YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VERİ MADENCİLİĞİNDE MARKET SEPET ANALİZİ VE BİRLİKTELİK KURALLARININ BELİRLENMESİ

Bilgisayar Müh. Ayhan DÖŞLÜ

FBE Bilgisayar Mühendisliği Anabilim Dalı Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Yrd. Doç. Dr. Songül ALBAYRAK

İSTANBUL, 2008

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

I	Nesneler kümesi (Set of items)
I_d	Nesne (item)
m	Nesne sayısı (number of items)
D	Hareketsel veritabanı (transaction database)
s	Destek (support)
c	Güven (confidence)
T	Hareket (transaction)
X, Y	Nesnekümeler (itemsets)
$X \Rightarrow Y$	Birliktelik kuralı (association rule)
L	Yaygın nesnekümeler kümesi (set of large itemsets)
l	Yaygın nesnekümesi (large itemset)
L_k	k uzunluklu yaygın nesnekümeler kümesi (set of large itemsets of size k)
l_k	k uzunluklu yaygın nesnekümesi (large itemset of size k)
C_k	k uzunluklu aday kümeler (candidate sets of size k)
$\overline{L_k}$	k uzunluklu ve TID içeren yaygın nesnekümeler kümesi (set of large itemsets of size k and the TID containing them)
$\overline{C_k}$	k uzunluklu ve TID içeren aday nesnekümeler kümesi (set of candidate itemsets of size k and the TID containing them)
D^i	D veritabanında i bölmesi (partition i for database D)
X^i	D^i bölmesindeki nesnekümesi (itemset for partition D^i)
L^i	D^i bölmesindeki yaygın nesnekümeler kümesi (set of large itemsets for partition D^i)
C^i	D^i bölmesindeki aday nesnekümeler kümesi (set of candidate itemsets for partition D^i)
p	bölme sayısı (number of partitions)

KISALTMA LİSTESİ

BFS	Breadth-First Search
DBA	Database Administrator
DBMS	Database Management Systems
DFS	Depth-First Search
FIFO	First In First Out
G/Ç	Giriş/Çıkış
IT	Information Technology
KDD	Knowledge Discovery in Databases
LIFO	Last In First Out
OLAP	On-line Analytical Processing
OLTP	On-line Transaction Processing
RDBMS	Relational Database Management Systems
SQL	Structured Query Language
TID	Transaction ID

ŞEKİL LİSTESİ Sayfa

ÇİZELGE LİSTESİ Sayfa

ÖNSÖZ

Verilerin sayısal ortamda saklanmaya başlanması ile birlikte, veri miktarının her yirmi ayda bir iki katına çıktığı varsayılmaktadır. Bu büyük miktardaki ham veri selinden gelecekle ilgili tahmin yapılmasını sağlayan anlamlı bilgilerin, bağıntı ve kuralların keşfedilmesi gerekir. Bu kurallara dayanarak belirlenen stratejiler ile şirket karı arttırılabilir. Örneğin, süpermarketlerde veri analizi yaparak her ürün için bir sonraki ayın satış tahminleri çıkarılıp birlikte satın alınan ürünler için promosyon uygulaması ve reyon dizilişleri yapılabilir, müşteriler satın aldıkları ürünlere göre gruplandırılabilir, yeni bir ürün için potansiyel müşteriler belirlenebilir. Binlerce ürünün ve müşterinin olacağı düşünülürse bu analizlerin gözle ve elle yapılamayacağı, bilgisayar programları aracılığıyla otomatik olarak yapılması gerektiği ortaya çıkar. Bu ihtiyaçlar veri madenciliğinin ve tekniklerinin ortaya çıkmasına sebep olmuştur.

Yakın geleceğin geçmişten çok fazla farklı olmayacağı varsayılırsa, geçmiş veriden çıkarılmış olan kurallar gelecekte de geçerli olacak ve ilerisi için doğru tahmin yapılmasını sağlayacaktır. Bu maksatla veri madenciliği alanında birliktelik kuralı madenciliği son zamanlarda önem kazanarak birçok araştırmanın konusu olmuştur.

Bu tez çalışmasında veri madenciliği ve veritabanlarında bilgi keşfi süreci içinde yer alan temel kavramlar, yöntem ve teknikler ele alınmış olup, birliktelik kuralları ve bu kuralların çıkarılması için kullanılan algoritmalar araştırılmıştır ve örnek veri setleri üzerinde uygulama yapılmıştır.

Çalışmam boyunca değerli fikir ve önerileriyle beni yönlendiren, her konuda destek veren tez danışmanım Sayın Yrd. Doç. Dr. Songül Albayrak'a, gerçek verileri içeren bir market alışveriş verisi almamda emeği geçen Migros Türk T.A.Ş. Genel Müdürlüğü personeline, anlayışları ve gösterdikleri hoşgörüden dolayı tüm iş arkadaşlarıma ve her zaman destek ve dualarını yanımda hissettiğim canım aileme saygı ve içtenlikle teşekkürlerimi sunarım.

VERİ MADENCİLİĞİNDE MARKET SEPET ANALİZİ VE
BİRLİKTELİK KURALLARININ BELİRLENMESİ

ÖZET

Günümüzde teknoloji sayesinde çok büyük miktarda veri elde edilip saklanabilmektedir. Ancak bu büyük miktardaki verilerden gözle görülemeyecek, elle analiz edilmesi zor bilgilerin gelişen bilgisayar teknolojisi ve bilgisayar programları ile otomatik olarak analiz edilmesinin gerekliliği ortaya çıkmaktadır. Verikümelerinden örüntülerin, eğilimlerin ve anormalliklerin bulunarak basit modeller şeklinde özetlenmesi, bilgi çağındaki büyük uğraşıların başında gelir. Veri madenciliği, büyük miktardaki mevcut veri içinden anlamlı, potansiyel olarak kullanışlı, gelecekle ilgili tahmin yapılmasını sağlayan bağıntı ve kuralların bilgisayar programları kullanarak bulunmasıdır. Birçok sektörde kullanımı giderek yaygınlaşan veri madenciliğinin uygulama alanlarından biri de süpermarketlerdeki müşteri, ürün ve satış bilgilerinden yararlanarak ilişki ve kuralların elde edildiği market sepet analizidir. Market sepet analizinde ürünlerin birbiriyle olan satış ilişkilerinin elde edilmesi ve veri madenciliği konularından biri olan birliktelik kurallarının çıkarılması, şirketlerin kârını arttırıcı etkenlerdir. Birliktelik kuralları, satış hareket verileri içinde birlikte hareket eden nesnelerin ve nesneler arasındaki bağıntıların keşfedilerek geleceğe yönelik tahminlerin üretilmesini sağlar. Bu kuralların elde edilebilmesi için 90'lı yılların başından itibaren birçok algoritma geliştirilmiştir. Bu algoritmaların birbirine göre farklı koşullar altında üstünlükleri ve farklı çalışma yöntemleri mevcuttur. Veritabanının taranması, birleştirme, budama yöntemlerinin uygulanması ve minimum destek değeri yardımı ile nesneler arasındaki birliktelik ilişkilerinin bulunması, algoritmaların genel mantığını teşkil eder.

Bu tez çalışmasında, veri madenciliği ile ilgili kavramlar ve özellikle market sepet analizinde kullanmak üzere birliktelik kuralları üreten temel algoritmalar detaylı bir şekilde ele alınmış ve birbiriyle karşılaştırılmıştır. Ayrıca, örnek veri setlerinden iki farklı algoritma ile birliktelik kurallarını bulan bir uygulama geliştirilmiştir.

Anahtar Kelimeler: Veri madenciliği, Market sepet analizi, Birliktelik kuralları, Birliktelik kural madenciliği algoritmaları, Apriori algoritması, FP-Growth algoritması.

MARKET BASKET ANALYSIS IN DATA MINING AND FINDING ASSOCIATION RULES

ABSTRACT

Today, large amounts of data can be collected and stored by using technology. However, there is a necessity of automatic analysis using computer technology and computer programmes which is developing day by day in order to analyze the data, that is difficult to be analyzed by manuel and can not be seen. Making summaries in the simple way by finding patterns, tendencies, anormalities from the database is one the most common thing in the information age. Data mining is the process of finding the rules and the correlations among the large amounts of data by the computer programmes, which are understandable, potentially useful and provide predictions about the future. The utilization of data mining in a wide selection of fields is increasing. One of the areas is the market-basket analysis that is to have the rules and associations from the data about customer, products and sales. In this analysis, gathering the association rules-one of the subjects in the data mining- and having the sales relationships between the products are two factors of increasing rate of profit in the companies. Association rules provide predictions about the future by discovering relations between the objects which act together in the transactional sales data and the objects. Lots of algorithms has been developed since the beginnings of 1990's. These algorithms have different working methods and different superiorities on each other in the different conditions. The common logic of these algorithms is that passing over the database, combining, pruning and finding the association rules between the items by using the minimum support threshold.

In this thesis, concepts about the data mining and basic algorithms especially using in the market-basket analysis to produce the association rules are examined in details and compared with each other. Also, an application is developed to find association rules from sample datasets by using two different algorithms.

Keywords: Data mining, Market-basket analysis, Association rules, Association rule mining algorithms, Apriori algorithm, FP-Growth algorithm.

1. GİRİŞ

Her geçen gün hem ucuzlayan hem de işlemci hızları ve disk kapasiteleri artan bilgisayar sistemlerinde büyük miktardaki veriler saklanıp işlenmektedir. Verilerin sayısal olarak toplandığı ve saklandığı bu teknolojilerde veri miktarının hızla artmasına rağmen bu artışa oranla bu verilerden elde edilen bilgi miktarının yeterli düzeyde olduğu söylenemez.

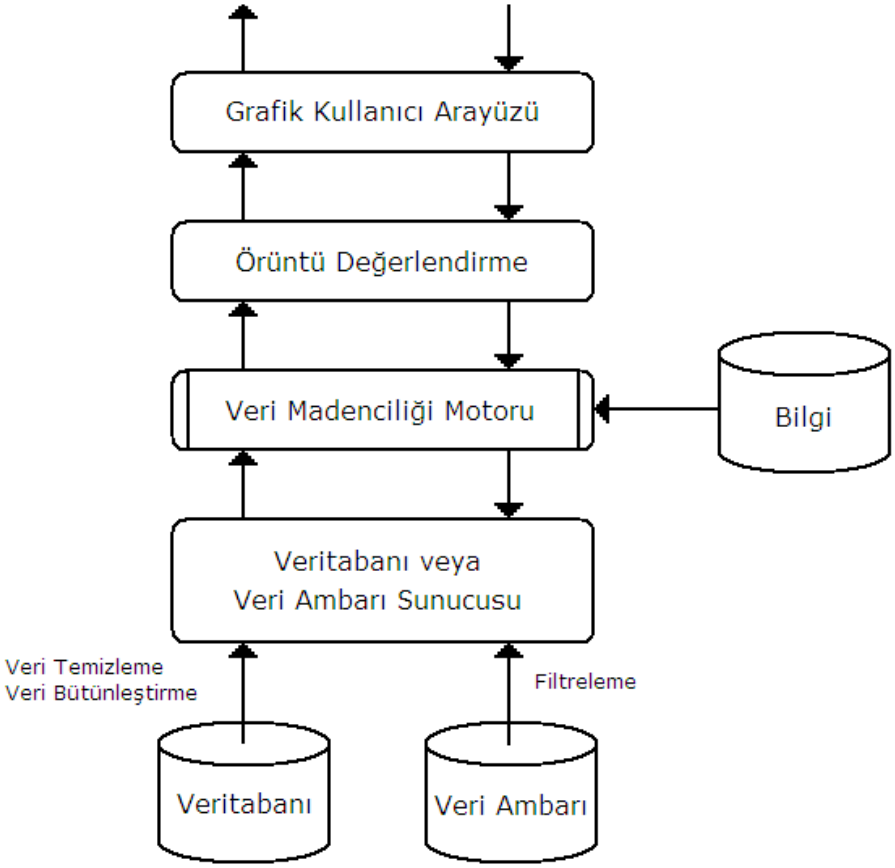
Bilgi sistemlerinin ve teknolojilerinin son zamanlarda gelişmesine paralel olarak büyük marketler, işletmeler ve diğer kuruluşlar kendi amaçlarına ve yapılarına göre veritabanlarında çeşitli türlerde veri toplamıştır. Alışveriş sektöründe, bankacılık işlemlerinde, kamusal alandaki işlemlerde ve buna benzer birçok alanda depolanan geniş hacimdeki ve dağınık verilerden anlamlı ve verimli şablon ve kuralların keşfedilmesine ihtiyaç duyulmaktadır. Saklı ve işlenmemiş bu verilerden yeni, geçerli, faydalı ve sonuç olarak anlaşılabilir örüntülerin çıkarılmasındaki bu bilgi keşfi sürecine *Veritabanlarında Bilgi Keşfi* (Knowledge Discovery in Databases - KDD) denir.

Veritabanlarında bilgi keşfi sürecinin bir aşaması olarak kabul gören *Veri Madenciliği* (Data Mining), eldeki verilerden üstü kapalı, çok net olmayan, önceden bilinmeyen ancak potansiyel olarak kullanışlı bilginin çıkarılmasını sağlar. Bir başka ifadeyle büyük miktardaki verinin analiz edilerek anlamlı şablon ve kuralların keşfedilmesine imkan verir (Berry ve Linoff, 2004).

Veritabanlarından birliktelik kurallarının bulunması veri madenciliğinin en önemli konularından biri olup, bir arada sık olarak görülen ilişkilerin ortaya çıkarılmasını ve özetlenmesini sağlar. Örneğin, bir alışveriş sırasında müşterinin hangi ürün veya hizmetleri satın almaya eğilimli olduğunun belirlenmesi, müşteriye daha fazla ürünün satılmasını sağlayarak şirket kârını arttırıcı rol oynar. Satın alma eğilimlerinin tanımlanmasını sağlayan birliktelik kuralları ve ardışık zamanlı örüntüler, pazarlama amaçlı olarak market sepet analizi adı altında veri madenciliğinde sıkça kullanılmaktadır. Market sepet analizine ek olarak bu teknikler tıp, finans gibi farklı olayların birbiriyle ilişkisinin belirlenmesinin gerekli olduğu alanlarda da tercih edilmektedir.

Birliktelik kuralları, satış hareket verileri içinde birlikte hareket eden nesnelerin ve nesneler arasındaki bağıntıların keşfedilerek geleceğe yönelik tahminlerin üretilmesini sağlar. Bu amaçla birliktelik kuralları madenciliğinde kullanmak üzere bu kuralların elde edilebilmesi için 90'lı yılların başından itibaren birçok algoritma geliştirilmiş olup bu algoritmaların birbirine göre farklı koşullar altında üstünlükleri ve farklı çalışma yöntemleri mevcuttur. Ancak birliktelik kurallarının çıkarılmasında en çok bilinen ve uygulanan algoritma, Apriori

algoritması olmuştur. Veritabanının taranması, birleştirme, budama yöntemlerinin uygulanması ve minimum destek değeri yardımı ile nesneler arasındaki birliktelik ilişkilerinin bulunması, algoritmaların genel mantığını teşkil etmektedir.



Şekil 1.1 Veri madenciliği sistemi mimarisi (Han ve Kamber, 2000)

Birliktelik kuralları aşağıdaki örneklerde de olduğu gibi eşzamanlı olarak gerçekleşen birlikteliklerin tanımlanmasında kullanılır:

- Kola satın alan müşteriler 40% olasılıkla patates cipsi de alırlar.
- Yağsız yoğurt ve düşük yağlı peynir alan müşteriler 85% olasılıkla diyet süt de satın alırlar.

Ardışık zamanlı örüntüler ise aşağıdaki örneklerde görüldüğü gibi birbirini izleyen dönemlerde gerçekleşen ilişkilerin tanımlanmasında kullanılır:

- X ameliyatından sonra onbeş gün içinde 45% olasılıkla Y enfeksiyonu oluşacaktır.
- Çekiç satın alan bir müşteri, ilk üç ay içerisinde 15%, bu dönemi izleyen üç ay içerisinde 10% olasılıkla çivi satın alacaktır.

Birliktelik kurallarının en önemli uygulama alanları arasında;

- Market sepet analizi,
- Çapraz-pazarlama (cross-marketing),
- Promosyon analizleri,
- Katalog ve yerleşim düzeni tasarımları bulunmaktadır.

Bu doğrultuda tez çalışmasının amacı, veri madenciliği ve veritabanlarında bilgi keşfi süreci içinde yer alan temel kavramların, yöntem ve tekniklerin araştırılması, market sepet analizinde birliktelik kurallarının keşfedilmesi sürecinin ve bu kuralların çıkarılması için kullanılan algoritmaların detaylıca incelenmesi ve kıyaslanmasıdır.

Çalışmanın ikinci bölümünde veri madenciliği kavramı öncesinde bilinmesi gereken veritabanı ve veri ambarı ile ilgili temel kavramlar hakkında bilgi verilmiştir. Üçüncü bölümde veri madenciliği hakkında inceleme yapılmış olup, veri madenciliği ve bilgi keşfi sürecine değinilmiştir. Dördüncü bölümde ise birliktelik kuralları madenciliği kavramı ve bu amaçla kullanılan algoritmalar ele alınmıştır.

Çalışmanın sonunda, en çok bilinen birliktelik kuralı algoritması olan Apriori algoritmasını baz alarak yazılan program ve sektörde aktif olarak kullanılan bir-iki veri madenciliği aracı tarafından örnek veri seti üzerinde birliktelik kurallarının çıkarılması ele alınacaktır.

2. VERİTABANI ve VERİ AMBARI KAVRAMLARI

2.1 Veri, Bilgi ve Metaveri

Günlük hayatta *veri* (data), *bilgi* (information) ile eş anlamlı olarak kullanılır. Ancak, düzenlenmemiş bir ölçüm olarak nitelendirilebilecek veri düzenlendiğinde bilgiye dönüşmektedir. Veri kendi başına değersizdir, hiçbir anlam ifade etmez. Örneğin veritabanından alınan "345" verisi müşteri ID'si mi, tutar mı yoksa ürün numarası mı diye bilinmiyorsa, bu veri bilgi içermez. İsteğimiz amacımız doğrultusunda bilgidir. Bilgi, bir amaca yönelik işlenmiş veridir. Bir soruya yanıt vermek için veriden çıkarılan olarak tanımlanabilir (Alpaydın, 2000).

Veri + Açıklama (+ Analiz) = Bilgi (2.1)

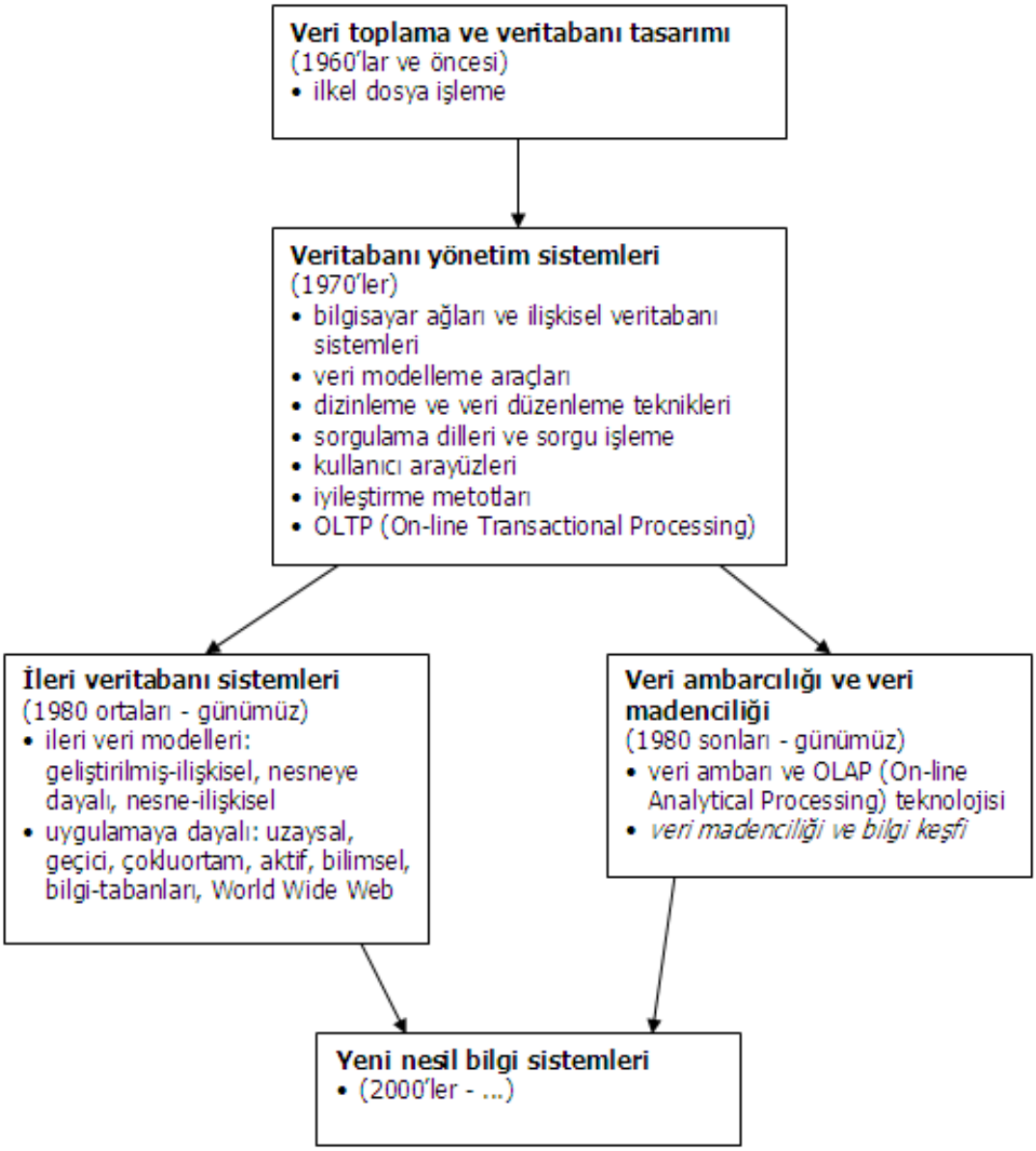
Veriyi bilgiye çevirmeye *veri analizi* denir. Veriyi oluşturan sayılar, harfler ve onların anlamı *metaveri* (metadata, üstveri) olarak bilinir. Metaveri, "veri hakkındaki veri" olarak tanımlanabilir.

2.2 Veritabanı Sistemleri

Veritabanı analizinde bir bilgi birçok veri kullanılarak elde edilebilir. İş dünyası ve şirketler, etkin yönetimi sağlamak ve kazançlarını ve gelirlerini en üst düzeye çıkarmak için bilgiyi yönetmeye ihtiyaç duyarlar. Birçok fatura ve kağıt parçası içinden yöneticinin sorduğu sorulara cevap vermek zor olsa da, bilgisayarların sevdiği iş olarak bu tekrarlanan görevleri yerine getirmek ve sorulara doğru cevaplar bulmak kolaylaşmaktadır.

Veritabanı (database), sistematik erişim imkanı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen düzenli bilgiler topluluğudur. Bir başka tanımı ise, bir bilgisayarda sistematik şekilde saklanmış, programlarca istenebilecek veri yığınlarına veritabanı denir. Bir veritabanını oluşturmak, saklamak, çoğaltmak, güncellemek ve yönetmek için kullanılan programlara *Veritabanı Yönetim Sistemleri* (Database Management Systems - DBMS) adı verilir. DBMS özelliklerinin ve yapısının nasıl olması gerektiğini inceleyen alan *Bilgi Bilimi* (Information Science)'dir.

Veritabanında kayıt yığını ya da bilgi parçalarının tanımlanmasına *şema* adı verilir. Şema, veritabanında kullanılacak bilgi tanımlarının nasıl modelleneceğini gösterir. Buna *veri modeli* (data model) denir. En yaygın olanı, verilerin tablolarda saklandığı *ilişkisel model* (relational model) 'dir. Tablolarda bulunan *satır*lar (row) kayıtların kendisini, *sütun*lar (column) ise bu kayıtları oluşturan bilgi parçalarının ne türden olduklarını belirtir.

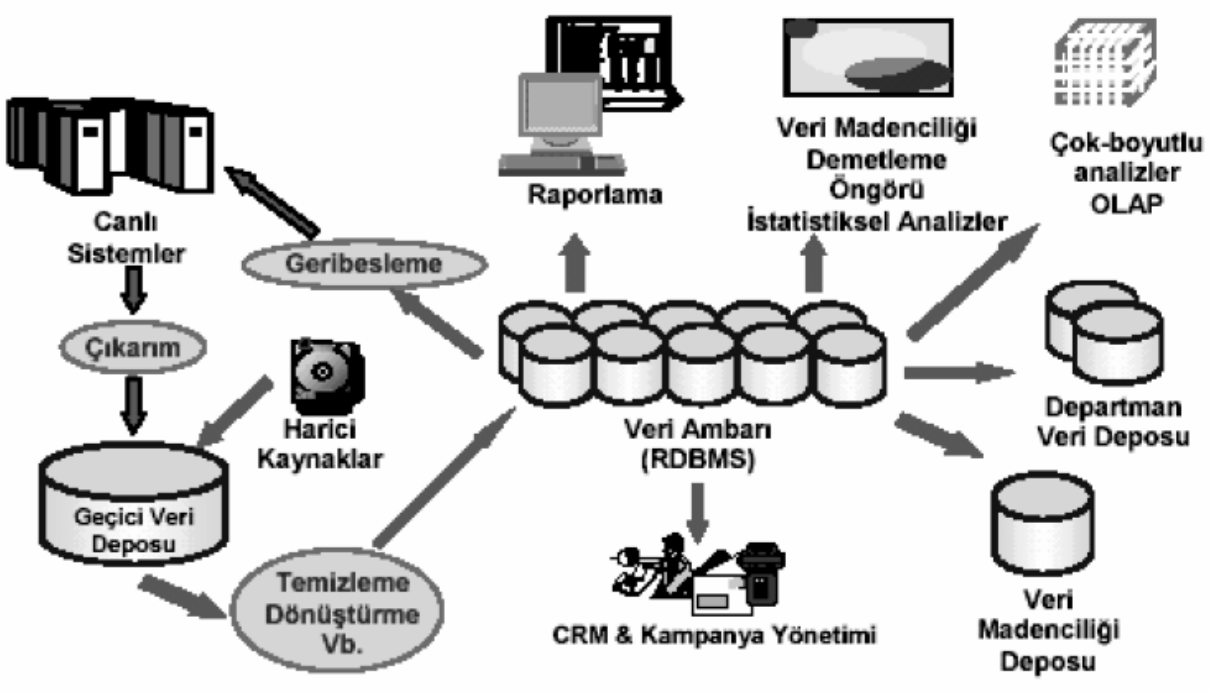


Şekil 2.1 Veritabanı teknolojisinin gelişimi (Han ve Kamber, 2000)

Veritabanı yazılımı ise verileri sistematik bir biçimde depolayan yazılımdır. Birçok yazılım bilgi depolayabilir ama aradaki fark, veritabanının bu bilgiyi verimli ve hızlı bir şekilde yönetip değiştirebilmesidir. Veritabanı, bilgi sisteminin kalbidir ve etkili kullanmakla değer kazanır. Bilgiye gerekli olduğu zaman ulaşabilmek esastır. *İlişkisel Veritabanı Yönetim Sistemleri* (Relational Database Management Systems - RDBMS) büyük miktardaki verilerin güvenli bir şekilde saklanabildiği, bilgilere hızlı erişim imkanının sağlanabildiği, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkanının sağlanabildiği programlardır [8].

2.3 Veri Ambarları

Veri ambarı (data warehouse) ilişkili verilerin sorgulanabildiği ve analizlerinin yapılabildiği bütünleşmiş bir bilgi deposudur. Veri ve bilgiler, üretildiklerinde heterojen kaynaklardan elde edilirler. Veri ambarları, başlangıçta farklı kaynaklardan gelen verinin üzerinde daha etkili ve daha kolay sorguların yapılmasını sağlamaktadır.



Şekil 2.2 Veri ambarı mimarisi (Tantuğ, 2002)

Veri ambarları, sağlık sektöründen coğrafi bilişim sistemlerine, işletmelerin pazarlama bölümünden üretime, geleceğe dönük tahminler yapmak, sonuçlar çıkarmak ve işletmelerin yönetim stratejilerini belirlemek için kullanılan bir sistemdir. Pahalı bir yatırım maliyeti olsa bile sonuç olarak getirisi ve yararı bu maliyetleri kat kat aşmaktadır.

İş organizasyonlarının bilgi akış mimarisinde veri ambarları iki amaçla oluşturulur:

1. Hareketsel ve organizasyonel görevler arasındaki depo ve analitik stratejik verilerin birikimini sağlar. Bu veriler daha sonra yeniden kullanılmak üzere arşivlenir. Veri ambarları verilerin sorgulanabildiği ve analiz yapılabilindiği bir depodur.
2. Pazarda yeni fırsatlar bulmak, rekabete katkı sağlamak, yoğun proje çevirimine yardımcı olmak, iş, envanter ve ürün maliyetlerini azaltmak gibi imkanların yanında farklı işlere ait verilerin ilişkilendirilmesi, alınan bilgiye hızlı cevap verebilme ve karar destek gibi birçok alanda veri ambarlarının katkısı büyüktür.

Veri ambarının en önemli bileşenlerinden biri metaveridir. Veri ambarında verilerin tanımlandığı kısımdır. Daha önce de belirtildiği gibi metaveri "veri hakkında veri" anlamındadır. Metaveri her veri elementinin anlamını, hangi elementlerin hangileriyle nasıl ilişkili olduğunu ve kaynak verisi ile erişilecek veri gibi bilgileri içermektedir.

Veri ambarındaki veriler, veri ambarı yöneticisinin kullandığı teknik veriler ve veri ambarı kullanıcılarının kullandığı iş verileri olarak ikiye ayrılır:

1. **Teknik veriler**: Operasyonel veritabanı tanımlarını ve veri ambarı tanımlarını içerir. Bu iki tanım veya şema veri ambarını çalıştırılabilmesini sağlayan veri taşıma operasyonlarını içerir. Bu bilgiler veri ambarı yöneticisine veri ambarında birbiriyle ilişkili verileri göstererek yardımcı olan bilgilerdir.
2. **İş verileri**: Kullanıcılara yardım eder. Kullanıcıların veritabanı oluşturan veriler dışındaki veri ambarında bulunan bilgilere ulaşmalarına yardımcı olur. Ayrıca veri ambarına verinin ne zaman ve nereden geldiği gibi bilgilerede ulaşılmasını sağlar [6].

2.3.1 Datamart

Datamart (veri pazarı) şirketlerde belirli bir bilgi kullanıcısı grubunun ihtiyacına yönelik olarak hazırlanan küçük boyutlu veri ambarı olarak tanımlanabilir. Datamartlar veri ambarının alt kümesi olan 1 - 10 GB'lık bölümsel ambarlardır ve organizasyonların ve işletmelerin belirli kullanıcıları için ayrılmış ve onlara ait verileri içerirler [6]. Datamartların tercih edilme nedenleri şunlardır:

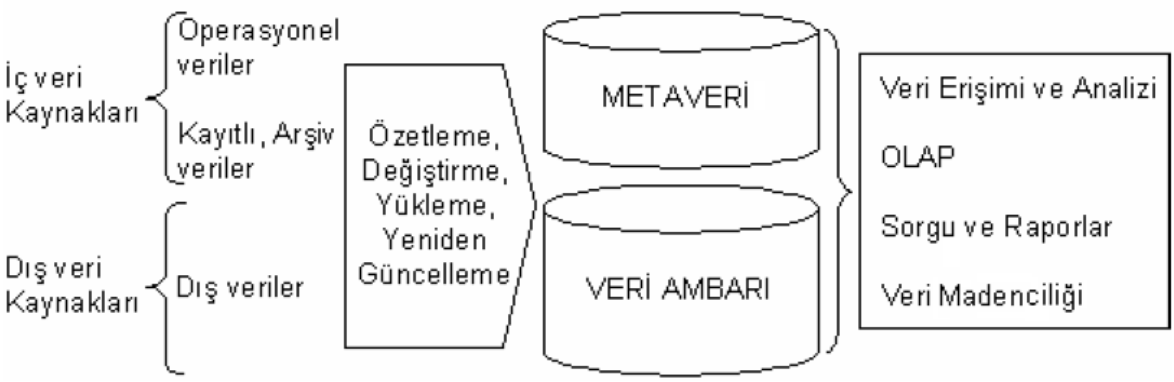
- Sık ihtiyaç duyulan veriye kolayca erişim sağlamak,
- Bir kullanıcı grubu için ortak bakış oluşturmak,
- Son-kullanıcı yanıt süresini geliştirmek,
- Kolaylıkla tasarlanabilmesi,
- Tüm veri ambarı tasarımına göre daha az maliyetli olması,
- Muhtemel kullanıcıların tüm veri ambarına göre daha net tanımlanabilmesi.

2.3.2 Veri Ambarı Bileşenleri ve Fonksiyonları

Veri ambarı bileşenleri ve fonksiyonları ise şu şekilde belirtilebilir:

- Değişik platformlar üzerindeki işletimsel uygulamalara ait verilere erişim ve gerekli verilerin bu platformlardan alınması,

- Alınan verilerin temizlenmesi, tutarlı duruma getirilmesi, özetlenmesi, birleştirme ve birbirleriyle entegrasyonunun sağlanması,
- Dönüştürülen verilerin veri ambarı veya datamart ortamına dağıtımı,
- Gönderilen verilerin bir veritabanında toplanması,
- Depolanan bilgi ile metaveride bulunan ilgili bilgilerin veri kataloğunda saklanması ve son kullanıcılara sunulması,
- Veri ambarı veya datamartlarda bulunan bilgileri uç kullanıcıların karar destek amaçlı kullanımının sağlanması.



Şekil 2.3 Veri ambarı bileşenleri

2.4 OLTP (Online Transaction Processing) Sistemleri

Birçok veritabanı sistemi *OLTP* (Online Transaction Processing) sistemlerde tutulur. Eşzamanlı veritabanı bağlantısını yönetmek için tasarlanmış bu sistemlerde az sayıdaki kayıt için kaydetme, güncelleme, sorgulama gibi işlemler yapılmaktadır.

OLTP sistemini içeren hareket (transaction) örneklerinden bazıları şöyledir:

- Satış veritabanına 325903 nolu müşteri için bir kayıt gir ve detaylarını gir,
- 583472 nolu müşterinin faturalanmamış siparişlerini göster,
- 1032 nolu tedarikçi firmanın adresini değiştir.

Bu gibi hareketlerin ortak paylaştığı özellikler ise:

- Belirli bir zamanda ayrık satırlardaki küçük sayıdaki verileri işleme,
- Herbir işlemin, verinin o anki gerçek değerleriyle güncellenmesini gerektirmesi,
- Kullanıcıların memnun olması için hemen hemen anlık yanıtlara ihtiyaç duyması.

“Şu ana kadar hangi ürün pazar payının en fazla azalmasına neden olmuştur?” gibi analitik bir sorgulama bu sistemlere sorulduğunda, sınıflandırma ve milyonlarca kayıttan cevabı bulma sırasında sistem zorlanmakta ve uzun süre almaktadır.

2.5 OLAP (Online Analytical Processing) Sistemleri

İlişkisel veri tabanlarının yaygınlığı ve sonrasında ortaya çıkan veri ambarlarının gelişmesi ile beraber, verilere daha hızlı şekilde erişme ve çok boyutlu analiz ihtiyaçları, bilim adamlarını ve yazılım şirketlerini, daha farklı yapılar geliştirmeye itmiştir.

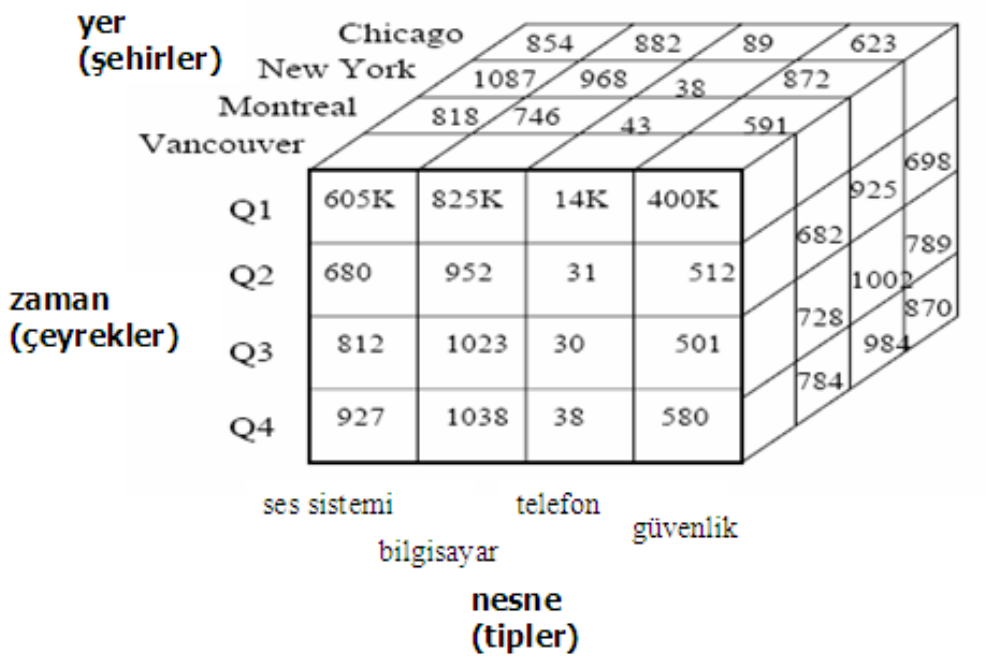
Bu amaçla geliştirilen bir teknoloji olan OLAP (Online Analytical Processing), ilişkisel veri tabanları gibi, bilimsel temeller üzerine değil, OLAP ürünleri üreten firmaların desteğinde çıkan bir teknoloji olmuştur.

OLAP, yöneticiler ve analistlerin, verilere çok hızlı şekilde, farklı açılardan bakabilmelerini sağlayan bir yapıdır. “Kim?” ve “Ne Zaman?” sorularından başka, “Neden?” ve “Eğer şu olursa...” sorularının da yanıtını verir. Örneğin; “Eğer şeker fiyatları 5% ve taşıma maliyetleri 10% düşerse, yıllık ve çeyrekler bazında kârlılık ne olur?” gibi soruların yanıtlarını bu sistemlerde akıllı raporlama araçları sayesinde almak oldukça kolaydır. Genel eğilimden farklılık gösteren, uç değerler teşkil eden elemanları birçok analiz aracı, sayısal detaylara girmeden, sadece renklerle bile görüntüleyebilmektedir.

Han ve Kamber’e göre veri ambarları ve OLAP teknolojisi çok boyutlu veri modelleri üzerine kurulmuştur. Çok boyutlu veri modeli veriyi *veri kübü* (data cube) formunda incelemektedir. Veri kübü, verinin çok boyutta modellenmesini ve incelenmesini sağlar ve *olgu tablosu*ndan (fact table) beslenen *boyutlar* (dimension) ve *ölçümler*den (measure) oluşur. Boyutlar, organizasyonun kayıtlarını tutmak istediği perspektifler veya varlıklar ile ilgili iken, ölçümler rakamsal değerler, elde edilmek istenen sonuçlardır.

OLAP basit olarak üç aşamadan oluşur; veri ambarından veri seçimi, veri kübü kurma ve veri kübü üzerinde çevrim içi analizin yapılacağı uygulama. Çok boyutlu veri küpleri üzerinde birçok OLAP işlemi uygulanmaktadır;

- Roll-up (detaylandırma),
- Drill-down (detay azaltma),
- Slice and dice (dilimleme ve parçalama),
- Pivot (çevirim), ... vb.



Şekil 2.4 yer, zaman, nesne boyutlarını (dimension) ve rakamsal ölçümleri (measure) içeren 3-boyutlu OLAP veri kübü (Han ve Kamber, 2000)

Bir veri ambarının olması, OLAP'a ihtiyaç olmadığı anlamına gelmez. Veri ambarları ve OLAP birbirlerini tamamlar. Veri ambarı verileri uygun şekilde tutmaya ve kontrol etmeye yarar. OLAP ise, veri ambarı verilerini stratejik bilgilere dönüştürmeye yarar.

Bir şirket yapısı içerisinde, departmanlar bazında incelenecek olursa:

- Pazarlama departmanlarında OLAP'ın en yaygın kullanım alanları, pazar araştırmalarında, satış tahminleri, promosyon ve kampanya analizleri, müşteri analizleri ve pazar/müşteri segmentasyonlarıdır. Veri madenciliği sonuçlarının değerlendirilmesi ve demografikler bazında incelenmesi seviyesinde de olmazsa-olmaz araçlardan biri olarak yer almaktadır.
- Üretim ile ilgili uygulamaları ise en yoğun olarak üretim planlama ve hata analizleridir. Özellikle senaryo geliştirmekte ve farklı ürün tipleri ile çalışılan yapılarda, çok boyutlu düşünme imkanı sayesinde maliyetler ve fiyatlamalar kolaylıkla çıkarılabilmektedir.
- Finans departmanları ise OLAP'ı bütçeleme, activity-based costing, finansal performans analizleri ve finansal modelleme amaçları ile kullanabilir. Özellikle birlik konusunda oluşturulacak modeller, çok büyük kolaylıklar sağlamaktadır. Strateji belirleme, satış analizleri ve gelecek tahminleri ise, satış departmanlarındaki OLAP

uygulamalarıdır.

2.5.1 OLAP Kuralları

OLAP teriminin ilk olarak ortaya çıkışı 1993 yılında, Dr. E. F. Codd'un ortaya koyduğu kurallar çerçevesinde olmuştur. Buna göre, bir veri yapısının OLAP olarak nitelendirilebilmesi için oniki kural belirlenmiştir. Bu kurallar sırası ile:

1. Çok boyutlu inceleme özelliğine sahip olması,
2. Şeffaflık,
3. Erişilebilirlik,
4. Her seviyede sorgulama için aynı performansı gösterebilme özelliği,
5. İstemci-Sunucu yapısında olması,
6. Sınırsız şekilde çarpraz raporlama olanağının olması,
7. En alt seviyedeki verilerin otomatik olarak ayarlanması,
8. Her şarta uygun boyutlandırılabilirlik,
9. Çok kullanıcı desteğinin olması,
10. Her seviyede verilerin değiştirilebilir olması,
11. Esnek raporlama özelliği,
12. Boyut ve gruplamalarda sınır olmamasıdır.

2.5.2 OLAP Özellikleri

Zaman kazancının dışında, OLAP üç çok önemli özelliği de beraberinde getirmektedir;

Verilere çok boyutlu bakabilme özelliği:

Analizler sırasında kullanılan her türlü kırılıma boyut adı verilir. Örneğin demografik veriler (yaş, cinsiyet, eğitim durumu), sayısal veriler, adetler, işlem miktarları, gerçekleşen ve bütçelenen değerler, ürün tüpleri, ürün özellikleri ve zaman. Yöneticiler ve analistler çalışmaları sırasında tüm bu tanımlanan verileri yatay veya düşey eksenlerde çakıştırarak görmek isteyebilirler.

İlişkisel veri tabanları, bu şekilde raporlara izin vermezler, fakat raporlama araçlarının yetenekleri ile, belirli bir noktaya kadar tolere edilebilir. Fakat daha karmaşık analizler işin içine girdiğinde, bir OLAP yapısı kurmadan bu raporları almak imkansız hale gelmektedir.

İlişkisel veri tabanları üzerinde karmaşık SQL kodları yazmak ya da raporlama aracının sahip olduğu programlama dili üzerinde uğraşmak gerekebilir. Bu da analizi yapan kişilerin işin özünden çıkarak, analiz gerektirebilecek verilere değil teknik olanaklara, daha kolay şekilde alabilecekleri verilere kanalize olmaları sonucunu doğurur. Bu nedenle iş zekası programlarının pratik olmasının yanında fazla teknik bilgi kullanmadan raporların alınabilir olması, farklı kaynakları bir arada kullanabilecek, konsolide edebilecek yapıda olmaları gerekir.

Boyutların başka bir özelliği de hiyerarşiler tanımlanabilmesidir. Hiyerarşiler sayesinde hem toplamlara ulaşmak kolaylaşmakta, hem de farklı gruplar için farklı senaryolar hazırlayabilme şansı doğmaktadır.

Karmaşık hesaplamalar:

Bir OLAP sisteminin gerçek performansı karmaşık hesaplamaları yapma gücü ile ölçülebilir. OLAP sistemleri sadece toplama işleminden başka işlemler de yapabilecek güçte olmalıdırlar. Gerçek hayat her zaman daha karmaşıktır. Analiz yapanlar için asıl rakamlardan çok, yüzdesel dağılımlar çok daha önemlidir. Birkaç yıllık satış içerisinde binlerce ürün türü için günlük bazda satışları yüzdesel olarak analiz edip sıraya dizebilmek bir RDBMS ile saatler sürecek bir raporun çalışmasını gerektirebilir. Oysa uygun bir OLAP sistemi ile bir günlük satışlar ve birkaç yıllık satış rakamı arasında bir fark olmamalıdır. Satış tahminlerinde genellikle “moving average” ve “yüzde artış” gibi trend analizleri kullanılır. Finansal analizlerde, envanter hesaplarında ve portföy performans hesaplarında, zamana göre ürünlerin toplanma sırası sonucu tamamen değiştirebilir. (yukarıdan aşağıya ya da aşağıdan yukarıya, LIFO-FIFO) Kullanılacak OLAP yapısında bu şekilde hesaplamalara da izin verir bir yapının olması gerekir.

Zaman kavramları:

Zaman boyutu neredeyse her analizin temel bileşenidir. Zaman, diğer boyutlardan farklı olarak kendine has bir sıralama içersinde gider. Alfabetik (Ocak her zaman Şubat’tan önce gelmelidir) veya nümerik sıralamalardan (12/31, 01/01’den önce gelmelidir) her zaman farklıdır. Gerçek OLAP sistemleri, zamanın bu şekilde sıralanmasını sağlar.

2.5.3 OLAP’ın Yararları

OLAP’ın yararları şu şekilde özetlenebilir:

- Analiz yapan kişiler kendine daha yeterli, IT’den bağımsız hale gelebilmektedirler.
- Düşük kapasiteli sistemlerde yaşanan zaman sıkıntısı problemleri ortadan

kalkmaktadır. Üretim sistemini rapor için hızlandıracak büyük yatırımlar yerine çok daha düşük maliyetli bir rapor sistemi kurmak bir çözüm olabilir. Yeni dönemde çıkan tümleşik OLAP yapılarında ilişkisel veri tabanı ve OLAP iç içe bir yapıda olduklarından üretim sistemeleri ya da veri ambarları üzerinde toplamlar gerektiğinde ilgili sorgulama OLAP küplerine yönlendirilerek çok yüksek ölçüde performans getirisi sağlanabilmektedir.

- Ayrıca bu yapılar sayesinde OLAP sistemi için hem yazılım hem de güncelleme anlamında ikinci kez masraf yapmak zorluğu da ortadan kalkmaktadır.
- Bu şekilde bir yatırımla var olan IT sistemi de rahatlamakta, üretim sistemi üzerinde yer alan raporlar ortadan kalkmaktadır.
- Farklı kaynaklardan alınan kaynaklar konsolide edilmekte ve veri güvenliği sağlanmaktadır.
- Veriler toplamları alınmış şekilde bulunduklarından toplam verilerin bulunması için gerekli raw-data analistin makinesine aktarılması gerekmediğinden network üzerinde büyük ölçüde bir trafik kazancı sağlanmaktadır.
- Zaman kazancı aynı zamanda kaynakların etkin kullanımı ve para kazancı anlamına da gelmektedir [7].

2.6 OLTP ve OLAP Sistemlerin Kıyaslanması

OLTP ve OLAP sistemleri arasında kullanıcı ve sistem yönelimi, veri içeriği, veritabanı tasarımı, görünüm ve erişim şablonları gibi konularda benzerlikler ve farklılıklar mevcuttur. Aşağıdaki çizelgede bu iki sistemin kıyaslanması özet olarak verilmiştir.

Çizelge 2.1 OLTP ve OLAP sistemlerin kıyaslanması (Han ve Kamber, 2000)

Özellik	OLTP	OLAP
Nitelik	hareketsel işleme	bilgisel işleme
Yönelim	hareket	analiz
Kullanıcı	tezgahtâr, DBA, veritabanı profesyoneli	bilgi çalışanı, yönetici, analist
Fonksiyon	günlük hareketler	uzun terimli bilgi gereksinimleri, karar destek
Veritabanı Tasarımı	E-R tabanlı, uygulamaya dayalı	yıldız/kar tanesi, özneye dayalı
Veri	güncel, güncellik garantisi	tarihsel, kesinlik zamanla sağlanır
Özetleme	ilkel, çok detaylı	özetlenmiş, birleştirilmiş
Görünüm	detaylı, düz ilişki	özetlenmiş, çok boyutlu
Çalışma Birimi	kısa, basit hareket	karmaşık sorgu
Erişim	oku/yaz	çoğunlukla oku
Odaklanma	gelen veri	çıkan bilgi
İşlemler	birincil anahtardaki indeks/hash	çoklu tarama
Erişilen Kayıt Sayısı	onlarca	milyonlarca
Kullanıcı Sayısı	binlerce	yüzlerce
Veritabanı Boyutu	100 MB'tan GB'a kadar	100 GB'tan TB'a kadar
Öncelik	yüksek başarı, yüksek kullanırlık	yüksek esneklik, son-kullanıcı yönetimi
Ölçüm	üretilen hareketsel iş	üretilen sorgusal iş, cevap süresi

3. VERİ MADENCİLİĞİ

İşlenmemiş verinin bilgiye çevrilmesi yeni bir problem değildir. Günümüzde hızla gelişen teknoloji ve yazılımlar sayesinde veriler çok hızlı şekilde depolanmaktadır. Bu depolar günümüzün yüksek kapasiteli donanımları sayesinde büyük verilerin elde edilmesini ve bunların saklanmasını sağlamaktadır. Bu depolanmakta olan verilerden anlamlı bilgi çıkartmak da o denli şekilde önem kazanmaktadır. Karar vermede en önemli ihtiyaç bilgidir. Gerçek zamanlı bir bilgi akışını sağlayabilmek için sürekli akan veri nehrinde, verileri çok hızlı toplayabilmeli, düzenleyebilmeli ve aynı oranda verilere ulaşabilmeli ve çözümleyebilmelidir. Bu anlamlı bilgi dönüşüm işlemi daha sonra stratejik karar verme sürecinde veya yeni bilimsel bulguların oluşturulmasında kullanılabilmektedir.

Binlerce kayıt içinden analizlerin gözle ve elle yapılamayacağı, otomatik olarak yapılması gerektiği ortaya çıkar. Veri madenciliği burada devreye girer:

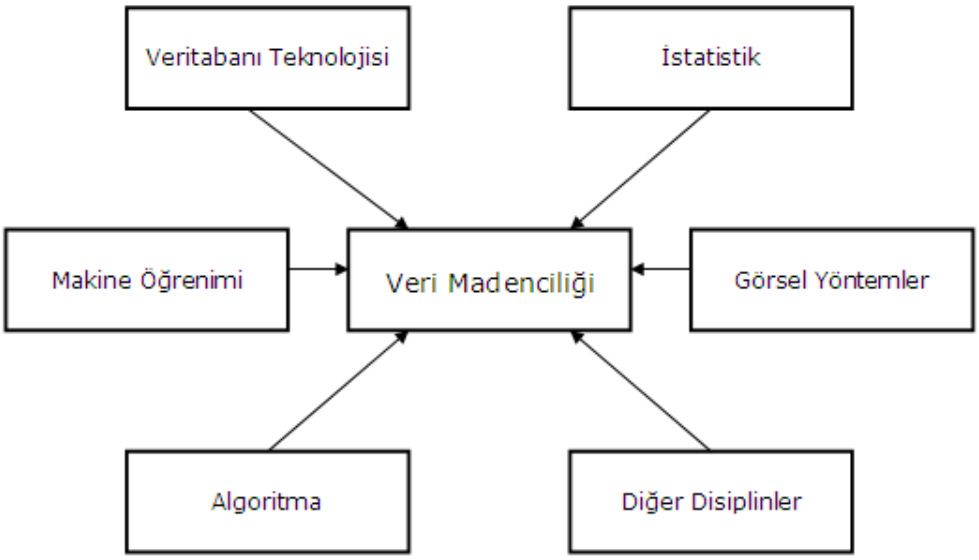
Veri Madenciliği; büyük miktardaki veri içinden gelecekle ilgili tahmin yapılmasını sağlayan bağıntı ve kuralların bilgisayar programları kullanarak bulunmasıdır.

Yakın geleceğin geçmişten çok fazla farklı olmayacağı varsayılırsa, geçmiş veriden çıkarılmış olan kurallar gelecekte de geçerli olacak ve ilerisi için doğru tahmin yapılmasını sağlayacaktır (Alpaydın, 2000).

Bu tanımlamalar doğrultusunda veri madenciliğinin kullanım amaçları şöyle özetlenebilir:

- Veri ambarında depolanmış verilerin içersinde bulunan bilgiyi çıkartma
- Çok büyük miktardaki veriden yeni ve gerekli olan anlamlı bilgileri üretme
- Verinin özelliklerinden yararlanarak eğilimlerini anlama
- Geleceğe yönelik tahminlerde bulunarak bilgiyi gelecekteki müşteri ilişkilerini yönlendirmek amacıyla değerlendirme.

İstatistiğin genel olarak tanımlayıcı ve yorumlayıcı oluşu veri madenciliğinde kümeleme, ilişki kurma, tahmin yürütme ve karşılaştırma amaçları ile kullanılmaktadır. Bu sayede birçok model çıkarılmaktadır. Ancak istatistiğin yanında veri tabanlarının ve bilgi öğrenme metotlarının gelişmesi, yeni algoritmaların geliştirilmesi ile veri madenciliği, birçok alanın kesişmesinin bir ürünü olarak ortaya çıkmaktadır [4].

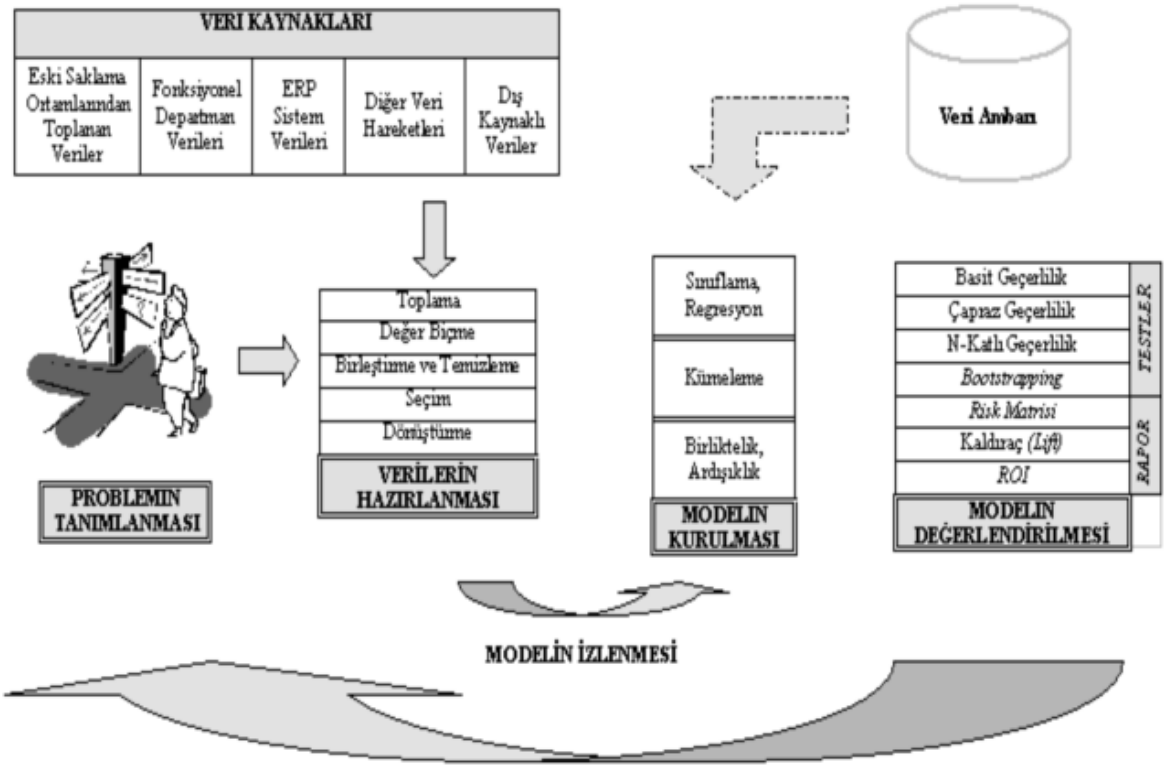


Şekil 3.1 Veri madenciliğinin disiplinler arası ilişkisi

3.1 Veritabanlarında Bilgi Keşfi Süreci

Veri madenciliği, veri ambarlarında tutulan ve ilk bakışta çok net şekilde anlaşılamayan bilgilerin sırlarını ortaya çıkartmak, bir anlamda bilgiyi keşfetmektir. Veri madenciliği matematiksel, istatiksel ve desen tanıma (pattern recognition) yöntemlerinden herhangi birini veya bir kaçını kullanarak büyük bir veri ambarı içerisindeki desenlerin, benzerliklerin ve korelâsyonların tespit edilmesi ve anlamlandırılması işlemidir.

Veritabanı sistemlerinin artan kullanımı ve hacimlerindeki olağanüstü artış, organizasyonları elde toplanan verilerden nasıl faydalanılabileceği problemi ile karşı karşıya bırakmıştır. Geleneksel sorgu (query) veya raporlama araçlarının veri yığınları karşısında yetersiz kalması, *Veritabanlarında Bilgi Keşfi - VTBK* (Knowledge Discovery in Databases - KDD) adı altında, sürekli ve yeni arayışlara neden olmaktadır. Bu süreç içerisinde, modelin kurulması ve değerlendirilmesi aşamalarından meydana gelen veri madenciliği en önemli kesimi oluşturmaktadır. Bu önem, bir çok araştırmacı tarafından VTBK ile veri madenciliği terimlerinin eş anlamlı olarak da kullanılmasına neden olmaktadır.



Şekil 3.2 Veritabanlarında bilgi keşfi süreci (Akpınar, 2000)

VTBK sürecinde izlenmesi gereken temel aşamalar şunlardır;

- Problemin tanımlanması,
- Verilerin hazırlanması,
- Modelin kurulması ve değerlendirilmesi,
- Modelin kullanılması ve
- Modelin izlenmesidir (Akpınar, 2000).

3.1.1 Problemin Tanımlanması

Veri madenciliği çalışmalarında başarılı olmanın ilk şartı, uygulamanın amacının açık bir şekilde tanımlanmasıdır. Amaç, problemin üzerine odaklanmış ve açık bir dille ifade edilmiş olmalı, elde edilecek sonuçların başarı düzeylerinin nasıl ölçüleceği tanımlanmalıdır.

3.1.2 Verilerin Hazırlanması

Modelin kurulması aşamasında ortaya çıkacak sorunlar, bu aşamaya sık sık geri dönülmesine ve verilerin yeniden düzenlenmesine neden olacaktır. Bu durum verilerin hazırlanması ve modelin kurulması aşamaları için, çözümleyicilerin veri keşfi sürecinin toplamı içerisinde

enerji ve zamanının % 50 - % 85'ini harcamasına neden olmaktadır. Veri hazırlamanın bütünüyle amacı veri madenciliği algoritması için girdi olabilecek veri kümesini oluşturabilmektir.

Verilerin hazırlanması aşaması kendi içerisinde toplama, değer biçme, birleştirme ve temizleme, seçme ve dönüştürme adımlarından meydana gelmektedir.

3.1.2.1 Toplama (Collection)

Toplama, tanımlanan problem için gerekli olduğu düşünülen verilerin ve bu verilerin toplanacağı veri kaynaklarının belirlenmesi adımıdır. Verilerin toplanmasında kuruluşun kendi veri kaynaklarının dışında, nüfus sayımı, merkez bankası kara listesi gibi veritabanlarından veya veri pazarlayan kuruluşların veritabanlarından da faydalanılabilir.

3.1.2.2 Değer biçme (Assessment)

Veri madenciliğinde kullanılacak verilerin farklı kaynaklardan toplanması, doğal olarak veri uyumsuzluklarına neden olacaktır. Bu uyumsuzlukların başlıcaları farklı zamanlara ait olmaları, kodlama farklılıkları ve farklı ölçü birimleridir. Bu nedenlerle, iyi sonuç alınacak modeller ancak iyi verilerin üzerine kurulabileceği için, toplanan verilerin ne ölçüde uyumlu oldukları bu adımda incelenerek değerlendirilmelidir.

3.1.2.3 Birleştirme ve temizleme (Consolidation and Cleaning)

Bu adımda farklı kaynaklardan toplanan verilerde bulunan ve değer biçme adımında belirlenen sorunlar mümkün olduğunca giderilerek veriler tek bir veritabanında toplanır. Ancak basit yöntemlerle ve baştan savma olarak yapılacak sorun giderme işlemlerinin, ilerideki aşamalarda daha büyük sorunların kaynağı olacağı unutulmamalıdır.

3.1.2.4 Seçim (Selection)

Bu adımda kurulacak modele bağlı olarak veri seçimi yapılır. Örneğin tahmin edici bir model için, bu adım bağımlı ve bağımsız değişkenlerin ve modelin eğitiminde kullanılacak veri kümesinin seçilmesi anlamını taşımaktadır. Modelde kullanılan veritabanının çok büyük olması durumunda rastgeleliği bozmayacak şekilde örnekleme yapılması uygun olabilir. Günümüzde hesaplama olanakları ne kadar gelişmiş olursa olsun, çok büyük veritabanları üzerinde çok sayıda modelin denenmesi çok uzun zaman alması nedeni ile mümkün olamamaktadır. Bu nedenle tüm veritabanını kullanarak birkaç model denemek yerine, rastgele örneklenmiş bir veritabanı parçası üzerinde birçok modelin denenmesi ve bunlar arasından en güvenilir ve güçlü modelin seçilmesi daha uygun olacaktır.

3.1.2.5 Dönüştürme (Transformation)

Çözümleme için kullanılması düşünülen verilere ilişkin değişkenlerin uygun şekle dönüştürülmesi gereklidir. Örneğin, kredi riskinin tahmini için geliştirilen bir modelde, borç/gelir gibi önceden hesaplanmış bir oran yerine, ayrı ayrı borç ve gelir verilerinin kullanılması tercih edilebilir. Ayrıca modelde kullanılan algoritma, verilerin gösteriminde önemli rol oynayacaktır. Örneğin bir uygulamada bir yapay sinir ağı algoritmasının kullanılması durumunda kategorik değişken değerlerinin evet/hayır olması, bir karar ağacı algoritmasının kullanılması durumunda ise örneğin gelir değişken değerlerinin yüksek/orta/düşük olarak gruplanmış olması modelin etkinliğini artıracaktır.

3.1.3 Modelin Kurulması ve Değerlendirilmesi

Bu adım; verilerin çözümlendiği, VTBK sürecinin en önemli aşaması olan veri madenciliği adımıdır.

Veri madenciliği; veritabanı sistemleri, verilerin depolanması, istatistik, makine öğrenimi gibi alanların kombinasyonundan oluşan disiplinler arası bir yöntemdir. Veri madenciliği, istatistik, veritabanı teknolojisi ve makine öğrenimi gibi diğer alanlara ait fikirleri, araçları ve yöntemleri de kullanır.

3.1.4 Modelin Kullanılması

Kurulan ve geçerliliği kabul edilen model doğrudan bir uygulama olabileceği gibi, bir başka uygulamanın alt parçası olarak da kullanılabilir.

3.1.5 Modelin İzlenmesi

Zaman içerisinde sistemlerin özelliklerinde ve dolayısıyla ürettikleri verilerde ortaya çıkan değişiklikler, kurulan modellerin sürekli olarak izlenmesini ve gerekiyorsa yeniden düzenlenmesini gerektirecektir (Akpınar, 2000).

3.2 Veri Madenciliği Uygulamaları

Bağıntı: “Çocuk bezi alan müşterilerin %30’u bira da satın alır.”

Market sepet analizinde (market basket analysis) müşterilerin beraber satın aldığı malların analizi yapılır. Buradaki amaç mallar arasındaki pozitif veya negatif korelasyonları bulmaktır. Çocuk bezi alan müşterilerin mama da satın alacağını veya bira satın alanların cips de alacağını tahmin edebiliriz ama ancak otomatik bir analiz bütün olasılıkları gözönüne alır ve kolay düşünülemeyecek, örneğin çocuk bezi ve bira arasındaki bağıntıları da bulur.

Sınıflandırma: "Genç kadınlar küçük araba satın alır, yaşlı, zengin erkekler büyük, lüks araba satın alır."

Amaç bir malın özellikleri ile müşteri özelliklerini eşlemektir. Böylece bir müşteri için ideal ürün veya bir ürün için ideal müşteri profili çıkarılabilir. Örneğin bir otomobil satıcısı şirket geçmiş müşteri hareketlerinin analizi ile yukarıdaki gibi iki kural bulursa genç kadınların okuduğu bir dergiye reklam verirken küçük modelinin reklamını verir.

Regresyon: "Ev sahibi olan, evli, aynı iş yerinde beş yıldan fazladır çalışan, geçmiş kredilerinde geç ödemesi bir ayı geçmemiş bir erkeğin kredi skoru 825'dir."

Başvuru skorlamada (application scoring) bir finans kurumuna kredi için başvuran kişi ile ilgili finansal güvenilirliğini notlayan örneğin 0 ile 1000 arasında bir skor hesaplanır. Bu skor kişinin özellikleri ve geçmiş kredi hareketlerine dayanılarak hesaplanır.

Zaman İçinde Sıralı Örüntüler: "İlk üç taksidinden iki veya daha fazlasını geç ödemiş olan müşteriler %60 olasılıkla kanuni takibe gidiyor."

Davranış skoru (behavioral score), başvuru skorundan farklı olarak kredi almış ve taksitleri ödeyen bir kişinin sonraki taksitlerini ödeme/geciktirme davranışını notlamayı amaçlar.

Benzer Zaman Sıraları: "X şirketinin hisse fiyatları ile Y şirketinin hisse fiyatları benzer hareket ediyor."

Amaç zaman içindeki iki hareket serisi arasında bağıntı kurmaktır. Bunlar örneğin iki malın zaman içindeki satış miktarları olabilir. Örneğin dondurma satışları ile kola satışları arasında pozitif, dondurma satışları ile sahlep satışları arasında negatif bir bağıntı beklenebilir.

İstisnalar (Fark Saptanması): "Normalden farklı davranış gösteren müşterilerim var mı?"

Amaç önceki uygulamaların aksine kural bulmak değil, kurala uymayan istisnai hareketleri bulmaktır. Bu da örneğin olası sahtekarlıkların saptanmasını (fraud detection) sağlar. Örneğin Visa kredi kartı için yapılan CRIS sisteminde bir yapay sinir ağı kredi kartı hareketlerini takip ederek müşterinin normal davranışına uymayan hareketler için müşterinin bankası ile temasa geçerek müşteri onayı istenmesini sağlar.

Döküman Madenciliği: "Arşivimde (veya internet üzerinde) bu dökümana benzer hangi dökümanlar var?"

Amaç dökümanlar arasında ayrıca elle bir tasnif gerekmeden benzerlik hesaplayabilmektir (text mining). Bu genelde otomatik olarak çıkarılan anahtar sözcüklerin tekrar sayısı sayesinde yapılır (Alpaydın, 2000).

3.3 Veri Madenciliği Yöntemleri

İstatistiksel Yöntemler: Veri madenciliği çalışması esas olarak bir istatistik uygulamasıdır. Verilen bir örnek kümesine bir kestirici oturtmayı amaçlar. İstatistik literatüründe son elli yılda bu amaç için değişik teknikler önerilmiştir. Bu teknikler istatistik literatüründe çokboyutlu analiz (multivariate analysis) başlığı altında toplanır ve genelde verinin parametrik bir modelden (çoğunlukla çokboyutlu bir Gauss dağılımından) geldiğini varsayar. Bu varsayım altında sınıflandırma (classification; discriminant analysis), regresyon, öbekleme (clustering), boyut azaltma (dimensionality reduction), hipotez testi, varyans analizi, bağıntı (association; dependency) kurma için teknikler istatistikte uzun yıllardır kullanılmaktadır.

Bellek Tabanlı Yöntemler: Bellek tabanlı veya örnek tabanlı bu yöntemler (memory-based, instance-based methods; case-based reasoning) istatistikte 1950'li yıllarda önerilmiş olmasına rağmen o yıllarda gerektirdiği hesaplama ve bellek yüzünden kullanılamamış ama günümüzde bilgisayarların ucuzlaması ve kapasitelerinin artmasıyla, özellikle de çok işlemcili sistemlerin yaygınlaşmasıyla, kullanılabilir olmuştur. Bu yönteme en iyi örnek en yakın k komşu algoritmasıdır (k-nearest neighbor).

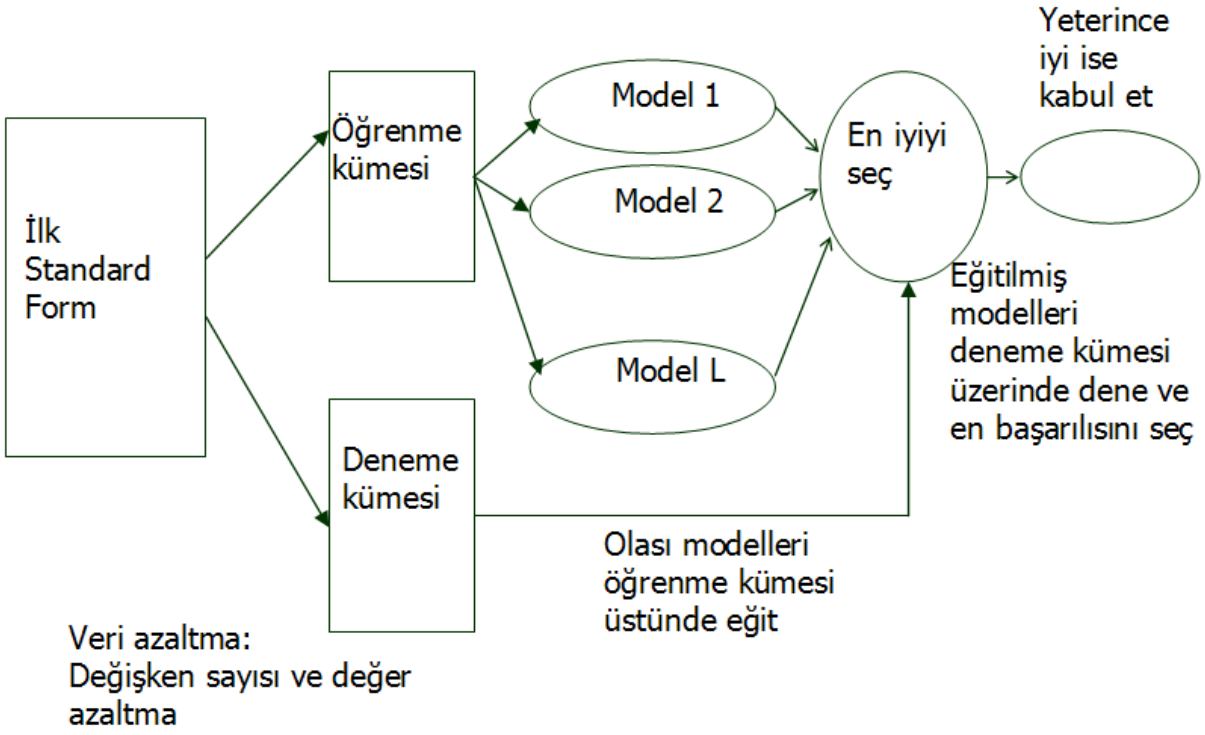
Yapay Sinir Ağları: 1980'lerden sonra yaygınlaşan yapay sinir ağlarında (artificial neural networks) amaç fonksiyon birbirine bağlı basit işlemci ünitelerinden oluşan bir ağ üzerine dağıtılmıştır. Yapay sinir ağlarında kullanılan öğrenme algoritmaları veriden üniteler arasındaki bağlantı ağırlıklarını hesaplar. YSA istatistiksel yöntemler gibi veri hakkında parametrik bir model varsaymaz yani uygulama alanı daha geniştir, ve bellek tabanlı yöntemler kadar yüksek işlem ve bellek gerektirmez.

Karar Ağaçları: İstatistiksel yöntemlerde veya yapay sinir ağlarında veriden bir fonksiyon öğrenildikten sonra bu fonksiyonun insanlar tarafından anlaşılabilecek bir kural olarak yorumlanması zordur. Karar ağaçları ise veriden oluşturulduktan sonra yukarıdaki örnekte de olduğu gibi ağaç kökten yaprağa doğru inilerek kurallar (IF-THEN rules) yazılabilir. Bu şekilde kural çıkarma (rule extraction), veri madenciliği çalışmasının sonucunun geçerlenmesini sağlar. Bu kurallar uygulama konusunda uzman bir kişiye gösterilerek sonucun anlamlı olup olmadığı denetlenebilir. Sonradan başka bir teknik kullanılacak bile olsa karar ağacı ile önce bir kısa çalışma yapmak, önemli değişkenler ve yaklaşık kurallar konusunda bize bilgi verir ve tavsiye edilir (Alpaydın, 2000).

3.4 Veri Madenciliği Metodolojisi

Bir veri madenciliği çalışmasında kullanılan metodoloji Şekil 3.3'te verilmiştir. Standart form içinde verilen veri, öğrenme ve deneme olmak üzere ikiye ayırılır. Her uygulamada kullanılabilecek birden çok teknik vardır ve önceden hangisinin en başarılı olacağını kestirmek olası değildir. Bu yüzden öğrenme kümesi üzerinde L değişik teknik kullanılarak L tane model oluşturulur. Sonra bu L model deneme kümesi üzerinde denenerek en başarılı olanı, yani deneme kümesi üzerindeki tahmin başarısı en yüksek olanı seçilir.

Eğer bu en iyi model yeterince başarılıysa kullanılır, aksi takdirde başa dönerek çalışma tekrarlanır. Tekrar sırasında başarısız olan örnekler incelenerek bunlar üzerindeki başarının nasıl arttırılabileceği araştırılır. Örneğin standart forma yeni alanlar ekleyerek programa verilen bilgi arttırılabilir; veya olan bilgi değişik bir şekilde kodlanabilir; veya amaç daha değişik bir şekilde tanımlanabilir.



Şekil 3.3 Veri madenciliği metodolojisi (Alpaydın, 2000)

3.5 Veri Madenciliği Modelleri

Veri madenciliğinde kullanılan modeller, tahmin edici (predictive) ve tanımlayıcı (descriptive) olmak üzere iki ana başlık altında incelenmektedir.

Tahmin edici modellerde, sonuçları bilinen verilerden hareket edilerek bir model geliştirilmesi ve kurulan bu modelden yararlanılarak sonuçları bilinmeyen veri kümeleri için sonuç

değerlerin tahmin edilmesi amaçlanmaktadır. Örneğin bir banka önceki dönemlerde vermiş olduğu kredilere ilişkin gerekli tüm verilere sahip olabilir. Bu verilerde bağımsız değişkenler kredi alan müşterinin özellikleri, bağımlı değişken değeri ise kredinin geri ödenip ödenmediğidir. Bu verilere uygun olarak kurulan model, daha sonraki kredi taleplerinde müşteri özelliklerine göre verilecek olan kredinin geri ödenip ödenmeyeceğinin tahmininde kullanılmaktadır.

Tanımlayıcı modellerde ise karar vermeye rehberlik etmede kullanılabilecek mevcut verilerdeki örüntülerin tanımlanması sağlanmaktadır. X/Y aralığında geliri ve iki veya daha fazla arabası olan çocuklu aileler ile, çocuğu olmayan ve geliri X/Y aralığından düşük olan ailelerin satın alma örüntülerinin birbirlerine benzerlik gösterdiğinin belirlenmesi tanımlayıcı modellere bir örnektir.

Veri madenciliği modellerini gördükleri işlevlere göre:

- Sınıflama (Classification) ve Regresyon (Regression),
- Kümeleme (Clustering),
- Birliktelik Kuralları (Association Rules) ve Ardışık Zamanlı Örüntüler (Sequential Patterns),

olmak üzere üç ana başlık altında incelemek mümkündür. Sınıflama ve regresyon modelleri tahmin edici, kümeleme, birliktelik kuralları ve ardışık zamanlı örüntü modelleri tanımlayıcı modellerdir.

3.5.1 Sınıflama ve Regresyon Modelleri

Mevcut verilerden hareket ederek geleceğin tahmin edilmesinde faydalanılan ve veri madenciliği teknikleri içerisinde en yaygın kullanıma sahip olan sınıflama ve regresyon modelleri arasındaki temel fark, tahmin edilen bağımlı değişkenin kategorik veya süreklilik gösteren bir değere sahip olmasıdır. Ancak çok terimli lojistik regresyon (multinomial logistic regression) gibi kategorik değerlerin de tahmin edilmesine olanak sağlayan tekniklerle, her iki model giderek birbirine yaklaşmakta ve bunun bir sonucu olarak aynı tekniklerden yararlanılması mümkün olmaktadır. Sınıflama ve regresyon modellerinde kullanılan başlıca teknikler:

- Karar Ağaçları (Decision Trees),
- Yapay Sinir Ağları (Artificial Neural Networks),
- Genetik Algoritmalar (Genetic Algorithms),

- K-En Yakın Komşu (K-Nearest Neighbor),
- Bellek Temelli Nedenleme (Memory Based Reasoning),
- Naïve-Bayes,
- Lojistik Regresyondur (Logistic Regression).

3.5.2 Kümeleme Modelleri

Kümeleme modellerinde amaç, küme üyelerinin birbirlerine çok benzediği, ancak özellikleri birbirlerinden çok farklı olan kümelerin bulunması ve veritabanındaki kayıtların bu farklı kümelere bölünmesidir. Başlangıç aşamasında veritabanındaki kayıtların hangi kümelere ayrılacağı veya kümelemenin hangi değişken özelliklerine göre yapılacağı bilinmemekte, konunun uzmanı olan bir kişi tarafından kümelerin neler olacağı tahmin edilmektedir.

3.5.3 Birliktelik Kuralları ve Ardışık Zamanlı Örüntüler

Bir alışveriş sırasında veya birbirini izleyen alışverişlerde müşterinin hangi mal veya hizmetleri satın almaya eğilimli olduğunun belirlenmesi, müşteriye daha fazla ürünün satılmasını sağlama yollarından biridir. Satın alma eğilimlerinin tanımlanmasını sağlayan birliktelik kuralları ve ardışık zamanlı örüntüler, pazarlama amaçlı olarak pazar sepeti analizi (Market Basket Analysis) adı altında veri madenciliğinde yaygın olarak kullanılmaktadır. Bununla birlikte bu teknikler, tıp, finans ve farklı olayların birbirleri ile ilişkili olduğunun belirlenmesi sonucunda değerli bilgi kazanımının söz konusu olduğu ortamlarda da önem taşımaktadır.

Birliktelik kuralları aşağıda sunulan örneklerde görüldüğü gibi eş zamanlı olarak gerçekleşen ilişkilerin tanımlanmasında kullanılır.

- Müşteriler bira satın aldığında, % 75 ihtimalle patates cipsi de alırlar,
- Düşük yağlı peynir ve yağsız yoğurt alan müşteriler, %85 ihtimalle diet süt de satın alırlar.
- Ardışık zamanlı örüntüler ise aşağıda sunulan örneklerde görüldüğü gibi birbirleri ile ilişkisi olan ancak birbirini izleyen dönemlerde gerçekleşen ilişkilerin tanımlanmasında kullanılır.
- X ameliyatı yapıldığında, 15 gün içinde % 45 ihtimalle Y enfeksiyonu oluşacaktır,
- İMKB endeksi düşerken A hisse senedinin değeri % 15'den daha fazla artacak olursa, üç iş günü içerisinde B hisse senedinin değeri % 60 ihtimalle artacaktır,

- Çekiç satın alan bir müşteri, ilk üç ay içerisinde % 15, bu dönemi izleyen üç ay içerisinde % 10 ihtimalle çivi satın alacaktır (Akpınar, 2000).

4. MARKET SEPET ANALİZİ ve BİRLİKTELİK KURALLARI

4.1 Market Sepet Analizi

Geçmiş tarihli hareketleri çözümlemek, karar destek sistemlerinde verilen kararın kalitesini artırmak için izlenen bir yaklaşımdır. 90'lı yılların başına değin teknik yetersizlikten dolayı, kurumlara veya müşterilere satış yapıldığı anda değil, belirli bir zaman aralığında (günlük, haftalık, aylık, yıllık) gerçekleşen satış hareketlerinin tamamına ilişkin genel veriler elektronik ortamda tutulmaktaydı. Barkod uygulamalarındaki gelişme ile, bir harekete ait verilerin satış hareketi oluştuğu anda toplanması ve elektronik ortama aktarılması olanaklı hale gelmiştir. Genellikle süpermarketlerin satış noktalarında bu tür veriler toplandığından, toplanan bu veriye market sepeti verisi adı verilmiştir. Market sepeti verisinde yer alan bir kayıtta, tekil olan hareket numarası, hareket tarihi ve satın alınan ürünlere ilişkin ürün kodu, miktarı, fiyatı gibi bilgiler yer almaktadır (Han ve Kamber, 2000).

*Market sepet analizi*nde (market basket analysis) amaç, satışlar arasındaki ilişkileri bulmak ve buna bağlı kuralları çıkarmaktır. Bu ilişkilerin bilinmesi, şirketin kârını arttırmak için kullanılabilir. Eğer X ürününü alanların Y ürününü de çok yüksek olasılıkla aldıkları biliniyorsa ve eğer bir müşteri X ürününü alıyor ama Y ürününü almıyorsa, o potansiyel bir Y müşterisidir denilebilir.

Buna benzer veri analizleri yaparak her ürün için bir sonraki ayın satış tahminleri çıkarılabilir, birlikte satın alınan ürünler için promosyon uygulaması ve reyon dizilişleri yapılabilir, müşteriler satın aldıkları ürünlere göre gruplandırılabilir, yeni bir ürün için potansiyel müşteriler belirlenebilir (Alpaydın, 2000).

4.2 Birliktelik Kuralları

Birliktelik kuralları (association rules), veri madenciliği alanında üzerinde çok fazla araştırma ve çalışma yapılmış olan ilgi çekici bir konudur. Birliktelik kuralları, aynı işlem içinde çoğunlukla beraber görülen nesneleri içeren kurallardır.

Birliktelik kurallarının kullanıldığı en tipik örnek market sepeti uygulamasıdır. Bu işlem, müşterilerin yaptıkları alışverişlerdeki ürünler arasındaki birliktelikleri bularak müşterilerin satın alma alışkanlıklarını çözümler. Bu tip birlikteliklerin keşfedilmesi, müşterilerin hangi ürünleri bir arada aldıkları bilgisini ortaya çıkarır ve market yöneticileri de bu bilgi ışığında raf düzenlerini belirleyerek satış oranlarını artırabilir ve etkili satış stratejileri geliştirebilirler. Market sepeti çözümlemesinin son zamanlarda çok büyük ilgi ile karşılaşmasının sebebi

kullanım kolaylığı ve anlaşılabilirliğidir.

Market sepet analizi ile birliktelik kuralları çıkarımı ilk olarak Agrawal ve diğerleri tarafından 1993 yılında ele alınmıştır. Çalışmada, X ve Y'nin nesnekümesi oluşu $X \Rightarrow Y$ (X birliktelik Y) şeklinde ifade edilmiş olup, birliktelik kurallarının matematiksel şekli belirlenmiştir. Kuralları oluşturabilmek için *destek* (support) ve *güven* (confidence) değerlerini kullanarak, kullanıcı tarafından belirlenmiş minimum destek ve minimum güven değerlerinden yaygın birlikteliklerin belirlenmesi amaçlanmıştır. Market sepet analizinde, nesneler müşteriler tarafından satın alınan ürünlerdir ve bir hareket (kayıt) birçok nesneyi içinde bulunduran tek bir satın almadır.

Birliktelik kurallarının kullanışlı olması için hem konu ile ilgili hem de anlaşılabilir olması gerekir. Birliktelik kuralları simgesel ve sezgisel yapıda olduğundan anlaşılabilirlik her zaman birliktelik kurallarının güçlü yönü olmuştur. Birliktelik kurallarında, kullanıcının kuralların tipini ve sayısını kontrol edebileceği çeşitli yollar vardır. En yaygın olarak kullanılan yöntem, eşik değerleri olarak bilinen minimum destek ve minimum güven değerlerinin belirlendiği yöntemdir. Bu yöntemde sadece kullanıcı tarafından belirlenen eşik değerlerinden büyük olan destek ve güven değerlerine sahip kurallar bulunur ve kullanılır. Diğer bir yöntemde kullanıcının sınırlanmış nesne tanımlamasıdır. Sınırlanmış nesne, kuralların içeriğinin sınırlanmasında kullanılan mantıksal bir ifadedir. Örneğin sınırlanmış nesne cips, kola ve hamburger olsun. Sadece cips, kola ve hamburger içeren kurallar ile ilgilenilir. Srikant ve diğerleri sınırlanmış nesne ile kurallar için çesitli etkin çözümleme yöntemleri geliştirmişlerdir.

Birliktelik kurallarındaki bir nesnenin ve bir işlemin tanımı uygulamaya bağlıdır. Market sepeti analizinde; nesneler, müsterilerin aldığı ürünler ve işlem, beraber alınan bütün nesnelerin kümesidir. Birliktelik kurallarında sıklıkla kullanılan birkaç önemli terim vardır. Bunlar; kuralın sol tarafını ifade eden önce (antecedent), kuralın sağ tarafını ifade eden sonuç (consequent), destek değeri, güven değeri, min_destek olarak gösterilen minimum destek değeri, min_güven olarak gösterilen minimum güven değeri, nesneküme, yaygın nesnekümesi ve aday nesnekümesidir (Dolgun, 2006).

Birliktelik kuralı madenciliği iki aşamalıdır:

- **Tüm yaygın nesnekümelerinin bulunması**: Her nesnekümesinin yaygın nesnekümesi olarak yer alabilmesi için, her nesnenin destek değerinin önceden tanımlanmış olan min_destek değerinden büyük olması gerekir.
- **Yaygın nesnekümelerinden güçlü birliktelik kurallarının elde edilmesi**: Bu

kurallar min_destek ve min_güven durumunu sağlamalıdır [10].

Birliktelik kuralı algoritmalarının performansını belirleyen adım birinci adımdır. Yaygın nesnekümeleri belirlendikten sonra, birliktelik kurallarının bulunması sıradan bir adımdır.

4.2.1 Birliktelik Kuralları Temel Kavramları

Birliktelik kuralının matematiksel modeli 1993 yılında Agrawal, Imielinski ve Swami tarafından ifade edilmiştir. Bu modele göre; $I = \{i_1, i_2, ..., i_m\}$ nesnelerin kümesi ve *D* işlemler kümesi olarak ifade edilir. Her *i*, bir nesne (ürün) olarak adlandırılır. *D* veritabanında her hareket (transaction) *T*, $T \subseteq I$ olacak şekilde tanımlanan nesnelerin kümesi (nesneküme) olsun. Her hareket bir tanımlayıcı alan olan TID ile temsil edilir. *A* ve *B* nesnelerin kümeleri olsun. Bir *T* işlemler kümesi ancak ve ancak $A \subseteq T$ ise yani *A*, *T*'nin alt kümesi ise *A*'yı kapsıyor denir. Bir birliktelik kuralı $A \Rightarrow B$ formunda ifade edilir. *A* önce ve *B* sonuç olarak adlandırılır. Burada, $A \subset I$, $B \subset I$ ve $A \cap B = \varnothing$ dir.

İlk olarak, $A \Rightarrow B$ kuralı için *d* olasılığı ile kuralın destek değeri tanımlanır. Destek, *T* işleminin $A \cup B$'yi içerme olasılığıdır. İkinci olarak, $A \Rightarrow B$ kuralının *g* ile gösterilen güven değeri tanımlanır. Bu olasılık, *T* işleminin *A*'yı ve aynı zamanda *B*'yi içermesidir. Matematiksel ifade ile kuralın destek ve güven değerleri;

$$Destek(A \Rightarrow B) = P(A \cup B) \tag{4.1}$$

$$Güven(A \Rightarrow B) = P(B/A) \text{ veya} \tag{4.2}$$

$$Güven(A \Rightarrow B) = Destek(A \Rightarrow B)/Destek(A) \tag{4.3}$$

şeklinde ifade edilir. Burada Destek (*A*) = Destek ($A \Rightarrow A$)' dır.

Başka bir ifade ile destek ve güven değerleri;

$$Destek(A) = |A|/|D| \tag{4.4}$$

$$Destek(A \Rightarrow B) = |A.B|/|D| \tag{4.5}$$

$$Güven(A \Rightarrow B) = Destek(A \Rightarrow B)/Destek(A) \tag{4.6}$$

olarak tanımlanır.

Burada; $|A|$; incelenen kayıtlardaki *A* ürününü içeren işlemlerin sayısını, $|A.B|$; incelenen kayıtlardaki *A* ve *B* ürünlerini birlikte içeren işlemlerin sayısını ve $|D|$; veritabanındaki bütün işlemlerin sayısını ifade etmektedir. Kuralın destek ve güven değerleri, kuralın ilginçliğini

ifade eden iki ölçüdür. Bu değerler sırasıyla keşfedilen kuralların yararlılığını (kullanışlılığını) ve kesinliğini (doğruluğunu) ifade eder (Han ve Kamber, 2000). Destek değeri, *A* ve *B* nesnelerinin birlikte bulunma olasılıklarını ifade eder. Güven değeri ise, *A*'yı içeren kayıtların *B*'yi de içereceğini ifade eder (Han ve Kamber, 2000).

Yüksek güven ve destek değerine sahip kurallara güçlü (strong) kurallar adı verilir (Agrawal ve diğerleri, 1993). Kullanıcı tarafından minimum destek eşik değeri (min_destek) ve minimum güven eşik değeri (min_güven) belirlenir. Bu belirlenen eşik değerlerini aşan birliktelik kuralları dikkate alınır ve ilginç olarak ifade edilir. İlginç bir örüntü, bilgi (knowledge) olarak ifade edilir. Genel olarak bu değerler 0 - 1 aralığından çok 0% - 100% aralığında ifade edilmektedir (Han ve Kamber, 2000). Verilen bir D işlemler kümesinde Birliktelik Kurallarının amacı , kullanıcı tarafından belirlenen minimum destek ve minimum güven değerinden büyük $A \Rightarrow B$ kurallarının bulunmasıdır.

Örneğin bir *A* ürününü satın alan müşteriler aynı zamanda *B* ürününü de satın alıyorlarsa, bu durum aşağıdaki birliktelik kuralı ile gösterilir.

$A \Rightarrow B$ [destek = 2%, güven = 60%] (4.7)

Buradaki destek ve güven ifadeleri, kuralın ilginçlik ölçüleridir. Eşitlik 4.7'deki birliktelik kuralı için destek ve güven değerleri şu şekilde yorumlanır. Çözümlenen bütün alışverişlerden 2%'sinde *A* ile *B* ürünlerinin birlikte alındığı (veya *A* ve *B*'nin toplam fiş hareketlerinde birlikte bulunması olasılığı 2%'dir) ve *A* ürününü satın alan müşterilerin 60%'ında aynı alışverişte *B* ürününü de satın aldığı (veya *A* alan bir müşterinin 60% olasılıkla *B* de aldığıdır) söylenir (Han ve Kamber, 2000). Nesnelerin kümesi, nesneküme (Veri madenciliği literatüründe nesneküme (itemset), nesne küme (item set) kullanımından daha sık yer almakta ve kullanılmaktadır) olarak ifade edilmektedir. Eğer bir küme *k* tane nesne içeriyorsa bu küme "*k*-nesneküme" olarak ifade edilir. Eğer bir nesneküme minimum destek değerini sağlıyor ise bu nesneküme yaygın (large, frequent ifadeleri aynı amaçla kullanılmaktadır, large kavram karmaşasına neden olduğundan frequent, yaygın tercih edilmektedir) nesneküme olarak adlandırılır. k-nesnekümenin yaygın kümeleri L_k şeklinde ifade edilmektedir (Han ve Kamber, 2000).

Minimum güven ve destek değerlerini sağlayan birliktelik kuralları çıkarım problemi iki adıma bölünmüştür (Agrawal ve Srikant, 1994);

1. Yaygın geçen nesnekümeler bulunur: Kullanıcı tarafından belirlenmiş olan minimum destek eşik değerini sağlayan nesnekümelere yaygın nesneküme adı verilmektedir. Bu

adımda yaygın nesnekümeleri bulan etkili yöntemler kullanılmalıdır.

2. Yaygın nesnekümelerden güçlü birliktelik kuralları oluşturulur: yaygın nesnekümeleri kullanarak minimum güven eşik değerini sağlayan birliktelik kurallarının bulunmasıdır. Bu adımdaki işlem oldukça basittir. Minimum güven eşik değerine göre taranarak bulunan birliktelik kuralları kullanıcının ilgilendiği ve potansiyel olarak önemli bilgiyi içeren kurallardır. Birliktelik kuralı algoritmasının performansını belirleyen adım birinci adımdır. Yaygın nesnekümeleri belirlendikten sonra, birliktelik kurallarının bulunması kolay bir adımdır (Han ve Kamber, 2000).

Sepet analizinin başarılı olduğu noktalar;

- Kolay ve anlaşılır sonuçlar üretir,
- Değişik boyutlardaki veriler üzerinde çalışabilir,
- Her ne kadar kayıtların sayısı ve kombinasyon seçimine göre işlem adedi artsa da sepet analizi için her adımda gerekli olan hesaplamalar diğer yöntemlere göre (genetik algoritmalar, yapay sinir ağları vb.) çok daha basittir.

Sepet analizinin başarısız olduğu noktalar;

- Sorunun boyutu büyüdükçe, gerekli hesaplamalar üstel olarak artmaktadır,
- Kayıtlarda çok az rastlanan ürünleri yoksayar. Sepet çözümlemesi yönteminin en doğru sonucu, tüm ürünlerin kayıtlar içinde yaklaşık aynı frekansta göründüğü durumlarda üretmektedir,
- Destek ve güven eşik değerleri üretilen kural sayısında sınırlama getirirler fakat eşik değerlerinin çok düşük belirlendiği durumda kullanıcı gerçekten ilgilendiği kuralları kaybetme tehlikesi ile karşı karşıya kalır.
- Birliktelik kurallarının keşfi katalog tasarımı, müşterilerin satın alma alışkanlıklarının belirlenmesi ve sınıflandırılması, mağaza ürün yerleşim planı gibi birçok uygulama alanında kullanılabilir. Sepet analizi yöntemi, birliktelik kurallarının uygulama alanındaki türlerinden birisidir.

4.2.2 Birliktelik Kuralları Çeşitleri

Birliktelik kurallarının birçok türü vardır. Birliktelik kuralları aşağıdaki kriterleri taşıyan çok değişik yollar ile sınıflandırılabilir.

- **Kuralda kullanılan değerlerin tiplerine göre:** Eğer bir kural nesnelerin varlığı ve yokluğu arasındaki birliktelikler ile ilgili ise, bu duruma mantıksal birliktelik kuralıdır. Örneğin 4.7 deki kural böyle bir kuraldır. Bu kurallar sepet analizinden elde edilirler. Kurallar nicel nesneler ya da özellikler arasındaki birliktelikleri tanımlıyor ise nicel birliktelik kuralıdır. Bu kurallarda, nesneler için nicel değerler ya da özellikler aralıklara bölünmüştür.
- **Kuralın içerdiği verinin boyutlarına göre:** Bir birliktelik kuralındaki özellikler (attribute) ya da nesneler sadece bir boyutu temsil ediyorlarsa, o zaman kurala tek boyutlu birliktelik kuralıdır denir.
- **Birliktelik kurallarının çeşitli boyutlarına göre:** Birliktelik kuralları çözümlemesi, korelasyon çözümlemesinin genişletilmişi olabilir. Aynı zamanda, "maxpattern" ve "frequent closed itemset" çözümlemelerinin genişletilmişi de olabilir. Bu iki yöntem, çözümleme sırasında oluşturulan yaygın nesnekümelerin sayısını azaltmak için kullanılmaktadır (Dolgun, 2006).

4.3 Birliktelik Kurallarının Belirlenmesinde Kullanılan Temel Algoritmalar

Yaygın nesnekümelerini (large itemsets) belirlerken birliktelik kuralları oluşturmak için kullanılan algoritmalar, sıralı (sequential) ve paralel olarak sınıflandırılabilir. Birçok durumda, nesne adına bağlı olarak nesnekümelerinin sözlük (lexicographic) sıralı olarak tanımlandığı ve depolandığı varsayılır. Bu sıralama, nesnekümelerinin üretilmesi ve sayılması sırasında kolaylık sağlayan bir tarz oluşturmaktadır ve sıralı algoritmalarda rastlanan olağan bir yaklaşımdır. Diğer yandan, paralel algoritmalar yaygın nesnekümelerinin bulunması işleminin paralelleştirilmesi üzerinde odaklanır.

4.3.1 Sıralı Algoritmalar

4.3.1.1 AIS Algoritması

AIS algoritması, Agrawal, Imielinski ve Swami tarafından 1993 yılında veritabanındaki tüm yaygın nesnekümelerini oluşturmak için geliştirilmiş ve yayınlanmış ilk algoritmadır. Karar destek sorgulamaları yapmak için veri tabanlarının işlevlerini artırmaya odaklanmıştır. Bu algoritmanın hedefi nitelikli kuralları bulmaktır. Bu teknik, sonuçta sadece bir nesne ile kısıtlanmıştır. Birliktelik kuralları $X \Rightarrow I_j \mid \alpha$ biçiminde tanımlanır. Burada X nesneler kümesi, I_j, I alanındaki tek bir nesne ve α de kuralın güven değeridir.

AIS algoritması, veritabanının üzerinden çoklu geçişler yapar. Her geçiş esnasında,

veritabanını tarar. İlk geçişinde tek nesnelerin desteğini sayar ve veri tabanında bunların hangi sıklıkta olduğunu belirler. Her bir geçişin yaygın nesnekümeleri aday nesnekümeleri üretmek için genişletilir. Bir tarama yapıldıktan sonra önceki taramadaki yaygın nesnekümeleri ile taranan nesnelerin arasındaki ortak nesnekümeleri belirlenir. Bu ortak nesnekümeleri yeni aday nesnekümeleri üretmek için veritabanındaki diğer nesneler ile genişletilir. Yaygın bir nesnekümesi olan *l*, yaygın ve sözlük sıralı *l*'deki her bir nesneden sonra gelen nesneler ile genişletilir. Bu işlemleri verimlice yapmak için AIS algoritması tahmin edici araçlar ve budama tekniği kullanır. Bu teknikler aday gruplarını, gereksiz nesneleri aday listesinden atarak belirlerler. Sonra her bir adayın grubu hesaplanır. Minimum destekten büyük veya minimum desteğe eşit olan aday grupları yaygın nesnekümeleri olarak seçilir. Yaygın nesnekümeleri sıradaki geçişte aday üretmek için genişletilirler. Bu işlemler yaygın nesnekümeleri bulunamayana kadar devam eder [5].

4.3.1.2 SETM Algoritması

SETM algoritması, 1995 yılında Houtsmal tarafından önerilmiş olup yaygın nesnekümelerinin hesaplanmasında SQL kullanılmasını baz almaktadır. Bu algoritmada yaygın nesnekümelerinin her bir üyesi, $\overline{L}_k$, TID birincil anahtar olmak üzere <TID, nesnekümesi> biçimindedir. Benzer şekilde aday kümelerinin her bir üyesi, $\overline{C}_k$ da <TID, nesnekümesi> biçimindedir.

AIS algoritmasına benzer olarak, SETM algoritması da veritabanı üzerinden çoklu geçişler yapar. İlk geçişte, ayrı ayrı her bir nesnenin destek sayısını sayar ve veritabanında hangilerinin büyük veya sık olduğunu bulur. Daha sonra, bir önceki geçişte oluşan yaygın nesnekümelerini genişleterek aday nesnekümelerini oluşturur. Ek olarak, SETM aday nesnekümelerinin TID'lerini de bilir. Aday nesnekümelerini oluştururken ilişkisel birleştirme işlemleri kullanılabilir (Srikant ve diğerleri, 1996b). Aday nesnekümelerini oluştururken aday nesnekümelerinin TID'leriyle birlikte bir kopyasını ardışıl biçimde saklar. Sonra, aday kümeleri nesnekümelerinin üzerinde sıralanır ve bir kabul fonksiyonuyla küçük nesnekümeleri silinir. Eğer veritabanı TID sıralı ise bir işlemde yer alan yaygın nesnekümeleri bir sonraki geçişte $\overline{L}_k$ 'yı TID'ye göre sıralanarak elde edilir. Bu yolla veritabanı üzerinden birçok geçiş yapılır. Algoritma daha fazla nesnekümesi bulamadığında sonlanır.

Bu algoritmanın ana dezavantajı $\overline{C}_k$ aday kümesinin sayısına bağlıdır (Agrawal ve diğerleri, 1994). Her bir aday nesnekümesinin TID içermesinden dolayı büyük sayıdaki TID'leri depolamak için daha fazla yere ihtiyaç duyar. Ayrıca, geçişin sonunda aday nesnekümelerinin

destek değerleri sayıldığında, $\overline{C}_k$ sıralı biçimde değildir. Bundan dolayı nesnekümeleri üzerinde yeniden sıralamaya ihtiyaç duyulur. Sonrasında, destek kısıtını sağlamayan aday nesnekümelerini devredışı bırakmakla aday nesnekümeleri budanır. $\overline{L}_k$ sonuç nesnekümesinde TID'lerin yeniden sıralanması gerekir. Daha sonra, sonraki geçişte aday kümelerini oluşturmak için $\overline{L}_k$ kullanılır. SETM algoritmasında bellek (tampon) yönetim tekniği ve iyileştirmesi ele alınmamıştır (Agrawal ve diğerleri, 1994). Ana bellekte $\overline{C}_k$'nın tutulabileceği varsayılmıştır. Bundan dolayı 1998 te Sarawagi, SETM'in etkili olmadığına ve ilişkisel VTYS üzerinde çalıştırıldığında rapor niteliğinde sonuç üretmediğine değinmiştir.

4.3.1.3 Apriori Algoritması

Agrawal ve diğerleri tarafından 1994 yılında geliştirilen Apriori algoritması, veri madenciliği tarihinde birliktelik kurallarının çıkarılması konusunda elde edilmiş büyük bir başarıdır. Birliktelik kuralları çıkarımında en çok bilinen algoritma olmuştur. Algoritmanın ismi, yaygın nesnelerin önsel bilgilerini kullanmasından yani bilgileri bir önceki adımdan almasından "önceki (prior)" anlamında aprioridir.

Bu teknik, yaygın bir nesnekümesinin tüm altkümeleri de yaygın olmalıdır kuralına dayanmaktadır. Ayrıca, bir nesnekümesindeki nesnelerin sözlük sıralı olduğunu varsayar. Bu algoritma AIS ve SETM algoritmalarına göre, aday nesnekümelerinin üretilme şeklinde ve sayılacak aday nesnekümelerinin seçilmesinde farklılık arzeder. Daha önce de değinildiği üzere, hem AIS hem de SETM algoritmalarında bir önceki geçişteki yaygın nesnekümeleri arasındaki ortak nesnekümeleri ve bir hareketteki nesneler elde edilir. Bu ortak nesnekümeleri hareket içindeki diğer bağımsız nesnelerle genişletilerek aday nesnekümeleri oluşturulur. Buna rağmen bu bağımsız nesneler yaygın olamazlar. Bilindiği gibi yaygın bir nesnekümesinin süperkümesi ile küçük bir nesnekümesi, küçük bir nesnekümesi sonucunu verir. Bu teknikler küçük olarak belirlenecek birçok aday nesnekümesi üretir. Apriori algoritması bu önemli nokta üzerinde odaklanır. Apriori, önceki geçişte oluşan yaygın nesnekümelerini birleştirerek aday nesnekümelerini oluşturur ve veritabanındaki hareketlerle ilgilenmeden, önceki geçişte oluşan altkümelerden küçük olanlarını siler. Önceki geçişte oluşan yaygın nesnekümelerinin dikkate alınmasıyla yaygın aday nesnekümelerinin sayısında anlamlı bir azalma olur.

İlk geçişte sadece bir nesne içeren nesnekümeleri sayılır. İlk geçişte bulunan yaygın nesnekümelerinden *apriori_gen()* fonksiyonu kullanılarak, ikinci geçiş için aday kümeleri oluşturulur. Aday nesnekümeleri bulunduğunda veritabanı taranarak iki uzunluklu yaygın nesnekümelerini bulmak için destek değerleri sayılır. İkinci geçişteki yaygın nesnekümeleri, üçüncü geçişteki yaygın nesnekümelerini oluşturacak aday kümeler olarak ele alınırlar. Yeni yaygın nesnekümesi bulunamayınca bu işlem sonlanır. Algoritma her *i*. geçişte veritabanını tarar ve *i* uzunluğundaki yaygın nesnekümelerini belirler. L_i ; *i* uzunluğundaki yaygın nesnekümelerini, C_i de *i* uzunluğundaki adayları ifade eder.

Apriori, seviye mantığı (level-wise) arama olarak bilinen yinelemeli bir yaklaşım kullanır. Bu yaklaşımda k-nesnekümeler, (k+1) nesnekümelerin araştırılması için kullanılır. İlk olarak, yaygın 1-nesnekümelerinin kümesi bulunur. Bulunan bu küme L_1 olarak adlandırılır. L_1, L_2'nin (yaygın 2-nesnekümelerin kümesi) bulunmasında kullanılır. L_2, L_3'ün bulunmasında

kullanılır ve algoritma bu şekilde daha fazla yaygın k-nesnekümeler bulamayıncaya kadar yinelemeli bir şekilde devam eder. Her L_k'nın bulunması bütün veritabanının taranması anlamına gelmektedir (Han ve Kamber, 2000).

apriori_gen() fonksiyonunun iki adımı vardır (Agrawal ve diğerleri, 1994). İlk adım esnasında, L_{k-1} kendisiyle birleştirilerek C_k elde edilir. İkinci adımda, birleşme sonucunda oluşmuş ve (k-1) altkümelerinden bir veya birkaçı L_{k-1} 'de olmayan tüm nesnekümeleri silinir. Kalan yaygın k-nesnekümeleri sonuç olarak döndürülür.

Çizelge 4.1 apriori_gen() fonksiyonunu kullanarak aday kümeleri bulma

Üçüncü geçişteki yaygın nesnekümeleri (L_3)	(L_3 , L_3) birleşimi	Dördüncü geçişteki aday kümeler (budama sonrası C_4)
{{Cips, Elma, Piliç}, {Cips, Elma, Kola}, {Elma, Kola, Piliç}, {Elma, Piliç, Yumurta}, {Cips, Kola, Piliç}}	{{Cips, Elma, Kola, Piliç}, {Elma, Kola, Piliç, Yumurta}}	{{Cips, Elma, Kola, Piliç}}

Çizelge 4.1'deki örneği dikkate alarak *apriori_gen()* fonksiyonunu örnekleyelim. Üçüncü geçiş sonucunda oluşan yaygın nesnekümeleri ilk kolonda gösterilmiştir. Bir hareketin {Cips, Elma, Kola, Piliç, Yumurta} 'dan oluştuğunu varsayalım. L_3 'ü kendisiyle birleştirince C_4 {{Cips, Elma, Kola, Piliç}, {Elma, Kola, Piliç, Yumurta}} olarak oluşur. Budama adımı {Elma, Kola, Piliç, Yumurta} nesnekümesini siler, çünkü bu kümenin 3 nesneli {Elma, Kola, Yumurta} altkümesi L_3 'te yoktur.

subset() fonksiyonu, bir harekette görülen aday kümelerin altkümelerini döndürür. Adayların destek değerlerinin sayılması, algoritmada zaman alan bir adımdır (Cengiz, 1997). Verilen bir harekette kontrol edilmesi gereken adayların sayılarını azaltmak için C_k aday nesnekümeleri bir hash ağacında saklanırlar. Hash ağacının bir düğümü bir yaprak düğümü veya bir hash tablosu (iç düğüm) içerir. Yaprak düğümler, aday nesnekümelerini sıralı olarak içerirler. Ağacın iç düğümlerinde çocuk düğümlere bağlantı içeren hash tabloları vardır. Nesneler, hash ağacına bir hash fonksiyonu kullanılarak eklenirler. Bir nesnekümesi eklendiğinde, kökten başlayarak ağacın bir yaprağına ulaşana dek gitmek gerekir. Ayrıca, budama adımını

hızlandırmak için L_k bir hash tablosunda saklanır (Srikant ve diğerleri, 1996b).

```
1)  L1 = find_frequent_1-itemsets(D);
2)  for (k = 2; Lk−1 ≠ φ; k++) {
3)      Ck = apriori_gen(Lk−1, min_sup);
4)      for each transaction t ∈ D { // scan D for counts
5)          Ct = subset(Ck, t); // get the subsets of t that are candidates
6)          for each candidate c ∈ Ct
7)              c.count++;
8)      }
9)      Lk = {c ∈ Ck | c.count ≥ min_sup}
10) }
11) return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k-1)-itemsets; min_sup: minimum support)
1)  for each itemset l1 ∈ Lk−1
2)      for each itemset l2 ∈ Lk−1
3)          if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
4)              c = l1 ⋈ l2; // join step: generate candidates
5)              if has_infrequent_subset(c, Lk−1) then
6)                  delete c; // prune step: remove unfruitful candidate
7)              else add c to Ck;
8)          }
9)  return Ck;

procedure has_infrequent_subset(c: candidate k-itemset; Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
1)  for each (k − 1)-subset s of c
2)      if s ∉ Lk−1 then
3)          return TRUE;
4)  return FALSE;
```

Şekil 4.1 Apriori Algoritması (Han ve Kamber, 2000)

4.3.1.4 Apriori-TID Algoritması

Daha önce de belirtildiği üzere, Apriori destek değerlerini saymak için her geçişte tüm veritabanını tarar. Ancak her geçişte tüm veritabanını taramak gereksizdir. Bu varsayıma dayanarak 1994 yılında Agrawal ve diğerleri, Apriori-TID ismiyle bir algoritma önermişlerdir. Apriori'ye benzer olarak, bir geçişin başında aday nesnekümelerini belirlemek için Apriori-TID de Apriori'nin aday üretim fonksiyonunu kullanır. Apriori'ye göre ana farkı ise ilk geçişten sonra destek değerlerini saymak için veritabanını kullanmaz. Bundan ziyade, $\overline{C_k}$ ile gösterilen, önceki geçişte kullanılan aday kümelerinin şifrelenmiş şeklini kullanır. SETM'deki gibi $\overline{C_k}$'nın her bir üyesi <TID, X_k> biçimindedir, TID tanımlayıcı, X_k ise potansiyel yaygın k-nesnekümesidir.

İlk geçişte, elde edilen C_1 veritabanına benzer. Buna rağmen, her nesne nesnekümesi tarafından yer değiştirilir. Diğer geçişlerde, T hareketinin yerini tutan $\overline{C_k}$'nın bir üyesi <TID, c> olarak ifade edilir. c, T'de bulunan C_k'ya bağlı bir adaydır. Bundan dolayı, $\overline{C_k}$'nın

büyüklüğü veritabanındaki hareket sayısından daha küçük olur. Ayrıca, $\overline{C}_k$'daki her bir kayıt, büyük k değerlerindeki yerini tutan hareketlerden daha küçüktür. Bunun nedeni çok az adayın harekette içerilmesidir. Başka şekilde ifade etmek gerekirse, $\overline{C}_k$'daki her bir kayıt küçük k değerlerindeki ilgili hareketten daha büyüktür (Srikant ve diğerleri, 1996b).

İlk olarak, tüm veritabanı taranır ve $\overline{C}_1$ nesnekümeleri elde edilir. $\overline{C}_1$'in her bir kaydı tüm nesneleri TID'leriyle birlikte içerir. 1-nesneli L_1 yaygın nesnekümeleri $\overline{C}_1$'deki kayıtlar sayılarak hesaplanır. Daha sonra C_2'yi elde etmek için *apriori_gen()* fonksiyonu kullanılır. Bir T hareketine ilişkin $\overline{C}_2$'deki kayıtlar, T'de bulunan C_2'nin üyelerinden elde edilir. Bunu gerçekleştirmek için tüm veritabanı yerine $\overline{C}_1$ taranır. Sonra, $\overline{C}_2$'deki destek değerleri sayılarak L_2 elde edilir. Bu işlem, boş aday nesnekümeleri oluşuncaya dek sürer.

Bu şifreleme fonksiyonunu kullanmanın avantajı, sonraki geçişlerde şifreleme fonksiyonunun büyüklüğü veritabanına oranla daha küçük olacağından okuma süresi azalır. AIS ve SETM tekniklerine kıyasla Apriori-TID daha iyidir. Çizelge 4.1'de verilen örnek kullanıldığında, L_3 şu şekilde bulunur; {{Cips, Elma, Piliç}, {Cips, Elma, Kola}, {Elma, Kola, Piliç}, {Elma, Piliç, Yumurta}, {Cips, Kola, Piliç}}. Apriori'ye benzer olarak Apriori-TID sonraki adımda sadece bir aday nesnekümesi oluşturur; {Cips, Elma, Kola, Piliç}. Daha önce de değinildiği üzere, hem AIS hem de SETM beş aday nesnekümesi oluşturmaktadır; {{Cips, Elma, Kola, Piliç}, {Cips, Elma, Piliç, Yumurta}, {Cips, Elma, Kola, Yumurta}, {Elma, Kola, Piliç, Yumurta} ve {Cips, Kola, Piliç, Yumurta}}.

Apriori-TID'de C_k'daki aday nesnekümeleri, C_k'daki nesnelerin TID'leriyle indekslenen bir dizi halinde saklanır. Her bir C_k sıralı yapıdadır. k. geçişte, L_{k-1} ve aday üretimi esnasında C_k için bellek alanına ihtiyaç duyar. Sayma safhasında ise C_{k-1}, C_k, $\overline{C}_k$ ve $\overline{C}_{k-1}$ için bellek alanına ihtiyaç vardır. Aday üretimi sırasında yaklaşık olarak belleğin (tamponun) yarısı adaylarla doldurulur. Bu, hem C_k hem de C_{k-1}'in konu ile ilgili parçasının hesaplama safhasında bellekte tutulmasını sağlar. Eğer L_k bellekte saklanamıyorsa, L_k'yı harici olarak sıralamak gerekir.

Apriori'ye benzer olarak bu algoritmanın performansı büyük bir IBM RS/6000 530H iş istasyonu üzerinde test edilmiştir (Agrawal ve diğerleri, 1994). Apriori-TID'in ilk geçişten sonra tüm veritabanı yerine C_k'yı kullanması ve sonraki geçişlerde $\overline{C}_k$'nın küçülmesiyle etkisi artmıştır. Buna rağmen, $\overline{C}_k$'ların büyük olabilmesi problemi SETM'de olduğu gibi

Apriori-TID'de de vardır, ancak Apriori-TID, SETM'in ürettiği aday nesnekümelerinden daha azını üretir. Ayrıca SETM'de olduğu gibi $\overline{C}_k$'nın sıralanmasına Apriori-TID'de ihtiyaç yoktur. $\overline{C}_k$'nın çok büyük olması durumunda bellek (tampon) yönetimiyle ilgili bazı sorunlar oluşur. Bellekte saklanabilecek $\overline{C}_k$ kümelerinin küçük sayıda olmasıyla Apriori-TID, Apriori'den daha iyi performans gösterir. Büyük veri kümeleri için ise Apriori'nin Apriori-TID'e göre performansının daha iyi olduğu belirtilmiştir (Agrawal ve diğerleri, 1994).

4.3.1.5 Apriori-Hybrid Algoritması

Bu algoritma, veri üzerindeki tüm geçişlerde aynı algoritmanın kullanılmasının zorunlu olmadığı fikrine dayanır. 1994 yılında Agrawal ve diğerleri tarafından değinildiği üzere, Apriori ilk geçişlerde daha iyi performansa sahiptir ve sonraki geçişlerde Apriori-TID, Apriori'den daha iyidir. Deneysel gözlemlere dayanarak Apriori-Hybrid tekniği, ilk geçişlerde Apriori'yi kullanacak şekilde tasarlanmıştır ve geçişlerin sonundaki $\overline{C}_k$ kümesinin bellekte karşılanacağını umarak Apriori-TID'e geçiş yapar. Bundan dolayı, her bir geçiş sonunda $\overline{C}_k$'nın tahmin edilmesi gerekmektedir. Ayrıca, Apriori'den Apriori-TID'e geçiş masraflıdır.

Bu tekniğin performansı büyük verisetleri üzerindeki tecrübelere dayanarak da hesaplanmıştır. Geçişlerin sonlarına doğru değişimin olması durumu dışında, Apriori-Hybrid'in Apriori'den daha iyi performans gösterdiği belirlenmiştir (Srikant ve diğerleri, 1996b).

4.3.1.6 OCD (Off-line Candidate Determination) Algoritması

OCD tekniği 1994 yılında Mannila tarafından önerilmiş olup, küçük örneklerin yaygın nesnekümelerini bulmada genellikle epey iyi olduğu fikrine dayanmaktadır. OCD tekniği, gereksiz aday kümelerini elemek için önceki geçişlerden elde edilen bilginin birleştirme analizi sonuçlarını kullanır. Bir $Y \subseteq I$ altkümesinin yaygın olmadığını bilmek için, *s* destek eşik değeri olmak üzere, en azından (1 - *s*) kadar hareket taranmalıdır. Bundan dolayı *s*'in küçük değerleri için neredeyse tüm ilişkinin okunması gerekir. Eğer veritabanı çok büyükse, veri üzerinde mümkün olduğunca az geçiş yapmanın önemli olduğu aşikârdır.

OCD, aday kümeleri belirlemek için AIS'ten farklı bir yaklaşım kullanır. OCD, mümkün olduğunca basit bir şekilde geçişi koruyarak, geçişler arasındaki aday kümelerini budamak için önceki geçişlerdeki tüm elverişli bilgiyi kullanır. *k* uzunluklu tüm yaygın nesnekümelerini içeren L_k kümesini oluşturur. L_k'daki yaygın nesnekümelerini içerenve L_{k+1}'de de olabilecek (*k* + 1) uzunluklu aday kümeler C_{k+1}'de bulunur. *e*, L_k'nın uzantısı

(extension) olmak üzere, eğer $X \in L_{k+e}$ ve $e \geq 0$ olduğu biliniyorsa, *X*, L_k'dan $\binom{k+e}{k}$ adet küme içerir. Örneğin; eğer *e*=1, *k*=2 ve $X \in L_3$ ise *X*, L_2'den $\binom{3}{2}$ yani 3 küme içerir. Benzer şekilde, L_4'ün her bir nesnesi L_3'ün 4 nesnesini içerir, ... vb. Örneğin L_2 = {{Elma, Muz}, {Lahana, Muz}, {Elma, Lahana}, {Elma, Yumurta}, {Muz, Yumurta}, {Dondurma, Elma}, {Lahana, Puding}} olsun. {Elma, Lahana, Muz} ve {Elma, Muz, Yumurta} kümelerinin L_3'ün olası üyeleri olduğu sonucu çıkarılabilir. Çünkü 2 uzunluklu tüm altkümeleri L_2'de olan 3 uzunluklu kümeler bunlardır. Bu safhada L_4 boştur. Bunun sebebi L_4'ün herhangi bir üyesi L_3'ün 4 nesnesini içermesi gerekirken L_3, 2 üyeye sahiptir. Bundan dolayı C_{k+1} aşağıdaki yöntemle sayılır:

$$C_{k+1} = \{ Y \subseteq I \; ; \; |Y| = k+1 \text{ ve } Y, \; L_k \text{'nın } (k+1) \text{ üyelerini içerir} \} \tag{4.8}$$

C_{k+1} bulma çözümü yorucu bir prosedürdür. Yorucu metodta, *k*+1 uzunluklu tüm altkümeler denetlenir. Buna rağmen, bu prosedür fazla sayıda gereksiz adaylar üretir ve bu masraflı bir tekniktir. Sayma işlemini uygun hale getirmek için OCD iki alternatif yaklaşım önermektedir. Bunlardan biri, ortak olarak (*k*-1) nesnelerini içeren, L_k birleşimlerini biçimlendirerek bir C'_{k+1} koleksiyonu hesaplamaktır:

$$C'_{k+1} = \{ Y \cup Y' \; ; \; Y, Y' \in L_k \text{ ve } |Y \cup Y'| = (k-1) \} \tag{4.9}$$

Daha sonra, $C_{k+1} \in C'_{k+1}$ ve C_{k+1}, C'_{k+1}'deki her bir küme için C_{k+1} denkleminin sağlanıp sağlanmadığı kontrolü yapılarak hesaplanabilir.

İkinci yaklaşım, L_k ve L_1'deki kümelerin birleşiminin biçimlendirilmesidir:

$$C''_{k+1} = \{ Y \cup Y' \; ; \; Y \in L_k \text{ ve } Y' \in L_1 \text{ ve } Y' \not\subset Y \} \tag{4.10}$$

Sonrasında, 1. denklemdeki içerme şartını kontrol ederek C_{k+1} hesaplanır.

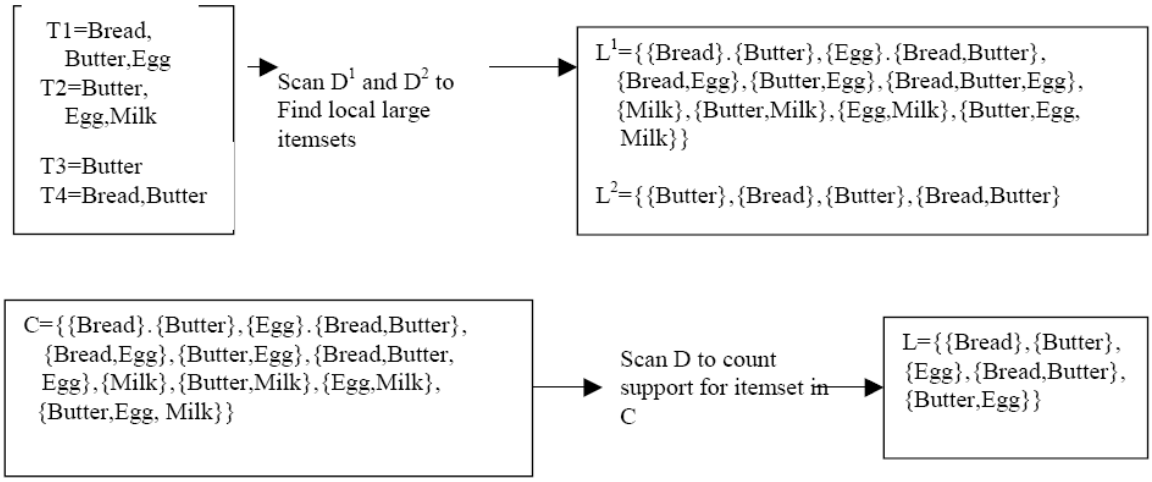
C_{k+1}'i oluşturmak için gereken işlem veritabanının boyutuna değil, L_k'nın boyutuna bağlıdır. Bir de *e*>1 için L_k'dan direkt olarak çeşitli C_{k+1}, C_{k+2}, ... hesaplanır. C'_{k+1}'den C_{k+1}'in elde edilmesi için gereken zaman karmaşıklığı $O(k|L_k|^3)$'tür. Diğer yandan, C''_{k+1}'den C_{k+1} elde edilirken geçen çalışma süresi doğrusal olarak veritabanının boyutuna (*n*) ve üssel olarak en

büyük yaygın nesnekümesinin boyutuna bağlıdır. Bundan dolayı, çok büyük n değerleri için algoritma çok yavaş çalışmaktadır. Yaygın nesnekümeleri için iyi bir tahmin yapabilmek, büyük veritabanının küçük örneklerinin analiziyle mümkündür (Lee, 1998). Mannila tarafından 1994 yılında yapılan teorik analizler göstermiştir ki yaygın nesnekümelerini bulmak için küçük örnekler çok iyidir. Ayrıca, destek eşik değerinin küçük olması durumunda, 3000 satır içeren bir örnek üzerinde yaygın nesnekümelerinin bulunması esnasında son derece iyi bir tahminleme yapılabilmiştir.

Bu algoritmanın performansı 1994'te Mannila tarafından iki veriseti üzerinde test edilmiştir. Bunlardan biri 4734 öğrenciye ait ders kayıtlarını içeren veritabanıdır. İkincisi ise bir telefon firmasının 30.000 kayıt içeren hata yönetim veritabanıdır. Deneysel sonuçlar, OCD'nin süre gereksiniminin AIS'in 10-20%'si olduğunu göstermiştir. Düşük destek eşik değerlerinde OCD'nin avantajı artmaktadır (Mannila, 1994). AIS ile oluşturulan aday sayıları OCD tarafından oluşturulanlara oranla çok fazladır. AIS, geçiş sırasında kopya adaylar oluştururken OCD bir adayı bir kez oluşturur ve veritabanında hesaplamaya girmeden önce altkümelerinin yaygın olup olmadığını kontrol eder.

4.3.1.7 Partitioning Algoritması

PARTITION, 1995 yılında Savasere tarafından önerilmiştir ve veritabanı tarama sayısını 2'ye indirir. Veritabanını küçük bölmelere ayırır ve her bir bölmenin ana bellekte tutulabileceğini varsayar. Veritabanı bölmelerinin D^1, D^2, ..., D^p olduğunu düşünelim. İlk taramada her bir bölme için *yerel yaygın nesnekümeleri* bulunur; D^i $(1 \le i \le p)$ ve $\{ X \left| X.count \ge s \times \right| D^i \left| \right\}$. L_i yerel yaygın nesnekümeleri, Apriori benzeri seviye-mantığı algoritması kullanılarak bulunur. Her bir bölme ana belleğe yerleştirilebildiği sürece, bölmeyi ana belleğe yükledikten sonra her bir bölme için ek G/Ç diskine gerek yoktur. İkinci taramada, tüm veritabanındaki yaygın bir nesnekümesinin, veritabanının en az bir bölmesinde de yerel olarak yaygın olması gerektiği özelliğini kullanır. Daha sonra, her bir bölmede bulunan yerel yaygın nesnekümelerinin birleşimi adaylar olarak kullanılır ve bütün yaygın nesnekümelerini bulmak için tüm veritabanı üzerinde sayılırlar.



Şekil 4.2 PARTITION algoritmasını kullanarak yaygın nesnekümelerinin bulunması

Şekil 4.2, Çizelge 4.1 üzerinde PARTITION'ın kullanımını ele almaktadır. Eğer veritabanı ikiye bölünürse, ilk bölme ilk iki hareketi ve ikinci bölme de kalan iki hareketi içerir. Minimum destek değeri 40% iken ve her bir bölmede sadece iki hareket mevcutken, bir kez görülen bir nesnekümesi yaygın olabilir. İki bölmedeki yerel yaygın nesnekümeleri hareketlerin tüm altkümeleridir. Birleşim kümesi, ikinci tarama için aday nesnekümelerini içerir. Algoritma, aşağıda gösterilmiştir. Veritabanı bölmelerini belirtmek için süperscriptler ve nesnekümelerinin boyutlarını belirtmek için ise altscriptler kullanılmıştır.

Input:
$I, s, D^1, D^2, ..., D^p$
Output:
L
Algorithm:
//scan one computes the local large itemsets in each partition
1) **for** i from 1 to p **do**
2) $L^i = Apriori(I,D^i,s)$; //L^i are all local large itemsets(all sizes) in D^i
//scan two counts the union of the local large itemsets in all partitions
3) $C = \bigcup_i L^i$;
4) $count(C, D) = \bigcup D^i$;
5) return $L = \{x \mid x \in C, x.count \geq s \times |D|\}$;

Şekil 4.3 PARTITION algoritması (Dunham ve diğerleri, 2000)

PARTITION algoritması homojen veri dağılımına ihtiyaç duyar. Bunun nedeni, eğer bir nesnekümesi her bir bölmede eşit sayıdaysa, ikinci taramada sayılacak çoğu nesnekümesinin sayısı büyük olur. Buna rağmen, dağınık veri dağılımlarında, ikinci taramadaki birçok

nesnekümesi küçük olarak nitelendirilir, bu da yanlış nesnekümelerini saymak için işlem süresini arttırır. AS-CPA (Anti-Skew Counting Partition Algorithm) (Lin, 1998), anti-dağınık algoritmalarının bir ailesidir ve dağınık veri dağılımlarında PARTITION'ın performansını arttırmak için önerilmiştir. İlk taramada, önceki bölmelerde bulunan nesnekümelerinin sayısı, diğer bölmelerde biriktirilir ve arttırılır. Biriktirilen sayılar küçük olabilecek nesnekümelerini budamak için kullanılır. İlk budama teknikleri nedeniyle, ikinci taramada sayılacak yanlış nesnekümelerinin sayısı azaltılmış olur.

4.3.1.8 Sampling Algoritması

1996'da Toivonen tarafından geliştirilen örnekleme (sampling) algoritması veritabanı tarama sayısını en iyi durumda bire, en kötü durumda ikiye indirir. İlk olarak veritabanından ana belleğe yerleştirilebiecek bir örnek alınır. Örnekteki yaygın nesnekümelerinin kümesi, Apriori benzeri seviye-mantığı algoritması kullanılarak bulunur. Örnekteki yaygın nesnekümelerinin kümesli *PL* olsun. Bu, muhtemel yaygın nesnekümelerinin bir kümesi olarak ve büyün veritabanında doğrulanabilen adayları oluşturmak için kullanılır. Adaylar *PL* üzerinde BD^{-} negatif kenar fonksiyonu uygulanarak elde edilir; $BD^{-}(PL) \cup PL$. *PL* nesnekümeleri kümesinin negatif kenarı, *PL*'de olmayan ancak tüm altkümeleri *PL*'de olan en küçük nesnekümeleri kümesidir. Negatif kenar fonksiyonu, Apriori'deki *apriori_gen()* fonksiyonunun genellenmişidir ve *pl*'deki tüm nesnekümelerinin aynı boyutta olmasını bekler, $BD^{-}(PL) = apriori\_gen(PL)$. Farkı ise, negatif kenar farklı boyuttaki nesnekümeleri kümesine uygulanabilirken, *apriori_gen()* fonksiyonu tek boyuta uygulanır. Adaylar oluşturulduktan sonra, adayların sayılarını belirlemek için tüm veritabanı bir kereliğine taranır. Eğer tüm yaygın nesnekümeleri *PL*'de ise ve $BD^{-}(PL)$'deki hiçbir nesnekümesi yaygın olarak nitelendirilemiyorsa, tüm yaygın nesnekümeleri bulunmuş olur ve algoritma sonlanır. Bu, tüm yaygın nesnekümelerinin bulunduğunu garantiler çünkü *PL* tüm *L* yaygın nesnekümelerini içeriyorsa ($L \subseteq PL$), $BD^{-}(PL) \cup PL$ Apriori'nin tüm aday nesnekümelerini içerir. Diğer türlü, $BD^{-}(PL)$'de kaçaklar varsa, tüm yaygın nesnekümelerinin bulunduğundan emin olmak için bazı yeni aday nesnekümeleri sayılmalıdır ve bu da fazla bir tane daha taramayı gerektirir. Bu durumda, $L \cap PL \neq \varnothing$ ise, ilk taramadaki aday nesnekümeleri Apriori'nin tüm aday nesnekümelerini içermez.

Örnekleme tekniğini örneklemek için; PL = {{A}, {B}, {C}, {A, B}} olsun. İlk taramada oluşan aday nesnekümeleri $BD^{-}(PL) \cup PL$ = {{A, C}, {B, C}} ∪ {{A}, {B}, {C}, {A, B}} = {{A}, {B}, {C}, {A, B}, {A, C}, {B, C}}. Eğer *L* = {{A}, {B}, {C}, {A, B}, {A, C},

{B, C}} ise, $BD^{-}(PL)$'de {A, C} ve {B, C} olarak iki kaçak vardır. Apriori'nin bir adayı olan ve yaygın olabilecek {A, B, C} nesnekümesi örneklemenin ilk taramasında sayılmamıştır. Bu nedenle örnekleme algoritması {A, B, C} gibi yeni aday nesnekümelerini saymak için bir adet daha taramaya ihtiyaç duyar. Negatif kenar fonksiyonunun kaçaklara tekrarlamalı olarak uygulanmasıyla yeni aday nesnekümeleri oluşturulur.

Input:
I, s, D
Output:
L
Algorithm:
//draw a sample and find the local large itemsets in the sample
1) Ds = a random sample drawn from D;
2) PL = Apriori(I,Ds,s);
//first scan counts the candidates generated from PL
3) C = PL $\cup$ BD^{-}(PL);
4) count(C, D);
//second scan counts additional candidates if there are misses in BD^{-}(PL)
5) ML = {x | x $\in$ BD^{-}(PL), x.count $\geq$ s $\times$ |D|}; //ML are the misses
6) **if** ML $\neq \varnothing$ **then** //MC are the new candidates generated from the misses
7) MC = {x | x $\in$ C, x.count $\geq$ s $\times$ |D|};
8) **repeat**
9) MC = MC $\cup$ BD^{-}(MC);
10) **until** MC doesn't grow;
11) MC = MC - C); //itemsets in C have already been counted in scan one
12) count(MC, D);
13) return L = {x | x $\in$ C $\cup$ MC, x.count $\geq$ s $\times$ |D|};

Şekil 4.4 Sampling algoritması (Dunham ve diğerleri, 2000)

4.3.1.9 DIC (Dynamic Itemset Counting) Algoritması

DIC, nesnekümelerini önceden oluşturmaya ve saymaya çalışır, bu da veritabanı tarama sayısını azaltır (Brin, 1997). Veritabanı aralıklı (interval) hareketler olarak gösterilir ve aralıklar ardışıl olarak taranır. İlk aralığın taranması sırasında, 1-nesnekümeleri oluşturulur ve sayılır. İlk aralığın sonunda, muhtemelen yaygın olan 2-nesnekümeleri oluşturulur. İkinci aralığı tararken, oluşturulan tüm 1-nesnekümeler ve 2-nesnekümeler sayılır. İkinci aralığın sonunda, muhtemelen yaygın olan 3-nesnekümeleri oluşturulur ve üçüncü aralığı tarama sırasında 1-nesnekümeler ve 2-nesnekümeler ile birlikte sayılır. Genellikle, *k*. aralığın sonunda, muhtemelen yaygın olan (*k* + 1)-nesnekümeleri oluşturulur ve sonraki aralıklarda önceki nesnekümeleriyle birlikte sayılırlar. Veritabanının sonuna ulaşıldığında, veritabanını

baştan itibaren yine ele alarak tam olarak sayılmamış nesnekümelerini sayar. Veritabanı taramalarının asıl sayısı aralık boyutuna bağlıdır. Eğer aralık yeterince küçükse, tüm nesnekümeler ilk taramada oluşturulur ve tam olarak ikinci taramada sayılır. Ayrıca, PARTITION'da olduğu gibi homojen bir dağılıma ihtiyaç duyulur.

4.3.1.10 CARMA (Continuous Association Rule Mining Algorithm) Algoritması

CARMA, küçük nesnekümelerinin hesaplanmasını çevrimiçi (online) olarak yapar (Hidb, 1999). Çevrimiçi olarak, CARMA, kullanıcıya mevcut birliktelik kurallarını gösterir ve veritabanının ilk taranmasındaki herhangi bir işlemde kullanıcının minimun destek ve minimum güven parametrelerini değiştirmesini sağlar. En çok 2 veritabanı taraması gerektirir. DIC'e benzer olarak, CARMA da ilk taramada nesnekümelerini oluşturur ve ikinci taramada tüm nesnekümelerinin sayılmasını bitirir. DIC'ten farklı olarak, CARMA hareketler üzerinden geçerken nesnekümelerini oluşturur. Her bir hareketi okuduktan sonra, ilk olarak hareketin altkümeleri olan nesnekümelerinin sayılarını arttırır. Daha sonra, eğer nesnekümesinin mevcut tüm altkümeleri minimum destek değerini sağlayacak şekilde ve veritabanının okunan bölümünde büyükseler, hareketten nesnekümeleri oluşturulur. Bir nesnekümesinin muhtemel olarak yaygın olmasının tam olarak kestirilebilmesi için, nesnekümesinin sayısı için bir üst sınır hesaplanır. Bu, mevcut sayısının toplamı ve nesnekümesi oluşturulmadan önce oluş sayısının tahmin edilmesidir. Oluş sayısının (maksimum kaçaklar) kestirilmesi işlemi, nesnekümesi ilk oluşturulduğunda hesaplanır.

4.3.1.11 FP-Growth (Frequent Pattern Growth) Algoritması

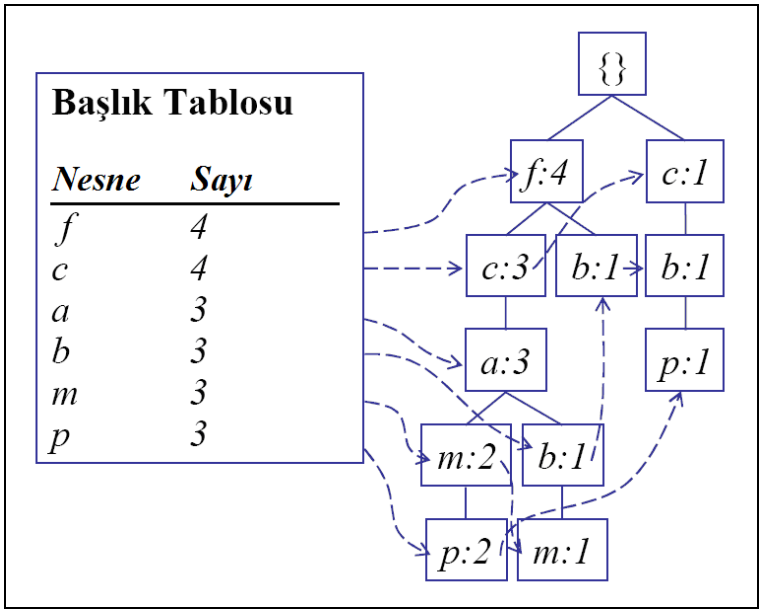
Aday üretmeksizin yaygın nesnekümelerinin bulunması için geliştirilmiş bir metoddur. İlk olarak veritabanı, yaygın nesneleri temsil edecek şekilde FP-Tree (Frequent Pattern Tree – FP-Ağacı) denilen ağaç yapısına sıkıştırılır. Bu ağaçta nesnekümelerinin birliktelik bilgileri yer almaktadır. Daha sonra, sıkıştırılmış veritabanı şartlı veritabanlarına bölünür. Her biri yaygın bir nesne ile ilişkilendirilmiştir ve bu veritabanları ayrı ayrı madenlenir.

FP-Growth metodu büyük yaygın nesnekümelerini bulma problemini tekrarlı bir şekilde küçüklerin araştırılması ve sonrasında soneklerinin (suffix) birleştirilmesi problemine dönüştürür. Az tekrarlı nesneleri sonek olarak kullanarak iyi seçicilik sağlar. Bu metod arama maliyetlerini önemli ölçüde azaltır.

TID	Nesneler	Yaygın nesneler (sıralı)
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
		minsup = 3

Şekil 4.5 FP-Growth örnek veri seti

Öncelikle veritabanı Apriori'deki gibi bir kez taranarak 1-nesnekümeler bulunur. Yaygın nesneler destek sayılarına göre büyükten küçüğe sıralanırlar (F-list = f, c, a, b, m, p). Ardından, veritabanı bir kez daha taranarak FP-Tree oluşturulur. FP-Tree, yaygın nesneleri bulmak için gerekli tüm bilgiyi barındırır. Yaygın olmayan nesneler ağaçta bulunmaz ve destek sayısı daha büyük olan nesneler köke daha yakındır. Ayrıca, FP-Tree asıl veri kümesinden daha büyük değildir [10].



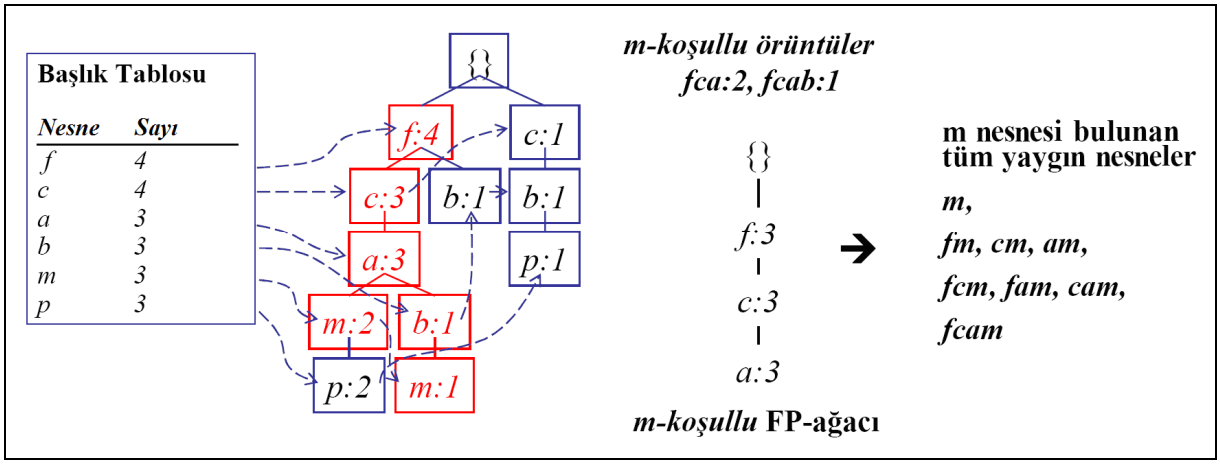
Şekil 4.6 Yaygın nesnekümelerinden, örnek veri seti kullanılarak FP-Tree'nin oluşturulması

Örüntüleri bulmak için yaygın nesneler, f-listesine göre altkümelere bölünür. Örnek veri seti için f-list = f, c, a, b, m, p dir. p nesnesi bulunan örüntüler, m nesnesi bulunan ancak p nesnesi bulunmayan örüntüler, ..., c nesnesi bulunan ancak a, b, m, p nesneleri bulunmayan örüntüler, f nesnesi bulunan örüntüler gibi tarama yapılarak örüntüler bulunur.

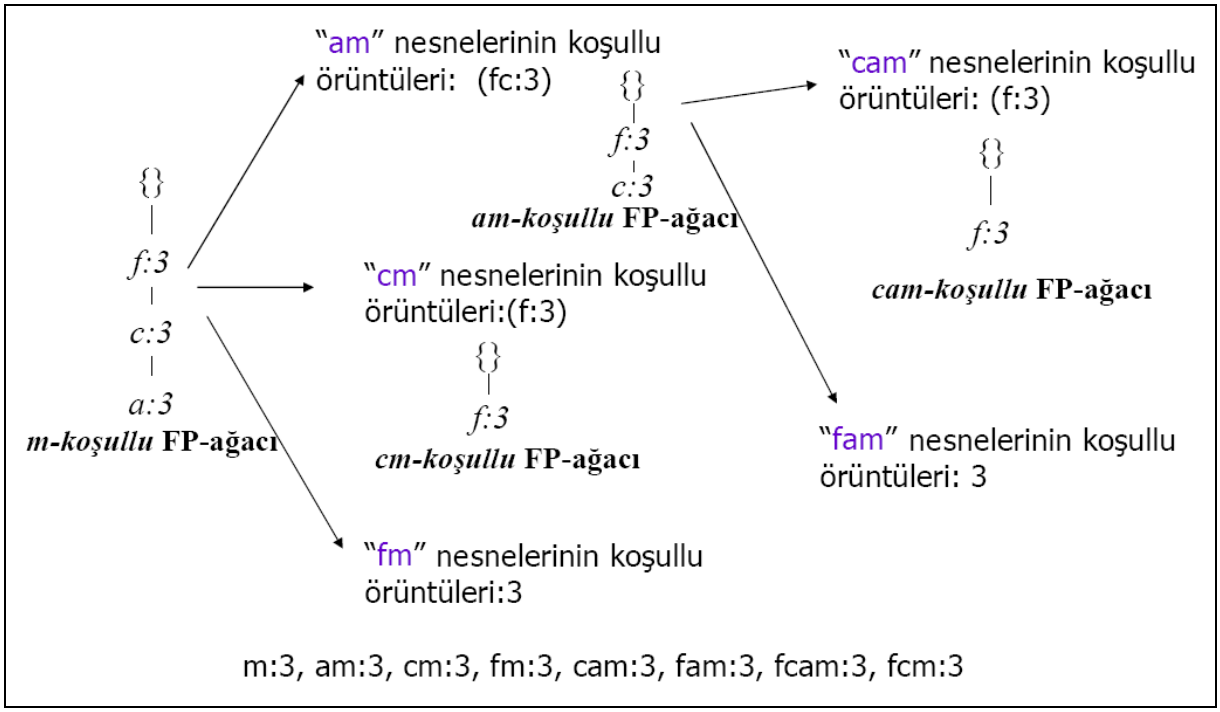
Nesne Koşullu Örüntü

nesne	*koşullu örüntü*
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

Şekil 4.7 FP-Tree'den elde edilen nesne koşullu örüntüler



Şekil 4.8 m-koşullu FP-Tree ve m nesnesi bulunan yaygın nesneler



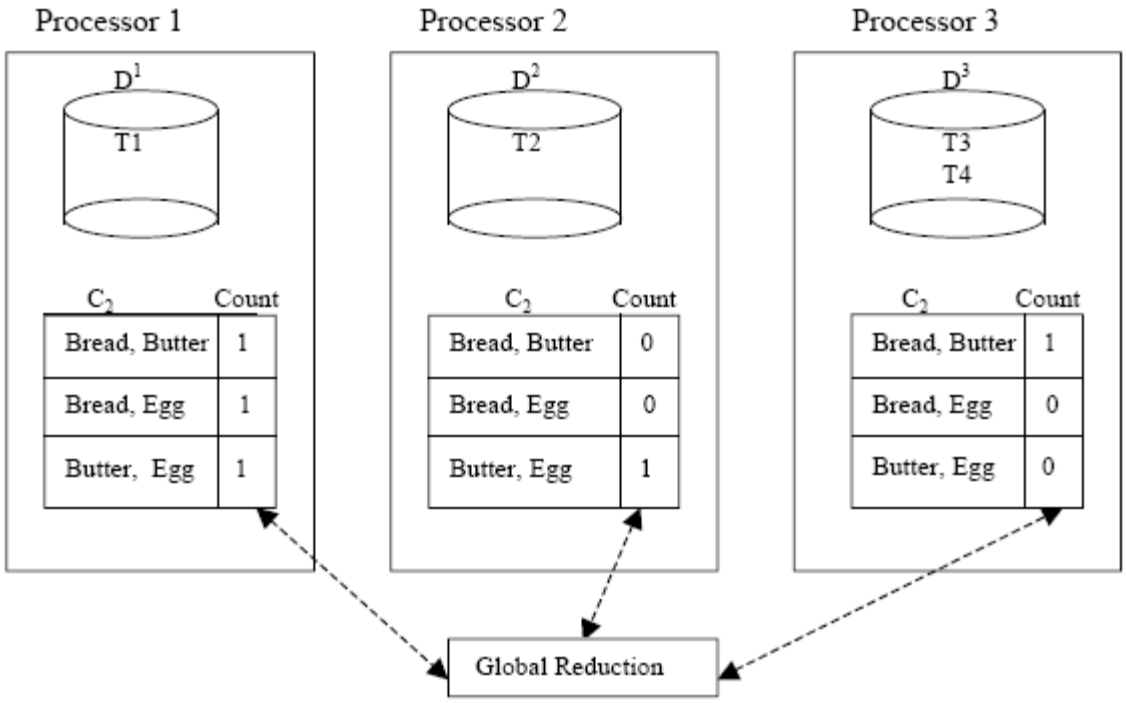
Şekil 4.9 m-koşullu FP-Tree'den yaygın nesnelerin bulunması [10]

FP-Growth metodu üzerinde yapılan performans çalışmaları göstermiştir ki, algoritma büyük yaygın nesnekümelerinin madenlenmesinde etkili ve ölçeklenebilir bir yapıdadır ve Apriori algoritmasından daha hızlı olduğu tespit edilmiştir (Han ve Kamber, 2000).

4.3.2 Paralel ve Dağıtılmış (Distributed) Algoritmalar

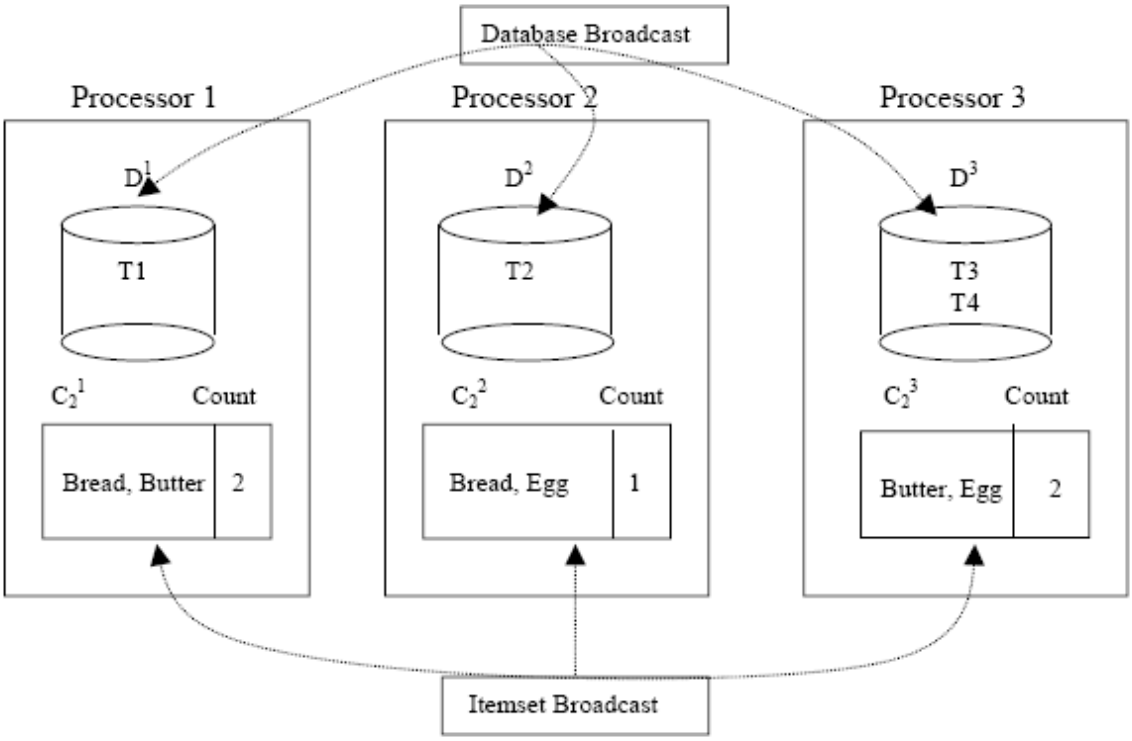
Günümüzdeki paralel ve dağınık algoritmaların çoğu Apriori seri algoritmasını baz alır. Zaki tarafından 1999 yılında yapılan mükemmel bir araştırma, algoritmaları yükleme-dengeleme (load-balancing) stratejisine, mimarisine ve paralelliğine göre sınıflandırır. Burada paralelleştirmeye odaklanılmıştır: *veri paralelleştirme* (data parallelism) ve *görev paralel-leştirme* (task parallelism) (Chat, 1997). İki modelin farkı, aday kümelerin işlemciler arasında dağıtılıp dağıtılmadığına bağlıdır. Veri paralelleştirme modelinde, her bir düğüm aynı aday kümelerini sayar. Görev paralelleştirme modelinde, aday küme bölümlendirilir ve işlemciler arasında dağıtılır ve her bir düğüm farklı bir aday kümesi sayar. Buna rağmen veritabanının teorik olarak iki modelde de bölümlendirilmesi gerekmez. Birçok etkili G/Ç uygulamasında genellikle veritabanının bölümlendirildiği ve işlemciler arasında dağıtıldığı varsayılır.

1996 yılında Agrawal tarafından önerilen sayma dağıtım algoritması veri paralelleştirme modelinde temsili bir algoritmadır. Adaylar tüm işlemcilerde kopyalanır ve veritabanı işlemciler arasında dağıtılır. Her bir işlemci, tüm adayların kendi veritabanı bölmesinde destek sayıları olan *yerel destek sayıları*nı (local support counts) hesaplamaktan sorumludur. Daha sonra tüm işlemciler veritabanının bütünündeki adayların toplam destek sayıları olan *küresel destek sayıları*nı (global support counts), yerel destek sayılarıyla takas ederek (küresel azaltım – global reduction) hesaplar. Sonuç olarak, yaygın nesnekümeleri her bir işlemcide bağımsız olarak hesaplanır. Veri paralelleştirme modeli, aşağıdaki şekilde ele alınmıştır. Dört hareket, üç işlemci arasında bölümlendirilir; 3 nolu işlemcide T3 ve T4 hareketleri, 1 nolu işlemcide T1 ve 2 nolu işlemcide T2 hareketleri mevcuttur. İkinci taramadaki üç aday nesnekümeleri her bir işlemcide kopyalanır. Yerel veritabanlarının taranmasından sonra yerel destek sayıları gösterilir.



Şekil 4.10 Veri paralelleştirme modeli (Dunham ve diğerleri, 2000)

1996 yılında Agrawal tarafından önerilen veri dağıtım algoritması görev paralelleştirme modelinde temsili bir algoritmadır. Veritabanındaki gibi aday kümeler işlemciler arasında bölümlendirilir ve dağıtılır. Her bir işlemci, adayların sadece bir altkümesi için küresel destek sayılarını tutmakla sorumludur. Bu yaklaşım her bir yinelemede iki iletişim çevrimine ihtiyaç duyar. İlk çevrimde, işlemcilerden her biri diğer tüm işlemcilere kendi veritabanı bölmesini gönderir. İkinci çevrimde, sıradaki yinelemede adayları hesaplamak için işlemcilerden her biri bulduğu yaygın nesnekümelerini diğer işlemcilere yayımlar. Görev paralelleştirme modeli, aşağıdaki şekilde gösterilmiştir. Dört hareket de veri paralelleştirmesindeki gibi bölümlendirilir. Her bir işlemcinin bir aday nesnekümesi içereceği şekilde, üç aday nesnekümesi işlemciler arasında bölümlendirilmiştir. Yerel veritabanı tarandıktan ve veritabanı bölmeleri diğer işlemciler tarafından yayımlandıktan sonra, her bir adayın küresel sayısı gösterilmiştir.



Şekil 4.11 Görev paralelleştirme modeli (Dunham ve diğerleri, 2000)

- **Veri Paralelleştirme Algoritmaları**

Veri paralelleştirme modelini benimseyen algoritmalar; CD (Agrawal, 1996), PDM (Park, 1995), DMA (Cheung, 1996) ve CCPD (Zaki, 1996)'dır. Bu paralelleştirme algoritmaları arasında aday budama veya etkili aday sayım tekniklerinin çalıştırılıp çalıştırılmaması ile ilgili farklar bulunur. CD temsili algoritması detaylı bir şekilde anlatılmış olup diğer üç algoritmanın yalnızca ek teknikleri ele alınmıştır.

4.3.2.1 CD (Count Distribution) Algoritması

CD tekniğinde, D veritabanı $\{D^1, D^2, ..., D^p\}$ olarak bölümlendirilir ve n adet işlemci arasında dağıtılır. Farkedileceği üzere, işlemci sayısını belirtmek için süperscript, aday boyutunu belirtmek içinse altscript kullanılmıştır. CD'nin i işlemcisindeki program kısmı $(1 \le i \le p)$, aşağıdaki şekilde gösterilmiştir. Temel olarak üç adım vardır. 1. adımda, D^i veritabanı bölmesindeki C_k adaylarının yerel destek sayıları bulunur. 2. adımda, her bir işlemci tüm adayların küresel destek sayılarını elde etmek için tüm adayların yerel destek sayılarını birbiriyle takas eder. 3. adımda ise, L_k küresel yaygın nesnekümeleri tanımlanır ve

her bir işlemcide bağımsız olarak L_k üzerinde *apriori_gen()* uygulanarak $k + 1$ uzunluktaki adaylar oluşturulur. Aday bulunamayana dek CD, 1. ve 3. adımlar arasını tekrarlar.

Input:
I, s, D^1, D^2, …, D^p
Output:
L
Algorithm:
1) C_1=I;
2) **for** k=1;$C_k \neq \emptyset$;k++ **do begin**
//step one: counting to get the local counts
3) count(C_k, D^i); //local processor is *i*
//step two: exchanging the local counts with other processors
//to obtain the global counts in the whole database.
4) **forall** itemset X ∈ C_k **do begin**
5) X.count=$\sum_{j=1}^{p}${X^j.count};
6) **end**
//step three: identifying the large itemsets and
//generating the candidates of size k+1
7) L_k={c ∈ C_k | c.count ≥ s × | $D^1 \cup D^2 \cup \ldots \cup D^p$ |};
8) C_{k+1}=apriori_gen(L_k);
9) **end**
10) return L=$L_1 \cup L_2 \cup \ldots \cup L_k$;

Şekil 4.12 CD algoritması (Dunham ve diğerleri, 2000)

4.3.2.2 PDM (Parallel Data Mining) Algoritması

PDM, doğrudan hashing tekniğini içerecek şekilde CD'nin değiştirilmesiyle oluşturulmuştur (Park, 1995). Hash tekniği sonraki geçişteki bazı adayları budamak için kullanılır. Apriori'nin L_1'den C_2 oluşturma sırasında budama işlemi yapmamasından dolayı özellikle ikinci geçişte kullanışlıdır. İlk geçişte, tüm 1-nesnekümelerinin sayılmasına ek olarak PDM, 2-nesnekümelerinin sayılarını saklamak için bir hash tablosu oluşturur. Hash tablosunda 2-nesnekümelerinin kendisinin tutulmasına gerek yoktur, sadece her bir kovanın (bucket) sayısı tutulur. Örneğin; {A, B} ve {C}'nin yaygın nesneler olduğu varsayılırsa, 2-nesnekümeler için hash tablosunda kova {AB, AD}'yi küçük yapmak için içerir (bu kovadaki sayı minimum destek sayısından küçüktür). PDM, hash tekniğiyle AB'yi 2 uzunluklu aday olarak oluşturmazken, ilk geçişten 2-nesnekümeleriyle ilgili hiçbir bilgi alınamadığı gibi Apriori ikinci geçiş için AB'yi aday olarak oluşturur. *k.* geçişte iletişim için, PDM hash tablosundaki $k + 1$ nesnekümelerinin yerel sayılarına ek olarak aday *k*-nesnekümelerinin yerel sayılarını takas etmeye ihtiyaç duyar.

4.3.2.3 DMA (Distributed Mining Algorithm) Algoritması

Aday budama tekniği ve iletişim mesajı azaltma tekniklerinin eklendiği veri paralelleştirme modeli bazlı bir tekniktir. Bir nesnekümesinin yoğun (hem bir veritabanı bölmesinde yerel olarak yaygın hem de tüm veritabanında küresel olarak yaygın) olup olmadığına karar vermek ve sonrasında yoğun yaygın nesnekümelerinden adayları oluşturmak için her bir işlemcideki yaygın nesnekümelerinin yerel sayılarını kullanır. Örneğin; A ve B sırasıyla 1 ve 2 nolu işlemcilerde yoğun olsun, bu durumda A küresel yaygın ve 1 nolu işlemcide yerel yaygındır ve B küresel yaygın ve 2 nolu işlemcide yerel yaygındır denir. DMA, AB'yi 2-nesneküme adayı olarak oluşturmazken Apriori, her bir işlemcideki yerel sayıları dikkate almayacağından dolayı AB'yi aday olarak oluşturur. İletişim için, CD'de olduğu gibi tüm adayların yerel sayılarının yayımlanması yerine DMA sadece yerel sayıları bir seçim yerine (polling site) gönderir, bu da mesaj boyutunu $O(p^2)$'den $O(p)$'ye düşürür.

4.3.2.4 CCPD (Common Candidate Partitioned Database) Algoritması

1996 yılında Zaki'nin önerdiği CCPD, bazı iyileştirmelerle SGI Power Challenge paylaşımlı bir bellek üzerinde CD'nin yürütülmesidir. Paylaşımlı bir bellek ortamında etkili bir şekilde aday üretilmesi ve sayılması için teknikler önermektedir. Yaygın nesnekümelerini ortak öneklere (genellikle ilk nesne) göre eşdeğer sınıflar şeklinde gruplandırır ve her bir eşdeğer sınıftan adayları oluşturur. Yaygın nesnekümelerinin gruplandırılması aday sayısını azaltmazken aday üretim süresini düşürür. Her bir harekette adayların etkili bir şekilde sayılması için kısa-döngülü (short-circuited) bir altküme kontrol metodunu öne sürmektedir.

- **Görev (Task) Paralelleştirme Algoritmaları**

Görev paralelleştirme modelini benimseyen algoritmalar; DD (Agrawal, 1996), IDD (Han, 1997), HPA (Shintani, 1996) ve PAR (Zaki, 1997)'dır. Bu algoritmalar adayları ve veritabanını işlemciler arasında bölüştürürler ve adayların ve veritabanının bölünme şekli farklılıklarını oluşturur. DD temsili algoritması detaylı bir şekilde anlatılmış olup diğer algoritmaların farklı tekniklerine değinilmiştir.

4.3.2.5 DD (Data Distribution) Algoritması

1996 yılında Agrawal tarafından önerilen DD tekniğinde adaylar, round-robin tarzında tüm işlemciler arasında bölüştürülmüş ve dağıtılmıştır. Teknik üç adımdan oluşur. İlk adımda, her bir işlemci kendi üzerinde dağıtılmış adayların yerel sayılarını elde etmek için yerel veritabanı bölmesini tarar. İkinci adımda, tüm işlemciler kendi veritabanı bölmesini diğer işlemcilere

yayımlar ve diğer işlemcilerden diğer veritabanı bölmelerini alarak tüm veritabanındaki küresel destek sayılarını elde etmek için alınan veritabanı bölmelerini tararlar. Son adımda ise, her bir işlemci kendi aday bölmesindeki yaygın nesnekümelerini hesaplar, tüm yaygın nesnekümelerini elde etmek için diğerleriyle takas eder ve sonra adayları ve bölmeleri oluşturarak, adayları tüm işlemciler arasında dağıtır. Bu adımlar, aday üretemeyene kadar devam eder. Veritabanı bölmelerini yayımlamanın iletişim maliyeti, iletişim ve hesaplamayı üst üste getiren asenkron iletişim ile azaltılabilir (Agrawal ve diğerleri, 1996).

Input:
$I, s, D^1, D^2, \ldots, D^p$
Output:
L
Algorithm:
1) $C_1^i \subseteq I$;
2) **for** ($k=1; C_k^i \neq \emptyset; k++$) **do begin**
//step one: counting to get the local counts
3) count(C_k^i, D^i); //local processor is *i*
//step two: broadcast the local database partition to others,
// receive the remote database partitions from others,
// scan $D^j (1 \leq j \leq p, j \neq i)$ to get the global counts.
4) broadcast(D^i);
5) **for** ($j=1$; ($j \leq p$ and $j \neq i$); $j++$) **do begin**
6) receive(D^j) from processor j;
7) count(C_k^i, D^j);
8) **end**
//step three: identify the large itemsets in C_k^i,
// exchange with other processors to get all large itemsets C_k,
// generate the candidates of size k+1,
// partition the candidates and distribute over all processors.
9) $L_k^i = \{c | c \in C_k^i, c.count \geq s * |D^1 \cup D^2 \cup \ldots \cup D^p|\}$;
10) $L_k = \bigcup_{i=1}^{p} (L_k^i)$;
11) C_{k+1} = apriori_gen(L_k);
12) $C_{k+1}^i \subseteq C_{k+1}$; //partition the candidate itemsets across the processors
13) **end**
14) return $L = L_1 \cup L_2 \cup \ldots \cup L_k$;

Şekil 4.13 DD algoritması (Dunham ve diğerleri, 2000)

4.3.2.6 IDD (Intelligent Data Distribution) Algoritması

IDD tekniği DD üzerinde geliştirmeler içeren bir tekniktir (Han, 1997). Adayların ilk nesnesine bağlı olarak adayları işlemciler arasında bölümlendirir. Bu, aynı ilk nesneli adayların aynı bölmede bölümlendirileceğini gösterir. Bundan dolayı, her bir işlemci sadece, işlemciye atanmış nesnelerden biriyle başlayan altkümeleri kontrol etme ihtiyacı duyar. Bu

DD'deki her bir işlemcinin her bir hareketteki tüm altkümeleri kontrol etme gibi birçok gereksiz hesaplamayı azaltır. Adayların yükleme-dengeleme (load-balanced) dağılımını elde etmek için adayları bölümlendirmeye yarayan bir kutu-paketleme (bin-packing) tekniğini kullanır. Bu teknik, her bir nesne için belirli nesne ile başlayan adayların sayısını hesaplar ve sonra her bir bölmedeki aday sayısı eşit olacak şekilde, aday bölmelere nesneleri atamak için kutu-paketleme algoritmasını kullanır. Ayrıca, iletişim masrafını azaltmak için bir halka mimarisini (ring architecture) benimser. Bu, yayımlama (broadcasting) yerine halkadaki komşular arasında noktadan noktaya eşzamansız iletişimi kullanmaktır.

4.3.2.7 HPA (Hash-based Parallel mining of Association rules) Algoritması

HPA, adayları farklı işlemcilere dağıtmak için bir hashing tekniği kullanır (Shintani, 1996). Örneğin; her bir işlemci, kendisine dağıtılmış olan adayları hesaplamak için aynı hash fonksiyonunu kullanır. Hesaplamada, işlemciler arasında bölünmüş veritabanlarında hareket etmek yerine, aynı hash tekniğiyle hareketlerin alt nesnekümelerini hedef işlemcilere hareket ettirirler. Böylelikle, bir hareketteki alt nesnekümesi *n* adet işlemci yerine sadece bir işlemciye gider. HPA, çarpık işleme tarzı (skew handling) tekniğiyle daha da geliştirilmiştir (Shintani, 1996). Çarpık işleme tarzı teknik, her bir işlemcide kullanılabilecek ana hafıza olması durumunda her bir işlemciye düşen işin daha orantılı olması için bazı adayların kopyalanmasıdır.

4.3.2.8 PAR (Parallel Association Rules) Algoritması

PAR, farklı adayları bölme ve saymada kullanılan bir grup algoritmadan oluşur (Zaki, 1997). Doğal yatay veritabanı bölmelerine (hareket listeleri) karşın dikey veritabanı bölmeleriyle (her bir nesnenin TID listesi) çalışırlar. Veritabanları için dikey organizasyonu kullanarak nesnekümesindeki nesnelerin TID listelerinin kesişimiyle bir nesnekümesinin hesaplanması daha kolay yapılmaktadır. Buna rağmen, veritabanı yatay olarak düzenlenmişse, dikey bölmelere dönüşümü gerekmektedir. Veritabanı, eşzamanlılığı azaltmak için kopyalanabilir. Par-Eclat ve Par-MaxEclat algoritmaları adayların ilk nesnelerine bağlı olarak eşitlik sınıfını kullanırken, Par-Clique ve Par-MaxClique algoritmaları adayları bölümlendirmek için maksimum hipergraf grubunu (hypergraph clique) kullanır. Hipergrafta, köşe (vertex) bir nesnedir, k köşesi arasındaki kenar, k köşesiyle ilgili nesneleri içeren nesnekümesidir ve bir grup birbiriyle bağlı tüm köşeleri içeren alt-graftır (sub-graph). Par-MaxEclat ve Par-MaxClique algoritmalarının bir özelliği azami (maximal) nesnekümelerini (diğerlerin altkümesi olmayan nesnekümeleri) bulmasıdır. Nesnekümesi sayımı alt-yukarı (bottom-up), üst-aşağı (top-down) veya melez (hybrid) şeklinde yapılabilir. Algoritmalar, adayları

bölümlendirmek için yaygın 2-nesnekümelerine ihtiyaç duyana dek (eşitlik sınıfı veya hipergraf grubu ile), tüm 2-nesnekümelerdeki oluşumları elde etmek için önişlem adımını kullanırlar.

- **Diğer Paralelleştirme Algoritmaları**

Tam manasıyla iki model arasında sınıflandırılamayan bazı diğer paralel algoritmalar mevcuttur. İki modeldeki benzer fikirleri paylaşmakla beraber ayrı özelliklere de sahiplerdir. Bu sebeple bu algoritmalar, diğer paralelleştirme algoritmaları olarak bölümlendirilmiştir. Bu paralel algoritmalar; Candidate Distribution (Agrawal, 1996), SH (Harada, 1998) ve HD (Han, 1997)'dir.

4.3.2.9 Candidate Distribution Algoritması

Agrawal tarafından 1996 yılında önerilen aday dağıtım (candidate distribution) algoritması, CD ve DD'deki eşzamanlılık ve iletişim ek yükünü azaltmak için geliştirilmiştir.

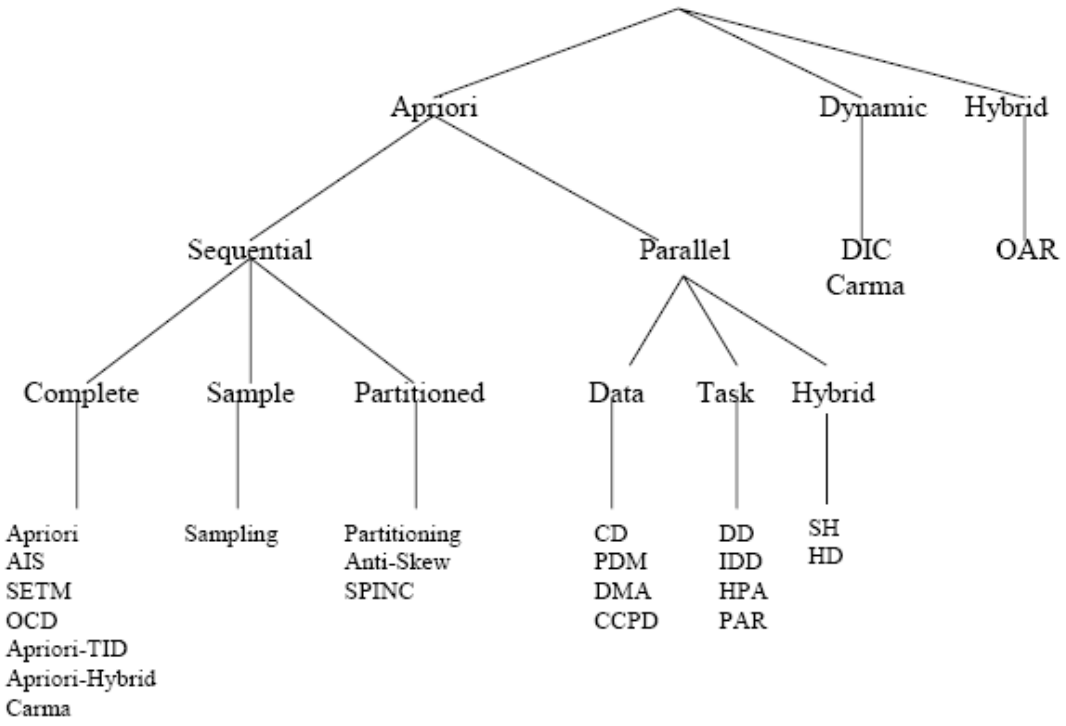
4.3.2.10 SH (Skew Handling) Algoritması

Harada tarafından 1998 yılında önerilen SH algoritmasında, Apriori sıralı algoritmasından farklı olarak, adaylar önceki yaygın nesnekümelerinden oluşturulmazlar. Bunun yerine her bir işlemcideki veritabanı bölmelerini tararken bağımsız olarak oluşturulur.

SH, Apriori'den farklı bir algoritma üzerine kurulmuş gibi görünse de, Apriori'ye çok yakın bir algoritmadır.

4.3.2.11 HD (Hybrid Distribution) Algoritması

HD, 1997 yılında Han tarafından önerilmiş olup, iki modelin birleşimidir. *p* işlemcilerinin *r* satırları ve *p/r* kolonları olarak iki boyutlu şekilde düzenlendiğini varsayar. Veritabanı *p* işlemcileri arasında eşit olarak bölümlendirilmiştir.



Şekil 4.14 Birliktelik algoritmalarının sınıflandırılması (Dunham ve diğerleri, 2000)

4.4 Birliktelik Kuralı Algoritmalarının Karşılaştırılması

Veritabanının herhangi bir taranmasında sayılan adayların azami sayısına bakılarak alan ihtiyacı tahmin edilebilir. Zaman ihtiyacını belirlemek için gereken veritabanı taramalarının azami sayısının (G/Ç tahmini) ve karşılaştırma işlemlerinin azami sayısının (işlemci tahmini) sayılması gerekir. Birçok hareketsel veritabanlarının ikincil disklerde saklanmasından ve G/Ç ek yüklerinin işlemci ek yüklerinden daha önemli olmasından bu yana, veritabanının bütününde yapılan tarama sayılarına odaklanılmıştır. Açıkça, veritabanındaki her bir hareketin tüm nesneleri içermesi en kötü durumdur. Her bir hareketteki nesne sayısı m ve D veritabanındaki k-nesnelerini içeren yaygın nesnekümeleri L_k olarak ifade edilmek üzere, yaygın nesnekümeleri sayısı 2^m kadardır. Seviye-mantığı (level-wise) tekniklerde (AIS, SETM, Apriori, ..., vb.), veritabanının ilk taranması sırasında tüm L_1 yaygın nesnekümeleri elde edilmiş olur. Benzer şekilde, tüm L_2 yaygın nesnekümeleri ikinci tarama sırasında elde edilir ve bu işlem böyle devam eder. m. taramada L_m nesnekümesi elde edilir. Yaygın nesnekümelerine ek girişler olmadığında ek taramaya gerek kalmayacağı için tüm algoritmalar sonlanır. Bu nedenle, tüm veritabanı en fazla (m+1) kez taranır. Apriori-TID tüm veritabanını ilk geçişte tarar. Sonrasında, (k+1). geçişte tüm veritabanı yerine $\overline{C_k}$'yı kullanır.

Buna rağmen, kötü durumlarda her zaman yardımcı olamaz. Bunun nedeni, tüm işlem sırasında $\overline{C_k}$'nın tüm hareketleri nesneleriyle birlikte içermesidir. Diğer taraftan, OCD tekniği algoritmanın başında L_1 yaygın nesnekümelerini elde etmek için tüm veritabanını bir kereliğine tarar. Sonra, OCD ve Sampling tüm veritabanının sadece bir bölümünü ve L_k'nın aday nesnekümelerini bulmak için $(1 < k \leq m)$ ilk geçişte elde edilen bilgiyi kullanır. İkinci taramada, her bir aday nesnekümesinin destek değerini hesaplar. Bu nedenle, verilen yeterli ana bellekte en kötü durumda iki tarama olur. PARTITION tekniği de, veritabanı taramalarını ikiye indirerek G/Ç ek yükünü azaltır. Benzer şekilde, CARMA en çok iki veritabanı taramasına ihtiyaç duyar.

Bir algoritmanın iyiliği, geliştirdiği "doğru" adayların sayısına bağlıdır. Daha önce de belirtildiği gibi, tüm algoritmalar aday kümeleri oluşturmak için önceki geçişlerde oluşan yaygın nesnekümelerini kullanır. Aday nesnekümelerini oluşturmak için bir önceki nesnekümelerinden oluşturulmuş yaygın nesnekümeleri ana belleğe alınır. Aday nesnekümeleri, destek değerlerinin sayılması için ana bellekte olmalıdır. Belleğin yetersiz gelmesi durumuna binaen farklı çeşitlilikte tampon (buffer) yönetimi ve depolama mimarilerine imkan veren çeşitli algoritmalar vardır. AIS, L_{k-1}'in ihtiyaç durumunda diskte kalmasını önermiştir. SETM, eğer $\overline{C_k}$ ana bellekte yer alamayacak kadar büyükse, FIFO (first in first out) tarzında belleğe yazılmasını önermiştir. Apriori ailesi, L_{k-1}'in diskte saklanmasını ve C_k'yı bulmak için bir bloğun ana belleğe alınmasına değinmiştir. Buna rağmen hem Apriori-TID hem de Apriori-Hybrid'de C_k, destek değerlerinin hesaplanması için ana bellekte olmalıdır. Diğer yandan, tüm diğer teknikler bu problemlerle baş etmek için yeterli belleğin olduğunu varsayarlar. Diğer tüm sıralı teknikler (PARTITION, Sampling, DIC ve CARMA) ana bellekte yer kaplayabilecek kadar tüm veritabanının elverişli bir bölümünün bulunmasına odaklanır. Apriori ailesi, yaygın nesnekümeleri için hash ağacı veya dizisi (hash tree or array) gibi uygun veri yapıları önermiştir. Buna rağmen AIS ve SETM, herhangi bir depolama mimarisi önermemiştir.

Birçok ticari şekilde uygulanabilir araçlar, birliktelik kuralları oluşturmak için Apriori tekniğini baz alırlar.

Bazı algoritmalar belirli durumlar altında kullanmak için son derece uygundur. AIS, veritabanındaki nesnelerin çok büyük sayıda olması durumunda iyi çalışmaz. Bundan dolayı AIS, küçük işlemsel veritabanları için uygundur. Daha önce de bahsedildiği üzere, Apriori ilk geçişlerde Apriori-TID'e oranla küçük çalışma sürelerine ihtiyaç duyar. Diğer yandan,

Apriori-TID sonraki geçişlerde Apriori'ye göre daha iyidir. Bundan dolayı uygun geçişle, Apriori-Hybrid en iyi performansı göstermektedir. Buna rağmen, Apriori'den Apriori-TID'e geçiş çok önemli ve ek yük getiren bir iştir. OCD, tahmini teknik olmasına rağmen, düşük destek eşik değerlerindeki yaygın nesnekümelerini bulmakta çok etkindir. CARMA, kullanıcı etkileşimli, geridönüşüm tabanlı online bir tekniktir ve hareket dizilerinin bir ağdan okunması sırasında etkilidir.

Aşağıdaki tablo, bahsedilen çeşitli algoritmaların kısaca karşılaştırılmasına ışık tutmakta ve özetlemektedir. Bu tablo azami tarama sayısını, önerilen veri yapılarını ve bazı açıklamaları içermektedir.

Çizelge 4.2 Algoritmaların karşılaştırılması (Dunham ve diğerleri, 2000)

Algoritma	**Tarama Sayısı**	**Veri Yapısı**	**Açıklamalar**
AIS	m+1	Belirtilmemiş	Belli başlı, az veri içeren seyrek hareketsel veritabanları için uygundur.
SETM	m+1	Belirtilmemiş	SQL uyumlu.
Apriori	m+1	L_{k-1} : Hash tablosu C_k : Hash ağacı	Belli başlı, orta düzey veri içeren hareketsel veritabanları için uygundur. AIS ve SETM'e oranla daha iyidir. Paralel algoritmaların baz aldığı algoritmadır.
Apriori-TID	m+1	L_{k-1} : Hash tablosu C_k : TID ile dizinlenmiş dizi $\overline{C_k}$: Sıralı mimari ID: bitmap	$\overline{C_k}$ 'nın büyük olması durumunda çok yavaştır. Küçük boyutlu $\overline{C_k}$ ile Apriori'den daha iyidir.
Apriori-Hybrid	m+1	L_{k-1} : Hash tablosu İlk safhada: C_k : Hash ağacı İkinci safhada: C_k : TID ile dizinlenmiş dizi $\overline{C_k}$: Sıralı mimari ID: bitmap	Apriori'den daha iyidir. Buna rağmen, Apriori'den Apriori-TID'e geçiş ek yük gerektirir. Geçiş noktasını saptamak çok önemlidir.
OCD	2	Belirtilmemiş	Düşük destek eşik değerleri ve çok büyük veritabanları için uygundur.
PARTITION	2	Hash tablosu	Büyük veritabanları için uygundur. Homojen veri dağılımını destekler.
Sampling	2	Belirtilmemiş	Düşük destek eşik değerleri ve çok büyük veritabanları için uygundur.
DIC	Aralık boyutuna bağlıdır	Ağaç	Veritabanı, hareketsel aralıklar şeklindedir. Yüksek boyutlu adaylar, bir aralığın sonunda oluşturulur.
CARMA	2	Hash tablosu	Hareket dizilerinin bir ağdan okunmasına uygundur. Online. Kullanıcılardan sürekli geri-besleme gelir ve destek ve/veya güven değerlerini işlem sırasında herhangi bir anda değiştirebilirler.
FP-Growth	2	FP-Tree	Aday üretimsiz bir tekniktir. Apriori ve türevlerinden daha iyi performansa sahiptir.
CD	m+1	Hash tablosu ve ağacı	Veri paralelleştirme.
PDM	m+1	Hash tablosu ve ağacı	Veri paralelleştirme, erken aday budaması.
DMA	m+1	Hash tablosu ve ağacı	Veri paralelleştirme, aday budaması.
CCPD	m+1	Hash tablosu ve ağacı	Veri paralelleştirme, paylaşımlı-bellek içeren makinelerde kullanılır.
DD	m+1	Hash tablosu ve ağacı	Görev paralelleştirme, round-robin bölümlendirmesi.
IDD	m+1	Hash tablosu ve ağacı	Görev paralelleştirme, ilk nesnelerle bölümlendirme.
HPA	m+1	Hash tablosu ve ağacı	Görev paralelleştirme, hash fonksiyonu ile bölümlendirme.
SH	m+1	Hash tablosu ve ağacı	Veri paralelleştirme, adaylar her bir işlemcide bağımsız olarak oluşturulur.
HD	m+1	Hash tablosu ve ağacı	Melez veri ve görev paralelleştirme, ızgara (grid) paralel mimarisi

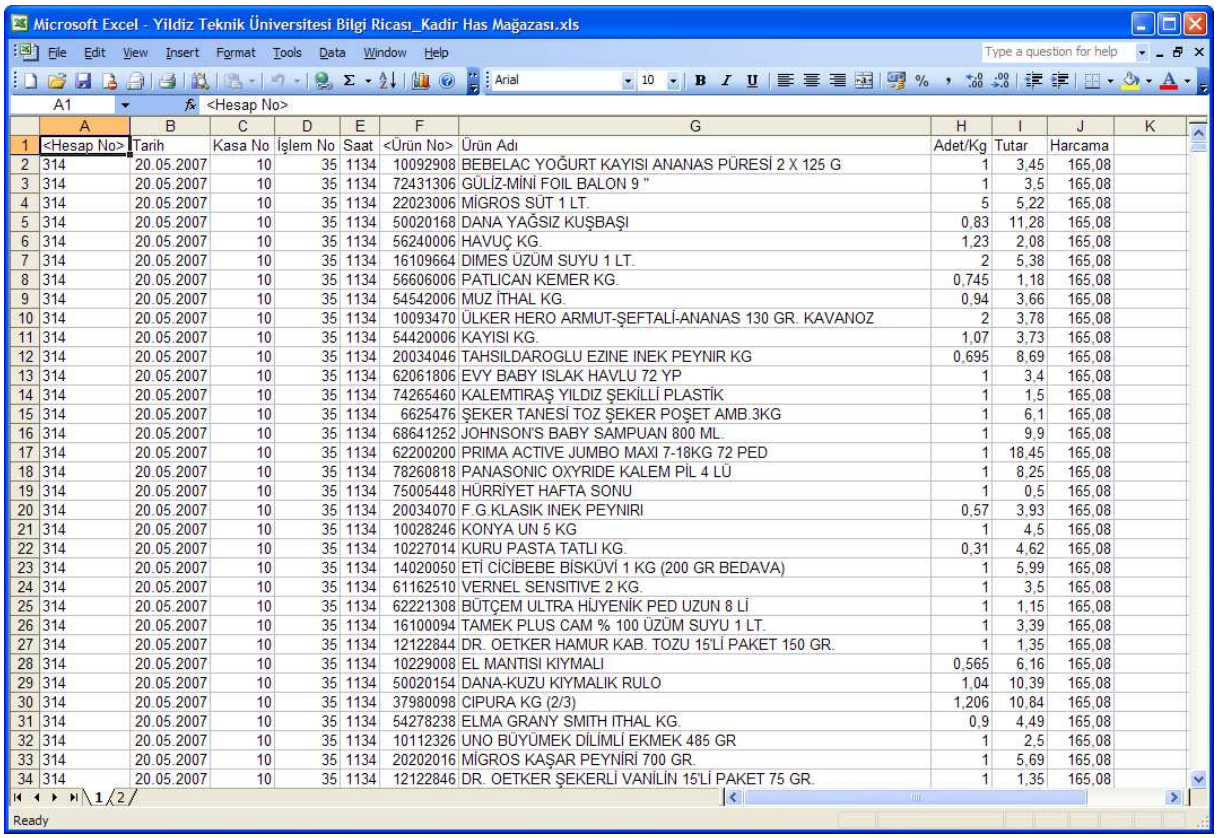
5. UYGULAMA

Bu bölüm, tez çalışmasında kullanılmak üzere Migros Türk T.A.Ş. Genel Müdürlüğü'nden talep edilen örnek veri seti üzerindeki analizleri, akademik çalışmalara destek olması açısından internetten erişime açık olan BMS-WebView-1 ve BMS-WebView-2 verileri üzerindeki analizleri ve Transact SQL kullanılarak hazırlanan algoritmaların bu örnek veriler üzerindeki karşılaştırmalı sonuçlarını içermektedir. Örnek veri setinden birliktelik kurallarının elde edilmesi için aday üretimli bir algoritma olan Apriori ile aday üretimsiz bir algoritma olan FP-Growth algoritmaları kullanılmıştır.

5.1 Örnek Veri Setleri

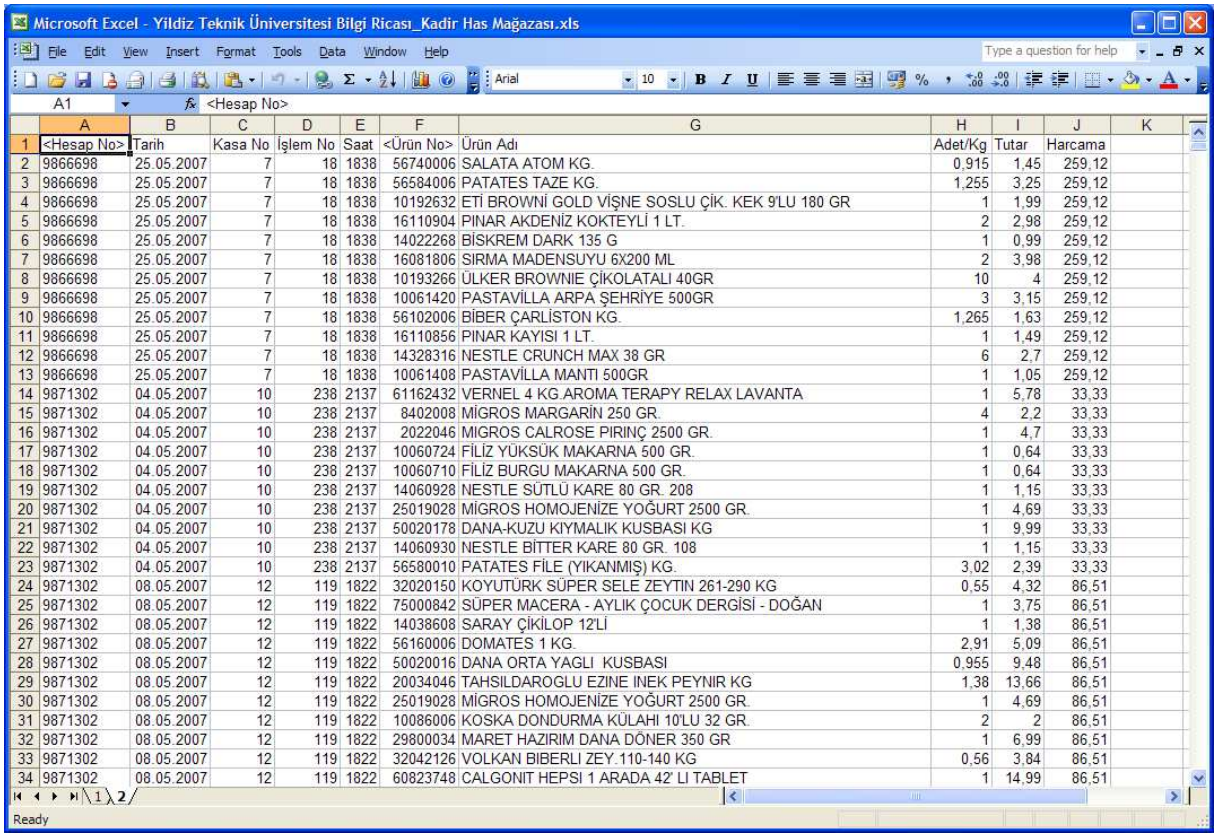
5.1.1 Migros Kadir Has Mağazası Veri Seti

Migros Kadir Has Mağazası'na ait örnek veri seti, 01.Mayıs.2007 ve 31.Mayıs.2007 tarihleri arasında elde edilmiş olan bir aylık market alışveriş verisini içermektedir. Talep sonrasında Microsoft Excel formatında paylaşılan 9,25 MB büyüklüğündeki veri 82902 adet kayıttan oluşmaktadır. Bu veriye ait örnek görüntüler Şekil 5.1 ve Şekil 5.2'de görülmektedir.


Microsoft Excel - Yildiz Teknik Üniversitesi Bilgi Ricası_Kadir Has Mağazası.xls

	A	B	C	D	E	F	G	H	I	J	K
1	<Hesap No>	Tarih	Kasa No	İşlem No	Saat	<Ürün No>	Ürün Adı	Adet/Kg	Tutar	Harcama	
2	314	20.05.2007	10	35	1134	10092908	BEBELAC YOĞURT KAYISI ANANAS PÜRESİ 2 X 125 G	1	3,45	165,08	
3	314	20.05.2007	10	35	1134	72431306	GÜLİZ-MİNİ FOIL BALON 9 "	1	3,5	165,08	
4	314	20.05.2007	10	35	1134	22023006	MİGROS SÜT 1 LT.	5	5,22	165,08	
5	314	20.05.2007	10	35	1134	50020168	DANA YAĞSIZ KUŞBAŞI	0,83	11,28	165,08	
6	314	20.05.2007	10	35	1134	56240006	HAVUÇ KG.	1,23	2,08	165,08	
7	314	20.05.2007	10	35	1134	16109664	DIMES ÜZÜM SUYU 1 LT.	2	5,38	165,08	
8	314	20.05.2007	10	35	1134	56606006	PATLICAN KEMER KG.	0,745	1,18	165,08	
9	314	20.05.2007	10	35	1134	54542006	MUZ İTHAL KG.	0,94	3,66	165,08	
10	314	20.05.2007	10	35	1134	10093470	ÜLKER HERO ARMUT-ŞEFTALİ-ANANAS 130 GR. KAVANOZ	2	3,78	165,08	
11	314	20.05.2007	10	35	1134	54420006	KAYISI KG.	1,07	3,73	165,08	
12	314	20.05.2007	10	35	1134	20034046	TAHSILDAROGLU EZINE INEK PEYNIR KG	0,695	8,69	165,08	
13	314	20.05.2007	10	35	1134	62061806	EVY BABY ISLAK HAVLU 72 YP	1	3,4	165,08	
14	314	20.05.2007	10	35	1134	74265460	KALEMTIRAŞ YILDIZ ŞEKİLLİ PLASTİK	1	1,5	165,08	
15	314	20.05.2007	10	35	1134	6625476	ŞEKER TANESİ TOZ ŞEKER POŞET AMB.3KG	1	6,1	165,08	
16	314	20.05.2007	10	35	1134	68641252	JOHNSON'S BABY SAMPUAN 800 ML.	1	9,9	165,08	
17	314	20.05.2007	10	35	1134	62200200	PRIMA ACTIVE JUMBO MAXI 7-18KG 72 PED	1	18,45	165,08	
18	314	20.05.2007	10	35	1134	78260818	PANASONIC OXYRIDE KALEM PİL 4 LÜ	1	8,25	165,08	
19	314	20.05.2007	10	35	1134	75005448	HÜRRİYET HAFTA SONU	1	0,5	165,08	
20	314	20.05.2007	10	35	1134	20034070	F.G.KLASIK INEK PEYNIRI	0,57	3,93	165,08	
21	314	20.05.2007	10	35	1134	10028246	KONYA UN 5 KG	1	4,5	165,08	
22	314	20.05.2007	10	35	1134	10227014	KURU PASTA TATLI KG.	0,31	4,62	165,08	
23	314	20.05.2007	10	35	1134	14020050	ETİ CİCİBEBE BİSKÜVİ 1 KG (200 GR BEDAVA)	1	5,99	165,08	
24	314	20.05.2007	10	35	1134	61162510	VERNEL SENSITIVE 2 KG.	1	3,5	165,08	
25	314	20.05.2007	10	35	1134	62221308	BÜTÇEM ULTRA HİJYENİK PED UZUN 8 Lİ	1	1,15	165,08	
26	314	20.05.2007	10	35	1134	16100094	TAMEK PLUS CAM % 100 ÜZÜM SUYU 1 LT.	1	3,39	165,08	
27	314	20.05.2007	10	35	1134	12122844	DR. OETKER HAMUR KAB. TOZU 15'Lİ PAKET 150 GR.	1	1,35	165,08	
28	314	20.05.2007	10	35	1134	10229008	EL MANTISI KIYMALI	0,565	6,16	165,08	
29	314	20.05.2007	10	35	1134	50020154	DANA-KUZU KIYMALIK RULO	1,04	10,39	165,08	
30	314	20.05.2007	10	35	1134	37980098	CIPURA KG (2/3)	1,206	10,84	165,08	
31	314	20.05.2007	10	35	1134	54278238	ELMA GRANY SMITH ITHAL KG.	0,9	4,49	165,08	
32	314	20.05.2007	10	35	1134	10112326	UNO BÜYÜMEK DİLİMLİ EKMEK 485 GR	1	2,5	165,08	
33	314	20.05.2007	10	35	1134	20202016	MİGROS KAŞAR PEYNİRİ 700 GR.	1	5,69	165,08	
34	314	20.05.2007	10	35	1134	12122846	DR. OETKER ŞEKERLİ VANİLİN 15'Lİ PAKET 75 GR.	1	1,35	165,08	

Şekil 5.1 Migros Kadir Has Mağazası'ndaki market alışverişlerine ait örnek veri seti - Sheet1 (1 - 60000 arası kayıtlar)



Microsoft Excel - Yildiz Teknik Üniversitesi Bilgi Ricası_Kadir Has Mağazası.xls

<Hesap No>	Tarih	Kasa No	İşlem No	Saat	<Ürün No>	Ürün Adı	Adet/Kg	Tutar	Harcama
9866698	25.05.2007	7	18	1838	56740006	SALATA ATOM KG.	0,915	1,45	259,12
9866698	25.05.2007	7	18	1838	56584006	PATATES TAZE KG.	1,255	3,25	259,12
9866698	25.05.2007	7	18	1838	10192632	ETİ BROWNİ GOLD VİŞNE SOSLU ÇİK. KEK 9'LU 180 GR	1	1,99	259,12
9866698	25.05.2007	7	18	1838	16110904	PINAR AKDENİZ KOKTEYLİ 1 LT.	2	2,98	259,12
9866698	25.05.2007	7	18	1838	14022268	BİSKREM DARK 135 G	1	0,99	259,12
9866698	25.05.2007	7	18	1838	16081806	SIRMA MADENSUYU 6X200 ML	2	3,98	259,12
9866698	25.05.2007	7	18	1838	10193266	ÜLKER BROWNIE ÇİKOLATALI 40GR	10	4	259,12
9866698	25.05.2007	7	18	1838	10061420	PASTAVİLLA ARPA ŞEHRİYE 500GR	3	3,15	259,12
9866698	25.05.2007	7	18	1838	56102006	BİBER ÇARLİSTON KG.	1,265	1,63	259,12
9866698	25.05.2007	7	18	1838	16110856	PINAR KAYISI 1 LT.	1	1,49	259,12
9866698	25.05.2007	7	18	1838	14328316	NESTLE CRUNCH MAX 38 GR	6	2,7	259,12
9866698	25.05.2007	7	18	1838	10061408	PASTAVİLLA MANTI 500GR	1	1,05	259,12
9871302	04.05.2007	10	238	2137	61162432	VERNEL 4 KG.AROMA TERAPY RELAX LAVANTA	1	5,78	33,33
9871302	04.05.2007	10	238	2137	8402008	MİGROS MARGARİN 250 GR.	4	2,2	33,33
9871302	04.05.2007	10	238	2137	2022046	MIGROS CALROSE PIRINÇ 2500 GR.	1	4,7	33,33
9871302	04.05.2007	10	238	2137	10060724	FİLİZ YÜKSÜK MAKARNA 500 GR.	1	0,64	33,33
9871302	04.05.2007	10	238	2137	10060710	FİLİZ BURGU MAKARNA 500 GR.	1	0,64	33,33
9871302	04.05.2007	10	238	2137	14060928	NESTLE SÜTLÜ KARE 80 GR. 208	1	1,15	33,33
9871302	04.05.2007	10	238	2137	25019028	MİGROS HOMOJENİZE YOĞURT 2500 GR.	1	4,69	33,33
9871302	04.05.2007	10	238	2137	50020178	DANA-KUZU KIYMALIK KUSBASI KG	1	9,99	33,33
9871302	04.05.2007	10	238	2137	14060930	NESTLE BİTTER KARE 80 GR. 108	1	1,15	33,33
9871302	04.05.2007	10	238	2137	56580010	PATATES FİLE (YIKANMIŞ) KG.	3,02	2,39	33,33
9871302	08.05.2007	12	119	1822	32020150	KOYUTÜRK SÜPER SELE ZEYTIN 261-290 KG	0,55	4,32	86,51
9871302	08.05.2007	12	119	1822	75000842	SÜPER MACERA - AYLIK ÇOCUK DERGİSİ - DOĞAN	1	3,75	86,51
9871302	08.05.2007	12	119	1822	14038608	SARAY ÇİKİLOP 12'Lİ	1	1,38	86,51
9871302	08.05.2007	12	119	1822	56160006	DOMATES 1 KG.	2,91	5,09	86,51
9871302	08.05.2007	12	119	1822	50020016	DANA ORTA YAGLI KUSBASI	0,955	9,48	86,51
9871302	08.05.2007	12	119	1822	20034046	TAHSILDAROGLU EZINE INEK PEYNIR KG	1,38	13,66	86,51
9871302	08.05.2007	12	119	1822	25019028	MİGROS HOMOJENİZE YOĞURT 2500 GR.	1	4,69	86,51
9871302	08.05.2007	12	119	1822	10086006	KOSKA DONDURMA KÜLAHI 10'LU 32 GR.	2	2	86,51
9871302	08.05.2007	12	119	1822	29800034	MARET HAZIRIM DANA DÖNER 350 GR	1	6,99	86,51
9871302	08.05.2007	12	119	1822	32042126	VOLKAN BİBERLİ ZEY.110-140 KG	0,56	3,84	86,51
9871302	08.05.2007	12	119	1822	60823748	CALGONIT HEPSI 1 ARADA 42' LI TABLET	1	14,99	86,51

Şekil 5.2 Migros Kadir Has Mağazası'ndaki market alışverişlerine ait örnek veri seti - Sheet2 (60001 - 82902 arası kayıtlar)

Migros verisinin her satırında müşteri hesap numarası, işlem tarihi, işlem yapılan kasa numarası, günlük işlem numarası, işlem saati, alınan ürün numarası, alınan ürünün adı, adet/kg, tutar ve ilgili işlem numarasında yapılmış toplam harcama bilgileri bulunmaktadır.

Veriye ait başlıca özellikler şunlardır:

- 82902 adet satırdan oluşmaktadır.
- 01.05.2007 Salı – 31.05.2007 Perşembe tarihleri arasındaki alışveriş bilgilerini içermektedir.
- 6401 adet hareketten (transaction – birlikte alınma işlemi) oluşmaktadır.
- 8939 adet üründen oluşmaktadır. Bu ürünlerden daha sonra 574 adet ürün kategorisi elde edilmiştir.
- 1273 adet farklı müşteri numarası ve 9 adet farklı kasa numarası yer almaktadır.
- En erken yapılan alışveriş saat 10:05'te (03.05.2007), en geç yapılan alışveriş saat 22:08'de (03.05.2007) gerçekleşmiştir.
- 720 adet farklı işlem numarası 1 – 1364 arasında değişmektedir.

5.1.2 BMS-WebView-1 ve BMS-WebView-2 Veri Setleri

BMS-WebView-1 ve BMS-WebView-2, veri madenciliği alanında yapılan akademik çalışmalara destek olması açısından, internetten erişime açık olan, iki e-ticaret web sitesinden elde edilmiş satış verileridir. Her iki veri de metin dosyasında bulunmakta ve satış fiş numarası ve ürün kodunu içeren iki kolondan oluşmaktadır. Şekil 5.3 ve Şekil 5.4'te bu iki verikümesine ait küçük birer örnek görülmektedir.

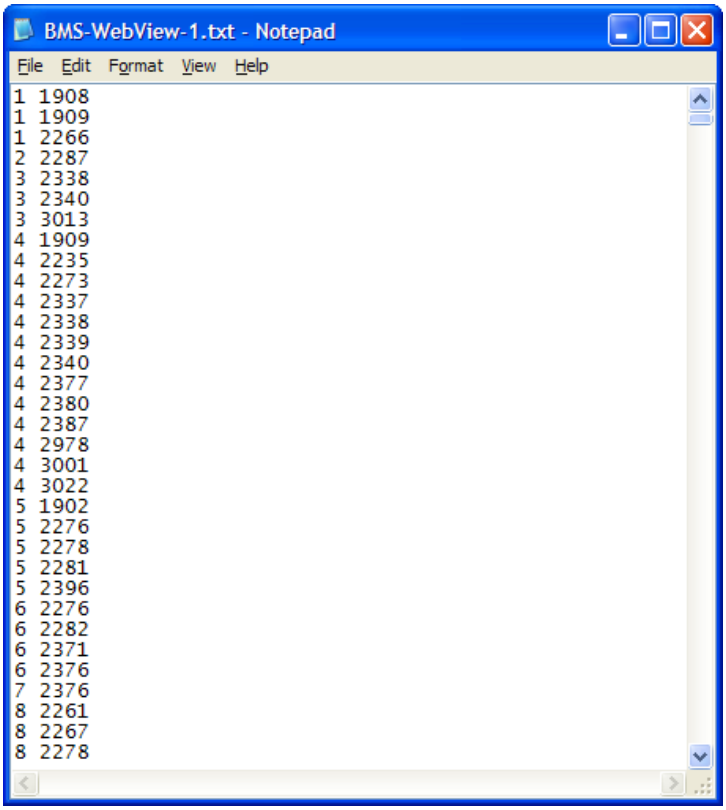
Bu iki veriden de birliktelik kuralları elde edileceğinden dolayı, verilerin veritabanına aktarılmasının ardından, sadece tek ürün içeren fiş numaralarına ait satırlar silinmiş olup, bölüm 3.1.2'de bahsedilen verilerin hazırlanması ile ilgili yöntemler uygulanmıştır.

Veritabanına aktarılmış BMS-WebView-1 verisine ait başlıca özellikler şunlardır:

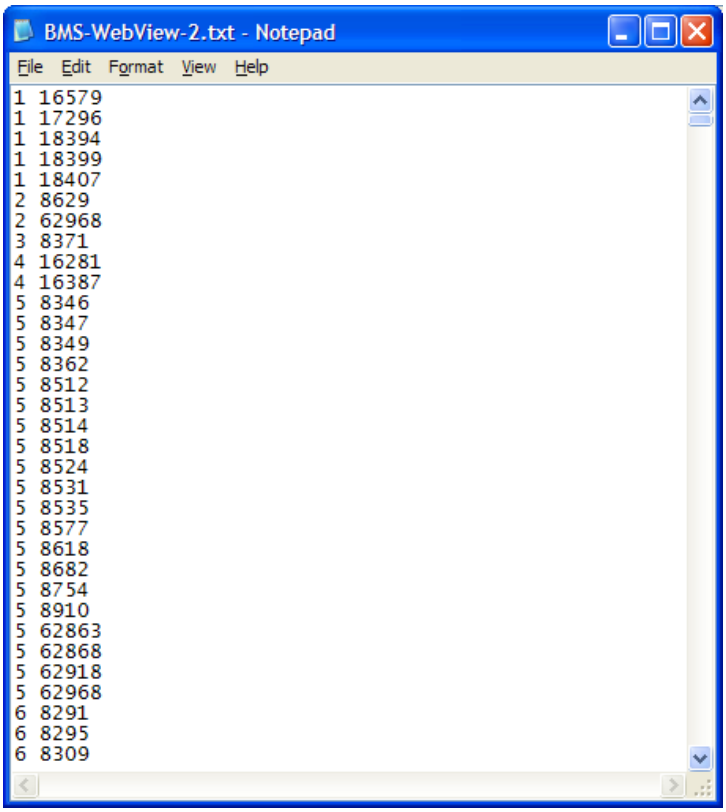
- 116704 adet satırdan oluşmaktadır.
- 26667 adet hareketten (transaction – birlikte alınma işlemi) oluşmaktadır.
- 497 adet farklı ürün içermektedir.
- Fiş numarası aralığı 1 – 60067'dir.
- Ürün Kodu aralığı 1902 – 4929'dur.
- 2498 adet ile en fazla fişte geçen ürün 2399'dur.
- 1204 adet ile en fazla fişte geçen 2-nesnekümesi {3996, 4002}'dir.

BMS-WebView-2 verisine ait başlıca özellikler ise şunlardır:

- 333385 adet satırdan oluşmaktadır.
- 52619 adet hareketten (transaction – birlikte alınma işlemi) oluşmaktadır.
- 3335 adet farklı ürün içermektedir.
- Fiş numarası aralığı 1 – 79455'dir.
- Ürün Kodu aralığı 8284 – 80004'tür.
- 3412 adet ile en fazla fişte geçen ürün 8284'tür.
- 1501 adet ile en fazla fişte geçen 2-nesnekümesi {8307, 8309}'dur.



Şekil 5.3 BMS-WebView-1 örnek veri seti – 149639 satır

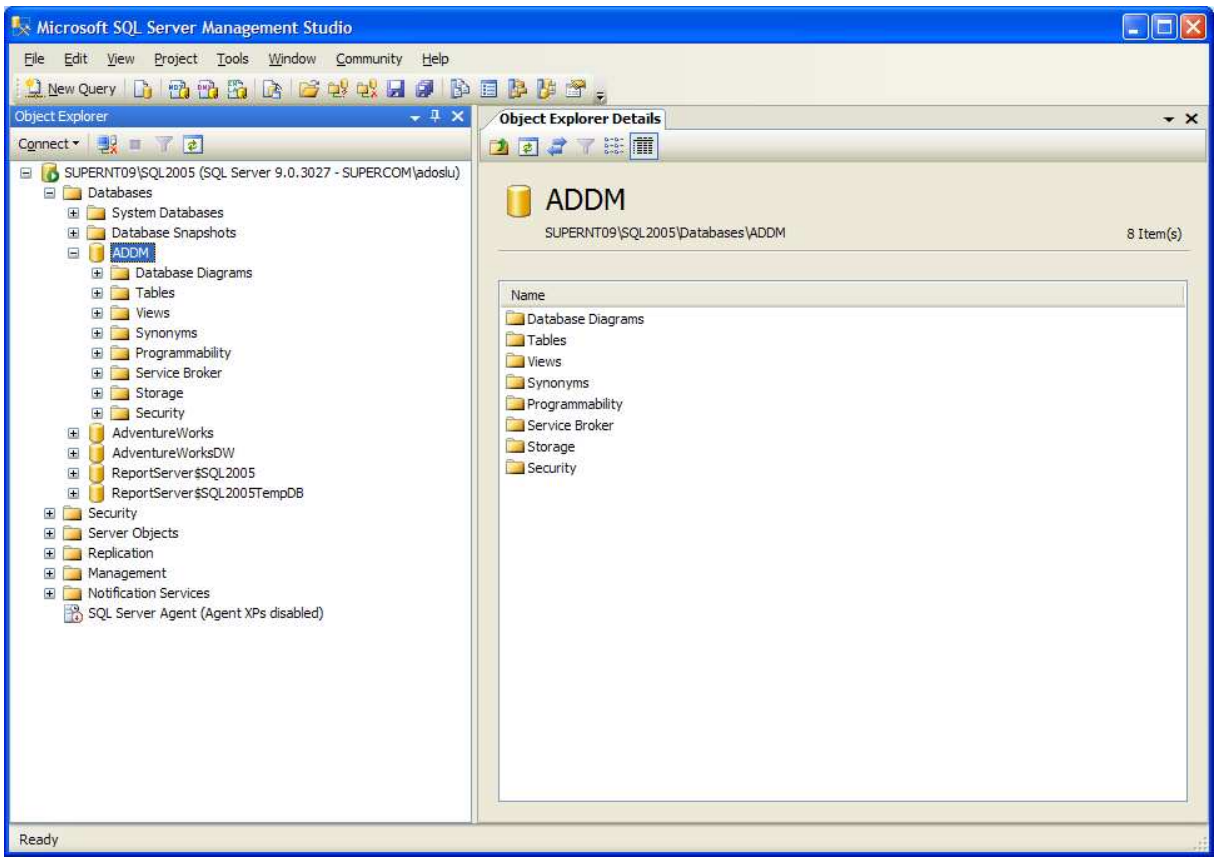


Şekil 5.4 BMS-WebView-2 örnek veri seti – 358278 satır

5.2 ADDM Veritabanı

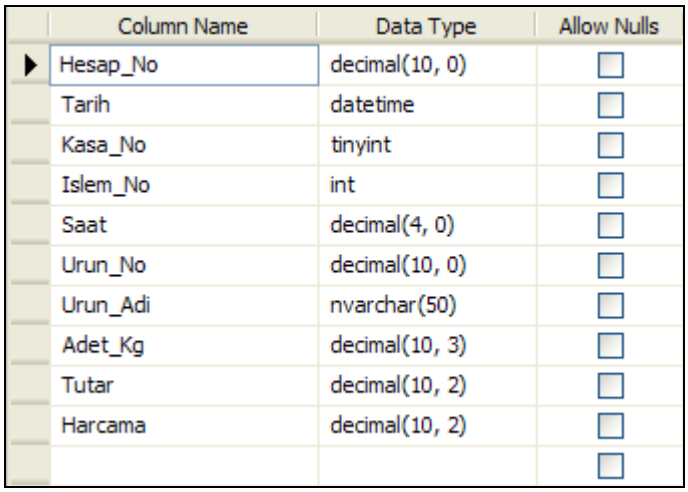
5.2.1 Veritabanının Oluşturulması ve Örnek Veri Setlerinin Veritabanına Alınması

Excel formatındaki Migros verisinin ve txt uzantılı BMS verilerinin üzerinde işlem ve sorgulamalar yapıp, Transact SQL dili ile yazılmış algoritmalardan sonuçlar alabilmek adına Microsoft SQL Server 2005'e ait SQL Server Management Studio (SSMS) üzerinde ADDM veritabanı oluşturulmuştur. Veritabanının ismi "Ayhan Döşlü Data Mining" kelimelerinin baş harflerinden gelmektedir.



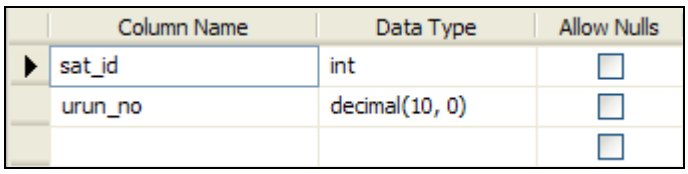
Şekil 5.5 SQL Server Management Studio'da oluşturulmuş olan ADDM veritabanı

Excel formatındaki Migros verisini ve txt uzantılı BMS verilerini ADDM veritabanına alabilmek için öncelikle, Excel'deki kolon sırasını ve isimlendirmelerinin benzerini içerecek şekilde t_migros_kadir_has isimli bir tablo ve BMS-WebView-1 ve BMS-WebView-2 verilerini içerecek olan t_bms_webview_1 ve t_bms_webview_2 tabloları oluşturulmuştur. t_migros_kadir_has tablosunda Hesap_No, Tarih, Kasa_No, Islem_No, Saat, Urun_No, Urun_Adi, Adet_Kg, Tutar, Harcama isimli kolonlar yer almaktadır. Yapıları benzer olan t_bms_webview_1 ve t_bms_webview_2 tablolarında ise sat_id ve urun_no kolonları bulunmaktadır. Tabloların yapısı Şekil 5.6 ve Şekil 5.7'de görülmektedir.



Column Name	Data Type	Allow Nulls
Hesap_No	decimal(10, 0)	☐
Tarih	datetime	☐
Kasa_No	tinyint	☐
Islem_No	int	☐
Saat	decimal(4, 0)	☐
Urun_No	decimal(10, 0)	☐
Urun_Adi	nvarchar(50)	☐
Adet_Kg	decimal(10, 3)	☐
Tutar	decimal(10, 2)	☐
Harcama	decimal(10, 2)	☐
		☐

Şekil 5.6 ADDM veritabanındaki t_migros_kadir_has tablosunun yapısı

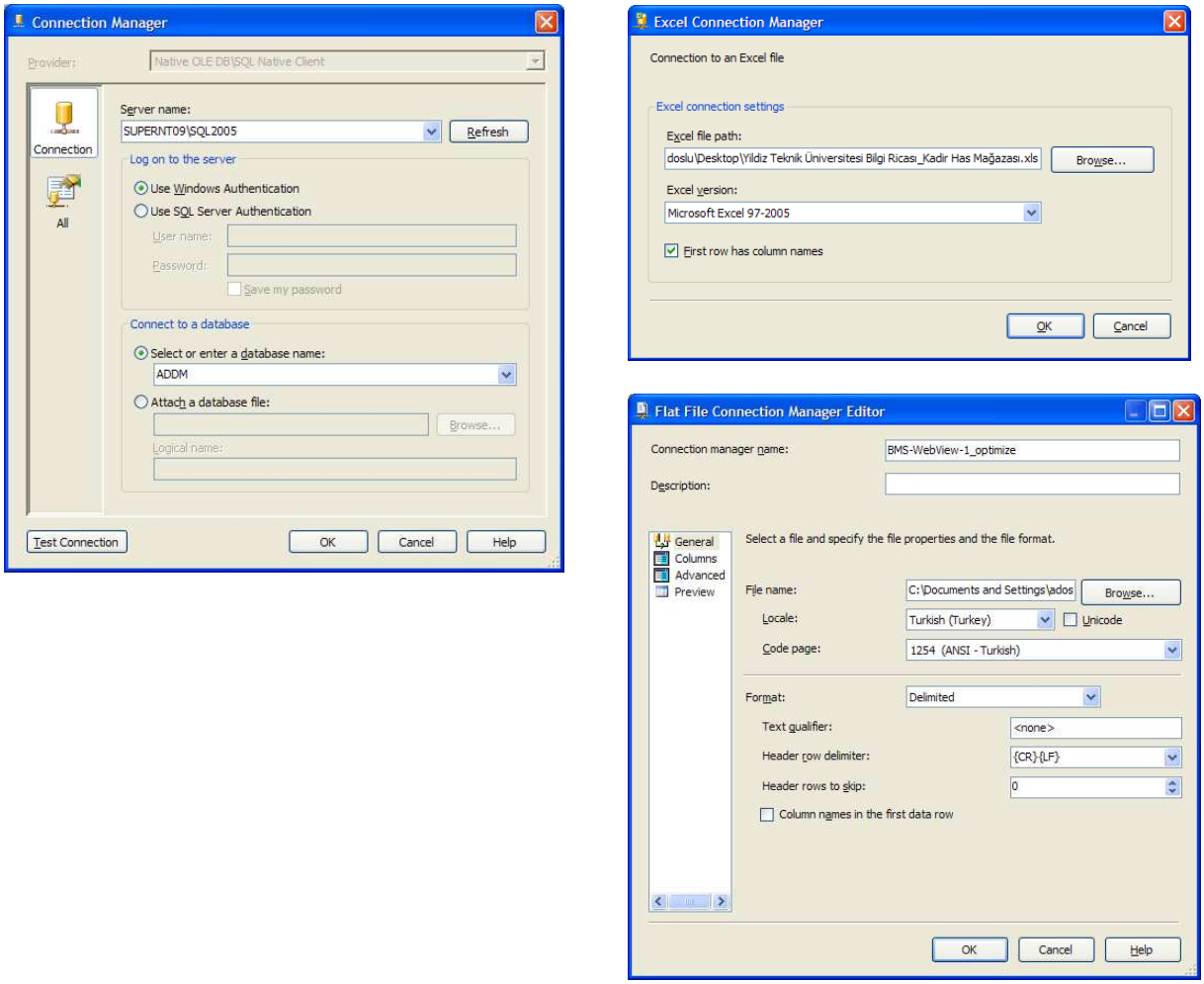


Column Name	Data Type	Allow Nulls
sat_id	int	☐
urun_no	decimal(10, 0)	☐
		☐

Şekil 5.7 ADDM veritabanındaki t_bms_webview_1 ve t_bms_webview_2 tablolarının yapısı

Oluşturulan t_migros_kadir_has tablosuna Excel formatındaki verinin, t_bms_webview_1 ve t_bms_webview_2 tablolarına da txt uzantılı verilerin atılabilmesi için, farklı kaynaklar arasında veri aktarımına imkan veren (ETL – Extract, Transform, Load) Microsoft SQL Server Integration Services (SSIS) kullanılmıştır.

Yapılacak aktarım işlemini tasarlamak için SQL Server Business Intelligence Development Studio'da ADDMSSIS isimli bir Integration Services Project oluşturulmuştur. Proje içinde SSIS Packages altında oluşturulan pMigrosSatis.dtsx'te, Excel ve SQL Server 2005 Veritabanı bağlantılarını sağlamak için bağlantı nesneleri oluşturulmuştur. Aynı şekilde, pBMS.dtsx içinde de Flat File (txt, csv, ... uzantılı dosyalar için) ve SQL Server 2005 Veritabanı bağlantılarını sağlamak için bağlantı nesneleri oluşturulmuştur.

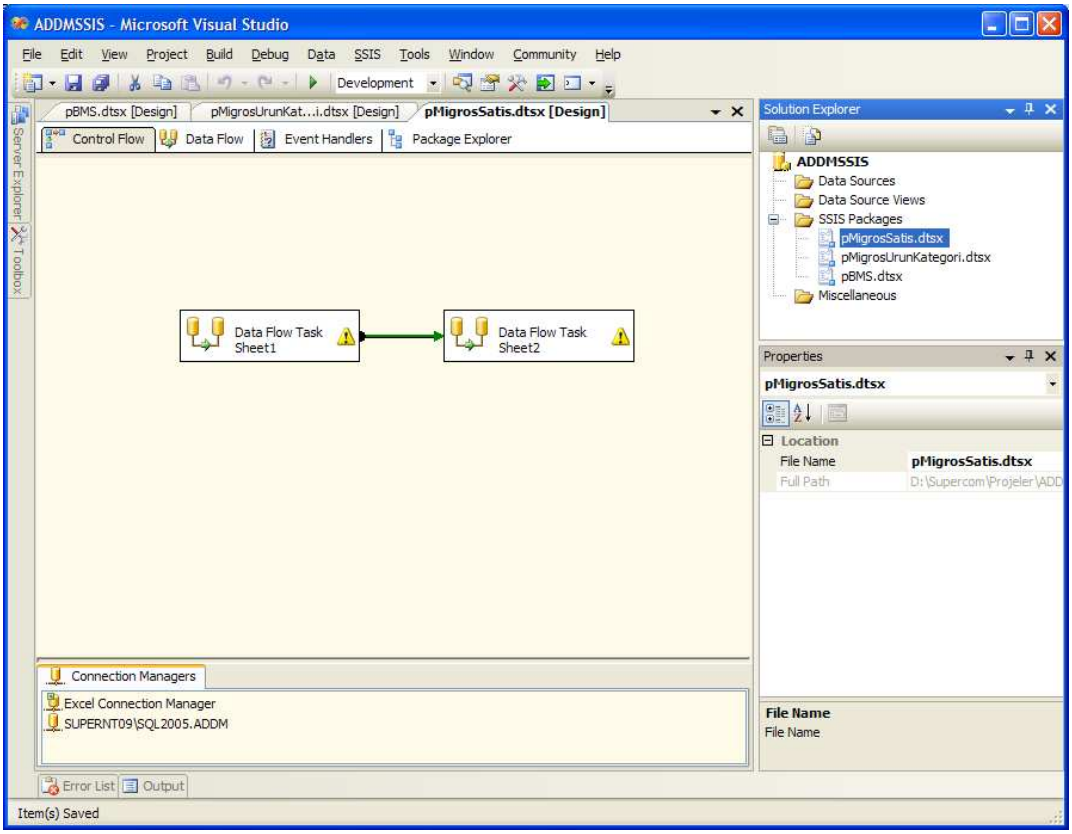


Şekil 5.8 ADDMSSIS projesinde yer alan
Excel, Flat File ve SQL Server 2005 Veritabanı bağlantı nesnelerinin yapılandırması

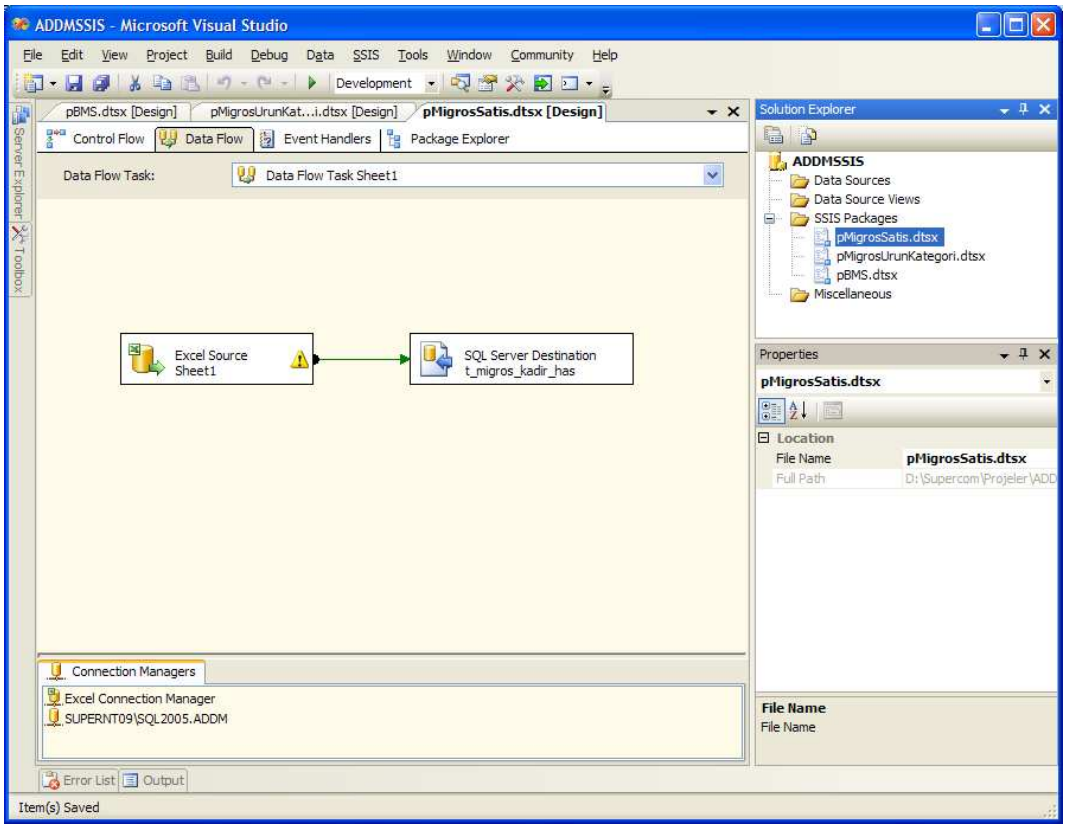
Ardından, bu bağlantı nesnelerini kullanarak Excel'deki Migros örnek veri setinin bulunduğu Sheet1 ve Sheet2'nin t_migros_kadir_has tablosuna atılması için iki adet Data Flow Task ve içerikleri tasarlanmıştır. pMigrosSatis.dtsx package'ının yapısı Şekil 5.9, Şekil 5.10 ve Şekil 5.11'de verilmiştir.

Metin formatındaki BMS veri setlerini t_bms_webview_1 ve t_bms_webview_2 tablolarına almak için ise, pBMS.dtsx package'ı tasarlanmıştır ve yapısı Şekil 5.12, Şekil 5.13 ve Şekil 5.14'te görülmektedir.

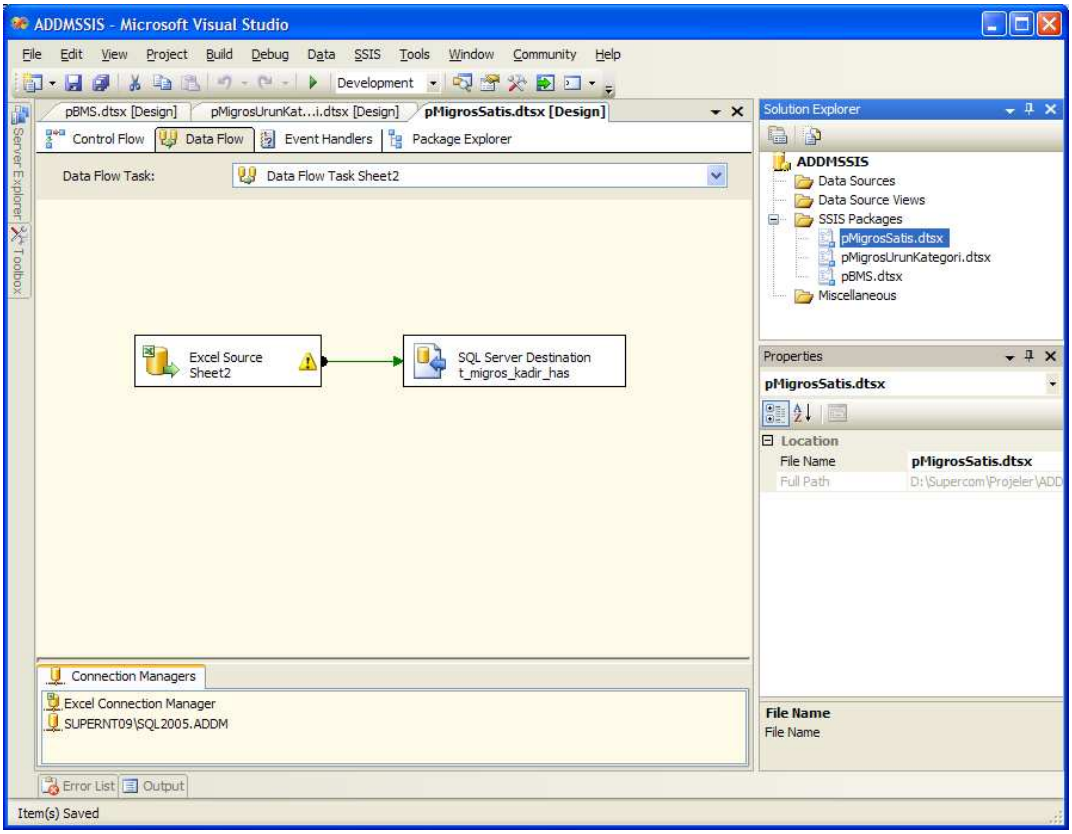
pMigrosSatis.dtsx ve pBMS.dtsx, tasarlandıktan sonra sırasıyla çalıştırılmıştır ve Excel dökümanındaki veriler t_migros_kadir_has tablosuna, txt uzantılı dökümanlardaki veriler t_bms_webview_1 ve t_bms_webview_2 tablolarına alınmıştır.



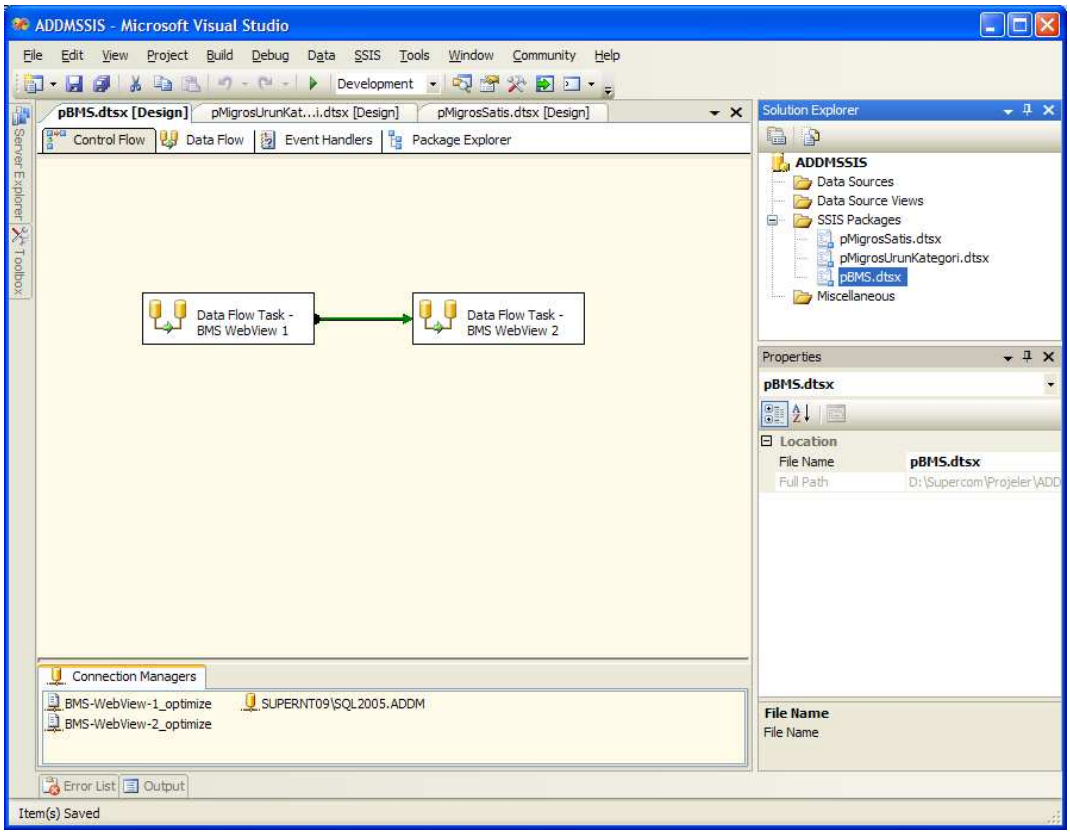
Şekil 5.9 pMigrosSatis.dtsx package'ı içinde yer alan Data Flow Task'ler ve bağlantıları



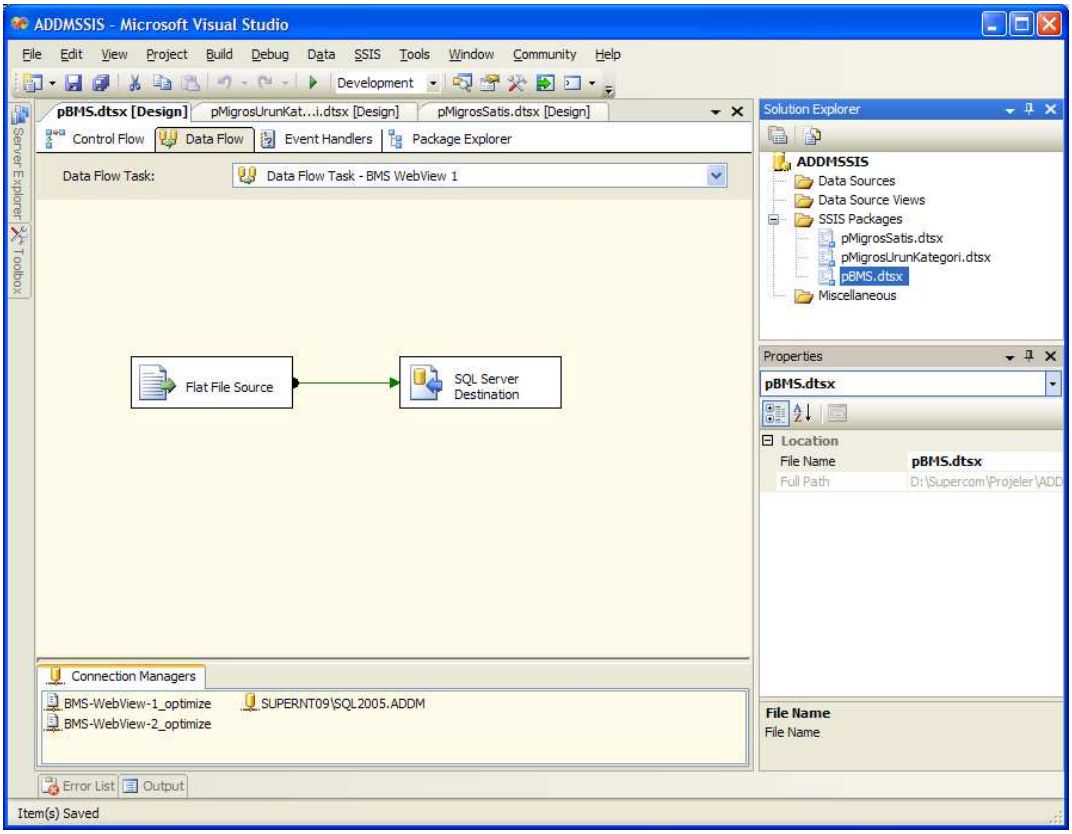
Şekil 5.10 pMigrosSatis.dtsx'teki Data Flow Task Sheet1'in içeriği



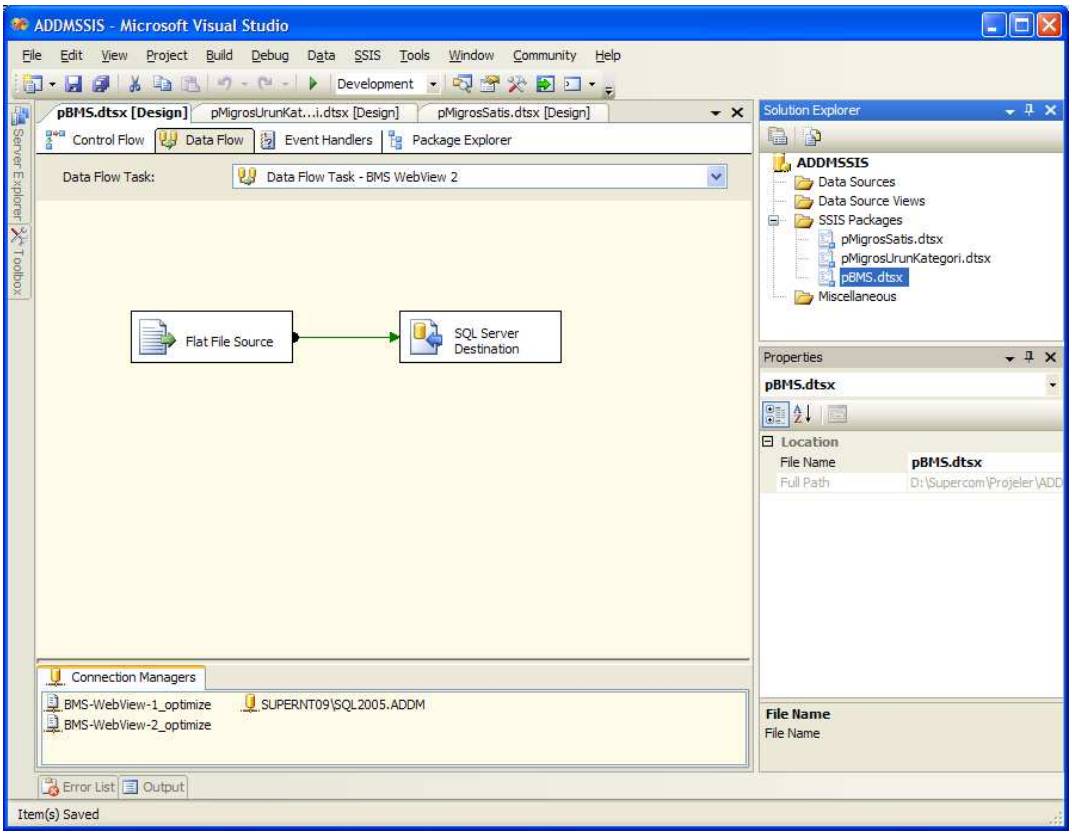
Şekil 5.11 pMigrosSatis.dtsx'teki Data Flow Task Sheet2'nin içeriği



Şekil 5.12 pBMS.dtsx package'ı içinde yer alan Data Flow Task'ler ve bağlantıları



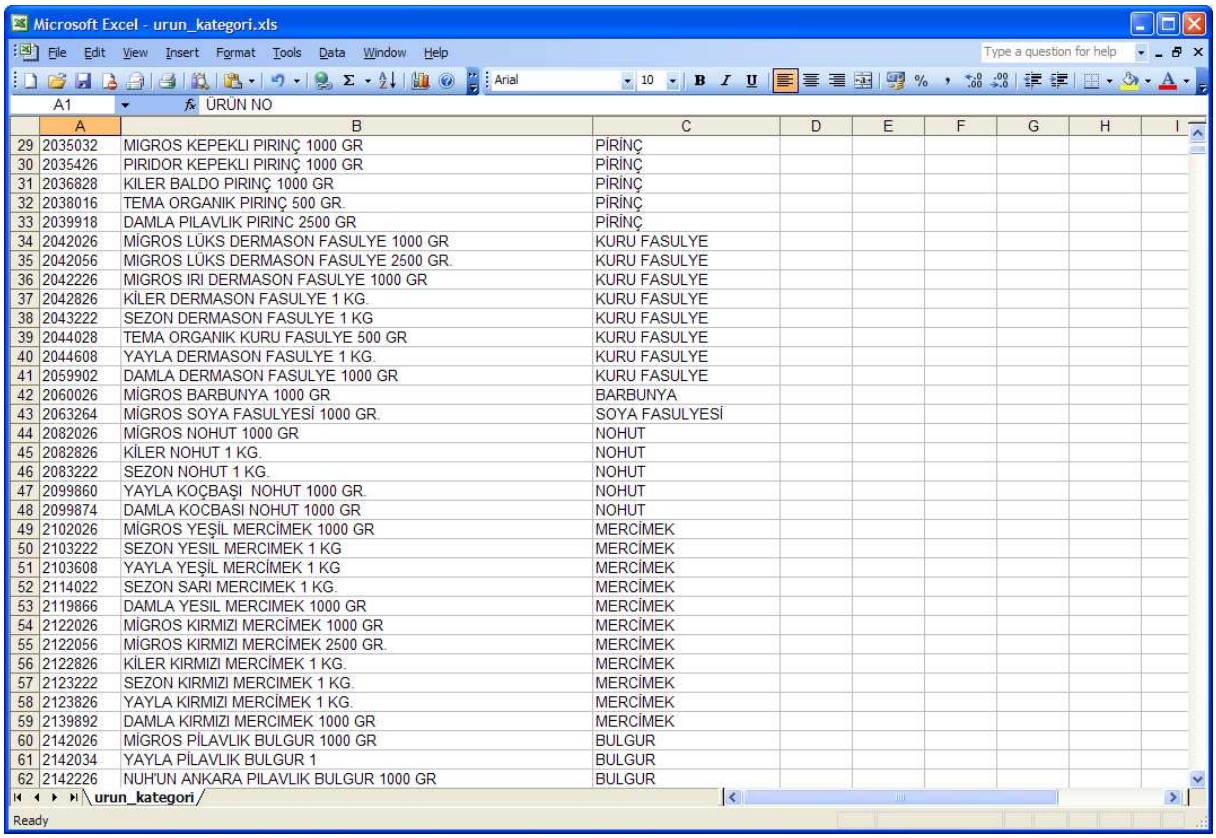
Şekil 5.13 pBMS.dtsx'teki Data Flow Task - BMS WebView 1'in içeriği



Şekil 5.14 pBMS.dtsx'teki Data Flow Task - BMS WebView 2'nin içeriği

5.2.2 Migros Veri Setindeki Ürünlerin Kategorik Hale Getirilmesi ve Veritabanına Alınması

Migros veri setinden daha anlamlı birliktelik kuralları elde edebilmek için 8939 adet ürünün kategorik hale getirilmesi gerekmektedir. Bundan dolayı, ürünler bir Excel dökümanında ürün numarasına göre dizilerek kategorik hale getirilmiştir. Her bir ürün tek tek gözden geçirilmiştir ve ilgili ürünün yanına Ürün Kategori Adı yazılmıştır. Şekil 5.15'de dökümanın örnek ekran görüntüsü bulunmaktadır.



	A	B	C
29	2035032	MIGROS KEPEKLI PIRINÇ 1000 GR	PİRİNÇ
30	2035426	PIRIDOR KEPEKLI PIRINÇ 1000 GR	PİRİNÇ
31	2036828	KILER BALDO PIRINÇ 1000 GR	PİRİNÇ
32	2038016	TEMA ORGANIK PIRINÇ 500 GR.	PİRİNÇ
33	2039918	DAMLA PILAVLIK PIRINC 2500 GR	PİRİNÇ
34	2042026	MİGROS LÜKS DERMASON FASULYE 1000 GR	KURU FASULYE
35	2042056	MIGROS LÜKS DERMASON FASULYE 2500 GR.	KURU FASULYE
36	2042226	MIGROS IRI DERMASON FASULYE 1000 GR	KURU FASULYE
37	2042826	KİLER DERMASON FASULYE 1 KG.	KURU FASULYE
38	2043222	SEZON DERMASON FASULYE 1 KG	KURU FASULYE
39	2044028	TEMA ORGANIK KURU FASULYE 500 GR	KURU FASULYE
40	2044608	YAYLA DERMASON FASULYE 1 KG.	KURU FASULYE
41	2059902	DAMLA DERMASON FASULYE 1000 GR	KURU FASULYE
42	2060026	MİGROS BARBUNYA 1000 GR	BARBUNYA
43	2063264	MİGROS SOYA FASULYESİ 1000 GR.	SOYA FASULYESİ
44	2082026	MİGROS NOHUT 1000 GR	NOHUT
45	2082826	KİLER NOHUT 1 KG.	NOHUT
46	2083222	SEZON NOHUT 1 KG.	NOHUT
47	2099860	YAYLA KOÇBAŞI NOHUT 1000 GR.	NOHUT
48	2099874	DAMLA KOCBASI NOHUT 1000 GR	NOHUT
49	2102026	MİGROS YEŞİL MERCİMEK 1000 GR	MERCİMEK
50	2103222	SEZON YESIL MERCIMEK 1 KG	MERCİMEK
51	2103608	YAYLA YEŞİL MERCİMEK 1 KG	MERCİMEK
52	2114022	SEZON SARI MERCIMEK 1 KG.	MERCİMEK
53	2119866	DAMLA YESIL MERCIMEK 1000 GR	MERCİMEK
54	2122026	MİGROS KIRMIZI MERCİMEK 1000 GR	MERCİMEK
55	2122056	MİGROS KIRMIZI MERCİMEK 2500 GR.	MERCİMEK
56	2122826	KİLER KIRMIZI MERCİMEK 1 KG.	MERCİMEK
57	2123222	SEZON KIRMIZI MERCIMEK 1 KG.	MERCİMEK
58	2123826	YAYLA KIRMIZI MERCİMEK 1 KG.	MERCİMEK
59	2139892	DAMLA KIRMIZI MERCIMEK 1000 GR	MERCİMEK
60	2142026	MİGROS PİLAVLIK BULGUR 1000 GR	BULGUR
61	2142034	YAYLA PİLAVLIK BULGUR 1	BULGUR
62	2142226	NUH'UN ANKARA PILAVLIK BULGUR 1000 GR	BULGUR

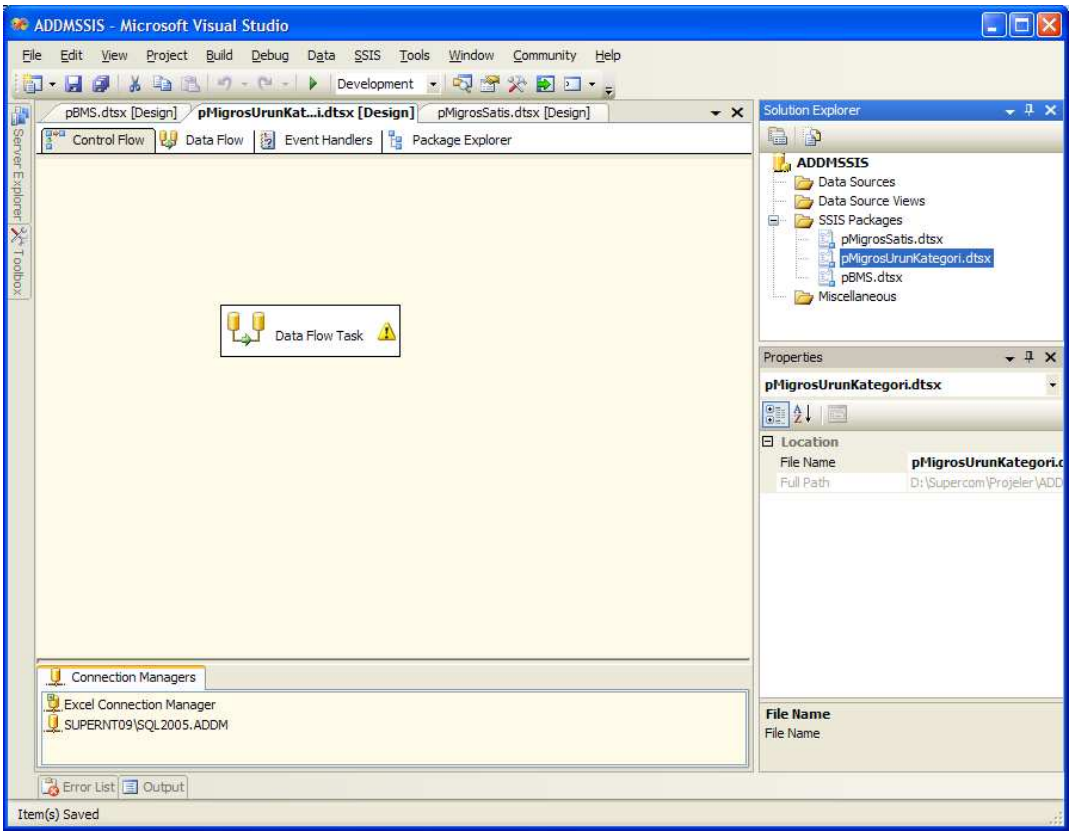
Şekil 5.15 8939 adet ürünün kategorik hale getirilmesi

Oluşturulan kategorik ürünler listesini ADDM veritabanında tutmak için t_urun isimli bir tablo tasarlanmıştır.

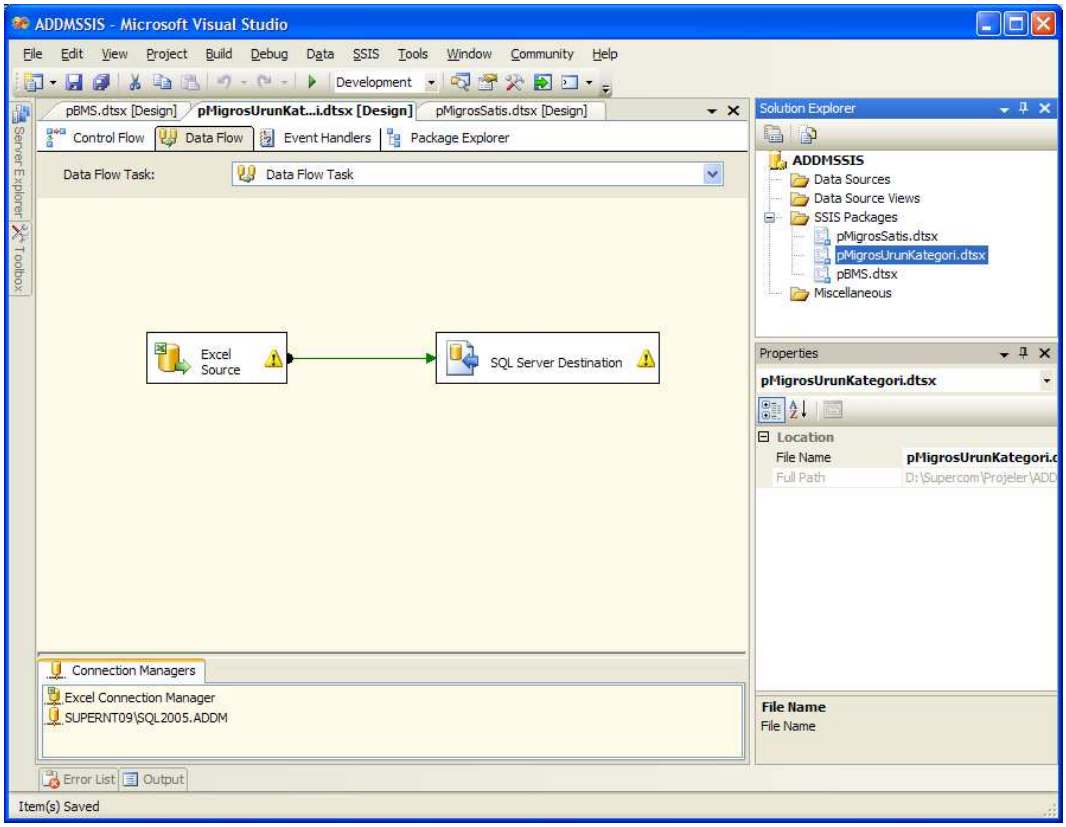
	Column Name	Data Type	Allow Nulls
🔑	urun_no	decimal(10, 0)	☐
	urun_ad	nvarchar(50)	☐
	urk_id	int	☑
			☐

Şekil 5.16 ADDM veritabanındaki t_urun tablosunun yapısı

Daha sonra, Excel'deki örnek veri setinin t_migros_kadir_has tablosuna alınmasında olduğu gibi, ADDMSSIS projesi altında pMigrosUrunKategori.dtsx isimli yeni bir package oluşturulmuştur.



Şekil 5.17 pMigrosUrunKategori.dtsx package'ı içinde yer alan Data Flow Task ve bağlantıları

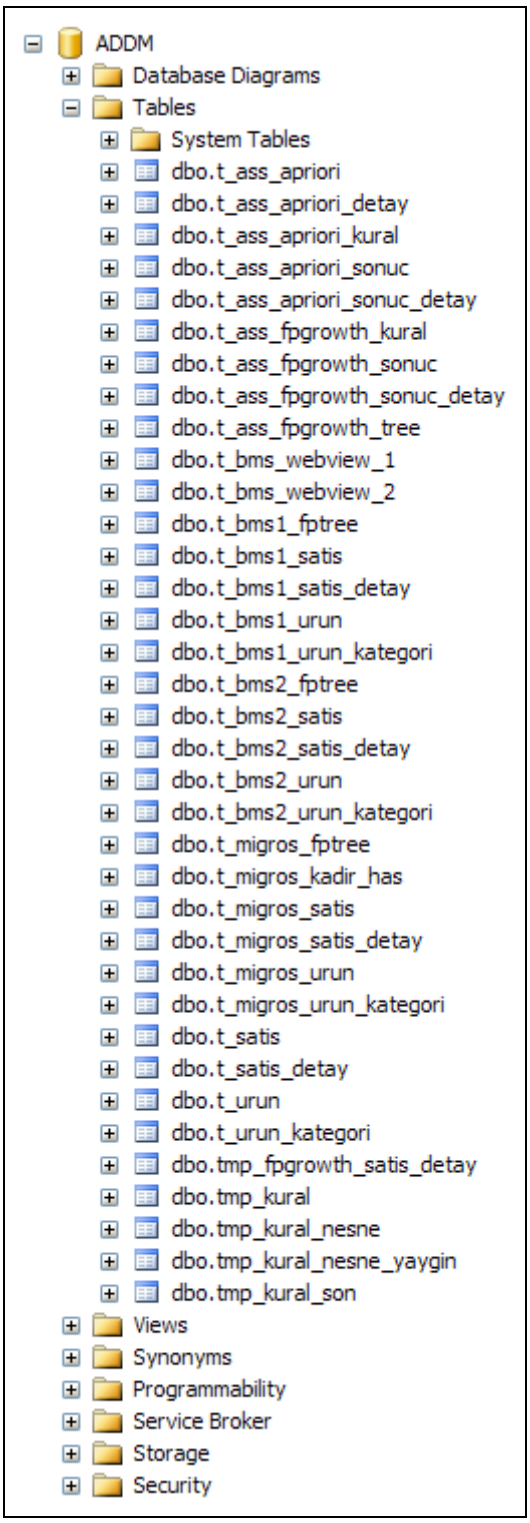


Şekil 5.18 pMigrosUrunKategori.dtsx'teki Data Flow Task'ın içeriği

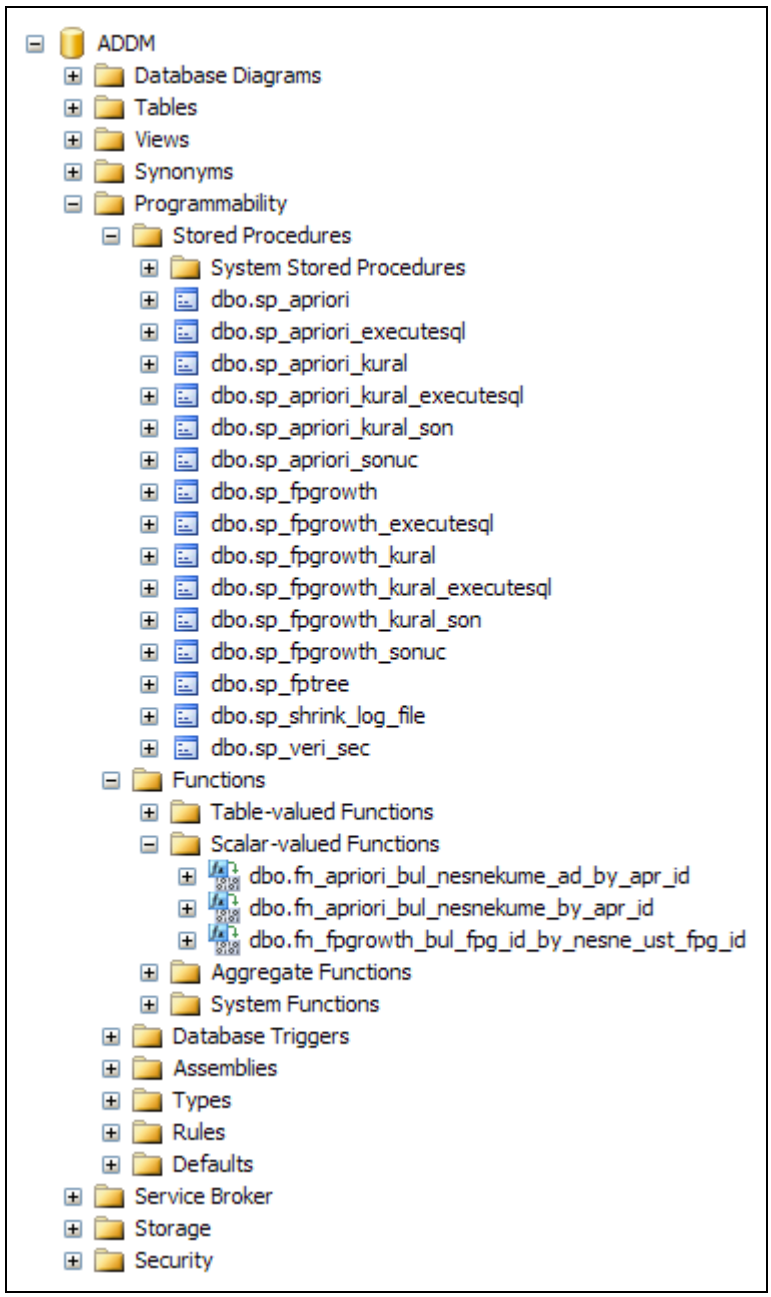
pMigrosUrunKategori.dtsx tasarlandıktan sonra çalıştırılmıştır ve Excel'deki kategorik ürün verileri t_urun tablosuna alınmıştır. Böylelikle, 8939 adet ürün 574 adet kategoriye ayrılarak, daha anlamlı birliktelikler ve kurallar oluşturulmuştur.

5.2.3 ADDM Veritabanının Yapısı

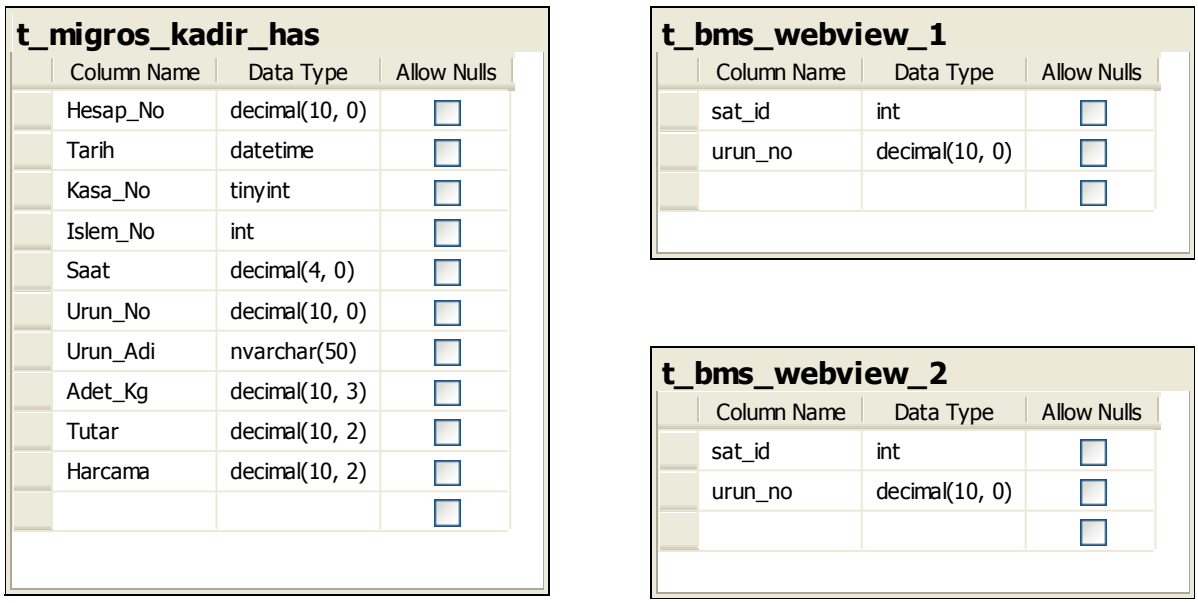
ADDM veritabanı D:\SQLData dizini altında bulunan, Primary filegroup unu kullanan ADDM_Data.mdf ve log bilgilerinin yer aldığı ADDM_Log.ldf dosyalarında tutulmaktadır. Veritabanında kullanılan veritabanı nesneleri belirli bir yapıya göre adlandırılmıştır. Tablolar için “t_”, stored procedure lar için “sp_”, fonksiyonlar için “fn_”, indeksler için “IX_”, ..., vb. ön ekleri kullanılmıştır. Şekil 5.19 ve Şekil 5.20’de, tasarlanmış tablo, stored procedure ve fonksiyonlar görülmektedir. Ayrıca Şekil 5.21 ile Şekil 5.27 arasında, tabloların kullanım alanlarına göre gruplandırılmış olarak bulunan tablo yapıları ve oluşturulmuş ilişkiler ile örnek stored procedure ve fonksiyonlardan kesitler yer almaktadır.



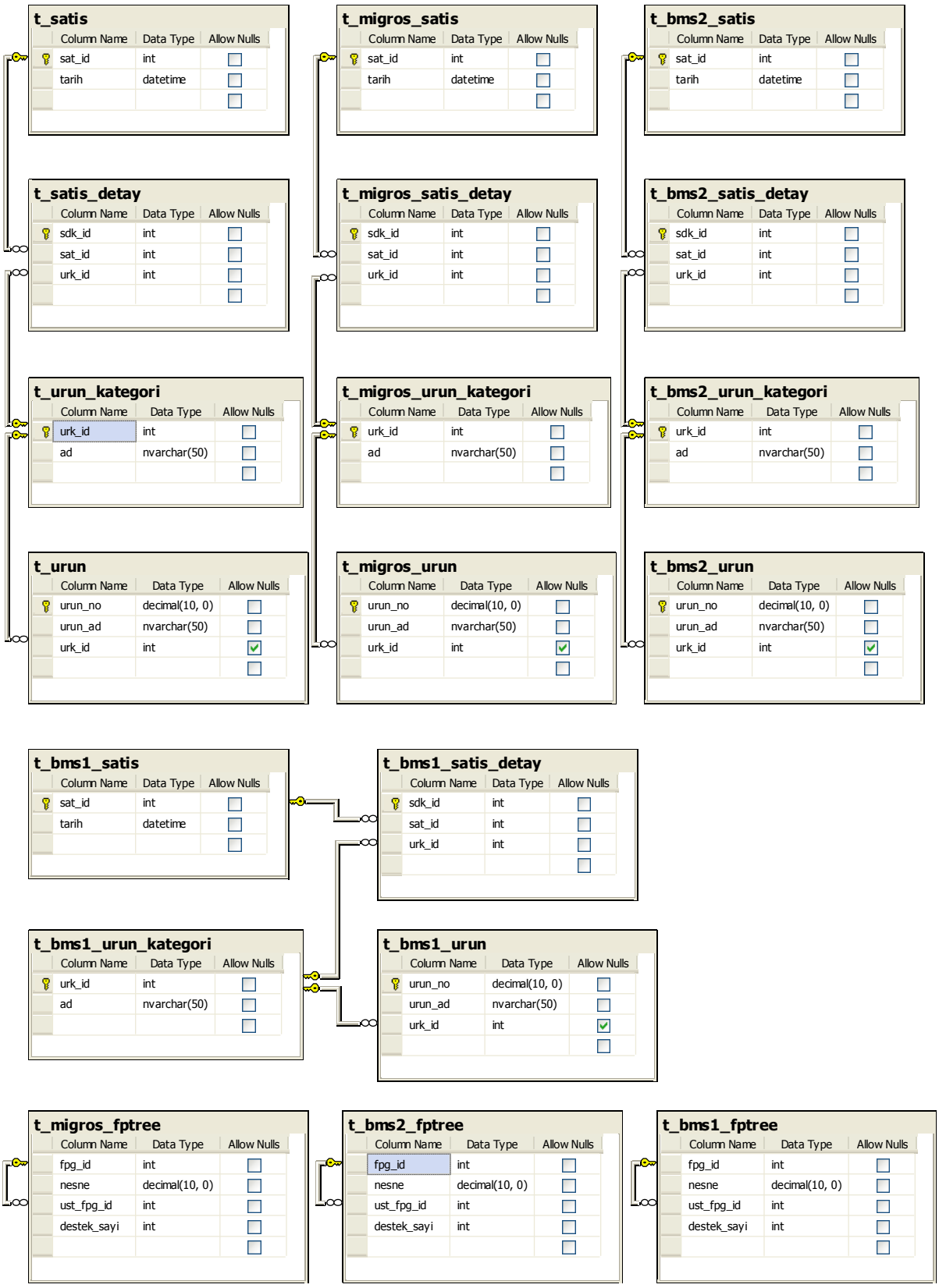
Şekil 5.19 ADDM veritabanı tabloları (36 adet)



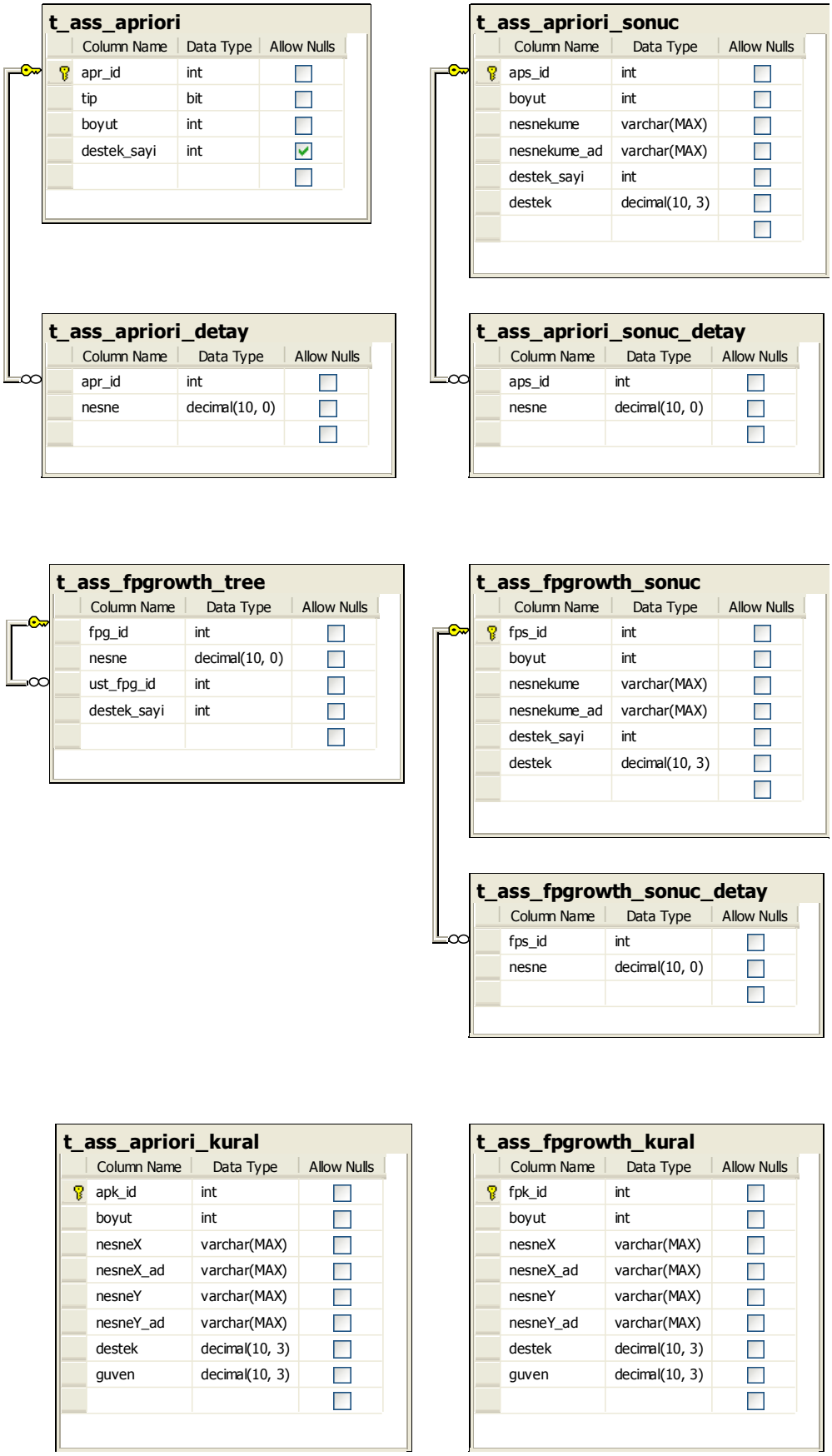
Şekil 5.20 ADDM veritabanı stored procedure ve fonksiyonları



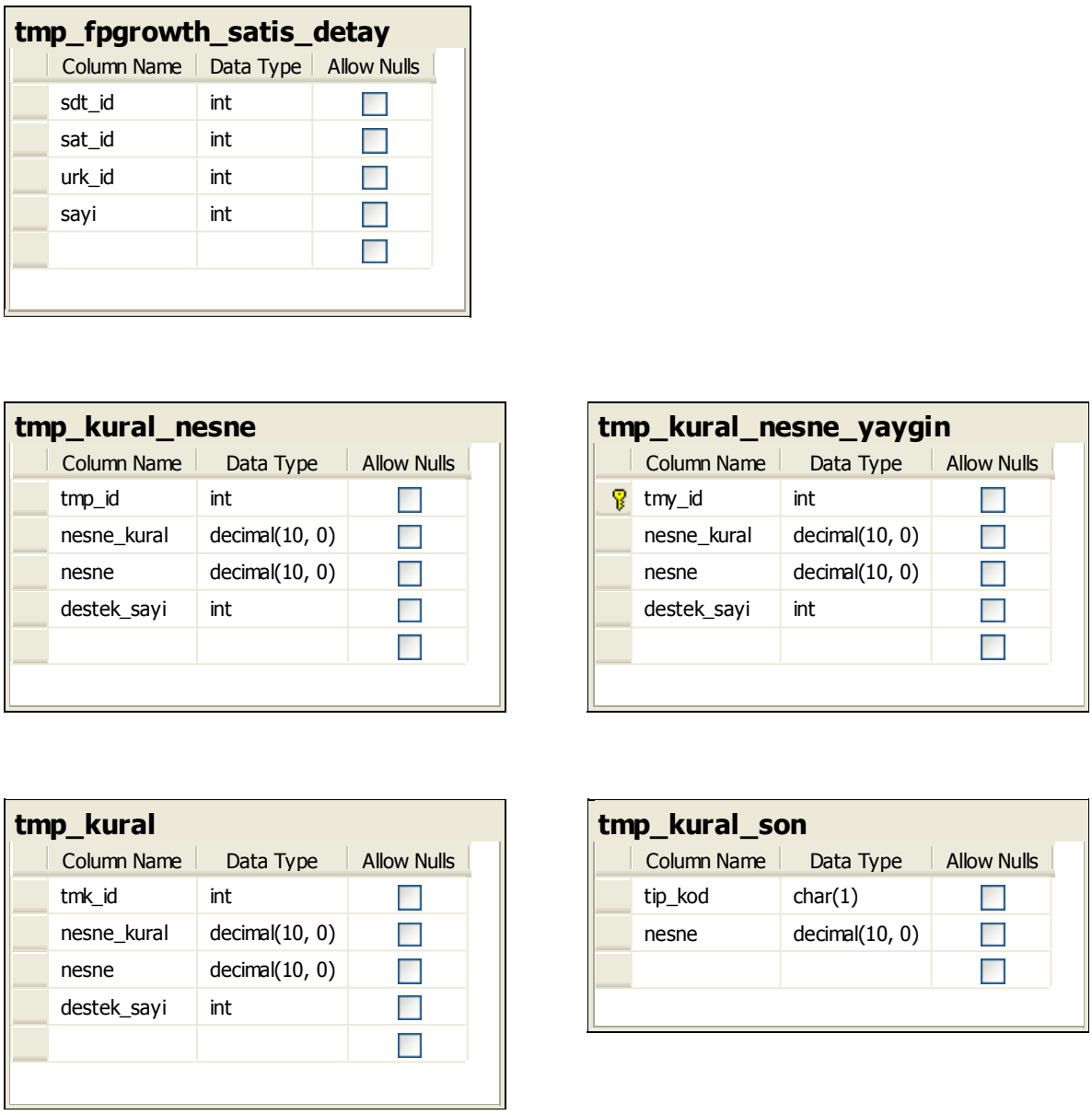
Şekil 5.21 Excel ve metin dosyalarındaki bilgilerin aktarıldığı, ham verilerin bulunduğu tablolar



Şekil 5.22 Ham verilerin bulunduğu tablolardan türetilmiş, algoritmalara veri sağlayan tablolar



Şekil 5.23 Algoritmaların çalışması esnasında kullanılan, sonuçların ve birliktelik kurallarının bulunduğu tablolar



Şekil 5.24 Algoritmaların çalışması esnasında ve birliktelik kuralları oluştururken kullanılan geçici tablolar

```
    select
        @ln_sayi = count(apr_id)
    from
        dbo.t_ass_apriori
    where
        boyut = @ln_boyut


    if @ln_sayi > 1 -- bir adet n boyutlu nesnekümesinden n+1 nesnekümesi üretilmeyeceği için "> 1"
    begin

        -- sınırsız sayıda Cx ve Lx oluşması için..
        while @ln_sayi > 1
        begin

            set @ln_boyut = @ln_boyut + 1

            exec dbo.sp_apriori_executesql @ln_boyut, @pn_ilk_gun, @pn_son_gun, @ln_min_sup

            select
                @ln_sayi = count(apr_id)
            from
                dbo.t_ass_apriori
            where
                boyut = @ln_boyut

        end

    end

end

exec dbo.sp_apriori_sonuc
```

Şekil 5.25 sp_apriori stored procedure'ünden bir kesit

```
set @ls_SQL = @ls_strDeclare + ' ' + @ls_strSet + ' ' +
              @ls_str1 +
              ' select ' +
                      '@ln_destek_sayi = isnull(sum(tmp1.destek_sayi), 0) ' +
                'from ' +
                      @ls_str2 +
                'where ' +
                      @ls_str3 +
                'having ' +
                      'sum(tmp1.destek_sayi) >= ' + cast(@pn_min_sup as varchar) + ' ' +

                'if isnull(@ln_destek_sayi, 0) > 0 ' +
                'begin ' +

                      'truncate table dbo.tmp_kural ' +

                      'insert into ' +
                            'dbo.tmp_kural ' +
                            '( ' +
                            ' nesne_kural ' +
                            ',nesne ' +
                            ',destek_sayi ' +
                            ') ' +
                      'select ' +
                            ' ' + cast(@pn_kural_nesne as varchar) + ' as nesne_kural ' +
                            ',' + cast(@pn_kural_nesne as varchar) + ' as nesne ' +
                            ',@ln_destek_sayi ' +
                      @ls_str4 +
                      'ORDER BY ' +
                            'nesne ' +

                      'exec sp_fpgrowth_sonuc ' + cast(@pn_hareket_sayi as varchar) + ' ' +
```

Şekil 5.26 sp_fpgrowth_executesql stored procedure'ünden bir kesit

```
USE [ADDM]
GO
/****** Object:  UserDefinedFunction [dbo].[fn_fpgrowth_bul_fpg_id_by_nesne_ust_fpg_id]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER function [dbo].[fn_fpgrowth_bul_fpg_id_by_nesne_ust_fpg_id]
        (@pn_nesne decimal(10,0)
        ,@pn_ust_fpg_id int)
returns int
as

-- 12/2007 AD tasarlandı.

begin
    declare @ln_fpg_id int

    select
        @ln_fpg_id = fpg_id
    from
        dbo.t_ass_fpgrowth_tree
    where
        nesne = @pn_nesne
        and ust_fpg_id = @pn_ust_fpg_id

    return isnull(@ln_fpg_id, -1) -- "-1" ise insert et

end
```

Şekil 5.27 fn_fpgrowth_bul_fpg_id_by_nesne_ust_fpg_id isimli fonksiyon

Excel'deki veri seti bilgilerini içeren t_migros_kadir_has tablosundan faydalanarak, hareketsel (transactional) satışları içerecek t_migros_satis ve her bir satışa ait alınan ürünler ve detaylarını içerecek t_migros_satis_detay tabloları tasarlanmıştır. Böylelikle, Excel'de tekrar eden Hesap_No, Tarih, Kasa_No bilgilerinin tekilliği (unique) kullanılarak, ilgili tarihteki ilgili satış numarası ana bilgileri, t_migros_satis tablosunda oluşturulmuştur. t_migros_satis_detay tablosuna ise, ilgili satış numarasında alınmış ürünler, satış numaralarıyla birlikte işlenmiştir. Migros verisi üzerinde yapılan bu işlemlerin benzeri, BMS verileri için de uygulanmıştır.

Şekil 5.22'de yer alan satış, satış detay, ürün, ürün kategori tablolarının yapıları aynıdır. Böylelikle, birliktelik kuralları çıkarılacak olan veri setindeki bilgiler, t_satis, t_satis_detay, t_urun ve t_urun_detay tablolarına sp_veri_sec stored procedure'ü yardımıyla aktarılır ve algoritmalar çalıştırılır. Satış ve satış detay bilgileri kullanılarak her bir veri seti için oluşturulmuş ve ancak veri seti değiştiğinde baştan oluşturulması gereken FP-Tree yapılarını bulunduran tablolar vardır. Bu tabloların amacı, veri seti değişmediği müddetçe sabit kalacak olan FP-Tree yapısının tekrar oluşturulmasını ve zaman kaybını önlemektir. Aynı şekilde bu tablolardaki veriler de, sp_veri_sec stored procedure'ü çalıştırıldığında t_ass_fpgrowth_tree

tablosuna aktarılır.

Tabloları dört gruba ayırmak mümkündür:

- Excel ve metin dosyalarındaki bilgilerin aktarıldığı, ham verilerin bulunduğu tablolar (Şekil 5.21)
- Ham verilerin bulunduğu tablolardan türetilmiş, algoritmalara veri sağlayan tablolar (Şekil 5.22)
- Algoritmaların çalışması esnasında kullanılan, sonuçların ve birliktelik kurallarının bulunduğu tablolar (Şekil 5.23)
- Algoritmaların çalışması esnasında ve birliktelik kuralları oluştururken kullanılan geçici tablolar (Şekil 5.24)

Stored procedure'ler, algoritmaların çalışmasından önce veri sağlayan ve performans arttırıcı bir grup ile algoritmaları içeren operasyon grubu olmak üzere iki gruptur. Fonksiyonlar ise, operasyonel stored procedure'ler tarafından çağrılır.

5.3 Algoritmaların Çalıştırılması

Üç örnek veri seti üzerinde de işlem yapabilen Apriori ve FP-Growth algoritmaları, T-SQL kodları kullanılarak kodlanmış stored procedure'lerde yer almaktadır. Bu stored procedure'ler çalışırken belirli bir sırayı takip ederler. İlk olarak sp_veri_sec stored procedure'üne 'migros', 'bms1' veya 'bms2' parametrelerinden biri verilerek, birliktelik kurallarının çıkarılması istenen veri seti hazır edilir. Ardından, veri seti üzerinde çalıştırılması istenen ve kural üretilecek olan stored procedure'ler çalıştırılır. Apriori algoritmasının çalışması ve kuralların oluşması için sırasıyla sp_apriori ve sp_apriori_kural stored procedure'leri çalıştırılır. FP-Growth için ise sırasıyla sp_fpgrowth ve sp_fpgrowth_kural stored procedure'leri çalıştırılır.

Algoritmaların sonuçlar üretmesini ve kuralların oluşmasını sağlayabilmek için, stored procedure'lere @pn_min_sup isimli, decimal(5,3) tipli parametrenin verilmesi gerekmektedir. Bu parametre 4,317 gibi yüzdesel bir değer olabilir ve bu yüzdesel değeri kullanarak sonuçlar ve kurallar oluşturulur. Örneğin, Migros verisi 6401 adet hareketten oluşmaktadır. @pn_min_sup = 15 değeri için, 6401 * (15 / 100) = 960 adet hareket e sahip hiçbir birliktelik olmadığından dolayı kural oluşmaz. Örnek veri setleri dikkate alındığında, @pn_min_sup parametresine verilmesi gereken en büyük değer, Migros verisi için 14,989 (959 adet), BMS1 verisi için 4,516 (1204 adet) ve BMS2 verisi için de 2,853 (1501 adet) olmalıdır. Ayrıca veri setlerinin büyüklüğü dikkate alındığında, @pn_min_sup parametresine küçük değerler verilmesi durumunda işlem sürelerinde ciddi artışlar olduğu gözlenmektedir.

5.4 Algoritmaların Karşılaştırılması

5.4.1 Yöntem Farklılıklarına Göre Karşılaştırma

Apriori, aday üretimli bir metod olup, veritabanındaki hareketleri göz önünde bulundurmadan veritabanı üzerindeki bir önceki geçişte yaygın olarak belirlenmiş nesnekümelerini kullanarak, bir sonraki geçişte sayılacak olan aday nesnekümelerini oluşturur. Ana mantığı yaygın bir nesnekümesinin alt kümelerinin de yaygın olması gerekliliğidir. Bu nedenle k nesne içeren adaylar, (k-1) nesne içeren yaygın nesnekümelerinin birleştirilmesiyle oluşturulur. Bu işlemler, daha fazla aday nesnekümesi elde edilemeyene kadar sürer.

FP-Growth ise, adaylar üretmeksizin yaygın nesnekümelerinin bulunması için geliştirilmiş bir metoddur. Veritabanından, yaygın nesneleri temsil edecek şekilde FP-Tree denilen ağaç yapısı elde edilir ve bu sayede veritabanını tarama sayıları azaltılmış olur. FP-Tree'de nesnekümelerinin birliktelik bilgileri yer almaktadır. Bu metod arama maliyetlerini önemli ölçüde azaltır.

FP-Growth metodu üzerinde yapılan çalışmalar sonucunda, algoritmanın büyük yaygın nesnekümelerinin madenlenmesinde etkili ve ölçeklenebilir bir yapıda olduğu ve Apriori algoritmasından daha hızlı olduğu tespit edilmiştir.

5.4.2 Performans Farklılıklarına Göre Karşılaştırma

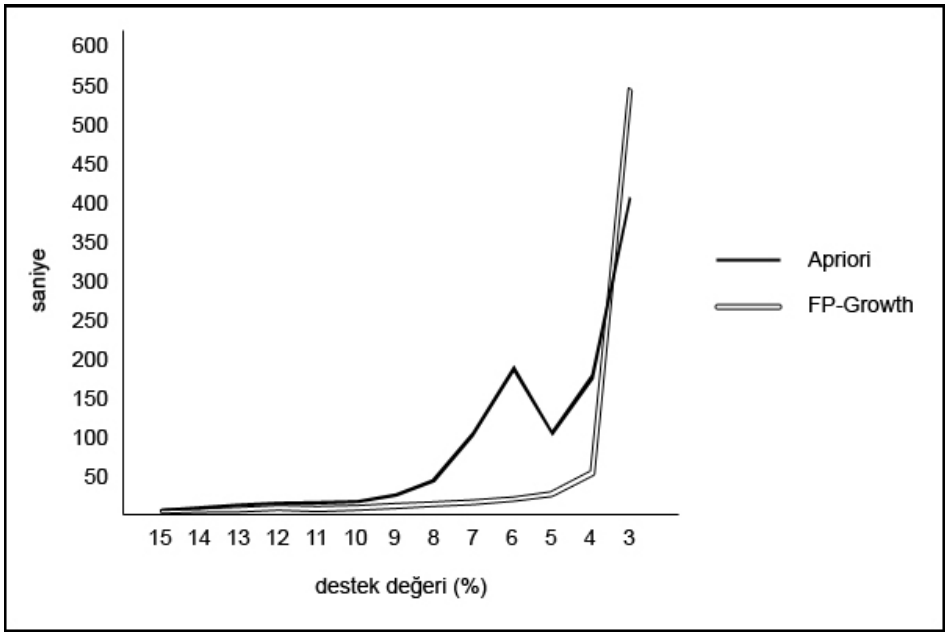
Bu tez çalışmasında uygulanan iki algoritmanın da farklı yöntemler kullanmasından dolayı farklı çalışma süreleri vardır. Veritabanını kullanma ve tarama şekli ile aldıkları parametreler farklıdır. Ancak, aynı destek değerleri için eşit sayıda yaygın nesnekümeleri bulmalarından dolayı, birliktelik kuralları oluşturma yöntemleri benzerdir.

İki algoritmadan da örnek veri setleri üzerinde elde edilen sonuçlar aşağıda belirtilmiştir. Algoritmalar Microsoft SQL Server 2005'te T-SQL ile kodlanmış olup, Intel Pentium 1.73 GHz işlemci ve 1GB RAM kapasiteli Microsoft Windows XP Professional işletim sistemine sahip bir bilgisayar üzerinde test edilmiştir.

5.4.2.1 Migros Verisi Sonuçları

Çizelge 5.1 Algoritmaların Migros verisi üzerindeki karşılaştırmalı sonuçları

Destek Değeri (%)	Destek Sayısı	Üretilen Kural Sayısı	k-nesnekume	Çalışma Süreleri (sn)	
				Apriori	FP-Growth
15,00	960	0	0	5	2
14,00	896	4	2	6	4
13,00	832	4	2	15	6
12,00	768	8	2	17	7
11,00	704	10	2	18	6
10,00	640	18	2	18	6
9,00	576	28	2	30	10
8,00	512	42	2	48	13
7,00	448	72	3	105	16
6,00	384	136	3	186	22
5,00	320	268	3	103	33
4,00	256	558	4	171	52
3,00	192	1618	4	404	546

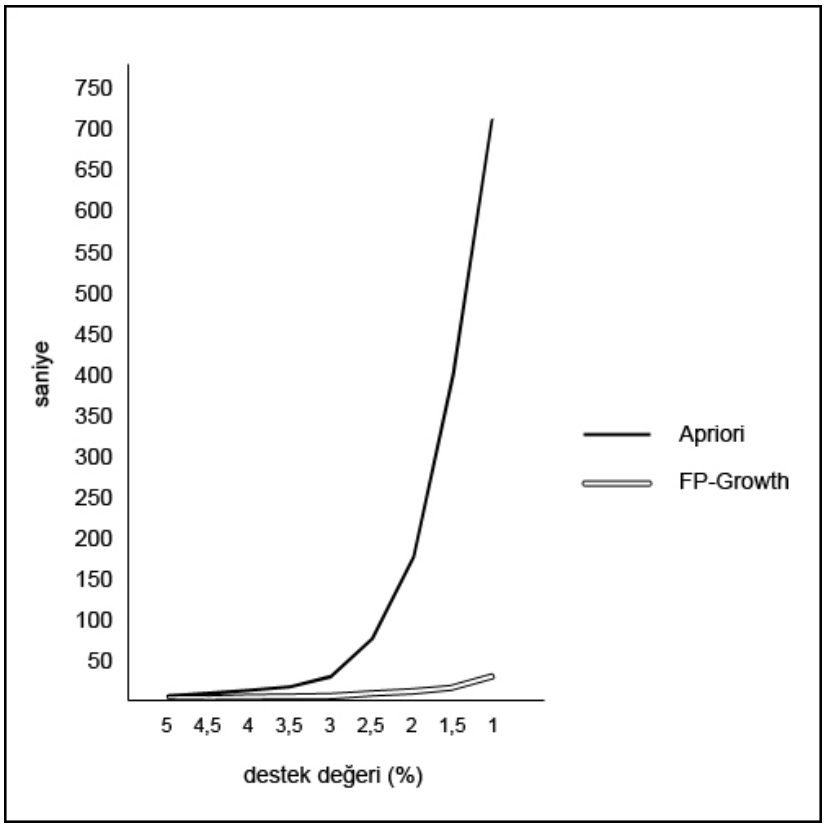


Şekil 5.28 Algoritmaların Migros verisi üzerindeki
ilgili destek değerlerinde toplam çalışma sürelerini gösteren grafik

5.4.2.2 BMS-WebView-1 Verisi Sonuçları

Çizelge 5.2 Algoritmaların BMS-WebView-1 verisi üzerindeki karşılaştırmalı sonuçları

Destek Değeri (%)	Destek Sayısı	Üretilen Kural Sayısı	k-nesnekume	Çalışma Süreleri (sn)	
				Apriori	FP-Growth
5,00	1333	0	0	8	3
4,50	1200	2	2	9	2
4,00	1067	2	2	10	2
3,50	933	2	2	17	3
3,00	800	6	2	33	5
2,50	667	12	2	76	8
2,00	533	28	2	175	13
1,50	400	58	3	403	20
1,00	267	184	3	717	32

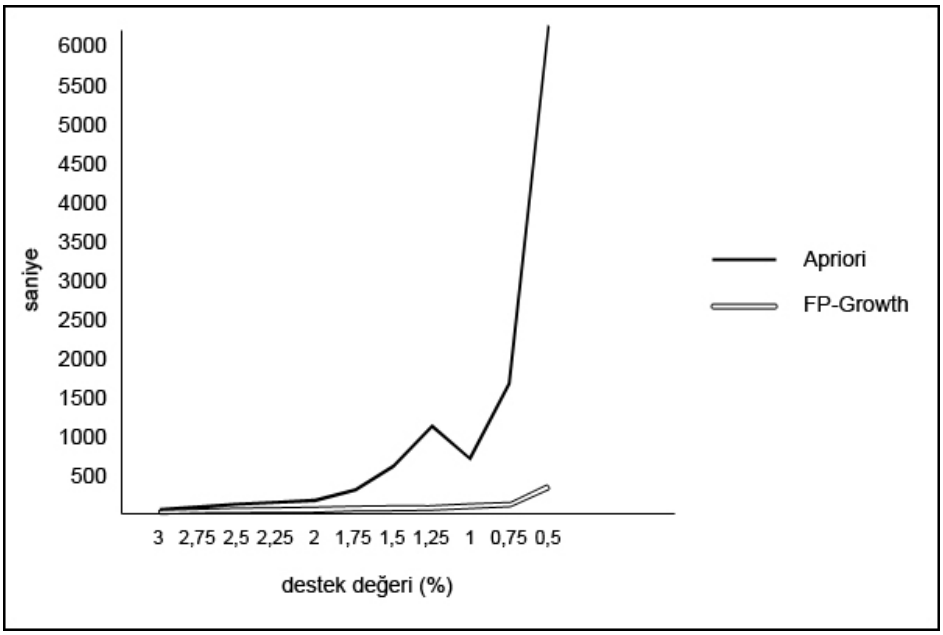


Şekil 5.29 Algoritmaların BMS1 verisi üzerindeki
ilgili destek değerlerinde toplam çalışma sürelerini gösteren grafik

5.4.2.3 BMS-WebView-2 Verisi Sonuçları

Çizelge 5.3 Algoritmaların BMS-WebView-2 verisi üzerindeki karşılaştırmalı sonuçları

Destek Değeri (%)	Destek Sayısı	Üretilen Kural Sayısı	k-nesnekume	Çalışma Süreleri (sn)	
				Apriori	FP-Growth
3,00	1579	0	0	70	3
2,75	1447	2	2	72	3
2,50	1315	4	2	106	4
2,25	1184	8	2	141	6
2,00	1052	12	2	198	8
1,75	921	30	3	374	14
1,50	789	58	3	571	16
1,25	658	124	3	1120	26
1,00	526	350	4	743	39
0,75	395	1132	5	1725	89
0,50	263	6242	6	6264	399



Şekil 5.30 Algoritmaların BMS2 verisi üzerindeki
ilgili destek değerlerinde toplam çalışma sürelerini gösteren grafik

5.4.3 Farklı Veri Setlerine Göre Karşılaştırma

Bu çalışmada, özelliklerinden daha önce bahsedilen üç farklı veri seti kullanılmıştır ve her veri setinden farklı sonuçlar elde edilmiştir. Veri setlerinin satır sayıları, hareket sayıları, ürün sayıları, ... vb. farklı olduğundan dolayı, aynı algoritmaların bu farklı veriler üzerindeki sonuçları da farklı olmuştur. Özellikle ürün çeşitliliği, üretilen yaygın nesnekümelerinin ve kurallarının sayısını ve çalışma sürelerini değiştirmiştir.

5.4.4 Yaygın Nesnekümelerine Göre Karşılaştırma

Her iki algoritma da, aynı veri seti ve aynı destek değeri için eşit sayıda yaygın nesnekümeleri oluşturur. Örneğin, 6401 hareketten oluşan Migros verisine %10'luk bir min_sup eşik değeri verildiğinde, 640 ve üzeri harekette birlikte alınmış olan ürünler belli olduğundan, iki algoritmanın da bu ürünleri yaygın nesnekümeleri olarak döndürmesi beklenir.

5.4.5 Eşik Destek Değerlerine Göre Karşılaştırma

Eşik destek değerine göre algoritmaların çalışma süreleri farklılık göstermektedir. min_sup değeri ne kadar küçük seçilirse, oluşturulacak yaygın nesnekümelerinin sayısı ve işlem süresi de o kadar fazla olur. Aynı veri seti üzerinde iki algoritmayı karşılaştırabilmek için parametre olarak aynı min_sup değeri verilmelidir. Bu parametre 4,317 gibi yüzdesel bir değer olabilir ve bu yüzdesel değeri kullanarak yaygın nesnekümeleri ve kurallar oluşturulur. Örneğin, Migros verisi 6401 adet hareketten oluşmaktadır ve min_sup = 5 değeri için, 6401 * (5 / 100) = 320 adet hareket e sahip yaygın nesnekümeleri bulunarak birliktelik kuralları oluşturulur.

Uygulamada kullanılan örnek veri setleri dikkate alındığında, min_sup parametresine verilmesi gereken en büyük değer, Migros verisi için 14,989 (959 adet), BMS1 verisi için 4,516 (1204 adet) ve BMS2 verisi için de 2,853 (1501 adet) olmalıdır. Ayrıca veri setlerinin büyüklüğü dikkate alındığında, min_sup parametresine küçük değerler verilmesi durumunda işlem sürelerinde ciddi artışlar olduğu gözlenmektedir.

min_sup değeri büyük verilirse, veri setinden önemli bilgiler taşıyabilecek olan bazı birliktelikler elde edilmeyebilir. min_sup değeri küçük verilirse, yöntem karmaşıklaşır, süre uzar ve çok fazla sayıda yaygın nesneler kümesi ve kurallar oluşur. Ayrıca tek bir destek değerinde çalışmak da doğru değildir, farklı destek değerlerinden alınacak sonuçlara göre kurallar değerlendirilmelidir.

5.5 Üretilen Birliktelik Kuralları

Daha önce de bahsedildiği üzere, birliktelik kural madenciliği iki temel adımdan oluşmaktadır. İlki, yaygın nesnekümelerinin bulunmasıdır. Diğeri ise, yaygın nesnekümelerini kullanarak birliktelik kurallarının bulunmasıdır.

Algoritmalar için asıl karşılaştırma kriteri, birinci adımı gerçekleştirme aşamasıdır. İkinci adım ise ortak bir prosedür ile gerçekleştirilebilir. Birinci adımda üretilen yaygın nesnekümeleri ilgili prosedüre girdi olarak verilir ve birliktelik kuralları elde edilir.

Bu çalışmada uygulanan algoritmaların, aynı verikümesi üzerinde ürettikleri yaygın nesnekümeleri aynı olduğundan, birliktelik kuralları da aynıdır. İki algoritma için de yazılmış olan kural üretme stored procedure'leri benzerdir.

5.5.1 Migros Verisine Ait Birliktelik Kuralları

Çizelge 5.4 Migros verisinin madenlenmesi ile oluşan birliktelik kuralları (güven sıralı)

k-nesnekume	Birliktelik Kuralları (min_sup = 6)	Destek (%)	Güven (%)
2	ZEYTİN ==> PEYNİR	8,70	79,00
2	HIYAR ==> DOMATES	9,60	72,80
2	SUCUK ==> PEYNİR	6,20	72,50
3	SÜT, YUMURTA ==> PEYNİR	6,90	70,00
2	BİBER ==> DOMATES	8,00	65,90
3	SÜT, YOĞURT ==> PEYNİR	7,80	62,60
2	YUMURTA ==> PEYNİR	11,60	62,20
2	MAKARNA ==> PEYNİR	8,00	60,60
3	PEYNİR, YOĞURT ==> SÜT	7,80	60,50
3	EKMEK, SÜT ==> PEYNİR	6,40	60,10
3	PEYNİR, YUMURTA ==> SÜT	6,90	59,20
2	MEYVELİ YOĞURT ==> PEYNİR	7,50	56,20
3	EKMEK, YOĞURT ==> PEYNİR	6,10	56,10
2	PEÇETE ==> PEYNİR	6,50	54,90
2	MAKARNA ==> SÜT	7,30	54,70
2	ÇİKOLATA ==> PEYNİR	10,00	54,30
2	HIYAR ==> PEYNİR	7,10	54,20
2	SALATALIK ==> DOMATES	6,30	53,30
2	MEYVE SUYU ==> PEYNİR	8,40	52,70
2	YUMURTA ==> SÜT	9,80	52,60
2	MEYVELİ YOĞURT ==> SÜT	7,00	52,50
2	BİSKÜVİ ==> PEYNİR	10,70	52,40
3	PEYNİR, SÜT ==> YOĞURT	7,80	51,80
2	KIRMIZI ET ==> PEYNİR	7,50	51,60
2	SÜT ==> PEYNİR	15,00	51,40
2	DOMATES ==> PEYNİR	10,80	51,30
2	GOFRET ==> PEYNİR	7,30	50,90
2	KURUYEMİŞ ==> PEYNİR	6,70	50,70
2	YOĞURT ==> PEYNİR	12,80	50,30
2	BİBER ==> PEYNİR	6,00	49,60
...	...	...	...

Çizelge 5.5 Migros verisinin madenlenmesi ile oluşan birliktelik kuralları (destek sıralı)

k-nesnekume	Birliktelik Kuralları (min_sup = 6)	Destek (%)	Güven (%)
2	SÜT ==> PEYNİR	15,00	51,40
2	PEYNİR ==> SÜT	15,00	43,00
2	EKMEK ==> PEYNİR	14,40	45,30
2	PEYNİR ==> EKMEK	14,40	41,30
2	YOĞURT ==> PEYNİR	12,80	50,30
2	PEYNİR ==> YOĞURT	12,80	36,90
2	YOĞURT ==> SÜT	12,40	48,60
2	SÜT ==> YOĞURT	12,40	42,60
2	YUMURTA ==> PEYNİR	11,60	62,20
2	PEYNİR ==> YUMURTA	11,60	33,40
2	DOMATES ==> PEYNİR	10,80	51,30
2	YOĞURT ==> EKMEK	10,80	42,50
2	EKMEK ==> YOĞURT	10,80	34,20
2	PEYNİR ==> DOMATES	10,80	31,00
2	BİSKÜVİ ==> PEYNİR	10,70	52,40
2	PEYNİR ==> BİSKÜVİ	10,70	30,80
2	SÜT ==> EKMEK	10,60	36,50
2	EKMEK ==> SÜT	10,60	33,50
2	ÇİKOLATA ==> PEYNİR	10,00	54,30
2	DOMATES ==> EKMEK	10,00	47,40
2	EKMEK ==> DOMATES	10,00	31,50
2	PEYNİR ==> ÇİKOLATA	10,00	28,60
2	YUMURTA ==> SÜT	9,80	52,60
2	SÜT ==> YUMURTA	9,80	33,80
2	HIYAR ==> DOMATES	9,60	72,80
2	DOMATES ==> HIYAR	9,60	45,50
2	BİSKÜVİ ==> SÜT	9,00	44,10
2	SÜT ==> BİSKÜVİ	9,00	30,90
2	ZEYTİN ==> PEYNİR	8,70	79,00
2	PEYNİR ==> ZEYTİN	8,70	25,00
2	ÇİKOLATA ==> SÜT	8,50	46,00
2	YUMURTA ==> YOĞURT	8,50	45,70
2	YOĞURT ==> YUMURTA	8,50	33,50
2	SÜT ==> ÇİKOLATA	8,50	29,00
2	MEYVE SUYU ==> PEYNİR	8,40	52,70
...	...	...	...

5.5.2 BMS-WebView-1 Verisine Ait Birliktelik Kuralları

Çizelge 5.6 BMS1 verisinin madenlenmesi ile oluşan birliktelik kuralları (güven sıralı)

k-nesnekume	Birliktelik Kuralları (min_sup = 1,5)	Destek (%)	Güven (%)
2	2375 ==> 2399	2,10	82,60
2	1904 ==> 1908	1,70	69,70
3	1908, 1909 ==> 1903	1,60	67,10
2	3998 ==> 4002	1,50	64,50
2	2379 ==> 2399	2,20	64,00
2	3991 ==> 4002	1,90	61,20
2	4002 ==> 3996	4,50	57,90
3	1903, 1909 ==> 1908	1,60	56,50
2	1917 ==> 1910	1,60	55,40
2	3996 ==> 4002	4,50	54,90
2	2264 ==> 2266	3,30	52,30
2	3991 ==> 3996	1,60	51,00
2	2338 ==> 2340	2,30	50,80
2	1903 ==> 1908	3,40	50,10
3	1903, 1908 ==> 1909	1,60	45,50
2	2266 ==> 2264	3,30	45,30
2	3941 ==> 2340	2,10	43,90
2	1908 ==> 1903	3,40	43,10
2	2394 ==> 2266	1,60	43,00
2	1903 ==> 1909	2,80	40,30
2	1903 ==> 1910	2,70	39,50
2	3941 ==> 2266	1,90	38,90
2	1909 ==> 1910	2,90	38,20
2	2340 ==> 2266	2,40	37,80
2	2340 ==> 2338	2,30	36,80
2	1909 ==> 1903	2,80	36,50
2	3941 ==> 1909	1,70	35,60
2	2340 ==> 1909	2,20	34,50
2	2340 ==> 3941	2,10	34,20
2	2266 ==> 2340	2,40	32,60
2	2266 ==> 1909	2,30	31,80
2	1910 ==> 1909	2,90	31,50
2	1909 ==> 1908	2,30	30,70
2	2334 ==> 2399	1,50	30,50
2	1909 ==> 2266	2,30	30,40
...	...	...	...

Çizelge 5.7 BMS1 verisinin madenlenmesi ile oluşan birliktelik kuralları (destek sıralı)

k-nesnekume	Birliktelik Kuralları (min_sup = 1,5)	Destek (%)	Güven (%)
2	4002 ==> 3996	4,50	57,90
2	3996 ==> 4002	4,50	54,90
2	1903 ==> 1908	3,40	50,10
2	1908 ==> 1903	3,40	43,10
2	2264 ==> 2266	3,30	52,30
2	2266 ==> 2264	3,30	45,30
2	1909 ==> 1910	2,90	38,20
2	1910 ==> 1909	2,90	31,50
2	1903 ==> 1909	2,80	40,30
2	1909 ==> 1903	2,80	36,50
2	1903 ==> 1910	2,70	39,50
2	1910 ==> 1903	2,70	29,50
2	2340 ==> 2266	2,40	37,80
2	2266 ==> 2340	2,40	32,60
2	2338 ==> 2340	2,30	50,80
2	2340 ==> 2338	2,30	36,80
2	2266 ==> 1909	2,30	31,80
2	1909 ==> 1908	2,30	30,70
2	1909 ==> 2266	2,30	30,40
2	1908 ==> 1909	2,30	29,20
2	2379 ==> 2399	2,20	64,00
2	2340 ==> 1909	2,20	34,50
2	1909 ==> 2340	2,20	28,50
2	2399 ==> 2379	2,20	23,60
2	2375 ==> 2399	2,10	82,60
2	3941 ==> 2340	2,10	43,90
2	2340 ==> 3941	2,10	34,20
2	2399 ==> 2375	2,10	22,10
2	3991 ==> 4002	1,90	61,20
2	3941 ==> 2266	1,90	38,90
2	2266 ==> 3941	1,90	26,10
2	4002 ==> 3991	1,90	24,50
2	1908 ==> 1910	1,90	23,30
2	1910 ==> 1908	1,90	20,20
2	2264 ==> 1909	1,80	28,00
...	...	...	...

5.5.3 BMS-WebView-2 Verisine Ait Birliktelik Kuralları

Çizelge 5.8 BMS2 verisinin madenlenmesi ile oluşan birliktelik kuralları (güven sıralı)

k-nesnekume	Birliktelik Kuralları (min_sup = 1,5)	Destek (%)	Güven (%)
3	8306, 8309 ==> 8307	1,90	78,70
3	8306, 8307 ==> 8309	1,90	74,50
3	8284, 8307 ==> 8309	1,50	69,20
3	8307, 8309 ==> 8306	1,90	67,50
2	8306 ==> 8307	2,60	63,40
3	8284, 8309 ==> 8307	1,50	62,60
2	8307 ==> 8309	2,90	61,80
2	8306 ==> 8309	2,40	60,00
2	8295 ==> 8307	2,00	57,90
2	8295 ==> 8309	1,90	57,40
2	8307 ==> 8306	2,60	56,00
2	8291 ==> 8309	1,50	55,50
3	8307, 8309 ==> 8284	1,50	53,60
2	8324 ==> 8309	2,10	51,60
2	8295 ==> 8284	1,70	50,40
2	8293 ==> 8309	1,80	50,10
2	8295 ==> 8306	1,70	49,90
2	8285 ==> 8284	2,00	49,20
2	8306 ==> 8284	2,00	48,60
2	8309 ==> 8307	2,90	48,10
2	8295 ==> 8293	1,60	48,00
2	8307 ==> 8284	2,20	47,90
3	8306 ==> 8307, 8309	1,90	47,20
2	8293 ==> 8307	1,70	46,50
2	8324 ==> 8307	1,80	45,50
2	8293 ==> 8295	1,60	44,70
2	8295 ==> 8285	1,50	44,60
2	8307 ==> 8295	2,00	42,50
2	8324 ==> 8284	1,70	42,00
2	8293 ==> 8284	1,50	42,00
3	8307 ==> 8306, 8309	1,90	41,70
2	8306 ==> 8295	1,70	41,50
2	8309 ==> 8306	2,40	41,30
2	8309 ==> 8284	2,40	41,20
2	8285 ==> 8309	1,60	40,70
...	...	...	...

Çizelge 5.9 BMS2 verisinin madenlenmesi ile oluşan birliktelik kuralları (destek sıralı)

k-nesnekume	Birliktelik Kuralları (min_sup = 1,5)	Destek (%)	Güven (%)
2	8307 ==> 8309	2,90	61,80
2	8309 ==> 8307	2,90	48,10
2	8306 ==> 8307	2,60	63,40
2	8307 ==> 8306	2,60	56,00
2	8306 ==> 8309	2,40	60,00
2	8309 ==> 8306	2,40	41,30
2	8309 ==> 8284	2,40	41,20
2	8284 ==> 8309	2,40	37,70
2	8307 ==> 8284	2,20	47,90
2	8284 ==> 8307	2,20	34,10
2	8324 ==> 8309	2,10	51,60
2	8309 ==> 8324	2,10	34,80
2	8295 ==> 8307	2,00	57,90
2	8285 ==> 8284	2,00	49,20
2	8306 ==> 8284	2,00	48,60
2	8307 ==> 8295	2,00	42,50
2	8284 ==> 8285	2,00	30,60
2	8284 ==> 8306	2,00	30,50
3	8306, 8309 ==> 8307	1,90	78,70
3	8306, 8307 ==> 8309	1,90	74,50
3	8307, 8309 ==> 8306	1,90	67,50
2	8295 ==> 8309	1,90	57,40
3	8306 ==> 8307, 8309	1,90	47,20
3	8307 ==> 8306, 8309	1,90	41,70
2	8309 ==> 8295	1,90	32,80
3	8309 ==> 8306, 8307	1,90	32,50
2	8293 ==> 8309	1,80	50,10
2	8324 ==> 8307	1,80	45,50
2	8307 ==> 8324	1,80	39,40
2	8309 ==> 8293	1,80	30,80
2	8295 ==> 8284	1,70	50,40
2	8295 ==> 8306	1,70	49,90
2	8293 ==> 8307	1,70	46,50
2	8324 ==> 8284	1,70	42,00
2	8306 ==> 8295	1,70	41,50
...	...	...	...

5.6 Algoritmaların Sonuçlarının Değerlendirilmesi

Yapılan çalışma sonucunda, her üç veri seti üzerinde algoritmaların hem düşük hem yüksek destek değerlerindeki performans sıralaması FP-Growth > Apriori şeklinde olmuştur.

Apriori, nesnekümelerini ve destek değerlerini hesaplarken veritabanından geçişler yapmaktadır, bu da çalışma süresinin artmasına neden olur. FP-Growth ise, oluşturulan FP-Tree yapısını kullanarak, düşük destek değerine sahip yaygın nesnekümelerinden başlayarak ilgili nesnekümesi için tüm birliktelikleri bulur. Veritabanı taramaları olmadığı için daha hızlı sonuç elde edilir.

Elde edilen sonuçlarda da görüldüğü üzere, destek değeri küçüldükçe her iki algoritmada da çalışma süreleri, üretilen yaygın nesnekümeleri ve kurallar artmaktadır.

6. SONUÇLAR VE ÖNERİLER

Veri madenciliğinde amaç çok büyük miktardaki ham veriden değerli bilginin çıkarılmasıdır. Çok miktarda güvenilir (hata ve eksiklerin olmadığı) veri önşarttır çünkü çözümün, yani çıkarılan kuralların kalitesi öncelikle verinin kalitesine bağlıdır.

Veri madenciliği çalışması bilgisayarcıların ve uygulama konusundaki uzmanların ortak çalışmasıdır. Her ne kadar olabildiğince otomatik olmasını istesek de uzmanların yardımı ve desteği olmadan başarılı olmak sözkonusu değildir. Uzmanlar amacı tanımlar. Uygulama ile ilgili sonuca yararlı olabilecek her tür bilginin sisteme verilmesi gerekir ve bunları da ancak uzmanlar bilir. Ayrıca çalışma ile alınan sonuçların yorumlanması ve geçerlenmesi uzmanlar tarafından yapılır. Veri madenciliği tek aşamalı bir çalışma olmayıp tekrarlıdır. Sistem ayarlanana dek birçok deneme gerektirebilir. Çalışma uzun olabilir, sabır gerektirir. Buna çalışan ekibin ve yönetimin hazırlıklı olması, kısa vadede çok büyük beklentilere sahip olunmaması gerekir.

Bu çalışmada ele alınan market sepet analizi ve birliktelik kuralları çıkarımı çerçevesinde incelenmiş birçok algoritma vardır. Bu farklı çalışma koşulları ve yöntemleri içeren algoritmaların, farklı veri yapıları üzerinde çalıştırılarak performans ölçümleri yapılabilir. Gerçek verilerle veya veri üreteçleri ile rastgele üretilen verilerle birliktelik kuralları çıkarımı yapılarak sonuçlar karşılaştırılabilir. Algoritmaların iyi özellikleri birleştirilerek yeni ve daha güçlü algoritmalar elde edilebilir.

KAYNAKLAR

Agrawal, R., Imielinski, T. ve Swami, A., (1993), Mining Association Rules Between Sets of Items in Large Databases, ACM SIGMOD Conference on Management of Data, Washington.

Agrawal, R. ve Srikant, R., (1994), Fast Algorithms for Mining Association Rules, VLDB Conference, Santiago, Chile.

Akpınar, H., (2000), "Veri Tabanlarında Bilgi Keşfi ve Veri Madenciliği", İstanbul Üniversitesi İşletme Fakültesi Dergisi, Nisan 2000 Sayısı, C: 29.

Alataş, B. ve Akın, E., (2004), Veri Madenciliğinde Yeni Yaklaşımlar, YA/EM'2004 Yöneylem Araştırması / Endüstri Mühendisliği, XXIV. Ulusal Kongresi, Gaziantep – Adana.

Alpaydın, E., (2000), "Zeki Veri Madenciliği: Ham Veriden Altın Bilgiye Ulaşma Yöntemleri", Bilişim 2000 Eğitim Semineri, 1-10.

Berry, M. J. A. ve Linoff, G. S., (2004), Data Mining Techniques for Marketing, Sales and Customer Relationship Management, Wiley Publishing, Inc., Indianapolis.

Dolgun, M. Ö., (2006), Büyük Alışveriş Merkezleri için Veri Madenciliği Uygulamaları, Yüksek Lisans Tezi, Hacettepe Üniversitesi İstatistik Anabilim Dalı, Ankara.

Dunham, M. H., Xiao, Y., Gruenwald L. ve Hossain, Z., (2000), "A Survey of Association Rules", Teknik Rapor, Southern Methodist University, Department of Computer Science, T R00-CSE-8.

Han, J. ve Kamber, M., (2000), Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, Burnaby.

Hegland, M., (2002), Algorithms for Association Rules, Lecture Notes, Australian National University, Australia.

Hipp, J., Güntzer, U. ve Nakhaeizadeh, G., (2000), Algorithms for Association Rule Mining – A General Survey and Comparison, 2000 ACM SIGKDD, Tübingen, Germany.

Koyuncu, F., (2004), Veri Madenciliğinde Apriori Temelli İlişkilendirme Kuralı Algoritmalarının Uygulama ve Karşılaştırması, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Özçakır, F. C., (2006), Müşteri İşlemlerindeki Birlikteliklerin Belirlenmesinde Veri Madenciliği Uygulaması, Yüksek Lisans Tezi, Marmara Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Özel, S. A. ve Güvenir, H. A., (2001), An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning, Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'2001), Gazimagusa, 257-264.

Srikant, R. ve Agrawal R., (1996), Mining Quantitive Association Rules in Large Relational Tables, ACM SIGMOD'96 International Conference of Management of Data, Montreal, 1-12.

Tantuğ, A. C., (2002), Veri Madenciliği ve Demetleme, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Uysal, İ. ve Güvenir, H. A., (1999), An Application of Inductive Learning for Mining Basket Data, Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'99), İstanbul, 30-39.

Vahaplar, A. ve İnceoğlu, M. M., (2001), Veri Madenciliği ve Elektronik Ticaret, Türkiye'de İnternet Konferansları VII, Ege Üniversitesi Bilgisayar Mühendisliği Bölümü, İzmir.

Witten, I. H. ve Frank, E., (2005), Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers, San Francisco.

Zaki, M. J. ve Ogihara, M., (1998), Theorical Foundations of Association Rules, SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), Seattle, WA.

İNTERNET KAYNAKLARI

[1] http://www.cmpe.boun.edu.tr/~ethem

[2] http://www.cs.bu.edu/faculty/gkollios/ada02/LectNotes/assoc1.ppt

[3] http://www.cs.bu.edu/faculty/gkollios/ada02/LectNotes/assoc2.ppt

[4] http://www.verimadenciligi.gen.tr

[5] http://www2.cs.uh.edu/~ceick/6340/grue-assoc.pdf

[6] http://mail.baskent.edu.tr/~20394676/0302/bil483/HW2.pdf

[7] http://www.danismend.com/konular/bilgiveteknoyon/bilgi_olap1.htm

[8] http://tr.wikipedia.org/wiki/Veri_taban%C4%B1

[9] http://cvs.buu.ac.th/~mining/Datasets/real_data/

[10] http://www3.itu.edu.tr/~sgunduz/courses/verimaden/

ÖZGEÇMİŞ

Doğum tarihi	17.12.1981	
Doğum yeri	Edirne	
Lise	1993-2000	Edirne Anadolu Lisesi
Lisans	2000-2004	Sakarya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
Yüksek Lisans	2004-2008	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

Çalıştığı kurumlar

2005-2006	Turkuaz Bilgisayar
2006-Devam ediyor	Supercom Bilişim Bilgisayar