BAĞLI VERİ ÜZERİNDE DAĞITIK SORGULAMA OPTİMİZASYONU

ETHEM CEM ÖZKAN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Nisan 2015

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Osman EROĞUL

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Ethem Cem Özkan tarafından hazırlanan Bağlı Veri Üzerinde Dağıtık Sorgulama Optimizasyonu adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. A. Murat ÖZBAYOĞLU _____

Üye : Doç. Dr. Erdoğan DOĞDU _____

Üye : Burak İbrahim SEVİNDİ _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Ethem Cem ÖZKAN

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Doç. Dr. Erdoğan Doğdu**
Tez Türü ve Tarihi : **Yüksek Lisans – Nisan 2015**

Ethem Cem ÖZKAN

BAĞLI VERİ ÜZERİNDE DAĞITIK SORGULAMA OPTİMİZASYONU

ÖZET

SPARQL anlamsal ağın (semantik web) standart sorgulama dilidir ve büyük anlamsal ağ veri kaynakları olan "bağlı veri" kaynaklarını sorgulamada kullanılmaktadır. SPARQL dağıtık sorgular yazılarak, dağıtık bağlı veri kaynaklarını sorgulamak içinde kullanılır. Bu işlemde sorgu veya alt sorguları farklı veri kaynaklarında çalıştırılır ve sonuçlar sorgunun sonucu olarak birleştirilir.

Bu tezde, "biricik yüklem veri kaynağı eleme" (unique predicate source pruning) (UPSP) adlı dağıtık SPARQL sorgusunda veri kaynağı seçen bir algoritma önerisi öneriyoruz. Algoritmanın amacı dağıtık SPARQL sorgusu çalıştırılmadan önce ilgili bağlı veri kaynaklarını bulmaktır. Bu sayede sorgu tüm veri kaynaklarına gönderilmek yerine, sorgu ile alakalı veri bulunduran dolayısı ile sorguya katkı sağlayabilecek veri kaynaklarına gönderilebilecektir. Önerdiğimiz algoritma, öncelikle sorgudaki yıldız, yol, alıcı ve hibrit adı verilen alt sorgu tiplerini eşleştirmektedir. Daha sonra sorgudaki tüm düğümler için özne-özne, özne-nesne, nesne-özne, nesne-nesne adı verilen uygun biricik yüklem tiplerini kontrol etmektedir. Eğer algoritma uygun biricik yüklem tipi ve alt sorgu tiplerini bulursa harici veri kaynaklarını elemektedir.

UPSP algoritması, önceden çevrim dışı oluşturulmuş dizin yapısı kullanmaktadır. Bu dizin yapısı bu alanda daha önce yapılmış olan Hibiscus çalışması ile uyumlu olacak şekilde tasarlanmıştır. Hibiscus dizin yapısına her biricik yüklem tipi için bir tane olmak üzere dört adet isteğe bağlı alan eklenmiştir.

UPSP algoritması, açık kaynak dağıtık sorgulama motoru olan Hibiscus üzerine gerçekleştirilmiştir. Algoritma, Hibiscus veri kaynağı eleme algoritmasından hemen önce çalışmaktadır.

Algoritmanın performansı, FedBench test aracı kullanılarak orijinal Hibiscus veri kaynağı eleme yöntemi ile karşılaştırıldı. Sonuçlar algoritmanın veri kaynağı seçimini bazı durumlarda %20'ye kadar iyileştirdiğini göstermektedir.

Anahtar Kelimeler: bağlı veri, dağıtık sorgulama, sorgu optimizasyonu, biricik yüklem

University : **TOBB University of Economics and Technology**

Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Associate Professor Dr. Erdoğan Doğdu**
Degree Awarded and Date : **M.Sc. – April 2015**

Ethem Cem ÖZKAN

FEDERATED QUERY OPTIMIZATION ON LINKED DATA

ABSTRACT

SPARQL is the standard query language of the semantic Web and it is used to query linked data sources which are big semantic Web data sources. SPARQL can also be used to query "distributed" linked data sources by writing federated SPARQL queries in which case query or its sub queries are executed in separate sites and the results are combined and returned as the result of the query.

In this thesis, we propose a new algorithm called "unique predicate source pruning" (UPSP) that reduces the federated SPARQL query execution time. The idea behind the algorithm is to find all relevant distributed linked data sources before executing federated SPARQL queries. This way the query is not sent to all data sources but only to the linked data sources that have data relevant to the query and therefore might return results. UPSP algorithm checks the sub query patterns in the query being processed first, looks for "star", "path", ”hybrid”, “sink” patterns. For each node UPSS algorithm checks appropriate unique predicate types which are subject-subject, subject-object, object-subject and object-object. If UPSP algorithm finds appropriate unique predicate type for query pattern it prunes all external sources.

UPSP algorithm uses an index structure that is built offline before the algorithm executes. UPSP algorithm index structure is designed to be compatible with Hibiscus index that was proposed in the literature before. UPSP algorithm index has four more optional fields which are for each unique predicate types.

We implemented UPSP algorithm on Hibiscus federated query engine which is an open source federated SPARQL query engine. UPSS algorithm executes just before Hibiscus pruning algorithm.

We evaluated UPSP using FedBench benchmark. We compared the performance of the algorithm against standard Hibiscus source selection. The results show that algorithm improves source pruning up to 20% in some cases.

Keywords: linked data, federated querying, query optimization, unique predicate

TEŞEKKÜR

Tez çalışmam süresince yardımlarını, bilge ve tecrübelerini esirgemeyen danışmanım Doç. Dr. Erdoğan Doğdu'ya, teze katkı veren Leipzig üniversitesinden doktora öğrencisi Muhammad Saleem ve danışmanı Axel-Cyrille Ngonga Ngomo'ye,

Yardımlarını esirgemeyen, tecrübeleri ile bana büyük yardımları dokunan başta Burak İbrahim Sevindi ve Seyfullah Demir olmak üzere tüm çalışma arkadaşlarıma,

Yüksek lisans çalışmalarımı destekleyen kurumum TÜBİTAK – BİLGEM –YTE'ye

Hep yanımda olan, yaptıkları fedakârlıklar ile başarılarıma en büyük katkıyı sağlayan başta annem Ayşegül Özkan, babam Celal Asım Özkan, teyzem Nigar Akpınar ve halam Nida Besbelli olmak üzere tüm aileme teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

KISALTMALAR

KISALTMA	AÇIKLAMA
SPARQL	SPARQL Protocol AND RDF Query Language
URI	Uniform Resource Identifier
IRI	Internationalized Resource Identifier
W3C	World Wide Web Consortium
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
XML	eXtensible Markup Language
OWL	Web Ontology Language
RDF	Resource Description Framework
JAAS	Join Aware Source Selection
UPSP	Unique Predicate Source Pruning
BGP	Basic Graph Pattern

1. GİRİŞ

Günümüzde "semantik Web" (Shadbolt, Hall, & Berners-Lee, 2006) ve "bağlı veri" (linked data) (Bizer, Heath, Idehen, & Berners-Lee, 2008) kavramları ile birlikte veriyi etkili bir şekilde sorgulamanın yöntemleri araştırılmaya başlanmıştır. Bağlı veri yapısı farklı kategorilerde birbirleri arasında ilişkiler bulunan devasa bir veri kümesidir ve bu veriler dağıtık olarak farklı sunucularda bulunabilmektedir. Bu büyük dağıtık verinin sorgulanabilmesi için dağıtık sorgulama motorları geliştirilmiştir. Anlamsal ağ sorgulama dili olan SPARQL 1.1 (Seaborne, Polleres, Feigenbaum, & Williams, 2013) dağıtık sorgulama ara yüzünün duyurulması ile birlikte farklı SPARQL 1.1 üzerine dağıtık sorgulama motorları çıkmıştır. SPARQL-DQP (Priyatna, Aranda, & Corcho, 2013) yaygın olarak bilinen SPARQL 1.1 üzerine kurulmuş dağıtık sorgulama motorudur. Ancak SPARQL 1.1 üzerine kurulan dağıtık sorgulama araçlarının kullanılabilir olduğu düşünülmediğinden (Rakhmawati, Umbrich, Karnstedt, Hasnain, & Hausenblas, 2013) SPARQL 1.0 üzerine dağıtık sorgulama üzerine çalışma yapıldı. SPARQL 1.1 ile gelen dağıtık sorgulama yapısında sorgu içerisine veri kaynağı adresi verilmesi gerekmektedir. Bu yöntem sorgunun hızlı ve doğru çalıştığını garanti etmektedir. Ancak dağıtık sorgu çalıştırılırken, veri kaynağı bilgisi yazmak bağlı veri boyutu sebebi ile veri kaynağı bilinemeyeceğinden dolayı kullanışlı değildir. Bu nedenle bu tez kapsamında SPARQL 1.0 üzerine geliştirilen dağıtık sorgulama motorları üzerine çalışılmıştır.

Dağıtık sorgulama motorunun verilen sorgu için en uygun veri kaynaklarını hızlı bir şekilde bulması gerekmektedir. Veri kaynağı seçme işlemi için farklı yollar kullanılmaktadır. Bu yöntemlerin hepsi verilen sorguyu alt parçalara bölerek bu parçaları ayrı ayrı işlemektedir. Yöntemlerden en yaygın ve en basit olanı; alt sorguları tüm veri kaynaklarına ayrı ayrı göndererek dönen sonuçları birleştirmektir. Bu anlaşılabileceği gibi doğru sonuç oluşturan ancak etkili olmayan bir yöntemdir. SPARQL sorgusu yapısında alt sorgular arasında ilişki olabilir. Her alt sorgu ayrı ayrı işlendiğinde bu ilişkiler kullanılamamaktadır. Bu ilişkiler kullanılarak alt

sorgular gruplanarak seçilen veri kaynaklarına gönderilmesi sağlanabilmektedir. Bu yöntem ile sorgu çalışma zamanı iyileştirilebilmektedir. Bu yönteme Birleşim-Farkında Veri Kaynağı Seçme (Join Aware Source Selection) (JASS) ismi verilmiştir. Bu yöntem ile alt sorgular arasındaki ilişkileri kullanarak hızlı ve doğru bir şekilde veri kaynakları seçilmektedir. SPENDID (Staab, 2011) ve Hibiscus (Saleem & Ngonga Ngomo, 2014) JAAS yöntemi ile geliştirilmiş yaygın kullanılan dağıtık sorgulama motorlarıdır.

Bu tez kapsamında Hibiscus (Saleem & Ngonga Ngomo, 2014) ve SPARQL 1.0 üzerine kurulmuş bir JASS algoritması olan biricik yüklem veri kaynağı eleme (UPSP) algoritması anlatılacaktır.

Hibiscus veri kaynağı eleme algoritmasını iyileştirmek amacıyla geliştirilen biricik yüklem eleme "unique predicate source pruning" (UPSP) algoritması, yüklem özellikleri ile alt sorgular arasındaki ilişkileri kullanarak veri kaynağı elemektedir.

Bu tezin katkıları şöyle özetlenebilir:

1) Biricik Yüklem Eleme (Unique Predicate Source Pruning) (UPSP) algoritması önerilmiştir,
2) Var olan Hibiscus dizin yapısı UPSP algoritması ile uyumlu olacak şekilde yapısı geliştirilmiştir
3) UPSP algoritmasının orijinal Hibiscus'a göre daha iyi sonuçlar verdiği veri kaynağı eleme testlerinde gösterilmiştir.

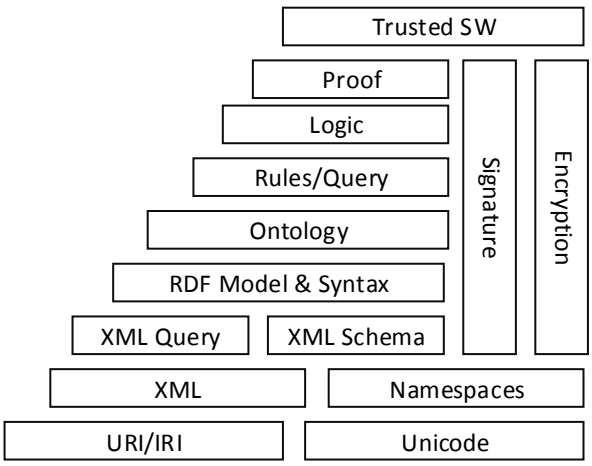
Tezin izleyen bölümleri şunları içermektedir: Bölüm 2'de bu tezin ilgili olduğu konular ve dağıtık sorgulama alanındaki daha önce yapılan çalışmalar incelenmiştir. Bölüm 3'de tez kapsamında önerilen Biricik Yüklem tabanlı dağıtık sorgulama iyileştirme yöntemi açıklanmıştır. Bölüm 4'de tez kapsamında önerilen iyileştirmenin testleri için kurulan ortam anlatılmış ve test için kullanılan veri kaynağı ve sorguları özetlenmiştir. Bölüm 5'de deney sonuçları detaylı olarak sunulmuş ve değerlendirilmiştir. Bölüm 6'da sonuçlar özetlenmiş ve gelecek çalışmalar için önerilerde bulunulmuştur.

2. ÖNCEKİ ÇALIŞMALAR VE MOTİVASYON

Bu bölümde tezle ilgili genel konular (semantik Web, bağlı veri, SPARQL sorgu dili) ile tezde çözüm getirilmeye çalışılan dağıtık sorgulama konusu, bu konuda daha önce yapılan çalışmalar, FedX, Splendid ve Hibiscus sistemleri konusunda bilgi verilecektir. Ayrıca yapılan çalışmanın gerekçesi açıklanacaktır.

2.1 Anlamsal Ağ (Semantik Web)

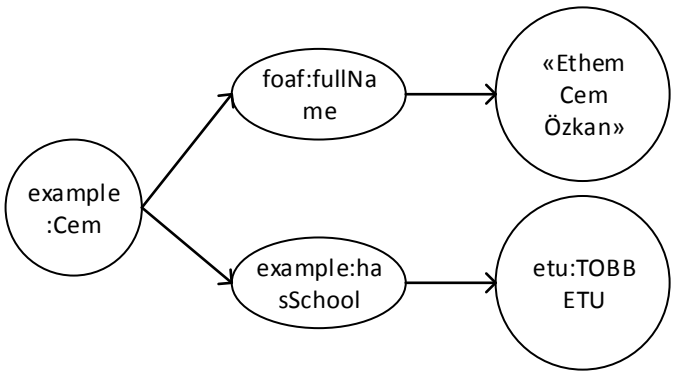
Web'in gelişimi 3 aşamadan oluşmaktadır. Bunlar Web 1.0 denilen statik web sayfalarının ağırlıkta olduğu internetin ilk çağları, Web 2.0 denilen dinamik sayfaların yaygınlaşmaya başladığı insanların forumlar, bloglar, yorumlar ile internete katkı yapmaya başladığı dönemler ve Web 3.0 ile anlamsal ağ çağıdır. Özellikle Web 2.0 sonrasında Web'deki veri miktarı gün geçtikçe katlanarak artmaktadır. Google gibi arama motorları yardımı ile insan kolaylıkla aradığı bilgiyi Web üzerinde bulabilir duruma gelmiştir. Ancak bu devasa veri temelde insanların anlamlandırabildiği, yorumlayabildiği bir formatta bulunmaktadır. Yani Web yazılımların yorumlayabileceği şekilde tasarlanmamıştır. Bu düşünce ile yola çıkarak Tim Berners-Lee ve James Hendler 2001 yılında "anlamsal ağ" (Berners-lee & Hendler, 2001) kavramını ortaya çıkarmıştır. Anlamsal ağ ile internet de bulunan bu verinin yazılımlar tarafından kullanılabilmesi ve yorumlanabilmesi amaçlanmaktadır. Yazılımların İnternet'te bulunan bu veriyi anlamlandırabilmesi için RDF (Resource Description Framework) ve diğer anlamsal ağ standartları W3C (World Wide Web Consortium) kurumu tarafından oluşturulmuştur.



Şekil 2.1 Anlamsal Ağ Katmanları

Şekil 2.1'de görülebildiği gibi anlamsal ağ standartları, katmanlı bir yapıda tasarlanmıştır. Anlamsal ağın temelini oluşturan URI/IRI ile tüm nesneler adreslendirilmektedir. RDF, XML üzerine oluşturulmuş bir standarttır. Nesneler ve bunlar arasındaki ilişkiyi belirler. RDFS ise veri modellemek için kullanılmaktadır. Bir düğümün türü RDFS modelleri ile belirtilmektedir. Ontoloji (Ontology) yani OWL katmanı ise RDF üzerindeki sınıfları ve yüklemleri tanımlamak için geliştirilmiştir. RDF üzerinde bulunan veriyi OWL katmanında tanımlanmış özelliklerini kullanarak sorgulamak için ise sorgu katmanı (Rules/Query) yani SPARQL dili geliştirilmiştir. Bunlar üzerine bilgisayarların anlamlandırma, yorumlama yapabilmesi için Mantık (Logic) katmanı bulunmaktadır. Bu katmanların hepsini dikey olarak kesen İmzalama (Signature) ve Şifreleme (Encription) katmanları ile güvenlik unsurları eklenmektedir.

Anlamsal ağda veriler ile üçlü (triple) denen bir veri yapısında gösterilirler. Bu veri yapısı özne-yüklem-nesne (subject-predicate-object) şeklinde gösterilmektedir. Bu veri formatında her düğüm (node) özne, yüklem veya nesne olarak görev alabilir.



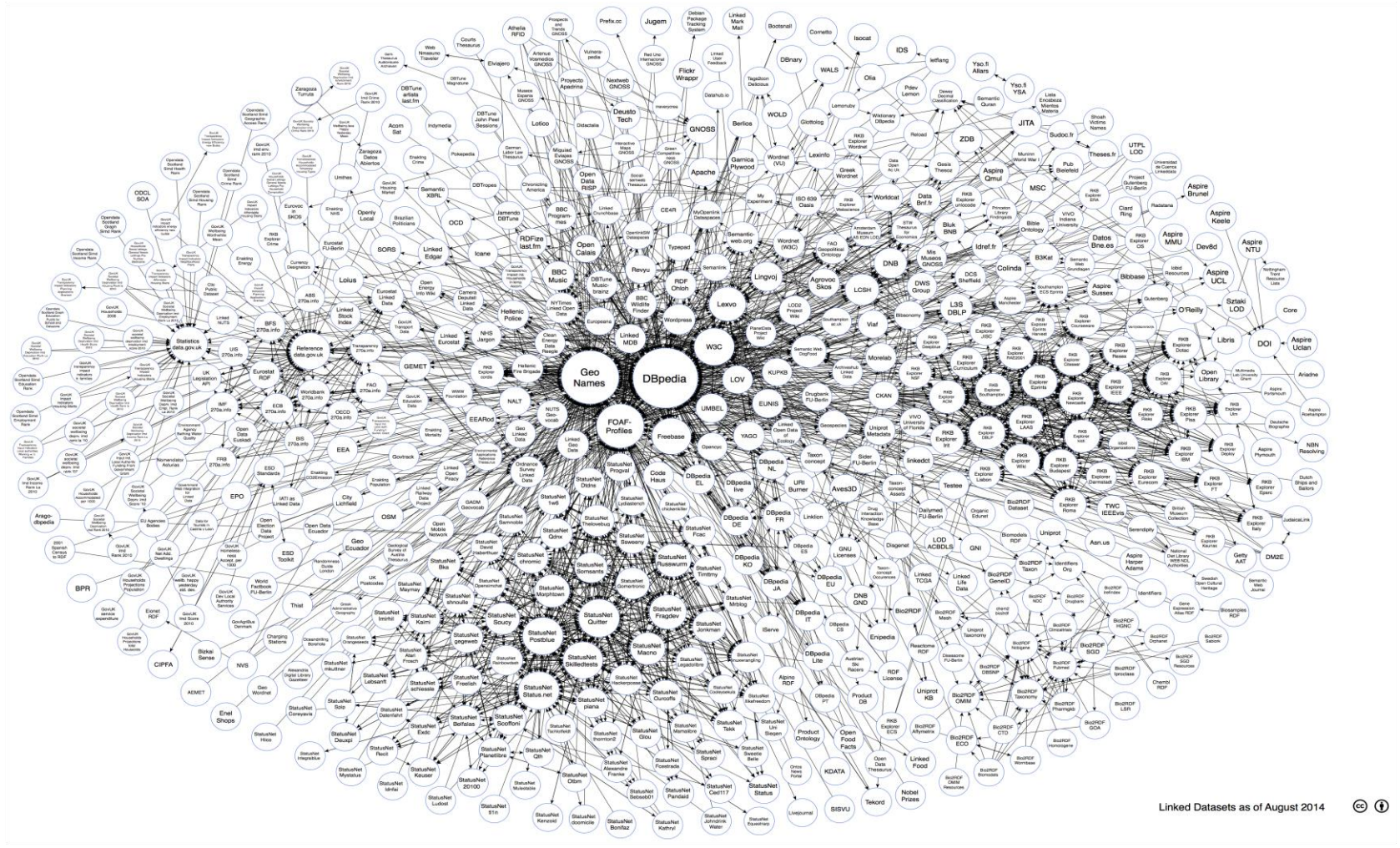
Şekil 2.2 Özne-Yüklem-Nesne Örneği

Şekil 2.2'de anlamsal ağda kullanılan özne-yüklem-nesne yapısının örneği görülmektedir. Şekilde example:Cem-foaf:fullName-"Ethem Cem Özkan" ve example:Cem-example:hasSchool-etu:TOBB ETU şeklinde iki adet üçlü bulunmaktadır. Yukarıdaki örnekte example:Cem özne, foaf:hasSchool yüklem ve etu:TOBB ETU nesne olarak görev almıştır. Şekilden görülebildiği gibi üçlüler arası bağlantılar eklenerek çizge şeklinde bir yapı oluşturulmaktadır. Bu şekilde düğümler arası ilişkiler bu yapıda da görülmektedir. Ayrıca düğümlerin başında bulunan foaf, example, etu gibi ifadeler düğümün alan adı (namespace) bilgisini ifade etmektedir. Alan adı bilgisi düğümün hangi veri kaynağına veya veri modeline ait olduğunu bulmakta kullanılabilir. Örneğin http://xmlns.com/foaf/spec/# URL'i kısaltılmış olarak foaf şeklinde kullanılmıştır.

2.2 Bağlı Veri (Linked Data)

Bağlı veri Bölüm 2.1'de anlatılan anlamsal ağ kavramı ile popüler olmuştur. Bağlı veri internet üzerine dağınık olan RDF verisinin birbirleri ile ilişkiye sahip olması ile oluşan bir veri kümesidir. Farklı noktalarda bulunan verilerin RDF bağları ile aralarında bağlantı kurulması sonucu oluşmaktadır. Örneğin DBpedia üzerindeki ilaç verisinde, etkileşime girdiği ilaç bilgileri bulunmamaktadır. Bu bilgi Drugbank veri kaynağında bulunmaktadır. Bu veriyi çekebilmek için DBpedia ve Drugbank arasında bulunan RDF bağları kullanılarak sorgu yapılması gerekmektedir.

Bu tez hazırlanırken bağlı veri bulutunun 30/8/2014 versiyonu bulunmaktadır ve birbirine bağlı 570 adet veri kaynağı içermektedir.



Şekil 2.3 Bağlı Veri Bulutu

Birbirlerine bağlı değişik veri kaynaklarının etkili bir şekilde sorgulanabilmesi için SPARQL sorgu dili ve bu dil üzerine yazılan dağıtık sorgulama motorları geliştirilmiştir.

2.3 SPARQL

SPARQL özne-yüklem-nesne şeklinde gösterilen veri yapısına sahip veri kaynaklarını sorgulayabilmek için geliştirilmiş bir dildir. W3C tarafından önerilen bir standarttır. Dilin sözdizimi veritabanı sorgulamada kullanılan SQL sorgulama diline benzemektedir. Ancak daha farklı bir veri modeli üzerinde çalıştığı için SQL ile arasında farklar bulunmaktadır. Özne-yüklem-nesne üçlüleri kullanılarak sorgulama işlemini yapabilmektedir. Sorgu içerisindeki her üçlü örüntüsüne BGP (Basic Graph Pattern) ismi verilmiştir. İlk çıkan versiyonu olan SPARQL 1.0 verinin tek veri kaynağı üzerinden sorgulanmasını desteklemektedir. Daha sonraki yıllarda çıkan SPARQL 1.1 ile veri üzerinde güncelleme(update) yapılması ve dağıtık sorgulama destekleri eklenmiştir. Jena (McBride, 2001) ve Sesame (Broekstra, Kampman, & Harmelen, 2002) yaygın olarak kullanılan SPARQL gerçekleştirimleridir. İki gerçekleştirim de Java kütüphanesi sunmaktadır. Bölüm 2.3.1 ve 2.3.2'de sırası ile SPARQL 1.0 ve 1.1 dilleri örnekler üzerinden anlatılacaktır.

2.3.1 SPARQL 1.0

2008 yılında çıkan ilk SPARQL standardıdır. Sadece verinin sorgulanması üzerine odaklanmıştır. "INSERT", "UPDATE" ve "DELETE" gibi veri manipülasyonu desteği bulunmamaktadır. Dağıtık sorgulama desteği ise, harici dağıtık sorgulama motorları ile sağlanmıştır. 4 çeşit sorgu tipi desteklemektedir. Bunlar aşağıda açıklamaları ile birlikte anlatılmıştır.

- **SELECT:** SPARQL veri kaynağından veri çekmek için kullanılan sorgu tipidir.
- **ASK:** Sorgu içerisinde verilen özne-yüklem-nesne üçlüsü veri kaynağında bulunuyor ise doğru, bulunmuyor ise yanlış olarak sonuç döner. Dağıtık sorgulama motorları için kullanışlı bir sorgu tipidir.

- **DESCRIBE:** Veri kaynağı istatistiklerini sorgular. Sorgu sonucunun herhangi bir standardı bulunmamaktadır.
- **CONSTRUCT:** Sorgu sonucu üzerinde veri kaynağında bulunmayan özne-yüklem-nesne üçlüleri tanımlamaya imkân veren sorgu çeşididir.

```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
    ?person foaf:name ?name .
    ?person foaf:mbox ?email .
}
```

Şekil 2.4 SPARQL 1.0 Örnek Select Sorgu

Şekil 2.5'de görülebilen örnek SPARQL 1.0 sorgudur. Sorgu üç bölümden oluşmaktadır. Bunlar alan adı kısaltmalarının belirtildiği PREFIX, sorgu sonucunda bulunmasını istediğimiz düğümleri yazdığımız SELECT ve sorgulamak istediğimiz özne-yüklem-nesne örüntülerini yazdığımız WHERE kısmıdır. Yukarıdaki sorgu veri kaynağında bulunan tüm kişilerin ad ve eposta bilgilerini dönmektedir.

```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
ASK {
    ?person foaf:name ?name .
}
```

Şekil 2.5 SPARQL 1.0 Örnek Ask Sorgu

Şekil 2.5'de bulunan örnek ASK sorgusu, veri kaynağında foaf:name yüklem olarak kullanılıyorsa doğru cevap dönecektir. ASK sorgu tipi iki bölümden oluşmaktadır. Bunlar alan adı kısaltmalarının belirtildiği PREFIX ve sorgulamak istenilen özne-yüklem-nesne örüntülerinin belirtildiği ASK kısmıdır.

SPARQL 1.0 üzerine dağıtık sorgulama yapılabilmektedir. Dağıtık sorgulama yapılabilmesi için, sorgunun dağıtık sorgulama motoru üzerinde çalıştırılması gerekmektedir. Dağıtık sorgulama motoru, veri kaynaklarının seçimi ve dönen sonuçların birleştirilmesi işlemlerini yapmaktadır.

2.3.2 SPARQL 1.1

2013 yılında W3C kurumu SPARQL'in bu tez yazıldığı tarihte en güncel hali olan SPARQL 1.1 versiyonunu duyurmuştur. Bu sürüm ile birlikte SPARQL diline 1.0 standardına ek olarak veri manipülasyonu ve SERVICE sözdizimi ile dağıtık sorgulama desteği gelmiştir.

SERVICE söz dizimi sorgu içerisinde veri kaynağı belirtilerek dağıtık sorgulama yapılmasına olanak vermektedir. *Şekil 2.6*'da SPARQL 1.1 standardında yazılmış dağıtık sorgu örneği bulunmaktadır. Bu örnekten görülebildiği gibi SPARQL 1.0 standardı üzerine SERVICE sözdizimi eklenerek dağıtık sorgulamaya uygun hale getirilmiştir.

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

Şekil 2.6 SPARQL 1.1 Dağıtık Sorgu Örneği

Şekil 2.6'da bulan sorgu DBpedia ve NYtimes veri kaynaklarını beraber sorgulamaktadır. DBpedia üzerinden Barack Obama'nın parti bilgisi ve NYtimes üzerinden haber bilgilerini sorgulayarak bunları tek bir sonuç seti üzerinde görüntülemiştir.

2.4 Dağıtık Sorgulama (Federated Query)

Bu bölümde tez için temel olan dağıtık sorgulama (federated query) yapısı açıklanmıştır. Bölüm 2.2'de anlatılan bağlı verinin etkin bir şekilde sorgulanabilmesi için dağıtık sorgulama motorları geliştirilmiştir. Bu motorlar kullanılarak, tek bir nokta üzerinden bağlı veri bulutu üzerinde bulunan RDF verisi sorgulanabilmektedir.

Dağıtık sorgulama Bölüm 2.3.1 ve 2.3.2'de anlatılan SPARQL standartlarının ikisi de kullanılarak yapılabilmektedir.

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

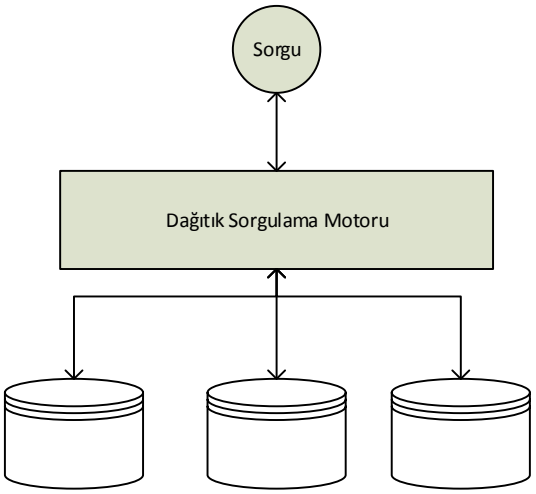
Şekil 2.7 SPARQL 1.0 Dağıtık Sorgulama Örneği

Şekil 2.6'da SPARQL 1.1 standardında yazılmış olan dağıtık sorgu örneğinin SPARQL 1.0 üzerinde yazılmış olan versiyonu *Şekil 2.7*'de görülmektedir. Resimlerden anlaşılabileceği gibi SPARQL 1.1 kullanarak dağıtık sorgulama yapıldığında sorgu içerisine veri kaynağı belirtmek gerekmektedir. Bu durum bağlı veri bulutunun büyüklüğü düşünüldüğünde sorgu yazmayı zorlaştırmaktadır. Sorgu yazılırken SPARQL 1.0 standardı kullanılarak yani veri kaynağı bağımsız yazıldığında tam anlamı ile dağıtık sorgulama yapıldığını düşünülmektedir. Çünkü sorgulanan verinin hangi veri kaynağında bulunduğunu kullanıcının bilmediği durumlar oluşabilmektedir.

Bu başlıkta bu alanda öne çıkan uygulamalar ve bu uygulamaların dağıtık sorgulama alanında yaptıkları iyileştirmeler özetlenmiştir.

2.4.1 Dağıtık Sorgulama Hakkında Temel Bilgiler

Dağıtık sorgulama, birden fazla bağlı veri kaynağını etkili bir şekilde sorgulamak için sunulmuş bir yapıdır. Günümüzde endüstri standardı olan ilişkisel veri tabanları dahil çeşitli veri kaynaklarını birlikte sorgulamak için kullanılabilir. Dağıtık sorgulama ile birden çok veri kaynağında bulunan bilgiler tek bir araç üzerinden sorgulanabilir.



Şekil 2.8 Dağıtık Sorgulama Yapısı

Dağıtık sorgulama motoru gelen sorguyu veri kaynaklarına göre alt sorgulara böler ve ilgili veri kaynaklarına alt sorguyu gönderir. Alt sorgulara bölme işlemi genellikle sorgu içerisindeki her üçlü örüntüsünü ayrı sorgu olarak yazmak ile çözülmektedir. Daha sonra alt sorgular aynı veri kaynaklarına gitmesi durumunda birleştirilebilmektedir. Alt sorgu sonuçları "hash join", "bind join" gibi birleşme algoritmalarını kullanarak birleştirir ve sonucu döner.

Dağıtık sorgulama alanında VoID (Akar, Halaç, Ekinci, & Dikenelli, 2012) dizinleri yaygın olarak kullanılmaktadır. VoID dizinleri veri kaynakları için toplam yüklem sayısı, toplam üçlü sayısı gibi istatistiksel bilgileri de içermektedir.

Dağıtık sorgulama işlemlerinin verimli bir şekilde yapılabilmesi için aşağıdaki maddelere dikkat edilmesi gerekmektedir.

- **Veri kaynağı seçimi:** Alt parçalara bölünen sorguların hızlı ve doğru sonuç verebilmesi ve gereksiz birleşimlerin (join) önüne geçilebilmesi için dağıtık sorgulama motorunun "kaynak seçimi" hızlı ve doğru olmalıdır. Veri kaynağı seçimi için yaygın olarak aşağıdaki metotlar kullanılmaktadır.
    - **ASK Sorgusu:** Özel SPARQL sorgu tipidir. Sorgu içerisinde verilen üçlü, veri kaynağında var ise doğru, yok ise yanlış olarak sonuç dönmektedir. Tek başına kullanıldığı durumlarda çok maliyetli

olabileceği için genellikle ask sorgu sonuçlarını ayrıca tutan bir önbellek yapısı eklenmektedir (Rakhmawati et al., 2013).

- **Data Kataloğu:** Bu yöntemde, çevrimiçi veya çevrimdışı tutulan, veri kaynağı listesindeki tüm kaynaklara alt sorgular gönderilir. En maliyetli yöntemdir. Çünkü sorgu sonucuna hiçbir şekilde katılmaması gereken veri kaynaklarına sorgu gönderilir ve bu sebeple birleşme süresi çok uzamaktadır. Özellikle tüm veri kaynaklarında bulunan owl:sameAs ve rdf:type gibi yüklemleri içeren sorgularda bu yöntem çok yavaş kalmaktadır (Rakhmawati et al., 2013).
- **Dizin:** Genellikle veri kaynakları ve içerdiği yüklemler üzerinden bir dizin oluşturulur. Dizin oluşturulması için ASK sorguları ve bunun yanında yüklemin özelliklerini çeken sorgular kullanılır. Dizin, önceden veya çalışma zamanında gelen sorgulara göre dinamik bir şekilde oluşturulabilir. Bölüm 2.4.4'te anlatılan Hibiscus (Saleem & Ngonga Ngomo, 2014) uygulaması ve bu yüksek lisans tezi kapsamında yapılan çalışma da bu yöntem kullanılmaktadır (Rakhmawati et al., 2013).

- **Birleşme algoritması seçimi:** Alt sorgu sonuçlarının hızlı bir şekilde birleştirilebilmesi (join) için birleşme algoritması seçimi önemlidir. Bu alanda yaygın kullanılan birleşme algoritmaları aşağıda özetlenmiştir.
  - **Nested Loop Join:** Sıra ile çalıştırılan alt sorgularda her sonuç önceki ile birleştirilir (Rakhmawati et al., 2013).
  - **Hash Join:** Bu birleştirme tipinde alt sorgular paralel olarak çalıştırıldıktan sonra oluşan ara sonuçlar yerelde birleştirilmektedir. Ara sonuçlar büyük değil ise iyi performans verdiği gözlemlenmiştir. Ancak alt sorgular üzerinde herhangi bir kısıt olmadığından dolayı veri aktarımının yüksek olduğu bilinmektedir. Bu nedenle ağ üzerinde aktarım maliyeti yüksektir (Rakhmawati et al., 2013).
  - **Bind Join:** Alt sorguları sıra ile çalıştırmaktadır ve dönen sonuçlar bir sonraki sorguya filtre olarak eklenmektedir. Bu sayede yerelde birleştirme ve ağ iletişim maliyetini düşürür. Ancak paralel işlem

yapılması mümkün olmadığı için çalışma süresinin uzun olduğu gözlemlenmiştir (Rakhmawati et al., 2013).

Burada dağıtık sorgulama için temel bilgiler verilmiştir. Bu başlıklarda anlatılan yöntemler, dağıtık sorgulama için oluşturulan çözümlerin temelini oluşturmaktadır.

2.4.2 FedX

FedX (Schwarte, Haase, Hose, Schenkel, & Schmidt, 2011) Sesame üzerine geliştirilmiş SPARQL 1.0 üzerinde çalışan bir dağıtık sorgulama motorudur. VoID (Zhao et al., 2009) istatistik bilgilerini ve ASK sorgularını kullanarak veri kaynağı eleme işlemini gerçekleştirir. ASK sorgularının maliyetini azaltmak için sonuçları gelecekteki kullanımlar için ön bellekte saklanmaktadır. FedX aracında oluşturulan alt sorguların sonuçları elde edildikten sonra maliyet hesaplaması yapılarak, maliyete göre birleşme sırası belirlenmektedir. Ayrıca "Exclusive Group" ismini verdikleri bir yöntem ile aynı veri kaynağında olduğu bilinen alt sorguları birleştirerek, tek sorgu olarak göndermektedir. Bu sayede alt sorguların ağ maliyetinde iyileşme olduğu gözlemlenmiştir. "Bound Join" yöntemi adını verdikleri bir birleşme algoritması yardımı ile alt sorgular tarafından oluşturulan ara sonuçlar etkili bir şekilde birleştirilebilmektedir.

2.4.3 Splendid

Splendid (Staab, 2011) Sesame SPARQL motoru üzerine geliştirilmiş bir dağıtık sorgulama aracıdır. SPARQL 1.0 üzerine dağıtık sorgulama yapılmasını desteklemektedir. Veri kaynağı eleme metodu olarak VoID bilgilerini ve ASK sorgularını kullanmaktadır. Veri kaynağı seçimi ve birleşme alanında aşağıdaki iyileştirmeleri eklemiştir.

- **Veri Kaynağı Seçimi:** Önbellekte bulunmayan üçlüler için ASK sorgusu oluşturarak veri kaynağı seçme işlemi yapmaktadır.
- **Birleşme:** Bölüm 2.4.1'de anlatıldığı gibi üç farklı birleşme yöntemi yaygın olarak kullanılmaktadır. Splendid dağıtık sorgulama motoru küçük veri kaynakları için "Hash Join", büyük veri kaynakları için "Bind Join"

kullanmaktadır. Veri kaynağı büyüklük seçimini ise VoID istatistiklerini kullanarak yapmaktadır.

2.4.4 Hibiscus

Hibiscus (Saleem & Ngonga Ngomo, 2014) dağıtık sorgulama yapılırken veri kaynağı seçimine farklı bir açıdan yaklaşmaktadır. Sorguyu hiper-çizge olarak gösterimini yapar ve çizge özelliklerini ve kendi dizin yapısını kullanarak veri kaynaklarını eler. FedX (Schwarte et al., 2011), Splendid (Staab, 2011) ve DarQ (Quilitz & Leser, 2008) dağıtık sorgulama motorları üzerine eklenerek testleri yapılmıştır.

Hibiscus (Saleem & Ngonga Ngomo, 2014) dağıtık sorgulama yapılırken veri kaynağı seçimine farklı bir açıdan yaklaşmaktadır. Sorguyu hiper-çizge olarak modeller ve çizge özelliklerini ve kendi dizin yapısını kullanarak veri kaynaklarını eler. Veri kaynağı listesini, üzerinde çalıştığı dağıtık sorgulama aracını kullanarak oluşturur. Üzerine eleme yapılmamış veri kaynağı listesi, sorgu içerisinde bulunan özne-yüklem-nesne üçlüleri için ASK sorgusu gönderilerek, gelen sonuçlara göre oluşturulur. Bu liste, üzerinde minimum sayıda eleme işlemi yapılmış bir listedir ve genellikle diğer üçlüler arasındaki ilişkilerden bağımsız ASK sorgusu gönderildiği için sorgu gönderilmesine gerek olmayan veri kaynaklarını da içermektedir. Bu sebeple üzerinde Hibiscus veya bu tez çalışması kapsamında geliştirilen UPSP algoritmalarının çalıştırılması gerekmektedir. Hibiscus, FedX (Schwarte et al., 2011), Splendid (Staab, 2011) ve DarQ (Quilitz & Leser, 2008) dağıtık sorgulama motorları üzerine eklenerek testleri yapılmıştır. Bu tez çalışmasında Hibiscus dizin yapısı üzerine biricik yüklem bilgisi eklenip, bu bilgi kullanılarak veri kaynağı eleme işlemi yapan bir algoritma (UPSP) önerildi. Aşağıda Hibiscus dizin yapısı ve ilgili veri kaynağı eleme algoritmaları anlatılacaktır.

2.4.4.1 Hibiscus Dizin Yapısı

Hibiscus, dizininde bulunan bilgileri kullanarak veri kaynağı eleme işlemini gerçekleştirmektedir. Dizin yapısı ve açıklaması aşağıda verilmiştir.

```
[] a ds:Service ;
    ds:url   <http://jamendo.ecozkan.com/sparql> ;
    ds:capability
        [
            ds:predicate  <http://purl.org/ontology/mo/time> ;
            ds:sbjAuthority <http://dbtune.org> ;
            ds:objAuthority <http://dbtune.org> ;
        ] ;
            .
[] a ds:Service ;
    ds:url   <http://kegg.ecozkan.com/sparql> ;
    ds:capability
        [
     ds:predicate  <http://bio2rdf.org/ns/kegg#xGene> ;
            ds:sbjAuthority <http://bio2rdf.org> ;
            ds:objAuthority <http://bio2rdf.org> ;
        ] ;
```

Şekil 2.9 Hibiscus Örnek Dizin Yapısı

Şekil 2.9'de görülebildiği gibi Hibiscus dizininde ds:Service, ds:url, ds:capability, ds:predicate, ds:sbjAuthority ve ds:objAuthority elemanları bulunmaktadır. Bu elemanların açıklamaları aşağıda listelenmiştir.

- Service: Veri kaynağı dizin yapısında ds:service olarak gösterilmektedir.
- url: Service olarak gösterilen veri kaynağının erişim adresini tutmaktadır. SPARQL sorguları için veri kaynağı seçilirken bu adres kullanılmaktadır.
- capability: Yüklemler dizin yapısında ds:capability olarak gösterilmektedir.
- predicate: Yüklemin URL olarak gösterimini tutmaktadır.
- sbjAuthority: Özne-yüklem-nesne (subject-predicate-object) üçlüsünde yüklemin ilgili veri kaynağında aldığı tüm öznelerin alan adı bilgilerini içerir.
- objAuthority: Özne-yüklem-nesne (subject-predicate-object) üçlüsünde yüklemin veri kaynağında aldığı tüm nesnelerin alan adı bilgilerini içerir.

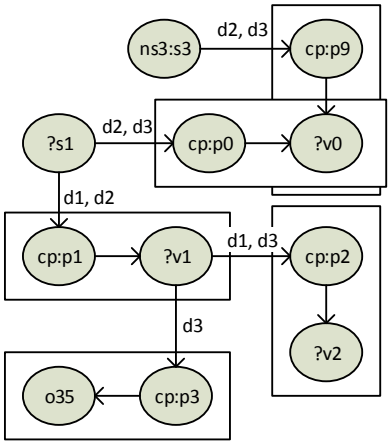
Alan adı URL bilgisinin üçüncü '/' karakterine kadar olan kısmıdır. Örneğin "http://dbpedia.org/ontology/municipality" düğümü için "http://dbpedia.org" alan adı bilgisidir.

Hibiscus bu dizinde ilgili veri kaynağına ait yüklemlerin "sbjAuthority" ve "objAuthority" alanlarında bulunan verileri kullanarak eleme algoritmasını çalıştırır. Eleme algoritmasının detayları Bölüm 2.4.4.4'da detaylı olarak anlatılmaktadır.

2.4.4.2 Hibiscus Sorgu Gösterimi

Hibiscus (Saleem & Ngonga Ngomo, 2014) veri kaynağı eleme işlemini yapabilmek için sorguları hiper-çizge (hypergraph) olarak modellemektedir.

```
PREFIX cp: <http://common/schema/>
PREFIX ns3:<http://auth3/schema/>
SELECT * WHERE
{
        ns3:s3 cp:p9 ?v0 . Kaynak: d2, d3
        ?s1 cp:p0 ?v0 .    Kaynak: d2, d3
        ?s1 cp:p1 ?v1 .    Kaynak: d1, d2
        ?v1 cp:p2 ?v2 .    Kaynak: d1, d3
        ?v1 cp:p3 "o35"    Kaynak: d3
}
```



Şekil 2.10 Hiper-Çizge Gösterimi

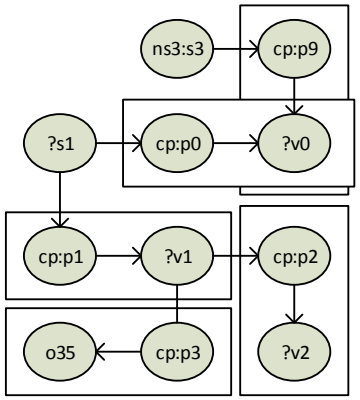
Şekil 2.10'de verilen örnek sorgu ve buna ait hiper-çizge gösterimi verilmiştir. Hiper-çizge gösterimi aşağıda açıklanmıştır.

- **Düğüm:** Hiper-çizge üzerindeki düğümlerdir. Sorgu üzerinde bulunan tüm özne yüklem ve nesne elemanlarını göstermektedir.
- **Bağ:** Düğümler arasında üçlülere göre oluşturulan yönlü bağlantılardır. Her üçlü için özneden yükleme ve yüklemden nesneye olmak üzere iki bağ vardır.
- **Hiper Bağ:** Hiper bağ, sorgunun özne-yüklem-nesne üçlüsünden oluşmaktadır. Özne ve "yüklem-nesne" ikilisi arasındaki bağlantıdır. Üzerinde ilgili özne-yüklem-nesne üçlüsünün sorgulanabileceği veri kaynağı listesini bulundurmaktadır.

2.4.4.3 Sorgu Tipleri

Sorguların başarılı bir şekilde dağıtık çalıştırılabilmesi için öncelikle sorgunun incelenmesi ve alt sorgulara ayrıştırılması gerekmektedir. Bu işlemi Hibiscus (Saleem & Ngonga Ngomo, 2014) sorgu üzerinden hiper-çizge oluşturarak yapmaktadır. Örnek bir sorgu ve bunun hiper-çizge gösterimi *Şekil 2.11* de verilmiştir.

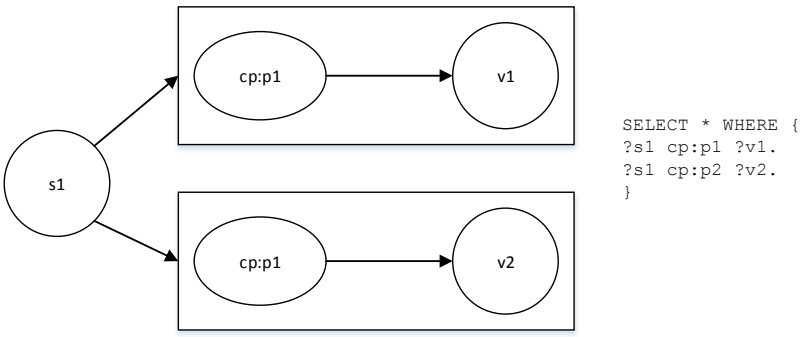
```
PREFIX cp: <http://common/
schema/>
PREFIX ns3:<http://auth3/
schema/>
SELECT * WHERE
{
       ns3:s3 cp:p9 ?v0.
       ?s1 cp:p0 ?v0.
       ?s1 cp:p1 ?v1 .
       ?v1 cp:p2 ?v2 .
       ?v1 cp:p3 "o35"
}
```



Şekil 2.11 Örnek Sorgu ve Hiper-Çizge Gösterimi

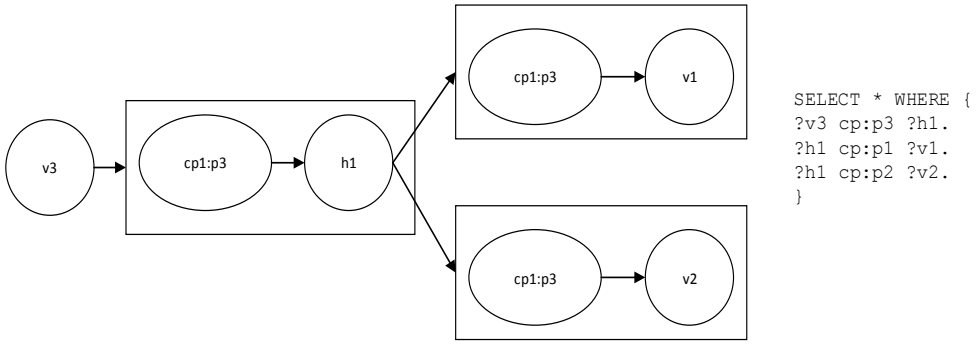
Hiper-çizge gösteriminin detaylı anlatımı Bölüm 2.4.4.2'te anlatılmıştır. Bir sorgunun hiper-çizge gösteriminde 5 farklı düğüm tipi çıkabilir. Bunlar açıklamaları ile beraber aşağıda verilmiştir.

- **Yıldız (Star):** Bir düğüm üzerinde birden fazla çıkan bağ ve hiç gelen bağ yok ise bu düğüm yıldız türündedir. Şekil *2.12*'de s1 düğümü yıldız türündedir.



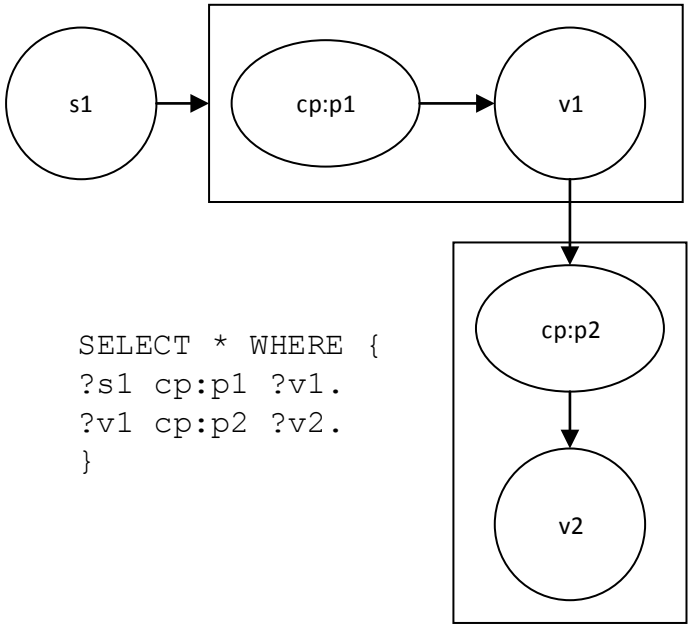
Şekil 2.12 Yıldız Düğüm

- **Hibrit (Hybrid):** Bir düğüm üzerinde en az bir çıkan ve birden fazla gelen bağ veya en az bir gelen ve birden fazla çıkan bağ olduğu durumda düğüm hibrit türündedir. *Şekil 2.13*'de h1 düğümü hibrit türündedir.



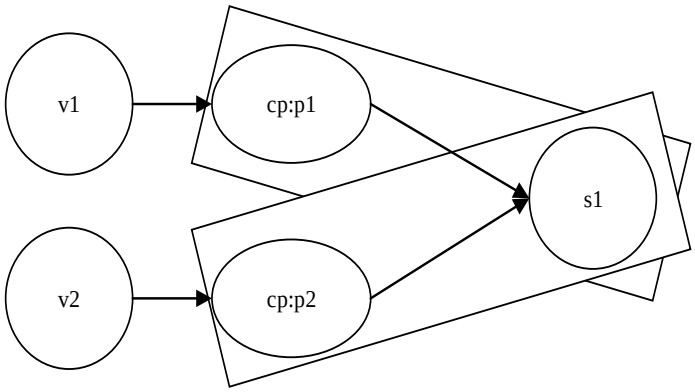
Şekil 2.13 Hibrit Düğüm

- **Yol (Path):** Bir düğüm üzerinde bir adet çıkan ve bir adet gelen bağ var ise yol türündedir. *Şekil 2.14*'de v1 düğümü yol türündedir.



Şekil 2.14 Yol Düğüm

- **Alıcı (Sink):** Bir düğüm üzerinde birden fazla gelen bağ var ise alıcı türündedir. *Şekil 2.15*'de s1 düğümü alıcı türündedir.



Şekil 2.15 Alıcı Düğüm

- **Basit (Simple):** Yıldız, Yol, Hibrit ve Alıcı düğüm tipi tanımlarına uymayan düğümlerdir.

2.4.4.4 Hibiscus Eleme Algoritması

Hibiscus (Saleem & Ngonga Ngomo, 2014) eleme algoritmasının çalışabilmesi için özne ve nesne alan adı bilgilerini içeren dizinin oluşturulmuş olması gerekmektedir. Eleme algoritması *Şekil 2.16*'de görülebilmektedir.

```
Require: DHG //disjunctive hypergraphs
 1: for each HG_i ∈ DHG do
 2:     for each v ∈ vertices(HG_i) do
 3:         if λ_vt(v) ≠ 'simple' then
 4:             SAuth = ∅; OAuth = ∅;
 5:             for each e ∈ E_out(v) do
 6:                 SAuth = SAuth ∪ {subjectauthories(e)}
 7:             end for
 8:             for each e ∈ E_in(v) do
 9:                 OAuth = OAuth ∪ {objectauthories(e)}
10:             end for
11:             A = SAuth ∪ OAuth // set of all authorities
12:             I = A.get(1) //get first element of authorities
13:             for each a ∈ A do
14:                 I = I ∩ a //intersection of all elements of A
15:             end for
16:             for each e ∈ E_in(v) ∪ E_out(v) do
17:                 label = ∅ //variable for final label of e
18:                 for d_i ∈ λ_e(e) do
19:                     if authorities(d_i) ∩ I ≠ ∅ then
20:                         label = label ∪ d_i
21:                     end if
22:                 end for
23:                 λ_e(e) = label
24:             end for
25:         end if
26:     end for
27: end for
```

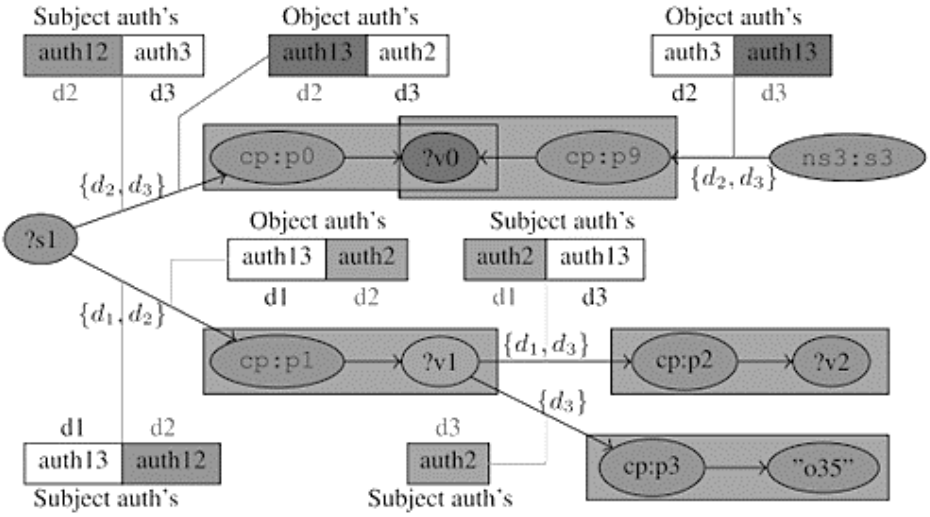
Şekil 2.16 Hibiscus Eleme Algoritması (Saleem & Ngonga Ngomo, 2014)

Algoritma, basit düğüm tipi dışındaki tüm düğümler için çalışmaktadır. Her bir düğüm için çıkan düğümlerin özne alan adları (sbjAuthority) SAuth listesine ve giren düğümlerin nesne alan adları (objAuthority) OAuth listesine eklenmektedir (Satır 5-9). SAuth ve OAuth listelerinde bulunan alan adlarının kesişimleri hesaplanır ve I listesine yazılır (Satır 11-15). İlgili düğümden gelen ve çıkan tüm hiper bağlar ve bunların üzerinde bulunan veri kaynaklarının alan adları ile kesişimleri boş küme değil ise alan adı bilgisi label değişkenine toplanır. Tüm hiper bağlar için bu işlemler yapıldıktan sonra label değişkeni yeni veri kaynağı listesi olarak hiper bağın üzerine yazılır (Satır 16-24) .

Şekil 2.17 Hibiscus Veri Kaynakları ve Örnek Sorgu (Saleem & Ngonga Ngomo, 2014)

Şekil 2.17 de bulunan sorgu ve veri kaynakları için Hibiscus (Saleem & Ngonga Ngomo, 2014) eleme algoritması çalıştırıldığında hibrit tipinde olan v1 düğümü için SAuth değişkeni {auth13, auth2} ve OAuth değişkeni {{auth13, auth2}, {auth2}} olarak yazılmaktadır. Bunların kesişimi ise {auth2} olarak I değişkenine yazılmaktadır. Son olarak v1'e giren bağın veri alanları olan d1 ve d2 arasında sadece d2'nin nesne alan adı {auth2}'yi içerdiği için d2 seçilir. Aynı şekilde çıkan bağın veri alanları olan d1 ve d3 arasında sadece d3 ün özne alan adı {auth2} ile kesişmektedir. Bu nedenle d3 seçilmiştir. *Şekil 2.18* üzerinde tüm düğümler için seçilen tüm alan adları görülebilmektedir.

Bu sayede özne ve nesne alan adı bilgisi kullanılarak eleme algoritması çalıştırılmıştır. Tez çalışması kapsamında bu işlem öncesinde çalışacak olan UPSP algoritması geliştirilmiştir. Bölüm 3.3'de bu yöntem detaylı olarak anlatılmaktadır.



Şekil 2.18 Alan adları ile Hiper-Çizge (Saleem & Ngonga Ngomo, 2014)

3. DAĞITIK SORGULAMA OPTİMİZASYONU

Dağıtık sorgulama alanında, sonucun doğru ve hızlı bir şekilde oluşturulabilmesi için, farklı yöntemler kullanılmaktadır. Bunlardan yaygın olarak kullanılanları Bölüm 0'te anlatılmıştır. Bölüm 2.3.2'de anlatılan SPARQL 1.1 standardı sonrasında dağıtık sorgulama optimizasyonu yaygın olarak araştırılan bir konu haline gelmiştir. Bu alanda SPARQL 1.0 ve SPARQL 1.1 kullanılarak oluşturulmuş çözümler bulunmaktadır. Ancak SPARQL 1.1 standardında sorgu hazırlanırken veri kaynaklarını bilmek gibi bir zorunluluk bulunmaktadır. Çünkü SPARQL 1.1 standardında dağıtık sorgulama yapılabilmesi için, sorgu içine veri kaynağı adresi yazılması gerekmektedir. Bu dağıtık sorgulama yaklaşımının kullanılabilir olduğu düşülmediğinden SPARQL 1.0 üzerine dağıtık sorgulama konusu üzerine çalışılmıştır. SPARQL 1.0 üzerine dağıtık sorgulama ile veri kaynağı bağımsız sorgu yazılabilir. Veri kaynağı seçme işlevini dağıtık sorgulama motoru yapmaktadır. Bu sebeple sorgunun doğru ve hızlı sonuç verebilmesi için veri kaynağı seçimi SPARQL 1.0 üzerine dağıtık sorgulama motorları için çok önemlidir. Bu başlık altında biricik yüklem yaklaşımı ve bunun veri kaynağı seçiminde kullanım alanları anlatılacaktır.

3.1 Biricik Yüklem (Unique Predicate)

Biricik yüklem sorgulanan veri kaynaklarından sadece bir tanesinde bulunabilen ve özne-özne, özne-nesne, nesne-özne ve nesne-nesne biricik özelliklerinden en az birine sahip olan yüklemdir.

Şekil 3.1 Biricik Yüklem Veri Kaynağı Örneği

Şekil 3.1 de tüm biricik yüklem durumlarını içeren örnek veri kaynakları görülebilmektedir. Biricik yüklem özellikleri bu şekilde bulunan veri kaynakları kullanılarak aşağıda anlatılmıştır.

- **Özne-özne:** Biricik yüklemin bulunduğu veri kaynağında geçen özneleri diğer veri kaynaklarında özne olarak kullanılmıyor ise yüklem özne-özne biriciktir. Örneğin *Şekil 3.1* de Veri Kaynağı 1 de bulunan cp:p1 yüklemi özne-özne biriciktir. Çünkü cp:p1 diğer veri kaynaklarında bulunmamakta ve Veri Kaynağı 1 de öznesi olarak geçen ns1:s1 ve ns1_1:s2 diğer veri kaynaklarında özne olarak kullanılmamaktadır.
- **Özne-nesne:** Biricik yüklemin bulunduğu veri kaynağında geçen özneleri diğer veri kaynaklarında nesne olarak kullanılmıyor ise yüklem özne-nesne biriciktir. *Şekil 3.1* de Veri Kaynağı 1 de bulunan cp:p3 yüklemi özne-nesne biriciktir. Çünkü cp:p3 diğer veri kaynaklarında bulunmamakta ve Veri Kaynağı 1 de öznesi olarak geçen ns2:s3 ve ns1_1:s1 diğer veri kaynaklarında nesne olarak kullanılmamaktadır. Ancak cp:p3 özne-özne biricik değildir çünkü öznesi olarak kullanılan ns2:s3 Veri Kaynağı 2 de özne olarak kullanılmaktadır.
- **Nesne-özne:** Biricik yüklemin bulunduğu veri kaynağında geçen nesneleri diğer veri kaynaklarında özne olarak kullanılmıyor ise yüklem nesne-özne

biriciktir. Örneğin *Şekil 3.1*'de Veri Kaynağı 2 de bulunan cp:p5 yüklemi nesne-özne biriciktir çünkü cp:p5 diğer veri kaynaklarında bulunmamakta ve Veri Kaynağı 2 de nesnesi olarak geçen ns3:o2 diğer veri kaynaklarında özne olarak kullanılmamaktadır.

- **Nesne-nesne:** Biricik yüklemin bulunduğu veri kaynağında geçen nesneleri diğer veri kaynaklarında nesne olarak kullanılmıyor ise yüklem nesne-nesne biriciktir. Veri Kaynağı 2 de bulunan cp:p5 yüklemi nesne-nesne biriciktir çünkü Veri Kaynağı 2 dışındaki veri kaynaklarında bulunmamakta ve nesnesi olan ns3:o2 diğer veri kaynaklarında nesne olarak kullanılmamaktadır.

Örneklerden anlaşıldığı gibi bir yüklem aynı zamanda birden fazla biricik yüklem özelliğine sahip olabilir. Bu yüksek lisans tezi kapsamında Biricik yüklem ve Bölüm 2.4.4.3'de anlatılan sorgu tipleri arasındaki ilişkileri kullanılarak biricik yüklem veri kaynağı eleme (UPSP) algoritması geliştirilmiştir. Bu algoritmanın çalışabilmesi için önceden dizin oluşturulması gerekmektedir. Dizin yapısı ve oluşturma algoritması detaylı olarak Bölüm 3.2'de anlatılmıştır.

3.2 Dizin Oluşturma

Bu tez çalışması kapsamında oluşturulan dizin yapısı Bölüm 2.4.4.1'de açıklanan Hibiscus (Saleem & Ngonga Ngomo, 2014) dizin yapısını temel almaktadır. Geliştirilen veri kaynağı eleme algoritması Hibiscus ile birlikte çalıştığı için iki dizin yapısının birbirleri ile uyumlu olması önemlidir. Dizin yapısı olarak Hibiscus'un var olan dizinine dört yeni alan eklenmiştir. Bu alanlar ssUnique, soUnique, osUnique ve ooUnique olarak isimlendirilmiştir ve sırası ile özne-özne, özne-nesne, nesne-özne ve nesne-nesne biricik yüklem özelliklerini tutmaktadırlar. Eğer bir yüklem örneğin nesne-özne biricik değil ise dizin boyutunu ufak tutmak amacı ile osUnique bilgisi dizine yazılmamaktadır. Yani varsayılan olarak tüm değerler yanlış (false) kabul edilmektedir. *Şekil 3.2*'de görülebildiği gibi http://dbpedia.org/ontology/casNumber yüklemi özne-nesne biricik olmadığı için soUnique değeri dizine hiç eklenmemiştir.

```
a ds:Service ;
     ds:url   <http://jamendo.ecozkan.com/sparql> ;
     ds:capability
          [
            ds:predicate
<http://purl.org/ontology/mo/track_number> ;
            ds:sbjAuthority <http://dbtune.org> ;
            ds:ssUnique true ;
            ds:soUnique true ;
            ds:osUnique true ;
            ds:ooUnique true ;
          ] ;
.
[] a ds:Service ;
     ds:url   <http://dbpedia.ecozkan.com/sparql> ;
ds:capability
          [
            ds:predicate  <http://dbpedia.org/ontology/casNumber> ;
            ds:sbjAuthority <http://dbpedia.org> ;
            ds:ssUnique true ;
            ds:osUnique true ;
            ds:ooUnique true ;
          ] ;
```

Şekil 3.2 Biricik Yüklem Dizini

Dizin yapısını oluşturmak için Hibiscus'un var olan dizin yapısına biricik yüklem kontrolü eklenmiştir. Biricik yüklem kontrolü algoritmasını *Şekil 3.3*'de görebilirsiniz.

Algoritma: Biricik Yüklem Dizin Oluşturma Algoritması

Requires endpoints //list of available datasets

```
1.  for each e1 in endpoints do
2.    for each p1 in predicates(e1) do
3.       sbjAuthP1 = sbjAuth(p1,e1)
4.       objAuthP1 = objAuth(p1,e1)
5.       if isUnique(p1)
6.            for each e2 in endpoints do
7.                 for each p2 in predicates(e2) do
8.                      sbjAuthP2 = sbjAuth(p2,e2)
9.                      objAuthP2 = objAuth(p2,e2)
10.                         if sbjAuthP1 ∩ sbjAuthP2
11.                            checkSubjectSubjectUnique(p1,p2,e1,e2)
12.                         if sbjAuthP1 ∩ objAuthP2
13.                            checkSubjectObjectUnique(p1,p2,e1,e2)
14.                         if objAuthP1 ∩ sbjAuthP2
15.                            checkObjectSubjectUnique(p1,p2,e1,e2)
16.                         if objAuthP1 ∩ objAuthP2
17.                            checkObjectObjectUnique(p1,p2,e1,e2)
```

Şekil 3.3 Biricik Yüklem Dizin Oluşturma Algoritması

Şekil 3.3'de görülen biricik yüklem dizin oluşturma algoritması ulaşılabilen tüm veri kaynaklarının tüm yüklemleri için Hibiscus (Saleem & Ngonga Ngomo, 2014) dizin yapısında kullanılan özne ve nesne alan adlarını (Subject Authority, Object Authority) sbjAuthP1 ve objAuthP1 listelerine atmaktadır (Satır 1-4). Eğer ilgili yüklem biricik yani başka bir veri kaynağında bulunmuyor ise diğer tüm veri kaynaklarında bulunan yüklemler ile özne-özne, özne-nesne, nesne-özne ve nesne-nesne karşılaştırılması yapılmaktadır (Satır 5-17). Bu karşılaştırma yapılmadan önce Bölüm 2.4.4'de anlatılan alan adı karşılaştırması yapılarak gereksiz sorgu çalıştırmanın önüne geçilmeye çalışılmıştır. Örneğin iki yüklemin özne alan adları arasında kesişme yok ise yüklemlerin aynı özneyi kullanma ihtimali bulunmamaktadır. Bu sayede gereksiz kontrol işlemleri engellenebilmektedir. Karşılaştırma işlemi özel bir SPARQL sorgu tipi olan ASK sorgusu ile yapılmaktadır. ASK sorgusu eğer verilen özne-yüklem-nesne üçlüsü veri kaynağında var ise doğru cevap dönmektedir. Karşılaştırma işlemi yapılırken veri kaynaklarının yeri bilindiğinden dolayı SPARQL 1.1 standardı kullanılmıştır. Özne-özne, özne-nesne, nesne-özne ve nesne-nesne kontrolleri için gönderilen şablon sorgular aşağıda açıklamaları ile birlikte verilmiştir.

- **Özne-Özne:** ASK { SERVICE <e2> {?s <p2> ?o2.} ?s <p1> ?o1 .} Bu sorgu e1 veri kaynağında çalıştırıldığında p1 ve p2 yüklemleri arasında ortak özne var ise doğru sonuç dönmektedir.
- **Özne-Nesne:** ASK { SERVICE <e2> {?s1 <p2> ?s.} ?s <p1> ?o1 .} Bu sorgu e1 veri kaynağında çalıştırıldığında p1 yükleminin öznesi, p2 tarafından nesne olarak kullanılıyorsa doğru sonuç dönmektedir.
- **Nesne-Özne:** ASK { SERVICE <e2> {?o1 <p2> ?s.} ?s1 <p1> ?o1 .} Bu sorgu e1 veri kaynağında çalıştırıldığında p1 yükleminin nesnesi, p2 tarafından özne olarak kullanılıyorsa doğru sonuç dönmektedir.
- **Nesne-Nesne:** ASK { SERVICE <e2> {?s2 <p2> ?o.} ?s1 <p1> ?o .} Bu sorgu e1 veri kaynağında çalıştırıldığında p1 ve p2 yüklemleri arasında ortak nesne var ise doğru sonuç dönmektedir.

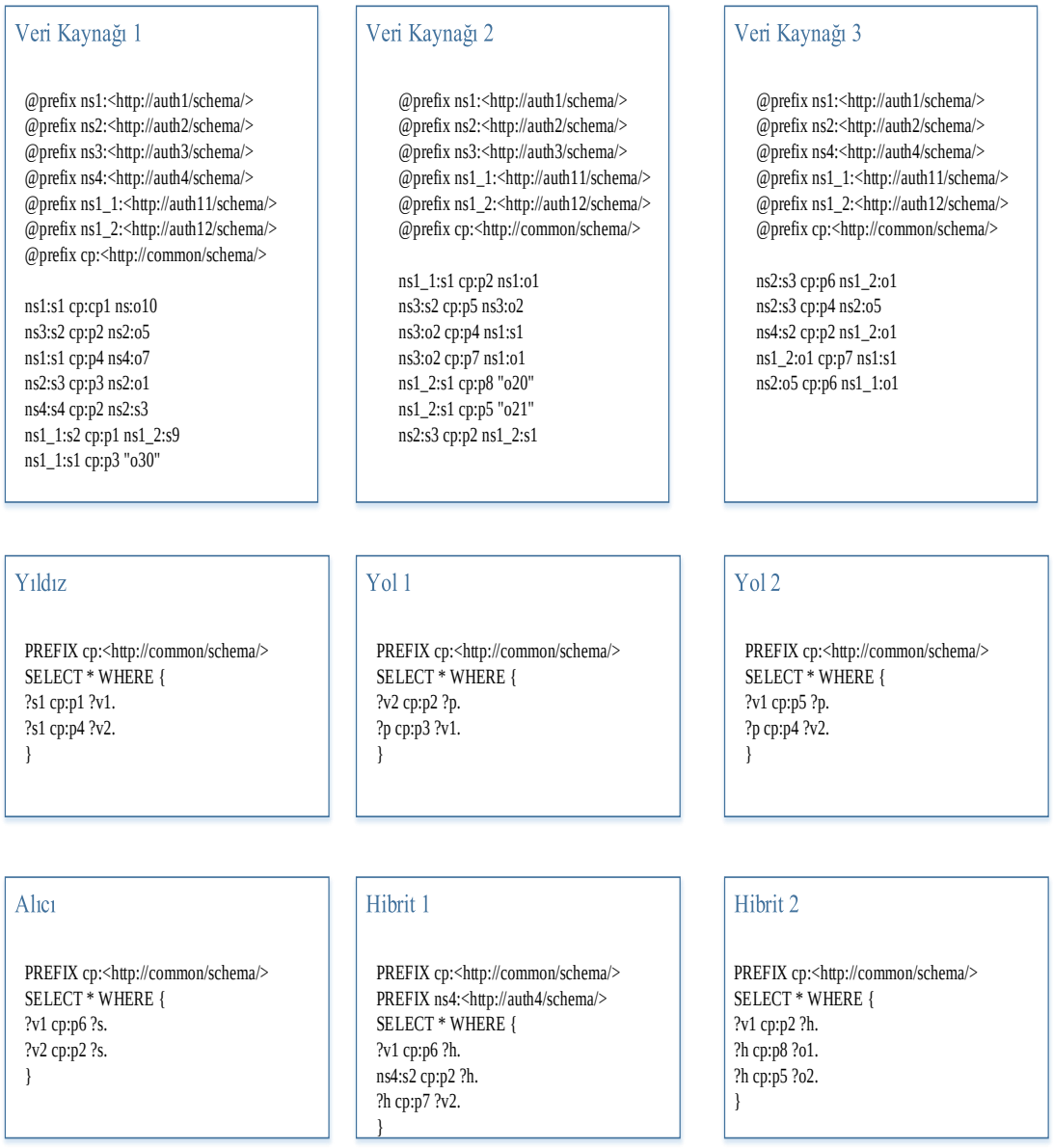
Bu işlemler sonucunda Hibiscus alan adı dizinine ekstra biricik yüklem bilgileri eklenerek dizin yapısı oluşturulmaktadır.

Dizin oluşturma algoritması gerçekleştirimi yapıldıktan sonra dizin oluşturma aşamasında eşzamanlı yapının yeterli olmayacağı tespit edildi. Bu sebeple java ThreadPool ve ConcurrentHashMap sınıfları kullanılarak çok iş parçacıklı şekilde gerçekleştirimi yapıldı. Her yüklem için bir işlem parçacığı açılarak dizin oluşturma işlemi başlatıldı. Biricik yüklem bulmanın ASK sorguları yüzünden maliyetli bir işlem olması sebebi ile çok iş parçacıklı yapı 600 tane ile dizin oluşturma süresini ortalama 60 saatten 36 saate düşürebildi. Sorgu eleme işleminde sağlayacağı kazanç düşünüldüğünde 36 saatin yeterli bir süre olduğunu düşünülmektedir. Bu dizin kullanılarak yapılan eleme işlemi Bölüm 3.3'de anlatılmaktadır.

3.3 Biricik Yüklem Elemesi ile Sorgu Optimizasyonu

Tez çalışması kapsamında geliştirilen UPSP algoritması Bölüm 2.4.4.4'de anlatılan Hibiscus eleme algoritması ile aynı döngü içerisinde Hibiscus elemesinden hemen önce çalışmaktadır. Algoritma Hibiscus eleme algoritmasının işini kolaylaştırmak amacı ile tasarlanmıştır ve Hibiscus hiper-çizge sorgu modelini kullanmaktadır. Yani Bölüm 2.4.4.3'de anlatılan hiper bağlar üzerinde bulunan veri kaynaklarını elemeyi

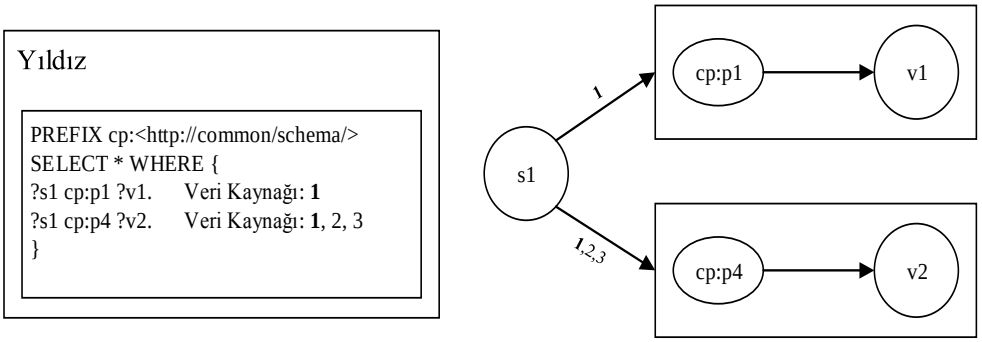
amaçlamaktadır. UPSP algoritması Bölüm 3.1'de açıklanan biricik yüklem tipleri ile Bölüm 2.4.4.3'de açıklanan sorgu tipleri arasındaki ilişkiyi kullanılarak veri kaynağı eleme işlemini gerçekleştirmektedir.



Şekil 3.4 Biricik Yüklem Örnek Veri ve Sorgular

Aşağıda *Şekil 3.4*'de bulunan örnek veri kaynakları ve sorgular üzerinden biricik yüklem ve sorgu tipleri arasındaki ilişki anlatılmaktadır.

- **Yıldız:** Yıldız sorgu tipi, özneler arası ilişki sonucu oluştuğu için, yıldız sorgu tipine giren yüklemlerden, en az biri özne-özne biricik ise diğer hiper bağlar üzerinde bulunan harici veri kaynaklarının hepsi silinebilir. Çünkü özne-özne biricik yüklem olduğu durumda diğer veri kaynakları ile birleşme işlemine girememektedir. Örneğin Yıldız örnek sorgusu düşünüldüğünde cp:p1 yüklemi özne-özne biriciktir. Yani Veri Kaynağı 1 üzerinde cp:p1 in öznesi olarak kullanılan düğümler, diğer veri kaynaklarında özne olarak kullanılmamıştır. Dolayısı ile harici bir veri kaynağı ile birleşme işlemine giremez. Bu sebeple yıldız sorgu tipine giren hiper bağların harici veri kaynakları silinebilir.

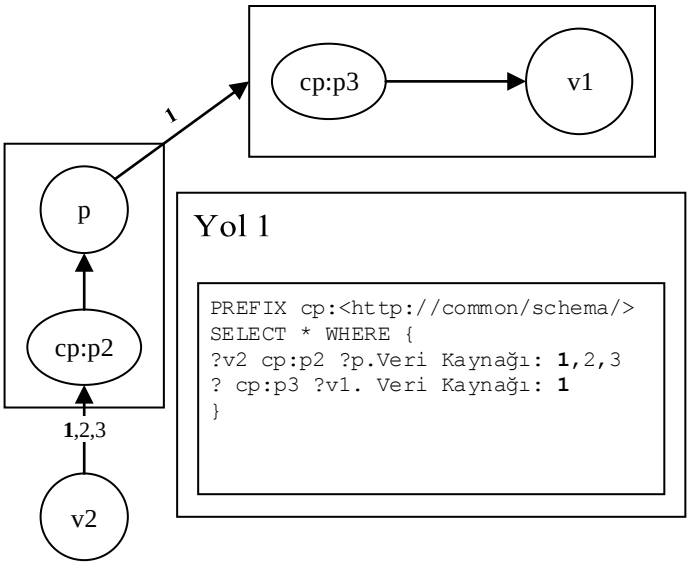


Şekil 3.5 Hiper-Çizge ile Yıldız Sorgu Tipi

Şekil 3.5'te görüldüğü gibi cp:p1 ile cp:p4 yüklemleri yıldız birleşim tipinde sorgu oluşturmuşlar. Cp:p4 yüklemi tüm veri kaynaklarında bulunduğu için hiper bağı üzerinde tüm veri kaynakları yazmaktadır. Ancak cp:p1 özne-özne biricik tipinde olduğu için cp:p4 hiper bağı üzerindeki harici veri kaynakları yani 2 ve 3 UPSP algoritması tarafından elenmiştir.

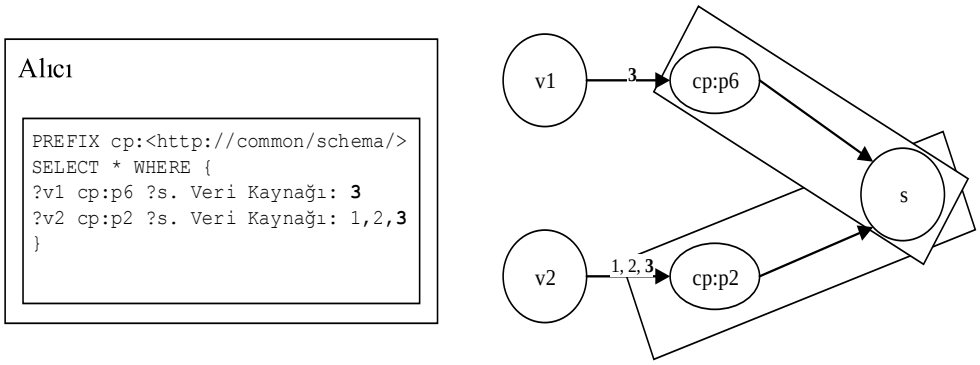
- **Yol:** Bir düğüm yol tipinde ise iki şekilde UPSP algoritması çalışabilir. Bunlardan ilki ilgili düğümden çıkan yüklemler, özne-nesne biricik ise düğüme giren hiper bağlar üzerindeki tüm harici kaynaklar silinebilir. İkincisi ise ilgili düğüme giren yüklemler, nesne-özne ise düğümden çıkan hiper bağlar üzerindeki harici kaynaklar da silinebilir. *Şekil 3.6* de bulunan Yol 1 örnek sorgusu düşünüldüğünde, p düğümü yol tipinde bir sorgu oluşturmuştur

ve p düğümünden çıkan cp:p3 yüklemi özne-nesne biricik olduğu için p düğümüne giren tüm hiper bağlar üzerinde harici veri kaynakları elenebilir.



Şekil 3.6 Hiper-Çizge ile Yol Sorgu Tipi

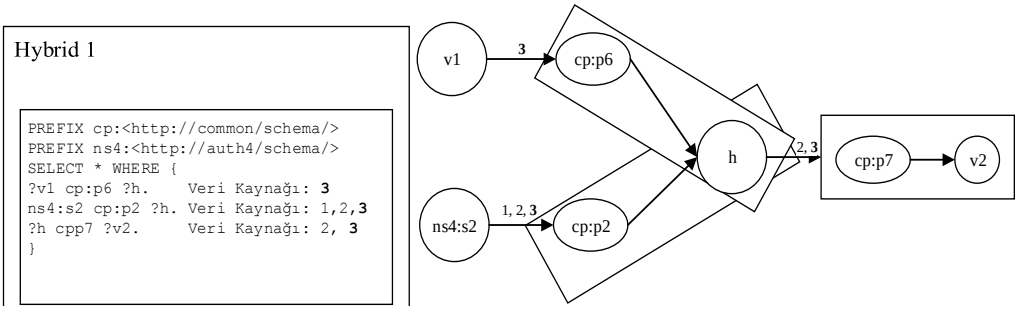
- **Alıcı:** Bir düğüm alıcı tipinde sorgu içinde bulunduğunda çıkan yüklemlerden, biri nesne-nesne biricik ise düğüme giren tüm hiper bağlar üzerindeki harici kaynaklar silinebilir. Çünkü alıcı düğüm tipi her zaman nesne-nesne ilişki ile oluşur. Yüklemlerden biri nesne-nesne biricik ise başka veri kaynağı ile birleşme işlemine giremeyeceği anlamına geldiğinden dolayı harici veri kaynakları elenebilir.



Şekil 3.7 Hiper-Çizge ile Alıcı Sorgu Tipi

Şekil 3.7'de hiper-çizge gösterimi verilen alıcı sorgu örneğinde cp:p6 nesne-nesne biricik olduğu için diğer hiper bağlar üzerinde veri kaynağı 3 dışındaki harici kaynaklar elenmiştir.

- **Hibrit:** Diğer tüm sorgu tiplerinin birleşimi olan hibrit sorgu tipi diğer durumların birleşiminden oluşmaktadır. Yani yıldız, yol ve alıcı için bulunan kurallar bu tip içinde uygulanabilir. Eğer bir düğüm hibrit tipinde bir sorgu içerisinde bulunuyorsa 4 farklı durum oluşmaktadır. İlgili düğümden
    - Gelen yüklemlerden en az biri
        - Nesne-özne biricik ise ilgili düğümden çıkan hiper bağlarından harici veri kaynakları elenebilir.
        - Nesne-nesne biricik ise ilgili düğüme gelen hiper bağlarından harici veri kaynakları elenebilir.
    - Çıkan yüklemlerden en az biri
        - Özne-nesne biricik ise düğüme gelen hiper bağlarından harici veri kaynakları elenebilir.
        - Özne-özne biricik ise düğümden çıkan hiper bağlarından harici veri kaynakları elenebilir.



Şekil 3.8 Hiper-Çizge ile Hibrit Sorgu Tipi

Şekil 3.8'de verilen hibrit tipindeki örnek sorguda cp:p6 yüklemi nesne-nesne ve nesne-özne biricik özelliklerine sahip olduğu için h düğümünden çıkan ve gelen hiper bağlar üzerindeki harici veri kaynakları elenebilir.

Örneklerden görülebileceği gibi biricik yüklem elemesi özne-özne, özne-nesne, nesne-özne ve nesne-nesne özelliklerini kullanarak harici veri kaynakları ile birleşme yapamayacağı durumları tespit etmek üzerine kurulmuştur. Dizin oluşturması maliyetli bir yöntem olmasına rağmen etkili bir şekilde veri kaynaklarını eleyebilmektedir. Eleme işlemini yapan algoritma *Şekil 3.9*'da görülebilmektedir.

Algoritma: Biricik Yüklem Veri Kaynağı Eleme (UPSP) Algoritması

Requires DHG // Hypergraph of query

```
1.  for each HG in DHG do
2.      for each v in vertices(HG) do
3.           if isStarNode(v)
4.              for each e in v.outEdge do
5.                if e.predicate is "Subject-Subject"
6.                     pruneExternal(e, v.outEdge)
7.              //Hibiscus pruning
8.           else if isPathNode(v)
9.              eOut = v.outEdge[0]
10.             eIn = v.inEdge[0]
11.             if eOut.predicate is "Subject-Object"
12.               pruneExternal(e, v.inEdge)
13.             if eIn.predicate is "Object-Subject"
14.               pruneExternal(e, v.outEdge)
15.             //Hibiscus pruning
16.          else if isSinkNode(v)
17.             for each e in v.inEdge do
18.               if e.predicate is "Object-Object"
19.                  pruneExternal(e, v.inEdge)
20.             //Hibiscus pruning
21.          else if isHybridNode(v)
22.             for each eIn in v.inEdge
23.               if eIn.predicate is "Object-Subject"
24.                  pruneExternal(eIn, v.outEdge)
25.               if eIn.predicate is "Object-Object"
26.                  pruneExternal(eIn, v.inEdge)
27.             for each eOut in v.outEdge
28.               if eOut .predicate is "Subject-Object"
29.                  pruneExternal(eOut , v.inEdge)
30.               if eOut .predicate is "Subject-Subject"
31.                  pruneExternal(eOut , v.outEdge)
32.             //Hibiscus pruning
```

Şekil 3.9 UPSP Algoritması

Şekil 3.9'da görülebilen UPSP algoritması hiper-çizgede bulunan tüm düğümler için

- Yıldız tipinde ise, çıkan düğümler arasında özne-özne biricik yüklem bulunuyorsa çıkan hiper bağlardan harici veri kaynaklarını eler (Satır 3-6).
- Yol tipinde ise çıkan düğüm özne-nesne ise gelen hiper bağdan harici veri kaynaklarını eler (Satır 11-12), gelen düğüm nesne-özne biricik ise çıkan hiper bağdan harici veri kaynaklarını eler (Satır 13-14).
- Alıcı ise gelen düğüm nesne-nesne ise gelen hiper bağdan harici veri kaynaklarını eler (Satır 16-19).

- Hibrit ise:
    - Gelen düğüm nesne-özne ise çıkan hiper bağlardan harici kaynakları eler (Satır 23-24).
    - Gelen düğüm nesne-nesne ise gelen hiper bağlardan harici kaynakları eler (Satır 25-26).
    - Çıkan düğüm özne-nesne ise gelen hiper bağlardan harici kaynakları eler (Satır 28-29).
    - Çıkan düğüm özne-özne ise çıkan hiper bağlardan harici kaynakları eler (Satır 30-31).

Algoritmadan görülebileceği gibi Biricik Yüklem elemesi Hibiscus eleme metoduna veri kaynağı seçme süresini uzatmamak için eklenti olarak tasarlanmıştır. Biricik yüklem algoritması testleri Bölüm 5'te detaylı olarak anlatılacaktır.

4. TEST ORTAMI

Biricik Yüklem eleme testleri, dağıtık sorgulama alanında yaygın olarak kullanılan Fedbench (Schmidt et al., 2011) test aracı ile yapıldı. Fedbench aracı DBPedia, NyTimes gibi yaygın olarak bilinen dağıtık veri kaynaklarını test verisi olarak kullanmaktadır ve bunlar arasında dağıtık sorgulama yapan sorgu setleri oluşturmuştur. Dağıtık sorgulama test aracı olarak öne çıkan başka alternatif olmadığı (Rakhmawati et al., 2013) için tez çalışması kapsamında deneyler Fedbench aracı kullanılarak yapıldı. Fedbench aracı dokuz adet bilinen veri kaynağını kullanmaktadır. Bunlar sırası ile DBpedia, NyTimes, Kegg, Drugbank, Chebi, Jamendo, LinkedMDB, Semantic Web Dog Food, Geo Names veri kaynaklarıdır. Bu veri kaynaklarının her biri ayrı sunucuya Amazon Web Services (AWS) altyapısı kullanılarak kurulmuştur. AWS altyapısı kolaylıkla yeni sunucu oluşturulabildiği, sunucu özellikleri ayarlanabildiği için tez çalışması için uygun bir ortam sağlamıştır. Ayrıca sunuculara SSD disk takılabilmesi ve yüksek konfigürasyona sahip sunucular oluşturulabileceği için ortam seçimini AWS olarak yapıldı. Test ortamı Konfigürasyonu, Veri Kaynakları ve Sorguları hakkındaki detaylar bu başlık altında anlatılacaktır.

4.1 Konfigürasyon

Test ortamı Amazon Web Services Elastic Compute Cloud (EC2) Windows Server 2012 R2 üzerine kurulmuştur. Ağ gecikmelerini en aza indirmek için bütün sunucular aynı bölge (Frankfurt – eu-central-1a) üzerinde oluşturulmuştur. Tüm sunucularda disk erişimini hızlandırmak için SSD disk kullanılmıştır. Veri kaynağı sunucusu olarak Virtuoso 07.10.3207 versiyonu kullanılmıştır. Tüm Virtuoso sunucularında dağıtık sorgulama alt yapısı açılmıştır ve işlem zaman aşımı limiti olarak 1200 saniye ayarlanmıştır.

Tez deneyleri kapsamında oluşturulan ortam ve özellikleri Tablo *1*'de listelenmiştir.

Tablo 1 Test Ortamı Konfigürasyonu

Sunucu Adı	AWS Sunucu Tipi	CPU	vCPU	RAM	HDD	IOPS
DBPedia	m3.xlarge	Intel Xeon E5-2670 v2 2.5 GHZ	4	15 GB	60 GB SSD	180/3000
GeoNames	m3.xlarge	Intel Xeon E5-2670 v2 2.5 GHZ	4	15 GB	35 GB SSD	105/3000
Dev Env	m3.xlarge	Intel Xeon E5-2670 v2 2.5 GHZ	4	15 GB	80 GB SSD	240/3000
Nytimes	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000
Kegg	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000
Drugbank	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000
Jamendo	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000
LinkedMDB	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000
Chebi	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000
Semantic Web Dog Food	t2.medium	Intel Xeon 2.5 GHZ	2	4 GB	30 GB SSD	90/3000

“Dev Env” sunucusu, tez çalışması kapsamında geliştirilen dağıtık sorgulama motorunun çalıştığı sunucudur. Diğer sunuculara FedBench veri kaynakları kurulmuştur ve üzerlerinde başka işlem çalışmamaktadır. AWS sunucu tipi seçimlerinde veri kaynağı sunucusunun gereksinimleri etkileyici olmuştur. DBpedia ve GeoNames sunucuları en büyük veri setlerini içerdiğinden dolayı m3.xlarge tipinde sunucular seçilmiştir. Ayrıca Hibiscus testlerinde kullanılan ortamın konfigürasyonuna yakın sunucular seçilmesine özen gösterilmiştir.

4.2 Veri Kaynakları

Fedbench test aracı çoğu test aracının aksine gerçek veri setlerini kullanmaktadır. Veri setleri sırası ile DBpedia, NyTimes, Kegg, Drugbank, Chebi, Jamendo, LinkedMDB, Semantic Web Dog Food, Geo Namesdir. Tez çalışması kapsamında, test ortamı kurulurken, Hibiscus çalışması kapsamında hazırlanmış Virtuoso

sunucularını kullanılmıştır. Fedbench tarafından yayınlanan veri setleri özellikleri Tablo *2*'de listelenmiştir.

Tablo 2 FedBench Veri Setleri Özellikleri

Kategori	Veri Kaynağı	Sürüm	Üçlü	Özne	Yüklem	Nesne	Tipler	Bağlar
Cross Domain	DBpedia Subset	3.5.1	43.6M	9.50M	1063	13.6M	248	61.5k
	NY Times	2010-01-13	335k	21.7k	36	192k	2	31.7k
	LinkedMDB	2010-01-19	6.15M	694k	222	2.05M	53	63.1k
	Jamendo	2010-11-25	1.05M	336k	26	441k	11	1.7k
	GeoNames	2010-10-06	108M	7.48M	26	35.8M	1	118k
	SW Dog Food	2010-11-25	104k	12.0k	118	37.5k	103	1.6k
Life Sciences	DBpedia Subset	3.5.1	43.6M	9.50M	1063	13.6M	248	61.5k
	KEGG	2010-11-25	1.09M	34.3k	21	939k	4	30k
	Drugbank	2010-11-25	767k	19.7k	119	276k	8	9.5k
	CheEBI	2010-11-25	7.33M	50.5k	28	772k	1	-

Tablo *2*'de görüldüğü gibi, veri setleri "Cross Domain" ve "Life Sciences" olarak ikiye ayrılmıştır. "Cross Domain" altında belirli bir kategoriye ait olmayan veriler yer almaktadır. "Life Sciences" kategorisi altında bulunan veri setlerinde ise tıbbi içerikli veriler bulunmaktadır. DBpedia ve GeoNames en büyük veri setleridir. Bu sebeple konfigürasyon yapılırken yüksek özellikli sunucular verilmiştir.

Biricik yüklem dizin oluşturma işlemi Tablo *2*'de listelenen veri setleri üzerinde çalıştırılmıştır.

Şekil 4.1'te FedBench veri setleri üzerinde çalıştırılan dizin oluşturma algoritması sonucunda bulunan toplam yüklem, özne-özne, özne-nesne, nesne-özne, nesne-nesne biricik yüklem sayıları gösterilmiştir.

Toplam Yüklem	2652
Özne-Özne Biricik Yüklem	1533
Özne-Nesne Biricik Yüklem	1477
Nesne-Özne Biricik Yüklem	1533
Nesne-Nesne Biricik Yüklem	1500

Şekil 4.1 Tüm Veri kümeleri için biricik yüklem İstatistikleri

Bölüm 4.3'de FedBench sorguları ve bunların biricik yüklem ilişkileri anlatılacaktır.

4.3 Sorgular

FedBench sorguları Bölüm 4.2'de anlatılan veri setleri üzerinde çalışmaktadır. Dört kategoriye ayrılmış şekilde toplam 33 sorgu bulunmaktadır. Tez çalışması sırasında Hibiscus deneyleri ile karşılaştırma yapılacağı için Hibiscus testlerinde kullanılan 25 tane sorgu üzerinden biricik yüklem testleri çalıştırılmıştır. Testler sırasında kullandığım FedBench sorguları EK 1'de listelenmiştir. Testler sırasında Cross Domain, Life Science ve Linked Data kategorilerini kullanılmıştır.

Tablo 3 Fedbench Sorgu Seti Özeti

Sorgu	Sorgu Sonucu Kayıt Sayısı	Biricik Yüklem İçeriyor mu?	Sorgu Tipi
CD1	90	Hayır	Yıldız
CD2	1	Evet	Yıldız
CD3	2	Evet	Yıldız, Hibrit
CD4	1	Evet	Yıldız, Yol
CD5	2	Evet	Yıldız, Yol
CD6	11	Hayır	Yıldız, Yol
CD7	1	Evet	Yıldız, Yol
LS1	1159	Hayır	Basit
LS2	333	Hayır	Yıldız
LS3	9054	Evet	Yıldız, Yol
LS4	3	Evet	Yıldız, Alıcı, Hibrit
LS5	393	Evet	Yıldız, Alıcı, Yol
LS6	28	Evet	Alıcı, Yıldız
LS7	145	Evet	Yıldız, Alıcı
LD1	309	Evet	Yıldız, Yol
LD2	186	Evet	Yıldız, Yol
LD3	162	Evet	Yıldız, Hibrit
LD4	50	Evet	Yıldız, Yol, Alıcı
LD5	28	Evet	Yıldız
LD6	39	Evet	Yıldız, Yol
LD7	1216	Hayır	Yıldız
LD8	22	Evet	Yıldız, Hibrit
LD9	1	Evet	Yıldız, Yol
LD10	3	Hayır	Yıldız, Yol
LD11	376	Evet	Yıldız, Yol

Tablo *3*'de testler kapsamında kullanılan sorguların, sonuç sayısı, biricik yüklem içerme durumu ve sorgu tipi özellikleri listelenmiştir. Sorgunun biricik yüklem

içeriyor olması, sorgu tipi alanlarından en az biri ile uyumlu biricik yüklem bulunması anlamına gelmektedir. Sorgu tipi ve biricik yüklem ilişkisi Bölüm 3.3'te detaylı olarak açıklanmıştır. Biricik Yüklem İçeriyor Mu kolonunda "Evet" şeklinde belirtilen sorgular üzerinde, Biricik Yüklem Eleme (UPSP) algoritmasının çalışması beklenmektedir. Şekil *4.2*'de örnek olarak verilen CD4 sorgusu LinkedMDB ve NYTimes veri kaynakları üzerinde çalışmaktadır. Yapılan testler kapsamında elde edilen sonuçlar Bölüm 5'de detaylı olarak anlatılacaktır.

```
SELECT ?actor ?news WHERE {
   ?film <http://purl.org/dc/terms/title> 'Tarzan' .
   ?film <http://data.linkedmdb.org/resource/movie/actor> ?actor .
   ?actor <http://www.w3.org/2002/07/owl#sameAs> ?x.
   ?y <http://www.w3.org/2002/07/owl#sameAs> ?x .
   ?y <http://data.nytimes.com/elements/topicPage> ?news
}
```

Şekil 4.2 Fedbench CD4 Sorgusu

4.4 Testler

UPSP algoritması elenen veri kaynağı sayısı, veri kaynağı eleme süresi sorgu çalışma süresi ve konfigürasyon yüklenme süreleri üzerinden Hibiscus algoritması ile karşılaştırılmıştır. UPSP ve Hibiscus algoritması yaygın olarak kullanılan ve Bölüm 2.4.2 ve Bölüm 2.4.3'te anlatılan FedX ve Splendid dağıtık sorgulama motorları üzerine geliştirilerek testleri yapılmıştır. Testler her iki dağıtık sorgulama motorunda da bulunan "ASK Dominant" ve "Index Dominant" kipleri için yapılmıştır ve UPSP ile Hibiscus algoritmalarının sonuçları kendi içlerinde karşılaştırılmıştır. Her sorgu beşer kere çalıştırılmıştır. Veri kaynağı eleme ve çalışma süreleri sonuçlarında ortalama alınmıştır. Ortalama alınan sonuçlarda maksimum ve minimum değerler dışarıda tutularak ortalamalar hesaplanmıştır. UPSP algoritması, Hibiscus algoritmasından önce çalışacak şekilde tasarlandığı için UPSP algoritması testlerinde UPSP algoritmasından hemen sonra Hibiscus veri kaynağı eleme algoritması da çalışmaktadır.

5. TEST SONUÇLARI VE DEĞERLENDİRME

UPSP algoritması testleri Bölüm 4'te detayları anlatılan ortam üzerinde çalıştırılmıştır. Bu tez kapsamında geliştirilen algoritma, Splendid ve FedX dağıtık sorgulama motorları üzerine eklenmiştir. Testler "ASK Dominant" ve "Index Dominant" kipleri için tekrarlanmıştır. Her sorgu beşer kere çalıştırılmıştır. "Toplam Çalışma Süresi", "Veri Kaynağı Eleme Süresi", "Konfigürasyon Yüklenme Süresi" ve "Elenen Veri Kaynağı Sayıları" Hibiscus değerleri ile karşılaştırılmıştır. "Veri Kaynağı Eleme Süresi", UPSP ve Hibiscus algoritmalarının veri kaynağı seçimi için harcanan zamandır. "Dizin Yüklenme Süresi" ise dağıtık sorgulama motoru ilk açıldığında, dizinin yüklenmesi ve konfigürasyonların yapılması için harcanan süredir. "Toplam Çalışma Süresi" ise veri kaynağı eleme süresi ve sorgu çalıştırılması için geçen sürenin toplamıdır. "Çalışma Süresi" ve "Veri Kaynağı Eleme Süresi" sonuçlarında ortalama alınmıştır. Ortalamalar en yüksek ve en düşük çıkan sonuçlar dışarıda bırakılarak hesaplanmıştır. "Elenen Veri Kaynağı Sayısı" UPSP ve Hibiscus algoritmaları tarafından elenen veri kaynaklarının sayısıdır. Deneyler sonucunda elde edilen detaylı sonuçlar EK 2 ve EK 3'de gösterilmektedir.

5.1 Veri Kaynağı Eleme Testleri

Veri kaynağı eleme testleri Bölüm 4.3 ve 4.2'de belirtilen sorgular ve veri kaynakları üzerinde önceden hazırlanmış dizin kullanılarak çalıştırılmıştır. Dizin oluşturulması Bölüm 3.2'de anlatılan iş parçacıklı yapı ile 36 saat sürmektedir. Hibiscus ve UPSP dizin yapısı benzerdir. Bu sebeple Hibiscus ve UPSP testleri için aynı dizin kullanılmıştır. Biricik yüklem eleme (UPSP) algoritması Hibiscus üzerine çalıştırılmış ve orijinal Hibiscus algoritması ile sonuçları karşılaştırılmıştır. Tablolarda iyileşme olan sonuçlar kalın yazılmıştır. Tablolarda bulunan sütunlar detaylı olarak aşağıda verilmiştir.

- **İlk Seçilen Kaynak:** UPSP ve Hibiscus algoritmalarının üzerine geliştirildiği dağıtık sorgulama motoru tarafından ilk önerilen kaynak sayısıdır. "ASK

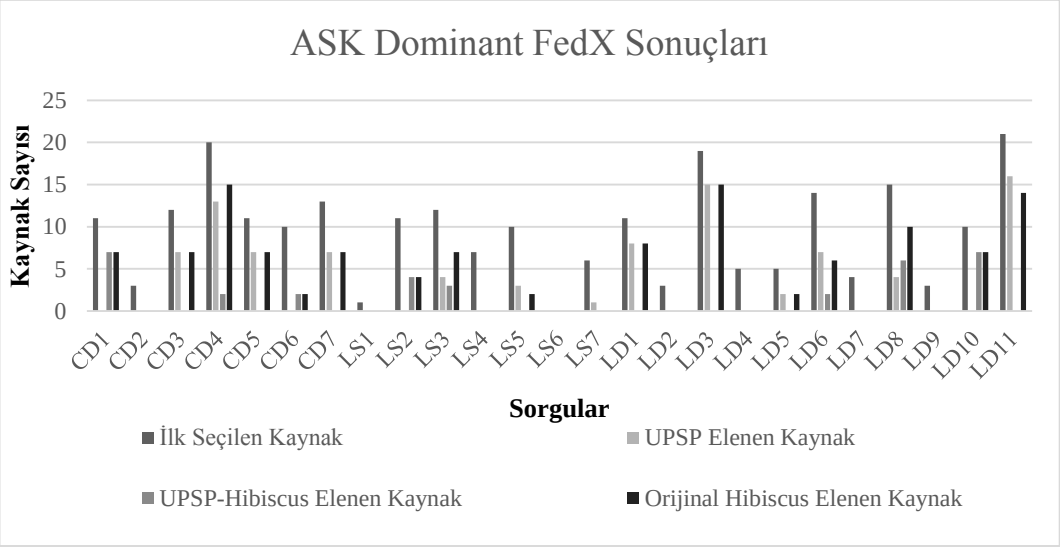
Dominant" kipi ise SPARQL ASK sorguları, "Index Dominant" kipinde ise VoID dizini yardımı ile kaynaklar önerilmektedir.

- **UPSP İle Elenen Kaynak Sayısı:** UPSP algoritması çalışması sonucunda elenen kaynak sayısıdır.
- **Hibiscus ile Elenen Kaynak Sayısı:** UPSP algoritması çalıştıktan sonra Hibiscus tarafından elenen kaynak sayısıdır.
- **UPSP+Hibiscus Elenen Kaynak Sayısı:** UPSP ve Hibiscus tarafından toplam elenen kaynak sayısıdır. İyileşme olan sonuçlar kalın olarak gösterilmiştir.
- **Orijinal Hibiscus:** UPSP algoritması yok iken Hibiscus tarafından elenen toplam kaynak sayısıdır.

Bu testler kapsamında elde edilen detaylı sonuçları EK 3'de bulabilirsiniz.

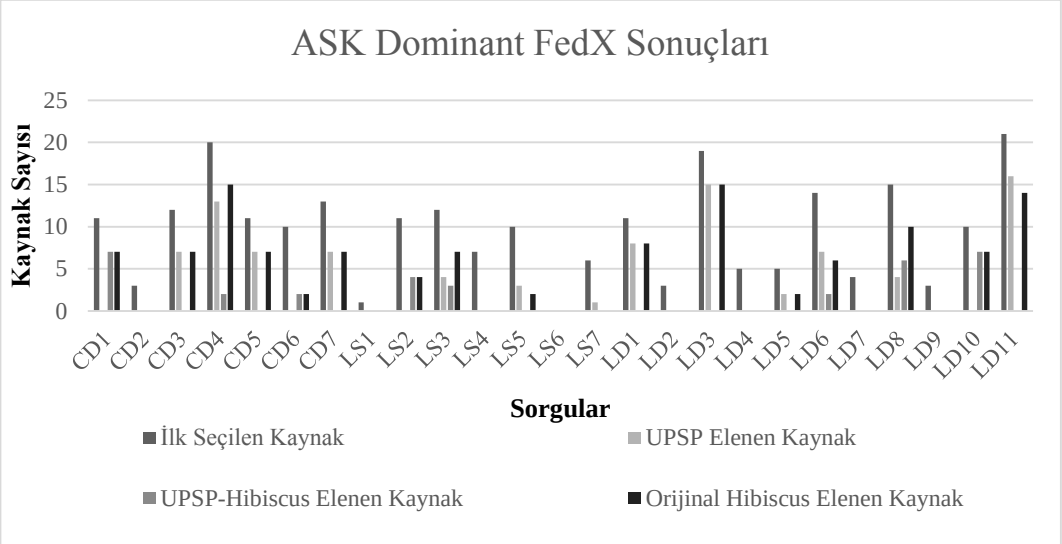
5.1.1 FedX

Bu bölümde FedX dağıtık sorgulama motoru üzerinde Hibiscus algoritması ve daha sonra UPSP algoritması çalıştırılarak yapılan deneyler açıklanmıştır. "Veri Kaynağı Eleme Sayısı" sonuçları "ASK Dominant" ve "Index Dominant" kipleri için ayrı ayrı karşılaştırılacaktır.



Şekil 5.1 ve Tablo *4*'de "ASK Dominant" kipinde çalışan FedX dağıtık sorgulama aracının UPSP algoritması testleri ile Hibiscus üzerine yapılan test sonuçları görüntülenmektedir. Her test 5 defa çalıştırılmıştır ve eleme sayılarında herhangi bir

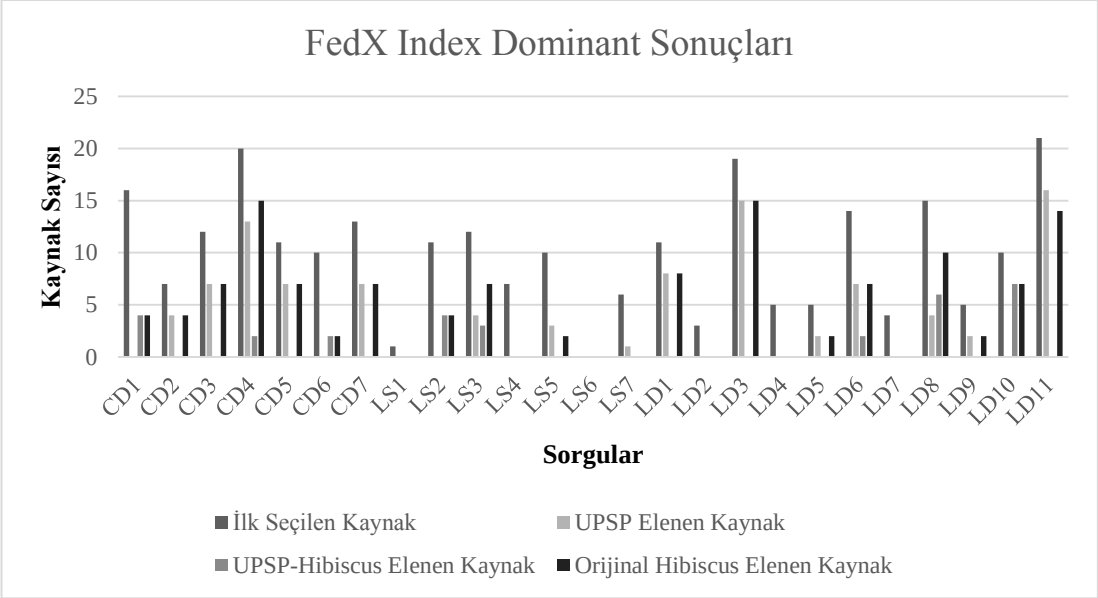
değişiklik olmadığı doğrulanmıştır. Sonuçlardan görülebileceği gibi LS5, LS7 LD6, LD11 sorgularında, orijinal Hibiscus eleme algoritmasından daha fazla kaynak elenerek daha az sorgu çalıştırılması sağlanmıştır. Bu alanda veri kaynağı eleme açısından, orijinal Hibiscus algoritmasına göre %20'lik bir artış gözlemlenmiştir.



Şekil 5.1 ASK Dominant FedX Eleme Sonuçları

Tablo 4 ASK Dominant FedX Eleme Sonuçları Tablosu

		UPSP + Hibiscus Kaynak Eleme			
Sorgu	**İlk Seçilen Kaynak**	**UPSP İle Elenen Kaynak Sayısı**	**Hibiscus ile Elenen Kaynak Sayısı**	**UPSP+Hibiscus Elenen Kaynak Sayısı**	**Orijinal Hibiscus**
CD1	11	0	7	7	7
CD2	3	0	0	0	0
CD3	12	7	0	7	7
CD4	20	13	2	15	15
CD5	11	7	0	7	7
CD6	10	0	2	2	2
CD7	13	7	0	7	7
LS1	1	0	0	0	0
LS2	11	0	4	4	4
LS3	12	4	3	7	7
LS4	7	0	0	0	0
LS5	10	3	0	**3**	2
LS6	-	-	-	-	-
LS7	6	1	0	**1**	0
LD1	11	8	0	8	8
LD2	3	0	0	0	0
LD3	19	15	0	15	15
LD4	5	0	0	0	0
LD5	5	2	0	2	2
LD6	14	7	2	**9**	6
LD7	4	0	0	0	0
LD8	15	4	6	10	10
LD9	3	0	0	0	0
LD10	10	0	7	7	7
LD11	21	16	0	**16**	14



Şekil 5.2 Index Dominant FedX Sonuçları

Şekil 5.2 ve Tablo *5*'de "Index Dominant" kipinde çalıştırılmış FedX dağıtık sorgulama motorunun biricik yüklem algoritması ile orijinal Hibiscus algoritmasının veri kaynağı eleme sonuçları görülebilmektedir. Bu deneyde LS5, LS7, LD6 ve LD11 sorgularında orijinal Hibiscus algoritmasından daha fazla veri kaynağı elenerek %20lik iyileşme olduğu gözlemlenmiştir.

FedX dağıtık sorgulama motoru sonuçları incelendiğinde; LS1, LD2, LD4, LD7 sorgularında, FedX'in önermiş olduğu "İlk Seçilen Kaynaklar" üzerinde eleme yapılamadığından, bu sorgularda Hibiscus ve UPSP algoritmaları çalışamamıştır. LS6 sorgusunda ise işlem zaman aşımına uğradığından dolayı sorgu sonuçları kaydedilememiştir.

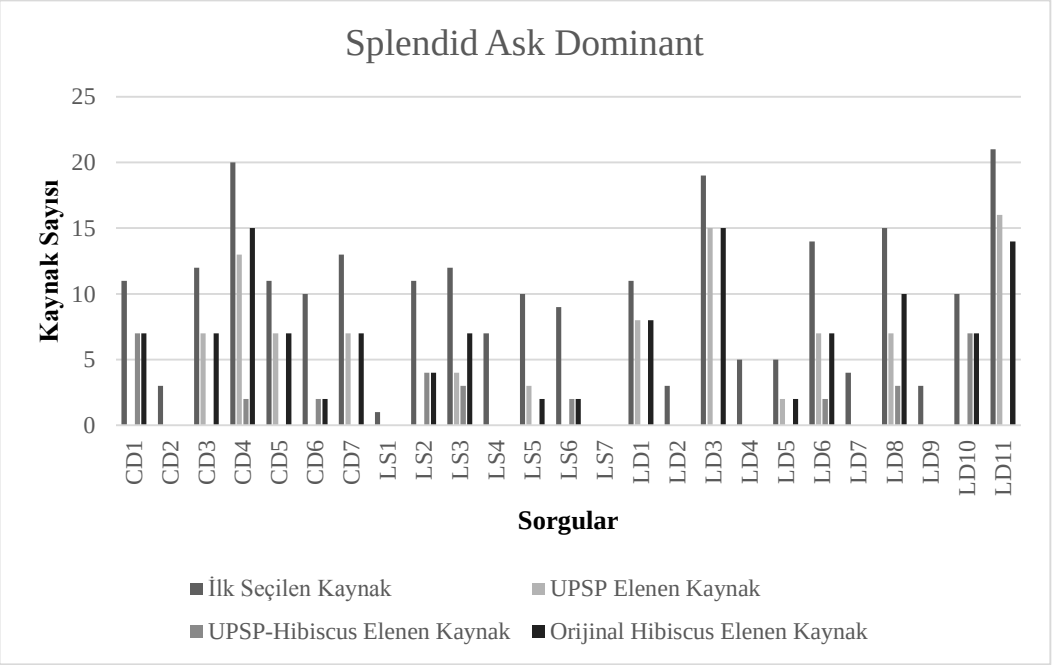
Tablo 5 Index Dominant FedX Eleme Sonuçları Tablosu

		UPSP + Hibiscus Kaynak Eleme			
Sorgu	**İlk Seçilen Kaynak**	**UPSP İle Elenen Kaynak Sayısı**	**Hibiscus ile Elenen Kaynak Sayısı**	**UPSP+Hibiscus Elenen Kaynak Sayısı**	**Orijinal Hibiscus**
CD1	16	0	4	4	4
CD2	7	4	0	4	4
CD3	12	7	0	7	7
CD4	20	13	2	15	15
CD5	11	7	0	7	7
CD6	10	0	2	2	2
CD7	13	7	0	7	7
LS1	1	0	0	0	0
LS2	11	0	4	4	4
LS3	12	4	3	7	7
LS4	7	0	0	0	0
LS5	10	3	0	**3**	2
LS6	-	-	-	-	-
LS7	6	1	0	**1**	0
LD1	11	8	0	8	8
LD2	3	0	0	0	0
LD3	19	15	0	15	15
LD4	5	0	0	0	0
LD5	5	2	0	2	2
LD6	14	7	2	**9**	7
LD7	4	0	0	0	0
LD8	15	4	6	10	10
LD9	5	2	0	2	2
LD10	10	0	7	7	7
LD11	21	16	0	**16**	14

5.1.2 Splendid

Bu başlıkta Splendid dağıtık sorgulama motoru üzerinde Hibiscus çalıştırılmış ve daha sonra UPSP algoritması çalıştırılarak yapılan deneyler açıklanacaktır. "Veri Kaynağı Eleme Sayısı" sonuçları "ASK Dominant" ve "Index Dominant" kipleri için ayrı ayrı karşılaştırılacaktır.

Şekil 5.3 ve Tablo 6'daki sonuçlar gözlemlendiğinde LS5, LD6 ve LD11 sorgularında orijinal Hibiscus algoritmasına göre daha fazla veri kaynağı elendiği görülebilmektedir. Diğer sorgularda toplamda Hibiscus ile aynı miktarda veri kaynağı elenmiştir. Bu deneyde UPSP algoritması orijinal Hibiscus algoritmasına oranla %12 daha başarılıdır.

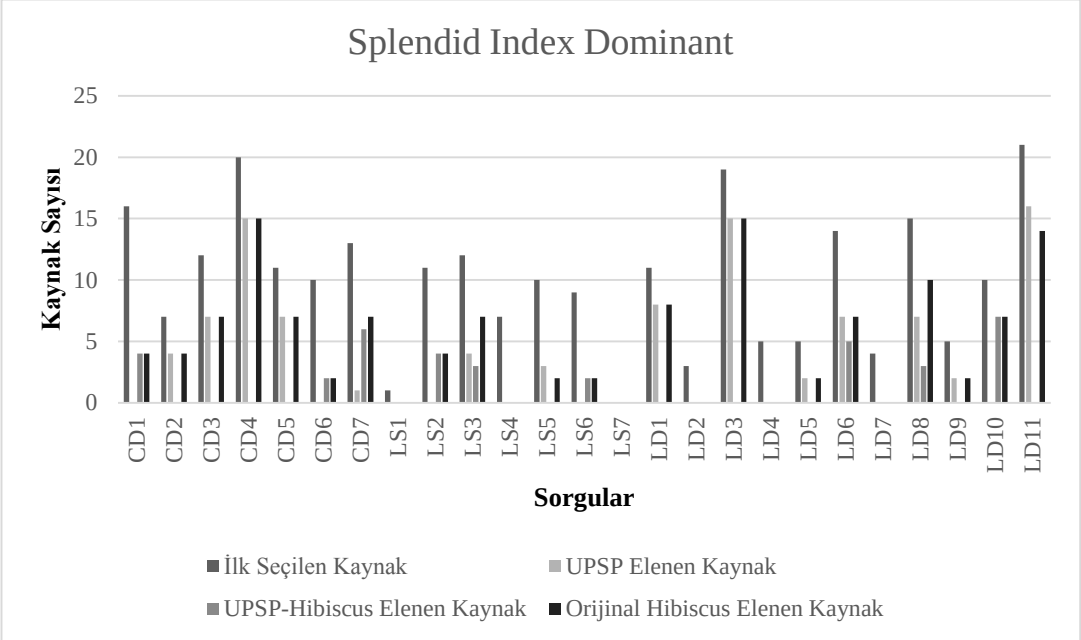


Şekil 5.3 ASK Dominant Splendid Sonuçları

Tablo 6 ASK Dominant Splendid Eleme Sonuçları Tablosu

		UPSP + Hibiscus Kaynak Eleme			
Sorgu	**İlk Seçilen Kaynak**	**UPSP İle Elenen Kaynak Sayısı**	**Hibiscus ile Elenen Kaynak Sayısı**	**UPSP+Hibiscus Elenen Kaynak Sayısı**	**Orijinal Hibiscus**
CD1	11	0	7	7	7
CD2	3	0	0	0	0
CD3	12	7	0	7	7
CD4	20	13	2	15	15
CD5	11	7	0	7	7
CD6	10	0	2	2	2
CD7	13	7	0	7	7
LS1	1	0	0	0	0
LS2	11	0	4	4	4
LS3	12	4	3	7	7
LS4	7	0	0	0	0
LS5	10	3	0	**3**	2
LS6	9	0	2	2	2
LS7	-	-	-	-	-
LD1	11	8	0	8	8
LD2	3	0	0	0	0
LD3	19	15	0	15	15
LD4	5	0	0	0	0
LD5	5	2	0	0	2
LD6	14	7	2	**9**	7
LD7	4	0	0	0	0
LD8	15	7	3	10	10
LD9	3	0	0	0	0
LD10	10	0	7	7	7
LD11	21	16	0	**16**	14

Şekil 5.4 ve Tablo *7*'de görülen sonuçlar incelendiğinde biricik yüklem eleme yönteminin LS5, LD6 ve LD11 numaralı sorgularda iyileştirme yaptığı görülebilmektedir. Bu deneyde UPSP algoritması veri kaynağı seçimini %12 daha iyi şekilde yapmıştır.



Şekil 5.4 Index Dominant Splendid Sonuçları

Tablo 7 Index Dominant Splendid Eleme Sonuçları Tablosu

		UPSP + Hibiscus Kaynak Eleme			
Sorgu	**İlk Seçilen Kaynak**	**UPSP İle Elenen Kaynak Sayısı**	**Hibiscus ile Elenen Kaynak Sayısı**	**UPSP+Hibiscus Elenen Kaynak Sayısı**	**Orijinal Hibiscus**
CD1	16	0	4	4	4
CD2	7	4	0	4	4
CD3	12	7	0	7	7
CD4	20	15	0	15	15
CD5	11	7	0	7	7
CD6	10	0	2	2	2
CD7	13	1	6	7	7
LS1	1	0	0	0	0
LS2	11	0	4	4	4
LS3	12	4	3	7	7
LS4	7	0	0	0	0
LS5	10	3	0	**3**	2
LS6	9	0	2	2	2
LS7	-	-	-	-	-
LD1	11	8	0	8	8
LD2	3	0	0	0	0
LD3	19	15	0	15	15
LD4	5	0	0	0	0
LD5	5	2	0	2	2
LD6	14	7	5	**12**	7
LD7	4	0	0	0	0
LD8	15	7	3	10	10
LD9	5	2	0	2	2
LD10	10	0	7	7	7
LD11	21	16	0	**16**	14

Splendid sonuçları incelendiğinde “ASK Dominant” kipinde CD2, LS1, LS4, LD2, LD4, LD7, LD9 sorgularında, “Index Dominant” kipinde ise LS1, LS4, LD2 ,LD4, LD7 sorgularında hem UPSP hem de orijinal Hibiscus algoritmalarının veri kaynağı elemediği gözlenmektedir. “Index Dominant” kipinde daha çok veri kaynağı eleme işlemi gerçekleştirilmiştir. Çünkü Splendid dağıtık sorgulama motoru “Index Dominant” kipinde çalıştığında daha fazla veri kaynağı önerdiği için daha çok eleme

işlemi yapılmaktadır. LS7 sorgusunda ise Splendid dağıtık sorgulama motoru zaman aşımı hatası aldığı için, bu sorguya ait sonuçlar bulunmamaktadır.

Bölüm 4.3'te özellikleri listelenen sorguların bazılarında biricik yüklem ve buna uygun sorgu tipleri olmasına rağmen biricik yüklem algoritmasının çalışmadığı durumlar oluşmuştur. LS4, LS6, LD2, LD4 sorgularında kullanılan dağıtık sorgulama motorunun oluşturduğu veri kaynağı listesi eleman sayısı, sorguda bulunan hiper bağ sayısından daha az olduğu için bu sorgularda eleme algoritmasının çalışmasına gerek olmamıştır.

5.2 Veri Kaynağı Eleme ve Çalışma Süresi Testleri

Bu başlık kapsamında FedX ve Splendid dağıtık sorgulama motorlarının "Veri Kaynağı Eleme Süreleri", "ASK Dominant" ve "Index Dominant" kiplerinde önce Hibiscus çalıştırılıp daha sonra UPSP algoritması çalıştırılacak karşılaştırılacaktır. Gösterilen tüm sonuçlar milisaniye bazındadır ve Java "System.currentTimeMillis" fonksiyonu kullanılarak ölçüm yapılmıştır. "Veri Kaynağı Eleme Süresi"; dağıtık sorgulama aracından elde edilen, veri kaynağı listesi üzerinde yapılan eleme işlemine harcanan süredir. "Çalışma Zamanı" ise veri kaynağı eleme süresine ilaveten sorgunun çalıştırılıp sonuçların kullanıcıya verilmesine kadar geçen süredir. Sonuçlar, beş defa çalıştırılan testlerin minimum ve maksimum değerleri çıkartılmasının ardından kalan sonuçların ortalaması alınarak elde edilmiştir. Bu başlık altında tüm sorgular açıklanmamıştır. Çünkü milisaniye bazında yapılan ölçümler ağ durumu, sunucu işlemleri gibi etkenlerden kolaylıkla etkilenebilmektedir. Bu sebeple UPSP algoritmasının fazladan veri kaynağı elediği sorgular açıklanmıştır. Testler kapsamında elde edilen sonuçlar için EK 4'e bakabilirsiniz. Diğer sorgularda sonuçlar yakın çıktığından dolayı tabloda verilmemiştir.

5.2.1 FedX

Bu başlıkta FedX motoru kullanılarak yapılan veri kaynağı eleme zaman testleri sonuçları açıklanacaktır.

Tablo 8 FedX ASK Dominant Ortalama Çalışma ve Eleme Süreleri

Sorgu	UPSP Ortalama Çalışma Süresi (ms)	Orijinal Hibiscus Ortalama Çalışma Süresi (ms)	UPSP Ortalama Eleme Süresi (ms)	Orijinal Hibiscus Ortalama Eleme Süresi (ms)	İlk Seçilen Kaynak	UPSP Seçilen Kaynak	Orijinal Hibiscus Seçilen Kaynak
LS5	1946	1948	31	26	10	**7**	8
LS7	1823	1793	27	21	6	**5**	6
LD6	**298**	314	36	35	14	**5**	8
LD11	**385**	1286	**31**	38	21	**5**	7

Tablo *8*'de FedX dağıtık sorgulama motoru "ASK Dominant" kipinde çalışıyor iken çalışma süresi ve veri kaynağı seçme süreleri karşılaştırmalı olarak görülebilmektedir. Tablodan görülebileceği gibi LD11 sorgusunda orijinal Hibiscus değerlerine göre daha fazla veri kaynağı eleyerek çalışma süresinde %70, veri kaynağı seçme süresinde ise %18'lik kazanç sağlanmıştır. LS5 ve LS7 sorgularında daha fazla veri kaynağı elediği halde daha fazla çalışma ve veri kaynağı seçim süresi harcamışlardır. LS5 ve LS7 sorgularında küçük veri kaynakları elendiğinden dolayı "Çalışma Süresi" ve "Veri Kaynağı Seçim Süresi" daha fazla çıkmıştır. Çünkü küçük veri kaynaklarını elemek bu veri kaynaklarına gönderilen sorguları işlemekten daha maliyetlidir. Fazladan elenen veri kaynaklarına gönderilen sorgular, yüksek maliyetli sorgular olmadığı durumda sonuçlar daha düşük performanslı çıkmıştır.

Tablo 9 FedX Index Dominant Ortalama Çalışma ve Eleme Süreleri

Sorgu	UPSP Ortalama Çalışma Süresi (ms)	Orijinal Hibiscus Ortalama Çalışma Süresi (ms)	UPSP Ortalama Eleme Süresi (ms)	Orijinal Hibiscus Ortalama Eleme Süresi (ms)	İlk Seçilen Kaynak	UPSP Seçilen Kaynak	Orijinal Hibiscus Seçilen Kaynak
LS5	2216	2195	102	78	10	**7**	8
LS7	2146	1986	87	65	6	**5**	6
LD6	**436**	496	**68**	112	14	**5**	7
LD11	**583**	1457	**92**	122	21	**5**	7

FedX dağıtık sorgulama motoru "Index Dominant" kipinde iken yapılan veri kaynağı seçme ve çalışma zamanı ölçümleri Tablo *9*'da görülebilmektedir. LD6 sorgusunda UPSP algoritması daha fazla veri kaynağı eleyerek, çalışma ve veri kaynağı eleme sürelerinde sırası ile %11 ve %39'luk iyileşme göstermiştir. Aynı şekilde LD11 sorgusunda ise %60 ve %24'lük iyileşme görülebilmektedir. LS5 ve LS7 sorguları ise "ASK Dominant" kipinde olduğu gibi eleme işlemine daha fazla kaynak harcamışlardır. Bu sebeple bu sorgularda UPSP algoritmasının çalışması daha uzun sürmüştür. Fazladan elenen veri kaynakları küçük olduğu durumda buralara gönderilen sorgular eleme işleminden daha az maliyetli olduğu için "Eleme ve Çalışma Süresi"'nde daha fazla süre harcanmıştır.

5.2.2 Splendid

Bu başlık kapsamında Splendid dağıtık sorgulama motoru kullanılarak yapılmış veri kaynağı eleme ve çalışma zamanı testleri açıklanacaktır.

Tablo 10 Splendid ASK Dominant Ortalama Çalışma ve Eleme Süreleri

Sorgu	UPSP Ortalama Çalışma Süresi (ms)	Orijinal Hibiscus Ortalama Çalışma Süresi (ms)	UPSP Ortalama Eleme Süresi (ms)	Orijinal Hibiscus Ortalama Eleme Süresi (ms)	İlk Seçilen Kaynak	UPSP Seçilen Kaynak	Orijinal Hibiscus Seçilen Kaynak
LS5	**13660**	16105	29	24	10	**7**	8
LD6	662	588	36	36	14	**5**	7
LD11	**417**	5578	33	33	21	**5**	7

Tablo *10*'da Splendid "ASK Dominant" kipi aktif iken yapılmış veri kaynağı seçme deneylerinin, "Çalışma ve Veri Kaynağı Seçme Süreleri" sonuçlarını karşılaştırmalı olarak görülmektedir. LS5 ve LD11 sorgularının sonuçları incelendiğinde UPSP algoritması çalıştığı durumda daha fazla veri kaynağı elediğinden dolayı toplam "Çalışma Süresinin" iyileştiği gözlemlenmektedir. LS5 sorgusunda daha çok veri kaynağı elenmesinden dolayı, "Veri Kaynağı Seçme Süresi" %22 artmasına rağmen veri kaynağı seçimi daha iyi yapıldığı için "Çalışma Süresinde" %15'lik bir düşüş

görülmüştür. LD11 sorgusunda ise "Veri Kaynağı Seçimi Süresinde" herhangi bir değişiklik gözlemlenmemesine rağmen daha az veri kaynağına sorgu gönderilmesi sebebi ile çalışma süresinde %92'lik bir iyileşme gözlemlenmiştir. LD6 sorgusunda ise kaynak seçimine harcanan süre fazla olduğu için toplam "Çalışma Süresi" yüksek gözükmektedir. Çünkü küçük veri kaynaklarını eleme işlemi buralara gönderilen sorgu daha az maliyetlidir.

Tablo 11 Splendid Index Dominant Ortalama Çalışma ve Eleme Süreleri

Sorgu	UPSP Ortalama Çalışma Süresi (ms)	Orijinal Hibiscus Ortalama Çalışma Süresi (ms)	UPSP Ortalama Eleme Süresi (ms)	Orijinal Hibiscus Ortalama Eleme Süresi (ms)	İlk Seçilen Kaynak	UPSP Seçilen Kaynak	Orijinal Hibiscus Seçilen Kaynak
LS5	14040	16033	81	67	10	**7**	8
LD6	**551**	743	**75**	105	14	**2**	8
LD11	**474**	5823	**64**	123	21	**5**	7

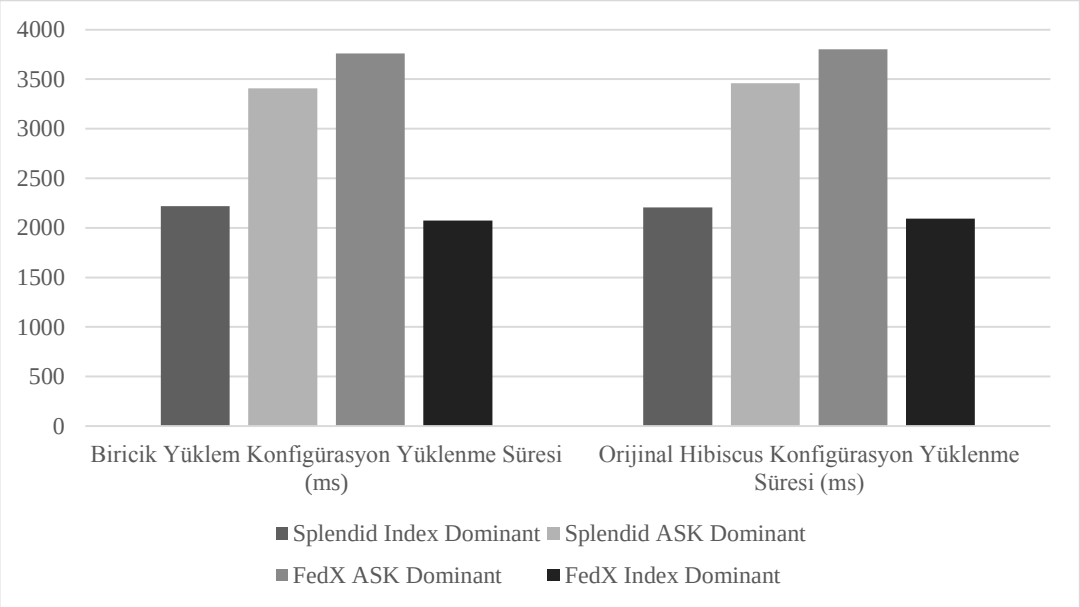
Splendid sorgu motoru ile "Index Dominant" kipinde yapılan deney sonuçları Tablo *11*'de görülmektedir. Tablodan görülebildiği gibi LD6 ve LD11 sorguları biricik yüklem elemesinde daha fazla veri kaynağı eleyerek toplam çalışma ve eleme süresinde daha iyi sonuçlar vermişlerdir. LD6 sorgusunda, çalışma süresi olarak %25 ve veri kaynağı seçme süresi olarak %28 kazanç sağlanmıştır. LD11 sorgusunda ise çalışma süresinde %91, veri kaynağı seçme süresinde %48 kazanç sağlanmıştır. LD6 ve LD11 sorgularında kaynak eleme süresinin daha düşük çıkmasının ana sebebi, UPSP algoritmasının Hibiscus algoritmasından önce çalışarak Hibiscus'a kaynak eleme açısından daha az iş bırakmasıdır. LS5 sorgusunda ise daha fazla kaynak elemesi yapıldığı için, eleme süresi daha yüksek çıksa da kaynak elemesi iyi yapılması nedeni ile daha az sorgu gönderildiği için toplam süre daha iyi çıkmıştır. Veri kaynağı seçme süresi olarak %20 kayba karşılık toplam sürede %12 kazanç sağlamıştır.

5.3 Konfigürasyon Yükleme Süresi Testleri

Bu başlık kapsamında dağıtık sorgulama motoru ilk çalıştırıldığında dizin yüklenmesi, ayarların yapılması için geçen süre ölçülmüştür. Diğer deneyler gibi

FedX ve Splendid SPARQL motorları üzerinde “ASK Dominant” ve “Index Dominant” kiplerinde çalıştırılmıştır. Deney sonuçları Hibiscus ile UPSP algoritması için karşılaştırılmıştır.

Konfigürasyon yüklenme süresi deneylerinde *Şekil 5.5*’de görülebileceği gibi birbirlerine yakın sonuçlar çıktığı gözlemlenmiştir. “ASK Dominant” kiplerde yapılan deney sonuçlarının dağıtık sorgulama motoru açılışında yapılan sorgular sebebi ile daha yüksek çıktığı gözlemlenmiştir.



Şekil 5.5 Konfigürasyon Yükleme Süreleri

6. SONUÇLAR VE GELECEK ÇALIŞMALAR

Dağıtık sorgulama, bağlı veri bulutunu tek nokta üzerinden sorgulamayı amaçlayan bir yapıdır. Dağıtık sorgulama optimizasyonunda önemli konulardan biri olan veri kaynağı seçme algoritmasının başarılı sayılabilmesi için hızlı ve doğru sonuç vermesi gerekmektedir. Dağıtık sorgulama için SPARQL 1.1 ve SPARQL 1.0 standartları üzerine kurulmuş dağıtık sorgulama motorları bulunmaktadır. Dağıtık sorgulama motorunun hızlı ve doğru sonuç vermesinin yanında, kullanılabilir olması da gerekmektedir. Bağlı veri gibi büyük bir veri bulutu düşünüldüğünde, veri kaynağı seçme işleminin kullanıcı tarafından yapıldığı SPARQL 1.1 standardı kullanılabilirlik alanında başarılı olamamaktadır. Çünkü büyük veri bulutu üzerinde aranılan verinin, hangi kaynak üzerinde bulunduğu bilgisini el ile bulmak zor ve zahmetli bir iştir. Kullanıcının üçlülerin hangi veri kaynağında bulunduğunu bilmeden sorgulama yapabilmesi gerekmektedir. Bu sebeple bu işlemin SPARQL 1.0 standardı üzerine yapılan eklentilerle otomatikleştirilmesi daha kullanışlı olmaktadır. SPARQL 1.0 üzerine geliştirilen dağıtık sorgulama motorlarının en önemli işlevleri veri kaynağını bulmak ve oluşturulan alt sorgu sonuçlarını doğru bir şekilde birleştirmektir.

Veri kaynağı eleme algoritmaları, genellikle VoID adı verilen veri kaynağı istatistikleri ve veri kaynaklarına SPARQL ASK sorguları gönderilerek gelen sonuçlara göre veri kaynaklarını seçme yöntemini kullanırlar. Bu yöntem doğru sonuç üretmesine rağmen hız ve ağ maliyeti açısından başarılı olmamaktadır. Bu nedenle Birleşim-Farkında Eleme (Join Aware Source Selection) (JAAS) Algoritmaları öne çıkmaktadır. Çünkü SPARQL sorgusu yapısı gereği özne-yüklem-nesne arasındaki bağları kullanarak yazılmaktadır. Bu bağlar dikkate alınarak veri kaynağı seçimi yapıldığında gereksiz alt sorguların önüne kolaylıkla geçilebilmektedir. Hibiscus dağıtık sorgulama motoru çalışmasında başarılı bir şekilde Birleşim-Farkında Eleme (JAAS) Algoritması geliştirilmiştir. Hibiscus veri kaynağı eleme algoritması yüklemin özne ve nesne olarak kullandığı düğümlerin alan adları bilgisini karşılaştırarak eleme işlemini yapmaktadır.

Bu tez kapsamında başarılı bir dağıtık sorgulama motoru olan Hibiscus'un veri kaynağı eleme algoritması üzerinde iyileştirme önerileri sunulmuş ve yapılan testler kapsamında bulunan sonuçlar paylaşılmıştır.

Tez çalışması kapsamında, Hibiscus eleme algoritmasından önce çalışan Biricik Yüklem Eleme (UPSP) algoritması önerilmiştir. Biricik yüklem sadece bir adet veri kaynağında bulunabilen ve özne-özne, özne-nesne, nesne-özne ve nesne-nesne ilişkilerinden en az birine sahip yüklemdir. İyileştirme önerisi Hibiscus sorgu gösterim modeli olan hiper-çizge üzerindeki sorgu tipleri ile biricik yüklem tipleri arasındaki ilişkiyi kullanarak veri kaynağı eleme işlemini gerçekleştirmektedir.

Biricik yüklem bilgileri önceden oluşturulmuş bir dizin üzerinden sorgulanarak elde edilmektedir. Oluşturulan bu dizin Hibiscus dizini ile uyumlu olacak şekilde tasarlanmıştır ve var olan Hibiscus dizinine isteğe bağlı biricik yüklem özelikleri alan olarak eklenmiştir. Biricik yüklem bulmak için yüklem diğer tüm yüklemler ile ortak ASK sorgusu gönderilmektedir. Bu nedenle biricik yüklem dizini oluşturmak oldukça maliyetli bir işlem olduğu için dizin oluşturma algoritması olabildiğince az ASK sorgusu yapacak şekilde geliştirilmiştir. ASK sorgusu oluşturulmadan önce yüklemlerin alan adı bilgileri karşılaştırılarak gereksiz sorguların önüne geçilmiştir.

Önerilen iyileştirme Hibiscus üzerine eklendikten sonra dağıtık sorgulama alanında en yaygın kullanılan FedBench test aracı kullanılarak 25 adet sorgu üzerinden deneyler yapılmıştır. Deneyler UPSP algoritması Hibiscus ile birlikte FedX ve Splendid aracına eklenerek yapılmıştır. "ASK Dominant" ve "Index Dominant" kiplerinde her sorgu beşer kere çalıştırılmıştır. Sonuçların ortalaması maksimum ve minimum sonuçlar çıkarılarak alınmıştır.

Deney sonuçları incelendiğinde veri kaynağı eleme performansı FedX aracı üzerinde %20, Splendid aracı üzerinde %12'lik iyileşme göstermiştir. Çalışma ve veri kaynağı eleme süreleri açısından yüzde olarak değer verilmemiştir. Çünkü milisaniye bazında ölçüm yapıldığı için ağ, sunucu işlemleri, önbellek gibi etkenler sonuçları değiştirebilmektedir. Bu sebeple biricik yüklem algoritmasının fazladan veri kaynağı elediği ve süre olarak yüksek iyileştirmelere sahip olan sorgular açıklanmıştır.

Çalışma ve veri kaynağı eleme süresi deneylerinde FedX "ASK Dominant" kipinde LD11 sorgusunda, UPSP Algoritması daha fazla veri kaynağı eleyerek %70'lik iyileştirme sergilemiştir. LS5 ve LS7 sorgularında daha fazla veri kaynağı elendiği halde çalışma ve veri kaynağı seçim süresi artmıştır. FedX "Index Domiant" kipinde sırası ile çalışma ve eleme süresinde LD6 %11 ve %39, LD11 ise %60 ve %24'lük iyileşme göstermiştir. LS5 ve LS7 sorguları daha fazla veri kaynağı elenmesine rağmen daha çok süre harcamışlardır. Çünkü küçük veri kaynaklarını elemek bu veri kaynaklarına gönderilen sorgulardan daha maliyetlidir.

Splendid çalışma süresi deneylerinde Splendid "ASK Dominant" kipinde LS5 sorgusu için sırası ile çalışma ve veri kaynağı eleme sürelerinde %15'lik iyileşme ve %22'lik artma gözlemlenmiştir. LD11 sorgusunda ise, çalışma süresinde %92'lik iyileşme gözlenirken veri kaynağı seçimi süresinde fark gözlemlenmemiştir. LD6 sorgusunun orijinal Hibiscus ile karşılaştırıldığında daha kötü sonuç verdiği gözlemlenmiştir. "Index Dominant" kipinde yapılan deneylerde ise LD6 ve LD11 sorgularında sırası ile çalışma süresi alanında %25 ve %92'lik, veri kaynağı seçme süresi alanında ise %28 ve %48'lik iyileşme gözlemlenmiştir. LS5 sorgusunda ise veri kaynağı eleme süresinde %20'lik düşüş gözlemlenirken, çalışma süresinde %12'lik iyileşme gözlemlenmiştir. Çalışma süresi alanı diğer sürelerin toplamını içerdiğinden dolayı veri kaynağı seçimi süresindeki artış çalışma süresini etkilemektedir. Bu etkinin elenen veri kaynaklarına harcanan süre ve veri kaynaklarının büyüklüğüne göre değiştiği gözlemlenmiştir. Daha fazla veri kaynağı elendiğinde, fazladan elenen kaynaklar büyük veya gönderilen alt sorgular maliyetli ise çalışma süresinde genel olarak bir düşüş gözlemlenmiştir. Küçük veri kaynağı olduğu ve maliyeti düşük bir alt sorgu olduğu durumda, kaynak elemesine harcanan fazladan sürenin hem çalışma hem de veri kaynağı eleme süresine artış olarak yansıdığı gözlemlenmiştir. Çünkü küçük veri kaynaklarını elemek sorguyu çalıştırmaktan daha maliyetlidir.

Dizin oluşturma süresi ise biricik yüklem bulma işlemi maliyetli bir işlem olduğundan dolayı 60 saatte gerçekleştirilmiştir. Ancak iş parçacıklı yapıya geçilerek bu değer 36 saate düşürülebilmiştir.

Gelecek çalışmalarda, biricik yüklem bulma algoritması yatayda ölçeklenebilir bir yapıya (Hadoop gibi) uyarlanarak dizin oluşturma süresi iyileştirilebilir. Yatay ölçeklenebilir yapıda metin dosyaları üzerinden "MapReduce" işleri çalıştırılarak dizin oluşturma aşamasında SPARQL kullanılmasının önüne geçilebilir. Dizin yapısının oluşturulma aşamasında veri kaynağı güncellemelerine bağlı olarak artımlı (incremental) yapıya geçilebilir.

KAYNAKLAR

[1] Akar, Z., Halaç, T. G., Ekinci, E. E., & Dikenelli, O. (2012). Querying the web of interlinked datasets using VOID descriptions. In *CEUR Workshop Proceedings* (Vol. 937).

[2] Berners-lee, B. T., & Hendler, J. (2001). The Semantic Web, *21*. doi:10.1007/978-3-642-29923-0

[3] Bizer, C., Heath, T., Idehen, K., & Berners-Lee, T. (2008). Linked data on the web (LDOW2008). In *Proceeding of the 17th international conference on World Wide Web - WWW '08* (p. 1265). New York, New York, USA: ACM Press. doi:10.1145/1367497.1367760

[4] Broekstra, J., Kampman, A., & Harmelen, F. Van. (2002). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *International Semantic Web Conference ISWC*, *1*, 54–68. doi:10.1007/3-540-48005-6_7

[5] McBride, B. (2001). Jena: Implementing the RDF Model and Syntax Specification. *Proceedings of the Second International Workshop on the Semantic Web SemWeb2001*, *40*, 23–28.

[6] Priyatna, F., Aranda, C. B., & Corcho, O. (2013). Applying SPARQL-DQP for federated SPARQL querying over google fusion tables. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7955 LNCS*, 189–193. doi:10.1007/978-3-642-41242-4_22

[7] Quilitz, B., & Leser, U. (2008). Querying distributed RDF data sources with SPARQL. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5021 LNCS*, 524–538. doi:10.1007/978-3-540-68234-9_39

[8] Rakhmawati, N. A., Umbrich, J., Karnstedt, M., Hasnain, A., & Hausenblas, M. (2013). Querying over Federated SPARQL Endpoints - A State of the Art Survey. *CoRR*, *abs/1306.1*(June). Databases; Distributed, Parallel, and Cluster Computing.

[9] Saleem, M., & Ngonga Ngomo, A. C. (2014). HiBISCuS: Hypergraph-based source selection for SPARQL endpoint federation. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8465 LNCS*, 176–191. doi:10.1007/978-3-319-07443-6_13

[10] Schmidt, M., Görlitz, O., Haase, P., Ladwig, G., Schwarte, A., & Tran, T. (2011). FedBench: A benchmark suite for federated semantic data query processing. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7031 LNCS*(PART 1), 585–600. doi:10.1007/978-3-642-25073-6_37

[11] Schwarte, A., Haase, P., Hose, K., Schenkel, R., & Schmidt, M. (2011). FedX: Optimization techniques for federated query processing on linked data. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7031 LNCS*(PART 1), 601–616. doi:10.1007/978-3-642-25073-6_38

[12] Seaborne, A., Polleres, A., Feigenbaum, L., & Williams, G. T. (2013). SPARQL 1.1 Federated Query.
[13] Shadbolt, N., Hall, W., & Berners-Lee, T. (2006). The semantic web revisited. *IEEE Intelligent Systems*. doi:10.1109/MIS.2006.62
[14] Staab, S. (2011). SPLENDID : SPARQL Endpoint Federation Exploiting VOID Descriptions.
[15] Zhao, J., Zhao, J., Cyganiak, R., Cyganiak, R., Hausenblas, M., & Hausenblas, M. (2009). Describing Linked Datasets. *Linked Data on the Web (LDOW2009)*.

EKLER

EK 1: Fedbench Sorguları

Cross Domain	

CD 1	SELECT ?predicate ?object WHERE {
	{ <http://dbpedia.org/resource/Barack_Obama> ?predicate ?object }
	UNION
	{ ?subject <http://www.w3.org/2002/07/owl#sameAs>
	<http://dbpedia.org/resource/Barack_Obama> .
	?subject ?predicate ?object }
	}

CD 2	SELECT ?party ?page WHERE {
	<http://dbpedia.org/resource/Barack_Obama>
	<http://dbpedia.org/ontology/party> ?party .
	?x <http://data.nytimes.com/elements/topicPage> ?page .
	?x <http://www.w3.org/2002/07/owl#sameAs>
	<http://dbpedia.org/resource/Barack_Obama> .
	}

CD 3	SELECT ?president ?party ?page WHERE {
	?president <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
	<http://dbpedia.org/ontology/President> .
	?president <http://dbpedia.org/ontology/nationality>
	<http://dbpedia.org/resource/United_States> .
	?president <http://dbpedia.org/ontology/party> ?party .
	?x <http://data.nytimes.com/elements/topicPage> ?page .
	?x <http://www.w3.org/2002/07/owl#sameAs> ?president .
	}

CD 4	SELECT ?actor ?news WHERE {
	?film <http://purl.org/dc/terms/title> 'Tarzan' .
	?film <http://data.linkedmdb.org/resource/movie/actor> ?actor .
	?actor <http://www.w3.org/2002/07/owl#sameAs> ?x.
	?y <http://www.w3.org/2002/07/owl#sameAs> ?x .
	?y <http://data.nytimes.com/elements/topicPage> ?news
	}

CD 5	SELECT ?film ?director ?genre WHERE {
	?film <http://dbpedia.org/ontology/director> ?director .
	?director <http://dbpedia.org/ontology/nationality>
	<http://dbpedia.org/resource/Italy> .
	?x <http://www.w3.org/2002/07/owl#sameAs> ?film .
	?x <http://data.linkedmdb.org/resource/movie/genre> ?genre .
	}

CD 6	SELECT ?name ?location ?news WHERE {
	?artist <http://xmlns.com/foaf/0.1/name> ?name .
	?artist <http://xmlns.com/foaf/0.1/based_near> ?location .
	?location <http://www.geonames.org/ontology#parentFeature> ?germany .
	?germany <http://www.geonames.org/ontology#name> 'Federal Republic of Germany'
	}

CD 7	SELECT ?location ?news WHERE {
	?location <http://www.geonames.org/ontology#parentFeature> ?parent .
	?parent <http://www.geonames.org/ontology#name> 'California' .
	?y <http://www.w3.org/2002/07/owl#sameAs> ?location .
	?y <http://data.nytimes.com/elements/topicPage> ?news
	}

Life Science	

LS 1	SELECT $drug $melt WHERE {
	{ $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/meltingPoint> $melt. }
	UNION
	{ $drug <http://dbpedia.org/ontology/Drug/meltingPoint> $melt . }
	}

LS 2	SELECT ?predicate ?object WHERE {
	{ <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00201> ?predicate ?object . }
	UNION
	{ <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00201> <http://www.w3.org/2002/07/owl#sameAs> ?caff .
	?caff ?predicate ?object . }
	}

LS 3	SELECT ?Drug ?IntDrug ?IntEffect WHERE {
	?Drug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Drug> .
	?y <http://www.w3.org/2002/07/owl#sameAs> ?Drug .
	?Int <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/interactionDrug1> ?y .
	?Int <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/interactionDrug2> ?IntDrug .
	?Int <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/text> ?IntEffect .
	}

LS 4	SELECT ?drugDesc ?cpd ?equation WHERE {
	?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory> <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/cathartics> .
	?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/keggCompoundId> ?cpd .
	?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/description> ?drugDesc .
	?enzyme <http://bio2rdf.org/ns/kegg#xSubstrate> ?cpd .
	?enzyme <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://bio2rdf.org/ns/kegg#Enzyme> .
	?reaction <http://bio2rdf.org/ns/kegg#xEnzyme> ?enzyme .
	?reaction <http://bio2rdf.org/ns/kegg#equation> ?equation .
	}

LS 5	SELECT $drug $keggUrl $chebiImage WHERE {
	$drug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugs> .
	$drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/keggCompoundId> $keggDrug .
	$keggDrug <http://bio2rdf.org/ns/bio2rdf#url> $keggUrl .
	$drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/genericName> $drugBankName .
	$chebiDrug <http://purl.org/dc/elements/1.1/title> $drugBankName .
	$chebiDrug <http://bio2rdf.org/ns/bio2rdf#image> $chebiImage .
	}

LS 6	SELECT ?drug ?title WHERE {
	?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory> <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/micronutrient> .
	?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/casRegistryNumber> ?id .
	?keggDrug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://bio2rdf.org/ns/kegg#Drug> .
	?keggDrug <http://bio2rdf.org/ns/bio2rdf#xRef> ?id .
	?keggDrug <http://purl.org/dc/elements/1.1/title> ?title .
	}

LS 7	SELECT $drug $transform $mass WHERE {
	{ $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/affectedOrganism> 'Humans and other mammals'.
	$drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/casRegistryNumber> $cas .
	$keggDrug <http://bio2rdf.org/ns/bio2rdf#xRef> $cas .
	$keggDrug <http://bio2rdf.org/ns/bio2rdf#mass> $mass
	FILTER ($mass > '5')
	}
	OPTIONAL { $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/biotransformation> $transform . }
	}

Linked Data	

LD 1	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
	SELECT * WHERE {
	?paper <http://data.semanticweb.org/ns/swc/ontology#isPartOf>
	<http://data.semanticweb.org/conference/iswc/2008/poster_demo_proceedings> .
	?paper <http://swrc.ontoware.org/ontology#author> ?p .
	?p rdfs:label ?n .
	}

LD 2	SELECT * WHERE {
	?proceedings
	<http://data.semanticweb.org/ns/swc/ontology#relatedToEvent>
	<http://data.semanticweb.org/conference/eswc/2010> .
	?paper <http://data.semanticweb.org/ns/swc/ontology#isPartOf>
	?proceedings .
	?paper <http://swrc.ontoware.org/ontology#author> ?p .
	}

LD 3	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
	SELECT * WHERE {
	?paper <http://data.semanticweb.org/ns/swc/ontology#isPartOf>
	<http://data.semanticweb.org/conference/iswc/2008/poster_demo_proceedings> .
	?paper <http://swrc.ontoware.org/ontology#author> ?p .
	?p owl:sameAs ?x .
	?p rdfs:label ?n .
	}

LD 4	SELECT * WHERE {
	?role <http://data.semanticweb.org/ns/swc/ontology#isRoleAt>
	<http://data.semanticweb.org/conference/eswc/2010> .
	?role <http://data.semanticweb.org/ns/swc/ontology#heldBy> ?p .
	?paper <http://swrc.ontoware.org/ontology#author> ?p .
	?paper <http://data.semanticweb.org/ns/swc/ontology#isPartOf>
	?proceedings .
	?proceedings
	<http://data.semanticweb.org/ns/swc/ontology#relatedToEvent>
	<http://data.semanticweb.org/conference/eswc/2010> .
	}

LD 5	PREFIX dbpedia: <http://dbpedia.org/resource/>
	PREFIX dbprop: <http://dbpedia.org/property/>
	PREFIX dbowl: <http://dbpedia.org/ontology/>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
	PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
	PREFIX factbook: <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
	PREFIX mo: <http://purl.org/ontology/mo/>
	PREFIX dc: <http://purl.org/dc/elements/1.1/>
	PREFIX fb: <http://rdf.freebase.com/ns/>
	SELECT * WHERE {
	?a dbowl:artist dbpedia:Michael_Jackson .
	?a rdf:type dbowl:Album .
	?a foaf:name ?n .
	}

LD 6	PREFIX dbpedia: <http://dbpedia.org/resource/>
	PREFIX dbowl: <http://dbpedia.org/ontology/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX linkedMDB: <http://data.linkedmdb.org/resource/>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	SELECT * WHERE {
	?director dbowl:nationality dbpedia:Italy .
	?film dbowl:director ?director.
	?x owl:sameAs ?film .
	?x foaf:based_near ?y .
	?y <http://www.geonames.org/ontology#officialName> ?n .
	}

LD 7	PREFIX dbpedia: <http://dbpedia.org/resource/>
	PREFIX dbowl: <http://dbpedia.org/ontology/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX linkedMDB: <http://data.linkedmdb.org/resource/>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	PREFIX gn: <http://www.geonames.org/ontology#>
	SELECT * WHERE {
	?x gn:parentFeature <http://sws.geonames.org/2921044/> .
	?x gn:name ?n .
	}

LD 8	PREFIX kegg: <http://bio2rdf.org/ns/kegg#>
	PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
	SELECT * WHERE {
	?drug drugbank:drugCategory <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/micronutrient> .
	?drug drugbank:casRegistryNumber ?id .
	?drug owl:sameAs ?s .
	?s foaf:name ?o .
	?s skos:subject ?sub .
	}

LD 9	PREFIX geo-ont: <http://www.geonames.org/ontology#>
	PREFIX dbpedia: <http://dbpedia.org/resource/>
	PREFIX dbprop: <http://dbpedia.org/property/>
	PREFIX dbowl: <http://dbpedia.org/ontology/>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
	PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
	PREFIX factbook: <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
	PREFIX mo: <http://purl.org/ontology/mo/>
	PREFIX dc: <http://purl.org/dc/elements/1.1/>
	SELECT * WHERE {
	?x skos:subject <http://dbpedia.org/resource/Category:FIFA_World_Cup-winning_countries> .
	?p dbowl:managerClub ?x .
	?p foaf:name "Luiz Felipe Scolari" .
	}

LD 10	PREFIX dbpedia: <http://dbpedia.org/resource/>
	PREFIX dbprop: <http://dbpedia.org/property/>
	PREFIX dbowl: <http://dbpedia.org/ontology/>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
	PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
	PREFIX factbook: <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
	SELECT * WHERE {
	?n skos:subject
	<http://dbpedia.org/resource/Category:Chancellors_of_Germany> .
	?n owl:sameAs ?p2 .
	?p2 <http://data.nytimes.com/elements/latest_use> ?u .
	}

LD 11	PREFIX geo-ont: <http://www.geonames.org/ontology#>
	PREFIX dbpedia: <http://dbpedia.org/resource/>
	PREFIX dbprop: <http://dbpedia.org/property/>
	PREFIX dbowl: <http://dbpedia.org/ontology/>
	PREFIX foaf: <http://xmlns.com/foaf/0.1/>
	PREFIX owl: <http://www.w3.org/2002/07/owl#>
	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
	PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
	PREFIX factbook: <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
	PREFIX mo: <http://purl.org/ontology/mo/>
	PREFIX dc: <http://purl.org/dc/elements/1.1/>
	SELECT * WHERE {
	?x dbowl:team dbpedia:Eintracht_Frankfurt .
	?x rdfs:label ?y .
	?x dbowl:birthDate ?d .
	?x dbowl:birthPlace ?p .
	?p rdfs:label ?l .
	}

EK 2: Detaylı Deney Sonuçları

A. Splendid-Hibiscus Biricik Yüklem Index Dominant Deney Sonuçları

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	860	298	461	403	546	8863	2399
Çalışma Süresi 2	688	289	547	465	586	8728	2368
Çalışma Süresi 3	709	333	464	453	366	8945	2407
Çalışma Süresi 4	790	376	463	498	381	8660	2362
Çalışma Süresi 5	669	366	392	404	384	8750	2440
Konfigürasyon Süresi 1	2240	2245	2278	2218	2249	2169	2224
Konfigürasyon Süresi 2	2178	4323	2134	2231	2189	2293	2184
Konfigürasyon Süresi 3	2178	2260	2290	2205	2355	2285	2159
Konfigürasyon Süresi 4	2214	2220	2250	2289	2239	2203	2176
Konfigürasyon Süresi 5	2199	2117	2143	2273	2194	2200	2139
Eleme Süresi 1	336	55	75	158	56	59	77
Eleme Süresi 2	261	58	82	119	186	69	56
Eleme Süresi 3	281	52	71	142	56	67	87
Eleme Süresi 4	334	68	73	184	72	62	74
Eleme Süresi 5	236	46	60	110	72	69	68

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	472	1234	5969	320	14040	1959	
Çalışma Süresi 2	451	911	6195	322	13881	1955	
Çalışma Süresi 3	464	914	6052	464	14026	1905	
Çalışma Süresi 4	487	896	6033	419	14055	2022	
Çalışma Süresi 5	469	948	5979	399	14086	2017	
Konfigürasyon Süresi 1	2174	2251	2247	2249	2190	2218	
Konfigürasyon Süresi 2	2181	2208	2172	2140	2320	2121	
Konfigürasyon Süresi 3	2214	2287	2190	2176	2121	2356	
Konfigürasyon Süresi 4	2130	2322	2199	2218	2160	2124	
Konfigürasyon Süresi 5	2132	2265	2219	2274	2171	2195	
Eleme Süresi 1	24	242	67	32	89	71	
Eleme Süresi 2	25	179	93	28	74	65	
Eleme Süresi 3	18	205	92	36	75	70	
Eleme Süresi 4	19	167	150	44	79	80	
Eleme Süresi 5	89	269	91	31	104	71	

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD 10	LD 111
Çalışma Süresi 1	368	364	366	236	293	553	862	104	258	377	520
Çalışma Süresi 2	295	260	355	354	258	510	845	953	223	307	426
Çalışma Süresi 3	399	300	303	213	270	589	855	896	233	362	455
Çalışma Süresi 4	392	372	429	290	206	507	853	100	246	435	448
Çalışma Süresi 5	401	326	361	231	272	611	839	104	240	356	548
Konfigürasyon Süresi 1	2130	2310	2116	2339	2312	2328	2219	2199	2233	2206	2179
Konfigürasyon Süresi 2	2359	2204	2304	2144	2192	2210	2370	2157	2269	2289	2358
Konfigürasyon Süresi 3	2172	2112	2206	2262	2142	2142	2424	2356	2255	2162	2201
Konfigürasyon Süresi 4	2176	2157	2124	2209	2257	2299	2154	2288	2153	2280	2130
Konfigürasyon Süresi 5	2209	2233	2226	2282	2366	2180	2384	2274	2291	2164	2172
Eleme Süresi 1	54	43	65	37	32	64	34	98	95	62	76
Eleme Süresi 2	43	24	68	30	41	69	53	61	40	63	56
Eleme Süresi 3	67	25	54	22	38	92	34	57	31	66	66
Eleme Süresi 4	45	23	77	33	33	55	36	76	41	94	59
Eleme Süresi 5	103	77	54	29	43	131	21	79	45	59	67

B. Splendid-Hibiscus Index Dominant Orijinal Hibiscus Deney Sonuçları

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	808	358	420	428	422	9130	2449
Çalışma Süresi 2	787	321	435	351	473	9151	2500
Çalışma Süresi 3	710	395	557	414	466	8786	2380
Çalışma Süresi 4	804	326	469	477	337	8912	2482
Çalışma Süresi 5	702	348	452	453	394	8894	2410
Konfigürasyon Süresi 1	2126	2095	2273	2153	2312	2232	2135
Konfigürasyon Süresi 2	2288	2232	2258	2304	2194	2176	2189
Konfigürasyon Süresi 3	2188	2128	2147	2250	2220	2245	2284
Konfigürasyon Süresi 4	2196	2359	2139	2158	2315	2164	2158
Konfigürasyon Süresi 5	2173	2278	2244	2313	2240	2362	2223
Eleme Süresi 1	255	65	80	140	77	102	68
Eleme Süresi 2	276	56	79	111	82	67	77
Eleme Süresi 3	279	64	142	169	80	80	58
Eleme Süresi 4	370	60	81	176	64	61	73
Eleme Süresi 5	251	74	85	215	93	56	84

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	471	1072	6372	379	16115	1943	
Çalışma Süresi 2	505	1081	6104	386	15984	1890	
Çalışma Süresi 3	458	941	6204	428	15838	1996	
Çalışma Süresi 4	451	1065	6072	457	16000	2011	
Çalışma Süresi 5	392	925	5971	374	16165	2082	
Konfigürasyon Süresi 1	2130	2171	2147	2241	2311	2298	
Konfigürasyon Süresi 2	2215	2151	2208	2093	2470	2180	
Konfigürasyon Süresi 3	2183	2084	2379	2220	2176	2121	
Konfigürasyon Süresi 4	2226	2141	2146	2145	2189	2183	
Konfigürasyon Süresi 5	2291	2193	2253	2115	2203	2094	
Eleme Süresi 1	20	306	84	31	78	66	
Eleme Süresi 2	68	259	91	30	56	56	
Eleme Süresi 3	20	233	119	101	65	74	
Eleme Süresi 4	22	283	91	34	75	69	
Eleme Süresi 5	12	248	81	37	62	65	

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD111
Çalışma Süresi 1	389	272	406	224	263	729	839	959	213	394	5867
Çalışma Süresi 2	372	337	368	288	233	774	847	941	241	289	5730
Çalışma Süresi 3	362	367	314	295	207	778	855	936	238	366	5947
Çalışma Süresi 4	319	283	431	224	305	707	866	926	242	459	5792
Çalışma Süresi 5	377	384	360	256	248	725	850	103	275	398	5811
Konfigürasyon Süresi 1	2194	2221	2187	2107	2359	2225	2333	2187	2239	2323	2277
Konfigürasyon Süresi 2	2127	2224	2156	2225	2309	2205	2261	2163	2208	2261	2260
Konfigürasyon Süresi 3	2294	2192	2249	2243	2185	2230	2368	2211	2204	2130	2142
Konfigürasyon Süresi 4	2199	2128	2135	2199	2253	2237	2228	2205	2272	2146	2122
Konfigürasyon Süresi 5	2160	2171	2160	2116	2106	2116	2221	2173	2267	2198	2068
Eleme Süresi 1	53	26	132	24	33	97	33	106	39	58	128
Eleme Süresi 2	61	51	82	36	46	106	93	86	45	47	115
Eleme Süresi 3	69	75	74	53	44	127	40	86	50	61	103
Eleme Süresi 4	61	22	88	29	64	93	32	82	46	57	127
Eleme Süresi 5	59	23	75	27	39	112	34	86	34	59	146

C. Splendid-Hibiscus ASK Dominant Biricik Yüklem

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	313	352	861	742	569	13908	2529
Çalışma Süresi 2	507	194	369	261	302	8746	2462
Çalışma Süresi 3	504	222	351	291	311	8667	2265
Çalışma Süresi 4	260	260	375	270	325	8540	2307
Çalışma Süresi 5	305	238	382	318	329	8669	2176
Konfigürasyon Süresi 1	3478	3557	3559	3448	3389	3410	3422
Konfigürasyon Süresi 2	4059	3567	3311	3509	3595	3470	3492
Konfigürasyon Süresi 3	3572	3472	3478	3495	3423	3326	3436
Konfigürasyon Süresi 4	3571	3489	3377	3366	3477	3541	3407
Konfigürasyon Süresi 5	3497	3406	3465	3529	3386	3402	3466
Eleme Süresi 1	106	8	34	36	29	18	32
Eleme Süresi 2	225	11	28	33	28	33	18
Eleme Süresi 3	243	9	24	41	25	29	20
Eleme Süresi 4	38	15	24	56	29	29	29
Eleme Süresi 5	83	14	26	53	23	30	16

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	499	884	8493	257	14017	2111	
Çalışma Süresi 2	364	829	6059	246	13660	1768	
Çalışma Süresi 3	484	589	5970	323	13645	1766	
Çalışma Süresi 4	519	734	5864	268	13676	1930	
Çalışma Süresi 5	363	762	5852	315	13638	1787	
Konfigürasyon Süresi 1	3357	3530	3606	3543	3460	3424	
Konfigürasyon Süresi 2	3361	3502	3600	3422	3323	3463	
Konfigürasyon Süresi 3	3419	3451	3483	3386	3640	3382	
Konfigürasyon Süresi 4	3534	3405	3454	3388	3424	3486	
Konfigürasyon Süresi 5	3485	3537	3364	3627	3463	3433	
Eleme Süresi 1	9	39	26	18	26	24	
Eleme Süresi 2	12	197	25	10	28	27	
Eleme Süresi 3	12	73	22	15	35	27	
Eleme Süresi 4	8	162	25	13	33	30	
Eleme Süresi 5	9	147	20	19	19	23	

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD11
Çalışma Süresi 1	410	300	302	275	201	573	847	242	335	404	309
Çalışma Süresi 2	282	256	296	249	230	895	814	805	240	274	402
Çalışma Süresi 3	299	309	302	204	260	523	829	790	209	291	405
Çalışma Süresi 4	359	277	301	205	256	668	824	LD	199	291	407
Çalışma Süresi 5	309	301	296	206	231	744	829	869	209	230	438
Konfigürasyon Süresi 1	3494	3509	3645	3467	3522	3542	3388	3367	3441	3460	3544
Konfigürasyon Süresi 2	3436	3433	3384	3563	3462	3419	3480	3399	3508	3369	3418
Konfigürasyon Süresi 3	3430	3588	3379	3489	3443	3297	3408	3341	3453	3463	3414
Konfigürasyon Süresi 4	3415	3598	3566	3417	3486	3660	3387	3357	3376	3507	3458
Konfigürasyon Süresi 5	3359	3437	3471	3522	3518	3397	3424	3369	3461	3393	3406
Eleme Süresi 1	18	11	27	14	23	31	18	25	11	32	23
Eleme Süresi 2	23	13	36	14	15	43	13	23	14	27	24
Eleme Süresi 3	18	9	35	14	21	34	18	20	10	27	40
Eleme Süresi 4	23	9	24	17	19	57	14	27	15	32	36
Eleme Süresi 5	25	14	24	14	12	31	17	29	8	25	68

D. Splendid-Hibiscus ASK Dominant Orijinal Hibiscus

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	365	228	357	277	285	9115	2368
Çalışma Süresi 2	342	214	331	353	353	8723	2603
Çalışma Süresi 3	405	247	331	275	344	8600	2365
Çalışma Süresi 4	276	221	376	325	257	8842	2280
Çalışma Süresi 5	433	228	318	290	265	8728	2298
Konfigürasyon Süresi 1	3469	3341	3325	3656	3385	3658	3382
Konfigürasyon Süresi 2	3432	3512	3473	3485	3452	3456	3433
Konfigürasyon Süresi 3	3450	3442	3483	3453	3375	3410	3397
Konfigürasyon Süresi 4	3496	3577	3601	3472	3508	3347	3400
Konfigürasyon Süresi 5	3513	3482	3512	3483	3370	3438	3483
Eleme Süresi 1	120	8	27	58	34	29	32
Eleme Süresi 2	99	8	25	42	35	20	34
Eleme Süresi 3	185	13	29	43	32	22	21
Eleme Süresi 4	58	14	38	43	26	15	35
Eleme Süresi 5	167	13	24	31	30	20	39

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	464	667	5853	270	16127	1748	
Çalışma Süresi 2	369	579	5823	272	16435	1855	
Çalışma Süresi 3	386	787	6713	273	16174	1744	
Çalışma Süresi 4	402	755	6543	241	16013	1847	
Çalışma Süresi 5	372	155	5875	297	15997	2038	
Konfigürasyon Süresi 1	3390	3349	3438	3430	3478	3531	
Konfigürasyon Süresi 2	3500	3506	3443	3554	3508	3361	
Konfigürasyon Süresi 3	3383	3536	3460	3360	3486	3504	
Konfigürasyon Süresi 4	3462	3521	3402	3492	3417	3486	
Konfigürasyon Süresi 5	3471	3515	3491	3435	3622	3473	
Eleme Süresi 1	8	54	34	17	28	20	
Eleme Süresi 2	8	62	24	19	28	17	
Eleme Süresi 3	8	197	32	17	15	23	
Eleme Süresi 4	11	155	36	18	28	21	
Eleme Süresi 5	8	80	28	15	14	19	

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD11
Çalışma Süresi 1	395	268	305	326	281	778	844	855	226	261	564
Çalışma Süresi 2	271	245	298	277	230	542	835	927	210	281	558
Çalışma Süresi 3	337	334	308	235	221	592	830	970	222	294	553
Çalışma Süresi 4	407	302	336	254	200	630	841	104	238	285	555
Çalışma Süresi 5	270	287	306	251	217	34	843	100	185	244	560
Konfigürasyon Süresi 1	3514	3362	3534	3580	3495	3486	3318	3369	3312	3630	3549
Konfigürasyon Süresi 2	3334	3333	3470	3495	3538	3362	3416	3492	3423	3389	3389
Konfigürasyon Süresi 3	3370	3536	3447	3489	3467	3461	3400	3542	3455	3482	3405
Konfigürasyon Süresi 4	3507	3501	3459	3343	3487	3467	3462	3433	3460	3478	3605
Konfigürasyon Süresi 5	3413	3401	3442	3506	3341	3495	3491	3340	3462	3492	3547
Eleme Süresi 1	19	11	29	16	13	29	11	24	15	25	32
Eleme Süresi 2	17	13	37	12	14	38	14	26	16	18	35
Eleme Süresi 3	19	9	44	10	16	35	13	25	15	28	33
Eleme Süresi 4	18	13	31	19	13	34	20	40	8	22	32
Eleme Süresi 5	16	12	32	14	17	40	16	21	9	24	51

E. FedX-Hibiscus Index Dominant Biricik Yüklem

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	631	317	535	644	470	1110	925
Çalışma Süresi 2	783	322	451	613	421	1046	786
Çalışma Süresi 3	593	327	604	587	245	1154	785
Çalışma Süresi 4	768	327	590	724	352	1169	853
Çalışma Süresi 5	644	374	425	598	482	1099	958
Konfigürasyon Süresi 1	2058	1982	2157	2015	2063	2001	2083
Konfigürasyon Süresi 2	2027	2090	2140	2234	2072	2085	2195
Konfigürasyon Süresi 3	2092	1993	1943	2037	2114	2021	2081
Konfigürasyon Süresi 4	2017	2067	1993	2052	2046	2148	2010
Konfigürasyon Süresi 5	2059	1949	2188	2010	2140	2101	2112
Eleme Süresi 1	279	78	98	192	85	79	87
Eleme Süresi 2	387	83	96	198	84	94	79
Eleme Süresi 3	269	86	135	170	61	94	75
Eleme Süresi 4	365	64	110	243	61	94	84
Eleme Süresi 5	323	83	85	173	91	97	89

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	605	779	3973	259	2143		2256
Çalışma Süresi 2	545	681	4180	310	2379		2169
Çalışma Süresi 3	547	887	4132	338	2118		1931
Çalışma Süresi 4	504	677	4059	270	2267		2082
Çalışma Süresi 5	523	738	4073	341	2237		2186
Konfigürasyon Süresi 1	2111	2115	2043	2045	2021		1884
Konfigürasyon Süresi 2	2070	2022	4031	2309	1985		2058
Konfigürasyon Süresi 3	2045	2064	1968	2275	2041		2089
Konfigürasyon Süresi 4	2047	2111	2125	2001	2106		2031
Konfigürasyon Süresi 5	2088	2127	2072	1959	2136		2031
Eleme Süresi 1	36	309	115	44	87		93
Eleme Süresi 2	37	239	2081	44	127		84
Eleme Süresi 3	30	305	119	41	75		66
Eleme Süresi 4	24	247	101	36	108		91
Eleme Süresi 5	25	270	116	55	110		87

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD11
Çalışma Süresi 1	420	333	362	271	328	409	118	699	227	494	539
Çalışma Süresi 2	413	373	413	277	319	452	906	679	253	479	541
Çalışma Süresi 3	399	324	396	299	297	403	105	607	245	495	681
Çalışma Süresi 4	423	334	387	303	278	448	953	616	247	470	576
Çalışma Süresi 5	416	455	415	254	341	506	953	648	283	609	631
Konfigürasyon Süresi 1	2019	2070	2089	2106	2118	2053	2019	1988	2082	2145	2106
Konfigürasyon Süresi 2	2019	2226	2042	2064	2023	2085	2092	1947	2033	2082	2132
Konfigürasyon Süresi 3	2170	2158	2098	2130	2063	2089	2116	2085	2084	2002	2082
Konfigürasyon Süresi 4	2019	2050	2060	2134	2035	2117	2066	2151	2295	2088	2075
Konfigürasyon Süresi 5	2133	2043	2126	2057	2071	2246	2066	2083	2115	2104	2093
Eleme Süresi 1	67	38	69	43	77	76	49	113	47	75	84
Eleme Süresi 2	65	34	95	33	56	66	39	116	59	71	93
Eleme Süresi 3	49	36	87	42	64	70	39	99	58	83	95
Eleme Süresi 4	69	28	94	45	53	39	48	100	53	77	99
Eleme Süresi 5	63	40	98	37	90	67	48	99	59	89	89

F. FedX-Hibiscus Index Dominant Orijinal Hibiscus

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	1571	415	2704	927	643	1103	952
Çalışma Süresi 2	685	324	549	606	405	1066	836
Çalışma Süresi 3	700	400	545	645	452	1096	900
Çalışma Süresi 4	623	374	422	604	430	1057	993
Çalışma Süresi 5	302	346	504	591	422	1176	891
Konfigürasyon Süresi 1	2477	2003	3541	2186	1968	2030	2048
Konfigürasyon Süresi 2	2111	2031	2048	2120	2065	2193	2102
Konfigürasyon Süresi 3	2203	2032	2112	2034	2070	2109	2139
Konfigürasyon Süresi 4	2190	2149	2141	2029	2165	2086	2016
Konfigürasyon Süresi 5	2077	2155	2025	2119	2147	2102	2132
Eleme Süresi 1	241	85	167	177	100	94	103
Eleme Süresi 2	384	72	144	188	85	78	86
Eleme Süresi 3	297	121	96	206	98	97	112
Eleme Süresi 4	281	89	80	188	99	70	124
Eleme Süresi 5	612	77	136	176	82	146	112

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	588	701	6429	433	2630		1893
Çalışma Süresi 2	563	669	4174	261	2161		1995
Çalışma Süresi 3	531	676	4212	267	2155		2101
Çalışma Süresi 4	539	896	4228	385	2071		1972
Çalışma Süresi 5	598	711	4071	286	2269		1992
Konfigürasyon Süresi 1	2108	2099	2060	2028	2022		2201
Konfigürasyon Süresi 2	2106	2091	2020	2055	2072		2158
Konfigürasyon Süresi 3	2314	2129	2108	2119	2089		2047
Konfigürasyon Süresi 4	2236	2030	2133	2009	2175		2088
Konfigürasyon Süresi 5	2108	2052	2094	2154	2031		2140
Eleme Süresi 1	31	241	88	49	64		60
Eleme Süresi 2	39	255	103	45	77		64
Eleme Süresi 3	38	233	108	48	78		67
Eleme Süresi 4	39	355	118	51	79		65
Eleme Süresi 5	39	269	84	46	93		73

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD11
Çalışma Süresi 1	517	377	394	289	538	791	125	629	404	529	165
Çalışma Süresi 2	442	404	414	306	223	443	903	662	280	460	146
Çalışma Süresi 3	503	334	426	297	276	486	999	691	253	450	141
Çalışma Süresi 4	409	449	440	276	256	550	104	675	238	556	148
Çalışma Süresi 5	405	292	423	261	242	451	955	611	244	578	139
Konfigürasyon Süresi 1	2061	2157	2100	2012	2090	2137	2119	2062	2387	2029	1982
Konfigürasyon Süresi 2	2140	2082	2141	2223	2145	2036	2161	2071	2138	2059	2067
Konfigürasyon Süresi 3	2091	2113	2074	2042	2150	2065	2038	2194	2033	2111	2102
Konfigürasyon Süresi 4	2065	2057	2044	2075	2059	2097	2035	2104	2043	2063	2078
Konfigürasyon Süresi 5	2080	2136	2006	2075	2156	2156	1937	2119	2101	2011	2167
Eleme Süresi 1	87	38	92	46	44	119	54	103	46	85	147
Eleme Süresi 2	82	39	97	44	40	105	39	100	54	53	114
Eleme Süresi 3	85	38	101	47	59	111	48	116	45	65	126
Eleme Süresi 4	70	37	114	36	41	131	47	102	47	82	127
Eleme Süresi 5	62	28	122	41	53	94	48	91	45	83	114

G. FedX-Hibiscus ASK Dominant Biricik Yüklem

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	1293	198	1147	681	503	841	739
Çalışma Süresi 2	291	202	311	383	180	868	600
Çalışma Süresi 3	277	188	312	378	264	923	631
Çalışma Süresi 4	298	181	297	357	303	843	657
Çalışma Süresi 5	316	182	314	393	267	813	637
Konfigürasyon Süresi 1	3969	3712	3742	3662	3887	3770	3845
Konfigürasyon Süresi 2	3740	3822	3870	3839	3695	3683	3827
Konfigürasyon Süresi 3	3727	3672	3875	3720	3751	3833	3723
Konfigürasyon Süresi 4	3826	3781	3871	3758	3651	3684	3871
Konfigürasyon Süresi 5	3755	3799	3733	3668	3875	3833	3736
Eleme Süresi 1	56	14	27	43	26	30	28
Eleme Süresi 2	54	16	36	55	25	25	23
Eleme Süresi 3	62	16	40	42	26	24	37
Eleme Süresi 4	58	12	31	34	34	23	36
Eleme Süresi 5	72	12	32	64	27	27	34

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	483	381	3902	386	1941		1843
Çalışma Süresi 2	384	393	3784	203	1938		1754
Çalışma Süresi 3	386	398	3877	192	2193		1800
Çalışma Süresi 4	380	400	3898	187	1919		1826
Çalışma Süresi 5	388	374	3907	190	1959		1886
Konfigürasyon Süresi 1	3757	3813	3845	4696	3634		3761
Konfigürasyon Süresi 2	3881	3655	3702	3857	3769		3700
Konfigürasyon Süresi 3	3786	3670	3812	3650	3855		3780
Konfigürasyon Süresi 4	3660	3796	3812	3849	3735		3709
Konfigürasyon Süresi 5	3768	3767	3728	3664	3830		3816
Eleme Süresi 1	16	66	38	30	27		30
Eleme Süresi 2	15	67	31	18	28		23
Eleme Süresi 3	19	64	33	16	34		23
Eleme Süresi 4	14	56	42	15	31		30
Eleme Süresi 5	18	59	30	15	36		27

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD11
Çalışma Süresi 1	351	240	293	223	185	283	985	488	317	444	384
Çalışma Süresi 2	283	222	252	195	172	322	697	442	162	328	381
Çalışma Süresi 3	282	231	252	187	177	300	712	559	175	346	389
Çalışma Süresi 4	278	221	254	186	193	297	731	453	158	365	399
Çalışma Süresi 5	268	224	246	181	185	296	734	448	158	348	383
Konfigürasyon Süresi 1	3801	3800	3767	3703	3746	3690	3763	3709	3700	3832	3698
Konfigürasyon Süresi 2	3723	3839	3657	3722	3653	3741	3864	3820	3722	3686	3961
Konfigürasyon Süresi 3	3816	3700	3779	3692	3831	3697	3749	3807	3893	3728	3764
Konfigürasyon Süresi 4	3678	3705	3904	3696	3720	3886	3746	3840	3697	3802	3762
Konfigürasyon Süresi 5	3751	3688	3636	3688	3727	3843	3726	3748	3678	3753	3777
Eleme Süresi 1	20	15	36	19	19	30	15	41	15	28	34
Eleme Süresi 2	20	12	25	17	19	44	15	33	14	32	27
Eleme Süresi 3	27	16	29	15	20	45	20	33	12	30	28
Eleme Süresi 4	19	12	31	13	22	30	15	35	12	31	30
Eleme Süresi 5	20	13	25	13	22	33	15	35	12	24	34

H. FedX-Hibiscus ASK Dominant Orijinal Hibiscus

Cross Domain

	CD1	CD2	CD3	CD4	CD5	CD6	CD7
Çalışma Süresi 1	331	192	292	446	277	915	641
Çalışma Süresi 2	278	205	305	388	276	834	688
Çalışma Süresi 3	323	184	326	472	347	857	704
Çalışma Süresi 4	292	185	306	390	305	883	628
Çalışma Süresi 5	307	186	301	384	271	839	598
Konfigürasyon Süresi 1	3763	3777	3810	3685	3912	3893	3749
Konfigürasyon Süresi 2	3731	3976	3747	3930	3797	3775	3758
Konfigürasyon Süresi 3	3765	3826	3706	3743	3763	3782	3801
Konfigürasyon Süresi 4	3774	3853	3967	3825	3795	3743	3836
Konfigürasyon Süresi 5	3869	3849	3732	3714	3686	3774	3700
Eleme Süresi 1	74	16	31	49	28	31	34
Eleme Süresi 2	55	16	29	59	24	27	26
Eleme Süresi 3	73	12	27	71	32	25	27
Eleme Süresi 4	71	12	33	60	23	26	28
Eleme Süresi 5	81	12	37	46	26	26	28

Live Science

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Çalışma Süresi 1	410	409	3958	196	1975		1818
Çalışma Süresi 2	384	542	3991	199	1915		1824
Çalışma Süresi 3	383	397	3963	196	1951		1768
Çalışma Süresi 4	389	389	3975	225	1918		1788
Çalışma Süresi 5	431	415	3972	193	2163		1774
Konfigürasyon Süresi 1	3907	3872	3846	3798	3798		3870
Konfigürasyon Süresi 2	3747	3800	3761	4916	3778		3753
Konfigürasyon Süresi 3	3756	3945	3781	3855	3734		3722
Konfigürasyon Süresi 4	3833	3929	3692	3745	3828		3953
Konfigürasyon Süresi 5	3919	3901	3883	3947	3768		3795
Eleme Süresi 1	17	75	30	17	28		20
Eleme Süresi 2	16	107	28	14	24		25
Eleme Süresi 3	14	67	29	16	26		20
Eleme Süresi 4	16	64	35	19	26		20
Eleme Süresi 5	19	67	31	15	25		23

Linked Data

	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	LD 8	LD 9	LD10	LD11
Çalışma Süresi 1	285	232	262	194	240	304	767	509	170	342	134
Çalışma Süresi 2	295	237	269	188	177	304	722	453	162	343	123
Çalışma Süresi 3	290	233	275	187	183	328	721	461	174	372	128
Çalışma Süresi 4	305	225	286	191	173	337	736	453	161	331	121
Çalışma Süresi 5	283	225	308	187	177	311	736	471	160	382	147
Konfigürasyon Süresi 1	3765	3830	3810	3792	3749	3666	3822	3686	3885	3762	3832
Konfigürasyon Süresi 2	3862	4006	3772	3761	3785	3802	3749	3852	3920	3760	3956
Konfigürasyon Süresi 3	3801	3740	3817	3761	3915	3753	3938	3757	3818	3810	3986
Konfigürasyon Süresi 4	3747	3745	3746	3871	3764	3698	3689	3754	3772	3882	3741
Konfigürasyon Süresi 5	3682	3801	3739	3680	3756	3779	3926	3787	3855	3736	3940
Eleme Süresi 1	22	12	32	16	23	31	17	34	12	32	46
Eleme Süresi 2	25	12	30	14	19	30	15	42	13	22	33
Eleme Süresi 3	24	14	44	12	22	39	16	34	15	25	34
Eleme Süresi 4	29	15	33	14	18	34	22	32	12	21	39
Eleme Süresi 5	22	12	41	13	17	40	18	37	12	28	40

EK 3: Veri Kaynağı Eleme Sonuçları

A. Splendid-Hibiscus ASK Dominant Veri Kaynağı Eleme

Sorgu	İlk Seçilen Kaynak	Biricik Yüklem Elenen Kaynak	Biricik Yüklem Sonrası Hibiscus Elenen	Toplam ASK	Orijinal Hibiscus
CD1	11	0	7	0	7
CD2	3	0	0	0	0
CD3	12	7	0	0	7
CD4	20	13	2	0	15
CD5	11	7	0	0	7
CD6	10	0	2	0	2
CD7	13	7	0	0	7
LS1	1	0	0	0	0
LS2	11	0	4	0	4
LS3	12	4	3	0	7
LS4	7	0	0	0	0
LS5	10	3	0	0	2
LS6	9	0	2	0	2
LS7					
LD1	11	8	0	0	8
LD2	3	0	0	0	0
LD3	19	15	0	0	15
LD4	5	0	0	0	0
LD5	5	2	0	0	2
LD6	14	7	2	0	7
LD7	4	0	0	0	0
LD8	15	7	3	0	10
LD9	3	0	0	0	0
LD10	10	0	7	0	7
LD11	21	16	0	0	14

B. Splendid-Hibiscus Index Dominant Veri Kaynağı Eleme

Sorgu	İlk Seçilen Kaynak	Biricik Yüklem Elenen Kaynak	Biricik Yüklem Sonrası Hibiscus Elenen	Toplam ASK	Orijinal Hibiscus
CD1	16	0	4	0	4
CD2	7	4	0	0	4
CD3	12	7	0	0	7
CD4	20	15	0	0	15
CD5	11	7	0	0	7
CD6	10	0	2	0	2
CD7	13	1	6	0	7
LS1	1	0	0	0	0
LS2	11	0	4	0	4
LS3	12	4	3	0	7
LS4	7	0	0	0	0
LS5	10	3	0	0	2
LS6	9	0	2	0	2
LS7					
LD1	11	8	0	0	8
LD2	3	0	0	0	0
LD3	19	15	0	0	15
LD4	5	0	0	0	0
LD5	5	2	0	0	2
LD6	14	7	5	0	7
LD7	4	0	0	0	0
LD8	15	7	3	0	10
LD9	5	2	0	0	2
LD10	10	0	7	0	7
LD11	21	16	0	0	14

C. FedX-Hibiscus ASK Dominant Veri Kaynağı Eleme

Sorgu	İlk Seçilen Kaynak	Biricik Yüklem Elenen Kaynak	Biricik Yüklem Sonrası Hibiscus Elenen	Toplam ASK	Orijinal Hibiscus
CD1	11	0	7	0	7
CD2	3	0	0	0	0
CD3	12	7	0	0	7
CD4	20	13	2	0	15
CD5	11	7	0	0	7
CD6	10	0	2	0	2
CD7	13	7	0	0	7
LS1	1	0	0	0	0
LS2	11	0	4	0	4
LS3	12	4	3	0	7
LS4	7	0	0	0	0
LS5	10	3	0	0	2
LS6					
LS7	6	1	0	0	0
LD1	11	8	0	0	8
LD2	3	0	0	0	0
LD3	19	15	0	0	15
LD4	5	0	0	0	0
LD5	5	2	0	0	2
LD6	14	7	2	0	6
LD7	4	0	0	0	0
LD8	15	4	6	0	10
LD9	3	0	0	0	0
LD10	10	0	7	0	7
LD11	21	16	0	0	14

D. FedX-Hibiscus Index Dominant Veri Kaynağı Eleme

Sorgu	İlk Seçilen Kaynak	Biricik Yüklem Elenen Kaynak	Biricik Yüklem Sonrası Hibiscus Elenen	Toplam ASK	Orijinal Hibiscus
CD1	16	0	4	0	4
CD2	7	4	0	0	4
CD3	12	7	0	0	7
CD4	20	13	2	0	15
CD5	11	7	0	0	7
CD6	10	0	2	0	2
CD7	13	7	0	0	7
LS1	1	0	0	0	0
LS2	11	0	4	0	4
LS3	12	4	3	0	7
LS4	7	0	0	0	0
LS5	10	3	0	0	2
LS6					
LS7	6	1	0	0	0
LD1	11	8	0	0	8
LD2	3	0	0	0	0
LD3	19	15	0	0	15
LD4	5	0	0	0	0
LD5	5	2	0	0	2
LD6	14	7	2	0	7
LD7	4	0	0	0	0
LD8	15	4	6	0	10
LD9	5	2	0	0	2
LD10	10	0	7	0	7
LD11	21	16	0	0	14

EK 4: Çalışma, Konfigürasyon ve Veri Kaynağı Eleme Süresi Ortalamaları (ms)

A. Splendid-Hibiscus ASK Dominant Çalışma, Konfigürasyon ve Veri Kaynağı Eleme Süresi Ortalamaları (ms)

Sorgu	Çalışma Süresi Ortalaması	Konfigürasyon Süresi Ortalaması	Eleme Süresi Ortalama	Orijinal Hibiscus Çalışma Süresi Ortalaması	Orijinal Hibiscus Konfigürasyon Süresi Ortalaması	Orijinal Hibisus Eleme Süresi Ortalaması
CD1	374	3547	138	371	3472	129
CD2	240	3506	11	226	3479	11
CD3	375	3440	26	340	3489	27
CD4	293	3484	43	297	3480	43
CD5	322	3430	27	298	3404	32
CD6	8694	3427	29	8764	3435	21
CD7	2345	3441	22	2344	3410	34
LS1	449	3422	10	387	3441	8
LS2	775	3494	127	667	3514	99
LS3	5964	3512	24	6090	3447	31
LS4	280	3451	15	272	3452	17
LS5	13660	3449	29	16105	3491	24
LS6	1828	3440	26	1817	3488	20
LS7						
LD1	322	3427	21	334	3430	18
LD2	293	3511	11	286	3421	12
LD3	300	3474	29	306	3459	33
LD4	220	3493	14	261	3497	14
LD5	249	3489	18	223	3483	14
LD6	662	3453	36	588	3471	36
LD7	8277	3407	16	8403	3426	14
LD8	1223	3364	25	968	3431	25
LD9	219	3452	12	219	3446	13
LD10	285	3439	29	276	3484	24
LD11	417	3430	33	5578	3500	33

B. Splendid-Hibiscus Index Dominant Çalışma, Konfigürasyon ve Veri Kaynağı Eleme Süresi Ortalamaları(ms)

Sorgu	Çalışma Süresi Ortalaması	Konfigürasyon Süresi Ortalaması	Eleme Süresi Ortalama	Orijinal Hibiscus Çalışma Süresi Ortalaması	Orijinal Hibiscus Konfigürasyon Süresi Ortalaması	Orijinal Hibisus Eleme Süresi Ortalaması
CD1	729	2197	292	767	2186	270
CD2	332	2242	55	344	2213	63
CD3	463	2224	73	452	2216	82
CD4	441	2241	140	432	2237	162
CD5	437	2227	67	427	2257	80
CD6	8780	2229	66	8979	2218	69
CD7	2391	2173	73	2447	2190	73
LS1	468	2162	23	460	2208	21
LS2	924	2268	209	1026	2154	263
LS3	6021	2203	92	6127	2203	89
LS4	380	2214	33	398	2160	34
LS5	1404	2174	81	16033	2234	67
LS6	1977	2179	71	1983	2161	67
LS7						
LD1	386	2186	55	370	2184	60
LD2	330	2198	31	329	2195	33
LD3	361	2185	62	378	2168	82
LD4	252	2251	31	256	2180	31
LD5	267	2254	37	248	2249	43
LD6	551	2230	75	743	2220	105
LD7	8516	2324	35	8510	2274	36
LD8	1001	2254	72	945	2188	86
LD9	240	2252	42	240	2238	43
LD10	365	2217	64	386	2202	58
LD11	474	2184	64	5823	2175	123

C. FedX-Hibiscus ASK Dominant Çalışma, Konfigürasyon ve Veri Kaynağı Eleme Süresi Ortalamaları (ms)

Sorgu	Çalışma Süresi Ortalaması	Konfigürasyon Süresi Ortalaması	Eleme Süresi Ortalama	Orijinal Hibiscus Çalışma Süresi Ortalaması	Orijinal Hibiscus Konfigürasyon Süresi Ortalaması	Orijinal Hibisus Eleme Süresi Ortalaması
CD1	302	3774	59	307	3767	73
CD2	189	3764	14	188	3843	13
CD3	312	3828	33	304	3763	31
CD4	385	3715	47	408	3761	56
CD5	278	3774	26	286	3785	26
CD6	851	3762	25	860	3777	26
CD7	642	3803	33	652	3769	28
LS1	386	3770	16	394	3832	16
LS2	391	3744	63	407	3901	70
LS3	3892	3784	34	3970	3796	30
LS4	195	3790	16	197	3867	16
LS5	1946	3778	31	1948	3781	26
LS6						
LS7	1823	3750	27	1793	3806	21
LD1	281	3758	20	290	3771	24
LD2	226	3735	13	230	3792	13
LD3	253	3734	28	277	3776	35
LD4	189	3697	15	189	3771	14
LD5	182	3731	20	179	3768	20
LD6	298	3760	36	314	3743	35
LD7	726	3753	15	731	3832	17
LD8	463	3792	34	462	3766	35
LD9	165	3706	13	164	3853	12
LD10	353	3761	30	352	3777	25
LD11	385	3768	31	1286	3909	38

D. FedX-Hibiscus Index Dominant Çalışma, Konfigürasyon ve Veri Kaynağı Eleme Süresi Ortalamaları (ms)

Sorgu	Çalışma Süresi Ortalaması	Konfigürasyon Süresi Ortalaması	Eleme Süresi Ortalama	Orijinal Hibiscus Çalışma Süresi Ortalaması	Orijinal Hibiscus Konfigürasyon Süresi Ortalaması	Orijinal Hibisus Eleme Süresi Ortalaması
CD1	681	2048	322	669	2168	321
CD2	325	2014	81	373	2071	84
CD3	525	2097	101	533	2100	125
CD4	618	2035	188	618	2091	184
CD5	414	2083	77	435	2094	94
CD6	1121	2069	94	1088	2099	90
CD7	855	2092	83	914	2094	109
LS1	538	2068	30	563	2151	39
LS2	733	2097	274	696	2081	255
LS3	4088	2080	117	4205	2087	100
LS4	306	2107	43	313	2067	48
LS5	2216	2056	102	2195	2064	78
LS6						
LS7	2146	2040	87	1986	2129	65
LD1	416	2057	65	451	2079	79
LD2	347	2093	36	372	2110	38
LD3	399	2082	92	421	2073	104
LD4	282	2100	41	287	2064	44
LD5	315	2056	66	258	2128	46
LD6	436	2097	68	496	2100	112
LD7	988	2075	45	999	2064	48
LD8	648	2052	104	655	2098	102
LD9	248	2094	57	259	2094	46
LD10	489	2091	78	515	2050	77
LD11	583	2094	92	1457	2082	122

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZKAN, Ethem Cem

Uyruğu : T.C.

Doğum tarihi ve yeri : 27.10.1989 Ankara

Medeni hali : Bekar

Telefon : 0 (536) 971 62 72

e-mail : ethemcem.ozkan@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	TOBB ETÜ/Bilgisayar Mühendisliği	2012

İş Deneyimi

Yıl	Yer	Görev
2012-…	TÜBİTAK BİLGEM YTE	Konfigürasyon Yöneticisi

Yabancı Dil

İngilizce

Yayınlar

TURKRES 2012 2. Kalkınmada Bölgesel Dinamikler Sempozyumu 2012 Ekim

Semantik Web ile Yönetim Bilişim Sistemleri Verilerinin Etkin Sorgulanması

Doç. Dr. Erdoğan DOĞDU, Ethem Cem ÖZKAN, Ender ÖZMEN