Veri Yoğun Bilgi Sistemleri İçin Melez Bir Veri Mimarisi Önerisi

Murat Osman Ünalır, Emrah İnan, Burak Yönyül, Emre Olca, Fatmana Şentürk, Vahab Mostafapour, Petek Yıldız, Dilek Yılmazer, Devrim İşli

Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi,
35100 Bornova, İzmir, Türkiye
{murat.osman.unalir,emrah.inan,fatmana.senturk}@ege.edu.tr
{burakyonyul,devrimisli,dilekylmzr,emreatlier,petekyildiz,v.mostafapour}@gmail.com

Özet. Bulut bilişim teknolojileriyle birlikte bilgi sistemlerinde işlenen veri hacmi artmaktadır. Veri hacminin artmasının yanında, veriye hızlı erişim en önemli gereksinimdir. İlişkisel veri modeli kullanımının yanında ilişkisel olmayan veri modellerinin de kullanımı önemli boyutlara ulaşmıştır. Bu çalışmada veri yoğun bilgi sistemleri için melez bir veri mimarisi önerisinde bulunulmaktadır. Öncelikle büyük veri kapsamında farklı veri modellerinin temel özellikleri tartışılmaktadır. Örnek bir bilgi sisteminin gereksinimleri doğrultusunda farklı veri modellerine duyulan gereksinimler vurgulanmaktadır. İlgili bilgi sistemini temel alan bir durum çalışması metamodel mimarisine uygun olarak ontolojik bir yapıda tasarlanmıştır. Melez veri mimarisini meydana getiren farklı veri modellerinin anahtar-değer veri modeline dönüştürülebilirliği gösterilmiştir. Bilgi sisteminin ilgili veri servislerinin temel özellikleri farklı teknolojiler kapsamında sunulmuştur.

1 Giriş

Bulut bilişim istenilen yerden bilgisayar ağları, sunucuları, depolama, uygulama gibi bilişim kaynaklarına servisler aracılığıyla erişimine olanak tanır [1]. Bu kaynakları kullanmak için yönetim ve servis etkileşimi maliyetlerini en aza indirmeyi hedefler [2]. Böylece altyapı, esneklik ve kaynakların kullanılabilirliği gibi sorunlarla uğraşılmadığından ve sonuç olarak ana işe odaklanılmakta işlenen verinin hacmi de artmaktadır. Ancak bu durum verinin yönetilmesini zorlaştırır. Büyük veri, ilişkisel veri modeli yaklaşımıyla ölçeklenemeyen verinin verimli bir şekilde depolanmasına, yönetilmesine ve işlenmesine olanak tanır [3]. Bulut bilişim, büyük verinin hesaplama ve işleme süreçlerine imkan sunmakla birlikte servis modeli olarak da hizmet eder.

Büyük verinin hacim, hız ve çeşitlilik olmak üzere 3 karakteristik özelliği vardır [4], [5]. Hacim, çok büyük miktarlardaki veriye, hız ise verinin kaynaklar arası aktarılma süratine karşılık gelmektedir. Çeşitlilik, farklı biçimde ve kaynaktaki verilerin toplanmasıyla ilgilenmektedir.

Farklı veri modellerinin bir bilgi sisteminde bir arada kullanılabilmesi için melez bir veri mimarisine gereksinim duyulmaktadır. Not only SQL (NoSql)

olarak adlandırılan ilişkisel olmayan veri modelleri bir veri mimarisinin çözümünde sadece ilişkisel veri modelinin kullanılamayacağını ifade etmektedir [6]. NoSql veri modelleri anahtar-değer, çizge, doküman ve sütun veritabanı olmak üzere 4 alt kategoriye ayrılır. Anahtar-değer veritabanı okuma ve yazma işlemlerini hızlandırır. Çizge veritabanı düğümlerin bağlar aracılığıyla kolay bir şekilde dolaşılmasını sağlar. Doküman veritabanı farklı biçimlerdeki veri kaynaklarının daha esnek işletilmesine olanak tanır. Sütun veritabanı ilişkili sütunların bir sütun ailesinde toplanmasıyla uygulamalara geniş kapsamlı sorgu ve veri çözümlemeleri yapabilme yeteneğini sunar.

Çok çeşitli kalıcılık (Polyglot persistence) [7], ihtiyaç duyulan melez yaklaşımı tanımlamak için kullanılır. Ancak buradaki her veri yönetim modeli ayrı birer yönetimsel karmaşıklığa sahiptir. Saklanan veri arttıkça performans kayıpları oluşmakta ve oluşan bu kayıpların giderilmesi de birbirinden farklılık gösterebilmektedir. Buna göre tasarlanan mimarinin gereksinimine uygun veri mo-delinin seçilmesi önem kazanmaktadır [8]. Bu çalışma kapsamında, Öğrenci Bilgi Sistemi (ÖBS) için farklı veri modellerini barındıran melez bir veri mimarisi önerilmektedir. ÖBS; öğretim üyeleri, öğrenciler ve derslere ilişkin bilgilerin saklandığı bir sistemdir. Öğretim üyelerinin verdiği dersler, öğrencilerin kayıtlandığı dersler, derse ait sınav sonuçları, öğrenci ve öğretim üyelerinin özgeçmişleri yer almaktadır.

Bu çalışmanın ikinci bölümünde büyük veri yönetimi için veri modelleri tanıtılmıştır. Üçüncü bölümünde ise bu veri modelleri üzerinde ÖBS ile ilgili durum çalışması yapılmıştır. Son olarak önerilen melez veri modelinin getirileri tartışılmış ve ileriki çalışmalar anlatılmıştır.

2 Büyük Veri Yönetimi için Veri Modelleri

Veri yönetimi için ilişkisel veritabanı sistemleri F. Codd [9] tarafından öne sürüldükten sonra zamanla geliştirilmiş ve böylece verilerin tutulması, saklanması ve işlenmesinde büyük kolaylıklar sağlanmıştır. Ancak zamanla sistemleri kullanan kişi sayısı ve buna paralel olarak saklanan veri boyutu artmaya başlamıştır. Farklı kullanıcılar aynı tabloya eş zamanlı olarak erişmek istediğinde, işlemlerin öncelik sırasına göre bitmesi beklenmektedir. Özellikle işlem hacminin yoğun olduğu sistemler üzerinde uzun süreli beklemeler olabilmekte, bu da sistemin kullanıcılara cevap veremez duruma gelmesine ve performans kayıplarına sebep olmaktadır. Büyük hacimli verilerin merkezi sunucularda depolanması ve geleneksel veri analiz yöntemleri ile analiz edilmesinin verimsiz ve pahalı olduğu kanıtlanmıştır [10]. İlişkisel veritabanlarında yapılan iyilileştirmeler veri boyutu uyarlanabilirlik gereksinimini ve performans ihtiyacını karşılayamamaktadır. Bu sebeple yatayda veya dikeyde veri boyutu uyarlanabilir bir veri saklama yapısına ihtiyaç duyulmuştur. Bu kapsamda farklı amaçlara göre özelleşmiş veritabanı çözümleri geliştirilmiştir. 2003 yılında ilk veri boyutu uyarlanabilir veritabanı ortaya çıkmış ve sonraki yıllarda da farklı amaçlar için yeni veritabanları geliştirilmeye devam etmiştir [6] .

Karmaşıklık ve boyuta göre veritabanları incelendiğinde ilişkisel veritabanı diğerlerine göre kolay çözüm sunmasına rağmen veri boyutu uyarlanabilir değil-

dir [11]. Diğer veritabanlarında ise veri boyutu arttıkça daha zor çözüm üretilmektedir. Hem karmaşıklık hem de veri boyutu uyarlanabilirlik kısıtlarını sağlamak amacıyla farklı veri modelleri içeren sistemlerde büyük veri destekleyen veri-tabanlarından birkaçının kullanılması gerekmektedir. Bu durum birden fazla veri deposunun işlenmesi nedeniyle veri yönetim sorunlarına neden olmaktadır. Karşılaşılan veri modelleme sorununa çözüm olarak farklı veri modellerinin tek bir veri depolama altyapısına dönüştürülmesi önerilmiştir.

Bu yaklaşım, verinin esnek olarak modellenmesine olanak sağlamaktadır [12]. Ancak verilerin hangi veri modeline uygun olduğunun belirlenmesi önemlidir. En az karmaşıklığa ve en uyarlanabilir veri boyutuna sahip veritabanı olması nedeniyle diğerlerinin anahtar-değer veritabanına dönüştürülmesi daha az maliyetlidir. Bu şekilde, farklı veritabanlarında yer alan verileri ortak bir veritabanına dönüştürerek farklı veri modellerini destekleyen bir veri modeli mimarisi önerilmiştir. Birden çok veri modelini anahtar-değer veri modeline dönüştüren çalışmalardan FoundationDB [13], farklı veri modellerindeki veriler için ilişkisel, çizge ve doküman veritabanlarını destekleyen bir yapı oluşturmuştur. ArangoDB [14] ise doküman, çizge, anahtar-değer veritabanlarını destekleyen esnek açık kaynak bir veritabanı olarak sunulmuştur. Veritabanları arasındaki dönüşüm işlemleri sırasında yaşanabilecek herhangi bir problem, veriler arasında tutarsızlık ve veri kaybına neden olabilmekte, veritabanları arası dönüşümler ise bekleme sürelerine yol açabilmektedir. Veritabanları arası dönüşüm yerine veri modellerine göre verileri; çizge, doküman, sütun ve anahtar-değer veritabanlarında saklama gereksimi vardır. Bu çalışma kapsamında, bu farklı modellerdeki veriler arasındaki ilişkileri anahtar-değer veritabanında tutan veri yoğun bilgi sistemlerine yönelik bir veri mimarisi önerilmektedir. Bundan sonraki alt başlıklarda sırasıyla ilişkisel, doküman, anahtar-değer, sütun ve çizge veritabanı kavramları detaylandırılmıştır.

2.1 İlişkisel Veritabanı

Verilerin birbirinden farklı ve belirli özelliklere göre ilişkilendirilmiş tablolarda tutulduğu veritabanı yönetim sistemidir. Tablolar satır ve sütunlardan oluşmaktadır. Tablo içerisindeki her sütun bir niteliğe karşılık gelmekte ve oluşturma aşamasında tanımlanmaktadır. Her satır ise bir kayda karşılık gelmekte ve veri girişi sırasında belirlenmektedir. Tabloda ayrıca tanımlayıcı niteliğinde birincil anahtar alan ve tablolar arası geçişi sağlayan ikinci anahtar alan bulunur. Bu alanlar kullanılarak farklı tablolar birleştirilebilir ve birbirine bağlı veriler tek seferde sorgulanabilir. İlişkisel veritabanı kavramı, verileri saklamak için kullanılan mevcut dosyalama sisteminin veri ilişkilerini ve veri bütünlüğünü sağlayamaması üzerine oluşturulmuştur. İlişkisel veritabanları sağladığı tutarlılık ve bütünlük avantajları sayesinde, uzun bir süre verilerin saklanması için yeterli olmuş ve pek çok veritabanı sisteminin temelini oluşturmuştur.

2.2 Doküman Veritabanı

Doküman veritabanı sistemleri, ilişkisel olmayan veri modellerinden biridir. Yarı-yapılandırılmış veriler olarak bilinen doküman tabanlı bilgilerin saklanması,

alınması ve yönetilmesinde kullanılan bir veri modelidir. Tablolara karşılık derlemler, satırlara karşılık dokümanlar, sütunlara karşılık da alanlar kullanılmaktadır. Sıradan bir şema yerine değişken bir şema temel alınmaktadır. Bazı alanların tüm kayıtlarda kullanılmadığı durumlarda, ilişkisel veritabanlarında fazladan yer kaplarken bu alanlar doküman veritabanlarında yer almaz. Veri modeli tek tip değildir ve veriye göre esneklik gösterir. Böylece bir dokümanda yer almayan bazı alanlar bir başka doküman içerisinde kullanılabilir. Herhangi bir alan eklemek veya çıkarılmak istendiğinde yapısal bir değişiklik yapılmamış olur ve sadece ilgili doküman değiştirilir. Böylece, gerekli değişikliklerin yapılması için fazladan bir zaman gerekmemektedir.

2.3 Anahtar-Değer Veritabanı

Veritabanı sistemleri içerisinde karmaşıklığı en az olanıdır. Bir anahtar ve bu anahtara karşılık gelen değerin saklandığı veritabanı sistemidir. Eğer bir anahtarın değeri biliniyor ise, anahtarın tuttuğu değer getirilebilir. Değer alanı, yapılandırılmamış veri türündedir. Yani değerin içeriği sayı, bağlantı, metin, vb. gibi farklı veri yapılarından oluşabilir. Örneğin, Twitter sosyal ağında atılan her tweet için bir tweet numarası ve içeriği bulunmaktadır. Böyle bir yapının, anahtar-değer veritabanında tutulması uygundur. Bu veritabanı türünde genel olarak değer kısmında yapısal olmayan veriler tutulmaktadır.

2.4 Sütun Veritabanı

Anahtar-değer depoları ve doküman veritabanları satır odaklıdır. Bu tür veritabanlarının amacı bir yada daha fazla kısıta göre belirlenen bütün veriyi getirmektir. Bazı durumlarda uygulamalar bütün belgenin derlemi yerine belirli sütunların oluşturduğu alt kümelere ihtiyaç duyar. Sütun veritabanı, satırları sütun derlemleri gibi yapılandırabilir. Tek bir satır birçok sütun içerebilir. İlişkili sütunların bir sütun ailesinde toplanması ile çok varlıklı sütun verilerine erişilebilir. Sütun ailesindeki bu esneklik, ilişkisel veritabanları tarafından sağlanan işlevselliğe benzer olarak, uygulamalara geniş kapsamlı karmaşık sorgu ve veri analizlerinin yapılmasına olanak sağlar.

2.5 Çizge Veritabanı

Veri hacmi arttıkça ilişkisel veritabanında veri üzerinde arama mümkün olmakta, ancak zincirleme ilişkiler getirileceği zaman birleştirme maliyetleri doğmaktadır. Doküman ve anahtar-değer veritabanlarında ise arama ilişkisel veritabanına göre daha zordur. Doküman veritabanlarında değerler döküman içerisinde, anahtar-değer veritabanlarında ise anahtara karşılık gelen değer kısmında yer alır. Zincirleme ilerlemek için herhangi bir birleşim yapısı olmadığından, değerler aranıp bulunmalı ve eşlenmelidir. Ayrıca arama mümkün olsa bile veri üzerinde dolaşıp ilerleme mümkün olmamaktadır. Çizge veritabanlarında veriler; düğümler, düğümler arasındaki ayrıtlar ve düğümler ile ayrıtların özellikleri şeklinde

tutulur. Ayrıtlar üzerinden ilerlenerek düğümler arasında gezmek ve arama yapmak mümkündür. Belirli düğümlere veya ayrıtlara özel aramalar yapılabilmektedir.

3 Durum Çalışması: Büyük Veri Modellerinin Öğrenci Bilgi Sistemi Üzerinde Uygulanması

ÖBS kapsamında; öğrenci bilgileri, öğretim üyesi bilgileri, dersler, öğrencinin aldığı ders bilgileri, öğrenci ve öğretim üyelerinin özgeçmişleri, öğrenci ve öğretim üyelerinin sosyal medya üzerinde yaptığı paylaşımlar yer alır. Dersler hakkındaki görüşler, öğrenci, öğretim üyesi ve derse ait sosyal ağ hesapları aracılığıyla toplanır. Öğrenciler, ÖBS'ye giriş yaptıktan sonra herhangi bir öğretim üyesi tarafından verilen bir veya birden fazla derse kayıt olabilmekte ve kayıtlı oldukları derslere ilişkin sınav sonuçlarına erişebilmektedir. Öğrenciler, öğretim üyeleri veya dersler bir sosyal ağ hesabına sahip olabilmektedir. Öğrenci ve öğretim üyelerinin sistemde kayıtlı bir veya birden fazla özgeçmiş belgesi bulunabilmektedir. Ayrıca öğrenci ve öğretim üyelerine ait kişisel bilgiler, iletişim bilgileri de saklanmaktadır.

3.1 Durum Çalışması Gereksinimleri

İlişkisel Veritabanı Gereksinimi Tasarlanan bilgi sisteminde öğrenci ve öğretim üyelerine ait kişisel bilgiler ilişkisel veritabanında ayrı tablolarda saklanır. Kişisel bilgiler tablosu kimlik numarası, adres, telefon numarası ve e-posta gibi bilgilerden oluşur. Bu bilgilere erişim öğrenciler için öğrenci numarası, öğretim üyeleri için öğretim üyesi numarası olarak tanımlanan anahtar sahalar ile sağlanır. Tanımlanan veri tipleri ve ilişkilere göre bu veriler ilişkisel veritabanında tutularak ihtiyaçlar karşılanabilmekte, veriler arasındaki ilişkiler kullanılarak öğrenci ve öğretim üyelerine ait kişisel bilgilere kolaylıkla ulaşılabilmektedir.

Doküman Veritabanı Gereksinimi ÖBS'de bazı veriler ilişkisel veri modelinde saklanırken, bazı verilerin ilişkisel olarak modellenmesinde zorluklar yaşanmaktadır. Genel olarak özgeçmiş biçiminin değişime açık olduğu ve kayıtlar arasında farklılık olabileceği görülmektedir. Ders içerik bilgilerinde de benzer bir durum söz konusudur. Ders bilgilerinin "dersin çıktıları" alanı her derse göre farklılık göstermektedir. Kayıt bazında alan özelleştirilmesine ihtiyaç duyulan bu gibi durumlarda klasik ilişkisel veri modeli yetersiz kalmaktadır. Özgeçmiş ve ders bilgileri için erişim maliyetlerini azaltan, performans artışı sağlayan ve yatayda boyutunun değiştirilebilindiği dinamik bir veri modeline ihtiyaç duyulmaktadır. Bu gereksinimleri karşılayabilecek bir veri modeli olarak doküman veritabanı kullanılabilir.

Anahtar-Değer Veritabanı Gereksinimi ÖBS'de; öğrenci, öğretim üyesi ve diğer kullanıcıların birer rolü bulunur. Bir öğrenci sadece kendisi ile ilgili bilgileri

görüp değişiklikler yapabilir. Benzer şekilde bir öğretim üyesi de kendi bilgileri ve verdiği derslerin bilgilerini görüp üzerinde işlemler yapabilir. Bu nedenle ÖBS'de her bir kullanıcıya kullanıcı numarası ve şifre verilip kullanıcı bu numara ve şifre ile sisteme giriş yaptığında kendi rolüne göre yetkileri belirlenir. Kullanıcı işlem yaparken de bu oturum bilgilerinden yararlanır. Kullanıcının numarası ve oturum bilgisi ayrı bir veri modelinde tutulur. Sistemde her öğrencinin aldığı derslerin listesi bulunur. Benzer şekilde her bir derse kayıtlı öğrencilerin de numaraları tutulmak istenmektedir. Bu gereksinime göre ders kodu ve öğrenci numarası gibi ikilileri saklayan bir veri modeli bulunmalıdır. Özellikle ders-öğrenci ilişkilerinin yoğun olduğu ders kayıt ve ekle-sil haftalarında bu yapının işlem hacminin artması ile sistem beklemelerinin en düşük düzeyde tutulması gerekmektedir. Bu kapsamda hızlı, esnek ve veri boyutu uyarlanabilir bir veri modeli kullanılmalıdır. En az karmaşıklığa sahip ve hızlı cevap verme gereksinimi söz konusu olduğunda çözüm olarak anahtar-değer veritabanı yaklaşımı önerilir.

Sütun Veritabanı Gereksinimi Sütun veritabanları verilere hızlı erişim sağlamakta ve özel sorgu setlerini desteklemektedir. Bu nedenle sütun veritabanlarından faydalanabilmek için, uygulamada çalışan genel sorguları eniyileyen sütun aileleri tasarlanmalıdır. ÖBS'de öğrencilere ait dersler ve notların bir arada tutulması gerekir. Her ne kadar tablo yapısı olarak ilişkisel veritabanına benzese de, ilişkisel veritabanında yer alan bir tablonun aksine, sütun ailesi içinde sütunlar, her satır için değişmez bir şemaya uymak zorunda değildir. Sütun ailesi anahtar-değer çiftlerinin haritası gibi düşünülürse, öğrenciler için ders notlarının bir arada tutulması istenen veriye doğrudan erişimi sağlayacaktır.

Çizge Veritabanı Gereksinimi ÖBS'de; öğrencilerin birbirlerini takip etme durumu, öğretim üyeleri ile olan danışmanlık ve takip ilişkileri, derse kayıtlanma ilişkileri, öğretim üyelerinin ders verme ilişkileri, öğrencilerin ve öğretim üyelerinin sosyal ağ hesaplarına ilişkin sayfaları bulunmaktadır. Bu verilere ve ilişkilere uygun bir çözüm çizge veritabanı olarak öngörülebilir. Buna göre öğrenciler, öğretim üyeleri, dersler ve bölümler çizgedeki düğümlere karşılık gelmektedir. Bu düğümlerin her birinin kendine özgü özellikleri bulunmakta ve düğümlerin de birbirleri ile ilişkileri yer almaktadır.

3.2 Durum Çalışması Tasarımı

Tasarlanan bilgi sisteminde ihtiyaç duyulan farklı veri modellerini kullanan servisleri yönetebilmek için bir üst modele ihtiyaç vardır. ÖBS'nin dayandığı bu üst modelde bilgi sistemindeki hangi veri modelinin hangi servis(ler) ile bağlantılı oldukları ve bu veri modellerinin birlikte çalışabilirliğinin nasıl yönetileceği tanımlanmalıdır. Öncelikle üst modelde tanımlı eşlemeye göre, sistemde birbirleri ile ilişkili ve geçişlere sahip modüllerin belirleyici özelliklerinin eşlemeleri ayrı bir anahtar-değer veritabanında tutulmalıdır. İkinci olarak üst modelde gelen isteklerin ilişkili modüllere iletilmesini sağlayan bir yönetim mekanizması yer almalıdır. Böylece bilgi sistemine gelen istekler tiplerine göre uygun modül ile

ilişkilendirilebilir. Tasarlanan bu üst model için ontolojik bir yaklaşım uygundur. Üst-Nesne Çerçevesi [15] açısından düşünüldüğünde tasarlanan mimarinin M2 ve M1 katmanları açısından incelenmesi gerekir.

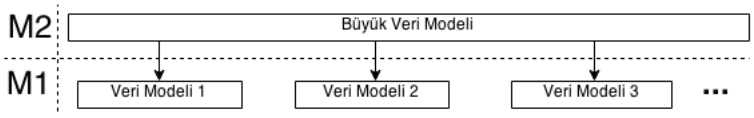

Fig. 1. Büyük Veri Modeli Taslağı

Şekil 1'de görüldüğü gibi Büyük Veri Modeli M2 katmanına karşılık gelmektedir. Bu model içerisinde birbirinden farklı veritabanlarında bulunan ortak özelliklerin ÖBS adı altında bir ontoloji [16] üzerinden bağlantıları kurulup daha sonra bu bağlantılar anahtar-değer veritabanında temsil edilmelidir. M1 katmanı ise öğrencilere ait bilgilerin tutulduğu ilişkisel veritabanı, özgeçmişlerini taşıyan doküman veritabanı vb. gibi farklı veri modellerini kapsamaktadır.

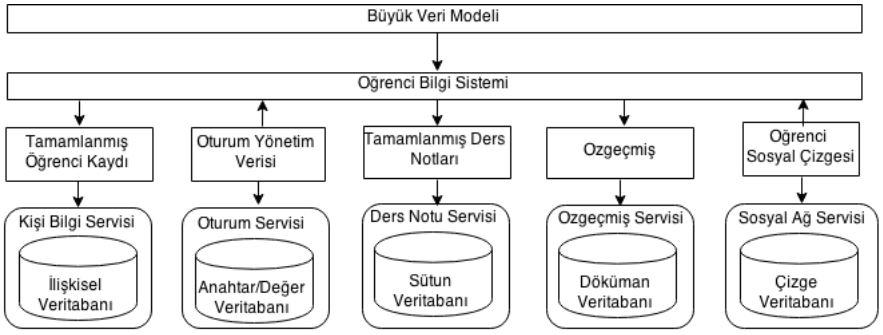

Fig. 2. Durum Çalışmasının Tasarımı

Şekil 2'de görüldüğü gibi durum çalışmasında örnek olarak tasarlanan ÖBS'de veriler birbirinden farklı veri modellerinde saklanmaktadır. Örneğin, öğrenci kayıtları ilişkisel veritabanında yer alırken, oturum verileri anahtar-değer veritabanında tutulmaktadır. Ders notlarını sütun veritabanında, öğrenci ve öğretim üyelerine ait özgeçmişleri doküman veritabanında ve son olarak sosyal ağa ait öğrenci ve ders bilgileri çizge veritabanında saklanmaktadır. Bu farklı veritabanlarına ait ortak özelliklerin bütünleştirilmesi bir üst model olan büyük veri modelinde sağlanmaktadır. Öğrenci Bilgi Sistemi için hazırlanan ÖBS ontolojisi kullanılarak Şekil 3'teki büyük veri modeli içindeki ontolojiye ait bağlantılar gösterilmiş, daha sonra bu bağlantıların temsili anahtar-değer veritabanında saklanmıştır.

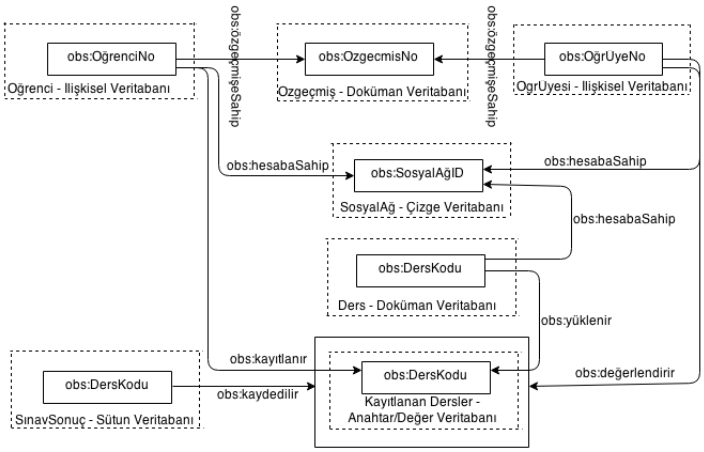

Fig. 3. Büyük Veri Modelinin Gösterimi

Öğrenci Bilgi Sistemi'nin melez veri modeli gerekliliği birden fazla veri modelini ilgilendiren karmaşık sorgularda ortaya çıkmaktadır. ÖBS'ye gelen karmaşık sorgu Şekil 3'teki ontolojik gösterimdeki eşlemelere [17] göre ayrıştırılarak sorguların işletileceği alt modüller belirlenir. Her bir alt sorgunun işletilmesiyle elde edilen sonuçlarının anahtar-değer eşlemeleri Şekil 4'teki gibi elde edilir. Ara sonuçlar bağlantılı olduğu modüle eşlenir ve bu modülde çalıştırılacak olan bağlı sorgu oluşturulur. Bu mantıkla sorgular zincirleme şekilde son alt sorguya kadar işletilir ve sonuç olarak karmaşık sorgunun cevabı elde edilir.

Örneğin ÖBS'de bir öğrencinin aldığı dersleri veren öğretim üyelerinin sosyal ağ bilgilerine erişilmek istensin. Bu karmaşık sorgu ontolojik gösterimine göre çözümlenerek kayıtlanılan dersler, öğretim üyeleri bilgileri ve sosyal ağ bilgileri olmak üzere üç alt sorguya ayrılır. Öncelikle kayıtlanılan ders bilgileri anahtar-değer veritabanından getirilir. Kayıtlanılan dersi veren öğretim üyelerine olan bağlantı ontolojik gösterime göre elde edilir ve bu bağlantı ilişkisel veritabanı olarak bulunur. İlişkisel veritabanına olan bağlantı anahtar-değer veritabanı dönüşümü üzerinden yapılır ve öğretim üyelerine ait bilgiler elde edilir. Ontoloji gösterimindeki ilişkilere göre öğretim üyelerinin sosyal ağ bilgileri çizge veritabanı üzerinden bulunur. Öğretim üyelerinin bilgileri anahtar-değer dönüşümü ile eşlenir ve sosyal ağ bilgilerini elde etmek için çizge veritabanında çalıştırılacak sorgu oluşturulur. Oluşturulan sorgu çizge veritabanında işletilerek gerekli bilgiler elde edilir.

3.3 Durum Çalışması Modülleri

Sosyal Ağ Çizgesi Modülü Öğrenciler, öğretim üyeleri, dersler gibi düğümlerin ve birbirleri arasındaki ilişkiler için çizge veritabanı uygun bir yaklaşımdır. Şekil-4.a'da ÖBS'deki örnek bir verinin çizge veritabanı üzerindeki temsili ve anahtar-değer veritabanına dönüştürülmüş hali görülmektedir. ÖBS üzerinden sosyal ağ

çizgesi modülüne gelen istekler modül tarafından ele alınarak işlem tipi belirlenir. İşlem tipi dolaşma veya sorgu olmasına göre çizge servisi üzerine farklı şekillerde yönlendirilir. Eğer istek dolaşma ise çizge servisinin dolaşma özelliğinin kullanılması için kullanıcı servis arayüzü üzerine yönlendirilir.

Özgeçmiş ve Ders Bilgileri İçin Doküman Veritabanı Modülü Dinamik bir yapıya sahip olan özgeçmiş ve ders bilgilerinin tutulması için doküman veritabanı seçilerek artan/azalan alanlar ve farklı biçimlere sahip şemalar için performanslı bir çözüm önerilmiştir. Özgeçmiş kayıtları ve ders bilgileri birbirlerine göre farklılıklar gösterdiği için veri tekrarları ya da boş alanlar olmadan doküman veri modeli olarak saklanmaktadır. İhtiyaç duyulduğunda kayda doküman olarak ulaşılabildiği gibi, dokümanın alt alanlarına da verilen etiket adıyla ulaşılarak arama işlemi gerçekleştirilebilmektedir.

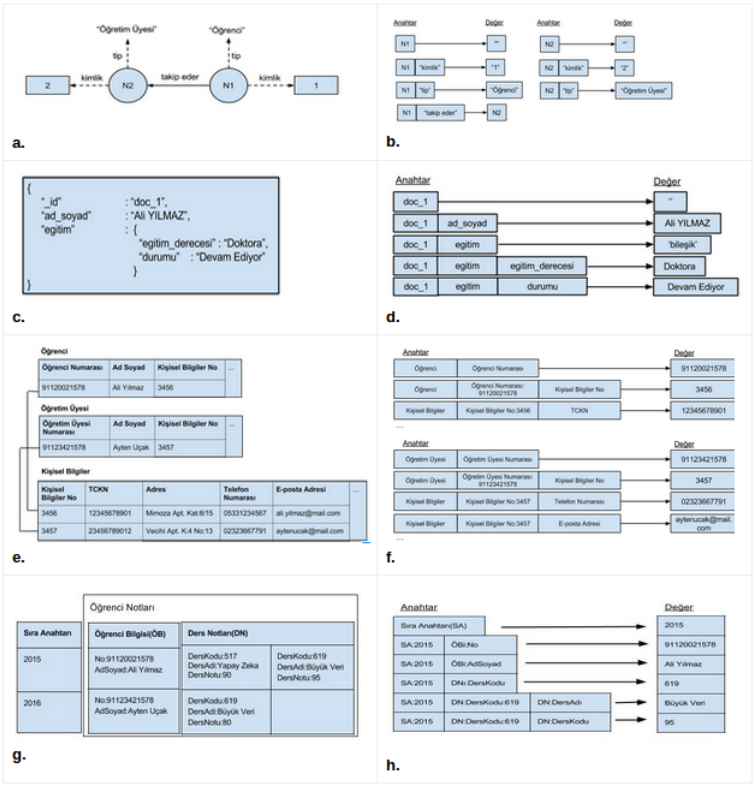

Fig. 4. Örnek Veri Modellerinin Anahtar-Değer Veritabanı Üzerinde Temsilleri

Öğrenci ve Öğretim Üyesi Bilgileri İçin İlişkisel Veritabanı Modülü Öğrenci ve öğretim üyesi bilgileri yapısal olarak değişiklik göstermeyen ve birbiri ile bağlantılı veriler olduğundan ilişkisel veritabanında tutulur. İlişkisel veritabanında bir veya birkaç adımda erişilecek veri için, anahtar-değer veritabanında pek çok adım gerekebilir. Şekil 4.e'de ilişkisel veritabanı üzerinde, Şekil 4.f'de ise anahtar-değer veritabanı üzerinde öğrenci ve öğretim üyesi tabloları temsili olarak gösterilmiştir. Öğrenci ve öğretim üyesine ait kişisel bilgiler için ise ayrı bir tablo yaratılmış, öğrenci ve öğretim üyesi tabloları ilişkilendirilmiştir.

Sütun Veritabanı Modülü ÖBS için geliştirilen sütun veritabanı yapısında süper sütun adı statik olarak "Öğrenci Notları" olarak belirlenmiştir. Ancak yapı gereği öğrenciler ve aldıkları dersler için gerekli olan sütun adları dinamik olarak tutulmaktadır. Şekil 4.g'de öğrenciler ve her birinin kayıtlandıkları ders ve derse ait notlar belirtilmiştir.

3.4 Durum Çalışması Teknolojileri

İlişkisel Veritabanı Servisi Bulut üzerinde yer alan bir ilişkisel veritabanının kolay bir şekilde kurulması, çalıştırılması sağlayan web servisleridir. Klasik ilişkisel veritabanından farklı olarak, maliyet açısından verimlidir ve yeniden boyutlandırılabilmektedir. Ayrıca, otomatik yedekleme ve hata denetimi, yazılım düzeltmeleri, geri yükleme işlemleri ve güvenli erişim alanlarının yönetilmesine imkan sağlar [18]. Amazon RDS [19], Microsoft SQL Azure [20] ve Google Cloud SQL [21] servisleri ile ilişkisel veritabanı hizmetleri kullanılabilir.

Doküman Veritabanı Servisleri ÖBS'de, öğretim üyesi özgeçmiş bilgileri ve ders bilgileri için doküman tabanlı veri modelinin kullanılması uygun görülmüştür. Kayıt içindeki alanların farklılık gösterdiği durumlarda verilerin ilişkisel modelde tutulması veri tekrarlarına veya hücrelerin boş kalmasına sebep olabilir. Farklı tablolarda tutulmasının sonucu olarak birleştirme maliyeti ortaya çıkabilir. Doküman veritabanı hizmetlerinin servis olarak kullanabileceği teknolojilere MongoDB [22] on Amazon EC2 [23] ve Google Cloud Platform [24] örnek verilebilir.

Anahtar-Değer Veritabanı Servisleri ÖBS'de oturum bilgileri kullanıcı adı ve şifre şeklinde tutulur. Anahtar-değer veritabanı yapısı ile paralellik sağlanarak tasarlanan sistemin hızının artırılması planlanmaktadır. Ayrıca kayıt zamanlarının da yoğunluğu göz önünde bulundurulduğunda öğrenci-ders ikilisinin saklanması da uygun olabilir. Anahtar-değer veritabanı yapısı ile kayıt zamanlarında sistemin kapasitesi artırılarak uyarlanabilir veri boyutu sağlanacak, sistemde beklemeler en düşük seviyeye indirgenecektir. Bu kapsamda kullanılabilecek anahtar-değer veritabanı servislerine Amazon DynamoDB [25] ve Google BigTable [26] üzerinde çalışan MapReduce [27] örnek olarak verilebilir.

Sütun Veritabanı Servisleri ÖBS'de öğrencilere ait ders notları Cassandra [28]'da tutulmaktadır. Satır ve sütunlardan oluşan çizelge şeklinde verilerin tutulabilir ve sütunlar, sütun ailesi olarak adlandırılan gruplara ayrılabilir. Sütun veritabanında her satır bir anahtar içerir ve bu anahtara göre veri getirilir. ÖBS kapsamında sütun veritabanı kullanılarak öğrenci bilgisi ve ders bilgisinin iki sütun ailesinde tutulması düşünülmüştür. Böylece çok kullanılacağı düşünülen, öğrencilerin ders ve sonuç bilgileri bir arada tutularak daha az okuma işlemiyle bu bilgilere erişilebilmektedir.

Çizge Veritabanı Servisleri ÖBS'de kullanılacak çizge veritabanının servis olarak sunması göz önünde bulundurularak Graphenedb [29] seçilmiştir. Neo4j [30]'nin desteklediği biçimde tutulan veriler Graphenedb ile de saklanabilmektedir. REST servisi üzerinden bir bağlantı adresi, kullanıcı adı ve şifre ile saklanan verilere erişilir. Ayrıca mevcut çizge yapısı bir arayüz ile sorgulanabilmekte, gezilebilmekte ve güncellenebilmektedir. Çizgenin görsel temsili de kullanıcılar açısından anlaşılmasına ve gezilebilmesine olanak sağlamaktadır.

4 Sonuç

Bu çalışmada büyük veri gereksinimlerine uygun veri modellerinin mimari yaklaşımları ele alınmıştır. Aynı anda birden çok mimariye olan gereksinime değinilmiş ve örnek bir durum çalışması oluşturulmuştur. Bu durum çalışması için melez bir veri modeli mimarisi önerilmiştir. ÖBS örnek mimarisinin gerçekleştiriminde tercih edilebilecek büyük veri teknolojileri ve bunları servis olarak sunan çözümler de ele alınmış ve bunların ileride hibrit modelin gerçekleştiriminde ne şekilde kullanılacağına ışık tutulmuştur.

İşlevsel fonksiyonlar kapsamlı bir şekilde gerçekleştirilmek istense de tüm veri erişim senaryoları tahmin edilemeyebilir. Bu çalışmada sunulan tasarımda istisna durumlar göz ardı edilebileceğinden, gerçekleştirim aşamasında çok çeşitli kalıcılık önemli hale gelmektedir. Yönetim modelinin düzgün bir şekilde gerçekleştirilmesi ve bu modeli kullanan bir veri yönetim katmanının uygulama bazında oluşturulması büyük önem taşımaktadır. Veri yönetim katmanı sayesinde farklı veri modelleri arasında iletişim ve tutarlılık sağlanıp ihtiyaca yönelik uygun veri modeli yaklaşımından faydalanan bir veri yönetim sistemi gerçeklenmiş olacaktır. İlerleyen aşamada önerilen mimarinin gerçekleştirimi yapılacak ve okuma-yazma sorgu performansları mevcut veri modelleriyle karşılaştırılacaktır.

References

1. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.
2. Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

3. James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers, and McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity. 2011.
4. Jules J. Berman. *Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.
5. Douglas Laney. 3d data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6, 2001.
6. Rick Cattell. Scalable sql and nosql data stores. *SIGMOD Rec.*, 39(4):12–27, May 2011.
7. Pramod J. Sadalage and Martin Fowler. *NoSQL distilled : a brief guide to the emerging world of polyglot persistence*, chapter Polyglot Persistence, pages 133–134.
8. Srikrishna Prasad and MS Nunifar Sha. Nextgen data persistence pattern in healthcare: Polyglot persistence. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–8. IEEE, 2013.
9. E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.
10. Nitin Sawant and Himanshu Shah. *Big Data Application Architecture Q&A: A Problem - Solution Approach*. Apress, Berkely, CA, USA, 1st edition, 2013.
11. Michael Stonebraker and Rick Cattell. 10 rules for scalable performance in 'simple operation' datastores. *Commun. ACM*, 54(6):72–80, June 2011.
12. Solving the nosql data modeling dilemma. `http://blog.foundationdb.com/video-recap-solving-the-nosql-data-modeling-dilemma`, (2015).
13. Foundationdb. `https://foundationdb.com`, (2015).
14. Arangodb. `https://www.arangodb.com`, (2015).
15. Meta-Object Facility (MOF). `http://www.omg.org/mof/`, (2015).
16. Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pages 108–117. Citeseer, 2001.
17. Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, September 2006.
18. Amazon RDS Documentation. `http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html`, (2015).
19. Amazon RDS. `http://aws.amazon.com/rds/`, (2015).
20. Microsoft SQL Azure. `http://azure.microsoft.com/tr-tr/services/sql-database/`, (2015).
21. Google Cloud SQL. `https://cloud.google.com/sql/`, (2015).
22. mongoDB. `https://www.mongodb.org/`, (2015).
23. Amazon Elastic Compute Cloud (EC2). `http://aws.amazon.com/ec2/`, (2015).
24. Google Cloud Platform. `https://cloud.google.com/`, (2015).
25. Amazon DynamoDB. `http://aws.amazon.com/dynamodb/`, (2015).
26. Google Big Table. `https://cloud.google.com/bigtable/`, (2015).
27. Hadoop Map Reduce. `http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html`, (2015).
28. Apache Cassandra. `http://cassandra.apache.org/`, (2015).
29. Graphenedb. `http://www.graphenedb.com/`, (2015).
30. Neo4j. `http://neo4j.com/`, (2015).