VERİ TABANINDA VERİ TEKİLLEŞTİRME: ATATÜRK ÜNİVERSİTESİ ÖĞRENCİ BİLGİ SİSTEMİ ÖRNEĞİ

Yakup BAYOĞLU

Yüksek Lisans Tezi
İşletme Anabilim Dalı
Doç. Dr. Abdulkadir ÖZDEMİR
2018

T.C.
ATATÜRK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
İŞLETME ANABİLİM DALI

Yakup BAYOĞLU

VERİ TABANINDA VERİ TEKİLLEŞTİRME: ATATÜRK ÜNİVERSİTESİ ÖĞRENCİ BİLGİ SİSTEMİ ÖRNEĞİ

YÜKSEK LİSANS TEZİ

TEZ YÖNETİCİSİ
Doç. Dr. Abdulkadir ÖZDEMİR

ERZURUM – 2018

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

21/09/2018

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE

BİLDİRİM

Atatürk Üniversitesi Lisansüstü Eğitim ve Öğretim Uygulama Esaslarının ilgili maddelerine göre hazırlamış olduğum "VERİ TABANINDA VERİ TEKİLLEŞTİRME: ATATÜRK ÜNİVERSİTESİ ÖĞRENCİ BİLGİ SİSTEMİ ÖRNEĞİ" adlı tezin tamamen kendi çalışmam olduğunu ve her alıntıya kaynak gösterdiğimi taahhüt eder, tezimin 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.

21.09.2018
Yakup BAYOĞLU

---

* **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

Doç. Dr. Abdulkadir ÖZDEMİR danışmanlığında, Yakup BAYOĞLU tarafından hazırlanan bu çalışma 20/09/2018 tarihinde aşağıda isimleri yazılı jüri tarafından. İşletme Anabilim Dalı'nda Yüksek Lisans Tezi olarak kabul edilmiştir.

Başkan : Prof. Dr. Abdullah NARALAN İmza:

Jüri Üyesi : Doç. Dr. Abdulkadir ÖZDEMİR İmza:

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

Prof. Dr. Mehmet TÖRENEK
Enstitü Müdürü

İÇİNDEKİLER

ÖZET

YÜKSEK LİSANS TEZİ

VERİ TABANINDA VERİ TEKİLLEŞTİRME: ATATÜRK ÜNİVERSİTESİ ÖĞRENCİ BİLGİ SİSTEMİ ÖRNEĞİ

Yakup BAYOĞLU

Tez Danışmanı: Doç. Dr. Abdulkadir ÖZDEMİR

2018, 113 sayfa

Jüri: Doç. Dr. Abdulkadir ÖZDEMİR
Prof. Dr. Abdullah NARALAN
Dr. Öğr. Üy. Serdar AYDIN

Veri kavramı hayatımızın her alanında kullanılmaktadır. Bilişim sistemleri söz konusu olduğunda; veri tabanı, veri güvenliği, veri tutarlılığı, veri kirliliği kavramlarında olduğu gibi yanına yeni sözcükler alarak karşımıza çıkmaktadır.

Atatürk Üniversitesi'nde kullanılan birçok sistem, çok fazla kişi tarafından kullanılmakta ve değişik şekillerde veriler girilmektedir. Girilen veriler aynı şeyi ifade etmesine rağmen kullanıcıların değişik bakış açılarından dolayı; bazen farkında olmadan, bazen de kasıtlı olarak farklı şekillerde girilebilmektedir. Bu farklılıklar sebebi ile tekrar tekrar eklenmiş olan veriler sistemin sağlıklı çalışmasını etkileyecek derecede veri kirliliğine sebep olabilmektedir. Sistemlerdeki bu tarz veri kirliliğinin önüne geçmenin çeşitli yöntemleri bulunmaktadır. Bu yöntemler kullanıcıların sistemdeki verilere müdahalesini kısıtlamak/engellemek ve yeni kayıt girilirken çeşitli kontroller yapıldıktan sonra eklemek şeklindedir.

Atatürk Üniversitesi oldukça büyük ve köklü bir geçmişe sahip olmasından dolayı Öğrenci Bilgi Sistemine (ÖBS) ait veri tabanındaki veriler çok eski tarihlere uzanmaktadır. Daha önceleri yukarıda bahsedilen tedbirler uygulanmadığından dolayı, sistemde eskiden kalma kirli veriler mevcuttur. Ders adlarındaki veri kirliliği, ÖBS de en çok karşılaşılan ve en çok soruna sebep olan kirliliktir. Bu çalışmamızda aynı isim ve krediye sahip derslerin tek kayıtta birleştirilmesi ve aslında aynı olup farklı şekilde yazılmış olan derslerin tespit edilerek bunların da tek bir kayıtta birleştirilmesi amaçlanmıştır. Bu çalışma kapsamında ders adlarında bazı düzeltmeler yapılmış ve akabinde aynı isimli dersler birleştirilmiştir. Dizge karşılaştırma algoritmalarından Damerau-Levenshtein algoritması kullanılarak benzer isimli dersler tespit edilmiş ve bunlardan da birleştirilmesi uygun görülen dersler birleştirilmiştir.

Anahtar Kelimeler: Veri tabanı, veri kirliliği, veri tekilleştirme, dizge karşılaştırma

ABSTRACT

MASTER'S THESIS

DATA DEDUPLICATION ON DATABASE: ATATÜRK UNIVERSITY STUDENT INFORMATION SYSTEM CASE

Yakup BAYOĞLU

Advisor: Assoc. Prof. Dr. Abdulkadir ÖZDEMİR

2018, Page: 113

Jury: Assoc. Prof. Dr. Abdulkadir ÖZDEMİR
Prof. Dr. Abdullah NARALAN
Assist. Prof. Dr. Serdar AYDIN

The concept of data is used in all areas of our lives. When it comes to information systems, the word data is used with another words alongside. Such as database, data safety, data integrity, data pollution etc.

The systems used in Atatürk University, are used by many people and data is entered in different ways. Although the input data indicate the same thing, due to the different approaches of the users; sometimes unwittingly, sometimes intentionally, it could be entered in different forms. Data which is repeatedly added due to these different approaches, could cause data corruption that will affect the healthy operation of the system. There are various methods of preventing such data pollution in systems. These methods are to restrict / prevent users from interfering with the data in the system and to make various checks while entering the new record.

Since Atatürk University has a very large and deep-rooted history, the data in the database of the Student Information System (SIS) goes back to very old dates. Since measures mentioned above had not been applied in earlier time, there are some old dirty data in the system. The data pollution in the course names is the most encountered pollution and causes the most problems in SIS. In this study, it is aimed to combine the courses with the same name and credits in a single record. It is also aimed to determine the courses that are actually the same but typed differently, and combine them in a single record. Within the scope of this study, some corrections have been made in the course names and then the courses with the same name were combined. Similarly named courses were determined by using Damerau-Levenshtein Algorithm which is one of string comparison algorithms. Similarly named courses which are agreed to be same ones, were combined in a single record.

Keywords: Database, data pollution, data deduplication, string comparison

KISALTMALAR DİZİNİ

ÖBS	: Öğrenci Bilgi Sistemi
AGNO	: Ağırlıklı Genel Not Ortalaması
VTYS	: Veri Tabanı Yönetim Sistemi
LD	: Levenshtein Distance
DLD	: Damerau-Levenshtein Distance
LCS	: Longest Common Subsequence
SP	: Stored Procedure

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

ÖNSÖZ

Bu çalışmanın amacı, Atatürk Üniversitesi Öğrenci Bilgi Sistemi veri tabanında yer alan ders bilgilerindeki kirliliğin, veri tekrarının ve tutarsızlıkların giderilmesi ve bu işlemlerin sonucunda; müfredatların kısmen de olsa temizlenmesi, öğrencilerin transkriptlerinde belirtilen toplam kredi ve ağırlıklı genel not ortalamalarının doğru bir şekilde hesaplanabilmesine olanak sağlanmasıdır.

Çalışmanın planlanıp uygulanmasında desteklerini esirgemeyen başta tez danışman hocam Doç. Dr. Abdulkadir ÖZDEMİR'e, Dr. Öğr. Üyesi Serdar AYDIN'a, Uzm. Metin MİKYAS liderliğindeki Öğrenci Bilgi Sistemi proje grubuna ve Öğrenci İşleri Daire Başkanı Metin SAYGILI'ya teşekkürü bir borç bilir ve çalışmanın konu ile ilgilenenlere faydalı olmasını dilerim.

Erzurum – 2018 **Yakup BAYOĞLU**

GİRİŞ

Veriye duyulan ihtiyaç, insanoğlunun tarihi kadar eskidir. Gittikçe artan bu ihtiyaç, hiçbir zaman bugün olduğu kadar şiddetli değildi; yarın da bugünden daha şiddetli olacaktır. Veriyi kaydetme aracı olarak ilkel zamanlarda taş tabletler kullanılırken; bu araçlar gitgide gelişmiş olup, günümüzde çoğunlukla veri tabanları kullanılmaktadır.

Veri tabanı, gereksiz verileri kontrol ederek ve merkezileştirerek, pek çok uygulamaya sunmak üzere düzenlenmiş veri yığını olarak tanımlanabilir. Veri tabanı yönetim sistemi (VTYS), bir kurumun verilerini merkezileştirmesine ve verilerini efektif bir şekilde kullanmasını sağlayan, uygulama programlarının depolanmış verilere erişimini ve yönetimini sağlayan yazılımdır. Günümüzde en gözde VTYS türü ilişkisel VTYS'dir. İlişkisel veri tabanında veriler, iki boyutlu tablolarda tutulur. Her bir tablonun benzersiz veri içeren birincil anahtar bir sütunu vardır. Tablolar arası ilişkiler, birincil anahtarlar vasıtasıyla kurulur.

Veri tabanında yer alan veriler, Structured Query Language (SQL) ile sorgulanmaktadır. SQL 1975 yılında IBM laboratuvarlarında geliştirilmiş olup tüm veri tabanı sorgulama dillerinde temel olarak alınmıştır. SQL dilinin yeteneklerinin sınırlı olması sebebi ile zamanla çeşitli firmalar SQL'e ekleme ve iyileştirmeler yaparak kendi sorgulama dillerini geliştirmişlerdir. Microsoft firması T-SQL'i, Oracle firması da PL/SQL'i geliştirmiş ve yazılımcılara sunmuştur.

Dizge karşılaştırma algoritmaları, iki metinsel ifadenin birbirlerine benzerliğini ölçmektedir. (http://alias-i.com/lingpipe/demos/tutorial/stringCompare/read-me.html) Bazı algoritmalar, bir ifadeyi diğerine dönüştürmek için gereken karakter bazında ekleme, silme ve değiştirme gibi işlemlerinin sayısını hesaplarken; bazıları da iki ifadenin birbirine benzerliğini yüzdesel olarak hesaplar.

Atatürk Üniversitesi Öğrenci Bilgi Sisteminde (ÖBS) aynı isim ve krediye sahip dersler bulunmaktadır. Benzer şekilde, aslında aynı olup yazılışı hatalı olduğu için farklı görünen dersler de mevcuttur. Veri kirliği oluşturan bu durum, birçok soruna sebep olmaktadır. Yaşanan temel sorunlar:

- Müfredatlarda fazla ders bulunması

- Derslerin birden fazla açılması sebebiyle, öğrenci işleri personelinin ve dersin öğretim üyesinin iş yükünün artması
- Dersin gruplara bölünmesi sebebi ile harf notu hesaplamasında tutarsızlıklar oluşması
- Öğrencilerin toplam kredi ve ağırlıklı genel not ortalamalarının yanlış hesaplanması
- Gerçekte bütün derslerini başarmış bir öğrencinin alttan dersi varmış gibi görünmesi

Şeklinde listelenebilir.

Bu çalışmanın amacı, yukarıda maddeler halinde belirtilen sorunları çözmek için, ÖBS'de yer alan aynı isim ve krediye sahip ders adlarını tek bir kayıt altında birleştirmektir. Aslında aynı olup yazılışı farklı olan dersleri tespit ederek, bu dersleri de tek bir kayıtta birleştirmek, yine bu çalışmanın amaçlarındandır. Bu şekildeki dersleri tespit edebilmek için, metinsel ifadelerin birbirlerine olan benzerliğini tespit eden dizge karşılaştırma algoritmalarından Damerau-Levenshtein Algoritması kullanılmıştır.

Birinci bölümde veri tabanı kavramı, veri tabanı yönetim sistemleri ve ilişkisel veri tabanlarından bahsedilmiştir. Veri bütünlüğünü korumak için veri tabanının ilişkisel olmasının önemine değinilmiştir.

İkinci bölümde sık kullanılan çeşitli dizge karşılaştırma algoritmaları anlatılmış ve örneklerle izah edilmiştir. Yapısı ve mantığı itibari ile bu algoritmaların hangi durumlarda kullanmaya daha elverişli olduğundan bahsedilmiştir. Bu çalışmada Damerau-Levenshtein algoritmasının kullanılması tercih edilmiştir.

Üçüncü bölümde Atatürk Üniversitesi ve Öğrenci Bilgi Sisteminden bahsedilmiş; ders birleştirme uygulamasına neden ihtiyaç duyulduğu, mevcut sorunlara dair örnekler verilerek anlatılmıştır.

Dördüncü bölümde uygulamadaki saklı yordamlar ve fonksiyonlar tanıtılmıştır. Uygulamayı hayata geçirmeden önce yapılan ön hazırlıklar ve çeşitli optimizasyonlardan bahsedilmiştir. Uygulamanın adım adım hayata geçirilmesi anlatılmıştır.

Sonuç kısmında uygulama sonucunda düzeltilmiş olan verilere dair istatistiki bilgiler verilmiş ve bu tür sorunların tekrar yaşanmaması adına ne tür tedbirler alınması gerektiğinden bahsedilmiştir. Bu çalışmanın daha ileriye götürülmesi adına Öğrenci Bilgi Sistemi proje ekibine ve Öğrenci İşleri Daire Başkanlığı personeline bazı önerilerde bulunulmuştur.

BİRİNCİ BÖLÜM

VERİ TABANI

1.1. VERİ TABANI KAVRAMI

Veri tabanı ve veri tabanı teknolojileri, bilgisayar kullanımının artmasında büyük öneme sahiptir. Veri tabanlarının, bilgisayar kullanımının yaygın olduğu; işletme, elektronik ticaret, sosyal medya, mühendislik, tıp, genetik, hukuk, eğitim gibi bilim dallarında kritik rol aldığı söylenebilir. Veri tabanı kavramı çok geniş bir alanda kullanılır ve en genel tanımıyla birbiri ile ilişki veriler topluluğu olarak tanımlanabilir. Bu tanımdaki veriden kasıt, kaydedilebilir ve net gerçeklerdir. Örneğin, cep telefonu rehberinde kaydedilen tüm isim, soyisim, numara, fotoğraf, e-posta adresi gibi veriler, birbirleri ile ilişkili veriler olup, bir veri tabanıdır. (Elmasri & Navathe, 2015)

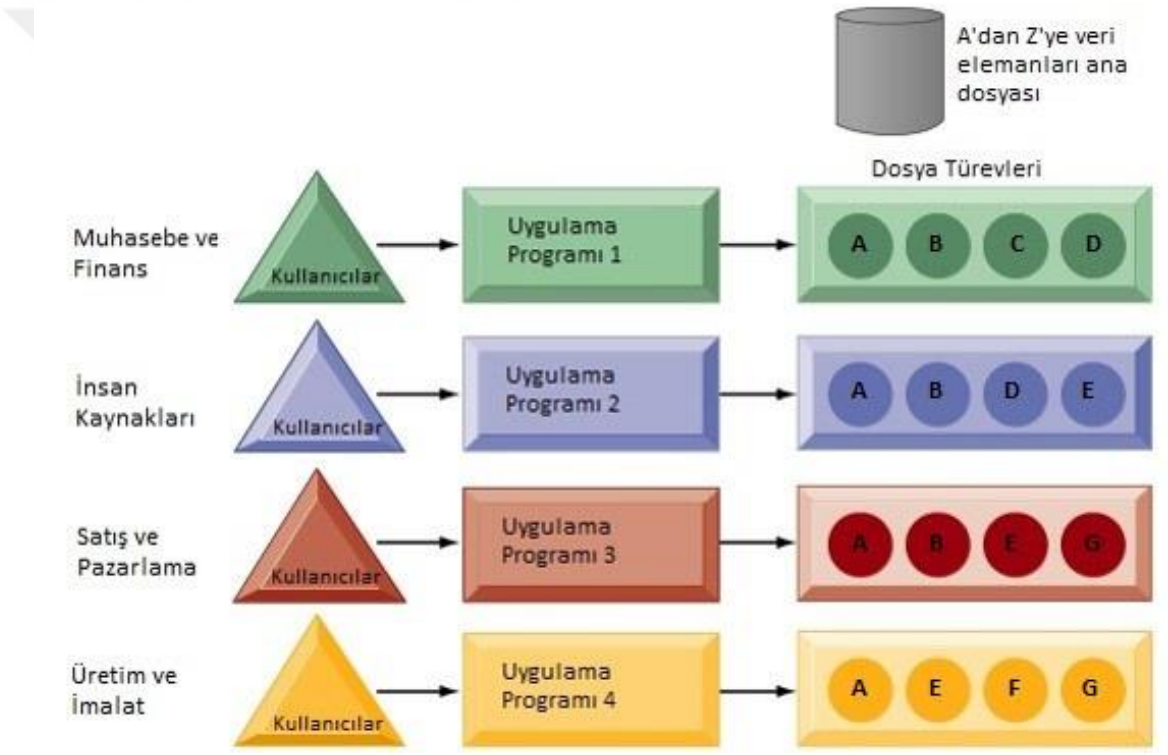
Veri tabanı, verilerin, birbirinden bağımsız olan farklı uygulamalar tarafından kullanılması amacıyla, gereksiz tekrarlardan kaçınılarak; gizliliği, güvenliği ve tutarlılığı sağlanarak; özel tekniklerle depolanması, güncellenmesi ve erişilmesine imkân veren bir yazılım sistemidir. (Batuk, 1997)

Veri tabanı kavramı, bilgi işlem dünyasında uzun tecrübe ve aşamalardan sonra ulaşılmış bir kavramdır ve klasik dosya yönetimine bir alternatif olarak, geniş kapasiteli, hızlı, büyük veri yığınlarını taşıyıp saklayabilen donanımlar ile bunlara uygun, kapsamlı, ağ ortamının isteklerine cevap veren yazılımların geliştirilmesinin sonucu ortaya çıkmıştır. Klasik bir dosyalama sisteminde en önemli özellik uygulamaya bağımlı olmaktır; yani bir dosya hangi yazılım tarafından oluşturulmuşsa o yazılıma bağımlı olarak dosyaya erişilebilir; oysa veri tabanı yönetiminde prensip olarak veri-uygulama bağımsızlığı vardır; yani bir kez oluşturulmuş verilere teorik olarak her tür programlama dili ya da uygulama programı ile erişme imkânı vardır. (Karaş, Baz, & Geymen, 2006)

Bilgi sistemlerinin temelinde verinin toplanması, depolanması, gösterilmesi ve bu süreçlerin performansı, tutarlılığı ve güvenliği yer alır. Sonraki tüm süreçler veri ile etkileşim içerisinde olacağından dolayı, verinin ne şekilde depolanacağı buradaki en önemli kısımdır. Başlangıçta veriyi depolamada efektif bir metot uygulanırsa, daha

sonraki süreçlerde kolaylık sağlanacaktır. Verinin depolanması, bilgi sistemleri için hayati önem arz ettiğinden dolayı, bu sürecin iyi yönetilebilmesi amacıyla veri tabanı sistemleri ortaya çıkmıştır. (Ünal, 2011)

Veri tabanı sistemleri henüz geliştirilmemişken, verileri depolamak için klasik dosya sistemleri kullanılırdı. Böyle bir sistemde her bir uygulama, kullanıcıların veriye erişmesi, işlemesi gibi ihtiyaçlarını karşılayan ayrı uygulama programları yer alırdı. Kullanıcıların yeni ihtiyaçlarına göre, sistem programcıları tarafından yeni uygulama programları yazılır ve sisteme eklenirdi. Böylece sistemin işlerliği için gereken dosya sayısı ve uygulama programları çoğalır ve ölçek büyüdükçe de sistemi yönetmek zorlaşırdı.



Şekil 1.1. Geleneksel Dosya Yapısı Kullanılmış Bir Bilgi Sistemi Örneği (Laudon)

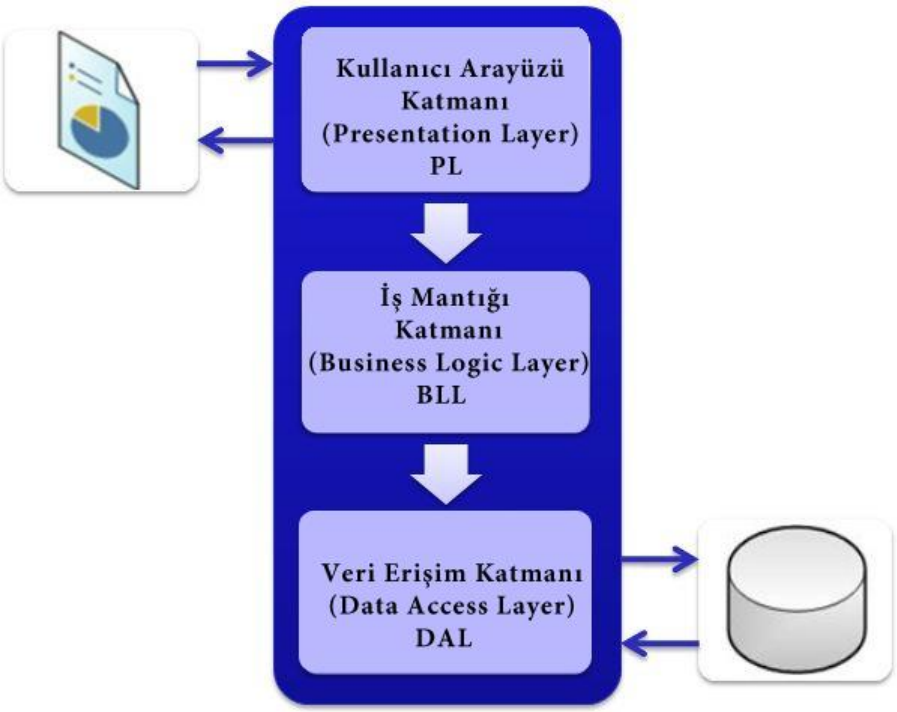
Şekil 1.1 de gösterilen örnek bir bilgi sisteminde 4 ayrı modül yer almaktadır. Her modülün diğerlerinden bağımsız veri dosyaları ve uygulama programları yer almaktadır. Her modül ihtiyaç duyduğu bilgileri kendi dosyalarında tutmaktadır ve A konulu verileri içeren dosyalar her uygulamada ayrı ayrı yer almaktadır. B ve E konulu dosyalar üç modül için ayrı ayrı yer almaktadır. Bu sistemde klasik dosyalama sistemi yerine, veri tabanı kullanılmış olsa idi, her modül için ayrı uygulama programlarına ihtiyaç

duyulmayacaktı. Benzer şekilde, aynı verilerin farklı modüllerde ayrıca tutulmasına gerek kalmayacaktı. Bu sistemde A konulu verilerde bir güncelleme ihtiyacı olduğunda, bu işlem her modül için ayrı ayrı yapılması gerekmektedir ve güncelleme işlemi dikkatli bir şekilde yapılmazsa veri tutarsızlığına sebep olacaktır. Laudon, klasik dosyalama sistemlerinin dezavantajlarını şu şekilde listelemektedir: (Laudon & Laudon, 2012)

- Veri fazlalığı ve tutarsızlık
- Program-veri bağımlılığı
- Yetersiz esneklik
- Zayıf güvenlik
- Veri paylaşımı ve erişiminde yetersizlik

Yukarıda belirtilen dezavantajlar, veri tabanı sistemlerinin geliştirilmesine sebep olmuştur.

Bilgi sistemleri geliştirilirken artık çok katmanlı yazılım mimarisi kullanılmaktadır ve genelde üç katmandan oluşur. Veri tabanları, en alt tabakada yani veri katmanında yer alır.



(http://www.webyazilimdilleri.com/web/yazilim/dilleri/makaleler/5282/C%23NET-TE-3-KATMANLI-MIMARI/default.aspx)

Şekil 1.2. Üç Katmanlı Mimari

1.2. VERİ TABANI YÖNETİM SİSTEMLERİ

Veri tabanı, anlamlı bir grup veriyi bir arada barındıran yapı olarak tanımlanmıştı. Veri tabanı yönetim sistemleri (VTYS) ise, bünyesinde bir ya da daha fazla veri tabanı barındıran ve bu veri taban(lar)ında veri ekleme, silme, güncelleme ve listeleme işlemleri ile çalışmak isteyen kullanıcıları yöneten gelişmiş sistemlerdir.(Çamoğlu, 2009)

Bir veri tabanı yönetim sistemi, hem birbiri ile ilişkili veriler topluluğunu, hem de bu verilere ulaşmak amacı ile kullanılan programları içerir. VTYS'nin esas işlevi, veri tabanı verilerinin saklanması, işletimi ve yönetimine yönelik kullanışlı bir ortam sağlamaktır denilebilir. VTYS, temeli IBM tarafından atılmış ve standart bir dil olan Structured Query Language (SQL) ile uygulamalara ve kullanıcılara hizmet verir.

Veri tabanı yönetim sistemlerinin temel bileşenleri şunlardır:

- Veri erişimini ve işlenmesini sağlayan arabirim: Verileri görme, silme, değiştirme, ekleme, düzenleme işlemlerini sağlayan arabirimdir.

- Uygulama ve veri tabanı geliştirme olanağı sağlayan arabirim: Uygulama geliştiriciler için uygun ortam sağlar.
- Veri sözlüğü: Veri yapısını gösterir ve sistem kataloğunda bulunur.
- Veri modeli: Veri tabanı tasarımı için geliştirilmiş bir arabirimdir.
- Metadata (Üstveri): Veri hakkındaki bilgidir. Örneğin tablodaki sütunların veri tipi, boyutu gibi bilgiler. (https://en.wikipedia.org/wiki/Metadata)
- Sorgulama Dili: Sorgulama için en yaygın olarak, SQL adı verilen yapısal sorgulama dili kullanılır. Farklı veri tabanı sistemleri için farklılıklar gösterir, ancak SQL temelde standart bir dildir.
- Güvenlik Sistemi: Kullanıcıların yetki ve izinlerinin düzenlendiği ortamdır.
- Raporlama: Sorgu sonuçlarının rapor halinde yönetimini sağlayan araçlardır.

1.2.1. Veri Tabanı Yönetim Sisteminin İşlevleri

Veri Tabanı Yönetim Sistemi, kullanıcıların, verilerini depolamasını, silmesini, güncellemesini sağlar. Kayıtlı verilere değişik şekillerde erişim imkânı verir.

Veri tabanları genelde birçok kullanıcının erişimine açık olur. VTYS birden fazla kullanıcının veri tabanına aynı anda erişmesine ve işlem yapabilmesine olanak sağlar. Birden fazla kullanıcının aynı anda işlem yapması durumunda ihtiyaç duyulabilecek olan kilitleme mekanizmasına sahiptir.

Farklı tablolarda yer alan veriler, birbirleri ile ilişki halinde olabilir. Bu durumda tablolar arasında ilişkilerin tanımlanmasına ve bu ilişkilere bağlı olarak ekleme, silme ve güncelleme işlemlerine çeşitli kısıtlar konulabilmesine imkân sağlar. Bu sayede veri bütünlüğü sağlanmış olur.

İşlem gruplarını yönetebilir. Birbiri ile ilişkili bir grup işlemin bir blok halinde ya hep ya hiç mantığıyla yerine getirilebilmesine imkân sağlar. Grup içindeki işlemlerden biri gerçekleşemez ise bütünlük bozulacağı için rollback yapılır; yani o ana kadar yapılan işlemler geri alınır. Bu yetenek veri bütünlüğü açısından önemlidir.

Veri Tabanı Yönetim Sistemi, verilere erişimde yetkilendirme yapar. Farklı kullanıcılara, sadece ihtiyaç duyacağı kadar yetki vererek verilerin korunmasına imkân

sağlar. Bu korumayı, farklı kullanıcılara; farklı işlem(ler) yetkisi; hatta farklı şemalara erişim yetkisi vermek sureti ile gerçekleştirir.

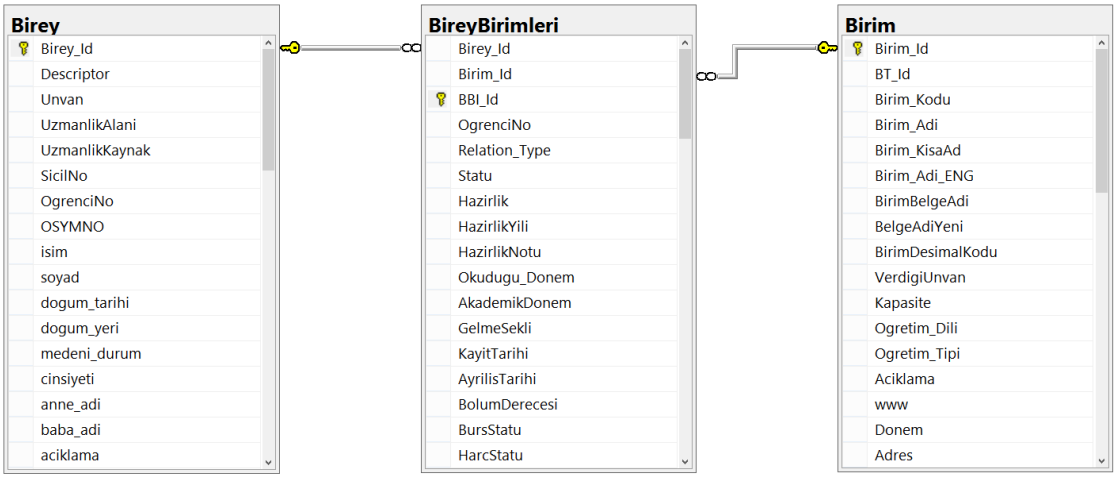
Veri tabanı Yönetim Sistemi, veri tabanının yedeklenebilmesine ve ihtiyaç halinde bu yedeklerden geri yüklenebilmesine imkân sağlar. Böylece sistemde meydana gelebilecek herhangi bir aksilik durumunda mevcut yedeklerden birine geri dönülebilir. Yedek alma işleminin belirli periyotlarla otomatik olarak yapılmasına da imkân sağlar.

1.3. İLİŞKİSEL VERİ TABANI YÖNETİM SİSTEMLERİ

İlişkisel veri tabanının temelleri 1970'li yıllarda E. F. Codd tarafından IBM laboratuvarlarında yapılan çalışmalar ile atılmıştır. Devamında yapılan çalışmalar neticesinde, 1981 yılında Codd, Turing ödülüne layık görülmüştür. 1983 yılında SQL (Structured Query Language) standartları tanımlanmış; 1987 yılında önce ISO, akabinde de ANSI tarafından standart olarak tanınmıştır. 1992 yılında yayınlanan ANSI SQL-92, birçok VTYS'nin kullandığı temel dil ve ifadeler için referans olmuştur. (Gözüdeli, 2010)

İlişkisel veri tabanı, günümüzde en çok tercih edilen veri tabanı sistemlerindendir. Satır ve sütunlara sahip tablolardan oluşur ve bu tablolar birbirleri ile ilişki halindedir. Yani bir veri tabanının ilişkisel olması için, veri tabanında birden fazla tablo yer almalı ve bunlar bir şekilde birbirleri ile ilişkilendirilebilmelidir. (Öztürk & Atmaca, 2017)

Tablolar verileri ihtiyaca uygun şekilde belli bir yapıda tutarlar. Veriler, farklı tablolarda tutulacak şekilde kendi içinde anlamlı bir şekilde gruplanıp bölünür. Farklı tablolardaki veriler, birbirleri ile ilişkilendirilir. Örnek bir öğrenci bilgi sisteminde öğrencilerin adı, soyadı, anne adı, baba adı, doğum tarihi, doğum yeri, TC kimlik numarası vs. gibi özlük bilgileri Birey isimli bir tabloda tutuluyor olsun. Birimlerin adı, kodu, kısa adı, İngilizce adı, düzeyi, öğretim tipi, öğretim dili, adresi vs. gibi bilgileri de Birim isimli bir tabloda tutuluyor olsun. Hangi öğrencinin hangi birimde kaydı olduğu, kayıt tarihi, öğrenci numarası, müfredat yılı, okuduğu dönem sayısı, sınıfı, mezuniyet tarihi, diploma numarası gibi bilgileri de BireyBirimleri isimli üçüncü bir tabloda tutuluyor olması ilişkisel veri tabanına güzel bir örnektir.



Şekil 1.3. Birbiri ile İlişkili Tablolar Örneği

Şekil 1.3'te yer alan BireyBirimleri tablosundaki Birey_Id ve Birim_Id sütunları sırası ile Birey tablosundaki Birey_Id sütunu ve Birimler tablosundaki Birim_Id sütunu ile ilişkilidirler.

1.4. T-SQL

SQL 1983 yılında, veri tabanında tablo oluşturmak, tabloyu değiştirmek; tabloya veri eklemek, silmek, güncellemek gibi temel işlevlerin geçekleştirilmesini sağlamak amacı ile IBM laboratuvarlarında geliştirildi. (Gözüdeli, 2010)

SQL dilinin yetenekleri sınırlı olduğu için çeşitli firmalar SQL üzerinde iyileştirmeler ve eklemeler yapmıştır. Oracle firması, SQL üzerine yaptığı iyileştirmeleri standartlaştırmış ve PL/SQL adı ile piyasaya sunmuştur. Microsoft firması da kendi platformu için SQL üzerine yaptığı iyileştirme ve eklemeleri T-SQL adı ile geliştiricilere sunmuştur. Transact-SQL'in kısaltması olan T-SQL, günümüz veri tabanı yönetim ihtiyaçlarının tamamını karşılayabilecek yeterliliktedir. T-SQL verileri manipüle etme, değişken kullanma ve hata ayıklama gibi birçok programlama yeteneğine sahiptir. Fakat programlama dili değil, gelişmiş bir sorgu dilidir. T-SQL'in yeteneklerinden istifade edebilmek için Microsoft ürünü olan SQL Server Management Studio'nun (SSMS) kullanılması gereklidir. (https://www.teknologweb.com/t-sql-nedir-transact-sql)

1.4.1. Saklı Yordam (Stored Procedure)

Saklı yordamlar, daha sonra tekrar eden kullanım için, sunucudaki Transact-SQL kod bloklarını kapsülleyen veri tabanı nesneleridir. İşlemlerin nasıl ve hangi sırada yapılması gerektiğini belirleyen algoritmaların T-SQL ile kodlanmasını sağlar. Saklı yordamlar, diğer programlama dillerindeki altprogramların T-SQL eşdeğerleridir. Özel veri tabanı uygulama geliştiricileri, saklı yordamları oluştururken tüm programlama yapılarını kullanabilir: (Sunderic, 2002)

- Değişkenler
- Veri tipleri
- Giriş / çıkış parametreleri
- Dönüş değerleri
- Koşullu yürütme
- Döngüler
- Yorumlar

Saklı yordam oluşturulurken geçilen aşamalar şu şekildedir:

- Ayrıştırma (Parsing): Bu aşamada sentaks kontrolü yapılır. Yani yazılan ifadelerin SQL yazım standartları ve kurallarına uyup uymadığı, komut ifadelerinin doğru yazılıp yazılmadığı vs. kontrol edilir. Veri tabanında bulunmayan bir tablodan select ile veri çekiliyor olması bu aşamada hata oluşturmaz; fakat "select" ifadesi "slect" şeklinde yazılırsa hata oluşur. Bu aşamada sorgu ağacı ya da sıra ağacı denen yapı ortaya çıkar.
- Derleme (Compiling): Bir önceki aşama olan ayrıştırma aşamasında oluşturulan sorgu ağacından çalışma planı çıkarılır, hak ve yetkiler kontrol edilerek güvenlik denetlenir. Çalışma planı, hangi aşamada hangi kontrollerin, kısıtların veya indekslerin kullanılacağı gibi tanımları içerir.
- Çalıştırma (Executing): Derleme aşamasında oluşturulmuş olan çalışma planı üzerinden işlemler gerçekleştirilir.

Saklı yordamlar ilk defa çalıştırılırken yukarıdaki aşamaların hepsinden geçilir; daha sonraki çalıştırılmalarında ise sadece sonuncu aşama olan çalıştırma aşaması gerçekleşir. Çünkü derleme aşamasından sonra oluşan çalışma planı hafızada saklanır

ve sonraki kullanımlarda yeniden oluşturulmasına gerek kalmaz. Bu yüzden saklı yordamlar aynı içeriğe sahip kod bloğuna göre daha hızlı çalışırlar. Bu durum saklı yordam kullanmanın en önemli avantajlarındandır.

Saklı yordam şu durumlarda çok kullanışlıdır: (Elmasri & Navathe, 2015)

- Bir veri tabanı programına farklı uygulamalar tarafından ihtiyaç duyulduğunda, program saklı yordam olarak sunucuda saklanıp herhangi bir uygulama tarafından çağrılabilir. Bu durum yazılımı daha modüler hale getirir.
- Saklı yordam kullanmak bazı durumlarda sunucu ve istemci arasındaki veri transferini ve iletişim maliyetini azaltır.
- Saklı yordam, daha karmaşık bir şekilde türetilmiş veriler elde etmeye izin verdiği için görüntülerin sağladığı modelleme gücünü artırır. Ayrıca tablolar üzerinde, tablo tanımında yapılan kısıtlamalar ve tetikleyicilerden daha karmaşık kontroller sağlar.

Saklı yordamlarda iki türlü parametre kullanılmaktadır:

- Giriş Parametresi: Saklı yordamın çalışırken ihtiyaç duyacağı veriler, giriş parametreleri vasıtasıyla yordama gönderilir.
- Çıkış Parametresi: Saklı yordama gönderilen çıkış parametresine, yordam içerisinde değer atanarak çağrıldığı yere geri döndürülür. Saklı yordam tanımlanırken, veri tipinden sonra ‘out’ ifadesi yazılarak çıkış parametresi olduğu belirtilir.

1.4.2. Fonksiyon

Fonksiyon kavramı programcılığın temel kavramlarındandır. Sıkça kullanılacak kod bloğu bir fonksiyon içine yazılır ve ihtiyaç duyulduğu her yerde ilgili kod bloğunun yeniden yazılması yerine fonksiyon çağırılır. Bu şekilde kod karmaşası azaltılır, programın okunurluğu ve anlaşılırlığı artar. Ayrıca saklı yordamlarda olduğu gibi fonksiyon tanımında değişiklik yapıldığı zaman, bu değişiklik fonksiyonun kullanılmış olduğu her yere yansır. Böylece programın modülerliği artar.

Microsoft SQL Server’ın kendi içinde tanımlı hazır fonksiyonları vardır. Matematiksel fonksiyonlar, metinsel işlemlere yönelik fonksiyonlar, tarih fonksiyonları,

dönüştürme fonksiyonları gibi çeşitli amaçlara yönelik çok zengin bir fonksiyon kütüphanesi mevcuttur. Bunlardan başka programcılar da kendi ihtiyaçlarına yönelik fonksiyon tanımlayabilirler.

Fonksiyonlar, görüntü ve saklı yordam kullanmanın avantajlarını kendi bünyesinde toplamıştır. Fonksiyon içerisinde, saklı yordamda olduğu gibi çeşitli T-SQL sorguları, kontrollü yürütme, döngüler vs. kullanılabilir, parametre alabilir. Fonksiyonun görüntüye benzer yönü ise select sorgularına dâhil edilebilmeleridir. Bu imkân yazılımcılara ciddi kolaylık sağlar.

Kullanıcı tanımlı fonksiyonlar iki gruba ayrılır:

- Skaler Fonksiyon: Geriye, sonuç olarak sadece tek bir değer döndürür.
- Tablo Fonksiyon: Geriye, sonuç olarak bir tablo döndürür.

1.4.3. Transaction Yapısı

Bazı durumlarda birden fazla işlem bir bütünün parçasıdır. Bu işlemlerden herhangi biri gerçekleşmediği durumda kalan diğer işlemler anlamsız kalabilir. Bu şekildeki işlem grubunun, tek bir işlem olarak ele alınması gerekebilir. Parçalanamaz işlemlerin ifade ettiği bu tek işleme transaction denir. Kısaca, daha küçük parçalara ayrılamayan işlem bloğudur.

Transaction bloğu “Begin Transaction” ifadesi ile başlar. Böylece yapılacak işlemlerin bütünlük arz ettiği, her an tamamının geçersiz sayılabileceği ilan edilmiş olur. Blok içerisindeki her bir işlemden sonra, işlemin başarılı bir şekilde gerçekleşip gerçekleşmediği kontrol edilir. İşlem başarılı ise sıradaki işleme geçilir; başarısız ise “Rollback Transaction” komutu ile o ana kadar yapılan işlemler geri alınır. Tüm işlemler başarılı bir şekilde tamamlandığında “Commit Transaction” komutu ile yapılan işlemler sabitlenir. (Gözüdeli, 2010)

Bir öğrenci bilgi sisteminde, öğrenciler mezun edilirken sırası ile aşağıdaki işlemler yapılır:

- Öğrencilik bilgilerinin yer aldığı tabloda statüsü mezun olarak değiştirilir.
- Mezuniyet işlemi öğrencinin arşiv bilgilerinin tutulduğu tabloya eklenir.
- Diploma bilgilerinin yer aldığı tabloda bu öğrenci için bir kayıt oluşturulur.

- Üretilen diploma numarası, öğrencilik bilgilerinin yer aldığı tabloda diplomaNo alanına yazılır.

Yukarıdaki işlemlerin tamamı bir bütün olduğu için bu işlemler transaction bloğu içerisinde yapılmalıdır. Transaction kullanılmadığı durumda; ilk iki işlem gerçekleşip, elektrik kesintisi vs. gibi bir sebepten dolayı 3. ve 4. işlemlerin gerçekleşmediği bir senaryoda; öğrenci mezun görünecektir ama diploma tablosunda bu öğrenci için bir kayıt bulunmayacaktır. Bu durum veri tutarsızlığı oluşturacaktır.

Uygulama programcılarının, eşzamanlı yürütme, kısmi yürütme, sistem çökmesi gibi tehditlerden dolayı verilerde oluşacak olan hasarı, tutarsızlığı önleme konusunda sorumlulukları, transaction kullanımını gerektirmektedir. (Liu & Özsu, 2009)

1.4.4. Tetikleyici (Trigger)

Tetikleyiciler, SQL Server'da saklı yordam, fonksiyon gibi bir bileşendir. SQL Server'da gerçekleşen bir işleme bağlı olarak başka bir işlemin de otomatik olarak yapılması istendiği durumda tetikleyici kullanılır. Tetikleyici hangi tablonun üzerinde tanımlandı ise; o tabloya veri eklendiğinde, silindiğinde, güncellendiğinde ilgili tetikleyici çalışır.

(http://www.bilisimlife.net/Yazi/148-SQL-Serverda-Trigger-Kullanimi.html)

Temel olarak, tetikleyiciler iki ana tipe ayrılır.

1.4.4.1. Ardı Sıra Tetikleyici (For Triggers)

Ardı sıra tetikleyiciler, bir tablo üzerinde ekleme, güncelleme veya silme işlemlerinden sonra çalışır. Ardı sıra tetikleyiciler 3 tipe ayrılır:

- Ekleme Sonrası Tetikleyici (After Insert Trigger)
- Güncelleme Sonrası Tetikleyici (After Update Trigger)
- Silme Sonrası Tetikleyicidirler (After Delete Trigger)

Örneğin bir tablodaki verilerin değişimini takip edebilmek amacı ile bir güncelleme sonrası tetikleyici tanımlanabilir. Tablodaki verilerin güncellenmesi halinde, ilgili satırların güncellenmeden önceki hali, başka bir log tablosuna, işlem

yapılan tarih ile beraber kaydedilebilir. Böylece tablodaki verilerin hangi tarihte, ne içerikte olduğu bilgisi arşivlenmiş olur.

1.4.4.2. Yerine Tetikleyici (Instead of Trigger)

Yerine tetikleyici, tablolar üzerinde yapılan işlemlerde bir önleyici ya da kontrol edici mekanizma olarak kullanılmaktadır. Tetikleyici içerisinde yazılan kontrollere göre işlem gerçekleştirilir ya da gerçekleştirilmez. Yerine tetikleyiciler üç sınıfa ayrılır:

- Ekleme Yerine Tetikleyici (Instead of Insert Trigger)
- Silme Yerine Tetikleyici (Instead of Delete Trigger)
- Güncelleme Yerine Tetikleyici (Instead of Update Trigger)

Bir tabloya silme işlemi için yerine tetikleyici tanımlandığında, silinmek istenen satırlar, silinmeden önce tetikleyici içerisindeki kontrollere sokulur. İlgili kontrollerden geçerse commit edilir, yani silme işlemi gerçekleştirilir; eğer kontrollerden geçemezse rollback yapılır; yani silme işlemi gerçekleşmez.

İKİNCİ BÖLÜM

DİZGE KARŞILAŞTIRMA ALGORİTMALARI

Bilgisayar sistemlerinde sayısal veriler eşitlik, büyüklük-küçüklük şeklinde karşılaştırılabilirken, metinsel ifadeler sadece eşitlik olarak karşılaştırılabilmektedir. Dizge karşılaştırma algoritmaları, birbirinden farklı iki veya daha fazla metinsel ifadenin birbirine benzerliğini ölçen algoritmalardır. Bazı algoritmalar sonuç olarak bir tamsayı hesaplar. Bu sayı, birinci metinsel ifadeyi diğerine dönüştürmek için karakter bazında ekleme, çıkarma, değiştirme gibi işlemlerden kaç tane yapılması gerektiğini belirtir. Bazı algoritmalar da 0 ile 100 arasında bir benzerlik yüzdesi hesaplar. Örneğin sadece ilk harfleri birbirinden farklı olan "bitap" kelimesi ile "kitap" kelimesi ele alındığında ilk kelimenin 'b' harfi, 'k' harfi ile değiştirilmesi durumunda "bitap" kelimesi "kitap" kelimesine dönüşecektir. Bu dönüşümün gerçekleşmesi için sadece bir adet değiştirme işlemi yapılması gerektiğinden dolayı, tamsayı sonuç üreten algoritmalar 1 sonucunu döndürecektir. Yüzde veren algoritmalar ise "bu kelimeler birbirine %80 benzemektedir" gibi bir sonuç dönecektir.

Atatürk Üniversitesi Öğrenci Bilgi Sisteminde yer alan dersler içerisinde aynı isimli olanları tespit etmek kolay bir işlemdir. Basit SQL sorguları ile tespit edilebilir; fakat "Makro İktisat" ve "Makro İkdisat" gibi benzer isimli dersleri tespit etmek, basit SQL sorguları ile mümkün değildir. Bundan dolayı uygulamada aslında aynı olup fakat yazılışı aynı olmayan dersleri tespit edebilmek için dizge karşılaştırma algoritması kullanmaya ihtiyaç vardır. Bu bölümde çeşitli dizge karşılaştırma algoritmaları anlatılmıştır.

2.1. LEVENSTEIN DISTANCE

Levenshtein Mesafesi 1965 yılında Rus bilim adamı Vladimir Levenshtein tarafından geliştirilmiştir. Algoritma aynı zamanda "Düzenleme mesafesi" (Edit Distance) ismi ile de anılmaktadır. (Augsten & Böhlen, 2013)

Levenshtein Mesafesi (LD) iki metinsel ifadenin birbirine benzerliğinin bir ölçüsüdür. LD algoritmasında kaynak metnin hedef metne dönüştürülebilmesi için

gerekli olan ekleme, yer değiştirme ve silme işlemlerinin sayısı Distance (mesafe)'dir. (Levenshtein, 1966)

Levenshtein Mesafesi ne kadar büyük olursa iki metinsel ifade birbirinden o kadar farklıdır. Levenshtein Mesafesi algoritması kullanım alanları aşağıda sıralanmıştır:

- Yazım hatalarını denetlemede
- Konuşma tanımada
- DNA analizlerinde
- İntihal tespitinde

Eğer *s* metinsel verisi "test" ise ve *t* metinsel ifadesi "test" ise LD(s,t)=0 olur, çünkü iki metin birbirine eşittir.

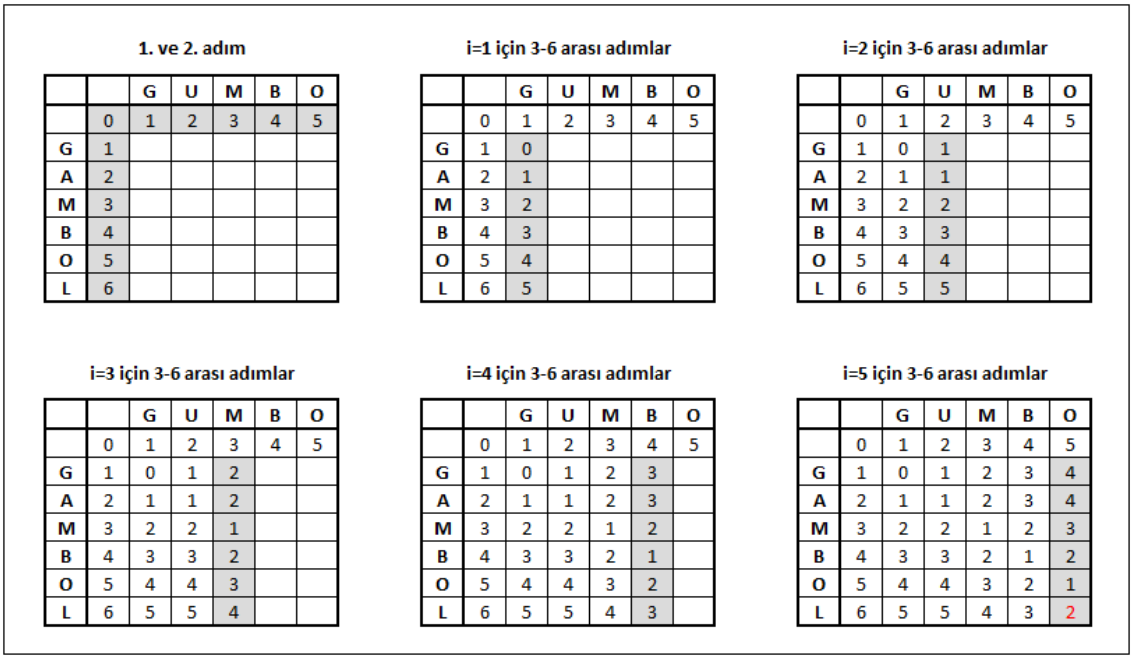
Eğer *s* metinsel verisi "test" ise ve *t* metinsel ifadesi "tent" ise LD(s,t)=1 dir. Burada yapılması gereken işlem sayısı bir olup, "n" karakterinin "s" karakteri ile değiştirilmesidir.

Tablo 2.1. Levenshtein Mesafesi Algoritması Hesaplama Adımları

Adım	**Açıklama**
1	n: s değişkeninin uzunluğudur m: t değişkeninin uzunluğudur Eğer n = 0, m değişkeni geri dönülür. Eğer m = 0 ise n değişkeni geri dönülür. 0..m ve 0..n sütun ihtiva eden bir matris kurulur.
2	İlk satırda 0 dan n'ye kadar olan sayılar yazılır. İlk sutünda 0 dan m'ye kadar olan sayılar yazılır.
3	i; 1 den n'ye kadar s değişkeninin her bir karakteri için 4. adım uygulanır.
4	j; 1 den m'ye kadar t değişkeninin her bir karakteri s[i] ile karşılaştırılır.

5	Eğer s[i] t[j] birbirine eşit ise, maliyet 0 dır. Eğer s[i] t[j] birbirine eşit değil ise, maliyet 1dir.
6	Matrisin d[i,j] hücresinin değeri: a. Üstteki hücrenin değerinin 1 fazlası: d[i-1,j] + 1. b. Soldaki hücrenin değerinin 1 fazlası: d[i,j-1] + 1. c. Sol üst çaprazda yer alan hücrenin değerinin, o hücredeki maliyet değeri miktarınca fazlası: d[i-1,j-1] + maliyet. a, b, ve c maddelerinde elde edilen değerlerin en küçüğüne eşitlenir.
7	3,4,5,6 adımlarının iterasyonu ile Levenshtein Distance derecesi d[n,m] hücresinde hesaplanır.

Levenshtein Mesafesi algoritmasının çalışma prensibine bir örnek olarak GAMBOL kelimesinin GUMBO kelimesi ile benzerliği tespit edilmesi Şekil 2.1'de adım adım gösterilmiştir.



Şekil 2.1. Adım Adım Gambol-Gumbo Dönüşüm Matrisinin Oluşturulması

Matrisin sağ alt köşesinde yer alan '2' değeri, GAMBOL kelimesinin GUMBO kelimesine dönüşebilmesi için gerekli olan değişiklik sayısını ifade eder. GAMBOL

kelimesi GUMBO kelimesine dönüştürülmesi için “A” harfinin “U” harfi ile değiştirilmesi ve “L” harfinin silinmesi gerekmektedir.

(http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm)

2.2. DAMERAU – LEVENSTEIN DISTANCE

Damerau-Levenshtein Distance algoritması Levenshtein algoritmasına yapılan bir ekleme ile meydana gelmiştir.

Levenshtein algoritmasında bir metnin diğer metne dönüştürülmesinde yapılan işlemler; ekleme, çıkarma ve değiştirmedir. Yan yana gelen harfleri yer değişmiş vaziyette yazmak çok sık yapılan yazım hatalarındandır. Örneğin “kitap” yerine “kiatp” yazmak gibi. Damerau, bu durumu göz önüne alarak, uzaklık mesafesi hesaplanırken yan yana olan iki karakterin birbiri ile yer değiştirilmesi işlemini tek bir işlem sayacak şekilde algoritmayı güncellemiştir. (Damerau, 1964)

Levenshtein algoritmasına göre “kiatp” kelimesinin “kitap” a dönüşmesi için üçüncü ve dördüncü karakterler için ayrı ayrı değişim işlemi yapılmalıdır. İki kelimenin birbirine uzaklığı 2 dir. Damerau-Levenshtein algoritmasına göre ise üçüncü ve dördüncü karakterler için kendi içinde yer değişmesi işlemi uygulanır ve tek bir işlem sonucunda “kiatp” kelimesi “kitap” haline gelir. Uzaklık 1 olarak hesaplanır.

2.3. LONGEST COMMON SUBSEQUENCE

En uzun ortak altküme problemi, birden fazla metin dizisindeki (genellikle 2 tane) tüm diziler için ortak olan en uzun alt diziyi bulma problemidir.

(https://en.wikipedia.org/wiki/Longest_common_subsequence_problem)

Metin dizilerine karakterlerden oluşan kümeler nazarı ile bakılırsa, iki kümenin ortak elemanlarının her iki kümede de aynı sırada yer aldığı en uzun ortaklığını tespit eder. Örnek olarak:

A->{X,M,J,Y,A,U}

B->{M,Z,J,A,W,X,U}

Şeklindeki iki kümenin en uzun ortak altkümesi: LCS -> {M,J,A,U}

Kümelerden birisi ana küme olarak seçilir ve bu kümenin elemanları, sırasıyla diğer kümenin elemanları ile karşılaştırılır. Ana küme olarak A nın seçildiği durumda, ilk elemandan başlayarak son elemana kadar ortak elemanlar aranacaktır. İlk eleman olan X, diğer kümede aranacak, bulunursa diğer kümede X den sonra gelen eleman olan U, A kümesinde aranacaktır. Bu şekilde her iki kümenin de sonuna gelene kadar arama işlemine devam edilir. Akabinde A nın ikinci elemanı olan M için aynı işlem yapılır ve bu şekilde A nın tüm elemanları için tek tek bu işlem yapıldığında aşağıdaki sonuç elde edilir:

LCS1->{X,U}

LCS2->{M,J,A,U}

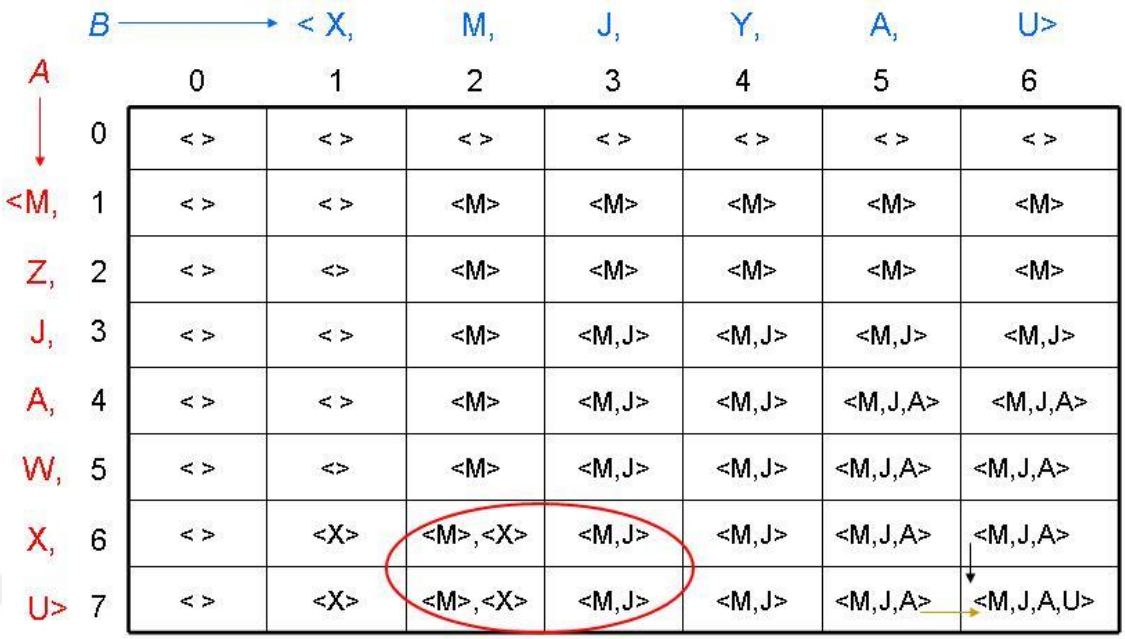
LCS3->{J,A,U}

LCS4->{} Y'nin eşi diğer kümede yok.

LCS5->{A,U}

LCS6->{U}

Görüldüğü üzere en uzun ortak altküme 2. adımda ortaya çıkan LCS2: {M,J,A,U} kümesidir. Problemin adım adım giderek tablo ile çözülmüş hali Şekil 2.2'de gösterilmiştir. Tablonun sağ alt köşesinde sonuç yer almaktadır. Tablo oluşturulması şu şekildedir: Başlangıçta 0. satır ve sütunlar boş bırakılır. Her hücre kendisinin bir üstündeki veya bir solundaki hücre içeriğini aynen alır ve kendi bulunduğu satır ve sütundaki değerler aynı ise bu değeri de kümeye ekler. Eğer üzerinde yer alan hücredeki küme ile solundaki hücrede yer alan küme farklı ise, bu durumda her iki küme de aynı hücreye yazılır. Bu durumun bir örneği tabloda kırmızı çember içerisinde görülmektedir. Daha uzun bir küme elde edilince kısa küme bırakılıp uzun küme ile devam edilir. (http://bilgisayarkavramlari.sadievrenseker.com/2007/12/04/en-uzun-ortak-kume-longest-common-subsequence-lcs/)



Şekil 2.2. LCS Algoritmasının Tablo ile Çözüm Örneği

LCS algoritması yazım denetleme, otomatik düzeltme gibi alanlarda kullanılmaktadır. En çok kullanıldığı alanlardan biri de genetik çalışmalardır; farklı canlılar için DNA sıralamasının benzerliklerinin araştırılmasında kullanılmaktadır. (http://w3.gazi.edu.tr/~akcayol/files/AAL8Dynamic.pdf)

2.4. BİTAP ALGORİTMASI

Unix agrep (approximate grep) uygulamasının altında yatan algoritmalardan birebir veya yaklaşık metin karşılaştırma yapan bir algoritmadır. Bu algoritmayı diğer metin karşılaştırma algoritmalarından ayıran temel fark, algoritmanın hızlı çalışmasını sağlayan bit bazlı çalışıyor olmasıdır.

Birebir metin karşılaştırma için kullanılan bitap algoritması, eşleşecek olan metin deseninin 31 bit uzunluğunda –alfabe uzunluğu- long tipinde bir bit matrisine dönüştürülmesi ile yapılır. Örneğin:

“abcc” gibi bir desenin bit matrise dönüştürülmesi:

11111111111111111111111111110 # desen/a

11111111111111111111111111101 # desen/b

11111111111111111111111110011 # desen/c

“a” deseninde “a” nın her görüldüğü yer için bir sıfır bulunmaktadır. Aynı şekilde “b” desenin de “b” nin bulunduğu her yer için sıfır bulunmaktadır. “c” desenine bakıldığında “c” nin görüldüğü 3. ve 4. karakterler için sıfır olduğu görülebilir. Desenin eşleştiği sütunlardaki bitler 0, eşleşme dışındaki bütün bitler 1 değerindedir.

“abcc” deseninin “aaabcca” metni üzerinde eşleşmesi için öncelikle ~1 değerinde bir long dizisi oluşturulur.

11111111111111111111111111110 # ilk durum

Karşılaştırma yapabilmek için metnin tüm karakterleri üzerinde bir döngü kurularak bitsel veya işlemi gerçekleştirilir. Öncelikle desen/a ile bitsel olarak karşılaştırılır ve ardından sola 1 bit bitsel kaydırma yapılır.

Metin: **a**aabcca

11111111111111111111111111110 # desen/a

11111111111111111111111111110 # ilk durum

11111111111111111111111111100 # ikinci durum

Yukarıdaki durumda ilk karakterin eşleşmesi durumunda karşılaştırma matrisinin değişimi gösterilmiştir. Bu dizide 0 eşleşme 1 ise eşleşmeme durumlarını göstermektedir. İkinci karakter için işlem yapıldığında

Metin a**a**abcca

11111111111111111111111111110 # desen/a

11111111111111111111111111100 # önceki durum

11111111111111111111111111100 # yeni durum

Halen sadece “a” eşleşti;

Metin aa**a**bcca

11111111111111111111111111110 # desen/a

11111111111111111111111111111100 # önceki durum

11111111111111111111111111111100 # yeni durum

Metin aaa**b**cca

11111111111111111111111111111101 # desen/b

11111111111111111111111111111100 # önceki durum

11111111111111111111111111111010 # yeni durum

Yukarıda görüldüğü gibi ikinci eşleşme gerçekleşti üçüncü karaktere bakacak olursak;

Metin aaab**c**ca

11111111111111111111111111110011 # desen/c

11111111111111111111111111111010 # önceki durum

11111111111111111111111111110110 # yeni durum

Üçüncü karakter eşleşti ve son karakter;

Metin aaabc**c**a

11111111111111111111111111110011 # desen/c

11111111111111111111111111110110 # önceki durum

11111111111111111111111111101110 # yeni durum

Desenin uzunluğu m kadar ise her döngüde m+1'inci bit eğer sıfır ise eşleşmenin tamamlandığı anlamına gelecektir. Bu algoritmanın dezavantajı 31 karakter veya daha kısa metinleri karşılaştırabilmesidir. (http://www.wucathy.com/blog/?p=680)

2.5. HAMMING DISTANCE

Hamming Distance (HD) aynı uzunluktaki iki dizgeden kaynak dizgenin hedef dizgeye dönüşebilmesi için gerekli olan değiştirme sayısını veren algoritmadır. Bu algoritmada ekleme ve çıkarma işlemi yoktur; sadece değiştirme işlemi vardır. Başka bir deyişle x, y olarak verilen iki metinsel ifadenin arasındaki Hamming Distance değeri $x_i \neq y_i$ olan durumların sayısıdır.

HD algoritma örnekleri:

- “karolin” ve “kathrin” karşılaştırıldığında sonuç 3 tür.
- “karolin” ve “kerstin” karşılaştırıldığında sonuç 3 tür.
- “1011101” ve “1001001” karşılaştırıldığında sonuç 2 dir.
- “2173896” ve “2233796” karşılaştırıldığında sonuç 3 tür.

Bu algoritmanın önemli bir dezavantajı karşılaştırılan metinlerin birebir aynı uzunlukta olması zorunluluğudur.

(http://www.ieee.ma/uaesb/pdf/distances-in-classification.pdf)

Hata tespiti ve düzeltilmesi, grafik dosyaları üzerinden şekil eşleştirmelerinin yapılması gibi hesaplamalarda kullanılmaktadır.

(https://www.buraksenyurt.com/post/hamming-distance-algoritmasinin-basit-kullanimi)

Hamming Distance, telekomünikasyon alanında, hata tahmini yapmak için sabit uzunluklu ikili bir metindeki bozulmuş bitleri saymada da kullanılır. Bu yüzden Signal Distance da denilir. (https://en.wikipedia.org/wiki/Hamming_distance)

2.6. JARO WINKLER DISTANCE

Jaro Winkler Distance (JWD), iki dizgenin arasındaki farklılığın tespiti için kullanılan algoritmalardan biridir. 1989 yılında Matthew A. Jaro tarafından geliştirilen Jaro Distance algoritmasının bir türevi olarak, 1990 yılında William E. Winkler tarafından geliştirilmiştir.

(https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance)

2.6.1. Jaro Benzerliği

Jaro benzerliği verilen iki metinsel ifade $s_1$ve $s_2$için

$|s_i|$ metinsel ifadelerin (s_i) uzunluğu,

m: eşleşen karakterlerin sayısı,

t: yer değiştirmelerin sayısı olmak üzere;

$$ben_j = \begin{cases} 0 & eğer\ m = 0\ ise; \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & diğer; \end{cases}$$

formülü ile hesaplanır.

(https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance)

2.6.2. Jaro Winkler Benzerliği

Jaro Winkler benzerliği bir p skaleri ve l uzunluk katsayısı kullanarak ilk karakterden itibaren eşleşen metinsel ifadeler için daha anlamlı sonuçlar sunar.

ben_j Jaro benzerliği,

l: metinsel ifadenin başlangıcından itibaren maksimum dört karaktere kadar eşleşmiş karakter katsayısı

p: skala katsayısı; varsayılan değeri 0.1 olup, maksimum 0.25 olabilen katsayıdır. Bu katsayı elde edilecek benzerlik değerinin 1 den küçük olmasını sağlamak için kullanılır.

$$ben_w = ben_j + (lp(1 - ben_j))$$

Formülüyle hesaplanır. JWD ise;

$$d_w = 1 - ben_w$$

Formülüyle hesaplanır.

(https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance)

ÜÇÜNCÜ BÖLÜM

ATATÜRK ÜNİVERSİTESİ ÖĞRENCİ BİLGİ SİSTEMİ VE YAŞANAN SORUNLAR

3.1. ATATÜRK ÜNİVERSİTESİ ÖĞRENCİ BİLGİ SİSTEMİ

Atatürk Üniversitesi, 1957 yılında kurulmuş olup, 61 yıllık geçmişi ile ülkemizin en eski üniversitelerinden biridir. 2018 yılı itibari ile bünyesinde 23 fakülte, 8 enstitü, 1 yüksekokul, 12 meslek yüksekokulu, 1 konservatuvar ve 29 araştırma merkezi barındırmaktadır. Bu birimler ilçeler de dâhil olmak üzere 11 ayrı yerleşkeye dağılmıştır.

Atatürk Üniversitesi'nden mezun olan öğrenci sayısı 304,428'tir. Bu sayı, bilgisayar ortamında verisi bulunan öğrencileri kapsamaktadır. "Dijital Arşiv Projesi" tamamlandığında gerçek sayıya ulaşılabilecektir. Temmuz 2018 itibari ile üniversitede aktif bir şekilde öğrenimine devam eden 306,296 öğrenci bulunmaktadır. Öğretim tipine göre, program ve öğrenci sayıları Tablo 3.1'de; öğrenim seviyelerine göre program ve öğrenci sayıları Tablo 3.2'de verilmiştir.

Tablo 3.1. Öğretim Tipine Göre Öğrenci Sayıları

Öğretim Tipi	Program Sayısı	Öğrenci Sayısı
Örgün öğretim	1,019	46,080
İkinci öğretim	104	15,661
Açık öğretim	40	241,969
Uzaktan eğitim	23	2,586
Toplam	1,186	306,296

(https://ubys.atauni.edu.tr/) (https://obs.atauni.edu.tr/)

Tablo 3.2. Eğitim Düzeyine Göre Öğrenci Sayıları

Öğrenim Seviyesi	Program Sayısı	Öğrenci Sayısı
Ön Lisans	208	209,948
Lisans	205	85,323
Yüksek Lisans	426	8,541
Doktora	344	2,465
Sanatta Yeterlik	3	19
Toplam	1,186	306,296

(https://ubys.atauni.edu.tr/) (https://obs.atauni.edu.tr/)

Atatürk Üniversitesinde öğrenci bilgileri, önceleri kâğıt ortamında manuel bir şekilde tutuluyorken, 2000 yılında ilk defa bilgisayarlı otomasyon sistemi kullanılmaya başlanmıştır. Oracle veri tabanı kullanılarak, Delphi programlama dili ile yazılmış olan bu otomasyon sistemi Bilkent Üniversitesi tarafından sağlanmış olup, 2009 yılına kadar kullanılmıştır. Bakım yetersizliği, personel için web ara yüzünün olmaması vs. gibi sebeplerden ötürü bu program artık ihtiyacı karşılayamaz duruma geldiği için, Üniversite yeni yazılım arayışına girmiş ve 2009 yılında özel bir firma ile geliştirici ortak statüsü ile bir yazılım satın almıştır. İlerleyen süreçte firma ile sorunlar yaşanmış ve firma desteğini tamamen çekmiştir. Bu aşamadan sonra Üniversite, bu projede geliştirici ortak olarak yer aldığı için kendi Öğrenci Bilgi Sistemi (ÖBS) proje ekibini oluşturmuştur. Bu ekip yazılımı çalışır hale getirmek için yoğun gayret sarf etmiştir. 2011 yılında proje ekibi kendi arayüzlerini yazmaya başlamış ve ilerleyen süreçte firmanın yazılımı ile ilişiğini tamamen kesmiştir. Üniversite, 2011 yılından itibaren Bilgisayar Bilimleri Araştırma ve Uygulama Merkezi bünyesindeki ÖBS proje ekibi tarafından yazılan ve sürekli geliştirilmeye devam edilen kendi Öğrenci Bilgi Sistemi (ÖBS) yazılımını kullanmaktadır.

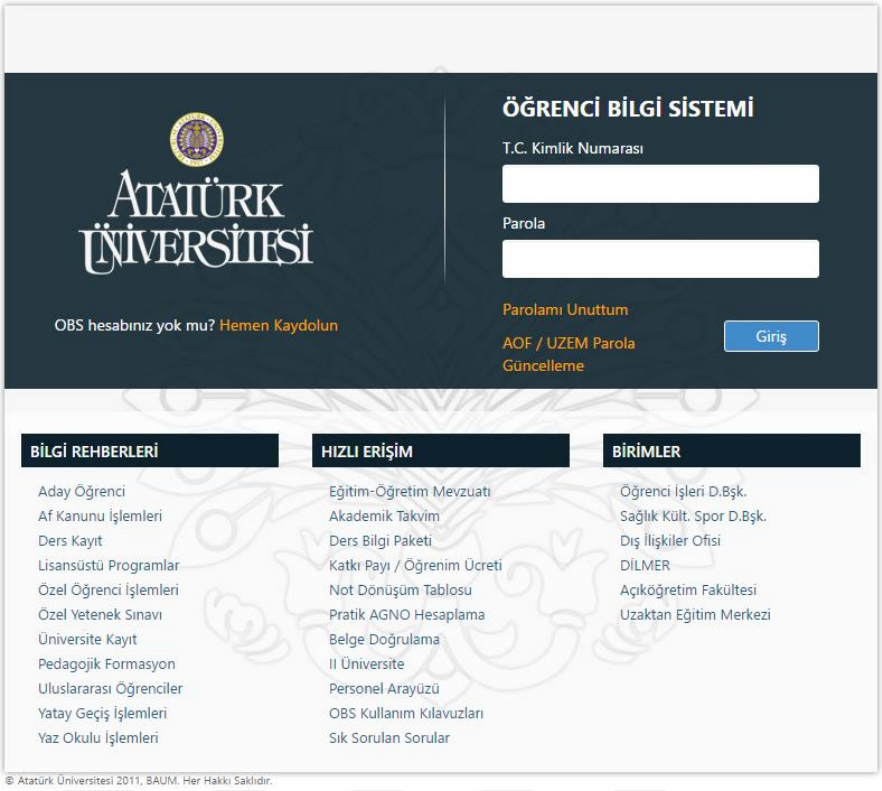
Atatürk Üniversitesi Öğrenci Bilgi Sistemi'nde, veri tabanı olarak Microsoft SQL Server 2017 sürümü kullanılmaktadır. Kullanıcı ara yüzlerinde bazı modüller ASP.Net ve C# ile yazılmış olup Microsoft IIS sunucusunda, diğer modüller PHP ile yazılmış olup Linux sunucusunda çalışmaktadır. Her iki sunucu arasında oturum paylaşımı vardır.

ÖBS ye giriş yapan her kullanıcı kendi kullanıcı grubunun ya da gruplarının sahip olduğu yetkilere göre işlem yapabilmektedir. Örneğin ders açma, açılan derse haftalık program girme, derse kontenjan tanımlama, derse öğretim üyesi atama vs. gibi işlemler "Fakülte/MYO Öğrenci İşleri Grubu" nun yetkisi dâhilinde iken, açılan derse sınav tanımlamak, derse kayıtlı öğrencilerin sınav notlarını girmek, devamsızlık işlemek, harf notlarını hesaplatmak, harf notunu ilan etmek gibi işlemler "Öğretim Üyeleri Grubu" nun yetkisindedir.

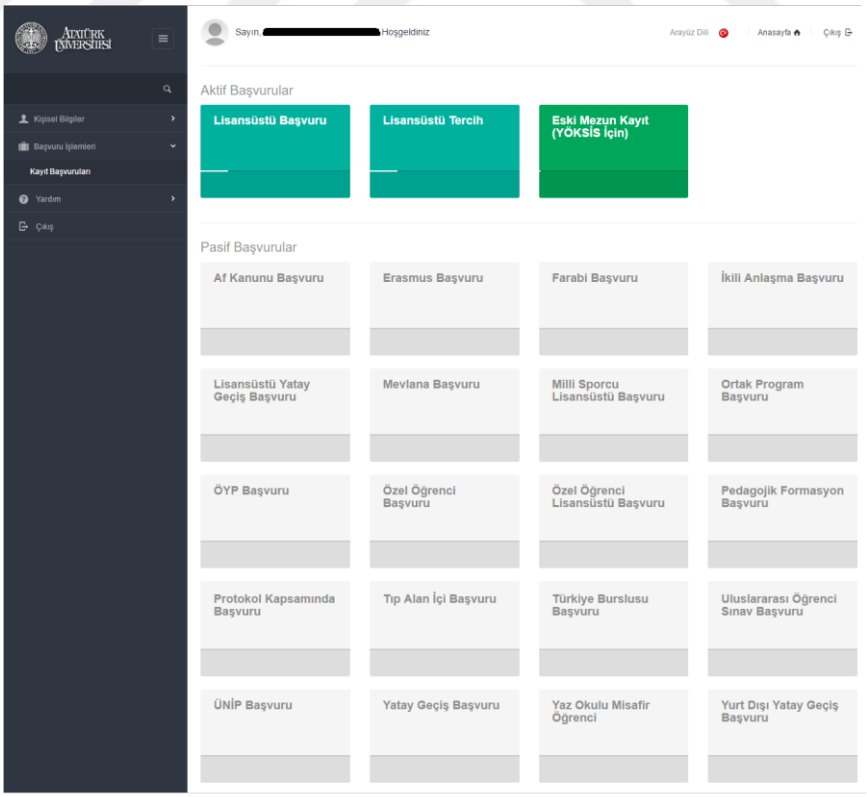
Öğrenci Bilgi Sisteminde yer alan temel kullanıcı grupları aşağıda listelenmiştir:

- Başvuru Grubu
- Öğrenci Grubu
- Öğretim Üyeleri Grubu
- Danışman Grubu
- Birim İdarecileri Grubu
- Fakülte/MYO Öğrenci İşleri Grubu
- Enstitü Öğrenci İşleri Grubu
- AÖF Öğrenci İşleri Grubu
- ÖİDB Personeli Grubu
- Yönetici Grubu
- Harçlar Yönetim Grubu

Öğrenci Bilgi Sistemine isteyen herkes, Şekil 3.1'de görülen giriş sayfasında yer alan "Hemen kaydolun" linki üzerinden cep telefon numarasına gelen doğrulama kodu ile kayıt olabilmektedir. Bu şekilde kayıt olan kullanıcılar, genel bir grup olan "Başvuru grubu" na dâhil olurlar. Bu gruptaki kullanıcılar, Şekil 3.2'de sol tarafta görüldüğü gibi özlük, iletişim vs. gibi bilgilerin yer aldığı "Kişisel Bilgiler" sayfası, "Yardım" ve "Başvurular" sayfalarını görüntüleyebilmektedir. Bu sayfalara erişim yetkisi tüm kullanıcı gruplarında mevcuttur. Şekil 3.2'de görülen başvurular sayfasında, ÖBS üzerinden alınan çeşitli başvurular listelenmekte ve aktif olan başvurulara müracaat edilebilmektedir.



Şekil 3.1. ÖBS Kullanıcı Giriş Ekranı



Şekil 3.2. ÖBS'de Yer Alan Başvurular Sayfası

3.1.1. Müfredat Kavramı

Atatürk Üniversitesi lisans ve ön lisans programlarında, her akademik yıl için, programa o yıl kaydolan öğrencilerin tabi olacağı bir müfredat bulunmaktadır. Müfredat içerisinde, öğrencinin alması gereken zorunlu dersler ve alabileceği seçmeli dersler ile bu derslerin kodu, bulunduğu dönemi, ortalamaya etkisi olup olmadığı, krediye etkisi olup olmadığı gibi bilgiler yer almaktadır. Öğrenciler ders seçimi yaparken sadece tabi olduğu müfredatta yer alan derslere kayıt yaptırabilmektedir.

3.2. ÖĞRENCİ BİLGİ SİSTEMİNDE DERS HAVUZU

Öğrenci Bilgi Sistemi ders havuzunda 38,156 adet ders bulunmaktadır. Bu havuzdaki dersler Üniversitedeki tüm programların kullanımına açıktır. Herhangi bir program için müfredat oluşturulurken dersler bu havuzdan çekilir. Ders havuzu, veri tabanında Dersler tablosunda yer almaktadır. Bu tabloda derslerin Id numarası, adı, kısa adı, kredisi, teori / uygulama / laboratuvar saat sayısı gibi bilgiler yer almaktadır. Bu havuza yeni bir ders eklemek isteyen kullanıcılar, ders talep sayfasından talepte bulunmaktadır. Bu talepler DersTalepleri tablosuna kaydedilmektedir. Ders ekleme talebinde bulunurken, sistem; istenen dersin adı, kredisi, teori / uygulama / laboratuvar saat sayısı bilgileri ile aynı olan bir dersin havuzda mevcut olup olmadığına bakmakta; bu bilgilere sahip bir ders havuzda mevcut ise talep kaydedilmemektedir. Talep edilen dersin kredisinin; teori, uygulama ve laboratuvar saat bilgileri ile uyumlu olup olmadığı da kontrol edilmektedir. Ders taleplerine bakan personel, yazılan ders adının, havuza eklemeye uygun olup olmadığına karar vermekte; buna göre talebi onaylamakta veya reddetmektedir. Personel onay / ret işlemini, Öğrenci İşleri Daire Başkanlığı (ÖİDB) ve ÖBS proje ekibi tarafından belirlenmiş olan “Ders talebinde bulunurken dikkat edilmesi gereken hususlar” a göre ve genel olarak ders adının anlaşılırlığına, Türk Dil Kurumu Yazım Kılavuzu’na uygunluğuna bakarak karar vermektedir. Ders talebinde bulunurken dikkat edilmesi gereken hususlar Ek 8’de yer almaktadır.

Havuza ders eklemek, yukarıda belirtildiği üzere belli bir sistematiğe bağlıdır. Fakat daha önceleri bu kontroller yapılmadan havuza dersler eklenmiş olup; şu anda farklı Id ye, fakat aynı isme sahip dersler mevcuttur. Örneğin 3 kredilik “Matematik I” dersi, dersler tablosunda 15 defa yer almaktadır.

Tablo 3.3. Dersler Tablosunda Yer Alan 3 Kredilik "Matematik I" Dersleri

Ders_Id	Ders_Adi	Kredi
5070	Matematik I	3
5757	Matematik I	3
5758	Matematik I	3
5759	Matematik I	3
7582	Matematik I	3
7583	Matematik I	3
15370	Matematik I	3
27141	Matematik I	3
27856	Matematik I	3
27857	Matematik I	3
27912	Matematik I	3
27913	Matematik I	3
27914	Matematik I	3
27915	Matematik I	3
27916	Matematik I	3

Veri tabanında, Tablo 3.3'te listelenen "Matematik I" dersinde olduğu gibi, aynı isim ve krediye sahip olup birden çok defa mevcut olan 3,534 adet ders adı bulunmaktadır. Bu 3,534 adet ders, ayrı ayrı sayıldığında toplam 8,530 tanedir ve tüm derslerin %22.4 üne tekabül etmektedir. En fazla tekrar eden aynı isim ve krediye sahip 30 ders Tablo 3.4'te yer almaktadır:

Tablo 3.4. En Fazla Tekrar Eden Aynı İsim ve Krediye Sahip Dersler

Ders Adı	**Kredi**	**Sayı**
Atatürk İlkeleri ve İnkılap Tarihi	0	8
Atatürk İlkeleri ve İnkılap Tarihi I	0	27
Atatürk İlkeleri ve İnkılap Tarihi I	2	14
Atatürk İlkeleri ve İnkılap Tarihi II	0	9
Atatürk İlkeleri ve İnkılap Tarihi II	2	10
Bilgisayar Destekli Tasarım I	2	12
Bilgisayar I	2	10
Bilgisayar I	3	9
Bilgisayar II	2	8
Bilimsel Araştırma Yöntemleri	3	8
Ekonometri II	3	7
Fizik I	3	7
Fizik I	4	8
Fizik I	5	9
Fizik II	3	8
Fizik II	4	8
Fizik II	5	9
Girişimcilik	3	7
İngilizce I	2	8
İnsan Kaynakları Yönetimi	3	10
Kalite Güvence ve Standartları	2	8
Matematik I	3	15
Matematik II	3	8
Mesleki Yabancı Dil II	3	7
ÖĞRETMENLİK UYGULAMASI	5	9
Sistem Analiz ve Tasarım II	2	8
Sistem Analiz ve Tasarımı	3	8
Türk Dili I	0	8
Uzmanlık Alan Dersi	0	12
Yabancı Dil I	0	20

3.3. DERS ADLARINDAKİ ÇOĞULLAMALARIN SEBEP OLDUĞU SORUNLAR

Ders adlarındaki çoğullama çok çeşitli sorunlara yol açmaktadır. Bu sorunların en önemlisi: aldığı bir dersi daha sonra tekrar alan bir öğrenci, sonraki alışında, ilkinde aldığı dersten farklı bir Id'ye sahip dersi alması durumudur. Bu durumda öğrencinin AGNO'su ve kredisi yanlış hesaplanır.

3.3.1. Aynı Dersin, Programın Farklı Yıllara Ait Müfredatlarında Farklı Id ile Yer Alması

Her program için yeni eğitim-öğretim yılı başında, programın o yılına ait müfredat oluşturulmaktadır. Müfredata ekleme yapılırken ders için, daha önce kullanılan dersten farklı Id ye sahip bir dersin eklenmesi halinde oluşan durumdur. Tablo 3.3'de yer alan "Matematik I" dersleri üzerinde giderek bir örnek verilebilir: İktisadi ve İdari Bilimler Fakültesinde yer alan İktisat programının 2017-2018 akademik yılına ait müfredatı oluşturulurken 1. döneme 5070 Id li "Matematik I" dersi eklenmiş olsun. Programın 2016-2017 yılına ait müfredatında da 5757 Id li "Matematik I" dersi yer alıyor olsun. Bu durumda 2017-2018 akademik yılı ders kayıtlarında İktisat programındaki öğrenciler için 5070 Id li "Matematik I" dersi açılacaktır. Fakat bir önceki yıl "Matematik I" dersinden kalmış olan 2016-2017 müfredatına tabi öğrenciler 5070 Id li derse kayıt yapamayacaklardır; çünkü bu Id li ders kendi müfredatlarında yer almamaktadır. Bu öğrenciler için ayrıca 5757 Id li "Matematik I" dersi açılması gerekecektir. Bu durumda Fakülte Öğrenci İşleri personeli iki ayrı "Matematik I" dersi açmak zorunda kalacaktır. Hem iş yükleri artacak hem de tutarsız durumlar oluşabilecektir. Aynı zamanda aynı sınıfta verilen bu derslere ait bilgilerde bir güncelleme yapılması gerektiğinde; örneğin dersin haftalık programının güncellenmesi gerektiğinde, değişikliğin bir derste yapılıp diğerinde yapılmaması durumunda dersin bilgilerinde tutarsızlık oluşacaktır. Dersin öğretim üyesi, ders için yapması gereken işlemleri her iki ders için ayrı ayrı yapmak durumunda kalacaktır. Her ders için ayrı ayrı sınav tanımlaması, harf notlarını ayrı ayrı hesaplatması gerekecektir. Derslerin yoklama listeleri de ayrı ayrı olacaktır. Aynı zaman ve ortamda beraber ders gören öğrenciler iki gruba bölünmüş olacaktır. Bu durum harf notu hesaplatılırken de sorun oluşturabilecektir. Bağıl değerlendirme sisteminde harf

notları, öğrencilerin vize-final ortalamalarından elde edilen ham başarı notuna, ham başarı notlarının sınıf ortalaması (SO) ile standart sapmasına (SS) bağlı olarak hesaplanmaktadır. Bu derste öğrenciler iki gruba bölündükleri için sınıfın iki ayrı SO ve SS değeri olacaktır. Bu durumda her grup için, kendi gruplarının SO ve SS dikkate alınarak harf notları hesaplanacaktır. Dolayısıyla farklı gruplarda yer alıp aynı ham başarı notuna sahip olan öğrenciler için, farklı harf notu hesaplanması ihtimali doğacaktır. Aynı zaman ve aynı ortamda, aynı öğretim üyesinden ders dinleyip, sınavlardan da aynı notu alan öğrencilerin dönem sonunda farklı harf notu alması etkili ölçme ve değerlendirme ilkeleri açısından ciddi bir sorundur.

3.3.2. Dersin, Programın Aynı Yılına Ait Müfredatında Farklı Id ler ile Birden Fazla Yer Alması

Müfredata ders eklenirken, eklenmek istenen dersin o müfredatta yer alıp almadığı kontrol edilmektedir. Bu kontrol dersin Id si üzerinden yapılmaktadır. Örneğin İktisat programının 2017-2018 müfredatına 5070 Id li "Matematik I" dersi yeniden eklenmek istenirse sistem buna izin vermeyecek; "Bu ders müfredatta mevcuttur" şeklinde uyarı verecektir. Fakat 5757 Id li "Matematik I" dersi eklenmek istenirse, sistem buna müsaade edecektir. Bu şekilde 2017-2018 müfredatına her iki "Matematik I" dersi de eklenirse ders kayıtlarında aynı müfredata tabi öğrencilerin bir kısmı 5070 Id li dersi, diğerleri 5757 Id li dersi alacağından dolayı, başlık 3.3.1 de anlatılan sorunlar bu durumda da yaşanacaktır. Bu senaryoda yaşanması muhtemel bir diğer sorun; kaldığı bir dersi daha sonra tekrar alan bir öğrencinin, sonraki alışında ilk aldığı dersin Id sinden farklı Id ye sahip bir dersi alması durumudur. Bu durumda öğrencinin AGNO'su ve toplam kredisi yanlış hesaplanır. Sonra alınan ders, önceki aldığı ders ile aynı isme sahip olmasına rağmen, Id si farklı olduğu için, sistem tarafından farklı bir ders olarak algılanır. Dolayısıyla toplam kredisi olması gerekenden daha fazla, AGNO'su daha düşük hesaplanır. Ders adlarındaki çoğullamanın oluşturduğu en büyük sorun budur. "İnsan Kaynakları Yönetimi" dersi için bu sorunu yaşayan bir öğrencinin transkripti Şekil 3.3 de gösterilmiştir.

3.3.3. Yaz Okulunda Yaşanan Sorunlar

Atatürk Üniversitesinde yaz okulunda ders açılırken genellikle fakülteye özel olan dersler fakülte içinde, genel dersler ise ders ile ilgili fakültede açılmaktadır. Türk Dili I/II, İngilizce I/II ve Atatürk İlke ve İnkılapları I/II dersleri tüm Üniversite genelinde ortak zorunlu derstir. Yaz okulunda bu dersler sadece Edebiyat Fakültesi bünyesinde açılmaktadır. Diğer fakültelerin öğrencileri de yaz okulunda bu dersleri almak isterlerse eğer, Edebiyat Fakültesi'nden almaları gerekmektedir. Benzer şekilde "Hukukun Temel Kavramları" dersi normal dönemde birçok fakültede açılırken, yaz okulunda sadece Hukuk Fakültesi'nde açılmaktadır. Ayrıca bazı programların müfredatlarında bu ders "Temel Hukuk" olarak geçmektedir. Ders havuzunda bu derslerden 8 tane bulunmakta olup bu dersler Tablo 3.5'te listelenmiştir. Her programın müfredatında bu dersler aynı Id ile yer almadıkları için ve öğrenciler de sadece müfredatlarında yer alan derslere kayıt yapabildikleri için; yaz okulunda Hukuk Fakültesinde bu ders açılırken Tablo 3.5'de yer alan tüm derslerin açılması gerekmektedir. Bu durumda başlık 3.3.1 de anlatılan sorunlar burada da ortaya çıkmaktadır. Üstelik bu senaryoda aynı ortamda ve zamanda ders gören öğrenciler 8 ayrı gruba bölünmesi gerekmektedir.

Tablo 3.5. Dersler Tablosunda Yer Alan 3 Kredilik "Hukukun Temel Kavramları" Dersleri ve "Temel Hukuk" dersleri

Ders_Id	**Ders_Adi**	**Kredi**
6282	Hukukun Temel Kavramları	3
6402	Hukukun Temel Kavramları	3
62087	Hukukun Temel Kavramları	3
62111	Hukukun Temel Kavramları	3
38529	Hukukun Temel Kavramları I	3
9557	Temel Hukuk	3
27212	Temel Hukuk	3
27928	Temel Hukuk	3

3.3.4. Transkriptte AGNO ve Toplam Kredinin Hatalı Hesaplanması

Aldığı bir dersi daha sonra tekrar alan bir öğrenci, sonraki alışında, ilkinde aldığı dersten farklı bir Id ye sahip dersi alması durumunda öğrencinin AGNO'su ve kredisi yanlış hesaplanır. Sonra alınan ders, önceki aldığı ders ile aynı isme sahip olmasına rağmen, farklı bir Id ye sahip olduğu için, sistem tarafından farklı bir ders olarak algılanır. Dolayısıyla toplam kredisi, olması gerekenden daha fazla, AGNO'su da daha düşük hesaplanır.

2011 - 2012 Akademik Yılı Bahar Yarıyılı						
▲ 218 İŞL 2	İnsan Kaynakları Yönetimi (3661)	3.0	0.0	0.0	3.0	FF
222 İŞL 2	İngilizce II (6211)	2.0	0.0	0.0	2.0	G
202 İŞL 2	Mikro İktisat II (7436)	3.0	0.0	0.0	3.0	CC
▲ 204 İŞL 2	İstatistik II (7343)	3.0	0.0	0.0	3.0	FF
▲ 210 İŞL 2	Makro İktisat (7151)	3.0	0.0	0.0	3.0	FF
212 İŞL 2	Pazarlama Yönetimi II (5938)	3.0	0.0	0.0	3.0	DD
▲ 220 İŞL 2	Şirketler Muhasebesi (6230)	3.0	0.0	0.0	3.0	FF
					Kredi	AGNO
		Genel			72.00	1.29
2011 - 2012 Akademik Yılı Yaz Okulu						
218 İŞL 2	İnsan Kaynakları Yönetimi (10171)	3.0	0.0	0.0	3.0	DD
▲ 203 İŞL 2	İstatistik I (3666)	3.0	0.0	0.0	3.0	FF
213 İŞL 2	Sosyal Bilimlerde Araştırma Yöntemleri (9372)	3.0	0.0	0.0	3.0	CC
▲ 204 İŞL 2	İstatistik II (7343)	3.0	0.0	0.0	3.0	FF
					Kredi	AGNO
		Genel			75.00	1.36

Şekil 3.3. AGNO'su ve Toplam Kredisi Hatalı Hesaplanan Örnek Bir Öğrenci

Şekil 3.3'te transkriptinin bir kısmı gösterilen öğrenci 2011 - 2012 Akademik Yılı yaz okulunda sadece daha önce alıp başarısız olduğu dersleri tekrar etmiştir. Dolayısı ile hiç yeni ders almadığı için, yaz okulundan sonraki toplam kredisinin, bahar dönemi sonundaki toplam kredisi ile aynı olması gerekmektedir. Öğrenci bahar döneminde 3661 Id li "İnsan Kaynakları Yönetimi" dersini; yaz okulunda ise 10711 Id li "İnsan Kaynakları Yönetimi" dersini almıştır. Farklı Id li bir dersi aldığı için sistem, yaz

okulunda aldığı “İnsan Kaynakları Yönetimi” dersini, bahar döneminde aldığı dersten farklı bir ders olarak; sanki yeni bir dersi ilk defa almış gibi değerlendirmiş ve toplam kredisini artırarak 75 olarak hesaplamıştır. Hâlbuki öğrenci aynı Id ye sahip dersi alsa idi; yaz okulu sonundaki kredi toplamı 72 olarak kalacak, AGNO’su da 1.42 olacaktı.

3.3.5. Üst Sınıftan Ders Almada Yaşanan Sorun

Aynı dersi, farklı Id ile tekrar almak, üst sınıftan ders almada da sorun çıkarmaktadır. Üniversitemizde ikinci sınıf ve üzeri lisans öğrencileri üst sınıftan ders alabilmektedir. Üstten ders alabilmenin 2 şartı vardır:

(http://www.mevzuat.gov.tr/Metin.Aspx?MevzuatKod=8.5.23757&MevzuatIliski=0&sourceXmlSearch=)

- AGNO 2.5 in üzerinde olmalı
- Alttan dersi bulunmamalı

Derslerini genel olarak yüksek notlarla geçtiği için AGNO’su 2.5 in üzerinde olan, fakat kaldığı için tek bir dersini tekrar eden bir öğrenci ele alındığında, eğer tekrar ettiği dersi, farklı Id ler ile almış olursa; bu durumda gerçekte öğrenci tüm derslerini başarmış olmasına rağmen alttan bir dersi varmış gibi görünecektir. Dolayısı ile de ders kayıt zamanında sistem, öğrencinin üst sınıftan ders almasına müsaade etmeyecektir.

3.3.6. Yüzde 10 Listesine Girmede Sorun

Üniversitemizde ikinci öğretim öğrencileri, % 10 a girmeleri durumunda gündüz öğretimine geçiş yapma hakkı elde etmektedirler. İkinci öğretimden gündüz öğretimine geçmenin 2 şartı vardır:

(http://www.mevzuat.gov.tr/Metin.Aspx?MevzuatKod=8.5.23757&MevzuatIliski=0&sourceXmlSearch=)

- Bulunduğu sınıfta AGNO sıralamasında ilk % 10 içerisinde yer almalı
- Alttan dersi bulunmamalı

Yüksek AGNO’ya sahip olduğu için program içerisinde ilk % 10 içerisinde yer alan, fakat kaldığı için tek bir dersini tekrar eden bir öğrenci ele alındığında, eğer tekrar

ettiği dersi, farklı Id ler ile almış olursa; bu durumda gerçekte öğrenci tüm derslerini başarmış olmasına rağmen alttan bir dersi varmış gibi görünecektir. Dolayısı ile de sene sonunda % 10 listeleri alınırken bu öğrenci listeye dâhil edilmeyecektir.

DÖRDÜNCÜ BÖLÜM

UYGULAMA

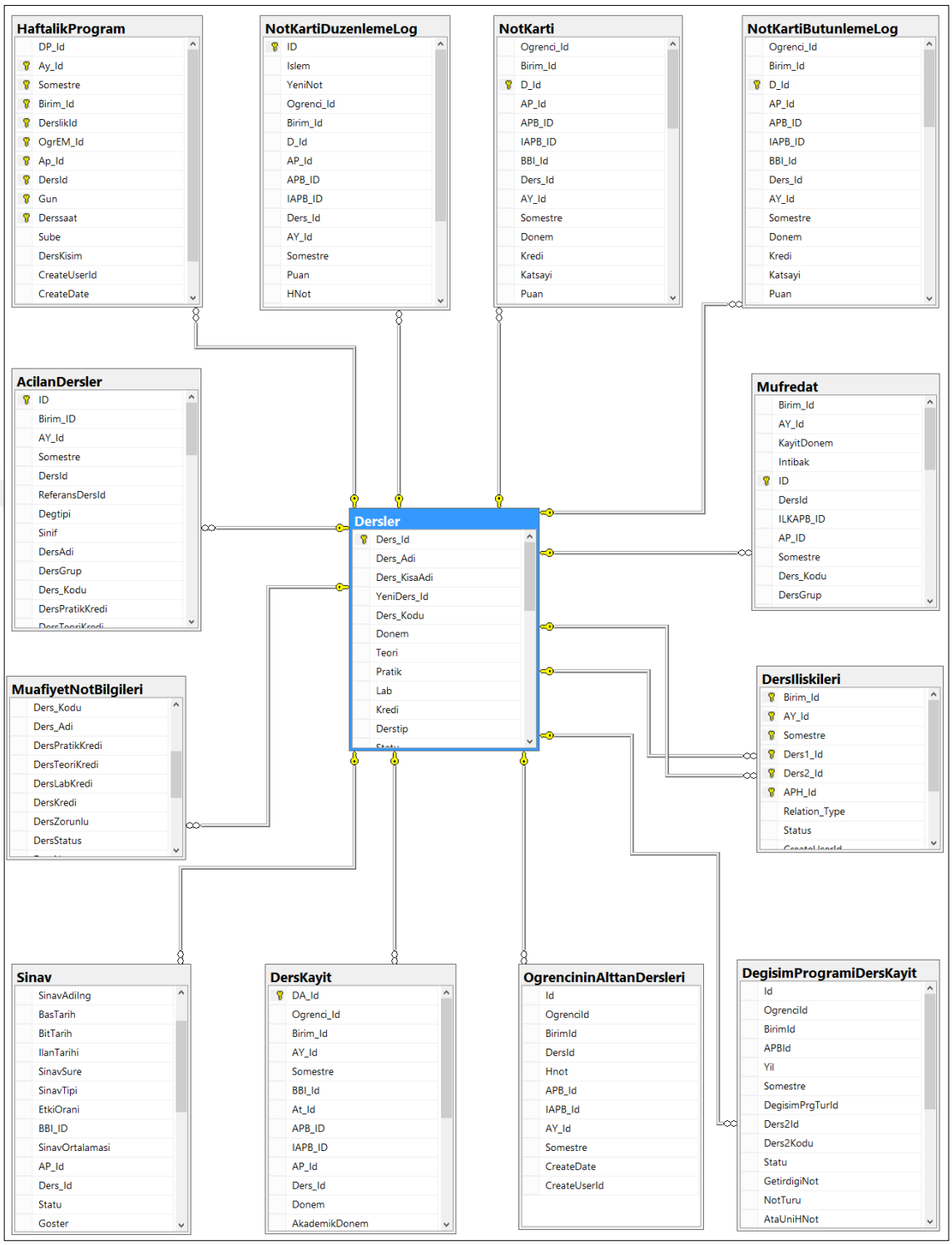
Bu çalışmanın ilk kısmında amaç aynı isim ve kredili dersleri, derslerden biri üzerinde birleştirmektir. “Fizik II” / “Fizik 2”; “Kimya I” / “KIMYA I”; “Mikroiktisat” / “Mikro iktisat” / “Mikro İktisad” / “Mikroekonomi” vs. gibi aynı isimli olmayan dersleri birleştirmek de çalışmanın kapsamındadır. Bu şekildeki yazımı farklı ama esasında aynı olan dersleri tespit etmek çalışmanın ikinci kısmını oluşturmaktadır. Üçüncü ve son kısımda, ÖİDB personelinin kullanımı için; yazımı birbirine benzeyen derslerin listelenmesi, uygun görülen derslerin birleştirme işleminin yapılmasını / geri alınmasını sağlayan bir arayüz yapılmıştır.

4.1. DERS BİRLEŞTİRME

Dersler tablosunda Ders_Id sütunu birincil anahtardır ve veri tabanındaki birçok tabloda yer alan sütunlara dış anahtar konumundadır. Bu şekilde Dersler tablosuna refere eden sütun içeren tablolardan veri bütünlüğü açısından önemli olan tablolar ve sütunları Tablo 4.1’de yer almaktadır. DersIliskileri tablosunun Dersler tablosuna refere eden iki sütunu olduğu için listede iki defa yer almıştır. Bu tablolar ile Dersler tablosunun ilişkileri Şekil 4.1’de gösterilmiştir. Ders birleştirme işleminde aynı isim ve farklı Id ye sahip olan dersler, derslerden biri üzerinde birleştirilir ve kalan dersler kullanıma kapatılır. Birleştirme ve yapılan birleştirmeyi geri alma işlemleri saklı yordamlar ile yapılmaktadır.

Tablo 4.1. Dersler Tablosuna Refere Eden Sütun İçeren Tablolar ve Sütun İsimleri

	TabloAdi	**KolonAdi**
1	AcilanDersler	DersId
2	DersIliskileri	Ders1_Id
3	DersIliskileri	Ders2_Id
4	Mufredat	DersId
5	NotKarti	Ders_Id
6	NotKartiButunlemeLog	Ders_Id
7	DersKayit	Ders_Id
8	HaftalikProgram	DersId
9	NotKartiDuzenlemeLog	Ders_Id
10	OgrencininAlttanDersleri	DersId
11	OgrencininAlttanDersleri_AOF	DersId
12	UzemAGNOicinDersKayit	Ders_Id
13	Sinav	Ders_Id
14	DegisimProgramiDersKayit	Ders2Id
15	MuafiyetNotBilgileri	Ders_Id
16	MuafiyetNotBilgileri_Log	Ders_Id



Şekil 4.1. Dersler Tablosu ile İlişkili Diğer Tablolar

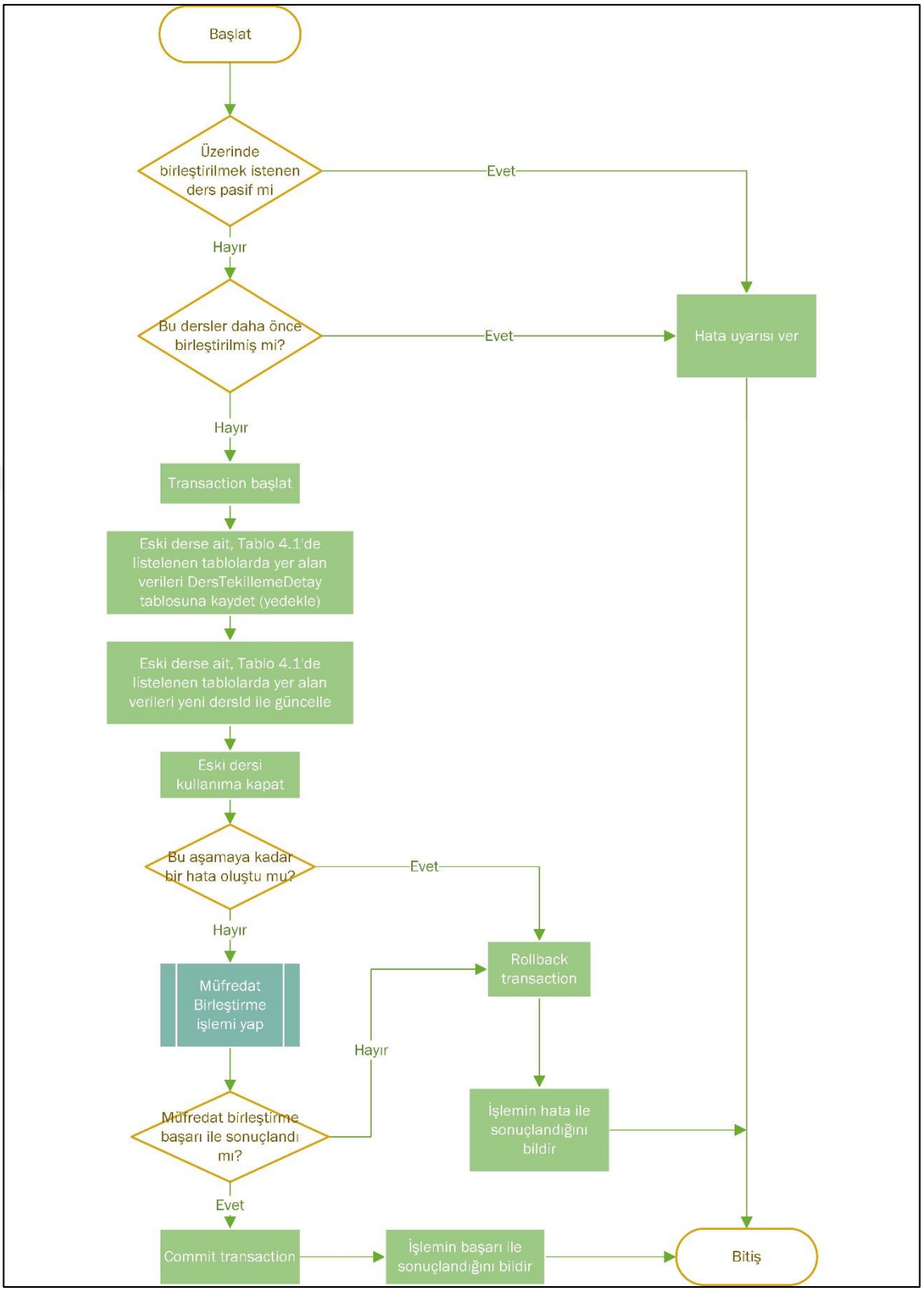
4.1.1. Ders Birleştirme Saklı Yordamı

Ders birleştirme saklı yordamı 3 parametre almaktadır:

- @YeniDersId: Tamsayı tipinde bir değişkendir. Üzerinde birleştirilecek olan dersin Id sidir.
- @KapanacakDersIdleri : Metin tipinde bir değişkendir. Birleştirilecek olan derslerden kapatılacak olanların Id leri, aralarında virgül olacak şekilde toplu olarak gönderilir.
- @UserId: Tamsayı tipinde bir değişkendir. İşlemi yapan kişinin kullanıcı Id sidir.

Saklı yordam, ilk olarak, gönderilen ders Id leri için gerekli kontrolleri yapar. Üzerine birleştirilecek olan ders aktif mi, kredi ile saat bilgileri uyuşuyor mu, kapanması için gönderilen derslerden hâlihazırda kapalı olan var mı gibi kontroller yapılır. Dersleri birleştirmeye engel bir durum yoksa işlem başlar. Tablo 4.1'de ismi geçen her tablonun belirtilen sütunlardaki verileri yeni ders_Id ile güncellenir. Güncelleme işleminden önce, yapılan birleştirme işleminin ileride geri alınmak istenmesi ihtimaline binaen, ilgili tablolardaki güncellenecek satırların yedeği DersTekillemeDetay tablosuna kaydedilir. Akabinde güncelleme yapılır.

Mevcut verilerin yedeklenmesi ve güncellenmesinin ardından, üzerinde birleştirilen dersin haricindeki dersler, daha sonra tekrar kullanılmaması için pasif duruma getirilir. Eğer bu noktaya kadar herhangi bir hata oluşur ve dolayısıyla saklı yordam birbiri ardına yapılan işlemlerden herhangi birini tamamlayamaz ise, veri bütünlüğü açısından ciddi sorun oluşacaktır. Bu yüzden tüm bu işlemler transaction blokları arasında yer alır. İşlemlerin bitiminde herhangi bir hata oluşup oluşmadığı kontrol edilir. Eğer hata oluşmuş ise "rollback" yapılır; yani transaction bloğu boyunca yapılmış olan işlemler geri alınır. Böylece veri tabanı, saklı yordamın çağrılmasından önceki haline geri döner. Hata oluşmamış ise son aşama olarak müfredat birleştirme işlemi gerçekleştirilir. Müfredat birleştirme işlemi de başarı ile sonuçlanırsa transaction bloğu "commit" edilir. Müfredat birleştirme işleminin başarı ile sonlanması durumu, ilgili saklı yordama gönderilen çıkış parametresi ile kontrol edilir. Müfredat birleştirme saklı yordamının akış diyagramı Şekil 4.3'te gösterilmiştir.



Şekil 4.2. Ders Birleştirme Saklı Yordamının Akış Diyagramı

Örnek olarak Tablo 3.5'te yer alan 3 'Temel Hukuk' dersini 9757 Id li ders üzerinde birleştirecek olan saklı yordam şu şekilde çağrılır:

```
Exec usp_admin_DersBirlestirme
  @YeniDersId = 9757,
      @Parametre = '27212,27928,',
      @UserId = 1
```

Yukarıdaki saklı yordam; 9757, 27212 ve 27928 Id li dersleri, 9757 Id li ders kalacak, diğerleri kapatılacak şekilde birleştirir. Bu işlemden sonra 27212 ve 27928 Id li dersler kullanıma kapatılmış olur.

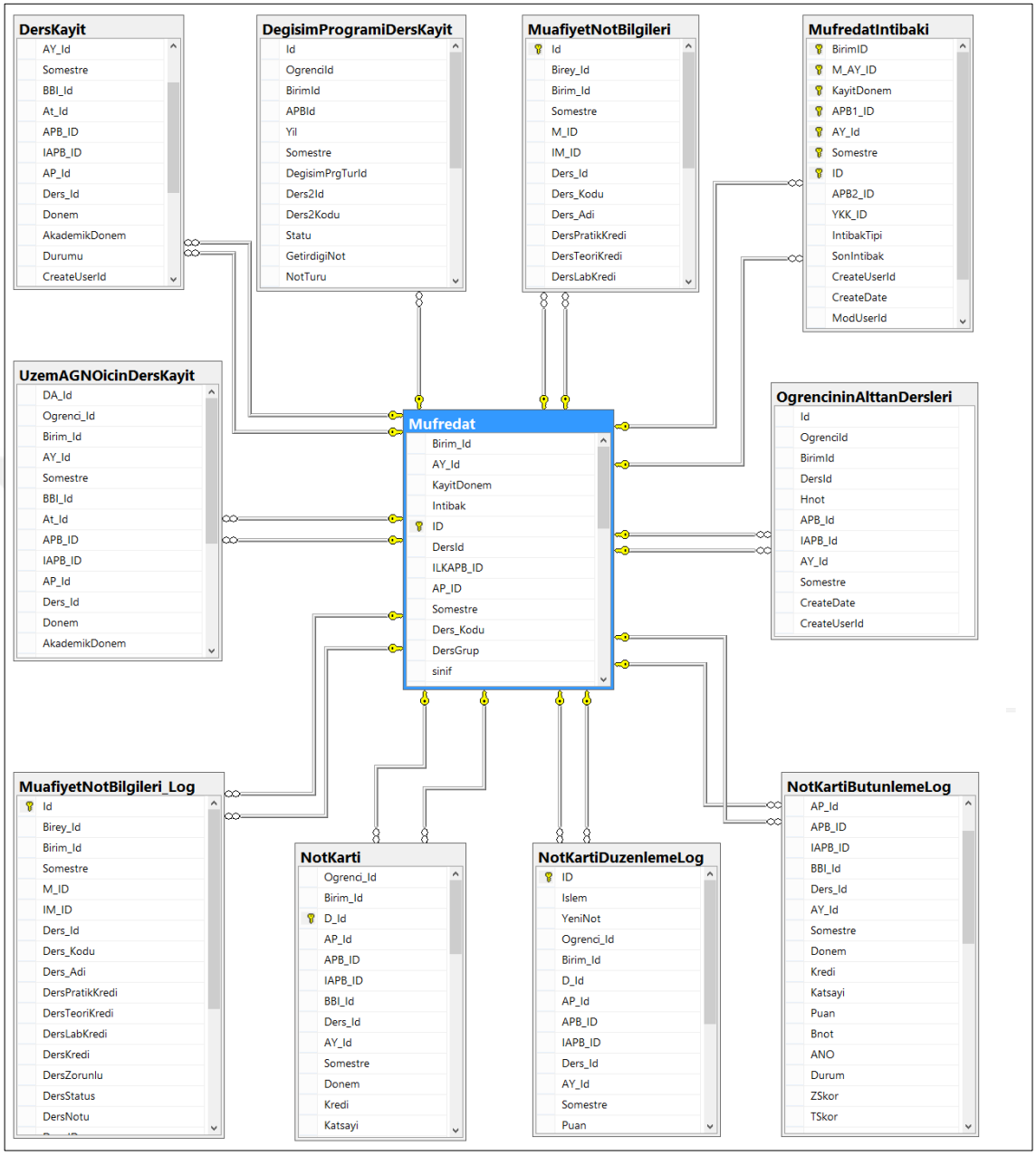
4.1.2. Müfredat Birleştirme Saklı Yordamı

Bir programın herhangi bir yılına ait müfredatta bir ders, sadece bir defa yer alabilir. Ders birleştirme işleminden sonra aynı dersin bir müfredatta birden fazla yer alıyor hale gelmesi muhtemeldir. Şekil 3.3'te transkripti gösterilmiş olan öğrenci İktisadi ve İdari Bilimler Fakültesi, İşletme Programı'nda okumakta olup "2009 - 2010 Akademik Yılı" müfredatına tabidir. Bu müfredatta hem 3661 Id li hem de 10711 Id li "İnsan Kaynakları Yönetimi" dersleri mevcuttur. Tüm "İnsan Kaynakları Yönetimi" derslerini, 3661 Id li ders üzerinde birleştirme işlemi yapıldıktan sonra, bu müfredattaki 10711 Id li ders güncellenip 3661 olacaktır. Dolayısıyla 3661 Id li ders, bu müfredatta iki defa yer almış olacaktır. Bir programın herhangi bir yılına ait müfredatta bir dersin sadece bir defa yer alabilmesi kuralı ihlal edilmiş olacağı için, ders birleştirme işleminin bir benzeri olarak müfredat birleştirme işleminin yapılması gerekecektir.

Ders birleştirme işlemi için, veri tabanında yer alan diğer tablolardaki, dersler tablosunun DersId sütununa refere eden sütunlar belirlendiği gibi; müfredat birleştirme için de aynı şekilde Müfredat tablosunun Id sütununa refere eden tablolar ve ilgili sütunları belirlendi. Bu tablolar Tablo 4.2'de listelenmiştir. Müfredat bilgilerinin tutulduğu Mufredat tablosu ve bu Mufredat ile ilişkili olan tabloların ilişki diyagramı Şekil 4.2'de gösterilmiştir.

Tablo 4.2. Müfredat Id ye Refere Eden Sütunlar ve Bu Sütunların Yer Aldıkları Tablolar

	TabloAdi	**KolonAdi**
1	NotKarti	APB_ID
2	NotKarti	IAPB_ID
3	NotKartiButunlemeLog	APB_ID
4	NotKartiButunlemeLog	IAPB_ID
5	DersKayit	APB_ID
6	DersKayit	IAPB_ID
7	MufredatIntibaki	APB1_ID
8	MufredatIntibaki	APB2_ID
9	NotKartiDuzenlemeLog	APB_ID
10	NotKartiDuzenlemeLog	IAPB_ID
11	OgrencininAlttanDersleri	APB_Id
12	OgrencininAlttanDersleri	IAPB_Id
13	UzemAGNOicinDersKayit	APB_ID
14	UzemAGNOicinDersKayit	IAPB_ID
15	OgrencininAlttanDersleri_AOF	APB_Id
16	OgrencininAlttanDersleri_AOF	IAPB_Id
17	DegisimProgramiDersKayit	APBID
18	MuafiyetNotBilgileri	M_ID
19	MuafiyetNotBilgileri	IM_ID
18	MuafiyetNotBilgileri_Log	M_ID
19	MuafiyetNotBilgileri_Log	IM_ID



Şekil 4.3. Müfredat Tablosu ile İlişkili Diğer Tablolar

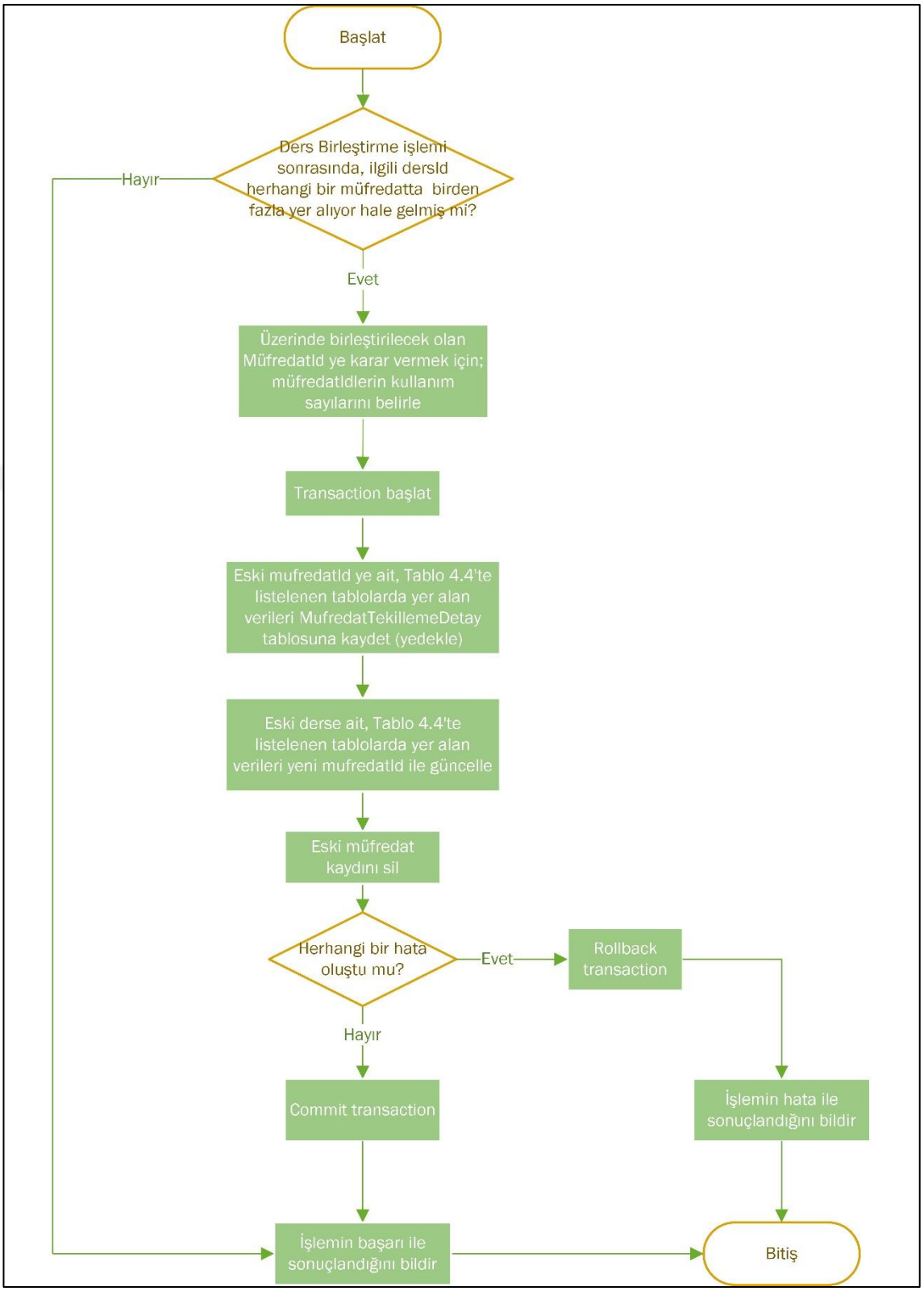
Müfredat birleştirme saklı yordamı, ders birleştirme saklı yordamı tarafından çağrılmaktadır. 3 Parametre almaktadır:

- @BirlestirilmisDersId: Tamsayı tipinde bir değişkendir. Bu yordamı çağıran ders birleştirme saklı yordamına gönderilip üzerinde birleştirme yapılmış dersId dir.

- @UserId: Tamsayı tipinde bir parametredir. İşlemi yapan kişinin kullanıcı Id sidir.
- @BasariliMi: Tamsayı tipinde bir çıkış parametresidir. Bu saklı yordamın başarı ile sonuçlanıp sonuçlanmadığı bilgisini, kendisini çağıran ders birleştirme saklı yordamına döndürür. Ders birleştirme saklı yordamı, kendi içindeki transaction bloğunu, bu parametreden gelen değere göre commit ya da rollback yapar.

Müfredat Birleştirme saklı yordamı, gönderilen dersId nin, herhangi bir müfredatta birden çok defa mevcut olup olmadığını kontrol eder; böyle bir durum yok ise sonlanır; var ise ders birleştirme işlemine benzer işlemler yapar. Öncelikle çoğullamış olan müfredat Id lerinden en çok kullanılmış olanı tespit eder; çünkü birleştirme işlemi bu kayıt üzerinde yapılacaktır.

Tablo 4.2'de listelenen tabloların ilgili sütunları yeni müfredat Id ile güncellenir. Güncelleme işleminden önce, güncellenecek olan verilerin yedeği MufredatTekillemeDetay tablosuna kaydedilir. Akabinde güncelleme işlemi yapılır. Ders birleştirme saklı yordamında olduğu gibi; bu saklı yordam da transaction blokları arasında yer alır. Herhangi bir sorun oluşursa rollback yapılarak tüm işlemler geri alınır; yordamın başarısızlıkla sonuçlandığı bilgisi, kendini çağıran ders birleştirme saklı yordamına @BasariliMi parametresi üzerinden gönderilir. Herhangi bir sorun yaşanmazsa transaction bloğu commit edilir ve kendisini çağıran yordama işlemin başarı ile sonuçlandığı bilgisi @BasariliMi parametresi üzerinden gönderilir. Müfredat birleştirme saklı yordamının akış diyagramı Şekil 4.4'te gösterilmiştir.



Şekil 4.4. Müfredat Birleştirme Saklı Yordamının Akış Diyagramı

4.1.3. Ders Birleştirmeyi Geri Alma Saklı Yordamı

Yapılan bir ders birleştirme işlemi, yanlış derslerin birleştirilmesi gibi sebeplerden dolayı geri alınmak istenebilir. Bu durumda ders birleştirmeyi geri alma saklı yordamı kullanılır. Bu yordam 3 parametre almaktadır:

- @YeniDersId: Tamsayı tipinde bir parametredir. Üzerinde birleştirilmiş olan dersin Id sidir.
- @EskiDersId: Tamsayı tipinde bir parametredir. @YeniDersId parametresinde gönderilen ders ile birleştirilerek pasif yapılmış bir dersin Id sidir.
- @UserId: Tamsayı tipinde bir değişkendir. İşlemi yapan kişinin kullanıcı Id sidir.

İlk olarak kapatılmış yani pasif hale getirilmiş olan ders yeniden aktif hale getirilir. Dersler birleştirilirken DersTekillemeDetay tablosuna kaydedilmiş olan yedekler vasıtasıyla tablo 4.1'de listelenen tablolar güncellenir. Bu işlemler transaction blokları içerisinde yapılır.

Örneğin daha önce 10064 Id li "Yabancı Dil I" dersi ile 35186 id li "Yabancı Dil I (İngilizce)" dersleri birleştirilmiş olup bu birleştirme yapılırken DersTekillemeDetay tablosuna kaydedilmiş verinin örnek bir kesiti Tablo 4.3'te gösterilmiştir.

Tablo 4.3. DersTekillemeDetay Tablosuna Örnek Bir Kesit

EskiDersId	**YeniDersId**	**TabloAdi**	**KolonAdi**	**TablodakiID**	**IdentityKolonAdi**
35186	10064	AcilanDersler	DersId	505651	ID
35186	10064	Mufredat	DersId	470069	ID
35186	10064	NotKarti	Ders_Id	7492192	D_Id
35186	10064	NotKartiButunlemeLog	Ders_Id	9197029	D_Id
35186	10064	DersKayit	Ders_Id	4935438	DA_Id
35186	10064	NotKartiDuzenlemeLog	Ders_Id	4315766	Id
35186	10064	OgrencininAlttanDersleri	DersId	191740	Id
35186	10064	Sinav	Ders_Id	174653	Sinav_Id

Geri alma işlemine başlamadan önce, müfredat birleştirmeyi geri alma işlemi yapılır; ondan sonra ders birleştirme işlemi geri alınır.

4.1.4. Müfredat Birleştirmeyi Geri Alma Saklı Yordamı

Ders birleştirme işlemi geri alınacağı zaman, bu işlemin sağlıklı olabilmesi için, öncelikle ders birleştirme işleminin akabinde yapılmış olan müfredat birleştirme işlemi geri alınması gerekir. Bu durumda müfredat birleştirmeyi geri alma saklı yordamı kullanılır. Bu saklı yordam 4 parametre almaktadır:

- @KalanDersId: Tamsayı tipinde bir parametredir. Üzerinde birleştirilmiş olan dersin Id sidir.
- @EskiDersId: Tamsayı tipinde bir parametredir. @KalanDersId parametresinde gönderilen ders ile birleştirilerek pasif yapılmış bir dersin Id sidir.
- @BasariliMi: Tamsayı tipinde bir çıkış parametresidir. Bu saklı yordamın başarı ile sonuçlanıp sonuçlanmadığı bilgisini, kendisini çağıran ders birleştirmeyi geri alma saklı yordamına döndürür. Ders birleştirmeyi geri alma saklı yordamı, kendi içindeki transaction bloğunu, bu parametreden gelen değere göre rollback yapar ya da işleme devam eder.
- @UserId: Tamsayı tipinde bir değişkendir. İşlemi yapan kişinin kullanıcı Id sidir.

Parametrede gönderilen dersler birleştirilirken müfredat birleştirme yapılması sonucu silinen müfredatların olup olmadığı tespit edilir. Silinen müfredat yoksa @BasariliMi parametresine 1 değeri atanarak yordam sonlanır. Silinen müfredat var ise MufredatTekillemeDetay tablosundan yola çıkarak gerekli güncellemeler yapılır. Bu güncellemeler transaction blokları arasında yer alır. Herhangi bir sorun oluşursa rollback yapılarak tüm işlemler geri alınır; yordamın başarısızlıkla sonuçlandığı bilgisi, kendini çağıran ders birleştirmeyi geri alma saklı yordamına @BasariliMi parametresi üzerinden gönderilir. Herhangi bir sorun yaşanmazsa transaction bloğu commit edilir ve kendisini çağıran yordama işlemin başarı ile sonuçlandığı bilgisi @BasariliMi parametresi üzerinden gönderilir.

4.2. BENZER İSİMLİ DERSLERİ TESPİT ETME

Ders birleştirme işlemi aynı isimli dersler arasında yapılabileceği gibi, ismi aynı olmayan dersler üzerinde de yapılabilir. “Mikro İktisat”, “MikroİktisatMikro İktisat” ve “Mikro Iktisat” şeklinde yazılışları farklı olan dersler aslında aynı derslerdir. Hatta “Mikro Ekonomi” dersi de yukarıdaki dersler ile aynı derstir. Dolayısı ile birleştirme işleminin yazımı aynı olmayan dersler üzerinde de gerçekleştirilebilmesi gerekmektedir.

Aynı isimli dersler, basit SQL sorguları ile tespit edilebilir; fakat yukarıda örnekleri verilen “Mikro İktisat” dersinin çeşitli versiyonları gibi yazımı aynı olmayan dersleri tespit etmek, basit sorgular ile mümkün değildir. Bu sebeple yazımı farklı olduğu halde aynı ders olma ihtimali olan dersleri tespit edebilmek için dizge karşılaştırma algoritmalarından “Damerau–Levenshtein Distance” algoritması kullanıldı. Bu algoritma için parametre olarak iki metinsel ifade alan ve sonuç olarak bu iki ifadenin DLU değerini geri döndüren bir fonksiyon yazıldı. Bu fonksiyonu kullanan saklı yordamlar vasıtasıyla tüm ders adları, diğer ders adları ile karşılaştırılarak her bir dersin diğer dersler ile olan Damerau–Levenshtein uzaklığı (DLU) hesaplandı. DLU değeri düşük olan, yani yazılışları birbirine benzeyen dersler tespit edildi.

4.2.1. Damerau-Levenshtein Uzaklık Hesabı Yapan Fonksiyon

Damerau-Levenshtein uzaklık hesabı yapan bu kullanıcı tanımlı fonksiyon, @Ifade1, @Ifade2 isimli parametreleri ile iki metinsel ifade almakta ve sonuç olarak, parametre ile aldığı ifadelerin birbirine olan Damerau-Levenshtein Uzaklığı değerini döndürmektedir. Fonksiyondan dönen değer, bir ifadenin diğerine dönüşmesi için gereken; ekleme, silme, değiştirme ve yer değiştirme işlemlerinin sayısıdır. Her iki ifade aynı ise, herhangi bir işlem yapmaya gerek olmayacağı için sonuç olarak 0 dönülecektir. Fonksiyonun dönebileceği en küçük değer 0; en büyük değer, iki ifadeden uzun olanının karakter sayısıdır. “Matematik I” ifadesi ile “Matamatik I” ifadesini karşılaştırmak için fonksiyon şu şekilde çağrılmalıdır:

```
Select dbo.fn_DamerauLevenshteinDistance(‘Matematik I’, ‘Matamatik I’)
```

Bu şekilde çağrılan fonksiyon 1 dönecektir.

Fonksiyonun çalışma süresini kısaltmak amacıyla, fonksiyona @limit isimli tamsayı tipinde üçüncü bir parametre daha eklenmiştir. Fonksiyon, çalışması esnasında, her bir adımda, o adımdaki mevcut uzaklık değerini limit değeri ile karşılaştırarak; bu değerin aşıldığı anda fonksiyonun sonlanması şeklinde değiştirilmiştir. Bu sayede fonksiyon, hesaplamaya devam etmenin gereksiz olduğunun anlaşılması halinde sonlanarak daha hızlı sonuç döndürmeye başlamıştır.

4.2.2. Bir Dersi Diğer Derslerle Karşılaştırma Saklı Yordamı

Bu saklı yordam parametre olarak dersId alır. Aldığı Id li ders ile karşılaştırılacak olan dersleri tespit edip geçici bir tabloya doldurur. Parametre ile gelen dersi; döngü vasıtasıyla, geçici tablodaki dersler ile tek tek karşılaştırarak DL uzaklığını hesaplar ve sonucu DersKarsilastirma (DK) tablosuna yazar.

4.2.3. Tüm Dersleri Diğer Dersler ile Karşılaştırma Saklı Yordamı

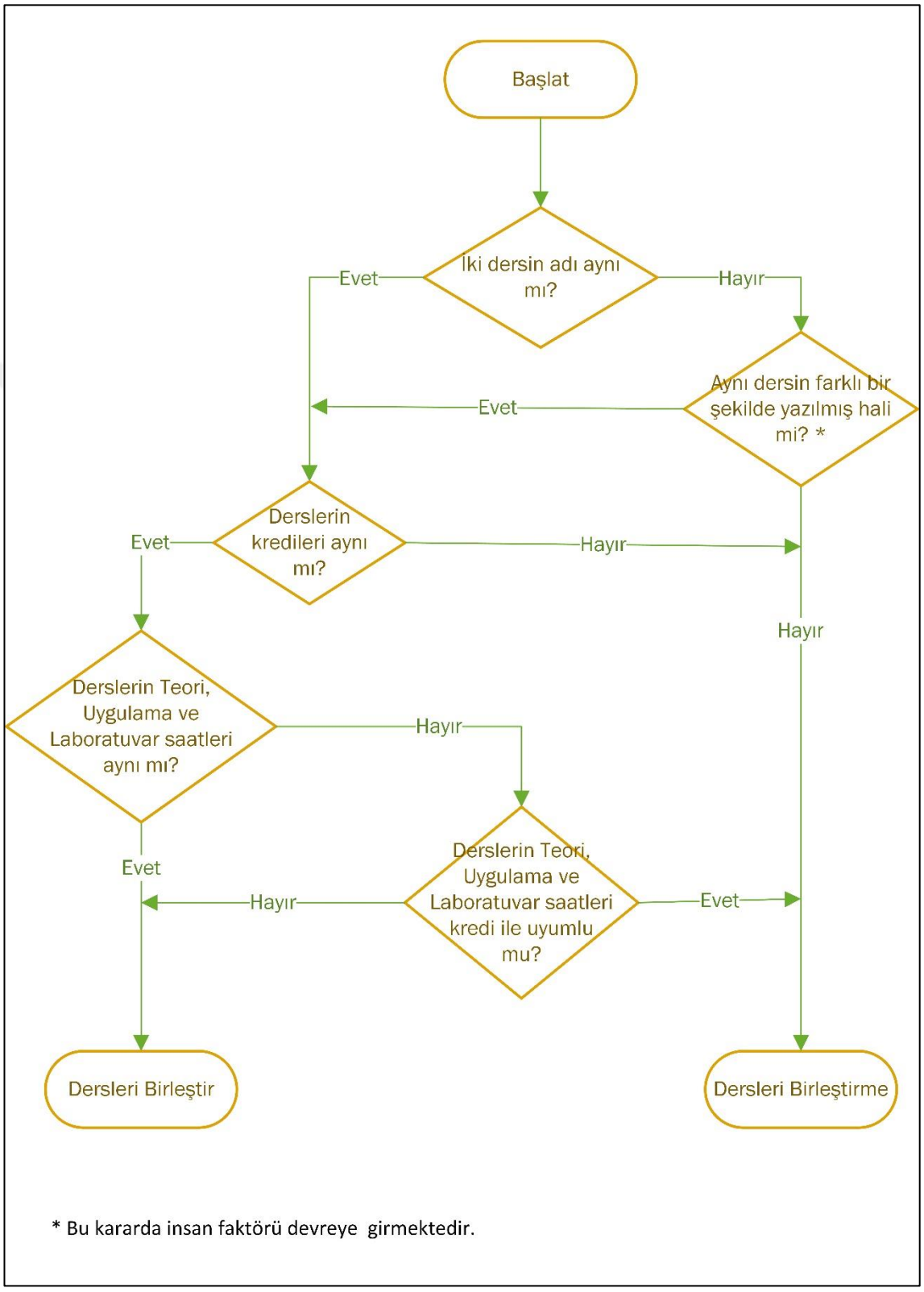
Bu saklı yordam herhangi bir parametre almadan, döngü vasıtasıyla, en küçük Id li dersten başlayarak, sırayla her bir ders için, bir dersi diğer tüm dersler ile karşılaştıran saklı yordamı çağırır. Bu saklı yordamın çalışması bittiğinde tüm dersler, diğer dersler ile karşılaştırılmış olur.

Bu saklı yordam, daha sonra asenkron çalıştırılabilmesi için parametrik hale getirilmiş ve iki parametre almaya başlamıştır. Birinci parametre, karşılaştırma yapmaya hangi dersten başlayacağını, ikinci parametre ise kaç ders için işlem yaptıktan sonra sonlanacağını ifade etmektedir. Böylece aynı anda, aynı saklı yordam, SQL Server Management Studio (SSMS) ortamında, farklı sayfalarda farklı parametreler ile çağrılarak tüm karşılaştırma işlemlerinin süresi kısaltılmıştır.

4.3. DERSLERİ BİRLEŞTİRME

Herhangi iki veya daha fazla dersi birleştirmek için derslerin adı ya aynı olmalı ya da yazımı farklı olsa da aynı manayı ifade etmesi gereklidir. Derslerin kredi saati; teori (T), uygulama (U) ve laboratuvar (L) saatleri de aynı olması gereklidir. Bazı derslerin kredisi ile T,U,L saatleri uyumlu değildir. Bu durumdaki T,U,L saat bilgileri hatalı olan

dersler birleştirilirken T,U ve L saatlerinin aynı olması zorunluluğu yoktur. Ders Birleştirme karar ağacı Şekil 4.5'te görülmektedir.



Şekil 4.5. Ders Birleştirme Karar Ağacı

4.3.1. Ön Hazırlık

Veri tabanında, Dersler tablosunda 38,156 adet ders bulunmaktadır. 38,156 adet dersi diğer 38,155 ders ile karşılaştırmak demek (38156 * 38155) / 2 = 727,921,090 defa DLD fonksiyonunu çalıştırmak demektir ki; bu sayıda karşılaştırma yapmak veri tabanı sunucusuna ciddi bir işlemci yükü bindirmek ve aynı zamanda tüm hesaplamaların bitmesi için çok uzun süre beklemek demektir. Dolayısı ile yapılacak olan karşılaştırma sayısının olabildiğince azaltılması gerekmektedir.

4.3.2. Toplam Karşılaştırma Sayısını Azaltmak İçin Yapılan İşlemler

Tüm derslerin kullanım geçmişleri incelenerek hiç kullanılmadığı tespit edilen 4,526 adet ders, direk veri tabanından silindi.

Kullanıma kapatılmış olan 330 adet ders, karşılaştırma listesinden çıkarıldı.

3 kredilik “Matematik I” dersinden 15 tane olduğu belirtilmişti. Farklı kredili dersler de dâhil edilince bu sayı 25 olmaktadır. Örneğin 5070 Id li “Matematik I” dersini diğer tüm dersler ile karşılaştırdıktan sonra, 5757 Id li “Matematik I” dersi için bu işlemi tekrar etmeye gerek yoktur. Çünkü 5070 Id li ders için elde edilen DLU değerleri tüm “Matematik I” dersleri için geçerlidir. Toplu karşılaştırma işlemi yapılırken yazılışı aynı olan derslerden sadece bir tanesi kalacak şekilde diğerleri listeden çıkarıldı.

Yukarıdaki işlemlerden sonra karşılaştırma listesinde 23,464 adet ders kaldı.

Ders isimleri genel bir şekilde gözden geçirildi. Fark edilebilen belirgin yazım hataları düzeltildi. Ders isimleri için bazı standartlar belirlendi. Örneğin birbirinin devamı niteliğindeki derslerde yer alan numaralar için Roma rakamları kullanılması ve rakamdan önce ‘-’ karakteri kullanılmaması şeklinde standartlar belirlendi ve mevcut dersler bu standartlara uyarlandı. Bu işlemden sonra 25 olan “Matematik I” dersi sayısı, 29 oldu. Ders isimlerinde yapılan kontrol ve düzeltmelerin bir kısmı aşağıda sıralanmıştır:

- Başında ve sonundaki boşluklar temizlendi.
- Çift boşluk karakterleri tek boşluk ile değiştirildi.

- Numaralı derslerde standart belirlenip buna göre düzenleme yapıldı.
- –, /, *, ? gibi karakter içeren dersler gözden geçirildi.
- Dersin adı ile alakası olmadığı halde adının sonuna parantez içerisinde eklenmiş olan (İngilizce), (erkek), (kız), (yıllık), (özel), (eski), (eczacılık), (doktora), (Tekrar), (A), (B), (Seçmeli), (S), (Zorunlu), (Z), (staj) vb. ifadeler dersin adından çıkarıldı.
- Parantez, virgül, nokta, üst üste iki nokta, noktalı virgül vb. işaretlerinden önce boşluk olması veya sonra boşluk olmaması gibi yazım hataları düzeltildi.
- Yan yana üç ünlü ve yan yana üç ünsüz harf geçen ders adları kontrol edildi.
- Çok uzun kelimeye sahip ders adları kontrol edildi.
- Birebir bakıldığında aynı olmayıp, içinden boşluklar ve noktalama işaretleri çıkarıldığında aynı olan dersler tespit edildi ve yazımı hatalı olan dersler düzeltildi. (“Makroiktisat” – “Makro İktisat” veyahut “İnsan, Toplum, Bilim” - “İnsan Toplum Bilim” gibi dersler bu sayede tespit edildi.)
- ‘Laboratuvar’, ‘makine’ gibi yazımında sıklıkla hata yapılan kelime içeren dersler kontrol edildi.

Bu kontrol ve düzeltmelerden sonra karşılaştırma listesindeki ders sayısı biraz daha azalmış ve 23,044 olmuştur. Şu hali ile toplam karşılaştırma sayısı (23044 * 23043) / 2 = 265,501,446 olmuştur.

Tüm karşılaştırma işlemleri bittiğinde, her bir ders için, benzerlik oranı %90 ve üzeri olan dersler ile aynı olma ihtimali değerlendirilecektir. Bu yüzden 7 karakterden oluşan “Kimya I” dersi ile 38 karakterden oluşan “Sosyal Bilimlerde Araştırma Yöntemleri” dersini karşılaştırmak, pek anlamlı olmayacaktır. Dolayısıyla her ders, diğer tüm derslerle değil; kendi uzunluğunun %10 fazlası ya da eksiği uzunluğundaki dersler ile karşılaştırıldı. Yani 38 karakter uzunluğundaki “Sosyal Bilimlerde Araştırma Yöntemleri” dersi, tüm diğer 23043 ders ile değil; sadece uzunluğu kendi uzunluğu ile %10 fark edecek olan 34-42 arası karakter uzunluğuna sahip 3729 ders ile karşılaştırıldı. Ders adlarının uzunluğuna göre yapılan bu filtrelemenin ardından, yapılan karşılaştırma

sayısı 51,904,774 oldu. Karşılaştırma sayısı, 727,921,090 dan 51,904,774 a; yani ilk sayının % 7 sine düşürülerek iş yükü ciddi bir şekilde azaltılmıştır.

4.3.3. Karşılaştırma Süresini Kısaltmak İçin Yapılan İşlemler

Benzerlik oranı %90 ve üzeri olan derslerin birbirleri ile aynı olma ihtimalinin değerlendirileceğinden yukarıda bahsedilmişti. Dolayısıyla 38 karakterden oluşan "Sosyal Bilimlerde Araştırma Yöntemleri" dersinin, DL uzaklığı en fazla 4 olan dersler ile aynı olma ihtimali dikkate alınacaktır. Bu yüzden uzaklık hesabı yapan fonksiyona limit değerini ifade eden üçüncü bir parametre daha eklendi. Sonuç değeri, bu limit değerini aşacağı belli olduğu anda fonksiyonun hesaplamaya devam etmeyip sonlanması sağlandı. Bu sayede fonksiyonun çalışma süresi kısaltıldı. İlk durumda "Sosyal Bilimlerde Araştırma Yöntemleri" dersini "Tarih Metodolojisi ve Bibliografya" dersi ile karşılaştırıldığında sonuç 35 çıkmakta idi. Fonksiyon limit değeri 4 olacak şekilde çağrıldığında ise, sonucun 4 den büyük olacağı anlaşıldığı anda fonksiyon sonlanacak ve -1 dönecektir. Limit değerini eklemek, fonksiyonun yapısını bozmamıştır. Limit değeri, -1 verilerek fonksiyon çağrıldığında, gerçek DLU değeri hesaplanmaktadır. Fonksiyon "Sosyal Bilimlerde Araştırma Yöntemleri" ve "Tarih Metodolojisi ve Bibliografya" ifadeleri için; limit değeri -1 olarak çağrıldığında 35, limit değeri 4 olarak çağrıldığında -1 sonucunu döndürmektedir. Fonksiyon çalışma süresi; limit değeri -1 olarak çağrıldığında 466 milisaniye iken, limit değeri 4 olarak çağrıldığında 62 milisaniye olmaktadır. Bu örnek için fonksiyon 7 kat hızlanmıştır.

Bir döngü vasıtasıyla, sırayla her bir dersin, diğer derslerle karşılaştırıldığı saklı yordam, asenkron çalıştırılabilmesi için parametrik hale getirildi ve iki parametre almaya başladı. Birinci parametre, karşılaştırma yapmaya hangi dersten başlayacağını, ikinci parametre ise, kaç ders için işlem yaptıktan sonra sonlanacağını ifade etmektedir. Böylece aynı anda, aynı saklı yordam, SSMS'da farklı sayfalarda, farklı parametreler ile çağrılmıştır. İşlem, paralel programlama mantığına yaklaşarak, parçalara bölünmüştür. Farklı sayfalarda aynı saklı yordam, değişik parametreler ile toplam 93 defa çağrılmıştır. Bu şekilde çağrılan saklı yordamlar Tablo 4.4'te görülmektedir. Bu esnada veri tabanı sunucusunun işlemcisine ciddi yük binmiştir; fakat tüm derslerin karşılaştırılması süresi de aynı oranda kısalmıştır. Bu işlem mesai bitiminden sonra başlatılmış olduğu ve ertesi gün mesai başlamadan büyük ölçüde bitmiş olduğu için

Öğrenci Bilgi Sisteminin işlerliğinde herhangi bir aksama yaşanmamıştır. Tüm dersleri, diğerleri ile karşılaştırmak başlangıçta 6 gün sürerken; asenkron hale getirdikten sonra 13 saatte bitmiştir. Yaklaşık 12 katlık bir hızlanma gerçekleşmiştir.

Tablo 4.4. Farklı Parametre ile Asenkron Olarak Çağrılan Saklı Yordamlar Listesi

Sorgu İfadesi
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 1, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 251, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 501, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 751, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 1001, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 1251, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 1501, @Aralik = 250
…
…
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 22751, @Aralik = 250
exec usp_admin_DersBenzerlikTesbit_TumDersler @MinDersId = 23001, @Aralik = 250

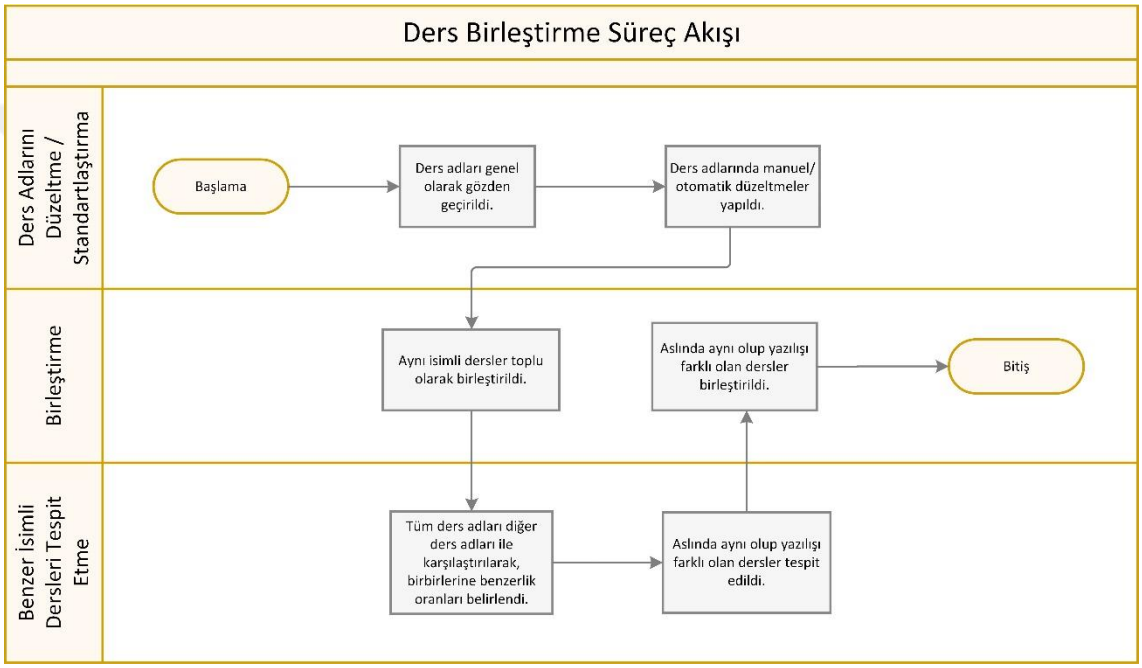
Tüm bu optimizasyon işlemlerinden sonra toplu karşılaştırma işlemi yapıldı ve birbirine benzerliği %90 ve üzeri olan ders kombinasyonları belirlendi. Bu liste gözden geçirilerek birbiri ile aynı olduğuna kesin kanaat getirilen ders kombinasyonları birleştirildi. Diğer kombinasyonlar değerlendirilmek üzere Öğrenci İşleri Daire Başkanlığı'na gönderildi.

4.3.4. Sistemin Uygulanması

Kodlar tekrar tekrar test edilmiş olsa da, bir aksilik çıkma ihtimaline binaen, eylem başlangıçta pilot olarak uygulandı. AGNO'yu etkilememesi ve en çok ders çoğullamasının da ortak zorunlu dersler içerisinde olması sebebi ile pilot uygulama olarak sadece ortak zorunlu dersler "Atatürk İlkeleri ve İnkılap Tarihi", "Yabancı Dil", "İngilizce", "Türk Dili", "Hukukun Temel Kavramları" birleştirildi. Öğrencilerden veya Öğrenci İşleri personeli tarafından herhangi bir hata bildirimi gelmesi beklendi. Geri

dönüşler içerisinde bir hata bildirimi olmaması sebebi ile sistemin sağlıklı çalıştığına kanaat getirildi ve "Ders birleştirme" işlemi test veri tabanında diğer dersler üzerinde de uygulandı.

Öncelikle yazılışı, teori saati, uygulama saati, laboratuvar saati ve kredisi aynı olan dersler gruplandı. Her grubun içinde, en çok kullanılmış olan ders belirlendi ve diğer dersler bu ders üzerinde birleştirildi. Daha sonra yazılışı aynı olmayıp diğer bilgileri aynı olan fakat aynı ders olduğu bariz belli olan bazı dersler birleştirildi. Ders Birleştirme sürecinin ilerleyişi Şekil 4.6'da gösterilmiştir.



Şekil 4.6. Ders Birleştirme Süreç Akışı

Yazımı aynı olmayıp fakat aynı ders olduğuna kanaat getirilip birleştirilen örnek bazı dersler Tablo 4.5'te gösterilmiştir.

Tablo 4.5. Yazımı Aynı Olmayıp Birleştirilen Bazı Dersler

Ders1Adi	**Ders2Adi**	**DLD**
Ağız Diş ve Çene Cerrahisi Stajı	Ağız, Diş ve Çene Cerrahisi Stajı	1
ANA ÇOCUK BESLENMESİ	Anne-Çocuk Beslenmesi	3
ANA ÇOCUK BESLENMESİ	Anne Çocuk Beslenmesi	2
Anorganik Kimya Labaratuvarı II	Anorganik Kimya Laboratuvarı II	1
Arap Grameri II	Arapça Gramer II	3
Besinlerde Antimikrobiyel Maddeler	Besinlerde Antimikrobiyal Maddeler	1
Bilgi İletişim Teknolojileri I	Bilgi ve İletişimTeknolojileri I	4
Bilimsel Araştıma Metodları II	Bilimsel Araştırma Metotları II	2
Biyoistatistik	Bioistatistik	1
Çaşdaş İslam Dünyası Tarihi	Çağdaş İslam Dünyası Tarihi	1
Edebiyat Eleştirisi	Edebiyat Eleştrisi	1
Ekonomik Faaliyetler II	Ekonomik Faliyetler II	1
Elektrik Makineleri III	Elektrik Makinaları III	3
Elektromagnetik Teorisi	Elektro Manyetik Teorisi	3
Gıda Katkı Madde ve Toksikoloji	Gıda Katkı Maddeleri ve Toksikoloji	4

"Makro İktisat" dersini 3 defa alıp; her seferinde farklı yazılmış ders adına sahip dersleri alan örnek bir öğrencinin transkriptinin ders birleştirme işleminden önceki hali Şekil 4.7'de ve birleştirme işleminden sonraki hali de Şekil 4.8'de gösterilmiştir.

Kodu	Ders Adı	TS	US	LS	Kredi	Notu
2010 - 2011 Akademik Yılı Güz Yarıyılı						
YNO 505	Stratejik Yönetim ve Rekabet Avantajı (21810)	3.0	0.0	0.0	3.0	BB
FBEB 512	C++ İle Nesne Tabanlı Programlama (27514)	3.0	0.0	0.0	3.0	AA
İKP 501	Makro iktisat I (23125)	3.0	0.0	0.0	3.0	FF
FBEB 518	C# İle Görsel Windows Programlama (27572)	3.0	0.0	0.0	3.0	CB
					Kredi	AGNO
		Genel			12.00	2.38
2010 - 2011 Akademik Yılı Bahar Yarıyılı						
SAY505	İşletmelerde Veri İletişimi (22382)	3.0	0.0	0.0	3.0	AA
SAY555	Elektronik Devlet (22240)	3.0	0.0	0.0	3.0	BB
İKP 501	Makro İktisat I (24076)	3.0	0.0	0.0	3.0	FF
EKO 519	Yöneylem Araştırması I (26745)	3.0	0.0	0.0	3.0	AA
SAY501	Uygulamalı İstatistik I (22864)	3.0	0.0	0.0	3.0	AA
					Kredi	AGNO
		Genel			27.00	2.72
2011 - 2012 Akademik Yılı Güz Yarıyılı						
EKOBHP101	Makro iktisat I (28216)	3.0	0.0	0.0	3.0	CC
					Kredi	AGNO
		Genel			30.00	2.65

Şekil 4.7. "Ders Birleştirme" İşleminden Önceki Transkript

Kodu	Ders Adı	TS	US	LS	Kredi	Notu
2010 - 2011 Akademik Yılı Güz Yarıyılı						
YNO 505	Stratejik Yönetim ve Rekabet Avantajı (21810)	3.0	0.0	0.0	3.0	BB
FBEB 512	C++ İle Nesne Tabanlı Programlama (27514)	3.0	0.0	0.0	3.0	AA
İKP 501	Makro iktisat I (23125)	3.0	0.0	0.0	3.0	FF
FBEB 518	C# İle Görsel Windows Programlama (27572)	3.0	0.0	0.0	3.0	CB
					Kredi	AGNO
		Genel			12.00	2.38
2010 - 2011 Akademik Yılı Bahar Yarıyılı						
SAY505	İşletmelerde Veri İletişimi (22382)	3.0	0.0	0.0	3.0	AA
SAY555	Elektronik Devlet (22240)	3.0	0.0	0.0	3.0	BB
İKP 501	Makro İktisat I (23125)	3.0	0.0	0.0	3.0	FF
EKO 519	Yöneylem Araştırması I (26745)	3.0	0.0	0.0	3.0	AA
SAY501	Uygulamalı İstatistik I (22864)	3.0	0.0	0.0	3.0	AA
					Kredi	AGNO
		Genel			24.00	3.06
2011 - 2012 Akademik Yılı Güz Yarıyılı						
EKOBHP101	Makro iktisat I (23125)	3.0	0.0	0.0	3.0	CC
					Kredi	AGNO
		Genel			24.00	3.31

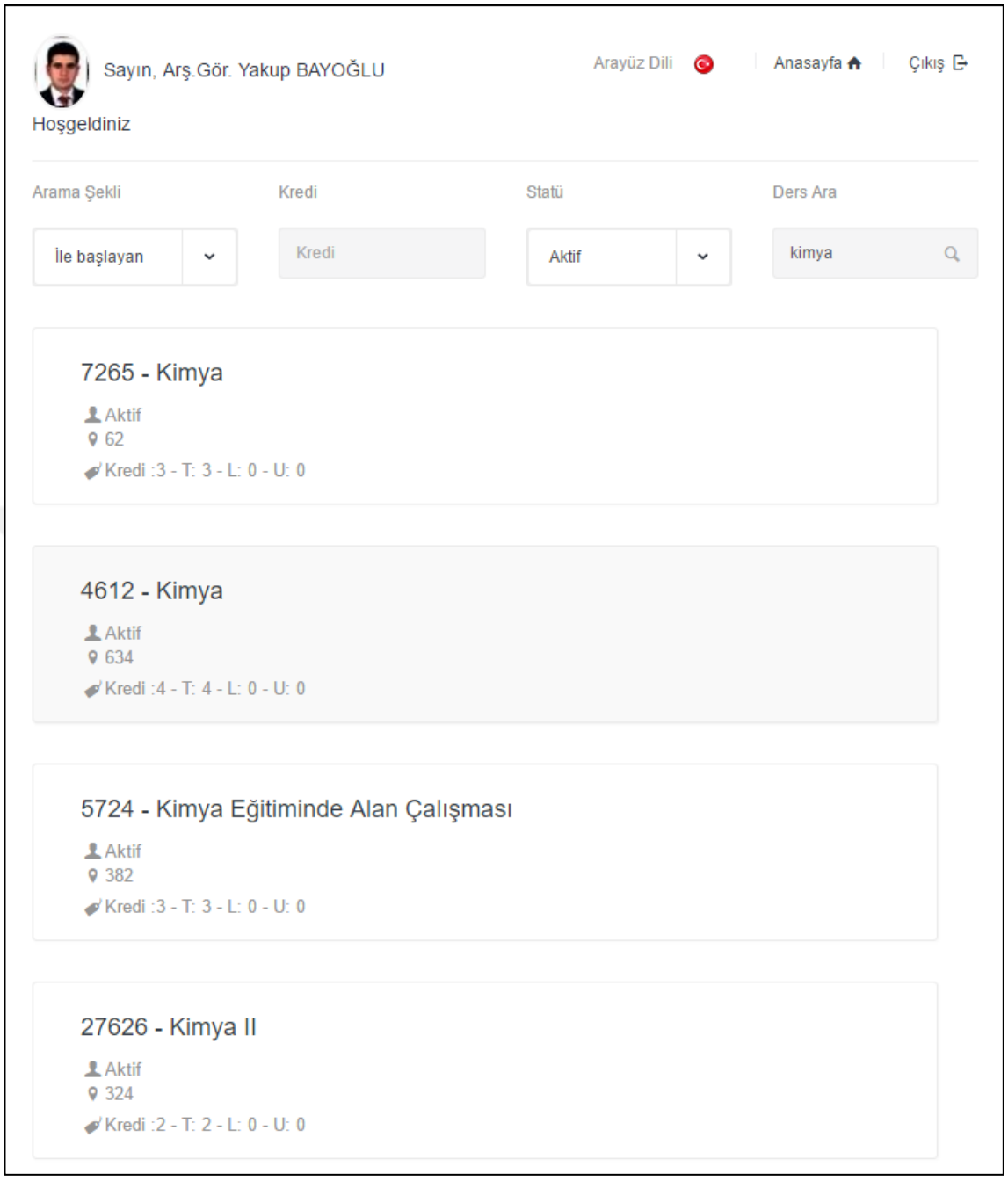
Şekil 4.8. "Ders Birleştirme" İşleminden Sonraki Transkript

Ders birleştirme işleminden önce, alınmış her "Makro İktisat I" dersi farklı Id lere sahip olduğu için, ders her alındığında farklı ders muamelesi görmüş ve her defasında kredi toplamına dâhil edilmiştir. Birleştirme işleminden sonra ise hepsi aynı ders olduğu için kredi toplamına sadece bir defa dâhil edilmiştir. Bu şekilde toplam kredi 30 dan 24 e düşmüş; AGNO'su 2.65 den 3.31 e yükselmiştir.

4.4. ÖĞRENCİ İŞLERİ DAİRE BAŞKANLIĞI İÇİN YAPILAN MODÜL

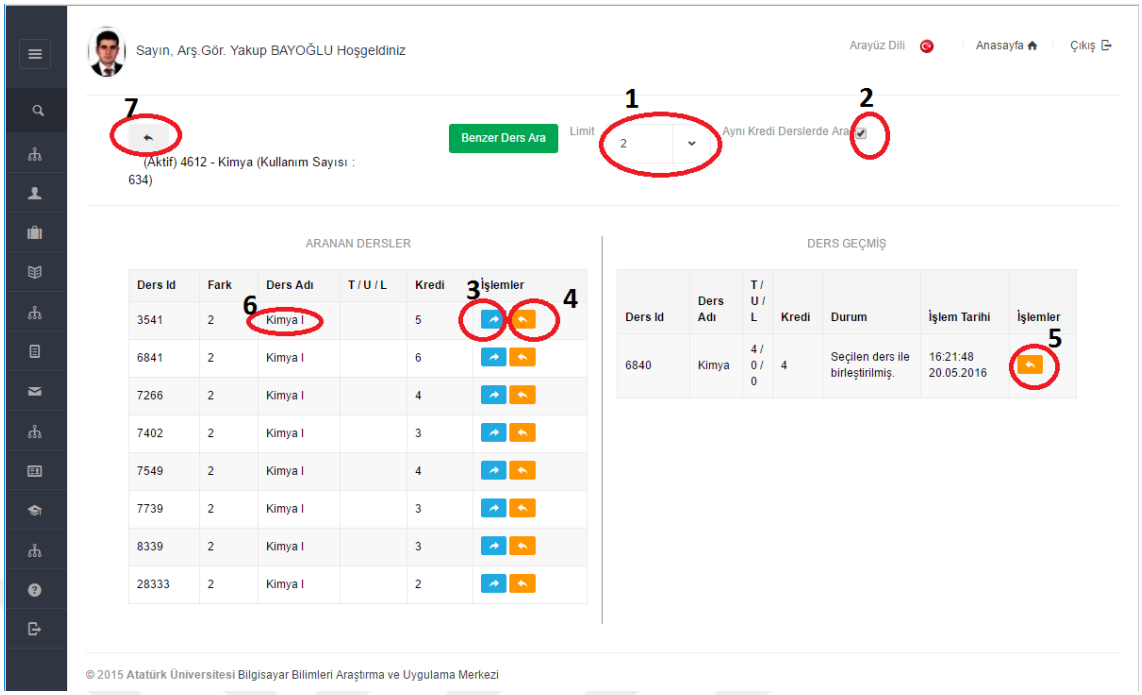
Öğrenci Bilgi Sistemi içerisinde yer alan bu modülde kullanıcıyı ilk olarak bir arama sayfası karşılamaktadır. Bu sayfada ismine göre ders aranmakta ve sonuçlar listelenmektedir. Arama sayfasında "Kimya" diye aratılınca gelen sonuç Şekil 4.9'da

gösterilmektedir. Burada aramak için yazılan ifade, ders adının başında geçen, sonunda geçen, içerisinde geçen ya da yazılan ifade ile birebir aynı olan şeklinde aranabilmektedir. Gelen sonuçlar ders statüsüne göre; aktif, pasif ya da tümü şeklinde filtrelenebilmektedir. Ayrıca aranan dersin ID si biliniyorsa eğer, bu Id değeri ile de arama yapmak mümkündür. Kullanıcı, arama kutusuna ders adı yerine, dersin Id sini yazabilir. Ders arama saklı yordamı, ders adı olarak gelen parametrenin içeriği nümerik bir değer ise dersId sütununa göre; diğer durumda dersAdi sütununa göre arama yapmaktadır.



Şekil 4.9. Ders Arama Sayfası ve Arama Sonuçları

Arama sonuçlarında derse tıklanınca, kullanıcı seçilen dersin bilgilerinin yer aldığı yeni bir sayfaya yönlendirilmektedir. Arama sonucundaki listeden 4612 Id li "Kimya" dersi seçildiğinde gelen sayfanın görüntüsü Şekil 4.10'da gösterilmiştir. Bu sayfada, iki liste yer almaktadır.



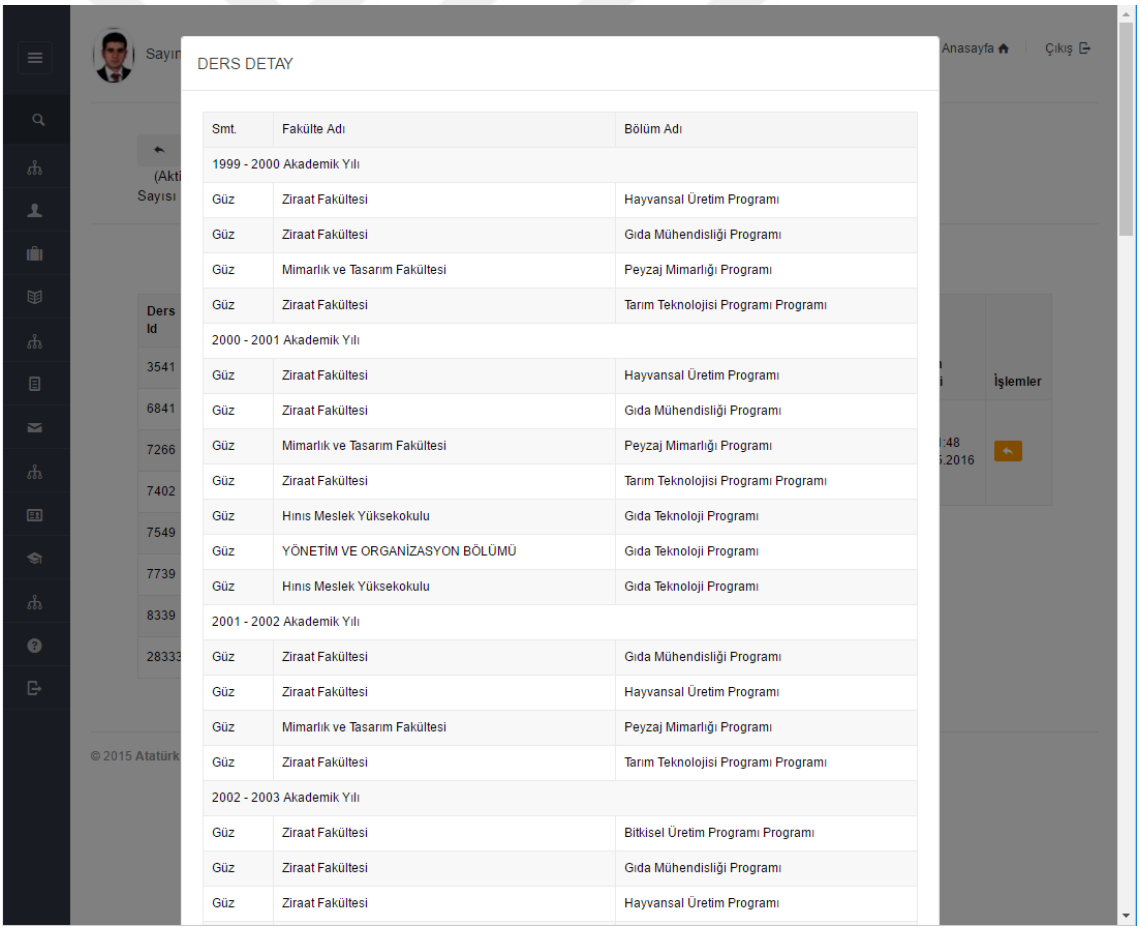
Şekil 4.10. Seçilen Dersin Detay Bilgilerinin Yer Aldığı Sayfa

Sayfanın sol tarafında yer alan birinci listede seçili dersin adı ile benzer ada sahip dersler listelenmektedir. Bu listedeki fark sütununda, arama sayfasında seçilen 4612 Id li dersin adı ile satırda yer alan 3541 Id li dersin adının birbirlerine olan DL uzaklığı yer almaktadır. "T/U/L" sütununda dersin teori, uygulama ve laboratuvar saati bilgileri yer almaktadır. İşlemler sütununda ise ders birleştirme işlemini gerçekleştirecek olan iki buton yer almaktadır. Her iki butonun da görevi dersleri birleştirmektir; farkları birleştirmenin hangi ders üzerinde yapılacağıdır. 3 numaralı buton kullanılırsa eğer listedeki 3541 Id li ders üzerinde birleştirilir. 4 numaralı buton kullanıldığında ise arama sayfasından seçilerek getirilmiş olan 4612 Id li ders üzerinde birleştirilir.

Sayfanın sağ tarafında yer alan ikinci listede, arama sayfasından seçilen 4612 Id li ders üzerinde önceden yapılmış olan birleştirme işlem(ler)i var ise eğer, bu dersler listelenir. Aynı şekilde arama sayfasında seçilen ders pasif bir ders ise; yani başka bir ders ile birleştirilmiş ise üzerinde birleştirildiği ders de listede yer alır. Durum sütununda, birleştirme işleminin hangi ders üzerinde yapıldığını belirten ifade yer almaktadır. Şekil 4.10'da 5 ile işaretlenen buton aracılığı ile önceden yapılmış olan bu birleştirme işlemi geri alınabilir.

Birinci liste maksimum uzaklık (Fark) değeri ile filtrelenebilir. Şekil 4.10'da 1 ile gösterilen kombodan seçilen değere göre "Benzer Ders Ara" butonuna tıklayarak benzer dersler listesi güncellenebilir. Listelenen dersler seçili ders ile aynı krediye sahip olma durumuna göre de filtrelenebilmektedir. Bunun için Şekil 4.10'da 2 ile işaretlenen yerdeki "Aynı Kredili Derslerde Ara" kutucuğu kullanılmaktadır.

Birinci listede yer alan herhangi bir satırdaki ders adına tıklandığında o dersin kullanım geçmişi listelenmektedir. Bu listede dersin hangi yıllarda hangi fakültenin hangi programlarında kullanıldığı bilgisi yer almaktadır. Dersleri birleştirecek olan ÖİDB personeli, yazılışı birbirine benzeyen derslerin aynı ders olup olmadığına dair şüphe duyması durumunda hangi Fakülte(ler)den bilgi alması gerektiğini bu sayede tespit edebilecektir. 3541 Id li "Kimya I" dersinin kullanım geçmişi Şekil 4.11'de gösterilmiştir.



Smt.	Fakülte Adı	Bölüm Adı
1999 - 2000 Akademik Yılı		
Güz	Ziraat Fakültesi	Hayvansal Üretim Programı
Güz	Ziraat Fakültesi	Gıda Mühendisliği Programı
Güz	Mimarlık ve Tasarım Fakültesi	Peyzaj Mimarlığı Programı
Güz	Ziraat Fakültesi	Tarım Teknolojisi Programı Programı
2000 - 2001 Akademik Yılı		
Güz	Ziraat Fakültesi	Hayvansal Üretim Programı
Güz	Ziraat Fakültesi	Gıda Mühendisliği Programı
Güz	Mimarlık ve Tasarım Fakültesi	Peyzaj Mimarlığı Programı
Güz	Ziraat Fakültesi	Tarım Teknolojisi Programı Programı
Güz	Hınıs Meslek Yüksekokulu	Gıda Teknoloji Programı
Güz	YÖNETİM VE ORGANİZASYON BÖLÜMÜ	Gıda Teknoloji Programı
Güz	Hınıs Meslek Yüksekokulu	Gıda Teknoloji Programı
2001 - 2002 Akademik Yılı		
Güz	Ziraat Fakültesi	Gıda Mühendisliği Programı
Güz	Ziraat Fakültesi	Hayvansal Üretim Programı
Güz	Mimarlık ve Tasarım Fakültesi	Peyzaj Mimarlığı Programı
Güz	Ziraat Fakültesi	Tarım Teknolojisi Programı Programı
2002 - 2003 Akademik Yılı		
Güz	Ziraat Fakültesi	Bitkisel Üretim Programı Programı
Güz	Ziraat Fakültesi	Gıda Mühendisliği Programı
Güz	Ziraat Fakültesi	Hayvansal Üretim Programı

Şekil 4.11. 3541 Id li "Kimya I" Dersinin Kullanım Geçmişi

SONUÇ VE TARTIŞMA

Veri tabanı kavramı, bilgi işlem dünyasında uzun tecrübe ve aşamalardan sonra ulaşılmış bir kavramdır ve en genel tanımıyla birbiri ile ilişki veriler topluluğu olarak tanımlanabilir. Günümüzde en çok tercih edilen veri tabanı sistemleri, ilişkisel veri tabanlarıdır. İlişkisel veri tabanlarında birden fazla tablo bulunur ve bu tablolar birbirleri ile ilişkilidir. Bu ilişkiler tablolardaki birincil anahtarlar ve dış anahtarlar vasıtasıyla kurulur.

Veri tekrarı, bir tabloda aynı bilginin birden fazla satırda yer almasıdır ve veri tabanı uygulamalarında karşılaşılan önemli sorunlardandır. Veri tabanının ilişkisel olması, veri tekrarını önleme açısından önemlidir, fakat tek başına yeterli değildir. Dolayısıyla uygulama geliştiricileri, veri tekrarı konusunda çeşitli önlemler almaları gerekir. Bu önlemlerin alınmadığı ya da yeterli olmadığı durumda veri tekrarı kaçınılmaz olacaktır. Atatürk Üniversitesi Öğrenci Bilgi Sisteminde (ÖBS) yer alan aynı isim ve krediye sahip dersler, veri tekrarına örnek olarak verilebilir.

Kirli veri, hatalı bilgi içeren veri olarak tanımlanabilir. Veri tekrarı için alınacak önlemler, kirli veriyi önleme konusunda yeterli olmayacaktır. ÖBS'deki yazımı hatalı dersler kirli veriye bir örnektir.

Veri tabanındaki tekrarlı veya kirli veriler çeşitli sorunlara yol açmaktadır. ÖBS'deki ders adlarındaki çoğullamalar ve yazım hataları; müfredatlarda fazla ders bulunması, öğrencilerin toplam kredi ve ağırlıklı genel not ortalamalarının yanlış hesaplanması, derslerin harf notu hesaplamasında tutarsızlıklar oluşması gibi sorunlara sebep olmaktadır. Bu çalışma ile ÖBS'de yer alan aynı isim ve krediye sahip ders adları tek bir kayıt altında birleştirildi. Aslında aynı olup yazılışı farklı olan dersler de tespit edilerek, bu dersler de tek bir kayıtta birleştirildi. Bu şekildeki dersleri tespit edebilmek için, metinsel ifadelerin birbirlerine olan benzerliğini tespit eden dizge karşılaştırma algoritmalarından Damerau-Levenshtein Algoritması kullanıldı. Ders birleştirme uygulaması sayesinde:

- Müfredatlarda fazla ders bulunması
- Öğrencilerin toplam kredi ve ağırlıklı genel not ortalamalarının yanlış hesaplanması

- Dersin gruplara bölünmesi sebebi ile harf notu hesaplamasında tutarsızlıklar oluşması
- Derslerin birden fazla açılması sebebiyle, öğrenci işleri personelinin ve dersin öğretim üyesinin iş yükünün artması

Sorunları büyük ölçüde çözülmüştür.

Ders birleştirme uygulaması sonucunda, başka bir ders üzerinde birleştirilip kapatılan ders sayısı 2,584'tür.

Ders birleştirme işlemleri esnasında, içinde dersId barındıran en önemli iki tablodan; müfredat bilgilerinin tutulduğu tabloda 50,635 kayıt, transkript bilgilerinin tutulduğu tabloda ise 187,651 öğrenciye ait 898,449 kayıt güncellenmiştir. Ders birleştirme işleminden sonra kredisi ve AGNO'su düzelen öğrenci sayısı 11,685'tir.

Üzerinde en fazla birleştirme yapılan ders 14170 id li "Atatürk İlkeleri ve İnkılap Tarihi I" dersidir ve 40 ders bu ders üzerinde birleştirilmiştir. Üzerinde en çok birleştirme yapılan ilk 15 ders aşağıdaki Tablo 1'de listelenmiştir.

Tablo 1. Üzerinde En Çok Birleştirme Yapılan İlk 15 Ders

DersId	**Ders Adı**	**Kredi**	**Sayı**
14170	Atatürk İlkeleri ve İnkılap Tarihi I	2	40
10064	Yabancı Dil I	2	27
14696	Atatürk İlkeleri ve İnkılap Tarihi II	2	19
10703	İngilizce I	2	16
6211	İngilizce II	2	16
6646	Türk Dili I	2	13
10775	Makro İktisat I	3	11
9474	Türk Dili II	2	11
6213	İngilizce II	4	11
13999	Atatürk İlkeleri ve İnkılap Tarihi	2	10
4762	Mikro İktisat I	3	9
9915	Yabancı Dil II	2	9
6208	İngilizce	2	8
4321	Makro İktisat II	3	8
7436	Mikro İktisat II	3	7

Öneriler

Öğrenci Bilgi Sistemi veri tabanında toplam 1345 tablo bulunmaktadır. Zamanında veri tabanı, mantık olarak ilişkisel kurulmuş; fakat bu ilişkiler veri tabanında tanımlanmamış idi. Hangi tabloların hangi sütunlarının Dersler tablosundaki DersId sütununa referans olduğu manuel olarak tespit edildi ve bunun için hayli emek ve zaman harcandı. Hâlbuki tablolar arasındaki ilişkiler, veri tabanına kaydedilmiş olsa idi, çok kolay bir şekilde tespit edilebilirdi. Bu çalışma esnasında, ilişkiler, veri tabanı düzeyinde tanımlandı.

Veri tabanına yeni bir ders ekleneceği zaman, mevcut kontrollere ilaveten benzerlik taraması yapılıp, ondan sonra dersi ekleyip eklememeye karar verilebilir. Örneğin veri tabanında "Kooperatif Muhasebesi" diye bir ders olmadığını; fakat bir harfi farklı olan "Kooparatif Muhasebesi" diye bir dersin mevcut olduğunu farz edelim. Bu durumda iken sisteme "Kooperatif Muhasebesi" diye bir ders eklenmek istenirse eğer, eklenir; çünkü bu isimde bir ders veri tabanında mevcut değildir. Fakat eklenmek istenen ders, eklenmeden önce mevcut dersler ile karşılaştırılırsa, bu dersin, aslında "Kooparatif Muhasebesi" şeklinde veri tabanında yer aldığı görülür ve yeni dersin eklenmesi yerine mevcut dersteki yazım hatası düzeltilmesi yoluna gidilebilir. Böylece, bundan sonra yaşanacak ders çoğullamaları engellenmiş olur.

Atatürk Üniversitesi Eğitim Öğretim Yönetmeliğine göre bir dersin kredisi aşağıdaki formüle göre hesaplanır:

Kredi = teori saati + uygulama saatinin yarısı + laboratuvar saatinin yarısı

Veri tabanında bu formüle uymayan 7,916 adet ders bulunmaktadır. Öğrenci İşleri Daire Başkanlığının kuracağı bir komisyon aracılığı ile bu derslerin doğru saat bilgileri tespit edilip düzeltilebilir, akabinde yeniden ders birleştirme yapılabilir.

ÖBS'de bu tip hatalı veri tanımlamalarının yaşanmaması için tüm konuları kapsayan genel bir ÖBS veri giriş yönergesi hazırlanıp ÖİDB ve akademik birimlerde görevli ÖBS kullanıcılarına tebliğ edilerek konu hakkında farkındalık kazanmaları sağlanabilir. Bununla birlikte ÖBS'ye veri girişi yapacak personelde bazı özelliklerin aranmasında fayda olduğu düşünülmektedir. Bu özelliklerin bazıları şöyle sıralanabilir:

- Analitik düşünme becerisine sahip olma,

- Durumlar arasındaki sebep sonuç ilişkilerini kavrayabilme,
- Sistemin güçlü ve zayıf yönlerini fark edip bunları ifade edebilme,
- Teknolojideki gelişmeleri takip etme ve bu gelişmelere kolayca uyum sağlayabilme,
- Kullanıcıların sorun yaşayabilecekleri durumları öngörebilecek tecrübeye sahip olma,
- Gerekli temel mevzuat bilgilerinden haberdar olma,
- Sistemin kullanımını kullanıcılara basit ve anlaşılır bir şekilde anlatabilecek anlatım becerisine sahip olma
- TDK yazım kurallarını bilme ve uygulayabilme.

Veri tabanında derslerde olduğu gibi, bireylerde de çoğul kayıtlar bulunmaktadır. Benzer bir tekilleme çalışması, birey kayıtları üzerinde de yapılabilir.

KAYNAKÇA

Augsten, N., & Böhlen, M. H. (2013)."Similarity joins in relational database systems". *Synthesis Lectures on Data Management",* 5(5), 1-124.

Batuk, F. (1997). *Arazi Bilgi Sistemi Ders Notları,* İstanbul: YTÜ Jeodezi ve Fotogrametri Mühendisliği Bölümü.

Çamoğlu, K. (2009). *Programlama & Veritabani Mantiği,* İstanbul: Kodlab Yayın Dağıtım Yazılım ve Eğitim Hizmetleri.

Damerau, F. J. (1964). "A technique for computer detection and correction of spelling errors". *Communications of the ACM, 7*(3), 171-176.

Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems*. 7th: New Jersey: Pearson.

Gözüdeli, Y. (2010). *Yazılımcılar İçin SQL Server 2008 R2 & Veritabanı Programlama* (5 ed.), Ankara: Seçkin Yayıncılık.

Karaş, İ. R., Baz, İ., & Geymen, A. (2006). "Farklı Formattaki Konumsal ve Özniteliksel Verilerin Otomatik Olarak Bir Coğrafi Veri Tabanına Dönüştürülmesi", 4. *Coğrafi Bilgi Sistemleri Bilişim Günleri*, 13-16.

Laudon, K. C., & Laudon, J. P. (2012). *Management Information Systems: Managing the Digital Firm,* Essex: Prentice Hall Press

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

Liu, L., & Özsu, M. T. (2009). *Encyclopedia of database systems* (Vol. 6), New York, NY, USA: Springer

Öztürk, S., & Atmaca, H. E. (2017). "İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi". *Bilişim Teknolojileri Dergisi, 10*(2), 199-209.

Sunderic, D. (2002). *Stored Procedure Programming,* New York, NY: McGraw-Hill, Inc.

Ünal, Ü. (2011). *Yüksek Lisans Öğrenci Ön Kabul Otomasyon Sistemi Için Veritabani Tasarlanmasi ve Uygulaması*. (Yüksek Lisans Tezi), Kütahya: Dumlupınar Üniversitesi.

İNTERNET KAYNAKLARI

http://www.webyazilimdilleri.com/web/yazilim/dilleri/makaleler/5282/C%23NET-TE-3-KATMANLI-MIMARI/default.aspx, Erişim: 14.06.2016

https://en.wikipedia.org/wiki/Metadata, Erişim: 20.06.2018

https://www.teknologweb.com/t-sql-nedir-transact-sql, Erişim: 17.07.2018

http://www.bilisimlife.net/Yazi/148-SQL-Serverda-Trigger-Kullanimi.html, Erişim: 17.07.2018

http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm, Erişim: 21.06.2018

https://en.wikipedia.org/wiki/Longest_common_subsequence_problem, Erişim: 21.06.2018

http://bilgisayarkavramlari.sadievrenseker.com/2007/12/04/en-uzun-ortak-kume-longest-common-subsequence-lcs/, Erişim: 21.06.2018

http://w3.gazi.edu.tr/~akcayol/files/AAL8Dynamic.pdf, Erişim: 14.06.2018

http://www.wucathy.com/blog/?p=680, Erişim: 22.06.2018

http://www.ieee.ma/uaesb/pdf/distances-in-classification.pdf, Erişim: 22.06.2018

https://en.wikipedia.org/wiki/Hamming_distance, Erişim: 22.06.2018

https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance, Erişim: 22.06.2018

https://ubys.atauni.edu.tr/ , Erişim: 16.07.2018

https://obs.atauni.edu.tr/, Erişim: 16.07.2018

http://www.mevzuat.gov.tr/Metin.Aspx?MevzuatKod=8.5.23757&MevzuatIliski=0&sourceXmlSearch=, Erişim: 16.07.2018

http://alias-i.com/lingpipe/demos/tutorial/stringCompare/read-me.html, Erişim: 23.07.2018

EKLER

EK 1. Ders Birleştirme Saklı Yordamı

```
CREATE procedure [dbo].[usp_admin_DersBirlestirme]
      @YeniDersId           int,
      @KapanacakDersIdleri  varchar(1000),
      @UserId               int
as
Begin
/*
Güncellenen tablolar ve eklenme tarihleri:
      AcilanDersler --> DersId
      DersIliskileri --> Ders1_Id
      DersIliskileri --> Ders2_Id
      Mufredat --> DersId
      NotKarti --> Ders_Id
      NotKartiButunlemeLog --> Ders_Id
      DersKayit --> Ders_Id
      HaftalikProgram --> DersId
      NotKartiDuzenlemeLog --> Ders_Id
      OgrencininAlttanDersleri --> DersId
      Sinav --> Ders_Id
      UzemAGNOicinDersKayit --> Ders_Id
29.09.2017
      OgrencininAlttanDersleri_AOF --> DersId
      DegisimProgramiDersKayit --> Ders2Id
      MuafiyetNotBilgileri --> Ders_Id
      MuafiyetNotBilgileri_Log --> Ders_Id

*/

      Declare @DegistirilmesiIstenenDersler table(
            DersId int
      )

      Declare
            @YeniDersTeori  int,
            @YeniDersLab    int,
            @YeniDersPratik int,
            @YeniDersKredi  float,
            @Sayac1         int,
            @Sayac2         int,
            @Basarili       int = 0


      if (select Statu from Dersler (nolock) where Ders_Id = @YeniDersId) in (0,
10)
      Begin
            select
                  'Hata' = 0,
                  'Mesaj' = 'Üzerinde birleştirilmesi istenen ders pasif bir ders
olduğu için işlem iptal edildi.'
            return
      End
```

```
	if @KapanacakDersIdleri not like '%,'
		select @KapanacakDersIdleri += ','

	Insert
		@DegistirilmesiIstenenDersler
	Select
		F.Id
	From
		dbo.fn_Split(@KapanacakDersIdleri) F
	Where
		F.ID <> @YeniDersId
	order by
		F.Id

	select
		@YeniDersId,
		D1.Ders_Adi,
		D2.Ders_Adi,
		D2.Ders_Id
	from
		@DegistirilmesiIstenenDersler D,
		Dersler D1,
		Dersler D2
	where
		D1.Ders_Id = @YeniDersId
		and D2.Ders_Id = D.DersId


	delete from @DegistirilmesiIstenenDersler
	where DersId in (select EskiDersId from DersTekilleme)

	if not exists (select * from @DegistirilmesiIstenenDersler)
	Begin
		select
			'Hata' = 2,
			'Mesaj' ='Bu ders(ler) daha önce birleştirilmiş'
		return
	End


	Begin Transaction

	update BB
	set
		AgnoGuncelmi = 0
	from
		NotKarti NK (nolock)
		inner join BireyBirimleri BB
			on BB.Birey_Id = NK.Ogrenci_Id
			and BB.Birim_Id = NK.Birim_Id
		inner join @DegistirilmesiIstenenDersler DRS
			on DRS.DersId = NK.Ders_Id


	insert into DersTekilleme
	(
		EskiDersId,
		YeniDersId,
		DegistirenKisi,
		DegismeTarihi
```

```
	)
	select
		D.DersId,
		@YeniDersId,
		@UserId,
		GETDATE()
	from
		@DegistirilmesiIstenenDersler D


-----------------------------------------------------------------------
-- AcilanDersler --> DersId
-----------------------------------------------------------------------

	Insert Into DersTekillemeDetay
	Select
		S.DersId,
		@YeniDersId,
		'AcilanDersler',
		'DersId',
		AC.[ID],
		'ID',
		@UserId,
		getdate(),
		null
	From
		AcilanDersler AC (nolock)
		Inner Join @DegistirilmesiIstenenDersler S
			on S.DersID = AC.DersID

	Update AC
	Set
		DersId = @YeniDersId,
		DersAdi = D.Ders_Adi,
		DersPratikKredi = D.Pratik,
		DersTeoriKredi = D.Teori,
		DersLabKredi = D.Lab,
		DersKredi = D.Kredi
	From
		AcilanDersler AC
		inner join @DegistirilmesiIstenenDersler S
			on S.DersId = AC.DersId
		Inner Join Dersler D (nolock)
			on D.Ders_ID = @YeniDersId


-----------------------------------------------------------------------
-- DersIliskileri --> Ders1_Id
-----------------------------------------------------------------------

	Insert Into DersTekillemeDetay
	Select
		S.DersId,
		@YeniDersId,
		'DersIliskileri',
		'Ders1_Id',
		DI.APH_Id,
		'APH_Id',
		@UserId,
		getdate(),
```

```
            null
      From
            DersIliskileri DI (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = DI.Ders1_Id

      Update DI
      Set
            Ders1_Id = @YeniDersId
      From
            DersIliskileri DI
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = DI.Ders1_Id


-------------------------------------------------------------------------
-- DersIliskileri --> Ders2_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'DersIliskileri',
            'Ders2_Id',
            DI.APH_Id,
            'APH_Id',
            @UserId,
            getdate(),
            null
      From
            DersIliskileri DI (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = DI.Ders2_Id

      Update DI
      Set
            Ders2_Id = @YeniDersId
      From
            DersIliskileri DI
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = DI.Ders2_Id


-------------------------------------------------------------------------
-- Mufredat --> DersId
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'Mufredat',
            'DersId',
            M.ID,
            'ID',
            @UserId,
            getdate(),
            null
      From
            Mufredat M (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
```

```
                on S.DersID = M.DersId

    Update M
    Set
        DersId = @YeniDersId,
        DersTeoriKredi = D.Teori,
        DersLabKredi = D.Lab,
        DersPratikKredi = D.Pratik,
        DersKredi = D.Kredi
    From
        Mufredat M
        inner join @DegistirilmesiIstenenDersler S
            on S.DersId = M.DersId
        inner join Dersler D (nolock)
            on D.Ders_Id = @YeniDersId


-------------------------------------------------------------------------
-- NotKarti --> Ders_Id
-------------------------------------------------------------------------
    Insert Into DersTekillemeDetay
    Select
        S.DersId,
        @YeniDersId,
        'NotKarti',
        'Ders_Id',
        NK.D_Id,
        'D_Id',
        @UserId,
        getdate(),
        null
    From
        NotKarti NK (nolock)
        Inner Join @DegistirilmesiIstenenDersler S
            on S.DersID = NK.Ders_Id

    Update NK
    Set
        Ders_Id = @YeniDersId
        --Kredi = D.Kredi,
        --BdsKatsayi = D.Kredi * Katsayi,
        --SAno = D.Kredi * DNot
    From
        NotKarti NK
        inner join @DegistirilmesiIstenenDersler S
            on S.DersId = NK.Ders_Id
        inner join Dersler D
            on D.Ders_Id = @YeniDersId


-------------------------------------------------------------------------
-- NotKartiButunlemeLog --> Ders_Id
-------------------------------------------------------------------------
    Insert Into DersTekillemeDetay
    Select
        S.DersId,
        @YeniDersId,
        'NotKartiButunlemeLog',
        'Ders_Id',
        NKBL.D_Id,
```

```
            'D_Id',
            @UserId,
            getdate(),
            null
      From
            NotKartiButunlemeLog NKBL (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = NKBL.Ders_Id

      Update NKBL
      Set
            Ders_Id = @YeniDersId,
            Kredi = D.Kredi,
            BdsKatsayi = D.Kredi * Katsayi,
            SAno = D.Kredi * DNot
      From
            NotKartiButunlemeLog NKBL
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = NKBL.Ders_Id
            inner join Dersler D (nolock)
                  on D.Ders_Id = @YeniDersId


-------------------------------------------------------------------------
-- DersKayit --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'DersKayit',
            'Ders_Id',
            DK.DA_Id,
            'DA_Id',
            @UserId,
            getdate(),
            null
      From
            DersKayit DK (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = DK.Ders_Id

      Update DK
      Set
            Ders_Id = @YeniDersId
      From
            DersKayit DK
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = DK.Ders_Id


-------------------------------------------------------------------------
-- HaftalikProgram --> DersId
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'HaftalikProgram',
            'DersId',
```

```
            HP.DP_Id,
            'DP_Id',
            @UserId,
            getdate(),
            null
      From
            HaftalikProgram HP (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = HP.DersId

      Update HP
      Set
            DersId = @YeniDersId
      From
            HaftalikProgram HP
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = HP.DersId


-------------------------------------------------------------------------
-- NotKartiDuzenlemeLog --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'NotKartiDuzenlemeLog',
            'Ders_Id',
            NKDL.Id,
            'Id',
            @UserId,
            getdate(),
            null
      From
            NotKartiDuzenlemeLog NKDL (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = NKDL.Ders_Id

      Update NKDL
      Set
            Ders_Id = @YeniDersId
      From
            NotKartiDuzenlemeLog NKDL
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = NKDL.Ders_Id


-------------------------------------------------------------------------
-- OgrencininAlttanDersleri --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'OgrencininAlttanDersleri',
            'DersId',
            OAD.Id,
            'Id',
            @UserId,
            getdate(),
```

```
            null
      From
            OgrencininAlttanDersleri OAD (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = OAD.DersId

      Update OAD
      Set
            DersId = @YeniDersId
      From
            OgrencininAlttanDersleri OAD
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = OAD.DersId


-------------------------------------------------------------------------
-- Sinav --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'Sinav',
            'Ders_Id',
            ST.Sinav_Id,
            'Sinav_Id',
            @UserId,
            getdate(),
            null
      From
            Sinav ST (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = ST.Ders_Id

      Update ST
      Set
            Ders_Id = @YeniDersId
      From
            Sinav ST
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = ST.Ders_Id


-------------------------------------------------------------------------
-- UzemAGNOicinDersKayit --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'UzemAGNOicinDersKayit',
            'Ders_Id',
            UADK.DA_Id,
            'DA_Id',
            @UserId,
            getdate(),
            null
      From
            UzemAGNOicinDersKayit UADK (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
```

```
                on S.DersID = UADK.Ders_Id

      Update UADK
      Set
            Ders_Id = @YeniDersId
      From
            UzemAGNOicinDersKayit UADK
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = UADK.Ders_Id


-------------------------------------------------------------------------
-- OgrencininAlttanDersleri_AOF --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'OgrencininAlttanDersleri_AOF',
            'DersId',
            OAD.Id,
            'Id',
            @UserId,
            getdate(),
            null
      From
            OgrencininAlttanDersleri_AOF OAD (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = OAD.DersId

      Update OAD
      Set
            DersId = @YeniDersId
      From
            OgrencininAlttanDersleri_AOF OAD
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = OAD.DersId


-------------------------------------------------------------------------
-- DegisimProgramiDersKayit --> Ders2Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'DegisimProgramiDersKayit',
            'Ders2Id',
            DPDK.Id,
            'Id',
            @UserId,
            getdate(),
            null
      From
            DegisimProgramiDersKayit DPDK (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = DPDK.Ders2Id

      Update DPDK
      Set
```

```
            Ders2Id = @YeniDersId
      From
            DegisimProgramiDersKayit DPDK
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = DPDK.Ders2Id


-------------------------------------------------------------------------
-- MuafiyetNotBilgileri --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'MuafiyetNotBilgileri',
            'Ders_Id',
            MNB.Id,
            'Id',
            @UserId,
            getdate(),
            null
      From
            MuafiyetNotBilgileri MNB (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = MNB.Ders_Id

      Update MNB
      Set
            Ders_Id = @YeniDersId
      From
            MuafiyetNotBilgileri MNB
            inner join @DegistirilmesiIstenenDersler S
                  on S.DersId = MNB.Ders_Id


-------------------------------------------------------------------------
-- MuafiyetNotBilgileri_Log --> Ders_Id
-------------------------------------------------------------------------
      Insert Into DersTekillemeDetay
      Select
            S.DersId,
            @YeniDersId,
            'MuafiyetNotBilgileri_Log',
            'Ders_Id',
            MNBL.Id,
            'Id',
            @UserId,
            getdate(),
            null
      From
            MuafiyetNotBilgileri_Log MNBL (nolock)
            Inner Join @DegistirilmesiIstenenDersler S
                  on S.DersID = MNBL.Ders_Id

      Update MNBL
      Set
            Ders_Id = @YeniDersId
      From
            MuafiyetNotBilgileri_Log MNBL
            inner join @DegistirilmesiIstenenDersler S
```

```
                on S.DersId = MNBL.Ders_Id


-------------------------------------------------------------------------
-- Dersin Pasif edilmesi
-------------------------------------------------------------------------


     Update D
     Set
          Statu = 10 -- 10: Ders birleştirme sebebi ile kullanıma kapatılma
     From
          Dersler D
          inner join @DegistirilmesiIstenenDersler S
               on S.DersId = D.Ders_Id

-------------------------------------------------------------------------
-- Sonuç mesajının döndürülmesi
-------------------------------------------------------------------------
     If @@ERROR = 0
     Begin
          exec usp_admin_DersBirlestirmeSonrasiMufredatTekilleme @YeniDersId,
@Basarili out

          if @Basarili = 1
          Begin
               Commit Transaction
               Select
                    'Hata' = 0,
                    'Mesaj'= 'İşlem başarı ile tamamlandı.'
          End
          Else
          Begin
               Rollback Transaction
               Select
                    'Hata' = 1,
                    'Mesaj'= 'İşlem ikinci adımında bir hata oluştu. Lütfen
işlemi yenileyin.'
          End
     End
     Else
     Begin
          Rollback Transaction
          Select
               'Hata' = 1,
               'Mesaj'= 'İşlem yapılırken bir hata oluştu. Lütfen işlemi
yenileyin'
     End

End
```

EK 2. Ders Birleştirme Sonrası Müfredat Tekilleme Saklı Yordamı

```
CREATE procedure [dbo].[usp_admin_DersBirlestirmeSonrasiMufredatTekilleme]
	@BirlestirilmisDersId int,
	@UserId		int,
	@Basarili	int out
as
Begin

	declare @Mufredatlar table
	(
		Id		int identity(1,1),
		BirimId	int,
		AYID	int,
		DersId	int,
		KalacakMufredatId	int
	)

	declare @KullanimSayilari table
	(
		Id			int identity(1,1),
		DerslerId		int,
		MufredatId		int,
		KullanimSayisi	int,
		YeniMufredatId	int
	)

	-- Müfredatlarda oluşan çoğullamalar tespit ediliyor
	insert into @Mufredatlar
	select
		M.Birim_Id,
		M.AY_Id,
		M.DersId,
		null
	from
		Mufredat M (nolock)
	where
		M.DersId = @BirlestirilmisDersId
		--and M.KayitDonem = 1
		--and M.Birim_Id > 0
		--and M.Birim_Id = 1100
	group by
		M.Birim_Id,
		M.AY_Id,
		M.DersId
	having
		COUNT(*) > 1

	--select * from @Mufredatlar

	-- Müfredatlarda çoğullama oluşmadı ise sonlansın
	if not exists (select 1 from @Mufredatlar)
	Begin
		select @Basarili = 1
		return
	End

	-- Çoğullayan müfredatId ler içersined en çok kullanılan MufredatId tespit
ediliyor
```

```
	insert into @KullanimSayilari
	select
		D.Id,
		M.ID,
		COUNT(NK.D_Id),
		null
	from
		@Mufredatlar D
		inner join Mufredat M (nolock)
			on M.AY_Id = D.AYID
			and M.Birim_Id = D.BirimId
			and M.DersId = D.DersId
		left join NotKarti NK (nolock)
			on NK.IAPB_ID = M.Id
	group by
		D.Id,
		M.ID

	--select * from @KullanimSayilari order by DerslerId, KullanimSayisi desc

	update D
	set
		KalacakMufredatId = K.MufredatId
	from
		@Mufredatlar D
		inner join @KullanimSayilari K
			on K.DerslerId = D.Id
		inner join (select
					D.Id,
					MAX(KullanimSayisi) KullanimSayisi
				from
					@Mufredatlar D
					inner join @KullanimSayilari K
						on K.DerslerId = D.Id
				group by
					D.Id ) S
		on S.Id = D.Id
		and S.KullanimSayisi = K.KullanimSayisi


	update @KullanimSayilari
	set
		YeniMufredatId = D.KalacakMufredatId
	from
		@KullanimSayilari K
		inner join @Mufredatlar D
			on D.Id = K.DerslerId


	--select * from @KullanimSayilari

	delete from @KullanimSayilari where YeniMufredatId is null

	delete from @KullanimSayilari where YeniMufredatId = MufredatId

	--select * from @Mufredatlar
	--select * from @KullanimSayilari

	--return
```

```
	BEGIN TRANSACTION

	insert into MufredatTekilleme(
		EskiMufredatId,
		YeniMufredatId,
		KalanDersId,
		DegismeTarihi
	)
	select
		MufredatId,
		YeniMufredatId,
		@BirlestirilmisDersId,
		GETDATE()
	from
		@KullanimSayilari


-------------------------------------------------------------------------
---- Yedek alınıyor
-------------------------------------------------------------------------

	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'NotKarti', 'APB_ID', NK.D_Id, 'D_Id', @UserId, getdate()
From NotKarti NK (nolock) Inner Join @KullanimSayilari K on K.MufredatId =
NK.APB_ID
	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'NotKarti', 'IAPB_ID',NK.D_Id, 'D_Id', @UserId, getdate()
From NotKarti NK (nolock) Inner Join @KullanimSayilari K on K.MufredatId =
NK.IAPB_ID

	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'NotKartiButunlemeLog', 'APB_ID', NKBL.D_Id, 'D_Id',
@UserId, getdate() From NotKartiButunlemeLog NKBL (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = NKBL.APB_ID
	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'NotKartiButunlemeLog', 'IAPB_ID',NKBL.D_Id, 'D_Id',
@UserId, getdate() From NotKartiButunlemeLog NKBL (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = NKBL.IAPB_ID

	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'DersKayit', 'APB_ID', DK.DA_Id, 'DA_Id', @UserId,
getdate() From DersKayit DK (nolock) Inner Join @KullanimSayilari K on
K.MufredatId = DK.APB_ID
	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'DersKayit', 'IAPB_ID',DK.DA_Id, 'DA_Id', @UserId,
getdate() From DersKayit DK (nolock) Inner Join @KullanimSayilari K on
K.MufredatId = DK.IAPB_ID

	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'MufredatIntibaki', 'APB1_ID', MI.ID, 'ID', @UserId,
getdate() From MufredatIntibaki MI (nolock) Inner Join @KullanimSayilari K on
K.MufredatId = MI.APB1_ID
	Insert Into MufredatTekillemeDetay Select K.MufredatId,
	K.YeniMufredatId, 'MufredatIntibaki', 'APB2_ID', MI.ID, 'ID', @UserId,
getdate() From MufredatIntibaki MI (nolock) Inner Join @KullanimSayilari K on
K.MufredatId = MI.APB2_ID
```

```
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'NotKartiDuzenlemeLog', 'APB_ID', NKDL.ID, 'ID', @UserId,
getdate() From NotKartiDuzenlemeLog NKDL (nolock) Inner Join @KullanimSayilari K
on K.MufredatId = NKDL.APB_ID
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'NotKartiDuzenlemeLog', 'IAPB_ID',NKDL.ID, 'ID', @UserId,
getdate() From NotKartiDuzenlemeLog NKDL (nolock) Inner Join @KullanimSayilari K
on K.MufredatId = NKDL.IAPB_ID

      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'OgrencininAlttanDersleri', 'APB_ID', OAD.ID, 'ID',
@UserId, getdate() From OgrencininAlttanDersleri OAD (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = OAD.APB_ID
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'OgrencininAlttanDersleri', 'IAPB_ID',OAD.ID, 'ID',
@UserId, getdate() From OgrencininAlttanDersleri OAD (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = OAD.IAPB_ID

      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'OgrencininAlttanDersleri_AOF', 'APB_ID', OAD.ID, 'ID',
@UserId, getdate() From OgrencininAlttanDersleri_AOF OAD (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = OAD.APB_ID
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'OgrencininAlttanDersleri_AOF', 'IAPB_ID',OAD.ID, 'ID',
@UserId, getdate() From OgrencininAlttanDersleri_AOF OAD (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = OAD.IAPB_ID

      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'UzemAGNOicinDersKayit', 'APB_ID', UADK.DA_ID, 'DA_ID',
@UserId, getdate() From UzemAGNOicinDersKayit UADK (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = UADK.APB_ID
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'UzemAGNOicinDersKayit', 'IAPB_ID',UADK.DA_ID, 'DA_ID',
@UserId, getdate() From UzemAGNOicinDersKayit UADK (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = UADK.IAPB_ID

      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'DegisimProgramiDersKayit', 'APBID',DPDK.ID, 'Id',
@UserId, getdate() From DegisimProgramiDersKayit DPDK (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = DPDK.APBID

      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'MuafiyetNotBilgileri', 'M_ID', MNB.ID, 'ID', @UserId,
getdate() From MuafiyetNotBilgileri MNB (nolock) Inner Join @KullanimSayilari K
on K.MufredatId = MNB.M_ID
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'MuafiyetNotBilgileri', 'IM_ID',MNB.ID, 'ID', @UserId,
getdate() From MuafiyetNotBilgileri MNB (nolock) Inner Join @KullanimSayilari K
on K.MufredatId = MNB.IM_ID

      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'MuafiyetNotBilgileri_Log', 'M_ID', MNBL.ID, 'ID',
@UserId, getdate() From MuafiyetNotBilgileri_Log MNBL (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = MNBL.M_ID
      Insert Into MufredatTekillemeDetay Select K.MufredatId,
      K.YeniMufredatId, 'MuafiyetNotBilgileri_Log', 'IM_ID',MNBL.ID, 'ID',
@UserId, getdate() From MuafiyetNotBilgileri_Log MNBL (nolock) Inner Join
@KullanimSayilari K on K.MufredatId = MNBL.IM_ID
```

```
------------------------------------------------------------------------
---- Güncelleniyor
------------------------------------------------------------------------

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
NotKarti T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
NotKarti T on K.MufredatId = T.IAPB_ID

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
NotKartiButunlemeLog T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
NotKartiButunlemeLog T on K.MufredatId = T.IAPB_ID

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
DersKayit T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
DersKayit T on K.MufredatId = T.IAPB_ID

      update T set APB1_ID = K.YeniMufredatId from @KullanimSayilari K inner join
MufredatIntibaki T on K.MufredatId = T.APB1_ID
      update T set APB2_ID = K.YeniMufredatId from @KullanimSayilari K inner join
MufredatIntibaki T on K.MufredatId = T.APB2_ID

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
NotKartiDuzenlemeLog T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
NotKartiDuzenlemeLog T on K.MufredatId = T.IAPB_ID

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
OgrencininAlttanDersleri T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
OgrencininAlttanDersleri T on K.MufredatId = T.IAPB_ID

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
OgrencininAlttanDersleri_AOF T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
OgrencininAlttanDersleri_AOF T on K.MufredatId = T.IAPB_ID

      update T set APB_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
UzemAGNOicinDersKayit T on K.MufredatId = T.APB_ID
      update T set IAPB_ID = K.YeniMufredatId from @KullanimSayilari K inner join
UzemAGNOicinDersKayit T on K.MufredatId = T.IAPB_ID

      update T set APBID = K.YeniMufredatId from @KullanimSayilari K inner join
DegisimProgramiDersKayit T on K.MufredatId = T.APBID

      update T set M_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
MuafiyetNotBilgileri T on K.MufredatId = T.M_ID
      update T set IM_ID = K.YeniMufredatId from @KullanimSayilari K inner join
MuafiyetNotBilgileri T on K.MufredatId = T.IM_ID

      update T set M_ID  = K.YeniMufredatId from @KullanimSayilari K inner join
MuafiyetNotBilgileri_Log T on K.MufredatId = T.M_ID
      update T set IM_ID = K.YeniMufredatId from @KullanimSayilari K inner join
MuafiyetNotBilgileri_Log T on K.MufredatId = T.IM_ID


      -- Fazlalık halindeki kayıtlar müfredattan siliniyor
```

```
	delete from Mufredat where Id in (select MufredatId from @KullanimSayilari
where YeniMufredatId != MufredatId)

	If @@ERROR = 0
	Begin
		Commit Transaction
		select @Basarili = 1
		--Select 'Message'= 'İşlem Başarı İle Tamamlandı'
	End
	Else
	Begin
		Rollback Transaction
		select @Basarili = 0
		--Select 'Message'= 'İşlem yapılırken bir hata oluştu. Lütfen işlemi
yenileyin'
	End

End
```

EK 3. Müfredat Birleştirmeyi Geri Alma Saklı Yordamı

```
CREATE procedure [dbo].[usp_admin_DersBirlestirmeSonrasiMufredatTekilleme_GeriAl]
	@KalanDersId	int,
	@EskiDersId	int,
	@Basarili	int out,
	@UserId		int
as
Begin

	declare @SilinenMufredatlar table(
		Id		int identity(1,1),
		EskiMufredatId	int,
		YeniMufredatId	int
	)

	-- Gönderilen dersler birleştirilirken müfredat birleştirme yapılması
sonucu silinen müfredatları tesbit et
	insert into @SilinenMufredatlar
	select
		EskiMufredatId,
		YeniMufredatId
	from
		MufredatTekilleme MT (nolock)
		inner join Mufredat_Log L (nolock)
			on L.ID = MT.EskiMufredatId
			and MT.KalanDersId = @KalanDersId
		inner join DersTekillemeDetay DTD (nolock)
			on DTD.EskiDersId = @EskiDersId
			and DTD.YeniDersId = @KalanDersId
			and DTD.TabloAdi = 'Mufredat'
			and DTD.TablodakiID = MT.EskiMufredatId


	-- Silinen müfredat yoksa
	if not exists (select 1 from @SilinenMufredatlar)
	Begin
		select @Basarili = 1
		return
	End

	Begin Transaction

	-- silinen müfredatlar yeniden ekleniyor.

	set Identity_Insert dbo.Mufredat on

	insert into Mufredat(
		ID,
		Birim_Id,
		AY_Id,
		KayitDonem,
		Intibak,
		DersId,
		ILKAPB_ID,
		AP_ID,
		Somestre,
		Ders_Kodu,
		DersGrup,
```

```
            sinif,
            DersPratikKredi,
            DersTeoriKredi,
            DersLabKredi,
            Yil,
            Donem,
            DersKredi,
            DersZorunlu,
            DersTez,
            Seviye,
            OrtEtkisi,
            Kontenjan,
            Statu,
            DersSure,
            SureTip,
            KreEtkisi,
            ECTS,
            PFDersiMi,
            CreateUserId,
            CreateDate,
            ModUserId,
            ModDate
      )
      select
            L.ID,
            L.Birim_Id,
            L.AY_Id,
            L.KayitDonem,
            L.Intibak,
            L.DersId,
            L.ILKAPB_ID,
            L.AP_ID,
            L.Somestre,
            L.Ders_Kodu,
            L.DersGrup,
            L.sinif,
            L.DersPratikKredi,
            L.DersTeoriKredi,
            L.DersLabKredi,
            L.Yil,
            L.Donem,
            L.DersKredi,
            L.DersZorunlu,
            L.DersTez,
            L.Seviye,
            L.OrtEtkisi,
            L.Kontenjan,
            L.Statu,
            L.DersSure,
            L.SureTip,
            L.KreEtkisi,
            L.ECTS,
            L.PFDersiMi,
            L.CreateUserId,
            L.CreateDate,
            L.ModUserId,
            L.ModDate
      from
            Mufredat_Log L (nolock)
            inner join @SilinenMufredatlar SM
```

```
                on EskiMufredatId = L.ID

      set Identity_Insert dbo.Mufredat off

/* müfredat birleştirilirken yapılan güncelleştirmeler geri alınıyor */

-------------------------------------------------------------------------
-- NotKarti --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
      update NK
      set
            APB_ID = SM.EskiMufredatId
      from
            NotKarti NK
            inner join MufredatTekillemeDetay MTD (nolock)
                  on MTD.TabloAdi = 'NotKarti'
                  and MTD.KolonAdi = 'APB_ID'
                  and MTD.TablodakiID = NK.D_Id
            inner join @SilinenMufredatlar SM
                  on SM.EskiMufredatId = MTD.EskiMufredatId
                  and SM.YeniMufredatId = MTD.YeniMufredatId

      update NK
      set
            IAPB_ID = SM.EskiMufredatId
      from
            NotKarti NK
            inner join MufredatTekillemeDetay MTD (nolock)
                  on MTD.TabloAdi = 'NotKarti'
                  and MTD.KolonAdi = 'IAPB_ID'
                  and MTD.TablodakiID = NK.D_Id
            inner join @SilinenMufredatlar SM
                  on SM.EskiMufredatId = MTD.EskiMufredatId
                  and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- NotKartiButunlemeLog --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
      update NKBL
      set
            APB_ID = SM.EskiMufredatId
      from
            NotKartiButunlemeLog NKBL
            inner join MufredatTekillemeDetay MTD (nolock)
                  on MTD.TabloAdi = 'NotKartiButunlemeLog'
                  and MTD.KolonAdi = 'APB_ID'
                  and MTD.TablodakiID = NKBL.D_Id
            inner join @SilinenMufredatlar SM
                  on SM.EskiMufredatId = MTD.EskiMufredatId
                  and SM.YeniMufredatId = MTD.YeniMufredatId

      update NKBL
      set
            IAPB_ID = SM.EskiMufredatId
      from
            NotKartiButunlemeLog NKBL
            inner join MufredatTekillemeDetay MTD (nolock)
                  on MTD.TabloAdi = 'NotKartiButunlemeLog'
                  and MTD.KolonAdi = 'IAPB_ID'
```

```
                and MTD.TablodakiID = NKBL.D_Id
            inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- DersKayit --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
    update DK
    set
        APB_ID = SM.EskiMufredatId
    from
        DersKayit DK
        inner join MufredatTekillemeDetay MTD (nolock)
            on MTD.TabloAdi = 'DersKayit'
            and MTD.KolonAdi = 'APB_ID'
            and MTD.TablodakiID = DK.DA_Id
        inner join @SilinenMufredatlar SM
            on SM.EskiMufredatId = MTD.EskiMufredatId
            and SM.YeniMufredatId = MTD.YeniMufredatId

    update DK
    set
        IAPB_ID = SM.EskiMufredatId
    from
        DersKayit DK
        inner join MufredatTekillemeDetay MTD (nolock)
            on MTD.TabloAdi = 'DersKayit'
            and MTD.KolonAdi = 'IAPB_ID'
            and MTD.TablodakiID = DK.DA_Id
        inner join @SilinenMufredatlar SM
            on SM.EskiMufredatId = MTD.EskiMufredatId
            and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- MufredatIntibaki --> APB1_ID, APB2_ID
-------------------------------------------------------------------------
    update MI
    set
        APB1_ID = SM.EskiMufredatId
    from
        MufredatIntibaki MI
        inner join MufredatTekillemeDetay MTD (nolock)
            on MTD.TabloAdi = 'MufredatIntibaki'
            and MTD.KolonAdi = 'APB1_ID'
            and MTD.TablodakiID = MI.Id
        inner join @SilinenMufredatlar SM
            on SM.EskiMufredatId = MTD.EskiMufredatId
            and SM.YeniMufredatId = MTD.YeniMufredatId

    update MI
    set
        APB2_ID = SM.EskiMufredatId
    from
        MufredatIntibaki MI
        inner join MufredatTekillemeDetay MTD (nolock)
            on MTD.TabloAdi = 'MufredatIntibaki'
            and MTD.KolonAdi = 'APB2_ID'
```

```
                and MTD.TablodakiID = MI.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- NotKartiDuzenlemeLog --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
     update NKDL
     set
          APB_ID = SM.EskiMufredatId
     from
          NotKartiDuzenlemeLog NKDL
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'NotKartiDuzenlemeLog'
                and MTD.KolonAdi = 'APB_ID'
                and MTD.TablodakiID = NKDL.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId

     update NKDL
     set
          IAPB_ID = SM.EskiMufredatId
     from
          NotKartiDuzenlemeLog NKDL
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'NotKartiDuzenlemeLog'
                and MTD.KolonAdi = 'IAPB_ID'
                and MTD.TablodakiID = NKDL.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- OgrencininAlttanDersleri --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
     update OAD
     set
          APB_ID = SM.EskiMufredatId
     from
          OgrencininAlttanDersleri OAD
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'OgrencininAlttanDersleri'
                and MTD.KolonAdi = 'APB_ID'
                and MTD.TablodakiID = OAD.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId

     update OAD
     set
          IAPB_ID = SM.EskiMufredatId
     from
          OgrencininAlttanDersleri OAD
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'OgrencininAlttanDersleri'
                and MTD.KolonAdi = 'IAPB_ID'
```

```
                and MTD.TablodakiID = OAD.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- OgrencininAlttanDersleri_AOF --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
     update OAD
     set
          APB_ID = SM.EskiMufredatId
     from
          OgrencininAlttanDersleri_AOF OAD
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'OgrencininAlttanDersleri_AOF'
                and MTD.KolonAdi = 'APB_ID'
                and MTD.TablodakiID = OAD.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId

     update OAD
     set
          IAPB_ID = SM.EskiMufredatId
     from
          OgrencininAlttanDersleri_AOF OAD
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'OgrencininAlttanDersleri_AOF'
                and MTD.KolonAdi = 'IAPB_ID'
                and MTD.TablodakiID = OAD.Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- UzemAGNOicinDersKayit --> APB_ID, IAPB_ID
-------------------------------------------------------------------------
     update UADK
     set
          APB_ID = SM.EskiMufredatId
     from
          UzemAGNOicinDersKayit UADK
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'UzemAGNOicinDersKayit'
                and MTD.KolonAdi = 'APB_ID'
                and MTD.TablodakiID = UADK.DA_Id
          inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId

     update UADK
     set
          IAPB_ID = SM.EskiMufredatId
     from
          UzemAGNOicinDersKayit UADK
          inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'UzemAGNOicinDersKayit'
                and MTD.KolonAdi = 'IAPB_ID'
```

```
                and MTD.TablodakiID = UADK.DA_Id
            inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- DegisimProgramiDersKayit --> APBID
-------------------------------------------------------------------------
      update DPDA
      set
            APBID = SM.EskiMufredatId
      from
            DegisimProgramiDersKayit DPDA
            inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'DegisimProgramiDersKayit'
                and MTD.KolonAdi = 'APBID'
                and MTD.TablodakiID = DPDA.Id
            inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- MuafiyetNotBilgileri --> M_ID, IM_ID
-------------------------------------------------------------------------
      update MNB
      set
            M_ID = SM.EskiMufredatId
      from
            MuafiyetNotBilgileri MNB
            inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'MuafiyetNotBilgileri'
                and MTD.KolonAdi = 'M_ID'
                and MTD.TablodakiID = MNB.ID
            inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId

      update MNB
      set
            IM_ID = SM.EskiMufredatId
      from
            MuafiyetNotBilgileri MNB
            inner join MufredatTekillemeDetay MTD (nolock)
                on MTD.TabloAdi = 'MuafiyetNotBilgileri'
                and MTD.KolonAdi = 'IM_ID'
                and MTD.TablodakiID = MNB.ID
            inner join @SilinenMufredatlar SM
                on SM.EskiMufredatId = MTD.EskiMufredatId
                and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- MuafiyetNotBilgileri_Log --> M_ID, IM_ID
-------------------------------------------------------------------------
      update MNBL
      set
            M_ID = SM.EskiMufredatId
      from
```

```
            MuafiyetNotBilgileri_Log MNBL
            inner join MufredatTekillemeDetay MTD (nolock)
                  on MTD.TabloAdi = 'MuafiyetNotBilgileri_Log'
                  and MTD.KolonAdi = 'M_ID'
                  and MTD.TablodakiID = MNBL.ID
            inner join @SilinenMufredatlar SM
                  on SM.EskiMufredatId = MTD.EskiMufredatId
                  and SM.YeniMufredatId = MTD.YeniMufredatId

      update MNBL
      set
            IM_ID = SM.EskiMufredatId
      from
            MuafiyetNotBilgileri_Log MNBL
            inner join MufredatTekillemeDetay MTD (nolock)
                  on MTD.TabloAdi = 'MuafiyetNotBilgileri_Log'
                  and MTD.KolonAdi = 'IM_ID'
                  and MTD.TablodakiID = MNBL.ID
            inner join @SilinenMufredatlar SM
                  on SM.EskiMufredatId = MTD.EskiMufredatId
                  and SM.YeniMufredatId = MTD.YeniMufredatId


-------------------------------------------------------------------------
-- Sonuç mesajının döndürülmesi
-------------------------------------------------------------------------
      If @@ERROR = 0
      Begin
            Commit Transaction
            select @Basarili = 1
            --Select 'Message'= 'İşlem başarı ile tamamlandı.'
      End
      Else
      Begin
            Rollback Transaction
            select @Basarili = 0
            --Select 'Message'= 'İşlem yapılırken bir hata oluştu. Lütfen işlemi
yenileyin.'
      End
End
```

EK 4. Ders Birleştirmeyi Geri Alma Saklı Yordamı

```
CREATE procedure [dbo].[usp_admin_DersBirlestirme_GeriAl]
	@YeniDersId	int,
	@EskiDersId	int,
	@UserId	int
as
Begin

	declare
		@BasariliMi	int

	select
		@YeniDersId,
		D1.Ders_Adi,
		d2.Ders_Adi,
		D2.Ders_Id
	from
		Dersler D1 (nolock),
		Dersler D2 (nolock)
	where
		D1.Ders_Id = @YeniDersId
		and D2.Ders_Id = @EskiDersId

	if not exists (select 1 from DersTekilleme where EskiDersId = @EskiDersId
and YeniDersId = @YeniDersId)
	Begin
		Select
			'Hata' = 2,
			'Mesaj' = 'Gönderdiğiniz dersler birleştirilmemiş ya da bu geri
alma işlemi daha önceden yapılmış.'
		return
	End

	-- geri alınmak istenen ders daha sonra başka bir ders ile birleştirilmiş
mi?
	if exists (select 1 from Dersler (nolock) where Ders_Id = @YeniDersId and
Statu != 1)
	Begin

		declare @SonrakiBirlesmeler table
		(
			Id		int identity(1,1),
			EskiDersId int,
			YeniDersId int
		)

		declare
			@YeniDersIdKopya int,
			@Aciklama		varchar(2000) = ''

		select @YeniDersIdKopya = @YeniDersId


		while exists (select 1 from DersTekilleme (nolock) where EskiDersId =
@YeniDersIdKopya)
		Begin
			insert into @SonrakiBirlesmeler
			select
```

```
                        EskiDersId,
                        YeniDersId
                  from
                        DersTekilleme (nolock)
                  where
                        EskiDersId = @YeniDersIdKopya

                  select
                        @YeniDersIdKopya = YeniDersId
                  from
                        DersTekilleme
                  where
                        EskiDersId = @YeniDersIdKopya
            End

            select * from @SonrakiBirlesmeler

            select
                  --@Aciklama += D1.Ders_Adi + ' (' + CAST(D1.Ders_Id as
varchar(10)) + ') dersi --> ' + D2.Ders_Adi + ' (' + CAST(D2.Ders_Id as
varchar(10)) + ') dersi ile birleştirilmiş. '
                  @Aciklama += CAST(Id as varchar(3)) + ') ' +  CAST(D1.Ders_Id as
varchar(10)) + ' --> '+ CAST(D2.Ders_Id as varchar(10)) + ' dersi ile
birleştirilmiş. '
            from
                  @SonrakiBirlesmeler S
                  inner join Dersler D1 (nolock)
                        on D1.Ders_Id = S.EskiDersId
                  inner join Dersler D2 (nolock)
                        on D2.Ders_Id = S.YeniDersId

            select
                  'Hata' = 3,
                  'Mesaj' = 'Geri alınmak istenen birleşim işleminden sonra başka
birleşim işlem(ler)i yapılmış; ' + @Aciklama + ' Öncelikle bu birleştirme
işlem(ler)ini (ters sıra ile) geri almalısınız.'

            --drop table #SonrakiBirlesmeler
            return
      End

      --return


      Begin Transaction

      -- önce müfredat birleştirme işlemi geri alınmalı
      exec usp_admin_DersBirlestirmeSonrasiMufredatTekilleme_GeriAl
            @EskiDersId,
            @YeniDersId,
            @BasariliMi out,
            @UserId

      -- Müferdat birleştirmeyi geri alma işlemi başarılı bir şekilde sonlanmadı
ise, ders birleştirmede gheri alınmasın.
      if @BasariliMi = 0
      Begin
            rollback transaction
            Select
                  'Hata' = 1,
```

```
                'Mesaj'= 'Müfredat birleşirmeyi geri almada yaşan sorun nedeni
ile işleme devam edilemedi.'
            return
      end

-------------------------------------------------------------------------
-- AcilanDersler --> DersId
-------------------------------------------------------------------------

      Update AC
      Set
            DersId = @EskiDersId,
            DersAdi = D.Ders_Adi,
            DersPratikKredi = D.Pratik,
            DersTeoriKredi = D.Teori,
            DersLabKredi = D.Lab,
            DersKredi = D.Kredi
      From
            AcilanDersler AC
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = AC.ID
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'AcilanDersler'
            Inner Join Dersler D (nolock)
                  on D.Ders_ID = @EskiDersId


-------------------------------------------------------------------------
-- DersIliskileri --> Ders1_Id
-------------------------------------------------------------------------

      Update DI
      Set
            Ders1_Id = @EskiDersId
      From
            DersIliskileri DI
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = DI.APH_Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'DersIliskileri'
                  and DTD.KolonAdi = 'Ders1_Id'


-------------------------------------------------------------------------
-- DersIliskileri --> Ders2_Id
-------------------------------------------------------------------------

      Update DI
      Set
            Ders2_Id = @EskiDersId
      From
            DersIliskileri DI
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = DI.APH_Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'DersIliskileri'
                  and DTD.KolonAdi = 'Ders2_Id'
```

```
-----------------------------------------------------------------------
-- Mufredat --> DersId
-----------------------------------------------------------------------

      Update M
      Set
            DersId = @EskiDersId,
            DersTeoriKredi = D.Teori,
            DersLabKredi = D.Lab,
            DersPratikKredi = D.Pratik,
            DersKredi = D.Kredi
      From
            Mufredat M
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = M.ID
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'Mufredat'
            inner join Dersler D  (nolock)
                  on D.Ders_Id = @EskiDersId

-----------------------------------------------------------------------
-- NotKarti --> Ders_Id
-----------------------------------------------------------------------

      Update NK
      Set
            Ders_Id = @EskiDersId,
            Kredi = D.Kredi,
            BdsKatsayi = D.Kredi * Katsayi,
            SAno = D.Kredi * SAno
      From
            NotKarti NK
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = NK.D_Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'NotKarti'
            inner join Dersler D  (nolock)
                  on D.Ders_Id = @EskiDersId


-----------------------------------------------------------------------
-- NotKartiButunlemeLog --> Ders_Id
-----------------------------------------------------------------------

      Update NKBL
      Set
            Ders_Id = @EskiDersId,
            Kredi = D.Kredi,
            BdsKatsayi = D.Kredi * Katsayi,
            SAno = D.Kredi * SAno
      From
            NotKartiButunlemeLog NKBL
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = NKBL.D_Id
                  and DTD.EskiDersId = @EskiDersId
```

```
                and DTD.YeniDersId = @YeniDersId
                and DTD.TabloAdi = 'NotKartiButunlemeLog'
          inner join Dersler D  (nolock)
                on D.Ders_Id = @EskiDersId


-------------------------------------------------------------------------
-- DersKayit --> Ders_Id
-------------------------------------------------------------------------

     Update DK
     Set
          Ders_Id = @EskiDersId
     From
          DersKayit DK
          inner join DersTekillemeDetay DTD (nolock)
                on DTD.TablodakiID = DK.DA_Id
                and DTD.EskiDersId = @EskiDersId
                and DTD.YeniDersId = @YeniDersId
                and DTD.TabloAdi = 'DersKayit'


-------------------------------------------------------------------------
-- HaftalikProgram --> DersId
-------------------------------------------------------------------------

     Update HP
     Set
          DersId = @EskiDersId
     From
          HaftalikProgram HP
          inner join DersTekillemeDetay DTD (nolock)
                on DTD.TablodakiID = HP.DP_Id
                and DTD.EskiDersId = @EskiDersId
                and DTD.YeniDersId = @YeniDersId
                and DTD.TabloAdi = 'HaftalikProgram'


-------------------------------------------------------------------------
-- NotKartiDuzenlemeLog --> Ders_Id
-------------------------------------------------------------------------

     Update NKDL
     Set
          Ders_Id = @EskiDersId
     From
          NotKartiDuzenlemeLog NKDL
          inner join DersTekillemeDetay DTD (nolock)
                on DTD.TablodakiID = NKDL.D_Id
                and DTD.EskiDersId = @EskiDersId
                and DTD.YeniDersId = @YeniDersId
                and DTD.TabloAdi = 'NotKartiDuzenlemeLog'


-------------------------------------------------------------------------
-- OgrencininAlttanDersleri --> Ders_Id
-------------------------------------------------------------------------

     Update OAD
     Set
```

```
            DersId = @EskiDersId
      From
            OgrencininAlttanDersleri OAD
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = OAD.Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'OgrencininAlttanDersleri'


-------------------------------------------------------------------------
-- Sinav --> Ders_Id
-------------------------------------------------------------------------

      Update S
      Set
            Ders_Id = @EskiDersId
      From
            Sinav S
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = S.Sinav_Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'Sinav'


-------------------------------------------------------------------------
-- UzemAGNOicinDersKayit --> Ders_Id
-------------------------------------------------------------------------

      Update UADK
      Set
            Ders_Id = @EskiDersId
      From
            UzemAGNOicinDersKayit UADK
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = UADK.DA_Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'UzemAGNOicinDersKayit'


-------------------------------------------------------------------------
-- OgrencininAlttanDersleri_AOF --> Ders_Id
-------------------------------------------------------------------------

      Update OAD
      Set
            DersId = @EskiDersId
      From
            OgrencininAlttanDersleri_AOF OAD
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = OAD.Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'OgrencininAlttanDersleri_AOF'


-------------------------------------------------------------------------
-- DegisimProgramiDersKayit --> Ders2Id
```

```
------------------------------------------------------------------------

      Update DPDA
      Set
            Ders2Id = @EskiDersId
      From
            DegisimProgramiDersKayit DPDA
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = DPDA.Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'DegisimProgramiDersKayit'


------------------------------------------------------------------------
-- MuafiyetNotBilgileri --> Ders_Id
------------------------------------------------------------------------

      Update MNB
      Set
            Ders_Id = @EskiDersId
      From
            MuafiyetNotBilgileri MNB
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = MNB.Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'MuafiyetNotBilgileri'


------------------------------------------------------------------------
-- MuafiyetNotBilgileri_Log --> Ders_Id
------------------------------------------------------------------------

      Update MNBL
      Set
            Ders_Id = @EskiDersId
      From
            MuafiyetNotBilgileri_Log MNBL
            inner join DersTekillemeDetay DTD (nolock)
                  on DTD.TablodakiID = MNBL.Id
                  and DTD.EskiDersId = @EskiDersId
                  and DTD.YeniDersId = @YeniDersId
                  and DTD.TabloAdi = 'MuafiyetNotBilgileri_Log'


------------------------------------------------------------------------
-- Dersin yeniden aktif edilmesi
------------------------------------------------------------------------

      Update D
      Set
            Statu = 1
      From
            Dersler D
      where
            Ders_Id = @EskiDersId


------------------------------------------------------------------------
```

```
-- DersTekilleme ve DersTekillemeDetay tablolarında temizlik yapılması
-----------------------------------------------------------------------

	--delete from
	--	DersTekilleme
	--where
	--	EskiDersId = @EskiDersId
	--	and YeniDersId = @YeniDersId

	update DersTekilleme
	set
		Durum = 2  -- 2: İşlem geri alındı
	where
		EskiDersId = @EskiDersId
		and YeniDersId = @YeniDersId


	--delete from
	--	DersTekillemeDetay
	--where
	--	EskiDersId = @EskiDersId
	--	and YeniDersId = @YeniDersId

	update DersTekillemeDetay
	set
		Durum = 2  -- 2: İşlem geri alındı
	where
		EskiDersId = @EskiDersId
		and YeniDersId = @YeniDersId


-----------------------------------------------------------------------
-- Sonuç mesajının döndürülmesi
-----------------------------------------------------------------------

	If @@ERROR = 0
	Begin
		Commit Transaction
		Select
			'Hata' = 0,
			'Mesaj'= 'İşlem Başarı İle Tamamlandı'
	End
	Else
	Begin
		Rollback Transaction
		Select
			'Hata' = 1,
			'Mesaj'= 'İşlem yapılırken bir hata oluştu. Lütfen işlemi
yenileyin'
	End
End
```

EK 5. Damerau-Levenshtein Distance Fonksiyonu

```
CREATE function [dbo].[fn_DamerauLevenshteinDistance](
      @Ifade1     nvarchar(150),
      @Ifade2     nvarchar(150),
      @Limit            int
)
RETURNS int
AS
begin

      Declare
            @i          int = 0,
            @j          int = 0,
            @a          int,
            @b          int,
            @Maliyet    int,
            @YeniDeger int,
            @Ekleme     int,
            @Cikarma    int,
            @Degisim    int,
            @TMaliyet   int,
            @Uzunluk1   int,
            @Uzunluk2   int,
            @Uzunluk3   int,
            @Ifade3     nvarchar(150)

      Declare @temp table(
            i     int,
            j     int,
            deger int
      )

      select
            @Uzunluk1 = len(@Ifade1),
            @Uzunluk2 = len(@Ifade2)

      if @Uzunluk1 < @Uzunluk2
      begin
            select @Ifade3 = @Ifade2
            select @Ifade2 = @Ifade1
            select @Ifade1 = @Ifade3

            select @Uzunluk3 = @Uzunluk2
            select @Uzunluk2 = @Uzunluk1
            select @Uzunluk1 = @Uzunluk3
      end


      while (@i <= @Uzunluk1)
      begin
            insert into @temp (
                  i,
                  j,
                  deger
            )
            values(
                  @i,
                  0,
```

```
                  @i
            )

            set @i = @i+1
      end --while

      while (@j <= @Uzunluk2)
      begin
            insert into @temp (
                  i,
                  j,
                  deger
            ) values(
                  0,
                  @j,
                  @j
            )

            set @j = @j+1
      end --while

      --select * from @temp

      set @i = 1
      while (@i <= @Uzunluk1)
      begin
            set @j = 1
            while (@j <= @Uzunluk2)
            begin
                  if (substring(@Ifade1, @i, 1) = substring(@Ifade2, @j, 1))
                        set @Maliyet = 0
                  else
                        set @Maliyet = 1

                  -- toplama, çıkarma, değiştirme maliyetleri
                  set @Ekleme  = (select deger + 1 from @temp where i = @i - 1 and
j = @j)
                  set @Cikarma = (select deger + 1 from @temp where j = @j - 1 and
i = @i)
                  set @Degisim = (select distinct deger from @temp where i = @i - 1
and j = @j - 1)
                  set @Degisim = @Degisim + @Maliyet
                  set @YeniDeger = dbo.Minimum(dbo.Minimum(@Ekleme, @Cikarma),
@Degisim)

                  insert into @temp (
                        i,
                        j,
                        deger
                  ) values(
                        @i,
                        @j,
                        @YeniDeger
                  )

                  if (@i > 1
                        and @j > 1
                        and substring(@Ifade1, @i, 1) = substring(@Ifade2, @j-1, 1)
                        and substring(@Ifade1, @i-1, 1) = substring(@Ifade2, @j,
1))
```

```
                  begin
                        set @a = (select deger from @temp where i = @i and j = @j)
                        set @b = (select distinct deger from @temp where i = @i - 2
and j = @j - 2)
                        set @b = @b + @Maliyet
                        set @TMaliyet = dbo.Minimum(@a, @b)

                        Update @temp
                        set
                              deger = @TMaliyet
                        where
                              i = @i
                              and j = @j
                  end

                  --select * from @temp where i = @i and j = @j

                  select @j += 1
            end --while

            --select * from @temp where i = @i order by j
            if @Limit != -1
            Begin
                  if not exists (select 1 from @temp where i = @i and deger <=
@Limit)
                  Begin
                        return -1
                        --return null
                  End
            End

            select @i +=1

      end --while

      --select
      --    @Uzaklik = deger
      --from
      --    @temp
      --where
      --    i=@Uzunluk1
      --    and j=@Uzunluk2

      --return   @Uzaklik
      return (select
                        deger
                  from
                        @temp
                  where
                        i = @Uzunluk1
                        and j = @Uzunluk2
                  )
end
```

EK 6. Bir Ders İçin Benzer Derslerini Tespit Eden Saklı Yordam

```
CREATE procedure [dbo].[usp_admin_DersBenzerlikTesbit]
	@Ders1Id	int
as
Begin

	set nocount on;

	Declare @KarsilastirilacakDersler table(
		Id		int identity(1,1),
		Ders2Id		int,
		Ders2Adi	nvarchar(150),
		Limit		int
	)

	Declare @Sonuc table(
		Id		int identity(1,1),
		Ders1Id		int,
		Ders2Id		int,
		Limit		int,
		DLD		int
	)

	Declare
		@Uzunluk	int,
		@MaxUzunluk		int,
		@MinUzunluk		int,
		@Ders2Id	int,
		@Ders1Adi	nvarchar(150),
		@Ders2Adi	nvarchar(150),
		@Sayac1		int = 1,
		@Sayac2		int,
		@Basarili	int = 0,
		@Limit			tinyint,
		@BuyukLimit		tinyint,
		@KucukLimit		tinyint,
		@MaxDers2Id		int


	select
		@Uzunluk = Uzunluk,
		@MaxUzunluk = MaxUzunluk,
		@MinUzunluk = MinUzunluk,
		@Ders1Adi = DersAdi,
		@BuyukLimit = CEILING(Uzunluk * 0.125),
		@KucukLimit = CEILING(Uzunluk * 0.1)
	from
		dbo.TDA (nolock)-- TDA: Tekil Ders Adları
	where
		Id = @Ders1Id

	select
		@MaxDers2Id = MAX(Ders2Id)
	from
		dbo.DK (nolock)		-- DK: Ders Karşılaştırma
	where
		Ders1Id = @Ders1Id
	--group by
```

```
--    Ders2Id

insert into @KarsilastirilacakDersler
select
      Id,
      DersAdi,
      case
            when Uzunluk > @Uzunluk then @BuyukLimit
            else @KucukLimit
      end
from
      dbo.TDA (nolock)
where
      Uzunluk between @MinUzunluk and @MaxUzunluk
      and Id < @Ders1Id
      and Id > isnull(@MaxDers2Id, 0)
order by
      Id

--select
--    '@Uzunluk' = @Uzunluk,
--    '@MaxUzunluk' = @MaxUzunluk,
--    '@MinUzunluk' = @MinUzunluk,
--    '@KucukLimit' = @KucukLimit,
--    '@BuyukLimit' = @BuyukLimit,
--    '@Ders1Adi' = @Ders1Adi,
--    '@MaxDers2Id' = @MaxDers2Id

--select * from @KarsilastirilacakDersler
--return

select
      @Sayac2 = MAX(Id)
from
      @KarsilastirilacakDersler

while @Sayac1 <= @Sayac2
Begin
      Select
            @Ders2Id = Ders2Id,
            @Ders2Adi = Ders2Adi,
            @Limit = Limit
      from
            @KarsilastirilacakDersler
      where
            Id = @Sayac1

      --insert into dbo.DK
      insert into @Sonuc
      select
            @Ders1Id,
            @Ders2Id,
            @Limit,
            dbo.fn_DamerauLevenshteinDistance(@Ders1Adi, @Ders2Adi, @Limit)

      if @Sayac1 % 20 = 0
      Begin
            insert into DK
            select
                  Ders1Id,
```

```
                    Ders2Id,
                    Limit,
                    DLD
               from
                    @Sonuc

               delete from @Sonuc
          End

          Select @Sayac1 += 1
     End

     insert into DK
     select
          Ders1Id,
          Ders2Id,
          Limit,
          DLD
     from
          @Sonuc

     select @Ders1Id

     set nocount off;

End
```

EK 7. Tüm Dersler İçin Benzer Dersleri Tespit Eden Saklı Yordam

```
CREATE procedure [dbo].[usp_admin_DersBenzerlikTesbit_TumDersler]
	@MinDersId int,
	@Aralik	int = 250
as
Begin
	set nocount on;

	Declare @Dersler table
	(
		Id	int identity(1,1),
		DersId	int
	)

	Declare
		@DersId	int,
		@Sayac1	int = 1,
		@Sayac2	int,
		@Basarili	int = 0,
		@limit		int,
		@Tm		tinyint

	insert into @Dersler
	Select
		Id
	from
		TDA (nolock)
	where
		Id between @MinDersId and @MinDersId + @Aralik - 1
		and ISNULL(Bitti, 0) = 0
	order by
		Id

	select
		@Sayac2 = MAX(Id)
	from
		@Dersler

	while @Sayac1 <= @Sayac2
	Begin
		Select
			@DersId = DersId
		from
			@Dersler
		where
			Id = @Sayac1

		exec dbo.usp_admin_DersBenzerlikTesbit @dersId

		update TDA set Bitti = 1 where Id = @DersId

		Select @Sayac1 += 1
	End

	set nocount off;

End
```

EK 8. Ders Talebinde Bulunurken Dikkat Edilmesi Gereken Hususlar

- Ekletmek istediğiniz dersin veri tabanında olup olmadığı kontrol edilmeli; veri tabanında yok ise talepte bulunulmalıdır.
- Ekletmek istenilen dersin saat bilgileri ve kredisinin çerçeve yönetmeliğin ilgili maddesine göre uyumlu olmasına dikkat edilmelidir.
- Dersin kısa adı: dersin adı 40 karakterden az ise boş bırakılmalı; diğer durumlarda 40 karakteri geçmeyecek şekilde anlamlı bir kısaltma yazılmalıdır.
- Dersin adını kelimenin sadece ilk harfi büyük olacak şekilde yazılmalıdır. (Örnek: "Sosyal Bilimlerde Araştırma Yöntemleri"). Fakat TBMM, TCMB, DNA … gibi yaygın olarak bilinen kısaltmalar küçültülmemelidir.
- Dersin adında tire (-) ve slaş (/) kullanılmamalıdır. (Örnek: "Türk Dili-II" şeklinde değil "Türk Dili II" şeklinde olmalı.)
- Numaralandırma için Roma Rakamları kullanılmalıdır. (Örnek: "Türk Dili 2" şeklinde değil "Türk Dili II" şeklinde olmalı.)
- Dersin adında parantez kullanılmamalıdır.
- "Dersin Adı" sütununda sadece dersin adını yazılmalı; herhangi bir açıklama, not vs. yazılmamalıdır.
- Bu form sadece yeni ders ekletmek için kullanılmalıdır.

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Yakup BAYOĞLU
Doğum Yeri ve Tarihi	Pasinler-ERZURUM / 16.04.1986
Eğitim Durumu	
Lisans Öğrenimi	İstanbul Teknik Üniversitesi Bilgisayar Mühendisliği
Bildiği Yabancı Diller	İngilizce
Bilimsel Faaliyetleri	
İş Deneyimi	
Stajlar	
Projeler	- Atatürk Üniversitesi Öğrenci Bilgi Sistemi
Çalıştığı Kurumlar	- Atatürk Üniversitesi İktisadi ve İdari Bilimler Fakültesi
İletişim	
E-Posta Adresi	ybayoglu@atauni.edu.tr
Tarih	09.07.2018