Servis Tabanlı Bir Melez Veri Erişim Mimarisi Önerisi

Nail Diker[1], Görkem Giray[2], Murat Osman Ünalır[3]

[1] Logo Yazılım San. ve Tic. A.Ş., İzmir
nail.diker@logo.com.tr
[2] gorkemgiray@gmail.com
[3] Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
murat.osman.unalir@ege.edu.tr

Özet. Günümüzde birçok olayı, durumu, etkileşimi verileştirmekteyiz. Verileştirme sonucunda devasa veri depoları ortaya çıkmaktadır. Bu eğilimle birlikte "büyük veri" kavramı ortaya çıkmış ve verinin hacmi, hızı, çeşitliliği gibi boyutlarının geleneksel yöntemlerden farklı ele alınması gerekliliğini ortaya koymuştur. Farklı özelliklere sahip verinin farklı gereksinimler doğrultusunda farklı şekillerde saklanması önemli bir problemdir. Bu çalışma kapsamında, bu veri çeşitliliği problemine çözüm önermek amacıyla bir veri yönetimi üst modeli ve servis tabanlı bir veri erişim mimarisi önerilmiştir. Bu mimarinin veri çeşitliliği problemini nasıl çözebileceği konusunda bir örnek verilmiştir.

Anahtar kelimeler: Büyük veri, veri yönetim üst modeli, melez veri, melez veri erişim mimarisi, veri çeşitliliği

1 Giriş

Son birkaç on yılda, veriyi işleyebilen ve saklayabilen makinelerin hayatımızdaki önemi oldukça artmıştır. Artık belirli bir andaki ruh durumumuzu bir sosyal ağ uygulamasında, düzenlediğimiz bir faturanın bilgilerini bir muhasebe programında, dağıtım yapan araçların gün içinde izledikleri rotayı sayısal bir harita üzerinde, finansal bilgilerimizi bankamızın bilgi sistemlerinde oluşturabilmekte, işleyebilmekte ve saklayabilmekteyiz. Kısacası artık birçok olayı, durumu, etkileşimi verileştirmekteyiz. Verileştirme sonucunda devasa veri depoları ortaya çıkmaya başlamıştır ve bu depoların hem kapasitesi hem de içindeki veri miktarı büyümeye devam etmektedir. Bu eğilimin iyice arttığı son yıllarda büyük veri kavramı artık birçok insan tarafından bilinen ve üzerinde düşünülen bir kavram olmuştur.

Büyük verinin beraberinde getirdiği en çok bilinen sorunlar hacim ("volume"), hız ("velocity") ve çeşitlilik ("variety") olarak sıralanabilir. Hacim sorunu verilerin fiziksel olarak saklanmaları için gerekli saklama kapasitesinin büyüklüğü ile ilgilidir. Hız sorunu ise büyük hacimli verilerin kabul edilebilir sürelerde analiz edilerek gerekli çıkarımların yapılabilmesiyle ilgilidir. Çeşitlilik sorunu verinin çok sayıda kaynakta ve farklı biçimlerde (yapısal ve yapısal olmayan) bulunmasını irdelemektedir. Farklı yapılara sahip ve yapısal olmayan verilerin saklanmasının,

birleştirilmesinin ve analiz edilmesinin zorluğu çeşitlilik başlığı altında incelenmektedir.

Paradigm4'un Temmuz 2014'te yaptığı bir araştırmada [1] katılımcıların %71'i büyük verideki çeşitliliğin hacimden daha büyük bir problem olduğunu belirtmişlerdir. Aynı araştırmada katılımcıların %91'i büyük veri ile ilgili bir çalışma içinde olduklarını veya gelecek bir iki yıl içerisinde çalışmaya başlamayı planladıklarını belirtmişlerdir. Bu araştırma, kurumların büyük veri kavramına ilgisini ortaya koymaktadır. Aynı araştırmada katılımcıların %49'u veriyi ilişkisel modele uygun hale getirmekte zorlandıklarını ifade etmektedir.

Veri hacminin sürekli artışıyla birlikte, ilişkisel veritabanı yönetim sistemlerinin (VTYS) analiz gereksinimlerine bazı durumlarda cevap veremediği gözlemlenmektedir [2]. Diğer taraftan ilişkisel VTYS'lerin yıllardır çok sayıda sistemde çalıştığı ve oldukça olgunlaştığı söylenebilir. Bundan dolayı verinin saklanması ve işlenmesi için ilişkisel VTYS'leri dışlamak yerine farklılaşan gereksinimlere göre farklı modelleri benimseyen VTYS'leri kullanmak mantıklı görünmektedir. Bir model yerine daha fazla sayıda model kullanmak bir taraftan gereksinimleri karşılamak açısından avantaj getirecekken diğer taraftan ise kullanım açısından zorluklar getirecektir. Bu zorluğu yönetmek için farklı modellere sahip VTYS'lere erişimi birleştirilmiş bir arayüzle sağlamak geliştiriciler ve kullanıcılar için oldukça önemlidir.

Bildirinin ikinci bölümünde bu çalışmaya temel sağlayan bağlam ve ilgili çalışmalar özetlenmiştir. Üçüncü bölümde, melez veri yönetimi üst modeli ve bu modelin tasarım süreci açıklanmıştır. Bunun yanında melez veri yönetimi üst modelini kullanan bir melez veri erişim mimarisi önerisi sunulmuştur. Dördüncü bölümde bir problemin çözümü için melez veri erişim mimarisinin kullanımı kavramsal olarak anlatılmıştır. Beşinci bölümde sonuçlar ve gelecek çalışmalar paylaşılmıştır.

2 Bağlam ve İlgili Çalışmalar

Farklı VTYS'ler farklı problemleri çözmek için tasarlanmıştır [3]. Bir VTYS'nin temelinde yatan model (ilişkisel model gibi) o sistemin yeteneklerini de belirli ölçüde belirlemektedir. Tüm gereksinimler için aynı VTYS'yi kullanmak genelde tatmin edici olmayan bir çözümle karşılaşmaya yol açmaktadır [3]. İşlemsel verinin saklanması, oturum bilgisinin önbellekte saklanması, bir müşterinin ve arkadaşlarının aldığı ürünleri içeren bir çizgenin dolaşılması farklı problemlerdir [3]. Farklı gereksinimler doğrultusunda veri saklama problemine melez bir yaklaşımla çözüm bulmaya çok çeşitli kalıcılık (polyglot persistence) denilmektedir [3].

Büyük veri ile birlikte hacim, hız ve çeşitlilik sorunları göz önüne alındığında ilişkisel VTYS'ler esneklik ve ölçeklenebilirlik açısından bazı gereksinimleri karşılamakta yetersiz kalmaktadır. Bundan dolayı büyük veri için ilişkisel olmayan VTYS'lerin kullanımının yaygınlaşması kaçınılmaz olmuştur. İlişkisel olmayan VTYS'ler farklı gerçekleştirimlere sahiptir ve genel olarak dört ana başlık altında sınıflandırılabilir [3]: (1) Anahtar/değer tabanlı VTYS'ler, (2) Doküman tabanlı VTYS'ler, (3) Sütun tabanlı VTYS'ler, (4) Çizge tabanlı VTYS'ler.

Her bir sınıfta farklı kullanım amaçlarını ve farklı sorunları irdeleyen ilişkisel olmayan VTYS'ler bulunmaktadır. Anahtar/değer tabanlı VTYS'ler için Redis [4] ve Riak [5], doküman tabanlı VTYS'ler için MongoDB [6] ve CouchDB [7], sütun tabanlı VTYS'ler için Cassandra [8] ve MonetDB [9], çizge VTYS'ler için Neo4J [10] ve GraphDB [11] örnek olarak verilebilir. İlişkisel VTYS'lere örnek olarak ise Microsoft SQL Server, Oracle, MySQL ve PostgreSQL verilebilir. VTYS'lerdeki ve her bir sınıftaki ürün çeşitliliği, veri saklama ve işleme için önemli bir zorluk oluşturmaktadır.

ODBAPI, ilişkisel ve ilişkisel olmayan veritabanlarında CRUD (Create, Read, Update, Delete) işlemlerini işletmek için birleştirilmiş bir REST uygulama programlama arayüzüdür [12]. ODBAPI kullanılarak farklı VTYS'ler aynı şekilde sorgulanabilmektedir. ODBAPI, sorgu dilleri arasındaki farklılıkları, sorguları dönüştürerek gidermektedir. Bu çalışmada önerilen veri erişim mimarisinde ODBAPI'den farklı olarak çok çeşitli kalıcılık da desteklenmektedir.

[13] çalışmasında ilişkisel olmayan veritabanlarındaki veri üçlüler halinde ilişkisel veritabanında saklanmaktadır. Böylece veriye, hem ilişkisel modelle hem de ilişkisel olmayan modelle saklanmasından bağımsız olarak, SQL sorgu diliyle erişilebilmektedir. Bu çalışmanın olumsuz tarafı ilişkisel veritabanlarının esneklik ve ölçeklenebilirlik problemlerine bir çözüm oluşturmamasıdır. Aynı zamanda verinin dönüştürülerek ilişkisel veritabanında saklanması senkronizasyon ve veri güncelliği problemlerini beraberinde getirmektedir.

[2] çalışmasında büyük verinin ilişkisel modelle ve geleneksel yöntemlerle nasıl uzlaştırılabileceği üzerinde durulmaktadır. Performans problemlerini gidermek için bellek veritabanı, paralel işletim gibi yöntemler kullanılmaktadır. Bu çalışmada da farklı modellerle saklanan veriye birleştirilmiş bir arayüz ile erişim çözümü bulunmamaktadır.

Nesneye yönelik programlama dilleriyle VTYS'ler arasında köprü görevi gören Object-Relational Mapping (ORM) ve Object-Graph Mapping (OGM) tekniklerini hayata geçiren çeşitli ürünler bulunmaktadır. Bu ürünlere örnek olarak Hibernate ORM [14], Hibernate OGM [15] ve DataNucleus [16] verilebilir. Bu ürünlerin kullanıldığı sistemlerde veritabanı şemasında herhangi bir değişiklik olduğunda değişikliğin geliştirici tarafından yapılması ve kodun tekrar derlenmesi gerekmektedir. Bu çalışmadaki yaklaşım, kodu derlemeye gerek kalmadan şemadaki değişiklikleri devreye almaya olanak sağlamaktadır. Ayrıca Hibernate ORM sadece ilişkisel VTYS'lerle, Hibernate OGM sadece ilişkisel olmayan VTYS'lerle kullanılabilmektedir; bu çalışmadaki yaklaşım ise hem ilişkisel hem de ilişkisel olmayan VTYS'leri kapsamaktadır. DataNucleus hem ilişkisel hem de ilişkisel olmayan VTYS'leri desteklese de bir nesnenin farklı VTYS'lerde saklanması ve bu nesnenin sorgulanması için gereken işlemlerin geliştirici tarafından yapılması gerekmektedir. Bu çalışmadaki yaklaşımda ise bu işlemler geliştirici için saydamdır.

Tozer, genel olarak üst veri ve üst model kullanımının faydalarını üç başlık altında özetlemiştir [17]: (1) kararlılığı sağlayan sağlamlık, (2) basitleştirmeyi sağlayan ortak bir yapı, (3) verimliliği sağlayan yaklaşım tutarlılığı. Tozer, veri yönetimi için bir üst model tanımlamıştır [17]. Bu çalışmada Tozer'in veri yönetimi üst modelinden alınan kavramlar kullanılmıştır.

[18] çalışmasında ilişkisel ve ilişkisel olmayan farklı VTYS'lerde verinin saklandığı bir öğrenci bilgi sistemi için bir büyük veri modeli önerilmiştir. Bu büyük

veri modeli sadece öğrenci bilgi sistemi alanına yöneliktir; bu çalışmadaki model ise ilişkisel ve ilişkisel olmayan VTYS'lere ait kavramları kapsayacak şekilde genişletilebilir bir veri yönetimi üst modeli sunmaktadır.

3 Melez Veri Yönetimi Üst Modeli ve Melez Veri Erişim Mimarisi

Bu bölümde, verinin farklı şekillerde saklanması için önerilen çözümlerin kısıtlarını ortadan kaldıracak bir melez veri erişim mimarisi önerilmiştir. Bu mimari içinde önemli bir yere sahip olan melez veri yönetimi üst modeli hakkında bilgi verilmiştir ve bu üst modelin tasarım süreci özetlenmiştir.

3.1 Melez Veri Yönetimi Üst Modeli

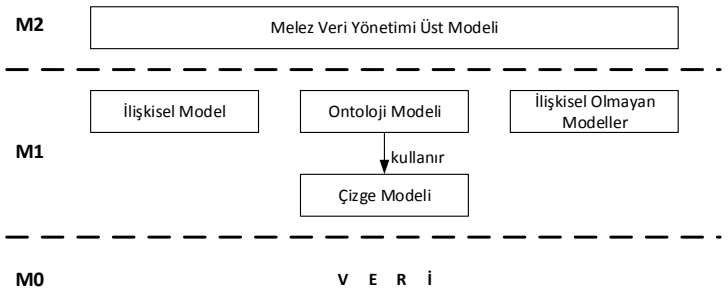
Tüm VTYS'lerde veri saklama yaklaşımları farklılık gösterse de soyut ve genel bir bakış açısıyla bakıldığında bazı ortak noktalar bulunmaktadır. Örneğin, Tablo 1'de ODBAPI projesindeki [12] veri yönetimi üst modeli gösterilmektedir. Bu modeldeki kavramlara bakıldığında farklı VTYS'lerdeki kavramların ortak genel kavramlarla karşılanabildiği görülmektedir.

Tablo 1. ODBAPI'deki kavramlar

İlişkisel Kavramlar	CouchDB Kavramları	Riak Kavramları	ODBAPI Kavramları
Database	Environment	Environment	Database
Table	Database	Database	Entity Set
Row	Document	Key/Value	Entity
Column	Field	Value	Attribute

ODBAPI projesinde, bir *entity set*'in farklı sistemlerde farklı yapıların birleşimlerinden oluşması durumu değerlendirilmemektedir. Bir başka deyişle ODBAPI veri üst modeli çok çeşitli kalıcılığı desteklememektedir. Çok çeşitli kalıcılığa bir örnek olarak, bir ürüne ait kod, açıklama, tedarikçi gibi bilgiler ilişkisel bir veritabanında saklanırken ürüne ait teknik çizimlerin bir doküman tabanlı veritabanında saklanması verilebilir.

Bu çalışma kapsamında tasarlanan melez veri yönetimi üst modelinin, modelleme hiyerarşisindeki yeri Şekil 1'de gösterilmektedir. M0 seviyesinde saklanan veri, M1 seviyesinde ise bu verinin saklandığı modele (ilişkisel, ilişkisel olmayan gibi) göre değişen üst verisi bulunmaktadır. M2 seviyesinde ise model farklılığından bağımsız olarak tasarlanmış olan melez veri yönetimi üst modeli bulunmaktadır.



Şekil 1. Melez veri yönetimi üst modelinin modelleme hiyerarşisindeki yeri

Melez veri yönetimi üst modelinde ilişkisel ve ilişkisel olmayan modeller tek bir üst model altında toplanmıştır. Doküman, sütun ve anahtar/değer tabanlı veritabanları ilişkisel olmayan modeller kapsamında yer almaktadır. Çizge tabanlı veritabanları ise ayrı ele alınmıştır. Bunun nedeni çizge tabanlı dışındaki ilişkisel olmayan modellerin aslında anahtar/değer sistemlerinin temellerini takip ederek geliştirilmiş olmasıdır. Melez veri yönetimi üst modeli kullanılarak farklı modelleri kullanan VTYS'lerin bir üst seviyede yönetilmesi hedeflenmiştir. Böylece M2 seviyesinde tüm veri saklama araçları için birleştirilmiş bir arayüz sunulması hedeflenmiştir.

3.2 Melez Veri Yönetimi Üst Modelinin Tasarımı

Melez veri yönetimi üst modelinin esnek ve genişleyebilir bir yapıda olması gerekmektedir. Değişen gereksinimler doğrultusunda, bundan önce de olduğu gibi, gelecekte farklı veri saklama yaklaşımları geliştirilecek ve bu yaklaşımların yaygınlaşmasına paralel olarak melez veri yönetimi üst modelinin bu yaklaşımları da kapsayacak şekilde genişletilmesi söz konusu olacaktır.

Melez veri yönetim üst modeli için Tozer'in geliştirdiği veri yönetimi üst modeli [17] temel alınmıştır. Tozer'in üst modeli sadece ilişkisel ve nesne tabanlı modelleri kapsamaktadır. Bu çalışma kapsamında bu üst model ilişkisel olmayan modelleri de destekleyecek şekilde genişletilmiştir.

Melez veri yönetim üst modelinin tasarım sürecinde öncelikle farklı veri saklama yaklaşımları temel alınarak tasarlanmış VTYS'lerin üst veri yapıları oluşturulmuştur. Bu kapsamda ilişkisel, anahtar/değer, doküman, sütun ve çizge tabanlı VTYS'lerin ortak özelliklerini temsil edecek *Datastore* kavramı tanımlanmıştır.

Datastore temel kavramının her bir VTYS için özelleşmiş hallerini temsil edecek kavramların oluşturulmasıyla tasarıma devam edilmiştir. Örneğin, en yaygın olarak kullanılan ilişkisel VTYS'lerde ortak olarak kullanılan kavramlar belirlenip melez veri yönetim üst modeli zenginleştirilmiştir: *Veritabanı*, *Tablo*, *Satır*, *Sütun*, *Birincil Anahtar*, *Yabancı Anahtar*.

Kullanımı giderek yaygınlaşan doküman tabanlı VTYS'ler için de aynı şekilde ortak kavramlar belirlenmiştir: *Veritabanı*, *Koleksiyon*, *Doküman*, *Alan*, *Değer*.

Her bir VYTS'yi temel alarak tasarlanmış kavramlar belirli bir özellik kümesine kadar ortaklaşmayı sağlamaktadır. Fakat bu yaklaşımlar kullanılarak geliştirilmiş ürünler temelinde değişen özelliklerin de melez veri yönetim üst modelinde

kapsanması gerekmektedir. Örneğin, ilişkisel model kullanılarak geliştirilmiş olan Microsoft SQL Server, veritabanı işlemlerini yönetmek için "Transact SQL" (T-SQL) sorgu dilini kullanmaktadır. T-SQL, SQL standardının üzerine birtakım gelişmiş özelliklere sahiptir. Oracle ise "Procedural Language/SQL" (PL/SQL) kullanarak ilişkisel modelin özelliklerini desteklemeyi tercih etmiştir.

Doküman tabanlı VTYS'ler için ise MongoDB ve CouchDB incelendiğinde her ikisinde de *dokümanlar* alanlardan oluşmakta ve her bir alanın kendine özgü değerleri bulunmaktadır. MongoDB'de *dokümanlar*, *koleksiyonlar* içerisinde saklanmaktadır. *Koleksiyonlar* ise *veritabanları* kapsamında yer almaktadır. Oysaki CouchDB'de bu tarz bir yapı bulunmamaktadır ve *dokümanlar* doğrudan *veritabanı* seviyesinde saklanmaktadırlar (Bkz. Tablo 2).

Anahtar değer ikililerini barındıran ve ilişkisel olmayan VTYS'nin en temel halini temsil eden sistemlerde de ilişkisel ve doküman tabanlı VTYS'lere benzer ürünlere özelleşmiş davranışlar ve yetenekler mevcuttur. Örneğin Riak değer olarak ne türde bir veri saklandığı ile ilgili hiçbir bilgi barındırmazken Redis daha yapısal bir yaklaşım benimseyerek değer türü olarak belirli tiplere izin vermektedir.

Ürünler arasında veri yönetimi açısından işlevsel olmayan farklılıklar da mevcuttur. Riak ve Redis ürünleri bu anlamda incelendiğinde Riak'ın Microsoft Windows işletim sistemleri üzerinde çalışamadığı, Redis'in ise bu işletim sistemini de desteklediği görülmektedir. Bu anlamda doküman VTYS'leri için ise CouchDB'nin CRUD işlemleri için REST ("REpresentational State Transfer") uyumlu olarak hazır bir hizmet sunarken, MongoDB'nin böyle bir hazır hizmeti bulunmamaktadır.

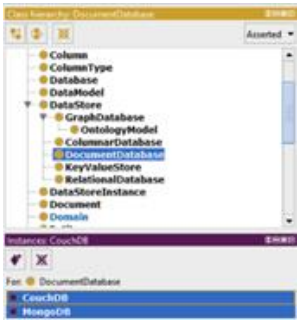
Tasarım sürecinin sonucunda, Tozer'in veri yönetim üst modeli, tüm bu örneklerden hareketle ürünlere özgü bu farklılıkları kapsayacak şekilde genişletilmiştir. Elde edilen melez veri yönetim üst modeli Tablo 2'de gösterilmektedir.

Tablo 2. Melez veri yönetim üst modelindeki kavramlar

Melez Veri Yönetim Üst Modeli						
İlişkisel VTYS		Doküman Tabanlı VTYS		Anahtar/Değer VTYS		Datastore
MS SQL Server	Oracle	MongoDB	CouchDB	Redis	Riak	Datastore individual
Database	Database	Database	Database	Instance	Instance	Data Model
Table	Table	Collection	-	Database	Database	Entity
Row	Row	Document	Document	Key/Value Pair	Key/Value Pair	Entity Occurence
Column	Column	Field	Field	Key	Key	Attribute
Column Value	Column Value	Field Value	Field Value	Value	Value	Attribute Value
Column Type	Column Type	Field Type	Field Type	Value Type	-	Attribute Type

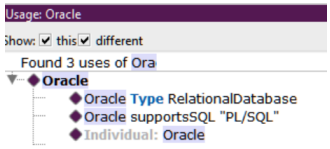
Tablo 2'de en sağ kolonda yer alan kavramlar Tozer'in üst modelinde bulunan kavramlardır. Bunların ilişkisel ve ilişkisel olmayan VTYS'lerde ürün düzeyindeki kavramsal karşılıkları tabloda belirtilmiştir. Tabloda yer alan kavramlar dışında üst model *Datastore* kavramı ile zenginleştirilmiştir. *Datastore* kavramını özelleştiren kavramlar ise veri saklama yaklaşımlarına göre tasarlanmıştır. Buna göre, *RelationalDatabase*, *DocumentDatabase*, *KeyValueStore* ve *GraphDatabase*

kavramları, *GraphDatabase* kavramının özelleşmiş hali olarak da *OntologyModel* kavramı oluşturulmuştur. Bu kavramlar, ontoloji düzeyinde sınıflar olarak tanımlanmıştır. Sınıflar ve hiyerarşik yapı Şekil 2'de gösterilmiştir.



Şekil 2. Ontolojideki sınıflar ve hiyerarşik yapı (kısmi)

Tablo 2'de ismi yer alan VTYS'ler (MS SQL Server, Oracle, MongoDB, vs.) ise takip ettikleri veri saklama yaklaşımına ait sınıfların birer "individual"ı olarak temsil edilmektedirler. Şekil 2'de *DocumentDatabase* sınıfının örnekleri olarak MongoDB ve CouchDB "individual"ları gösterilmektedir. Veri saklama yaklaşımlarını temsil eden sınıfların kendilerine özgü veri ve nesne özellikleri bulunmaktadır. Örneğin, ilişkisel bir VTYS olan Oracle'ın PL/SQL yapısal sorgulama dilini desteklediği bilgisi, ontolojide Şekil 3'te gösterildiği gibi ifade edilmiştir.

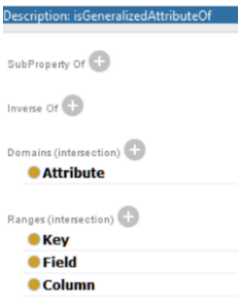


Şekil 3. Oracle VTYS'nin PL/SQL sorgulama dilini desteklediğinin ontolojide ifade edilmesi

Tablo 2'deki ürün ismi ile isimlendirilen kolonlarda yer alan kavramlar ise ontolojide sınıf olarak oluşturulmuştur. Bu sınıfların veri saklama yaklaşımları ile ilişkileri nesne özellik tanımları ile sağlanmıştır. Örnek olarak ilişkisel model için olan kavramlar Şekil 4'te gösterilmiştir.

Melez veri üst modelindeki kavramlar ile veri saklama yaklaşımlarında yer alan kavramların eşlenmesi için *isGeneralizedOf* nesne özelliği kullanılmaktadır. Bu nesne özelliği alanı melez veri üst modelindeki kavramları, aralığı ise veri saklama yaklaşımındaki kavramları temsil eder. Örnek olarak, Şekil 5'te *Attribute* sınıfı eşleşmeleri gösterilmiştir. Örneği detaylandırırsak *Attribute* sınıfı, *Column* sınıfının genel halidir (*isGeneralizedAttributeOf*). Bir önceki örnekte ise *Column* sınıfının *RelationalDatabase* sınıfına ait bir kavram olduğu görülmekteydi. Bu iki bilgi üzerinden basit bir çıkarsama ile ilişkisel veri tabanı yönetim sistemlerinde sütunların özellikleri temsil ettiklerini söyleyebiliriz. Bu çıkarsama işleminin mevcut çıkarsama motorları ile yapılabilmesi için de veri yönetimi üst modelini temsil etmek için ontoloji kullanılmıştır.

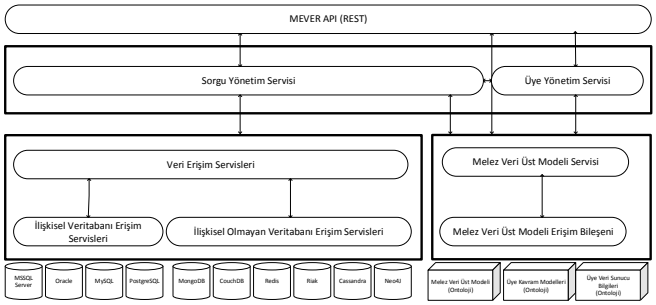
Şekil 4. İlişkisel model için ontolojide tanımlanan kavramlar



Şekil 5. *Attribute* sınıfı eşleşmeleri

3.3 Bir Melez Veri Erişim Mimarisi Önerisi

Bir önceki bölümde açıklanan melez veri yönetim üst modelini gerçekleyen bir mimari Şekil 6'da gösterilmektedir.



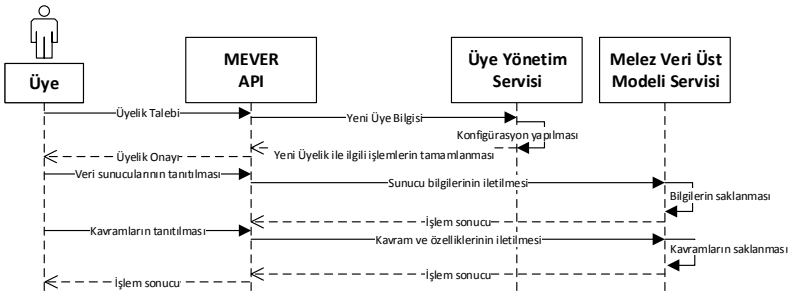
Şekil 6. Melez veri erişim mimarisi

Mimarinin bileşenlerinin sorumlulukları şu şekildedir:

- ***MEVER (Melez Veri Erişim Çerçevesi) API***
  Sistemin dışa açılan yüzünü temsil etmektedir. Sistem işlevlerinin tümü bu API üzerinden sunulmaktadır. Bu işlevler, (1) Yeni üyelik işlemi; (2) Üye VTYS'lerin tanımlanması; (3) Üye kavramlarının tanımlanması; (4) Kavram sorgulama; (5) Önerilerin alınması olarak sıralanabilir.
- ***Sorgu Yönetim Servisi***
  Gelen sorguların iletildiği servistir. Melez veri modeli servisi ile haberleşerek üyeye özel kavram tanımı, özellikleri ve özelliklere karşılık gelen veri modeli düzeyindeki bilgileri alır. Sorguyu kavramsal düzeyden veri erişim servislerinde çalışacak VTYS sorgularına dönüştürmekle görevlidir. Bu görevinin yanı sıra veri erişim servislerinden dönen sonuç kümelerine göre sorgu cevabını oluşturacak kavram ya da kavramları oluşturur. Oluşturduğu kavram ya da kavramları MEVER API'ye iletir.

- ***Üye Yönetim Servisi***
  Yeni üye kayıt işlemleri, üyeye ait VTYS ve kavram tanımlarının alınması ve bu bilgilere erişilmesinden sorumludur. Bir kavram sorgusu yapıldığında üyenin bilgilerini kontrol eder.
- ***Veri Erişim Servisleri***
  Üyeye ait VTYS'ler ile doğrudan veya VTYS servisleri üzerinden kendisine iletilen sorguları çalıştırır ve sonuç kümelerini geri döndürür. İlişkisel ve ilişkisel olmayan VTYS'leriyle ilgili servisler aracılığıyla haberleşmektedir.
- ***Melez Veri Üst Modeli Servisi***
  Melez veri modeline, üye kavram ve veri sunucusu bilgilerine erişimi yöneten servistir.
- ***Üye Kavram Modelleri***
  Üyelere ait kavramları Melez Veri Modeline uygun şekilde saklayan ontolojilerdir.
- ***Üye Veri Sunucu Bilgileri***
  Üyelere ait VTYS sunucuların erişim yöntemlerini Melez Veri Modeline uygun şekilde saklayan ontolojilerdir.

Melez veri erişim mimarisinin dinamik davranışını tanımlamak için Şekil 7'deki ve Şekil 8'deki sıra diyagramları çizilmiştir. Şekil 7'te yeni üye kaydetme senaryosunun sıra diyagramı gösterilmektedir. MEVER API'ye gelen yeni üye bilgileri üye yönetim servisi tarafından kaydedilmektedir ve kullanıcıya üyeliğin tamamlanması onayı gönderilmektedir. Sonrasında veri sunucuları sisteme tanıtılmaktadır. Bunun için MEVER API melez veri modeli servisini kullanmaktadır. Veri sunucuları sisteme tanıtıldıktan sonra kavramlar ve özellikleri melez veri modeli servisi tarafından kaydedilmektedir. Böylece üyelik işlemleri sonlandırılmaktadır.

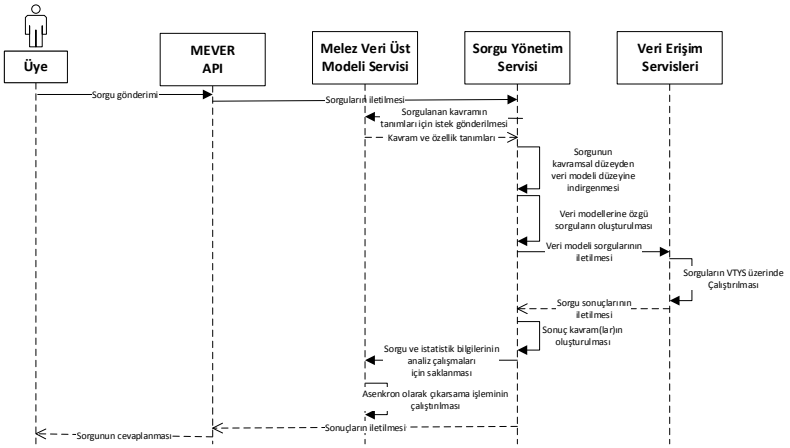


Şekil 7. Yeni üye kaydetme senaryosunun sıra diyagramı

Şekil 8'de sorgu çalıştırma senaryosunun sıra diyagramı gösterilmektedir. MEVER API'ye gelen sorgular, sorgu yönetim servisi tarafından kavramsal düzeyden (Şekil 1'deki M2 seviyesinden) veri modeli düzeyine (Şekil 1'deki M1 seviyesine) dönüştürülür. Bu dönüşüm için gerekli bilgiler, Melez Veri Üst Modeli Servisi tarafından sunulan kavramın daha önce üye tarafından tanımı sisteme verilmiş olan veri modeli karşılıklarıdır. Gereksinim duyulan veri modellerine özgü sorgular yine sorgu yönetim servisi tarafından oluşturulur. Sorguların çalışacağı VTYS sunucusu

veya servisinin bilgileri yine Melez Üst Veri Modeli Servisi tarafından sunulur. VTYS sunucu erişim bilgileri ve veri modeli sorguları veri erişim servislerine iletilir. Veri erişim servisleri veriyi bir dönüşüm işlemine tabi tutmadan kendi kaynaklarında gelen sorgu içeriğine göre sorgular. Her bir veri erişim servisi, sorgu sonuçlarını sorgu yönetim servisine iletir. Tüm sonuçlar sorgu yönetim servisine iletildikten sonra sonuç kümeleri üzerinden sorguya cevap olarak verilecek kavramlar oluşturulur. Oluşturulan kavramlar sorgu cevabı olarak MEVER API'ye iletilir.

Yukarıda bahsedilen adımlarda rol alan sorgulanan kavram, sorgu içeriği, oluşturulan veri modeli sorgularının içerikleri, bu sorguların VTYS sunucusu üzerinde çalışma zamanı, dönen sonuç kümesinin büyüklüğü, VTYS sunucularının erişim zamanı gibi bilgiler toplanmaktadır. Bu toplanan bilgiler, Melez Veri Üst Modeli Servisi'ne iletilir. Bunun amacı, bu bilgiler üyeye özgü olarak ontoloji düzeyinde saklanarak, daha sonrasında üyeye veri modeline hatta veri saklama yaklaşımlarına dair öneriler oluşturabilme hedefidir. Örneğin kapsam olarak büyük boyutlu bir dosya içeriği saklayan bir özelliğin ilişkisel modeldeki cevap süresinin beklenen azami süreden uzun olması durumu tespit edilip bunun farklı bir veri saklama yöntemi ile ilişkisel olmayan doküman tabanlı bir sistemde saklanması önerilebilir.



Şekil 8. Sorgu çalıştırma senaryosunun sıra diyagramı

4 Örnek Durum Senaryosu

Melez veri erişim mimarisinin kullanılabileceği çok sayıda gerçek dünya problemi bulunabilir. Artık birçok alanda çok çeşitli verinin farklı gereksinimleri yerine getirecek şekilde saklanması ve bu verinin gerektiğinde bütünleştirilerek istemcilere sunulması gerekmektedir. Böyle bir duruma örnek olarak insan kaynakları alanından bir problem seçilmiştir.

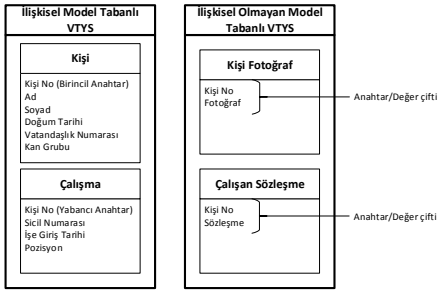
Çoğu firma, faaliyetini sürdürebilmek için çalışanlar istihdam etmektedir. Firmanın, çalışanlarıyla ilgili sakladığı verinin kapsamı firmaların faaliyet gösterdiği

sektörlere ve işleyişlerine göre değişiklik göstermektedir. Genel olarak değişiklik göstermeyen veri için çalışanın adı, soyadı, doğum tarihi, kan grubu, vatandaşlık numarası, fotoğrafı gibi kişiye özgü bilgiler; firmayla olan çalışma ilişkisini temsil eden veri için de sicil numarası, firmadaki pozisyonu, işe giriş tarihi, sözleşmesi ele alınabilir.

Bu veriyi MEVER kapsamında düşündüğümüzde sisteme üye olan firmanın "çalışan" kavramı (Şekil 9) bulunmaktadır. "Çalışan" kavramının, kişiye özgü ve firmada çalışmasından ötürü elde ettiği özellikler bulunmaktadır.

Kavramsal düzeyde çalışan kavramını Şekil 9'daki gibi temsil etmek mümkünken mantıksal düzeyde veri modeline bakıldığında, bu verinin saklandığı VTYS'lerin veri saklama yaklaşımları çeşitlilik gösterebilir. Veri saklama modeli olarak Şekil 10'da gösterildiği gibi bir yöntem izlenmiş olabilir.

Şekil 9. Çalışan kavramı



Şekil 10. Çalışan kavramına ait verinin farklı VTYS'lerde saklanması

"Çalışan" kavramının MEVER sistemine tanıtılabilmesi için öncelikle VTYS'lerin üye firma tarafından tanımlanmış olması gerekmektedir. Bundan sonra "çalışan" kavramı ve özellikleri sisteme tanıtılır. Tanıtma işlemi, kavram özellikleri ve her bir özelliğin mantıksal modelde karşılığı olan VTYS, VTYS kapsamındaki veri tabanı (üst modeldeki *Data Model*), veri tabanı nesnesi (*Entity*) ve bu nesnedeki özellik (*Attribute*) bilgilerini kapsamaktadır.

"Çalışan" kavramına ait bir sorgu MEVER API servisine iletildiğinde ise sorgu, sorgu yönetim servisine iletilir. Yukarıdaki bahsedilen tanımlar Melez Veri Üst Modeli servisi aracılığı ile alınır ve sorgu yönetim servisi her bir VTYS üzerinde çalışacak sorguları hazırlar. Sorgular, verinin saklandığı VTYS'nin tanıdığı sorgu dilinin sözdizimine uygun biçimde oluşturulur. Veri erişim servisleri, sorumluluğunda bulunan VTYS üzerinde sorguyu çalıştırır ve sonuç kümesini sorgu yönetim servisine geri döndürür. Sorgu yönetim servisi, kavram tanımlarını temel alarak sonuç kümelerini bir araya getirir ve böylelikle sorguya cevap olabilecek kavram ya da kavramları oluşturur.

İlgili kavrama ilişkin sorgu, veri erişim servislerinde çalışan sorgular, bu sorgulara ait çalışma zamanları, dönen sonuç kümelerinin büyüklüğü gibi işlevsel özelliğe sahip olmayan istatistiki bilgiler melez veri modeli servisine iletilir. Bu bilgilerin elde edilmesi ve melez veri üst modeli servisi üzerinden ontolojiler olarak saklanmasının amacı çıkarsamalar yapılmasıdır. Çıkarsamalar, işlevsel olmayan ihtiyaçların (cevap dönme hızı, veri hacmi gibi) belirli kural kümelerine göre öneriler oluşturulması için çalıştırılacaktır.

5 Sonuçlar ve Gelecek Çalışmalar

Bu çalışmada veri çeşitliliği problemine çözüm olarak bir melez veri erişim mimarisi önerilmiştir. Günümüzde verinin dağıtık doğası düşünüldüğünde bu mimari servis tabanlı olarak tasarlanmıştır. Veri yönetim üst modeli farklı veri saklama yöntemlerini birleştiren soyut ve genel model olarak M2 seviyesinde konumlandırılmıştır. Mimaride bu üst model ontolojilerle gerçeklenmiştir.

Gelecek çalışmalar kapsamında, ontolojiler üzerinde çıkarsamalar yapılarak veri saklama gereksinimlerine daha iyi cevap verilmesi ve bazı durumlarda geçerleme (ontoloji üzerine kural kümeleri işletilerek) yapılması planlanmaktadır. Tüm VYTS'lerin tek bir sorgu diliyle sorgulanabilmesi için M2 seviyesinde konumlandırılan ontoloji temel alınarak bir sorgu dili geliştirilmesi planlanmaktadır. Ayrıca MEVER'e iletilen sorguların parçalanması, sorguların ilgili VTYS'de işletilmesi ve elde edilen sonuçların birleştirilmesi gerekmektedir. Bunlar yapılırken her bir VTYS'nin kendi işlem yönetiminden (transaction management) bağımsız olarak bir üst seviyede de işlem yönetimi gereksinimi bulunmaktadır.

Kaynakça

1. Paradigm4: Leaving Data on the Table: New Survey Shows Variety, Not Volume, is the Bigger Challenge of Analyzing Big Data, http://www.prnewswire.com/news-releases/leaving-data-on-the-table-new-survey-shows-variety-not-volume-is-the-bigger-challenge-of-analyzing-big-data-265365761.html, (2014), Erişim tarihi: 12 Haziran 2016.
2. Arnold, J., Glavic, B., Raicu, I.: HRDBMS: Combining the Best of Modern and Traditional Relational Databases, Technical Report, (2015)
3. Sadalage, P.J., Fowler, M.: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, 1. Baskı, Addison-Wesley Professional (2012)
4. http://redis.io/. Erişim tarihi: 20 Mayıs 2016.
5. http://basho.com/products/riak-kv/. Erişim tarihi: 20 Mayıs 2016.
6. https://www.mongodb.com/. Erişim tarihi: 20 Mayıs 2016.
7. http://couchdb.apache.org/. Erişim tarihi: 20 Mayıs 2016.
8. http://cassandra.apache.org/. Erişim tarihi: 20 Mayıs 2016.
9. https://www.monetdb.org/Home. Erişim tarihi: 20 Mayıs 2016.
10. http://neo4j.com/. Erişim tarihi: 20 Mayıs 2016.
11. http://ontotext.com/products/graphdb/. Erişim tarihi: 20 Mayıs 2016.
12. Sellami, R., Bhiri, S., Defude, B.: ODBAPI: a unified REST API for relational and NoSQL data stores, Big Data (BigData Congress), 2014 IEEE International Congress on, 653-660 (2014)
13. Roijackers, J.: Bridging SQL and NoSQL, Yüksel Lisans Tezi, Eindhoven University of Technology, Department of Mathematics and Computer Science (2012)
14. http://hibernate.org/orm/. Erişim tarihi: 13 Eylül 2016.
15. http://hibernate.org/ogm/. Erişim tarihi: 13 Eylül 2016.
16. http://www.datanucleus.org/. Erişim tarihi: 13 Eylül 2016.
17. Tozer, G.V.: Metadata Management for Information Control and Business Success, Artech Print on Demand (1999)
18. Ünalır, M.O., İnan, E., Yönyül, B., Olca, E., Şentürk, F., Mostafapour, V., Yıldız, P., Yılmazer, D., İşli, D.: Veri Yoğun Bilgi Sistemleri İçin Melez Bir Veri Mimarisi Önerisi, Ulusal Yazılım Mühendisliği Sempozyumu (2015)