Dokuz Eylül Üniversitesi-Mühendislik Fakültesi
Fen ve Mühendislik Dergisi
Cilt 19, Sayı 57, Eylül 2017

Dokuz Eylul University-Faculty of Engineering
Journal of Science and Engineering
Volume 19, Issue 57, September 2017

DOI: 10.21205/deufmd.2017195769

Çizge Veri Tabanı Kullanılarak Geliştirilen Yazılım Lisans Yönetimi Amaçlı Veri Görselleştirmesi Uygulaması: BigLogVis

Gizem Nur KARAGÖZ[1], Murat KOMESLİ[*2]

[1]Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, 06800, Ankara
[2]Yaşar Üniversitesi, Mühendislik Fakültesi, Yazılım Mühendisliği Bölümü, 35100, İzmir

(Alınış / Received: 20.11.2016, Kabul / Accepted: 31.05.2017,
Online Yayınlanma / Published Online: 20.09.2017)

Anahtar Kelimeler
Çizge Veri Tabanı, Veri Görselleştirme, Kayıt Dosyası, Lisans Takip

Özet: Çizge veri tabanları, verinin dinamik olarak saklanıp işlenmesine olanak sağlayan veri tabanı sistemleridir. Bu doğrultuda, sistemde esneklik gerektiren, aralarında çok sayıda ilişki bulunduran ve değişen yapıya sahip verilerin çizge veri tabanında tutulup işlenmesi fayda sağlamaktadır. Proje kapsamında görselleştirilmesi amaçlanan kayıt dosyalarının çizge veri tabanı üzerinde tutulması, verinin dinamik oluşu ve veri içinde tekrar eden ilişkilerin sayısı nedeniyle önem arz etmektedir. Çalışmanın amacı, saklanan bu kayıt dosyalarını, işlevi doğrultusunda, etkili bir şekilde kullanabilmek ve bu sayede karar destek sistemi olarak hizmet etmesini sağlamaktır. Böylece modüllerin lisans kullanım bilgilerini içeren bu kayıt dosyalarının görselleştirilmesi, gereksiz harcanmış olan lisans ücretlerini ortaya koymakta ve alınan kararların bu bağlamda maliyet etkin olmasını sağlamaktadır.

Data Visualization Application for Software License Management by Utilizing Graph Database: BigLogVis

Keywords
Graph Database, Data Visualization, Log File, License Tracking

Abstract: Graph databases are database systems that are useful for dynamic data storage and manipulation. Storing and manipulating such data that requires flexibility and includes many relations, provides advantages on systems. The data which is aimed to visualize within this study, include all of these properties such as dynamism and repeating relationships. Thus, using graph database on this study is essential. The aim of the study is to utilize the stored log files effectively as a decision support system. Through this, unnecessary purchased licences can be detected and reduced by the visualization of the log files that include the information of modules and its usage. Consequently, decisions that are taking through this system are becoming cost-effective.

*Sorumlu yazar: murat.komesli@yasar.edu.tr

1. Giriş

Türkiye'de yazılım ithalatı çok yüksek meblağlara ulaşmıştır. Öyle ki 2009 yılında tüm ülkelerden yapılan toplam yazılım ithalatı 110.860.601 ₺ olarak belirlenmiştir [1]. Bu oranın yüksekliği, satın alınan lisanslara etkin olarak karar verilememesiyle tetiklenmektedir. Alınan lisansların büyük bölümü aslında ihtiyaç olmadığı halde satın alınırken, diğer taraftan lisans miktarı yetersiz olan yazılımların da belirlenmesi güçleşmektedir. Tüm bu sorunların ortadan kalkması adına, lisans takip sistemlerinin, yazılım modülü kullanım bilgilerini düzenli olarak kayıt altına alması gerekmektedir. Fakat bu durum sonucunda sadece sürekli kaydedilen, çok sayıda büyük boyutta kayıt dosyalarının oluşmasına neden olmaktadır. Çözüm, karar vericilerin kolaylıkla kayıt dosyaları içindeki özet veriyi elde edebileceği görselleştirmeler sunmaktır.

Çizge, objeler (düğüm) ve ilişkilerden (ayrıt) oluşan grafiksel veri yapısıdır. Karmaşık veriler bu yapıyla görselleştirildiğinde daha anlaşılır sonuçlar çıkmaktadır. Bu yöntem uzun süredir kullanılmaktadır [2]. Çizge veri tabanı da aynı doğrultuda verinin bu yapı üzerinde saklandığı, tüm düğümlerin ve ayrıtların da özelliklerinin olabildiği veri tabanı sistemidir. İlişkisel veri tabanlarının aksine tablolar yer almamaktadır. Bu sayede, sorgular sırasında performansı en çok etkileyen tablolar arası geçiş işlemleri çizge veri tabanında ortadan kalkmıştır. Özellikle veri boyutu büyüdükçe, çizge veri tabanının performansı ilişkisel veri tabanına göre çok daha büyük bir ivmeyle artmaktadır [3].

Veri görselleştirilmesi, çok geniş bir kavram olup, birçok alt başlığı da beraberinde bulundurur [4]. Ayrıca, çıkarımlarla veri arasında bağ kurabilmek veri görselleştirme ile mümkün olmaktadır. Bu amaç doğrultusunda birçok görselleştirme tipi kullanılmaktadır. Örneğin grafikler, çizgeler, tablolar ve hatta haritalar dahi görselleştirme tipleri arasında sayılabilir [2]. Veri görselleştirme, verinin içindeki temel bilginin alınması amacıyla gerçekleştirilir. Özetlerde ve raporlarda verinin tamamı aktarılamayacağı için sadece bir bölümü sunulabilir. Fakat veri görselleştirme ile verinin tamamı hiçbir kayba uğramadan işlenebilir. Bu durum veri görselleştirmenin gücünü ve önemini kanıtlar niteliktedir.

Bu çalışmada, bir çizge veri tabanı olan Neo4j kullanarak büyük kullanıcı kayıt dosyalarındaki veriler yardımıyla lisans kullanımının görselleştirilmesi üzerinde çalışılmıştır. Ayrıca, çizge ve ilişkisel veri tabanları arasında karşılaştırma yapılmıştır. Çizge veri tabanı sorgulama dili Cypher ile ilişkisel veri tabanı sorgulama dili SQL arasındaki farklar/benzerlikler ortaya konulmuştur. Bunun neticesi olarak karar vericilerin daha maliyet-etkin kararlar almasını sağlayan bir uygulama yazılımı (BigLogVis) geliştirilmiştir. İkinci bölümde, bu konuda yapılan çalışmalar ve araştırmalarla ilgili geniş bilgi verilmektedir. Üçüncü bölümde, geliştirilen uygulama yazılımı açıklanmıştır. Son bölümde ise, konu ile ilgili karşılaştırmalar, sonuç ve önerilere yer verilmiştir.

2. İlgili Çalışmalar

Fry [5]'a göre veri görselleştirme yedi aşamadan oluşmaktadır.

Verinin toplanması: Bu aşama en önemlisidir. Çünkü doğru verinin toplanması ve örnek uzayın seçimi, sonuçlar için önem taşımaktadır. Veriye bağlı olarak bu süreç, kolay veya zor hale gelebilir. Veri toplanması aşamasının planlanması da projenin sonunu önemli ölçüde etkiler.

Veri yapısının belirlenmesi: Bu aşamada elde edilen verinin program tarafından

algılanabilir yapıları saptanmalıdır. Veri yapılarının hangi sıklıkla tekrar ettiği gibi bilgiler saptanıp omurga ortaya çıktıktan sonra veri, yazılım tarafından kullanılmaya hazır hale gelir. Buna göre yazılım geliştirilebilir.

Gereksiz verinin belirlenip, çıkarılması: Çalışmada toplanan verinin tamamının kullanılmasının gereksiz olacağı ve sonuçları olumsuz yönde etkileyeceğinden verilerin sadece ilgili kısmının kullanılması önem arz etmektedir. Örneğin, çalışmada veri olarak kullanılan kayıt dosyalarının oluşturulduğu sunucu bilgisayar bilgileri kayıtlarda yer almaktadır. Ancak, bu bilgiler görselleştirmeyi etkilemeyeceğinden kullanılmamıştır.

Veri üzerinde, veri madenciliği veya istatistik yöntemleri uygulayarak veriyi bilgiye çevirme: Bu aşamada verinin işlenmesi adına matematik, istatistik veya veri madenciliği yöntemleri kullanılır. Böylece gerekli olan bilginin veriden çıkarılması sağlanmış olur.

Verinin görselleştirilme yöntemine karar verme: En az verinin toplaması kadar önemli olan bu aşamanın daha veri toplanırken üzerinde düşünülmesi gerekir. Kullanılan birçok veri görselleştirmesi tipi arasından veri için doğru olanı seçmek, ulaşılmak istenen sonucu en doğru sunabilmek adına önemlidir. Örneğin, üç boyutlu verilerde kabarcık grafiklerinin kullanılması veya koordinat bilgisi içeren verinin harita üzerinde görselleştirilmesi amacına ulaşmasını kolaylaştıracaktır.

Görselleştirmeyi iyileştirme ve anlamını güçlendirme: Görselleştirilen veride dikkat çekmesi gereken bölümlerin incelenmesi ve ilk bakışta algının istenen bölgeye çekilebilmesi için üzerinde durulması gereken aşamadır. Bu aşamada kontrast renkler kullanmak, şekillerde ve boyutlarda değişiklik yapmak etkin olarak kullanılmaktadır.

Görselleştirme ile veri arasındaki ilişkiyi kurma: Son olarak bu aşama verinin görselle bağlantısının kurulması aşamasıdır. Bu aşamada görsel üzerinde verinin tipine göre farklı ifadeler kullanılabilir [6]. Örneğin haritalar üzerinde sadece illerin isimleri gösterilirken kullanıcının interaktif olarak illerin üzerine tıkladığında ayrıntılı bilgilere ulaşması sağlanabilir. Ayrıca, haritanın ölçeğine göre yaklaşıp uzaklaşmaya bağlı olarak sunulan veri, değişiklik gösterebilir. Böylece görselin anlaşılabilirliği ve kolay yoldan veri ile görsel arasında bağlantı kurularak görselleştirmenin amacına ulaşması sağlanabilir.

Brath vd. [2], çizge ve çizge veri tabanı modellerinin tanımları ve kullanımlarını açıklamıştır. Diğer veri tabanları ve modellerinin karşılaştırmalarının yer aldığı çalışmalarında özellikle çizge veri tabanı ile ilişkisel veri tabanı arasındaki farklar ortaya konulmuştur. Ardından detaylı olarak çizge veri tabanı modelleri ve sorgu dilleri tek tek incelenmiş, hangi amaçla kullanıldıkları, güçlü ve zayıf yönleri ele alınmıştır. Örneğin gen haritalarının saklanabilmesi adına kullanılan "GGL"(Graph Database System for Genomics) çizge veri tabanı sistemi incelenmiş, verilerin ikili çizgeler üzerinde tutulduğu, böylece gen haritalarının bu yapı üzerinde saklandığı anlatılmıştır. Son olarak konu ile ilgili açık olmayan taraflardan bahsedilmiş ve 2002 yılına kadar olan modeller ile sorgu dilleri belli metrikler doğrultusunda karşılaştırılmıştır.

Batra vd. [7], çizge veri tabanının önemine vurgu yapmıştır. İlişkisel veri tabanlarının bahsedilen amaçlar doğrultusunda yetersiz kaldığından bahsedilmiştir. İlişkisel ve çizge veri tabanlarının karşılaştırılması güvenlik ve esneklik gibi metrikler üzerinden yapılmıştır. Ardından her ikisinin de uygulama aşamalarından ve farklarından bahsedilmiştir. Son olarak karşılaştırma test sonuçları sunulmuş, bunun sonucunda Neo4J çizge veri tabanının

MySQL ilişkisel veri tabanından performans açısından daha üstün olduğu gösterilmiştir.

Çizge veri tabanlarından ve kullanımlarından bahseden bir araştırmasında Miller [8], çizge veri tabanlarının amaçları, olumlu ve olumsuz yönleri ile ilişkisel veri tabanları ile arasındaki karşılaştırmalardan bahsetmiştir. Çizge verinin nasıl saklanıp sorgulandığını ve aslında bu iki kavramın farklı amaçlar doğrultusunda kullanıldığı konusunda detaylı anlatıma yer vermiştir.

Bunun yanı sıra, çizge veri tabanı uygulamalarından; sosyal çizgeler, makine öğrenmesi doğrultusunda gerçekleştirilen öneri sistemleri (recommender systems) ve biyoenformatik için çizge veri tabanlarının nasıl kullanıldığı anlatılmıştır. Son olarak, performans metriklerinin bu bağlamda olumlu ve olumsuz durumlarından bahsedilmiştir.

2013 yılında yayımlanan bir araştırmasında Thompson [9], çizge veri tabanı literatürü ve özellikleri üzerinde durmuştur. Bununla birlikte, çizge veri tabanı platformları, sınıflandırılmış ve sınıflandırılmamış platformlar başlıkları altında detaylı olarak incelenmiştir. Ayrıca, her biri için uygulama, kullanım alanları ve metrikler üzerinden açıklamalar ve karşılaştırmalar yapılmıştır.

Çizge veri tabanları ve ilişkisel veri tabanları karşılaştırılması amaçlanan araştırmaya göre [3], önce çizge veri tabanları ve ilişkisel veri tabanları üzerine tanımlama ve açıklamalar yapılmış, sonrasında tüm bu tanımlar doğrultusunda kullanımlarından bahsedilmiş, karşılaştırmalar yapılmıştır. Karşılaştırmalı değerlendirme (benchmark) test sonuçları ortaya konmuştur. Bu sonuçlar, altı farklı sorgu tipi için gerçekleştirilmiş olup, sorgular aşağıdaki gibidir:

-Tüm tek (bağlantısı olmayan) düğümleri bulan sorgu
-Derinliği 4 ve 128 olan çizgeler üzerinde gezen sorgu
-Bir özelliği, belirli bir değere eşit olan düğümleri bulan sorgu
-Bir özelliği, belirli bir değerden küçük olan düğümleri bulan sorgu
-Belirli bir metin parçasını içeren düğümleri bulan sorgu.

Son üç sorgu tipinde, çizge veri tabanı olan Neo4J'nin performansı indekslemeye ve verinin boyutuna göre değişmektedir.

İlişkisel veri tabanı üzerinden çalışan bir sağlık bilgi sistemi uygulaması, performansının artacağı öngörülerek Sing ve Kaur [10] tarafından çizge veri tabanına taşınmıştır. Orta-büyük ölçekli bir veritabanı yapısına sahip olan bu uygulamada, veriler ilişkisel veritabanından çizge veri tabanına aktarılırken kullanılan metodoloji, en çok kullanılan verilerin birbirine yakın kaydedilmesi ve böylece sorgularda performansın artması üzerinedir.

3. Geliştirilen Yazılım: BigLogVis

Yazılımın amacı, kullanıcı sanayi kuruluşun isterleri kapsamında kayıt dosyalarının görselleştirilerek analiz edilmesi sonucunda mevcut ticari yazılım lisans miktarının ortaya konulmasıdır. Bu amaç doğrultusunda kayıt dosyalarının dinamikliğine uygun olabilmesi adına çizge veri tabanı kullanılmıştır.

3.1. Alt yapı

Verilerin saklanması için, açık kaynak olan ve Java ile geliştirilen Neo4J çizge veri tabanı kullanılmıştır. Bu veri tabanının, SQL yerine kendine özgü kullandığı 'Cypher' sorgu dili, SQL ile benzerlik göstermektedir [11]. Her iki

dilin karşılaştırılmasının ayrıntılarıyla bu çalışmanın temel katkılarından biri olarak kabul edilen Tablo 1'de sunulmuştur. SQL'deki veri getirme işlemi, Cypher'da da benzer mantıkta gerçekleşmektedir.

SQL'de tablo adı ve özelliği seçilirken, Cypher'da düğüm tipi ve onun özelliği sorguyu belirlemek için kullanılmaktadır. Sorgu sonuçları SQL'de 'SELECT' anahtar kelimesiyle elde edilirken, Cypher'da bunun yerine 'RETURN' anahtar kelimesi kullanılmaktadır. Sonuçları sıralama ve sınırlandırma işlemleri bire bir aynı gerçekleştirilirken, belirli bir özelliğe göre sorgulamak adına Cypher kısayol sunmaktadır. Düğüm tipinin belirlendiği bölümde özellik araması, ilave bir satıra gerek duymadan gerçekleşmektedir. Böylece, sorgularda en çok kullanılan bu bölüm daha kolay hale gelmektedir. Bu aşamaya kadar basit sözdizimsel farklılıklar yer alırken, en önemli ve büyük fark, tablo birleştirmelerinde ortaya çıkmaktadır. Tablo 1'de son iki satırda görüldüğü gibi, SQL'de istenen sorguyu elde etmek için 3 tabloyu birbiriyle bağlamak gerekmektedir. Bunun için satırlarca 'JOIN' işlemine ihtiyaç duyulmaktadır. Hâlbuki Cypher'da sadece bir satırlık sorgu ile istenen sonuç çok daha kolay elde edilmektedir.

Tablo 1. SQL-Cypher sorgulama dilleri işlem komutlarının karşılaştırılması

İşlem	SQL	Cypher
Select İşlemi	SELECT p.* FROM Products AS p;	MATCH (p:Product) RETURN p;
	SELECT p.ProductName, p.UnitPrice, FROM Products as p ORDER BY p.unitprice DESC LIMIT 10;	MATCH (p:Product) RETURN p.ProductName, p.UnitPrice, ORDER BY p.unitprice DESC LIMIT 10;
Bir Özellik ile 'SELECT'	SELECT p.ProductName FROM Products AS p WHERE p.ProductName = 'book';	MATCH (p:Product) WHERE p.ProductName="book" RETURN p.ProductName; --OR— MATCH (p:Product{productName = "book"}) RETURN p.ProductName;
In İşlemi	SELECT p.UnitPrice FROM Products AS p WHERE p.ProductName IN ('chocolate','chai');	MATCH (p:Product) WHERE p.ProductName IN ['chocolate','chai'] RETURN p.UnitPrice;
LIKE İşlemi	SELECT p.ProductName FROM Products AS p WHERE p.ProductName LIKE 'C%' AND UnitPrice > 100;	MATCH (p:Product) WHERE p.ProductName =~"C.*" AND UnitPrice > 100 RETURN p.ProductName;
JOIN İşlemi	3 Inner Join is required between Product, Company and Order Entities	MATCH (p:Product {productName ="book"}) <- [:INCLUDE] – (:ORDER) <-[PURCHASED] – (c:CUSTOMER) RETURN c.CompanyName;
GROUP BY İşlemi	SELECT e. EmpId, count(*) AS CNT FROM Employee AS e JOIN Order AS o ON (o.EmpId = e.EmpId) GROUPBY e.EmpId ORDER BY CNT DECS LIMIT 10;	MATCH (o:Order) <- [:SOLD] - (e:Employee) RETURN e.name, count(*) AS CNT ORDER BY CNT DECS LIMIT 10; **Neo4J gruplama işlemini otomatik gerçekleştirdiğinden, Cypher'da "GROUP BY" işlemine gerek yoktur.

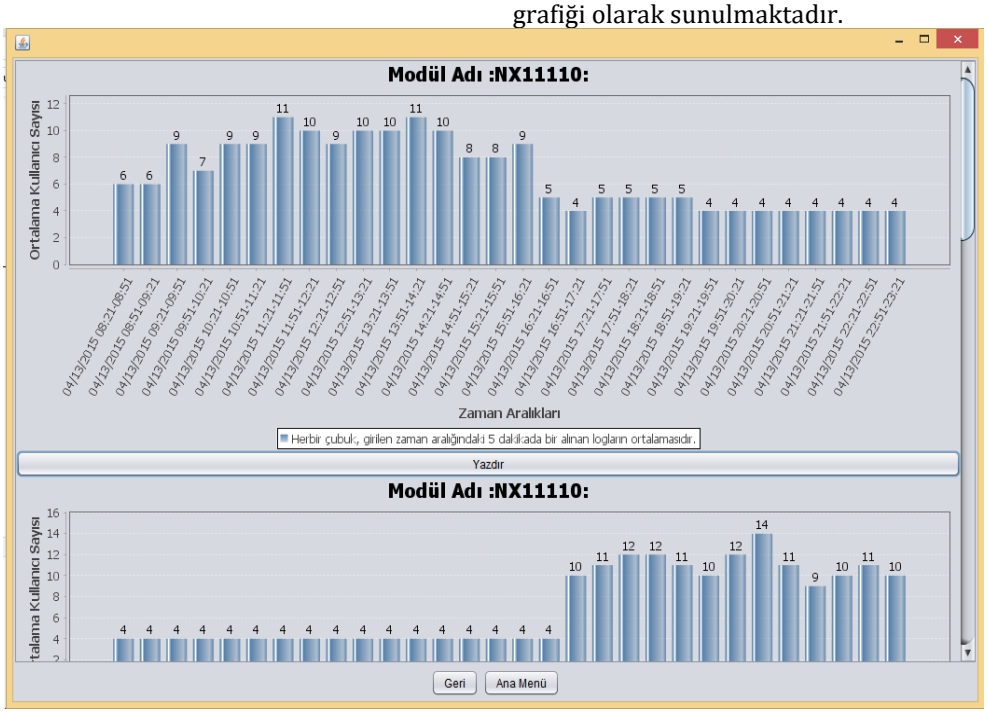
3.2. BigLogVis Uygulama Yazılımı

Geliştirilen yazılım çerçevesinde, detaylı 'yazılım modülü kullanım bilgileri' içeren kayıt dosyalarının görselleştirilmesi, böylece kayıt dosyalarının amaçları doğrultusunda karar destek sistemi olarak kullanılabilmesi sağlanmıştır.

Kayıt dosyaları, kendi içinde her beş dakikada bir veri tabanına kopyalanan kullanıcı kayıtlarından oluşmaktadır. Her bir kayıt, o kayıt içerisinde kullanılan modülleri ve her bir modül o anki kayıt içerisindeki kullanıcıları içermektedir. BigLogVis, dört ana bölümden oluşmaktadır.

3.2.1. Seçilen Bir Yazılım Modülünün Kaç Kişi Tarafından Kullanıldığı Bilgisi

Yazılımın bu bölümünde projenin sanayi destekçisi kullanıcı kurumun bünyesinde kullandığı yazılımların içerisinden seçilen bir yazılım modülünün yine kullanıcı tarafından girilen zaman aralıklarında ortalama kaç kişi tarafından kullanıldığı bilgisi hesaplanmaktadır. Bu bilgi, sütun grafiği ile interaktif olarak görselleştirilmiştir (Şekil 1.). Her bir sütun üzerine tıklandığında o sütunun zaman aralığı içinde yer alan 5-dakikalık kayıtlardaki tam kullanıcı sayıları sütun grafiği olarak sunulmaktadır.



Şekil 1. Yazılım modülünün kullanıcı bilgisini gösteren sütun grafiği

3.2.2. Seçilen Bir Yazılım Modülünün Hangi Kullanıcılar Tarafından Kullanıldığı Bilgisi

Bu özellik kapsamında, kullanıcı kaydından alınan modül adı, tarih aralığı ve saat dilimi özellikleri ile her bir kullanıcının ilgili saat diliminde modülü kullanıp kullanmadığı bilgisi tablo ile interaktif olarak görselleştirilmiştir. İmleç ile üzerinde durulduğunda, gerek kullanıcı bilgilerine gerekse kullanıcının ilgili modülü kullandığı detaylı tarih ve saatlere ulaşılabilmektedir. Bu özellik kısa süreli incelemeler için geliştirilmiştir. Eğer kullanıcı detaylı bilgiye ihtiyaç duyursa yeni bir ekranla detaylara ulaşılabilmektedir.

3.2.3. Tüm Kullanıcıların Kullandığı Yazılım Modülleri

Bu özellik doğrultusunda kullanıcıdan yalnızca tarih aralığı ve saat dilimi bilgisinin girilmesi istenir. Bu tarihler arasında tüm kullanıcıların kullandığı tüm yazılım modülleri yine interaktif tablo ile görselleştirilip kullanıcıya sunulur.

3.2.4. Modülün Tüm Lisanlarının Kullanılma Bilgisi

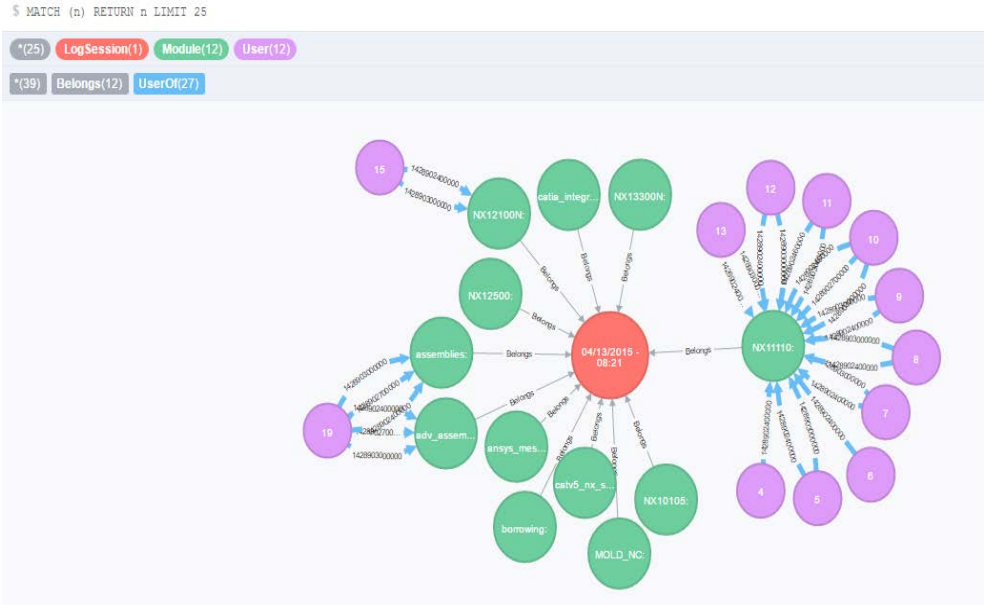
Bu özellik yazılım lisans miktar alımlarının uygunluğunu ölçmek için geliştirilmiştir. Kullanıcının girdiği modül ve zaman aralığına göre hangi saatlerde maksimum lisans sayısına ulaşıldığı bilgisini içerir. Böylece, sürekli maksimum lisans sayısında kullanılan bir modülün yetersiz/fazla lisans miktarı tespit edilmiş olur.

3.2.5. Çizge Veri Tabanı ile Görselleştirme

Bu görselleştirme sayesinde, ilk bakışta hangi modülün ne yoğunlukta kullanıcısı olduğu, hangi modüllerin hiç kullanılmadığı tespit edilmektedir. Tespitinden sonra diğer program özellikleri sayesinde istenen modül hakkında detaylı bilgiye doğrudan ulaşmak mümkün olmaktadır.

3.3 Geliştirilen Mimari

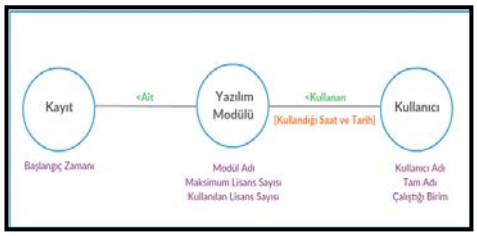
Proje Kapsamında kullanılan çizge veri tabanı (Neo4J) yapısı (Şekil 2.) ve alternatif olarak sunulan ilişkisel veri tabanı şeması (Şekil 4.) sunulmuştur. Şemalardan da kolaylıkla görüldüğü üzere, çizge veri tabanındaki yapı son derece basit ve herhangi bir ilişki eklenmek istenmesi durumunda sadece yeni bir düğüm ve ayrıtla kolayca yanıt verebilecek bir yapı olduğu görülmektedir. Fakat, ilişkisel veri tabanı şeması (Şekil 4.) içine herhangi bir yeni ilişki eklenmesi, yeni bir tablo veya tablolar eklenmesini gerektirmektedir. Neo4J içinde, SQL'de olduğu gibi 'date' veri tipi olmadığından, gün ve saat bilgileri 'Unix Time' şeklinde tutulmuştur. Unix zamanına göre tüm tarih, saniye cinsine çevrilir. Başlangıç zamanı olarak 1 Ocak 1970 baz alınır. Bu tarihten önceki tarihler için negatif sayılar kullanılmaktadır [12].



Şekil 2. Çizge veri tabanı

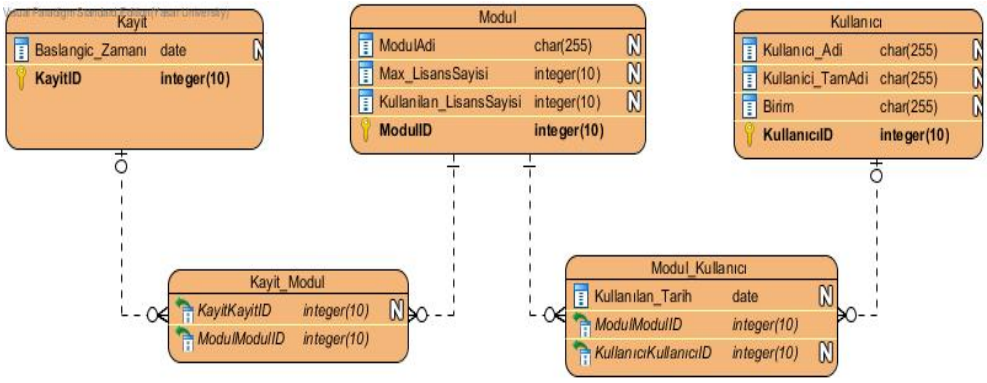
Çizge veri tabanı yapısında (Şekil 3.) düğüm tipleri 'Kayıt', 'Yazılım Modülü' ve 'Kullanıcı' olarak belirlenmiştir. Her bir yazılım modülünün hangi kayda ait olduğu bilgisi iki düğüm arasındaki "Ait" ayrıtı ile belirtilmiştir. Aynı şekilde Yazılım Modülü ile Kullanıcı arasındaki kullanma ilişkisi de "Kullanan" ayrıtı ile belirlenmiş ve bu ayrıta kullanıcının yazılım modülünü kullandığı tarih ve saat

özellik olarak eklenmiştir. Böylece ilişkisel veri tabanı tasarımındaki gibi yeni bir tablo oluşturma durumuna gerek kalmamaktadır.



Şekil 3. Çizge veri tabanı yapısı

İlişkisel veri tabanı yapısında (Şekil 4.) görüldüğü gibi çoğa-çok ilişkiler için yeni bir tablo yaratılmış ve verilerin bu şekilde saklanması öngörülmüştür. Bu durumda sorgu sırasında tüm bu tablolara erişebilmek için tabloları birleştirmek gerekmektedir. Bu durum hem yazılım performansını, hem de kodlama performansını düşürmektedir. BigLogVis programında bu veri yapılarının dışında gerekli olan özellikleri elde edebilmek için üç katmanlı mimari kullanılmıştır. Kullanıcı ara yüzü, program için gereken işlemlerin yapıldığı katman ve dosya/veri tabanı işlemleri tamamen birbirinden bağımsız çalışmaktadır.



Şekil 4. İlişkisel veri tabanı mantıksal modeli

4. Karşılaştırma

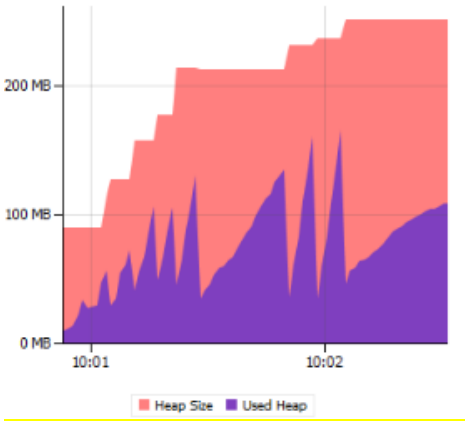
Çalışma kapsamında, kullanıcı kayıt dosyaları ile projenin sanayi destekçisi kurumun ihtiyaç duyduğu gerekli/gereksiz yazılım lisans miktarlarının tespit edilmesi amacıyla bir konsol yazılım uygulaması (BigLogVis) geliştirilmiştir. Böylelikle kurum içindeki karar vericiler, yazılım lisans alımında daha yerinde ve maliyet-etkin kararlar vermektedirler. Bu şekliyle, sanayi destekçisi kullanıcı kurum bünyesinde kayıt dosyalarının görselleştirilmesi projesi, bir yenilik unsuru barındırmaktadır.

BigLogVis uygulaması, öncelikle ilişkisel veri tabanı üzerinde geliştirilmiştir. Daha sonra performansın arttırılması adına çizge veri tabanına taşınmıştır. Şekil 5.'de çizge veri tabanında, Şekil 6.'da ilişkisel veri tabanında veri girilmesi esnasında Java sanal makinesi nesne kullanımı değerleri gösterilmektedir. Şekillerdeki turuncu renkli alanlar sanal makine tarafından yaratılan "heap" alanı, mavi renkli alan ise, yazılım programı tarafından kullanılan alanı göstermektedir. Veri tabanının kayıt dosyaları ile beslendiği aşamada, Şekil 5. ve Şekil 6.'da mavi renk ile gösterilen alanın, makine tarafından ayrılan kırmızı renkli alana göre daha küçük olması önem taşımaktadır. Bu durum, ilişkisel veri tabanının gerek CPU performansı, gerekse Java sanal makinesinde kapladığı

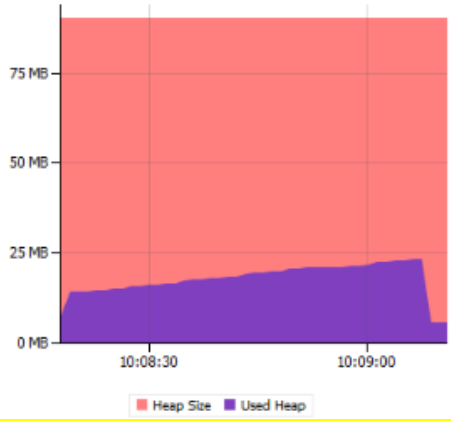
alan bakımından performansının üstün olduğunu göstermektedir. Söz konusu değerler, Netbeans 8.0.2 ile ölçülerek gözlenmiştir.

Bunun sebebi, Neo4J veri tabanının sağladığı kütüphane ile veri tabanını oluştururken her bir yeni kayıt için yeni düğüm nesnesi yaratmasıdır. Dolayısıyla, hem sanal makinede nesnelerin kapladığı alanın ilişkisel veri tabanına göre çok fazla olması, hem de nesne oluşumunun yarattığı zaman kaybına sebep olmaktadır.

Bunun yanısıra, çizge veri tabanının (Neo4J) gerek yazılımın geliştirilmesi ve veri tabanının oluşturulması aşamasında, gerekse sorgu aşamasında ilişkisel veri tabanına (MySQL) oranla performans açısından önemli derecede avantajlı olduğu Şekil 7.'de yapılan karşılaştırma sonucu ortaya çıkmıştır. Şöyle ki, çizge veri tabanında birleştirilmiş sorgunun cevap süresi 2.012 mili saniye iken, aynı sorgunun ilişkisel veri tabanındaki cevap süresi ise 17.5 mili saniye olarak gözlenmiştir. Bu durum çizge veri tabanı cevap süresinin yaklaşık 9 kat daha hızlı olduğunu göstermektedir. Proje, kayıt dosyalarının saklanması ve görselleştirilmesi üzerine olduğu için, karşılaştırmada seçim sorgusu, birleşim sorgusu ve kayıt verisinin veri tabanına işlenebilmesi adına gerekli olan kayıt ekleme sorguları kullanılmıştır.



Şekil 5. Çizge veri tabanı nesne kullanımı



Şekil 6. İlişkisel veri tabanı nesne kullanımı

Şekil 7. CPU zamanı ve bellek kullanımının karşılaştırılması

5. Sonuç ve Tavsiyeler

Çizge veri tabanının sağladığı kütüphane ile veri tabanı oluşturulurken her bir yeni kayıt için yeni düğüm nesnesi yaratılmakta, bu da çizge veri tabanının Şekil 5. ve Şekil 6.'daki değerlerini düşürmektedir. Performans kaybının sebebinin hem sanal makinede nesnelerin kapladığı alanın ilişkisel veri tabanına göre çok fazla olması, hem de nesne oluşumunun yarattığı zaman kaybından kaynaklandığı saptanmıştır. Yaşanan bu kaybın veri büyüdükçe ortadan kalkacağı, ilişkisel veri tabanına oranla performansının artacağı öngörülmektedir.

Çalışma kapsamında geliştirilen yazılım, projenin sanayi destekçi kurum tarafından kullanılmaktadır. Bu kullanım, geliştirilen yazılımın test aşaması olarak kabul edilerek, alınan geri beslemeler doğrultusunda iyileştirme ve yenileştirmeler yapılmıştır.

Projenin bir ileri aşaması olarak, veri tabanının beslenme performansını arttırmak adına büyük veri teknikleri uygulanabileceği değerlendirilmektedir.

Ayrıca ilişkisel veri tabanından çizge veri tabanına veri aktarımında performansı arttıracak farklı yaklaşımlar bulunmaktadır [13]. Sıklıkla kullanılan verilerin birbirlerine daha yakın kaydedilmesini öngören bu yaklaşım performansı arttıracaktır.

Teşekkür

Bu çalışma, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu tarafından desteklenmiştir (TÜBİTAK; Program: 2209B Proje Numarası: 1139B411503258). Yazarlar, projenin sanayi destekçi kurumu ASELSAN A.Ş.'den sanayi danışmanı Rıdvan TOROSLU'ya katkılarından dolayı teşekkür ederler.

Kaynakça

[1] Yazılım Sektörü Raporu, Batı Akdeniz Kalkınma Ajansı, 2012.

[2] R. Brath and D. Jonker. Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data. John Wiley Sons, Inc., 2015

[3] Zhonga C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen and D. Wilkins, "A Comparison of a Graph Database and a Relational Database", ACMSE '10, April 15-17, 2010, Oxford, MS, USA, 2010.

[4] I. Robinson, J. Webber, and E. Eifrem. Graph Databases: New Opportunity for Connected Data. O'Reilly Media, Inc., 2 Edition, 2015.

[5] B. Fry. Visualizing Data: Exploring and Explaining Data with the Processing Environment. O'Reilly Media, Inc., 2 Edition, 2008.

[6] R. Angles and C. Gutierrez, "Survey of graph database models", CSUR, vol. 40, no. 1, pp. 1-39, 2008.

[7] S. Batra and C. Tyagi, "Comparative Analysis of Relational and Graph Databases", International Journal of Soft Computing and Engineering (IJSCE), vol. 2, no. 2, 2016.

[8] J. Miller, "Graph Database Applications and Concepts with Neo4j", in Southern Association for Information Systems Conference, Atlanta, 2013.

[9] B. Thompson, "Literature Survey of Graph Databases", 2013.

[10] M. Singh and K. Kaur. "SQL2Neo: Moving Health-Care Data from Relational to Graph Databases". *IEEE Projects,* ieeeproject.org/project/page/3 (Erişim Tarihi: 27.03.2017).

[11] Neo4j: The world's leading graph database. http://neo4j.com/product/ (Erişim Tarihi: 01.03.2017).

[12] Unix time. https://en.wikipedia.org/wiki/Unix_time (Erişim Tarihi: 21.03.2017).

[13] R. De Virgilio, A. Maccioni, and R. Torlone, "Converting Relational to Graph Databases" in First International Workshop on Graph Data Management Experiences and Systems. ACM, 2013, p. 1.