BÜYÜK ÖLÇEKLİ BİLGİ İŞLEM MERKEZLERİNDE SERVİS ODAKLI MİMARİ KULLANARAK ORTAK VERİ TABANI OLUŞTURMA: ATATÜRK ÜNİVERSİTESİ ÖRNEĞİ

Gökhan TUTAR

Yüksek Lisans Tezi
Yönetim Bilişim Sistemleri Anabilim Dalı
Dr. Öğr. Üyesi Serdar AYDIN
2019

T.C.
ATATÜRK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI

Gökhan TUTAR

BÜYÜK ÖLÇEKLİ BİLGİ İŞLEM MERKEZLERİNDE SERVİS ODAKLI MİMARİ KULLANARAK ORTAK VERİ TABANI OLUŞTURMA: ATATÜRK ÜNİVERSİTESİ ÖRNEĞİ

YÜKSEK LİSANS TEZ ÇALIŞMASI

TEZ YÖNETİCİSİ
Dr. Öğr. Üyesi Serdar AYDIN

ERZURUM - 2019

T.C.
ATATÜRK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
TEZ BEYAN FORMU

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE

BİLDİRİM

Atatürk Üniversitesi Lisansüstü Eğitim ve Öğretim Uygulama Esaslarının ilgili maddelerine göre hazırlamış olduğum "**BÜYÜK ÖLÇEKLİ BİLGİ İŞLEM MERKEZLERİNDE SERVİS ODAKLI MİMARİ KULLANARAK ORTAK VERİ TABANI OLUŞTURMA: ATATÜRK ÜNİVERSİTESİ ÖRNEĞİ**" adlı tezin/raporun tamamen kendi çalışmam olduğunu ve her alıntıya kaynak gösterdiğimi taahhüt eder, tezimin/raporumun kâğıt ve elektronik kopyalarının Atatürk Üniversitesi Sosyal Bilimler Enstitüsü arşivlerinde aşağıda belirttiğim koşullarda saklanmasına izin verdiğimi onaylarım:

Lisansüstü Eğitim ve Öğretim Uygulama Esaslarının ilgili maddeleri uyarınca gereğinin yapılmasını arz ederim *.

☐ Tezimin/Raporumun tamamı her yerden erişime açılabilir.

☐ Tezimin/Raporumun makale için **altı ay,** patent için **iki yıl** süreyle erişiminin ertelenmesini istiyorum.

22.07.2019

Gökhan TUTAR

---

* **LİSANSÜSTÜ TEZLERİN ELEKTRONİK ORTAMDA TOPLANMASI, DÜZENLENMESİ VE ERİŞİME AÇILMASINA İLİŞKİN YÖNERGE**

.....

ÜÇÜNCÜ BÖLÜM

Çeşitli ve Son Hükümler

Lisansüstü tezlerin erişime açılmasının ertelenmesi MADDE 6– (1) Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulu iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.

(2) Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulunun gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.

Gizlilik dereceli tezler MADDE 7– (1) Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, tezin yapıldığı kurum tarafından verilir. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, ilgili kurum ve kuruluşun önerisi ile enstitü veya fakültenin uygun görüşü üzerine üniversite yönetim kurulu tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir.

(2) Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir.

T.C.
ATATÜRK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

TEZ KABUL TUTANAĞI

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Dr. Öğr. Üyesi Serdar AYDIN danışmanlığında, Gökhan TUTAR tarafından hazırlanan bu çalışma 22 / 07 / 2019 tarihinde aşağıda isimleri yazılı jüri tarafından Yönetim Bilişim Sistemleri Anabilim Dalı'nda Yüksek Lisans Tezi olarak kabul edilmiştir.

Başkan : Prof. Dr. Üstün ÖZEN İmza: .....

Jüri Üyesi : Doç Dr. Handan ÇAM İmza: .....

Jüri Üyesi : Dr. Öğr. Üyesi Serdar AYDIN İmza: .....

Prof. Dr. Sait UYLAŞ
Enstitü Müdürü

İÇİNDEKİLER

BEŞİNCİ BÖLÜM

ARAŞTIRMA BULGULARI VE TARTIŞMA

ÖZET

YÜKSEK LİSANS TEZİ

BÜYÜK ÖLÇEKLİ BİLGİ İŞLEM MERKEZLERİNDE SERVİS ODAKLI MİMARİ KULLANARAK ORTAK VERİ TABANI OLUŞTURMA: ATATÜRK ÜNİVERSİTESİ ÖRNEĞİ

Gökhan TUTAR

Tez Danışmanı: Dr. Öğr. Üyesi Serdar AYDIN

2019, 130 sayfa

Juri: Prof. Dr. Üstün ÖZEN
Doç. Dr. Handan ÇAM
Dr. Öğr. Üyesi Serdar AYDIN

Dijital gelişmelere bağlı olarak bilgi yönetim sistemlerinin sayısı ve çeşitliliği de artmaktadır. Farklı ihtiyaçlara yönelik farklı bilgi yönetim sistemlerinin olması, bilgiyi yönetmeyi de zorlaştırmaktadır. Bu sorunu çözmenin en ideal yöntemi kurum/kuruluşun bütün ihtiyaçlarını karşılayacak tek bir bilgi yönetim sistemi oluşturmaktır. Ancak maliyet, personel yetersizliği ve zaman sınırlandırması gibi birçok nedenden dolayı tek bir bilgi sistemi oluşturulması mümkün değildir.

Yönetim bilgi sistemleri birbirlerinden bağımsız olsalar bile birbirlerinin verilerine ihtiyaç duymaktadırlar. İhtiyaç duyulan verilerin güvenli bir şekilde paylaşımı için web servisler kullanılır. Bilgi yönetim sistemlerinin çeşitliliği web servislerin sayısını da arttıracaktır. Web servis sayısının fazla olması hem bilgi yönetim sistemlerinin yükünü artıracak hem de web servislerinin yönetimini zorlaştıracaktır.

Bu tezde, ortak bir veri tabanı oluşturularak; bilgi sistemleri arasındaki iletişimi en üst düzeye çıkarmak, web servislerin yönetimini kolaylaştırmak, bilgi sistemlerinin üzerindeki yükü hafifletmek ve veri paylaşımını belirli bir düzen içerisinde yapmak amaçlanmıştır.

Ortak veri tabanı oluşturulması birçok fayda sağlamaktadır. Bunlardan bazıları şunlardır: Anlık güncelleme işlemlerinin yükünü kendi üzerine alarak diğer bilgi sistemlerinin iş yükünü hafifletir; servis odaklı mimari kullanıldığı için diğer bilgi sistemlerinin iş yoğunluklarından etkilenmez; verileri tek bir merkezde toplayarak veri erişimini kolaylaştırır; devlet kurumları tarafından sağlanan web servislerin (KPS, ÖSYM ve YÖKSİS) kullanıcı bilgilerini diğer bilgi sistemleriyle paylaşmadığından web servislerin güvenliğini artırır; web servislerden gelen cevapların sürelerini cache yöntemiyle hızlandırır.

Anahtar Kelimeler: Ortak veri tabanı, web servis, servis odaklı mimari, wsdl

ABSTRACT

MASTER'S THESIS

BUILDING A COMMON DATABASE USING SERVICE ORIENTED ARCHITECTURE IN LARGE SCALE INFORMATION TECHNOLOGIES CENTERS: ATATURK UNIVERSITY SAMPLE

Gökhan TUTAR

Advisor: Assist. Prof. Dr. Serdar AYDIN

2019, Page: 130

Jury: Prof. Dr. Üstün ÖZEN
Assoc. Prof. Dr. Handan ÇAM
Assist. Prof. Dr. Serdar AYDIN

The number and diversity of information management systems are increasing due to digital developments. The existence of different information management systems for different needs makes it difficult to manage information. The ideal way to solve this problem is to create a single information management system that will meet all the needs of the organization. However, it is not possible to establish a single information system for many reasons such as cost, lack of personnel and time limitation.

Even though management information systems are independent of each other, they need each other's data. Web services are used for sharing of needed data securily. The diversity of information management systems will increase the number of web services. The high count of web services will increase the burden of information management systems and make it difficult to manage web services.

In this thesis, by creating a common database; it is aimed to maximize communication between information systems, to simplify the management of web services, to reduce the burden on information systems and to share data in a specific order.

Creating a common database provides many benefits. Some of these are: It relieves the workload of other information systems by taking the burden of instant updates on itself; it is not affected by the workload of other information systems because it uses service-oriented architecture; simplifies data access by collecting data in a single center; enhances the security of web services as the web services provided by government agencies (KPS, OSYM and YÖKSİS) do not share user information with other information systems; speeds up the response time from web services with cache method.

Keywords: Common database, web service, service oriented architecture, wsdl

KISALTMALAR DİZİNİ

AKB : Akıllı kart basım programı
AKK : Akıllı kart kapı geçişleri
AKY : Akıllı kart yemekhane geçişleri
BT : Bilgi Teknolojileri
HGS : Hızlı Geçiş Sistemi
HTML : Hypertext Markup Language
KBS : Kütüphane bilgi sistemi
KPS : Kimlik Paylaşım Sistemi
MERNİS : Merkezi Nüfus İdaresi Sistemi
OVT : Ortak Veri Tabanı
ÖBS : Öğrenci bilgi sistemi
ÖSYM : Ölçme, Seçme ve Yerleştirme Merkezi
RPC : Remote Procedure Call
SOA : Service Oriented Architecture
SOAP : Simple Object Access Protocol
W3C : World Wide Web Consortium
WSDL : Web Services Description Language
XML : Extensible Markup Language
YÖK : Yüksek Öğretim Kurulu

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

ÖNSÖZ

Bu çalışmanın gerçekleştirmesinde bilgi ve deneyimini benden esirgemeyip her türlü desteği veren danışmanım sayın Dr. Öğr. Üyesi Serdar AYDIN'a teşekkürlerimi sunarım.

Ayrıca tez çalışmamda beni yönlendiren ve destek veren Dr. Öğr. Üyesi Ahmet Kamil Kabakuş'a, Bilgisayar Bilimleri Araştırma ve Uygulama merkezi yönetimi, çalışma arkadaşlarıma şükranlarımı sunarım.

Hayatım boyunca yanımda olan ve desteklerini her zaman hissettiğim aileme sonsuz teşekkürlerimi sunar çalışmanın ilgilenenlere faydalı olmasının dilerim.

Erzurum – 2019 **Gökhan TUTAR**

GİRİŞ

Günümüzün gelişen teknoloji ve iletişim çağında, organizasyonlar, iş süreçlerini iyileştirmek ve maliyetlerini düşük tutmak, üretim hızını artırmak ve rekabet güçlerini artırmak amacıyla etkinliklerinin büyük bir kısmını dijital ortamada gerçekleştirmektedir. Bu amaç doğrultusunda sürekli olarak yeni ve farklı bilişim teknolojilerine ihtiyaç duymaktadırlar. Gündelik hayatın vaz geçilmez bir parçası haline gelmiş internet, iş, sağlık, eğitim ve eğlence gibi birçok alanda bilgilere kolay, hızlı ucuz bir şekilde erişilebilmesine imkân sağlamaktadır. İnternet, özellikle iş modellerinin temel bileşenlerinden biri olarak kullanılmakta olup hem kurumlar hem de kullanıcılar, internet tabanlı uygulamalarını yaygın olarak kullanmaktadır. Şirketlerin yanı sıra kamu kurumları ve sosyal platformlar da biz kullanıcılara veya diğer şirket ve kurumlara sunmuş oldukları hizmet ve ürünlerde internet teknolojilerini etkin olarak kullanmaktadırlar.

Dünyada devletler ve büyük şirketler verinin yönetilmesinde bilgi güvenliğini önemli bir strateji olarak benimsemişlerdir. Ülkemizin 2019-2023 stratejik planı doğrultusunda çalışmaları tamamlanan “Dijital Türkiye Yol Haritası” ile tüm sektörlerde dijital dönüşümü sağlamış, kamuda ve özel sektörde kurumsal kaliteyi artırmış bir Türkiye hedeflenmektedir. (Sanayi Bakanlığı – Dijital Türkiye Yol Haritası)

Dijital dönüşümün başarılı bir şekilde gerçekleşmesi ve bunun sürdürülebilirliğinin sağlanması kamu kuruluşları için bilgi işlem merkezlerinin önemini göstermektedir. Kamu kuruluşlarında bilgi işlem daire başkanlıkları bulunması zorunludur ve bu birim kurumun en temel yapıları arasındadır ("Yükseköğretim Üst Kuruluşları ile Yükseköğretim Kurumlarının İdari Teşkilatı Hakkında Kanun Hükmünde Kararname" 1983). Hatta bazı üniversiteler bilgi işlem daire başkanlığının idari yapı olmasından kaynaklanan zorlukları aşmak ve akademik çalışmaların yapılabilmesi için Bilgisayar Bilimleri Uygulama ve Araştırma Merkezlerini kurmuşlardır.

Kamu kurumlarında uzun süren bürokratik işlemlerin hızlandırılması ve reformlar yapabilmenin en hızlı yöntemi bilgi işlem merkezlerini kullanmaktır. Kamu kurumun geniş görüşlülüğünü gerçekleştirmedeki en büyük etkenlerden birisi kurumundaki bilgi işlem merkezinin yeterliliğidir (Cordella ve Tempini 2015).

Kamu kurumlarının hizmet verdiği kitleler farklılık gösterdiğinden dolayı kurumlarda bulunan bilgi işlem merkezlerinin büyüklükleri de farklılık göstermektedir. Bu durum büyük ölçekli bilgi işlemi kavramının doğmasına neden olmuştur. Bilgi işlem biriminin büyük ölçekli bilgi işlem birimi olabilmesi için Maliye Bakanlığı'nın 31 Aralık 2008 tarihinde 27097 sayılı Kamu kurum ve kuruluşlarının büyük ölçekli bilgi işlem biriminde sözleşmeli bilişim personeli istihdamına ilişkin esas ve usuller hakkındaki yönetmeliğin beşinci maddesinde aşağıdaki yedi ölçüte sahip olması gerektiği belirtilmiştir.

1- Kamu kurum ve kuruluşunun görev alanı kapsamında bilgi işlem birimi tarafından internet/intranet ortamında sunulan uygulamalarının fiilen en az beş bin kullanıcısının olması veya bu uygulamalardan faydalanan en az bin hizmet biriminin bulunması ya da il ve ilçelerin en az 1/3 ünde bu uygulamalardan faydalanan birimin bulunması, bu çerçevede internet/intranet ortamında sunulan uygulamaların, veri girişi, verinin değiştirilmesi ve onaylanması, rapor alma ve bilgiye erişim hizmetlerinin sunumuna elverişli olması ve bu uygulamaların fiilen kullanılması,
2- Merkezi internet ve/veya intranet uygulamalarına açık olması ve uygulamaların servis odaklı (diğer farklı sistemlere veri alışverişine müsait) bir mimariyi desteklemesi,
3- Ağ yönetimi ve yazılım hizmetlerinin merkezi olarak sunulması,
4- Ağ yönetimi ve yazılım hizmetlerini istihdam edeceği sözleşmeli bilişim personeli ile yapabilme kapasitesine sahip olması,
5- Ayrı bir fiziki merkezde kurulu olan acil durum merkezinde yedekleme yapabilmesi, felaket durumunda bu merkezin kritik bilgi işlem uygulamalarının devamlılığını sağlayabilmesi ve bu durumda sistemin nasıl devreye alınacağı ile ilgili dokümanların bulunması,
6- Çağrı merkezi ve sesli yanıt sistemine sahip olması, gelen çağrıları kayıt altına alabilmesi, interaktif hizmet verebilmesi ve sesli mesaj alabilmesi,
7- Haftada 7 gün ve günde 24 saat hizmet sunma kapasitesinin bulunması olarak tanımlanmıştır ("Kamu Kurum ve Kuruluşlarının Büyük Ölçekli Bilgi İşlem Birimlerinde Sözleşmeli Bilişim Personeli İstihdamına İlişkin Esas ve Usuller Hakkında Yönetmelik" 2008).

Kurumun internet hizmetlerinin çok çeşitli olması işlenecek verilerin boyutu da o denli büyük ve çeşitli olacağı anlamına gelmektedir. Büyük ve çeşitli verinin yönetilebilmesi için daha büyük bilgi işlem merkezlerine ihtiyaç duyulacağı açıktır. Bilgi sistemlerinin çok sayıda ve farklı çeşitlilikte olması kurumdaki veri giriş sayısının artmasına bu da dijital ortamda bürokratik işlemlerin artması anlamına gelmektedir.

Dijital ortamdaki bürokrasi gerçek hayattakinden daha hızlı olsa da işlemlerini yavaşlatacağı ve gereksiz işlemler olacağı için insanları yoracaktır. Son yirmi yılda dijitalleşme adına birçok örnek bulunmaktadır. Bu dijitalleşmelerden başarılı olanlar kadar başarısız olanlar da bulunmaktadır. Dijitalleştirme yapılırken gerçek hayatta yapılan işlemin tamamen aynısını dijital ortama taşımak pek mümkün değildir çünkü gerçek hayatta vatandaşların karşısında takdir yetkisini kullanabilecek veya vatandaşla birebir iletişime geçerek ona, neyin nasıl yapılması gerektiğini anlatacak, hangi formları doldurmasını ve formlardaki alanlara neler yazmasını gösterebilecek bir memur bulunmaktadır. Bunun yanından vatandaşlar dijital ortama aşinalık gibi bir dizi problem yaşayabilmektedir (Pimenidis, Iliadis, ve Georgiadis 2011).

Dijital bürokrasiye örnek verilmesi gerekilirse; kurumdaki birçok bilgi sistemine aynı verileri tekrar tekrar girilmesini istemek verilebilir. Yani kurumda bilgi işleme sistemi olarak personel bilgi yönetim sistemi ve elektronik belge yönetim sistemi olduğu durumda kişinin hem personel bilgi yönetim sistemine hem de elektronik belge yönetimine ayrı ayrı kayıt olması durumudur. Bu bürokrasi sadece kişi için zaman kaybı değil aynı zamanda kurum için masraf ve iş gücü kaybı anlamına gelmektedir. Çünkü aynı veri iki farklı veritabanlarında barındırılır. Fazladan bakım hizmeti ve veri alanı gerektiren bu durumun en ideal çözümü kurumun bütün ihtiyaçlarını tek bir yazılımda toplamaktır ancak tek bir yazılımın büyük ölçekli bilgi işlem merkezine sahip bir kurumun bütün ihtiyaçlarını karşılayabilmesi pek olası değildir. Büyük ölçekli bilgi işlem merkezleri genellikle kendi yazılımlarını üreten birimlerdir ancak bilgi işlem merkezleri ne kadar büyük olsa da bazı durumlarda yazılımları dışardan temin etmek zorunda kalabilirler. Bu durum farklı yazılımlar için farklı sistemleri kurmaya neden olacaktır.

Kurum tarafından geliştirilmemiş dışardan satın alınan yazılımın en büyük sorunu bilgi işlem bünyesindeki hali hazırda çalışan yazılımlar ile iletişime geçmesidir çünkü

farklı yazılımlar, farklı kodlamalar, farklı veri yapılarının bir noktada eşleşmesi gerekir. Bu tezde büyük ölçekli bilgi işlem merkezlerinde barındırılan kurum içerisinde geliştirilen ve kurum dışından satın alınan yazılımların servis odalık mimari ile ortak bir veri tabanı oluşturarak iletişime geçmesi planlanmış, uygulanmış ve model anlatılmıştır.

Birinci bölümde servis odaklı mimari kavramından ve servis odaklı mimarinin faydalarından bahsedilmektedir. Ayrıca bu bölümde servis odalı mimari ile yakından ilgisi olan web servisler ve web servislerin parçaları hakkında bilgi verilmektedir.

İkinci bölümde veri tekilleştirme hakkında bilgiler verilmektedir. Veri tekilleştirme algoritmaları, bu algoritmaların çalışma prensipleri örneklerle anlatılarak ve veri tekilleştirmede kullanılacak olan MERNİS projesi hakkında bilgiler bahsedilmektedir.

Üçüncü bölümde Atatürk Üniversite tarafından kullanılan bilgi sistemleri ve bu bilgi sistemlerinin birbirleriyle olan ilişkisi anlatılmaktadır. Ortak veri tabanı öncesinde ve sonrasında bilgi sistemlerinin birbirleriyle olan ilişkileri şemasal şekilde gösterilerek bilgi sistemlerinin hangi veriye ihtiyaç duydukları ve sağlayabilecekleri verilerin analizi yapılmıştır.

Dördüncü bölümde ortak veri tabanı oluşturulması için kullanılan materyallerden bahsedilerek çalışmanın adım adım hayata geçirilmesi için gerekli tablolar ve fonksiyonlar anlatılmaktadır.

Beşinci bölümde çalışma hakkında istatistik bilgiler paylaşılarak, bu istatistik bilgiler doğrultusunda çalışmanın sağladığı faydalardan bahsedilmektedir.

Sonuç kısmında ise ortak veritabanı oluşturulmasındaki adımlar, bu adımlar sırasında karşılaşılabilecek sorunlar ve bu sorunları hakkında öneriler verilerek çalışmanın daha ileri götürebilmesi adına gelecekte yapılacak çalışmalardan bahsedilmektedir.

BİRİNCİ BÖLÜM

SERVİS ODAKLI MİMARİ

1.1. SERVİS ODAKLI MİMARİ NEDİR?

Servis Odaklı Mimari (SOA)'yı farklı bakış açılarına göre çeşitli yollarla tanımlanabilir. Bir işletme açısından, firmanın müşterileri ve tedarikçileri ile iş yapma yeteneğini geliştiren bir dizi hizmeti temsil eder. Teknoloji açısından bakıldığında modülerlik, işlerin ayrılması, hizmetin yeniden kullanımı ve kompozisyonun anlamına gelir. Bunların yanı sıra web hizmetlerini içeren standartlara ve araçlara dayanan yeni bir programlama yöntemi ile karakterize edilen yeni bir proje felsefesidir. Bilgi teknolojileri yönetim bakış açısından, uygulamaları kavramsallaştırmak ve tasarlamak için yeni bir yoldur (Ordanini ve Pasini 2008).

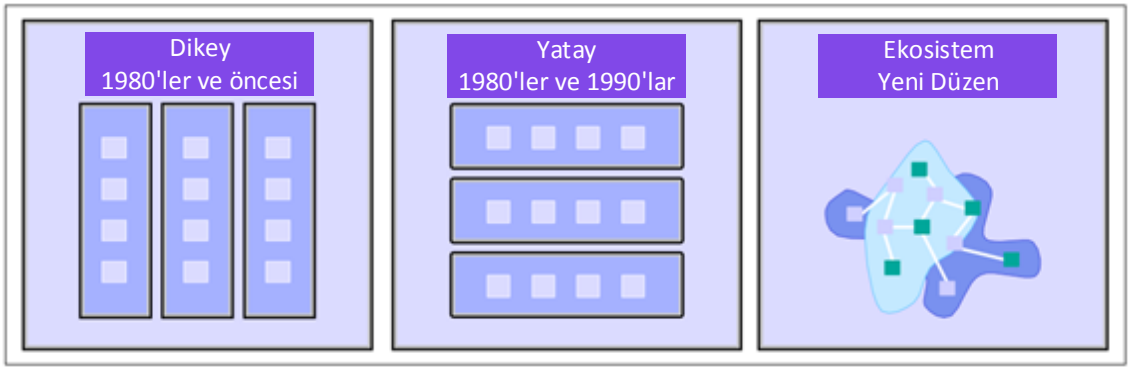
SOA (servis odaklı mimari) fikri wsdl servislerden daha önce ortaya atılmasına rağmen, wsdl servislerin kullanımının artması ile 2000'li yıllardan itibaren çok daha fazla kullanılmaya ve önem arz etmeye başlamıştır. SOA sadece yazılım teknoloji alanında değil birçok alanda kullanılabilir (Jammes ve Smit 2005).

SOA, hizmet odaklı dağıtılmış kurumsal uygulamaların geliştirilmesine olanak sağlayan, sistem geliştirme yöntemleri, teknikleri ve ilgili teknolojileri içeren bir dizi mimari kavram ve ilke olarak tanımlanmaktadır. Günümüz kurumsal bilgi işlem ortamlarında etkili olmak için, SOA kapsamının SaaS (Hizmet Olarak Yazılım), IaaS (Hizmet Olarak Altyapı), Hizmet Olarak Platform (PaaS) dâhil olmak üzere farklı türdeki hizmet modellerini kapsayacak şekilde genişletilmesi gerekir. Bu önemli hizmet yönelimli bilgi işlem eğilimleri SOA ile paralel olarak gelişmektedir ve şimdi SaaS, IaaS, PaaS ve Web 2.0'ın kurumsal hesaplamanın temel bileşenlerini oluşturan bir olgunluk aşamasına erişmiştir (Feuerlicht ve Govardhan 2009).

Servis odaklı mimari birçok servisin birbiri ile iletişime geçerek bir bilgi sistemi oluşumundaki mimari olarak da tanımlanabilir. Servisler, belirli bir standartta oluşturulmuş prosedürleri çağrılabilen ve bağımsız olarak işlevi yerine getirebilen yapılardan oluşurlar. Bütün servislerin oluşturulmasında işletim sistemi, ortam veya programlama dili gibi farklı faktörler üzerinde yürütülebilmektedir. Bundan dolayı yeni

servisler kolayca eklenebilir, eklenen servisler değiştirilebilir ve oluşturulan servisler tekrar tekrar kullanılabilir (Komoda 2006).

Günümüzde teknolojinin çok sık ve hızlı bir şekilde değişmesi cep telefonu, tablet veya akıllı cihazlar gibi daha önceden var olmayan yeni alanlar ortaya çıkarmıştır. Bu yeni alanların ortaya çıkmasıyla bilgi teknolojilerindeki rekabet daha artmış ve firmaları bu alanlarda yeni yazılımlar üretmeye itmiştir ancak firmalar hali hazırda kullandıkları yazılımları da devam ettirerek, yeni ortamlar için oluşturulan yazılımlar beraber çalışmasını sağlamak zorundaydılar. Yeni üretilecek yazılımların var olan yazılımlarla uyum sorunu yaşaması büyük problemlere neden olabilir. Bu aşamada firmalar esnek yapıda bir platform ihtiyacı duymaktadır çünkü eski yazılımlar ile yeni ortamlarda geliştirilen yazılımların dilleri, çalışma ortamları hatta çalışma mantıkları bile birbirinden farklıdır. Bilgi teknolojilerindeki bu gelişmeler sonucunda firmalar 1980'li yıllarda dikey, 1990'lı yıllarda yatay ve 2000'li yıllarda ise ekosistem şeklinde evrimleşmiştir. Bilgi teknolojilerindeki bu evrimleşmede SOA anlayışının büyük katkısı bulunmaktadır (Endrei ve ark. 2004). Aşağıdaki **Şekil 1.1**'de bu evrimleşmeyi göstermektedir.



Şekil 1.1. İşin Evrimleşmesi (Endrei ve ark. 2004).

Farklı sistemlerin birbirleriyle olan iletişiminde servis odaklı mimari yapısını kullanmak; hem dağıtık yapıda çalışma imkânı hem de bu iletişimin güvenilir, kontrollü, hızlı ve farklı yapılara kolay adapteyi sağlayacaktır.

1.1.1. Servis Odaklı Mimari Faydaları

SOA tabanlı mimarilerin ana faydaları arasında büyük ölçekli kurumsal sistemlerin yönetilebilir büyümesini kolaylaştırmak, hizmet kullanımını kolaylaştırmak ve organizasyondaki işbirliğine organizasyon maliyetlerini azaltmak yer almaktadır (MacKenzie ve ark. 2006).

SOA'nın diğer bir faydası ayrı ayrı bileşenlerde var olan değeri gerçekleştirmek için birlikte çalışabilirlik gerektiren büyük sistem ağlarını düzenlemek için basit bir ölçeklenebilir model sağlamasıdır. SOA prensiplerini kullanan bir mimar, ölçeklenebilir, geliştirilebilir ve yönetilebilir sistemler geliştirmek için daha donanımlıdır. İşlevsellik sahiplik sınırlarına nasıl entegre edileceğine karar vermek daha kolaydır. Örneğin, daha küçük bir şirket satın alan büyük bir şirket, edinilen bilgi teknolojileri altyapısını genel bilgi teknolojileri portföyüne nasıl entegre edeceğini hızlı şekilde belirleyebilir (MacKenzie ve ark. 2006).

SOA, doğal ölçeklendirme ve gelişim kabiliyeti sayesinde, belirli bir problem alanı veya süreç mimarisinin çeşitli ihtiyaçlarına uyarlanabilen bir bilgi teknolojileri portföyüne olanak tanır. SOA'nın desteklediği altyapı, üstel bir çift yönlü arayüzler üzerine kurulu bir sistemden daha çevik ve duyarlıdır. Bu nedenle, SOA ayrıca iş çevikliği ve uyarlanabilirliği için sağlam bir temel sağlar (MacKenzie ve ark. 2006).

1.2. XML ve WSDL TANIMLARI

XML (Extensible markup language), W3C tarafından geliştirilen ortamdan bağımsız bir yazım dilidir. XML, 1996 yılında World Wide Web Konsorsiyumu (W3C) bünyesinde kurulan çalışma grubu tarafından geliştirildi. XML dili başta SGML (Standard Generalized Markup Language) dilini temel alıp basitleştirilerek oluşturulan bir dildir. XML dilinin oluşturulma amacı; yazımı kolay, hızlı bir şekilde oluşturulabilecek, farklı yazılım dilleri ve uygulamalar tarafından desteklenen bir yazım dili oluşturulmasıdır (Bray ve ark. 2006).

XML, yazılımlar arası veri transferi yapmak için kullanılan hiyerarşik bir yapıya sahip olan bir yazım dilidir. XML'in hızlı yaygınlaşmasının sebebi; XML sayesinden

yeni diller oluşturulabilmesidir. Veriye hızlı erişilebilmesi ve hızlı işlem yapılabilmesi XML'in diğer avantajlarıdır (Erdinç, KILIÇASLAN, ve Erdem 2008).

XML'in yapısı HTML yapısına benzemektedir. HTML tarayıcı tarafından yorumlanarak görsel oluşturma amacı taşırken XML veriyi aktarma amacı taşımaktadır. XML de HTML'de olduğu gibi dışardan bakınca hemen anlaşılacak bir yapıya sahiptir (Walsh 1998). Aşağıda bir XML örneği verilmiştir;

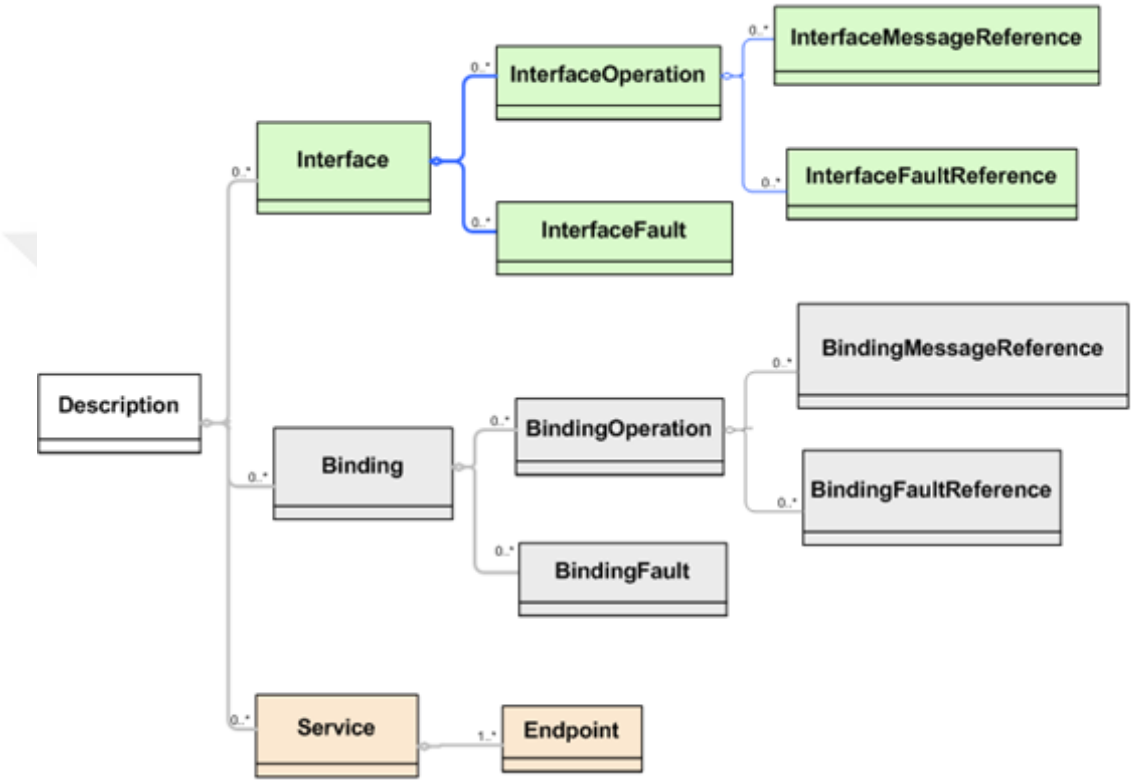
```
1  <kisiler>
2      <kisi id="1">
3          <ad>Gokhan</ad>
4          <soyad>TUTAR</soyad>
5      </kisi>
6      <kisi id="2">
7          <ad>Ahmet</ad>
8          <soyad>CAN</soyad>
9      </kisi>
10     <kisi id="3">
11         <ad>Mehmet</ad>
12         <soyad>POLAT</soyad>
13     </kisi>
14     <kisi id="4">
15         <ad>Hasan</ad>
16         <soyad>TALİP</soyad>
17     </kisi>
18 </kisiler>
```

Şekil 1.2. XML Örneği

WSDL, XML'in bir çeşidi değildir. XML standartları kullanılarak web servis oluşturulmasında faydalanılan bir yazım biçimidir. WSDL oluşturmak için XML'e tamamen hâkim olunması zorunlu değildir. XML'in standartlarının bilinmesi yeterlidir (Erl 2004).

WSDL dosyalarının esas kullanım amacı bir web servisi tarif etmektir. Web serviste bulunan fonksiyonlar, bu fonksiyonların aldığı parametreler, bu parametrelerin tipleri ve alabilecekleri değerler gibi bütün bilgiler WSDL dosyalarında mevcuttur. WSDL dosyalarının web servisin bir ara yüzü olarak tanımlanabilir (Chinnici ve ark. 2007).

W3C tarafında oluşturulan WSDL'in temel amacı web servislerde bir standart oluşturulmasıdır. WSDL tıpkı XML'de olduğu gibi hiyerarşik (parent-childeren) bir yapıya sahiptir ve kendi içerisinde belirli gruplar halindedir. Her grubun kendine has bir işlevi bulunmaktadır (Booth ve Liu 2007). Aşağıdaki şema WSDL bileşenlerinin hiyerarşik yapısına genel bakışı göstermektedir.



Şekil 1.3. WSDL Bileşenlerinin Hiyerarşisi (Booth ve Liu 2007).

1.3. SOAP ve WEB SERVİSLER

SOAP (Simple Object Access Protocol) yani basit nesne erişim protokolü dağıtık yapılarda ve web servislerinin birbirleriyle haberleşmesinde kullanılmak üzere tasarlanan, RPC (Remote Procedure Call) modelini ve istemci/sunucu mantığını kullanan bir protokoldür. Bir başka deyişle SOAP, web servisleri oluştururken kullanılmak üzere geliştirilmiş kurallardır (Box ve ark. 2000).

SOAP kodlama stili, programlama dillerinde, veri tabanlarında ve yarı yapılandırılmış verilerde sistemlerde bulunan ortak özelliklerin genelleştirilmesi olan basit bir sisteme dayanır. SOAP basit tip veya birkaç parçanın kompoziti olarak inşa

edilebilir. İlk olarak, açıklanan tip sistemi ile tutarlı herhangi bir notasyonda bir şema verildiğinde, bir XML dilbilgisi için bir şema oluşturulabilir. İkincisi, bir tip-sistem şeması ve bu şemaya uygun belirli bir değerler grafiği verildiğinde, bir XML örneği oluşturulabilir. Tersine, bu kurallara uygun olarak üretilmiş bir XML örneği verildiğinde ve orijinal şema da verildiğinde, orijinal değer grafiğinin bir kopyası oluşturulabilir (Box ve ark. 2000).

SOAP protokolü 3 temel parçadan oluşmaktadır. Bunlar; Envelope, Header ve Body dir.

1.3.1. Envelope

Soap mesajında ne olduğunu ifade etmek için genel bir çerçeve tanımlar. Kiminle iletişime geçecek veya zorunlu mu gibi bilgiler bulunur. Envelope Soap'ın en dış katmanıdır ve Header ve Body parçalarını da barındırır.

1.3.2. Header

Header parçası, SOAP'ın ilk alt öğesi olarak kodlanmıştır. Başlık ögesinin tüm acil alt öğeleri başlık girişleri olarak adlandırılır. Bu kısımda meta-data denilen doküman bilgi bulundurulur.

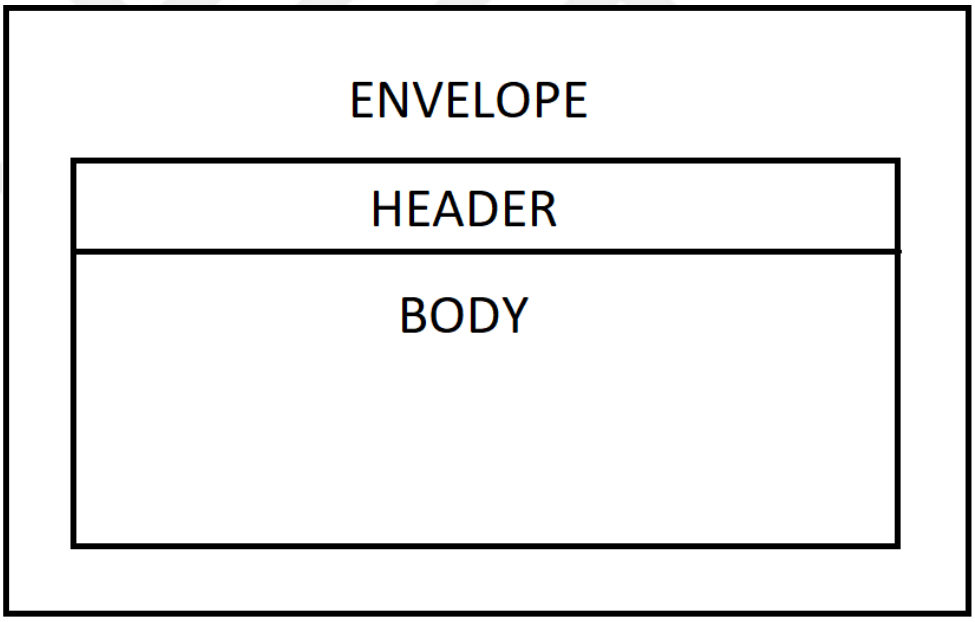
1.3.3. Body

Body kısmı Soap mesajındaki en önemli kısmı oluşturur. Bu kısım web servisteki metotların adları, parametreleri ve geri dönüş değerleri gibi hayati öneme sahip bilgileri taşımaktadır. Mesajın alıcısına yönelik zorunlu bilgi alışverişinde bulunmak için basit bir mekanizma sağlar. Header parçasının tipik kullanımları arasında marshalling RPC çağrıları ve hata raporlamayı barındırır.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”>
    <SOAP-ENV:Header>
    ….
    ….
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
    ….
    ….
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Şekil 1.4. SOAP Örneği

Şekil 1.5.’de SOAP yapısının temel parçalarının şekilsel gösterimi bulunmaktadır. Şekilden de görülebileceği gibi Envelope kısmı hem Header hem de Body kısmını kapsamaktadır.



Şekil 1.5. SOAP kısımları

Web servisler, farklı platformlarda çalışan uygulamaları bir araya getirerek veri alışverişini mümkün kılar. Web servislerin platform bağısız olması bu sistemin en büyük tercih nedenlerinden birisidir (Ferris ve Farrell 2003). Web servislerin platform bağımsız olarak çalışabilmesinin en büyük nedeni XML, WSDL ve SOAP standartlarını kullanmasıdır.

Web Servisler, iş veya bireyler için bir dizi işlevsellik sağlayan açık standart diller aracılığıyla web üzerinden erişilebilen, kendi kendine yeten modüler uygulamalardır. Web hizmetlerini çekici kılan, kullanıcının gereksinimlerini karşılamak için farklı kuruluşlar tarafından geliştirilen web hizmetlerini bir araya getirme becerisidir. Bu entegrasyon, Web hizmetlerini uygulamak için kullanılan dillere ve Web hizmetlerinin yürütüldüğü platformlara bakılmaksızın, Web hizmeti arabirimlerinin ortak standartlarına dayanır (Rao 2004).

Web servisler, iletişim kuralı yığınının alt düzeyinden kaynaklanır ve TCP / IP üzerinden iletişime geçerek veri aktarımında bulunurlar. Web servis hizmetinin sınıflandırması, güvenlik gereksinimleri, maliyet ve kalite parametreleri gibi hizmetlere özgü bilgileri açıklar. Verilen web servisler için kullanıcı ara yüzünü açıklayan sunum katmanını içerir (Ferris ve Farrell 2003).

İKİNCİ BÖLÜM

VERİ TEKİLLEŞTİRME

2.1. VERİ TEKİLLEŞTİRME NEDİR?

Veri Tekilleştirme, farklı kayıtlarda aynı veriyi farklı şekilde gösteren bir veri setindeki kayıtları bulma ve silme görevi anlamına gelir. Kayıt şekli, saklama yeri veya tercihindeki farklılıklar nedeniyle olduğu gibi, ortak bir tanımlayıcıyı paylaşabilen/paylaşamayan varlıklar temelinde veri kümelerini birleştirirken, kayıt tekilleştirmesine ihtiyaç duyulur (Gujar, Desai, ve Technology 2013).

Ortak veri tabanı oluşturulmasındaki en büyük problemlerden biri de veri çoğullaşmasıdır. Veri çoğullaşması kontrol altında tutulmazsa, sadece kirli verilerden oluşmuş, büyümesi kontrolsüzce artan veri yığını oluşur (Santos ve ark. 2007).

Veri çoğullaşması sorunu kaçınılmazdır. Kaçınılmaz olmasının nedeni; birbirinden bağımsız birkaç merkezinden veri toplandığında farklı kaynaklardan gelen bu verilerin farklı şekilde ifade edilen aynı veriler mi yoksa tamamen farklı veriler mi olduğunu karar verme işleminin zorluğudur (Chandrasekar 2013).

Ortak veri tabanına gelen verilerin aynı veya farklı olduklarına karar verirken kendi yapınıza uygun bir algoritma kullanılması önemlidir. Veriler anlık olarak geliyor ve anında cevap verilmesi isteniyorsa; veri tekilleştirmesi algoritması seçilirken performans, doğruluk ve operasyonel maliyet gibi konulara dikkat edilerek seçilir (Chandrasekar 2013).

2.1.1. Metin Karşılaştırma Algoritmaları

Çoğullaşan kayıtları bulup tekilleştirmek için, iki farklı metni karşılaştırıp bunların aynı metin olduğuna karar verebilmek gerekir. İnsan gözü ile iki metini karşılaştırıp aynı olup olmadıklarına karar vermek kısmen kolay ve yüksek oranda doğruluk ile sonuçlanacaktır. Ancak veri tabanındaki kayıtların çok fazla olması ve sürekli yeni verilerin gelmesi bu işlemin bir insan tarafından yapılmasını mümkün kılmamaktadır.

Bilgisayar ortamında iki metni birebir karşılaştırmak yani iki metnin harfi harfine birbirine eşitliğinin karşılaştırılması çoğullaşan kayıtları bulmada çok faydalı olmayacaktır. Çünkü metin içerisindeki bazı yazım hataları veya kısaltmalardan dolayı, iki metin birbirinden farklı görünse de aslında aynı metindirler. Farklı görünen ancak aynı olan metinleri belirlemek için metin karşılaştırma algoritmaları kullanılabilir. İki metnin harfi harfine karşılaştırılması siyah-beyaz bir karşılaştırma olduğunu varsayarsak, metin karşılaştırma algoritmaları da gri karşılaştırmadır. Gri karşılaştırmalar kesin sonuç vermezler. Metinlerin ne kadar benzer olduğunu veya ne kadar farklı olduğunu sunarlar (Cohen, Ravikumar, ve Fienberg 2003).

Birçok metin karşılaştırma algoritması bulunmaktadır. Bu algoritmaların kendilerine has farklı mantıkları bulunmaktadır. Bu algoritmalardan bazıları iki metin arasındaki benzerliğe odaklanır, bazıları iki metin arasında farklılığa, bazıları da metni parçalayarak alt metinlerdeki benzerliğe veya farklığa odaklanırlar. Yapılan işlemler her ne kadar farklı olsa da aslında yapılan işlem iki metnin birbirlerine ne kadar benzediğini bulmaktır. (Jokinen ve ark. 1996).

2.1.1.1. The Naive Algoritması

The Naive diğer adıyla kaba kuvvet algoritması en basit metin karşılaştırma algoritmalarından birisidir. Bu algoritmadaki mantık "A" metnindeki alt dizeleri arasında eşleşmeye çalışmasıdır. Bu algoritmanın örneği aşağıda verilmiştir (Baeza-Yates 1989).

```
naive_arama( metin, n, desen, m )
char metin[], desen[] ;
int n, m;
{
    int i, j, k, lim;
    lim = n-m+1;
    for( i = 1; i <= lim; i++ )
    {
        k=i;
        for( j=i; j<=m && metin[k] == desen[j]; j++ ) k++;
        if( j > m ) eslesen_pozisyonlari_raporla( i-j+1 );
    }
)
```

Şekil 2.1. The Naive Algoritması (Baeza-Yates 1989).

Naive algoritma iki metinin karşılaştırılmasından çok metin içerisinde arama yapmak için kullanılan bir algoritmadır. Metin içerisinde arama yapıldıktan sonra çıkan sonuca göre iki metinin birbirlerine olan yakınlığı için bir değer üretilebilir ancak arama işlemi bire bir yapıldığı için bu algoritma metin karşılaştırma işlemlerinde pek faydalı değildir.

2.1.1.2. Levenshtein Distance Algoritması

Levenshtein Distance algoritması iki farklı metinin birbirlerine olan benzerliğinden yola çıkarak metinler aralarındaki mesafeyi ölçer. Yani "A" metnini "B" metnine dönüştürmek için yapılması gerekli olan ekleme, çıkarma ve güncellemelere göre metinler arasında bir mesafe belirler. Bu ekleme, çıkarma ve güncelleme işlemleri ne kadar çoksa iki metin arasındaki mesafe o kadar artacaktır. Mesafenin büyük olması demek iki metin arasındaki bağın az olması anlamına gelir (Levenshtein 1966) (Haldar ve Mukhopadhyay 2011).

Örnek verilecek olursa; "elma" kelimesinin "alma" kelimesine mesafesini bulmak için Levenshtein Distance algoritması uygulanırsa mesafe "1" çıkacaktır. Çünkü "alma" kelimesinin ilk harfini "e" olarak değiştirilirse "elma" kelimesi elde edilir, yapılan işlem sayısı da "1" olduğu için iki kelime arasındaki mesafe de "1" olur.

Levenshtein Distance algoritması kullanılarak "TİPO" kelimesinin "POSTA" kelimesine olan mesafesini ölçmek için gerekli uygulama adımları sırasıyla aşağıda verilmiştir.

1. adım: İlk kelime "s" adında bir değişken olarak kabul edilir ve "n" bu değişkenin uzunluğudur. İkinci kelime "t" adında bir değişken kabul edilir ve "m" bu değişkenin uzunluğudur. Algoritma uygulanacak kelimelerin sığacağı bir matris düzenini oluşturulur.

Tablo 2.1. Levenshtein Distance Algortiması 1.adım

		P	O	S	T	A
	0	1	2	3	4	5
T	1					
İ	2					
P	3					
O	4					

2. adım: “i” için 1 den “n” e kadar olan ve “j” için 1 den “m” e kadar olan harfler karşılaştırılır. Eğer s[i] t[j] ye eşitse maliyet “0”, s[i] t[j] ye eşit değilse “1” dir. Maliyeti “0” olan hücrelere, hücrenin sol çaprazında bulunan değer yazılır. Maliyeti “1” olan hücrelere ise; hücrenin üstündeki, solundaki ve sol çaprazındaki sayıların en küçüğü alınarak 1 eklenir.

Tablo 2.2. Levenshtein Distance Algortiması 2.adım

		P	O	S	T	A
	0	1	2	3	4	5
T	1	1				
İ	2	2				
P	3	2				
O	4	3				

Yukarıdaki 2. adımı sırasıyla yaparsak;

1- T-P -> T’yi P’ye dönüştürmek için “1” işlem (P yerine T yazmak) gerekir. Bundan dolayı **Tablo 2.2**’de sarı renkli hücreye “1” yazılır. Bu işlem algoritmaya göre yapılırsa; harfler (P-T) farklı olduğu için maliyet 1 dir. Maliyet 1 olduğu için hücreye, hücresini üstü, sol çaprazı ve solundaki sayıların en küçüğünün (sırasıyla 1,0,1) bir fazlası yani 0+1 = 1 yazılır.

2- Tİ-P -> Tİ’yi P’ye dönüştürmek için “2” işlem (P yerine T yazmak ve İ’yi silmek) gereklidir. Bundan dolayı **Tablo 2.2**‘de mavi renkli hücreye “2”

yazılır. Bu işlem algoritmaya göre yapılırsa; harfler (P-İ) farklı olduğu için maliyet 1 dir. Maliyet 1 olduğu için hücreye, hücresini üstü, sol çaprazı ve solundaki sayıların en küçüğünün (sırasıyla 1,1,2) bir fazlası yani 1+1 = 2 yazılır.

3- TİP-P -> TİP'i P'ye dönüştürmek için "2" işlem (T ve İ harflerini silmek) gereklidir. Bundan dolayı **Tablo 2.2**'de turuncu renkli hücreye "2" yazılır. Bu işlem algoritmaya göre yapılırsa; harfler (P-P) aynı olduğu için maliyet 0 dır. Maliyet 0 olduğu için hücreye, hücresini üstü çaprazındaki sayı yani 2 yazılır.

4- TİPO-P -> TİPO'yu P'ye dönüştürmek için "3" işlem (T, İ ve O harflerini silmek) gereklidir. Bundan dolayı **Tablo 2.2**'de gri renkli hücreye "3" yazılır. Bu işlem algoritmaya göre yapılırsa; harfler (P-O) farklı olduğu için maliyet 1 dir. Maliyet 1 olduğu için hücreye, hücrenin üstü, sol çaprazı ve solundaki sayıların en küçüğünün (sırasıyla 2,3,4) bir fazlası yani 2+1 = 3 yazılır.

3. adım: 2. Adımda yapılan işlemler d[m,n] değerleri için uygulanır. Bu işlemler tamamlandıktan sonra oluşan tablonun sağ alt köşesindeki sayı iki kelime arasındaki mesafeyi verir. **Tablo 2.3**'te de gösterildiği gibi TİPO kelimesinin POSTA kelimesine dönüşmesi için gerekli işlem sayısı 5 tir. Bu sayı aynı zamanda TİPO ile POSTA kelimelerinin arasındaki mesafeyi gösterir (Haldar ve Mukhopadhyay 2011).

Tablo 2.3. Levenshtein Distance algortiması 3.adım

		P	O	S	T	A
	0	1	2	3	4	5
T	1	1	2	3	3	4
İ	2	2	2	3	4	4
P	3	2	3	3	4	5
O	4	3	2	3	4	5

2.1.1.3. Damerau Levenshtein Distance Algoritması

Damerau Levenshtein Distance algoritması Levenshtein Distance algoritmasını temel alarak oluşturulan bir algoritmadır. Damerau Levenshtein Distance algoritmasının oluşturulmasındaki amaç; Levenshtein Distance algoritmasındaki yazım yanlışlarından

kaynaklı kelimeler arasındaki mesafeyi azaltmaktır. İnsanlar klavyeden hızlı bir şekilde yazı yazarken yanlışlıkla kelimelerin harfleri bazen yer değiştirebilir. Bir kelimedeki bir harfi yer değiştiğinde (POSTA -> POTSA) oluşan yeni kelime ile eski kelimenin arasındaki mesafe Levenshtein Distance algoritmasına göre “2” dir. Bu durum Damerau Levenshtein Distance algoritmasına göre “1” dir çünkü Damerau Levenshtein Distance algoritması harflerin yanlışlıkla yer değiştirme ihtimaline de göz önüne almıştır (Damerau 1964) (Bard 2007).

2.2. MERNİS

MERNİS fikri 5 Mayıs 1972 tarih ve 1587 sayılı Nüfus Kanunun çıkmasıyla beraber ortaya atıldı. MERNİS dünyadaki ilk e-devlet projelerinden biri olmuştur. MERNİS projesindeki amaç nüfus hizmetlerindeki verilerin ortak bir merkezde toplanarak bu verilerin hem hızlı hem de güvenli bir şekilde paylaşımı ve verilerin güncellenmesidir. MERNİS fikri her ne kadar 1972 yılında ortaya atılsa da fikrin tam olarak işlevsel hale gelmesi Kasım 2002 sonunu bulmuştur ("MERNİS" 2019).

2.2.1. MERNİS Projesinin Tarihçesi

MERNİS projesinin tam olarak hayata geçmesi yaklaşık 30 yıl sürmüştür. Bu 30 yıllık süreçte yapılan işlemler “Nüfus ve Vatandaşlık İşleri Genel Müdürlüğü” tarafından kronolojik olarak aşağıdaki şekilde verilmiştir.

5 Mayıs 1972 tarih ve 1587 sayılı Nüfus Kanunu ile MERNİS projesi fikri doğdu.

1976 yılında Devlet Planlama Teşkilatı (DPT) tarafından projelendirildi.

1980 yılında proje, Ortadoğu Teknik Üniversite'sine (ODTÜ) ihale edildi.

1982 yılında projeyi uygulama çalışmaları başladı.

1982-1996 yılları arasında proje çalışmalarına devam edildi.

1996 yılında Dünya Bankası MERNİS projesini özelleştirme ve Sosyal Güvenlik Ağı (PIAL) kapsamına aldı ve proje fizibilite çalışması yapıldı.

1997 yılında yürürlüğe giren 4300 sayılı kanunla sağlanan ödeneğin kullanılması ile MERNİS projesi hız kazandı. MERNİS projesi yönetim şeması oluşturuldu.

1997 yılında Dünya bankası MERNİS projesine kaynak aktardı.

1997-1999 yılları arasında Genel Müdürlük ve 923 ilçe nüfus müdürlüğünün altyapısı tamamlanarak bilgisayar sistemleri kuruldu.

Genel Müdürlük ile İlçe Nüfus Müdürlüklerinde bulunan Sunucu ve Kişisel Bilgisayarlara İşletim Sistemleri ve Veri Tabanı Yönetim Sistemleri kuruldu.

1997-1999 yılları arasında nüfus kayıtları bilgisayar ortamına aktarıldı.

1998 yılında ilçe nüfus müdürlüklerine destek vermek amacıyla Acil Destek Merkezi (call center) kuruldu.

1998-2000 yılları arasında MERNİS uygulama yazılımları gerçekleştirildi.

1997-2002 tarihleri arasında 6500 personele bilgisayar teknolojileri konusunda eğitim verildi.

28 Ekim 2000 tarihinde Türkiye Cumhuriyeti Kimlik Numarası tüm nüfus kayıtlarına verildi.

Eylül 2000 tarihinde merkezi sunucu sistemi, depolama sistemi ve yedekleme sistemleri satın alınarak 16 Kasım 2000 tarihinde hizmete açıldı.

Kamu kurum ve kuruluşlarına ve vatandaşlara T.C. Kimlik Numarasını yaygınlaştırmak amacıyla Nüfus Bilgi Bankası kuruldu.

13 Aralık 2001 tarihinde ilçe nüfus idarelerinin merkezle çevrimiçi çalışmasını sağlamak amacıyla nüfus idarelerinin iletişim alt yapısının kurulması ve online uygulaması ihalesi yapıldı.

18 Mart 2002 tarihinden itibaren online uygulama Ankara ve Kırıkkale iline bağlı ilçelerde pilot uygulama olarak başlanarak, 2002 yılı sonu itibariyle projenin bitirilmesi hedeflendi.

Kasım 2002 sonu itibariyle MERNİS veri tabanı kurulmuş ve MERNİS sistemin online olarak çalışması sağlanmıştır ("MERNİS" 2019).

MERNİS projesi ile Türkiye Cumhuriyeti vatandaşlarına birer kimlik numarası verilmesi sadece nüfus müdürlükleri için değil aynı zamanda diğer devlet kurumları için de büyük kolaylıklar sağlamaktadır. Kimlik numarası 11 haneli eşsiz bir sayıdan

oluşmaktadır. Bu sayı eşsizdir yani bir vatandaşa verildiğinde başka bir vatandaş için sayıyı kullanamaz.

Bir vatandaş devlet kurumlarından hizmet almak için gittiğinde kurumda yapılan ilk işlem kişi bilgilerini kayıt altına almaktır. Bu vatandaş ister bir öğrenci olsun ister kurumun kendi personeli olsun isterse sadece bilgi edinmek isteyen kişi olsun kişinin bilgileri kayıt altında tutulur. Bu kişiyi kayıt altına alma işlemi sırasında gerek yazım hataları gerek isim kısaltmalarından dolayı bir kişinin aynı kurumda birden çok kaydı oluşur. Bunu önüne geçmek için her kurum vatandaşlara sicil, öğrenci veya vergi numarası gibi adlar altında birtakım numaralar vermiştir. Bu numaralar genellikle ardışık sayılardan oluşmaktadır. Her ne kadar bu sistem veri çoğullaşmasını bir nebze azaltsa da vatandaşlara her kurumun farklı bir numara vermesi karışıklığa neden olabilmekteydi. Bu karışıklıklar hem istatistiksel bilgilerde bir karmaşaya hem de kişinin geçmişi ile ilgili hatalı yönlendirmelere sebep olabilir. Bu hatalı yönlendirmeler bazen hayati yanlışlara sebebiyet verbilmektedir. Buna örnek verecek olursa; hasta bir sağlık kuruluşuna gittiğinde bazı tahliller verir ancak eski tahlil bilgileri farklı bir kayıtta tutulduğu için doktoru hastalığı doğru teşhis edemeyebilir.

MERNİS projesi kapsamında her vatandaşın bütün kurumlar tarafından kullanılabilecek bir tane vatandaşlık numarası verilmektedir. Bu numaranın ardışık olmayan belirli bir algoritmaya göre oluşturulması kişi bilgilerindeki yanlış veri girişi ve veri çoğullaşma sorunlarını büyük bir oranda ortadan kaldırmıştır.

ÜÇÜNCÜ BÖLÜM

KULLANILMAKTA OLAN BİLGİ SİSTEMLERİ VE BİLGİ SİSTEMLERİNİN BİRBİRLERİYLE OLAN İLİŞKİSİ

3.1. KULLANILMAKTA OLAN BİLGİ SİSTEMLERİ

Büyük ölçekli bilgi işlem merkezlerinde, çok sayıda farklı platformlarda yazılan bilgi sistemleri bulunabilir. Bilgi sistemlerinini farklı platformlarda yazılması bilgi sistemlerinin birbirleriyle iletişime geçip veri aktarım işini zorlaştırmaktadır. Bu veri aktarım işlemlerinde platformdan bağımsız olduğu için genellikle web servisleri tercih edilir. Her bilgi sistemi diğer bir bilgi sistemlerindeki veriye ihtiyaç duyacağı için çok sayıda servis yazılması gerekmektedir.

Bilgi işlem merkezinde bulunan farklı bilgi sistemlerinin kendi aralarında web servisler ile iletişime geçmesi yerine; ortak bir veri tabanı oluşturulması ve bu veri tabanının diğer bilgi sistemleriyle iletişime geçmesi zaman, güvenlik ve gereksiz kod yazımı açısından fayda sağlayacaktır.

Ortak veri tabanı oluşturmadan önce kurumda kullanılan bilgi sistemleri ve bilgi sistemlerinin ihtiyaç duyduğu verilerin incelenmesi gerekir. Bu tezde Atatürk Üniversitesi örneği ele alındığı için öncelikle kurumda kullanılan bilgi sistemleri incelenecektir.

3.1.1. Öğrenci Bilgi Sistemi (ÖBS)

En basit tabirle öğrencilerin özlük ve not bilgilerinin bulunduğu bilgi sistemidir. Bu bilgi sisteminde sadece öğrencilerin bilgileri değil aynı zamanda çoğu akademik personellerin de bilgileri bulunmaktadır. Öğrenci bilgi sistemi öğrenci odaklı çalıştığından dolayı öğrenci işleriyle direkt bir etkileşimi olmayan idari personellerin (örnek olarak; Strateji Geliştirme Daire Başkanlığı personelleri) ve akademik personelin (örnek olarak; ders vermeyen uygulamalı birimlerde bulunan öğretim görevlileri) bilgileri yer almamaktadır.

3.1.2. Web Portalı

Üniversite web sitesinde bulunan veriler portal sayesinde sağlanmaktadır. Bu bilgi sisteminde duyurular, haberler ve idari – akademik bütün personelin özlük, unvan, makam ve birim bilgileri bulunmaktadır.

3.1.3. Hızlı Geçiş Sistemi (HGS)

Üniversite kampüsü içerisinde bulunan araçların yetkileri dâhilinde gişelerden geçişlerine izin veren sistemdir. Bu sistemde üniversite en az bir arabası olan personel ve öğrencilerin özlük bilgileri bulunmaktadır. Bu sistem üniversitede çalışmayan ancak üniversitede işleri bulunan kişilerin de özlük bilgisi bulunmaktadır. Yani üniversite personelinin ve öğrencilerinin bir kısmı ile üniversiteyle ilişkisi bulunan kişilerin bilgileri bulunmaktadır. Bu bilgiler öğrenci bilgi sistemi veya portal bilgi sisteminde bulunan bilgiler kadar detaylı değildir. Kişinin öğrenci/sicil numarası, adı, soyadı, birimi, arabanın plakası ve yetkisi (arabanın geçebileceği kapılar) gibi sınırlı bilgileri içerir.

3.1.4. Akıllı Kart Kapı Girişleri (AKK)

Bu sistem fakülte ve diğer binalardaki kapılarda kişilerin yetkileri dâhilinde binalara giriş ve çıkışlarında turnikeler tarafından kullanılan sistemdir. Bu sistemde bütün öğrenci, idari ve akademik personel bilgileri bulunmaktadır. Bu bilgiler kişinin öğrenci/sicil numarası, adı, soyadı, birimi ve yetkisi (kişinin hangi kapılardan geçebileceği gibi) sınırlı bilgilerdir.

3.1.5. Akıllı Kart Yemekhane Girişleri (AKY)

Öğrenci ve akademik – idari bütün personelin yemekhaneye giriş ve yemek ücretleri akıllı kartlar aracılığıyla yapması için kullanılan sistemdir. Bu sistemde bütün öğrenci, idari ve akademik personel bilgileri bulunmaktadır. Bu bilgiler kişinin öğrenci/sicil numarası, adı, soyadı, birimi, akıllı kartında bulunan para miktarı ve yetkisi gibi sınırlı bilgilerdir.

3.1.6. Akıllı Kart Basım Programı (AKB)

Üniversitedeki öğrenci, akademik ve idari personelin akıllı kartlarını basımı için kullanılan sistemdir. Bu sistem hem öğrenci hem de bütün personelin bilgilerine ihtiyaç duymaktadır ancak kendine ait bir veri tabanı bulunmamaktadır.

3.1.7. Kütüphane Bilgi Sistemi (KBS)

Üniversite personeli ve öğrencilerin kütüphanede bulunan kaynakları ödünç almalarını sağlamak ve bu işlemlerin takibini yapmak için kullanılan sistemdir. Kütüphane bilgi sisteminde kitap ödünç alma veya iade etme gibi işlemler de akıllı kartlar kullanılarak yapılmaktadır. Bu sistemde de öğrenci/sicil numarası, adı, soyadı birimi, alınan ödünç kitaplar gibi sınırlı bilgiler içermektedir.

3.1.8. Kurum Dışı Servislerden Faydalanan Bilgi Sistemleri

Bu bilgi sistemleri üniversitenin diğer bilgi sistemleri ile veri alış verişi yapmamaktadır. Ancak devlet kurumlarının üniversiteye sağladığı web servislerden faydalanmaktadırlar. Bu bilgi sistemleri irili ufaklı yaklaşık on bir adettir. Bu bilgi sistemlerinin en büyükleri arasında Sağlık Araştırma ve Uygulama Merkezine bağlı olan hastane tarafından kullanılan "Tıp Fakültesi Hasta Takip Bilgi Sistemi" ve Diş Hekimliği Fakültesi tarafından kullanılan "Diş Hekimliği Fakültesi Hasta Takip Bilgi Sistemi" bulunmaktadır. Diğer bilgi sistemlerinin (DBS) listesi **Tablo 3.1**.'de verilmiştir.

Tablo 3.1. Kurum dışı servislerden fardalanan bilgi sistemleri

Sıra	Bilgi sisteminin adı
1	AOF Bilgi Sistemi
2	UZEM Bilgi Sistemi
3	Atateknokent Bilgi Sistemi
4	DAYTAM Bilgi Sistemi
5	Elektronik Belge Yönetim Sistmei
6	Yabancı Diller Yüksekokulu Bilgi Sistemi
7	Eğitim Fakültesi Bilgi Sistemi
8	Spor Bilimleri Fakültesi Bilgi Sistemi
9	Elektronik Arşiv Bilgi Sistemi

10	Ağkayıt Bilgi Sistemi
11	ÜNİP bilgi sistemi

3.2. BİLGİ SİSTEMLERİNİN BİRBİRLERİYLE OLAN İLİŞKİSİ

Atatürk üniversitesinde kullanılan bilgi sistemleri farklı amaç doğrultusunda ancak ortak veri setlerine ihtiyaç duyarak çalışan sistemlerdir. Tabi, bu sistemler birbirlerinden bağımsız sistemler oldukları için her birinin veri tabanları, yazım standartları ve hatta yazılım dilleri birbirlerinden farklıdır. Bu sistemler farklı veri tabanları kullandıkları için ortak veri setleri her bir sistemde tekrar tekrar bulunmak zorundadır.

Bu sorunun ideal çözümü bütün sistemlerin aynı veri tabanından veri çekmesidir. Ancak yukarıdaki sistemlerin bazıları üniversite tarafından geliştirilen yazılımlarken, bazıları da dışardan satın alınan paket yazılımlardır. Yukarıda bahsedilen bu yazılımların bir kısmının kaynak kodları üniversitede bulunmamaktadır. Böyle bir durumda da bütün bu yazılımları tek çatı altına toplayıp veri tabanlarının birleştirilmesi pek mümkün değildir. Hatta üniversite tarafından geliştirilen farklı sistemler farklı veri tabanları kullanabilmektedir.

Aynı veri setleri farklı yazılımlarda tekrarlansa da bu verileri kişiler tarafında tekrar tekrar girmesini istemek hem zaman kaybına hem de kişilerin iş yükünün artmasına neden olacaktır. Bunun için bir bilgi sistemlerine girilen veri seti diğer sistemlerin ihtiyaç duyulması halinde ortak veri tabanına kaydedilirse, kişilerden tekrar tekrar veri girilmesine gerek kalmadan web servisler aracılığıyla arka planda veri alış verişi yapılabilir.

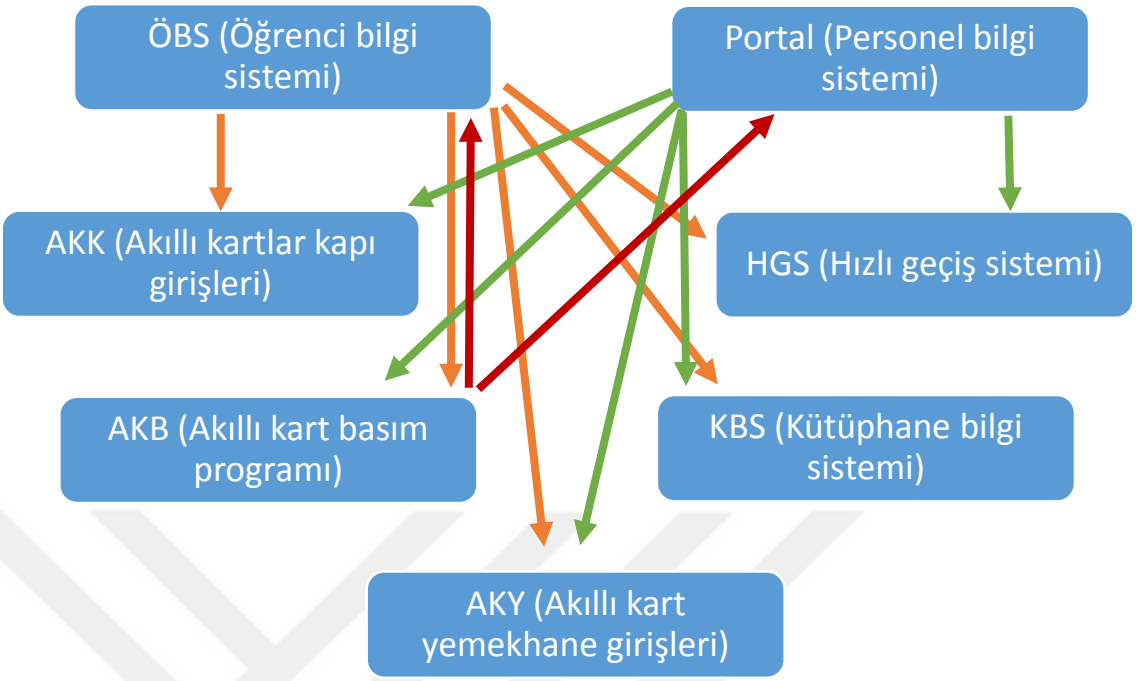
Atatürk Üniversitesinin yazılımlardaki veri setleri bakımından genel olarak hizmet verdiği üç farklı kitle bulunmaktadır. Bunlar; öğrenciler, idari personel ve akademik personeldir. Ortak veri tabanı yapısı oluşturmak için üniversite tarafından kullanılan bilgi sistemlerinde hangi bilgilerin var olduğu aşağıdaki **Tablo 3.2**‘de verilmiştir.

Tablo 3.2. Üniversitede kullanılan bilgi sistemlerde var olan bilgiler

Bilgi sistemi	**Akademik Personel Bilgisi**	**İdari Personel Bilgisi**	**Öğrenci Bilgisi**
Öğrenci bilgi sistemi	Çoğu	Kısmen	Tamamı
Portal	Tamamı	Tamamı	Yok
Hızlı geçiş sistemi	Kısmen	Kısmen	Kısmen
Akıllı kart kapı girişleri	Tamamı	Tamamı	Tamamı
Akıllı kart yemekhane girişleri	Kısmen	Kısmen	Kısmen
Akıllı kart basım programı	Tamamı	Tamamı	Tamamı
Kütüphane bilgi sistemi	Tamamı	Tamamı	Tamamı

Tablo 3.2.'de üniversitede kullanılan sistemlerin hepsinde ayrı ayrı kişi bilgileri bulunduğu gösterilmektedir. Bu bilgi sistemlerinden en gelişmişleri ve en fazla bilgi barındıranları öğrenci bilgi sistemi ve portal sistemidir. Bu iki bilgi sistemi (ÖBS ve Portal) en fazla bilgi barındırdığı ve üniversitenin temel bilgi sistemleri arasında yer aldığından dolayı diğer bilgi sistemleri tarafından veri kaynağı olarak kullanılmaktadır.

İster öğrenci ister personel olsun kişiler üniversiteye ilk geldiğinde bilgi sistemlerine kayıt yaptırmaları gerekmektedir. Bilgi sistemleri arasında web servisleri aracılığıyla bilgi paylaşımı olmadan kullanıldığında personel ve öğrenciler bürokrasiye takılacaklardır. Yani üniversiteye yeni gelen bir personel veya öğrenci üniversitede bulunan sistemlere kayıt olmak için tek tek bütün sistemleri dolaşıp kendilerini kaydettirmeleri gerekmektedir. Bu kayıt işlemleri zaman kaybı ve gereksiz iş yükü gibi bilgi sistemlerinden memnun olmama sebeplerine dönüşebilmektedir. Bu sorunu çözmek için bütün bilgi sistemleri birbirleriyle iletişime geçmesi ve ihtiyaç duydukları veri setlerini diğer bilgi sistemlerinden alması gerekmektedir. Her bir bilgi sistemin diğer bilgi sistemleriyle iletişime geçerek veri alması/vermesi sonucunda çok sayıda servise ihtiyaç duyulacaktır. Her bilgi sisteminin farklı yapıda olduğu düşünülürse, her bir sistemin kendini diğer sistemlere göre yapılandırması gerekir. Böyle karmaşık bir sistemde bir hata ile karşılaşınca hatanın nerde olduğunu bulmak bile zor olacaktır.



Şekil 3.1. Bilgi Sistemleri Arasındaki İlişki

Şekil 3.1.'de servisler sayesinde üniversitedeki sistemler birbiriyle olan ilişkileri gösterilmektedir. Öğrenci bilgi sistemi ve Portal sistemi veri kaynağı olduğu için diğer sistemler bu iki sistem tarafından oluşturulan web servisleri aracılığıyla personel ve öğrenci verilerine ulaşırlar. Bu sistem sayesinden personel ve öğrencilerin sistemlere kayıt olmak için bütün sistemleri tek tek dolaşmasına gerek kalmaz. Bu yöntem insanların işini kolaylaştırsa da içerisinde dijital bürokrasi barındırmaktadır. Dijital bürokrasi kurumun kaynaklarını israf edecektir. İsrafın nedeni kurumun sahip olduğu bütün sistemler hem öğrenci bilgi sistemi ile hem de portal sistemi ile ayrı ayrı eşleşmeler yapması gerekir. Bu eşleşmeler birim hiyerarşisi, unvan ve kişi gibi birçok alanda yapılması gerekir. Öğrenci bilgi sistemi ve Portal birbirinden farklı ve bağımsız sistemler olduğu için veri eşleşmeleri ayrı ayrı yapılmalıdır. Kuruma yeni kazandırılacak sistemlerin hem öğrenci bilgi sistemiyle hem de portal ile eşleşme yapması gerekecektir. Bu işlemlerde hem kaynak hem de zaman kaybına neden olacaktır.

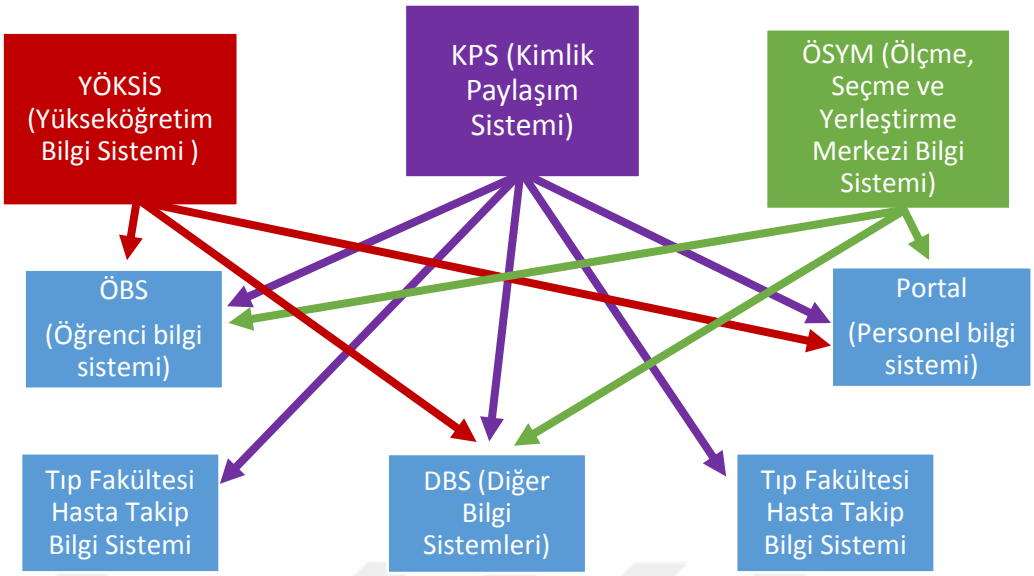
Bu web servislerle iletişimi daha optimal hale getirmek için ortak bir veri tabanı tasarlanması planlanmaktadır. Bu veri tabanı öğrenci, idari personel ve akademik personelin birim, unvan, özlük ve akıllı kart bilgilerini barındıracak şekilde tasarlanacaktır çünkü kurumda bulunan bilgi sistemlerinin (AKK, KBS, AKB gibi) ihtiyaç duyduğu temel bilgiler unvan, özlük ve akıllı kart bilgileridir.

Devlet kurumları tarafından veri paylaşımında bulunmak adına bazı web servis hizmetleri sunulmaktadır. Bu servisler ve servislerin sunduğu bilgiler **Tablo 3.3**.'de gösterilmiştir.

Tablo 3.3. Devlet Kurumları Tarafından Üniversiteye Sağlanan Web Servisler

Servis	**Kurum**	**Sunduğu Hizmet**
KPS (Kimlik Paylaşım Sistemi)	T.C. İçişleri Bakanlığı	Türk vatandaşlarının ve Türkiye'de yaşayan yabancı uyruklu kişilerin özlük ve adres bilgileri
YÖKSİS (Yükseköğretim Bilgi Sistemi)	Yüksek Öğretim Kurulu	Türkiye'deki bütün akademik personel, öğrenci bilgileri
ÖSYM (Ölçme, Seçme ve Yerleştirme Merkezi Bilgi Sistemi)	Ölçme, Seçme ve Yerleştirme Merkezi	ÖSYM tarafından yapılan bütün sınavların sonuç bilgileri

Üniversitenin bilgi sistemleri ihtiyaçları doğrultusunda bu web servis hizmetlerini kullanmaktadır. Üniversitenin sahip olduğu bilgi sistemlerinin hangi kurumun web servisleri kullandığı şeması **Tablo 3.3 Şekil 3.2**.'de verilmiştir.



Şekil 3.2. Devlet Kurumlarının Web Servisleri ile Bilgi Sistemleri Arasındaki İlişki.

Şekil 3.2.'de diğer devlet kurumlarının üniversiteye sağladığı web servis hizmetlerinden faydalanan bilgi sistemleri gösterilmektedir. Bilgi paylaşımı için sunulan bu web servislerin yetkilendirmeleri tek bir şifre ile sağlanmaktadır. Servisin şifre bilgisine sahip olan bilgi sistemi bütün veriye erişebilmektedir. Yani bir bilgi sistemine KPS şifresi verildiğinde bu bilgi sistemi KPS tarafından sağlanan bütün verilere (nüfus cüzdan bilgileri dahil) erişebilmektedir. Bu durum büyük güvenlik açıklarına sebebiyet verebilmektedir. Güvenlik açığının yanı sıra ÖBS tarafından KPS'den sorgulanan kişi bilgileri farklı bir sistem tarafından sorgulandığında fazladan iş yükü oluşturmaktadır.

3.2.1 Kullanılmakta Olan Bilgi Sistemlerinin Veri İhtiyaç Analizi

Ortak veri tabanı (OVT) tarafından barındırılacak verileri belirlemek için hangi bilgi sisteminde hangi bilgilerin var olduğu ve hangi bilgi sisteminin hangi verilere ihtiyaç duyduğunun belirlenmesi gerekir. **Tablo 3.4.**'de bilgi sistemlerinin ihtiyaç duyduğu veya sağlayabileceği veriler listelenmiştir.

Tablo 3.4. Bilgi Sistemlerinin İhtiyaç Duyduğu ve Sağlayabileceği Veriler.

Bilgi Sistemi	**İhtiyaç duyulan veri**	**Sağlayabileceği veri**
HGS	• Kişi özlük bilgileri (Kimlik Numarası, Sicil / Öğrenci Numarası, Ad, Soyad) • Birim - Unvan bilgileri • İletişim Bilgileri (Cep telefonu – Eposta adresi, Adresi)	
AKB	• Kişi özlük bilgileri (Kimlik Numarası, Sicil / Öğrenci Numarası, Ad, Soyad) • Birim - Unvan bilgileri • Kişinin fotoğrafı • İletişim Bilgileri (Cep telefonu – Eposta adresi, Adresi)	• Akıllı kart no (RFID)
KBS	• Kişi özlük bilgileri (Kimlik Numarası, Sicil / Öğrenci Numarası, Ad, Soyad) • Birim - Unvan bilgileri • Kart bilgileri (RFID) • Kişinin fotoğrafı • İletişim Bilgileri (Cep telefonu, Eposta adresi, Adresi)	
AKY	• Kişi özlük bilgileri (Kimlik Numarası, Sicil / Öğrenci Numarası, Ad, Soyad) • Birim - Unvan bilgileri • Kart bilgileri (RFID)	
AKK	• Kişi özlük bilgileri (Kimlik Numarası, Sicil / Öğrenci Numarası, Ad, Soyad) • Birim - Unvan bilgileri • Kart bilgileri (RFID)	

Tablo 3.4. (Devamı)

ÖBS		• Öğrenci özlük bilgileri (Kimlik Numarası, Öğrenci Numarası, Ad, Soyad) • Öğrenci Birim - Unvan bilgileri • Öğrenci İletişim Bilgileri (Cep telefonu – Eposta adresi, Adresi) • Öğrencinin fotoğrafı
PORTAL		• Personel özlük bilgileri (Kimlik Numarası, Sicil Numarası, Ad, Soyad) • Personel Birim - Unvan bilgileri • Personel İletişim Bilgileri (Cep telefonu – Eposta adresi, Adresi) • Personelin fotoğrafı

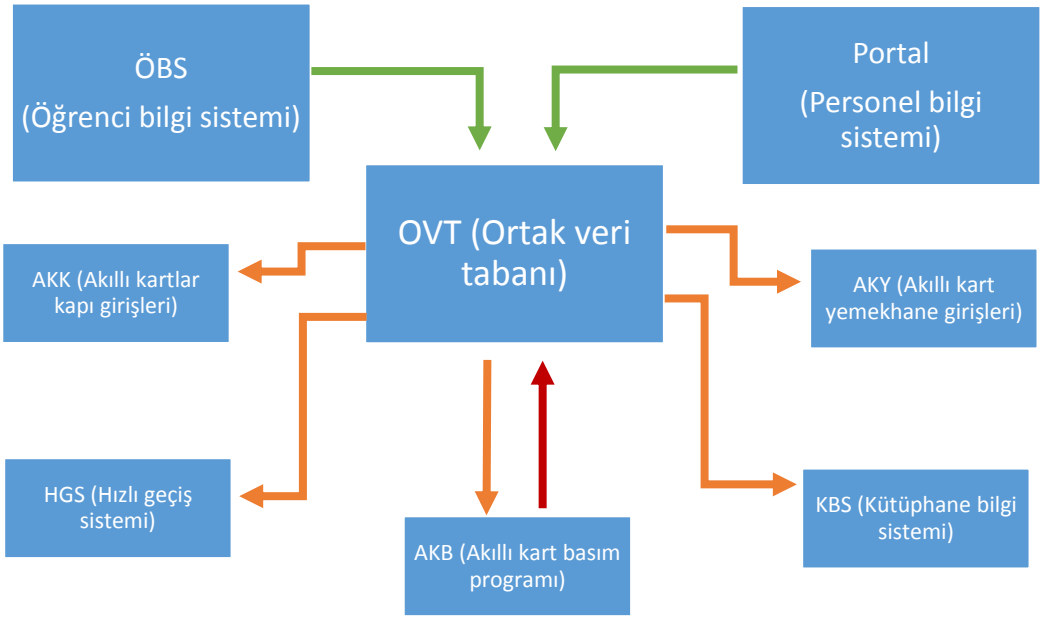
Tablo 3.4.'e göre ihtiyaç duyulan verilerin birleşim kümesinde; Kişi özlük bilgileri (Kimlik Numarası, Sicil / Öğrenci Numarası, Ad, Soyad), Birim - Unvan bilgileri, Kişinin fotoğrafı, İletişim Bilgileri (Cep telefonu – Eposta adresi, Adresi) ve Kart bilgileri (RFID) bulunmaktadır. OVT'yi oluştururken en az bu birleşim kümesindeki verileri saklayabilecek yapıda tabloların oluşturulması gerekmektedir.

DÖRDÜNCÜ BÖLÜM

UYGULAMA

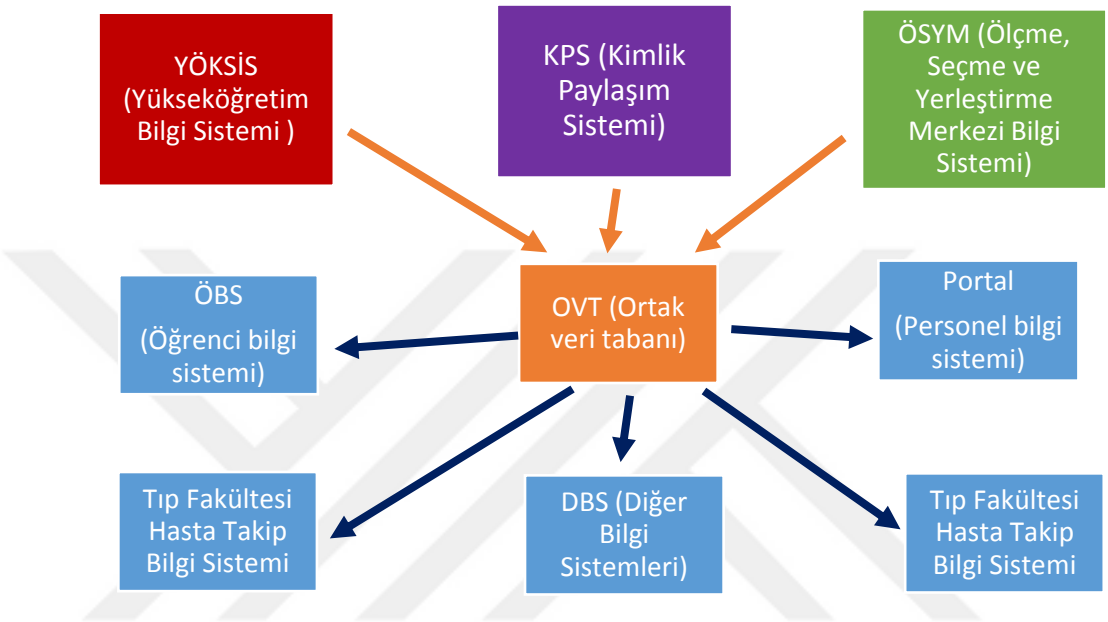
4.1. ORTAK VERİ TABANI

Oluşturulacak yeni sistemde ortak bir veri tabanı kullanılacaktır. ÖBS ve Portal sistemlerinde bulunan kişiler ortak veri tabanına aktarılacaktır. ÖBS ve Portal sistemlerinde yapılacak veri güncellemelerin anlık olarak ortak veri tabanına aktarımı sağlanacaktır. Ortak veri tabanında toplanan bu veriler üniversite tarafından kullanılan diğer sistemlere (AKK, KBS, AKB gibi) hem anlık hem de isteğe bağlı olarak zamanlanmış görevlerle web servisler aracılığıyla aktarılacaktır. Bu sayede, üniversite tarafından kullanılan sistemler her defasında hem ÖBS hem de Portal sistemleri ile eşleşmek zorunda kalmadan sadece OVT ile eşleşmeleri yeterli olacaktır. OVT oluşturulmadan önce kişilere verilen kart bilgilerinin (RFID) bir kısmı ÖBS sisteminde bir kısmı da Portal sisteminde bulunmaktaydı. OVT oluşturularak bütün kart bilgileri tek bir veri tabanında toplanması sağlanmıştır. Oluşturulacak yeni sistemi şeması **Şekil 4.1**.'de verilmiştir.



Şekil 4.1. OVT Sonrası Bilgi Sistemleri Arasındaki İlişki.

Devlet kurumlarının üniversiteye sağladığı servislerin şifrelerini diğer bilgi sistemleri ile paylaşmamak, ihtiyacı olan bilgi sistemine ihtiyacı kadar veriyi paylaşmak ve tekrarlanan sorguların hızını arttırmak için; web servislerin ortak veri tabanını üzerinden hizmet verilmesi planlanmıştır. Oluşturulacak yeni sistemin şeması **Şekil 4.2**'de verilmiştir.



Şekil 4.2. OVT Sonrası Kurum Servisleri ile Bilgi Sistemleri Arasındaki İlişki.

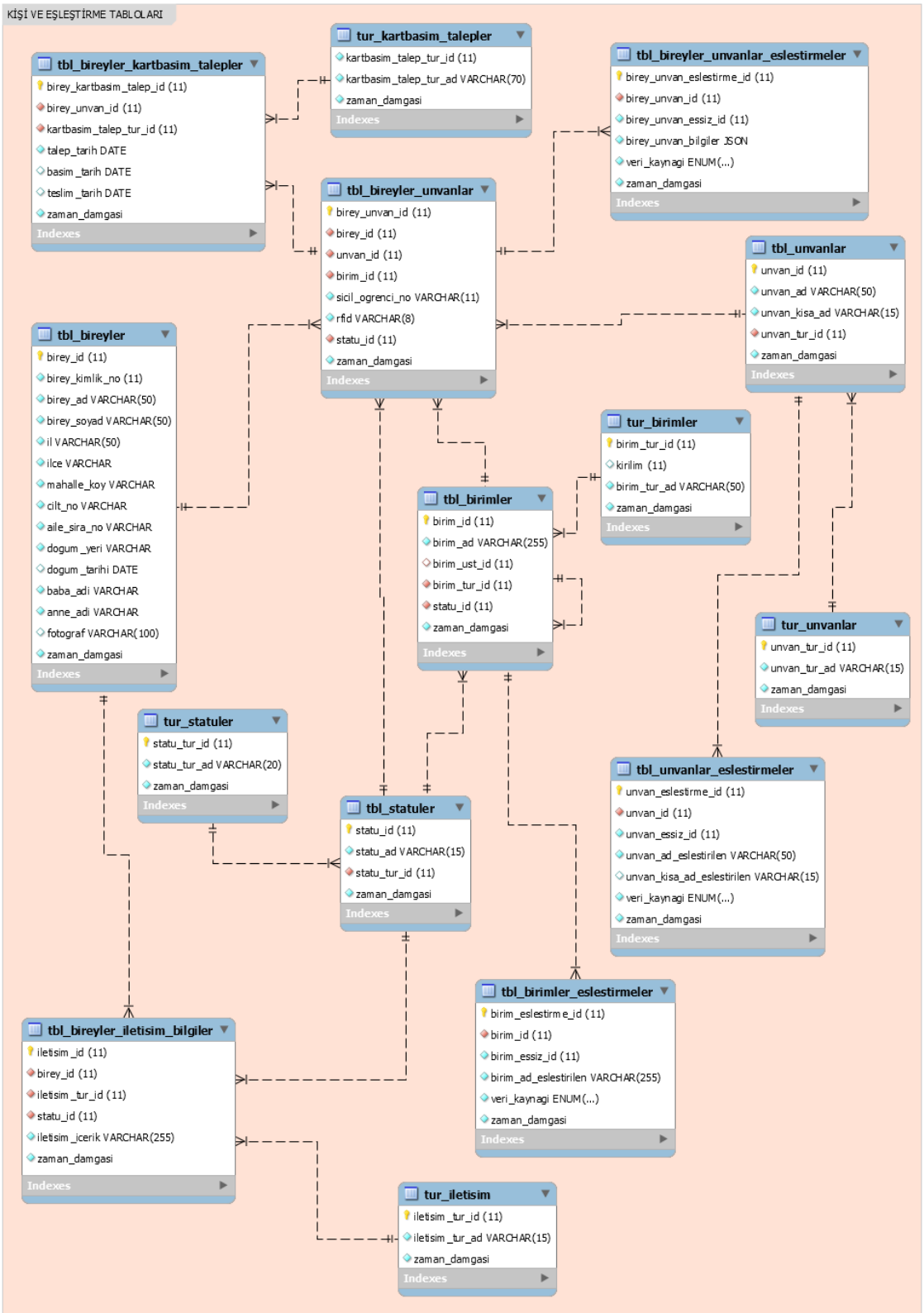
4.2. MATERYAL

Bu çalışmada web servisler oluşturulurken PHP tercih edilmiştir. Veri tabanı olarak ise PHP ile en iyi uyumda çalışan MYSQL seçilmiştir. Yine sunucu olarak ise hem PHP hem de MYSQL ile en iyi uyuma sahip Centos (Linux) işletim sistemi seçilmiştir. Web sunucu programı olarak Apache kullanılmıştır. Tabi, proje oluşturmada c# gibi diğer diller veya MSSQL gibi diğer veri tabanlarının kullanımında herhangi bir sakınca bulunmamaktadır. Proje oluşturulmasındaki ana düşünce servis odaklı mimari ile oluşturulması ve diğer sistemlerle çakışma yaşamadan çalışabilmesidir.

4.3. VERİ TABANI YAPISI

Ortak veri tabanı oluşturmadaki temel işlem veri tabanı yapısını iyi hazırlamaktır. Veri tabanı üzerinde daha sonradan yapılacak değişiklikler yazılacak bütün servisleri ve diğer bilgi sistemlerini etkileyeceği için OVT sistemini kullanacak çok sayıda bilgi sistemi olduğu için web servislerde veya veri tabanındaki yapılacak bir değişiklik bütün sistem yöneticilerine iletilmelidir.

Analiz kısmında OVT'nin ihtiyacı olan verilerin kişi özlük bilgileri, unvan bilgileri ve iletişim bilgileri olduğu belirlenmiştir. Ortak veri tabanından sadece kişilerin verilerinin tutulacağı alanlar değil aynı zamanda servis bilgileri, servislerde bulunan metotlar, kullanıcı bilgileri, unvan eşleştirmeleri ve birim eşleştirmeleri gibi daha farklı ihtiyaçlara yönelik tablolar da oluşturulacaktır.



Şekil 4.3. OVT Tabloları ve İlişkileri – 1.

Şekil 4.3'de OVT'nin kişi bilgilerini tutmak için ihtiyaç duyacağı bütün tablolar ve bu tablolar arasındaki ilişkiler bulunmaktadır. Şimdi de bu tabloların kullanım amaçları sunulacaktır.

tbl_bireyler: Bu tabloda kişilerin adı, soyadı, nüfusa kayıtlı olduğu ili gibi kişinin özlük bilgileri bulunmaktadır. Bu tablo başlangıç veya yola çıkış tablosu olarak adlandırılabilir. OVT'nin en temel tablosudur. Bu tablodaki birincil anahtar "birey_id" alanıdır ve bu alan auto increment yani otomatik artan numaradır.

OVT'de bulunan ve kişi bilgilerinin saklandığı (iletişim ve unvan gibi) diğer tabloların hepsi "tbl_bireyler" tablosunun "birey_id" alanıyla ilişkilidir.

birey_id	birey_kimlik_no	birey_ad	birey_soyad	il	ilce	mahalle_koy	cilt_no	aile_sira_no	dogum_yeri	dogum_tarihi	baba_adi	anne_adi	fotograf	zaman_damgasi
15562		AHMET		Muş	Muş merkez	Ulukaya			Muş				/foto/0f290c07bbcaf7f2d6ac267a...	
33421		Hatice		Bilecik	Bilecik merkez	Cumhuriyet			Bilecik				/foto/1b4337fc955095f4091c62b...	
34921		BUKET		Kırşehir	Mucur	Yörücek			Mucur				/foto/339e26645effe742512e9e4...	
17099		Nur		Erzurum	Yakutiye	Yukarı Hasanibasrı			ERZURUM				/foto/889d1cf6f47042d48e8c09d...	
34833		Aleyna		Trabzon	Akçaabat	Dörtyol-pazarok			Trabzon				/foto/875df84a814ca4c39c828ce...	
40336		GÖKHAN		Hatay	Kırıkhan	408 Evler			Kırıkhan				/foto/9b0af04c46f29da8ce41f370...	
28406		Furkan Oğuzhan		Erzurum	Yakutiye	Alpaşa			ERZURUM				/foto/a0a63a3201667bf3f6e310d...	
22592		GAMZE		Erzurum	Narman	Tuzla			Suruç				/foto/ecf4ccf61337e5c39fff01635...	
40242		Mustafa		Çankırı	Ilgaz	Yeni			Ilgaz				/foto/ca95899d32bafbb9f5e645f...	
21881		DERYA		Aksaray	Eskil	Karakol			Tortum				/foto/019d960217d5e0982de1d3...	
24230		Osman		Erzurum	Uzundere	Cömertler			Tortum				/foto/5f015ccee595f6ea6f7279b7...	
18103		REYHAN		Niğde	Niğde merkez	Gümüşler Ksb./es...			Gönen				/foto/1b6d363663dbc03f3cb0552...	
39640		AYSUN		Erzurum	Narman	Kışlaköy			Erzurum				/foto/47f824b055e790cccb4f14c6...	
19587		Sebahattin		Erzurum	Hınıs	Kayabaşı			Hınıs				/foto/3e6f45948439c945ffbe28b...	
40372		FETİH		Elazığ	Maden	Kayalar			Maden				/foto/49d665b60c9778a8eab5d2...	
32025		NECDET		Ardahan	Posof	Kır			Bakırköy				/foto/8e5c718cd86271b894643f6...	
24424		ÖZGE		Erzurum	Palandöken	Yenişehir			Erzurum				/foto/e1ca88ed6da19fb5068aa0d...	
24796		Selçuk		Erzurum	Yakutiye	Yeğenağa			ERZURUM				/foto/4440d1f4e2442d4c79d4e0f...	

Şekil 4.4. tbl_bireyler Tablosunun Örnek Görünümü.

Ayrıca oluşturulan her tabloda "zaman_damgasi" adında bir alan bulunmaktadır. Bu alan tabloya bir kayıt eklendiğinde veya güncellendiğinde o anki zamanı otomatik alarak (varsayılan değeri CURRENT_TIMESTAMP) yapılan son işlemin zamanını tutmaktadır. Bu alan sadece o satırdaki son yapılan işlemin zamanını tutmaktadır. Log tutma işlemlerinden ilerleyen bölümlerde bahsedilecektir.

Şekil 4.4.'deki örnek tablo görüntüsünde ve bundan sonraki örnek görüntülerde verilen bilgiler sadece gösterim amaçlı olup gerçek bilgiler değillerdir.

tbl_statuler: Bu tabloda kişinin, birimin ve kişi iletişim bilgisinin statülerini (aktif, pasif ve emekli gibi) saklamak için kullanılmaktadır.

Dört farklı tablo ile ilişkisi vardır. Bunlar; "tbl_bireyler_unvanlar", "tbl_birimler" ve "tbl_bireyler_iletisim_bilgiler" tablosunda bulunan "statu_id" alanı üzerinden ve "tur_statuler" tablosunda bulunan "statu_tur_id" alanı üzerindendir. Tabloda bulunan

“statu_tur_id” alanı statünün türünü yani bu statü iletişim bilgisine mi ait yoksa birim bilgisine mi ait olduğunu belirlemek için kullanılır.

statu_id	statu_ad	statu_tur_id	zaman_damgasi
1	Aktif	1	2019-03-25 16:33:33
2	Pasif	1	2019-03-25 16:33:33
3	Emekli	1	2019-03-25 16:33:33
4	Ders Almamış	1	2019-03-25 16:33:33
5	Mezun	1	2019-03-25 16:33:33
6	Aktif	2	2019-04-09 14:31:36
7	Pasif	2	2019-04-09 14:31:36
8	Yarı Aktif	2	2019-04-09 16:02:53
9	Pasif	3	2019-04-09 14:31:36
10	Aktif	3	2019-04-09 14:31:36
NULL	NULL	NULL	NULL

Şekil 4.5. tbl_statuler Tablosunun Örnek Görünümü.

tur_statuler: Bu tabloda statülerin (aktif, pasif gibi) türlerini (birim, iletişim veya birey birim) saklamak için kullanılmaktadır. Statü bilgisi genel olarak iki tanedir (aktif ve pasif) ancak kişilerde emekli veya birimlerde yarı aktif (yeni öğrenci almıyor ancak var olan öğrenciler devam ediyor) gibi farklı statüler bulunabilir. Statülerin birimler için mi yoksa kişiler için mi olduğu bilgisi bu tabloda saklanmaktadır.

Bu tablonun tek ilişkisi “tbl_statuler” tablosunda bulunan “statu_tur_id” alanı üzerindendir.

statu_tur_id	statu_tur_ad	zaman_damgasi
1	Birey Birim Statüler	2019-04-09 14:34:54
2	Birim Statüler	2019-04-09 14:34:54
3	İletişim Statüler	2019-04-09 14:34:54
NULL	NULL	NULL

Şekil 4.6. tur_statuler Tablosunun Örnek Görünümü.

tbl_bireyler_iletisim_bilgiler: Bu tabloda kişilerin e-posta, cep telefonu, sabit telefon veya adres gibi bütün iletişim bilgileri, bu bilgilerin türleri ve durumları (aktif veya pasif gibi) saklanmaktadır.

Bu tablo ile

- “tbl_bireyler” tablosunda bulunan “birey_id” alanı üzerinden,

- “tbl_statuler” tablosunda bulunan “statu_id” alanı üzerinden,
- “tur_iletisimler” tablosunda bulunan “tur_iletisim_id” alanı üzerinden ilişki kurulmuştur.

iletisim_id	birey_id	iletisim_tur_id	statu_id	iletisim_icerik	zaman_damgasi
1	3	1	10	Atatürk Üniversitesi Erzurum	2019-03-25 13:30:26
2	3	2	10	5325555555	2019-03-25 13:31:38
3	3	4	10	gokhan@atauni.edu.tr	2019-03-25 13:31:38
4	3	3	10	4422310000	2019-03-25 13:31:59
NULL	NULL	NULL	NULL	NULL	NULL

Şekil 4.7. tbl_bireyler_iletisim_bilgiler Tablosunun Örnek Görünümü.

tur_iletisim: Bu tabloda “tbl_bireyler_iletisim_bilgiler” tablosunda saklanacak iletişim bilgilerinin türleri (adres, e posta ve cep telefonu gibi) yer almaktadır.

Bu tablonun tek ilişkisi “tbl_bireyler_iletisim_bilgiler” tablosunda bulunan “iletişim_tur_id” alanı üzerindendir.

iletisim_tur_id	iletisim_tur_ad	zaman_damgasi
1	Adres	2019-03-22 14:56:07
2	Cep Telefonu	2019-03-22 14:56:07
3	Sabit Telefon	2019-03-25 13:29:41
4	E Posta	2019-03-25 13:29:41
NULL	NULL	NULL

Şekil 4.8. tur_ iletisim Tablosunun Örnek Görünümü.

tbl_bireyler_eslestirmeler: OVT’ye gelecek veriler iki farklı veri kaynağından geleceği için bir kişinin verisi farklı kişiler gibi birden fazla defa gelebilir. Veri tabanında aynı kaydın çoğullaşmaması için web servislerde gerekli kontroller yapılacaktır. Bu tablo daha önceden eşleştirilen bir kişiyi tekrar aynı kontrollere gerek kalmadan veya daha az kontrolden geçerek “tbl_bireyler” tablosuyla ilişkilendirebilmek için kullanılacaktır. Tabloda bulunan “birey_essiz_id” alanı bireyin veri kaynağındaki (ÖBS veya Portal) birincil anahtar değeridir.

Bu tablonun tek ilişkisi “tbl_bireyler” tablosunda bulunan “birey_id” alanı ile kurulmuştur.

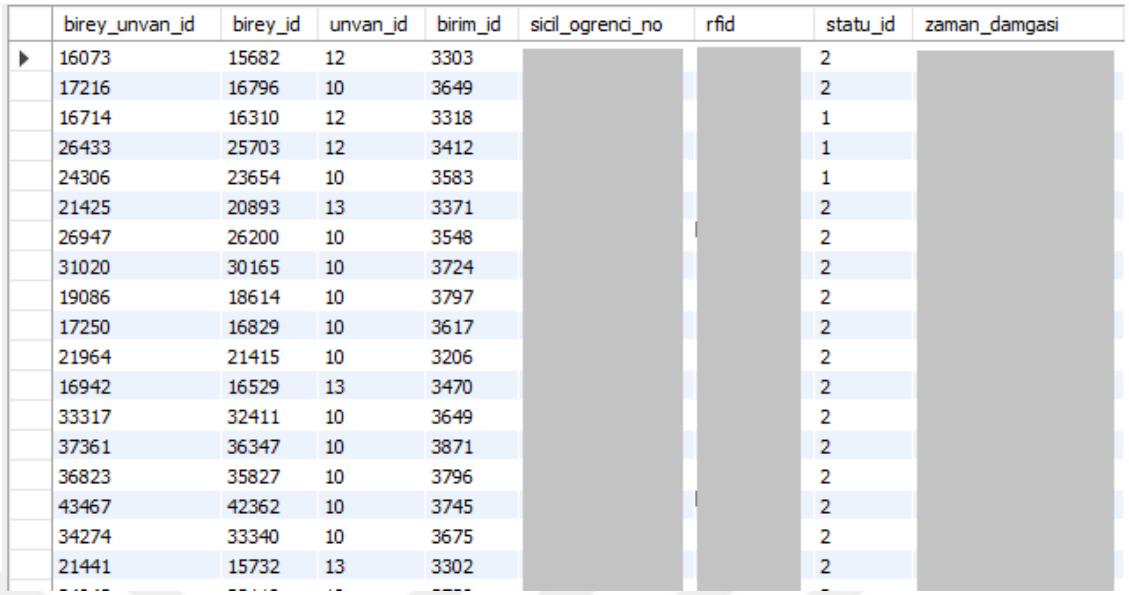
birey_eslestirme_id	birey_id	birey_essiz_id	birey_kimlik_no	birey_bilgiler	veri_kaynagi	zaman_damgasi
15491	15489	97894		{"il": "Erzurum", "ilce": "Horasan", "cilt_no": "95", "anne_adi":...	PORTAL	
15492	15490	100847		{"il": "Ağrı", "ilce": "Patnos", "cilt_no": "27", "anne_adi": "Şirin...	OBS	
15493	15491	100894		{"il": "Sinop", "ilce": "Durağan", "cilt_no": "28", "anne_adi": "H...	OBS	
15494	15492	100925		{"il": "Adana", "ilce": "Karaisalı", "cilt_no": "34", "anne_adi": "...	OBS	
15495	15493	100926		{"il": "Erzurum", "ilce": "Tortum", "cilt_no": "3", "anne_adi": "V...	OBS	
15496	15494	100929		{"il": "Erzurum", "ilce": "Yakutiye", "cilt_no": "76", "anne_adi":...	OBS	
15497	15495	100936		{"il": "Erzurum", "ilce": "Yakutiye", "cilt_no": "139", "anne_adi...	OBS	
15498	15496	100937		{"il": "Kayseri", "ilce": "Develi", "cilt_no": "69", "anne_adi": "S...	OBS	
15499	15497	100961		{"il": "Erzurum", "ilce": "Pasinler", "cilt_no": "52", "anne_adi": ...	OBS	
15500	15498	100962		{"il": "Samsun", "ilce": "Bafra", "cilt_no": "102", "anne_adi": "...	OBS	
15501	15499	100965		{"il": "Trabzon", "ilce": "Maçka", "cilt_no": "29", "anne_adi": "...	OBS	
15502	15500	100968		{"il": "Erzurum", "ilce": "Tortum", "cilt_no": "17", "anne_adi": "...	OBS	
15503	15501	100970		{"il": "Erzurum", "ilce": "İspir", "cilt_no": "23", "anne_adi": "Na...	OBS	
15504	15502	100972		{"il": "Muğla", "ilce": "Fethiye", "cilt_no": "67", "anne_adi": "F...	OBS	
15505	15503	100978		{"il": "Kayseri", "ilce": "Bünyan", "cilt_no": "45", "anne_adi": "...	OBS	
15506	15504	100981		{"il": "Erzurum", "ilce": "Horasan", "cilt_no": "94", "anne_adi":...	OBS	

Şekil 4.9. tbl_bireyler_eslestirmeler Tablosunun Örnek Görünümü.

tbl_bireyler_unvanlar: Bu tabloda kişilerin öğrencilik bilgileri, personel bilgileri, akıllı kart ilgileri (RFID), unvan ve bu unvana ait diğer bilgiler (sicil no, öğrenci no veya hangi birimde bulunduğu gibi) saklanmaktadır. Bu tablo adeta OVT'nin merkezinde bulunur ve en çok ilişki yapan tablolar arasındadır.

Bu tablo ile

- "tbl_bireyler" tablosunda bulunan "birey_id" alanı üzerinden,
- "tbl_statuler" tablosunda bulunan "statu_id" alanı üzerinden,
- "tbl_birimler" tablosunda bulunan "birim_id" alanı üzerinden,
- "tbl_bireyler_kartbasim_talepler" tablosunda bulunan "birey_unvan_id" alanı üzerinden,
- "tbl_bireyler_kartbasim_talepler" tablosunda bulunan "birey_unvan_id" alanı üzerinden,
- " tablosunda bulunan "birey_unvan_id" alanı üzerinden,
- "tbl_unvanlar" tablosunda bulunan "unvan_id" alanı üzerinden ilişki kurulmuştur.



birey_unvan_id	birey_id	unvan_id	birim_id	sicil_ogrenci_no	rfid	statu_id	zaman_damgasi
16073	15682	12	3303			2	
17216	16796	10	3649			2	
16714	16310	12	3318			1	
26433	25703	12	3412			1	
24306	23654	10	3583			1	
21425	20893	13	3371			2	
26947	26200	10	3548			2	
31020	30165	10	3724			2	
19086	18614	10	3797			2	
17250	16829	10	3617			2	
21964	21415	10	3206			2	
16942	16529	13	3470			2	
33317	32411	10	3649			2	
37361	36347	10	3871			2	
36823	35827	10	3796			2	
43467	42362	10	3745			2	
34274	33340	10	3675			2	
21441	15732	13	3302			2	

Şekil 4.10. tbl_bireyler_unvanlar Tablosunun Örnek Görünümü.

tbl_bireyler_unvanlar_eslestirmeler: OVT'ye gelecek veriler iki farklı veri kaynağından geleceği için bir kişinin unvan verisi farklı bilgiler gibi birden fazla defa gelebilir. Veri tabanında aynı kaydın çoğullaşmaması için web servislerde gerekli kontroller yapılacaktır. Bu tablo daha önceden eşleştirilen bir kişinin unvan bilgilerinin tekrar aynı kontrollere gerek kalmadan veya daha az kontrolden geçerek "tbl_bireyler" tablosuyla ilişkilendirebilmek için kullanılacaktır. Tabloda bulunan "birey_unvan_essiz_id" alanı kişi unvanının veri kaynağındaki (ÖBS veya Portal) birincil anahtar değeridir.

Bu tablonun tek ilişkisi "tbl_bireyler_unvanlar" tablosunda bulunan "birey_unvan_id" alanı ile kurulmuştur.

birey_unvan_eslestirme_id	birey_unvan_id	birey_unvan_essiz_id	birey_unvan_bilgiler	veri_kaynagi	zaman_damgasi
44133	44232	125961	{"birim": [{"birim_ad": "Tıp Fakültesi", "statu_id": ...	PORTAL	
44132	44231	125948	{"birim": [{"birim_ad": "Tıp Fakültesi", "statu_id": ...	PORTAL	
44131	44230	124344	{"birim": [{"birim_ad": "İktisadi ve İdari Bilimler Fa...	PORTAL	
44130	44229	122982	{"birim": [{"birim_ad": "Fen Fakültesi", "statu_id"...	PORTAL	
44129	44228	44625	{"birim": [{"birim_ad": "Edebiyat Fakültesi", "stat...	OBS	
44128	44227	44624	{"birim": [{"birim_ad": "İletişim Fakültesi", "statu_...	OBS	
44127	44226	44623	{"birim": [{"birim_ad": "Fen Fakültesi", "statu_id"...	OBS	
44126	44225	44622	{"birim": [{"birim_ad": "Açıköğretim Fakültesi", "st...	OBS	

Şekil 4.11. tbl_bireyler_unvanlar_eslestirmeler Tablosunun Örnek Görünümü.

tbl_unvanlar: Bu tabloda OVT'de bulunabilecek unvan (Profesör Doktor, Öğretim Görevlisi veya Lisans Öğrencisi gibi) listesi bulunmaktadır. Unvan adının çok

uzun olduğu durumlarda personel veya öğrenci kimlik kartlarına sığmamasına karşı "unvan_kisa_ad" alanı eklenmiştir.

Bu tablo ile

- "tbl_bireyler_unvanlar" tablosunda bulunan "unvan_id" alanı üzerinden,
- "tbl_unvanlar_eslestirmeler" tablosunda bulunan "unvan_id" alanı üzerinden,
- "tur_unvanlar" tablosunda bulunan "unvan_tur_id" alanı üzerinden ilişki kurulmuştur.

unvan_id	unvan_ad	unvan_kisa_ad	unvan_tur_id	zaman_damgasi
1	Profesör Doktor	Prof. Dr.	1	2019-03-25 16:25:38
2	Doçent Doktor	Doç. Dr.	1	2019-03-25 16:25:38
3	Doktor Öğretim Üyesi	Dr. Öğr. Üyesi	1	2019-03-25 16:25:38
4	Öğretim Görevlisi	Öğr. Gör.	1	2019-03-25 16:25:38
5	Araştırma Görevlisi	Arş. Gör.	1	2019-03-25 16:25:38
6	Bilgisayar İşletmeni	Bil. İşt.	2	2019-03-25 16:25:38
7	Hemşire	Hemş.	2	2019-03-25 16:25:38
8	Elektronik Mühendisi	Elektronik Müh.	2	2019-03-25 16:25:38
9	Tekniker	Tekn.	2	2019-03-25 16:25:38
10	Önlisans Öğrencisi	Önlisans Ögr.	3	2019-03-25 16:25:38
11	Lisans Öğrencisi	Lisans Ögr.	3	2019-03-25 16:25:38
12	Yüksek Lisans Öğren...	Y. Lisans Ögr.	3	2019-03-25 16:25:38
13	Doktora Öğrencisi	Doktora Ögr.	3	2019-03-25 16:25:38
NULL	NULL	NULL	NULL	NULL

Şekil 4.12. tbl_unvanlar Tablosunun Örnek Görünümü.

tur_unvanlar: Bu tablo "tbl_unvanlar" tablosunda bulunan unvanların türünü (akademik, idari ve öğrenci gibi) saklamak için kullanılır. Bu tablo liste veya rapor hazırlarken daha kısa ve hızlı sorgular kullanmak amacıyla oluşturulmuştur. Bu tablo kullanılmadan OVT'de bulunan bütün öğrenciler listelenmek istendiğinde, oluşturulacak sorguya öğrenci unvanına ait bütün "unvan_id" alanlarının yazılması gerekir. Bütün öğrenci unvanlarının yazılması da hem uzun ve yavaş sorgulara hem de küçük dikkatsizlikler sonucunda eksik verilere neden olabilir. Bu tablonun oluşturulması zorunlu değildir.

Bu tablonun tek ilişkisi "tbl_unvanlar" tablosunda bulunan "unvan_tur_id" alanı üzerindendir.

	unvan_tur_id	unvan_tur_ad	zaman_damgasi
▶	1	Akademik	2019-03-22 14:56:25
	2	İdari	2019-03-22 14:56:25
	3	Öğrenci	2019-03-25 16:18:18
*	NULL	NULL	NULL

Şekil 4.13. tur_unvanlar Tablosunun Örnek Görünümü.

tbl_unvanlar_eslestirmeler: Bu tablo ÖBS veya Portal yönetim bilgi sistemlerinden gelen unvan bilgilerinin OVT'deki unvanlarla eşleştirmek için kullanılmaktadır. Bu tablo kullanılarak dış kaynaktan gelen unvan bilgisiyle OVT'deki unvan bilgisi eşleştirilir veya dış kaynaktan gelen yeni unvan bilgisi OVT'ye eklenir. Tabloda bulunan "unvan_essiz_id" alanı unvanın kaynaktaki (ÖBS veya Portal) birincil anahtar değeridir.

Bu tablonun tek ilişkisi "tbl_unvanlar" tablosunda bulunan "unvan_id" alanı üzerindendir.

	unvan_eslestirme_id	unvan_id	unvan_essiz_id	unvan_ad_eslestirilen	unvan_kisa_ad_eslestirilen	veri_kaynagi	zaman_damgasi
▶	5	2	2	Doçent Doktor	Doç. Dr.	PORTAL	2019-04-18 14:00:49
	6	15	117	Sistem Uzmanı	Sis. Uzm.	PORTAL	2019-04-18 15:30:18
	7	12	31	Yüksek Lisans Öğrencisi	Yük. Lis. Öğr.	OBS	2019-04-18 15:46:37
	8	10	21	Önlisans Öğrencisi	Önl. Öğr.	OBS	2019-04-18 15:46:37
	9	10	33	Lisans Öğrencisi	Lis. Öğr.	OBS	2019-04-18 16:05:08
	10	13	43	Doktora Öğrencisi	Dokt. Öğr.	OBS	2019-04-18 16:05:09
	11	1	1	Profesör Doktor	Prof. Dr.	PORTAL	2019-04-18 22:41:09
	12	4	4	Öğretim Görevlisi	Öğr. Gör.	PORTAL	2019-04-18 22:41:12
	13	5	5	Araştırma Görevlisi	Araş. Gör.	PORTAL	2019-04-18 22:45:16
	14	3	3	Doktor Öğretim Üyesi	Dr. Öğr. Üyesi	PORTAL	2019-04-18 22:49:32
	15	16	126	Araştırma Görevlisi Do...	Araş. Gör. Dr.	PORTAL	2019-04-18 22:49:35
	16	7	34	Hemşire	Hemş.	PORTAL	2019-04-19 09:55:20
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Şekil 4.14. tbl_unvanlar_eslestirmeler Tablosunun Örnek Görünümü.

tbl_birimler: OVT'de bulunan birimlerin hepsini saklamak için kullanılan tablodur. Bu tabloda birimlerin adları, türleri (fakülte, enstitü, bölüm ve program gibi), hiyerarşik bilgileri ve durumları (aktif, pasif gibi) saklanmaktadır.

Bu tablonun dört adet farklı tablodaki alanla bir tane de kendi içindeki bir alanla ilişkisi bulunmaktadır.

Farklı tablolarla olan ilişkileri

- "tbl_bireyler_unvanlar" tablosunda bulunan "birim_id" alanı üzerinden,
- "tur_birimler" tablosunda bulunan "birim_tur_id" alanı üzerinden

- “tbl_birimler_eslestirmeler” tablosunda bulunan “birim_id” alanı üzerinden,
- “tbl_birimler_eslestirmeler” tablosunda bulunan “birim_id” alanı üzerindendir

Kendi içerisinde de “birim_id” alanıyla “birim_ust_id” alanı ilişkilendirilmiştir. Tablonun kendi içerisindeki ilişkilendirme, birimler arasındaki hiyerarşiyi kurabilmek için yapılmıştır.

birim_id	birim_ad	birim_ust_id	birim_tur_id	statu_id	zaman_damgasi
3774	Nanoelektronik	3240	10	6	2019-04-18 23:01:38
3742	Tarım Ekonomisi	3204	8	6	2019-04-18 23:00:54
3352	Temel İletişim Bilimleri	3351	10	6	2019-04-18 22:55:02
3379	Çocuk Gelişimi	3153	8	6	2019-04-18 22:55:17
3922	Resim	3830	8	6	2019-04-18 23:10:16
3889	Görsel İşitsel Teknikler ve Medya Yapımcılığı	3315	8	6	2019-04-18 23:06:00
4230	İktisat Teorisi	3566	10	6	2019-04-19 09:55:06
3853	Moleküler Biyoloji ve Genetik	3852	10	6	2019-04-18 23:04:22
3544	Veterinerlik Patolojisi	3359	10	6	2019-04-18 22:58:50
3492	Gıda İşleme	3315	8	6	2019-04-18 22:57:07
4084	Mikrobiyoloji	3219	10	6	2019-04-18 23:25:32
3312	İslam Tarihi ve Sanatları	3150	8	6	2019-04-18 22:54:51
3819	Sanat Tarihi (İ.Ö.)	3355	8	6	2019-04-18 23:02:59
3382	Halkla İlişkiler ve Tanıtım(İ.Ö.)	3381	10	6	2019-04-18 22:55:18
3657	Hemşirelik	3656	10	6	2019-04-18 22:59:31
3968	Resim (İ.Ö.)	3967	10	6	2019-04-18 23:17:46
3971	Heykel	3830	8	6	2019-04-18 23:17:48
3653	Radyo TV ve Sinema	3190	8	6	2019-04-18 22:59:30

Şekil 4.15. tbl_birimler Tablosunun Örnek Görünümü.

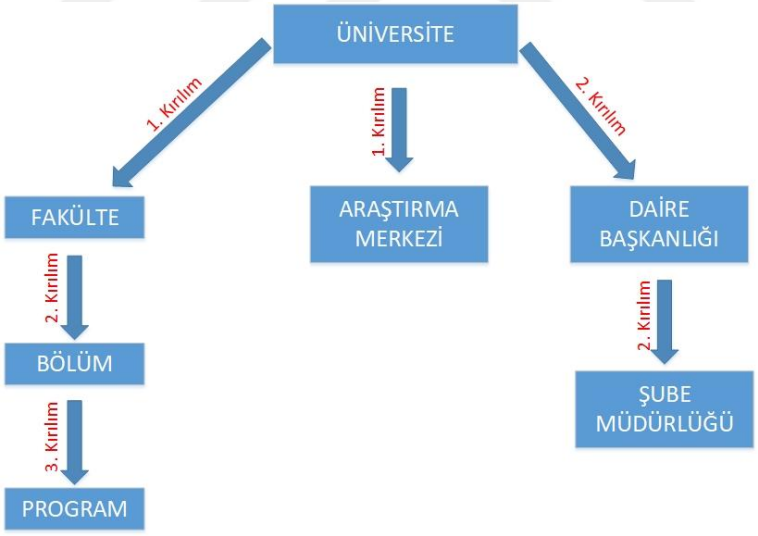
tur_birimler: OVT’de var olan birim türlerini (fakülte, bölüm, program ve enstitü gibi) ve hiyerarşideki yerlerini (kirilim) saklamak için kullanılan tablodur.

Bu tablonun iki ilişkisi bulunmaktadır. Bunlar; “tbl_birimler” tablosunda bulunan “birim_tur_id” alanı üzerinden ve “tbl_statuler” tablosunda bulunan “statu_id” alanı üzerindendir.

birim_tur_id	kirilim	birim_tur_ad	zaman_damgasi
1	NULL	Üniversite	2019-03-25 16:13:30
2	1	Fakülte	2019-03-25 16:13:30
3	1	Meslek Yüksek Okulu	2019-03-25 16:13:30
4	1	Yüksek Okul	2019-03-25 16:13:30
5	1	Konservatuar	2019-03-25 16:13:30
6	1	Araştırma Merkezi	2019-03-25 16:13:30
7	1	Daire Başkanlığı	2019-03-25 16:13:30
8	2	Bölüm	2019-03-25 16:13:30
9	2	Anabilim Dalı	2019-03-25 16:13:30
10	3	Program	2019-03-25 16:13:30
11	3	Bilim Dalı	2019-03-25 16:13:30
12	1	Enstitü	2019-03-26 13:56:44
NULL	NULL	NULL	NULL

Şekil 4.16. tur_birimler Tablosunun Örnek Görünümü

Tabloda bulunan "kirilim" alanınun amacı birim türünün (fakülte, bölüm ve program gibi) hiyerarşide kaçıncı basamakta yer aldığını belirlemek için kullanılmaktadır. Başka bir deyişle kırılım bir birimin, hiyerarşik olarak en üstte bulunan birimden uzaklığı olarak tanımlanabilir.



Şekil 4.17. Hiyerarşideki Kırılım

tbl_birimler_eslestirmeler: OVT'nin iki farklı veri kaynağı olduğu için bir birimin verisi farklı birimler gibi birden fazla defa gelebilir. Veri tabanında aynı kaydın çoğullaşmaması için web servislerde gerekli kontroller yapılacaktır. Bu tablo ise daha önceden eşleştirilen bir birimi tekrar aynı kontrollere gerek kalmadan veya daha az

kontrolden geçerek "tbl_birimler" tablosuyla ilişkilendirebilmek için kullanılacaktır. Tabloda bulunan "birim_essiz_id" alanı birimin kaynaktaki (ÖBS veya Portal) birincil anahtar değeridir.

Bu tablonun tek ilişkisi "tbl_birimler" tablosunda bulunan "birim_id" alanı üzerindedir.

birim_eslestirme_id	birim_id	birim_essiz_id	birim_ad_eslestirilen	veri_kaynagi	zaman_damgasi
3476	3153	1761	Açıköğretim Fakültesi	OBS	2019-04-18 22:54:18
4983	3153	118	Açıköğretim Fakültesi	PORTAL	2019-04-19 16:58:21
4552	4001	2333	Tarih	OBS	2019-04-18 23:22:57
3702	3302	343	Türk İslam Sanatları	OBS	2019-04-18 22:55:05
3658	3323	89	İktisat	OBS	2019-04-18 22:54:53
4014	3574	878	Tıp	OBS	2019-04-18 22:58:59
3604	3271	92	İlköğretim	OBS	2019-04-18 22:54:42
3825	3283	2586	Fars Dili ve Edebiyatı	OBS	2019-04-18 22:56:20
3687	3347	1946	İlköğretim Din Kültürü ve Ahlak Bilgisi Eğitimi Ana	OBS	2019-04-18 22:55:01
3573	3243	698	Tıbbi Farmakoloji	OBS	2019-04-18 22:54:35
4944	3536	413	Zootekni	PORTAL	2019-04-19 16:58:14
3563	3234	522	Tarım Politikası Ve Yayım	OBS	2019-04-18 22:54:33
3580	3249	20	Bitki Koruma	OBS	2019-04-18 22:54:36
4129	3683	1221	Endüstri Mühendisliği (İ.Ö.)	OBS	2019-04-18 22:59:52
3735	3226	243	Endüstri Mühendisliği	OBS	2019-04-18 22:55:18
3509	3186	449	Genel Biyoloji	OBS	2019-04-18 22:54:22
3846	3318	558	Yönetim ve Organizasyon	OBS	2019-04-18 22:56:30
4305	3841	1457	Uluslararası İlişkiler	OBS	2019-04-18 23:03:37
5009	4348	P.UNI.001.I...	Sağlık Kültür ve Spor Daire Başkanlığı	PORTAL	2019-04-19 16:58:26

Şekil 4.18. tbl_birimler_eslestirmeler Tablosunun Örnek Görünümü.

Şekil 4.18'da "birim_ad_eslestirlen" alanındaki bazı değerler (Açıköğretim Fakültesi) aynıdır. Bunun nedeni; Açıköğretim Fakültesi'nin OVT'ye hem ÖBS'den hem de Portal'dan sanki iki farklı birimler olarak gelmesidir. OVT bu iki birimin aslında aynı birim olduğuna karar verdiği için her ikisinin de "birim_id" alanına "3153" değerini yazmıştır. Yani ÖBS veri kaynağından "birim_essiz_id" değeri "1761" olan bir birim geldiğinde OVT bunun aslında kendinde bulunan ve "birim_id" alanı "3153" olan "Açıköğretim Fakültesi" olduğunu anlayacaktır. Aynı şekilde Portal veri kaynağından "birim_essiz_id" değeri "118" olan birim geldiğinde de OVT bunun kendindeki "birim_id" değeri "3153" olan "Açıköğretim Fakültesi" olduğunu anlayacaktır.

tbl_bireyler_kartbasim_talepler: Üniversitede bulunan öğrenciler ve personel kimlik kartlarını (RFID) kaybettiklerinde veya kartların bozulması/kırılması durumunda yeni kimlik kartı basımı için taleplerini çevrim için olarak ÖBS veya Portal bilgi sisteminden talep edebilirler. Bu talepler her ne kadar ÖBS veya Portal bilgi sisteminden yapılsa da talep kayıtları ve talep nedenleri OVT'de tutulmaktadır. OVT'de

bu talepleri akıllı kart basım programına (AKB) servisler aracılığıyla iletmektedir. Daha öncede bahsedildiği gibi akıllı kart basım programının kendine ait bir veri tabanı bulunmamaktadır. İhtiyaç duyduğu bütün veriler OVT tarafından karşılanmaktadır. Akıllı kart basım programının en çok ihtiyaç duyduğu veriler bu tabloda yer almaktadır. Bu tabloda ise kişilerin talep ettikleri kart basım taleplerinin kayıtları tutulmaktadır.

Bu tablonun iki ilişkisi bulunmaktadır. Bunlar; "tbl_bireyler_unvanlar" tablosunda bulunan "birey_unvan_id" alanı üzerinden ve "tur_kartbasim_talepler" tablosunda bulunan "kartbasim_talep_tur_id" alanı üzerindendir.

birey_kartbasim_talep_id	birey_unvan_id	kartbasim_talep_tur_id	talep_tarih	basim_tarih	teslim_tarih	zaman_damgasi
11	9	1	2019-04-11	NULL	NULL	2019-04-11 14:05:45
12	5	1	2019-04-11	2019-04-12	2019-04-18	2019-04-18 09:15:25
13	2	7	2019-04-01	2019-04-02	NULL	2019-04-02 10:51:35
14	6	6	2019-04-16	2019-04-17	2019-04-18	2019-04-18 17:59:45
15	11	4	2019-04-19	NULL	NULL	2019-04-19 16:10:36
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Şekil 4.19. tbl_bireyler_kartbasim_talepler Tablosunun Örnek Görünümü.

tur_kartbasim_talepler: Bu tabloda kişilerin hangi sebeplerle yeni kimlik kartı (RFID) basım talebinde bulunabileceklerini saklamak için kullanılan tablodur.

Kişiler ÖBS veya Portal bilgi sisteminden yeni kart basımı talep etmeleri durumunda karşılarına bu tablodaki liste çıkmaktadır. Kişiler kendilerine uygun olan seçeneği seçerek yeni kimlik kartı talebinde bulunabilirler. Bu türlerin veri tabanından tutulma amacı seçeneklerden bazılarında kart basımı ücretsiz iken bazılarında kimlik kartı (RFID) ücreti alınabilmektedir. Örnek olarak kişi tarafından "Daha önceden kartım çıkarılmadı" seçeneği seçilirse ve kişinin OVT'de daha önce basılmış kart bilgisi yoksa bu kişiden ücret talep edilmemektedir. Bu seçenekte ücret talep edilmemesinin nedeni kişinin kartının basılmamış olması kişiden kaynaklı bir sorun değildir. Kişi kartımı kaybettim seçeneğini seçerse kişiden belirli bir miktar kart ücreti talep edilmektedir. Bu seçenekte ücret alınmasının nedeni ise kişinin kartının kaybetmesi kişiden kaynaklı bir sorun olduğu içindir.

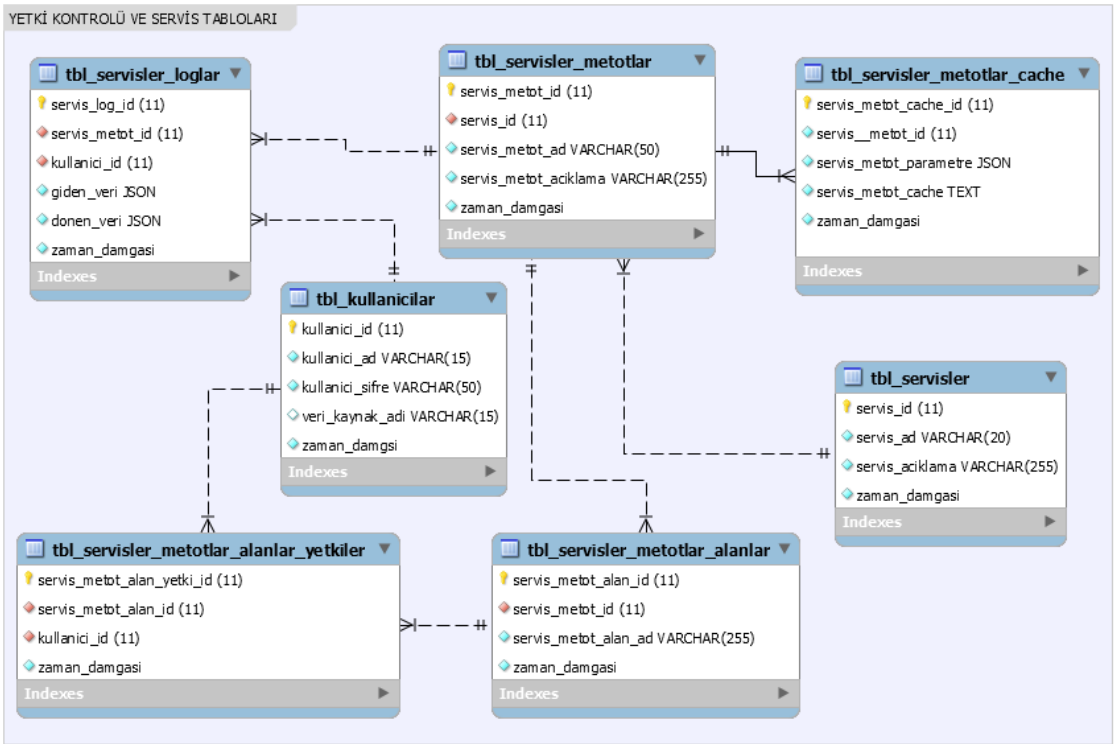
Sonuç olarak kişiler yeni kart basım talebinde bulunduğunda talep türüne göre ücret alınır veya alınmaz. Bu kart basım taleplerin sebeplerini OVT'de tutulması gerekmektedir.

Bu tablonun tek ilişkisi “tbl_bireyler_kartbasim_talepler” tablosunda bulunan “birey_kartbasim_talep_id” alanı üzerindedir.

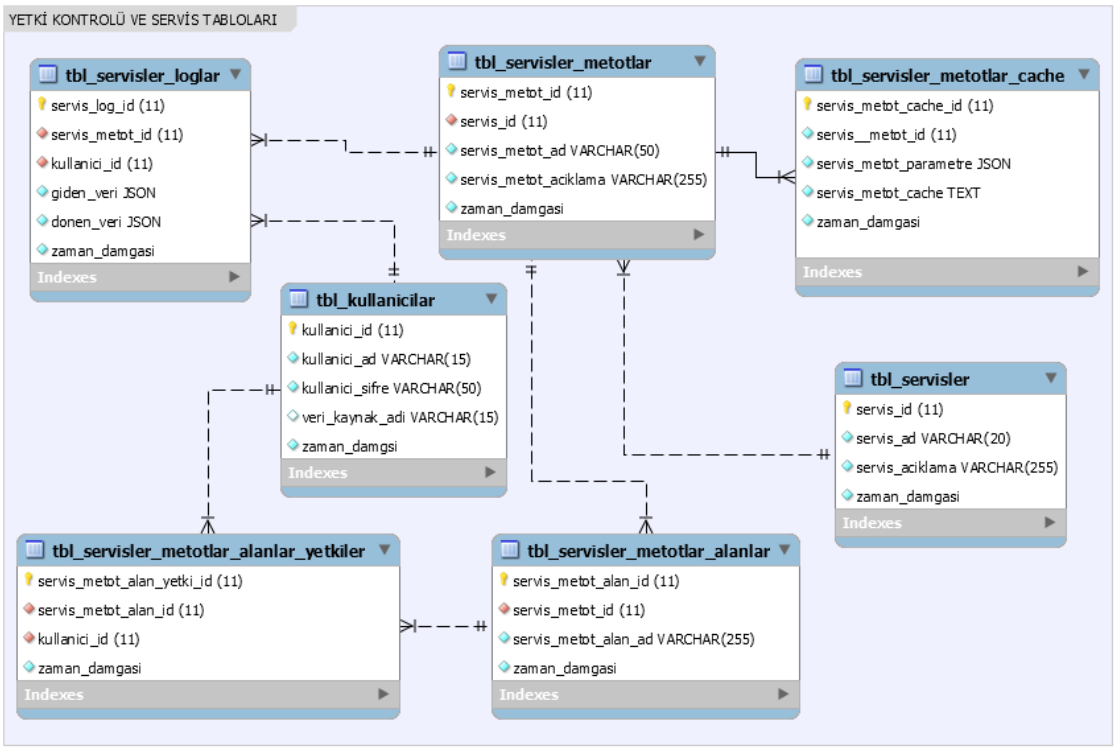
kartbasim_talep_tur_id	kartbasim_talep_tur_ad	zaman_damgasi
1	Daha önce adıma kart çıkartılmadı.	2019-04-11 13:38:16
2	Kartımı kaybettim.	2019-04-11 13:38:16
3	Kartımı çaldırdım.	2019-04-11 13:38:16
4	Kartım ilk kullanımdan beri arızalı.	2019-04-11 13:38:16
5	Kartım kırıldı.	2019-04-11 13:38:16
6	Kart üzerindeki bilgi değişikliği (TC, Öğrenci No, ...	2019-04-11 13:38:16
7	Ünvan Değişikliği	2019-04-11 13:38:16
8	Kart üzerindeki fotoğraf değişikliği	2019-04-11 13:38:16
NULL	NULL	NULL

Şekil 4.20. tur_kartbasim_talepler Tablosunun Örnek Görünümü.

Şimdiye kadar anlatılar sadece OVT’de kişi, birim, unvan veya iletişim bilgilerinin saklanabilmesi için gerekli olan tablolardır. OVT’de sadece bu bilgileri değil aynı zamanda servis, kullanıcı/şifre veya yetki bilgileri gibi OVT’nin güvenliğini sağlayacak tablolar da bulunmaktadır.



Şekil 4.21. OVT Tabloları ve İlişkileri – 2.



Şekil 4.21.'da OVT'de kullanıcı, yetki ve servis bilgilerini tutmak için ihtiyaç duyacağı bütün tablolar ve bu tablolar arsındaki ilişkiler bulunmaktadır. Şimdi de bu tabloların kullanım amaçları sunulacaktır.

tbl_kullanicilar: Bu tabloda web servislere yetkili olan bütün bilgi sistemlerinin kullanıcı adı, şifresi veya veri kaynağına verilen ad (ÖBS veya POSTAL gibi) bilgileri saklanmaktadır.

Bu tablonun iki farklı tablo ile ilişkisi bulunmaktadır. Bunlar; "tbl_servisler_loglar" tablosunda bulunan "kullanici_id" alanı üzerinden ve "tbl_servisler_alanlar_yetkiler" tablosunda bulunan "kullanici_id" alanı üzerindendir.

kullanici_id	kullanici_ad	kullanici_sifre	veri_kaynak_adi	zaman_damgsi
1	OBS-GIRIS	7793d599ac04c7f586d6643c14c0fca8	OBS	2019-03-27 13:26:10
2	Portal-GIRIS	beffd539bb7065192f50c5aabf7cbec4	PORTAL	2019-03-27 13:26:10
3	Kutuphane	94b9224dcab60b8ee5ea4f6a889c27f8	NULL	2019-03-27 13:26:10
4	HGS	e99ab0614e2e3cf753e6432869f3b181	NULL	2019-03-27 13:26:10
5	KART_BASIM	2e7654fa390f9a87e2f75c8d8c2d936b	NULL	2019-03-27 13:26:10
NULL	NULL	NULL	NULL	NULL

Şekil 4.22. tbl_kullanicilar Tablosunun Örnek Görünümü.

tbl_servisler: Bu tablo OVT'de var olan servisleri ve bu servislerin bilgilerini saklamak için kullanılmaktadır.

Bu tablonun tek ilişkisi "tbl_servisler_metotlar" tablosunda bulunan "servis_id" alanı üzerindedir.

servis_id	servis_ad	servis_aciklama	zaman_damgasi
1	BilgiAktarimServis	OVT'ye kişi, birim veya ünvan bilgilerini aktarmak...	2019-03-27 13:34:44
2	BilgiSorgulamaServis	OVT'de olan bilgileri sorgulamak için kullanılan se...	2019-03-27 13:34:44
NULL	NULL	NULL	NULL

Şekil 4.23. tbl_servisler Tablosunun Örnek Görünümü.

tbl_servisler_metotlar: Bu tabloda serviste bulunan metotlar ve bu metotların bilgileri saklanmaktadır.

Bu tablo ile

- "tbl_servisler" tablosunda bulunan "servis_id" alanı üzerinden,
- "tbl_servisler_metotlar_alanlar" tablosunda bulunan "servis_metot_id" alanı üzerinden,
- "tbl_servisler_metotlar_cache" tablosunda bulunan "servis_metot_id" alanı üzerinden,
- "tbl_servisler_loglar" tablosunda bulunan "servis_metot_id" alanı üzerinden ilişki kurulmuştur.

servis_metot_id	servis_id	servis_metot_ad	servis_metot_aciklama	zaman_damgasi
5	1	kaydetKisiOzlukBilgiler	kişinin özlük bilgilerini kaydeden metot	2019-03-27 13:51:46
6	2	getirKisiKartBilgilerByKimlikNo	kişinin kart bilgilerini kimlik numarasına göre getir...	2019-03-27 13:51:46
7	2	getirKisiBilgilerByKimlikNo	kişinin bilgilerini kimlik numarasına göre getiren s...	2019-03-27 13:51:46
8	2	getirKisiBilgilerByKartNo	kişinin bilgilerini kart numarasına göre getiren se...	2019-03-27 13:51:46
NULL	NULL	NULL	NULL	NULL

Şekil 4.24. tbl_servisler_metotlar Tablosunun Örnek Görünümü.

tbl_servisler_metotlar_cache: Bu tabloda devlet kurumları tarafından sunulan web servislerin sonuçları saklanmaktadır. Bu tablonun oluşturulma amacı aynı parametre ile sorgulanan devlet kurumu servislerinin cevap sürelerini azaltmaktır. Tablo günlük olarak boşaltılmaktadır.

Bu tablo ile "tbl_servisler_metotlar" tablosunda bulunan "servis_metot_id" alanı üzerinden ilişki kurulmuştur.

tbl_servisler_metotlar_alanlar: Bu tabloda serviste bulunan metotlar tarafından dönen alanlar saklanmaktadır.

Bu tablo ile

- "tbl_servisler_metotlar" tablosunda bulunan "servis_metot_id" alanı üzerinden,
- "tbl_servisler_metotlar_alanlar_yetkiler" tablosunda bulunan "servis_metot_alan_id" alanı üzerinden ilişki kurulmuştur.

servis_metot_alan_id	servis_metot_id	servis_metot_alan_ad	zaman_damgasi
1	7	Ad	2019-08-05 00:44:56
2	7	Soyad	2019-08-05 00:44:56
3	7	CiltNo	2019-08-05 00:44:56
4	7	AileSiraNO	2019-08-05 00:44:56
NULL	NULL	NULL	NULL

Şekil 4.25. tbl_servisler_alanlar_metotlar_alanlar Tablosunun Örnek Görünümü.

tbl_servisler_metotlar_alanlar_yetkiler: Bu tabloda OVT'de kayıtlı kullanıcıların hangi metotların hangi kolonlarına yetkili olduğu bilgisi saklanmaktadır.

Bu tablonun iki farklı tablo ile ilişkisi bulunmaktadır. Bunlar; “tbl_kullanicilar” tablosunda bulunan “kullanici_id” ve “tbl_servisler_metotlar_alanlar” tablosunda bulunan “servis_metot_alan_id” alanı üzerindendir.

servis_metot_alan_yetki_id	servis_metot_alan_id	kullanici_id	zaman_damgasi
1	1	3	2019-03-27 14:15:22
2	1	1	2019-03-27 14:15:22
3	2	2	2019-03-27 14:15:22
4	3	4	2019-03-27 14:15:22
NULL	NULL	NULL	NULL

Şekil 4.26. tbl_servisler_metotlar_alanlar_yetkiler Tablosunun Örnek Görünümü.

tbl_servisler_loglar: Bu tabloda OVT’de bulunan servis ve servis metotlarının çalıştırılmasında metodu çalıştıran kullanıcı, metoda gelen ve metottan dönen veri gibi log bilgileri tutulmaktadır. Gelen ve dönen verilerin büyüklükleri değişkenlik gösterebilir. Yani bir serviste istenen veri sayısı 10 iken başka serviste 2 olabilir. Bundan dolayı her bir veri için ayrı alan tanımlanmamış bunun yerine gelen ve giden veri olarak iki “json” tipinde alan tanımlanmıştır. Alan tipinin “json” olmasının diğer avantajı ise; bir alanın içerisindeki verileri sanki ayrı bir alanlar tanımlanmış gibi arama yapmaya olanak tanımasıdır.

Bu tablonun iki farklı tablo ile ilişkisi bulunmaktadır. Bunlar; “tbl_kullanicilar” tablosunda bulunan “kullanici_id” ve “tbl_servisler_metotlar” tablosunda bulunan “servis_metot_id” alanı üzerindendir.

servis_log_id	servis_metot_id	kullanici_id	giden_veri	donen_veri	zaman_damgasi
1	5	1	{"PROFIL_OZLUK_BILGILER": {"il": "Erzurum", "il...	{"hata": 0, "mesaj": "Veriler kaydedildi"}	2019-03-27 14:31:37
2	5	1	{"PROFIL_OZLUK_BILGILER": {"il": "Ankara", "ilc...	{"hata": 0, "mesaj": "Veriler kaydedildi"}	2019-03-27 14:31:37
3	8	3	{"kart_no": "4D335FC6"}	{"hata": 0, "mesaj": "Veriler listelendi", "profil_a...	2019-03-27 14:37:18
4	6	4	{"kimlik_no": "1111111111112"}	{"hata": 0, "mesaj": "Veriler listelendi", "kart_no...	2019-03-27 14:37:18
5	5	2	{"PROFIL_OZLUK_BILGILER": {"il": "Erzurum", "il...	{"hata": 0, "mesaj": "Veriler kaydedildi"}	2019-03-27 14:38:37
6	7	4	{"kimlik_no": "44444444440"}	{"hata": 0, "mesaj": "Veriler listelendi", "kart_no...	2019-03-27 14:57:18
NULL	NULL	NULL	NULL	NULL	NULL

Şekil 4.27. tbl_servisler_loglar Tablosunun Örnek Görünümü.

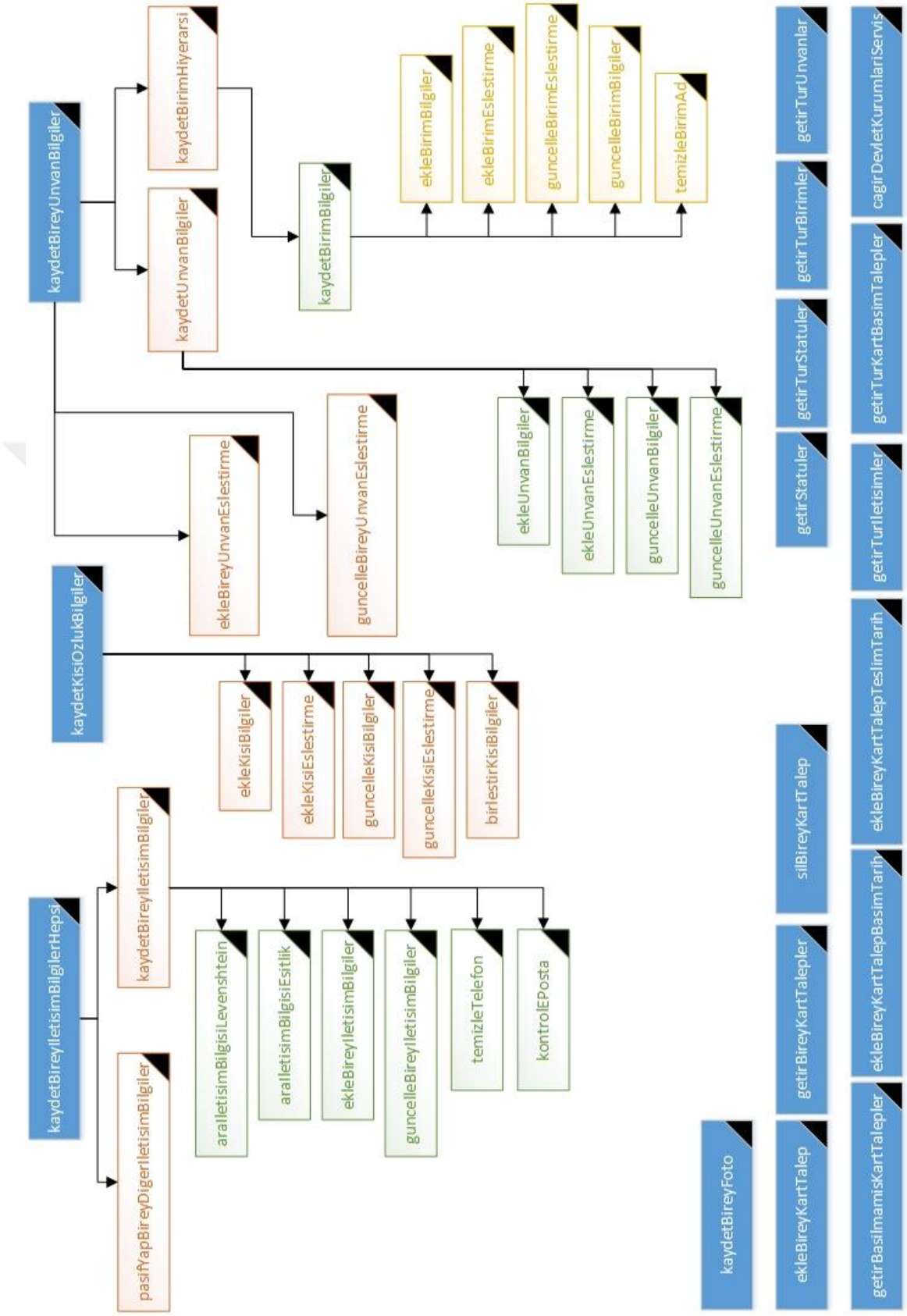
4.4. SERVİSLER

OVT’de genel olarak iki tip servis olacaktır. Bunları veri alan ve veri gönderen servis olarak tanımlayabiliriz. Veri alan servis ÖBS veya Portal tarafından gönderilen verileri gerekli kontrollerden geçirdikten sonra OVT’ye kaydeden servislerdir. Veri

gönderen servisler ise; OVT'nin diğer paydaşlarına (AKK, AKY, KBS ve AKB gibi) belirli kıstaslara göre kişi verilerini paylaşan servislerdir.

4.4.1. Veri Alan Servis

Bu servise kişilerin üç temel veri grubundaki bilgileri gelmektedir. Bu üç temel bilgi grubunun içerisinde kişi özlük bilgileri (adı, soyadı, ili, ilçesi ve doğum tarihi gibi), kişi unvan bilgileri (unvanın ne olduğu, hangi birimde olduğu, unvanın statüsü ve sicil numarası gibi), kişinin iletişim bilgileri (cep telefonu, adresi ve e posta adresi gibi) bulunmaktadır. Her veri grubundaki bilgiler kaydedilmeden önce bazı denetimlerden geçmektedir. Bu denetimler kayıtların çoğullaşmaması için veya veri bütünlüğünü sağlamak için yapılmaktadır. OVT'de oluşabilecek en büyük problemlerden bir tanesi veri çoğullaşmasıdır. Veri çoğullaşmasını engellemek veri grubuna göre farklı taktikler uygulanacaktır. Örnek olarak kişi özlük bilgilerinde veri çoğullaşmasını engellemek için kimlik numaraları üzerinden kontroller yapılacaktır ancak unvan, birim veya iletişim bilgilerinde kimlik numarası gibi eşsiz bir değer bulunmamaktadır. Bundan dolayı unvan ve iletişim bilgilerinde ise Levenshtein Distance algoritması gibi algoritmalar kullanılmıştır.



Şekil 4.28. Oluşturulacak Fonksiyonlar ve İlişkileri.

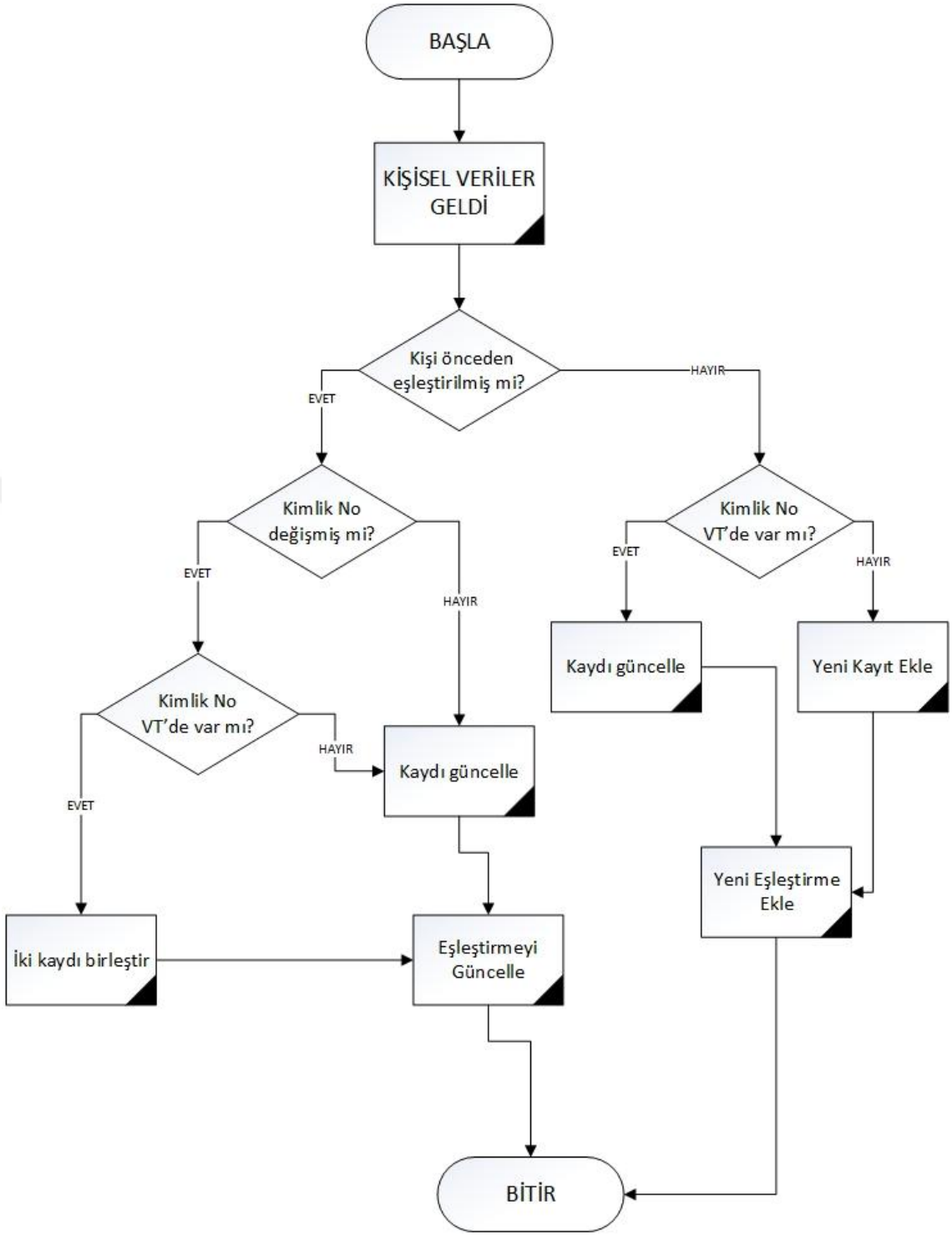
Veri alan servis, bilgi grubuna (özlük, unvan ve iletişim) göre parçalanarak anlatılacaktır. Veri alan servisin en önemli kısmı gelen verilerin denetlenme işlemidir çünkü eğer denetlenme işlemi başarısız olursa veri tabanının bütünlüğü korunamayabilir. Oluşturulacak web servisler içerisinde çeşitli metotlar bulunmaktadır. Servislerde bulunacak bu metotlar fonksiyonlara bölünmüştür. Oluşturulacak bütün fonksiyonlar dışarıdan erişime açık değildir. Bazıları sadece OVT'nin iç işleyişi için kullanılacaktır. Dış erişime açık olan fonksiyonlar "public" türünde tanımlanmıştır. OVT'nin kendi iç işleyişi için kullanılacak fonksiyonlar ise "private" tipinde tanımlanmıştır. **Şekil 4.28.**'de oluşturulacak fonksiyonlar ve bu fonksiyonlar arasındaki ilişki verilmiştir. Mavi renkli kutucuklar "public" fonksiyonlarını temsil etmektedir. Diğer bütün kutucuklar ise "private" fonksiyonları temsil etmektedir.

4.4.1.1. Özlük Bilgisi Denetimleri ve Fonksiyonları

Kişinin özlük bilgileri denetimi KPS servisleri tarafından sağlanan kimlik numarası sayesinde çok kolay hale gelmiştir. KPS servisleri tarafından verilen kimlik numarası çok özel bir durum (mahkeme kararı gibi) olmadıkça değişmemektedir. ÖBS'ye yeni kayıt yaptıran yabancı uyruklu öğrencilerin durumu biraz daha farklıdır. Yabancı uyruklu öğrenciler üniversiteye kayıt hakkı kazandıklarında, bu öğrencilere KPS tarafından bir yabancı uyruk numarası verilmemiş olabilir. Yabancı uyruk numarası olmayan öğrencilere üniversitenin talebi doğrultusunda YÖK tarafından 98 ile başlayan 11 haneli yabancı uyruk numarasına benzer bir numara verilmektedir. Kişi daha sonra yabancı uyruk numarası alırsa kimlik numarası değişmektedir. Bu da OVT'de kayıtların çoğullaşması anlamına gelebilmektedir.

Atatürk Üniversitesi Türkiye'nin en eski üniversiteleri arasındadır. Üniversite 1957 yılında kurulduğunda bilgi sistemleri mevcut değildi. Bundan dolayı üniversitenin dijital ortama olmayan ancak arşivde bulunan kayıtlar bulunmaktaydı. Bu kayıtların dijital ortama aktarılması için Atatürk Üniversitesi tarafından gerekli çalışmalar başlattırılarak fiziksel arşivin çok büyük bir kısmı dijital ortama aktarılmıştır. Dijitalleşme çalışmaları kapsamında verilerin eski olması nedeniyle bazı öğrencilerin kimlik numaraları ilk başta bulunamamış sonraki çalışmalar sıranda bulunmuştur.

Sonraki çalışmalar sırasında bulunan kimlik numaraları OVT'de kayıt çoğullaşmasına sebep olabilir.



Şekil 4.29. Kişi Eşleştirme Akış Şeması.

OVT'deki veri alan servisinde kişi verilerinin çoğullaşması sorunun engellemek

için **Şekil 4.29**.'daki akış şeması uygulanmıştır. Akış şemasının (Bkz. **Şekil 4.29.**) uygulanması için oluşturulacak fonksiyonlar "private" ve "public" olarak üzere ikiye ayrılmışlardır. Private olan fonksiyonlar OVT'nin kendi iç işlerini yapmak için kullanılmaktadır. İki tane olan public fonksiyonlar ise yani "kaydetKisiOzlukBilgiler" ve "kaydetBireyFoto" diğer bilgi sistemleri tarafından çalıştırılması için oluşturulmuştur.

Şekil 4.29.'daki akış şemasını koda dökmeden önce kişi özlük bilgilerini güncelleyen, yeni kişi oluşturan, eşleşme bilgilerini güncelleyen, eşleşme bilgilerini kaydeden ve kayıt çoğullaşması ihtimaline karşı iki kaydı tekilleştiren fonksiyonların oluşturulması gerekmektedir.

```
private function ekleKisiBilgiler( $ozluk )
{
    $sorgu = $this->db->prepare ("INSERT INTO tbl_bireyler
(birey_kimlik_no, birey_ad, birey_soyad, il, ilce, mahalle_koy,
cilt_no, aile_sira_no, dogum_yeri, dogum_tarihi, baba_adi,
anne_adi) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)");

    return
        $sorgu->execute(array(
            $ozluk["birey_kimlik_no"],
            $ozluk["birey_ad"],
            $ozluk["birey_soyad"],
            $ozluk["il"],
            $ozluk["ilce"],
            $ozluk["mahalle_koy"],
            $ozluk["cilt_no"],
            $ozluk["aile_sira_no"],
            $ozluk["dogum_yeri"],
            $ozluk["dogum_tarihi"],
            $ozluk["baba_adi"],
            $ozluk["anne_adi"]
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birey_id"  => $this->db->lastInsertId()
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birey_id"  => null
            ) ;
}
```

Şekil 4.30. Kişi Ekleme Fonksiyonu.

Şekil 4.30.'deki fonksiyon OVT'ye yeni kişi ekleme fonksiyonudur. Fonksiyon dizi $ozluk adındaki dizi tipi bir değişken almaktadır. Bu değişkenin içerisinde kişinin bütün özlük bilgileri beklenmektedir. Fonksiyon gelen kişi bilgilerine hiçbir müdahalede bulunmadan OVT'ye eklemektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birey_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi eklenmiş anlamına gelmektedir. Eğer kişi eklemede sorun çıkmamış ise dönütte "birey_id" anahtarı son eklenen id değerini vermektedir.

```
private function guncelleKisiBilgiler( $birey_id, $ozluk )
{
    $sorgu = $this->db->prepare("UPDATE tbl_bireyler SET
birey_kimlik_no = ?, birey_ad = ?, birey_soyad = ?, il = ?, ilce =
?, mahalle_koy = ?, cilt_no = ?, aile_sira_no = ?, dogum_yeri = ?,
dogum_tarihi = ?, baba_adi = ?, anne_adi = ? WHERE birey_id = ?");

    return
        $sorgu->execute(array(
            $ozluk["birey_kimlik_no"],
            $ozluk["birey_ad"],
            $ozluk["birey_soyad"],
            $ozluk["il"],
            $ozluk["ilce"],
            $ozluk["mahalle_koy"],
            $ozluk["cilt_no"],
            $ozluk["aile_sira_no"],
            $ozluk["dogum_yeri"],
            $ozluk["dogum_tarihi"],
            $ozluk["baba_adi"],
            $ozluk["anne_adi"],
            $birey_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birey_id"  => $birey_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birey_id"  => null
            ) ;
}
```

Şekil 4.31. Kişi Güncelleme Fonksiyonu.

Şekil 4.31.'deki fonksiyon OVT'de bulunan bir kişinin bilgilerini güncelleme fonksiyonudur. Fonksiyon iki tane parametre almaktadır. Bunlar; $birey_id adındaki integer tipi değişken ve $ozluk adındaki dizi tipi değişkenlerdir. Adı $ozluk olan

değişkenin içerisinde kişinin bütün özlük bilgileri beklenmektedir, adı $birey_id olan değişkende ise kişinin OVT'deki "tbl_bireyler" tablosundaki eşsiz alan olan "birey_id" değeri beklenmektedir. Fonksiyon gelen kişi bilgilerine hiçbir müdahalede bulunmadan OVT'deki "birey_id" alanına göre güncelleme yapmaktadır. Dönüt olarak üç adet anahtarı (hata, mesaj, birey_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi güncellenmiş anlamına gelmektedir. Hata anahtarının "1" olması kişinin güncellenemediği anlamına gelmektedir.

```
private function ekleKisiEslestirme( $birey_id, $ozluk,
$veri_kaynagi )
{
    $sorgu = $this->db->prepare ("INSERT INTO
tbl_bireyler_eslestirmeler (birey_id, birey_essiz_id,
birey_kimlik_no, birey_bilgiler, veri_kaynagi) VALUES (?,?,?,?,?)");

    return
        $sorgu->execute(array(
            $birey_id,
            $ozluk["birey_essiz_id"],
            $ozluk["birey_kimlik_no"],
            json_encode($ozluk),
            $veri_kaynagi
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birey_id"  => $birey_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birey_id"  => null
            ) ;
}
```

Şekil 4.32. Kişi Eşleştirmeyi Ekleme Fonksiyonu.

Şekil 4.32.'daki fonksiyon OVT'de daha önceden eşlememiş bir kişiyi eşleştirerek kayıt altına alan fonksiyondur. Fonksiyon üç tane parametre almaktadır:

- $birey_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_bireyler" tablosundaki eşsiz alan olan "birey_id" değeri içermektedir.
- $ozluk: Dizi tipinde değişkendir. Kişinin bütün özlük bilgilerini (ad, soyad, kimlik numarası gibi) içermektedir.

- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır.

Dönüt olarak üç adet anahtarı (hata, mesaj, birey_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi bilgilerinin güncellendiği anlamına, hata anahtarının "1" olması ise kişi bilgilerinin güncellenemediği anlamına gelmektedir.

```
private function guncelleKisiEslestirme( $birey_eslestirme_id,
$birey_id, $ozluk, $veri_kaynagi )
{
    $sorgu = $this->db->prepare("UPDATE tbl_bireyler_eslestirmeler
SET birey_id = ?, birey_essiz_id = ?, birey_kimlik_no = ?,
birey_bilgiler = ?, veri_kaynagi = ? WHERE birey_eslestirme_id =
?");

    return
        $sorgu->execute(array(
            $birey_id,
            $ozluk["birey_essiz_id"],
            $ozluk["birey_kimlik_no"],
            json_encode($ozluk),
            $veri_kaynagi,
            $birey_eslestirme_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birey_id"  => $birey_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birey_id"  => null
            ) ;
}
```

Şekil 4.33. Kişi Eşleştirmeyi Güncelleme Fonksiyonu.

Şekil 4.33.'da fonksiyon OVT'de daha önceden eşleşmiş bir kişinin eşleşme bilgilerini güncelleyen fonksiyondur. Fonksiyon dört tane parametre almaktadır:

- $birey_eslestirme_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_bireyler_eslestirmeler" tablosundaki eşsiz alan olan "birey_eslestirme_id" değeri içermektedir.
- $birey_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_bireyler" tablosundaki eşsiz alan olan "birey_id" değeri içermektedir.

- $ozluk: Dizi tipinde değişkendir. Kişinin bütün özlük bilgilerini (ad, soyad, kimlik numarası gibi) içermektedir.
- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birey_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi bilgilerinin güncellendiği anlamına, hata anahtarının "1" olması ise kişi bilgilerinin güncellenemediği anlamına gelmektedir.

```
private function birlestirKisiBilgiler($eski_birey_id,
$yeni_birey_id)
{
    $sorgu = $this->db->prepare
    (
        "SET @eski_birey_id = :eski_birey_id;
        SET @yeni_birey_id  = :yeni_birey_id;

        UPDATE tbl_bireyler_iletisim_bilgiler SET birey_id =
@yeni_birey_id WHERE birey_id = @eski_birey_id;
        UPDATE tbl_bireyler_unvanlar SET birey_id = @yeni_birey_id
WHERE birey_id = @eski_birey_id;
        UPDATE tbl_bireyler_eslestirmeler SET birey_id =
@yeni_birey_id WHERE birey_id = @eski_birey_id;
        DELETE FROM tbl_bireyler WHERE birey_id = @eski_birey_id;"
    );
    $b =
        $sorgu->execute(array(
            "yeni_birey_id" => $yeni_birey_id,
            "eski_birey_id" => $eski_birey_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birey_id"  => $yeni_birey_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birey_id"  => null
            ) ;
}
```

Şekil 4.34. Kişi Birleştiren Fonksiyonu.

Şekil 4.34.'deki fonksiyon OVT'de aynı kişinin iki farklı kaydı olması durumunda bu kayıtları teke indirmek için kullanılmaktadır. Fonksiyon iki parametre almaktadır. Bunlar; $eski_birey_id adındaki tam sayı tipindeki değişken ve

$yeni_birey_id adındaki tam sayı tipindeki değişkenlerdir. Fonksiyon OVT'de "birey_id" alanı bulunan tablolardaki eski birey_id bilgisini yeni birey_id bilgisi ile günceller ve son olarak "tbl_bireyler" tablosundaki eski birey_id'ye sahip kaydı silmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birey_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi bilgilerinin birleştirildiği anlamına, hata anahtarının "1" olması ise kişi birleştirilemediği anlamına gelmektedir.

```
public function kaydetKisiOzlukBilgiler($ozluk)
{
    $sorgu = $this->db->prepare("SELECT birey_kimlik_no, birey_id,
birey_eslestirme_id FROM tbl_bireyler_eslestirmeler WHERE
birey_essiz_id = :birey_essiz_id AND veri_kaynagi = :veri_kaynagi");
    $sorgu->execute(array(
            "birey_essiz_id" => $ozluk["birey_essiz_id"],
            "veri_kaynagi"   => $this->veri_kaynagi
        ));
    $vt = $sorgu->fetch(2);
    $sorgu = $this->db->prepare("SELECT birey_id FROM tbl_bireyler
WHERE birey_kimlik_no = :birey_kimlik_no");
    $sorgu->execute(array("birey_kimlik_no" =>
$ozluk["birey_kimlik_no"]));
    $ara_birey_id = $sorgu->fetch(2)["birey_id"];
    if(strlen($vt["birey_kimlik_no"])>0){
        $s = $vt["birey_kimlik_no"] != $ozluk["birey_kimlik_no"] ?
                strlen($ara_birey_id) > 0 ?
                    $this->birlestirKisiBilgiler($vt["birey_id"],
$ara_birey_id):
                    $this->guncelleKisiBilgiler($vt["birey_id"],
$ozluk):
                $this->guncelleKisiBilgiler($vt["birey_id"],
$ozluk);
        return
            $s["hata"] == "0" ?
                $this->guncelleKisiEslestirme
                (
                    $vt["birey_eslestirme_id"],
                    $vt["birey_id"],
                    $ozluk,
                    $this->veri_kaynagi
                ):
                $s;
    }
    else{
        if(strlen($ara_birey_id)>0) {
            $s = $this->guncelleKisiBilgiler($ara_birey_id, $ozluk);
            return
                $s["hata"] == "0" ?
                    $this->ekleKisiEslestirme($ara_birey_id, $ozluk,
$this->veri_kaynagi):
                    $s;
        }
        else{
            $s = $this->ekleKisiBilgiler($ozluk);
            return
                $s["hata"] == "0" ?
                    $this->ekleKisiEslestirme($s["birey_id"],
$ozluk, $this->veri_kaynagi):
                    $s;
        }
    }
}
```

Şekil 4.35. Kişi Bilgilerini Kaydeden Fonksiyon.

Şekil 4.35.'deki fonksiyon gerekli kontrolleri yaparak ve daha önceden oluşturulan 5 fonksiyonu (ekleKisiBilgiler, guncelleKisiBilgiler, ekleKisiEslestirme, guncelleKisiEslestirme, birlestirKisiBilgiler) kullanarak Şekil 4.29'daki algoritmayı yürütmektedir. Fonksiyon sadece bir parametre almaktadır. Bu parametre $ozluk adında ve dizi tipindeki değişkendir. Dizinin içerisinde kişiye ait bütün özlük bilgileri (ad, soyad, kimlik numarası, anne adı gibi) bulunmaktadır.

Dönüt olarak üç adet anahtarı (hata, mesaj, birey_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi özlük bilgilerinde gerekli değişikliklerin yapıldığı anlamına, hata anahtarının "1" olması ise kişi özlük bilgilerinde gerekli değişikliğin yapılamadığı anlamına gelmektedir. Kişi bilgilerinin kaydedilmesinde bir sorun ile karşılaşılmadıysa dönütte birey_id anahtarı yer alır.

```
public function kaydetBireyFoto($birey_essiz_id, $resim_base64)
{
    $sorgu = $this->db->prepare ("SELECT birey_id FROM
tbl_bireyler_eslestirmeler WHERE birey_essiz_id = ? AND veri_kaynagi
= ?");
    $sorgu->execute(array(
        $birey_essiz_id,
        $this->veri_kaynagi
    ));
    $birey_id = $sorgu->fetch(2)["birey_id"];
    if(strlen($birey_id)>0)
    {
        $resim_yol = "foto/".md5($birey_id. uniqid()).".jpg";
        $resim = file_put_contents($resim_yol,
base64_decode($resim_base64));
        if($resim !== false)
        {
            $sorgu = $this->db->prepare ("UPDATE tbl_bireyler SET
foto = ? WHERE birey_id = ?");
            $sorgu->execute(array(
                $resim_yol,
                $birey_id
            ));
        }
        else
            return
                array(
                    "hata" => "1",
                    "mesaj" => "Resim kopyalanamadı"
                );
    }
    else
        return
            array(
                "hata" => "1",
                "mesaj" => "Kişi bulunamadı"
            );
}
```

Şekil 4.36. Kişinin Fotoğrafını Güncelleme Fonksiyonu.

Daha önceden kişinin özlük bilgilerini kaydeden fonksiyon (Bkz. **Şekil 4.35**.) oluşturulmuştu ancak kişi fotoğrafı bu fonksiyonda istenmemişti. Bunun nedeni fotoğraf verileri çok büyük boyutlarda olabilir. Büyük boyutlu olduğu için fotoğraf kaydedilirken ve gönderilirken zaman kaybına neden olabilir ve web servislerin çalışmasını yavaşlatabilir. Aynı zamanda veri kaynakları (ÖBS ve Portal) kişinin özlük bilgilerindeki en ufak değişikliği OVT'ye göndereceklerdir ve bu gönderim sırasında OVT kişinin diğer bilgilerini de isteyecektir. Bunu örnekle açıklamak gerekirse; ÖBS'de "A" kişinin soyadı değiştiğinde, OVT'deki servis sadece soyadını değil kişinin bütün bilgilerini isteyecektir çünkü ÖBS'de bu kişinin daha önceden OVT'ye gönderilip

gönderilmediği hakkında bir kayıt tutulmamaktadır. Dolayısıyla kişinin sadece soyadı gönderilirse ve bu kişi OVT'de kayıtlı değilse OVT kişiyi veri tabanına eklemek için diğer özlük bilgilerine de ihtiyaç duyacaktır. Kişi özlük verilerinin gönderimi sırasında kişinin fotoğrafı da gönderilirse veri boyutu bir hayli artacaktır. Bu hem web servislerde yavaşlamaya hem de aynı resmin tekrar tekrar kaydedilmesine neden olacaktır. Bunu engellemek için kişinin fotoğrafının kaydedildiği servis özlük bilgileri servisi ile ayrıştırılmıştır.

ÖBS ve Portal bilgi sisteminde kişinin ayrı ayrı iki fotoğrafı tutulmaktadır. Ancak OVT'de kişinin sadece bir tane resmi bulunmaktadır. OVT'ye en son gelen fotoğraf kişinin güncel fotoğrafı olarak kabul edilmiştir.

Şekil 4.36.'deki fonksiyon kişi "jpg" uzantılı fotoğraf dosyasının içeriğini base64 ile encode edilmiş halini "foto" adındaki klasörüne kaydetmektedir. Base64 encode kullanılmasındaki amaç veri gönderimi sırasında özel karakterlerin oluşturacağı problemi engellemektir. Resim dosyasının adı ise, kişinin OVT'deki eşsiz "id"si ve zamana bağlı olarak php tarafından oluşturulan bir eşsiz "id" değeri toplanarak md5 algoritmasıyla şifrelenmiştir. Bunun nedeni gelen her fotoğraf için eşsiz bir ad üreterek eski fotoğrafların da aynı klasörde saklanabilmesidir.

Fonksiyonda kişinin diğer bilgi sistemlerindeki (ÖBS veya Portal) eşsiz "id" değeri istenerek kolay eşleştirme yapılması amaçlanmıştır. Fonksiyonda kişinin resmini kaydetmesi için, kişinin daha önceden OVT'ye aktarılmış olması gerekir. Kişi OVT'de yoksa kişinin fotoğrafı kaydedilmez.

Fonksiyon dönüt olarak iki anahtarı olan bir dizi dönmektedir. Dizide bulunan hata alanın "0" olması kişinin fotoğrafının güncellendiği anlamına gelmektedir, "1" olması ise işlemler sırasında bir hata olduğunu belirtir, bu hatanın açıklaması "mesaj" alanında verilmiştir.

4.4.1.2. Birim Bilgisi Denetimleri ve Fonksiyonları

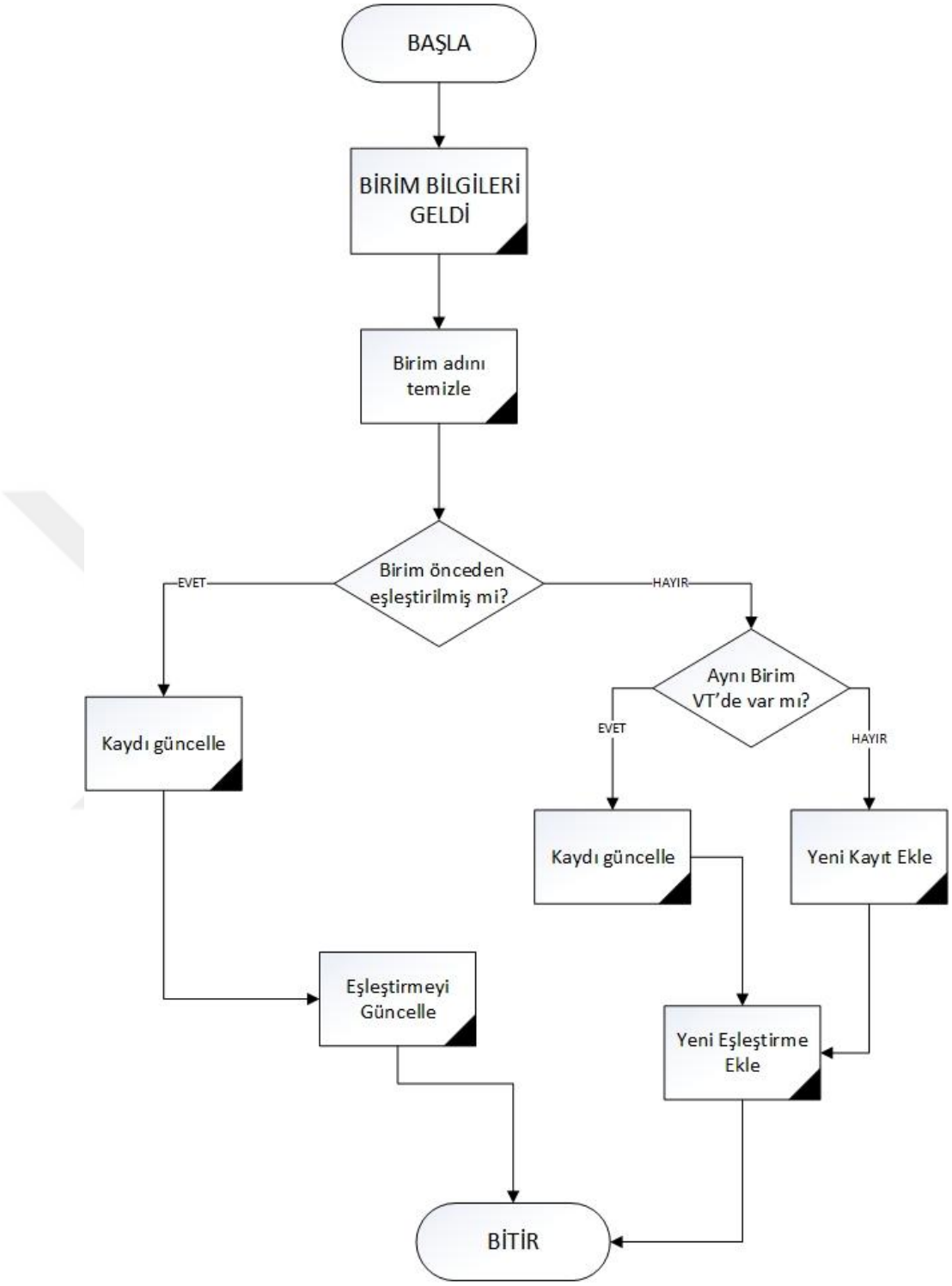
Üniversitede bulunan bilgi sistemleri içerisindeki hiyerarşileri kendi içlerinde tutarlı olmasına rağmen, farklı sistemlerde birim adlarında küçük değişiklikler olabilmektedir. Örnek olarak ÖBS'nin veri tabanında "Yönetim Bilişim Sistemleri Bölümü" adındaki bölüm Portal'ın veri tabanında "Yönetim Bilişim Sistemleri" olarak

anılabilmektedir veya yazım hatalarından kaynaklı sorunlardan dolayı farklı sistemlerdeki aynı birimler birbirlerinden ayrışabilmektedir.

Farklı sistemlerde farklı şekilde anılan ancak gerçekte aynı birim olan bu yapıları karşılaştırılmış ve bu birimler OVT'de teke düşürülmüştür. Bu teke düşürme işleminde birimin bütün hiyerarşik ve diğer bilgileri (birim tipi ve birim adı gibi) alınarak o birimin kendi seviyesindeki diğer birimlerle karşılaştırılması yapılmıştır. Karşılaştırmadan önce birimin sonundan bulunan "Bölümü", "Anabilim Dalı", "Programı" veya "Tezli Yüksek Lisans Programı" gibi aslında birimin adında bulunmayan kelimeler silinmiştir. Bu kelimelerin silinmesindeki amaç karşılaştırmanın daha sağlıklı yapılabilmesidir. Birimlerin karşılaştırılmasında birebir bir eşitlik yerine Levenshtein algoritması kullanılmıştır. Bunun amacı birim adlarındaki küçük yazım farklılıklarından kaynaklı sorunların üstesinden gelebilmektir.

Bir birimde yapılacak eşleme için birimin bütün hiyerarşisine ihtiyaç vardır çünkü program adı aynı olan fakat başka fakülte veya meslek yüksekokulunun altında yer alan programlar olabilir. Örneğin "Bilgisayar Programcılığı" programı hem Erzurum Meslek Yüksekokulunun altında hem de Pasinler Meslek Yüksekokulunun altında yer almaktadır. Bunlara benzer birimleri ayırt edebilmek için o birimin bütün üst birimlerinin bilgileriyle beraber karşılaştırılması gerekir.

Akış şemasının (Bkz. **Şekil 4.37**.) uygulanması için oluşturulacak bütün fonksiyonlar "private" olarak ayarlanmıştır. Yani OVT'nin kendi iç işlerini yapmak için kullanılacaktır. Diğer bilgi sistemleri tarafından çağırılmayacaktır.



Şekil 4.37. Birim Eşleştirme Akış Şeması.

Şekil 4.37.'deki akış şemasını koda dökmeden önce birim bilgilerini güncelleyen, yeni birim oluşturan, eşleşme bilgilerini güncelleyen, eşleşme bilgilerini kaydeden ve birim adını temizleyen fonksiyonların oluşturulması gerekmektedir.

```
private function ekleBirimBilgiler( $birim_ad, $birim_ust_id,
$birim_tur_id, $statu_id )
{
    $sorgu = $this->db->prepare ("INSERT INTO tbl_birimler
(birim_ad, birim_ust_id, birim_tur_id, statu_id) VALUES (?,?,?,?)");

    return
        $sorgu->execute(array(
            $birim_ad,
            $birim_ust_id,
            $birim_tur_id,
            $statu_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birim_id"  => $this->db->lastInsertId()
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birim_id"  => null
            ) ;
}
```

Şekil 4.38. Birim Ekleme Fonksiyonu.

Şekil 4.38.'deki fonksiyon OVT'ye yeni birim ekleme fonksiyonudur. Fonksiyon gelen birim bilgilerine hiçbir müdahalede bulunmadan OVT'ye eklemektedir. Fonksiyon dört tane parametre almaktadır:

- $birim_ad: Metin tipinde değişkendir. Birimin adını içermektedir.
- $birim_ust_id: Tam sayı tipinde değişkendir. Birimin bir üst biriminin OVT'deki "tbl_birimler" tablosundaki "birim_id" değerini içermektedir.
- $birim_tur_id: Tam sayı tipinde değişkendir. Birimin program, bölüm, fakülte gibi birim türünü içeren bilgidir. Bu bilginin karşılıkları "tur_birimler" tablosundaki "birim_tur_id" alanında yer almaktadır.
- $statu_id: Tam sayı tipinde değişkendir. Birimin aktif veya pasif gibi birim durumunu içeren bilgidir. Bu bilginin karşılıkları "tbl_statuler" tablosundaki "statü_id" alanında yer almaktadır.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması birimin eklenmiş olduğu anlamına gelmektedir. Eğer birim eklemede sorun çıkmamış ise dönütte "birim_id" anahtarı son eklenen birimin id değerini vermektedir. Hata anahtarının "1" olması birim eklenirken sorun çıktığı anlamına gelmektedir.

```
private function guncelleBirimBilgiler( $birim_id, $birim_ad,
$statu_id )
{
    $sorgu = $this->db->prepare("UPDATE tbl_birimler SET birim_ad =
?, statu_id = ? WHERE birim_id = ?");

    return
        $sorgu->execute(array(
            $birim_ad,
            $statu_id,
            $birim_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti".
json_encode($sorgu->errorInfo()),
                "birim_id"  => $birim_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birim_id"  => null
            ) ;
}
```

Şekil 4.39. Birim Güncelleme Fonksiyonu.

Şekil 4.39.'daki fonksiyon OVT'de bulunan birimin adını ve statüsünü güncellemektedir. Fonksiyon gelen birim adı bilgisine hiçbir müdahalede bulunmadan OVT'deki "birim_id" alanına göre güncelleme yapmaktadır.

Fonksiyon üç tane parametre almaktadır:

- $birim_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_birimler" tablosundaki eşsiz alan olan "birim_id" değeri beklenmektedir.
- $birim_ad: Metin tipinde değişkendir. Birimin adını içermektedir.
- $statu_id: Tam sayı tipinde değişkendir. Birimin aktif veya pasif gibi birim durumunu içeren bilgidir. Bu bilginin karşılıkları "tbl_statuler" tablosundaki "statü_id" alanında yer almaktadır.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması birim adının güncellenmiş olduğu anlamına, hata anahtarı "1" olması ise birim adının güncellenemediği anlamına gelmektedir. Bu fonksiyon birimin sadece adını güncellemekte, birimin üst id değerini veya birim türünü güncellememektedir çünkü birimin hiyerarşisinin ve türünün değişme ihtimali pek bulunmamaktadır. Ancak yanlış veri gelmesi durumunda birim türünün veya hiyerarşisinin değişmesine ihtiyaç duyulabilir. Bu durumda ise birim bilgileri (birim_tur_id ve birim_ust_id) manuel olarak güncellenebilir.

```
private function ekleBirimEslestirme( $birim_id, $birim_essiz_id,
$birim_ad_eslesen, $veri_kaynagi )
{
    $sorgu = $this->db->prepare ("INSERT INTO
tbl_birimler_eslestirmeler (birim_id, birim_essiz_id,
birim_ad_eslestirilen, veri_kaynagi) VALUES (?,?,?,?)");

    return
        $sorgu->execute(array(
            $birim_id,
            $birim_essiz_id,
            $birim_ad_eslesen,
            $veri_kaynagi
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birim_id"  => $birim_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu,
                "birim_id"  => null
            ) ;
}
```

Şekil 4.40. Birim Eşleştirmeyi Ekleme Fonksiyonu.

Şekil 4.40.'deki fonksiyon OVT'de daha önceden eşlememiş bir birimi kayıt altına alan fonksiyondur. Fonksiyon dört tane parametre almaktadır:

- $birim_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_birimler" tablosundaki eşsiz alan olan "birim_id" değeri içermektedir.
- $birim_essiz_id: Tam sayı tipinde değişkendir. OVT'ye veri gönderen bilgi sistemleri (ÖBS veya Portal) tarafından birim için kullanılan eşsiz birim id değeri içermektedir.

- $birim_eslestirilen_ad: Metin tipinde değişkendir. OVT'de birim eşleştirmesi yapılırken kullanılan birim adını içermektedir.
- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması birim eşleşme bilgilerinin eklendiği anlamına, hata anahtarının "1" olması ise kişi bilgilerinin güncellenemediği anlamına gelmektedir.

```
private function guncelleBirimEslestirme( $birim_id,
$birim_eslestirme_id, $birim_ad_eslesen )
{
    $sorgu = $this->db->prepare("UPDATE tbl_birimler_eslestirmeler
SET birim_ad_eslestirilen = ? WHERE birim_eslestirme_id = ?");

    return
        $sorgu->execute(array(
            $birim_ad_eslesen,
            $birim_eslestirme_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "birim_id"  => $birim_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "birim_id"  => null
            ) ;
}
```

Şekil 4.41. Birim Eşleştirmeyi Güncelleme Fonksiyonu.

Şekil 4.41.'deki fonksiyon OVT'de daha önceden eşleşmiş bir birimin eşleşme bilgilerini güncelleyen fonksiyondur. Fonksiyon üç tane parametre almaktadır:

- $birim_id: Tam sayı tipinde değişkendir. Bu değişken kullanılarak herhangi işlem yapılmamaktadır. Bu değişkenin amacı $birim_id değişkenini geri döndürebilmesidir.
- $birim_eslestirme_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_birimler_eslestirmeler" tablosundaki eşsiz alan olan "birim_eslestirme_id" değeri içermektedir.

- $birim_ad_eslesen: Metin tipinde değişkendir. Eşleştirilen birimin adı içermektedir.

Birim eşleştirmesini güncelleyen fonksiyonda sadece eşleşen birimin adının güncellenme nedeni; birim bilgileri güncellenirken kişi bilgilerinde olduğu gibi tekilleştirme işlemine tabi tutulmamaktadır. Birim bilgilerinde tekilleştirme işlemi olmadığı için "birim_id" gibi bilgilerin sonradan değişmemektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması birim eşleşme bilgilerinin güncellendiği anlamına, hata anahtarının "1" olması ise kişi bilgilerinin güncellenemediği anlamına gelmektedir.

```
public function temizleBirimAd($birim_ad)
{
    return
        str_replace
        (
            array
            (
                "Bilim Dalı",
                "Anabilim Dalı",
                "Programı",
                "Bölümü",
                "Tezli Yüksek Lisans",
                "Tezli Yükseklisans",
                "Tezsiz Yüksek Lisans",
                "Doktora",
                " - ",
            ),
            "",
            $birim_ad
        );
}
```

Şekil 4.42. Birim Adını Temizleyen Fonksiyon.

Şekil 4.42.'daki fonksiyon OVT tarafından veri kaynağı olarak kullanılan diğer bilgi sistemlerinin (ÖBS ve Portal) kendi ihtiyaçlarına yönelik kullandığı birim adlarındaki fazlalıkları silen fonksiyondur. Birim adlarında bulunan bu fazlalıklar (bilim dalı, programı, bölümü gibi) birim eşleştirmelerinde daha iyi sonuçlar alabilmek için silinmiştir. Fonksiyon metin tipinde $birim_ad adında tek bir parametre almaktadır ve birim adında bulunan fazlalıklar silindikten sonraki değeri geri döndürür.

```
private function kaydetBirimBilgiler($birim_essiz_id, $birim_ad,
$birim_ust_id, $birim_tur_id, $statu_id, $veri_kaynagi)
{
    $birim_ad = $this->temizleBirimAd($birim_ad);

    $sorgu = $this->db->prepare("SELECT birim_id,
birim_eslestirme_id FROM tbl_birimler_eslestirmeler WHERE
birim_essiz_id = :birim_essiz_id AND veri_kaynagi = :veri_kaynagi");
    $sorgu->execute(array(
            "birim_essiz_id" => $birim_essiz_id,
            "veri_kaynagi"   => $veri_kaynagi
        ));
    $vt = $sorgu->fetch(2);

    if(strlen($vt["birim_id"])>0) {
        $s = $this->guncelleBirimBilgiler($vt["birim_id"],
$birim_ad, $statu_id);
        return
            $s["hata"] == "0" ?
                $this->guncelleBirimEslestirme($vt["birim_id"],
$vt["birim_eslestirme_id"], $birim_ad):
                $s;
    }
    else
    {
        $sorgu = $this->db->prepare("SELECT birim_id FROM
tbl_birimler WHERE birim_ust_id = :birim_ust_id AND birim_tur_id =
:birim_tur_id AND levenshtein(birim_ad, :birim_ad )<5");
        $sorgu->execute(array(
                "birim_ust_id" => $birim_ust_id,
                "birim_tur_id" => $birim_tur_id,
                "birim_ad" => $birim_ad
            ));
        $ara_birim_id = $sorgu->fetch(2)["birim_id"];

        if(strlen($ara_birim_id)>0) {
            $s = $this->guncelleBirimBilgiler($ara_birim_id,
$birim_ad, $statu_id);
            return
                $s["hata"] == "0" ?
                    $this->ekleBirimEslestirme($ara_birim_id,
$birim_essiz_id, $birim_ad, $veri_kaynagi):
                    $s;
        }
        else{
            $s = $this->ekleBirimBilgiler($birim_ad, $birim_ust_id,
$birim_tur_id, $statu_id);
            return
                $s["hata"] == "0" ?
                    $this->ekleBirimEslestirme($s["birim_id"],
$birim_essiz_id, $birim_ad, $veri_kaynagi):
                    $s;
        }
    }
}
```

Şekil 4.43. Birim Bilgilerini Kaydeden Fonksiyon.

Şekil 4.43'deki fonksiyon gerekli kontrolleri yaparak ve daha önceden oluşturulan beş fonksiyonu (ekleBirimBilgiler, guncelleBirimBilgiler, ekleBirimEslestirme, guncelleBirimEslestirme, temizleBirimAd) kullanarak **Şekil 4.37**'deki algoritmayı yürütmektedir. Fonksiyon beş parametre almaktadır:

- $birim_essiz_id: Tam sayı tipinde değişkendir. Diğer bilgi sistemlerinin (ÖBS veya Portal) kendi içerisinde birimler için kullandıkları eşsiz "id" değerini içermektedir.
- $birim_ad: Metin tipinde değişkendir. Birimin adını içermektedir.
- $birim_ust_id: Tam sayı tipinde değişkendir. Biriminde bir kademe üstündeki bulunan eşsiz "id" değerini içermektedir.
- $birim_tur_id: Tam sayı tipinde değişkendir. Birimin program, bölüm, fakülte gibi birim türünü içeren bilgidir. Bu bilginin karşılıkları "tur_birimler" tablosundaki "birim_tur_id" alanında yer almaktadır.
- $statu_id: Tam sayı tipinde değişkendir. Birimin aktif veya pasif gibi birim durumunu içeren bilgidir. Bu bilginin karşılıkları "tbl_statuler" tablosundaki "statü_id" alanında yer almaktadır.
- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması birim bilgilerinde gerekli değişikliklerin yapıldığı anlamına, hata anahtarının "1" olması ise birim bilgilerinde gerekli değişikliğin yapılamadığı anlamına gelmektedir.

Şekil 4.43.'daki algoritma sadece bir birimin OVT'ye eklenme veya güncellenme işlemini yapmaktadır. OVT'de bulunan birimler birbirleriyle ilişkilidir. Bu ilişkiyi kurabilmek için birimin bütün hiyerarşisine ihtiyaç duyulmaktadır. **Şekil 4.43**'da bulunan "kaydetBirimBilgiler" fonksiyonunun aldığı parametrelerden biri olan "$birim_ust_id" değeri diğer bilgi sistemlerinde bulunmamaktadır. Bu değer OVT'de "tbl_birimler" tablosundaki "birim_ust_id" değeridir. Örnek olarak OVT'ye "Yönetim Bilişim Sistemleri Bilim Dalı" eklenmek istendiğinde bu birimin bütün üst birim bilgileri de aşağıdaki **Şekil 4.44**'deki gibi gönderilmelidir.

```
Array
(
    [0] => Array
        (
            [birim_essiz_id] => 156
            [birim_ad] => "İktisadi ve İdari Bilimler Fakültesi"
            [birim_tur_id] => 2
            [statu_id] => 6
        )

    [1] => Array
        (
            [birim_essiz_id] => 587
            [birim_ad] => "Yönetim Bilişim Sistemleri Bölümü"
            [birim_tur_id] => 8
            [statu_id] => 6
        )

    [2] => Array
        (
            [birim_essiz_id] => 598
            [birim_ad] => "Yönetim Bilişim Sistemleri Programı"
            [birim_tur_id] => 10
            [statu_id] => 6
        )
)
```

Şekil 4.44. Yönetim Bilişim Sistemleri Programının Hiyerarşik Yapısı.

Şekil 4.44.'de "Yönetim Bilişim Sistemleri Programı" birimini OVT'ye eklemek için gerekli bütün veriler bulunmaktadır. Birim bilgileri OVT'ye aktarılırken en üst birimden başlanarak en alta doğru ilerlemelidir. Üst birimin eşsiz "id" değeri bilinmeden alt birimle olan ilişki oluşturulamaz. Birim bilgilerindeki sıralama da önemlidir. Sıralamanın yanlış olması veya değiştirilmesi OVT'deki hiyerarşiyi de etkileyecek ve yanlış bir hiyerarşi oluşmasına neden olacaktır. OVT'de en gelen birimin en üst birimi varsayılan olarak "Atatürk Üniversitesi" olarak kabul edilecektir. Bundan dolayı diğer bilgi sistemleri birim bilgilerinde Atatürk Üniversitesi'ni göndermeyeceklerdir.

Birim hiyerarşisi her zaman üç kademeli (fakülte, bölüm, program) olmak zorunda değildir. Birimin yapısına bağlı olarak bazen bir kademe veya iki kademli de olabilmektedir. Aşağıdaki şekilde Personel Daire Başkanlığına bağlı olan Kadro Şubesinin hiyerarşik yapısı gösterilmektedir.

```
Array
(
    [0] => Array
        (
            [birim_essiz_id] => 75
            [birim_ad] => "Personel Daire Başkanlığı"
            [birim_tur_id] => 7
            [statu_id] => 6
        )

    [1] => Array
        (
            [birim_essiz_id] => 89
            [birim_ad] => "Kadro Şubesi"
            [birim_tur_id] => 13
            [statu_id] => 6
        )
)
```

Şekil 4.45. Kadro Şubesinin Hiyerarşik Yapısı.

Birim bilgilerindeki hiyerarşik kademe sayısı değişkenlik gösterdiği için OVT'de birim kaydetme işlemlerinde dinamik yapı ihtiyacı doğmuştur. Bu dinamik yapı OVT'de bir fonksiyon ile gerçekleştirilmiştir. Bu fonksiyon daha önceden oluşturulan "kaydetBirimBilgiler" fonksiyonunu kullanarak birimin bütün hiyerarşisini kayıt altına almaktadır. Bu işlemi yaparken de gelen verideki ilk birimin üst birimi Atatürk Üniversitesi ("birim_id" değeri 1) kabul edecektir. İkinci birimin üst birimini de birinci birimin "birim_id" değerini kabul edecektir. Bu işlem ta ki gelen verideki bütün birimler bitene kadar devam edecektir. Birim ekleme işlemleri sırasında oluşacak bir hatada işlemi durduracak ve geri dönüş yapacaktır çünkü fonksiyon bir birim eklenemediğinde sonraki birimin "ust_birim_id" değerine erişemeyecektir. Bu da sonraki birim ekleme işlemini engelleyecektir. Bu dinamik yapının kullanıldığı fonksiyon aşağıdaki **Şekil 4.46**'de gösterilmiştir.

```
private function kaydetBirimHiyerarsi($b, $veri_kaynagi)
{

    $birim_ust_id = 1;
    foreach ($b as $k => $v)
    {
        $s =
            $this
                ->kaydetBirimBilgiler
                (
                    $v["birim_essiz_id"],
                    $v["birim_ad"],
                    $birim_ust_id,
                    $v["birim_tur_id"],
                    $v["statu_id"],
                    $veri_kaynagi
                );
        if($s["hata"] == "0"){
            $birim_ust_id = $s["birim_id"];
        }
        else
            return $s;
    }
    return $s;
}
```

Şekil 4.46. Birimin Bütün Hiyerarşisini Kaydeden Fonksiyon.

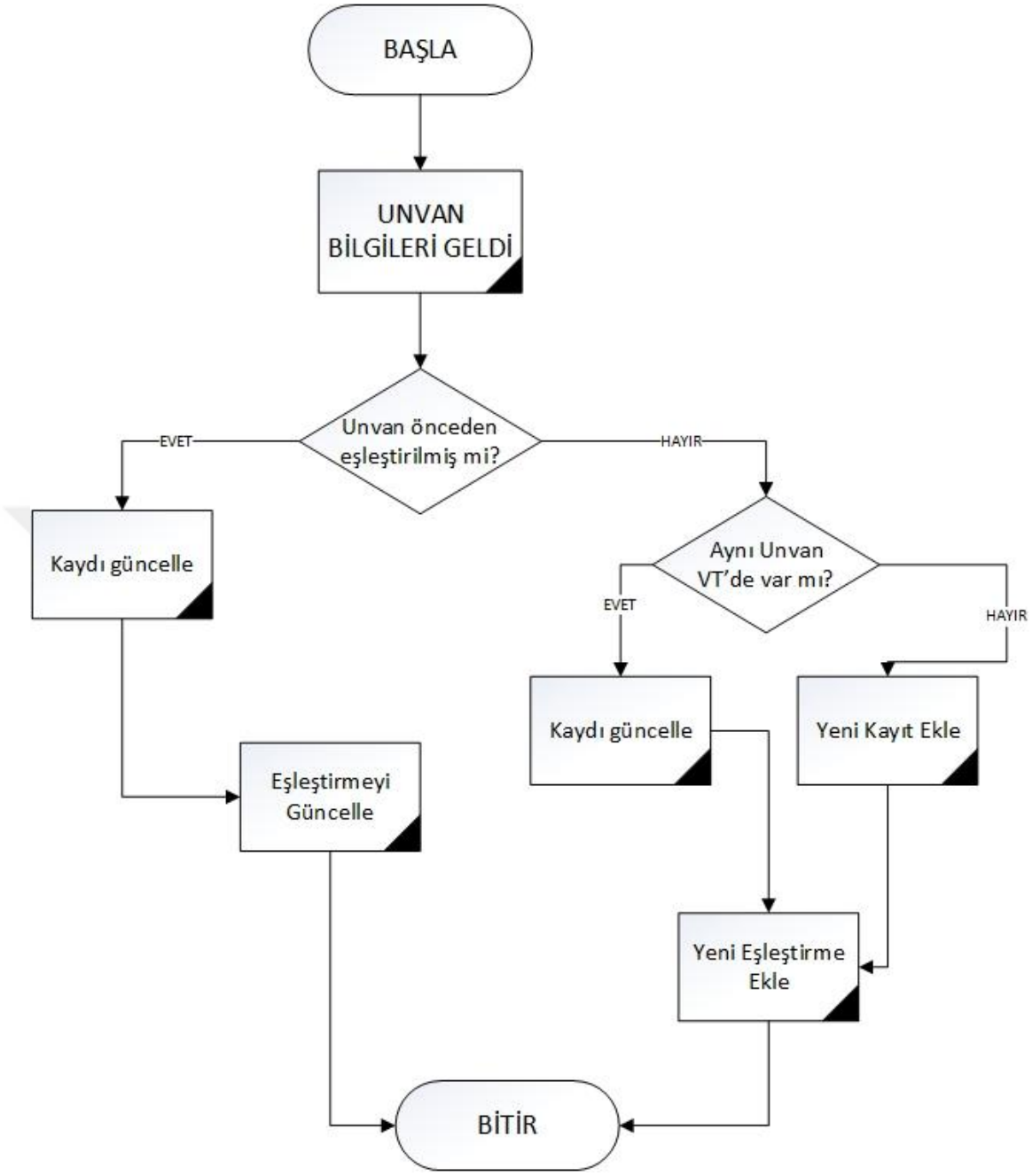
Şekil 4.46.'deki fonksiyon bir birimin bütün hiyerarşisini kaydetmektedir. Fonksiyon iki tane parametre almaktadır. Bunlar; $b adındaki dizi tipi değişken ve $veri_kaynagi adındaki metin tipi değişkenlerdir. Adı $b olan değişken bir birimin bütün hiyerarşik yapısı, adı ve türü gibi bilgileri içermektedir (Şekil 25 ve Şekil 26'da örnek veriler bulunmaktadır), adı $veri_kaynagi olan değişken gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Birim hiyerarşi kayıtlarında herhangi bir hata olması durumunda döngüyü durdurarak hata anahtarı "1" olan bir diziyi döndürür. Kayıtlar sırasında herhangi bir hata olmaması durumunda ise hata anahtarı "0" olan bir dizi döndürür. Yine hata olmaması durumunda en son eklenen birimin OVT'deki eşsiz "id" değeri de döndürülen dizinin içerisinde bulunur.

4.4.1.3. Unvan Bilgisi Denetimleri ve Fonksiyonları

Üniversitede çalışan personelin çok çeşitli olmasından dolayı unvan sayısı oldukça fazladır. Aynı unvan bilgileri farklı bilgi yönetim sistemleri tarafından çoğu zaman aynı isimle anılsa da bazen yazım hatalarından veya unvanın çok uzun olması durumunda kısaltmalardan kaynaklı olarak birkaç harfte değişiklik olabilir. Farklı sistemlerde farklı şekilde anılan ancak gerçekte aynı unvanlar karşılaştırılmış ve OVT'de teke düşürülmüştür. Unvanların karşılaştırılmasında birebir bir eşitlik yerine Levenshtein algoritması kullanılmıştır. Bunun amacı unvan adlarındaki küçük yazım farklılıklarından kaynaklı sorunların üstesinden gelebilmektir.

Akış şemasının (Bkz. **Şekil 4.47**.) uygulanması için oluşturulacak bütün fonksiyonlar "private" olarak ayarlanmıştır. Yani OVT'nin kendi iç işlerini yapmak için kullanılacaktır. Diğer bilgi sistemleri tarafından çağırılmayacaktır.



Şekil 4.47. Unvan Eşleştirme Akış Şeması.

Şekil 4.47.'deki akış şemasını koda dökmeden önce unvan bilgilerini güncelleyen, yeni unvan oluşturan, eşleşme bilgilerini güncelleyen ve eşleşme bilgi fonksiyonlarının oluşturulması gerekmektedir.

```
private function ekleUnvanBilgiler( $unvan_ad, $unvan_kisa_ad,
$unvan_tur_id )
{
    $sorgu = $this->db->prepare ("INSERT INTO tbl_unvanlar
(unvan_ad, unvan_kisa_ad, unvan_tur_id) VALUES (?,?,?)");

    return
        $sorgu->execute(array(
            $unvan_ad,
            $unvan_kisa_ad,
            $unvan_tur_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "unvan_id"  => $this->db->lastInsertId()
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "unvan_id"  => null
            ) ;
}
```

Şekil 4.48. Unvan Ekleme Fonksiyonu.

Şekil 4.48.'deki fonksiyon OVT'ye yeni unvan ekleme fonksiyonudur. Fonksiyon gelen unvan bilgilerine hiçbir müdahalede bulunmadan OVT'ye eklemektedir. Fonksiyon üç tane parametre almaktadır:

- $unvan_ad: Metin tipinde değişkendir. Unvanın adını içermektedir.
- $unvan_kisa_ad: Metin tipinde değişkendir. Unvanın kısaltılmış halini içermektedir.
- $unvan_tur_id: Tam sayı tipinde değişkendir. Unvanın akademik, idari veya öğrenci gibi unvan türünü içeren bilgidir. Bu bilginin karşılıkları "tur_unvanlar" tablosundaki "unvan_tur_id" alanında yer almaktadır.

Dönüt olarak üç adet anahtarı (hata, mesaj, birim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması unvan eklenmiş anlamına gelmektedir. Eğer unvan eklemede sorun çıkmamış ise dönütte "unvan_id" anahtarı son eklenen unvanın id değerini vermektedir. Hata anahtarının "1" olması unvan eklenirken sorun çıktığı anlamına gelmektedir.

```
private function guncelleUnvanBilgiler( $unvan_id, $unvan_ad,
$unvan_kisa_ad )
{
    $sorgu = $this->db->prepare("UPDATE tbl_unvanlar SET unvan_ad =
?, unvan_kisa_ad = ? WHERE unvan_id = ?");

    return
        $sorgu->execute(array(
            $unvan_ad,
            $unvan_kisa_ad,
            $unvan_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "unvan_id"  => $unvan_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "unvan_id"  => null
            ) ;
}
```

Şekil 4.49. Unvan Güncelleme Fonksiyonu.

Şekil 4.49.'daki fonksiyon OVT'de bulunan unvanın adını ve kısa adını güncellemektedir. Fonksiyon üç tane parametre almaktadır:

- $unvan_id: Tam sayı tipinde değişkendir. Unvanın OVT'deki eşsiz "id" değerini içermektedir.
- $unvan_ad: Metin tipinde değişkendir. Unvanın adını içermektedir.
- $unvan_kisa_ad: Metin tipinde değişkendir. Unvanın kısaltılmış halini içermektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, unvan_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması unvan bilgilerinin güncellenmiş olduğu, hata anahtarı "1" olması ise unvan bilgilerinin güncellenemediği anlamına gelmektedir. Bu fonksiyon unvanın sadece adını güncellemekte, unvanın türünü (akademik, idari veya öğrenci gibi) güncellememektedir çünkü unvan türünün değişme ihtimali pek bulunmamaktadır. Ancak yanlış veri gelmesi durumunda unvan türünün değişmesine ihtiyaç duyulabilir. Bu durumda ise unvan türü manuel olarak güncellenebilir.

```
private function ekleUnvanEslestirme( $unvan_id, $unvan_essiz_id,
$unvan_ad_eslesen, $unvan_kisa_ad_eslesen, $veri_kaynagi )
{
    $sorgu = $this->db->prepare ("INSERT INTO
tbl_unvanlar_eslestirmeler (unvan_id, unvan_essiz_id,
unvan_ad_eslestirilen, unvan_kisa_ad_eslestirilen, veri_kaynagi)
VALUES (?,?,?,?,?)");
    return
        $sorgu->execute(array(
            $unvan_id,
            $unvan_essiz_id,
            $unvan_ad_eslesen,
            $unvan_kisa_ad_eslesen,
            $veri_kaynagi
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "unvan_id"  => $unvan_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu". json_encode($sorgu-
>errorInfo()),
                "unvan_id"  => null
            ) ;
}
```

Şekil 4.50. Unvan Eşleştirmeyi Ekleme Fonksiyonu.

Şekil 4.50.'deki fonksiyon OVT'de daha önceden eşlenmemiş bir unvanı kayıt altına alan fonksiyondur. Fonksiyon beş tane parametre almaktadır:

- $unvan_id: Tam sayı tipinde değişkendir. Unvanın OVT'deki eşsiz "id" değerini içermektedir.
- $birim_essiz_id: Tam sayı tipinde değişkendir. OVT'ye veri gönderen bilgi sistemleri (ÖBS veya Portal) tarafından unvan için kullanılan eşssiz unvan id değeri içermektedir.
- $unvan_ad_eslesen: Metin tipinde değişkendir. OVT'de birim eşleştirmesi yapılırken kullanılan birim adını içermektedir.
- $unvan_kisa_ad_eslesen: Metin tipinde değişkendir. Unvanın kısaltılmış halini içermektedir.
- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, unvan_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması unvan eşleştirme bilgilerinin güncellendiği anlamına, hata anahtarının "1" olması ise güncellenemediği anlamına gelmektedir.

```
private function guncelleUnvanEslestirme( $unvan_id,
$unvan_eslestirme_id, $unvan_ad_eslesen, $unvan_kisa_ad_eslesen )
{
    $sorgu = $this->db->prepare("UPDATE tbl_unvanlar_eslestirmeler SET
unvan_ad_eslestirilen = ?, unvan_kisa_ad_eslestirilen = ? WHERE
unvan_eslestirme_id = ?");

    return
        $sorgu->execute(array(
            $unvan_ad_eslesen,
            $unvan_kisa_ad_eslesen,
            $unvan_eslestirme_id
        )) ?
            array(
                "hata"      => "0",
                "mesaj"     => "İşlem gerçekleşti",
                "unvan_id"  => $unvan_id
            ) :
            array(
                "hata"      => "1",
                "mesaj"     => "Hata oluştu",
                "unvan_id"  => null
            ) ;
}
```

Şekil 4.51. Unvan Eşleştirmeyi Güncelleme Fonksiyonu.

Şekil 4.51.'deki fonksiyon OVT'de daha önceden eşleşmiş bir unvanın eşleşme bilgilerini güncelleyen fonksiyondur. Fonksiyon dört tane parametre almaktadır:

- $unvan_id: Tam sayı tipinde değişkendir. Bu değişken kullanılarak herhangi işlem yapılmamaktadır. Bu değişkenin amacı $unvan_id değişkenini geri döndürebilmesidir.
- $unvan_eslestirme_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_unvanlar_eslestirmeler" tablosundaki eşsiz alan olan "unvan_eslestirme_id" değeri içermektedir.
- $unvan_ad_eslesen: Metin tipinde değişkendir. Eşleştirilen unvanın adı içermektedir.
- $unvan_kisa_ad_eslesen: Metin tipinde değişkendir. Eşleştirilen unvanın kısaltılmış adı içermektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, unvan_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi bilgilerinin güncellendiği anlamına, hata anahtarının

"1" olması ise unvan eşleştirme bilgilerinin güncellenemediği anlamına gelmektedir.

```
private function kaydetUnvanBilgiler($unvan_essiz_id, $unvan_ad,
$unvan_kisa_ad, $unvan_tur_id, $veri_kaynagi)
{
    $sorgu = $this->db->prepare("SELECT unvan_id, unvan_eslestirme_id
FROM tbl_unvanlar_eslestirmeler WHERE unvan_essiz_id = :unvan_essiz_id
AND veri_kaynagi = :veri_kaynagi");
    $sorgu->execute(array(
            "unvan_essiz_id" => $unvan_essiz_id,
            "veri_kaynagi"   => $veri_kaynagi
        ));
    $vt = $sorgu->fetch(2);

    if(strlen($vt["unvan_id"])>0)
    {
        $s = $this->guncelleUnvanBilgiler($vt["unvan_id"], $unvan_ad,
$unvan_kisa_ad);
        return
            $s["hata"] == "0" ?
                $this->guncelleUnvanEslestirme($vt["unvan_id"],
$vt["unvan_eslestirme_id"], $unvan_ad, $unvan_kisa_ad):
                $s;
    }
    else
    {
        $sorgu = $this->db->prepare("SELECT unvan_id FROM tbl_unvanlar
WHERE unvan_tur_id = :unvan_tur_id AND levenshtein(unvan_ad, :unvan_ad
)<5");
        $sorgu->execute(array(
                "unvan_tur_id" => $unvan_tur_id,
                "unvan_ad" => $unvan_ad
            ));
        $ara_unvan_id = $sorgu->fetch(2)["unvan_id"];

        if(strlen($ara_unvan_id)>0)
        {
            $s = $this->guncelleUnvanBilgiler($ara_unvan_id, $unvan_ad,
$unvan_kisa_ad);
            return
                $s["hata"] == "0" ?
                    $this->ekleUnvanEslestirme($ara_unvan_id,
$unvan_essiz_id, $unvan_ad, $unvan_kisa_ad, $veri_kaynagi):
                    $s;
        }
        else
        {
            $s = $this->ekleUnvanBilgiler($unvan_ad, $unvan_kisa_ad,
$unvan_tur_id);
            return
                $s["hata"] == "0" ?
                    $this->ekleUnvanEslestirme($s["unvan_id"],
$unvan_essiz_id, $unvan_ad, $unvan_kisa_ad, $veri_kaynagi):
                    $s;
        }
    }
}
```

Şekil 4.52. Birim Bilgilerini Kaydeden Fonksiyon.

Şekil 4.52.'daki fonksiyon gerekli kontrolleri yaparak ve daha önceden oluşturulan dört fonksiyonu (ekleUnvanBilgiler, guncelleUnvanBilgiler, ekleUnvanEslestirme ve guncelleUnvanEslestirme) kullanarak **Şekil 4.47**'deki algoritmayı yürütmektedir. Fonksiyon beş parametre almaktadır:

- $unvan_essiz_id: Tam sayı tipinde değişkendir. Diğer bilgi sistemlerinin (ÖBS veya Portal) kendi içerisinde unvan için kullandıkları eşsiz "id" değerinin içermektedir.
- $unvan_ad: Metin tipinde değişkendir. Unvanın adını içermektedir.
- $unvan_kisa_ad: Metin tipinde değişkendir. Unvanın kısaltılmış adını içermektedir.
- $unvan_tur_id: Tam sayı tipinde değişkendir. Unvanın akademik, idari veya öğrenci gibi unvan türünü içeren bilgidir. Bu bilginin karşılıkları "tur_unvanlar" tablosundaki "unvan_tur_id" alanında yer almaktadır.
- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, unvan_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması unvan bilgilerinde gerekli değişikliklerin yapıldığı anlamına, hata anahtarının "1" olması ise unvan bilgilerinde gerekli değişikliğin yapılamadığı anlamına gelmektedir.

4.4.1.4. Kişi Unvan Denetimleri ve Fonksiyonları

Kişiler OVT'ye gönderilirken bu kişilerin unvan (öğretim görevlisi, bilgisayar işletmeni veya lisans öğrencisi gibi) bilgilerinin de gönderilmesi gerekir. Daha önceden OVT ile diğer bilgi sistemlerinin birim ve unvan bilgilerindeki eşleşmeleri anlatılmıştı. Bu kısımda da kişilerin unvan bilgilerinin (sadece kişinin unvanı değil aynı zamanda unvanın kullanıldığı birim bilgisi) OVT'ye aktarımı anlatılacaktır.

Kişi unvan bilgileri aktarılmadan önce kişi bilgilerinin OVT'ye aktarılması zorunludur. Kişi bilgileri OVT'de yok ise kişinin unvan bilgileri kabul edilmeyecektir. Kişi unvan bilgileri aktarılmadan önce kişinin daha önceden eşleştirilmiş bir kaydı olup olmadığına bakılacak eğer eşleştirilmiş kaydı yoksa kişi unvan bilgisi eklenecek, eşleştirilmiş kaydı var ise var olan kişi unvan bilgisi güncellenecektir.

```
private function ekleBireyUnvanEslestirme($birey_unvan_id,
$birey_unvan_essiz_id, $bu_bilgiler, $veri_kaynagi)
{
    $sorgu = $this->db->prepare ("INSERT INTO
tbl_bireyler_unvanlar_eslestirmeler (birey_unvan_id,
birey_unvan_essiz_id, birey_unvan_bilgiler, veri_kaynagi) VALUES
(?,?,?,?)");
    return
        $sorgu->execute(array(
            $birey_unvan_id,
            $birey_unvan_essiz_id,
            json_encode($bu_bilgiler),
            $veri_kaynagi
        )) ?
            array(
                "hata"       => "0",
                "mesaj"      => "İşlem gerçekleşti",
                "birey_unvan_id" => $birey_unvan_id
            ) :
            array(
                "hata"       => "1",
                "mesaj"      => "Hata oluştu"
                "birey_unvan_id" => null
            ) ;
}
```

Şekil 4.53. Kişi Unvan Eşleştirme Bilgilerini Ekleyen Fonksiyonu.

Şekil 4.53.'deki fonksiyon OVT'de daha önceden eşlememiş bir kişi unvan bilgisini kayıt altına alan fonksiyondur. Fonksiyon dört tane parametre almaktadır:

- $birey_unvan_id: Tam sayı tipinde değişkendir. OVT'deki "tbl_bireyler_unvanlar" tablosundaki eşsiz alan olan "birey_unvan_id" değerini içermektedir.
- $birey_unvan_essiz_id: Tam sayı tipinde değişkendir. OVT'ye veri gönderen bilgi sistemleri (ÖBS veya Portal) tarafından kişi unvanı için kullanılan eşsiz "id" değerini içermektedir.
- $bu_bilgiler: Dizi tipinde değişkendir. Diğer bilgi sistemlerinden gelen kişi unvan bilgilerini içerir.
- $veri_kaynagi: Metin tipinde değişkendir. Gelen verilerin kaynak ismini barındırmaktadır. ÖBS veya Portal değerlerinden biri olması gerekmektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, birey_unvan_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişi unvan eşleştirme bilgilerinin eklendiği

anlamına, hata anahtarının “1” olması ise kişi bilgilerinin güncellenemediği anlamına gelmektedir.

```
private function guncelleBireyUnvanEslestirme($birey_unvan_id,
$birey_unvan_eslestirme_id, $bu_bilgiler)
{
    $sorgu = $this->db->prepare ("UPDATE
tbl_bireyler_unvanlar_eslestirmeler SET birey_unvan_bilgiler = ?
WHERE birey_unvan_eslestirme_id = ?");
    return
        $sorgu->execute(array(
        json_encode($bu_bilgiler),
            $birey_unvan_eslestirme_id
        )) ?
            array(
                "hata"       => "0",
                "mesaj"      => "İşlem gerçekleşti",
                "birey_unvan_id" => $birey_unvan_id
            ) :
            array(
                "hata"       => "1",
                "mesaj"      => "Hata oluştu",
                "birey_unvan_id" => null
            ) ;
}
```

Şekil 4.54. Kişi Unvan Eşleştirme Bilgilerini Güncelleyen Fonksiyon.

Şekil 4.54.’deki fonksiyon daha önceden eşleştirilen bir kişi unvan bilgisini güncellemektedir. Fonksiyon üç parametre almaktadır. Bunlar; $birey_unvan_id kişi unvanının OVT’deki eşsiz “id” değeri, $birey_unvan_eslestirme_id kişi unvanının eşleştirilme eşsiz değeri ve $bu_bilgiler diğer bilgi sistemlerinden (ÖBS veya Porta) gelen kişi unvan bilgilerinin tamamıdır.

Kişi unvan bilgilerinin aktarımı için bir tane “public” yani diğer bilgi sistemleri tarafından çağrılabilen fonksiyon oluşturulmuştur. Ancak bu fonksiyon içerisinde daha önceden oluşturulan fonksiyonlardan “kaydetUnvanBilgiler”, “kaydetBirimBilgiler”, “ekleBireyUnvanEslestirme” ve “guncelleBireyUnvanEslestirme” fonksiyonları da kullanılmıştır.

```
public function kaydetBireyUnvanBilgiler($birey_essiz_id,
$birey_unvan_essiz_id, $u, $b, $sicil_ogrenci_no, $statu_id)
{
    $sorgu = $this->db->prepare ("SELECT birey_id FROM
tbl_bireyler_eslestirmeler WHERE birey_essiz_id = ? AND veri_kaynagi
= ?");
    $sorgu->execute(array(
        $birey_essiz_id,
        $this->veri_kaynagi
    ));
    $birey_id = $sorgu->fetch(2)["birey_id"];
    if(strlen($birey_id)==0)
        return
            array(
                "hata" => "1",
                "mesaj" => "Kişi daha önceden eşleştirilmemiş.
Lütfen önce kişi bilgilerini eşleştiriniz"
            );

    $unvan =
        $this->kaydetUnvanBilgiler
        (
            $u["unvan_essiz_id"],
            $u["unvan_ad"],
            $u["unvan_kisa_ad"],
            $u["unvan_tur_id"],
            $this->veri_kaynagi
        );
    if($unvan["hata"] != "0")
        return $unvan;

    $birim = $this->kaydetBirimHiyerarsi($b, $this->veri_kaynagi);
    if($birim["hata"] != "0")
        return $birim;


    $sorgu = $this->db->prepare ("SELECT birey_unvan_id,
birey_unvan_eslestirme_id FROM tbl_bireyler_unvanlar_eslestirmeler
WHERE birey_unvan_essiz_id = ? AND veri_kaynagi = ?");
    $sorgu->execute(array(
        $birey_unvan_essiz_id,
        $this->veri_kaynagi
    ));
    $bi = $sorgu->fetch(2);
    $birey_unvan_id = $bi["birey_unvan_id"];
    $birey_unvan_eslestirme_id = $bi["birey_unvan_eslestirme_id"];
    if(strlen($birey_unvan_id)>0)
    {
        $sorgu = $this->db->prepare ("UPDATE tbl_bireyler_unvanlar
SET unvan_id = ?, birim_id = ?, sicil_ogrenci_no = ?, statu_id = ?
WHERE birey_unvan_id = ?");
        $s =
            $sorgu->execute(array(
                $unvan["unvan_id"],
                $birim["birim_id"],
```

```
                $sicil_ogrenci_no,
                $statu_id,
                $birey_unvan_id
            )) ?
                array(
                    "hata"  => "0",
                    "mesaj" => "Kişi unvan bilgileri kaydedildi.",
                    "birey_unvan_id" => $birey_unvan_id
                ):
                array(
                    "hata"  => "1",
                    "mesaj" => "Kişi unvan bilgileri eklenemedi.",
                    "birey_unvan_id" => null
                );
        return
            $s["hata"] == "0" ?
                $this->guncelleBireyUnvanEslestirme($birey_unvan_id,
$birey_unvan_eslestirme_id, array("birim" => $b,"unvan" => $u)):
                $s;
    }
    else
    {
        $sorgu = $this->db->prepare ("INSERT INTO
tbl_bireyler_unvanlar (birey_id, unvan_id, birim_id,
sicil_ogrenci_no, statu_id) VALUES (?,?,?,?,?)");
        $s =
            $sorgu->execute(array(
                $birey_id,
                $unvan["unvan_id"],
                $birim["birim_id"],
                $sicil_ogrenci_no,
                $statu_id
            )) ?
                array(
                    "hata"  => "0",
                    "mesaj" => "Kişi unvan bilgileri kaydedildi.",
                    "birey_unvan_id" => $this->db->lastInsertId()
                ):
                array(
                    "hata"  => "1",
                    "mesaj" => "Kişi unvan bilgileri
kaydedilemedi.".json_encode($sorgu->errorInfo()),
                    "birey_unvan_id" => null
                );
        return
            $s["hata"] == "0" ?
                $this-
>ekleBireyUnvanEslestirme($s["birey_unvan_id"],
$birey_unvan_essiz_id, array("birim" => $b,"unvan" => $u), $this-
>veri_kaynagi):
                $s;
    }
}
```

Şekil 4.55. Kişi Unvan Bilgilerini Kaydeden Fonksiyon.

Şekil 4.55.'deki fonksiyon diğer bilgi sistemlerinden (ÖBS veya Portal) aldığı kişinin unvanına ait bilgileri gerekli kontrollerden geçirerek OVT'ye kaydetmektedir. Fonksiyon yedi parametre almaktadır:

- $birey_essiz_id: Tam sayı tipinde değişkendir. Diğer bilgi sistemlerinin (ÖBS veya Portal) kendi içerisinde birey için kullandıkları eşsiz "id" değerini içermektedir.
- $birey_unvan_essiz_id: Tam sayı tipinde değişkendir. Diğer bilgi sistemlerinin (ÖBS veya Portal) kendi içerisinde bireyin unvanı için kullandıkları eşsiz "id" değerini içermektedir.
- $u: Dizi tipinde değişkendir. İçerisinde unvan bilgileri beklemektedir. Bu dizinin içerisinde "unvan_essiz_id", "unvan_ad", "unvan_kisa_ad" ve "unvan_tur_id" anahtarları bulunmaktadır.
- $b: Dizi tipinde değişkendir. Bu değişken iki boyutlu bir dizidir ve içerisinde birimin bütün hiyerarşik bilgileri bulunur (Bkz. **Şekil 4.44** ve **Şekil 4.45**.). Bu dizinin içerisinde "birim_essiz_id", "birim_ad", "birim_tur_id" ve "statu_id" anahtarları bulunmaktadır.
- $sicil_ogrenci_no: Metin tipinde değişkendir. Verisi gelen kişi öğrenci ise öğrenci numarası değilse sicil numarasını içermektedir.
- $statu_id: Tam sayı tipindeki değişkendir. Kişinin unvanının statüsünü (devamlı öğrenci, aktif, pasif veya emekli gibi) içermektedir.

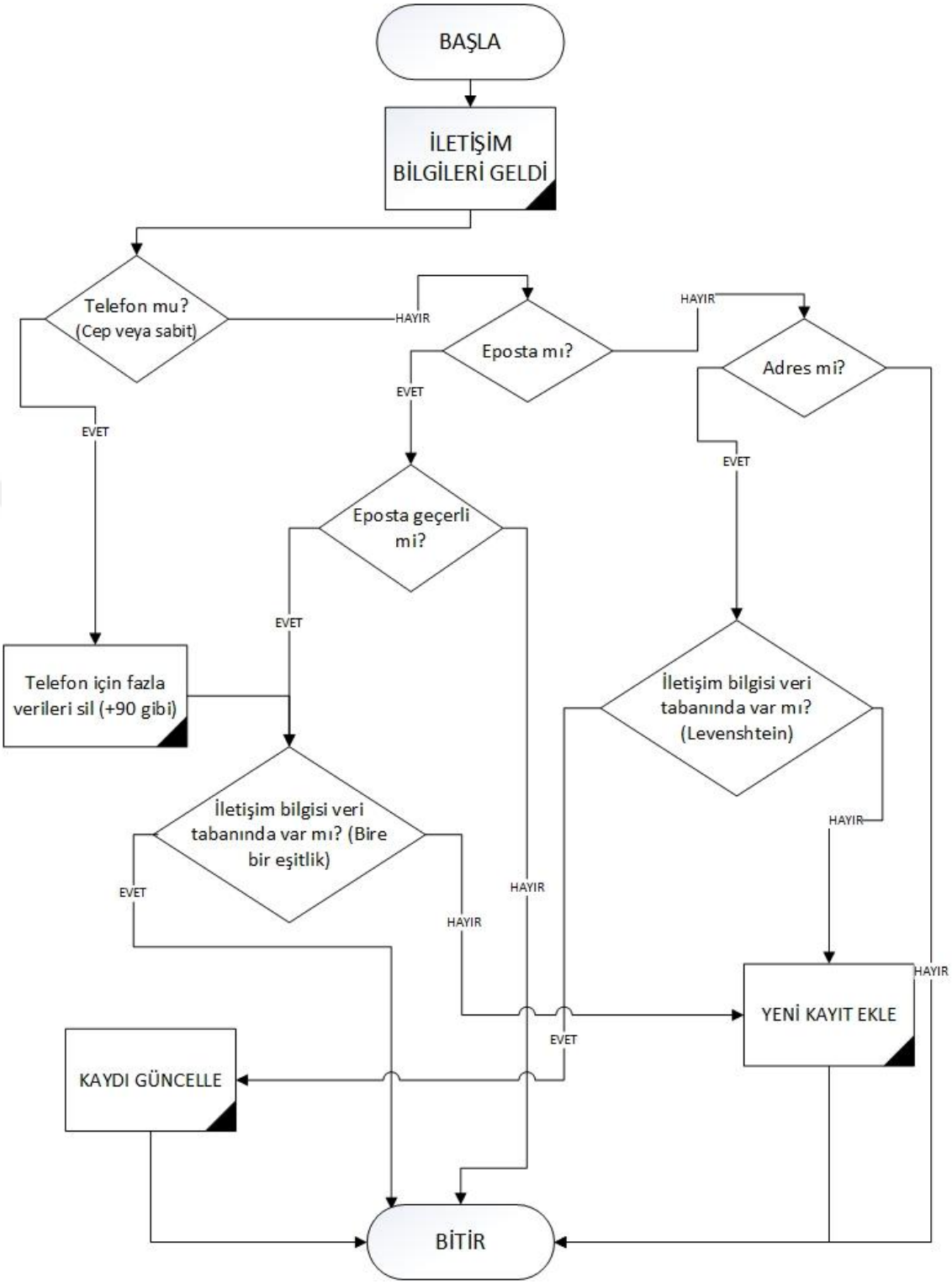
Dönüt olarak iki adet anahtarı (hata, mesaj) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişinin unvan bilgilerinde gerekli değişikliklerin yapıldığı anlamına, hata anahtarının "1" olması ise unvan bilgilerinde gerekli değişikliğin yapılamadığı anlamına gelmektedir.

4.4.1.5. İletişim Bilgisi Denetimleri ve Fonksiyonları

OVT'de kişilere ait iletişim bilgileri dört farklı tür olarak tutulmaktadır. Bunlar; adres, cep telefonu, sabit telefon ve e-postadır. Diğer bilgi sistemlerinden (ÖBS veya Portal) gelen iletişim bilgileri aynı kişi için farklı bilgiler veya aynı kişi için aynı bilgiler olabilir. Bundan dolayı OVT'ye gelen iletişim bilgisi veri tabanında aratılır eğer iletişim bilgisi veri tabanında yoksa yeni kayıt eklenir. Tabi, bu iletişim bilgisinin veri

tabanında olup olmadığı kontrolü iletişim verisinin türüne göre yapılır. Yani gelen veri cep telefonu ise veri tabanında cep telefonları arasında arama yapılır. Her iletişim tipi kendine göre farklı kontrollerden geçmektedir. Örneğin e-posta adresi belirli bir düzene (pattern) göre oluşturulduğu için önce e-postanın geçerli olup olmadığına daha sonra da veri tabanında bu kaydın olup olmadığı kontrol edilir.

Aşağıdaki akış şemasının (Bkz. **Şekil 4.56**.) uygulanması için oluşturulacak fonksiyonlar "private" ve "public" olmak üzere ikiye ayrılmıştır. Private olan fonksiyonlar OVT'nin kendi iç işlerini yapmak için kullanılan fonksiyonlarıdır. Bir tane olan public fonksiyon yani "kaydetBireyIletisimBilgilerHepsi" diğer bilgi sistemlerinin çalıştırılması için oluşturulmuştur.



Şekil 4.56. İletişim Bilgileri Kayıt Akış Şeması.

Yukarıdaki akış şeması (Bkz. **Şekil 4.56**.) sadece bir iletişim bilgisinin kayıt adımlarını göstermektedir. Bu şemayı koda dökmeden önce iletişim bilgilerini güncelleyen, yeni iletişim bilgisini ekleyen, e-posta adresini doğrulayan ve telefon bilgisinde bulunan fazlalıkları silen fonksiyonların oluşturulması gerekmektedir.

```
private function ekleBireyIletisimBilgiler($birey_id,
$iletisim_tur_id, $iletisim_icerik)
{
    $sorgu = $this->db->prepare ("INSERT INTO
tbl_bireyler_iletisim_bilgiler (birey_id, iletisim_tur_id,
iletisim_icerik) VALUES (?,?,?)");

    return
        $sorgu->execute(array(
            $birey_id,
            $iletisim_tur_id,
            $iletisim_icerik
        )) ?
            array(
                "hata"          => "0",
                "mesaj"         => "İşlem gerçekleşti",
                "iletisim_id"   => $this->db->lastInsertId()
            ) :
            array(
                "hata"          => "1",
                "mesaj"         => "Hata oluştu",
                "iletisim_id"   => null
            ) ;
}
```

Şekil 4.57. Birey İletişim Bilgisini Kaydeden Fonksiyon.

Şekil 4.57.'deki fonksiyon OVT'ye bireyin yeni iletişim bilgisini ekleyen fonksiyondur. Fonksiyon gelen iletişim bilgilerine hiçbir müdahalede bulunmadan OVT'ye eklemektedir. Fonksiyon üç tane parametre almaktadır:

- $birey_id: Tam sayı tipinde değişkendir. İletişim bilgisi eklenecek kişinin OVT'de bulunan eşsiz "id" değerini içermektedir.
- $iletisim_tur_id: Tam sayı tipinde değişkendir. İletişim bilgisinin türünü (cep telefonu, eposta veya adres gibi) içermektedir.
- $iletisim_icerik: Metin tipinde değişkendir. İletişim bilgisini içermektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, iletisim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması iletişim bilgisinin eklenmiş olduğu anlamına gelmektedir. Eğer iletişim bilgisi eklenebilmişse dönütte "iletisim_id" anahtarı son eklenen iletişim

id değerini vermektedir. Hata anahtarının "1" olması iletişim bilgisi eklenirken sorun çıktığı anlamına gelmektedir.

```
private function guncelleBireyIletisimBilgiler( $iletisim_id,
$iletisim_icerik )
{
    $sorgu = $this->db->prepare("UPDATE
tbl_bireyler_iletisim_bilgiler SET iletisim_icerik = ? WHERE
iletisim_id = ?");

    return
        $sorgu->execute(array(
            $iletisim_icerik,
            $iletisim_id
        )) ?
            array(
                "hata"          => "0",
                "mesaj"         => "İşlem gerçekleşti",
                "iletisim_id"   => $iletisim_id
            ) :
            array(
                "hata"          => "1",
                "mesaj"         => "Hata oluştu",
                "iletisim_id"   => null
            ) ;
}
```

Şekil 4.58. Birey İletişim Bilgisini Güncelleyen Fonksiyon.

Şekil 4.58.'deki fonksiyon bireyin OVT'de kayıtlı olan iletişim bilgisini güncellemek için kullanılmaktadır. Fonksiyon sadece iletişim bilgisinin kendisini güncellemektedir iletişim türünü (adres veya eposta gibi) değiştirmemektedir. Fonksiyon iki tane parametre almaktadır:

- $iletisim _id: Tam sayı tipinde değişkendir. Güncellenmek istenen kaydın "tbl_bireyler_iletisimler" tablosundaki eşsiz iletişim id değerini beklemektedir.
- $iletisim_icerik: Metin tipinde değişkendir. İletişim bilgisinin içermektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, iletisim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması iletişim bilgisinin güncellenmiş olduğu anlamına gelmektedir. Eğer iletişim bilgisi güncellenebilmişse dönütte "iletisim_id" anahtarı fonksiyona gelen iletişim id değerini vermektedir. Hata anahtarının "1" olması iletişim bilgisi eklenirken sorun çıktığı anlamına gelmektedir.

```
private function kontrolEPosta($eposta)
{
    return
        filter_var($eposta, FILTER_VALIDATE_EMAIL) ?
            array(
                    "hata"      => "0",
                    "mesaj"     => "E-Posta geçerli"
                ) :
                array(
                    "hata"      => "1",
                    "mesaj"     => "E-Posta geçerli değil"
                ) ;
}
```

Şekil 4.59. E-Postanın Geçerliliğini Kontrol Eden Fonksiyon.

Şekil 4.59.'deki fonksiyon gelen e-posta bilgisinin geçerli olup olmadığını kontrol etmektedir. E-postanın geçerli olması durumunda "true" geçersiz olması durumunda ise "false" yanıtı dönmektedir. E-Posta kontrolü yaparken PHP'nin kendi fonksiyonu olan "filter_var" kullanılmaktadır.

```
private function temizleTelefon($telefon)
{
    $telefon = filter_var($telefon, FILTER_SANITIZE_NUMBER_INT);
    return
        str_replace
        (
            "+90",
            "",
            $telefon
        )*1;
}
```

Şekil 4.60. Telefonda Bilgisini Temizleyen Fonksiyon.

Şekil 4.60.'daki fonksiyon gelen telefon bilgisindeki fazlalıkları temizleyerek geri döndürmektedir. Fonksiyon öncelikle "filter_var" fonksiyonu ile telefon verisinin içerisinde bulunan "+", "-", "." ve sayısal değerler haricindeki bütün veriyi siler daha sonra numaranın içerisinde "+90" ifadesi silinir son olarak çıkan sonuç 1 ile çarpılarak numaranın başında "0" varsa bu veri de silinir.

```
private function araIletisimBilgisiEsitlik($birey_id,
$iletisim_tur_id, $iletisim_icerik)
{
    $sorgu = $this->db->prepare ("SELECT iletisim_id FROM
tbl_bireyler_iletisim_bilgiler WHERE birey_id = ? AND
iletisim_tur_id = ? AND iletisim_icerik = ?");
    $sorgu->execute(array(
        $birey_id,
        $iletisim_tur_id,
        $iletisim_icerik
    ));
    return
        $sorgu->fetch(2)["iletisim_id"];
}
```

Şekil 4.61. İletişim Bilgisini Birebir Arayan Fonksiyon.

Şekil 4.61.'deki fonksiyon gelen iletişim bilgisini veri tabanında birebir arayarak, aynı iletişim bilgisinden bulması durumunda OVT'de bulunan iletişim bilgisinin eşsiz "id" değerini döndürmektedir. İletişim bilgisini bulamaması durumunda ise boyutu sıfır olan bir boşluk değeri dönmektedir.

```
private function araIletisimBilgisiLevenshtein($birey_id,
$iletisim_tur_id, $iletisim_icerik)
{
    $sorgu = $this->db->prepare ("SELECT iletisim_id FROM
tbl_bireyler_iletisim_bilgiler WHERE birey_id = ? AND
iletisim_tur_id = ? AND levenshtein(iletisim_icerik, ?)<5");
    $sorgu->execute(array(
        $birey_id,
        $iletisim_tur_id,
        $iletisim_icerik
    ));
    return
        $sorgu->fetch(2)["iletisim_id"];
}
```

Şekil 4.62. İletişim Bilgisini Levenshtein Algoritmasına Göre Arayan Fonksiyon.

Şekil 4.62.'deki fonksiyon gelen iletişim bilgisini veri tabanından levenshtein algoritmasına göre arayarak, benzer iletişim bilgisinden bulması durumunda OVT'de bulunan iletişim bilgisinin eşsiz "id" değerini döndürmektedir.

```
private function kaydetBireyIletisimBilgiler($birey_id,
$iletisim_tur_id, $iletisim_icerik)
{
    switch ($iletisim_tur_id)
    {
        case "1"://Adres
            $iletisim_id = $this-
>araIletisimBilgisiLevenshtein($birey_id, $iletisim_tur_id,
$iletisim_icerik);
            return
                strlen($iletisim_id)>0 ?
                    $this-
>guncelleBireyIletisimBilgiler($iletisim_id, $iletisim_icerik):
                    $this->ekleBireyIletisimBilgiler($birey_id,
$iletisim_tur_id, $iletisim_icerik);
        break;
        case "2"://Cep Telefonu
        case "3"://Sabit Telefon
            $iletisim_icerik = $this-
>temizleTelefon($iletisim_icerik);
            $iletisim_id = $this-
>araIletisimBilgisiEsitlik($birey_id, $iletisim_tur_id,
$iletisim_icerik);
            return
                strlen($iletisim_id)>0 ?
                    array( "hata"  => "0", "mesaj" => "İletişim
bilgisi zaten kayıtlı", "iletisim_id" => $iletisim_id ):
                    $this->ekleBireyIletisimBilgiler($birey_id,
$iletisim_tur_id, $iletisim_icerik);
        break;
        case "4"://E Posta
            if($this->kontrolEPosta($iletisim_icerik))
            {
                $iletisim_id = $this-
>araIletisimBilgisiEsitlik($birey_id, $iletisim_tur_id,
$iletisim_icerik);
                return
                    strlen($iletisim_id)>0 ?
                        array( "hata"  => "0", "mesaj" => "İletişim
bilgisi zaten kayıtlı", "iletisim_id" => $iletisim_id ):
                        $this->ekleBireyIletisimBilgiler($birey_id,
$iletisim_tur_id, $iletisim_icerik);
            }
            else return array( "hata" => "1", "mesaj" => "E-posta
adresi geçerli değil", "iletisim_id" => null );
        break;

        default:
            return array( "hata"  => "1", "mesaj" => "İletişim türü
geçerli değilssss1".$birey_id."1aaaa", "iletisim_id" => null );
        break;
    }
}
```

Şekil 4.63.'daki fonksiyon gerekli kontrolleri yaparak ve daha önce oluşturulan altı fonksiyonları (ekleBireyIletisimBilgiler, guncelleBireyIletisimBilgiler, kontrolEPosta, temizleTelefon, araIletisimBilgisiEsitlik ve araIletisimBilgisiLevenshtein) kullanarak **Şekil 4.56**'deki algoritmayı yürütmektedir. Fonksiyon üç parametre almaktadır:

- $birey_id: Tam sayı tipinde değişkendir. İletişim bilgisi eklenecek kişinin OVT'de bulunan eşsiz "id" değerini beklemektedir.
- $iletisim_tur_id: Tam sayı tipinde değişkendir. İletişim bilgisinin tipini (adres, telefon veya eposta gibi) içermektedir.
- $iletisim_icerik: Metin tipinde değişkendir. İletişim bilgisinin kendisini içermektedir.

Dönüt olarak üç adet anahtarı (hata, mesaj, iletisim_id) olan bir dizi dönmektedir. Hata anahtarının "0" olması kişinin iletişim bilgilerinde gerekli değişikliklerin yapıldığı anlamına, hata anahtarının "1" olması ise kişi iletişim bilgilerinde gerekli değişikliğin yapılamadığı anlamına gelmektedir.

Bir kişinin birden fazla adresi veya cep telefonu OVT'de bulunabilir. Ancak kişi ile iletişime geçilmek istendiğinde bu bazı sorunlara neden olabilir. Örneğin OVT kişinin kimlik kartı basıldığında kişinin cep telefonuna kısa mesaj atmaktadır. Kişinin birden fazla telefonu OVT'de bulunabildiği için kişiye kayıtlı bütün telefonlara kısa mesaj atmak gerekmektedir. Cep telefonuna gönderilen mesajların ücretli olduğu düşünüldüğünde hem masraflıdır hem de kişi cep telefonu numarasını değiştirmiş olabileceği için kısa mesaj yanlış kişilere gönderilebilir.

OVT'nin bilgi kaynakları (ÖBS ve Portal) kişilerin iletişim bilgilerini doğrulayarak kayıt altına almaktadır. Yani kişi sisteme (ÖBS veya Portal) cep telefonunu eklediğinde kişiye kısa mesaj yoluyla bir kod gönderilir kişi o kodu sisteme girince kişinin sistemdeki cep telefonu güncellenir. Bu işlem sonucunda kişilerin iletişim bilgilerinin doğruluğundan emin olunur. Ancak kişilerin zaman içinde iletişim bilgileri değişebilmektedir. Kişiler bilgi sistemlerine girdikleri iletişim bilgilerinin güncelliğini ihtiyaç olmadıkça pek umursamamaktadır veya sadece bir tane bilgi sisteminde (ÖBS veya Portal) değiştirmektedirler. Böyle bir durumda OVT'de kişinin hem eski ve hem de yeni telefon numaraları bulunmaktadır. OVT'de aynı türdeki iki

iletişim bilgisinin aktif olmasını engellemek için kişinin son gelen iletişim bilgisi haricindeki kayıtlar "statu_id" alanı "9" yapılarak pasife çekilirler.

```
private function pasifYapBireyDigerIletisimBilgiler($birey_id,
$iletisim_id, $iletisim_tur_id)
{
    $sorgu = $this->db->prepare ("UPDATE
tbl_bireyler_iletisim_bilgiler SET statu_id = 9 WHERE birey_id = ?
AND iletisim_tur_id = ? AND iletisim_id != ?");

    return
        $sorgu->execute(array(
            $birey_id,
            $iletisim_tur_id,
            $iletisim_id
        )) ?
            array(
                "hata"          => "0",
                "mesaj"         => "İşlem gerçekleşti"
            ) :
            array(
                "hata"          => "1",
                "mesaj"         => "Hata oluştu"
            ) ;
}
```

Şekil 4.64. İletişim Bilgilerini Pasif Yapan Fonksiyon.

Şekil 4.64.'daki fonksiyon gelen iletişim id dışında kalan, kişinin iletişim bilgilerini pasif yapmak için kullanılmaktadır. Bu pasif yapma işleminde iletişim türü göz önünde bulundurulmuştur. Fonksiyon üç parametre almaktadır:

- $iletisim _id: Tam sayı tipinde değişkendir. Pasif yapılmayacak kaydın "tbl_bireyler_iletisimler" tablosundaki eşsiz iletişim id değerini beklemektedir.
- $birey_id: Tam sayı tipinde değişkendir. İletişim bilgisi pasif edilecek kişinin OVT'de bulunan eşsiz "id" değerini beklemektedir.
- $iletisim_tur_id: Tam sayı tipinde değişkendir. İletişim bilgisinin tipini (adres, telefon veya e-posta gibi) içermektedir.

Dönüt olarak iki adet anahtarı (hata, mesaj) olan bir dizi dönmektedir. Hata anahtarının "0" olması fonksiyona gelen eşsiz "id" dışındaki iletişim bilgilerinin pasif yapıldığı anlamına, hata anahtarının "1" olması ise bu işlemler sırasında bir hata oluştuğu anlamına gelmektedir.

```
Array
(
    [0] => Array
        (
            [iletisim_tur_id] => 3
            [iletisim_icerik] => "+90 442 231 00 00"
        )

    [1] => Array
        (
            [iletisim_tur_id] => 4
            [iletisim_icerik] => "gokhan@atauni.edu.tr"
        )

    [2] => Array
        (
            [iletisim_tur_id] => 1
            [iletisim_icerik] => "Atatürk Üniversitesi / Erzurum"
        )

)
```

Şekil 4.65. İletişim Bilgileri Örneği.

İletişim bilgilerinin kaydedildiği fonksiyonda sadece bir tane iletişim bilgisi eklenmekte veya güncellenmektedir. Ancak diğer bilgi sistemlerinden (ÖBS ve Portal) **Şekil 4.65**'daki gibi kişinin bütün iletişim bilgileri aynı anda istenmektedir.

Bu iletişim bilgilerinin hepsinin kaydedildiği fonksiyon aşağıdaki şekilde verilmiştir. Aşağıdaki fonksiyon kişinin bütün iletişim bilgileri kayıt altına almaktadır. Fonksiyon iki tane parametre almaktadır. Bunlar; tam sayı tipindeki $birey_essiz_id kişinin bilgi sistemindeki (ÖBS veya Portal) eşsiz "id" değerini beklemektedir ve dizi tipindeki (Bkz. **Şekil 4.65.**) $i değişkeni ise kişinin iletişim bilgilerini içermektedir. Fonksiyonun içerisinde kullanılan metin tipindeki ve $this->veri_kaynagi adındaki değişken ise public bir değişken olup web servis kullanıcısı doğrulandığında atanmaktadır. Bu değişken verinin hangi kaynaktan geldiği (ÖBS veya PORTAL değerlerinden birini alabilir) bilgisini içermektedir. Gelen iletişim bilgileri tek tek döngü kullanılarak kaydedilir. Döngünün altında iletişim bilgisinin kaydı esnasında bir sorunla karşılaşılmadıysa aynı türdeki diğer iletişim bilgileri pasif yapılmaktadır. Döngüdeki kayıt esnasında hata ile karşılaşılsa bile döngü sonraki adımına devam etmektedir çünkü bir iletişim bilgisi diğer iletişim bilgisini etkilememektedir. Dönüt olarak iki boyutlu bir dizi döner. Bu dizinin içerisinde her bir iletişim bilgisinin kaydının yapılıp yapılamadığı bilgisi vardır.

```
public function kaydetBireyIletisimBilgilerHepsi($birey_essiz_id,
$i)
{
    $sorgu = $this->db->prepare ("SELECT birey_id FROM
tbl_bireyler_eslestirmeler WHERE birey_essiz_id = ? AND veri_kaynagi
= ?");
    $sorgu->execute(array(
        $birey_essiz_id,
        $this->veri_kaynagi
    ));
    $birey_id = $sorgu->fetch(2)["birey_id"];
    foreach ($i as $key => $value)
    {

        $s[] = $s2 =
            $this->kaydetBireyIletisimBilgiler
            (
                $birey_id,
                $value["iletisim_tur_id"],
                $value["iletisim_icerik"]
            );

        if($s2["hata"] == "0")
            $this->pasifYapBireyDigerIletisimBilgiler($birey_id,
$s2["iletisim_id"], $value["iletisim_tur_id"]);
    }
    return $s;
}
```

Şekil 4.66. Kişinin Bütün İletişim Bilgilerini Kaydeden Fonksiyon.

4.4.1.6. Kart (RFID) Bilgisi Denetimleri ve Fonksiyonları

Üniversitede çalışan akademik, idari personel ve bütün öğrencilere akıllı kimlik kartı (RFID) verilmektedir. Bu akıllı kimlik kartları fakülte girişlerindeki kapılardan geçişlere, yemekhanedeki ücretlerin tahsiline ve kütüphanedeki kitapları ödünç alınmasına kadar birçok alanda kullanılmaktadır. Hem üniversite personeli hem de öğrencilerin bilgisi OVT'de olduğu için akıllı kimlik kartlarının bilgisi de OVT'de tutulmaktadır.

Üniversite akıllı kartları basmak için gerekli donanıma sahiptir. Ancak dönem başında gelen öğrenci sayısı çok fazla olduğu için dönem başındaki kart basım işlemleri firmalar aracılığıyla yapılmaktadır. Akıllı kartların toplu basımında OVT'den gönderilen Excel listesi firmaya iletilir, firma bu listedeki verilere göre kartları basar daha sonra kartların idlerini (RFID) Excel listesine ekler, kart idlerinin olduğu liste OVT'ye yüklenir. OVT'den liste alma işlemi ve kart idleri yükleme işlemi akıllı kart

basım programı aracılığıyla gerçekleştirilir. Akıllı kart programı, verileri servisler aracılığıyla alır ve gönderir.

Üniversite tarafından basılan kartlar genellikle tekil işlemlerdir. Yani personel veya öğrenci kimlik kartlarını kaybettiklerinde veya dönem başladıktan sonra gelen öğrenciler (yatay geçiş ile gelen öğrenci veya ek kontenjan ile yerleşen öğrenciler gibi) için kart basım işlemi yapılmaktadır. Yeni akıllı kart talebinde bulunacak öğrenci ÖBS sistemi üzerinden personel ise Portal bilgi sistemi üzerinden bu taleplerini gerçekleştirebilirler. Bir kişinin akıllı kart talep ve basım adımları şu şekilde gerçekleşir:

1. ÖBS ve Portal bilgi sistemi bu talepleri web servisler aracılığıyla OVT'ye iletilir,
2. Talep OVT tarafından web servisler aracılığıyla akıllı kart basım programına iletilir,
3. Akıllı kart basıldıktan sonra, akıllı kart programı OVT'ye kişinin kart bilgilerini gönderir,
4. OVT'te kart bilgisini veri tabanına kaydeder ve kişiyi kısa mesaj yoluyla bilgilendirir.

ÖBS veya Portal bilgi sistemlerinde kişilerin akıllı kart basım talebi için yani yukarıdaki birinci adımı gerçekleştirmek için üç tane fonksiyona ihtiyaç vardır. Bunlar; kişilerin yeni kart talebinde bulunabilmeleri için "ekleBireyKartTalep", daha önceden bulundukları kart basım taleplerinin ne aşamada olduklarını görebilmeleri için "getirBireyKartTalepler" ve eğer talep edilen akıllı kart henüz basılmamışsa, akıllı kart basım taleplerini silmek için "silBireyKartTalep" fonksiyonlarıdır. Akıllı kart basımı için oluşturulan fonksiyonlar kendi içlerinde gerekli kontrolleri yaptıktan sonra ekleme veya silme işlemi gerçekleştirirler. Bu üç fonksiyon (ekleBireyKartTalep, getirBireyKartTalepler ve silBireyKartTalep) ÖBS ve Portal bilgi sistemleri tarafından kullanılmaktadır.

```
public function ekleBireyKartTalep($birey_unvan_essiz_id,
$kartbasim_talep_tur_id)
{
    $sorgu = $this->db->prepare ("SELECT birey_unvan_id,rfid FROM
tbl_bireyler_unvanlar JOIN tbl_bireyler_unvanlar_eslestirmeler
USING(birey_unvan_id) WHERE birey_unvan_essiz_id = ? AND
veri_kaynagi = ?");
    $sorgu->execute(array(
        $birey_unvan_essiz_id,
        $this->veri_kaynagi
    ));
    $bu = $sorgu->fetch(2);

    if($bu["birey_unvan_id"]>0)
    {
        if($bu["rfid"]>0 && $kartbasim_talep_tur_id == 1)
        {
            return
                array(
                    "hata" => "1",
                    "mesaj" => "Kişinin bu ünvanı için daha önceden
kart çıkarıldığı için bu seçeneği seçemezsiniz"
                );
        }

        $sorgu = $this->db->prepare ("SELECT
birey_kartbasim_talep_id FROM tbl_bireyler_kartbasim_talepler WHERE
birey_unvan_id = ? AND basim_tarih IS NULL");
        $sorgu->execute(array( $bu["birey_unvan_id"] ));
        if($sorgu->fetch(2)["birey_kartbasim_talep_id"]>0)
        {
            return
                array(
                    "hata" => "1",
                    "mesaj" => "Daha önceden kart talebinde
bulunmauşsunuz. Kart basım işlemi gerçekleştirildikten sonra kısa
mesaj yoluyla bilgilendirileceksiniz."
                );
        }

        $sorgu = $this->db->prepare ("SELECT
birey_kartbasim_talep_id FROM tbl_bireyler_kartbasim_talepler WHERE
birey_unvan_id = ? AND teslim_tarih IS NULL");
        $sorgu->execute(array( $bu["birey_unvan_id"] ));
        if($sorgu->fetch(2)["birey_kartbasim_talep_id"]>0)
        {
            return
                array(
                    "hata" => "1",
                    "mesaj" => "Henüz basılmamış bir akıllı kart
talebiniz bulunduğu için yeni talepte bulunamazsınız"
                );
        }

        $sorgu = $this->db->prepare ("INSERT INTO
```

```
tbl_bireyler_kartbasim_talepler (birey_unvan_id,
kartbasim_talep_tur_id, talep_tarih) VALUES (?,?,now())");
        return
            $sorgu->execute(array(
                $bu["birey_unvan_id"],
                $kartbasim_talep_tur_id
            )) ?
                array(
                    "hata" => "0",
                    "mesaj" => "Akıllı kart basım talebiniz
kaydedilmiştir."
                ):
                 array(
                    "hata" => "1",
                    "mesaj" => "Akıllı kart basım talebiniz
kaydedilemedi."
                );
    }
    else
        return
            array(
                "hata" => "1",
                "mesaj" => "Kişi bulunamadı"
            );
}
```

Şekil 4.67. Kart Talebi Ekleme Fonksiyonu.

Şekil 4.67.'deki fonksiyon kişilerin (akademik, idari personel ve öğrencilerin) yeni akıllı kart basımı için talep oluşturdukları fonksiyondur. Bu fonksiyon iki parametre almaktadır:

- $birey_unvan_essiz_id: Tam sayı tipinde değişkendir. Kişi unvanının diğer bilgi sistemlerindeki (ÖBS veya Portal) eşsiz "id" değerini beklemektedir. Kişinin birden çok unvanı (bir personel veya iki öğrenci gibi) bulunabilir. Bu unvanlardan hangisine yeni kart basımı yapılması gerektiği bilgisine ihtiyaç duyulduğundan dolayı birey eşsiz "id" değeri yerine birey unvan eşsiz "id" değeri kullanılmıştır.
- $kartbasim_talep_tur_id: Tam sayı tipinde değişkendir. Kişinin ne sebeple (kayıp çalıntı veya daha önceden çıkartılmadı gibi) yeni akıllı kart basımını talep ettiği bilgisidir. Bu değişkenin alabileceği değerler "tur_kartbasim_talepler" tablosunda bulunmaktadır.

Dönüt olarak iki adet anahtarı (hata, mesaj) olan bir dizi dönmektedir. Hata anahtarının "0" olması akıllı kart basım talebinin alındığı anlamına gelmektedir. Eğer işlemler sırasında hata meydana gelmişse dönen dizinin hata anahtarı "1" olur ve hatanın açıklaması "mesaj" anahtarında verilir.

```
public function getirBireyKartTalepler($birey_unvan_essiz_id)
{
    $sorgu = $this->db->prepare
    (
        "SELECT
            birey_kartbasim_talep_id,
            kartbasim_talep_tur_ad AS 'talep_nedeni',
            talep_tarih,
            basim_tarih,
            teslim_tarih
        FROM tbl_bireyler_unvanlar
        JOIN tbl_bireyler_unvanlar_eslestirmeler
USING(birey_unvan_id)
        JOIN tbl_bireyler_kartbasim_talepler USING(birey_unvan_id)
        JOIN tur_kartbasim_talepler USING(kartbasim_talep_tur_id)
        WHERE birey_unvan_essiz_id = ? AND veri_kaynagi = ?"
    );
    $sorgu->execute(array(
        $birey_unvan_essiz_id,
        $this->veri_kaynagi
    ));
    return $sorgu->fetchAll(2);
}
```

Şekil 4.68. Akıllı Kart Basım Taleplerinin Listesini Dönen Fonksiyon.

Şekil 4.68.'deki fonksiyon kişinin akıllı kart taleplerinin listelendiği fonksiyondur. Fonksiyonun amacı hem kişi talep ettiği akıllı kartın hangi aşamada (basıldı veya teslim edildi gibi) olduğunu görmesi hem de eğer kartı henüz basılmadıysa bu talebi silebilme imkânını sağlamaktır. Bu fonksiyon sadece bir parametre almaktadır. Bu parametre yani $birey_unvan_essiz_id, kişi unvanının diğer bilgi sistemlerindeki eşsiz "id" değeridir.

Dönüt olarak içerisinde kişinin akıllı kart taleplerinin listesini iki boyutlu bir dizi olarak verir. Dizi talebin eşsiz "id" değerini "birey_kartbasim_talep_id", talep nedenini "kartbasim_talep_tur_ad", talep edilen tarihi "talep_tarih", basım tarihini "basim_tarih" ve teslim tarihini "teslim_tarih" barındırır.

```
public function silBireyKartTalep($birey_unvan_essiz_id,
$birey_kartbasim_talep_id)
{
    $sorgu = $this->db->prepare
    (
        "SELECT basim_tarih
        FROM tbl_bireyler_unvanlar
        JOIN tbl_bireyler_unvanlar_eslestirmeler
USING(birey_unvan_id)
        JOIN tbl_bireyler_kartbasim_talepler USING(birey_unvan_id)
        WHERE
            birey_unvan_essiz_id = ? AND
            birey_kartbasim_talep_id = ? AND
            veri_kaynagi = ?"
    );
    $sorgu->execute(array(
        $birey_unvan_essiz_id,
        $birey_kartbasim_talep_id,
        $this->veri_kaynagi
    ));
    $basim_tarih = $sorgu->fetch(2)["basim_tarih"];
    if($basim_tarih)
    {
        return
            array(
                "hata" => "1",
                "mesaj" => "Akıllı kart basımı yapıldığı için bu
talebi silemezsiniz"
            );
    }
    else
    {
        $sorgu = $this->db->prepare("DELETE FROM
tbl_bireyler_kartbasim_talepler WHERE birey_unvan_essiz_id = ? AND
birey_kartbasim_talep_id = ?");
        return
            $sorgu->execute(array(
                $birey_unvan_essiz_id,
                $birey_kartbasim_talep_id
            )) ?
                array(
                    "hata" => "0",
                    "mesaj" => "Akıllı kart basım talebiniz
silinmiştir."
                ):
                array(
                    "hata" => "1",
                    "mesaj" => "Akıllı kart basım talebiniz
silinememiştir."
                );
    }
}
```

Şekil 4.69. Kişinin Kart Talebini Silen Fonksiyon.

Şekil 4.69.'deki fonksiyon kişinin talep ettiği akıllı kart basımını silmek için kullanılmaktadır. Aslında kişinin akıllı kart basım talebinin silmek içi sadece "tbl_bireyler_kartbasim_talepler" tablosundaki eşsiz alan olan "birey_kartbasim_talep_id" yeterlidir. Ancak bu sayı ardışık olarak arttığı için bir hataya sebep olabilir. Kişi unvanının diğer bilgi sistemlerindeki eşsiz "id" değeri olan "birey_unvan_essiz_id" değeri de istenmiştir. Bu fonksiyon iki parametre almaktadır:

- $birey_unvan_essiz_id: Tam sayı tipinde değişkendir. Kişi unvanının diğer bilgi sistemlerindeki (ÖBS veya Portal) eşsiz "id" değerini beklemektedir.
- $birey_kartbasim_talep_id: Tam sayı tipinde değişkendir. Kişinin talep ettiği akıllı kart basımının eşşiz id değeridir.

Şimdiye kadar kişilerin akıllı kart talep işlemlerinin ÖBS ve Portal ayağının (akıllı kartların talep edilmesi, taleplerin listelenmesi ve gerekli durumlarda silinmesi) fonksiyonları oluşturuldu. Şimdi ise akıllı kart talep işlemlerinin KBS yani kart basım program ile alakalı kısımları için fonksiyonlar oluşturulacaktır. Bu fonksiyonlar akıllı kart basımı taleplerinin listesini alma, kart basımından sonra kart bilgilerini OVT'ye gönderme ve kart tesliminde ise tarih bilgisini OVT'ye gönderme işlemlerini içermektedir.

```
public function getirBasilmamisKartTalepler()
{
    $sorgu = $this->db->prepare
    (
        "SELECT birey_unvan_id, birey_ad, birey_soyad,
birey_kimlik_no, unvan_ad, birim_ad, sicil_ogrenci_no, il, ilce,
cilt_no, aile_sira_no, mahalle_koy, dogum_tarihi, dogum_yeri,
baba_adi, anne_adi, birey_kartbasim_talep_id,
kartbasim_talep_tur_ad, kartbasim_talep_tur_id
        FROM tbl_bireyler_kartbasim_talepler
        JOIN tur_kartbasim_talepler USING(kartbasim_talep_tur_id)
        JOIN tbl_bireyler_unvanlar USING(birey_unvan_id)
        JOIN tbl_unvanlar USING(unvan_id)
        JOIN tbl_birimler USING(birim_id)
        JOIN tbl_bireyler USING(birey_id)
        WHERE basim_tarih IS NULL"
    );
    $sorgu->execute();
    return $sorgu->fetchAll(2);
}
```

Şekil 4.70. Basılmayan Akıllı Kart Basım Taleplerini Dönen Fonksiyon.

Şekil 4.70.'daki fonksiyon hiçbir parametre almadan akıllı kart basım talebinde bulunup basılmayan bütün kayıtları listeler. Dönen verinin içerisinde akıllı kart basım talebinde bulunan kişilerin özlük bilgileri (adı, soyadı, doğum tarihi gibi), talep nedeni ve bu talebin eşsiz "id" değerini bulunmaktadır. Talebin eşsiz "id" değerinin gönderilmesindeki amaç; akıllı kart basım işlemi yapıldıktan sonra kart bilgisinin (RFID) OVT'ye bu eşsiz değerle gönderilmesi gerektiğindedir. Talep sebebinin gönderilme nedeni ise; kart teslim edilirken ücrete tabi olup olmadığı bilgisi içindir.

```
public function ekleBireyKartTalepBasimTarih($birey_unvan_id,
$birey_kartbasim_talep_id, $rfid)
{
    $sorgu = $this->db->prepare("UPDATE
tbl_bireyler_kartbasim_talepler SET basim_tarih = now() WHERE
birey_kartbasim_talep_id = ? AND birey_unvan_id = ?");
    if($sorgu->execute(array($birey_kartbasim_talep_id,
$birey_unvan_id)))
    {
        $sorgu = $this->db->prepare("UPDATE tbl_bireyler_unvanlar
SET rfid = ? WHERE birey_unvan_id = ?");
        if($sorgu->execute(array($rfid, $birey_unvan_id)))
        {
            $sorgu = $this->db->prepare
            (
                "SELECT b.birey_id, b.birey_ad, b.birey_soyad,
i.iletisim_icerik AS 'cep_tel'
                FROM tbl_bireyler_kartbasim_talepler kt
                JOIN tbl_bireyler_unvanlar USING(birey_unvan_id)
                JOIN tbl_bireyler b USING(birey_id)
                JOIN tbl_bireyler_iletisim_bilgiler i ON i.birey_id
= b.birey_id AND i.statu_id = 10 AND i.iletisim_tur_id = 2
                WHERE birey_kartbasim_talep_id = ? AND
birey_unvan_id = ?"
            );
            $sorgu->execute(array(
                $birey_kartbasim_talep_id,
                $birey_unvan_id
            ));
            $b = $sorgu->fetch(2);
            if(strlen($b["cep_tel"]) == "11")
                $this->gonderSMS($b["cep_tel"], "Sayın" .
$b["birey_ad"] ." ". $b["birey_ad"]. " kimlik kartınız basılmıştır.
Kimlik kartınızı E-Kartlar biriminde alabilirsiniz.");
            return array("hata" => "0", "mesaj" => "Kişinin kart
bilgisi güncellendi");
        }else return array("hata" => "1", "mesaj" => "Kişinin kart
bilgisi güncellenemedi");
    }else return array("hata" => "1", "mesaj" => "Kart teslim tarihi
güncellenemedi");
}
```

Şekil 4.71. Kişi Akıllı Kart Basım Tarihi Ekleme Fonksiyonu.

Şekil 4.71.'deki fonksiyon KBS (kart basım programı) ile gelen kart talep id değerine göre önce kartın teslim tarihini günceller daha sonra OVT'de bulunan akıllı kartın bilgisini (RFID) günceller son olarak da OVT'de kişin telefonu varsa kişiye kimlik kartının basıldığı hakkında kısa mesaj atarak kişiyi bilgilendirir. Fonksiyon üç parametre almaktadır. Bunlar; $birey_kartbasim_talep_id, $birey_unvan_id ve $rfid adındaki değişkenlerdir. Bu değişkenlerden $rfid değişkeni akıllı kartın numarasını barındırır. Diğer iki değişkende tam sayı tipindedir. Aslında fonksiyon sadece $birey_kartbasim_talep_id kullanarak işlemlerini gerçekleştirebilir ancak bu değer ardışık sayılardan ibaret olduğu için en küçük hatada yanlış kişiye mesaj gitmesi gibi durumlar olabilir. Bu sorunu engellemek için $birey_unvan_id kontrol amaçlı eklenmiştir.

```
public function ekleBireyKartTalepTeslimTarih($birey_unvan_id,
$birey_kartbasim_talep_id)
{
    $sorgu = $this->db->prepare("UPDATE
tbl_bireyler_kartbasim_talepler SET teslim_tarih = now() WHERE
birey_kartbasim_talep_id = ? AND birey_unvan_id = ?");
    return
        $sorgu->execute(array($birey_kartbasim_talep_id,
$birey_unvan_id)) ?
            array(
                "hata" => "0",
                "mesaj" => "Kart teslim tarihi güncellendi"
            ):
            array(
                "hata" => "1",
                "mesaj" => "Kart teslim tarihi güncellenemedi"
            );
}
```

Şekil 4.72. Kişi Akıllı Kart Teslim Tarihi Ekleme Fonksiyonu.

Şekil 4.72.'deki fonksiyon akıllı kartın basımı sonrasında kişiye teslim edilirken çalıştırılacak fonksiyondur. Bu fonksiyon akıllı kart teslimi sonrasında çalıştırılmazsa kişi yeni akıllı kart talebinde bulunamaz.

4.4.1.7. Tür Bilgilerini Listeleyen Fonksiyonları

OVT'ye birim, unvan ve iletişim gibi bilgiler eklenirken bazı tür değerleri istenmektedir. Örneğin iletişim bilgisi eklenirken bu bilginin türü (telefon veya adres

gibi) istenmektedir. Tabi, bu değerler metin olarak değil de tam sayı tipinde istenmektedir. Ancak bu tam sayı tipindeki değerler diğer bilgi sistemlerinde (ÖBS veya Portal) bulunmamaktadır. Diğer bilgi sistemlerinin bu verileri alabilmesi için **Şekil 4.73**.'daki fonksiyonlar oluşturulmuştur.

```
public function getirTurKartBasimTalepler() {
    $sorgu = $this->db->prepare ("SELECT kartbasim_talep_tur_id,
kartbasim_talep_tur_ad FROM tur_kartbasim_talepler");
    $sorgu->execute();
    return $sorgu->fetchAll(2);
}

public function getirTurIletisimler() {
    $sorgu = $this->db->prepare ("SELECT iletisim_tur_id,
iletisim_tur_ad FROM tur_iletisim");
    $sorgu->execute();
    return $sorgu->fetchAll(2);
}

public function getirTurUnvanlar() {
    $sorgu = $this->db->prepare ("SELECT unvan_tur_id, unvan_tur_ad
FROM tur_unvanlar");
    $sorgu->execute();
    return $sorgu->fetchAll(2);
}

public function getirTurBirimler() {
    $sorgu = $this->db->prepare ("SELECT birim_tur_id,
birim_tur_ad, kirilim FROM tur_birimler");
    $sorgu->execute();
    return $sorgu->fetchAll(2);
}

public function getirTurStatuler() {
    $sorgu = $this->db->prepare ("SELECT statu_tur_id, statu_tur_ad
FROM tur_statuler");
    $sorgu->execute();
    return $sorgu->fetchAll(2);
}

public function getirStatuler($statu_tur_id) {
    $sorgu = $this->db->prepare ("SELECT statu_id, statu_ad FROM
tbl_statuler WHERE statu_tur_id = ?");
    $sorgu->execute(array($statu_tur_id));
    return $sorgu->fetchAll(2);
}
```

Şekil 4.73. Tur Listelerini Döndüren Fonksiyonlar.

4.4.2. Veri Gönderen Servis

Bu servisler, bilgi sistemlerinin (AKK, AKY, HGS veya KBS gibi) ihtiyaç duyduğu verileri sağlamak için oluşturulmuştur. Bilgi sisteminin ihtiyaç duyduğu ve elinde olan veriler farklılık gösterebilir. Örneğin HGS kişi bilgilerini kişinin kimlik numarasını göndererek sorgularken, AKK ise kişinin bilgilerini akıllı kart bilgisini (RFID) göndererek sorgulama yapar. Bundan dolayı bu servisteki metotlar üç temel gruba ayrılmıştır. Bunlar; kart bilgisiyle veri gönderen fonksiyonlar, kişi kimlik numarasıyla veri gönderen fonksiyonlar ve devlet kurumlarının üniversiteye sağladığı servis fonksiyonlarıdır.

4.4.2.1. Kart Bilgisiyle Veri Gönderen Fonksiyonlar

Kart bilgisiyle veri gönderen fonksiyonlar üç tanedir. Bunlardan birincisi kişinin fotoğrafını döndüren fonksiyon, ikincisi kişinin özlük ve unvan bilgilerini döndüren fonksiyon, üçüncüsü ise kişinin iletişim bilgilerini döndüren fonksiyondur. Kişinin özlük bilgileri ve kişinin fotoğrafı aynı fonksiyon içerisinde döndürülebilirdi. Ancak bilgi sistemleri (AKK, AKY, HGS gibi) kişinin fotoğrafına her zaman ihtiyaç duymazlar. Kişi bilgilerinin her sorgulanışında büyük boyutlardaki fotoğrafların veri transferini yavaşlatmaması için kişi bilgileri ve kişinin fotoğrafı iki farklı fonksiyona parçalanmıştır.

Kart bilgisiyle veri gönderen fonksiyonlar AKK (Akıllı kart kapı girişleri), AKY (Akıllı kart yemekhane girişleri) ve KBS (Kütüphane bilgi sistemi) tarafından kullanılması amacıyla yazılmıştır çünkü yukarıdaki bu sistemler kapı girişlerinde kullandıkları kiosk cihazlarıyla kişilerin akıllı kartlarını okuyarak kendi iç işlemlerini gerçekleştirmektedirler.

```
public function getirBireyFotografByRFID($rfid)
{
    $sorgu = $this->db->prepare ("SELECT fotograf FROM
tbl_bireyler_unvanlar JOIN tbl_bireyler USING(birey_id) WHERE rfid =
? ");
    $sorgu->execute($rfid);
    $foto = $sorgu->fetch(2)["fotograf"];
    return
        base64_encode(file_get_contents($foto));
}
```

Şekil 4.74. Kart Bilgisine Göre Kişinin Fotoğrafını Dönen Fonksiyon.

Şekil 4.74.'deki fonksiyon gönderilen akıllı kart numarasına (RFID) göre kişinin fotoğraf bilgisini base64 encode şeklinde döndürmektedir. Fonksiyon $rfid adında ve metin tipinde tek bir parametre almaktadır. Web servisin veri transferi sırasında fotoğraf dosyasının içerisindeki özel karakterler sorun teşkil edebileceği için dosyanın içeriği base64 encode işleminden geçirilmiştir.

```
public function getirBireyBilgilerByRFID($rfid)
{
    $sorgu = $this->db->prepare
    (
        "SELECT bu.birey_unvan_id, bu.sicil_ogrenci_no,
b.birey_kimlik_no, b.birey_ad, b.birey_soyad, s.statu_ad,
s.statu_id, u.unvan_ad, u.unvan_id, tu.unvan_tur_ad,
tu.unvan_tur_id, bi.birim_ad, bi.birim_id
        FROM tbl_bireyler_unvanlar bu
        JOIN tbl_unvanlar u USING(unvan_id)
        JOIN tur_unvanlar tu USING(unvan_tur_id)
        JOIN tbl_birimler bi USING(birim_id)
        JOIN tbl_bireyler b USING(birey_id)
        JOIN tbl_statuler s ON s.statu_id = bu.statu_id
        WHERE rfid = ?"
    );
    $sorgu->execute(array($rfid));
    return $sorgu->fetch(2);
}
```

Şekil 4.75. Kart Bilgisine Göre Kişinin Bilgilerini Getiren Fonksiyon.

Şekil 4.75.'deki fonksiyon gelen akıllı kart bilgisine göre kişinin özlük ve unvan bilgilerini döndürmektedir. Fonksiyon dizi tipindeki veriyi döndürmektedir. Dizinin içerisinde kişinin istenilen bütün verileri bulunmaktadır.

```
public function getirBireyIletisimBilgilerByRFID($rfid,
$iletisim_tur_id) {
    $sorgu = $this->db->prepare (
        "SELECT iletisim_icerik
        FROM tbl_bireyler_unvanlar bu
        JOIN tbl_bireyler_iletisim_bilgiler i ON
            bu.birey_id = i.birey_id AND
            i.statu_id = 10
        WHERE rfid = ? AND iletisim_tur_id = ?"
    );
    $sorgu->execute($rfid, $iletisim_tur_id);
    return $sorgu->fetch(2);
}
```

Şekil 4.76. Kart Bilgilerine Göre Kişinin İletişim Bilgisini Dönen Fonksiyon.

Şekil 4.76.'deki fonksiyon gönderilen akıllı kart bilgisine ve iletişim türüne göre kişinin o türdeki aktif iletişim bilgisini döndürmektedir.

4.4.2.2. Kimlik Numarasıyla Veri Gönderen Fonksiyonlar

Kimlik numarasıyla veri gönderen fonksiyonlar da kart bilgileriyle veri gönderen fonksiyonlar gibi üç tanedir. Kimlik numarasıyla veri gönderen fonksiyonlar ile kart bilgileriyle veri gönderen fonksiyonlar temel olarak aynı işlemleri yapmaktadır. Bu fonksiyonların aralarındaki en büyük fark birisinde kart bilgisi kullanılması diğerinde ise kişinin kimlik numarası kullanılmasıdır. Bu fonksiyonlar arasındaki diğer bir fark; kart bilgileriyle veri gönderen fonksiyonlarda en fazla bir kayıt dönmesi, kimlik numarasıyla veri gönderen fonksiyonda ise bir veya birden fazla kayıt dönmesidir. Kart bilgisi kişinin unvan bilgisi ile eşleştirilmiş eşsiz bir değerdir yani kart numarası kişinin sadece tek bir unvanı için verilir. Eğer kişi hem üniversitenin personeli hem de öğrencisi ise bu kişiye kart numaraları farklı iki tane kart verilmektedir. Kimlik numarasıyla veri gönderen fonksiyonlarda gelen kişiye ait bütün unvan bilgileri döndürmektedir.

Kart bilgisiyle veri gönderen fonksiyonlar HGS (hızlı geçiş sistemi) tarafından kullanılması amacıyla yazılmıştır. HGS sistemi diğer sistemlerden (AKK, AKY ve KBS gibi) biraz daha farklı çalışmaktadır. Diğer sistemler sadece üniversite personeli veya öğrencileri tarafından kullanılan sistemlerdir. Ancak HGS yani hızlı geçiş sistemi hem üniversite içerisindeki kişilerin (personel-öğrenci) kullandığı hem de üniversite ile dışardan ilişkisi olan kişi veya kurumların kullandığı bir sistemdir. Üniversitenin kampüsüne araba ile girecek olan kişilerin HGS etiketlerinin olması zorunludur. Bu etiketi almak isteyen kişiler HGS sistemine giriş yaparak başvuruda bulunurlar. Bu başvuru sırasında eğer kişi üniversite personeli veya öğrencisi olduğunu belirtirse web servisler aracılığıyla OVT'den kişinin bilgileri istenir. HGS OVT'den alınan bilgilere göre kişinin doldurması gereken alanları otomatik olarak doldurur.

```
public function getirBireyFotografByKimlikNo($kimlik_no)
{
    $sorgu = $this->db->prepare ("SELECT fotograf FROM tbl_bireyler
WHERE birey_kimlik_no = ? ");
    $sorgu->execute(array($kimlik_no));
    $foto = $sorgu->fetch(2)["fotograf"];
    return
        base64_encode(file_get_contents($foto));
}
```

Şekil 4.77. Kimlik Numarasına Göre Kişinin Fotoğrafını Dönen Fonksiyon.

Şekil 4.77.'deki fonksiyon gönderilen kimlik numarasına göre kişinin fotoğraf bilgisini base64 encode şeklinde döndürmektedir.

```
public function getirBireyBilgilerByKimlikNo($kimlik_no)
{
    $sorgu = $this->db->prepare
    (
        "SELECT bu.birey_unvan_id, bu.sicil_ogrenci_no,
b.birey_kimlik_no, b.birey_ad, b.birey_soyad, s.statu_ad,
s.statu_id, u.unvan_ad, u.unvan_id, tu.unvan_tur_ad,
tu.unvan_tur_id, bi.birim_ad, bi.birim_id
        FROM tbl_bireyler b
        JOIN tbl_bireyler_unvanlar bu USING(birey_id)
        JOIN tbl_unvanlar u USING(unvan_id)
        JOIN tur_unvanlar tu USING(unvan_tur_id)
        JOIN tbl_birimler bi USING(birim_id)
        JOIN tbl_statuler s ON s.statu_id = bu.statu_id
        WHERE birey_kimlik_no = ?"
    );
    $sorgu->execute(array($kimlik_no));
    return $sorgu->fetch(2);
}
```

Şekil 4.78. Kimlik Numarasına Göre Kişinin Bilgilerini Getiren Fonksiyon.

Şekil 4.78.'deki fonksiyon gelen kimlik numarasına göre kişinin özlük ve unvan bilgilerini döndürmektedir. Fonksiyon sonucunda kişinin birden fazla kayıt bilgisi döndürebilir. Fonksiyon dizi tipindeki veriyi döndürmektedir. Dizinin içerisinde kişinin istenilen bütün verileri bulunmaktadır.

```
public function getirBireyIletisimBilgilerByKimlikNo($kimlik_no,
$iletisim_tur_id)
{
    $sorgu = $this->db->prepare
    (
        "SELECT iletisim_icerik
        FROM tbl_bireyler b
        JOIN tbl_bireyler_iletisim_bilgiler i ON
            b.birey_id = i.birey_id AND
            i.statu_id = 10
        WHERE birey_kimlik_no = ? AND iletisim_tur_id = ?"
    );
    $sorgu->execute(array($kimlik_no, $iletisim_tur_id));
    return $sorgu->fetch(2);
}
```

Şekil 4.79. Kimlik Numarasına Göre Kişinin İletişim Bilgisini Dönen Fonksiyon.

Şekil 4.79'deki fonksiyon gönderilen kimlik numarası ve iletişim türüne göre kişinin o türdeki aktif iletişim bilgisini döndürmektedir.

4.4.2.3. Devlet Kuurmlarının Üniversiteye Sağladığı Servis Fonksiyonu

Bu fonksiyon bilgi sistemlerinin ihtiyaçları doğrultusunda devlet kurumlarının üniversiteye sağladığı web servislerin (KPS,ÖSYM, YÖKSİS) erişimi için yapılmıştır. Fonksiyon kullanıcının çağrılan web servisin yetkili olduğu alanlara göre daha önceden aynı parametreler ile çağrılmışsa cache verisinden çağrılmamışsa web servisten alarak verileri döndürmektedir. Bu işlemi yaparken eğer servisin cache verisi yoksa servisten dönen cevabı gerekli tabloya kaydederek servisin aynı parametrelerle çağrılma ihtimaline karşı önlem almaktadır.

Fonksiyon üç tane parametre almaktadır. Bunlar; çağrılacak olan servisi belirlemek için $servis_id, servisin metodunu belirlemek için $metot_id ve metoda gönderilecek veri veya verileri belirlemek için $parametre değişkenleridir.

```
public function cagirDevletKurumlariServis($servis_id, $metot_id,
$parametre)
{
      $sorgu = $db->prepare ("SELECT servis_metot_alan_ad FROM
tbl_kullanicilar k JOIN tbl_servisler_metotlar_alanlar_yetkiler y
USING(kullanici_id) JOIN tbl_servisler_metotlar_alanlar a
USING(servis_metot_alan_id) JOIN tbl_servisler_metotlar m
USING(servis_metot_id)
JOIN tbl_servisler s USING(servis_id)
WHERE k.kullanici_id = :kullanici_id AND servis_id = :servis_id AND
servis_metot_id = :metot_id");

      $alanlar = $sorgu->execute(array( "kullanici_id" =>
KULLANICI_ID, "servis_id" => $servis_id, "metot_id" => $metot_id));

      $sorgu = $db->prepare("SELECT servis_metot_cache FROM
tblservisler_metotlar_cache WHERE parametre = :parametre AND
servis_metot_id = :metot_id");

      $cache = $sorgu->execute(array("parametre" => $parametre,
"metot_id"  => $metot_id));
      if(!$cache) {
      switch ($servis_id){
         case "4":
            include KPS;
            $servis_cevap = new kps($metot_id, $parametre)
         break;
         case "5":
            include OSYM;
            $servis_cevap = new osym($metot_id, $parametre)
         break;
         case "6":
            include YOKSIS;
            $servis_cevap = new yoksis($metot_id, $parametre)
         break;

         default:
            return array("hata"=>1,"mesaj"=>"Servis bulunamadı");
         break;
      }
      $sorgu = $db->prepare("INSERT INTO tblservisler_metotlar_cache
(servis_metot_id, parametre, cache) VALUES (?,?,?)");
      $sorgu->execute(array("parametre" => $parametre, "metot_id" =>
$metot_id, "cache" => $servis_cevap));
      }
      else
            $servis_cevap = $cache;

      foreach ($servis_cevap as $key => $value)
      {
            if(!array_key_exists($key, $alanlar))
                  $servis_cevap[$key] = "";
      }
      return $servis_cevap;
}
```

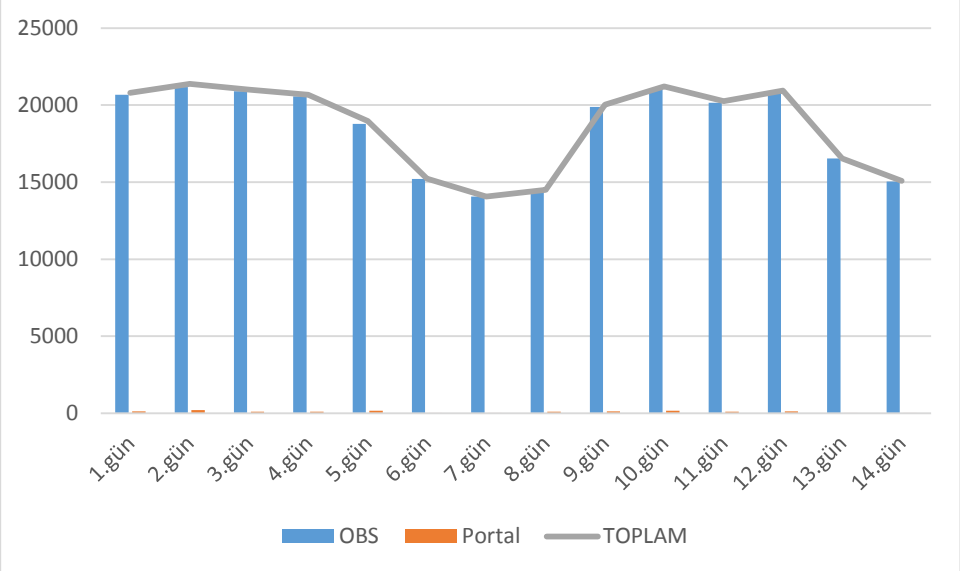
Şekil 4.80. Devlet Kurumları Web Servis Bilgileri Dönen Fonksiyon.

BEŞİNCİ BÖLÜM

ARAŞTIRMA BULGULARI VE TARTIŞMA

5.1. ÇALIŞMA İSTATİSTİKLERİ VE FAYDALARI

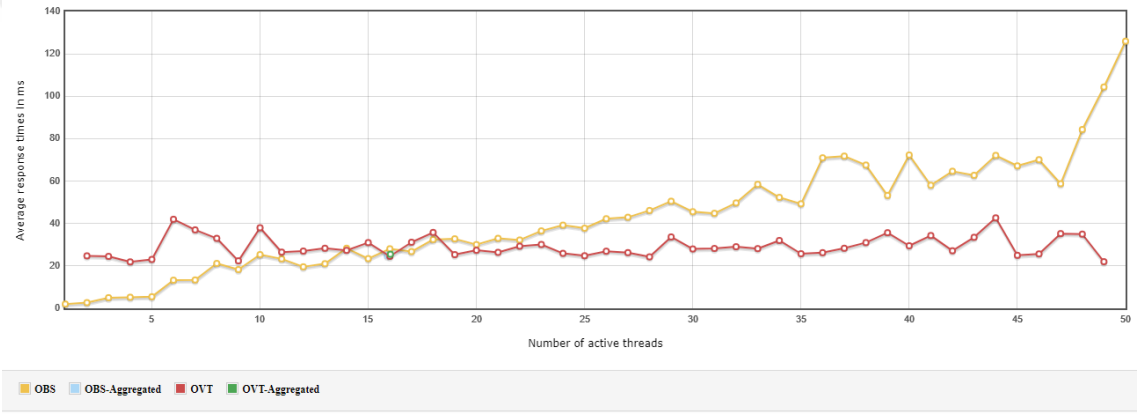
OVT oluşturulmadan önce üniversite tarafından kullanılan bilgi sistemleri birbirleriyle veri alış verişinde bulunmalarına rağmen veri alış verişlerinin bir düzeni bulunmamaktaydı. OVT bütün bilgi sistemleri arasında köprü görevi kurarak bilgi veri alışverişine bir düzen sağlanmıştır. Bu düzen sayesinde bazı kazançlar elde edilmiştir. Bu kazançlardan ilki bilgi sistemleri üzerindeki yükü hafifletmektir. OVT ve Servis Odaklı Mimari alt yapısı bulunmadığı durumda ÖBS'de bir öğrenci mezun olduğunda veya Portalda bir personelin bilgisi güncellendiğinde diğer bilgi sistemlerinde (HGS, AKK, AKY ve KBS) yer alan bilgiler güncellenmemekte, her bilgi sisteminde görevli veri kontrol hazırlama giriş persoenli bu bilgileri diğer bilgi sistemlerinde de güncelleştirmesi gerekmektedir. Bu işlem OVT oluşturulmadan sadece Servis Odaklı Mimari bulunan sistemde ise; ÖBS ve Portal dört ayrı bilgi sistemine ayrı ayrı sırayla bağlanarak bilgi göndermesi gerekmektedir. Servis odaklı mimariye OVT modülü eklendiğinde ise ÖBS ve Portal verileri sadece OVT'ye göndermekte OVT ise diğer dört farklı bilgi sistemine göndermektedir.



Şekil 5.1. OVT'ye Gönderilen Veriler (ÖBS ve Portal Bilgi Sistemleri).

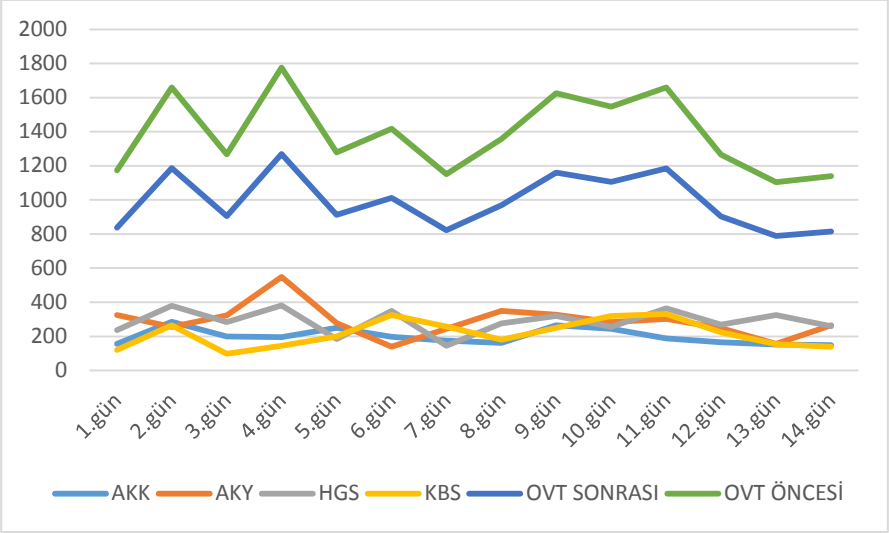
Şekil 5.1.' de OVT'ye ÖBS ve Portal tarafından günlük olarak gönderilen veri sayıları bulunmaktadır. ÖBS ve Portal on dört günde yaklaşık 250 bin kişinin verisini OVT'ye göndermektedir. OVT oluşturulmadan önce bu sayı dört kat fazla yani 1 milyon civarında olması gerekmekteydi. Bu verilerin alındığı 14 günlük periyotta OVT ÖBS üzerindeki yaklaşık 750 bin veri güncelleme yükünü kendi üzerine alarak ÖBS'nin yükünü hafifletmiştir.

ÖBS'de sanal bir yoğunluk oluşturmak ve aynı zamanda da OVT isteklerde bulunmak için "Apache" tarafından geliştirilen "JMeter" adlı program kullanılmıştır. JMeter programı belirlenen sayıda thread ve istek oluşturarak sistem testleri yapımında kullanılmaktadır.



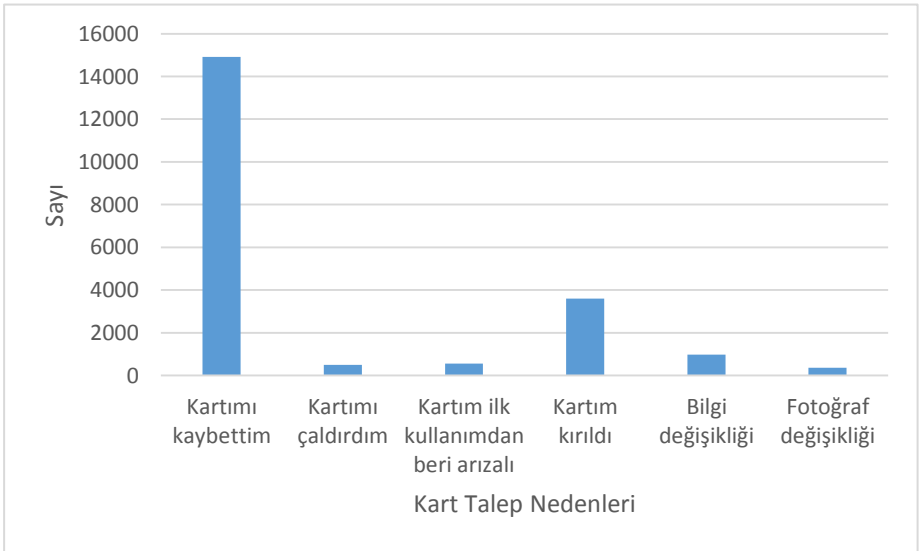
Şekil 5.2. ÖBS'nin Yoğun Zamanlarında OVT'nin Cevap süreleri.

ÖBS sistemi ders alma veya sınav notu açıklanma zamanlarında çok fazla yoğunluk yaşayabilmektedir. ÖBS'de yoğunluk yaşandığında kullanıcılara dönülen cevap süreleri uzamaktadır. ÖBS'nin cevap süresinin uzaması diğer bilgi sistemleri (HGS, AKK, AKY ve KBS) kötü yönde etkilemektedir. OVT sayesinde bu kötü etki ortadan kaldırılmıştır. **Şekil 5.2**'de ÖBS'nin yoğun olduğu zamanlarda cevap süresinin uzadığını ve OVT'nin ise bu yoğunluktan etkilenmediği gösterilmiştir.



Şekil 5.3. OVT'den Çekilen Verilerin Sayıları (AKK, AKY, HGS ve KBS).

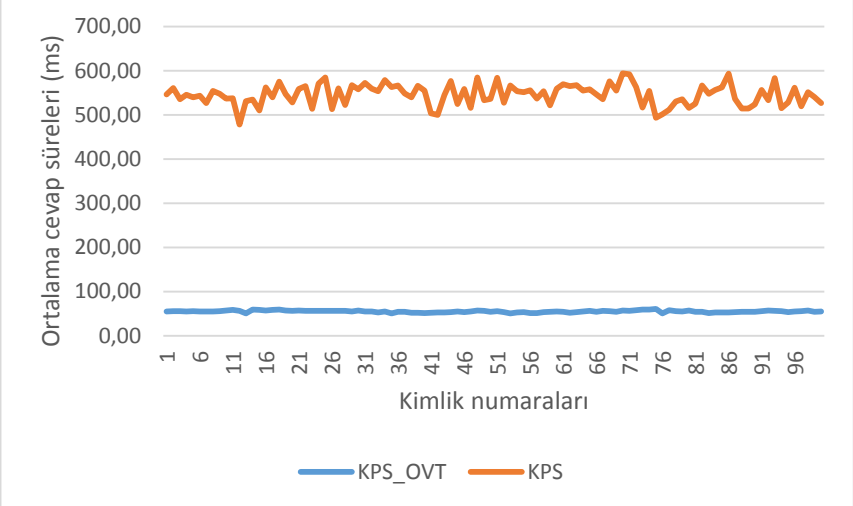
OVT oluşturulmadan önce bir bilgi sistemleri RFID veya kişinin kimlik numarasıyla kişi bilgilerine erişim sağlaması gerektiğinde OBS ve Portal sistemlerine iki ayrı istek yaparak kişinin bilgilerine erişebilmekteydiler. OVT oluşturulduktan sonra kişi ve kart bilgileri tek bir veri tabanından birleştirildiği için istek sayısı bire düşürülmüştür. **Şekil 5.3**.'de bilgi sistemlerinin (AKK, AKY, HGS ve KBS) günlük olarak OVT'den veri çekim sayıları gösterilmiştir. On dört günlük sayılara göre bilgi sistemleri OVT'den önce toplam 20 bin civarında istek yaparken OVT sonrası bu sayı yaklaşık 14 bin civarına düşürülmüştür.



Şekil 5.4. OVT Aracılığıyla Talep Edilen Kartların Sayısı

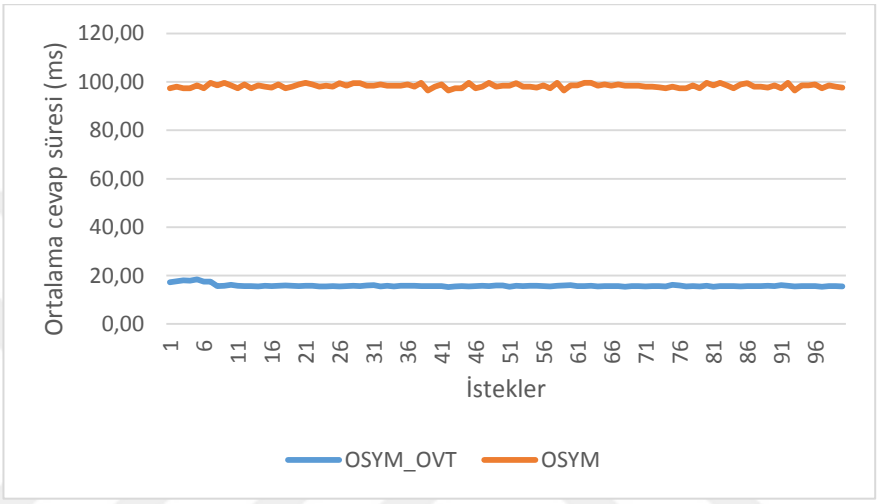
Şekil 5.4'de personel veya öğrencilerin OVT'te aracılığıyla talep ettikleri kimlik kartlarının sayıları ve talep nedenleri yer almaktadır. OVT aracılığıyla toplamda 20.889 adet kimlik talebi gerçekleştirilmiştir. OVT öncesinde kimlik kart talepleri kişinin birime bizzat başvurmaları gerekmekteydi. OVT sayesinde kimlik kart talepleri online olarak yapılabilir hale getirilerek kullanıcıların internet bağlantısı olan her yerden kart talebi oluşturabilmeleri sağlanmıştır.

Devlet kurumları tarafından üniversiteye sunulan web servisler (KPS, ÖSYM ve YÖKSİS) gün içerisinde aynı kişi için birden fazla çağrılabilir. Bunun nedeni; üniversitede bilgi sistemlerine giriş yapmış kullanıcılar kendilerine ait olan ekrandan KPS servisleri aracılığıyla kimlik bilgilerini güncelleyebilirler. Kullanıcılar gün içinde kimlik bilgilerini günceleme işlemini birden fazla yapabilirler veya ÖBS tarafından sorgulanan bir kişinin bilgisi gün içerisinde diğer bilgi sistemleri tarafından sorgulanabilir. Gün içerisinde aynı kişi için yapılan bu sorgular KPS, ÖSYM ve YÖKSİS servislerinin yükünü arttırmaktadır. OVT gün içerisinde KPS, ÖSYM ve YÖKSİS servislerinden yapılan sorgulamaları kayıt altına tutmakta ve ikinci defa yapılan istekte bu web servislere bağlanmadan daha önceden kendi veri tabanına kaydettiği veriyi dönmektedir. Bu sayede hem OVT üzerinde yapılan KPS, ÖSYM ve YÖKSİS servislerinin hızları arttırılmış hem de KPS, ÖSYM ve YÖKSİS servislerinin yükü hafifletilmiştir.



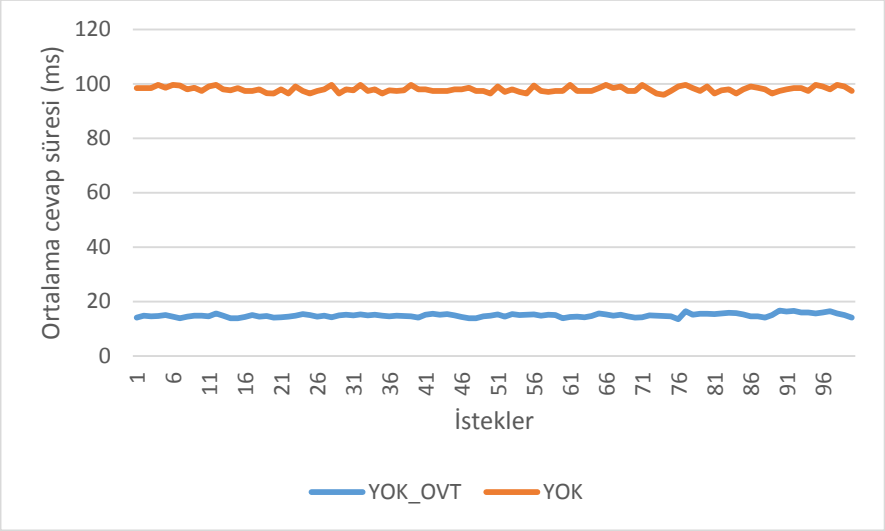
Şekil 5.5. KPS ve OVT Servislernin Ortalama Cevap Süreleri

Yüz farklı kimlik numarasının yüz defa art arda hem KPS servisinden direk olarak hem de OVT aracılığıyla sorgulanmış ve servislerin cevap sürelerini oluşturan grafik **Şekil 5.5**'de gösterilmiştir. OVT aracılığıyla yapılan yapılan sorgunun ortalama süresi yaklaşık 55.02 milisaniye olmasına karşın KPS'den yapılan sorguların ortalama cevap süresi 536.67 milisaniyedir. OVT sayesinde KPS sorguları yaklaşık olarak on üç kat daha hızlı gerçekleşmektedir.



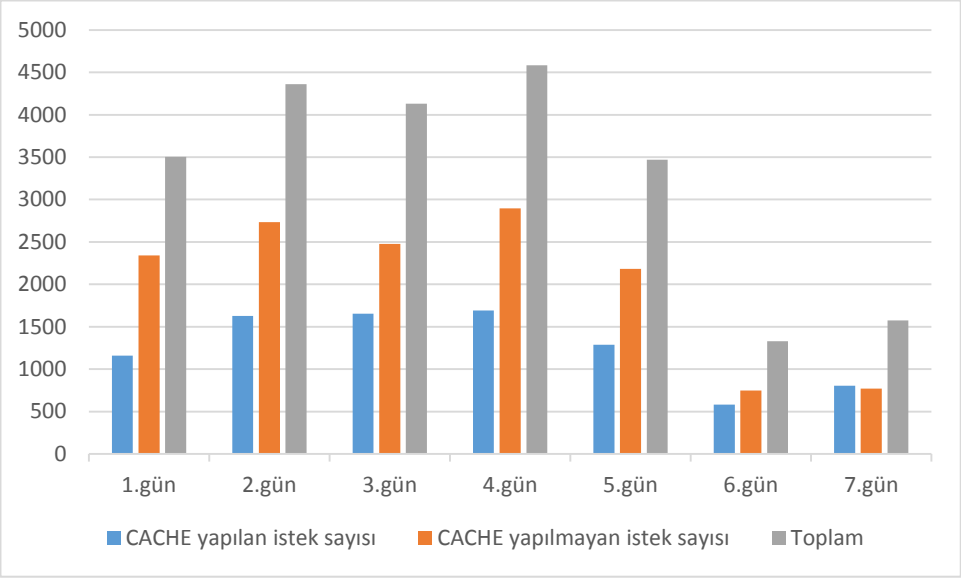
Şekil 5.6. OSYM ve OVT Servislernin Ortalama Cevap Süreleri

Yüz farklı kişinin aynı sınav sonucu yüz defa art arda hem ÖSYM servisinden direkt olarak hemde OVT aracılığıyla sorgulanmış ve servislerin cevap sürelerini oluşturan grafik **Şekil 5.6**'de gösterilmiştir. ÖSYM servisinin ortalama cevap süresi 99.35 mili saniye iken OVT aracılığıyla ÖSYM servislerinin ortalama cevap süresi 17.16 mili saniyedir. OVT sayesinde ÖSYM servislerinde yaklaşık 5.78 kat kazanç sağlanmıştır.



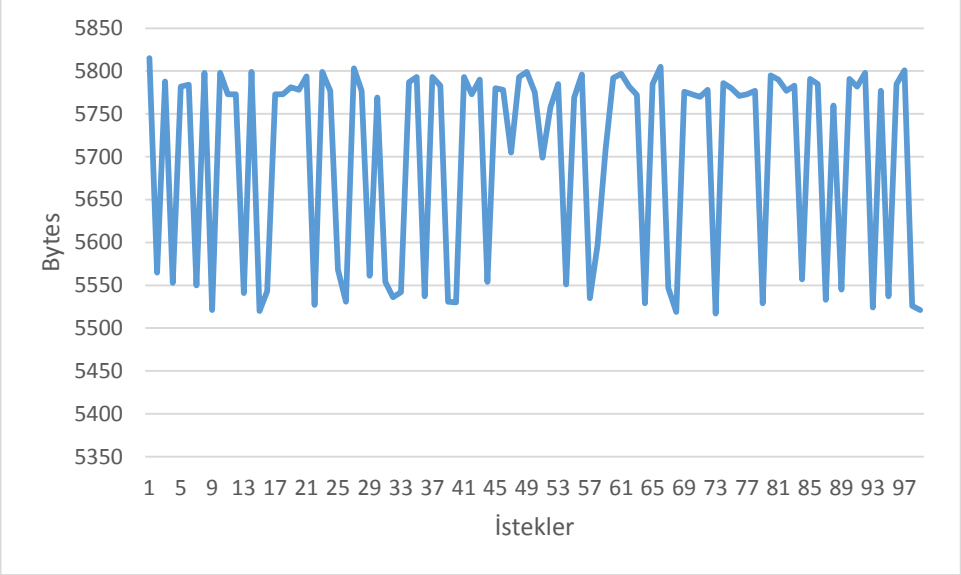
Şekil 5.7. YÖKSİS ve OVT Servislernin Ortalama Cevap Süreleri

Yüz farklı akademik personelin bilgilerini yüz defa art arda hem YÖKSİS servisinden direk olarak hemde OVT aracılığıyla sorgulanmış ve servislerin cevap sürelerini oluşturan grafik **Şekil 5.7**'de gösterilmiştir. YÖKSİS servisinin ortalama cevap süresi 99.95 milisaniye iken OVT aracılığıyla ÖSYM servislerinin ortalama cevap süresi 16.95 mili saniyedir. OVT sayesinde YÖKSİS servislerinde yaklaşık 5.89 kat kazanç sağlanmıştır.



Şekil 5.8. KPS Servislerine Yapılan İstek Sayıları.

Şekil 5.8.'de KPS servislerine yapılan bir haftalık istek sayıları ve cache yapılan sorgu sayıları bulunmaktadır. Yedi günlük verilere göre ortalama 22955 istek yapılmaktadır. Bu isteklerin 8805 tanesine KPS servislerine gitmeden OVT tarafından yanıt döndürülmektedir. Bu istekler direkt olarak KPS servislerinden sorgulanması durumunda; servisin cevap süresi 545.65 (KPS servisinin ortalama yanıt süresi) – 55.02 (OVT aracılığıyla yapılan KPS servisinin ortalama cevap süresi) * 8805 milisaniye uzayarak toplamda yaklaşık olarak haftalık 72 dakika gibi bir sürenin kayıp olmasına neden olacaktır. Ayrıca isteklerin %38.35'i dış bilgi sistemlerine gönderilmeden doğrudan cevaplanarak KPS sistemi üzerindeki yükün daha düşük olması sağlanmıştır.



Şekil 5.9. KPS Servisinin Yanıt Boyutları

KPS servis cevaplarını cache yapma işlemi hem servis cevap sürelerini hızlandırır hemde üniversitenin bant genişliğinde tasarruf yapılmasını sağlamaktadır. **Şekil 5.9**.'da KPS servislerine yapılan yüz farklı istek sonucunda dönen yanıtların boyutu gösterilmektedir. KPS servislerinin yanıtı ortalama 5700 byte dır. Bir haftalık verilere göre; 8805 (cache yapılan istek sayısı) * 5700 (ortalama yanıt boyutu) btye yani 47,8 mb üniversitenin bant genişliğinden tasarruf sağlanmaktadır.

Tablo 5.1. Gün içerisinde en fazla sorgulanan kimlik numaraları.

Kimlik Numarası	Sorgulanma sayısı
Kimlik Numarası 1	1100
Kimlik Numarası 2	180
Kimlik Numarası 3	119
Kimlik Numarası 4	106
Kimlik Numarası 5	81

OVT sayesinde bilgi sistemlerinin web servis kullanım bilgileri kayıt altında tutularak oluşabilecek anormallikler tespit edilebilir hale gelmektedir. **Tablo 5.1**.'de bir gün içerisinde en fazla sorgulanan beş kimlik numarası listelenmektedir. Bu listede özellikle "Kimlik Numarası 1" gün içerisinde çok fazla sayıda sorgulandığı dikkat çekmektedir. Yapılan araştırma sonucunda "Kimlik Numarası 1" sorgu sayısını fazla olmasının nedeninin bir yazılım hatası olduğu ortaya çıkmıştır. Sorguyu yapan bilgi sisteminde çalışan personel yazılım geliştirirken yanlışlıkla kendi kimlik numarasını kodların arasına eklemiş ve sürekli olarak aynı kimlik numarası sorgulanmıştır. OVT sayesinde kurumda oluşacak bu ve bunun gibi anomaliler kolaylıkla tespit edilebilmekte ve engellenebilmektedir.

SONUÇ VE ÖNERİLER

OVT, SOA kullanılarak oluşturulan bir yapıdır. Bu yapı Atatürk Üniversitesinde bulunan bilgi sistemlerinin sadece OVT ile iletişime geçmesiyle hem diğer bilgi sistemlerindeki verilere hem de devlet kurumları tarafından sağlanan web servislere erişim imkanı sağlamaktadır. OVT Atatürk Üniversite tarafından kullanılan bilgi sistemlerinin arasındaki ve devlet kurumları tarafından sağlanan web servislerin bilgi sistemleri arasındaki ilişki SOA kullanılarak en üst düzeye çıkarılmış, web servislerin yönetimi kolaylaştırılmış, bilgi sistemleri üzerindeki yükler hafifletilmiş ve veri paylaşımı belirli bir düzen içinde yapılması sağlanmıştır.

Yapılan araştırmalar sonucunda OVT oluşturulmasında kullanılabilecek en iyi mimarinin SOA mimarisi olduğuna karar verilmiştir. OVT'nin paydaşları olan diğer bilgi sistemlerinde veri alışverişlerinde sorun yaşanmaması ve web servislerinde bir standart olması için SOA mimarisi uygulanırken WSDL web servislerin kullanılması tercih edilmiştir.

Yapılan literatür çalışması sonucunda ortak veri tabanı oluştururken karşılaşılabilecek en büyük problemlerden birinin veri kaynaklarından gelen verilerin çoğullaşması kanısına varılmıştır. Veri çoğullaşmasının önüne geçilmediği takdirde ortak veri tabanının bir veri yığını haline geleceğinden bunun önüne geçmek için veri tekilleştirme algoritmaları hakkında literatür çalışması yapılmıştır. Bu araştırma sonucunda verinin niteliğine göre farklı yöntemler uygulanması kararlaştırılmıştır. Örnek olarak kişi verileri için MERNİS tarafından sağlanan kimlik numaralarına göre kontrol yapılırken kişilerin adresleri veya birim adları için Levenshtein Distance algoritması kullanılmıştır.

Veri tabanı oluşturulmadan önce bu veri tabanından hangi bilgilerin saklanacağı, hangi verilerin hangi bilgi sistemlerinden sağlanabileceği ve bilgi sistemlerinin ihtiyaç duyacağı veriler için gerekli analiz çalışması yapılmıştır. Bu analiz çalışması sonucunda veri tabanında gerekli tablolar ve bu tabloların alanları oluşturulmuştur.

OVT mimari olarak SOA kullanıldığı için yazılım dilinin hangi dil olacağı önemini yitirmiştir. Bundan dolayı materyal seçimi yapılırken sadece materyallerin kendi aralarındaki uyumuna dikkat edilmiştir. Yani bu uygulamada php kullanıldığı için Linux işletim sistemi tercih edilmiştir. OVT'nin esnek yapısı sayesinde Windows

işletim sistemi veya Visual c#.NET programlama dili gibi Microsoft ürünlerinin çalışmasında herhangi bir engel buunmamaktadır. Uygulama kısmında web servisleri oluşturacak fonksiyonların daha iyi anlaşılabilmesi için küçük parçalara ayrıştırılarak anlatılmıştır.

OVT sayesinde üniversitedeki bütün personel ve öğrenciler ek bir işleme gerek duymadan üniversitenin sunduğu diğer bilgi sistemlerinin avantajlarından yararlanabilirler. Daha önceden bu sistemlerden faydalanmak isteyen personeller veya öğrenciler kendilerinin üniversitenin bir personeli/öğrencisi olduğunu kanıtlamak durumunda kalmalarına rağmen OVT bu zorunluluğu ortadan kaldırmıştır. OVT sayesinde; öğrencilerin HGS etiketi alırken öğrenci belgesini ibraz etmek zorunluluğu, bu işlemler sırasında canlı veri görülemediği için belge kontrolü sırasında karşılaşabilecek küçük hatalarda HGS etiketi almaması gereken kişiler HGS etiketi alınabilmesi gibi istenmeyen durumların önüne geçilmiştir. Bu kontroller sırasında hiçbir hata olmasa bile öğrencilerin ÖBS'den alacağı öğrenci belgesini ibraz etme zorunluluğu olduğundan dolayı hem öğrenci hem de kontrol edecek personel için fazladan iş yükü ve işlemlerde daha fazla bürokrasi getirecektir. Bu işlemler OVT olmadan bütün sistemlerin birbirleriyle web servisler aracılığıyla iletişime geçmesi sayesinde de yapılabilir ancak bu defa da çok karmaşık bir web servis ağı oluşturulması gerekir. Böyle bir web servis ağının oluşturulmas ve yürütülmesi sürecinde fazla personel kaynağı gerekeceği gibi yapının karmaşıklığı nedeniyle çeşitli problemlerle karşılaşılabilir. Bu nedenle esnek, modüler ve güvenli sistem olan OVT hayata geçirilmiştir.

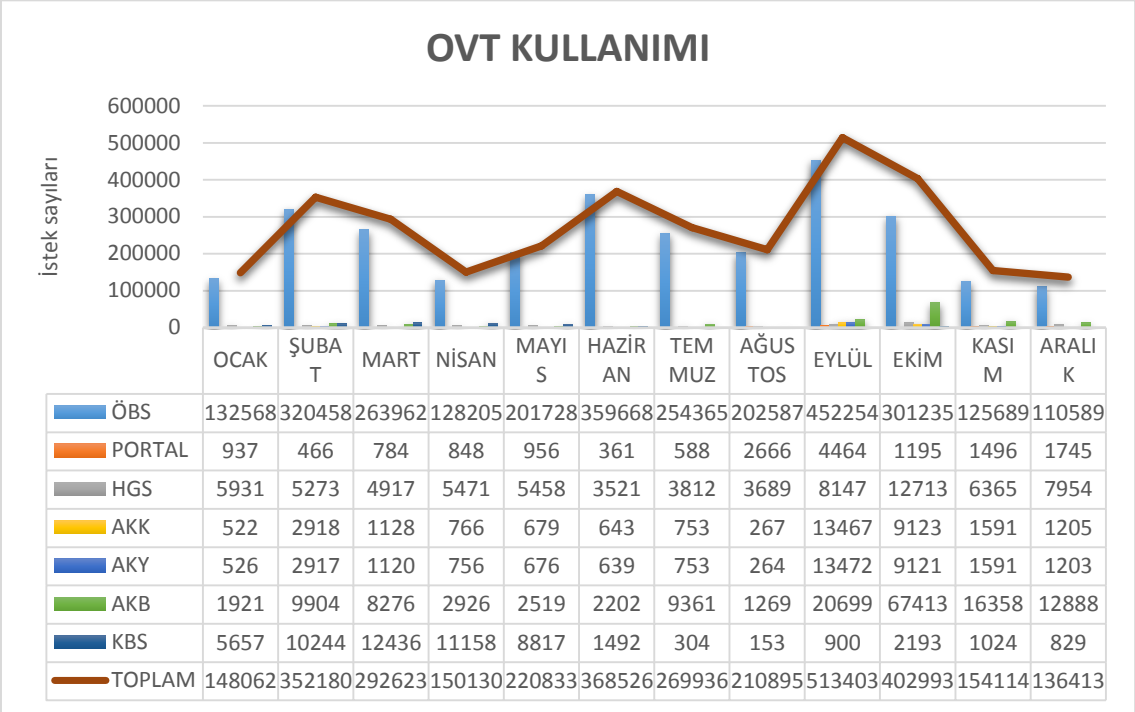
Büyük ölçekli bilgi işlem merkezlerinde ortak bir veri tabanı oluşturulmasının birçok avantajı vardır. Bunlardan bazıları:

- OVT'de bilgi güncelleme işlemi anlık olarak yapıldığından dolayı bütün bilgi sistemleri güncel verilere anında erişebilmektedir.
- Anlık güncelleme işlemlerinin yükünü OVT kendi üzerine alarak, iş yükü fazla olan (ÖBS ve Portal gibi) bilgi sistemlerinin yükünü hafifletmektedir.
- OVT'de bilgi sistemlerinden bağımsız ve servis odaklı mimari kullanılarak oluşturulduğu için diğer bilgi sistemlerinin iş yoğunluklarından etkilenmeden istikrarlı olarak çalışmaya devam edebilmektedir.

- Veriler tek bir merkezde toplandığı için diğer bilgi sistemlerinin veriye erişmesinde ve veri madeni çalışması yapımında büyük kolaylık sağlamaktadır. OVT olmadan yapılacak bir veri madeni çalışmasında bütün bilgi sistemlerinde veri alınarak bu veriler birleştirilmesi gerekir. Tabi, bu veri birleştirme işlemi sırasında kayıt çoğullaşması gibi veri madenciliği çalışmasını yanlış yönde etkileyebilecek hatalar meydana gelebilir.
- Atatürk Üniversitesi tarafından alınacak veya geliştirilecek bir bilgi sistemi diğer bütün bilgi sistemleriyle çok hızlı bir şekilde iletişime geçebilmesi sağlanmaktadır. Çünkü yeni bilgi sistemi diğer bilgi sistemlerine adapte olmasına gerek duymadan sadece OVT'ye adapte olarak diğer bilgi sistemleri ile konuşabilir hale gelebilmektedir.
- OVT bilgi sistemlerindeki anormal davranışlar tespit edilebilir hale gelmesini sağlamaktadır. Çünkü bilgi sistemlerinin yaptığı bütün işlemler OVT'de toplanmakta ve bu işlemlerin log kayıtları tutulmaktadır. Bu log kayıtları sayesinde bilgi sistemlerinin anormal davranışları tespit edilebilmektedir.
- Devlet kurumları tarafından sağlanan web servislerin (KPS, ÖSYM ve YÖKSİS) kullanıcı bilgileri diğer bilgi sistemleriyle paylaşmadığı ve bilgi sistemlerinin sadece ihtiyacı kadar veriye erişebildiği için web servislerin güvenliğini artırmaktadır.
- Devlet kurumlarının sağladığı web servislerin (KPS, ÖSYM, YÖKSİS) hem cevap süreleri cache yapılarak çok büyük oranda kısaltılmakta hem de üniversitenin bant genişliğinden tasarruf sağlamaktadır.

Atatürk Üniversitesi Ortak Veri Tabanında yaklaşık on bin personel bilgisi ve dört yüz binden fazla öğrenci bilgisi bulunmaktadır ve bu bilgiler her geçen gün artmaya devam etmektedir ("Atatürk Üniversitesi" 2019). OVT'ye en çok istek yapan bilgi sistemi ÖBS olmasının nedeni öğrenci sayısının personel sayısında çok fazla olmasıdır. OVT'de aylık çekilen/gönderilen kişi verileri değişkenlik göstermektedir. Özellikle akademik yılın başında yeni öğrenciler, sonunda ise mezun/ayrılan öğrenciler için yapılan veri gönderim/çekim işlemler istatistikleri artmaktadır. Tabi, dönemin ortalarında bu istekler göreceli olarak düşmektedir. OVT'ye yapılan aylık veri gönderimi/alımı istek sayısı yaklaşık olarak iki yüz altmış sekiz bindir. Bu sayı da

gösteriyor ki OVT'nin kullanımı oldukça yüksektir. Öğrenci/personel sayısı arttıkça veya OVT'ye yeni bilgi sistemleri eklendikçe bu sayının artması beklenmektedir.



	OCAK	ŞUBAT	MART	NİSAN	MAYIS	HAZİRAN	TEMMUZ	AĞUSTOS	EYLÜL	EKİM	KASIM	ARALIK
ÖBS	132568	320458	263962	128205	201728	359668	254365	202587	452254	301235	125689	110589
PORTAL	937	466	784	848	956	361	588	2666	4464	1195	1496	1745
HGS	5931	5273	4917	5471	5458	3521	3812	3689	8147	12713	6365	7954
AKK	522	2918	1128	766	679	643	753	267	13467	9123	1591	1205
AKY	526	2917	1120	756	676	639	753	264	13472	9121	1591	1203
AKB	1921	9904	8276	2926	2519	2202	9361	1269	20699	67413	16358	12888
KBS	5657	10244	12436	11158	8817	1492	304	153	900	2193	1024	829
TOPLAM	148062	352180	292623	150130	220833	368526	269936	210895	513403	402993	154114	136413

Şekil 6.1. OVT Kullanım İstatistikleri.

OVT sistemini kurgulama işlemi Atatürk Üniversitesinde hâlihazırda bulunan bilgi sistemlerinin verileri ve bilgi sistemlerinin ihtiyaç duyduğu verilere göre yapılmıştır. Bu yapıyı kullanarak kişilerin farklı bir bilgisi istendiğinde (öğrencilerin ağırlıklı genel ortalamaları gibi) yeni bir web servis ve veri tabanından yeni tablo/alan oluşturulmasına ihtiyaç duyulacaktır. Gelecekteki çalışmalarda daha dinamik yapı kurularak bilgi sistemlerinin kendi belirleyecekleri verileri yine kendi belirleyecekleri bilgi sistemleri ile otomatik olarak paylaşabilen bir OVT oluşturulması planlanmaktadır. Bu sayede bilgi sistemlerinin OVT'deki verileri kontrol etmesi sağlanacaktır. Yeni web servis yazımına ve veri tabanında herhangi bir değişikliğe ihtiyaç duyulmadan yeni veriler istenilen bilgi sistemine aktarılabilecektir.

Sonuç olarak büyük ölçekli bilgi işlem merkezlerinde, bütün işlemleri yapabilen tek bir bilgi sistemi tarafından sağlanan faydalar OVT oluşturularak hangi programlama dili ve alt yapı ile çalıştığı fark etmeksizin birden fazla bilgi sistemi aracılığı ile uygun maliyete, daha hızlı ve daha esnek bir şekilde gerçekleştirilebilir.

KAYNAKLAR

Atatürk Üniversitesi (2019), *Kayıtlı Öğrenci Sayıları*, Erişim (10 Temmuz 2019), https://atauni.edu.tr/aktif-ogrenci-sayilari.

Baeza-Yates, R.A., (1989). "Algorithms for string searching". *AcM sIGIR Forum*, 34-58.

Bard, G.V., (2007). "Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric". *Proceedings of the fifth Australasian symposium on ACSW, 68*, 117-124.

Booth, D. ve Canyang K. L., (2007). "Web services description language (WSDL) version 2.0 part 0: Primer". *World Wide Web Journal*, *26*, 15-19.

Box, D., Ehnebuske D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., ve Winer. D. (2000). "Simple object access protocol (SOAP) 1.1.". *World Wide Web Journal*, *08*, 3-8

Bray, T., Paoli, J., Sperberg-McQueen, C M., Maler, E., ve Yergeau, F. (2006). "Extensible markup language (XML) 1.1". *World Wide Web Journal*, 02, 27-66.

Chandrasekar, C. (2013). "An optimized approach of modified bat algorithm to record deduplication". *International Journal of Computer Applications*, *62*, 10-15.

Chinnici, R., Moreau, J.J., Ryman, A., ve Weerawarana, S. (2007). "Web services description language (wsdl) version 2.0 part 1: Core language". *W3C Recommendation, 26*, 19-38.

Cohen, W.W., Ravikumar, P., ve Fienberg. S.E. (2003). "A Comparison of String Distance Metrics for Name-Matching Tasks". *IIWeb*, 2003, 73-78.

Cordella, A., ve Tempini. N. (2015). "E-government and organizational change: Reappraising the role of ICT and bureaucracy in public service delivery". *Government Information Quarterly*, *32*, 279-286.

Damerau, F.J. (1964). "A technique for computer detection and correction of spelling errors". *Communications of the ACM*, *7*, 171-176.

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, M. ve Newling, T. (2004). *Patterns: service-oriented architecture and web services*. USA: IBM Corporation, International Technical Support Organization.

Uzun, E., Kılıçaslan Y. ve UÇAR E. (2008). "HTML, XML ve Web Servislerinin İnternet Sunucuları Üzerindeki Etkisinin İncelenmesi". *Trakya Üniversitesi Fen Bilimleri Dergisi*, 8, 81-85.

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. New Jersey, NY: Prentice Hall PTR.

Ferris, C., ve Farrell., J. (2003). "What are web services?". *Communications of the ACM*, 46, 29-34.

Feuerlicht, G., ve Govardhan., S. (2009). "SOA: Trends and directions". *International Conference on Systems Integration*. 17, 149-154.

Gujar, P.P., Desai P. (2013). "A survey of record deduplication techniques", *International Journal of Latest Trends in Engineering*. 2(4), - 246-250.

Haldar, R., ve Mukhopadhyay. D. (2011). "Levenshtein distance technique in dictionary lookup methods: An improved approach". *arXiv*, abs/1101.1232,

Jammes, F., ve Smit. H. (2005). "Service-oriented architectures for devices-the SIRENA view". *INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics*, 2005, 140-147.

Jokinen, P., Tarhio, J. ve Ukkonen, E. (1996). "A comparison of approximate string matching algorithms". *Software: Practice*, 26, 1439-1458.

Kamu Kurum ve Kuruluşlarının Büyük Ölçekli Bilgi İşlem Birimlerinde Sözleşmeli Bilişim Personeli İstihdamına İlişkin Esas ve Usuller Hakkında Yönetmelik. (2008). *T. C. Resmi Gazate*, 27097, 31 Aralık 2008.

Komoda, N.. (2006). "Service oriented architecture (SOA) in industrial systems". *4th IEEE International Conference on Industrial Informatics*, 2006,1-5.

Levenshtein, V.I. 1966. "Binary codes capable of correcting deletions, insertions, and reversals". *Soviet physics doklady*, 10(8),707-710.

MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R. ve Hamilton. B.A., 2006. "Reference model for service oriented architecture 1.0". *OASIS standard*, 12, 1-31.

MERNİS, (2019), *MERNİS Hakkında***,** Erişim (12 Mart 2019), https://www.nvi.gov.tr/hakkimizda/projeler/mernis.

Ordanini, A., ve Pasini. P. (2008). "Service co-production and value co-creation: The case for a service-oriented architecture (SOA)". *European Management Journal*, 26, 289-297.

Pimenidis, E., Iliadis, L. ve Georgiadis. C.K. (2011). "Can e-Government Systems Bridge the Digital Divide?". *Proceedings of the 5th European Conference on Information Management and Evaluation*, 403-410.

Rao, J. (2004). *Semantic web service composition via logic-based program synthesis* (Doktora Tezi). Norway Trondheim : Department of Computer and Information Science Norwegian University of Science and Technology.

Santos, W., Teixeira, T., Machado, C., Meira Jr, W., Ferreira, R. ve Guedes, D. (2007). "A scalable parallel deduplication algorithm.". *19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07), 19*, 79-86.

Walsh, N. (1998). "What is XML". https://www.xml.com/pub/a/98/10/guide0.html

Yükseköğretim Üst Kuruluşları ile Yükseköğretim Kurumlarının İdari Teşkilatı Hakkında Kanun Hükmünde Kararname. (1983). *T. C. Resmi Gazete*, 18228, 7 Ekim 1983.

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Gökhan TUTAR
Doğum Yeri ve Tarihi	Erzurum / 15.12.1986
Eğitim Durumu	
Lisans Öğrenimi	Bilgisayar ve Öğretim Teknolojileri Eğitimi
Y. Lisans Öğrenimi	
Bildiği Yabancı Diller	İngilizce
Bilimsel Faaliyetleri	
İş Deneyimi	
Stajlar	
Projeler	
Çalıştığı Kurumlar	Atatürk Üniversitesi Bilgisayar Bilimleri Araştırma ve Uygulama Merkezi
İletişim	
E-Posta Adresi	gokhan@atauni.edu.tr
Tarih	22/07/2019