kümesinin elemanı ise, uygun bağseti tanımlamalarının kaynak verisetleri tp_i için dışsal ilişkili verisetleridir.

$\forall \lambda_m, tp_i, tp_j((o_{tp_i} = o_{tp_j}) \wedge Compatible(\lambda_m, tp_i) \wedge (\delta^{to}_{\lambda_m} \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i)$, $o_{tp_i} \in \mathcal{V}, p_{tp_j} \notin \mathcal{P}^{link}$ *iken.*

?person facebook:likes ?movie.
?film owl:sameAs ?movie.

üçlü desenleri örneğine göre *Nesne Paylaşan Üçlü Desenleri Dışsal Kuralları*'nı açıklayalım. Her iki üçlü desenindeki yüklemlerin bağ yüklem olduğunu ve örnek üçlü desenlerine hiçbir analiz uygulanmadığını varsayalım. Buna göre üçlü desenleri için sorgulanacak verisetleri kümeleri $\mathcal{Q}_{tp_1} \equiv \Delta$ ve $\mathcal{Q}_{tp_2} \equiv \Delta$ şeklindedir. Şekil 3.2'ye göre $\delta_{DBpedia}$, $\delta_{NYtimes}$ ve $\delta_{LinkedMDB}$ verisetleri owl:sameAs bağ yüklemi içeren bağ üçlüleri içermektedir. Sadece $\delta_{Facebook}$ veriseti facebook:likes bağ yüklemi içeren bağ üçlülerini içermektedir ve bu bağ üçlülerinin nesneleri de $\delta_{LinkedMDB}$ verisetinde bulunmaktadır. Buna göre birinci üçlü deseni için dışsal ilişkili verisetleri

$\rho^{ext}(\delta_{Facebook}, tp_1)$ şeklindedir. İkinci üçlü deseni de, birinci üçlü deseni için elde edilen bilgilere göre sadece LinkedMDB'deki kaynaklara owl:sameAs ile bağlı kaynakları içeren verisetlerinden sorgulanmalıdır. Çünkü ?film değişkeni ile eşleşen ve diğer verisetlerinde tanımlı olan kaynaklar birinci üçlü deseni için bulunan ?movie değişkenine karşılık gelen kaynaklarla eşleşmeyecektir. Bu yüzden de ikinci üçlü deseni sadece $\delta_{DBpedia}$ verisetinden sorgulanır, $\rho^{ext}(\delta_{DBpedia}, tp_2)$. Şimdi de sadece birinci üçlü deseninin yükleminin bağ yüklem olduğunu ve üçlü desenlerinin sorgulanacak verisetleri kümelerinin $\mathcal{Q}_{tp_1} \equiv \Delta$ ve $\mathcal{Q}_{tp_2} = \{\delta_{DBpedia}, \delta_{LinkedMDB}\}$ olduğunu varsayalım. Bu durumda Şekil 3.2'deki örnek VOID modellerine göre sadece Facebook veriseti ikinci üçlü deseninin sorgulanacak verisetlerinden LinkedMDB verisetine facebook:likes bağ yüklemi ile bağlıdır. Buna göre birinci üçlü deseni için dışsal ilişkili veriseti sadece $\delta_{Facebook}$ veriseti olur, $\rho^{ext}(\delta_{Facebook}, tp_1)$.

Diğer Bağ Analizi bakış açısı altındaki kurallarda olduğu gibi *Nesne Paylaşan Üçlü Desenleri İçsel* ve *Dışsal Kuralları* ile elde edilen içsel ve dışsal ilişkili verisetleri Kural 4 ile birleştirilirler.

Son kural, Paylaşılan Değişken Analizi bakış açısı altındaki *Özne Paylaşan Üçlü Desenleri Kuralı* olarak adlandırılır. Adından da anlaşılacağı gibi aynı değişkeni özne olarak içeren iki üçlü desenini ele alır. Burada herhangi bir linkset tanımı analiz için kullanılmadığından Bağ Analizi bakış açısı kullanılmaz, bu yüzden içsel ve dışsal ilişkili olarak verisetleri bulunmaz. Bu analizi incelemek için şu üçlü desenlerini örnek olarak inceleyeceğiz:

?city dbpprop:name "İzmir"@en.
?city dc:subject ?subject.

Daha önceki tanımlamalarımıza göre aynı özneyi içeren üçlüler aynı verisetlerinde bulunurlar. Buna göre Kural 12 özneleri aynı olan üçlü desenleri için ilişkili verisetlerini onların sorgulanacak verisetleri kümelerinin kesişimini alarak bulur. Örneğe göre üçlü desenlerinin o anki sorgulanacak verisetleri kümelerinin $\mathcal{Q}_{tp_1} = \{\delta_{DBpedia}\}$ ve $\mathcal{Q}_{tp_2} \equiv \Delta$ olduğunu varsayalım. Özne Paylaşan Üçlü Desenleri Kuralı uygulandıktan sonra ise üçlü desenlerinin sorgulanacak verisetleri kümeleri $\mathcal{Q}^{new}_{tp_1} \equiv \mathcal{Q}^{new}_{tp_2} = \{\delta_{DBpedia}\}$ olur.

Kural 12 *Eğer iki üçlü deseni aynı değişkeni özne olarak içeriyorsa, bu üçlü desenlerinin sorgulanacak verisetlerinin ortak elemanları bu üçlü desenlerinin yeni sorgulanacak verisetleri kümesidir.*

$\forall \delta_x, tp_i, tp_j((s_{tp_i} = s_{tp_j}) \wedge (\delta_x \in (\mathcal{Q}_{tp_i} \cap \mathcal{Q}_{tp_j})) \rightarrow \rho(\delta_x, tp_i) \wedge \rho(\delta_x, tp_j))$, $s_{tp_i} \in \mathcal{V}$ *iken.*

Yinelemeli Analiz aşamasındaki kurallar bu kadardır. Şimdi analiz süreci sorgu örneğinin bu aşama ile nasıl analiz edildiğini inceleyelim. Buradaki analiz aşamasında üçlü desenlerinin tüm ikili kombinasyonları üzerinde kurallar birer kez işletilir.

- Bu aşamaya girdi olarak verilen $\mathcal{Q}_{BGP}^{constrained}$ kümesi $\{\mathcal{Q}_{tp_1} = \{\delta_{LinkedMDB}\}, \mathcal{Q}_{tp_2} = \{\delta_{DBpedia}, \delta_{LinkedMDB}, \delta_{NYtimes}, \delta_{MusicBrainz}, \delta_{Facebook}\}, \mathcal{Q}_{tp_3} = \{\delta_{DBpedia}, \delta_{LinkedMDB}, \delta_{NYtimes}, \delta_{MusicBrainz}, \delta_{Facebook}\}, \mathcal{Q}_{tp_4} = \{\delta_{NYtimes}\}\}$ şeklindedir.

- Birinci ve ikinci üçlü desenleri birlikte ele alındığında üçlü desenleri yapısının Zincirleme Üçlü Desenleri Kuralları'na uydukları görülmektedir. Zincirleme Üçlü Desenleri İçsel Kuralı'na göre birinci üçlü deseninin ilişkili verisetleri ikinci üçlü deseni için de aynı olmalıdır. Zincirleme Üçlü Desenleri Dışsal Kuralı'na göre birinci üçlü deseninin bir bağ yüklem içermesi gerektiğinden dışsal ilişkili veriseti bulunmaz. Bu yüzden bulunan içsel ilişkili veriseti yani LinkedMDB ikinci üçlü deseni için ilişkili verisetidir ve diğer verisetleri elenmiş olur, $\mathcal{Q}_{tp_2} = \{\delta_{LinkedMDB}\}$.

- Birinci üçlü deseni ile üçüncü ve dördüncü üçlü desenleri herhangi bir kurala uymazlar ve bir değişiklik olmaz.

- İkinci üçlü deseni ile diğer üçlü desenleri ele alındığında sadece üçüncü üçlü deseni ile Nesne Paylaşan Üçlü Desenleri Kuralları'na uyan bir durum olduğu görülür. Her iki üçlü deseni için de öyle bağseti tanımlamaları bulunmalıdır ki aynı verisetine owl:sameAs ile bağlı oldukları tanımlı olmalıdır. İkinci üçlü deseni için daha az ilişkili veriseti olduğu için ikinci üçlü deseni için uygun bağseti tanımlamalarından yola çıkılır. İkinci üçlü deseni LinkedMDB'den sorgulanır. Nesne Paylaşan Üçlü Desenleri Dışsal Kuralı'na göre kaynak veriseti LinkedMDB olan ve bağ yüklemi owl:sameAs olan bağseti tanımlamalarının hedef veriseti ?x değişkenine karşılık gelen kaynakların bulunduğu verisetidir. Şekil 3.2'ye göre hedef veriseti sadece DBpedia'dır. O halde üçüncü üçlü deseni için hedef veriseti DBpedia olan ve bağseti:owl:sameAs olan tüm bağseti tanımlamalarının kaynak verisetleri üçüncü üçlü deseni için dışsal ilişkili verisetleridir.

Bunlar da NYtimes ve LinkedMDB verisetleridir. Nesne Paylaşan Üçlü Desenleri İçsel Kuralı'na göre üçüncü üçlü deseni için içsel ilişkili veriseti de sadece LinkedMDB'dir. Buna göre içsel ve dışsal ve ilişkili verisetlerinin bileşimine göre üçüncü üçlü deseni için ilişkili verisetleri $\mathcal{Q}_{tp_3} = \{\delta_{LinkedMDB}, \delta_{NYtimes}\}$ olur.

- Üçüncü üçlü deseni ile diğer üçlü desenlerini ele aldığımızda ise ikinci üçlü deseni ile Nesne Paylaşan Üçlü Desenleri Kuralları'nın işletilebileceği görülür ama eleme yapılacak yeni bir belirtim çıkarsanmaz ve az sorgulanacak verisetleri kümeleri aynı kalır.

- Üçüncü üçlü deseni ile dördüncü üçlü desenini ele aldığımızda ise Özne Paylaşan Üçlü Desenleri Kuralı'na uygunluk görülmektedir. Her iki üçlü deseninin ilişkili verisetlerinin kesişimi her ikisi için de ilişkili verisetleri kümesi olacaktır. Yani, $\mathcal{Q}_{tp_3} = \{\delta_{LinkedMDB}, \delta_{NYtimes}\}$ ve $\mathcal{Q}_{tp_4} = \{\delta_{NYtimes}\}$ kümelerinin kesişim olan $\delta_{NYtimes}$ veriseti her ikisi için de ilişkili verisetidir. Sonuç olarak Yinelemeli Analiz aşamasına verilen ilişkili verisetleri kümesi $\mathcal{Q}_{tp_3} = \{\delta_{NYtimes}\}$ ve $\mathcal{Q}_{tp_4} = \{\delta_{NYtimes}\}$ şeklinde olur.

- Bu analiz sonucunda üretilen $\mathcal{Q}^{new}_{BGP} = \{\mathcal{Q}_{tp_1} = \{\delta_{LinkedMDB}\}, \mathcal{Q}_{tp_2} = \{\delta_{LinkedMDB}\}, \mathcal{Q}_{tp_3} = \{\delta_{NYtimes}\}, \mathcal{Q}_{tp_4} = \{\delta_{NYtimes}\}\}$ olarak üretilir.

- $\mathcal{Q}^{new}_{BGP}$ kümesi girdi olarak verilen $\mathcal{Q}^{constrained}_{BGP}$ kümesinden farklı olduğu için tekrar Yinelemeli Analiz aşamasına verilir.

- Buraya kadar işletilen kurallar tekrar işletilir ancak bu sefer herhangi bir eleme gerçekleştirilmez, çünkü sorgulanacak verisetleri kümelerindeki elemanlar için artık ilişkisiz olma durumu çıkarsanmaz. Bu nedenle üretilen $\mathcal{Q}^{new}_{BGP}$ kümesi girdi olarak verilen $\mathcal{Q}^{constrained}_{BGP}$ kümesinden farklı olmadığı için yinelemeli analiz aşaması sona erer ve $\mathcal{Q}^{new}_{BGP}$ kümesi $\mathcal{Q}^{final}_{BGP}$ olarak kabul edilir.

WoDQA analiz süreci içerisindeki kurallar ile ilişkisiz verisetleri elenerek ilişkili verisetleri seçilmeye çalışılır. Farklı bakış açıları ile daha çok ilişkisiz veriseti bulunup daha iyi bir seçilim de yapılabilir ancak bu noktada ve bu kurallarla diğer sorgulama motorlarına kıyasla daha iyi bir elemenin yapıldığı Bölüm 5.5'te yapılan değerlendirmelerle de gözlenmiştir. Analiz süreci bittiğinde her üçlü deseninin nereden sorgulanacağının belirlenmesi bitirilmiş olur. Bu aşamadan sonra analiz sonuçları

ile birleştirilmiş sorgular oluşturulmadan önce bazı sorgulama eniyileştirmeleri gerçekleştirilir. Bu eniyileştirme adımlarının bazıları gerçekleştirilirken analiz sonuçlarından da faydalanılır. Bu amaçla analiz sonuçları ile ham sorgu Sorgu Eniyileştirici modülüne verilir.

3.2 Sorgu Eniyileştirici

Sorgu eniyileştirme, IBM tarafından bir veritabanı sistemi projesi olan System R[3] ortaya atıldığından beri bir araştırma konusu olmuştur (Bernstein et al., 2007). Anlamsal sorgulama araçları da veb üzerinde yapılan sorgulamaların kısa sürede performanslı bir şekilde çalıştırılabilmesi için sorgu eniyileştirmelerini gerçekleştirmektedir. SPARQL sorgulama dili ile yapılan sorguların eniyileştirimleri *mantıksal* ve *fiziksel eniyileştirmeler* olmak üzere iki sınıfta gruplandırılabilir (Quilitz and Leser, 2008).

Aşağıda eniyileştirme adımları ve Sorgu Eniyileştirici modülü ile nasıl gerçekleştirileceği açıklanmıştır. Ancak bu tez kapsamında bu eniyileştirmelerden sadece bazıları gerçekleştirilmiştir. Bunlar Bölüm 3.2.3'te verilmiştir.

3.2.1 Mantıksal eniyileştirimler

Mantıksal sorgu eniyileştirmelerinin amacı, sorgudaki ifadelerin işlevlerini ve hedeflenen sorgulama sonucunu değiştirmeyecek şekilde sorgunun daha az maliyetle çalışacak bir biçimde yeniden yazılmasıdır. Aşağıda bazı mantıksal sorgu eniyileştirme adımları anlatılmaktadır.

3.2.1.1 Filter bloğunun yeniden yazılması

Sorgu içerisideki FILTER bloğunun içerisindeki yer alan değişkenlere ve sabitlere göre, sorgudaki değişkenlerin yerine sabit değerler koyulabilir ve sorgu yeniden yazılabilir. Örneğin Şekil 3.4a'da FILTER bloğu içerisindeki *firstname* ve *lastname* değişkenleri sabit birer literal ile sınırlandırılmak istenmiştir. Bunu FILTER bloğu içerisinde belirtmek yerine Şekil 3.4b'deki gibi üçlü deseni içerisinde yerine koyarak daha erken bir filtreleme sağlanabilir ve getirilen ara sonuçların sayısı azaltılır. Böylece daha iyi bir sorgulama performansı elde edilir.

[3]http://www.mcjones.org/System_R/

```
SELECT ?person ?firstname ?title
WHERE {
        ?person person:firstname ?firstname .
        ?person person:lastname ?lastname .
        ?person person:age ?age .

        ?publication publication:author ?person .
        ?publication publication:title ?title .

        FILTER (?firstname = "Donald"
                && ?lastname = "Chamberlin" && ?age > 30)
}
```

(a)

```
SELECT ?person ?firstname ?title
WHERE {
        ?person person:firstname  "Donald" .
        ?person person:lastname  "Chamberlin".
        ?person person:age ?age .

        ?publication publication:author ?person .
        ?publication publication:title ?title .

        FILTER (?age > 30)
}
```

(b)

Şekil 3.4: FILTER bloğunun yeniden yazımı

3.2.1.2 Filter bloğunu taşıma

Sorgu içerisindeki FILTER bloğu ne kadar erken işletilirse ara sonuçlar o kadar erken filtrelenir ve daha küçük bir ara sonuç ile sorgulamaya devam edilir. Bunun için FILTER bloğunun sorguda olabildiğince erken işletilmesi için sorgu yeniden yazılır. Şekil 3.5a'daki sorguda görülen FILTER bloğu, içerisindeki *?age* değişkeninin en son kullanıldığı üçlü deseninin hemen ardından Şekil 3.5b'deki gibi yazılmıştır. Böylece mümkün olan en erken şekilde filtreleme yapılıp ara sonuçlar azaltılacaktır.

```
SELECT ?person ?firstname ?title
WHERE {
        ?person person:firstname  "Donald" .
        ?person person:lastname  "Chamberlin".
        ?person person:age ?age .

        ?publication publication:author ?person .
        ?publication publication:title ?title .

        FILTER (?age > 30)

}
```

(a)

```
SELECT ?person ?firstname ?title
WHERE {
        ?person person:firstname  "Donald" .
        ?person person:lastname  "Chamberlin".
        ?person person:age ?age .
        FILTER (?age > 30)
        ?publication publication:author ?person .
        ?publication publication:title ?title .


}
```

(b)

Şekil 3.5: FILTER bloğunu taşıma

3.2.1.3 Üçlü desenlerinin sıralanması

Sorgu çalıştırımı sırasında üçlü desenlerinin hangi sırada yazıldığı sorgulama performansını büyük ölçüde etkilemektedir. Üçlü desenleri aslında verisetinin belirli bir kısmını ifade etmektedir. Bir üçlü deseninin sorgulanması ile verisetinin belirli bir kısmı yani bir ara sonuç getirilir, ondan sonraki üçlü deseninin çalıştırılmasında da getirilen bu sonuçlar biraz daha daraltılır ve bu, tüm üçlü desenleri için tekrar eder. Bu mantığa göre ilk kez sorgulanacak üçlü deseninin olabildiğince az üçlü içeren bir ara sonuç getirmesi, sonrasında yapılacak üçlü deseni sorgulamaları için daha az işlem yükü anlamına gelir. O halde sorgu içerisindeki üçlü desenleri, tek başına sorgulandıklarında

```
:DBpedia a void:Dataset;
    void:classPartition [
        void:class foaf:Person;
        void:entities 312000;
    ];
    void:propertyPartition [
        void:property foaf:name;
        void:triples 312000;
        void:distinctSubjects 312000;
        void:distinctObjects 100000;
    ];
    void:triples 1000000;
    .
```

Şekil 3.6: VOID istatistiksel verileri

en az üçlü içeren ara sonuçları getiren sırada olmalıdır.

Bir üçlü deseninin, kaç üçlü içeren bir ara sonuç getireceğinin bilgisi, veriseti üstverilerinden elde edilebileceği gibi verisetlerine bazı sorgular atılarak da elde edilebilir. Ayrıca verisetinde toplam kaç tane üçlü olduğu, belirli bir kaynağın kaç üçlüde özne olarak bulunduğu, kaç üçlüde nesne olarak bulunduğu veya kaç üçlüde yüklem olarak bulunduğu, bir kaynağın nesne, başka bir kaynağın da yüklem olarak bulunduğu üçlüler gibi ikili kombinasyonlar gibi istatistiki bilgiler de elde edilebilirler. Bu sonuçlara göre üçlü desenleri sorguları performans açısından eniyileştirilmek için yeniden sıralanırlar. Bu veriler VOID belgelerinin ya da verisetlerinin sorgulanmasıyla elde edilebilir.

İstatisiksel veriler üçlü desenlerinin sorgulanacakları veriset(ler)inden veya veriset(ler)inin üstverilerinden elde edileceği için WoDQA sorgu analizi sürecinde üçlü desenleri için yapılan veriseti analizinden sonra istatistiki bilgiler sorgulanmalıdır. Bu bilgilere göre üçlü desenleri yeniden sıralandıktan sonra Sorgu Düzenleyici ile sorgu yeniden yazımı gerçekleştirilir. Üçlü desenlerinin sıralanması için WoDQA iki yaklaşım izlemektedir. Bunlar istatistiki bilgilere dayalı olarak üçlü desenlerinin sıralanması ve sezgisel yöntemlere dayalı olarak üçlü desenlerinin sıralanmasıdır.

VOID depolarının istatistiksel bilgiler için sorgulanması

Bölüm 2.3'te belirtilen VOID depoları içerisindeki VOID belgeleri, verisetinin içerdiği kaynaklarla ilgili bazı istatistikleri de içermektedir. Şekil 3.6'da DBpedia veriseti için bazı kaynaklara ait istatistiksel veriler görülmektedir.

VOID belgelerinde yer alan bu istatistiksel verilerin anlamları aşağıda açıklanmıştır.

void:classParitition: Belirli bir RDFS sınıfına ait istatistiki bilgileri tanımlamada kullanılır. void:class ile hangi sınıf hakkında istatistik tutulduğu, void:entities ile de o sınıf altında kaç adet bileşen tanımlandığı belirtilir.

void:propertyPartition: Belirli bir yükleme ait istatistiksel bilgileri tanımlamada kullanılır. void:property ile hangi yüklem ile ilgili bilgilerin tutulduğu, void:triples ile ilgili yüklemi içeren kaç adet üçlü olduğu, void:distinctSubjects ile ilgili yüklemi içeren ve özne pozisyonunda farklı URI veya boş düğümler içeren kaç tane üçlü olduğu, void:distinctObjects ile ilgili yüklemi içeren ve nesne pozisyonunda farklı URI, boş düğüm veya literal içeren kaç farklı üçlü olduğu tutulur. void:triples aynı zamanda tüm veriseti içerisinde kaç tane üçlü olduğuna dair bilgi tutmamızı da sağlar.

Şekil 3.6'da DBpedia veriseti ile ilgili foaf:Person tipinde kaç tane örnek tanımlandığı ve içerisinde foaf:name bulunduran üçlüler ile ilgili istatistiksel bilgiler tanımlanmıştır. VOID belgelerinde bu tarz istatistikler modellenebilirken daha fazla istatistiksel bilgi için datasetler bazı sorgular[4] ile doğrudan da sorgulanabilirler. Böylece herhangi bir URI'nin özne pozisyonunda veya nesne pozisyonunda olduğu üçlü sayısı bile elde edilebilir.

Üçlü desenlerini istatistiksel bilgilere göre sıralamak için *seçicilik maliyeti tahminlemesi(selectivity cost estimation)*[OPTARQ] yapılır. Bu hesaplamaya göre bir üçlü deseni için olan maliyet $c(t)$; özne için olan maliyetin $c(s)$, yüklem için olan maliyetin $c(p)$ ve nesne için olan maliyetin $c(o)$ çarpımıyla hesaplanır. Üçlü deseni maliyeti 0 ve 1 arasında bir değerdir. Değerin 1'e yaklaşması maliyetin artması, 0'a yaklaşması ise maliyetin azalması anlamına gelir.

$$c(t) = c(s) * c(p) * c(o)$$

Şekil 3.7: Üçlü deseni seçicilik maliyeti hesaplama formülü

Özne maliyeti hesaplaması: Bir üçlü deseninin öznesi bir değişken ya da bir IRI olabilir. Değişken olduğu durumda tüm özneler kast edildiğinden maliyet $c(s) = 1$'dir. IRI olduğu durumda ise verisetindeki toplam tanımlı kaynak sayısına bağlı olarak, maliyet $c(s) = \frac{1}{void : entitites}$.

Yüklem maliyet hesaplaması: Bir üçlü deseninin yüklemi bir değişken ya da bir IRI olabilir. Değişken olduğu durumda tüm yüklemler kast edildiğinden maliyet

[4]http://code.google.com/p/void-impl/wiki/SPARQLQueriesForStatistics

	t	c(s)	c(p)	c(o)	c(t)
1	?s ?p ?o	1.0	1.0	1.0	1.0
2	:s ?p ?o	0.008747	1.0	1.0	0.008747
3	?s rdfs:label ?o	1.0	0.0865604	1.0	0.0865604
4	:s rdfs:label ?o	0.008747	0.0865604	1.0	0.0007571
5	?s rdfs:label "XQuery: A Query ..."	1.0	0.0865604	0.0129081	0.0011173
6	:s rdfs:label "XQuery: A Query ..."	0.008747	0.0865604	0.0129081	0.0000097

Şekil 3.8: Üçlü deseni maliyet değerleri örnekleri

$c(p) = 1$'dir. IRI olduğu durumda ise yüklemin tanımlı olduğu üçlü sayısının verisetindeki toplam üçlü sayısına oranına göre maliyet $c(p) = \frac{p(void:triples)}{void:triples}$.

Nesne maliyet hesaplaması: Bir üçlü deseninin nesnesi bir değişken, bir IRI ya da bir literal olabilir. Değişken olduğu durumda tüm nesneler kast edildiğinden maliyet $c(o) = 1$'dir. Eğer nesne bir IRI veya literal ise nesne maliyeti iki şekilde hesaplanır:

$$c(o) = \left\{ \begin{array}{cc} eğer\, p\, IRI\, ise; & c(p,o) \\ diğer\, durumlarda; & \sum_{p_i \in P} c(p_i, o) \end{array} \right\}$$

Üçlü desenleri için yapılan seçicilik meliyet tahminlemesinden sonra en düşük maliyetten en yüksek maliyete sahip olana doğru üçlü desenleri sıralanırlar.

Sezgisel yöntemlerle sıralama Üçlü desenleri sayısal istatistikler baz alınarak sıralanabileceği gibi bazı sezgisel yöntemlerle de sıralanabilirler. Kesin olan bazı bilgiler sayısal verilere ihtiyaç duyulmadan da elde edilebilir. Sezgisel yöntemlerdeki esas alınan kurallar şunlardır:

- İçerisinde öznesi ile birlikte yüklemi veya nesnesi değişken olmayan üçlü desenleri(birinci sezgisel seviye),
- İçerisinde öznesi bir IRI, yüklemi ve nesnesi değişken olarak bulunan üçlü desenleri(ikinci sezgisel seviye),
- İçerisinde öznesi değişken olan yüklemi ve nesnesi değişken olmayan üçlü desenleri(üçüncü sezgisel seviye),
- İçerisinde öznesi ile birlikte yüklemi veya nesnesi değişken olan üçlü desenleri(dördüncü sezgisel seviye),
- Tüm elemanları değişken olan üçlü desenleri(beşinci sezgisel seviye),

şeklinde üçlü desenleri sıralanmalıdır. Bu sıralama genellikle maliyeti en az olan üçlü deseninden maliyeti en çok olan üçlü desenine göre sıralama yapabilmemizi sağlar.

Buraya kadar üçlü desenlerinin sayısal yöntemlere ve sezgisel yöntemlere dayanarak nasıl sıralandığını tanımladık. Üçlü desenleri sıralamasında bu yöntemler tek başına kullanılabileceği gibi ikisi bir arada da kullanılabilir. Ancak sıralama amacıyla ele alınan üçlü desenlerinin birden fazla verisetinden sorgulandığı durumlarda istatistiksel verilere dayanarak sıralama her veriseti için ayrı ayrı gerçekleştirilmelidir. Çünkü her veriseti için farklı sorgulama maliyetleri hesaplanır ve üçlü desenlerinin sırası değişebilir.

Mantıksal eniyileştirmeler sorguların daha performanslı çalışması için yapıldığı halde bazen daha kötü bir performansa da neden olabilir. Bu nedenle mantıksal eniyileştirmelerin gerçekleştirildiği ve gerçekleştirilmediği durumlarda maliyetin ne kadar olacağı da hesaplanmalıdır. Bu hesaplamalar da fiziksel eniyileştirmelerde farklı sorgu çalıştırma planlarının maliyetlerinin belirlenmesi ile gerçekleştirilir. Ancak bu tez kapsamında herhangi bir fiziksel eniyileştirme adımı gerçekleştirilmemektedir.

3.2.2 Fiziksel eniyileştirimler

Fiziksel eniyileştirmenin amacı sorguların; ağın bant genişliği kısıtı, HTTP istekleri gibi nedenlerden kaynaklanan gecikmelerden daha az etkilenmesi için etkin bir sorgu çalıştırma planının belirlenmesidir. Bunun için sorguların farklı verisetlerinden sorgulanmak üzere düzenlenmiş alt sorgularının nasıl çalıştırılacağına bu eniyileştirmelerle karar verilir. Alt sorguların çalıştırılması için bazı yaklaşımlar vardır. Nested-loop join, bind join, hash join gibi. Bunlardan en önemli iki tanesi açıklanacaktır.

- **Nested-loop join:** Bu yaklaşımda üçlü desenleri tek tek ele alınırlar ve sırayla ilk üçlü deseninden elde edilen ara sonuçlardaki eşleşmeler bir sonraki üçlü deseninde yerine koyularak sorgulamalara devam edilir. Çok büyük ara sonuçlar gelmesine neden olacak seçici olmayan üçlü desenlerinin bulunduğu sorgularda etkindir. Ancak bu yaklaşımın dezavantajı çok fazla ağ isteği oluşturmasıdır. Bir önceki üçlü deseninin sorgulanmasında bulunan her bir eşleşmenin bir sonraki üçlü deseninde yerine koyulup sorgulanması için bir istek oluşturulur. Bu sorun aynı verisetinden sorgulanacak üçlü desenlerinin gruplanmasıyla aza indirgenebilir.

- **Bind join:** Üçlü desenlerinin hepsi tek başına sorgulanırlar ve ara sonuçları, sorgu sonucu oluşturmak üzere yerelde toplanır. Ancak hepsini birden sağlayan ortak çözümler sonuç olarak kabul edilir. Bu yaklaşım az boyutta veriyi sorguladığımızda avantajlıdır. Ağ isteği gönderme sayısı bu yaklaşımda azdır.

3.2.3 WoDQA sorgu eniyileştirme gerçekleştirimleri

WoDQA'nın bu tez kapsamında belirtilen sürümünde veriseti seçilimine ve sorguların birleştirilmesine odaklanıldığından dolayı sorgu eniyileştirimlerinden sadece mantıksal olan bazıları gerçekleştirilmiştir. FILTER bloğunun taşınması ve üçlülerin sıralanması gerçekleştirilen eniyileştirmelerdir. Üçlüler sıralanırken öncelikle sezgisel tabanlı olarak sıralama yapılmıştır. Sezgisel sıralama yapıldıktan sonra üçlü desenleri sezgisel sıralamada bahsedilen sezgisel seviyelere ayrılmış olur. Daha sonra her seviye içerisindeki üçlü desenleri de kendi aralarında istatistiksel bilgilere göre yeniden sıralanır. Bu istatistiksel bilgiler analiz sürecinde üretilen üçlü desenlerinin ilişkili verisetlerindeki üçlülerin sayısıdır. WoDQA bu bilgiyi verisetlerine ait VOID belgelerinden alır. Hangisine ait veriseti daha az üçlü içeriyorsa o daha önce yazılır. Ayrıca WoDQA'nın bu sürümünde bir sezgisel seviye içerisinde birden fazla verisetinden sorgulanacak üçlü desenleri için istatistiksel bilgilere göre sıralama yapılmaz, sadece sezgisel tabanlı sıralama yapılır. Eniyileştirmeler gerçekleştirildikten sonra sorgularda herhangi bir değişiklik gözlenmemesi sorgunun zaten en iyi halinde olması anlamına gelir.

Ham sorgular Sorgu Eniyileştirici modülü ile eniyileştirildikten sonra analiz sürecinde üretilen ilişkili veriseti bilgileri ile birlikte birleştirilmiş sorgular üretilmek üzere Sorgu Düzenleyici modülüne verilir.

3.3 Sorgu Düzenleyici

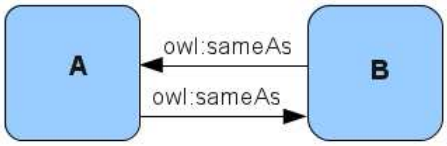
Sorgu Düzenleyici modülü, Veriseti Analizcisi tarafından belirlenen her üçlü deseni için sorgulanacak verisetlerini ($\mathcal{Q}_{BGP}^{final}$) kullanarak sorguların birleştirilmiş olarak yazılmasından sorumludur. Birleştirilmiş sorgular SPARQL 1.1 birleştirilmiş sorgu[5] söz dizimi ve anlamsal biçimine uygun olarak yazılırlar.

WoDQA tarafından analiz edilecek ham sorgular üçlü desenlerinden oluşan basit sorgulardır ve nereden sorgulanacakları belli olmadığından çalıştırılamazlar. Sorgu

[5]http://www.w3.org/TR/sparql11-federated-query/

Düzenleyici bu sorguları alt sorgulara böler ve sorguyu servis çizge desenlerinden oluşan bir biçime dönüştürür. Bir servis çizge deseni alt sorguların sorgulanacağı verisetlerinden[6] (Srv) ve alt sorgunun içerdiği üçlü desenlerinden ($SubTp$) oluşur ve $sgp_\alpha = \langle Srv_\alpha, SubTp_\alpha \rangle$ şeklinde temsil edilir. Bir service çizge desenindeki alt sorgunun çalıştırılması demek, alt sorgunun tüm verisetlerinden ayrı ayrı sorgulanıp alınan sonuçların birleştirilmesi demektir.

Sorguların yeniden yazılması sırasında dikkat edilmesi gereken en önemli nokta alt sorgularda yer alacak olan üçlü desenlerinin belirlenmesidir. WoDQA Sorgu Düzenleyici ile üçlü desenlerini kullanarak alt sorgular oluştururken üçlü desenlerinin sıraları değiştirilmez. Yani alt sorgular oluşturulduktan sonra da üçlü desenlerinin sıraları aynı kalır. Çünkü üçlü desenleri sorgu eniyileştirme amacıyla Bölüm 3.2'de anlatılan bazı yöntemlerle yeniden sıralanmıştır. Bu nedenle aynı verisetlerinden sorgulanacak üçlü desenleri ardışık olmadıkları sürece gruplanamazlar.



Şekil 3.9: Sorgu yeniden düzenleme örnek sorgusu VOID modelleri

Şimdi Sorgu Düzenleyisi'nin üçlü desenleri ile alt sorguları nasıl oluşturduğu açıklanacaktır. Performans nedeniyle aynı verisetlerinden sorgulanacak olan üçlü desenleri aynı alt sorgularda yer almalıdır. Böylece daha az ağ trafiği oluşturularak sorgulamalar çalıştırılır. Ancak bu gruplamayı yaparken dikkat edilmesi gereken bazı noktalar vardır. Bu durumu örnekler üzerinde açıklayalım.

```
?s owl:sameAs ?o.
?o ?p ?x.
```

örnek sorgusunu Şekil 3.9'daki örnek VOID modellerini dikkate alarak inceleyelim. WoDQA Sorgu Analizcisi, bu üçlü desenleri için Zincirleme Üçlü Desenleri Kurallarına göre bazı sonuçlar üretir. Bu sonuçlar hem İçsel hem de Dışsal bakış açılarıyla ele alınarak üretilmiş olan sonuçlardır. İçsel bakış açısıyla baktığımızda her iki üçlü deseni de aynı verisetinden sorgulanabilir ancak dışsal bakış açısıyla baktığımızda her iki üçlü deseni de farklı verisetlerinden sorgulanmalıdır. Buna göre birinci üçlü deseni

[6] Yeniden yazımın gerçekleştiriminde, SERVICE ifadeleri içerisinde verisetlerinin SPARQL uçnoktaları kullanılır.

için *A* veriseti sorgulandığında gelen sonuçlar ikinci üçlü deseninde yerine konulup *B* veriseti sorgulanmalı, ilk üçlü deseni için *B* veriseti sorgulandığında ise gelen sonuçlar yerine konulup *A* veriseti sorgulanmalıdır. Yani her iki üçlü deseni de *A* ve *B* verisetlerinden sorgulanıyor diye aynı alt sorguda yer alırlarsa her ikisi de *A* verisetinden ve *B* verisetinden ayrı ayrı sorgulanmaya çalışılacaktır ve sorgulama sonucunda kayıp olacaktır. Bu nedenle gruplanmaya çalışılan üçlü desenlerinin sorgulanacakları verisetleri arasında dışsal bakış açısıyla bulunmuş ilişkili bir veriseti varsa bu üçlü desenleri aynı alt sorguda yer alamazlar. Gruplanabilecek üçlü desenleri şu iki koşul altında gruplanabilirler:

- Gruplanacak üçlü desenlerinin sorgulanacak ilişkili veriseti sayısı bir olmalı ve aynı veriseti olmalıdır.

- Gruplanacak üçlü desenlerinin sorgulanacak ilişkili verisetleri birden fazla ise hem hepsi birbirinin aynı hem de hiçbiri dışsal bakış açısı ile bulunmuş olmamalıdır.

Hangi üçlü desenlerinin hangi servis çizge desenindeki alt sorgularda yer alacağı Algoritma 3 ile bulunmaktadır.

Algoritma 3 Sorguyu alt sorgulaya parçalayan algoritma

FUNCTION $DivideTriples()$
INPUT $bgp = \{tp_1, ..., tp_n\}$; //$n$ tane üçlü deseni içeren sorgu deseni
LET $i := 1, \alpha := 1$;
$SubTp_\alpha := tp_1$;
WHILE $i < n$ DO
  IF ($\mathcal{Q}_{tp_{i+1}} = \mathcal{Q}_{tp_i}$ && ($|\mathcal{Q}_{tp_{i+1}}| = 1$ || $\exists^0 \rho^{ext}(\delta_x, tp_{i+1})$)) THEN
    LET $SubTp_\alpha := SubTp_\alpha \cup \{tp_{i+1}\}$;
  ELSE
    LET $SubTp_{\alpha+1} := \{tp_{i+1}\}$;
    LET $\alpha := \alpha + 1$;
  LET $i := i + 1$;

Bir alt sorgunun sorgulanacağı verisetleri $Srv_\alpha = \{\delta_x | \delta_x \in \mathcal{Q}_{tp_i}, tp_i \in SubTp_\alpha\}$ şeklinde biçimselleştirilir. Verisetlerinin sorgulanacakları SPARQL uçnoktası URL'leri ise VOID belgelerinden elde edilir. Bir alt sorgu için birden fazla sorgulanacak veriseti olduğunda sorgulanacak SERVICE URL'leri Şekil 3.10'daki gibi *BIND* ve *UNION* anahtar kelimeleri ile birleştirilerek SPARQL sorgusu içerisinde yer alır.

$$\{BIND(< serviceURL_1 > AS\ ?service)\}$$
$$UNION$$
$$\{BIND(< serviceURL_2 >)AS\ ?service\}$$
$$UNION$$
$$...$$
$$\{BIND(< serviceURL_x >)AS\ ?service\}$$
$$SERVICE\ ?service$$

Şekil 3.10: Üçlü desenlerinin birçok verisetinden sorgulandığı servis bloğu

Sorgu Düzenleyici'nin ürettiği çıktı $ReorganizedBGP = \langle sgp_1, \ldots, sgp_m \rangle$ olarak biçimselleştirilir. Burada $1 \leq m \leq n$ ve n ham sorgudaki üçlü desenlerinin sayısıdır. $ReorganizedBGP$ servis çizge desenlerini içeren ham sorgunun birleştirilmiş halidir. WoDQA bu biçimdeki sorguları Jena ARQ sorgu motorunu kullanarak çalıştırır.

Sorgudaki üçlü desenlerinin gruplanmasının yanında, WoDQA bazı sorgu eniyileştirimleri de gerçekleştirmektedir. Bunlar Bölüm 3.2'de açıklanmaktadır. Ayrıca WoDQA sorgu içerisinde kullanılabilecek olan UNION, OPTIONAL, FILTER, LIMIT, OFFSET ifadelerini desteklemektedir. GRAPH ifadesi ve boş düğümler içeren sorguları desteklememektedir.

3.4 Kullanım Durumu

Kullanıcılar WoDQA'yı üç şekilde kullanabilirler. Birincisi ham sorguların WoDQA'nın SPARQL uçnoktasına[7] yönlendirilmesi ile gerçekleştirilir. Bunun için bir SERVICE bloğu içerisine istenilen ham sorgu yazılır ve gönderilecek servis olarak da WoDQA'nın SPARQL uçnoktası adresi verilirse sorgu WoDQA'ya yönlendirilmiş olacaktır. WoDQA SPARQL uçnoktasına gelen sorgu birleştirilmiş bir hale dönüştürülüp kullanıcıdan saydam bir şekilde gerekli verisetleri üzerinde çalıştırılacak ve sonuçlar kullanıcıya döndürülecektir. Şekil 3.11 bu kullanım yöntemi göstermektedir.

```
String queryString = "SELECT * WHERE {dbpedia:Barack_Obama ?p ?o}";
String wodqaEndpoint = "http://seagentdev.ege.edu.tr:8180/joseki/sparql";
Query query = QueryFactory.create(queryString);
QueryExecution queryExecution = QueryExecutionFactory.sparqlService(wodqaEndpoint, query);
ResultSet result = queryExecution.execSelect();
```

Şekil 3.11: WoDQA SPARQL uçnoktası ile sorgu çalıştırımı

İkinci kullanım yönteminde ise WoDQA kütüphanesi doğrudan uygulamanın

[7]WoDQA'nın güncel SPARQL uçnoktası adresi http://seagent.ege.edu.tr/etmen/wodqa.html sayfasında bulunmaktadır.

kaynak kodu içinde kullanılabilir[8]. Daha sonra Şekil 3.12'deki gibi doğrudan kaynak kodları çalıştırarak kullanılabilir. Girdiler tüm kullanılacak VOID belgeleri, çalıştırılmak istenen ham sorgu ve analiz sürecinde ASK sorguları ile daha fazla verisetinin elenmeye çalışılıp çalışılmayacağı bilgileridir. Sonuçta birleştirilmiş bir sorgu üretilir ve normal bir sorgu çalıştırılıyormuş gibi sorgu çalıştırılır.

```
QueryExecutor executor=new QueryExecutor();
String federatedQuery = executor.createQuery(allVoidModels, queryString, askOpt);
Query query = QueryFactory.create(federatedQuery);
QueryExecution queryExecution = QueryExecutionFactory.
                                create(query,ModelFactory.createDefaultModel());
ResultSet result = queryExecution.execSelect();
```

Şekil 3.12: WoDQA kaynak kodu kullanarak sorgu çalıştırımı

WoDQA'nın bir diğer kullanım yöntemi ise WoDQA'nın veb arayüzüdür[9]. Bu arayüzden ham sorgunun yeniden düzenlenmiş biçimi, SELECT ve CONSTRUCT sorgulara gelen sonuçlar ve işletim zamanı gözlemlenebilir. Bu bölümde WoDQA arayüzünde örnek bir sorgu çalıştırımı durum çalışması olarak anlatılacaktır.

Şekil 3.13'te verilen WoDQA arayüzünde görülen sorgu "Alman film yapımcıları tarafından yapılmış filmleri hangi Facebook kullanıcıları beğenmiştir?" sorusuna cevaplar aramaktadır. Bu sorgu analiz edilirken Şekil 3.2'deki örnek VOID modelleri baz alınmıştır. Bu sorgu $BGP =< tp_1, tp_2, tp_3, tp_4, tp_5 >$ olarak biçimselleştirilebilir.
$tp_1 =< ?faceUser, facebook : likes, ?movie >,$
$tp_2 =< ?movie, linkedMDB : producer, ?producer >,$
$tp_3 =< ?dbProducer, owl : sameAs, ?producer >,$
$tp_4 =< ?anyMovie, dbpo : producer, ?dbProducer >,$
$tp_5 =< ?dbProducer, dbpo : birthPlace, dbpedia : Germany > .$

Şimdi Şekil 3.3'teki analiz sürecine göre sorgulanacak ilişkili verisetlerinin nasıl bulunacağı anlatılacaktır. İlk olarak Tek Adım Analiz aşaması altındaki kurallar işletilecektir. $\mathcal{Q}_{tp_1}$ ve $\mathcal{Q}_{tp_2}$ kümesindeki ilişki verisetleri Tek Adım Analizi'ndeki Sözlük Eşleşmesi Kuralı ile elenirler ve son durumları $\mathcal{Q}_{tp_1} = \{\delta_{Facebook}\}$ ve $\mathcal{Q}_{tp_2} = \{\delta_{LinkedMDB}\}$ şeklinde olur. tp_3 için Tek Adım Analiz aşamasında hiçbir ilişkili veriseti bulunamaz, çünkü owl:sameAs genel bir özelliktir ve hiçbir VOID belgesinde sözlük tanımı olarak içerilmez. Bu yüzden $\mathcal{Q}_{tp_3}$ hala tüm verisetlerini içermeye devam eder. tp_4 ve tp_5 için de Sözlük Eşleşmesi Kuralı ile ilişkili verisetleri bulunur ve sorgu-

[8]http://etmen.ege.edu.tr/etmen/snapshots/Seagent/wodqa/ adresinde MAVEN kullanılarak nasıl yararlanılabilceği açıklanmıştır.

[9]http://seagent.ege.edu.tr/etmen/wodqa.html

Şekil 3.13: WoDQA ile örnek bir sorgu çalıştırımı

lanacak verisetleri $\mathcal{Q}_{tp_4} = \mathcal{Q}_{tp_5} = \{\delta_{DBpedia}\}$ olur. Daha sonra da ASK sorguları ile üçlü desenlerinin bu verisetlerinde olduğuna dair olumlu cevaplar alınmıştır.

Tek Adım Analizi tüm üçlü desenlerine uygulandıktan sonra, Yinelemeli Analiz aşaması uygulanır ve tp_3 için sorgulanacak verisetleri burada belirlenip ilişkisiz verisetleri elenirler. tp_3 ve tp_5 aynı özneye sahip olduklarından dolayı Özne Paylaşan Üçlü Desenleri Analizi ile $tp3$'ün sorgulanacak ilişkili verisetleri kümesi belirlenir. Bu kurala göre $\mathcal{Q}_{tp_3} = \{\delta_{DBpedia}\}$ olur.

Şekil 3.13'te üçlü desenleri için yeniden sıralama eniyileştirimi yapılmadan yukarıda açıklanan analiz sonuçları kullanılarak sorgu, Sorgu Düzenleyici tarafından yeniden oluşturulmuştur ve $ReorganizedBGP =< sgp_1, sgp_2, sgp_3 >$ şeklinde temsil edilir. Buradaki servis çizge desenleri

$sgp_1 =< \{\delta_{Facebook}\}, \{tp_1\} >$,

$sgp_2 =< \{\delta_{LinkedMDB}\}, \{tp_2\} >$,

$sgp_3 =< \{\delta_{DBpedia}\}, \{tp_3, tp_4, tp_5\} >$ şeklindedir.

Yeniden düzenleme süreci bittikten sonra servis çizge desenleri içerisindeki üçlü desenleri ilgili SPARQL uçnoktalarından Jena ARQ aracılığı ile çalıştırılırlar ve sorgu sonuçları birleştirilerek toplanır. Sonuçta sorgu ile ilgili elde edilen sonuçlar Şekil 3.13'te görülen sorgulama arayüzü sayfasının alt kısmında görülebilir.

4 DENEYSEL DEĞERLENDİRME

WoDQA'nın analiz süreci için temel gereksinimi tam ve doğru şekilde verisetlerini temsil eden VOID belgelerinin var olmasıdır. Sadece bu VOID belgelerinin varlığı ile WoDQA'yı diğer verisetlerinden farklı kılan veriseti analiz süreci değerlendirilebilir. Ancak var olan VOID depolarındaki VOID belgeleri tam olarak verisetlerinin içeriklerini yansıtmamaktadırlar[1]. Bunun için erişilebilir SPARQL uçnoktalarına sahip olan verisetlerinin VOID'lerini otomatik olarak oluşturan *VOID Oluşturucu* denilen bir araç geliştirilmiştir. Bu araç sayesinde WoDQA'nın sorgulanabilir tüm verisetlerini dikkate alarak analiz yapılması sağlanacak ve WoDQA'nın kayda değer bir değerlendirmesi ve ölçümü yapılabilecektir.

4.1 Otomatik VOID Oluşturucu

VOID Oluşturucu, veb üzerinde var olan herhangi bir VOID deposundan SPARQL uçnoktaları tanımlı tüm verisetlerini belirler ve her bir SPARQL uçnoktasına sorgulara cevap verilebilidiğini test etmek amacıyla basit ASK sorguları atar. Sorgulara cevap veremeyen verisetlerinin VOID'leri oluşturulmaya çalışılmaz. Verisetleri ile ilgili VOID belgelerinin içeriğini oluşturan sözlük bilgilerinin, kaynak tanımları için ortak veri uzayı bilgilerinin ve linkset bilgilerinin oluşturulması verisetleri üzerinde bazı analizler yapılmasını gerektirmektedir. Bu analizler ile VOID belgelerinin içerisindeki bilgilerin nasıl oluşturulduğu ayrıntılı bir şekilde aşağıda açıklanmaktadır.

4.1.1 Kullanılan sözlüklerin belirlenmesi

Her veriseti bazı ontoloji şemalarını yani sözlükleri kullanarak kaynaklarını tanımlamaktadır. Bu sözlükler, foaf, owl, rdf gibi her verisetinde kullanılabilmesi muhtemel çok yaygın sözlükler olabileceği gibi veriseti yayımlayıcıları tarafından oluşturulmuş az yaygın olan bir sözlük de olabilir. Bir verisetinde kullanılan tüm sözlüklerin bulunması için iki çeşit sorgu ile verisetleri sorgulanır. Birincisi

```
Select ?p Where {?s ?p ?o}
```

sorgusu ile bir verisetinde kullanılan tüm yüklemler incelenerek kullanılmış olan her

[1] 15 Nisan 2012 tarihindeki *http://dsi.lod-cloud.net/sparql* üzerinden erişilen VOID belegelerine göre söylenmiştir.

farklı sözlük bulunur.

Select ?o Where {?s rdf:type ?o }

sorgusu ile de kaynakların tiplerini tanımlamak için kullanılan tüm sözlükler bulunur.

Ancak bazı verisetleri çok fazla büyük olduğundan dolayı tek bir sorgu ile tüm verisetinin analizinin yapılması sorguların zaman aşımına uğraması nedeniyle mümkün olmayabilir. Bu yüzden verisetleri parça parça sorgulanmalıdır. Bu da LIMIT ve OFFSET anahtar sözcükleri ile gerçekleştirilebilir.

Select ?p Where {?s ?p ?o} Limit 0 OFFSET 10000

sorgusu verisetinin ilk kaydından itibaren 10000 tane sonuç getirene kadar çalışacaktır. Bu sorgu çalıştırıldıktan sonra OFFSET değeri 10000 olarak ayarlanarak verisetinin 10000. kaydı ile 20000. kaydı arasındaki sonuçlar getirilir. LIMIT ve OFFSET anahtar sözüklerinin bu şekilde kullanımından diğer VOID özellikleri oluşturulurken de yararlanılacaktır.

4.1.2 Kullanılan ortak isim uzaylarının belirlenmesi

Verisetleri kaynak tanımlamaları için genellikle ortak bir veya birkaç isim uzayını kullanırlar. Bunun için veriseti içerisindeki tüm üçlülerin öznelerinin incelenmesi gerekmektedir. Bunun için

Select ?s Where {?s ?p ?o}

sorgusu kullanılır. Yine çok büyük verisetlerinin de sorgulanabilmesi için sorgu LIMIT ve OFFSET anahtar kelimeleri kullanılarak sorgular çalıştırılmalıdır.

4.1.3 Bağseti bilgilerinin belirlenmesi

Bir verisetinin bir verisetine bağlı olması demek, bir veriseti içerisindeki bir kaynağın başka bir veriseti içerisinde tanımlı olan bir kaynağa bir bağ yüklemle bağlı olması demektir. Bir verisetinin linkset tanımlarının oluşturulması için o veriseti içerisindeki nesnesi bir URI olan tüm üçlülerin incelenmesi gerekmektedir. Bu biçimdeki her üçlü için nesnesinin sahip olduğu isim uzayının hangi VOID belgesinde bir isim uzayı tanımı olarak var olduğunun bulunması gerekir. Ancak bunun için öncesinde tüm verisetleri için isim uzaylarının belirlenmiş olması gerekmektedir.

Select ?p ?o Where {?s ?p ?o}

sorgusu sonucu elde edilen bir üçlünün <lgdo:Bursa, owl:sameAs, dbpedia:Bursa>

gibi olduğunu varsayalım. VOID belgesi içerisinde isim uzayı (void:uriSpace) özelliği olarak dbpedia: içeren tüm verisetleri için sorgulanan verisetinin VOID belgesi içerisinde bir linkset tanımı yapılmalıdır. Bağseti tanımlarının hedef veriseti özelliği dbpedia: isim uzayını içeren verisetleridir ve bağ yüklemi de bu üçlüye göre owl:sameAs yüklemidir. Böylece analiz sırasında olası tüm verisetlerinin keşfedilmesi sağlanarak tam ve eksiksiz bir sorgulamaya zemin hazırlanmış olur.

Verisetlerine ilişkin tam ve doğru VOID belgeleri, yukarıdaki yöntemler izlenerek oluşturulmuştur. Ancak bu yöntemler uygulanırken bazı sorunlarla karşılaşılmıştır. Büyük verisetleri için sorgulamaların LIMIT ve OFFSET anahtar sözcükleri kullanılarak gerçekleştirilmesi bir çözüm sağlamıştır. Ancak DBpedia ve LinkedGeoData gibi içerisinde yaklaşık 300 milyon üçlü[2] bulunduran verisetlerinin VOID belgelerinin oluşturulması için tüm bir verisetinin sorgulanmasının tamamlanması ortalama bir bilgisayarda yaklaşık 20 gün sürmektedir. Bu nedenden dolayı bu gibi verisetlerinin çözümlenmesi için daha farklı yöntemler izlenmiştir. Örneğin bağseti bilgileri için veb üzerinde var olan verisetinin hangi verisetlerine bağlandığına dair hazır bilgilerden yararlanılmıştır. Tüm verisetini sorgulamak yerine verisetini oluşturan dump'lar indirilmiş ve kısmi olarak sorgulanmışlardır.

Verisetlerinin içeriklerini yansıtan bu VOID belgeleri, WoDQA sorgulama motorunun sorgulama performansını ölçebilmek için gerekli olan veriseti üstverilerini sağlamaktadır. VOID Oluşturucu'nun bu sürümde üretebildiği VOID belgeleri ile WoDQA kısmen de olsa veri vebi üzerinde test edilmiş olacaktır.

4.2 Değerlendirme

WoDQA'nın temel amacı veri vebi üzerindeki tüm verisetlerini dikkate alarak sorguları analiz etmesi ve çalıştırmasıdır. VOID Oluşturucu ile yaratılan tüm VOID belgelerinin dikkate alınarak sorguların analiz edilmesi ve yeniden oluşturularak çalıştırılması bu tez kapsamında WoDQA'nın değerlendirilmesi için bir öncalışmadır.

VOID Oluşturucu ile *http://dsi.lod-cloud.net/sparql* adresindeki VOID deposu kullanılarak SPARQL uç noktasına erişilebilen ve sorgulanabilen 67 veriseti[3] için VOID belgeleri oluşturulmuştur. Bunların arasında DBpedia, LinkedGeoData, Linked-

[2] 1 Haziran 2012 tarihinde http://dbpedia.org/sparql ve http://linkedgeodata.org/sparql verisetlerine COUNT sorgusu atılarak elde edilen yaklaşık sonuçştur.

[3] Oluşturulan VOID belgeleri içerisindeki verisetlerinin SPARQL uçnoktaları Ek 'te gösterilmiştir.

MDB, NewYorkTimes gibi popüler verisetleri de bulunmaktadır. Şimdi WoDQA'nın *http://code.google.com/p/fbench/* adresinde dağıtık sorgulama ölçümleri yapabilmek için yer alan bazı sorguları hangi sürelerde çalıştırdığı ve sonuçları incelenecektir. Ayrıca sorguların analiz aşamalarındaki elenen veriseti sayıları da gösterilmektedir. Sorgulamalar verisetlerinin daha çok elenmesi amacı ile ASK sorgularının çalıştırılması ve WoDQA sorgu eniyileştirmeleri (üçlüleri sezgisel olarak sıralanması ve filter bloğunun taşınması eniyileştirmeleri) uygulanarak ve uygulanmayarak dört farklı kombinasyonda çalıştırılacaktır. Her bir çalıştırım için Sorgu Düzenleyici ile oluşturulmuş birleştirilmiş sorgular Ek2'de görülmektedir. Aşağıdaki tüm sorgu çalıştırımları 8 çekirdekli 3.07 GHz işlemcili, 8 GB bellekli bir makine üzerinde denenmiştir. Tablolardaki zaman aşımı ise sorguların çalıştırıldığı verisetlerinde uzun süre çalışmasından dolayı zaman aşımına uğrayıp çalışmaması anlamına gelmektedir.

```
SELECT ?predicate ?object
WHERE
{
 { dbpedia:Barack_Obama ?predicate ?object }
   UNION
 { ?subject owl:sameAs dbpedia:Barack_Obama.
   ?subject ?predicate ?object
 }
}
```

ASK Sorguları	Sorgu Eniyileştirme	Süre(sn)	Birleştirilmiş Sorgu
√	√	13,5	BS1
√	χ	13,3	BS2
χ	√	15,9	BS3
χ	χ	15,8	BS4

(a) Sorgu çalışma süreleri

CD1				ASK analizi varsa						ASK analizi yoksa		
	Tek Adım Analiz			ASK Analizi			Yinelemeli Analiz			Yinelemeli Analiz		
	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.
tp1	11	56	56	1	10	66	1	0	66	11	0	56
tp2	18	49	49	3	15	64	3	0	64	18	0	49
tp3	67	0	0	67	0	0	3	64	64	18	49	49
İl.:İlişkili veriseti sayısı, El.:İlgili analiz aşamasında elenen veriseti sayısı, Top El.:Toplam Elenen veriseti sayısı												

(b) Sorgu analiz sürecindeki ilişkili verisetleri sayıları

Tablo 4.1: CrossDomain Sorgu 1

Tablo 4.1'deki sorgu incelendiğinde dbpedia:Barack_Obama adlı kaynağa ait bilgiler ve bu kaynağa owl:sameAs bağ yüklemi ile bağlı kaynaklara ait bilgilerin sorgulanması amaçlanmaktadır. Bunun için var olan tüm VOID belgeleri analiz edilerek dbpedia:Barack_Obama kaynağının bulunduğu verisetleri ve dbpedia: isim uzayına owl:sameAs ile bağlı olan verisetleri de bulunarak sorgulanacaktır. Tablo 4.1'deki ham sorguya bakıldığında üçlü desenlerinin sıralamasının en iyi dizilimde olduğu görülmektedir. Bu yüzden WoDQA eniyileştirme adımı daha da iyi bir sonuç alabilecek herhangi bir eniyileştirme sağlamaz. Ancak EKLER bölümündeki ASK

sorgularının analizde kullanılmadığı BS3 ve BS4 incelendiğinde üçlü desenlerinin pek çok verisetinden sorgulandığı görülmektedir. Bu da daha fazla verisetine istek göndermeye ve gecikmelere neden olur. ASK sorgularının sağladığı bu performans Tablo 4.1'deki tabloda görülmektedir.

```
SELECT ?party ?page WHERE {
 dbpedia:Barack_Obama dbpedia-owl:party ?party .
 ?x nytimes:topicPage ?page .
 ?x owl:sameAs dbpedia:Barack_Obama }
```

ASK Sorguları	Sorgu Optimizasyonu	Süre(sn)	Birleştirilmiş Sorgu
√	√	7,6	BS5
√	χ	9,6	BS6
χ	√	5,0	BS7
χ	χ	6,2	BS8

(a) Sorgu çalışma süreleri

CD2				ASK analizi varsa						ASK analizi yoksa		
	Tek Adım Analiz			ASK Analizi			Yinelemeli Analiz			Yinelemeli Analiz		
	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.
tp1	3	64	64	1	2	66	1	0	66	3	0	64
tp2	1	66	66	1	0	66	1	0	66	1	0	66
tp3	18	49	49	3	15	64	1	2	66	1	17	66
İl.:İlişkili veriseti sayısı, El.:İlgili analiz aşamasında elenen veriseti sayısı, Top El.:Toplam Elenen veriseti sayısı												

(b) Sorgu analiz sürecindeki ilişkili verisetleri sayıları

Tablo 4.2: CrossDomain Sorgu 2

Tablo 4.2'deki sorgu dbpedia:Barack_Obama kaynağının bulunduğu verisetlerindeki dbpedia-owl:party özelliğini ve bu kaynağa owl:sameAs bağ yüklemi ile bağlı diğer verisetlerindeki kaynakların nytimes:topicPage özelliklerinin getirilmesini amaçlamaktadır. Sorgu çalıştırma sürelerine baktığımızda sorgu eniyileştirme uygulanan sorguların daha performanslı çalıştığı gözlenmiştir. Çünkü Tablo 4.2'deki ham sorguda yer alan üçüncü üçlü deseninin seçiciliği içerisinde sadece tek bir değişken olmasından dolayı ikinci üçlü deseninden daha fazladır. Sezgisel yöntemlerle üçlü desenleri sıralamasına göre eniyileştirme uygulanan bu sorguda ikinci ve üçüncü üçlü desenleri yer değiştirirler. Ek2'de BS5 ve BS7'de sorguların eniyileştirilmiş halleri görülmektedir. Ancak ASK sorguları ile analiz gerçekleştirildiğinde sürelerde herhangi bir iyileşme görülmemektedir. Çünkü ASK sorguları kullanılmadan yapılan analizlerde Ek2'deki BS7 ve BS8 sorgularında da görüldüğü gibi sonuç içermeyen çok fazla veriseti yoktur. ASK sorguları ile bu az sayıda sonuç içermeyen verisetleri elenirler ancak ASK sorgularının getirdiği maliyet daha fazla olduğu için az sayıda sonuç içermeyen verisetinin sorgulanması daha etkili bir çalışma sağlamıştır.

Tablo 4.3'e bakıldığında ise dbpedia-owl:nationality özelliği dbpe-

```
SELECT ?pres ?party ?page WHERE {
  ?pres rdf:type dbpedia-owl:President .
  ?pres dbpedia-owl:nationality dbpedia:United_States .
  ?pres dbpedia-owl:party ?party .
  ?x nytimes:topicPage ?page .
  ?x owl:sameAs ?pres }
```

ASK Sorguları	Sorgu Optimizasyonu	Süre(sn)	Birleştirilmiş Sorgu
√	√	8,8	BS9
√	χ	6,4	BS10
χ	√	zaman aşımı	BS11
χ	χ	zaman aşımı	BS12

(a) Sorgu çalışma süreleri

CD3				ASK analizi varsa						ASK analizi yoksa		
	Tek Adım Analiz			ASK Analizi			Yinelemeli Analiz			Yinelemeli Analiz		
	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.
tp1	4	63	63	1	3	66	1	0	66	3	1	64
tp2	3	64	64	1	2	66	1	0	66	3	0	64
tp3	4	63	63	3	1	64	1	2	66	3	1	64
tp4	67	0	0	67	0	0	1	66	66	1	66	66
tp5	1	66	66	1	0	66	1	0	66	1	0	66
İl.:İlişkili veriseti sayısı, El.:İlgili analiz aşamasında elenen veriseti sayısı, Top El.:Toplam Elenen veriseti sayısı												

(b) Sorgu analiz sürecindeki ilişkili verisetleri sayıları

Tablo 4.3: CrossDomain Sorgu 3

dia:United_States, rdf:type özelliği dbpedia-owl:President ve en az bir dbpedia-owl:party özelliği olan kaynaklar ve bu kaynakların owl:sameAs ile bağlı oldukları diğer verisetlerindeki kaynakların nytimes:topicPage özellikleri sorgulanmak istenmektedir. Sorgulama sonuçları incelendiğinde Ek2'de BS11 ve BS12'de var olan bazı ilişkisiz verisetlerinden dolayı sorguların zaman aşımına uğradığı görülür. ASK sorguları ile analiz edilen BS9 ve BS10 birleştirilmiş sorgularında ise ilişkisiz verisetleri daha fazla elendiği için hem Yinelemeli Analiz aşaması daha kısa sürer hem de daha az istek ile daha performanslı bir çalıştırım sağlanmıştır. Üçlü desenlerinin sıralamasında veriseti içerisindeki üçlü sayısı nedeniyle DBpedia veriseti daha fazla üçlü içerdiği için aynı sezgisel seviyede olan üçüncü, dördüncü ve beşinci üçlü desenlerinden üçüncü üçlü deseni sorgunun sonunda yer almak durumunda kalır. Ancak içerisinden bulunan değişkenlere karşılık gelen kaynaklar önceki üçlü desenlerinde yeterince elendiği için çok kritik bir performans katkısı olmaz.

Tablo 4.4'deki sorguda dc:title özelliği *"Tarzan"* olan kaynağın linked-MDB:actor özelliğinin farklı verisetlerinde bulunan owl:sameAs ile bağlı olduğu kaynaklara yine owl:sameAs ile bağlı olan kaynakların nytimes:topicPage özellikleri sorgulanmak istenmektedir. Sorgulama sonuçları incelendiğinde eniyileştirme uygylanan BS13 ve BS15 birleştirilmiş sorgularının çalışmadığı gözlenmektedir. Çünkü veriseti üçlü sayısı nedeniyle seçiciliği son dört üçlü deseninden daha fazla olan beşinci

```
SELECT  ?actor ?news
WHERE
 { ?film dc:title "Tarzan" .
   ?film linkedMDB:actor ?actor .
   ?actor owl:sameAs ?x .
   ?y owl:sameAs ?x .
   ?y nytimes:topicPage ?news.
 }
```

ASK Sorguları	Sorgu Optimizasyonu	Süre(sn)	Birleştirilmiş Sorgu
√	√	zaman aşımı	BS13
√	χ	12,9	BS14
χ	√	zaman aşımı	BS15
χ	χ	1,7	BS16

(a) Sorgu çalışma süreleri

CD4				ASK analizi varsa						ASK analizi yoksa		
	Tek Adım Analiz			ASK Analizi			Yinelemeli Analiz			Yinelemeli Analiz		
	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.
tp1	44	23	23	2	42	65	1	1	66	1	43	66
tp2	1	66	66	1	0	66	1	0	66	1	0	66
tp3	67	0	0	67	0	0	1	66	66	1	66	66
tp4	67	0	0	67	0	0	1	66	66	1	66	66
tp5	1	66	66	1	0	66	1	0	66	1	0	66
İl.:İlişkili veriseti sayısı, El.:İlgili analiz aşamasında elenen veriseti sayısı, Top El.:Toplam Elenen veriseti sayısı												

(b) Sorgu analiz sürecindeki ilişkili verisetleri sayıları

Tablo 4.4: CrossDomain Sorgu 4

üçlü deseni ikinci sıraya getirilimiştir. Bu üçlü deseninin bu sırada sorgulanması zaman aşımına neden olmaktadır. BS14 ve BS16 incelendiğinde ise ASK sorgularının çok fazla maliyete neden olması ve ASK sorguları kullanılmadığında da verisetlerinin elenebiliyor olması nedeniyle BS16 sorgusu daha performanslı çalıştırılmıştır.

Tablo 4.5'deki sorgu dbpedia-owl:director özelliğinin dbpedia-owl:nationality özelliği dbpedia:Italy olan kaynağa owl:sameAs bağ yüklemi ile bağlı olan kaynakların linkedMDB:genre özelliklerini sorgulamayı amaçlamaktadır. Ek2'deki ASK sorguları ile analiz edilen BS17 ve BS18 sorgularına baktığımızda ASK sorguları ile analiz edilmeyen BS19 ve BS20 sorgularından farklı olmadıklarını görürüz. Çünkü ASK sorguları ile analiz yapılmadan da tüm ilişkisiz verisetleri elenmiştir. Eniyileştirme adımı ise BS17 ve BS19'daki sorgularda sezgisel yöntemlere dayanarak ilk üçlü deseni ile ikinci üçlü deseninin yerlerini değiştirir. Çünkü ikinci üçlü deseni diğer tüm üçlü desenlerine göre içerisinde bulunan iki IRI nedeniyle daha seçicidir. Birinci üçlü deseni üçüncü ve dördüncü üçlü desenleri ile aynı sezgisel seviyeye sahiptir ancak birinci üçlü deseni DBpedia verisetinden sorgulanacağından dolayı içerisinde daha fazla üçlü bulunduğu için en alt sıraya yerleştirilir. Bu eniyileştirme adımı nedeniyle üçncü üçlü desenindeki ?film değişkenine karşılık gelen kaynaklar için erken bir filtrelenme sağlanmamış olur ve eniyileştirme adımları uygulanan BS17 ile BS19 sorguları zaman aşımına uğrarlar. Bu örnekteki gibi mantıksal eniyileştirme

```
SELECT ?film ?director ?genre WHERE {
  ?film dbpedia-owl:director ?director.
  ?director dbpedia-owl:nationality dbpedia:Italy .
  ?x owl:sameAs ?film .
  ?x linkedMDB:genre ?genre }
```

ASK Sorguları	Sorgu Optimizasyonu	Süre(sn)	Birleştirilmiş Sorgu
√	√	zaman aşımı	BS17
√	χ	15,5	BS18
χ	√	zaman aşımı	BS19
χ	χ	13,3	BS20

(a) Sorgu çalışma süreleri

CD5				ASK analizi varsa						ASK analizi yoksa		
	Tek Adım Analiz			ASK Analizi			Yinelemeli Analiz			Yinelemeli Analiz		
	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.	İl.	El.	Top El.
tp1	4	63	63	2	2	65	1	1	66	1	3	66
tp2	3	64	64	1	2	66	1	0	66	1	2	66
tp3	67	0	0	67	0	0	1	66	66	1	66	66
tp4	1	66	66	1	0	66	1	0	66	1	0	66
İl.:İlişkili veriseti sayısı, El.:İlgili analiz aşamasında elenen veriseti sayısı, Top El.:Toplam Elenen veriseti sayısı												

(b) Sorgu analiz sürecindeki ilişkili verisetleri sayıları

Tablo 4.5: CrossDomain Sorgu 5

adımları gerçekleştirildiğinde bazen olumsuz sonuçlar ortaya çıkabilir. Bu nedenle sorgu çalıştırma planlarının maliyetleri de hesaplanmalı ve eniyileştirme adımları o şekilde gerçekleştirilmelidir.

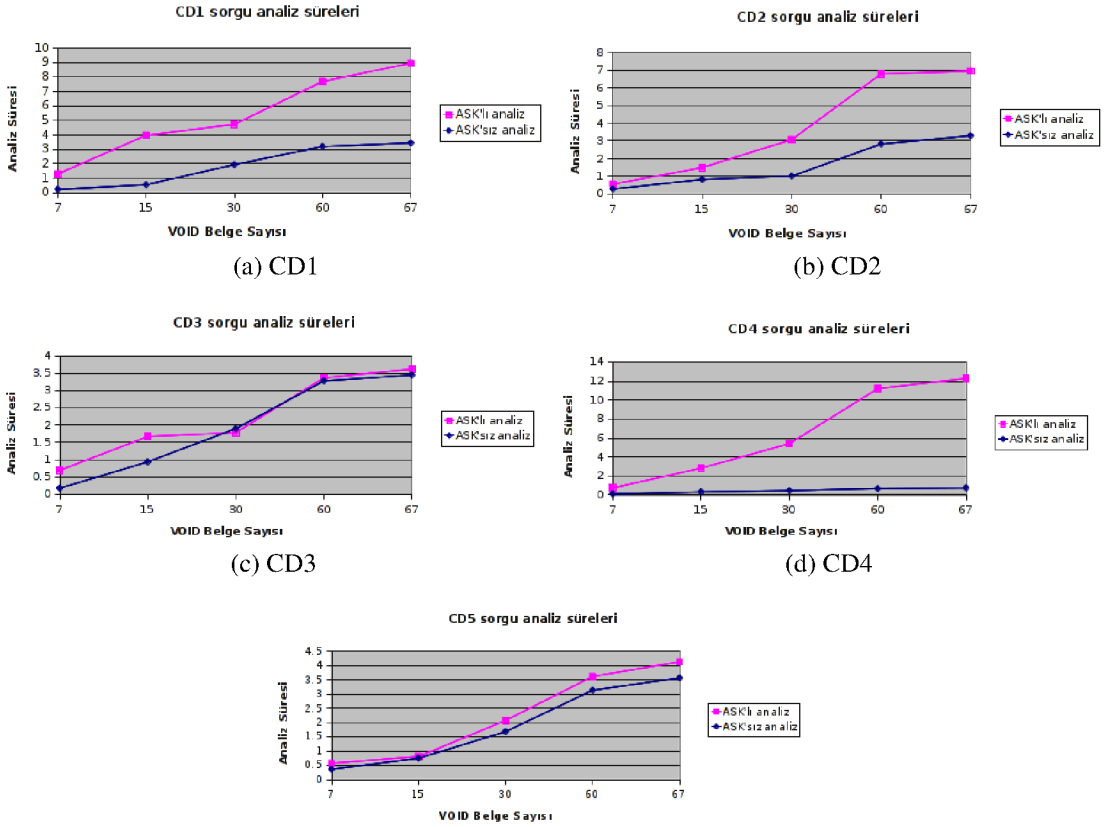
Yukarıdaki sorgular incelendiğinde zaman aşımına uğramayan sorgular hariç hiçbir sorgunun bir sonuç kaybına uğramadığı sadece sorgulama sürelerinin farklı oldukları görülmüştür. WoDQA analiz sürecinde uygulanan ASK sorguları ile daha fazla veriseti eleme yönteminin bazı sorgularda olumlu bazılarında ise olumsuz bir şekilde sorgu çalıştırımını etkilediği gözlemlenmiştir. Bu olumlu etkiler Tek Adım Analiz aşamasından sonra az sayıda verisetine ASK sorgusu atılması durumunda ve ASK sorguları ile analiz uygulanmadan Yinelemeli Analiz aşamasında elenmeyen verisetlerinin olduğu durumda görülmektedir. ASK sorguları bu gibi durumlarda hem daha az istek gönderimi ile sorguların çalışmasını sağlamakta hem de daha çok ilişkisiz verisetini elenmesini sağlamaktadır. Ancak Tek Adım Analiz aşamasından sonra az sayıda ilişkili veriseti bulunduysa bunların içerisindeki ilişkisiz olanları ASK sorguları ile elemeye çalışmanın daha maliyetli olduğu da görülmüştür. Bu gibi durumlarda ASK sorgularının çalıştırılmaması daha etkin sorgu çalıştırımı sağlayabilir. Öte yandan sorgu eniyileştirme adımları da eniyileştirilebilecek ham sorgulara uygulandığında üçlü desenlerinin sıralamasını değiştirerek sorgu çalıştırımı üzerinde bazen olumlu bazen olumsuz etkiler göstermiştir. Çünkü mantıksal eniyileştirmelerin sorgu çalıştırımlarının maliyetini olumsuz etkilemesi söz konusudur. Bu durum da fiziksel

eniyileştirme adımlarının gerçekleştirilmesi ile giderilebilir.

WoDQA sorgu motoru, etkinliği büyük ölçekli veriseti çizgeleri üzerinde çalıştırıldığında görülen, hem sonuç kaybetmeden hem de daha etkin olarak sorgular çalıştırabilen bir araçtır. Diğer sorgu federasyonu yaklaşımını kullanan araçlar küçük ölçekli veriseti çizgelerinde çalıştıklarından dolayı o araçlarla ayrıntılı bir kıyaslama ortaya koyulmamıştır. Ancak analiz süresinin var olan VOID belgesi sayısına göre nasıl değiştiği de Tablo 4.6'daki gibidir.

Sorgu/	CD1		CD2		CD3		CD4		CD5	
VOID_BS	**ASK**	-	**ASK**	-	**ASK**	-	**ASK**	-	**ASK**	-
7	1,283	0,217	0,536	0,266	0,693	0,176	0,745	0,116	0,571	0,362
15	3,957	0,539	1,481	0,804	1,670	0,937	2,828	0,319	0,810	0,747
30	4,720	1,929	3,066	0,995	1,782	1,899	5,416	0,465	2,076	1,684
60	7,687	3,182	6,790	2,816	3,370	3,281	11,222	0,671	3,619	3,137
67	8,955	3,431	6,954	3,289	3,624	3,452	12,294	0,753	4,131	3,568
VOID_BS:VOID Belge Sayısı, CD:CrossDomain. Tablodaki süreler saniye cinsindendir.										

Tablo 4.6: Cross Domain Sorgularının WoDQA ile Analiz Süreleri



(a) CD1

(b) CD2

(c) CD3

(d) CD4

(e) CD5

Şekil 4.1: WoDQA Cross Domain sorguları analiz grafikleri

Tablo 4.6'daki veriler her sorgu için Tek Adım Analiz aşamasından sonra

ASK sorguları kullanılarak ve kullanılmayarak analiz yapılmak üzere farklı sayıda VOID belgeleri ile denenmiştir. Çalışılan VOID belgeleri 67 tane olduğunda sorgu için kesin çözümler bulunmaktedır. Ancak rastgele seçilen 7, 15, 30 ve 60 adet VOID belgeleri arasında sorgular için çözüm içeren verisetlerinin VOID belgeleri olmayabilir. Amaç sadece analiz sürelerini ölçmektir. Her analiz süresi on işletimden elde edilen ortalama sonuçlardır. Her on işletimin her birinde VOID belgeleri tekrar rastgele seçilmiştir. Tablodaki her sorgu için VOID belge sayısı arttıkça analiz sürelerinin artışlarını gözlemlediğimizde sürelerde üstel bir artış olmadığı Şekil 4.1'deki grafiklerden de görülebilmektedir. VOID belge sayısı arttıkça doğrusal artışa yakın bir artış gözlenmektedir.

5 İLGİLİ ÇALIŞMALAR

Anlamsal veb için var olan sorgulama yaklaşımları ele alındığında iki grupta sınıflandırılabilir. Bunlar *merkezi sorgulama* yaklaşımları ve *dağıtık olarak sorgulama* yaklaşımlarıdır. Merkezi sorgulama bağlı verilerin tek bir merkezi veri deposunda toplanması ve verilerin buradan sorgulanması fikrine dayanmaktadır. Bu yaklaşım, önceden belirlenmiş veri depolarını bünyesinde bulunduran *veri ambarlama* ve RDF bağlarını takip ederek ve indeksleyerek veb üzerindeki verileri getiren *arama motorları* yöntemlerini kapsamaktadır(Hartig and Langegger, 2010). Ancak bu yaklaşımın önemli bir dezavantajı sorgulanan verinin o anki güncel veri olmamasıdır. Yani veriler orijinal kaynaktan belli bir zamanda kopyalanmış olan verilerdir. Ayrıca arama motorları da tüm vebi tarayamazlar ve sorgulara tam bir cevap sağlayamazlar. İndekslerin güncelliğine bağlı olarak sorgu sonucu getirilen veriler günceldirler.

Diğer bir yaklaşım olan dağıtık sorgulama yaklaşımında ise, sorgular alt sorgulara ayrılırlar ve bu alt sorgular dağıtık halde bulunan orijinal veri kaynakları üzerinde dağıtılarak işletilir ve sonuçlar elde edilir. Sorgu federasyonu *(Görlitz and Staab, 2011; Haase et al., 2010)* ve bağ dolanımı *(Hartig et al., 2009)* en temel dağıtık sorgulama yaklaşımlarıdır. DARQ (Quilitz and Leser, 2008), FedX (Schwarte et al., 2011a) ve SPLENDID (Görlitz and Staab, 2011) sorgu federasyonu yaklaşımının gerçekleştirilmiş örnekleridir. SQUIN(Hartig et al., 2009) ise bağ dolanımı yaklaşımının bir gerçekleştirimidir.

5.1 DARQ

DARQ, kendisini kullanacak geliştiriciler tarafından oluşturulan verisetleri için üstveriler sunan Service Descriptions[1] denilen belgeleri kullanarak bir sorguyu parçalar ve veri kaynaklarına dağıtır. Sorguları eniyileştirmek içinse üçlüler ve veriseti içerisindeki bazı kavramların ve sınıfların sayıları gibi bazı istatistiklerden yararlanılır. Service Descriptions dosyalarında her bir yüklem için servis tanımlamaları yapılır. Yani içerisinde belli bir yüklemi içeren üçlülerin nereden sorgulanacakları sorgulama zamanından önce burada tanımlanmaktadır. Fakat bu durum, sorgu içinde yüklemi değişken olarak bulunan üçlü desenlerinin nereden sorgulanacağının bulunamaması gibi bir soruna ne-

[1] Service Descriptions ilgili bildiride üçlüler ile ilgili bilgiler, erişim desenleri kısıtları ve istatistiki bilgiler olarak tanımlanmıştır.

den olur[2].

DARQ'ta eniyileştirme için gerçekleştirilen bazı adımlar vardır. Üçlü desenlerinin sorgulanmaları için Service Descriptions dosyalarından yararlanılarak yüklem tabanlı kaynak seçilimi yapılır. Ayrıca eniyileştirme adımı için veriseti içerisinde nesne konumunda kullanılan RDF kavramlarının sayısı ile ilgili bilgiler de tutulur daha etkin seçilim gerçekleştirilebilir. Aynı verisetinden sorgulanacak üçlü desenleri aynı alt sorgular altında gruplanırlar ve FILTER bloğu içerisindeki uygun sabitler sorgu içerisindeki değişkenlerin yerine yeniden yazılarak sorgu yeniden yazımı gerçekleştirilir. Üçlüler yeniden sıralanmazlar. Alt sorguları çalıştırmak için kullanılan planlarda bind join ve nested loop join sorgu çalıştırma yaklaşımlarını kullanır. Sorgu çalıştırım planlarının hangisinin daha etkili olacağına verisetleri için tanımlanmış olan üstverilerdeki bazı istatistikler kullanılarak karar verilir(Quilitz and Leser, 2008).

5.2 FedX

Sorgu federasyonu yaklaşımını kullanarak dağıtık sorgulama yapan bir diğer araç FedX'tir. FedX, AliBaba[3] tarafından sağlanan Fedaration SAIL sorgulama aracının genişletilmiş bir sürümüdür. Federation SAIL aracı da DARQ gibi sorgulanacak verisetlerinin belirlenmesi için Service Descriptions benzeri bir ayar dosyası kullanmaktadır. Burada da yüklem tabanlı veriseti seçilimi yapmak için tanımlamalar yer almaktadır. Ancak FedX sorgulama aracı Federation SAIL'deki gibi ayrıntılı bir ayar dosyası kullanmak yerine sadece sorgulamada hangi verisetlerinin kullanılacağına dair veriler içeren bir üstveriye ihtiyaç duyar. Bu veriler sadece verisetlerinin sorgulanacakları SPARQL uçnoktalarıdır. Bir sorgudaki üçlü desenlerinin her birinin hangi verisetlerinden sorgulanacağının belirlenebilmesi için FedX tüm SPARQL uçnoktaları üzerinde her bir üçlü deseni ile ASK sorguları çalıştırır. Böylece bir üçlü deseninin hangi verisetlerinden sorgulanacağı kesin olarak belirlenir. Sonuçta sorgular belirlenen verisetleri üzerinde hiçbir sonuç kaybedilmeden çalıştırılabilir. Ancak sorgu çalıştırılmadan önce sorguların işletileceği verisetleri en başta belirtilerek sınırlandırıldığı için tüm veb üzerinde sorgu çalıştırılması önlenmiştir. Ayrıca bir üçlü deseninin nereden sorgulanacağının ASK sorgularıyla belirlenmesi üçlü desenini gereksiz yere bazı verisetlerinden sorgulamaya neden olabilir. Çünkü bir üçlü deseninin bazı veriset-

[2]http://darq.sourceforge.net/#Limitations and known issues
[3]http://www.openrdf.org/doc/alibaba/2.0-beta6/alibaba-sail-federation/

lerinden sorgulanmasıyla getirilen sonuçları sorgu içerisindeki diğer üçlü desenleri için uygun eşleşmeler olmayabilir. FedX üçlü desenleri arasındaki ilişkileri dikkate almaz ve üçlü desenlerini gereksiz yere başka verisetlerinden sorgulayabilir.

FedX sorguların daha performanslı bir şekilde çalışması için bazı eniyileştirme adımları gerçekleştirmiştir. FILTER bloğunun sorgu içerisinde uygun yere taşınması ile daha erken bir ara sonuç elemesi yapılır ve sorgulama performansı arttırılır. Ayrıca ASK sorgularının tekrar tekrar aynı üçlüler için atılmasını önlemek amacıyla ASK sorgularının sonuçları bir önbellekte tutulur ve önbellekteki bu bilgiler daha sonra da kullanılır. Sezgisel tabanlı olarak üçlü desenlerinin sorgulanmasının maliyet hesaplamasını yapan ve üçlüleri yeniden sıralayan bir algoritma kullanılır. Ardışık olarak aynı verisetinden sorgulanacak olan üçlü desenleri gruplandırılır ve tek tek gönderilmek yerine grup olarak gönderilir ve istek sayısı azaltılmış olur. Ayrıca alt sorguların çalıştırılması için farklı join yaklaşımları desteklenmektedir(Schwarte et al., 2011b).

5.3 SPLENDID

SPLENDID de sorgu federasyonu yaklaşımını gerçekleştiren araçlardan biridir. Bu araç veriseti tanımlama üst verisi olarak VOID sözlüğünü kullanmaktadır. Ancak ortak kullanılan owl, rdf, rdfs gibi sözlüklerin yüklem olarak kullanıldığı üçlü desenlerini belirlemek için ASK sorguları kullanılabilir. Örneğin

?x rdfs:label “ID_1652”

gibi bir üçlü deseni VOID belgeleri ile analiz edildiğinde rdfs sözlüğü hemen hemen tüm verisetleri tarafından kullanıldığı için herhangi bir veriseti seçilimi yapılamaz. Bu durumda ASK sorguları ile yüklemi rdfs:label ve nesnesi “ID_1652” olan üçlülerin varlığı tespit edilerek veriseti seçilimi yapılabilir. SPLENDID’i diğer sorgu federasyonu yöntemlerine göre farklı kılan VOID sözlüğü içerisindeki bir verisetine ilişkin üçlülerin sayısı, tanımlı özelliklerin sayısı ve buna benzer istatistiki bilgileri kullanarak sorgu eniyileştirmelerini gerçekleştirmesidir.

SPLENDID, sorgu eniyileştirme adımlarını üç kategoriye ayırmıştır: Sorgu yeniden yazımı, sorgulanacak verisetlerinin seçilmesi ve maliyet tabanlı join-order eniyileştirmesi. Sorgu yeniden yazımı olarak FILTER bloğunun gerekirse taşınması ve yeniden yazılmasını gerçekleştirir. Teorik olarak tüm verisetleri bir sorgudaki üçlü desenleri için sorgulanabilir. Üçlü desenlerinin sorgulanacakları verisetlerinin elen-

mesi için sözlük tabanlı veriseti seçilimi için VOID belgeleri, rdfs gibi popüler sözlüklerin yüklem olarak kullanıldığı üçlü desenlerinin veriseti seçilimini yapabilmek için ise ASK sorguları kullanılır. Ayrıca aynı verisetinden sorgulanacak üçlü desenleri tek bir alt sorgu altında gruplanırlar. Farklı join stratejileri ile sorgunun en etkin sorgu planıyla çalıştırılması hedeflenmektedir. Sorgu planlarının maliyetlerinin hesaplanması için VOID belgelerinde yer alan istatistikler kullanılır(Görlitz and Staab, 2011).

5.4 SQUIN

Dağıtık sorgulama yaklaşımlarından bir diğeri olan bağ dolanımının ilk olarak ortaya atıldığı çalışma Olaf Hartig ve arkadaşları tarafından gerçekleştirilen SQUIN(Hartig et al., 2009) sorgulama aracıdır. Burada herhangi bir veriseti üst verisine veya başka bir ön bilgiye ihtiyaç yoktur. Üçlü desenlerinin sorgulanacağı verisetleri sorgu çalıştırımı esnasında keşfedilir. Verisetleri içerisinde tanımlanan kaynaklar arasındaki bağları takip etmeye ve çözümlemeye dayalı bir yaklaşımdır. Sorguların çalıştırılması için içerisindeki öznesi veya nesnesi URI olan bir üçlü deseni gerekmektedir. Başlangıç URI'si denilen bu URI'nin tanımlı olduğu RDF belgesine erişilir ve incelenen üçlü deseni için bu belge içerisinde çözüm aranır. Getirilen bu belge diğer üçlü desenleri için de bir çözüm sunabilir. Getirilen belge içerisindeki çözümlenmesi gereken diğer URI'ler de aynı şekilde çözümlenir ve RDF belgelerine erişilir ve bu RDF belgelerinin sunduğu çözümler birleştirilir. Bu durumu bir örnek ile açıklayalım.

```
:Ziya foaf:knows ?arkadas.
?arkadas ?p ?bilgiler.
```

gibi bir sorgu örneği için ilk üçlü deseninde :Ziya kaynağının tanımlı olduğu RDF belgesi getirilir. Buradaki üçlüler :Ziya kaynağının foaf:knows ile bağlı olduğu kaynakların URI'lerini nesne olarak bulundurmaktadır. İkinci üçlü deseni için gerekli kaynaklar ise :Ziya kaynağının foaf:knows ile bağlı olduğu kaynakların RDF belgelerinde yer almaktadır. İlgili belgeler çözümlenip sonuçlar birleştirildiğinde sorgu tamamlanmış olur. Ancak görüldüğü gibi bu yaklaşımda bir başlangıç URI'sine ihtiyaç duyulmaktadır. Getirilen RDF belgelerinin çok büyük belgeler olması ve URI çözümlemelerinin çok uzun sürmesi ihtimali de performansı olumsuz etkileyebilecek faktörlerdir. Ayrıca sorguların başarılı olarak çalıştırılması için sorgu içerisindeki üçlü desenlerinin sıralaması da önemlidir.

Üçlü desenlerinin sorgulanırken klasik döngüsel paradigma ile sorgulanması diğer üçlü desenlerini sorgulamadan önce beklemeye neden olmaktadır. SQUIN, bu gecikmeyi önleyerek sorgu çalışıtırımının performansını arttırmak amacıyla *engellenemez döngüsel paradigma* denilen bir yaklaşım kullanarak paralel bir şekilde üçlü desenlerinin sorgulanmasını sağlamaktadır(Hartig et al., 2009). Bu yaklaşımda herhangi bir üst veri kullanılmadığı için, sorgu eniyileştirimi için herhangi bir istatistiki veri kullanılmaz.

Bu yaklaşım daha sonra Hartig vd. tarafından biraz daha geliştirilmiştir. Sorgunun üçlü desenlerinin ele alınma sırası için sezgisel bir yöntem geliştirmişlerdir ve bununla beraber sorgulama maliyeti düşürülmüş ve daha kapsamlı sonuçlar sağlanmıştır(Hartig, 2011). Fakat sorguların başarılı bir şekilde işletilmesi ve sonuçlar alınabilmesi için başlangıç URI'sine ve üçlülerin işlenme sırasına hala sıkı sıkıya bir bağımlılık vardır.

5.5 Diğer Yaklaşımlar ve WoDQA

Yukarıda incelenen sorgu federasyonu yaklaşımlarında sorgulanan verisetleri birbirine bağlı olmasına rağmen kaynaklar arasında var olan ve aynı zamanda da verisetlerini birbirine bağlayan bağlar göz ardı edilmektedir. Bu nedenle bu sorgu federasyonu gerçekleştirimlerinin vebi sorgulama açısından bazı eksiklikleri vardır. Bu eksikliklerin en önemlisi yüklem tabanlı veriseti seçiliminin etkin olmayışıdır. Çünkü foaf:knows, owl:sameAs, rdfs:label gibi bazı yüklemler her verisetinde kullanılabilirler ve bunların kullanıldığı verisetlerini yüklem tabanlı veriseti seçilimi ile belirlemek imkansızdır. Ayrıca yüklemler üçlü desenleri içerisinde değişken olarak da var olabilirler, bu durumda da sorgulanacak verisetleri için hiçbir seçilim yapılamaz. Bu durumda ASK sorguları ile verisetlerini belirlemek üçlü desenleri için veriseti seçiliminde daha etkin bir yöntem olabilir. Ancak bu yöntemle üçlü desenlerinin bulunduğu verisetlerini belirlemek hem fazladan sorgulama maliyeti doğurur hem de verisetleri arasındaki bağlar ve üçlü desenleri arasındaki ilişkiler dikkate alınmadığından etkin bir biçimde seçilim sağlanmaz. Çünkü yaygın olarak kullanılan yüklemlerin bulunduğu üçlü desenleri hemen hemen her verisetinden sorgulanabileceği için etkin bir veriseti seçilimi yapılmamış olur. Ayrıca ilişkisiz verisetlerinin daha etkin bir biçimde elenebilmesi için verisetleri arasındaki bağların da sorgu içerisindeki üçlü desenlerinin birbirleri ile olan

ilişkilerinin de dikkate alınması gerekmektedir.

WoDQA sorgulama motorunun diğer sorgu federasyonu yaklaşımlarına göre sahip olduğu en önemli fark etkin veriseti seçilimi mekanizmasıdır. Performans ve tamlık açısından ilk yapılması gereken iş veriseti seçiliminin etkin bir şekilde gerçekleştirilmesidir. Bunu sorgu çalıştırımından önce hem üçlü desenleri arasındaki ilişkileri hem de verisetleri arasındaki bağları dikkate alarak gerçekleştirir. Diğer sorgu federasyonu araçlarından FedX, veriseti seçilimi için ASK sorgularını kullanmaktadır. Tablo 5.1 incelendiğinde sorguların 67 VOID belgesi için analiz edilmesi her üçlü deseni için 67 ASK sorgusunun çalıştırılması demektir. Bu da üçlü deseni sayısı ve çalışılan VOID belgesi sayısı fazlalaştıkça artan bir süre demektir. ASK sorguları ile belirlenen verisetlerinden gelecek olan birçok sonuç diğer üçlü desenleri için sorgulanan verisetlerinden gelen birçok sonuçla eşleşmeyeceği için pek çoğu gereksiz yere sorgulamaya dahil edilmiş olur. SPLENDID sorgulama aracı ise WoDQA gibi VOID belgelerini ve ASK sorgularını kullanmaktadır. Analiz etme süresi VOID belgelerinin sayısının artması ile artmaz ancak verisetleri arasındaki bağlar analiz edilmediğinden çok fazla ilişkisiz veriseti de sorgulamaya dahil edilebilir. Bu nedenle sorgulama süreleri uzayacaktır. Tablo 5.1'de 67 VOID belgesinin varlığında analiz edilen bazı sorguların üçlü desenleri için FedX, SPLENDID ve WoDQA araçları ile bulunan ilişkili verisetleri sayıları verilmiştir. Ayrıca her yaklaşımın her sorgu için attığı ASK sorguları sayısı da tabloda belirtilmiştir. DARQ sorgulama motoru bu tabloya dahil edilmemiştir. Çünkü veriseti seçim mekanizmasının uygulanabilirliği çok kısıtlıdır. Yüklem tabanlı veriseti seçilimi yaparken içerisinde yüklemi değişken olan her sorguyu çalıştıramaz ve çalıştırılabilecek sorgular ciddi bir şekilde kısıtlıdır. Ayrıca sorguların analizinde kullanılacak olan servis tanımlama dosyaları kullanıcılar tarafından tanımlanmak zorundadır. Verisetleri arasındaki bağlar da dikkate alınmamaktadır.

		FedX	SPLENDID	WoDQA(ASK'lı)	WoDQA(ASK'sız)
CD1	tp1	2	2	1	11
	tp2	4	4	3	18
	tp3	64	64	3	18
Atılan Toplam ASK sorgusu :		201	201	29	0
CD2	tp1	2	1	1	3
	tp2	1	1	1	1
	tp3	4	4	1	1
Atılan Toplam ASK sorgusu :		201	71	22	0
CD3	tp1	2	1	1	3
	tp2	2	1	1	3
	tp3	4	4	1	3
	tp4	50	67	1	1
	tp5	1	1	1	1
Atılan Toplam ASK sorgusu :		335	8	12	0
CD4	tp1	2	2	1	1
	tp2	2	1	1	1
	tp3	50	67	1	1
	tp4	50	67	1	1
	tp5	1	1	1	1
Atılan Toplam ASK sorgusu :		335	44	46	0
CD5	tp1	3	4	1	1
	tp2	2	1	1	1
	tp3	50	67	1	1
	tp4	2	1	1	1
Atılan Toplam ASK sorgusu :		268	4	8	0

Tablo 5.1: Veriseti Seçilimi Karşılaştırmaları

6 SONUÇ

Bu tezde, verisetleri için tanımlanmış olan VOID üstverilerini kullanarak sorguları verisetleri üzerinde dağıtık bir şekilde çalıştıran WoDQA sorgulama motoru ortaya koyulmuştur. WoDQA'nın diğer sorgulama motorlarına göre sahip olduğu yenilik veriseti seçilimini eksiksiz bir şekilde gerçekleştiren bir analiz sürecine sahip olmasıdır. Bunu da sorgulardaki üçlü desenleri arasındaki ilişkileri ve VOID denilen güçlü veriseti üstverisi tanımlama sözlüğü ile tanımlanmış verisetleri arasındaki ilişkileri dikkate alarak gerçekleştirmektedir. Kural tabanlı bir yaklaşım ile ilişkisiz verisetlerinin elenerek ilişkili olanların belirlenmesi sağlanır. Bu yaklaşımda kullanılacak VOID tanımlarının, verisetlerinin sorgulanabilmeleri için gerekli olan SPARQL uçnoktası tanımlarını içermeleri, veriseti içerisinde kullanılan sözlükleri ve kaynak tanımlama desenlerini tam ve doğru olarak yansıtmaları ve diğer verisetleri ile olan ilişkilerini tam olarak temsil etmeleri gerekmektedir. WoDQA, kullanıcılarına sorgunun alt sorgulara nasıl bölüneceğini ve hangi alt sorgunun nerede çalıştırılacağını bilmeden ham sorguları çalıştırmalarını sağlar. Sorgular için bulunan sonuçlar tüm VOID tanımlarının eksiksiz ve doğru olması durumunda tamdır.

WoDQA'nın bu ilk sürümünde sorgu federasyonu yaklaşımının kullanılmasından dolayı doğan bazı dezavantajlar vardır. Daha önceden de belirtildiği üzere bağ dolanımı yaklaşımında bazı sonuçları kaybedilebilir ve çok büyük RDF belgeleri getirilmeye çalışılmak zorunda kalınabilirdi. Buna benzer sorunlar sorgu federasyonu yaklaşımında da ortaya çıkabilir. İlk olarak, sorgulardan tam sonuçlar elde edebilmek için verisetlerinin üstverileri iyi bir şekilde ve tam olarak tanımlanmış olmalıdır. Gerçekleştirdiğimiz VOID Oluşturucu aracı verisetlerinin içeriklerini yansıtacak VOID belgelerini yaratır ancak bu tanımlamaların verisetinin değişimlerine göre sürekli güncellenmesi gerekmektedir. VOID Oluşturucu'nun aynı zamanda VOID belgeleri üzerinde güncelleme işlemleri de yapabilmesi gerekmektedir. Sorgu federasyonu yaklaşımından kaynaklanan bir diğer olası sorun ise sorgulanan SPARQL uçnoktalarının sorgulandıkları anlarda servis verememesi nedeniyle sonuç kaybına uğramaktır.

Veriseti yayımlayıcılarının verilerini yapılandırmalarından da kaynaklanan sorunlar olabilir. Örneğin bir veriseti yayımlayıcısı veriseti içerisindeki verilerine ulaşılması için bağlı veri prensipleri gereği verilerini diğer verisetindeki verilerle ilişkilendirmelidir. Böylelikle bu verisetinin WoDQA sorgu motoru tarafından keşfi

mümkün olur.

Bir diğer karşılaşılabilecek sorun ise verisetlerinin çok fazla elenemeyip çok fazla sorgulama yapılması nedeniyle veya eniyileştirilmemiş sorgular yüzünden gecikmelerin yaşanması olabilir. Daha etkin veriseti seçilim mekanizmalarının geliştirilmesi ve sorguların eniyileştirilmesi bu sorunlar için çözüm olabilir. Üçlü desenlerinin çalıştırılma sıralarının değiştirilmesi, bazı fiziksel sorgulama eniyileştirmeleri ve daha fazla veriseti istatistiğinin dikkate alınması gerçekleştirilebilecek bazı bazı eniyileştirme adımlarıdır. WoDQA'nın bu ilk sürümünde daha çok veriseti seçilimi üzerine yoğunlaşıldığından dolayı sorgu eniyileştirme belli bir seviyede gerçekleştirilmiştir. İleriki çalışmalarda üçlü desenlerinin seçiliminin daha iyi olmasını sağlamak için daha fazla istatistiksel bilgi dikkate alınarak gecikmeler önlenebilir ve veriseti seçilimi daha iyi hale getirilebilir.

KAYNAKLAR DİZİNİ

Alexander, K., Cyganiak, R., Hausenblas, M. and Zhao, J., 2009, Describing linked datasets - on the design and usage of voiD, the 'Vocabulary of Interlinked Datasets', WWW 2009 Workshop : Linked Data on the Web (LDOW2009), Madrid.

Berners-Lee, T., 2006, Linked Data - Design Issues, http://www.w3.org/DesignIssues/LinkedData.html (Erişim tarihi: 10 Aralık 2011).

Berners-Lee, T., Fielding, R. and Masinter, L., 1998, Uniform resource identifiers (uri): Generic syntax, Society, 2396(2396).

Bernstein, A., Kiefer, C. and Stocker, M., 2007, OptARQ: A SPARQL optimization approach based on triple pattern selectivity estimation, Technical Report ifi-2007.03, Zurich.

Bizer, C., Heath, T., Ayers, D. and Raimond, Y., 2007, Interlinking open data on the web, www4.wiwiss.fu-berlin.de/bizer/pub/LinkingOpenData.pdf (Erişim tarihi: 12 Mayıs 2009).

Bizer, C., Heath, T. and Berners-Lee, T., 2009, Isim, Linked data – the story so far, Int. J. Semantic Web Inf. Syst., 5(3), 1-22p.

Buil, C., Arenas, M. and Corcho, O., 2011, Semantics and optimization of the SPARQL 1.1 federation extension, Proc. of 8th Extended Semantic Web Conference (ESWC 2011), 6644, Crete, 1-15p..

Duerst, M. and Suignard, M., 2005, Internationalized resource identifiers (IRIs), IETF.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T., 1999, RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1, The Internet Society.

Görlitz, O. and Staab, S., 2011, Federated data management and query optimization for linked open data, New Directions in Web Data Management 1, 331, Germany, 109-137p.

Görlitz, O. and Staab, S., 2011, SPLENDID: SPARQL endpoint federation exploiting VOID descriptions, Proceedings of the 2nd International Workshop on Consuming Linked Data, Bonn.

KAYNAKLAR DİZİNİ (devam)

Haase, P., Mathäβ, T. and Ziller, M., 2010, An evaluation of approaches to federated query processing over linked data, Proceedings of the 6th International Conference on Semantic Systems, New York.

Hartig, O., 2011, Zero-Knowledge query planning for an iterator implementation of link traversal based query execution, ESWC, 6643, Crete, 154-169p.

Hartig, O., Bizer, C. and Freytag, J. C., 2009, Executing SPARQL queries over the web of linked data, International Semantic Web Conference, Washington, 293-309p.

Hartig, O. and Langegger, A., 2010, A database perspective on consuming linked data on the web, Datenbankspektrum.

Heath, T. and Bizer, C., 2011, Linked Data: Evolving the Web into a Global Data Space, Morgan & Claypool, San Rafael, CA, 1 edition.

Quilitz, B. and Leser, U., 2008, Querying distributed RDF data sources with SPARQL, The Semantic Web: Research and Applications, 5021, Berlin, 524-538p.

Schwarte, A., Haase, P., Hose, K., Schenkel, R. and Schmidt, M., 2011, FedX: A federation layer for distributed query processing on linked open data, The Semanic Web: Research and Applications, 6644, Walldorf, 481-486p.

Schwarte, A., Haase, P., Hose, K., Schenkel, R. and Schmidt, M., 2011, FedX: Optimization techniques for federated query processing on linked data, ISWC 2011, 7031, Bonn, 601-616p.

Sheth, A. P. and Larson, J. A., 1990, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Comput. Surv., 22(3), New York, 183-236p.

ÖZGEÇMİŞ

Ad Soyad	Ziya Akar
Doğum Tarihi	08.09.1987
Doğum Yeri	Kırcali
Uyruğu	Türkiye Cumhuriyeti
Eğitim	
Yüksek Lisans	2009-... Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı
Lisans	2005-2009 Ege Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
Lise	2001-2005 Bursa Çelebi Mehmet Lisesi
Araştırma Alanları	Anlamsal veb, etmen tabanlı yazılım geliştirme, dağıtık sistemler, bağlı veri, dağıtık sorgulama

Yayın Listesi

1. Ziya Akar, Tayfun Gökmen Halaç, Erdem Eser Ekinci, Oguz Dikenelli. "Querying the Web of Interlinked Datasets by Using VOID Descriptions", Linked Data On the Web Workshop on World Wide Web Conference, 2012.

EKLER

Ek1 Türkçe - İngilizce Terimler Sözlüğü

Anlamsal: Semantic

Alt sorgu: Subquery

Bağ: Link

Bağlı veri bulutu: Linked data cloud

Bağlı açık veri bulutu: Linked open data cloud

Bağ dolanımı: Link traversal

Birleştirilmiş sorgu: Federated query

Belirtim: Assertion

Boş düğüm: Blank node

Bağseti: Linkset

Biçimselleştirme: Formalization

Çizge: Graph

Dağıtık: Distributed

Değişken: Variable

Döngüsel: Iterator

Eniyileştirme: Optimization

Engellenemez döngüsel paradigma: Non-blocking iterator paradigm

Hedef veriseti: Referenced dataset

Ham sorgu: Raw query

İlişkili veriseti.: Relevant dataset

İlişkisiz veriseti: Irrelevant dataset

Kaynak: Resource

Kaynak veriseti: Referrer dataset

Nesne: Object

Önek: Prefix

Özne: Subject

Önbellek: Cache

Parçacıklı: Granularity

Servis çizge deseni: Service graph pattern

Sorgu federasyonu: Query federation

Söz dizimi: Syntax

Sözlük : vocabulary

Tamlık: Completeness

Uçnokta: Endpoint

Üçlü: Triple

Üçlü deseni: Triple pattern

Üstveri: Metadata

Veri ambarlama: Datawarehousing

Veri deposu: Data store

Veriseti: Dataset

Veriseti keşif kuralları: Dataset discovery rules

VOID deposu: VOID store

Yayımlayıcı: Publisher

Yüklem: Predicate

Ek2 Birleştirilmiş Sorgular

BS1:

```
SELECT  ?predicate ?object
WHERE
 {   { SERVICE <http://dbpedia.org/sparql>
       { dbpedia:Barack_Obama ?predicate ?object }
     }
   UNION
     {   { BIND(<http://el.dbpedia.org/sparql> AS ?ser684197) }
       UNION
         { BIND(<http://dbpedia.org/sparql> AS ?ser684197) }
       UNION
         { BIND(<http://api.talis.com/stores/nytimes/services/sparql> AS ?ser684197) }
       SERVICE ?ser684197
         { ?subject owl:sameAs dbpedia:Barack_Obama.
           ?subject ?predicate ?object
         }
     }
 }
```

BS2:

```
SELECT  ?predicate ?object
WHERE
 {   { SERVICE <http://dbpedia.org/sparql>
       { dbpedia:Barack_Obama ?predicate ?object }
     }
   UNION
     {   { BIND(<http://el.dbpedia.org/sparql> AS ?ser115282) }
       UNION
         { BIND(<http://dbpedia.org/sparql> AS ?ser115282) }
       UNION
         { BIND(<http://api.talis.com/stores/nytimes/services/sparql> AS ?ser115282) }
       SERVICE ?ser115282
         { ?subject owl:sameAs dbpedia:Barack_Obama.
           ?subject ?predicate ?object
         }
     }
```

BS3:

```
SELECT  ?predicate ?object
WHERE
 {   {   { BIND(<http://api.talis.com/stores/theviewfrom/services/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://data.archiveshub.ac.uk/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://api.talis.com/stores/productdb/services/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://data.fundacionctic.org/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://sparql.linkedopendata.it/musei> AS ?ser894009) }
      UNION
        { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://www.morelab.deusto.es/joseki/articles> AS ?ser894009) }
      UNION
        { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://api.talis.com/stores/climb/services/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://en.openei.org/sparql> AS ?ser894009) }
      UNION
        { BIND(<http://dbpedia.org/sparql> AS ?ser894009) }
      SERVICE ?ser894009
        { dbpedia:Barack_Obama ?predicate ?object }
    }
   UNION
    {   { BIND(<http://greek-lod.math.auth.gr/fire-brigade/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://en.openei.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://el.dbpedia.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://geo.linkeddata.es/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://linkedgeodata.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://dbpedia.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://api.talis.com/stores/nytimes/services/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://linkedscotland.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://data.linkedmdb.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://api.talis.com/stores/theviewfrom/services/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://data.archiveshub.ac.uk/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://api.talis.com/stores/productdb/services/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://data.fundacionctic.org/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://sparql.linkedopendata.it/musei> AS ?ser056305) }
      UNION
        { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://www.morelab.deusto.es/joseki/articles> AS ?ser056305) }
      UNION
        { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser056305) }
      UNION
        { BIND(<http://api.talis.com/stores/climb/services/sparql> AS ?ser056305) }
      SERVICE ?ser056305
        { ?subject owl:sameAs dbpedia:Barack_Obama.
          ?subject ?predicate ?object
        }
    }
 }
```

BS4:

```
SELECT  ?predicate ?object
WHERE
 {  {  { BIND(<http://api.talis.com/stores/theviewfrom/services/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://data.archiveshub.ac.uk/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://api.talis.com/stores/productdb/services/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://data.fundacionctic.org/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://sparql.linkedopendata.it/musei> AS ?ser894009) }
     UNION
      { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://www.morelab.deusto.es/joseki/articles> AS ?ser894009) }
     UNION
      { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://api.talis.com/stores/climb/services/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://en.openei.org/sparql> AS ?ser894009) }
     UNION
      { BIND(<http://dbpedia.org/sparql> AS ?ser894009) }
     SERVICE ?ser894009
      { dbpedia:Barack_Obama ?predicate ?object }
   }
  UNION
   {  { BIND(<http://greek-lod.math.auth.gr/fire-brigade/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://en.openei.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://el.dbpedia.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://geo.linkeddata.es/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://linkedgeodata.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://dbpedia.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://api.talis.com/stores/nytimes/services/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://linkedscotland.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://data.linkedmdb.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://api.talis.com/stores/theviewfrom/services/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://data.archiveshub.ac.uk/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://api.talis.com/stores/productdb/services/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://data.fundacionctic.org/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://sparql.linkedopendata.it/musei> AS ?ser056305) }
     UNION
      { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://www.morelab.deusto.es/joseki/articles> AS ?ser056305) }
     UNION
      { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser056305) }
     UNION
      { BIND(<http://api.talis.com/stores/climb/services/sparql> AS ?ser056305) }
     SERVICE ?ser056305
      { ?subject owl:sameAs dbpedia:Barack_Obama.
        ?subject ?predicate ?object
      }
   }
 }
```

BS5:

```
SELECT  ?party ?page
WHERE
 { SERVICE <http://dbpedia.org/sparql>
    { dbpedia:Barack_Obama dbpedia-owl:party ?party }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
    { ?x owl:sameAs dbpedia:Barack_Obama.
      ?x nytimes:topicPage ?page
    }
 }
```

BS6:

```
SELECT  ?party ?page
WHERE
 { SERVICE <http://dbpedia.org/sparql>
    { dbpedia:Barack_Obama dbpedia-owl:party ?party }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
    { ?x nytimes:topicPage ?page .
     ?x owl:sameAs dbpedia:Barack_Obama
    }
 }
```

BS7:

```
SELECT  ?party ?page
WHERE
 {   { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser008575) }
  UNION
   { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser008575) }
  UNION
   { BIND(<http://dbpedia.org/sparql> AS ?ser008575) }
  SERVICE ?ser008575
   { dbpedia:Barack_Obama dbpedia-owl:party ?party }
  SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
   { ?x owl:sameAs dbpedia:Barack_Obama.
     ?x nytimes:topicPage> ?page
    }
 }
```

BS8:

```
SELECT  ?party ?page
WHERE
 {   { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser008575) }
  UNION
   { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser008575) }
  UNION
   { BIND(<http://dbpedia.org/sparql> AS ?ser008575) }
  SERVICE ?ser008575
   { dbpedia:Barack_Obama dbpedia-owl:party ?party }
  SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
   { ?x nytimes:topicPage> ?page
     ?x owl:sameAs dbpedia:Barack_Obama.
    }
 }
```

BS9:

```
SELECT  ?president ?party ?page
WHERE
 { SERVICE <http://dbpedia.org/sparql>
    { ?president rdf:type dbpedia-owl:President.
     ?president dbpedia-owl:nationality dbpedia:United_States
    }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
    { ?x owl:sameAs ?president .
     ?x nytimes:topicPage ?page
    }
   SERVICE <http://dbpedia.org/sparql>
    { ?president dbpedia-owl:party ?party }
 }
```

BS10:

```
SELECT  ?president ?party ?page
WHERE
 { SERVICE <http://dbpedia.org/sparql>
    { ?president rdf:type dbpedia-owl:President.
     ?president dbpedia-owl:nationality dbpedia:United_States.
     ?president dbpedia-owl:party ?party

    }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
    { ?x owl:sameAs ?president .
     ?x nytimes:topicPage ?page
    }
 }
```

BS11:

```
SELECT  ?president ?party ?page
WHERE
 {   { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser439692) }
   UNION
    { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser439692) }
   UNION
    { BIND(<http://dbpedia.org/sparql> AS ?ser439692) }
   SERVICE ?ser439692
    { ?president rdf:type dbpedia-owl:President.
      ?president dbpedia-owl:nationality dbpedia:United_States
    }
    SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
     { ?x owl:sameAs ?president .
       ?x nytimes:topicPage ?page
     }
     { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser235671) }
   UNION
    { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser235671) }
   UNION
    { BIND(<http://dbpedia.org/sparql> AS ?ser235671) }
   SERVICE ?ser235671
    { ?president dbpedia-owl:party ?party }
 }
```

BS12:

```
SELECT  ?president ?party ?page
WHERE
 {   { BIND(<http://api.talis.com/stores/jgoodwin-genealogy/services/sparql> AS ?ser439692) }
   UNION
    { BIND(<http://api.talis.com/stores/pbac/services/sparql> AS ?ser439692) }
   UNION
    { BIND(<http://dbpedia.org/sparql> AS ?ser439692) }
   SERVICE ?ser439692
    { ?president rdf:type dbpedia-owl:President.
      ?president dbpedia-owl:nationality dbpedia:United_States.
      ?president dbpedia-owl:party ?party
    }
    SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
     { ?x owl:sameAs ?president .
       ?x nytimes:topicPage ?page
     }
 }
```

BS13:

```
SELECT  ?actor ?news
WHERE
{ SERVICE <http://data.linkedmdb.org/sparql>
   { ?film dc:title "Tarzan" }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
   { ?y owl:sameAs ?x .
     ?y nytimes:topicPage ?news
   }
   SERVICE <http://data.linkedmdb.org/sparql>
   { ?film linkedMDB:actor ?actor .
     ?actor owl:sameAs ?x
   }
}
```

BS14:

```
SELECT  ?actor ?news
WHERE
 { SERVICE <http://data.linkedmdb.org/sparql>
    { ?film dc:title "Tarzan" .
      ?film linkedMDB:actor ?actor .
      ?actor owl:sameAs ?x
    }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
    { ?y owl:sameAs ?x .
      ?y nytimes:topicPage ?news
    }
 }
```

BS15:

```
SELECT  ?actor ?news
WHERE
 { SERVICE <http://data.linkedmdb.org/sparql>
   { ?film dc:title "Tarzan" }
   SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
   { ?y owl:sameAs ?x .
     ?y nytimes:topicPage ?news
   }
   SERVICE <http://data.linkedmdb.org/sparql>
   { ?film linkedMDB:actor ?actor .
     ?actor owl:sameAs ?x
   }
 }
```

BS16:

```
SELECT  ?actor ?news
WHERE
  { SERVICE <http://data.linkedmdb.org/sparql>
     { ?film dc:title "Tarzan" .
       ?film linkedMDB:actor ?actor .
       ?actor owl:sameAs ?x
     }
    SERVICE <http://api.talis.com/stores/nytimes/services/sparql>
     { ?y owl:sameAs ?x .
       ?y nytimes:topicPage ?news
     }
  }
```

BS17:

```
SELECT  ?film ?director ?genre
WHERE
 { SERVICE <http://dbpedia.org/sparql>
   { ?director dbpedia-owl:nationality dbpedia:Italy> }
   SERVICE <http://data.linkedmdb.org/sparql>
   { ?x owl:sameAs ?film .
     ?x linkedMDB:genre ?genre
   }
   SERVICE <http://dbpedia.org/sparql>
   { ?film dbpedia-owl:director ?director }
}
```

BS18:

```
SELECT  ?film ?director ?genre
WHERE
  { SERVICE <http://dbpedia.org/sparql>
     { ?film dbpedia-owl:director ?director.
       ?director dbpedia-owl:nationality dbpedia:Italy
     }
    SERVICE <http://data.linkedmdb.org/sparql>
     { ?x owl:sameAs ?film .
       ?x linkedMDB:genre ?genre
     }
  }
```

BS19:

```
SELECT  ?film ?director ?genre
WHERE
 { SERVICE <http://dbpedia.org/sparql>
   { ?director dbpedia-owl:nationality dbpedia:Italy> }
   SERVICE <http://data.linkedmdb.org/sparql>
   { ?x owl:sameAs ?film .
     ?x linkedMDB:genre ?genre
   }
   SERVICE <http://dbpedia.org/sparql>
   { ?film dbpedia-owl:director ?director }
}
```

BS20:

```
SELECT  ?film ?director ?genre
WHERE
 { SERVICE <http://dbpedia.org/sparql>
    { ?film dbpedia-owl:director ?director.
      ?director dbpedia-owl:nationality dbpedia:Italy
    }
   SERVICE <http://data.linkedmdb.org/sparql>
    { ?x owl:sameAs ?film .
      ?x linkedMDB:genre ?genre
    }
 }
```

Ek3 Önek Tablosu

rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
dbpedia:	<http://dbpedia.org/resource/>
dbpedia-owl:	<http://dbpedia.org/ontology/>
dbpprop:	<http://dbpedia.org/property/>
linkedMDB:	<http://data.linkedmdb.org/resource/movie/>
owl:	<http://www.w3.org/2002/07/owl#>
dc:	<http://purl.org/dc/terms/>
foaf:	<http://xmlns.com/foaf/0.1/>
facebook:	<http://155.223.25.235:8180/FLE/ontology/socsem.owl#>
film:	<http://data.linkedmdb.org/resource/film/>

Ek4 Tüm VOID Belgeleri Uçnoktaları

http://linkedscotland.org/sparql

http://rdf.myexperiment.org/sparql/

http://sparql.linkedopendata.it/los

http://sparql.linkedopendata.it/grrt

http://api.talis.com/stores/theviewfrom/services/sparql

http://www4.wiwiss.fu-berlin.de/stitch/sparql

http://data.archiveshub.ac.uk/sparql

http://api.talis.com/stores/pokedex/services/sparql

http://greek-lod.math.auth.gr/police/sparql

http://api.talis.com/stores/smcjournals/services/sparql

http://api.talis.com/stores/productdb/services/sparql

http://sparql.data.southampton.ac.uk/

http://www4.wiwiss.fu-berlin.de/eurostat/sparql

http://cultura.linkeddata.es/sparql

http://api.talis.com/stores/nvd/services/sparql

http://data.fundacionctic.org/sparql

http://crime.rkbexplorer.com/sparql/

http://services.data.gov.uk/patents/sparql

http://sparql.linkedopendata.it/musei

http://api.talis.com/stores/trafficscotland/services/sparql

http://doc.metalex.eu:8000/sparql/

http://sparql.linkedopendata.it/cap

http://id.sgcb.mcu.es/sparql

http://services.data.gov.uk/statistics/sparql

http://api.talis.com/stores/jgoodwin-genealogy/services/sparql

http://kwijibo.talis.com/kasabi/eumida/sparql

http://www.morelab.deusto.es/joseki/articles

http://api.talis.com/stores/pbac/services/sparql

http://sparql.linkedopendata.it/grrp

http://www4.wiwiss.fu-berlin.de/eures/sparql

http://api.talis.com/stores/massobservation/services/sparql

http://services.data.gov.uk/business/sparql

http://dewey.info/sparql.php

http://api.talis.com/stores/mesh-norwegian/services/sparql

http://api.talis.com/stores/bbc-wildlife/services/sparql

http://api.talis.com/stores/climb/services/sparql

http://api.talis.com/stores/wordnet/services/sparql

http://services.data.gov.uk/reference/sparql

http://api.talis.com/stores/moseley/services/sparql

http://sparql.linkedopendata.it/scuole

http://www4.wiwiss.fu-berlin.de/diseasome/sparql

http://api.talis.com/stores/schemapedia/services/sparql

http://www4.wiwiss.fu-berlin.de/medicare/sparql

http://services.data.gov.uk/research/sparql

http://greek-lod.math.auth.gr/intervalue/sparql

http://affymetrix.bio2rdf.org/sparql

http://id.ndl.go.jp/auth/ndla/

http://greek-lod.math.auth.gr/fire-brigade/sparql

http://sparql.linkedopendata.it/istat

http://genbank.bio2rdf.org/sparql

http://omim.bio2rdf.org/sparql

http://sgd.bio2rdf.org/sparql

http://el-devtc01.tugraz.at/~ldv/interlinking/endpoint_handler.php

http://hgnc.bio2rdf.org/sparql

http://en.openei.org/sparql

http://el.dbpedia.org/sparql

http://unists.bio2rdf.org/sparql

http://data.linkedct.org/sparql

http://mgi.bio2rdf.org/sparql

http://www.lotico.com:2020/lotico

http://geo.linkeddata.es/sparql

http://homologene.bio2rdf.org/sparql

http://kent.zpr.fer.hr:8080/educationalProgram/sparql

http://linkedgeodata.org/sparql

http://dbpedia.org/sparql

http://data.linkedmdb.org/sparql

http://api.talis.com/stores/nytimes/services/sparql