İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VERİ AMBARLARINDA VERİLERİN TEMİZLENMESİ

YÜKSEK LİSANS TEZİ

Bilgisayar Müh. Metin ÇINAR

Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Programı : BİLGİSAYAR MÜHENDİSLİĞİ

MAYIS 2003

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VERİ AMBARLARINDA VERİLERİN TEMİZLENMESİ

YÜKSEK LİSANS TEZİ

Bilgisayar Müh. Metin ÇINAR
504001566

Tezin Enstitüye Verildiği Tarih : 5 Mayıs 2003
Tezin Savunulduğu Tarih : 29 Mayıs 2003

Tez Danışmanı : Prof.Dr. Eşref ADALI

Diğer Jüri Üyeleri : Prof.Dr. Tevfik AKGÜN (Y.T.Ü.)

Doç.Dr. Coşkun SÖNMEZ (İ.T.Ü.)

MAYIS 2003

ÖNSÖZ

Tez çalışmam süresince teknik anlamda yardımlarını esirgemeyen sayın hocam Prof. Dr. Eşref Adalı'ya bana her konuda destek olan aileme ve son olarak da benden desteğini esirgemeyen herkese sonsuz teşekkürlerimi sunarım.

Mayıs 2003 Metin ÇINAR

İÇİNDEKİLER

KISALTMALAR

SNM : Sorted Neighbourhed Method
DSS : Decision Support Systems
OLAP : On-line Analyitical Processing
SQL : Structured Query Language
DDİ : Doğal Dil İşleme

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

VERİ AMBARLARINDA VERİLERİN TEMİZLENMESİ

ÖZET

Günümüz teknolojik seviyesinde bilgisayarın hayatın her alanında kullanılmasından dolayı her an çok büyük boyutlarda veri saklanması, kaydedilmesi ya da toplanması zorunluluk halini almıştır. Herhangi bir alışveriş işleminde, bir telefon görüşmesinde ya da bir bankacılık işleminde kaydedilen veriler çok büyük veri dağlarının ortaya çıkmasına yol açmaktadır. Bu tip günlük ihtiyaçlar dışında ayrıca uydu, uzay araçları ya da uzaktan algılayıcı gibi sistemlerin muazzam boyutlarda verilerin saklanmasını gerektirdiği de aşikardır.

Günlük işlemlerin yerine getirilmesi amacıyla kullanılan canlı veri tabanı sistemlerinin belirli bir anlamsal bütünlük içerisinde bir araya getirildiği sistemler veri ambarı sistemleri olarak adlandırılırlar. Veri ambarına kaynaklık eden sistemlerin her biri genellikle birbirinden bağımsız olarak tasarlanmış ve geliştirilmiş sistemlerdir. Bunlar aynı organizasyon bünyesinde yer alabilecekleri gibi tamamen organizasyon harici sistemler de olabilirler. Bu verilerin bir veri ambarı sisteminde belirli bir şema yapısı ile anlamsal bütünlük içerisinde bir araya getirilmesi ile veri ambarı inşaa edilmiş olur.

Veri tabanı sistemlerinin artan kullanımı ve veri miktarlarındaki büyük artışlar işletme ya da organizasyonları bu verilerden yararlanma amacıyla çeşitli çözümler üretmeye itmiştir. Veri ambarlarının asıl kullanım amaçları içerdikleri verilerden yararlanılarak işletmelerin gelecekteki stratejilerinin belirlenmesi, kâr oranlarının artırılması gibi yönetimsel kararların alınmasıdır.

Bir veri ambarı sisteminden gerçek anlamıyla yararlanabilmenin ilk koşulu veri ambarında yer alan verilerin tutarlılığı ve doğruluğudur. Sistemde yapılacak analizlerin ve araştırmaların güvenilirliği, üzerinde çalışılan verilerin kalitesi ile doğrudan ilişkilidir. Veri kalitesinin düşük olması pek çok sebepten kaynaklanabilmektedir. Kaynak veri tabanlarının tasarım aşamasında çok iyi tasarlanmaması, bütünlük kısıtlayıcılarının yeterince etkin olarak kullanılmaması, veri girişinden kaynaklanan hatalar gibi çeşitli hatalar ile karşılaşılabilmektedir. Kaynak veri tabanlarının çok iyi tasarlanmış olması ya da tamamen hatasız olması da çoğu zaman veri ambarı sistemindeki veri kalitesi problemlerini gidermemektedir. Birleşmeden kaynaklanan karışık(heterojen) şema yapısı ve olası tekrarlı kayıt problemleri veri kalitesini düşürmektedir. Veri kalitesinin artırılması için gerekli olan çalışmaların üzerinde yeterince durulmaması daha sonraki aşamalarda çok daha yüksek maliyetli çalışmalar yapılmasını zorunluluk haline getirecektir.

Bu çalışmada veri ambarı sistemlerinde karşılaşılan çeşitli veri kalitesi problemleri sınıflandırılmış ve veri kalitesinin artırılması için kullanılan çeşitli teknikler üzerinde durulmuştur. Veri ambarlarında giderilmesi en güç olan ve üzerinde en fazla çalışılan konuların başında tekrarlı kayıtların tespit edilip ayıklanması gelmektedir. Diğer problemler ise nispeten daha basit bazı yöntemler kullanılarak giderilebilmektedir. Özellikle tekrarlı ve mükerrer kayıtların ortaya çıkma sebeplerinin başında yazım hataları gelmektedir. Tamamen aynı olan iki kayıt sisteme girilirken sezilip engellenebilir, fakat girilen kayıtlar arasında yazım farklılıkları olduğunda bunların veri girişi sırasında sezilmesi pek mümkün değildir. Bunun için bu çalışmada öncelikle yazım hatalarının belirlenebilmesi için sözlük kullanılmadan, Türkçe'ye özgü kurallardan yararlanılarak yazım hatalarının tespit edilmesi amaçlanmıştır. Bunun yanında n-gram metodu ile istatistiksel olarak da yazım denetiminin yapılması amaçlanmıştır. Daha sonraki adımda ise sistemde yer alan tekrarlı kayıtların belirlenip ayıklanması amacıyla sıralı komşu metodu olarak bilinen yöntemin bazı ek kurallar ile zenginleştirilerek uygulaması yapılmıştır.

DATA CLEANING IN DATA WAREHOUSES

SUMMARY

Today, computers are used in everypart of our life, for this reason, very big amount of data must be stored, saved or collected. When you are shopping, speaking on phone or banking, you generate data. In addition to these daily requirements, satellites, space vehicles and remote perceivers generate great amount of data which must be stored too.

A data warehouse is a repository of information gathered from multiple sources, stored under a unified schema. Data has been periodically collected from various discrete operational systems that has been used to perform daily operations and pushed into the warehouse by preserving the consistency of the data warehouse. A number of analysis and investigations has been done on the data warehouse that contains data which is not update. Some of these operations are statistical reporting, multi-dimensional analysis and data mining.

Excessive use of database systems and increasing amount of data forced companies and organizations to find a practical solution for taking advantage of these useful data. Data warehouses enables companies to determine future straregies of companies, increasing profit rates and other managerial decisions.

If a data warehouse doesn't contain consistent and reliable data, it is not possible to benefit from this warehouse, efficiently. The reliablity of analysis and researches generated from data warehouses is directly related with quality of data that stored by warehouse. The low quality of data appears by many reasons. Such as, the bad design of the source databases, insufficient use of integrity restrictions, poor design of data entry tools and human factor during data entry. Even if the accomplishment of terms above is performed, it is not enough for high quality of data in data warehouses. Because, heterogenous schema structures, duplicate records, inconsistent records, erroneous records may be generated during merging source databases. If the criterias mentioned above are not considered and planning of data warehouse system is not done accordingly, it will be more expensive to solve these problems in the future.

In this thesis data quality problems are classified and the different methodologies applicable to the variety of data cleaning problems are presented. As mentioned above, the main data quality problem in a data warehouse is duplicate records, so main consideration of this work is detection and elimination of the duplicate records. Other data quality problems can be solved rather easier methods. Incorrect or missing data values, inconsistent value naming conventions, and incomplete information cause "dirty" data files. Hence, it is not surprise to encounter multiple records referring to the same real world entity. Exactly same records can be detected easily

during data entry but if there are slightly differencies, it is almost impossible to detect them at data entry step. Therefore, at first step it is aimed to detect spelling errors using Turkish grammer rules without using a Turkish dictionary. In addition to these grammer rules n-gram statistic techniques are used to increase succesfully detected misspelled words. For this goal, di-gram, tri-gram and four-gram statistic tables generated using some turkish corpus and Turkish spelling guide**.** At second step, it is aimed to detect and eliminate duplicate records in the system using enriched sorted neighbourhood method(SNM) with field weights.

1. VERİ AMBARI KAVRAMI

1.1. Veri Ambarı Tanımı

Günümüzde büyük firmaların herbiri çok farklı yerlerde çok büyük hacimlere sahip verileri bulunabilmektedir.Bu tür firmalarının yüzlerce, hatta binlerce farklı yerde şubeleri bulunabilmektedir. Örneğin sigortacılık ya da bankacılık sektöründe yer alan firmalar çok fazla sayıda şubeye sahiptir ve bu şubelerden elde edilen verilerin toplam miktarı çok büyük hacimlere erişebilmektedir. Ayrıca, büyük organizasyonlar farklı türdeki verilerin farklı lokasyonlarda tutulmasından veya farklı işletim sistemlerine ya da farklı şemalara sahip olmalarından dolayı oldukça karmaşık iç yapılara sahiptir. Şirketlerdeki stratejik karar verme yetkisine sahip olan yöneticilerin bütün bu şekildeki dağıtılmış canlı sistemlerde yer alan bilgilere erişebilmesi gereklidir. Herbir veri kaynağı üzerinde tek tek yapılacak sorgulamalar doğru ve tutarlı bilgi alma ve karar verme konusunda yetersiz kalacaktır. Ayrıca herbir kaynaktaki veriler muhtemelen güncel veriler olacağından, karar vericilerin eski verilere ulaşması da bu canlı veritabanları üzerinde pek mümkün olamayacaktır. Veri ambarı sistemleri bu problemlere çözüm sunabilmektedir.

Veri ambarı, bir işletmenin ya da kurumun çeşitli birimleri tarafından canlı sistemler aracılığı ile toplanan verilerin, karar destek sistemlerinde(DSS) ya da çeşitli analizlerde kullanılabilmesi amacıyla biraraya getirilmesi sonucu oluşan çok büyük ölçekli bir merkezi veri deposudur. Burada sözü edilen canlı sistem kavramı literatürde genellikle "operasyonel veri bloğu" olarak adlandırılmaktadır.

Canlı sistemlerde güncel veriler bulunduğundan dolayı sürekli olarak bu verilere erişmek ve gerektiğinde arama, güncelleme, silme ya da kayıt ekleme gibi standart işlemleri yapmak gerekmektedir.Günlük yapılan işlemleri gerçeklemek ve sonuçları saklamak için tasarlanan bu sistemlere uçak ya da otobüs şirketleri tarafından kullanılan çevrimiçi(on-line) rezervasyon işlemlerinin yapıldığı bilgi sistemleri örnek

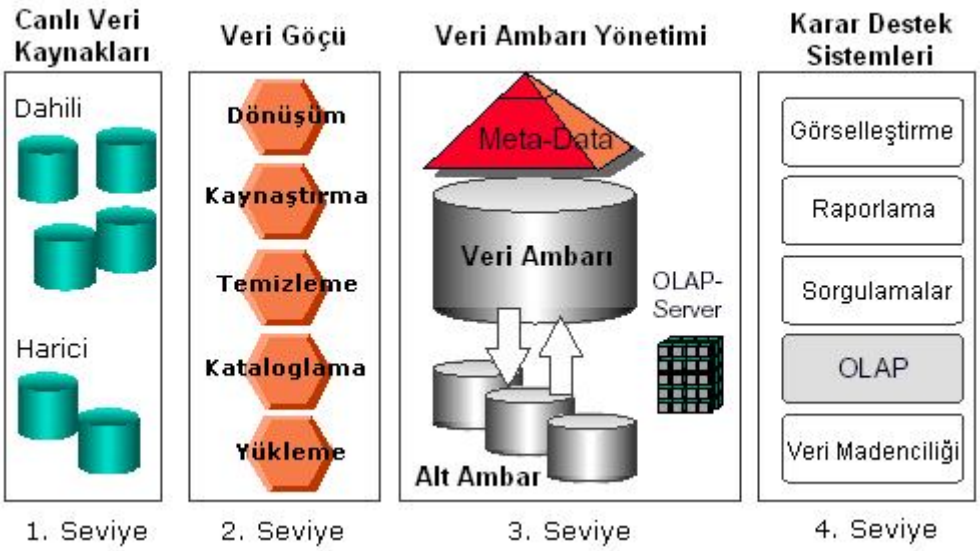
olarak verilebilir. Bu tür sistemler gerçek zamanlı sistemler olduklarından işin en kısa sürede tamamlanması öncelikli hedeftir. Veriye hızlı erişimin bu sistemler için kritik öneme sahip olmasından dolayı canlı sistemler çevrimiçi çalışma prensibine dayalı olarak tasarlanırlar.

Veri ambarlarının asıl kullanım amaçları içerdikleri veriler üzerinde çeşitli incelemeler ve araştırmalar yapılarak "*karar destek sistemleri*" olarak kullanılmalarıdır. Bu araştırmalar, işletmenin gelecekteki stratejilerinin belirlemesi, pazar araştırmaları yapılarak müşterilerin demografik yapılarının belirlenmesi, müşteri tercihleri doğrultusunda yeni alanlara açılım gibi çok çeşitli şekillerde işletme yöneticileri tarafından değerlendirilerek kullanılmaktadır. Bundan dolayı veri ambarlarında veriye erişim canlı sistemlerde olduğu gibi kritik bir öneme sahip değildir. Buradaki asıl kritik nokta çok büyük boyuttaki veriler kullanılarak yapılan işlemlerin mümkün olduğunca yüksek performansla yapılmasıdır. Çok büyük boyutlu veriler üzerinde yapılan incelemelerin ve araştırmaların sistemde aşırı kaynak sarfiyatına sebep olmaması için performansı artıracak tedbirlerin alınmasını gerekmektedir.

1.2. Veri Ambarı Mimarisi

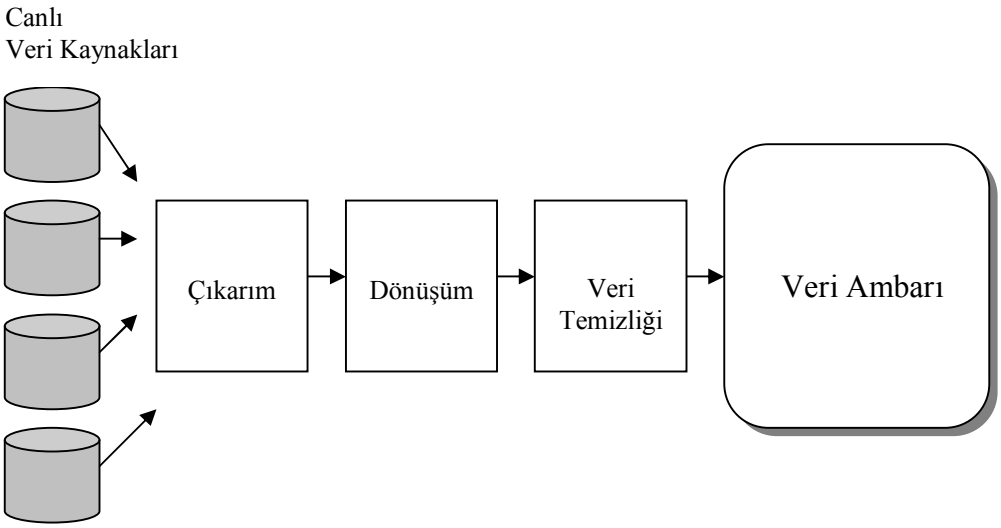
Canlı sistemler tarafından üretilen veriler belirli periyotlarla veri ambarına aktarıldıklarından veri ambarında yer alan veri miktarı sürekli olarak artış göstermektedir. Aktarım periyotları tamamen işletme ihtiyaçları göz önünde bulundurularak belirlenir, bu periyot gün mertebesinde olabileceği gibi hafta ya da ay mertebesinde de olabilir. Bundan dolayı herhangi bir anda veri ambarında yer alan kayıtların güncel kayıtlar olması beklenemez ve çok büyük oranda yer alan kayıtlar güncel değildir. Veri ambarını besleyen canlı sistemlerin hepsi aynı özelliklere sahip olmayabilir. Yani verilerin alındığı kaynaklar aynı işletmenin alt bölümlerinde kullanılan veri tabanları olabileceği gibi tamamen farklı işletmelere ait olan veri tabanları da olabilir. Bundan dolayı genellikle veri ambarı için kaynak olarak kullanılan veri kaynakları farklı şemalara sahip olduklarından veri ambarına aktarımdan önce bazı dönüşüm ve şema değişimleri zorunlu olmaktadır. Bu aktarım sürecinde daha önce veri ambarında yer alan verilerin tekrar aktarılmaması, bazı kayıtların güncellenmesi veya silinmesi gibi işlemlerinde yapılması gerekli

olmaktadır. Bu ön işlemlerin tamamlanması ile veriler veri ambarına tek bir şema yapısına uygun, anlamsal bir bütünlük içinde yerleştirilebilecektir. Şekil 1.1 de tipik bir veri ambarı mimari yapısı verilmiştir.



Şekil 1.1 – Veri ambarı mimarisi

Kaynak veritabanlarında yer alan ham verilerin karar destek sistemlerinde kullanılabilir hale getirilmeleri için çeşitli bazı işlemlere tabi tutulmaları gerektiğinden bahsedilmişti. Temel olarak uygulanması gerekli olan işlemler şunlardır.



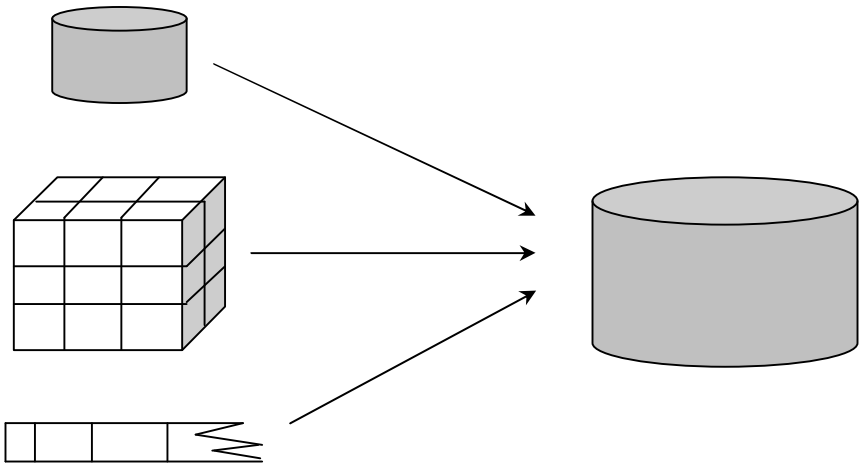
Şekil 1.2 – Ham verilerin veri ambarına aktarım süreci

- *Çıkarım(Extraction)*: Farklı kaynaklardaki farklı formatlara sahip olan verilerin elde edilmesi aşaması.

- *Dönüşüm(Transformation)*: Kaynaklarda yer alan ham veriler gerektiğinde karar destek sistemleri için en uygun şekilde dönüştürülürler. Örneğin iki farklı kaynakta aynı türden sayısal değerler bulunuyorsa bu değerlerin birleşmeden sonra anlamlı olabilmeleri için normalizasyon işleminden geçirilmeleri gerekli olabilir.

- *Temizleme(Data Cleaning/Cleansing)*: Eksik veriler mümkünse uygun değerler kullanılarak doldurulur. Hatalı veriler düzeltilir. Tutarsızlıklar ortadan kaldırılır. Karar destek sistemlerinden doğru ve güvenilir sonuçlar elde edilebilmesinin ilk şartı üzerinde çalışılan verilerin tutarlı ve doğru olmasıdır. Veri ambarı sistemlerinde üzerinde en fazla durulması gereken işlemlerin başında verileri temizlenmesi gelmektedir.

- *Kaynaştırma(Integration)*: Farklı özelliklere sahip kaynaklardan elde edilen veriler merkezi veri ambarında belirli bir şema yapısı altında kaynaştırılır(birleştirilir). Kaynaklar veritabanları, veri dosyaları, veri küpleri olabilir.

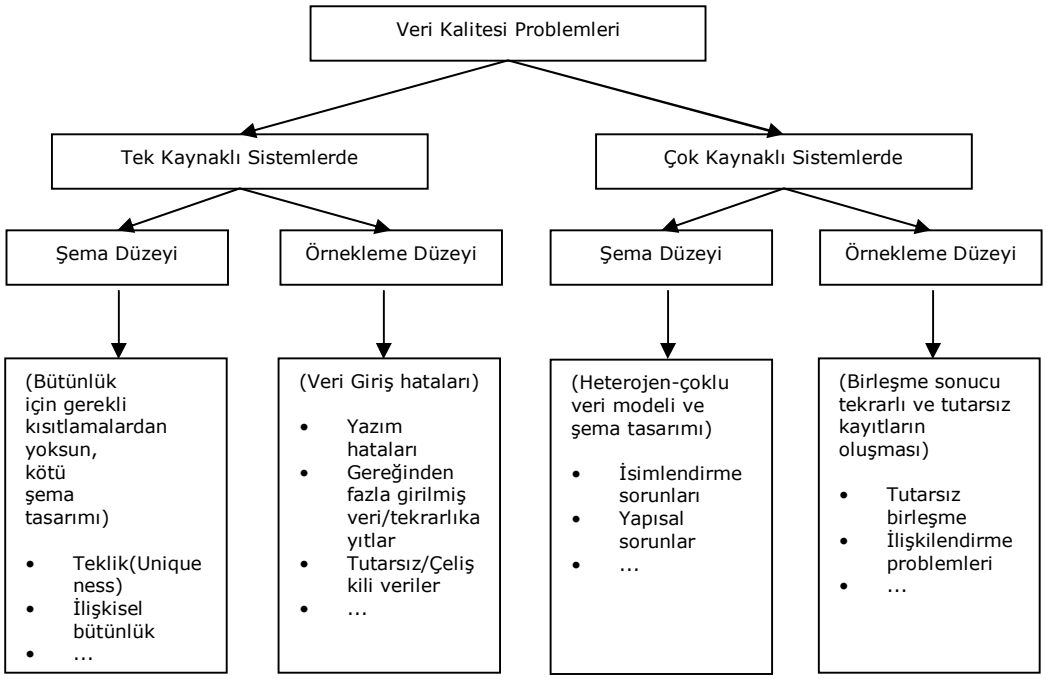


Şekil 1.3 – Farklı kaynaklardaki kayıtların veri ambarında birleştirilmesi

2. VERİ KALİTESİ PROBLEMLERİ

Bu bölümde verilerin temizlenmesi ve dönüştürülmesi ile çözümlenebilen başlıca veri kalitesi sorunları sınıflandırılmaktadır. Daha sonra da görülebileceği gibi bu problemler birbirlerine oldukça yakındır ve bundan dolayı birlikte ele alınmaları uygun olacaktır. Verilerin dönüştürülmesi yapısal, gösterimsel ya da veri içeriğinin değişimi şeklinde olabilir. Pek çok durumda bu dönüşümlerin yapılması bir zorunluluktur. Örneğin eski bir sistemden yeni bir bilgi sistemine geçilmesi ya da farklı veri kaynaklarının birleştirilmesi şema değişimini zorunluluk haline getirmektedir.

Veri Temizleme Problemleri Şekil 2.1 de görüldüğü gibi kabaca tek-kaynak ve çok-kaynaklı problemler olarak ikiye ayrılabilir. Bunlar da kendi aralarında şema düzeyi ve örnek-düzeyi olarak ikiye ayrılabilir. Şema düzeyindeki problemler aynı zamanda örnekleme seviyesindeki hatalara da sebep olmaktadırlar. Örnekleme düzeyindeki hatalar ya da tutarsızlıklar ise şema düzeyinde olmayan gerçek veri üzerindeki hatalardır. Şekil 2.1 de değişik durumlardaki tipik hatalar verilmiştir. Şekilde verilmediği halde tek-kaynaklı problemlerin tamamı çok-kaynaklı sistemler için de geçerlidir.



Şekil 2.1 – Veri kalitesi problemlerinin kabaca sınıflandırılması

2.1. Veri Temizliğinin Gerekliliği

Bir veri ambarının gerçek anlamda etkin kullanılabilmesi için gerekli olan en önemli unsur veri ambarının içerdiği verilerin tutarlılığı ve gerçekliğidir. Eğer veri ambarında yer alan bilgiler de hatalar, tutarsızlıklar, kayıt tekrarları varsa bu durumda bu sistem üzerinde yapılacak olan sorgulamalar neticesindeki alınacak kararlar önemli ölçüde hatalı olacaktır. Bundan dolayı bir veri ambarı sitemininin oluşumundaki en önemli adımların başında verilerin temizlenmesi gelmektedir. Bu adım üzerinde yeterince durulmaması daha sonra çok daha maliyetli çözümler üretilmesini zorunlu kılacaktır.

Aşağıdaki listede veri temizliğinin önemine işaret eden bazı örnekler yer almaktadır.

- *Pazarlama Sektöründe İletişim*: Hatalı adresler, hatalı yazılmış isimler, aynı kişiye birden fazla mektup gönderilmesine sebep olmaktadır.

- *Hedefe Yönelik Pazarlama*: Müşterilere ait demografik ve davranış özelliklerinin doğruluğu ve eksiksiz olması oldukça önemlidir.

- *Dış veriler ile iç verilerin birleştirilmesi*: Ürün isimleri, müşteri isimleri ya da coğrafik isimler uyuşmayabilir.

2.2. Tek Kaynaklı Sistemlerde Karşılaşılan Veri Kalitesi Problemleri

Bir kaynakta yer alan verilerin kalitesi şema(tasarım) ve bütünlük kısıtlayıcılarının izin verdiği veri değerlerine doğrudan bağımlıdır. Veri dosyaları gibi, hangi verilerin girileceği ve sistemde tutulacağı üzerinde çok fazla kısıtlama olmayan, belirli bir şemaya sahip olmayan kaynaklarda hata ve tutarsızlıklara rastlanması olasılığı çok daha yüksektir. Diğer taraftan veritabanı sistemleri ise uygulamaya özel kısıtlamaları içerebilir. (ilişkisel veritabanı yaklaşımı basit özellik değerleri ve referans bütünlüğünün sağlanması). Bundan dolayı şema ile ilgili veri kalitesi problemleri uygulamaya özel, uygun bütünlük kısıtlamalarının kullanılmamasından ileri gelmektedir. Örneğin veri modeli kısıtlamalarından, kötü şema tasarımdan ya da bütünlük kısıtlamaları için az sayıda tanımlama yapılmış olması bu sebeplerden bazılarıdır. Örnekleme seviyesi hataları şema seviyesinde engellenemeyecek hatalardır. (Örn: yazım hataları ya da veri girişinde kullanılan cihazlardan kaynaklanan hatalar.)

Tablo 2.1 – Tek kaynaklı sistemlerdeki şema düzeyindeki problemler.

Sorun		**Hatalı Veri**	**Sebepler/Görüşler**
Özellik	Hatalı veri	Dtarihi=25.15.1970	Değer alan sınırlarını aşmış
Kayıt	Kuralsız özellik bağımlılıkları	Yaş=20, Dtarihi=11.01.1977	Yaş=(bugünkü tarih – doğum tarihi) şeklinde tutulmalıdır
Kayıt Tipi	Tek olması gereken değerin birden fazla kullanılması	İşçi1=(adı="Ali Koşar",SSKNo="12345") İşçi2=(adı="Aylin Gezer",SSKNo="12345")	SSK numarasının tek olması gereklidir.
Kaynak	Referans bütünlüğü hatası	İşçi=(adı="Ali Yiğit",bölümno=145)	145 numaralı bölüm tanımlanmamış

Şema ve örnekleme düzeyi problemleri için farklı problem alanları oluşturulabilir. Tablo 2.1 ve Tablo 2.2 de değişik durumlar için bazı örnekler verilmiştir. Dikkat edilirse şema seviyesinde tanımlanan teklik(uniqueness) kısıtlamaları tekrarlı kayıtları engelleyememektedir. Örneğin aynı gerçek dünya değeri farklı değerler kullanılarak birden fazla girilirse tekrarlı kayıtlar ortaya çıkabilmektedir.

Tablo 2.2 – Örnekleme düzeyindeki tek kaynaklı sistemlerdeki problemler.

Sorun		**Hatalı Veri**	**Sebepler/Görüşler**
Özellik	Eksik değerler	Tel=999-9999999	Mevcut olmayan veri girişi (null değerler)
	Hatalı yazım	Şehir="İstanbol"	Genelde yanlış anlama ve klavye hataları
	Kısaltmalar	Meslek="Bilg. Müh"	
	Gömülü değer	İsim="A. Yiğit 12.09.1970 Ankara"	Birden fazla değerin aynı alana girilmesi
	Hatalı yere yazım	Şehir="Türkiye"	
Kayıt	Özellik bağımlılık hatası	Şehir="istanbul" plaka="38"	şehir ve plaka uyuşmazlığı var
Kayıt Tipi	Sözcüklerin yer değişmesi	İsim1="A. Yiğit", İsim2="Mehmet C. "	Genelde giriş formundan kaynaklanır
	Tekrarlı kayıtlar	İşçi1=(isim="Ali Yiğit",...) İşçi2=(isim="A. Yiğit",...)	Veri giriş hatalarından dolayı bazı kayıtlar tekrarlıdır
	Çelişkili kayıtlar	İşçi1=(isim="Ali Yiğit",dtarihi="11.01.1970") İşçi2=(isim="Ali Yiğit",dtarihi="11.11.1970")	Aynı veri iki farklı değer olarak girilmiş (Hangi kaydın doğru olduğu belirsiz)
Kaynak	Hatalı Referans	İşçi1=(isim="Ali Yiğit",bölüm=18)	18 numaralı bölüm referans verilmiş fakat doğru değil

Görüldüğü gibi bu hataların bulunup ortaya çıkarılması oldukça zor ve maliyetli bir süreç gerektirir. Bundan dolayı veri girişi sırasında hatalı veri girişini önleyici tedbirlerin alınması veri temizlemedeki en önemli adımdır. Bu da uygun bir veritabanı şeması tasarımı ve bütünlük kısıtlamalarının yanında iyi bir veri giriş uygulamasına bağlıdır.

2.3. Çok Kaynaklı Sistemlerdeki Problemler

Birleştirilmesi gereken kaynaklarda sorun olması birleştirme sürecini daha da ağırlaştırmaktadır. Her kaynak kalitesiz veri içerebilir ve kaynaklardaki veriler farklı gösterime sahip olabilirler. Hatta bu kaynaklardaki veriler birbirleriyle çelişki içerisinde olabilirler. Çünkü genellikle kaynakların herbiri farklı görevlerde kullanılmak üzere diğerlerinden bağımsız olarak geliştirilmiş ve kullanılmıştır. Bu durum birleşme sonrası yüksek dereceli heterojenliğe(karışıklık) sahip bir veri modeli, şema tasarımı ve verilerin ortaya çıkmasına sebep olur.

Şema düzeyinde, veri modeli ve şema tasarımı farklılıkları sırasıyla şema dönüşümü ve şema bütünleşmesi adımları olarak gerçeklenir. Şema tasarımındaki asıl sorun isimlendirme ve yapısal uyuşmazlık sorunlarıdır. İsimlendirme sorunları farklı nesnelerin aynı isimleri kullanmasından (eşadlı) ya da aynı nesnelerin farklı

isimlendirilmesinden kaynaklanmaktadır(eşanlamlı). Yapısal uyumsuzluklar ise farklı kaynaklardaki aynı nesnelerin değişik şekillerde temsil edilmesinden kaynaklanmaktadır ve çok farklı şekillerde olabilir. Örneğin tablo gösterimleri, farklı bileşen yapıları, farklı veri yapıları, farklı bütünlük kısıtlamaları gibi...

Şema düzeyi uyuşmazlıkların yanısıra, pekçok uyuşmazlık da örnekleme seviyesinde ortaya çıkmaktadır. Tek kaynaklı sistemlerdeki bütün problemler farklı şekillerde çok kaynaklı sistemlerde de ortaya çıkmaktadır. (Örn: tekrarlı kayıtlar, çelişkili kayıtlar vs.) Bununla birlikte, aynı özellik isimleri ve veri tipleri kullanılsa bile bunlar farklı değerleri temsil edebilir(örn: cinsiyet) veya kayıtlar arasında farklı şekilde yorumlanabilir(örn: ölçüm birimleri Dolar veya Euro gibi).

Çok kaynaklı sistemlerdeki veri temizliğindeki asıl sorunlardan biri kayıtların üst üste binme problemidir. Yani gerçek dünyada denk olan kayıtların birleşmeden sonra tekrarlı kayıt haline gelmesi durumudur. Bu problem literatürde aynı zamanda *object elimination problem*, *duplicate elimination* veya *merge/purge problem* olarak da bilinir. Genelde bilgi kısmen eksiktir ve kaynaklar birbirlerini tamamlayarak kayıt hakkındaki ek bilgileri sağlarlar. Bundan dolayı tekrarlı kayıtlar tamamlayıcı bilgiler aracılığı ile temizlenmeli ve tek kayıt halinde birleştirilmelidirler.

Tablo 2.3 – Bir numaralı bir kaynağa ait "müşteri" isimli tablo

Mno	***İsim***	***Adres***	***Şehir***	***Cinsiyet***
11	Mahmut Gezer	Aydoğan Mah.Selvi Sok.	58456 Sivas	1
24	Melahat Gezer	Aydoğan Mah.Selvi sok.	Sevas	0

Tablo 2.4 – İki numaralı bir kaynağa ait "alıcı" isimli tablo

ANo	***Soyad***	***Ad***	***Adres***	***Tel/Faks***	***Cinsiyet***
24	Gezer	Mehmet	Aydoğdu Mah. Kavak Sok No:45 06451 Ankara	222-1111111/ 333-5555555	K
493	Gezer	Mamut A.	Aydoğan Mah.Selvi Sok. No:11 58456 Sivas	888-4444444	E

Tablo 2.5 – Temizlenmiş "müşteriler" hedef tablo

No	*Soyad*	*Ad*	*C*	*Adres*	*Şehir*	*PK*	*Tel*	*Faks*	*Mno*	*Ano*
1	Gezer	Mahmut A.	E	Aydoğan Mah. Selvi Sok	Sivas	58456	888-44444444		11	493
2	Gezer	Mehmet	E	Aydoğdu Mah. Kavak Sok.	Ankara	065451	222-11111111	333-444444	24	
3	Gezer	Melahat	K	Aydoğan Mah.Selvi sok.	Sivas	58456				24

Örnekteki iki kaynakta ilişkisel yapıdadır fakat şema ve veri uyuşmazlığını göz önüne sermektedir. Şema seviyesinde, isimlendirme uyuşmazlıkları (*müşteri/alıcı,Mno/Ano*) ve yapısal uyuşmazlıklar (*isim ve adres alanındaki farklı gösterimler*) görülebilmektedir. Örnekleme seviyesinde ise cinsiyet gösterimindeki farklılık ("0/1" ve "K/E") ve muhtemelen tekrarlı bir kayıt dikkat çekmektedir. Son bir gözlem ise *Mno/Ano* değerlerinin herbir kaynağa özel ve karşılaştırılabilir değerler olmadığıdır, bu değerlerin farklı(11/493) olduğu kayıtlar aynı kişiyi gösterebilir aynı zamanda bu değerlerin aynı(24) olduğu kayıtlar farklı kişileri gösterebilir. Bu problemlerin çözümü şema bütünlüğü ve verilerin temizlenmesini gerektirmektedir; üçüncü tablo muhtemel bir çözümü göstermektedir. Veri temizliğinin yapılmasından önce şema uyumsuzluğunun giderilmesi gerektiği açıktır.(Aynı gösterime sahip isim ve adres alanları ve cinsiyet değerlerinin denkleştirilmesi).

2.4. Oluşabilecek Hataların Tipik Olarak Sınıflandırılması

Birbirlerinden bağımsız olarak tasarlanmış ve kurulmuş olan veri kaynaklarının birbirlerinden farklı şemalara sahip olması doğaldır. Hatta herbir kaynağın farklı veri modellerine sahip olması da olasıdır. Veri ambarı oluşum aşamasında şema bütünleştirme ve verilerin birleştirilmesinden önce şema dönüşümünün yapılması gereklidir. Sonuç olarak veri ambarında saklanacak olan veriler sadece kaynaklardaki verilerin bir kopyası değildir, aslında kaynaklardaki verilerin bir izdüşümüdür.

Veri ambarındaki veri kirliliği şu şekilde sınıflandırılabilir.

- **Eksik Veri**

1. Verinin olmasının beklendiği fakat olmadığı durumlar(Bazen veritabanı tasarımındaki eksiklikten dolayı gerekli bir veri de sistemde var olmayabilir)
2. Verinin olduğu ama uygun olmadığı ya da uygulanabilir olmadığı durumlar

Eksik verilerin ortaya çıkarılması genellikle çok karmaşık değildir.

- **Hatalı Veriler**

  Gerçekte olması gereken bir değer yerine farklı bir değerin tutulmasıdır. Bu ortaya çıkarılması oldukça zor olan bir hata türüdür.(Ör. Bir ismin hatalı yazılması)

- **Tekrarlı Veri** iki şekilde olabilir

  1. Bazı alanları farklı değerlere sahip tekrarlı kayıtlar
  2. Gerçek dünyaya ait bir değerin farklı tanımlamaları(örneğin doğum tarihinin iki farklı değer olarak girilmesi, bu kayıtların eş kayıtlar olduğu tespit edilse bile hangisinin doğru olduğu bilinemez)

- **Karışık(heterojen) Veriler**

  Birden fazla kaynak kullanılarak oluşturulan veritabanlarında ortaya çıkar.

  1. Yapısal karışıklık
  2. Anlamsal karışıklık. Farklı kaynaklarda aynı türden verilerin farklı anlamlarda kullanılması

Tablo 2.6 – Hatalı ve eksik veriler için örnekler

ID	Adı	Soyadı	Yaş	Gerçek	Sınıflandırma
1254C	Mehmet	Yılmaz	42	Mehmet Yılmaz 42 yaşındadır.	Doğru
4512A	Metin	Yeşilyurt		Metin Yeşilyurt 18 yaşındadır	Eksik
124SA	Fatma	Doğan	25	Fatma Doğan 16 yaşındadır.	Hatalı

Tablo 2.7 – Basit tekrarlı kayıt örneği

ID	Adı	Soyadı	Saat	Tekrar Durumu	Sınıflandırma
4121S	Onur	Ş.	2:55		
6554A	Ali	Yiğit	4:40		
66AQF	Onur	Şentürk	3:54	4121S ile aynı kayıt	Semantik
6554A	Ali	Yiğit	4:40	Tekrarlı kayıt	Basit tekrar

Şema Uyuşmazlıkları

Farklı şemalardaki nesnelerin değişik gösterimlerinden dolayı şemalar arasında çatışma ya da uyuşmazlıklar ortaya çıkabilir. Bu uyuşmazlıklar isimlendirme ve yapısal olarak iki şekilde ortaya çıkabilir.

İsimlendirme uyuşmazlığı: Aynı organizasyonun farklı uygulama alanlarında çalışan insanlar aynı veri için değişik terminoloji ve isimlendirme kullanabilirler. Bu durum muhtemel bir tutarsızlığa yol açabilecektir. *Eşsesli* ve *eşanlamlı* olarak iki şekilde ortaya çıkabilir.

Eşseslilik iki farklı kavram ya da büyüklük için aynı ismin kullanılmasıdır.

Örn: Bir şemada DONANIM alanı, "bilgisayar" anlamı taşırken, bir başkasında "mobilya" için kullanılması

Eşanlamlılık ise aynı kavramın iki ya da daha fazla isimle tanımlanmasıdır.

Örn: MUSTERI ya da ALICI

Eşseslilik, farklı şemalardaki aynı isme sahip kavramların karşılaştırılması ile ortaya çıkarılabilir, eşadlı olma durumu ise ancak dışarıdan yapılacak bir tanımlama ile belirlenebilir.

Aynı kavramın aynı isimlere sahip olduğu fakat verilen değerlerin uyuşmadığı bazı durumlarda da yine eşseslilik sorunu ortaya çıkabilir. Bu durum çok çeşitli seviyelerde ortaya çıkabilmektedir. Örneğin, özellik seviyesinde bir şemada "beden" giysi bedeni(tek haneli tamsayı) olarak kullanılmış, bir başka şemada ise pantolon bedeni(iki haneli tamsayı) olarak kullanılmışsa bu durumda yine şamalar arasındaki eşseslilik sorunu vardır.

Yapısal uyulmazlık: Bu şekildeki uyuşmazlıklar modellemedeki farklı seçimlerden ve bütünlük kısıtlayıcılarından kaynaklanmaktadır.

Tip Uyuşmazlıkları: Aynı kavramın farklı modellemelerde farklı şema yapıları içerinde yer almasından dolayı kaynaklanmaktadır. Örneğin bir şema yapısında bütün bir yapı olarak ele alınan bir nesne sınıfı bir başka şema içerisinde bir özellik olarak ele alınması.

Bağımlılık Uyuşmazlıkları: Bir grup kavramın farklı şemalar içerisinde değişik bağımlılıklara sahip olmasından kaynaklanmaktadır. Bir şema içerisinde 1:1 (işçi-bölüm) olan ilişki bir başka şema içerinde m:n (işçi-çalışılan bölümler) şeklinde ise ilişkisel uyuşmazlık ortaya çıkmaktadır.

Anahtar Alan Uyuşmazlıkları: Farklı şemalardaki aynı kavramlar için farklı anahtar alanların belirlenmesinden dolayı ortaya çıkan uyuşmazlıklardır. Örneğin, SSK no ve IsciID iki farklı şemadaki işçi tablosundaki anahtar alanlar ise anahtar uyuşmazlığı ortaya çıkmaktadır.

Davranışsal Uyuşmazlıklar: Farklı şemalardaki veri giriş ve silme kontrollerinden kaynaklanan uyuşmazlıklardır. Örneğin, bir şemada bir bölüm hiç işçiye sahip olmasa bile varlığını sürdürebilirken, bir başkasında bir bölümde çalışan son işçinin bu bölümden ayrılması ile bu bölümünde sistemden silinmesi iki şema arasındaki tipik bir davranışsal uyuşmazlıktır.

3. VERİ TEMİZLEME YÖNTEMLERİ

Bu bölümde hataların ve tutarsızlıkların tespit edilmesi ve giderilmesi için uygulanması gerekli olan teknikler üzerinde durulmaktadır.

Aşağıda odaklanılacak alanlar kategorize edilmiştir.

- Türkçe yazım hatalarının belirlenmesi
- Eksik değerler
- Aykırı değerler
- Tutarsız kodlar
- Şema bütünlüğü
- Tekrarlamalar

Veri hataları ile ilgili genel bir sınıflandırma 2. bölümde yapılmıştır.

Veri tabanlarında yer alan bütün hataların otomatik araçlar kullanılarak giderilmesi mümkün değildir. Bundan dolayı pek çok hata için insan müdahalesi zorunludur.

3.1. Türkçe için sözlüksüz yazım denetimi

Yazım hatalarının çok büyük bir bölümünde sözcüklerde bir karakter eksik, bir karakter fazla ya da iki karakterin ters sırada yazılması görülmektedir. Bir sözcükte eğer birden fazla karakter hatası yer alıyorsa bu sözcükteki hatanın sezilmesi olasığı yükselecektir. Özellikle Türkçe için düşünüldüğünde, yapılan hatalar eğer sözcükteki kurallı yapının bozulmasına neden oluyorsa bu hatanın sezilmesi kolaylaşmaktadır.

Türkçe, yapısı itibariyle kurallı bir dil olduğundan Türkçe yazım denetiminin bu kurallardan faydalanılarak yapılabileceği rahatlıkla söylenebilir. Bir sözcüğün doğru yazılıp yazılmadığının sınanması ve hataların bulunması, üzerinde çok fazla çalışılan bir konudur. Türkçe dahil pek çok dil için çeşitli yazım denetimi çalışmaları

yapılmaktadır, fakat bunların neredeyse tamamına yakınında bir sözlük kullanılması gerekmektedir. Bu çalışmada ise, amaç herhangi bir sözlük kullanılmadan Türkçe yazım denetiminin yapılabilmesi için izlenmesi gerekli olan adımların ve kuralların ortaya çıkarılmasıdır. Daha önce yapılmış Doğal Dil İşleme(DDİ) çalışmalarında doğrudan sözlüksüz yazım denetimi ile ilgili bir çalışma yer almadığından bu tez çalışmasında ortaya konulan yaklaşımlar bundan sonraki çalışmalar için kaynaklık edebilecektir. Özellikle Türkçe kurallara ek olarak di-gram ve tri-gram tablolarından faydalanılması hatalı sözcüklerin bulunma olasılığını artırabilecektir.

3.1.1. Türkçe'de heceler

Türkçe'de altı temel hece yapısı bulunmaktadır. Aşağıda hece yapıları verilirken S sesli ve Z sessiz harfleri simgelemek amacıyla kullanılmıştır.

- Bir sesliden meydana gelen hece yapısı (S) : **e**-vi-miz
- Bir sesli ve bir sessizden meydana gelen hece yapısı (SZ) : **el**-di-ven
- Bir sessiz ve bir sesliden meydana gelen hece yapısı (ZS) : **ye**-mek
- Bir sesli, bir sessiz ve bir sesliden meydana gelen hece yapısı (SZS) : **kay**-bet-mek
- Bir sesli ve iki sessizden meydana gelen hece yapısı (SZZ) : **ilk**-ba-har
- Bir sessiz, bir sesli ve iki sessizden meydana gelen hece yapısı (ZSZZ) :**kork**-mak

Tükçe'de kullanılan bazı yabancı kökenli sözcükler bu altı hece yapısına uymazlar. Örneğin gram, tren, elektrik ... gibi.

3.1.1.1. Hece yapısı için bazı kurallar

1. Dört harfli hecelerde (ZSZZ) son iki harf kesinlikle farklıdır.
2. Üç harfli (SZZ) türünden hecelerde son iki harf kesinlikle farklıdır.
3. Bir sözcükteki hece sayısı o sözcükte yer alan sesli harf sayısı kadardır.
4. Tek harfli heceler (S) kesinlikle ilk hece olmak zorundadır.
5. İki harfli (SZ) türünden heceler kesinlikle ilk hece olmak zorundadır.
6. Üç harfli (SZZ) türünden heceler kesinlikle ilk hece olmak zorundadır.

3.1.1.2. Türkçe sözcük heceleme yöntemi

Sözcük ilk sesli bulunana kadar sondan başa doğru taranır. Bu sesliden önceki sessizde alınarak ilk hece bulunur. Ardından yeni bir sesli bulunana kadar başa doğru ilerlenir ve yeni hece bir önceki adımdaki gibi bulunur. Bu işlem sözcük sonlanana kadar sürdürülür. Bu şekilde sözcüklerin hecelerine ayrıştırılmasında yabancı dillerden Türkçe'ye geçmiş olan sözcükler dikkate alınmamaktadır. Ayrıca Türkçe bir ek olan -*imtrak* eki de Türkçe hece kurallarına uymamaktadır. Bu nedenle bir hece bulunurken eğer *r* harfinden önce *t* varsa bu harf de heceye dahil edilmektedir.

İçerdiği heceler bu altı kurala uymayan bir sözcük hatalı yazılmış bir sözcük olarak değerlendirilebilir. Ancak Türkçe'de kullanılan fakat bu kurallara uymayan sözcüklere de rastlanmasının olası olduğu göz ardı edilmemelidir. Örneğin elektrik sözcüğü hecelerine ayrıştırıldığında **trik** hecesinden dolayı bu sınamalar sonucunda hatalı olarak değerlendirilmesi gerekecektir. Bu tip durumlarla karşılaşılmaması için bu gibi aykırı sözcüklerin bir tabloda tutulması gerekebilir. Bir sözcükte hece yapısı uyuşmazlığı saptandığında doğrudan hatalı olarak değerlendirilmesi yerine bu tabloda yer alıp almadığı kontrol edilebilir. Eğer bu tabloda yer alıyorsa hatalı olarak değerlendirilmez. Bu tip sözcük sayısının normal hece yapısına uyan sözcük sayısının yanında çok daha az olduğu düşünüldüğünde bu çözümün oldukça mantıklı olduğu anlaşılacaktır.

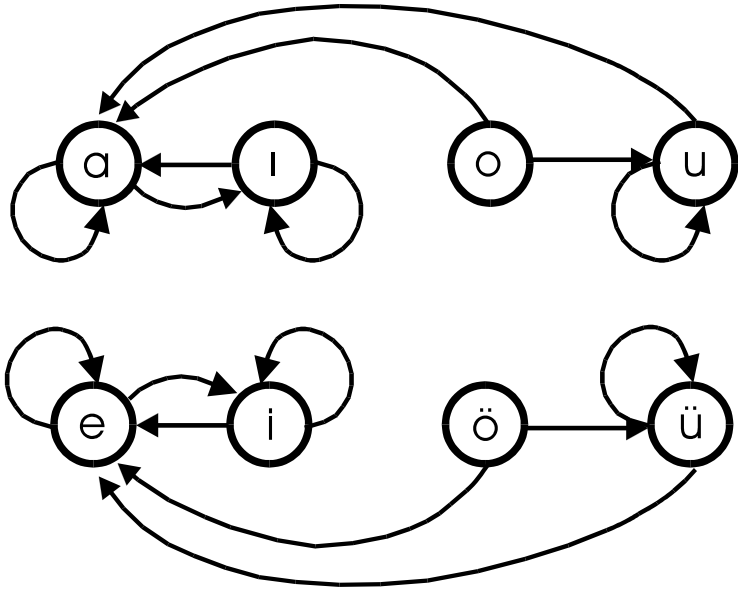
3.1.2. Sesli uyumu

Türkçe sözcüklerdeki seslilerin dizilişi belirli kurallara bağlıdır. Şekil 3.1 de bu kurallar durum diyagramı ile verilmiştir.

Kural olarak ince seslilerden sonra ince sesliler, kalın seslilerden sonra ise kalın sesliler gelir. Buna **büyük sesli uyumu** adı verilir. Bir diğer kural ise; bir sözcüğün ilk hecesinin seslisi düz seslilerden biri ise sonraki hecelerin seslileri de düz sesli olur, ilk hecesinin seslisi yuvarlak seslilerden biri ise sonraki hecelerin seslileri ya dar yuvarlak ya da düz geniş seslilerden biri olur. Bu kural **küçük sesli uyumu** olarak adlandırılır.

Ancak sesli uyumuna uymayan bazı durumlar da vardır. Örneğin bileşik sözcüklerde sesli uyumu olmayabilir, yabancı dillerden alınmış sözcüklerin de bir kısmı bu kurala

uymayabilir. Yine bazı Türkçe ekler de sesli uyumuna uymazlar, bunlara örnek olarak –yor, -ken, -ki, -leyin, -gil vs. ekleri verilebilir.



Şekil 3.1 – Türkçe'de seslilerin diziliş kuralları

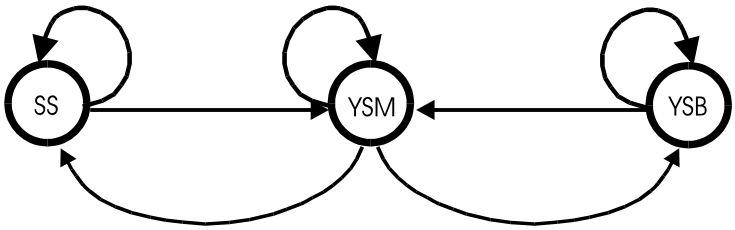
Yukarıda bahsedilen nedenlerden ötürü sesli uyumuna uymayan sözcük sayısının oldukça fazla olmasından dolayı sesli uyumunun doğrudan uygulanması halinde çok sayıda hatalı sonuca ulaşılması olasılığı yüksektir. Bunun için sesli uyumuna uymayan sözcüklerin bir aykırı sözcükler tablosunda tutulması gerekmektedir. Aksi halde bu kuralın uygulanması pek doğru değildir.

3.1.3. Sessiz uyumu

Türkçe'de sesizlerin sözcük içinde dizilişleri de kurallara bağlanmıştır. Türkçe'de sessiz uyumu sözcüklerde yanyana gelen sessizlerin ton bakımından birbirine uygun olmasına dayanır. Bu uyum kuralı Şekil 3.2 de durum makinesi ile açıkça belirtilmiştir. Buna ünsüz uyumu veya ünsüz benzeşmesi denir. Örnek: a**ş-ç**ı, a**t-k**ı, i**ş-ç**i, ta**ş-t**an, Tür**k-ç**e.

Tablo 3.1 – Sessiz uyumu için sessizlerin sınıflandırılması

Sert Sessizler(SS)	**ç, f, h, k, p, s, ş, t**
Sert Karşılığı Bulunmayan Yumuşak Sessizler (YSM)	**l, m, n, r, y**
Sert Karşılığı Bulunan Yumuşak Sessizler (YSB)	**b, c, d, g, ğ, j, v, z**



Şekil 3.2 – Sessiz uyumu için durum makinesi

3.1.4. Hece ve sözcük sonunda bulunabilecek çift sessizler

Türkçe'de bir diğer sessiz uyum kuralı sözcüğün veya hecenin sonunda bulunabilecek çift sesssizle ilgilidir. Türkçe sözcüklerde iki sessiz başta bulunmaz ve sözcüklerde başta ve sonda üç sessiz bulunmaz. Türkçe'de sözcük ve hece sonunda bulunan sessiz çiftlerine ilişkin tablo aşağıda verilmiştir (Tablo 3.2). `kalk, kart, kalp, ilk,kast` sözcükleri bu kurala uyarlar.

Tablo 3.2 – Hece ve sözcük sonunda çift sessiz kuralı

L	**ç, k, p, t**
N	**ç, k, t**
R	**ç, k, p, s, t**
S	**t**
Ş	**t**

Bu kuralların sözcüklere ait hecelerin sınanması esnasında kullanılması hatalı hecelerin bulunması noktasında ayırıcılığı artıracaktır.

3.1.5. Türkçe'de sözcük sonunda bulunamayan sessizler

Türkçe sözcüklerin sonunda *b, c, d, g* sessizleri bulunmaz. Alıntı sözcüklerdeki bu sesler sert karşılıkları olan *p, ç, t, k* sessizlerine çevrilir:
âhen**g** → ahen**k**, fer**d** → fer**t**, ihrâ**c** → ihra**ç**, kitâ**b** → kita**p**, kal**b** → kal**p**, leven**d** → leven**t**
Ad, sac, od, öd gibi sözcükler istisnadır.

3.1.6. Türkçe için n-gram kayan pencere yapısının uygulanması

Bir n-gram (n=1,2,3... harfler) test edilecek olan sözcükten alınan karakter katarının alt kümesidir. Sözcüklerin alt kümesi olan bu n-gramlar daha sonra bir n-gram tablosundan aranırlar. Eğer sözcük çok az rastlanan n-gramlara sahip ise ya da hiç kullanılmayan n-gramlara sahip ise büyük ihtimalle hatalı yazılmıştır.

Örneğin: İkili di-gram arama tablosu

Eğer Türkçe sözcükler için bir tablo oluşturulacak ise 31x31'lik (â ve î harfleri imla klavuzunda yer aldığından bu harfler de alınmıştır) alfabenin bütün iki karakterli kombinasyonlarını içeren bir dizi oluşurulur. Herbir dizi elemanı di-gram'ın herhangi bir sözcükte bulunup bulunmadığını gösteren 1 ya da 0 değerine sahiptir.

Bu yöntem veri kümesi içerisindeki tekrarlı kayıtların ortaya çıkarılması amacıyla genişletilebilir. Örneğin "müşteri adı" özelliği için üç harfli bir kayan pencere kullanıldığında. "Mehmet" kelimesi için 3-gramlar meh, ehm, hme, met şeklinde olacaktır. Veri kümesinin veritabanınından okunması ile birlikte 3-gramlarda oluşturulursa ve daha sonra kayıtlar arasındaki benzerlik bunlardan yararlanılarak belirlenirse tekrarlı kayıtlar belirlenebilir.

Bu çalışmada oluşturulan n-gram tabloları için http://www.dilimiz.gen.tr internet sitesinde yer alan imla klavuzundan yararlanılmıştır. Bu klavuzda toplam olarak yaklaşık 60.000 birbirinden farklı sözcük yer almaktadır. Ayrıca imla klavuzunda sözcüklerin ek almış hallerinin bulunmamasından dolayı ve bu türden sözcüklerinde n-gramların oluşturulması aşamasında kullanılması gerekli olduğundan bir adet 350 sayfalık teknik bir kitap ve iki adet ortalama 60'ar sayfalık tez çalışmasından da sözcükler elde edilerek kullanılmıştır. Sonuç olarak yaklaşık 70.000 farklı sözcükten yararlanılarak di-gram, tri-gram ve four-gram frekansları elde edilmiştir. Aşağıda bu tabloların oluşturulma aşamalarından bahsedilmiştir.

3.1.6.1. İkili harf çiftleri (di-gram) tablosunun oluşturulması

Di-gram tablosunun oluşturulabilmesi için Türkçe imla kurallarına uygun olarak yazılmış sözcüklerin ikili karakter grupları halinde ayrıştırılması ve herbir di-gram'ın kaç adet bulunduğunun bir tabloya işlenmesi gerekmektedir. Yukarıda bahsedilen 70.000 sözcük üzerinde yapılan istatistik sonucunda toplam olarak 534495 di-gram'ın yer aldığı görülmüştür. Bu di-gramların dağılımı aşağıdaki gibidir.

Tablo 3.3 – Türkçe sözcükler için harf çiftleri tablosu

Harf Çifti	Frekans
aâ	0
aı	0
bç	0
bt	0
cş	0
cp	0
⋮	⋮
öf	42
ph	42
hn	42
hb	42
mt	44
oç	44
⋮	⋮
ak	8802
la	8964
an	9108
me	10257
ma	12587

31*31=961

+ ≅ 540000 Toplam harf çifti sayısı

140 adet di-gram kaynak olarak kullanılan sözcüklerin hiçbirinde yer almamaktadır. (aı,bç,bt,cp...)

3.1.6.2. Üçlü harf (tri-gram) tablosunun oluşturulması

Tri-gram tablosu da aynen di-gram tablosunun oluşturulma mantığından hareket edilerek oluşturulabilir. 70.000 sözcük üzerinde yapılan istatistik sonucunda toplam olarak 440997 tri-gram'ın yer aldığı görülmüştür. Türkçe alfabede yer alan 29 harfin yanında â ve î harfleri de imla klavuzunda yer aldığından toplam 31*31*31=29.791 adet üçlü harf kombinasyonu oluşturulabilir.

Tablo 3.4 – Türkçe sözcükler için harf üçlüleri tablosu

Harf Üçlüsü	Frekans
abc	0
abm	0
acç	0
bye	0
cdu	0
ddö	0
⋮	⋮
asf	11
ape	11
aid	11
ahy	12
avz	12
hım	14
⋮	⋮
lik	2345
lık	2413
ama	2649
mek	4147
mak	4454

31*31*31=29791

+ ≅ 440000
Toplam harf üçlüsü sayısı

Tri-gram tablosunda yer alan 21538 trigram'ın yararlanılan sözcük kümesinde yer alan sözcükler tarafından içerilmediği görülmüştür. (abm, bye, ahg, bye...)

3.1.6.3. Dörtlü harf grupları (four-gram) tablosunun oluşturulması

Dörtlü harf grupları için bütün kombinasyonların toplam sayısı 923.521 dir. Fakat elde edilen istatistik sonuçları oldukça ilginçtir. Bu kombinasyonlardan sadece 33.387 tanesi Türkçe sözcüklerde kullanılmaktadır. Yani 890.134 kombinasyon herhangi bir Türkçe sözcükte yer almamaktadır. Buradan şu sonuca varmak mümkün olabilir. Türkçe'de ekler kullanılarak çok fazla sayıda farklı sözcük elde edilebilir, fakat elde edilen sözcüklerde kullanılan harf kombinasyonlarının sayısı çok yüksek değildir. Four-gram tablosunun çok fazla sayıda kombinasyon içermesinden dolayı, yazım denetimi esnasında frekansı sıfır olan kombinasyonlar tablodan çıkarılmıştır. Aksi halde tabloda yapılacak arama işlemi oldukça yavaş yapılabilmektedir.

Tablo 3.5 – Türkçe sözcükler için harf dörtlüleri tablosu

Harf Dörtüsü	Frekans
abko	0
abkö	0
acud	0
adse	0
ahja	0
akge	0
⋮	⋮
loke	3
antç	4
nzal	5
marh	6
matb	7
sasl	8
⋮	⋮
leme	1160
anma	1276
tmek	1511
lama	1599
etme	1645

31*31*31*31=923521

+ ≅ 350000
Toplam harf dörtlüsü sayısı

3.1.6.4. İstatistik tablolarının kullanımı

Görüldüğü gibi bu tabloların oluşturulmasının ardından bir sözcüğün Türkçe yazım kurallarına uyularak yazılıp yazılmadığı sınanabilir. Yapılması gereken ilk iş bu tabloların birer diziye aktarılmasından sonra bu dizilerin di-gram, tri-gram ve four-gramlara göre sıralanmasıdır. Daha sonra ise kontrol edilecek sözcüğe ait harf grupları (n-gramlar) bu tablolarda karmaşıklığı *O(LogN)* olan ikili arama algoritmasından yararlanılarak aranırlar. Bu tablolardan birincisinde 961 adet harf çiftinin yer aldığı düşünüldüğünde en fazla 10 karşılaştırma işlemi yapılacağı görülecektir. İkinci tabloda ise 29791 adet harf üçlüsü yer almaktadır. İkinci tablodaki aramada ise en fazla 15 karşılaştırma sonucunda aranan harf üçlüsü tabloda bulunacaktır. Eğer aranan n-grama ait frekans değeri belirli bir eşik değerinin altında ise bu durumda bu sözcük için hatalıdır denilebilir. Yalnız istatistik tabloları için kullanılacak eşik değerinin farklı olması gerektiği açıktır.

Örneğin di-gram tablosu için eşik değeri 45 olarak seçildiğinde "oç" harf çiftini içeren bir sözcük hatalı olarak kabul edilecektir. Ancak gerçekten de bu sözcük

doğru bir sözcük olma ihtimaline de sahiptir(örneğin "koç"). Bunun için eşik değerinin altında yer alan harf çiftlerine sahip olan sözcükler bir tabloda tutulabilirse ve hatalı olarak kabul edilmeden önce bu tabloda yer alıp almadıkları kontrol edilirse bu tür hataların önüne geçilebilir. Burada 45 eşik değerinin altında 444 adet harf çiftinin yer aldığı ve bunlardan 140 adedinin herhangi bir Türkçe sözcüklerde yer alma ihtimalinin olmadığı düşünüldüğünde bu yöntemin etkili bir yöntem olarak kullanılabilmesinin mümkün olabileceği görülebilecektir.

Ancak kullanılan imla klavuzunda doğal olarak sözcükler yalın hallerinde tutulduklarından bazı durumlar için n-gramlar gerçekte olması gerekenden daha az sayıda görülmektedirler. Örneğin imla klavuzunda "sözcük" kelimesi yer almaktadır. Fakat, "sözcüğün" kelimesi için bir analiz yapılması halinde "cüğ" harf üçlüsünün tabloda 5 değerine sahip olduğu, "zcüğ" harf dörlüsünün 4 değerine sahip olduğu yine "üğ" harf ikilisinin de ilgili tabloda 148 değerine sahip olduğu görülmüştür. Bu durumda bu kelimenin n-gram analizi sonucunda hatalı olduğu sonucuna ulaşılacaktır. (Eğer tri-gram için eşik değeri 4'ten büyükse) Buradan da anlaşılacağı gibi bu şekilde sözlüksüz olarak yapılacak bir analizde başarıya ulaşılabilmesi için n-gram tablolarının oluşturulması için kullanılan klavuzun sözcüklerin ek almış hallerini de içermesi ya da en azından daha sonradan bu tür belirli n-gramların değerlerine dışarıdan müdahale edilmesi gerekmektedir. Bu zaafın giderilmesi amacıyla çeşitli farklı kaynaklardan elde edilen sözcüklerin n-gram tablolarının oluşturulması aşamasında kullanılması gereklidir. Örnek sayısının artması ile elde edilecek frekanslar çok daha doğru şekilde ortaya çıkacaktır.

3.2. Eksik Değerler

Eksik değerler ilişkili diğer özelliklerden yararlanılarak bazı kurallar kullanılarak öngörülebilir veya kestirilebilirler.

1. Değer öngörümü veya kestirimi için kuralların kullanılması

   Örneğin

   (a) Eğer yönetici düzeyindeki bir kişinin maaşı mevcut değilse, aynı seviyede görev yapan diğer yöneticilerin veritabanında yer alan maaş bilgileri eksik veri hakkında ipucu verebilir

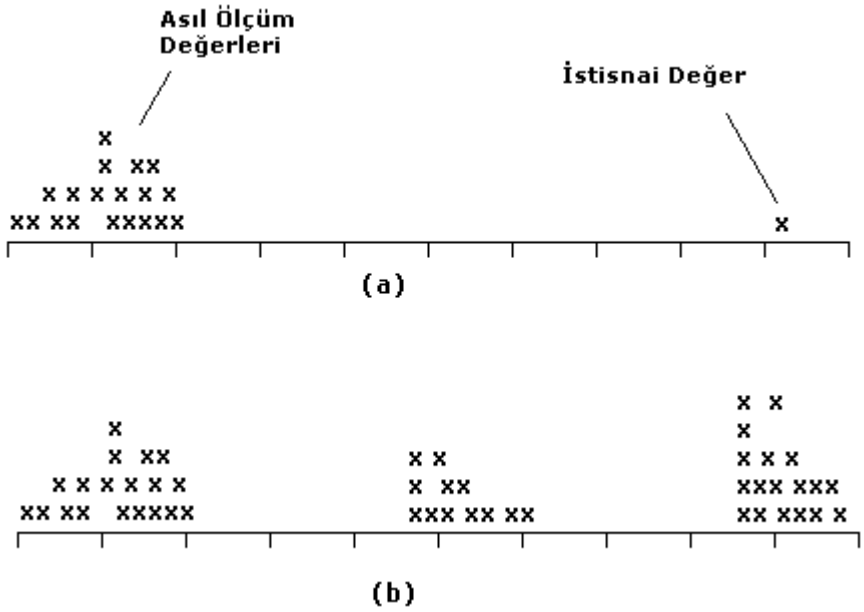
   (b) Geçmişte yapılan satışlara dayanarak satış miktarı hakkında öngörüde bulunabilmek de başka bir örnek olabilir.

2. Bir diğer yöntem ise üzerinde çalışılılan özellik üzerine bir polinom uydurulmasıdır. Değerlerin taranması ile polinom derecesi ve sabitleri elde edilebilir. Uydurulan bu polinomdan yararlanılarak eksik veriler çıkarılabilir

3.3. Aykırı Değerler (Outliers)

Canlı veritabanı sistemlerinin hemen hemen tamamına yakınında belirli miktarda aykırı verinin bulunması genelde sık karşılaşılan bir durumdur. Literatürde bu tür aykırı değerler "outlier" olarak adlandırılmaktadır. Bu tür verilerin ayıklanması hem veri kalitesinin artırılmasını hem de bu verilerin çeşitli analiz sonuçlarını etkilememesini sağlamaktadır.

Aykırı değerlerin ayıklanması için en sık kullanılan yöntem değerlerin bir grafik şeklinde gösteriminin ardından manuel olarak ayıklama işleminin yapılmasıdır. Sayısal büyüklükler için bu işlem verinin taranması ve belirli sınırların aşılıp aşılmadığının belirlenmesi ile yapılabilir. Sınırlar dışına taşmış olan değerler muhtemelen elenmesi gereken değerlerdir. Ayıklama işlemi ilişkili diğer özellikler üzerinde kurallar uygulaması ile onaylanabilir.



Şekil 3.3 – Örnek aykırı değerler

Özellikle deneysel ortamlarda elde edilen ölçümlerdeki istisnai değerlerin ayıklanması yapılacak olan hesaplamaların hata oranlarının azaltılmasını sağlayacaktır. Örneğin otomatik olarak bir gerilim ölçümünün bilgisayara doğrudan aktarıldığı bir durumda çeşitli dış etkilerden dolayı bazı değerlerin normal sınırların çok dışında olması bu değerler kullanılarak yapılacak olan hesaplamaların hatalı olmasına yol açacaktır. Bunun için veritabanlarında ya da veri ambarlarında hesaplama, analiz ya da öngörü yapılmadan önce istisnai büyüklüklerin ayıklanması mutlaka üzerinde durulması gerekli bir işlemdir.

3.4. Tutarsız Kodların Düzeltilmesi

Veri kümesinde bazı özellikler için kullanılan kodlar veri kümesinin bütünüyle karşılaştırıldığında sonlu küçük bir küme oluştururlar. Bunun için bu kodlar için bir tablo hazırlanabilir. Böylece bir kodun doğruluğunun sınanması bu tablo kullanılarak yapılabilir. Örneğin, semtlere ait posta kodlarının doğruluğunun sınanması için semtler ile posta kodlarının yer aldığı bir başvuru tablosu oluşturulabilir ve veri kümesi içerisinde posta kodu ve semt bilgisi hatalı olan kayıtlar belirlenebilir.

3.5. Şemaların Kaynaştırılması

Şema bütünlüğü için gerekli temel adımlar aşağıda listelenmiştir.

1. Şemalar arası uyumluluk
2. Uyuşmazlıkların aşılması için şema dönüşümlerinin yapılması
   Ör. Tip dönüşümleri, tekrar isimlendirmeler, gereksiz alanların elenmesi
3. Birleştirme ve yeniden sistemin yapılandırılması

Dönüşümler anlaşılırlık(understandability), minimalite(minimality) ve tamlık(completeness) kalitesinin artırılması amacıyla yapılmaktadır.

- Tamlık genelleştirme/özelleştirme, yeni özelliklerin eklenmesi gibi bazı yapısal adımları içerir.

- Minimalitenin temel amacı ise gereksiz fazlalıkların keşfedilip elimine edilmesidir.

- Anlaşılabilirlik doğrudan ölçülebilir bir büyüklük değildir. Fakat, grafiksel bir gösterim içerisinde diyagramın şekli,bağlantıların toplam uzunluğu gibi özellikler anlaşılabilirlik için birer parametre sayılabilirler.

Şema bütünleştirme hâlâ insan müdahalesi gerektirmektedir. Fakat aşağıda değinildiği gibi bir algoritma şeklinde de şema bütünleştirme üzerinde çalışılabilir.

1. İlk adım olarak özellik isimleri eşadlı/eşanlamlı olup olmadıklarının belirlenmesi için karşılaştırılır.

2. Daha sonra özelliklerin veri tipleri karşılaştırılır.

3. Son olarak özellik değerleri denetlenir. Eğer orjin ve sınırlar aynı/benzer ise muhtemelen uyumluluk yüksektir.

3.6. Tekrarlı Kayıtlar

Tekrarlı kayıtların sistemde bulunma sebeplerinin başında kayıtların farklı kaynakların birleştirilmesi sonucunda elde edilmesi gelmektedir. Diğer olası sebepler ise kullanılan veri giriş araçlarında yeterli kısıtlamaların bulunmaması ya da veri tabanında yeterli miktarda bütünlük kısıtlayıcısının yer almamasıdır. Hem veri giriş araçlarının hem de bütünlük kısıtlayıcılarının çok iyi tasarlanması da çoğu zaman tekrarlı kayıtların engellenmesini sağlayamamaktadır. Örneğin veri girişi esnasında aynı kişiye ait bir kayıt birkaç farklılıkla iki farklı kayıt olarak kaydedildiğinde bunun o anda otomatik olarak farkedilmesi pek mümkün değildir. Çünkü bu tip hataların veri girişi sırasında sezilebilmesi için o anda çeşitli analizlerin yapılması gerekmektedir. Bu analizlerin yapılması halinde ise veri giriş hızında düşüşler olacaktır. Veri giriş hızının düşmesi ise bir canlı sistemde en istenmeyecek durumlardan biridir.

Tekrarlı kayıtlar iki temel yöntem kullanılarak temizlenebilirler.

- Veritabanında yer alan kayıtlar öncelikle yine veritabanındaki kayıtlardan yaralanılarak sıralanırlar. Böylece birbirine benzeyen kayıtlar birbirine yakın olacak şekilde bu sıralamada yer alırlar.

- Daha sonra sıralanmış olan bu kayıtlar arasında bazı karmaşık karşılaştırma işlemleri uygulanarak tekralı kayıtlar belirlenir.

Tekrarlı kayıtların belirlenebilmesi için büyük veri kümelerinde kullanılabilecek iki uygun algoritma vardır. Bu algoritmalar ile ilgili ayrıntılı açıklamalar aşağıda verilmiştir.

3.6.1. Sıralı-komşu yöntemi

3.6.1.1. Temel sıralı-komşu metodu tanımı

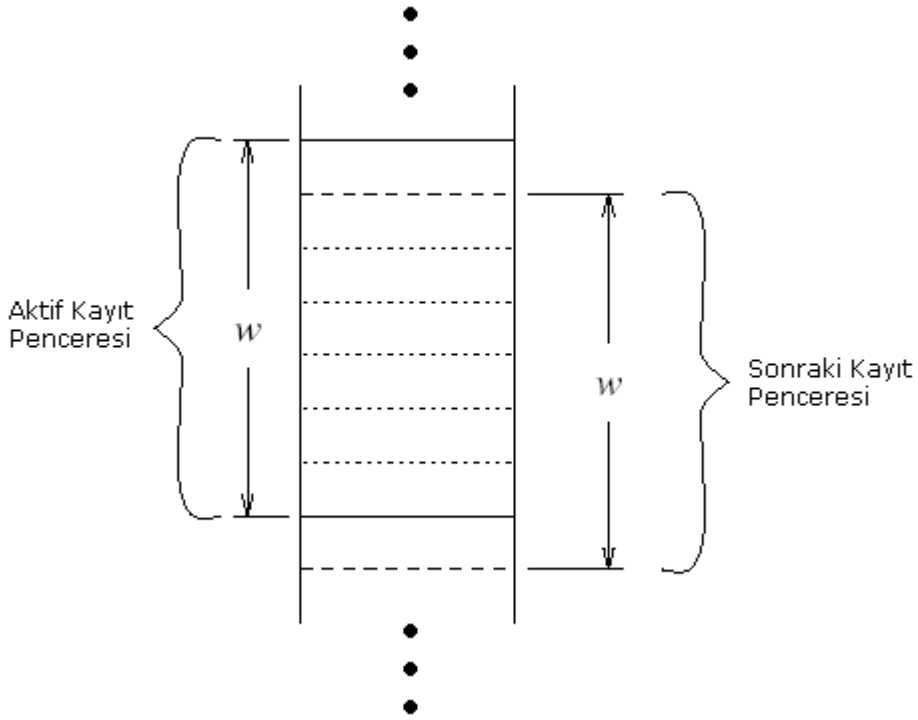
Sıralı-Komşu metodu veritabanlarında tekrarlı kayıtların ortaya çıkarılması amacıyla kullanılan bir yöntemdir. Literatürde "sorted-neighbourhood method" ya da kısaltılmış olarak "SNM" olarak anılmaktadır. İki ya da daha fazla sayıda veritabanı üzerinde çalışıldığında ilk olarak veriler N elemanlı bir ardışıl liste şeklinde

birleştirilir, daha sonra bu liste üzerinde sıralı-komşu metodu uygulanır. Sıralı-Komşu metodu üç adımda özetlenebilir:

1. **Anahtarların oluşturulması :** Listede bulunan her kayıt için kayıt alanlarının bir kısmı ya da tamamı kullanılarak bir anahtar oluşturulur. Anahtarın seçimi bilgi yoğunluğu ve konu ile ilgili alana özgü bir "hata yapısı" na bağlı olarak yapılır. Sıralı-komşu yönteminin verimli olması çok büyük oranda seçilen anahtarın uygun olmasına bağlıdır. Hatalı verilerin birbirlerine mümkün olduğunca yakın anahtarlara sahip olması bulunmalarını kolaylaştıracaktır.

2. **Verilerin Sıralanması :** Kayıtlar birinci adımda elde adilen anahtara göre sıralanır.

3. **Birleştirme :** Ardışıl kayıt listesindeki kayıtların karşılaştırılma sayısını sınırlayan sabit boyutlu bir pencere liste üzerinde hareket ettirilir. Eğer pencere boyutu *w* kayıt ise, bu durumda pencere içerisine giren her yeni kayıt "benzer" kaydın bulunabilmesi için *w-1* kayıtla karşılaştırılır. (Şekil 3.4)

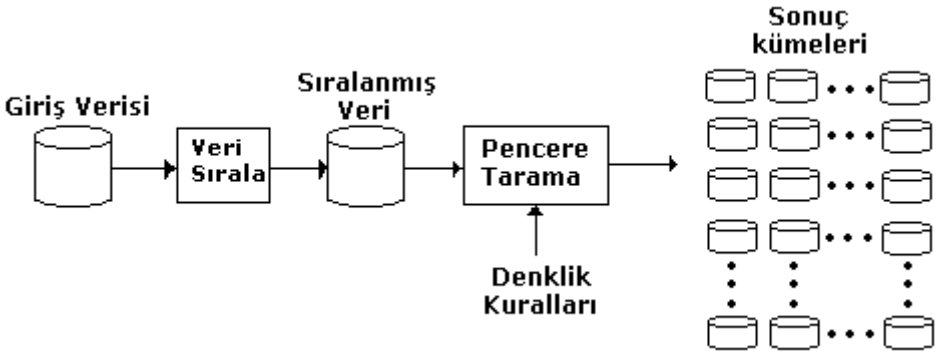


Şekil 3.4 – Veri temizleme sırasında kullanılan pencere tarama işlemi

Bu adımlar sırasıyla yürütüldüğünde, anahtarların oluşturulması aşaması *O(N)*, sıralama aşaması *O(N logN),* birleştirme aşaması ise *O(wN)* işlem gerektirir. Burada

N veritabanında yer alan kayıt sayısıdır. Sonuç olarak, bu metodun toplam zaman karmaşıklığı eğer $w < log\ N$ ise $O(N\ logN)$ aksi halde $O(wN)$ ' dir. Fakat ifadelerde yer alan sabit değerler çok fazla değişebilirler. Anahtarların oluşturulması aşamasında bir kayıttan ilgili anahtar değerlerin elde edilmesi nispeten daha masraflı olabilir. Sıralama anahtar değerler üzerinde birkaç karşılaştırma işlemi gerektirmektedir. Birleştirme aşaması ise potansiyel olarak çok sayıda kuralın iki kaydın karşılaştırılması sırasında uygulanmasını gerektirmektedir. Bundan dolayı en büyük sabit faktörü birleştirme aşamasında ortaya çıkmaktadır.

Burada dikkat edilirse *w* pencere tarama prosedürü için bir parametredir. *w* için geçerli değer aralığı 2 (sadece ardarda gelen iki kayıt karşılaştırılır) ile N (her kayıt diğer bütün kayıtlar ile karşılaştırılır). Son durumda işlem karmaşıklığı $O(N^2)$ olacaktır ve maksimum doğruluğa erişilebilecektir(Denklik kurallarına göre bütün tekrarlı kayıtlar doğru bir şekilde bulunabilecektir). İlk durumda ise (*w N* yanında oldukça küçük bir değere sahip) en uygun zaman performansı (sadece $O(N)$ zamanda) elde edilecek fakat doğruluk derecesi de minimum seviyede kalacaktır. Burada temel problem maksimum doğruluk ve minimum karmaşıklık için *w* sabitinin alması gereken değerin ne olacağıdır.



Şekil 3.5 – Temel sıralı komşu metodu

Bütün büyük veritabanları için G/Ç (I/O) baskın olan maliyettir, dolayısıyla veri seti üzerinden her geçiş bu maliyet yükünü artıracaktır. Bu durumda, en az üç geçiş gerekli olacaktır. İlk geçiş anahtarların hazırlanması, ikinci geçiş hızlı bir sıralama ile verilerin sıralanması ve son bir geçiş pencere sürecine geçilmesi ve pencereye giren her kayıt için "rule" yani belirlenmiş olan kuralların uygulanması olacaktır. Kural

programının karmaşıklığı ve *w* pencere boyutuna göre son geçiş büyük ihtimalle en büyük maliyete yol açacaktır.

Tablo 3.6 – Örnek kayıtlar ve anahtarlar

Ad	Soyad	Adres	ID	Anahtar
Rıza	Yeşilyurt	Beytepe Mah. 1456 Sok.	123456789	YELRIZBYTP123
Rıza	Yeşilyurt	Beytepe Mah. 1456 Sok.	123456780	YELRIZBYTP123
Rıza	Yeşilyutr	Beytepe Mah. 1456 Sok.	123456789	YELRIZBYTP123
Rıza	Yeşilvatan	Baytipi Mah. 1456 Sok.	123498745	YELRIZBYTP123

3.6.1.2. Anahtarların seçimi

Sıralı-komşu yönteminin verimli olabilmesi kayıtların sıralanması için seçilmiş olan anahtara doğrudan bağlıdır. Burada anahtar alan, kayıt içerisinde yer alan alt özelliklerin ya da bu özelliklere ait katarların bir kısmının birleştirilmesi ile elde edilmektedir. Örneğin Tablo 3.6 da yer alan dört kayıt düşünüldüğünde bu özel durum için anahtar, soyad alanının ilk iki ve beşinci karakteri, isim alanının ilk üç karakteri ardından adresin ilk bölümünün sessiz harfleri ve takip eden kimlik numarasının ilk üç karakterinin birleştirilmesi ile elde edilmiştir. Bu seçim de "anahtar tasarlayıcısı" nın soyad alanının genelde yanlış anlamalardan dolayı hatalı yazılmasından dolayı bu alanının ilk iki karakterini almaya karar verdiğini düşünebiliriz. Aynı şekilde isimler genelde daha fazla yaygın olduğundan ve hatalı yazılmaması için daha fazla özen gösterildiğinden isim alanı tamamıyla alınmıştır. Bunun sonrasında bu anahtarlar kayıtların sıralanması için kullanılacaktır, böylece anahtar alanları birbirine yakın olan kayıtlar aynı pencere içerisinde yer alabilecekler ve çift kayıtların bulunması kolaylaşacaktır. Dikkat edilirse dört kayıt içinde anahtar alanlar aynıdır. Burada açıkça görülüyor ki ilk iki kayıt gerçekten aynı kayıtlardır. Üçüncü kayıtta ise soyad alanında bir klavye hatası sonucu yazım hatası olduğu görülmektedir. Son kayıt ise diğerleri ile aynı anahtar alana sahip olmasına rağmen muhtemelen aynı kişi olmadığı görülmektedir. Pencere tarama işlemi ile kayıtlar arası benzerliklerin ortaya çıkarılması ile denklik olup olmadığı belirlenmektedir.

3.6.1.3. Denklik teorisi

Birleştirme aşamasında kayıtların denklik karşılaştırılması sıralama anahtarlarından ziyade kayıtlarda yer alan verilerin karmaşık bir süreçten geçirilmesi ile yapılır. Örneğin iki isim birbirine yakın fakat tamamen aynı olmayacak şekilde yazılmışsa ve

bu kayıtların adresleri aynı ise, buradan bu iki kişinin muhtemelen aynı kişiler olduğu bilgisi çıkarsanabilir. Diğer taraftan, iki kayıt aynı sosyal sigorta numarasına sahip, fakat isim ve adres bilgileri tamamen farklı olduğu durumda ise şu çıkarımlar yapılabilir; ya bu iki kayıt adres ve isim değiştirmiş aynı kişiye ait kayıttır ya da kayıtlardan birinde sosyal sigorta numarası alanı hatalı girilmiştir. Burada eğer ek bir bilgi yoksa ikinci seçeneğin doğru sayılması daha mantıklı olacaktır. Kayıtlarda ne kadar çok bilgi varsa o oranda çıkarımlar doğru sonuçlar verecektir. Örneğin **Suna Akın** ve **Sunay Akın** aynı adrese sahip iseler ve isimleri de makul derecede birbirine yakın olduğu için eğer yaş ve cinsiyet bilgileri mevcut ise **Suna** ve **Sunay**'ın evli ya da kardeş oldukları sonucuna varılabilir.

Bu çıkarımların basit bir şekilde değer ya da katarların karşılaştırılması ile değil açıkça belirlenmiş bir "denklik teorisinin" oluşturulması ile yapılması halinde sağlıklı sonuçlar elde edilmesi mümkün olabilecektir.

Örnek olarak aşağıda basit bir kural verilmiştir.

```
R1 ve R2 gibi iki kayıt için olsun
IF R1'in soyadı ile R2'nin soyadı aynı ise,
      AND isimlerinde az farklılık var ise,
      AND R1 ve R2 aynı adrese sahip ise,
THEN
      R1 ve R2 aynı (denk) kayıtlardır
```

Burada "**az farklılık**" kayıtlara ait isim alanlarına *uzaklık fonksiyonu*(*distance function*) uygulandıktan sonra fonksiyonun dönüş değerinin bir *eşik değerinin* üzerinde olmasını ifade etmektedir. Uzaklık fonksiyonun ve uygun eşik değerinin seçimi deneysel sonuçlarla belirlenmektedir. Uygun olmayan bir eşik değeri seçimi ya hatalı olarak denklik verilen kayıt sayısında artışa sebep olacaktır ya da denkliği belirlenmesi gereken kayıt sayısında düşüşe sebep olacaktır. "**Benzerlik**" ya da "**az farklılık**" üç farklı şekilde düşünülebilir.

1. *Değişiklik Uzaklığı(Edit-Distance):* İki sözcüğün birbiri ile eşitlenebilmesi için gerekli olan harf değiştirme/ekleme/silme sayısı

2. *Ses/Söyleniş Benzerliği(Phonetic Similarity):* Sözcüklerinin söylenişinin birbirine yakınlığının test edilmesi

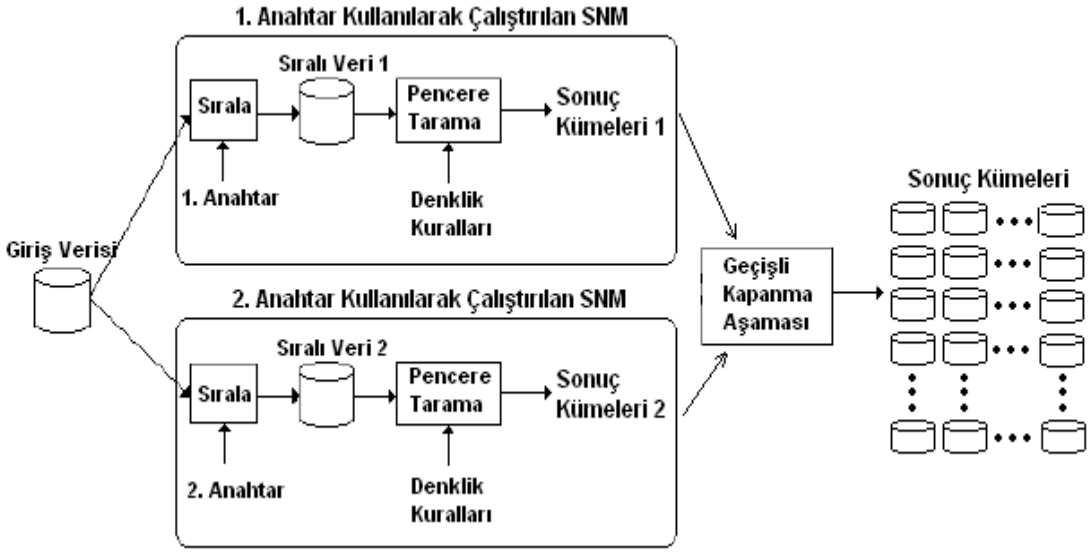
3. *Daktilo/Klavye Uzaklığı(Typewriter Distance):* Sözcüklerin içerdiği karakterlerin klavye üzerindeki yerlerine göre birbirine uzaklığı. Ör: Mehmet ile Nehmet birbirine yakın sözcüklerdir, çünkü Q klavye üzerinde '**N**' ile '**M**' yan yanadır.

Uygulanan kurallara kayıtların her zaman aynı özelliklerinde karşılaştırılma yapılması gibi bir zorunluluk yoktur. Örneğin, bir kişinin adı ve soyadının ters yazılıp yazılmadığının ortaya çıkarılabilmesi için şu şekilde bir kural yazılabilir: Birinci kayda ait isim ile ikinci kayda ait soyad ve birinci kayda ait soyad ile ikinci kayda ait isim alanları karşılaştırılabilir.

İyi bir denklik teorisinin oluşturulması süreci uzman sistemlerdeki gibi iyi bir bilgi-tabanı oluşturma sürecine benzer. Karmaşık problemlerde sistemde kullanılması gereken kuralların belirlenebilmesi için bir uzmanın yardımına ihtiyaç duyulabilir. Çıkarılan bu kurallar test edilip çeşitli veriler üzerinde denendikten sonra kural setinin tamamlanabilmesi için uzmana pek çok defa danışılması gerekliliği ortaya çıkabilir.

3.6.1.4. Bağımsız yürütmeler sonucunda geçişli kapanma durumlarının belirlenmesi

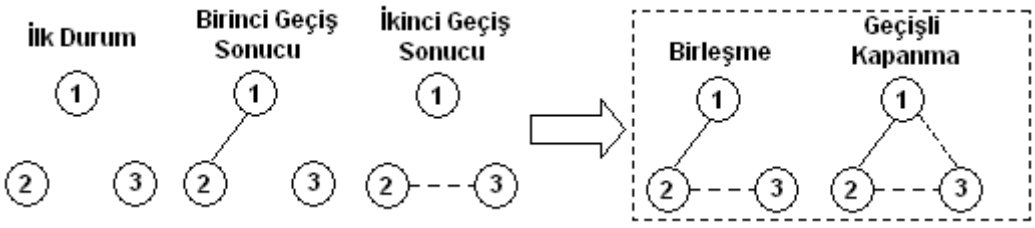
Genel olarak tek bir anahtarın kullanılması bütün eşleşen kayıtların bulunması için yeterli değildir. Anahtar da kullanılan ilk alan ya da bölüm kayıtların sıralanmasında etkili olduğu için yüksek bir ayrıştırma gücüne sahip olmaktadır. Bundan dolayı, eğer bir kayıttaki hata belli bir alanda ya da alanın bir bölümünde oluşuyorsa ve bu bölüm anahtarın en önemli parçasını oluşturuyorsa, anahtara göre sıralama yapıldığında bu türde hatalı kayıtların aynı pencere içerisine düşmesi mümkün olabilecektir. Örneğin iki işçiye ait kayıtta birinci işçinin sosyal sigorta numarası **193456782** ve diğer işçinın sosyal sigorta numarası da **913456782** ise (ilk iki rakam ters yazılmış) ve sosyal sigorta numarası alanı anahtarın ilk alanı olarak kullanılmışsa bu iki kaydın aynı pencere içerisine düşme olasığı çok zayıftır. Çünkü anahtara göre yapılan sıralamada iki kayıt birbirinden çok uzakta kalacaktır.



Şekil 3.6 – SNM'nin çok geçişli uygulaması

Denkliği bulunabilecek kayıt sayısının artırılabilmesi için iki seçenek bulunmaktadır. İlki *w* sabitini artırarak tarama pencere boyutunun büyütülmesidir. Şüphesiz bu seçenek hesaplama karmaşıklığını artıracaktır ve fakat doğru şekilde tespit edilen benzer kayıt sayısını istenen düzeyde artıramayacaktır.

Uygulanabilcek diğer bir seçenek ise sıralı-komşu metodunun birbirinden bağımsız olarak birkaç defa yürütülmesidir. Bu durumda her yürütmede farklı bir anahtar ve göreceli olarak daha küçük bir *pencere boyutu* kullanılması gereklidir. Bu yöntem *çok geçişli yaklaşım* olarak adlandırılır.(Şekil 3.6) Örneğin bir yürütmede adres alanı anahtarın ilk parçası olarak kullanılırken bir başka yürütmede ise soyad alanı anahtarın ilk parçası olarak kullanılabilir. Herbir bağımsız yürütme sonucunda birleştirilebilecek farklı kayıt çifti kümeleri elde edilecektir. Daha sonra, elde edilen bu kayıt çiftleri üzerinde geçişli kapanma uygulanabilir. Şekil 3.7 de geçişli kapanma durumu örneklenmiştir.



Şekil 3.7 – Geçişli kapanma aşaması

3.6.2. Alandan (uygulamadan) bağımsız öncelik-kuyruğu algoritması

Bu yöntemde de yine SNM'de olduğu gibi kayıtlar sıralanmakta ve daha sonra kayıtlar arası bazı karşılaştırmalar ile tekrarlanmış kayıtlar belirlenmektedir. Uygulamadan bağımsız denilmesinin sebebi ise anahtarların oluşturulması aşamasından kaynaklanmaktadır. Bu yöntemde anahtarlar bütün alanlar kullanılarak yaratılır. Veri kümesi üzerinden iki geçiş yapılmaktadır.İlk geçişte, anahtar bütün alanların normal sıra ile taranması ile elde edilir, ikinci geçişte ise anahtar bütün alanların ters sırada taranması ile elde edilir.

- Sıralama: Sıralama sıralı-komşu yönteminde olduğu gibi yapılır
- Birleştirme: Birleştirme aşağıdaki gibi yapılır:

Veri kümesi bir graf gibi düşünülür, ilk durumda veritabanında bulunan bütün kayıtları gösteren ve birbirleriyle bağlantısı bulunmayan N adet düğüm bulunur. Düğüm çiftlerine karşılık gelen kayıtların eşleşmesi şartıyla iki düğüm arasında bir yönsüz kenar bulunur.

Bağlantılı bir küme içerisinde yer alan düğümler denk kayıtlardır. Eğer iki ayrı bağlantılı küme içerisinde yer alan herhangi iki düğüm denk düğümler ise bu durumda bu kümeler birleştirilir. Bunun için bir birleştirme-bulma(union-find) veri yapısı kullanılır.

Birleştirme(union) ile iki küme biraraya getirilir.

1. *Union(x,y)* x ve y düğümlerini içeren kümeler Sx ve Sy ise bu kümeler yeni bir küme çatısı altında birleştirilir ve bir küme temsilcisi belirlenir.

2. Bulma işlemi ile kümeler içerisindeki bir düğüm aranarak, hangi küme içerisinde yer aldığı bilgisi döndürülür.

Find(x) x düğümünü içeren küme bilgisi döndürülür.

Algoritma kayıt kümelerinin yer aldığı bir öncelik kuyruğundan yararlanır. Algoritma ile veritabanı ardışıl olarak taranır ve her bir kaydın taranıp taranmadığı ya da öncelik kuyruğundaki bir kümeye ait olup olmadığı belirlenir.

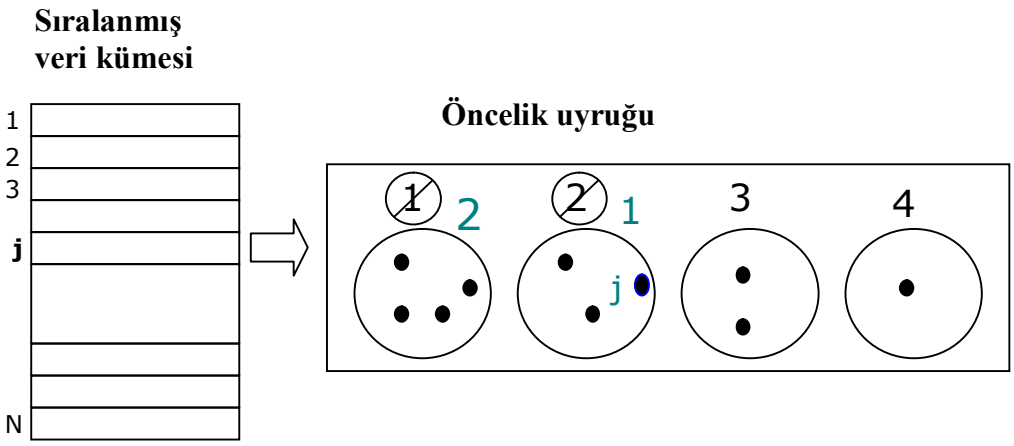
1. Eğer zaten öncelik kuyruğunda yer alan bir kümeye ait ise bu durumda kayıt atlanır.

2. Eğer öncelik kuyruğunda yer alan bir kümeye ait değilse, bu durumda kayıt Smith-Waterman algoritması kullanılarak öncelik kuyruğunda yer alan kümedeki temsilci kayıtlar ile karşılaştırılır.

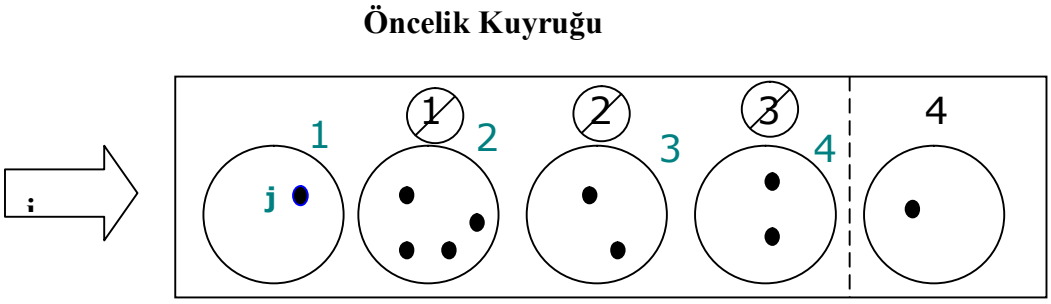
   (a) Karşılaştırma olumlu ise, küme için uygun bir kayıttır ve iki küme union işlemi kullanılarak birleştirilir. (Şekil 3.8)

   (b) Eğer hiçbir karşılaştırma olumlu sonuç vermezse kayıt öncelik kuyruğu içerisinde bağımsız bir küme olarak saklanır. (Şekil 3.9)

   (c) Yeni/birleştirilen kümeye en yüksek önceliğe sahip olacak şekilde öncelik değeri atanır.Eğer öncelik kuyruğunun boyutu belirli bir sınırı aşıyorsa en düşük önceliğe sahip küme silinir.



Şekil 3.8 – Karşılaştırmalar olumlu ise yürütülen işlemler



Şekil 3.9 – Karşılaştırmalar olumsuz ise yürütülen işlemler

Algoritmanın çalışma biçimi aşağıda daha ayrıntılı bir şekilde açıklanmıştır. Algoritma veritabanındaki kayıtları ardışıl olarak taramaktadır. R_j belirli bir anda veritabanından okunmuş bir kayıt olsun. Algoritma, ilk olarak R_j nin öncelik kuyruğunda yer alan bir kümeye ait olup olmadığını sınamaktadır.Bu işlem R_j ile küme temsilcilerinin karşılaştırılması ile gerçeklenir. Eğer bu karşılaştırmalardan biri olumlu sonuç verirse, R_j zaten öncelik kuyruğunda yer alan bir kümeye üyedir. Bu küme öncelik kuyruğunda en yüksek önceliğe sahip hale getirilir ve bir sonraki kayıt işlenmek için veritabanından okunur.

Sonuçları ne olursa olsun, bu karşılaştırma işlemleri sisteme maliyet açısından büyük bir yük teşkil etmemektedir. Çünkü bu karşılaştırma sürecinde sadece *Find* işlemi kullanılmaktadır.

İkinci durum ise R_j nin öncelik kuyruğu içerisindeki hiçbir kümenin üyesi olmamasıdır. Bu durumda *Smith-Waterman* algoritması kullanılarak R_j ile öncelik kuyruğunda yer alan kayıtlar arasında karşılaştırma yapılır. Kuyruktaki herbir küme için *Ri* gibi bir kayıt ile R_j karşılaştırılır. Eğer bu işlem sonucunda bir denklik ortaya çıkarsa R_j ye ait küme ile R_i ye ait küme, *Union*(R_j,R_j) kullanılarak R_i ye ait küme üzerinde birleştirilir.

Son olarak, eğer R_j için hiçbir karşılaştırma olumlu sonuç vermediyse R_j o anda öncelik kuyruğunda yer almayan bir kümeye aittir. Bu durumda R_j öncelik kuyruğunda tek başına bağımsız bir küme olarak saklanır ve en yüksek öncelik verilir. Eğer bu durumda kuyruk boyutu sınırı aşılırsa en düşük öncelikli olan küme kuyruktan silinir.

4. KAYIT YA DA KATAR BENZERLİKLERİ İÇİN KULLANILAN YÖNTEMLER

Verilerin temizlenmesi sürecinde en fazla üzerinde durulan konu tekrarlı kayıtların elimine edilmesidir. Genel olarak tekrarlı kayıtların ortaya çıkarılıp verilerin temizlenmesi üç adımda gerçekleştirilmektedir. Daha önce de değinildiği gibi öncelikle veritabanındaki kayıtlardan yararlanılarak anahtarlar oluşturulur. Daha sonra ise kayıtlar bu anahtarlar kullanılarak sıralanır. Son adımda ise birbirine yakın sıralanmış olan kayıtlar karşılaştırılarak tekrarlı kayıtlar ortaya çıkarılır. Kayıtların karşılaştırılması ve denkliklerinin belirlenebilmesi oldukça karmaşık işlemler gerektirmektedir. Bu denklik karşılaştırmaları uygulamaya özel olabileceği gibi uygulamadan bağımsız da yapılabilir.

- *Denklik Teorisi (Equational Theory)*

  Bu yaklaşım kural tabanlı bir karşılaştırma sistemine dayanmaktadır. İki kayıt arasında denklik olup olmadığı karakter katarları karşılaştırılmasının yanında bir dizi koşulun(kuralın) sağlanıp sağlanmadığının test edilmesiyle belirlenir. Bu yöntemle ilgili detaylı bilgiler 3.6.1.3 bölümünde verilmiştir.

- *Benzerlik Derecesi (Degree of Similarity)*

  Bir diğer yaklaşım ise kayıtlar arasındaki benzerlik derecesinin hesaplanmasıdır. Benzerlik derecesi 0,0 ile 1,0 arasında değişen bir ondalıklı sayı ile ifade edilir. Eğer iki kayıt arasındaki benzerlik derecesi 0.0 ise bu iki kayıt arasında hiçbir benzerlik olmadığını ifade eder, tam tersi olarak eğer bu değer 1,0 ise bu durumda iki kayıt tamamen aynıdır. Benzerlik derecesi asıl olarak karakter katarları için düşünülmüş olsa da sayısal değerler (SSK numarası gibi) içeren veritabanı kayıtları için de verimli bir şekilde uygulanabilmektedir.

  1. *Kayıt Benzerliği (Record Similarity)*

Bu yaklaşımda kayıtların denkliği üç aşamalı olarak ele alınmaktadır: jeton(token), alan(field) ve kayıt(record). Kayıtları oluşturan her bir alan içerisindeki jetonlar(parçalar), boşluklar ya da noktalama işaretlerinden yararlanılarak ayrıştırılır. Daha sonra bu ayrıştırılmış parçalar sıralanır. Böylece SNM 'deki anahtarların oluşturulması ve bu anahtarlara göre sıralamanın yapılması sonucunda daha fazla benzer kayıt birbirlerine yakın sıralanır. Kayıt içerisinde yer alan her bir alanın farklı bir önemi olduğundan, alanlar için ağırlık derecesi belirlenir. Bir kayıt içerisinde yer alan bütün alanların ağırlıklarının toplamı 1'e eşittir. Örneğin isim, soyad, cinsiyet, adres gibi dört alanı olan bir kayıt için alan ağırlıkları şu şekilde belirlenebilir; adres=0,6 soyad=0,3 isim=0,2 cinsiyet=0,1. İki kayıt arasındaki kayıt benzerliğinin belirli bir eşik değerini,örneğin 0.8, aşması durumunda bu kayıtlar denk olarak düşünülür.

2. *Katar Benzerliği (Similarity of Strings)*

Karakter katarları arasındaki benzerlik oranının belirlenebilmesi amacıyla pratikte pek çok farklı yöntem kullanılmaktadır. En fazla kullanılan yöntemlerin başında edit uzaklığı, klavye uzaklığı, karakter kullanım benzerliği gibi yöntemler gelmektedir.

Aşağıda genel olarak çok sık kullanılan katar ya da kayıtlar arası benzerlik yöntemleri üzerinde durulmuştur.

4.1. Kayıt Benzerliği

Veritabanında yer alan kayıtların sıralanmasının ardından, kayıtlar arası karşılaştırma sayısını sınırlandırmak için *w* boyutlu bir pencere kayıt kümesi üzerinde hareket ettirilir. Pencere içerisine giren her yeni kayıt *w*-1 adet kayıt ile karşılaştırılır ve pencerede ki ilk kayıt pencere dışına atılır.

İki kayıt arasındaki benzerlik derecesinin belirlenebilmesi için etkili bir karşılaştırma yapılması gereklidir. Bu yöntemde diğer yöntemlerden farklı olarak kayıtlara ait alanlara göreceli olarak "alan ağırlığı" verilir. Örneğin bir kayıtta yer alan isim alanı doğal olarak cinsiyet alanına göre daha yüksek bir ağırlığa sahip olmalıdır. Alan

ağırlıkları kullanıcı tarafından belirlenir ve bir kayda ait alan ağırlıklarının toplamı 1'e eşit olmalıdır. Örneğin isim ve adres alanlarının göz önüne alınarak bir tekrarlı kayıt analizi yapılmak isteniyorsa ve bu alanların eşit olarak ağırlıklandırılması isteniyorsa her iki alana 0,5 ağırlık değeri verilir, diğer alanlar ise 0 ağırlıklı olarak değerlendirilir. Böylece tekrarlı kayıt analizi işlemi sürecinde sadece isim ve adres alanları göz önüne alınmış olacaktır.

İki kayıt arasındaki benzerlik derecesinin hesaplanması süreci denk alanlara ait jetonların(alt katarların) karşılaştırılması ile başlar. Jetonlar tam katar karşılaştırılması, tek-hatalı katar eşleştirilmesi, kısaltma eşleştirmesi ve ön-ek eşleştirmesi kullanılarak karşılaştırılır. Alanlara ait jetonların(alt katarların) karşılaştırılma sonuçları temel alınarak bütün alan için benzerlik derecesi hesaplanır. Son olarak bütün alanların benzerlik dereceleri kullanılarak kayıtlar arası benzerlik derecesi belirlenir.

Alan Benzerliğinin Hesaplanması

X kaydı için bir alan t_{x1}, t_{x2} , ... ,t_{xn} jetonlara(alt katarlara) sahip olsun.
Y kaydı için aynı alan t_{y1}, t_{y2} , ... ,t_{ym} jetonlara(alt katarlara) sahip olsun.

$1 \le i \le n$ olmak üzere, her t_{xi} jetonu, $1 \le i \le m$ olmak üzere t_{yi} jetonu ile karşılaştırılır. t_{x1}, t_{x2} , ... ,t_{xn} ve t_{y1}, t_{y2} , ... ,t_{ym} için hesaplanan maksimum benzerlik dereceleri sırasıyla $DoS_{x1},...,DoS_{xn}$, $DoS_{y1},...,DoS_{ym}$ ise X ve Y kayıtları için alan benzerlikleri şu şekilde verilir.

$$(\sum_{i=1}^{n} DoSxi + \sum_{i=1}^{m} DoSyi)/(n+m) \qquad (4.1)$$

Kayıt Benzerliğinin Hesaplanması

$F_1, F_2, ..., F_n$ alanlarına sahip bir veritabanındaki alan ağırlıkları sırasıyla $W_1, W_2, ..., W_n$ olsun. X ve Y kayıtları için hesaplanmış olan alan benzerliklerinin Sim_{F1}(X,Y),..., Sim_{F2}(X,Y) olduğu göz önüne alınırsa X ve Y kayıtları için kayıt benzerliği şu şekilde hesaplanır.

(4.2)

$$\sum_{i=1}^{n} SimFi(X,Y) * Wi$$

Burada bir eşik değeri belirlenerek belirli bir benzerlik derecesinin aşılması halinde ilgili kayıtların denk kayıtlar olarak eşlenmesi sağlanabilir. Eğer iki jeton tam olarak denk ise bu durumda benzerlik derecesi 1 dir. Aksi halde her farklı karakter için eğer toplam jeton boyu x karakter ise 1/x değeri 1'den çıkarılır. Örnek olarak "kat" ve "dart" gibi iki jeton karşılaştırıldığında DoS_{kat} = 1- 1/3 = 0.67 olur çünkü "kat" içerisindeki k karakteri "dart" içinde yer almamaktadır. DoSdart = 1-2/3 = 0.33 olur çünkü d ve r karakterleri "kat" sözcüğü içinde yer almamaktadır.

Kayıt benzerliği yönteminde jetonlar arası karşılaştırma amacıyla aşağıdaki yöntemler kullanılabilir. Bu yöntemler çok daha değişik şekillerde uygulalanabilirler ve sayıları artırılabilir.

1. **Tam katar karşılaştırma**

   Standart *strcmp()* fonksiyonunu iki katar tam olarak denk ise 1 aksi halde 0 değeri döndürür.

2. **Tek-Karakter Hatası Benzerliği**

   Fazla karakter yazımı, eksik yazım, karakter yerlerinin hatalı olması gibi durumlar için kullanılır.

Tablo 4.1 – Tek hatalı sözcük karşılaştırma

Sözcük1	**Sözcük2**	**$DoS_{Sözcük1}$**	**$DoS_{Sözcük2}$**
BİLGİSAYAR	BİLGİBSAYAR	1,0	0,9
BİLGİSAYAR	BİLGSAYAR	0,9	1,0
BİLGİSAYAR	BİLGİSAYER	0,9	0,9
BİLGİSAYAR	BİLGİSAYRA	1,0	1,0

3. **Kısaltma Benzerliği**

   Dışarıdan bir dosya kullanılarak sözcük kısaltmaları belirlenebilir. Bir A sözcüğü eğer B sözcüğünün bütün karakterlerini B sözcüğünde yer alan sırasıyla içeriyorsa bu durumda A B'nin kısaltması olabilir. Bu tip bir durumda A ve B için benzerlik derecesi olarak 1 değeri verilir.

Tablo 4.2 – Örnek bir kısaltma eşleme tablosu

Kısaltma	Sözcük
LTD	Limited
ŞTİ	Şirket
TİC	Ticaret

4. Ön-Ek Karşılaştırması

Burada ise birinin diğerinin ön-ek hali olan benzer iki sözcük üzerinde durulmaktadır. Örnek olarak "Tekn" ve "Teknoloji" veya "Int" ve "International" sözcükleri verilebilir. Dikkat edilirse $DoS_{Tekn} = 1$'dir çünkü "Tekn" içerisindeki bütün karakterler "Teknoloji" içerisinde de yer almaktadır. Fakat tersten bakıldığında $DoS_{Teknoloji} = 0{,}44$ değerine sahiptir çünkü "Teknoloji" sözcüğünün 5 karakteri "Tekn" içinde yer almamaktadır. Yalnız, eğer bir sözcük diğerinin içerisinde yer alıyor fakat ön-ek durumunda değilse benzerlik dereceleri yüksek olmamalıdır. Örneğin "net" ve "internet" sözcükleri bu örnek için uygundur. Bu iki sözcük için benzerlik derecesi olarak 0,0 değeri verilmelidir.

4.2. Katarlar Arası Benzerlik Oranlarının Belirlenme Yöntemleri

4.2.1. Edit uzaklığı

İki karakter katarının benzerlik yönünden karşılaştırılmasında oldukça kullanışlı bir yöntemdir. Bir katarın bir başka katara dönüştürülebilmesi için gerekli olan minimum karakter ekleme, karakter çıkarma ya da karakter değiştirme sayısı "edit uzaklığı" olarak tanımlanmaktadır. Bir benzerlik değerinin elde edilebilmesi için "edit uzaklığı" normalize edilir. Genel olarak normalizasyonda payda da kullanılmak üzere uzun olan katarın karakter sayısı kullanılmaktadır.

$x = x_1 \ldots x_n$ ve $y = y_1 \ldots y_n$ gibi iki karakter katarı arasındaki edit uzaklığı x katarının y katarına dönüştürülebilebilmesi için gerekli olan minimum işlem sayısı olarak tanımlanabilir. Uygulanabilecek işlemler şunlardır:

- karakter ekleme

$ekle(x,i,a) = x_1 x_2 \ldots x_i a x_{i+1} \ldots x_n$

- karakter silme

$sil(x,i) = x_1 x_2 \ldots x_{i-1} x_{i+1} \ldots x_n$

- karakter değiştirme

değiştir(x,i,a) = $x_1 x_2 \ldots x_{i-1} a x_{i+1} \ldots x_n$

Örneğin, $x = aabab$ ve $y = babb$ ise x katarının y katarına dönüştürülebilmesi için 3 adım gereklidir.

a a b a b x
b a a b a b x' *ekle(x,0,b)*
b a b a b x'' *sil(x',2)*
b a b b x''' *sil(x'',4)*

yine 3 adımda şu şekilde de yapılabilir.
a a b a b x
a b a b x' *sil(x,1)*
b a b x'' *sil(x',1)*
b a b b x''' *ekle(x'',3,b)*

Edit uzaklığının belirlenebilmesi için n ve m karşılaştırılan katarların uzunluğu olmak üzere n*m adet işlem yapılması gerekmektedir. Bundan dolayı problem karmaşıklığı O(nm) olarak verilir. Yukarıdaki x=*aabab* ve y=*babb* katarları tekrar gözönüne alınırsa, aşağıdaki matris elde edilir.

	λ	*b*	*a*	*b*	*b*
λ	0	1	2	3	4
a	1	1	1	2	3
a	2	2	1	2	3
b	3	2	2	1	2
a	4	3	2	2	2
b	5	4	3	2	**2**

Bu durumda kullanılan matris M ile simgelenirse x ve y katarları arasındaki edit uzaklığı M[n,m] olarak verilir. Aşağıda matrisinin oluşturulmasında kullanılan algoritma verilmiştir.Görüldüğü üzere yukarıdaki denemelerde 3 olarak bulunan uzaklık optimum uzaklık değildir. Algoritmanın sonucu olarak verilen uzaklık optimum uzaklıktır.

Optimum çözüme aşağıdaki adımlarla ulaşılabilmektedir.

a a b a b x
b a b a b x' değiştir*(x,1,b)*
b a b b x'' *sil(x',4)*

Algoritmada *Op[.,.]* gibi bir yardımcı tablo kullanılmaktadır. Tablonun amacı $x_1 \ldots x_i \; y_1 \ldots y_i$ dönüşümü aşamasında optimal dönüşümlerin belirlenebilmesi için atılan adımların takip edilebilmesidir.

```
EdDist(x,y)
   n=length(x)
   m=length(y)
   for i=0 to n
      M[i,0] = i
      for j=0 to m
         M[0,j] = j
   for i=1 to n
      for j=1 to m
         if xi==yi then change =0 else change=1
         M[i,j]= M[i-1,j]+1; Op[i,j] = delete(x,i)
         if m[i,j-1]+1 < M[i,j] then
            M[i,j]=m[i,j-1]+1; Op[i,j]=insert(x,i,yj)
         if M[i-1,j-1]+change < M[i,j] then
            M[i,j]=M[i-1,j-1]+change
            if (change==0) then Op[i,j]=none
            else Op[i,j]=change(x,i,yj)
```

4.2.2. Klavye(daktilo) uzaklığı

Karakter katarları arasındaki karşılaştırmalarda kullanılabilecek bir diğer yaklaşım da yine bulanık(fuzzy) katar karşılaştırma yöntemlerinden biri olan "klavye uzaklığı" yöntemidir. Klavye uzaklığı, klavye üzerinde yer alan iki tuş arasındaki mesafenin göz önüne alınmasıyla belirlenen bir ölçüdür. Örneğin Q tipi klavye göz önüne alındığında 'g' tuşu klavye üzerinde 'r', 't', 'y', 'f', 'h', 'v', 'b' ve 'n' tuşlarına bir birim uzaklıkta yer almaktadır. Çaprazda yer alan 'r', 'y', 'v' ve 'n' tuşları 1.414 yerine kolaylık açısından 1 birim uzaklıkta yer alıyor diye düşünülmektedir.

İki katar arasındaki benzerlik oranı karakter uzaklıklarının toplamının, uzun olan katar boyunun en uzak iki karakter uzaklığına çarpılarak bölünmesi ile elde edilir.

$$1 - \frac{D}{LxM} \qquad (4.3)$$

D : katarlar arası uzaklık

L : uzun katarın boyu

M : maksimum karakter uzaklığı

Karakterler arası uzaklık aşağıdaki tuş matrisi kullanılarak belirlenmektedir.

Tablo 4.3 – Q Klavye üzerindeki karakterler arası uzaklık matrisi

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	‘	1	2	3	4	5	6	7	8	9	0	*	-	
1		q	w	e	r	t	y	u	ı	o	p	ğ	ü	
2		a	s	d	f	g	h	J	k	l	ş	i		
3		z	X	c	v	b	n	m	ö	ç	.			

Örnek olarak A = ahmet ile B = samet katarları arasındaki klavye uzaklığı için

A	h	m	e	t		
S	a	M	e	t		
1 +	5 +	0 +	0 +	0 +	=	1-(6 / (5*5))= 0,76

Şekil 4.1 – İki karakter katarı arasındaki klavye uzaklığı hesabı

4.2.3. Karakter kullanım benzerliği

Bu yöntemde ise karşılaştırılacak olan iki katarın ortak kullandığı karakter sayılarından yararlanılarak katarlar arasındaki benzerlik oranı tespit edilir. Aşağıda karakter kullanım benzerliğinin elde edilebileceği bir algoritma verilmiştir.

```
float get_Similarity(char* s1, char* s2)
{
        int number[256] = {0};
        int i = 0, total = 0;
        int len1=strlen(s1), len2 = strlen(s2);

        for (i=0;i<len1;i++)
          number[s1[i]]++;
        for (i=0;i<len2;i++)
          if (number[s2[i]]>0)
                {
                  number[s2[i]] --;
                  total++;
                }
        return (float) total / max(len1,len2)
}
```

Aşağıda iki katarın "*karakter kullanım benzerliği*" kullanılarak benzerliklerinin belirlenmesi örneklenmiştir.

A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
0	0	0	1	0	2	0	0	0	0	0	3	0	2	1	0	0	0	1	0	1	0	0	1	0	0	0	0	1

ELEKTRİKÇİ

A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
0	0	0	1	0	2	0	1	0	0	0	3	0	3	2	0	1	0	1	0	1	0	0	1	0	0	0	0	1

ELEKTRİKÇİLİK

İlk aşamada, her iki dizide de aynı indise sahip elemanların arasındaki farklar bulunur. Karakter katarlarını karşılaştırmak için kullanılan alfabeye Ω denilirse, n de bu alfabedeki harf sayısını temsil etmek üzere her karakter katarı için n boyutlu ve indisleri i ∈ Ω olmak üzere S dizisi oluşturulur. Bu S dizisinin her elemanı Si, o karakter katarı içerisinde i. indis olan harften kaç adet bulunduğunu içermektedir. Bu koşullar altında iki karakter katarına ait yukarıda anlatılan şekilde oluşturulmuş S ve P dizileri kullanılarak şu şekilde hesap yapılır:

$$d_{SP} = \sum_{i=1}^{n} |S_i - P_i|, \quad i \in \Omega \tag{4.4}$$

Örneğin yukarıda yer alan elektrikçi ve elektrikçilik sözcüklerinin birbirlerine olan uzaklıkları 3'dür. Bu uzaklık değerinin uzun olan katar boyuna bölünmesi ile katarlar arasındaki benzerlik derecesi elde edilebilmektedir.

Ancak dikkat edilmesi gereken bir nokta vardır. Bu yöntemin tek başına kullanılması halinde doğru benzerlik derecelerinin elde edilmesi pek mümkün değildir. Eğer yine ELEKTRİKÇİ ve ELEKTRİKÇİLİK sözcükleri ele alınırsa bu sözcüklerde yer alan harfler aynı kaldığı sürece ne kadar farklı yazım olursa olsun elde edilecek olan uzaklık her zaman 3 değerini alacaktır. Çünkü bu benzerlik yönteminde sadece kullanılan harflerin sayıları önemlidir.

4.2.4. Fonetik(sesçil) benzerlik

Yazımlarından bağımsız olarak katarlar arasındaki söyleniş yönünden olan benzerlik oranlarının göz önüne alındığı bir karşılaştırma yöntemidir. Temel olarak bu yöntemde esas alınan özellik, sözcüklerin yapıtaşlarının sesbirimlerinden oluşmasıdır. Bu sesbirimleri uluslararası standartlarca belirlendiklerinden üzerinde

çalışılan dilden bağımsızdır. Fonetik benzerlik için basit teknikler kullanılması halinde elde edilecek sonuç diğer bahsedilen teknikler yanında tatmin edici olamayacaktır.

4.2.5. N-gram uzaklığı

Türkçe yazım denetimi için kullanılan n-gram yöntemi katarlar arası benzerlik oranlarının belirlenmesinde de kullanılabilir. Bir *s* katarı ve *n* tamsayısı için, *s*'ye ait *n-gramlar s* katarı üzerinde *n* uzunluklu bir pencerenin kaydırılması ile elde edilirler. Örneğin :

G("Harrison Ford") = {'Ha', 'ar', 'rr', 'ri', 'is', 'so', 'on', 'n ', ' F', 'Fo', 'or', 'rd' }

G("Harison Fort") = {'Ha', 'ar', 'ri', 'is', 'so', 'on', 'n ', ' F', 'Fo', 'or', 'rt' }

s_1 ve s_2 arasındaki bağıl *n-gram* uzaklığı aşağıdaki gibi verilir.

$$\Delta_q(s_1, s_2) = 1 - \frac{|G(s_1) \cap G(s_2)|}{|G(s_1) \cup G(s_2)|} \qquad (4.5)$$

Örneğin Δ_n("Harrison Ford", "Harison Fort") = 10/13 ≅ 0,77 Benzerlik oranının 1'e yaklaşması iki katar arasındaki benzerliğin fazla oduğunu göstermektedir.

5. UYGULAMA

5.1. Amaç

Bu çalışmada veri tabanları veya veri ambarlarındaki hatalı ve kalitesiz verilerin tespit edilip hataların giderilmesi amacıyla izlenmesi gereken adımlar ve yöntemler hakkında detaylı bilgiler verilmiş ve uygulaması yapılmıştır.

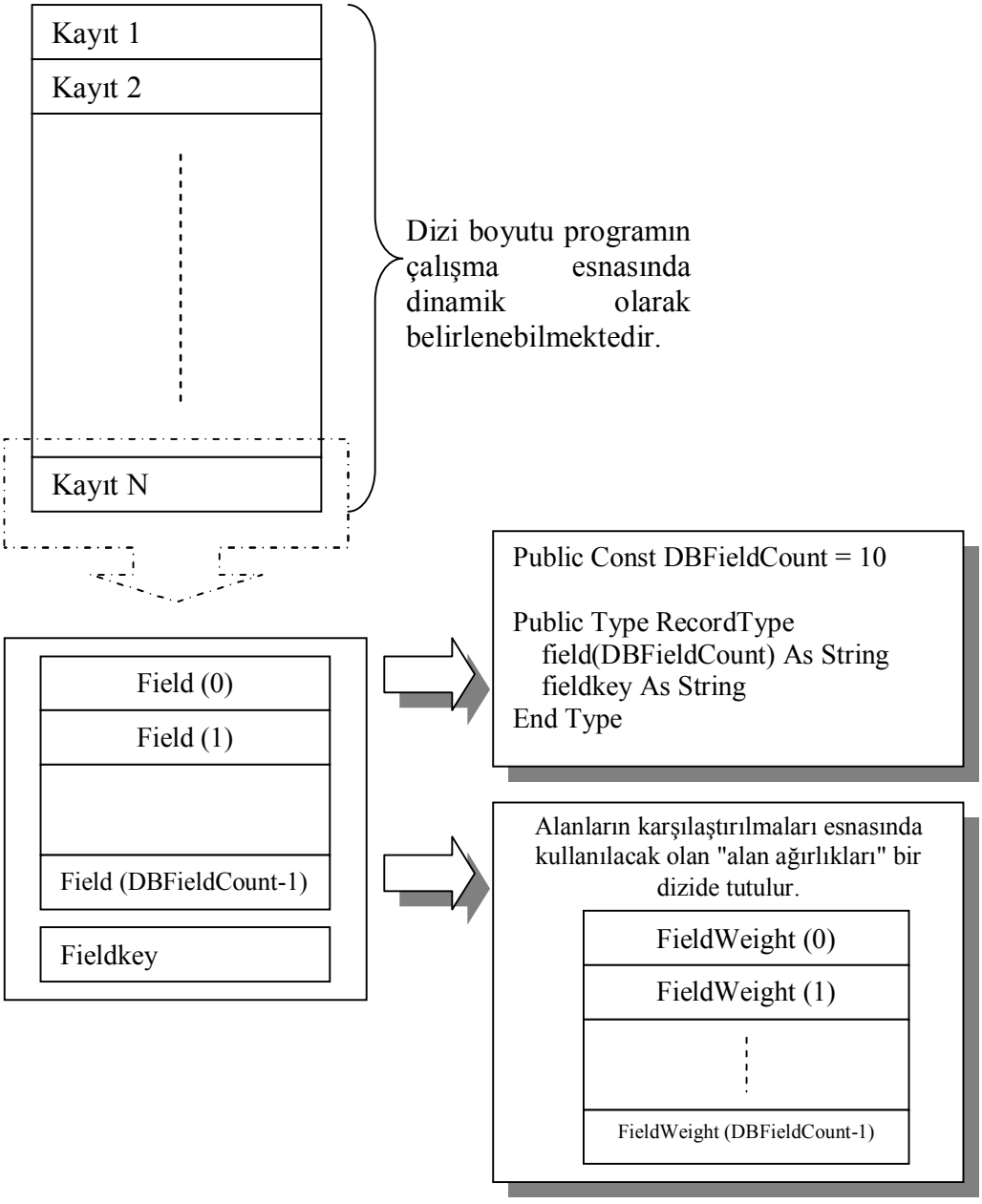
Veri temizleme uygulamaları genel olarak çok esnek yapıya sahip değildir. Çünkü her veritabanı kendine has bir şema yapısına sahiptir. Yani birbirinden bağımsız olarak farklı görevlerde kullanılmak için tasarlanmış olan veritabanlarının sadece tek bir veri temizleme aracı ile hatalarından arındırılması kolay bir iş değildir. Eğer çok genel amaçlı ve bütün veritabanları için kullanışlı bir uygulama geliştirilmek istenirse bu uygulamanın özel olarak her veri kümesi üzerinde etkili olabilmesi pek mümkün değildir. Bu çalışmada mümkün olduğunca esnek bir veri temizleme uygulamasının geliştirilmesi hedeflenmiştir.

Genel olarak veri temizleme, özel olarak tekrarlı kayıtların elimine edilmesi problemi hem ölçeklenebilirlik açısından hem de doğruluk açısından üstesinden gelinmesi zor bir problemdir. Bunun yanında tekrarlı kayıtların ortaya çıkmasına sebep olan yazım hatalarının da öncelikli olarak belirlenip sistemde gerekli düzeltmelerin yapılması ile tekrarlı kayıtlar için yapılacak analizden daha doğru sonuçların alınması hedeflenmiştir. Sözlüksüz olarak Türkçe'nin kurallı yapısından yararlanılarak ve n-gram tabloları ile istatistiksel olarak yanyana gelmesi ihtimali düşük olan harf kombinasyonlarından yazım denetimi yapılması amacıyla yararlanılması da daha önce çok fazla üzerinde çalışılmamış bir konu olması bakımından önemlidir. Bundan sonraki çalışmalarda yazım denetiminin daha etkili olabilmesi amacıyla bazı geliştirmeler yapılabilir.

5.2. Tekrarlı Kayıtların Tespiti

5.2.1. Kullanılan veri yapısı

Tekrarlı kayıtların belirlenmesi amacıyla Şekil 5.1 de verilmiş olan blok veri yapısı kullanılmıştır. Herbir kayıt veri tabanındaki alanlarına ek olarak bir adet anahtar alan ile temsil edilmektedir. Bu anahtar alanın değeri kayıt içerisinde yer alan alanlardan yararlanılarak her kayıt için bağımsız olarak oluşturulmaktadır. Daha sonra bu anahtar alan kayıtların sıralanması amacıyla kullanılacaktır.

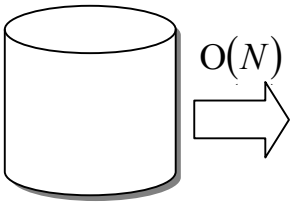


Şekil 5.1 – Kayıtlar için tutulan veri yapısı

Ayrıca kayıtların alanlarına ağırlık derecesi verilebilmesi amacıyla bir ağırlık dizisi kullanılmaktadır. Bu dizi aracılığı ile herbir alanının sonuçlara aynı oranda etki yapmasının önüne geçilebilir. Örneğin adres alanı için %50 ağırlık verilirken soyad alanı için %30, isim alanı için %25 ve cinsiyet alanı için %5 ağırlık verilebilir. Bu şekilde her veritabanı için farklı ağırlık tanımlanması yapılması sağlanarak daha genel bir temizleme uygulaması amaçlanmıştır.

5.2.2. Kayıtların okunması ve anahtarların oluşturulması

Yukarıda sözü edilen veri yapısına sahip olan dizi, programın çalışması esnasında dinamik olarak boyutlandırılabilmektedir. Böylece üzerinde çalışılan bilgisayarın bellek sınırlamaları çerçevesinde veritabanındaki kayıtlar doğrudan bu diziye aktarılabilmektedir. Bundan sonraki aşamalarda bütün işlemler için bu dizi kullanılmakta, disk işlemlerinin en aza indirilmesiyle işlemler çok daha hızlı yürütülebilmektedir. Diziye aktarım esnasında anahtar alanlar da yaratılmaktadır. (Şekil 5.2) Bu işlem $O(N)$ karmaşıklığı ile gerçeklenmektedir. Ancak anahtar alanların yaratılması katar işlemleri ile yapıldığından ve diskten okuma yapıldığından bu işlem süre yönünden sıralamaya göre daha uzun sürmektedir.

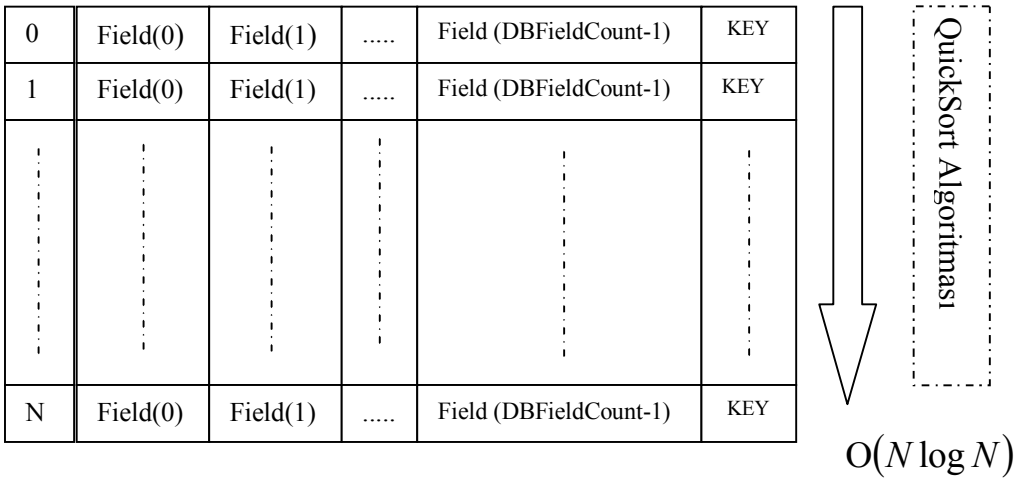


0	Field(0)	Field(1)	.....	Field (DBFieldCount-1)	KEY
1	Field(0)	Field(1)	.....	Field (DBFieldCount-1)	KEY
⋮	⋮	⋮	⋮	⋮	⋮
N	Field(0)	Field(1)	.....	Field (DBFieldCount-1)	KEY

Şekil 5.2 – Veritabanından kayıtların okunması ve anahtarların oluşturulması

5.2.3. Anahtarlara göre sıralama yapılması

Diziye aktarılmış ve anahtar alanları oluşturulmuş olan kayıtlar bu anahtarlardan yararlanılarak "Quicksort" algoritması ile sıralanırlar. (Şekil 5.3) Bu sıralama işlemi için karmaşıklık $O(N \log N)$ olarak verilebilir. Görüldüğü gibi sıralama işlemi için gerekli karmaşıklık anahtarların oluşturulması aşamasındaki karmaşıklıkdan daha büyüktür, fakat, sıralama işlemi yaklaşık olarak 3 kat daha hızlı olarak yürütülebilmektedir. Bu sıralama işlemininin yapılmasındaki amaç daha önce de bahsedildiği gibi benzerlikleri yüksek olan kayıtların birbirlerine yakın olarak sıralanmalarıdır.

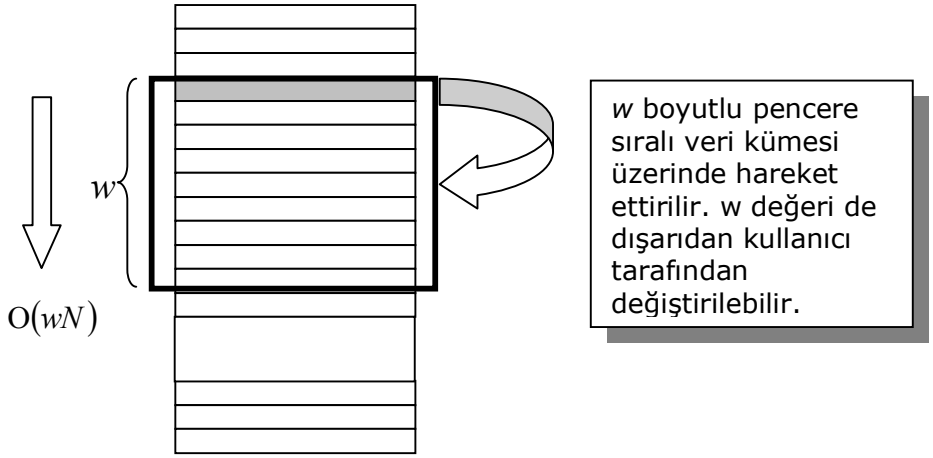


Şekil 5.3 – Anahtar alan üzerinde "quicksort algoritması" ile kayıtlar sıralanır.

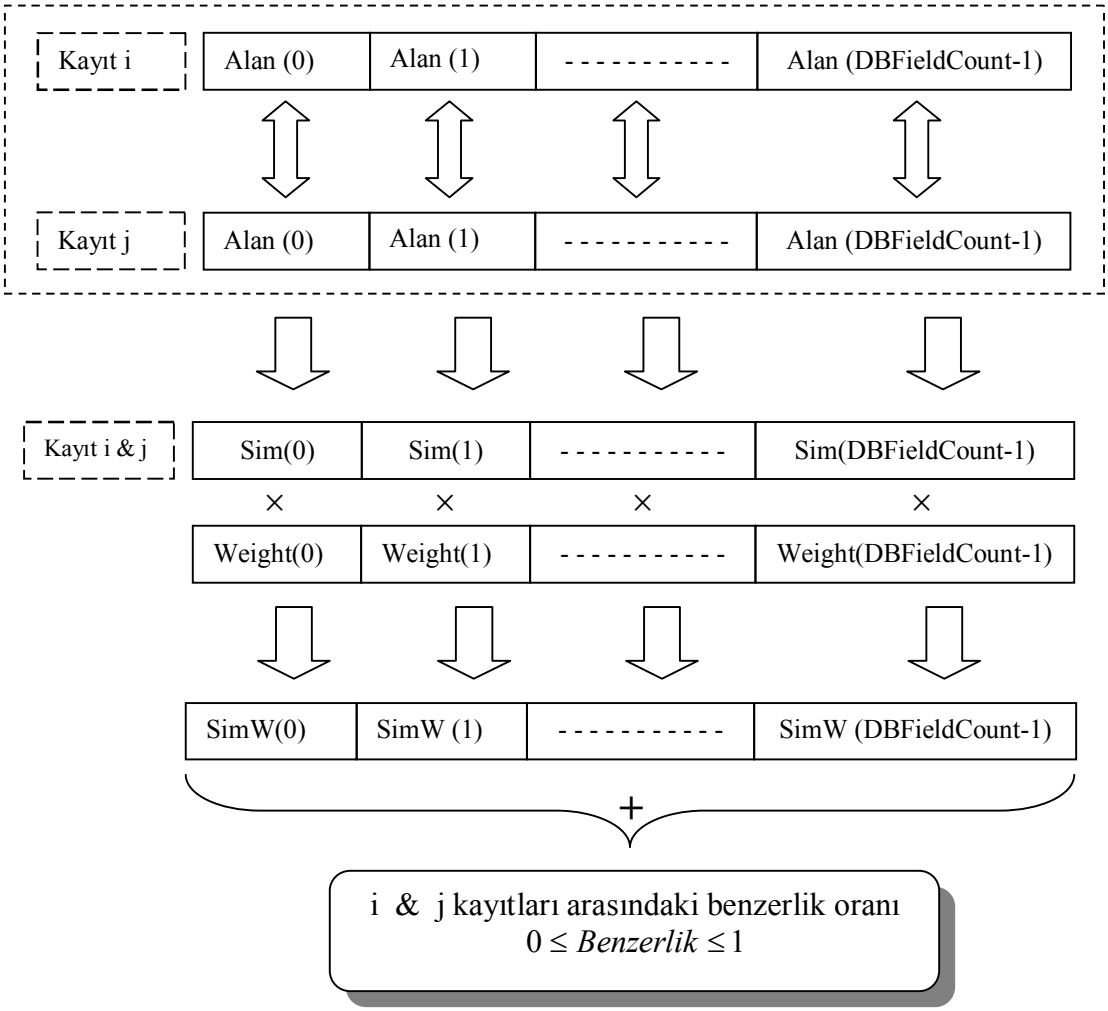
5.2.4. Sıralanmış olan kayıtların alan ağırlıkları ile karşılaştırılmaları

Bu aşamada sıralı kayıtlar üzerinde sabit *w* boyutlu bir pencerenin hareket ettirilmesi ile her kayıt kendisinden sonraki w-1 kayıt ile karşılaştırılır. Bu karşılaştırma sırasında kayıtlar arası benzerlik oranının belirlenmesi amacıyla edit uzaklığı yöntemi kullanılmaktadır. (Şekil 5.4) Herbir alan için edit uzaklığı yöntemi ile elde edilmiş olan benzerlik oranı daha sonra ilgili alana ait ağırlık derecesi ile çarpılarak herbir alanın ağırlıklandırılmış benzerlik katsayısı elde edilmektedir. Son olarak ise bütün bu benzerlik oranları toplanarak karşılaştırılan kayıtlar arası benzerlik oranı elde edilmektedir. Eğer bu oran dışarıdan kullanıcı tarafından belirlenebilen eşik değerini aşıyorsa bu durumda bu kayıtlar tekrarlı kayıt olarak kabul edilirler. Şekil

5.5'te iki kayıt arasındaki benzerliğin belirlenmesi için yürütülen aşamalar blok şema olarak verilmiştir.



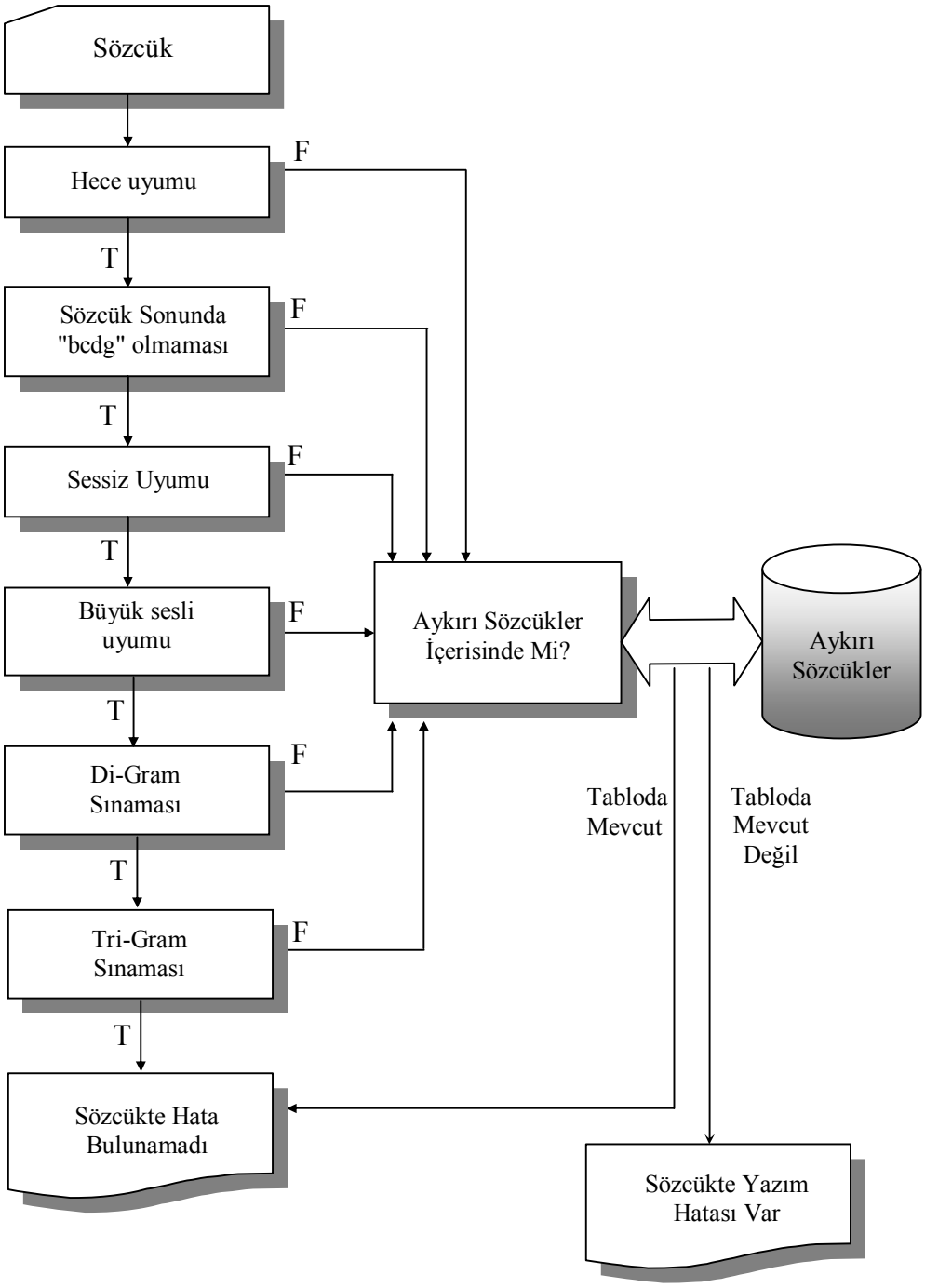
Şekil 5.4 – Pencere tarama işlemi



Şekil 5.5 – Kayıt karşılaştırma sürecindeki adımlar.

5.3. Türkçe Yazım Denetimi

Bölüm 3.1'de verilmiş olan Türkçe kurallarının ve n-gram tablolarının kullanılmasıyla bir sözlüksüz yazım denetimi uygulamasının gerçeklenmesi hedeflenmiştir.



Şekil 5.6 – Yazım hatalarının denetlenmesine ilişkin blok şema

Uygulamada sesli uyumu için sadece büyük sesli uyumunun varlığı aranmıştır. Ayrıca aykırı durumların aşılabilmesi için bir adet aykırı sözcükler tablosuna da yer verilmiştir. Kontrol edilen sözcük hatalı olarak değerlendirilmeden önce bu tabloda yer alıp almadığı kontrol edilmektedir (Şekil 5.6). Eğer bu tabloda yer alıyorsa hatalı olarak değerlendirilmemektedir. Kullanıcı programın yürütülmesi esnasında isterse doğru olarak kabul edilen sözcüğü bu tabloya ekleyebilmektedir, böylece program bir yandan da öğrenme yeteneğine sahip olmaktadır.

Tablo 5.1 – İki karakterin ters yazılması durumunda hatanın sezilmesi

	Sözcük	DiGram (Min)	Trigram (Min)	FourGram (Min)	Sessiz Uyumu	Sonda "bcdg"	Sesli Uyumu	Hece Uyumu
	Adımda	143	20	4	T	T	T	T
*	Adıdma	13	0	0	T	T	T	T
	Çıkarılır	1230	69	59	T	T	T	T
*	Çıkaırlır	0	0	0	T	T	T	F
	Oldukça	601	12	1	T	T	T	T
*	Odlukça	77	1	0	T	T	T	T
*	Olduçka	99	2	0	T	T	T	T
	Sağlanıp	234	5	4	T	T	T	T
*	Sağlaınp	0	0	0	T	T	T	T
	Hesaplanmasıdır	267	35	30	T	T	F	T
*	Hesaplamnasıdır	56	7	0	T	T	F	T
	Yaklaşımda	143	20	1	T	T	T	T
*	Yaklaşıdma	13	0	0	T	T	T	T
	Olduğundan	335	28	16	T	T	T	T
*	Olduğudnan	3	0	0	T	T	T	T
	Yapmazsanız	54	4	0	F	T	T	T
*	Yapmaszanız	0	0	0	F	T	T	T
	Değiştirildi	768	117	27	T	T	T	T
*	Değitşirildi	1	0	0	T	T	T	T
*	Değiştirilid	661	74	8	T	F	T	T
*	Değiştriildi	85	0	0	T	T	T	F
	Kahraman	169	33	10	T	T	T	T
*	Karhaman	133	27	2	T	T	T	T
	Denetim	2103	218	71	T	T	T	T
*	Deneitm	60	1	0	T	T	T	F
	Danışmanlarından	1591	193	18	T	T	T	T
*	Danışmalnarından	15	4	0	T	T	T	T
	Hükümetindeki	179	30	0	T	T	T	T
*	Hükümetindeik	60	4	0	T	T	T	F

Tablo 5.1'de doğru yazılmış ve hatalı olarak iki karakteri yer değiştirilmiş sözcükler için yazım denetiminde kullanılan testlerin sonuçları verilmiştir. Tablodaki en çarpıcı nokta di-gram ve tri-gram sonuçlarıdır. Görüldüğü gibi hiçbir durumda di-gram tek başına hatayı sezememiştir. Buradan di-gram'ın Türkçe yazım denetimi için çok yüksek bir ayırıcılığa sahip olmadığı sonucuna varılabilir. Tabii bazı durumlarda hatalı sözcüğün yakalanamadığı ya da hatalı olmadığı halde hatalı olarak tespit

edilen sözcüklerin de bulunabileceği göz ardı edilmemelidir. (Örneğin "sağlanıp" sözcüğü sınamalar sonucunda hatalı olarak değerlendirilmiştir.)

Tablo 5.2 – Bir karakterin eksik yazılması durumunda hatanın sezilmesi

	Sözcük	DiGram (Min)	Trigram (Min)	Four Gram (Min)	Sessiz Uyumu	Sonda "bcdg"	Sesli Uyumu	Hece Uyumu
	Saldırganlık	707	53	8	T	T	T	T
*	Saldıranlık	768	137	8	T	T	T	T
*	Saldıganlık	49	0	0	T	T	T	T
	Bırakmayacaklardır	287	41	1	T	T	T	T
*	Bırakmaycaklardır	45	11	0	T	T	T	T
	Gidemeyeceksiniz	661	66	4	T	T	T	T
*	Gidemeyceksiniz	45	3	0	T	T	T	T
	Ayrıcalıklı	426	75	6	T	T	T	T
*	Ayıcalıklı	496	75	2	T	T	T	T
*	Ayrıclıklı	16	0	0	T	T	T	T
	Verimsizlik	514	114	13	T	T	T	T
*	Verimizlik	1082	132	34	T	T	T	T
*	Vermsizlik	514	0	0	T	T	T	F
	Karmakarışık	1194	192	45	T	T	T	T
*	Karmkarışık	68	0	0	T	T	T	F
	Dilbilimcilik	112	18	5	T	T	T	T
*	Dibilimcilik	353	15	3	T	T	T	T
*	Dilbilmcilik	112	2	2	T	T	T	F
	Uygulamada	228	112	21	T	T	T	T
*	Uygulamda	143	28	5	T	T	T	T
*	Ugulamada	70	7	0	T	T	T	T
	Ağırlıklarından	1862	157	32	T	T	T	T
*	Ağrlıklarından	267	0	0	T	T	T	T
	Sözlerinin	231	118	37	T	T	T	T
*	Sözlernin	231	29	0	T	T	T	T
*	Sölerinin	231	5	0	T	T	T	T
	Çalışmalarıyla	345	16	4	T	T	T	T
*	Çaışmalarıyla	0	0	0	T	T	T	F

Bir karakterin eksik yazıldığı sözcükler için de yine di-gram'ın bu örnekler için hatayı tek başına sezemediği görülmektedir (Tablo 5.2). Tek karakterin eksik yazılması durumunda hece yapısındaki bozulmalar da hatanın sezilmesini sağlayabilmektedir.

Tablo 5.3 – Tek harfin hatalı yazıldığı durumlar

	Sözcük	DiGram (Min)	Trigram (Min)	FourGram Min	Sessiz Uyumu	Sonda "bcdg"	Sesli Uyumu	Hece Uyumu
	Sormayacaklarmış	516	60	1	T	T	T	T
*	Sormıyacaklarmış	345	11	0	T	T	T	T
	Olduğundan	335	28	16	T	T	T	T
*	Oldugundan	70	3	0	T	T	T	T
	Kararsızlık	628	98	8	T	T	T	T
*	Kararsıslık	615	7	0	T	T	T	T
	Geleceklermiş	995	68	5	T	T	T	T
*	Geliceklermiş	539	26	0	T	T	T	T
*	Geleçeklermiş	529	22	1	T	T	T	T
	Yükselmektedir	576	89	5	T	T	T	T
*	Yükselmekdetir	64	8	0	F	T	T	T

Tablo 5.3'de sözcük içerisindeki tek harfin hatalı olduğu durumlardaki hata saptanması durumları verilmiştir. Özellikle bu tip durumların çok yüksek oranlarda tespit edilemediği gözlenmiştir.

Tablo 5.4 – Sesli harflerin uygun kullanılmadığı durumlarda hataların sezilmesi

	Sözcük	DiGram (Min)	Trigram (Min)	FourGram (Min)	Sessiz Uyumu	Sonda "bcdg"	Sesli Uyumu	Hece Uyumu
	Sayısında	805	38	29	T	T	T	T
*	Sayisinda	605	5	0	T	T	F	T
	Yapılan	546	115	52	T	T	T	T
*	Yapilan	582	9	0	T	T	F	T
	Sorulmaktadır	516	67	20	T	T	T	T
*	Sorulmaktadir	516	67	8	T	T	F	T
	Halkımızı	391	54	17	T	T	T	T
*	Halkımızi	391	2	0	T	T	F	T
	Tanımlanmasıyla	345	26	7	T	T	T	T
*	Tanımlanmasiyla	544	31	0	T	T	F	T

Sesli uyumunun ihlal edilmesi durumunda da yine hatanın sezilmesi mümkün olabilmektedir. Yalnız sesli uyumunun oldukça fazla sözcük tarafından ihlal edilmesinden dolayı hatalı durumların ortaya çıkması olasılığı da oldukça yüksektir.

Buradaki örneklerden de anlaşılacağı gibi tri-gram ve four-gram tarafından hatalı sözcükler oldukça yüksek bir oranda tespit edilebilmektedir. Aslında ne kadar fazla harf kombinasyonu için istatistik elde edilebilirse o kadar yüksek oranda doğru hata tespiti mümkün olabilmektedir.

6. SONUÇLAR

Veri ambarlarında veri kalitesinin artırılması özellikle son yıllarda veri ambarlarının kullanımlarının artmasıyla üzerinde çalışılmaya başlanmış bir konudur. Bu çalışmanın ana hedefi olarak veri ambarlarındaki hatalı verilerin belirlenmesi ve gerekli düzeltmelerin yapılabilmesi amacıyla ne tür yöntemlerin izlenmesi gerektiği üzerinde durulmuş ve bu yöntemlerin bir uygulaması yapılmıştır.

Çalışmada genel olarak iki ana problemin üzerinde odaklanılmıştır.

1. Veritabanlarında Türkçe yazım denetiminin sözlük kullanılmadan yapılabilmesi için Türkçe'nin kurallı yapısından faydalanılmasının yanında istatistiksel yöntemlerin kullanılması.

    - Hece uyumu
    - Sesli uyumu
    - Sessiz uyumu
    - Hece ve sözcük sonunda bulunamayan çift sessizler
    - Sözcük sonunda bulunamayan sessizler(b,c,d,g)
    - N-gram tabloları ile istatistiki denetim

2. Özellikle sistemlerde yer alan tekrarlı kayıtların tespit edilebilmesi için kullanılması gereken yöntemler hakkında bilgiler verilmiş bu yöntemlerin daha etkili bir biçimde uygulanabilmesi için bazı ek özellikler ile zenginleştirilmesi amaçlanmıştır.

    - Sıralı Komşu Metodu (SNM)
    - Dışarıdan kullanıcı tarafından belirlenebilen alan ağırlıkları ile daha etkili bir analiz yapılması hedeflenmiştir.

- Kurulan sistemin her veritabanı için kullanılabilir bir jenerik yapı ile oluşturulması amaçlanmıştır. Böylece farklı sistemler için aynı yapı herhangi bir ek müdahale olmaksızın kullanılabilmektedir.

- Kullanıcı dışarıdan alan ağırlıklarına ek olarak pencere boyutunu ve kayıt benzerlikleri için esas alınacak eşik değerini de girebilecektir.

Özellikle tekrarlı olarak tanımlanan kayıtların belirlenmesi amacıyla hazırlanmış olan uygulama oldukça iyi sonuçlar elde edilmesini sağlamıştır. Bahsi geçen türde kayıtların çok büyük bir kısmının bir istatistiki sonuç elde edilmemiş olmasına rağmen oldukça yüksek oranlarda tespit edilebildiği görülmüştür.

Türkçe yazım denetimi uygulaması için ise çok iyi bir sonuç elde edilemediği görülmektedir. Bunun çeşitli sebepleri vardır. Öncelikle n-gram tablolarının oluşturulması esnasında çok daha fazla teknik ya da sosyal konularda yazılı dökümandan faydalanılması gerektiği açıktır. Çünkü kullanılan örnek sayısının artması ile çok daha sağlıklı n-gram frekanslarının elde edilmesi mümkün olabilecektir. Bu frekans değerlerinin doğruya en yakın hale geldiği durumda elde edilecek olan sonuçlar da çok daha tatmin edici olabilecektir. Bunun dışında diğer Türkçe kurallarından *hece uyumu*, *sessiz uyumu*, *hece ve sözcük sonunda bulunabilecek çift sessizler* ve *sözcük sonunda bulunamayan sessizler* kuralları hataların sezilebilmesi konusunda oldukça iyi sonuçlar vermektedirler. Ancak *sesli uyumu* için bunu söylemek pek mümkün değildir, oldukça fazla sayıda sözcüğün bu kurala uymuyor olması bu kuralın uygulanabilirliği konusunda sıkıntılar doğurmaktadır. Çünkü bu kural kullanılmadığında gerçekten hatalı olan pek çok sözcük sezilememektedir, kullanıldığında ise hatalı olmadığı halde hatalı olarak değerlendirilen çok fazla sözcük ortaya çıkmaktadır. Bunun için bir adet aykırı sözcükler tablosu kullanılarak bu sorun aşılmaya çalışılmıştır. Kullanıcı istediği sözcükleri bu tabloya ekleyerek sistemin bir daha bu sözcükleri hatalı olarak değerlendirmesini engelleyebilecektir. Böylece uygulamanın zamanla, kullanıldıkça çok daha iyi sonuçlar çıkarması mümkün olabilecektir.

Türkçe sözlüksüz yazım denetimi Bilgisayar Mühendisleri tarafından üzerinde çalışılmaya layık bir konudur ve daha sonraki çalışmalarda daha iyi sonuçlar elde

edilebilmesi olasılığı oldukça yüksektir. Bu çalışmalar ile Türkçe'nin henüz gün ışığına çıkmamış bazı özelliklerinin ve kurallarının da bilgisayar desteği ile ortaya çıkarılması ihtimali de bulunmaktadır. Bu çalışmada kullanılan kurallara ek olarak bazı yeni kuralların da kullanılması mümkün olabilirse daha olumlu sonuçlar elde edilebilmesi mümkün olabilecektir. Buna örnek olarak sözcüklerin yapım ve çekim eklerine ayrıştırılarak köklerinin elde edilmesi ve daha sonra elde edilmiş olan eklerin Türkçe kurallarına uyup uymadıklarının kontrol edilmesi verilebilir. Çünkü bu çalışmada sözcüklerin ekleri ile ilgili kontrol yapılmamıştır.

KAYNAKLAR

Hernandez, M.A. and Stolfo, S.J., 1998. Real World Data is Dirty: Data Cleansing and The Merge/Purge Problem. *To appear on the Journal of Data Mining and Knowledge Discovery, Vol 2,1998*

Jin, L., Li, C. and Mehrotra, S., 2003. Efficient Record Linkage in Large Data Sets *8th International Conference on Database Systems for Advanced Applications (DASFAA), March 2003*

Kukich, K., 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys, 24(4):377-439, December 1992*

Last, M. and Kandel, A., 2001. Automated Detection of Outliers in Real-World *Data Proceedings of the Second International Conference on Intelligent Technologies (InTech'2001), pp. 292-301, Bangkok, Thailand, November 27-29,2001*

Lee, M.L., Lu, H., Ling, T.W. and Ko, Y.T., 1999**.** Cleansing Data for Mining and Warehousing. *In Proceedings of sixth ACM SIGKDD international conferenceon Database and Expert Systems Applications (DEXA), pages 751-760, 1999*

Monge, E.A and Elkan, C.P., 1997. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. *In Proceedings of the 1997 SIGMOD Workshop on Research Issues on DMKD, 1997*

Pyle, D., 1999**.** Data Preparation for Data Mining. *Morgan Kaufmann Inc., ISBN 1-55860-529-0, 1999*

Rahm, E. and Do, H. H., 2000. Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin, 23(3), September, 2000*

Raisinghani, T. V., 1999**.** Cleaning Methods In Data Warehousing, *KR School of Information Technology IIT, Bombay, December 1999*

Sung, S.Y., Li, Z. And Sun, P., 2002. A Fast Filtering Scheme for Large Database *Cleansing. International Conference on Information and Knowledge Management (CIKM'02), McLean, Virginia, USA, November 4-9, (EI),2002*

Tantuğ, A. C., 2002. Veri Madenciliği ve Demetleme, *Yüksek Lisans Tezi, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul, Mayıs 2002*

ÖZGEÇMİŞ

Metin ÇINAR, 15.08.1977 tarihinde Sivas'da doğmuştur. Lise eğitimini İstanbul Alibeyköy Teknik Lisesi Elektrik Bölümü'nde tamamladıktan sonra İstanbul Teknik Üniversitesi Elektrik-Elektronik Fakültesi Kontrol ve Bilgisayar Mühendisliği bölümününden ilk yılı ingilizce hazırlık olmak üzere beş yıllık eğitimin ardından 2000 yılında mezun olmuştur. 2000-2003 yılları arasında ağırlıklı olarak kablosuz cihazlar için yazılım geliştirmekte olan özel bir şirkette Bilgisayar Mühendisi olarak çalışmıştır. Halen aynı firmada çalışmalarına devam etmektedir.