İLİŞKİSEL VERİ TABANLARINDA ANAHTAR KELİME ARAMA

Serap DEMİRCİOĞLU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NİSAN 2012
ANKARA

Serap DEMİRCİOĞLU tarafından hazırlanan "İLİŞKİSEL VERİ TABANLARINDA ANAHTAR KELİME ARAMA" adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Doç. Dr. Suat ÖZDEMİR …….
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans olarak kabul edilmiştir.

Prof. Dr. M. Ali AKÇAYOL …….
Bilgisayar Mühendisliği Anabilim Dalı, G.Ü.

Doç. Dr. Suat ÖZDEMİR …….
Bilgisayar Mühendisliği Anabilim Dalı, G.Ü.

Yrd. Doç. Dr. Süleyman TOSUN …….
Bilgisayar Mühendisliği Anabilim Dalı, A.Ü.

Tarih: 20/04/2012

Bu tez ile G.Ü. Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onamıştır.

Prof. Dr. Bilal TOKLU …….
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Serap DEMİRCİOĞLU

İLİŞKİSEL VERİ TABANLARINDA ANAHTAR KELİME ARAMA
(Yüksek Lisans Tezi)

Serap DEMİRCİOĞLU

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
NİSAN 2012

ÖZET

İlişkisel veri tabanlarında anahtar kelime aramak için birçok çalışma yapılmıştır. Son kullanıcının veri tabanı yapısından habersiz ve SQL sorgu dilini kullanmadan veri tabanından sorgu yapabilme ihtiyacı ilişkisel veri tabanlarında anahtar kelime arama uygulamaları ihtiyacını doğurmuştur. Yapılan çalışmaların ortak özelliği metin alanlarda metin arama işlemini gerçekleştirmeleridir. Bu çalışmaların amacı kullanıcının dışarıdan girdiği anahtar kelimeleri kullanarak veri tabanı sorgusu oluşturmak ve elde edilen sonuçları kullanıcıya sunmaktır. Burada önemli olan tablolar arası ilişkilerin doğru tanımlanabilmesi ve sonuç olarak verilecek veri setinin doğru sıralanabilmesidir. Yapılan çalışmalarda performans ikinci planda tutularak doğru sırada doğru sonuç üretebilme hedeflenmiştir. Çalışmalar veri tabanına özel geliştirildiklerinden genele hitap etmemektedirler. Bu durum yapılan çalışmaların dezavantajı gibi görünse de aslında olması gereken bir durumdur. Bu çalışmada tablolar arası ilişkinin tanımlanmasında sadece dış anahtar, birincil anahtar ilişkisinin kullanılmasının yeterli olmayacağını göstererek kayıtlar için tanımlayıcı olabilecek diğer alanlar üzerinden de ilişkiler tanımlamıştır. Tanımlanan bu ilişkiler elde edilen bilginin detaylanmasını sağlamıştır. Uygulanan yöntemin adımları ve elde edilen sonuçlar detaylı bir şekilde sunulmaktadır.

Bilim Kodu : 902.1.067
Anahtar Kelimeler : Anahtar kelime arama, ilişkisel veri tabanları
Sayfa Adedi : 95
Tez Yöneticisi : Doç. Dr. Suat ÖZDEMİR

KEYWORD SEARCH IN RELATIONAL DATABASES
(M.Sc. Thesis)

Serap DEMİRCİOĞLU

GAZI UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
APRIL 2012

ABSTRACT

As most of end users are not aware of database structure and want to make query without using SQL, keyword search in relational databases have been studied in the literature extensively. The common idea of the existing studies is to search keywords in text areas of the databases. These studies focused on two main points. First, the relation between tables should be well defined and the second, the results should be sorted in logical manner. However, run time performance of these systems is overlooked by the existing studies. In addition, current studies are developed for specific database schemas and they could not be extended for general purpose databases. Although this appears to be disadvantage, it is a necessity. In this study, we show that primary key, foreign key relation is not enough to construct relation between tables. In addition to this relation we also define new relations by using other fields which hold unique data like e-mail address or identity number. Performance analysis shows that, by using these newly introduced relations, query results are enriched.

Science Code : **902.1.067**
Key Words : **Keyword search, Relational database**
Page Number : **95**
Adviser : **Assoc. Prof. Dr. Suat ÖZDEMİR**

TEŞEKKÜR

Çalışmalarım boyunca yardım ve katkılarıyla beni yönlendiren danışman hocam Sayın Doç. Dr. Suat ÖZDEMİR'e, Yüksek Lisans eğitimim süresince yardımlarını esirgemeyen hocalarım Sayın Prof. Dr. Şeref SAĞIROĞLU, Sayın Prof. Dr. M. Ali AKCAYOL ve Yrd. Doç. Dr. Hacer KARACAN'a, çalışmalarım sırasında manevi desteğini ve değerli bilgilerini esirgemeyen eşim Erşan DEMİRCİOĞLU'na ayrıca maddi ve manevi her türlü destekleriyle beni hiçbir zaman yalnız bırakmayan çok değerli aileme teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

1. GİRİŞ

İnternette anahtar kelime araması, arama motorlarına verilen kelimenin tüm internet dokümanlarında aranması ve yakınlığı olan dokümanların belli bir mantık sırasında kullanıcıya sunulması şeklinde gerçekleştirilir.

Veritabanında anahtar kelime araması internet üzerindeki aramalardan farklıdır. Veri tabanlarında istenilen bir bilgiye ulaşılması internette metin üzerinde yapılan aramaya göre daha karmaşıktır. Çünkü istenilen bir bilgi tek bir tabloda değil bir kaç tabloya dağıtılmış şekilde tutulmaktadır. Veri tabanlarında gerçekleştirilen bu normalizasyon işlemi, veri tabanlarında anahtar kelime aramasını güçleştirmektedir. İstenilen bilgiye ulaşmak için aranan kelimenin bulunduğu ve bu kayıt ile ilişki içinde bulunan tablolardan arama yapılması gerekir. Bundan dolayı arama işlemi bize aranan kelimenin bulunmasının yanında kelimenin bulunduğu satırın ilişkilerinin de bilinmesi ve arama işleminin o satırlara da genişletilmesi iş yükünü getirir. Peki, veritabanında anahtar kelime ile arama yapılması gerçekten gerekli midir [9]?

Günümüzde birçok firma bilgilerini ilişkisel veritabanları üzerinde tutmaktadır. Bu durumda kullanıcıların veritabanındaki veriye ulaşmaları önem kazanmaktadır. Çünkü firmanın her personeli ve personelin her ihtiyaç duyduğu bilgiye yönelik bit rapor hazırlanması işlemi daha büyük bir iş gücü kaybına neden olmaktadır. Bu ihtiyacı gidermek için ilişkisel veritabanlarına uygun, firmaya özel arama motorları tasarlanmalıdır. İlişkisel veri tabanları bir firma için çok gerekli ve yararlıdır. Fakat bunun yanında ilişkisel veri tabanlarında arama yapmak internette arama yapmak gibi kolay olmadığından veri tabanları geliştirilme esnasında kısıtlı olarak kelime arama desteği vermiştir. Fakat veri tabanlarının anahtar kelime arama desteği çok kısıtlıdır. Aynı zamanda internet üzerinde arama yapan arama motoru teknikleri ilişkisel veri tabanlarında doğrudan kullanılamaz. İlişkisel veritabanlarında arama yapmayı sağlayabilmek için veritabanın yapısının ve

ilişkilerinin bilinmesi gerekmektedir. Ayrıca veritabanın normalizasyonu da bu işlemi güçleştirmektedir.

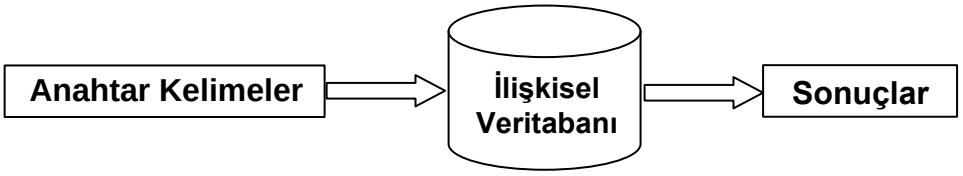
İlişkisel veri tabanlarında anahtar kelime araması önemli bir ihtiyaçtır. Çünkü ilişkisel veritabanında anahtar kelime araması sıradan bir kullanıcı için çok zordur. Bunun nedeni, ilişkisel veritabanlarında istenilen bilgiye ulaşılabilmesi için SQL sorgu yapısının bilinmesi ve sorgulanacak veri tabanında hangi bilginin hangi tabloda bulunduğunun ve bu tabloların hangi tablolar ile ilişki içerisinde olduğunun bilinmesi gerekmektedir. Oysaki sıradan bir kullanıcı SQL sorgu dilini, verinin şema ve ilişkisel yapısını bilmez. Tek istediği anahtar kelimeleri verip ilgili sonuçları almaktır. Bu amaçla önerdiğimiz yöntem kullanıcının girdiği anahtar kelime kayıtlarını elde eder ve bu kayıtların ilişkili olduğu kayıtları bulur. Önerdiğimiz yöntem literatürdeki çalışmalardan farklı olarak, bir kaydın ilişkili olduğu kayıtları elde etmede dış anahtar → birincil anahtar ilişkisinin yanında benzersiz alan tanılamalarının oluşturduğu ilişkileri de kullanmıştır. Benzersiz alan tanımlaması veri tabanı yöneticisi tarafından tanımlanır. Bu alanlar bir kişi veya nesne için belirleyici olma özelliğine sahiptirler. Fakat veri tabanında dış anahtar → birincil anahtar olarak tanımlanmamışlardır. Örneğin veri tabanının iki veya daha fazla tablosunda dış anahtar → birincil anahtar olarak tanımlanmamış fakat ek bilgi verme amaçlı kullanılmış olan TC kimlik numarası alanı veri tabanı yönrticisi tarafından benzersiz alan olarak tanımlanmış ve kayıtlar arası ilişkileri ortaya çıkarmada kullanılmıştır. Kullanıcının girmiş olduğu anahtar kelimeler için yapılan aramada benzersiz alan tanımlamalarının da kullanılması ile arama genişletilmiş hatta hiç ulaşılamayacak kayıtlara da ulaşılabilmesi sağlanmıştır. Önerilen yöntem ile ilgili sonuçlar 6. bölümde ayrıntılı olarak verilmiştir.

Hazırlanan çalışmanın 2. bölümünde problemin tanımından, öneminden ve yapılan katkılardan bahsedilmiştir. 3. bölümde literatürde yapılan çalışmalar incelenmiştir. 4. bölümde literatürdeki çalışmaların değerlendirilmesi yapılmıştır. 5. bölümde önerilen yöntem ayrıntılı bir şekilde anlatılmıştır. 6.

bölümde önerilen yöntem bir veri tabanı üzerinde çalıştırılmış ve uygulama sonuçları örnekler ile verilmiştir. Son olarak sonuç bölümünde önerilen yöntemin sağladığı katkılardan bahsedilmiştir.

2. PROBLEME GENEL BAKIŞ

Literatürde ilişkisel veritabanlarında arama yapmak için bazı çalışmalar yapılmıştır. Yapılan çalışmalar ortak olarak üç ana görevi barındırmaktadır. Bu görevler Şekil 2.1'de gösterildiği gibidir.



Şekil 2.1. İlişkisel veri tabanlarında anahtar kelime arama akışı

- *Veritabanının modellenmesi:* İlişkisel veri tabanında yer alan tablo ve verilerin birbirleri arasındaki ilişkinin, önerilen algoritmaların gereksinimine göre modellenmesi işlemidir.

- *İndeksleme:* Anahtar kelimeyi içeren verilere daha hızlı ulaşmak için veri tabanı üzerinde indeksleme işlemdir. Bu indeksleme çoğu zaman veri tabanlarının indeksleme yöntemlerini kullansa da önerilen algoritmaların ihtiyaçlarına göre değiştirilebilmektedir.

- *Sonuçların sıralanması:* Önerilen algoritmalar tarafından bulunan kayıtların, kullanıcıya gösterilmesinden önce anahtar kelimeye en yakın olan kayıtların ilk sıralara taşınması işlemidir.

2.1. Problemin Tanımı

Günümüz bilgi çağında her türlü bilgiye internet ortamından rahatça ulaşılabilmektedir. Hatta ihtiyaç olan bilginin tam olarak tanımlanamaması bile bilgiye ulaşmak için engel değildir. Elde edilmek istenen bilgi hakkında bir kaç kelime ile sınırlı anahtar kelimenin bilinmesi bilgiyi ulaşılabilecek şekle

dönüştürür. Bu kadar kısıtlı ön bilgi ile bilginin detaylarına ulaşabilme sıradan kullanıcılar için oldukça önemlidir.

Bilginin artması bilginin depolandığı alanların daha performanslı bir şekilde kullanılma ihtiyacını doğurmuştur. Bu amaçla bilginin depolama alanları olan veri tabanı geliştiricileri veriye daha hızlı ulaşılabilecek ve verinin daha az yer kaplamasını sağlayacak yöntem olan veri tabanı normalizasyonu yöntemini kullanmışlardır [17]. Veri tabanlarında gerçekleştirilen normalizasyon yöntemi ile veriler tek bir tablo içerisinde uzun uzun yer almak yerine bilginin farklı özellikleri farklı tablolarda yer alır. Bu şekilde ortak özelliğe sahip veriler için veri tabanında bilgi tekrarından kurtulunmuş olunur. Veri tabanlarına bu özelliğin kazandırılması veri tabanlarında yapılacak olan arama işlemini de sıradan bir kullanıcı için güçleştirmiş hata imkânsızlaştırmıştır.

Normalizasyon uygulanmamış veri tabanı tablo örneğini Şekil 2.2'de görebiliriz. Görüldüğü gibi bir öğrenciye ait bilgiler tek bir tablo içerisinde verilmiştir. Şekil 2.3 Şekil 2.2'deki tabloya ait kayıt örneğini göstermektedir. Kayıtlardan da görüleceği gibi bir öğrencinin birden fazla ders alması durumunda aynı kişi için ad, soyad, tc kimlik no bilgilerini içeren veriler tekrar etmektedir. Bu durum veri tabanları için hem performans kaybına hem de büyük depolama alanları ihtiyacına sebep olmaktadır. Bu nedenle veri tabanlarında normalizasyon işlemine ihtiyaç duyulmuştur. Fakat normalizasyon gerçekleşmemiş bir yapı için en büyük avantaj istenilen bilgiye çok kolay ulaşmaktır. Şekil 2.3'de görüldüğü gibi Ayşe hakkında bilgi edinilmek istenirse sadece Ayşe isminin geçtiği satırlar bize bu bilgiyi sağlayacaktır.

Ogrenci
Id
OgrenciNo
Adi
Soyadi
TcKimlikNo
DersKodu
DersAdi
AldigiNot
Sinifi
OgretmenSicilNo
OgretmenAdi
OgretmenSoyadi
OgretmenTcKimlikNo

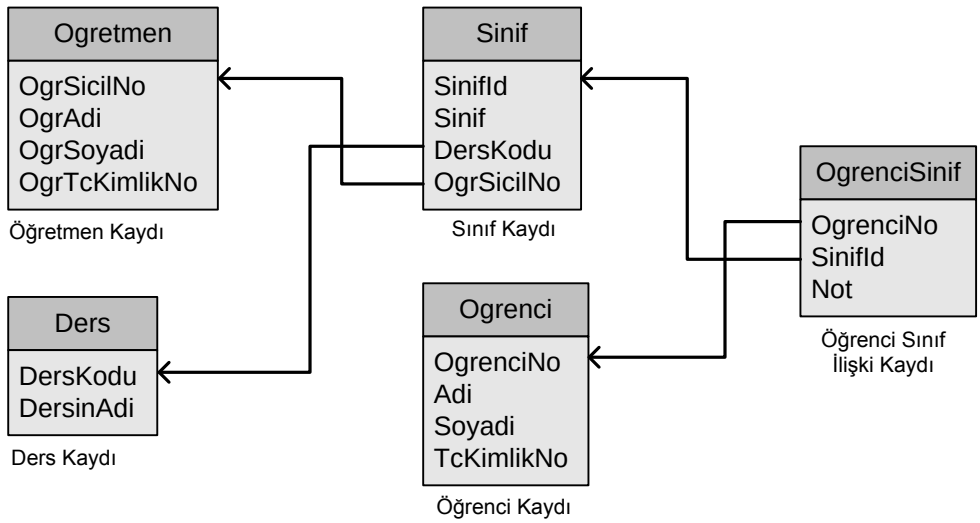
Öğrenci Tablosu

Şekil 2.2. Normalizasyon yapılmamış tablo örneği

Id	OgrenciNo	Adi	Soyadi	TcKimlikNo	DersKodu	DersAdi	AldigiNot	Sinifi	OgrSicilNo	OgrAdi	OgrSoyadi	OgrTcKimlikNo
1	1	Ayşe	Güzel	12345678912	101	Fizik	100	1	123	Ahmet	Çalış	32145675673
2	1	Ayşe	Güzel	12345678912	110	Kimya	80	1	234	Ali	Doğru	43256712349
3	2	Can	Atıl	21232343567	101	Fizik	90	1	123	Ahmet	Çalış	32145675673

Şekil 2.3. Normalizasyon yapılmamış tablo için kayıt örneği

Fakat veri tabanında normalizasyon işleminin gerçekleştirilmesi istenilen bilginin aranmasını güçleştirmiştir. Çünkü bilgi tek bir satırda değil farklı tabloların farklı satırlarına dağılmış durumdadır. Normalizasyon uygulanmış veri tabanı örneği Şekil 2.4'de gösterildiği gibidir. Şekil 2.2'de bir öğrenciye ait sınıf, öğretmen ve ders bilgisi tek bir satırda ifade edilirken Şekil 2.4'deki tablo yapısı bu tek satırı altı farklı tablodan elde etmektedir. Altı farklı tablodan bilgi elde etmesine rağmen Şekil 2.3'deki kayıt örneği Şekil 2.5'deki kayıt örneğine göre daha fazla yer kaplamaktadır.



Şekil 2.4. Normalizasyon uygulanmış veri tabanı örneği

OgrenciNo	Adi	Soyadi	TcKimlikNo
1	Ayşe	Güzel	12345678912
2	Can	Atıl	21232343567

Öğrenci Kaydı

OgrSicilNo	OgrAdi	OgrSoyadi	OgrTcKimlikNo
123	Ahmet	Çalış	32145675673
234	Ali	Doğru	43256712349

Öğretmen Kaydı

DersKodu	DersAdi
101	Fizik
110	Kimya

Ders Kaydı

SinifId	Sinif	DersKodu	OgrSicilNo
1	1	101	123
2	1	110	234

Sınıf Kaydı

OgrenciNo	SinifId	Not
1	1	100
2	1	90
1	2	80

Öğrenci Sınıf İlişki Kaydı

Şekil 2.5. Normalizasyon uygulanmış tablo kayıt örneği

Daha önce de belirtildiği gibi normalizasyon işlemi veri tabanında arama işlemini güçleştirmektedir. Örneğin Şekil 2.5'deki kayıt örneğinde Ayşe hakkında bilgiye ulaşılmak istensin. Bu bilgi beş farklı tablodan geleceğinden tablolar arası ilişki ve bilgiyi elde edecek SQL sorgu dili bilinmek zorundadır. Bu durumda sıradan kullanıcılar için ilişkisel veri tabanlarında istenilen bilginin elde edilmesi imkânsızlaşmıştır. Bu da ilişkisel veri tabanlarında anahtar kelime arama problemini ortaya çıkarmıştır.

2.2. Önemi

İlişkisel veri tabanlarında anahtar kelime arama özellikle bilgilerini veri tabanlarında saklayan tüm kuruluşlar için çok önemlidir. Çalışan tüm personelin veri tabanı sorgulama dili olan SQL sorgu dilini bilmediği kabul edilirse ilişkisel veri tabanlarında anahtar kelime aramanın önemi artmaktadır.

Her çalışana her ihtiyacı olduğu bilgiye yönelik ayrı bir raporun olması veya her ihtiyaç duyulan bilgi için yeni bir raporun hazırlanması hem performans hem de iş gücü kaybına neden olacağından mümkün değildir.

Özellikle sürekli güncel veri sorgulama ihtiyacı olduğu yerlerde herhangi bir rapora ihtiyaç duyulmaksızın ilişkisel veri tabanlarından anahtar kelime arayan uygulamaların geliştirilmesi önemlidir. Özellikle hayati önem taşıyan verilerin tutulduğu hastane veri tabanlarında anahtar kelime arayan uygulamalar raporlama ihtiyacını büyük oranda giderecektir.

2.3. Katkılar

Literatürde ilişkisel veri tabanlarında anahtar kelime arama alanında çeşitli çalışmalar yapılmıştır [1,2,3,4,5,6,7,8,14,16]. Çalışmaların tamamı veri tabanı yapısını bir çizge üzerinde ifade etmiş ve arama işlemini çizge üzerinde gerçekleştirmiştir. Bir çizge üzerinde düğümler verileri veya kayıtları, düğümler arası kenarlar ise veriler veya kayıtların bulunduğu tablolar arası ilişkiyi ifade etmektedir.

Çalışmaların çoğunda indeksleme olarak veri tabanı indekslemesi kullanılmıştır [1,2,4,6,7]. Çalışmalardan BLINKS çizge üzerindeki her düğüm için anahtar kelimeye olan uzaklıkların ve yolların tutulduğu bir indeksleme tabloları oluşturmuş ve bu indeksleme tablolarını kullanmıştır [3]. DBxplorer yönteminde de veri tabanındaki her bir kelime için kelimenin bulunduğu

sütün, kolon ve hücre bilgilerini tutan indeksleme listeleri oluşturulur [6]. Aramada bu listeler kullanılmıştır. Kendi indeksleme sistemlerini oluşturan çalışmaların diğer çalışmalara göre daha hızlı arama yapıp daha hızlı sonuca ulaşma gibi bir avantajları mevcutken, veri tabanında meydana gelen değişikler için sürekli güncel tutulmaları da bir dezavantaj oluşturmaktadır.

Literatürdeki çalışmalarda sonuçları sıralama kriteri olarak da farklı yöntemlerden bahsedilmektedir. Bunlardan biri bulunan sonuç ağaçlarının düğümleri arası kenar ağırlıklarının toplamına göre veya bulunan sonuç ağaçlarındaki mevcut bağlantı sayısına göredir. Kenar ağırlıklarının hesaplanması veya bağlantı sayısının hesaplanması çalışmalarda farklılık göstermektedir.

Literatürde gerçekleştirilen çalışmalardan biri olan Rsearch yöntemi diğer yöntemlerden farklı olarak çift kayıt problemini ele almıştır [7]. Bu problem veri tabanlarında oluşabilecek aynı verinin tekrarının oluşması durumudur. Bu durumda yapılacak arama işleminin tek satır üzerinden değil de tekrarlamış olabilecek kayıtlar üzerinden de yapılmasını içerir.

Yapılan çalışmalar tablolar arası ilişkiyi dış anahtar → birincil anahtar ilişkisine göre kurarak ilgili kayıtları bulmaktadır. Bu şekilde tablolar arası açık olarak görülebilen ilişkiler sonuca yansıtılmış olur. Ancak, veri tabanı yapısı içinde açıkça görünmese de telefon numarası, e-posta adresi gibi bazı alanlar üzeriden ilişki kurarak gizli ilişkilerde ortaya çıkartılabilir. Bu tezde, veri tabanında mevcut olan dış anahtar → birincil anahtar ilişkisinin yanında tanımlanmış olan belirleyici alanlar da ilişkilerin tanımlanmasında kullanılarak, kayıtlar arası gizli ilişkilerin de ortaya çıkarılması sağlanmıştır.

3. İLİŞKİSEL VERİ TABANLARINDA ANAHTAR KELİME ARAMA ÜZERİNE YAPILMIŞ ÇALIŞMALAR

İlişkisel veri tabanlarında anahtar kelime arama konusunda çeşitli çalışmalar gerçekleştirilmiştir. Bu çalışmalardan önemlileri sırasıyla açıklanmıştır. Literatür çalışmalarından da görüleceği gibi yapılan çalışmalar bazı ortak yöntemlere sahiptirler. Örneğin çalışmaların çoğu veri tabanında arama gerçekleştirirken bir çizge yapısından yararlanmıştır. Yine birçoğu kendilerinin tasarlamış olduğu bir indeksleme sistemini kullanmışlardır. Sonuçların sıralanmasında da her uygulama geliştirilen uygulamaya uygun olarak bir sıralama kriteri belirlemiş ve onu kullanmıştır.

Yapılan çalışmalarda en büyük farkı indeksleme yöntemleri ve sonuçların sıralanmasında kullanılan kriter oluşturmuştur. Aynı zamanda her çalışma veri tabanı aramasında karşılaşılabilecek bir soruna çözüm aramıştır.

Çalışmalar gösteriyor ki henüz ilişkisel veri tabanlarında arama yaparken SQL veri tabanı sorgu dilinin gösterdiği başarıyı gösteremiyor. Fakat veri tabanı bilgisine ihtiyaç duyan her kişinin veri tabanı yapısını ve SQL sorgu dilinin bilmesinin imkânsızlığı ilişkisel veri tabanlarında anahtar kelime arama çalışmalarını devam ettirmektedir.

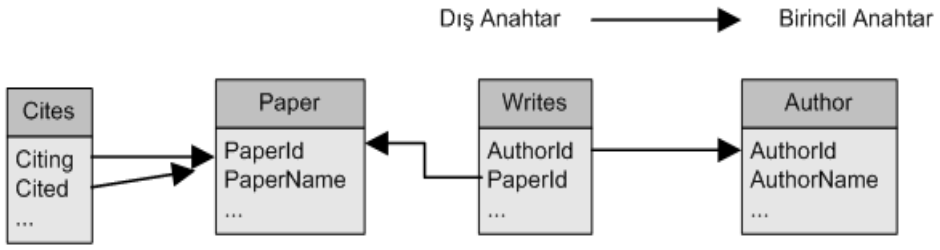
İlişkisel veri tabanlarında anahtar kelime arama uygulamalarını diğer bir ortak özelliği de geliştirilen uygulamaların veri tabanına özel uygulamalar olmasıdır. Çünkü veri tabanının indeks yapısı, tablo ilişkileri, tablolarda tutulan veri tipleri, veri tabanının büyüklüğü, veri tabanındaki tutulan verinin düzgünlü vs. geliştirilen uygulamayı etkileyecektir.

İncelenen makalelerde de görüleceği gibi çalışmalar örnek veri tabanları üzerinde geliştirilmiştir.

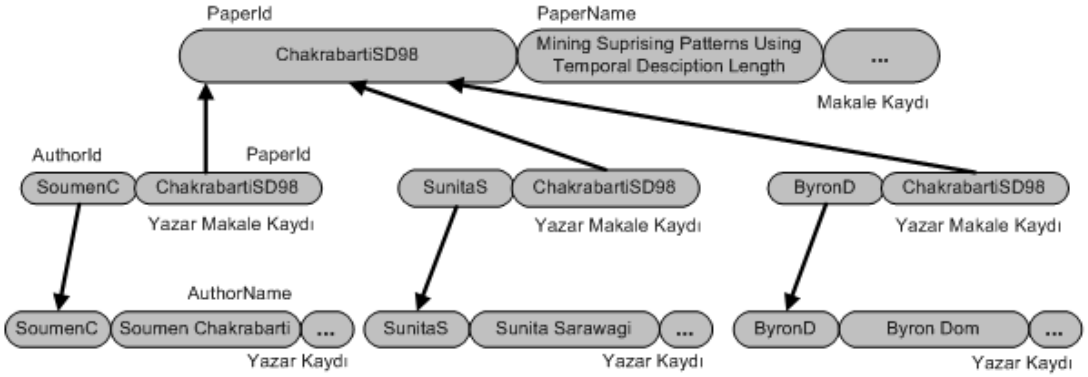
3.1. BANKS Yöntemi

Banks modelinde veri tabanı bir çizge olarak tanımlanmış ve kayıtlar çizgenin düğümlerini oluşturmuştur [1]. Düğümler arası bağlantılar da dış anahtar → birincil anahtar ilişkisini göstermektedir.

İlişkisel veri tabanlarında normalizasyondan dolayı anahtar kelimeler ile anahtar kelimeleri tamamlayan diğer bilgiler tek bir tablo veya kayıtta yer almaz. Birden fazla tablo ve kayıtlara bölünmüş durumdadır. Bu nedenle veri tabanında arama yapabilmek için veri tabanındaki verilerin aramaya uygun bir yapıda ifade edilmesi gerekmektedir.



Şekil 3.1. Veri tabanı şeması [1]



Şekil 3.2. Kayıt örneği [1]

Örneğin makalede incelenen veri tabanı şeması Şekil 3.1'de gösterildiği gibidir. Veri tabanı bir makaleye ait bilgileri içermektedir. Veri tabanında yapılan normalizasyon işlemi ile makale başlıkları, onların yazarları ve

referansları farklı tablolarda yer almaktadır. Tablolar arası ilişki de Şekil 3.1'de gösterildiği gibidir. Şekil 3.2 de veri tabanının bir parçasının çizge üzerindeki gösterimini ifade etmektedir. Şekil 3.2 makale başlığı ve yazar hakkında kısmı bilgi içermektedir. Şekilde de görüldüğü gibi bir kayıt birbiri ile birincil anahtar - dış anahtar ilişkisi ile bağlı yedi farklı tablodaki verilerden oluşmaktadır. Örnekteki ChakrabartiSD98 paperId ile tanımlı makaleyi arayan bir kullanıcının anahtar kelime olarak "sunita temporal" veya "soumen sunita" anahtar kelime çiftlerini vermesi yeterlidir. Anahtar kelime bazlı aramalarda anahtar kelimeleri içeren ve anahtar kelimeler ile ilişkide olan kayıtlar sonuç olarak bulunmaktadır.

Banks yönteminde anahtar kelimeler verilerde arandığı gibi aynı zamanda kolon isimlerinde ve ilişki isimlerinde de aranır. Banks, veri tabanını çizge üzerinde Şekil 3.2'de gösterildiği gibi tanımlar. Aynı zamanda Banks veri tabanını dış anahtar → birincil anahtar olmak üzere yönlü bir çizge olarak tanımlar. Çizge üzerinde iki kayıt arasındaki ilişki ağırlıklandırılır. Fakat Bank modeli yönlü çizge modelinde iki kayıt arasındaki yönlü ilişkiyi tanımlarken aynı zamanda bu ilişkinin tersi yönde bir ilişkinin de varlığını kabul etmektedir. Bu nedenle her ilişkiyi tanılarken geriye doğru ikinci bir ilişkiyi de tanımlar. Örneğin Şekil 3.2 incelenecek olursa SunitaS, SoumenC ve ByronD kayıtları için PaperId alanı ChakrabartiSD98 olan kayıt ortak bir düğümdür. Eğer ters yönlü ilişki tanımlanmamış olsa idi bu üç yazar arasındaki ilişkiye ulaşılamazdı. Ters yönlü ilişkinin tanımlanması ile üç yazar arasındaki ilişki ortaya çıkmıştır.

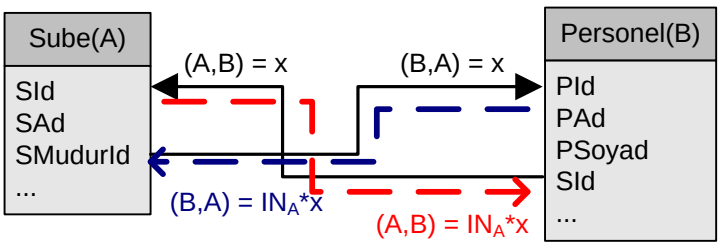
Bank kayıtlar arası yönlü dış anahtar → birincil anahtar ilişkisini sabit bir değer ile tanımlarken bunun tersi ilişkiyi düğüm prestiji ile düğümler arası kenara verilen sabit değerin çarpımı şeklinde tanımlamıştır. Bu şekilde her kenarın her iki yönde de en az bir ağırlığı mevcuttur. Makale düğüm prestijini de o düğüme gelen bağlantı sayısı olarak tanımlar. Yani ilgili kaydın indegree sayısı o kaydın düğüm prestijini verir.

Banks yöntemi iki kayıt arasındaki karşılıklı dış anahtar → birincil anahtar ilişkisi içerisindeki iki kaydın kenar ağırlığı için tek yönde hesaplanan kenar ağırlıklarından küçük olanı kabul etmiştir. Örneğin Şekil 3.3 karşılıklı ilişki içerisindeki iki tabloyu göstermektedir. Bu tablolarda yer alan kayıtlar arasında birbirini referans eden kayıtlar mevcuttur. BANKS yöntemi bu tip kayıtlar için aynı yönde iki ilişki tanımlar ve bunlardan küçük olanını kullanır. Örneğin iki düğüm A ve B arasında A'dan B'ye ve B'den A'ya olacak şekilde dış anahtar → birincil anahtar ilişkisi olsun. Bu durumda iki düğüm arasındaki ilişkiyi gösteren kenar hem ileri yönde hem de tersi yönde iki ağırlık değerine sahip olacaktır. Bu ilişkilerden A'dan B'ye dış anahtar → birincil anahtar ilişkisini gösteren kenar ağırlığını ele alacak olursak; A'dan B'ye dış anahtar → birincil anahtar ilişkisi için A ile B arasındaki ileri yöndeki kenar ağırlığı için belirlenmiş sabit değer ve B ile A arasındaki ters yöndeki kenar ağarlığı için hesaplanmış B ile A arasındaki ileri yön ilişkinin sabit değer ile B düğümünün düğüm prestij değerinin çarpımı şeklinde iki değer elde edilecektir. A ile B arasındaki ileri yöndeki kenar ağırlığının sabit değeri x kabul edilirse B ile A arasındaki tersi yöndeki kenar ağırlığı da IN_B ile x değerinin çarpımı olacaktır.

$(A,B) = x$

$(B,A) = IN_A * x$

Bu değerlerden küçük olanı A'dan B'ye dış anahtar → birincil anahtar ilişkisinin kenar ağırlığını vermektedir.



Şekil 3.3. Karşılıklı ilişki örneği

Banks modeli oluşturmuş olduğu bu yönlü ve ağırlıklandırılmış çizge üzerinde verilen anahtar kelimeleri arar. Her anahtar kelime için o anahtar kelimenin bulunduğu kayıtları içeren düğümlerden birer küme oluşturur. Örneğin “a” anahtar kelimesini içeren A,D,E düğümleri bir kümeyi oluştururken “b” anahtar kelimesini içeren A,F,C düğümleri diğer düğüm kümesini oluşturmaktadır. Her anahtar kelime için bir düğüm kümesi oluşturulmakta ve anahtar kelime sayısı kadar düğüm kümesi oluşmaktadır.

Banks oluşan bu düğüm kümelerinin her bir elemanından başlayarak tüm anahtar kelimeleri içerecek şekilde çizge üzerinde bir yol bulur. Bu yol bulma işlemini düğüm kümelerinin her bir elemanı için gerçekleştirir. Oluşan bu yolların kesişim düğümleri oluşturulacak olan cevap çizgesinin kök düğümlerini belirlemektedir. Bulunan her bir kesişim düğümü kök düğümü olacak şekilde ve her ağaç tüm anahtar düğümleri içerecek şekilde oluşturulan ağaçlar da cevap ağaçlarını vermektedir. Oluşan bu cevap ağaçları sahip oldukları ağırlıklara göre sıralanır ve cevapları oluşturur.

Cevap ağaçlarının ağırlıkları da düğüm ağırlığı ve bulunan kenar ağırlıklarının toplamı şeklinde ifade edilmektedir.

Düğüm ağırlığı da ilgili düğüme gelen bağlantı sayısını vermektedir. Örneğin “a” kaydı ile “b” kaydı arasında (a,b) ilişkisi tanılı olsun. Bunun tersi ilişki (b,a) şeklindedir. (b,a) ters ilişkisinin ağırlığı hesaplanırken kullanılan düğüm prestiji a kaydının bulunduğu tablodan b kaydına gelen ilişki sayısına eşittir.

Fakat b kaydının bulunduğu düğümün düğüm ağırlığı ise b kaydına diğer tüm tablolardan gelen ilişki sayısına eşittir.

Bank yönteminin en büyük problemi cevap ağaçlarını oluşturma sırasında çok büyük cevap ağaçlarının oluşabilmesi olmuştur.

3.2. İki Yönlü Arama (Bidirectional Expansion For Keyword Search) Yöntemi

Çizge üzerinde anahtar kelime aramada asıl problem veri çizge üzerinden istenen en iyi küçük bir sonuç ağacının elde edilmesidir. Geriye genişleyen (backward expanding) arama algoritması anahtar kelimeyi içeren düğümlerden başlayarak birbirini izleyen köklere doğru çalışır [2,15]. Bu arama algoritması genellikle düz yazıya dayalı aramalarda kullanılır. Fakat bu arama algoritması anahtar kelime birçok düğümde yer alıyor veya düğümün derecesi çok fazla ise iyi bir performans göstermez.

Makale geriye genişleyen arama algoritmasından yaralanarak olası bir kökten ileri arama yapacak şekilde bir algoritma geliştirilmiştir [2,15]. Bu algoritmanın esnekliğinden yararlanmak için yayma hareketine dayanan uç önceliklendirme tekniği olarak yeni bir arama tasarlamışlardır.

[2] numaralı çalışmada geliştirilen algoritma “İki yönlü arama (bidirectional search)” olarak adlandırılmıştır. Çizge üzerinde agnostik çizelge (schema-agnostic) metin aramak için tasarlanmıştır. Makalede sunulan algoritma cevap ağacı olabilecek olası kökler olan düğümlerden ileri bir yol izleyerek arama yapar. Örneğin “transaction” anahtar kelimesi birçok düğümle ilişkilendirilip buna karşın “Gray” anahtar kelimesi de daha az sayıda düğüm ile ilişkilendirilirse her iki anahtar kelimeden de ileriye doğru bir arama yapılır ve her iki yolun kesiştiği yollar çözümü oluşturur.

Makalede oluşturulmuş olan çizge yönlü ve ağırlıklandırılmış bir çizgedir. Çizge üzerinde düğümler varlıkları, kenarlar ise ilişkileri temsil etmektedir. Düğümler bir veri tabanındaki veriyi ya da satırı gösterirken kenarlar ise birincil anahtar → dış anahtar ilişkisini göstermektedir. Dış anahtar ilişkisi ile bağlı olan her düğüm arasında yönlü bir kenar vardır. Yönlü bir kenar kullanılmasının amacı her bağlantının eşit ağırlıkta olmamasıdır.

Basit olarak makalede uygulanan arama yöntemi yönlü bir çizge üzerinde her bir anahtar kelimeyi içeren düğümlerin bulunması ve bulunan düğümler arası ilişkiden cevap için cevap dizin ağaçlarının oluşturulmasıdır. Makalede kullanılan yöntemde yönlü bir çizge kullanıldığından oluşan cevap dizin ağaçları da yönlü olmaktadır. Dizin ağaçlarında anahtar kelimelerin bulunduğu düğümler arası yol anahtar kelimelerin veri tabanındaki ilişkisini açıklamaktadır.

Makale cevapların sıralanmasından çok arama algoritması üzerinde durmuştur. Sıralama için BANKS ve ObjectRank yöntemlerinde kullanılan sıralama algoritması kenar ağırlıklarını ve düğüm prestijini değerlerini kullanmaktadır [1,14]. Bu yöntemde düğüm prestiji bir düğüme gelen ilişki sayısını yani bir düğümün diğer düğümler tarafından referans gösterilme sayısını ifade etmektedir. Kenar ağırlığı ise bir birincil anahtar ile dış anahtar ilişkisi için sabit bir değeri göstermektedir. Makalede sıralama kriteri olarak kenar ağırlıklandırması veri tabanı şemasında tanımlanan ileri yönlü kenarlar için varsayılan değer 1 kabul edilmiştir. BANKS ve ObjectRank yöntemlerinde kullanılan sıralama algoritması bu iki değeri kullanarak her kenar için bir ağırlık belirler [1,14]. Örneğin A ve B olan iki düğüm düşünelim. A düğümünden B düğümüne bir bağlantı olsun. Bu durumda A düğümündeki dış anahtar ile B düğümündeki birincil anahtar bağlantısından kenarın makalede belirlenen sabit bir değeri olacaktır. Makale, eğer ileri yönlü bir ilişki varsa bunun tersine bir ilişkinin olduğunu düşünmüş ve ileri bağlantının tersi yönünde bir bağlantı daha tanımlamıştır ve bunu da geriye kenar olarak

adlandırmıştır. Geriye kenarın ağırlığı da ileri kenara verilen sabit değer ve düğüm prestijinin çarpımı şeklinde hesaplanır.

A düğümü ile B düğümü arasındaki ilişkinin yanı sıra B düğümü ile A düğümü arasında da bir ilişki olduğu durumda A ve B arasındaki dış anahtar → birincil anahtar kenar ağırlığının hem A ve B arasındaki dış anahtar → birincil anahtar ilişkisinden sabit bir değeri olacak hem de B ile A arasındaki ters ilişki nedeniyle A düğümünün prestiji ile B ile A arasındaki dış anahtar → birincil anahtar ilişkinin kenar ağırlığı çarpımından bir değeri olacaktır. Bu durumda elde edilen değerlerden küçük olan değer A'dan B'ye yönlü ilişkiyi gösteren kenarın ağırlığını ifade etmektedir.

Herhangi bir anahtar kelimeye sahip düğüm için oluşturulan yolun ağırlığı kök düğümden anahtar kelimenin bulunduğu yaprak düğüme kadar olan kenar ve düğüm ağırlıklarının toplamı şeklindedir.

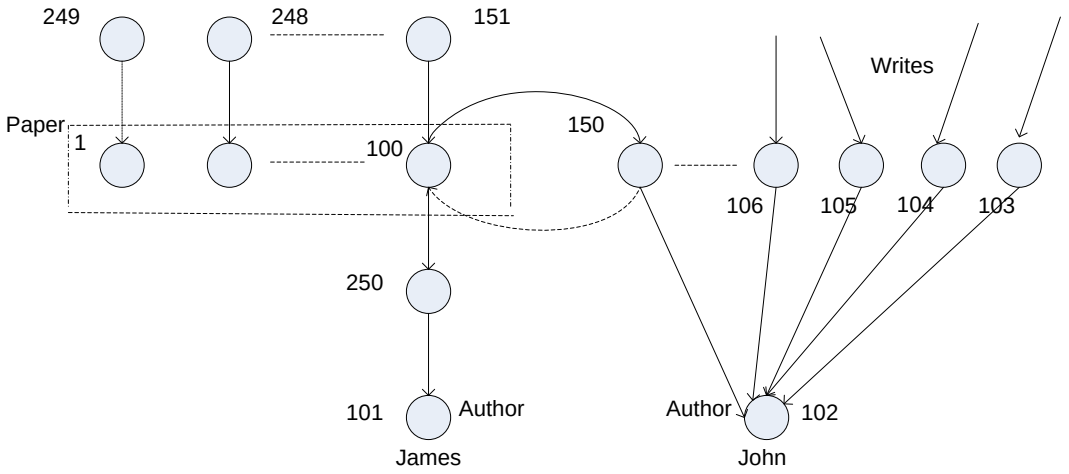
İki yönlü arama yöntemi cevapların sıralanmasında mevcut literatür çalışmalarından yararlanmıştır [2]. İki yönlü arama yöntemi bunun sebebini bu alanda yapılmış olan çalışmaların yetersiz olması şeklinde açıklamaktadır [2].

Makalenin örnek olarak ele aldığı ve kendi yöntemini dayandırdığı arama algoritması geriye genişleyerek arama yöntemini inceleyecek olursak; bu yöntem arama kriteri olarak verilen tüm kelimeleri öncelikle düğümlerde arar [2,15]. Anahtar kelimelerin bulunduğu düğümler anahtar kelime düğümü olarak adlandırılırlar. Bulunan her anahtar kelimenin hangi tablo veya kayıtta geçtiğinin bilinmesinin kolaylaştırmak için anahtar kelime ve tablo ismi ya da kayıt kimliğini tutan bir harita tanımlanır.

Anahtar kelimelerin bulunduğu düğümler her bir anahtar kelime için kümelenir. Bu şekilde anahtar kelime sayısı kadar bunların bulunduğu düğümleri içeren düğüm kümeleri oluşmuş olur. Düğüm kümelerinin elemanı

olan her düğümden diğer anahtar kelimeleri içerecek şekilde çizge üzerinde bir yol çizilir. Bu yol bulma tüm düğüm kümesi elemanları için gerçekleştirilir.

Geriye Genişleyen algoritması (Backward Expanding) her iterasyonda bütün anahtar kelime düğümleri ile ilişkilendirilir. Eğer herhangi bir iterasyon herhangi bir düğüme ulaşmak için uzun bir yok çizerse bu durumda da algoritma birçok düğümü aramak durumunda kalır bu da cevap ağacının büyümesine neden olur.



Şekil 3.4. İki yönlü arama örneği [2]

Makalede önerilen iki yönlü arama algoritması geriye genişleyerek arama algoritmasında olduğu gibi veri tabanı çizgesini parçalayarak küçük cevap ağaçları oluşturmayı hedefler [2,15]. Fakat Geriye Genişleyen algoritmasının büyük cevaplar oluşturma ihtimalini yok etmeyi hedefler. Bunun için çoklu iterasyonu değil tekli iterasyonu kullanır. Yani her anahtar düğümlerden tüm anahtar düğümlere ulaşmak yerine farklı anahtar kelimelerin bulunduğu anahtar düğümlerin hepsinden aynı anda köke doğru aramaya başlar ve hepsinin kesişimi bir cevap anahtarını oluşturur. Farklı anahtar kelimelerin bulunduğu anahtar düğümlerin kombinasyonları diğer cevap ağaçlarını oluşturur. Şekil 3.4 makalede önerilen Biderectional arama yöntemini göstermektedir. Kullanıcının James ve John yazarlarının yazmış olduğu ortak

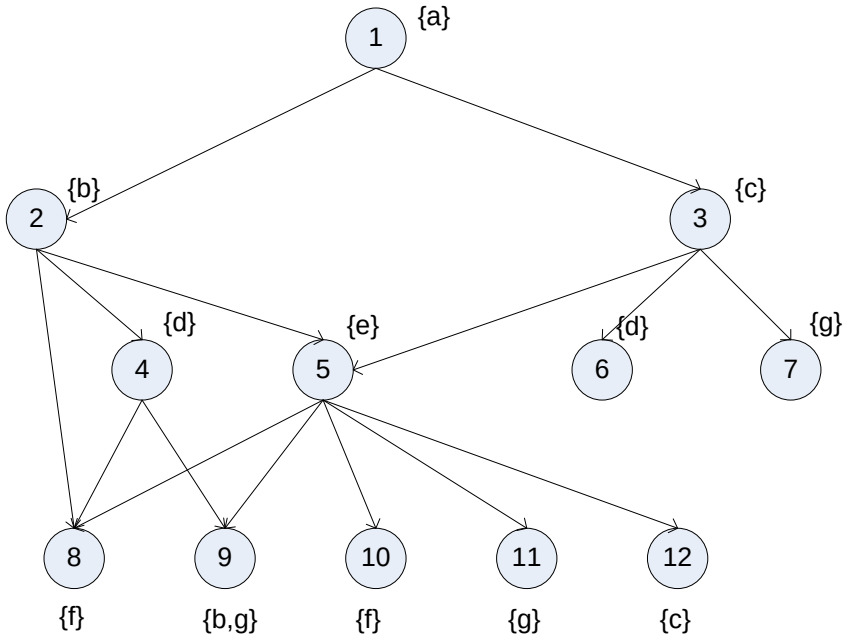
yazıları bulmak için James ve John anahtar kelimelerini aradığı düşünülecek olursa; Şekil 3.4 James ve John anahtar kelimeleri için yapılan aramanın bir iterasyonunu göstermektedir. James ve John anahtar kelimelerinin bulunduğu birer düğümden aynı anda köke doğu ilerlerler. Her adımda anahtar kelimenin bulunduğu düğüme bir sonraki bağlı olan düğümler eklenir. Her iki düğümün ortak düğüme ulaşması ile arama sonlanır. Ortak düğüm kök düğümünü oluşturacak şekilde bir cevap ağacı oluşur. Her iterasyon sonucunda oluşmuş olan cevap ağaçları cevap ağaçlarının ağılıklarına göre sıralanarak sonucu oluşturmuş olur.

Bu yöntemin benzer bir arama tekniği kullanan BANKS yöntemine göre avantajı arama adımlarının kısalmasıdır [1]. BANKS yöntemi aynı arama için bir anahtar kelimenin bulunduğu düğümden diğer anahtarın bulunduğu düğüme ulaşmak için aradaki tüm düğümleri gezmesi gerekirken makalede geliştirilmiş olan yöntem ile aynı arama dört adımda gerçekleştirilmiştir [1].

3.3. Blinks Yöntemi

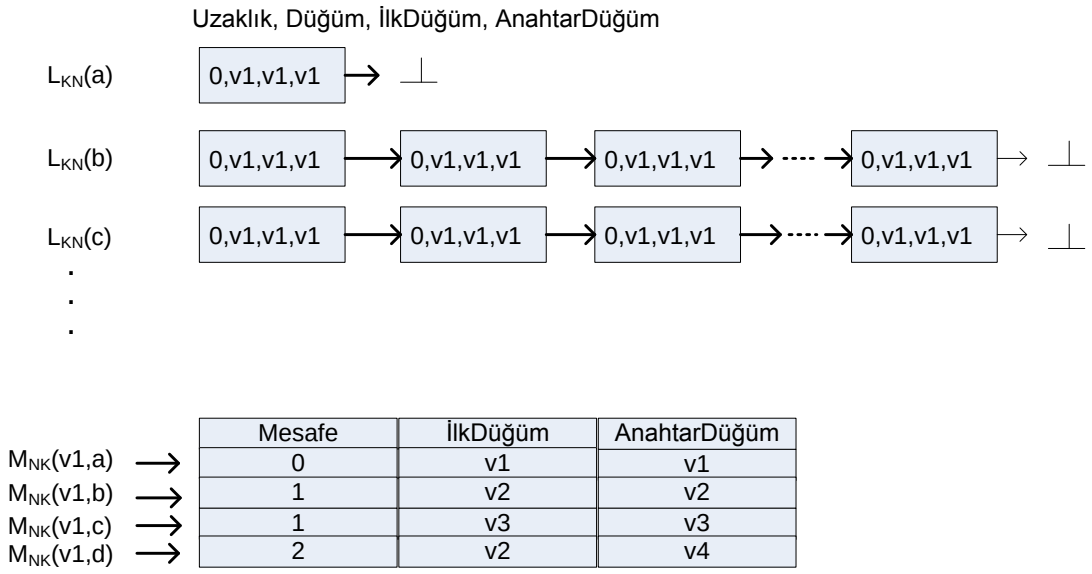
BLINKS yöntemi kullanılmış olan indeksleme yöntemi ve arama algoritması önceden kullanılmış olan tek seviye indeksleme (single-level index) olarak adlandırılan indeksleme yöntemi ve bu indeksleme yöntemi ile yapılmış arama algoritmasından esinlenerek geliştirilmiştir [3]. BLINKS öncelikle tek seviye indeksleme yöntemini ve bu yöntem kullanılarak geliştirilmiş arama algoritmasını incelemiştir [3].

Tek seviye indeksleme olarak adlandırılan indeksleme yönteminde anahtar kelime-düğüm listesi (keyword-node list) ve düğüm–anahtar kelime haritası (node-keyword map) şeklinde adlandırılan iki liste oluşturulmuştur.



Şekil 3.5. Veri tabanı çizgesi [3]

Bu listelerden anahtar kelime–düğüm listesinde bütün düğümlerden her bir anahtar kelimeye yönlü olarak gidilen yolların mesafesi küçükten büyüye dizilmiş şekilde tutulur. Şekil 3.5 veri tabanı çizgesini göstermekte ve bu veri tabanı çizgesi için oluşturulan listeleri de Şekil 3.6 göstermektedir. Anahtar kelime-düğüm listesi $L_{KN(w)}$ olarak ifade edilir ve w anahtar kelimeyi ifade etmektedir. Örneğin $L_{KN(a)}$ listenin bir elemanı, a anahtar kelimesine bütün düğümlerden gelinen yolları ifade etmektedir. Bu a anahtar kelimesine bakıldığında a anahtar kelimesi çizgenin en başında yer alan düğümde yer almaktadır ve sadece kendi bulunduğu düğümden ulaşılabilmektedir. Bu nedenle. $L_{KN(a)}$ $(0,v_1,v_1,v_1)$ şeklinde tek bir elemana sahiptir. Burada 0, a anahtar kelimesine ulaşmak için gerekli olan mesafeyi, ilk v_1 hangi düğümden başlanıldığını, ikici v_1 başlangıç noktası düğümden sonraki düğümü ve son v_1 ise anahtar kelimenin olduğu düğümü göstermektedir. Bu şekilde tüm anahtar kelimeler için tüm düğümlere olan uzaklıklar hesaplanır.



	Mesafe	İlkDüğüm	AnahtarDüğüm
$M_{NK}(v1,a)$ →	0	v1	v1
$M_{NK}(v1,b)$ →	1	v2	v2
$M_{NK}(v1,c)$ →	1	v3	v3
$M_{NK}(v1,d)$ →	2	v2	v4

Şekil 3.6. Anahtar kelime-düğüm listesi ve düğüm-anahtar kelime haritası

İkinci indeksleme listesi olan düğüm–anahtar kelime haritası ise her düğümden bütün anahtar kelime düğümlerine gidilen en kısa yol mesafesini tutar. Şekil 3.5 için tanımlanan bu listeyi Şekil 3.6 göstermektedir. Bu listenin her bir elemanı da $M_{NK(v,w)}$ şeklinde ifade edilir. Burada v her bir düğümü ifade ederken w anahtar kelimeleri ifade etmektedir. Örneğin $M_{NK}(v_1,a)$ için v_1 veri tabanı çizgesinin bir düğümünü ifade ederken a anahtar kelimelerden birini ifade etmektedir. $M_{NK}(v_1,a)$ ise v_1 düğümünden a anahtar kelimesinin bulunduğu düğüme olan en kısa mesafeyi ifade etmektedir. Düğüm–anahtar kelime haritası listesinin her bir elemanı bütün düğümlerden her bir anahtar kelime düğümüne olan en kısa mesafeyi ifade etmektedir. Bunu da (mesafe, ilk düğüm ve anahtar kelime düğümü) şekilde gösterir. Örneğin listenin. $M_{NK}(v_1,a)$ elemanı $(0,v_1,v_1)$ bilgisine sahiptir. Bu bilgi v_1 düğümünden a anahtar kelimesinin bulunduğu v_1 düğümüne olan mesafenin 0 olduğunu ifade eder.

Buradaki düğüm–anahtar kelime haritası ve anahtar kelime–düğüm listesi beraber tek seviye indeksleme tanılamasını oluşturmaktadır.

Makale bu indeksleme yönteminden yola çıkarak performans açısından daha kazançlı olacağını düşündüğü bir yöntem sunmuştur. Tek seviye indeksleme yöntemi küçük veri tabanlarında uygulanabilir bir yöntem fakat büyük veri tabanları için uygulanamaz bir yöntemdir. Bu nedenle makalede veri tabanını bloklara bölme yöntemini kullanan iki seviyeli indeksleme (bi-level index) olarak adlandırılan indeksleme yöntemi kullanılmıştır.

Bu yaklaşımda kayıtlardan oluşan çizge bloklara bölünmüş ve yeni bir indeksleme sistemi oluşturularak arama yapılmıştır. Geliştirilen bu yöntem BLINKS olarak adlandırılmıştır. BLINKS yönteminde iki düzey indeksleme (bi-level index) olarak adlandırılan indeksleme yöntemi kullanılmıştır. Bu indeksleme yöntemi blok içi indeksleme (intra-blok index) ve blok indeksleme yöntemlerini içermektedir.

BLINKS adıyla sunulan yöntem veri tabanını bloklara bölme üzerine kurulmuştur [3]. Bu nedenle veri tabanının bloklara bölünmesi işleminin nasıl gerçekleştirileceğine karar verilmesi gerekmektedir. Çizge bölümlemesi alanında yapılan çalışmalar bölümlemenin iki şekilde yapılabilir olduğunu göstermiştir. Bunlardan biri düğümler arası kenarlar üzerinden yapılan bölme işlemi diğeri ise düğümler üzerinden yapılan bölme işlemidir. Makale düğümler üzerinden yapılan bölme yöntemini kullanmıştır. Makalede bu yöntemin seçilmesi için iki neden öne sürülmüştür. Bunlar;

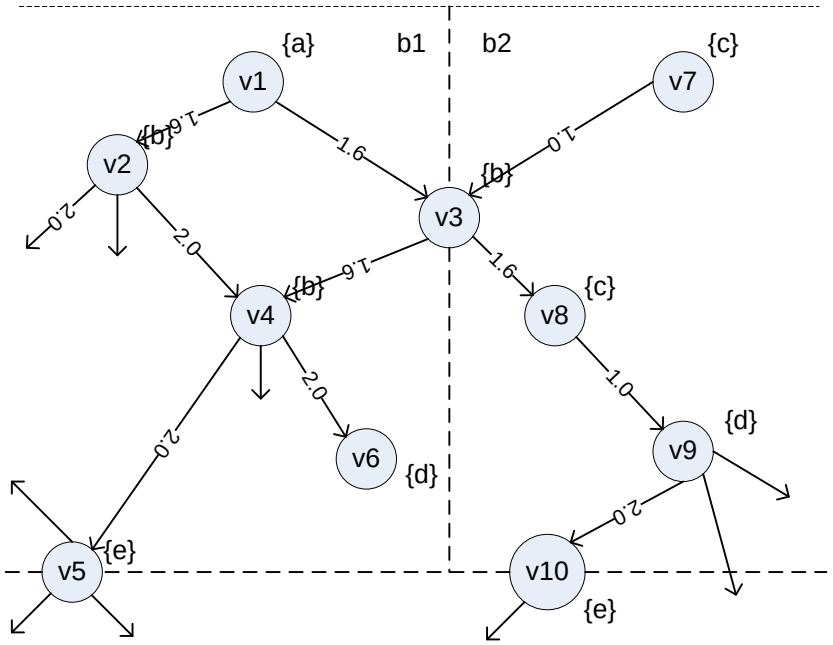
- Düğümler üzerinden yapılacak bölümlemenin daha az olması ve bölümlemede daha az bilgiye ihtiyaç duyulması.

- İlişkisel veri tabanı konu başlığında arama yapılacak noktanın düğümler olması.

Düğümler üzerinden yapılan bölme işleminde iki kavram ortaya çıkmaktadır. Bunlardan biri portal içi (in-portal) diğeri ise portal dışı (out-portal) kavramlarıdır. Bu iki kavramı şu şekilde tanılayabiliriz;

- *Portal içi (in-portal) :* Bir düğüme başka bir bloktan en az bir kenar giriyorsa ve bu düğümden en az bir kenar da aynı bloktaki bir düğüme gidiyorsa bu düğümü o blok için in-portal olarak tanımlayabiliriz.

- *Portal dışı (out-portal) :* Eğer bir düğümden başka bir bloğa en az bir kenar çıkıyor ise ve o bloktaki bir düğümden bir kenar geliyor ise o düğüm blok için out-blok düğüm olarak adlandırılır.



Şekil 3.7. Portal ve blok örneği [3]

Örneğin Şekil 3.7 ele alınacak olursa b_1 bloğu için v_5 düğümü out-portal olarak adlandırılır. Çünkü v_5 düğümünden diğer bloğa iki kenar çıkmış ve aynı bloktaki bir düğümden de v_5 düğümüne bir kenar gelmektedir. Diğer bir düğüm olan v_3 incelenirse bu düğümün b_1 ve b_2 blokları için hem in-blok hem de out-blok olduğu görülebilir.

Düğümler için yapılan bu tanımlamalar da kullanılarak iki düzey indeksleme yönteminin içerdiği blok içi indeksleme yöntemi geliştirilmiştir. Blok için indekslemede dört farklı indeks listesi tanımlanır. Bunlar blok içi anahtar

kelime–düğüm listesi (intra-blok keyword-node list), blok içi düğüm–anahtar kelime haritası (intra-blok node-keyword map), blok içi portal–düğüm listesi (intra-blok portal-node list), blok içi düğüm–portal mesafe haritası (intra-blok node-portal distance map) şeklindedir.

- *Blok içi anahtar kelime–düğüm listesi:* Blok içindeki tüm düğümlerden her anahtar kelimeye olan mesafeleri küçükten büyüye sıralı şekilde tutar ve $L_{KN}(b,w)$ şeklinde ifade edilir.

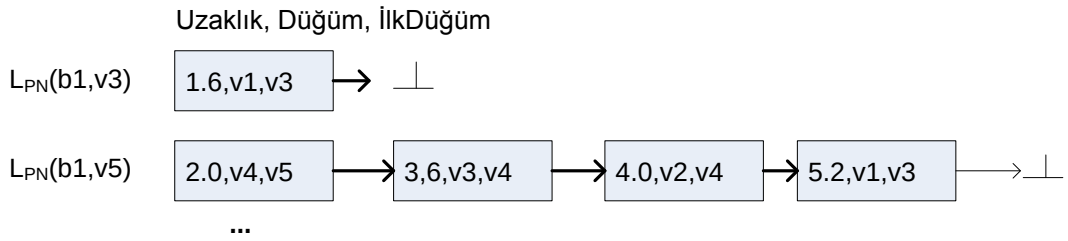
- *Blok içi düğüm–anahtar kelime listesi:* Blok içerisinde diğer bloklara taşmadan her düğümden anahtar kelimelere giden en kısa yol mesafesini tutar ve $M_{NK}(b,u,w)$ şeklinde ifade edilir.

- *Blok içi portal–düğüm listesi:* B bloğunu elemanı olan her out-portal p için $L_{PN}(b,p)$ şeklinde ifade edilir ve blok içindeki her düğümden bloktan ayrılmadan p portal düğümüne olan mesafeleri küçükten büyüye olacak şekilde tutar.

- *Blok içi düğüm–portal mesafe haritası:* B bloğundaki herhangi bir düğüm için $D_{NP}(b,u)$ şeklinde ifade edilir ve b bloğu içerindeki bütün düğümlerden b bloğunun out-portal düğümüne olan en kısa mesafeleri tutar.

Makalede kullanılan blok içi indeksleme yönteminin tek seviye indeksleme yönteminden tek farkı indeksleme listelerinin sadece blok içerisinde oluşturulmuş olmasıdır. Blok içi indeksleme yönteminde blok dışına çıkılamaması farklı bloklarda olup ta iki düğüm arasında ilişki varsa bu ilişkinin çıkarılamamasına neden olur. Bu durumda iki düğüm arasındaki mesafe sonsuz olarak gösterilir. İki düğüm arasındaki bağlantı portal düğüm listeleri sayesinde sağlanır. Portal–düğüm liste örneğini Şekil 3.8 göstermektedir.

BLINK yönteminde kullanılan iki seviye indeksleme yönteminde kullanılan diğer bir indeksleme yöntemi ise blok indeksleme yöntemidir. Blok indeksleme anahtar kelime–blok listesini (keyword-blok list) ve portal-blok listesini (portal-blok list) içerir. B bloğu için portal-düğüm listesi Şekil 3.8'de örneklendiği gibidir.



Şekil 3.8. B bloğunun portal-düğüm listesi

- *Anahtar kelime – blok listesi:* Her anahtar kelime için anahtar kelimeyi içeren blok listesini verir. Örneğin w anahtar kelimesini içeren blok listesi için $L_{KB}(w)$ tanımlaması kullanır. Şekil 3.7 incelenirse a anahtar kelimesi sadece b_1 bloğunda yer almaktadır bu denenle $L_{KB}(a) = \{b_1\}$ olacaktır. Bunun yanında d anahtar kelimesi de hem b_1 hem de b_2 bloğunda yer aldığından $L_{KB}(d) = \{b_1,b_2\}$ şeklinde olacaktır. Aynı durum portal düğümler üzerindeki anahtar kelimeler için de geçerlidir. V_3 portal düğümü b_1 ve b_2 blokları arasında yer aldığından b anahtar kelimesi için $L_{KB}(b) = \{b_1,b_2\}$ şeklinde olacaktır.

- *Portal – blok listesi:* Her portal düğümünün hangi bloklar için out-portal olduğunun listesini verir. Örneğin v_3 portalı hem b_1 hem de b_2 için out-portal bu nedenle $L_{PB}(v_3)=\{b_1,b_2\}$ şeklinde tanımlanır. Bunun yanında v_5 sadece b_1 için out-portal olduğu için $L_{PB}(v_5) = \{b_1\}$ şeklinde tanımlanır.

İlk aşama olarak çizge yapısı bloklara bölünmüştür. Makaledeki yöntemde çizgenin bloklara bölünmesi METIS-Based Partitioning algoritması ile gerçekleştirilmiştir. METIS-Based Partitioning algoritması çizgeleri kenarları

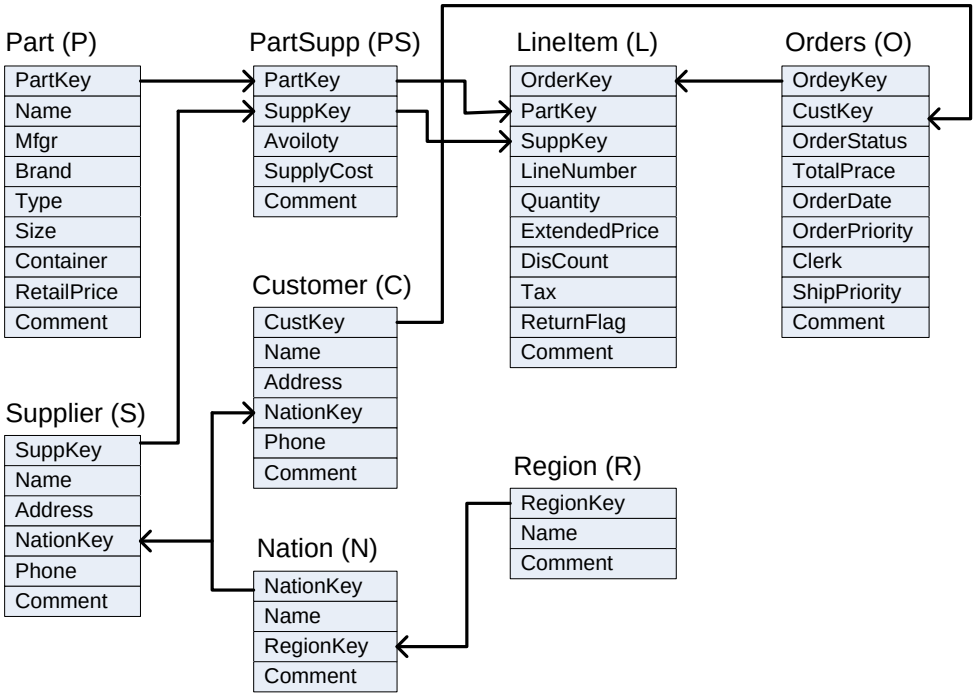
baz alarak bloklara böler daha sonra da node-based Partitioning algoritması ile düğüme dayalı bloklara bölme dönüşümü yapılır. Öncelikle kenara dayalı bölmenin yapılmasının sebebi düğüme dayalı bölümleme çok karmaşık olması ve bu katmaşıklığın azaltılmak istenmesidir. Kenara dayalı bölümleme algoritması her bloktaki düğüm sayısını ve blok ağırlığını eşit tutmayı hedefler.

Bu şekilde çizgenin bloklara bölünmesi ile özellikle büyük veritabanlarında işlem kolaylığı sağlanmıştır. Çok büyük veritabanlarında bütün düğümlerden bir anahtar kelimeye olan yollarının çizilmesi çok zaman alacağından ve karmaşıklık yaratacağından veri tabanı çizgesinin bloklara bölünmesi ve bu bloklara kendi içerisinde indeksleme algoritmasının uygulanması özellikle büyük veri tabanlarında işlem kolaylığı sağlamıştır.

Aranan anahtar kelimenin portal-düğüm listesi sayesinde hangi blok içerisinde olduğu bilgisi sonucuna ulaşılır. Anahtar kelimenin bulunduğu bloklarda indeksleme işlemi yapıldıktan sonra farklı bloklardaki anahtar kelimeler arası bağlantı düğüm-portal mesafe haritası listeleri sayesinde sağlanır.

3.4. Keşif (Discovery) Yöntemi

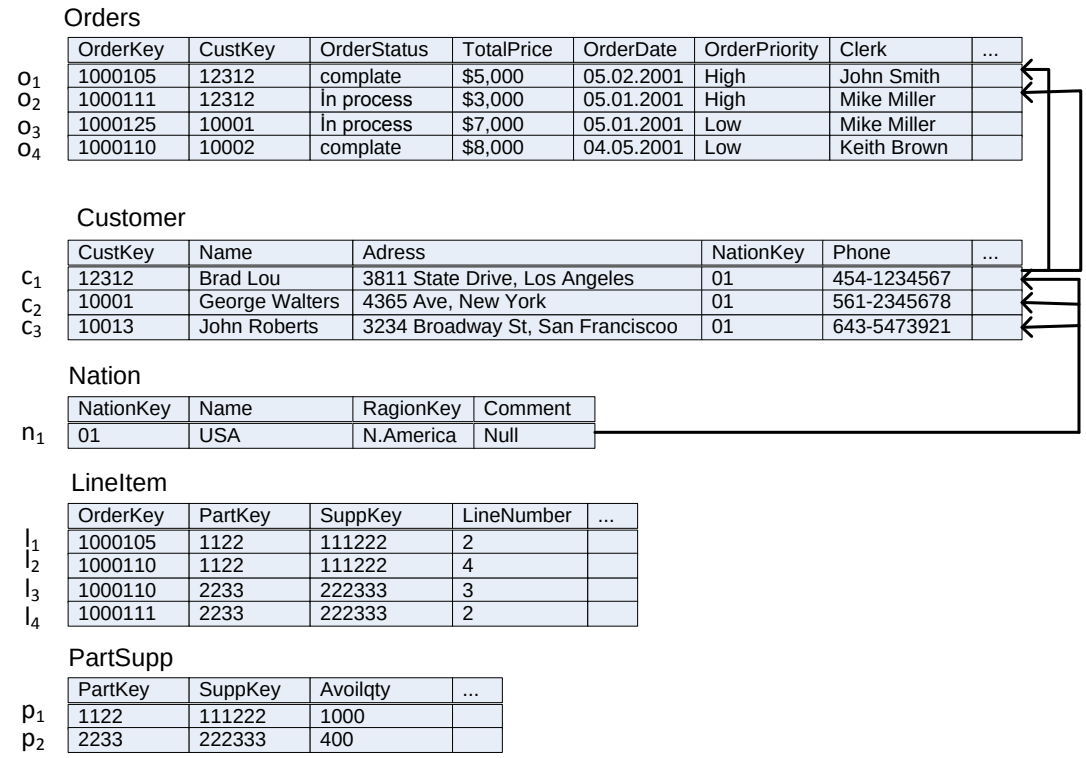
Discovery yönteminde Oracle 8i veri tabanı kullanılmış ve Oracle'ın indeksleme servisi Master İndex kullanılmış [4,18]. Discovery yönteminde öncelikle sisteme verilen anahtar kelimeler hangi tabloların hangi kayıtlarında geçiyor bilgisi elde edilir. Bu amaçla kullanılan ana indeks anahtar kelimelerin geçtiği kayıtları verir. Örneğin Şekil 3.9 makalede kullanılan veri tabanı ve alanlarını göstermektedir [4]. Verilen tablolarda Smith ve Miller kelimeleri aranmak istensin. Şekil 3.10 veri tabanının bir parçasının kayıt örneğini ve birbiri ile ilişkilerini göstermektedir. Veri tabanı kayıt örneğinden de görülebileceği gibi Master index Smith için o_1 Miller için o_2, o_3 kayıtlarını verecektir.



Şekil 3.9. Veri tabanı tabloları ve alanları [4]

Discovery yönteminde elde edilen bu kayıtlara göre aday ağ oluşturulur. Aday ağ oluşturulurken aynı zamanda sonuç elde edilemeyecek aday ağların elenmesi işlemi de gerçekleştirilir.

Öncelikle o_1 kaydı alınarak o_1 üzerinden hangi tabloların hangi kayıtlarına gidilebileceğine bakılır. O_1 üzerinden Lineitem ve Customer tablolarına gidilebileceği görülmektedir. Burada oluşan aday ağlar $O^{smith} \leftrightarrow L^{\{\}}$ ve $O^{smith} \leftrightarrow C^{\{\}}$ aday ağlarıdır.



Şekil 3.10. Veri tabanı kayıtları ve ilişkileri [4]

Şekil 3.10'da verilen veri tabanı için oluşan bu aday ağlar kullanılarak yeni aday ağlar elde edilir. Miller kaydını içeren bir network elde edildiğinde bu network sonuç ağlarından biridir. Örneğin ikinci adıma gelindiğinde ilk ele alınacak $O^{smith}\leftrightarrow L^{\{\}}$ aday ağı Order ve Partsupp tabloları ile ilişkilidir. Bu aday ağı Order tablosu ile ilişkilendirildiğinde tekrar "Smith" kaydına ulaşarak tekrarlayan bir döngüye dönüşür. Bu nedenle $O^{smith}\leftrightarrow L^{\{\}}\leftrightarrow O^{\{\}}$ aday ağı elenir. Çünkü Order tablosunun birincil anahtar alanı ile Lineitem dış anahtar alanı ilişkilidir ve Lineitem tablosundan tekrar Order tablosuna gidildiğinde Order tablosunun yine aynı birincil anahtar alanına gidilecek ve sonsuz bir döngü oluşacaktır. Aynı şekilde $O^{smith}\leftrightarrow L^{\{\}}$ aday ağından "Miller" kaydına da ulaşılamaz çünkü ulaşılabilir olması durumunda O^{smith} ve O^{miller} kayıtlarının aynı kayıt olması gerekmektedir. Bu nedenle $O^{smith}\leftrightarrow L^{\{\}}\leftrightarrow O^{miller}$ aday ağı da elenmiş olur. Ama bunun yanında $O^{smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}$ aday ağı elenmez çünkü bu aday ağından bir sonraki adımda diğer tablolara ilişki tanımlanabilir. Böylece $O^{smith}\leftrightarrow L^{\{\}}$ aday ağının ilişki kurabileceği tablolar bitmiştir. Bir sonraki

adımda da $O^{smith}\leftrightarrow C^{\{\}}$ aday ağından elde edilebilecek yeni aday ağları incelenir. Bu aday ağa bakıldığında Order ve Nation tabloları ile ilişkide olduğu görülebilir. Burada oluşacak $O^{smith}\leftrightarrow C^{\{\}}\leftrightarrow O^{miller}$ aday ağı sonuç ağı olur çünkü Order tablosunun dış anahtar alanı ile Customer tablosunun birincil anahtar alanı ilişkilidir ve Customer tablosunun birincil anahtar alanı ile Order tablosunun dış anahtar alanı ilişkilidir. Bu nedenle Customer tablosu üzerinden O^{smith} ve O^{miller} arasında bir ilişki kurulabilir. Böylece $O^{smith}\leftrightarrow C^{\{\}}\leftrightarrow O^{\{\}}$ ve $O^{smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}$ aday ağları bir sonraki adım içi aday ağ kümesini oluşturmaktadır.

1a	O^{Smith}
2a b	$O^{Smith}\leftrightarrow L^{\{\}}$/1a $O^{Smith}\leftrightarrow C^{\{\}}$/1a
3a b c d e f	$O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow O^{\{\}}$(pruned)/2a $O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow O^{Miller}$(pruned)/2a $O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}$/2a $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow O^{\{\}}$/2b $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow O^{Miller}$/2b $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}$/2b
4a b c d .	$O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}\leftrightarrow P^{\{\}}$ /3c/ $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow O^{Miller}$ $O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}\leftrightarrow L^{\{}$/4c $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow O^{\{\}}\leftrightarrow C^{\{\}}$(pruned)/3d $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}\leftrightarrow C^{\{\}}$/3f ...
5a b c d e .	$O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}\leftrightarrow P^{\{\}}\leftrightarrow PS^{\{\}}$(pruned)/4a $O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}\leftrightarrow L^{\{\}}\leftrightarrow O^{Miller}$/4b $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}\leftrightarrow C^{\{\}}\leftrightarrow O^{Miller}$/4d $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}\leftrightarrow C^{\{\}}\leftrightarrow O^{\{\}}$/4d $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}$(pruned)/4d ...
6a	$O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}\leftrightarrow C^{\{\}}\leftrightarrow O^{\{\}}\leftrightarrow C^{\{\}}$(pruned) /5d/ $O^{Smith}\leftrightarrow C^{\{\}}\leftrightarrow N^{\{\}}\leftrightarrow C^{\{\}}\leftrightarrow O^{Miller}$ .../ / $O^{Smith}\leftrightarrow L^{\{\}}\leftrightarrow PS^{\{\}}\leftrightarrow L^{\{\}}\leftrightarrow O^{Miller}$
7a	...

Şekil 3.11. Aday ağ örneği [4]

Discovery yönteminde aday ağının tüm anahtar kelimeleri içermesi ile aday ağı sonuç kümesine girer. Şekil 3.11'de Smith ve Miller anahtar kelimeleri için oluşturulan aday ağlar adım adım gösterilmiştir. Discovery'de amaç oluşan aday ağların anahtar kelimelerin tümünü içermesidir. Sonuçta elde edilen sonuç ağı arasında en az bağlantı sayısından başlamak üzere sıralama

yapılır. Her sonuç ağı için SQL sorgusu hazırlanır ve dönen sonuçlar kullanıcıya sunulur.

İlerideki çalışmalarda Discovery yöntemi tablo isimlerinin de çizge üzerinde tutulması ve tablo isimleri üzerinde de arama yapmayı hedeflemektedir. Yani kullanıcı “customer smith” anahtar kelimelerini girdiğinde kullanıcının aslında customer tablosundaki smith kaydı ile ilgilendiği bulunabilecektir.

3.5. Querying Communities Yöntemi

Son zamanlarda ilişkisel veri tabanlarında anahtar kelime üzerine birçok çalışma yapılmıştır. Bu çalışmaların çoğunda amaç anahtar kelimelerin tümünü içeren veri tabanı çizge yapısının en az bağlantıya sahip olmasını sağlamaktır. Banks, iki yönlü arama (bidrectional search) ve yakınlık (proximity) araması çalışmaları en az bağlantıya sahip ilk k kaydı bulmuştur [1,2,16].

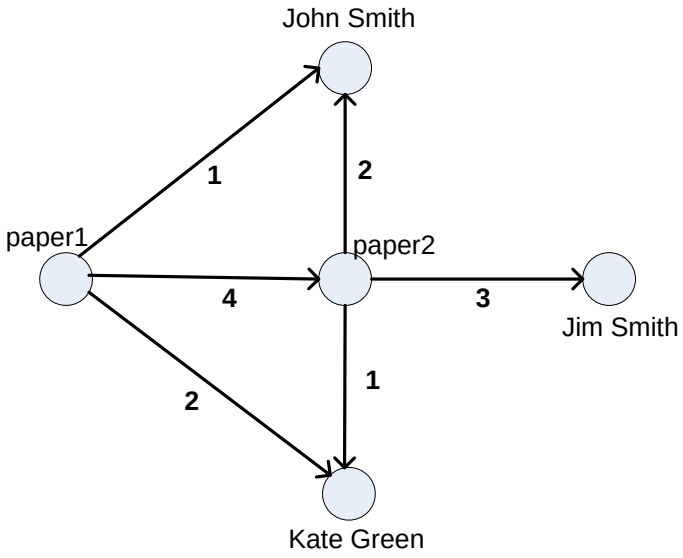
Makale diğer çalışmalardan farklı olarak gördüğü iki anahtar sorununa çözüm aramıştır [6,1,2,16]. Geliştirilen yöntemde çözüm aranan sorunlar şu şekildedir [6];

- Kullanıcı için en iyi olarak kabul edilen sonuç veri tabanı çizge yapısındaki en az bağlantıya sahip ağaç mıdır?

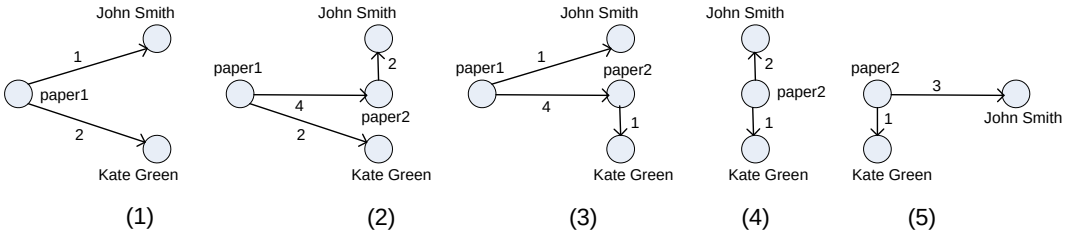
- Kullanıcının aradığı anahtar kelimeler ile en etkili alt çizgeler nasıl bulunmalıdır?

Bu amaçla makalede küçük bir çizge ele alınmıştır. Makalede üzerinde çalışılan çizge Şekil 3.12’de gösterildiği gibidir. Veri tabanını bir parçasını oluşturan çizge makale ve yazar tablolarını içermektedir. Makaleler paper1 ve paper2, yazarlar ise John Smith, Jim Smith ve Kate Green’dir. Makalede aynı zamanda kayıtlar arası bağlantılara da ağırlıklar verilmiştir. Ağırlıklar

Şekil 3.12'de çizge üzerinde gösterilmiştir. Paper1 ile John Smith arasındaki bağlantıya John Smith'in makalenin ilk yazarı olmasından dolayı 1, paper1 ve Kate Green arasındaki bağlantıya da Kate Green'in makalenin ikici yazarı olmasından dolayı 2 ağırlığını verilmiştir. Makalede makale kayıtları arasındaki bağlantının ağırlığı da 4 olarak kabul edilmiştir.



Şekil 3.12. Çizge [6]



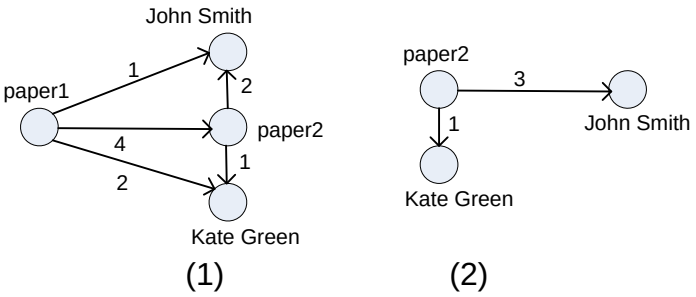
Şekil 3.13. Beş alt çizge [6]

Örneğin Kate ve Smith anahtar kelimelerini düşünelim. Oluşacak alt çizgeler Şekil 3.13'de gösterildiği gibidir. Şekilde de görüldüğü gibi 5 tane alt çizge oluşmuştur. Fakat oluşan bu alt çizgeler bize bazı bilgileri verememektedir. Örneğin kullanıcı John Smith ve Kate Green tarafından yazılmış kaç makale var bilmek istiyor. Alt çizgeler incelendiğinde 1. ve 4. çizgeler den iki yazarın

ortak yazmış olduğu makalelere ulaşabiliyoruz. Fakat bu bilgiye tüm cevapları inceleyerek ulaşabiliriz.

Bu yaklaşımda diğer bir problem ise verilen anahtar kelimeler için çok fazla sonuç oluşması ve kullanıcı bu sonuçlar arasından kendine ihtiyacı olan bilgiyi bulamamasıdır.

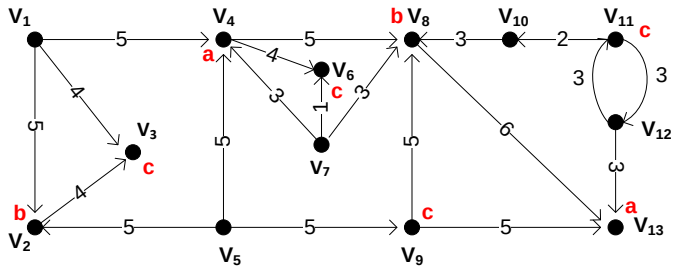
Makalede buna çözüm olarak merkez düğümler (center nodes), yol düğümleri (path nodes) ve anahtar kelime düğümleri (keyword nodes) tanımlamaları yapılmıştır. Buna göre merkez düğümler tüm anahtar kelimeler ile ilişkisi olan düğüm olarak tanımlanmıştır, yol düğümleri anahtar kelime ve merkez düğümler arasındaki düğümler olarak tanımlanmıştır ve anahtar kelime düğümleri ise anahtar kelimelerin bulunduğu düğümler olarak tanımlanmıştır.



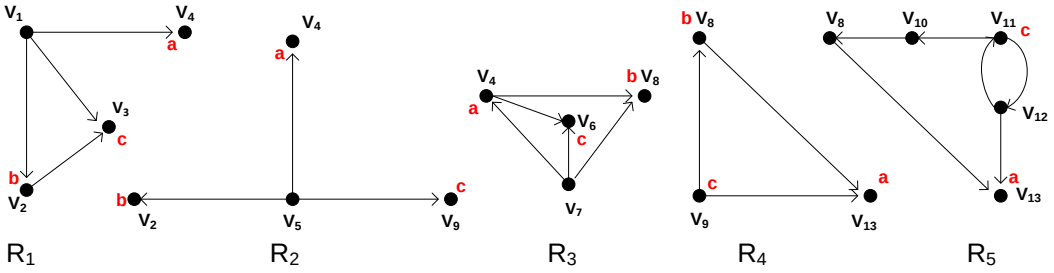
Şekil 3.14. Communities yöntemi ile oluşan çizgeler [6]

Makalede alt çizgelerin oluşturulmasında Communities yöntemi yani merkez düğümlerin baz alındığı yöntem kullanılmıştır. Makaledeki yaklaşımda merkez düğümler baz alınarak oluşturulan alt ağaçlar Şekil 3.14'de gösterilmiştir. Şekil 3.14'deki 1. çizge hem anahtar kelimeler Kate ve Smith'i hem de merkez düğüm paper1 ve paper2'yi içermektedir. Aynı zamanda Şekil 3.13'de oluşturulmuş olan ilk 4 alt çizgeyi içermektedir. Şekil 3.14'deki 2. çizge de aynı şekilde hem anahtar kelimeleri içermekte hem de merkez düğüm olan paper2'yi içermektedir.

Merkez düğümler baz alınarak oluşturulmuş alt çizgeler üzerinde anahtar kelimelerin tümünü içeren en kısa yollar hesaplanır. Yani her çizgede her anahtar kelimeye giden maliyetler hesaplanır ve toplanır. Her çizge için en iyi çözüm en düşük maliyetli çözümdür. Daha sonra kullanıcının önüne her çizge için hesaplanan en düşük maliyetli yol, maliyet sırasına göre verilir.



Şekil 3.15. Örnek veri tabanı çizgesi [6]



Şekil 3.16. Merkez düğümler baz alınarak oluşturulan alt çizgeler [6]

Örnek olarak Şekil 3.15 incelenirse merkez düğümler baz alınarak 5 tane alt ağaç oluşturulur. Her merkez düğümün her anahtar kelimeye olan uzaklığı hesaplanır ve toplanır. Örneğin Şekil 3.16'da oluşan alt çizgelerden 5. çizgeyi incelersek iki merkez düğüm olduğunu görürüz. Bunlar V11 ve V12 düğümleridir. V11 düğümünden tüm anahtar kelimelere maliyet hesaplandığında anahtar düğümler olan V8, V11, V13 için maliyet (2+3)+0+(3+3) = 11 olmaktadır. Yine aynı alt çizgede merkez düğüm V12 için maliyet hesaplanırsa (3+2+3)+3+3 = 14 olmaktadır. Bu çizge için düşük maliyetli olan V11 merkez düğümünün sonucu alınır. Diğer çizgeler de aynı

yöntem ile incelenir ve en düşük maliyetli yollar alınır ve kendi aralarında sıralanırlar. Bu sıralanmış çizelge Çizelge 3.1'de görüldüğü gibidir.

Çizelge 3.1. Çizgelerin maliyetlerine göre sonuç çizgelerinin sıralanması

Sıralama	Anahtar Kelime Düğümleri			Çizge	Maliyet	Merkez Düğüm
	a	b	c			
1	v_4	v_8	v_6	R_3	7	{ v_4, v_7}
2	v_{13}	v_8	v_9	R_4	10	{ v_9}
3	v_{13}	v_8	v_{11}	R_5	11	{ v_{12}, v_{11}}
4	v_4	v_2	v_3	R_1	14	{ v_1}
5	v_4	v_2	v_9	R_2	15	{ v_5}

3.6. Rsearch Yöntemi

Rsearch yöntemi ilişkisel veri tabanlarında anahtar kelime arama yaparken birçok veri tabanında karşılaşılabilecek problemlerden birine çözün aramıştır [7]. Makalede bu amaçla tespit ettiği problemi çözmek için bu problemin oluşabileceği bir veri tabanında çalışılmıştır. Bu yöntemde amaç çift kayıtları ele alarak anahtar kelime araması yapmaktır. Bu problem veri tabanına özel bir problem olup veri tabanına özel yöntem geliştirilerek çözüm üretilmeye çalışılmıştır.

Author

id	name	email
a_1	Alon Y. Halevy	alon@cs.washington.edu
a_2	Halevy A	alon@cs.washington.edu
a_3	Halevy A	avinoams@clalit.org.il
a_4	Halevy A	
a_5	Halevy Alon	halevy@google.com
a_6	Halevy AY	halevy@google.com
a_7	Dong XL	lunadog@research.att.com

Write

id	aid	pid
w_1	a_1	p_6
w_2	a_2	p_7
w_3	a_3	p_5
w_4	a_4	p_1
w_5	a_4	p_2
w_6	a_4	p_4
w_7	a_5	p_2
w_8	a_6	p_3
w_9	a_7	p_2

Source

id	name	ISSN
s_1	Comminication of the ACM	0001-0782
s_2	IEEE Intelligent System	1541-1672
s_3	Journal of Child Neurology	0883-0738
s_4	VLDB J.	1066-8888
s_5	VLDB Journal	1066-8888

Paper

id	title	sid	year	Vol(number)	page	Citation-times
p_1	Data integration with uncetainty	s_5	2009	18(2)	469-500	0
p_2	Representing uncertain data ...	s_5	2009	18(5)	989*1019	0
p_3	The Claremont Report on Database Research	s_1	2009	52(6)	56-65	0
p_4	The Unreasonable Effectiveness of data	s_2	2009	24(2)	8-12	0
p_5	... Complex Visual Hallucinations	s_3	2009	24(8)	1005-1007	0
p_6	Schame mediation ... Sematic data sharing	s_4	2005	14(1)	68-83	11
p_7	MiniCon ... Answering queries using views	s_4	2001	10(2-3)	182-198	33

Şekil 3.17. Uygulamanın gerçekleştirildiği örnek veri tabanı [7]

Şekil 3.17'de görüldüğü gibi Author tablosunda 3 satırda "name" alanının değerleri aynıdır. Fakat "email" alanını değerleri farklıdır. "email" alanına bakıldığında ise a_1 id'sine sahip yazar Alon Y.Halevy ile a_2 id'sine sahip yazar Halevy A satırlarının mail bilgileri aynıdır. Bu bilgilere dayanarak a_4'ün mail bilgisi boş olduğundan a_4,a_1 ve a_2 aynı yazar olabilir fakat a_3 aynı yazar değildir çıkarımı yapılabilir. Bunun yanında a_4'ün de aynı yazar olup olmadığı sonucuna varılması daha kompleks bir aramayı gerektirmektedir. Bu arama Source tablosu üzerinden gerçekleştirilebilir. Yine aynı şekilde a_4,a_1 ve a_2 makalelerinin kaynaklarına bakıldığında aynı ISSN değerlerini görebiliriz. Bu durumda bu üç yazarın aynı yazar olduğu bilgisine ulaşılabilir.

Bu yöntemde amaç oluşabilecek iki probleme çözüm bulmaktır. Bunlar;

- Kullanıcının yanlış sonuçlar araması

- Kullanıcının bulması gereken asıl sonuçlara ulaşamaması

☑ **1. Title:** Representing uncertain data: models, properties and algorithms
Author: Das Sarma A, Benjelloun O, Halevy A, et al.
Publisher: VLDB JOURNAL Vol:18 No:5 Pages:989-1019 Year:OCT 2009
Citation: 0

☐ **2. Title:** Methylphenidate Induction of Complex Visual Hallucinations
Author: Halevy A, Shuper A
Publisher: JOURNAL OF CHILD NEUROLOGY Vol:24 No:8 Pages:1005-1007 Year:AUG 2009
Citation: 0

☑ **3. Title:** Gastrointestinal Stromal Tumars: A 19 Year Experience
Author: Rabin I, Chikman B, Lavy R, et al
Publisher: ISRAEL MEDICAL ASSOCIATION JOURNAL Vol:11 No:2 Pages:98-102 Year:FEB 2009
Citation: 1

☑ **4. Title:** Data integration with uncertainty
Author: Dong XL, Halevy A, Yu C
Conference: 33rd International Conference on Very Large Data Bases, SEP 23-28, 2007 Univ Vienna, Vienna, AUSTRIA
Publisher: VLDB JOURNAL Vol:18 No:2 Pages:469-500 Year:APR 2009
Citation: 0

☑ **5. Title:** The Unreassonable Effectiveness of Data
Author: Halevy A, Norvig P, Pereira F
Publisher: IEEE INTELLIGENT SYSTEMS Vol:24 No:2 Pages:8-12 Year:MAR-APR 2009
Citation: 0

Şekil 3.18. Halevy A ve 2009 anahtar kelimeleri ile ilişkisel veri tabanında yapılan arama sonucu [7]

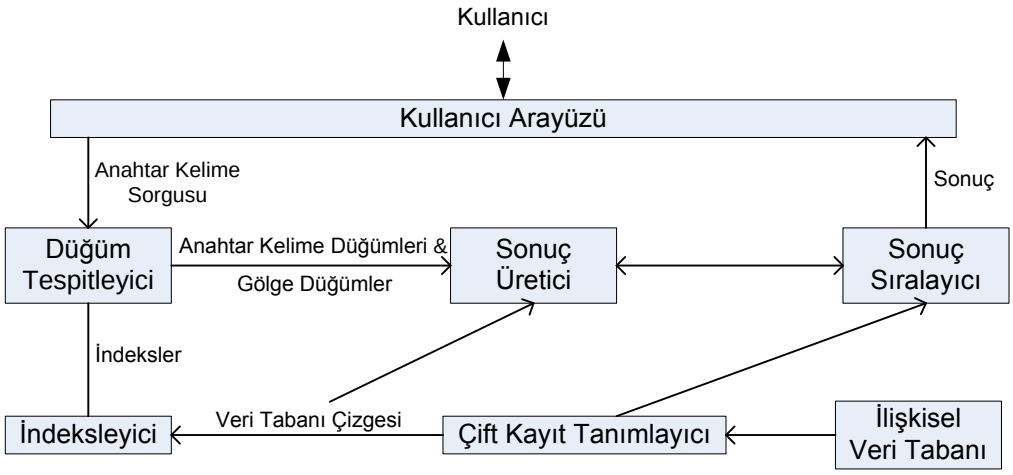
Örneğin, Science Citation Index Expanded veri tabanında SCI-indexed dergisinde 2009 yılında Alon Y. Halevy tarafından yazılmış makaleler aranıyor olsun. Bunun için Alon Y. Halevy ve 2009 anahtar kelimelerini vererek arama yapıldığında bu anahtar kelimeler ile herhangi bir sonuca ulaşılamadığı görülür. Fakat bu anahtar kelimelerin yerine Halevy A ve 2009 anahtar kelimelerini aratıldığında Şekil 3.18'de görüldüğü gibi 5 sonuca ulaşılabilinir. Bu sonuçlardan sadece 3 tanesi arama ile ilgili sonuçlar olurken diğerleri ise arama ile ilgili değildir. Aslında bulunan bu sonuçlar ilk aramanın sonuçlarını da vermektedir. Bunu da çift kayıt ilişkisinden bulabiliyoruz. Veri tabanı incelendiğinde Alon Y. Halevy kelimesinin bulunduğu satırın ve Halevy A. Kelimesinin bulunduğu satırın "email" alanındaki değerleri aynıdır. Bu bilgi de bu iki kişinin aynı olabileceği bilgisini verir. Çünkü mail bilgisi kişisel bir bilgi olup her kişi için farklı olması beklenir. Aynı durum ayırt edici kayıt özelliği olan diğer alanlar için de geçerlidir. Yine aynı şekilde aramada anahtar kelime olarak Halevy AY ve 2009 kullanıldığında ise Şekil 3.19'da görüldüğü gibi tek sonuç dönmektedir. Fakat veri tabanına bakıldığında yine

ayırt edici alan olan “email” alanı bilgisinden Halevy Alon kaydının da aynı kişiye ait olabileceği bilgisine ulaşılabilir.

☑ **1. Title:** The Claremont Report on Databases Research
Author: Agrawal R, Ailamaki A, Bernstein PA et al.
Publisher: COMMUNICATIONS OF THE ACM Vol:52 No:6 Pages:56-65 Year:JUN 2009
Citation: 0

Şekil 3.19. Halevy AY ve 2009 anahtar kelimelerinin arama sonucu [7]

Bu makalede RSEARCH adı verilen bir sistem uygulanmış ve veri tabanındaki kayıtlar arası ilişkiler analiz edilmiştir [7]. Bu sayede çift kayıtlar tanımlanarak arama kabiliyeti artırılmaya çalışılmıştır.



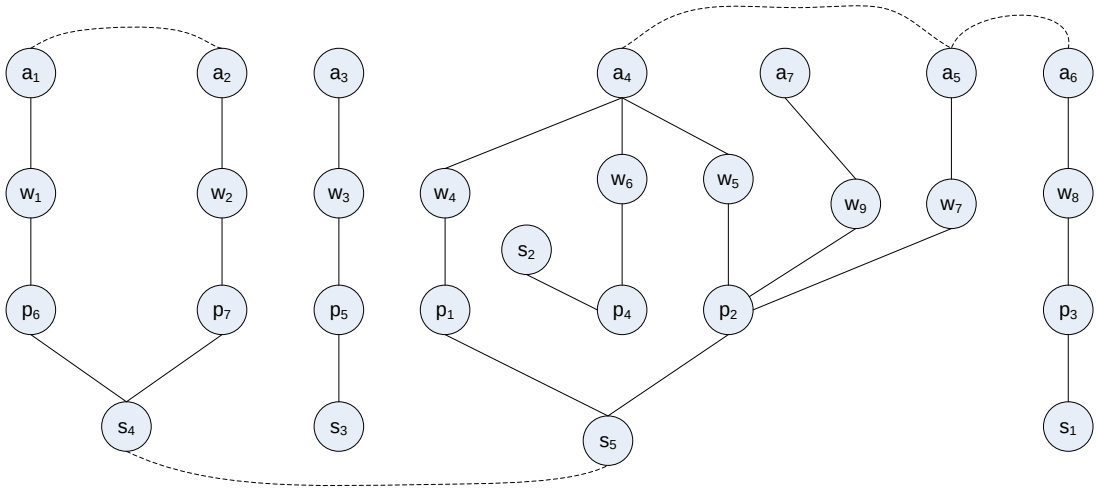
Şekil 3.20. Makalede geliştirilmiş olan Research arama sisteminin yapısı

RSEARCH sisteminin yapısını Şekil 3.20’de görüldüğü şekilde tanımlayabiliriz.

- *Çift kayıt tanımlayıcı (Nearly duplicate records identifier) :* Bu yapı verileri analiz eder, çift kayıtları tanımlar ve veri tabanı çizge yapısını oluşturur. Oluşturulan veri tabanı çizgesi G= (V,E_f,E_d) şeklinde ifade edilebilir. V veri tabanındaki her kaydı ifade ederken köşeleri oluşturur ve bu köşe bir kaydın tüm özellik değerlerini içerir. E_f ise bir kenardan diğerine olan dış anahtar bağlantısı kuran kenarları e€E_f ifade eder.

Diğer yaklaşımlardan farklı olarak bu yaklaşımda kullanılan E_d' yi başak bir kenar tipi olarak tanımlayabiliriz. Bu iki çift kayıt arasındaki ilişkiyi belirten çift kenar olarak adlandırılmıştır.Şekil 3.21,Şekil 3.17'deki veri tabanını çizge yapısını göstermektedir. Çizge yapısında çift kenarlar kesikli çizgi ile gösterilmiştir.

Bir kaydın çift olarak tanımlanabilmesi için kaydın bilgilerini tutan sadece bazı alanlar kullanılabilir. Örneğin Author tablosunun email alanı, Source tablosunun ISSN alanı ve Paper tablosunun sid, vol(number) ve page alanları birlikte çift kayıtları tanımlamak için kullanılır. Bu alanlar çift kayıt tanımlayıcı olarak adlandırılırlar. Çift kayıt tanımlayıcı olan alanlar eğer var ise her tablo için önceden belirlenmelidir.



Şekil 3.21. Veri tabanı kayıtlarının çizge üzerinde gösterimi [7]

- *İndeksleyici (Indexer) :* veri tabanı çizgesine göre indeksler oluşturur. Bununla birlikte birbirinin aynısı olabilecek kayıtları da indekslerin içerisinde tutar. İndeks yapısı bir ağaç olarak ifade edilmiştir.

- *Düğüm tespitleyici (Node locator) :* Öncelikle kullanıcı bir sorgu oluşturur. Düğüm tespitleyici indeksleyiciye ulaşarak çizge üzerinde eşleşen kayıtları alır. Bu eşleşen kayıtların bulunduğu düğümlere anahtar kelime düğümü adı verilir. İndeksleyici sadece anahtar kelime düğümlerine konumlanmanın yanında çizge G üzerindeki çift kenarları E_d'yi kullanır. Aynı zamanda bu düğümler üzerine de konumlanır ve bunlar anahtar kelimeleri içermeleri halinde anahtar kelime düğümleri ile aynı kümede değerlendirir. Bu düğümler gölge düğüm (shadow node) olarak tanımlanır. Bu durumda gölge düğüm ile anahtar kelime düğümü arasında çift kenar vardır.

- *Sonuç üretici (Result generator) :* Tanımlanan iki çeşit düğüm, anahtar kelime düğümü ve gölge düğüm çeşidine dayanır. Sonuçları oluşturmak için bu iki çeşit düğüm arasındaki bağlantının nasıl olduğuna karar verir. Oluşturulan arama sonuçları anahtar kelime düğümünü içermeyebilir fakat sonuçlar kullanıcının ilgilendiği sonuçlardır. Örneğin iki anahtar kelime Alon Halevy ve VLDB Journal için gerçekleştirilen aramada Şekil 3.21'deki a_2-w_2-p_7-s_4 yolu anahtar kelime düğümü içermemektedir. Burada a_2 ve s_4 gölge düğümleridir. Eğer bu aramada veri tabanı çizgesi üzerinde çift kenar düşünülmemiş olsaydı sistem kullanıcıyı ilgilendiren bu sonucu üretmeyecekti.

- *Sonuç sıralayıcı (Result ranker) :* Sonuç sıralayıcı, sonuç üretici tarafından üretilen sonuçları farklı sıralama fonksiyonları ile sıralar. Bu fonksiyonlar eşleşen anahtar kelime sayısı, veri tabanı çizge üzerindeki ağırlık ve sorgu sonuç büyüklüğü gibi olabilir. Bu çalışmada basitlik açısında çift kenar ağırlıkları 0 olarak kabul edilmiştir.

Makalede geliştirilen yöntem bazı veri tabanlarında arama yapıldığında yanlış sonuçlara ulaşılması veya istenilen sonuçlara ulaşılamamasını engellemek için aynı kayıt olabileceğini düşündüğü kayıtlar üzerinde de arama yaparak aramayı genişletmiştir.

3.7. DBXplorer Yöntemi

Makalede çalışılan yöntem DBXplorer olarak adlandırılmıştır [8]. DBXplorer'da amaç aranan anahtar kelimenin tümünü içeren sonuçların bulunmasıdır [8]. İki aşamada gerçekleşir. İlk aşama açığa çıkarma ikici aşama ise arama aşamasıdır.

Açığa çıkarmada arama yapılacak veri tabanının tüm tabloları alanları ile tanımlanır. Açığa çıkarma aşamasında, arama aşamasında kullanılacak arama tabloları oluşturulur. Bu tablo veri tabanı tablo isimleri, kolon adları ve tablo satırlarından oluşmuştur. Bu oluşturulan tablo sembol tablosu olarak adlandırılır ve anahtar kelimelerin aranmasında en etkili yöntemdir.

Arama aşamasında arama için oluşturulan sembol tablosu kullanılarak anahtar kelime veri tabanının tablolarının alan ve satırlarında aranır. Anahtar kelimelerin tümünün aynı tabloda olup olmadığı bağlantı ağacından bakılır ve tümünü içeren satırlar sonuç kümesine eklenir. Aranan kelimelerin farklı tablolarda olması durumunda bağlantı ağacı için SQL sorgusu çalıştırılır ve tüm anahtar kelimeleri içeren sonuçlar sonuç kümesine eklenir.

DBXplorer'da en önemli yapı sembol tablosunun tanımlanmasıdır. Sembol tablosu veri tabanındaki anahtar veri yapısıdır. Sembol tablosunun yapısı aranan anahtar kelimesinin yerinin belirlenmesinde önemlidir. Sembol tablosu tasarımında en önemli şey, anahtar kelimenin aranması sırasında veri tabanındaki anahtar kelimenin yerinin hızlı bir şekilde bulunabilesi için sembol tablosunun nasıl dizayn edilmesi gerektiğidir. Bunun için dikkat edilmesi gereken iki ayrıntı seviyesi vardır. Bunlardan biri sütun seviyesi diğeri ise satır seviyesidir.

Sütun seviyesinde sembol tablosu veri tabanını alanlarının tüm kelimeleri için o kelimeyi içeren sütun bilgisini içerir. Bu şekilde oluşturulan sembol tabloları Pub-Col sembol tablosu olarak adlandırılır.

Hücre ayrıntı seviyesinde ise, sembol tablosu veri tabanındaki alanların içerdiği her kelime için o kelimenin hücre bilgisini içerir. Bu şekilde oluşturulan sembol tabloları ise Pub-Cell olarak adlandırılırlar.

Uygulamalarda bu sembol tablolarının herhangi birinin tercih edilmesinde fazla fark olduğu söylenemez. Hücre ayrıntı seviyesindeki sembol tablosunun işlevsellik açısından tek avantajı hem sütun hem de satır bilgisini içermesidir. Fakat uygulamalarda oluşacak birçok kriter sembol tablosunun seçiminde etkilidir. Bunlardan ön önemlilerini şu şekilde sıralayabiliriz.

- Sembol tablosunun oluşmasında ihtiyaç olan yer ve zaman faktörü,
- Anahtar kelimenin aranmasında performans etkinliği,
- Sembol tablosunun güncelliğinin sağlanmasının kolaylığı

Sembol tablolarında yer ve zaman ihtiyacını değerlendirecek olursak Pub-Col sembol tablosu Pub-Cell sembol tablosuna göre daha avantajlıdır. Pub-Col sembol tablosu genelde Pub-Cell sembol tablosuna göre daha küçüktür ve Pub-Col sembol tablosunun oluşturulma zamanı Pub-Cell sembol tablosunun oluşturulma zamanını göre daha kısadır. Çünkü bir sütun birbirinden farklı kayıtlar içerdiğinden birden fazla hücreye sahiptir.

Anahtar kelime aramasında performans etkinliğine bakıldığında ise bunun SQL sorgusunun türüne ve çalıştırılmasına bağlı olduğu görülür. SQL sorguları anahtar kelimeleri aramada tablo ismi ve sütun ismine ihtiyaç duyar. Örneğin bir Order tablosuna sahip bir veri tabanı ve bu Order tablosunun o-orderpriority sütunu olsun. Order tablosu 150.000 satırlı verilerden oluşmuş olsun. Order tablosundaki o-orderpriority sütununda bu 150.000 satırda 5 farklı değer olsun. Bu 5 farklı değerin her biri için Pub-Cell sembol tablosu bu değerlerden biri ile eşleşen 30.000 satırın hücre bilgisini içerir. Arama

yapılırken aranan anahtar kelime için 30.000 hücre gösterilir. Eşleşen satırları almak için bu 30.000 satırın rowid'leri oluşturulur.

Pub-Col sembol tablosunda ise bu 5 farklı değerden biri için o-orderpriority sütunun ismi ile eşleşen bir satır yer alır. Arama için de basit bir SQL sorgusu oluşturulur. Oluşan SQL sorgusu "select * from Orders where Orders.o-orderpriority = @AnahtarKelime" şeklindedir. Böylece oluşturulan SQL sorgusu etkili bir şekilde çalışır. Fakat bu yöntem aranan sütuna index koyulmuş ise performansı yüksektir.

Sembol tablosunun güncelliğinin sağlanması kriterine bakıldığında, bu kriter veri tabanındaki verinin değişiminde önem kazanmaktadır. Pub-Col sembol tabloları bir satırda yeni bir değer araya giriyor ya da ekleniyor ise güncellemeye ihtiyaç duyar. Pub-Cell sembol tabloları ise tabloya her satır eklendiğinde o sütundaki değerlerden farklı olsun olması güncellenmeye ihtiyaç duyar. Sembol tablolarındaki değişiklikler bir trigger veya time stamps yardımı ile gerçekleştirilir.

Sonuç olarak şunu söyleyebiliriz ki eğer sütunlarda indeksleme yapılmış ise her zaman Pub-Col sembol tabloları Pob-Cell sembol tablolarına göre daha avantajlıdır.

Pub-Col sembol tabloları iki sütundan oluşur. Pub-Col sembol tablosundaki sütunlardan biri anahtar kelime bilgisini içerirken diğeri de anahtar kelimelerin bulunduğu tablodaki sütun bilgisini içeren ColId değerinin içerir. Sembol tabloları anahtar kelimeleri direk olarak içermezler. Anahtar kelimeler sembol tablosunda hash algoritmasında geçirilerek tutulurlar. Bunun sebebi de anahtar kelimenin uzunluğunun çok büyük olma ihtimalidir. Şekil 3.22'de bir Pub-Col sembol tablosu örneğini görebiliriz.

Şifrelenmiş Dğer	Kolon İd
v_1	c_1
v_2	c_1
v_3	c_1
v_4	c_1
v_2	c_2
v_3	c_2
v_4	c_2
v_5	c_2

Şekil 3.22. Pub-Col sembol tablosu örneği [8]

Şekil 3.22'de görüldüğü gibi v_1, v_2, v_3, v_4 ve v_5 değerleri aranabilecek kelimelerin hash algoritmasından geçirilmiş şifrelenmiş değerleridir. ColId alanında bulunan değerler ise hash algoritmasından geçirilmiş kelimelerin veri tabanındaki hangi sütunda yer aldıkları bilgisidir. Şekil 3.22'de de görüldüğü gibi tüm kelimeler sembol tablosunda yer aldığından sembol tablosunun büyüklüğü çok fazla olacaktır. Sembol tablosunun boyutunun küçültülmesi için sıkıştırma yöntemleri geliştirilmiştir.

Sıkıştırma yöntemlerinden biri FK-Comp'tur (Foreign Key Compression). Bu sıkıştırma yönteminde c_1 sütunu dış anahtar ilişkisinden dolayı c_2 sütununun alt kümesi ise tek bir hash tablosu tutulur.

Diğer bir sıkıştırma yöntemi de CP-Comp (General Compression Technique) olarak adlandırılır. Bu yöntemde aynı kelimenin birden fazla sütunda bulunması durumunda, sütun eşleşme tablosu kullanılarak sembol tablosundaki iki veya daha fazla satır tek satıra indirilir. Şekil 3.23'de sütun eşleşme tablosu kullanılarak sıkıştırılmış sembol tablosu oluşturulmuştur.

Yeni Kolon İd	Kolon İd
x	c_1
x	c_2

Şifrelenmiş Değer	Kolon İd
v_1	c_1
v_2	x
v_3	x
v_4	x
v_5	c_2

Şekil 3.23. Eşleşme tablosu ve sıkıştırılmış Pub-Col sembol tablosunu [8]

Şekil 3.23'de görüldüğü gibi v_2, v_3, v_4 hem c_1 hem de c_2 sütunlarında yer almaktadır. Oluşturulan eşleşme tablosunda x değeri c_1 ve c_2 sütunlarını işaret etmektedir. Oluşturulan eşleşme tablosu sayesinde 8 satırlık sembol tablosu 5 satıra düşmüştür.

Sembol tablosu S'in sıkıştırılmasında, S tablosundaki her bir şifrelenmiş kelimelerin değerler için kelimelerin geçtiği ColId listesi oluşturulur. Buna göre {{c_1},{c_1, c_2},{c_2}} listesi oluşur. Oluşturulan ColId listesi kullanılarak HashVal listesi oluşturulur. ColId listesinin her bir elemanı için o sütunda geçen HashVal değerlerini içerir. Oluşan liste {{v_1, v_2, v_3, v_4},{v_2, v_3, v_4},{v_2, v_3, v_4, v_5}} şeklindedir. Oluşturulan listelerde listenin her bir elemanı için $|ColId_i|*|HV_i| > |ColId_i|+|HV_i|$ şartının doğruluğu aranır. Şartın sağlanması ile şartın sağlandığı HV_i elemanının elemanları S sembol tablosunda kaldırılır ve yerlerine HV_i elemanları ile x ColId değeri eklenir. Aynı şekilde x değerinin karşılığının tutulduğu sütun eşleşme tablosuna $ColId_i$ elemanının elemanları ve karşılığında x yeni ColId bilgisi eklenir.

Pub-Cell sembol tablosu da Pub-Col sembol tablosunda olduğu gibi iki sütundan oluşmaktadır. Bu sütunlardan biri anahtar kelimelerin hash algoritmalarından geçirilmiş değerlerinin tutulduğu sıkıştırılmış değer "şifrelenmiş değer" sütunu, diğeri ise anahtar kelimelerin hücre bilgisinin tutulduğu "hücre id" sütunudur. Pub-Cell sembol tabloları için bir anahtar kelimenin başka sütun veya satırlarda tekrar etmesi bir dezavantajdır. Tüm anahtar kelimelerin hücre bilgileri sembol tablosunda tutulur. Bu nedenle

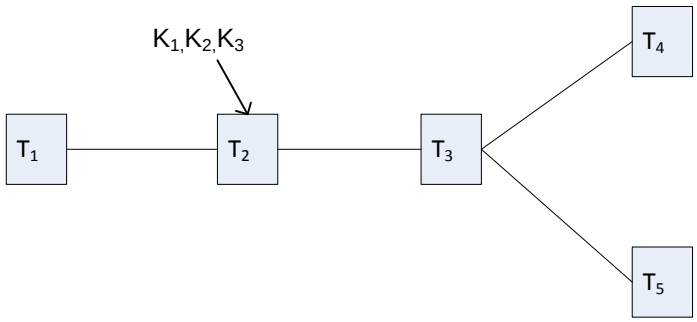
arama sırasında SQL ifadesi için istenen bütün lokasyonların alınması uzun zaman alabilir. Bu nedenle HashVal ve CellId listeleri iyi dizayn edilmelidir.

Pub-Cell sembol tablosunun yapısı gereği, anahtar kelime için alınan tüm lokasyonlar sembol tablosunun tek bir sütununda arama yapılarak elde edilir. Pub-Cell sembol tablosu tüm anahtar kelimelerin hücre lokasyonlarını içerdiğinden bu tabloda sıkıştırma yapılamaz.

İlişkisel veritabanlarında arama yapmanın ilk adımı sütun veya hücre bilgilerini içeren sembol tablolarında arama yapma işlemidir. Bu işlem oluşturulan SQL sorgusu ile gerçekleştirilir.

İkinci adım ise bağlantı ağacının numaralandırması ve eşleşen satırların tanımlanmasıdır.

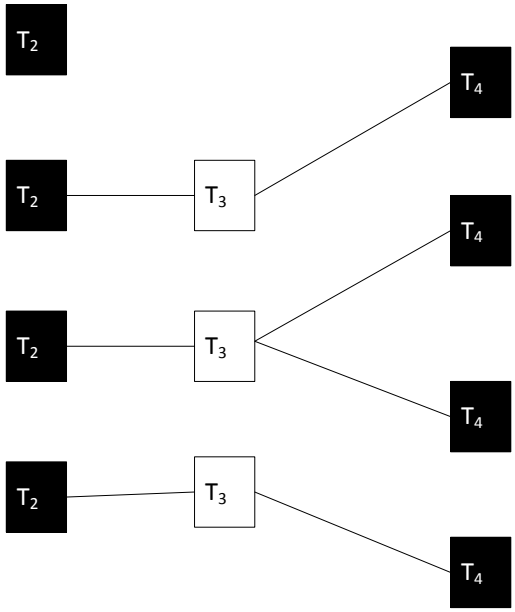
Bağlantı ağacının numaralandırmasında veri tabanında bulunan tabloların yönsüz bağlantılar ile birbirine bağlanması ile oluşan şema grafiğini G olarak düşünelim. G'nin alt ağaçları aranan kelimelerin hepsini içeren alt ağaçlardır.



Şekil 3.24. Örnek bağlantı ağacı gösterimi [8]

Şekil 3.24'deki gibi bir G bağlantı ağacıda K_1,K_2,K_3 anahtar kelimeleri aranırsa K_1,K_2,K_3 anahtar kelimelerinin üçünün de beraber olduğu satırlar seçilir ve bu satırların bulunduğu bağlantı ağaçları alt bağlantı ağaçlarını oluşturur. Alt bağlantı ağaçları da Şekil 3.25'de ifade edildiği gibidir. Anahtar

kelimelerin herhangi birini içeren tablo siyaha boyalıdır. Bu alt bağlantı ağaçları da G' olarak ifade edilir.



Şekil 3.25. Bağlantı ağacından elde edilen sonuçlar [8]

Veri tabanında arama işlemi, oluşturulan G çizgesi ile ilişkilendirilir. Elde edilen G' çizgeleri, sonuçları ifade eder. Makaledeki çalışmada sıralama işlemi ise bağlantı sayısına göre yapılmıştır. En az bağlantıya sahip sonuç ilk sırada gelirken en çok bağlantıya sahip sonuç en son sırada gelmektedir.

Makaledeki çalışmada Pub-Col sembol tablosunda karşılaşılan bir soruna çözüm olarak Pub-Prefix sembol tablosu oluşturulmuştur. Şekil 3.26'da görüldüğü gibi bir veri tabanı tablosunda "string", "ball", ve "round" kelimelerini aranmaktadır. Bu kelimelerin şifrelenmiş karşılıklarının da 1, 2, 3 olduğu kabul edilmektedir.

Satır İd	C
1	This is a string
2	This string
3	This is a ball
4	x
5	Any ball is round

Şekil 3.26. Pub-Cell sembol tablosu örneği [8]

Pub-Col sembol tablolarında arama performansı bir tablodaki sütunun uzunluğuna bağlıdır. Eğer ilgili sütunda index yok ise sütun genişliği ne kadar büyük ise arama işlemi o kadar zorlaşır. Buna çözüm olarak Pub-Prefix sembol tablosu oluşturulmuştur. Pub-Prefix sembol tablosunda alternatif bir indeks oluşturulur. Arama bu indeks üzerinden gerçekleştiğinden arama çok daha hızlı bir şekilde gerçekleşir.

Pub-Prefix tablo yapılmamış bir tabloda arama yapılırken aranan “string” kelimesi için WHERE T.C LIKE ‘%string%’ şartı oluşmaktadır. Bu şart ile arama yapıldığında arama işlemi uzun sürmektedir.

Pub-Prefix sembol tablosu oluşturularak arama işlemi çok daha hızlandırılabilir. Pub-Prefix sembol ilgili sütunun ilk iki harfi alınarak index oluşturulur. Oluşturulan Pub-Prefix sembol tablosunu Şekil 3.27’de gösterildiği gibidir.

Şifrelenmiş Değer	Sütün İd	Prefix
1	c	th
1	c	no
2	c	th
2	c	an
3	c	an

Şekil 3.27. Pub-Prefix sembol tablosunun örnek gösterimi [8]

Anahtar kelime “string” için şifrelenmiş değeri 1’dir. “string” anahtar kelimesi C sütununda 3 satırda geçmektedir. “string” için oluşturulacak indeksler “th” ve “no” olur. Çünkü “string” anahtar kelimesinin geçtiği satırların ilk iki harfi

alınmaktadır. Şekil 3.27'de görüldüğü gibi bir Pub-Prefix sembol tablosu oluşturulur. Oluşturulan bu tablodan arama yapılmak istenirse oluşacak SQL sorgu kriteri WHERE (T.C LIKE 'th%string%') OR (T.C LIKE 'no%string%') şeklinde oluşmaktadır ve arama çok daha hızlı olmaktadır.

Pub-Prefix sembol tabloları sütun uzunluklarının kısa, 100 karakterden daha az olduğu durumlarda daha performanslıdır.

Pub-Col sembol tabloları bir sütun üzerinde metnin tamamında indeks varsa, Pub-Prefix sembol tabloları sütun uzunlukları kısa ve indeksleme yok ise, Pub-Cell sembol tabloları ise diğer durumlarda yani sütun uzunluklarının çok uzun olduğu durumlarda tercih edilmelidir.

Fakat Pub-Prefix sembol tablosunda da karşılaşılan bazı sorunlar vardır. Örneğin "cat" ve "cats" ile başlayan sütunlar için oluşan index "ca" şeklindedir. Oluşacak SQL sorgusu WHERE Tablo_İsim LIKE 'ca%' şeklinde oluşur. Bu durumda "cat" veya "cats" için yanlış sonuçlar da döner.

Bunun için gelecek çalışmalarda bu tip karşılaşılacak problemlere çözümler aranmaktadır.

3.8. Proximity Yöntemi

Literatürde veri tabanında anahtar kelime arama için yapılan çalışmalardan biri de yakınlık aramasıdır. Yapılan çalışma yakınlık araması (proximity search) olarak adlandırılmıştı [16]. Yakınlık aramasında amaç aranan anahtar kelimeleri bulunduran objelerin birbiri ile ilişkisinin bulunmasıdır. Bunun için rastgele seçilmiş bir veri tabanında, veri tabanın objeleri arasındaki ilişki durumu çıkarılır.

Örneğin veri tabanında bir kişinin arandığını düşünelim. Aranacak kişinin ismini anahtar kelime olarak verdiğimizde ilgili kişi ile ilgili tüm satırlar bize sonuç olarak döner. Fakat bu kişi öğretmen mi, öğrenci mi yoksa yönetici mi

olup olmadığını bilmeyiz, bundan dolayı sorgu tüm veri tabanı sonuçlarını bize verir. Eğer bu kişinin öğrenci kayıtları ile ilgili olduğunu biliyorsak her kaydın öğrencilik ile yakınlığını dizerek daha iyi sonuçlar elde edebiliriz.

Yakınlık araması, veri tabanını birbiri ile ilişkili objeler bütünü olarak görür. Bu ilişkiyi belirleyen mesafe fonksiyonudur. Mesafe fonksiyonu sistem yöneticisi tarafından belirlenir. Bu fonksiyon ile objeler arası yakınlığın ne kadar kuvvetli olduğu belirlenir. Bir personel veri tabanı düşünüldüğünde objeler arası bağlantı sayısı, objelerin ne kadar yakın ilişkili olduğunun bir göstergesidir. Örneğin aynı departmanda çalışan iki kişiyi düşünelim her iki çalışanın da aynı departmana linki olduğunda iki çalışan yakın ilişkilidir diyebiliriz. Diğer taraftan bir de iki departman olduğunu ve bu iki departmanın aynı ürün üzerinde çalıştığını düşünelim. Çalışanlardan biri bir departmanda diğeri diğer departmanda çalışıyor ise bu iki personel arasında yine bir ilişki mevcuttur fakat bu daha zayıf bir ilişkidir.

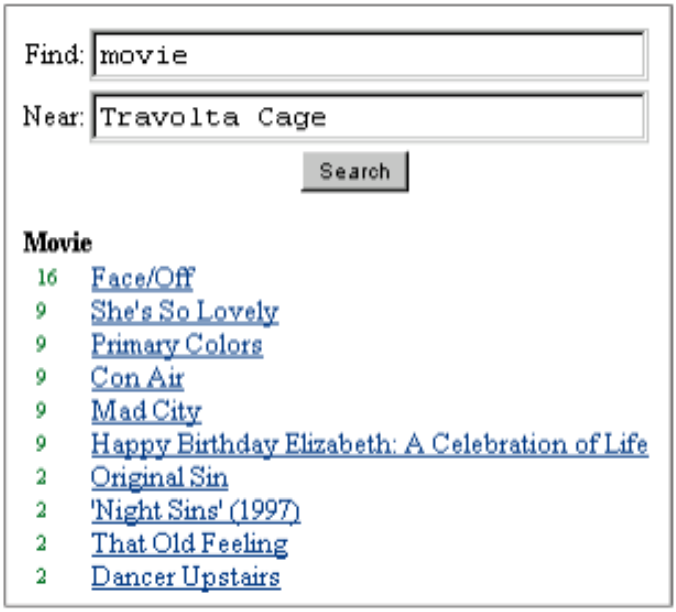
Makalede yapılan çalışmada veri tabanı olarak bir film veri tabanı incelenmiştir. Bu veri tabanı linklenmiş objeler seti olarak görüntülenir. Burada objeleri movie, actors ve directors temsil eder. Yakınlık aramasında veri tabanına özel “find set” ve “near set” listeleri oluşturulur.

- Yakınlık aramasında tanımlanan Find Set veri tabanındaki tüm objeleri içerir. Find sorgusuna verilen bir kelime Find Set içerisinde aranır. Örneğin Find Movie sorgusu bizim için tipi movie olan ve içinde movie geçen bütün objeleri ifade eder.

- Yakınlık aramasında tanımlanan Near Set ise Find sorgusu ile bulunan objelerden oluşur. Near sorgusuna verilen kelime bu set içerisinde aranır. Sonra Find Set Near Set’in sonuçlarına göre sıralanır.

Örneğin bir kullanıcı movie içerisinde John Travolta ve Nicolas Cage ile ilgileniyor olsun. Bu durumda oluşacak arama ifadesi “Find movie Near

Travolta Cage" olur. Bu sorgu Travolta ve Cage kelimelerini sadece movie içerisinde aramaz aynı zamanda veri tabanı içerisindeki ayrı objeler içinde de arar. Çünkü movie objesinin diğer objeler ile de ilişkisi olduğundan title, actor ve date bilgilerini de tanımlayan diğer objelere de linki vardır. Şekil 3.28'de görüldüğü gibi "Find movie Near Travolta Cage" sorgusu için sonuçlar sıralanmıştır. Şekil 3.28'den de anlaşıldığı gibi her iki aktör de Face/Off filmi için en kısa yola sahiptir.



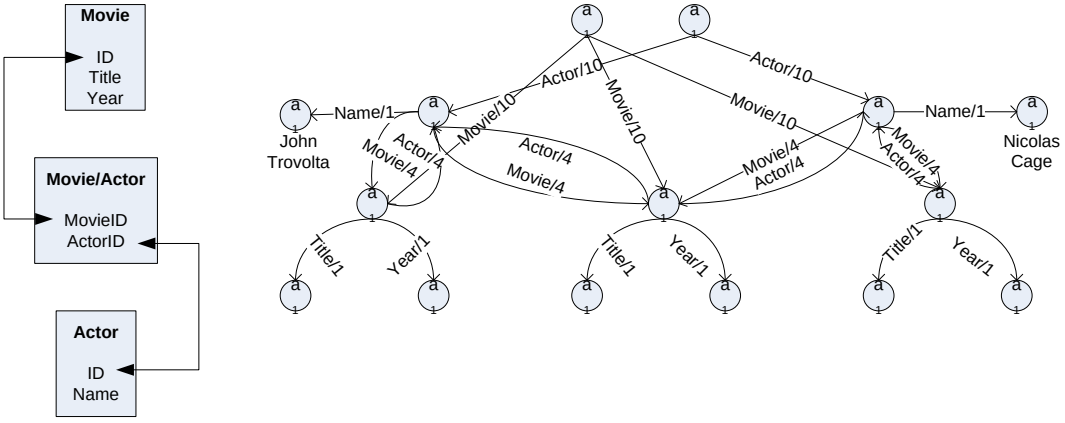
Şekil 3.28. Yakınlık araması yönteminin sonuçları [16]

Yakınlık araması yine de tam bir netlik sağlamamaktadır. Eğer tam bir sonuç elde edilmek isteniyorsa yine geleneksel veri tabanı sorgularına başvurulmalıdır. En iyi sonuç o şekilde alınacaktır. Fakat yakınlık araması özel sorguların pratik olmadığı durumlarda çok yararlıdır.

Yakınlık araması uygulamasında arama yapılacak veri tabanı, objeleri oluşturur. Uygulama Find ve Near sorgularını oluşturur. Veri tabanı bu sorguları değerlendirerek Find ve Near objelerinin sonuç setlerini oluşturur. Veri tabanı objeleri uygulama için anlamsızdır bunun için uzaklık bilgisi kullanılarak arama seti tekrar dizilir. Dizilmiş olan arama seti aynı zamanda

sonuç setini de verir. Objeler arası uzaklığın hesaplanmasında uygulamaya (X,Y,d) üçlüsü sağlanır. Burada X ve Y iki objeyi d ise bu iki obje arasındaki uzaklığı temsil eder.

Şekil 3.29'da bir veri tabanı için veri tabanı ilişkisi ve ilişkiler arası uzaklık grafiği gösterilmiştir.

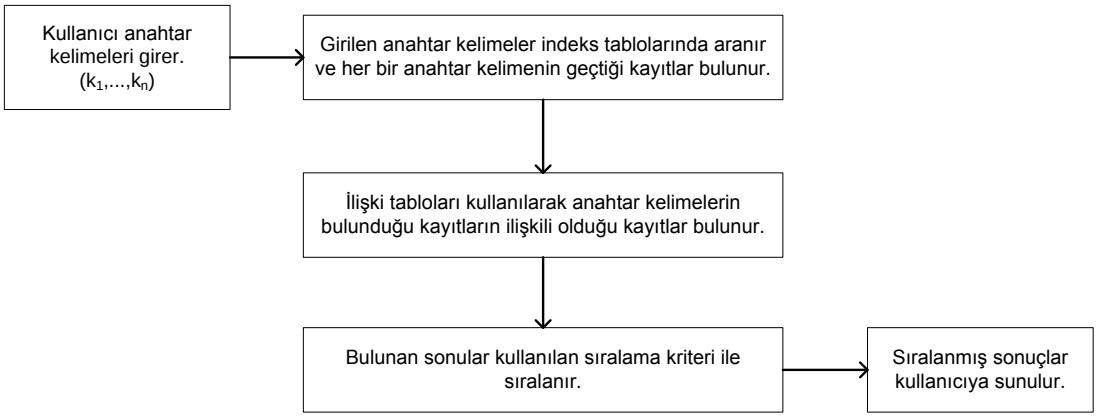


Şekil 3.29. Veritabanı objeleri arası ilişki ve uzaklık bilgileri [16]

4. MEVCUT YÖNTEMLERİN KARŞILAŞTIRILMASI

İlişkisel veri tabanlarında anahtar kelime arama işlemi; kullanıcının aramak istediği bilgi ile ilgili anahtar kelimeleri girmesi, anahtar kelimelerin ilişkisel veri tabanı tablolarında aranması, anahtar kelimenin geçtiği kayıtların tespit edilmesi, bulunan kayıtlar ile ilişkili kayıtların bulunması ve bulunan bu kayıtların sıralanıp kullanıcıya sunulması şeklinde tanımlanabilir. Kullanıcı, veri tabanını şeması hakkında bilgi sahibi olmadığından ve SQL sorgu dilini kullanamadığından verdiği anahtar kelimeler ile aramak istediği bilgiye ulaşmayı hedefler. Uygulama, kullanıcının bu isteğini yerine getirebilmek için ilişkisel veri tabanında anahtar kelimeleri arama ile başlayıp sonuçların kullanıcıya verilmesine kadar geçen işlemleri gerçekleştirir.

İlişkisel veri tabanlarında anahtar kelime probleminin çözümü için önerilen yöntemlerin akış diyagramı Şekil 4.1'de verildiği gibidir. Şekil 4.1'de de görüldüğü gibi kullanıcının tek yapması gereken istediği bilgi ile ilgili anahtar kelimeleri girmektir. Kullanıcı anahtar kelimeleri girdikten sonra istediği bilgi ile ilgili sonuçları alır. Anahtar kelimelerin girilmesi ve sonuçların elde edilmesi arasındaki işlemler uygulama tarafından gerçekleştirilir.



Şekil 4.1. İlişkisel veri tabanlarında anahtar arama uygulamalarının blok diyagramı

Anahtar kelimelerin girilmesinden sonra uygulama girilen anahtar kelimeleri indeks tablolarında arama işlemini gerçekleştirir. Anahtar kelimelerin arandığı indeks tabloları uygulama tarafından oluşturulabileceği gibi veri tabanının oluşturmuş olduğu indeks tabloları da kullanılabilir.

İndeks tablolarının uygulama tarafından oluşturulmasının tüm kelimeler için indeks tablosu oluşturulması açısından bir avantajı vardır. Tüm kelimeler için oluşturulmuş indeks tablolarında aramanın gerçekleştirilmesi oldukça hızlı olacaktır. Fakat bir dezavantajı da indeks tablolarının güncel tutulmasının gerekliliğidir. Bu ihtiyaç indeks tablolarının sürekli güncellenmesi anlamına gelmektedir ki bu da artı bir işlem ve zaman ihtiyacı demektir. Veri tabanlarının oluşturmuş olduğu indeks tablolarının kullanılması ise indeks tablolarının güncellenmesi işlemini ortadan kaldırmaktadır. Artık günümüzde veri tabanları tüm alanlar üzerinde indeks oluşturmakta ya da istenilen alanlar üzerinde indeks oluşturmaya imkân tanımaktadır. Bu nedenle veri tabanı indeks tablolarının kullanılması işlem karmaşıklığını azaltmaktadır.

İlişkisel veri tabanları üzerinde anahtar kelimelerin aranması uygulamada tercih edilen indeks tabloları üzerinde gerçekleştirilerek anahtar kelimelerin geçtiği kayıtlar elde edilir. Elde edilen bu kayıtlar aynı zamanda bize bu kaydın hangi tabloda olduğu bilgisini de verir. Fakat anahtar kelimenin

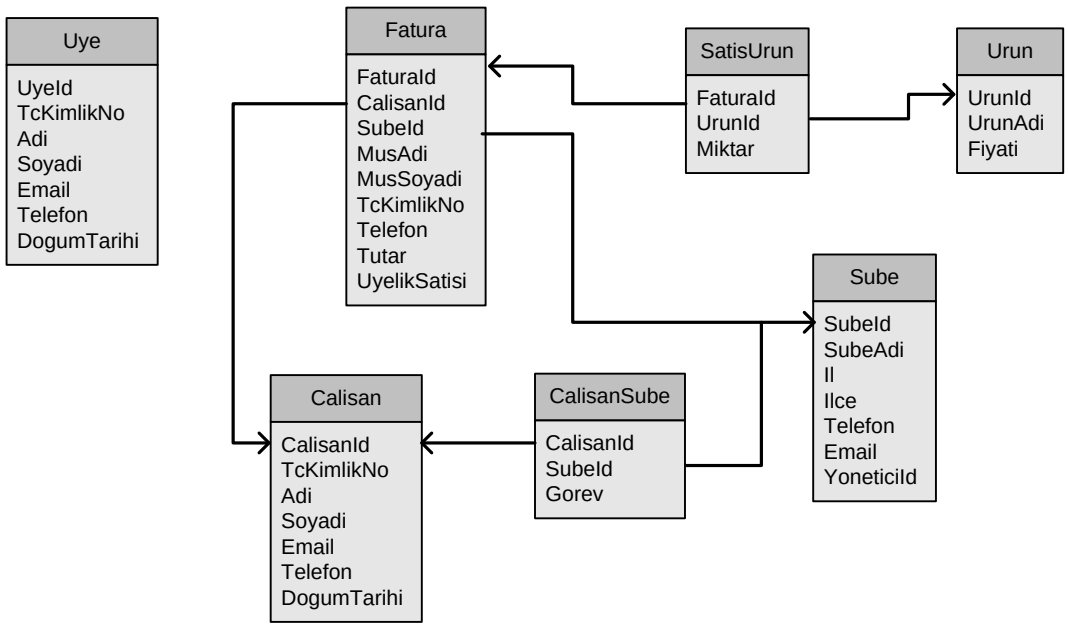
bulunduğu kaydın elde edilmesi tek başına bir anlam ifade etmemektedir. Çünkü ilişkisel veri tabanlarında bir veri ile ilgili bilgilerin tümü tek bir satırda değil farklı tabloların farklı satırlarına dağılmış durumdadır. İlişkisel veri tabanlarının bu özelliğinden dolayı anahtar kelimenin geçtiği kayıtların yanı sıra bu kayıtların ilişkili olduğu kayıtların da bulunması gerekmektedir. Bir kaydın diğer kayıtlar ile ilişkisinin elde edilmesi ilişki tablolarının kullanılması ile gerçekleştirilir.

İlişki tabloları bir tablonun başka bir tablo ile hangi alanlar üzerinden ilişkili olduğu bilgisini tutar. Bu ilişki genellikle dış anahtar → birincil anahtar ilişkisini içermektedir. İlişkisel veri tabanlarında anahtar kelime arama işlemi sonunda anahtar kelimelerin bulunduğu kayıtlar elde edilir. Bulunan bu kayıtların ilişkili olduğu kayıtların bulunması işlemi tanımlanmış olan ilişki tabloları üzerinden gerçekleştirilir. İlişki tabloları yardımı ile anahtar kelimelerin bulunduğu kayıtların ilişkili olduğu kayıtların da bulunması ve bu şekilde anahtar kelime ile alakalı olabilecek kayıtların elde edilmesi işlemi gerçekleştirilmiş olur.

Bundan sonraki adım, bulunan bu kayıtların kullanıcıya sunulacak şekle getirilmesi işlemidir. Anahtar kelime ile ilişkili bulunan tüm kayıtların kullanıcının önüne sunulması kullanıcı için çoğu zaman bir anlam ifade etmeyecektir. Çünkü kullanıcının girmiş olduğu kelimeler birçok tabloda birçok defa geçebilir. Kullanıcı birden çok anahtar kelime girmesi durumunda, her bir kelime için elde edilen kayıtlar arasında ilişki kurularak tüm anahtar kelimeleri içeren kayıtlar elde edilecektir. Fakat bazı kayıtlar sadece bir kaç tablonun birleşiminden elde edilirken, bazı kayıtlar ise birçok tablonun birleşiminden elde edilmektedir. Bunun sebebi, bir anahtar kelimenin bulunduğu kayıt ile diğer bir anahtar kelimenin bulunduğu kayıt arasında direk ya da dolaylı bir ilişkinin var olmasıdır. Bu durumda kullanıcının ilk ilgisini çekecek sonuçlar, direk bağlantıların tanılanmış olduğu kayıtlar olacaktır. Bundan dolayı uygulamanın bu kayıtları kullanıcıya ilk sıralarda sunulması önemlidir. Bu amaçla, sonuç kayıtlarının önem sırasına göre

dizilebilmesi için tablolar arası ilişki ağırlıkları tanımlanarak sonuçlar arasında sıralama gerçekleştirilir.

İlişkisel veri tabanlarında anahtar kelimelerin bulunduğu kayıtların ilişkili olduğu kayıtların da elde edilmesinden sonra sonuçların kullanıcıya sunulacak şekilde sıralanması işlemi gerçekleştirilir. Sıralama işleminin de tamamlanması ile ilişkisel veri tabanlarında anahtar kelime arama işlemi tamamlanmış olur. Yukarıda bahsettiğimiz akışın daha iyi anlaşılabilmesi için Şekil 4.2'de görülen örnek veri tabanı şeması kullanılabilir.



Şekil 4.2. Örnek veri tabanı şeması

Şekil 4.2'de de görüldüğü gibi tablolar arası dış anahtar → birincil anahtar ilişkisi tanımlanmıştır. Şekil 4.2'de gösterilen veri tabanı şema yapısı için kayıt örneği Şekil 4.3'de verildiği gibidir. Şekil 4.3'de verilen kayıt örneği üzerinden arama işlemi gerçekleştirildiğinde kayıtlar arası ilişkiye ilişki tablosundan erişilmektedir. Örneğin Şekil 4.3'deki kayılar arasından Ahmet Ak üyesi hakkında bilgi elde edilmek istensin. Kullanıcı bu bilgiyi elde etmek için anahtar kelime olarak *Ahmet* ve *Ak* anahtar kelimelerini girer. Uygulama, Şekil 4.1 'deki akış diyagramına göre girilen kelimeleri işler.

UyeId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	12345678934	Ahmet	Ak	a.ak@gazi.edu.tr	5123445566	01.01.1982
2	98765432145	Cem	Demir	c.demir@gazi.edu.tr	5125443214	02.10.1980

Üye Kaydı

UrunId	Adi	Fiyati
1	buzdolabı	1000
2	Çamaşır makinası	500
3	ütü	300

Ürün Kaydı

FaturaId	UrunId	Miktar
1	1	1
1	2	1
2	3	1
2	1	1
3	2	1
3	1	1
3	3	1

Satış Ürün İlişki Kaydı

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
1	1	1	Ahmet	Ak	12345678934	5123445566	1800	1
2	1	1	Fatif	Ok	12345678934	5123445566	1000	1
3	3	1	Kerim	Yay	32134565467	5129876757	500	0

Fatura Kaydı

SubeId	Adi	Il	Ilce	Telefon	Email	YoneticiId
1	Kızılay	Ankara	Çankaya	5124443322	Kızılay.ankara@k.c	3
2	Tunalı	Ankara	Çankaya	5125453212	Tunalı.ankara@k.c	4
3	Çaylayan	Ankara	Çankaya	5126567854	Caylayan.ankara@k.c	5

Şube Kaydı

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	43216547892	Filiz	Güzel	Filiz.guzel@k.c	5125463321	03.04.1990
2	12345678934	Ali	Ak	Ali.ak@k.c	5123445566	01.01.1982
3	32134565467	Kerim	Yay	Kerim.yay@k.c	5129876757	04.02.1980
4	76543216745	Fuat	Deli	Fuat.deli@k.c	5125673456	05.05.1978
5	87654321897	Arif	Kuru	Arif.kuru@k.c	5121234321	03.02.1976

Çalışan Kaydı

CalısanId	SubeId	Gorev
1	1	Satış Elemanı
2	2	Satış Elemanı
3	1	Yönetici
4	2	Yönetici
5	3	Yönetici

Çalışan Şube İlişki Kaydı

Şekil 4.3. Veri tabanı kayıt örneği

Adım 1: Anahtar kelimeleri veri tabanı tablolarında arama işlemin gerçekleştirir. Uygulama, arama adımı ile üye, fatura ve çalışan kayıtlarının tutulduğu tablolardan üç farklı kayda ulaşır. *Ahmet* ve *Ak* kelimeleri ile yapılan veri tabanı araması sonucunda elde edilen sonuçlar Şekil 4.4'de gösterildiği gibidir. Şekil 4.4'de de görüldüğü gibi veri tabanı araması verilen anahtar kelimelerin geçtiği kayıtlara ulaşmıştır. Henüz anahtar kelimelerin bulunduğu kayıtların ilişkili olduğu kayıtlar elde edilmemiştir. Anahtar kelimelerin bulunduğu kayıtlar bu şekli ile kullanıcı için bir anlam ifade etmez. Elde edilen kayıtlar ile üye kayıtlarının tutulduğu tablodan *Ahmet* ve *Ak* anahtar kelimeleri aynı satırda yer aldığından bir kayıt döner ve bu kayıttan email, telefon ve doğum tarihi bilgilerine ulaşabiliriz. Aynı anahtar kelimeler için fatura kayıtlarının tutulduğu tablodan da her iki anahtar kelime aynı satırda yer aldığından bir kayıt sonuç olarak döner. Bu kayıttan da anahtar kelimeler ile ilgili TC kimlik numarası, telefon, tutar ve üyelik satışı bilgilerine ulaşabiliriz. Son olarak ta çalışan kayıtlarının tutulduğu tablodan *Ak* anahtar kelimesine ulaşılır ve bu tablodan da tek bir kayıt sonuç olarak döner. Çalışan kaydının tutulduğu tablodan da anahtar kelimeler ile ilgili email, telefon ve doğum tarihi bilgilerine erişilebilir.

UyeId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	12345678934	Ahmet	Ak	a.ak@gazi.edu.tr	5123445566	01.01.1982

Üye

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
1	1	1	Ahmet	Ak	12345678934	5123445566	1800	1

Fatura

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
2	12345678934	Ali	Ak	Ali.ak@k.c	5123445566	01.01.1982

Çalışan

Şekil 4.4. Ahmet ve Ak kelimelerinin veri tabanında arama sonucu

Adım 2: Uygulamalarda anahtar kelimelerin bulunduğu kayıtların elde edilmesinden sonra ikinci adımda ilişki tablosundan yararlanılarak ilişkili kayıtların elde edilmesi gerçekleştirilmiştir. Ele alınan örnekte anahtar

kelimelerin ilişkisel veri tabanında aranması ile elde edilen üç kaydın ilişkili olduğu tabloların bulunabilmesi için genellikle kullanılan yöntem olan dış anahtar → birincil anahtar ilişki tablolarının kullanılmasıdır. Bu ilişki tablosunun kullanılması ile *Ahmet* ve *Ak* anahtar kelimelerinin bulunduğu kayıtların ilişkili olduğu kayıtlar Şekil 4.5'te gösterildiği gibidir. Şekil 4.5'ten de görüleceği gibi ilişki tablolarının kullanılması ile ilgili kaydın detay bilgileri elde edilmiştir. Örneğin ilişki tablosunun kullanılması ile fatura tablosunda bulunan kaydın hangi çalışan tarafından ve hangi şubede faturalandığı aynı zamanda da faturalamayı yapan çalışanın hangi şubede ne görevde olduğu bilgilerine erişilir. Üye bilgilerinin tutulduğu tablonun dış anahtar → birincil anahtar ilişkisine sahip olmaması bu üye kaydı için herhangi bir detay bilgiye erişilemez. Çalışan bilgilerinin tutulduğu tablodaki kayıt için ise çalışanın hangi şubede hangi görevde olduğu bilgisine ve aynı zamanda hangi faturaları düzenlediği gibi detay bilgilere erişilir.

UyeId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	12345678934	Ahmet	Ak	a.ak@gazi.edu.tr	5123445566	01.01.1982

Üye

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
1	1	1	Ahmet	Ak	12345678934	5123445566	1800	1

Fatura

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	43216547892	Filiz	Güzel	Filiz.guzel@k.c	5125463321	03.04.1990

Çalışan

SubeId	Adi	Il	Ilce	Telefon	Email	YoneticiId
1	Kızılay	Ankara	Çankaya	5124443322	Kızılay.ankara@k.c	3

Şube

CalısanId	SubeId	Gorev
1	1	Satış Elemanı

ÇalışanŞube

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
2	12345678934	Ali	Ak	Ali.ak@k.c	5123445566	01.01.1982

Çalışan

CalısanId	SubeId	Gorev
2	2	Satış Elemanı

ÇalışanŞube

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
2	1	1	Fatif	Ok	12345678934	5123445566	1000	1

Fatura

Şekil 4.5. Anahtar kelime kayıtlarının ilişkili olduğu kayıtlar

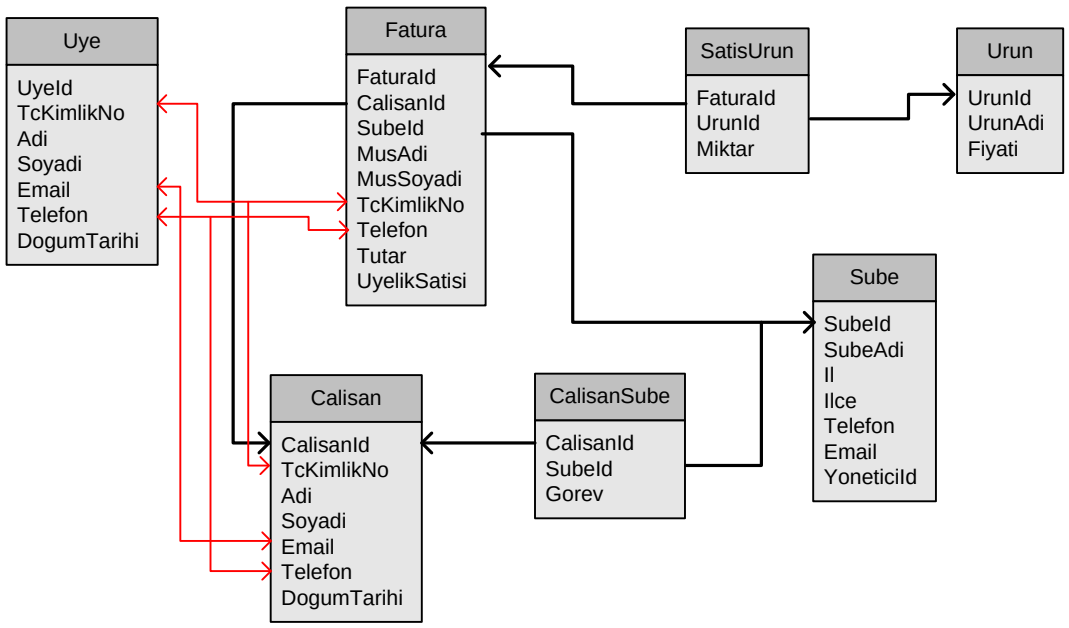
Adım 3: İlişkilerin çıkarılmasından sonraki adım sonuçların sıralanması ve kullanıcıya sunulması adımıdır. Sonuçların sıralanmasında tablolar arası ilişkiler ağırlıklandırılır, bir sıralama sağlanır ve sonuçlar kullanıcıya sunulur.

Örnekten de görülebileceği gibi üye bilgilerinin tutulduğu tablo ile diğer tablolar arasında dış anahtar → birincil anahtar ilişkisi bulunmadığından bir üye kaydının hangi alışverişleri yaptığı, aynı zamanda bir çalışan mı, çalışan ise hangi şubede çalışmaktadır gibi bilgilerine ulaşılamamaktadır. Üye tablosunun bir kaydının detay kayıtlarına ulaşılamamasının sebebi ilişki tablosunu sadece dış anahtar → birincil anahtar ilişkisinin oluşturmasıdır.

Oysaki veri tabanı kayıtları incelendiğinde üye bilgilerinin tutulduğu tablo ile fatura ve çalışan bilgilerinin tutulduğu tablolar arasında ilişki kurulabilir. Çünkü üye tablosu incelendiğinde tckimlikno, email ve telefon alanları ilgili kayıt için ayıt edici bir özelliktir. Tckimlikno ve telefon alanlarının fatura bilgilerinin tutulduğu tabloda da yer alması iki tablo arasında bir ilişkiyi ortaya çıkarabilir. Yine aynı şekilde çalışan bilgilerinin tutulduğu tablodaki email, telefon ve tckimlikno alanları üzerinden kurulacak ilişki üye bilgilerinin tutulduğu tablo ile aradaki ilişkiyi ortaya çıkarabilir.

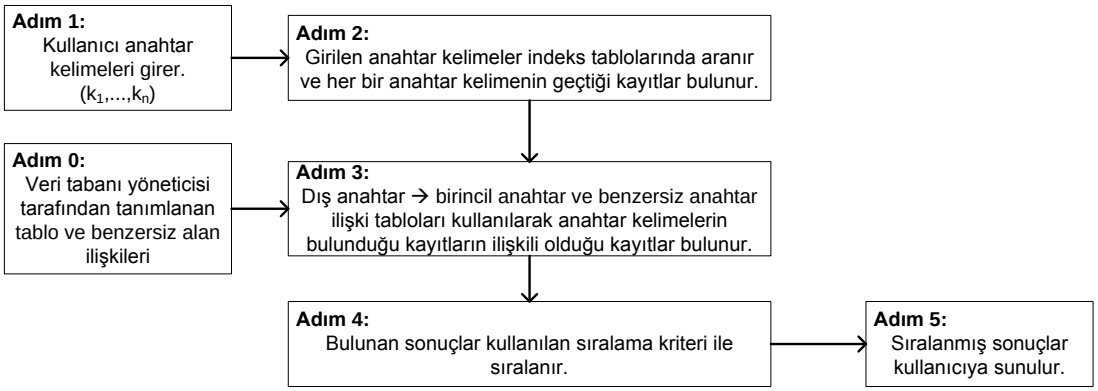
5. ÖNERİLEN YÖNTEM

Önceki bölümde incelediğimiz mevcut yöntemlerde tablolar arası ilişkiler veri tabanı tasarımı sırasında tanımlanmış dış anahtar → birincil anahtar ilişkisi üzerinden kurulmaktadır. Ancak birçok veri tabanında tasarım sırasında tanımlanmasa bile kullanım sırasında ortaya çıkan mantıksal ilişkiler vardır. Örnek vermek gerekirse veri tabanı tasarımı sırasında telefon numarası içeren alanlar genellikle kayıtları ilişkilendirmek için kullanılmazlar. Bu alanlar kayıtla ilgili ek bilgi olarak düşünülürler, ancak mantıksal olarak bir telefon numarası aslında bir kişiyi ya da aile gibi ilişkili kişileri ifade eder. Bu alanlar üzerinde de tanımlanacak ilave ilişkiler ile ilişkisel veri tabanlarında anahtar kelime arama sonuçları genişletilebilir. Bu amaçla bu çalışmada önerilen yöntem, ilişkilerin çıkarılmasında ayırt edici özelliğe sahip bu alanlar üzerinden de ilişki tanımlayarak aramanın genişletilmesidir. Bir önceki bölümde kullandığımız veri tabanı şeması üzerine ayırt edici özelliklerin bulunduğu alanlar arasındaki ilişkiler de eklenirse tasarım sırasında ortaya çıkmayan ilişkiler de elde edilmiş olur. Ayırt edici alanlar üzerinden kurulan ilişkiler Şekil 5.1'de ifade edilmiştir.



Şekil 5.1. Anahtar olmayan alanlar üzerinden ilişkilerin tanımlanması

Önerilen bu yöntem Şekil 4.1 ile verilen anahtar kelime araması blok diyagramının üçüncü adımına katkı sağlamaktadır. Önerilen yöntemin blok diyagramı Şekil 5.2'de gösterildiği gibidir.



Şekil 5.2. Önerilen yöntem blok diyagramı

Önerilen yöntemin blok diyagramı incelendiğinde arama sonucu elde edilecek sonuçların detaylandığı görülebilir.

Adım 0: Önerilen yöntemde veri tabanı yöneticisi veri tabanı şeması üzerindeki telefon numarası, e-posta adresi gibi benzersiz alan ilişkilerini önceden tanımlar. Önerilen yöntem bu ilişkileri kayıtlar arası ilişkileri bulmada kullanır.

Adım 1: Kullanıcı arama yapmak istediği kelime kümesi K'yı tanımlar. K kümesi birden fazla kelime içerebilir ($K= \{k_0, ... ,k_n\}$).

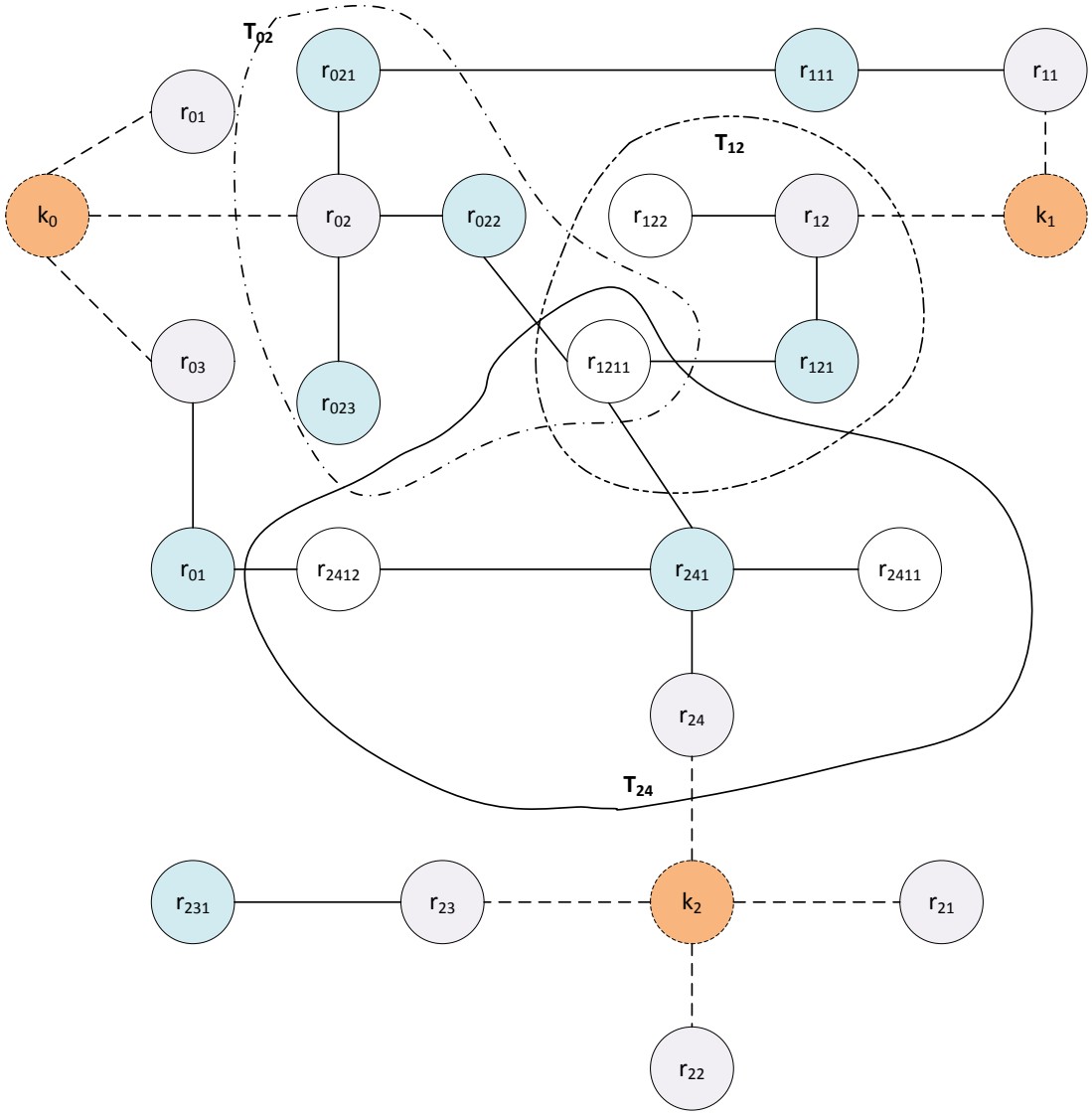
Adım 2: Literatürdeki çalışmalarda da olduğu gibi öncelikle uygulamaya girilen anahtar kelimelerin veri tabanı tablolarında arama işlemi gerçekleştirilir. Bu arama sonucunda her bir anahtar kelimenin geçtiği kayıtlar bulunur ve anahtar kelime ile ilişkili bir küme içinde tutulur ($R_i = \{r_{i0}, ... r_{im}\}$). Veri tabanında kelimeler için ayrı ayrı yapılan aramalar R kümesi altında birleştirilir ($R = \{R_0, ... , R_n\}$).

Adım 3: R kümesi içinde yer alan her bir r_{ij} kaydı (anahtar kelime k_i'nin j. kaydı) ile ilişkili diğer kayıtlar Adım 0'da tanımlanan tablolar arası ilişkiler kullanılarak bulunur. Bulunan kayıtlar T_{ij} ağaç yapısı içinde saklanır. Böylece r_{ij} kaydı ile ilişkili bulunan diğer kayıtlar bir hiyerarşik yapı içinde gösterilebilmiş ve bulunan bu kayıtlar ile r_{ij} arasındaki yakınlık ilişkisinin mesafesi korunabilmiş olur. Bu noktada her girilen anahtar kelime ile ilişkili kayıtlar ve o kayıtlar ile ilişkili diğer kayıtlar bulunmuştur, ancak kullanıcının asıl isteği girdiği anahtar kelimelerin hepsi ile ilgili olan kayıtları görmektir. Bu amaçla her bir kelime için bulunan T_{ij} ağaçlarının kesişimi alınır.

Adım 4: Anahtar kelimelerin bulunduğu kayıtlar ve bu kayıtların ilişkili olduğu kayıtların elde edilmesinden sonra literatürde kullanılmış yöntemler ile sonuçların sıralanması işlemi gerçekleştirilir.

Adım 5: Bulunan sonuçlar kullanıcıya gösterilir.

Önerilen yöntem mevcut yöntemlerin izlediği akışa ek olarak Adım 0'ı eklemekte ve Adım 3 bu eklemeye göre değişiklik yapmaktadır. Bu tez kapsamında Adım 0 ve 3 üzerine yoğunlaşılmış ve diğer adımlar üzerinde bir iyileştirme yapılmamıştır.

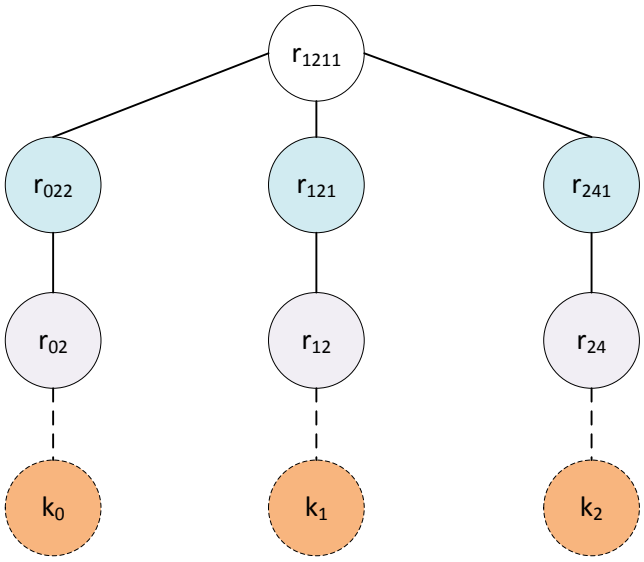


Şekil 5.3. k_0, k_1, k_2 anahtar kelimeleri için bulunan T_{ij} ağaç yapısı

Şekil 5.3'te kullanıcının aramak istediği anahtar kelime kümesi K = $\{k_1,k_2,k_3\}$ şeklindedir. Her bir anahtar kelime k_i için veri tabanında arama yapıldığında R_i kümesi elde edilir. Şekil 5.3'te her R_i R_0 = $\{r_{01},r_{02},r_{03}\}$, R_1 = $\{r_{11},r_{12}\}$, R_2 = $\{r_{21},r_{22},r_{23},r_{34}\}$ kümeleri şeklindedir.

Bir sonraki adımda ise anahtar kelimelerin bulunduğu tablolar ile diğer tablolar arasında tanımlı ilişkilerden T_{ij} ağaç yapıları oluşur. Şekil 5.3 T_{02}, T_{12}, T_{24} ağaç yapılarını göstermektedir. Şekil 5.3'te görüldüğü gibi T_{ij} ağaç yapıları birbirleri ile kesişebilmektedir. Bu durum da k_1, k_2, k_3 anahtar kelimelerinin bulunduğu kayıtların veri tabanında tanımlı yada sonradan tanımlanmış tablolar arası ilişkiden aynı kayıtlara ulaştığını göstermektedir.

Her bir anahtar kelime için oluşturulmuş olan ağaçlar içerisinde farklı anahtar kelimeler için aynı dallar mevcuttur. Uygulamadaki amaç, bu ortak dallar üzerinden birden fazla anahtar kelimeyi içeren ağaçlar oluşturmaktır. Bu amaçla bir anahtar kelime için oluşturulmuş ağacın dalları diğer bir anahtar kelime için oluşturulmuş ağaç içerisinde aratılır. Ulaşılan ortak dal üzerinden en az iki anahtar kelimeyi içerecek şekilde sonuç ağaçlarına ulaşılır. Aranan anahtar kelimeler k_0, k_1 ve k_2 için oluşturulmuş ağaçları gösteren Şekil 5.3'ten elde edilen sonuç ağacı Şekil 5.4'de gösterildiği gibidir.



Şekil 5.4. T_{01}, T_{11}, T_{21} ağalarının kesişimi

Şekil 5.4'de görüldüğü gibi iki veya daha fazla anahtar kelime için ortak kayıtlar bulunmaktadır. Aynı şekilde Şekil 5.4'de k_0, k_1 ve k_2 anahtar

kelimeleri için T_{01}, T_{11} ve T_{21} ağaç yapılarının ortak dalı olan r_{1211} dalı üzerinden ilişkilidir.

Önerilen yöntemin adımlarını daha iyi açıklamak için daha önce incelenen örnek ele alınacak olursa burada üye bilgilerinin tutulduğu tablodaki *tckimlikno* alanı, çalışan bilgilerinin tutulduğu tablodaki *tckimlikno* alanı ve fatura bilgilerinin tutulduğu tablodaki *tckimlikno* alanı benzersiz anahtar alanı olarak değerlendirilebilir. Bu durumda tablolar incelendiğinde üye bilgilerinin tutulduğu tablodan elde edilen kaydın *tckimlikno* alanındaki değer ile fatura ve çalışan kayıtlarının tutulduğu tablolardaki *tckimlikno* alanlarındaki değerler eşleşmektedir. Hatta fatura bilgilerinin tutulduğu iki satırda aynı *tckimlikno* alan bilgisine rastlanmaktadır. Bu da faturada geçen iki isim arasında ilişki olduğu bilgisine erişmemizi sağlamaktadır. Şekil 5.5'de görüldüğü gibi üye, çalışan ve fatura bilgilerinin tutulduğu tablolarda *tckimlikno* alanı aynı değere sahiptir. Bu alan üzerinden kurulan ilişki ile üyenin aynı zamanda bir çalışan olduğu bilgisine ve aynı zamanda da bu üyenin alışveriş bilgilerine erişilmiştir. Oysaki sadece dış anahtar → birincil anahtar ilişkisi kullanılarak yapılan arama bize bu detay bilgileri sağlayamamaktadır. Önerilen yöntem ile ayırt edici olduğu belirtilmiş olan alanlar üzerinden tanımlanmış ilişkiler ile dış anahtar → birincil anahtar ilişkisinden elde edilemeyen sonuçlara ulaşılabilmektedir.

UyeId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	12345678934	Ahmet	Ak	a.ak@gazi.edu.tr	5123445566	01.01.1982

Üye

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
2	12345678934	Ali	Ak	Ali.ak@k.c	5123445566	01.01.1982

Çalışan

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
1	1	1	Ahmet	Ak	12345678934	5123445566	1800	1
2	1	1	Fatif	Ok	12345678934	5123445566	1000	1

Fatura

Şekil 5.5. Tanımlanan ilişkiler ile üye tablosu için elde edilen detay bilgiler

Önerilen yöntemde dış anahtar → birincil anahtar ilişkilerinin kullanılmasının yanında, dış anahtar → birincil anahtar ilişkisi tanımlanmamış alanlar üzerinden kurulacak ilişkiler de ele alınarak ilişkisel veri tabanlarında anahtar kelime aramanın ikinci adımı olan ilişkili kayıtların bulunmasına katkı sağlanmıştır.

5.1. Benzersiz Alanların Tanımlanması

Benzersiz alan tanımlaması veri tabanı yöneticisi tarafından gerçekleştirilir. Veri tabanı yöneticisi önceden tanımlanmış etiketlere uygun olarak veri tabanı tablolarının alanlarını etiketler. Her hangi bir kısıt olmamakla birlikte önerilen alan etiketleri aşağıdaki listede sıralanmıştır.

1. *TC Kimlik No:* Türkiye Cumhuriyeti vatandaşları arasında eşsiz bir özellikte dağıtılmıştır. Geçerli bir TC Kimlik numarası her zaman için bir bireyi eşsiz olarak tanımlar.

2. *E-Posta Adresi:* Bir e-posta adresi eşsiz olarak bir e-posta hesabını tanımlar. Aynı kişinin birden fazla e-posta adresi olabilir ancak mantıksal olarak bir e-posta adresi birden fazla kişiye ait olamaz. Kişiler genellikle bir e-posta adresini aktif olarak kullanmakta ve bir çok resmi işlem için bu e-posta adresini vermektedir.

3. *Telefon Numarası:* Telefon numaraları da aynı e-posta adresleri gibi telefon sisteminde eşsiz olarak tanımlanırlar. Bir kişiye ait birden fazla telefon numarası olacağı gibi aynı telefon numarası başka kişiler tarafından da kullanılabilmektedir (Ör: aile, işletme vb.). Aslında bu özelliği istenilen bir durumdur. Böylece aile gibi kişiler arası ilişkilerde ortaya çıkartılabilmektedir.

Önerilen alan etiketleri bunlar olmasına rağmen önerilen yöntem tarafından bir kısıt bulunmamaktadır. Kullanılan veri tabanına yada uygulamaya göre farklı etiketler de tanımlanabilir.

Veri tabanı yöneticisi alanları etiketledikten sonra önerilen yöntem bu tablolar arası ilişkiyi bu etiketler üzerinden kurarak dış anahtar → birincil anahtar ilişkisi ile oluşturduğu tablolar arası ilişkiyi günceller.

5.2. İlişkili Kayıtların Bulunması

Önerilen yöntem aranmak istenilen bir anahtar kelime (k_i) için öncelikle bu anahtar kelimenin geçtiği kayıtları (r_{ij}) veri tabanından bulur. Bu adımda veri tabanı indeksleri kullanılabileceği gibi, amaca uygun olarak daha karmaşık indeksleme yapıları da geliştirilebilir ya da çok basit bir şekilde istenilen kelime veri tabanındaki tablolarda tek tek aranabilir. Bu çalışma kapsamında anahtar kelimelerin aranması konusunda bir öneri sunulmamıştır.

Bu arama sonucunda bulunan her bir kayıt anahtar kelime ile doğrudan ilişkilidir ve sonuçlarda yer alacaktır. Bir sonraki adımda önerilen yöntem özyinelemeli bir biçimde bu kayıtlara bağlı diğer kayıtları veri tabanında arayacaktır. Arama sonuçları bir ağaç yapısı içinde tutulurlar bu şekilde kayıtlar arası ilişkiler korunurken kayıtlar arası yakınlık ve uzaklık ilişkisi de korunmuş olur. İlk arama sonucu elde edilen anahtar kelimeyi içeren kayıt oluşturulacak ağaç yapısını kökünü oluşturur. Ağaç yapısında kayıtların köke olan uzaklığı kaydın ağaç içindeki seviyesine göre belirlenir. Kök kaydın

seviyesi 0 olarak atanır. Her yeni eklenen kaydın seviyesi bağlandığı kaydın seviyesinden bir fazladır. Önce, kök kayıt ile dış anahtar → birincil anahtar ilişkisi içinde olan doğrudan ilişkili kayıtlar, daha sonra tanımlanan benzersiz alanlara göre dolaylı ilişki içeren kayıtlar bulunmaktadır. İlk aşamada r_{ij} kaydını içeren tablo ile dış anahtar → birincil anahtar ilişkisi içinde olan diğer tablolar anahtar alan üzerinden ilişkilendirilir. Bu ilişki veri tabanın tasarımı sırasında oluşturulduğundan, bu ilişkinin varlığı açık ve kesindir. Bu nedenle bu ilişki doğrudan ilişki olarak nitelendirilebilir. İkinci aşamada eğer r_{ij} kaydını içeren tablonun kolonlarından biri benzersiz alan olarak işaretlendi ise bu benzersiz alanı içeren diğer tablolarda bu alan üzerinden aramaya tabi tutulur. Bu ilişki mantıksal olarak kurulduğundan dolaylı ilişki olarak adlandırılabilir.

İlişkiler üzerinden bulunan bu kayıtlar kök kaydın altına eklenirler ve Seviye 1 kayıtları oluştururlar. Her bir kayıt için arama işlemi tekrarlanır ve o kayıtla ilişkili diğer kayıtlara ulaşılır. Arama sonucunda bulunan kayıtlar arama işlemini tetikleyen kaydın altına eklenirler. Arama sonucunda bir kayda birden fazla farklı kayıttan ulaşılmış olabilir. Bu aynı kaydın birden fazla tekrarına ve arama işlemini uzamasına neden olacaktır. Bu durumu engellemek için, bulunan her bir kayıt öncelikle ağacın içinde aratılır. Eğer kayıt bulunmamışsa, kaynak kaydın altına eklenir ve seviyesi kaynak kaydın bir fazlası olarak işaretlenir. Eğer kayıt başka bir kaynak kayıt tarafından da ulaşılmışsa seviyesi düşük olan kayıt korunurken diğeri ağaçtan silinir. Bu arama işlemi yeni kayıt bulunamayana kadar devam eder.

5.3. Bulunan Sonuçların Birleştirilmesi

Kullanıcı birden fazla anahtar kelime ile aratma yaptığı durumda, her bir anahtar kelime için bulunan sonuçların birleştirilerek kullanıcıya sunulması gerekir. Kullanıcı arama yaptığı anahtar kelimelerin hepsi ile ilgili kayıtları görmek ister. Önerilen yöntem bu gereksinimi karşılamak için, her anahtar kelime sonuç ağacında bulunan aynı kayıtları seçer ve onların anahtar

kelimeler ile olan ilişkilerini bir ağaç yapısı altında gösterir. Böylece kullanıcı her anahtar kelime ile ilişkisi bulunan kaydı görüntülerken aynı zamanda onların anahtar kelimeler ile aralarında nasıl bir ilişki olduğunu da görebilir.

6. SONUÇLAR

6.1. Ortam

Uygulamada ücretsiz ve yaygın olarak kullanılmasından dolayı MySQL veri tabanı tercih edilmiştir. Uygulama Java programlama dili kullanılarak Windows 7 ortamında gerçekleştirilmiştir. Veri tabanı bağlantısı olarak ta MySQL tarafından sağlanılan JDBC (Java Database Connectivity) kütüphanesi kullanılmıştır.

6.2. Veri Kümesi

Bir alışveriş firması için tasarlanmış veri tabanın küçük bir parçası kullanılmıştır. Veri tabanı Uye, Fatura, Calisan, CalisanSube, SatisUrun, Sube ve Urun tablolarından oluşmaktadır.

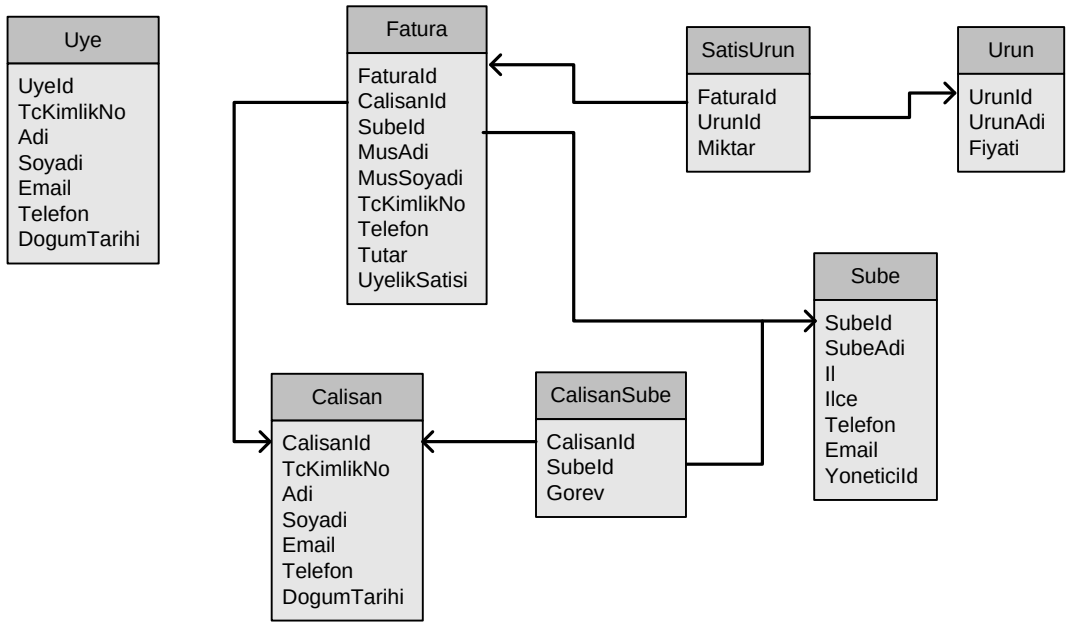
Veri tabanı ilk tasarım sırasında Fatura, Calisan, CalisanSube, SatisUrun, Sube ve Urun tablolarından oluşmaktadır. Daha sonradan Uye tablosuna ihtiyaç duyulmuş ve Uye tablosu yaratılmıştır. Yeni oluşturulan Uye tablosu var olan tablolar ile ilişkilendirilemediğinden Uye tablosunun diğer tablolar ile herhangi bir dış anahtar → birincil anahtar ilişkisi yoktur.

Uye tablosu 30 farklı üyenin kaydından oluşmakta, Fatura tablosu 100 farklı fatura bilgisinden oluşmakta, Calisan tablosu 15 farklı çalışan kaydından oluşmakta, CalisanSube tablosu 15 farklı veriden oluşmakta, SatisUrun tablosu 100 farklı veriden oluşmakta, Sube tablosu 3 farklı kayıttan oluşmakta ve Urun tablosu ise 40 farklı veriden oluşmaktadır.

6.2.1. Veri tabanı şeması

Bu çalışmada kullanılan veri tabanı şeması Şekil 6.1'de gösterildiği gibidir. Şekil 6.1'de de görüldüğü gibi veri tabanı şeması Uye, Fatura, Calisan, CalisanSube, SatisUrun, Sube ve Urun tablolarından oluşmaktadır. Uye

tablosu UyeId, TckimlikNo, Adi, Soyadi, Email, Telefon ve DoğumTarihi olmak üzere toplan 7 alandan oluşmaktadır. Fatura tablosu FaturaId, CalisanId, SubeId, MusAdi, MusSoyadi, TckimlikNo, Telefon, Tutar ve UyelikSatisi olmak üzere 9 alandan oluşmaktadır. Calisan tablosu CalisanId, TckimlikNo, Adi, Soyadi, Email, Telefon ve DogumTarihi olmak üzere 7 alandan oluşmaktadır. CalisanSube tablosu CalisanId, SubeId ve Gorev olmak üzere 3 alandan oluşmaktadır. SatisUrun tablosu FaturaId, UrunId ve Miktar olmak üzere 3 alandan oluşmaktadır. Urun tablosu UrunId, UrunAdi ve Fiyati olmak üzere 3 alandan oluşmaktadır. Son olarak Sube tablosu da SubeId, SubeAdi, Il, Ilce, Telefon, Email ve YoneticiId olmak üzere 7 alandan oluşmaktadır. Tablolar arası dış anahtar → birincil anahtar ilişkisi Şekil 6.1'de gösterildiği gibidir.



Şekil 6.1. Uygulamada kullanılan örnek veri tabanı şeması

6.2.2. Örnek kayıt

Çalışmada kullanılan veri tabanı şemasına bakıldığında tabloları oluşturan kayıtlardan bir örnek Şekil 6.2'de gösterildiği gibidir.

UyeId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	12345678934	Ahmet	Ak	a.ak@gazi.edu.tr	5123445566	01.01.1982

Üye

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
1	1	1	Ahmet	Ak	12345678934	5123445566	1800	1

Fatura

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
1	43216547892	Filiz	Güzel	Filiz.guzel@k.c	5125463321	03.04.1990

Çalışan

SubeId	Adi	Il	Ilce	Telefon	Email	YoneticiId
1	Kızılay	Ankara	Çankaya	5124443322	Kızılay.ankara@k.c	3

Şube

CalısanId	SubeId	Gorev
1	1	Satış Elemanı

ÇalışanŞube

CalisanId	TcKimlikNo	Adi	Soyadi	Email	Telefon	DogumTarihi
2	12345678934	Ali	Ak	Ali.ak@k.c	5123445566	01.01.1982

Çalışan

CalısanId	SubeId	Gorev
2	2	Satış Elemanı

ÇalışanŞube

FaturaId	CalisanId	SubeId	MusteriAdi	MusteriSoyad	TcKimlikNo	Telefon	Tutar	UyelikSatisi
2	1	1	Fatif	Ok	12345678934	5123445566	1000	1

Fatura

Şekil 6.2. Uygulamada kullanılan veri tabanına ait kayıt örneği

6.3. Yöntem

Şekil 6.1 veri tabanı şemasında görüldüğü gibi tablolar arası dış anahtar → birincil anahtar ilişkisi mevcuttur. Veri tabanı şeması incelendiğinde ise görülebilir ki bazı alanlar ilgili kayıt için benzersiz alanlardır. Bu tip alanlar sayesinde diğer tablolar ile ilişki oluşturulabilecekken bu alanlar üzerinden herhangi bir ilişki tanımlanmamıştır.

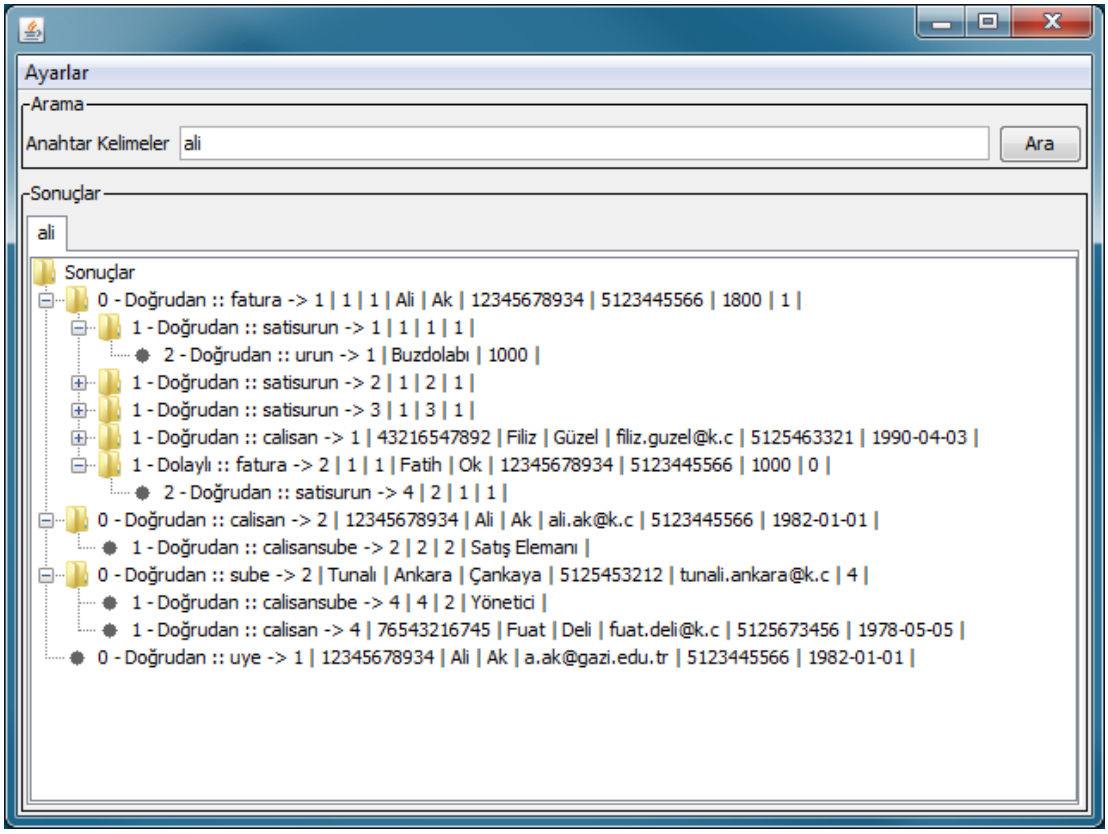
Çalışmada benzersiz alanlar üzerinden tanımlanan tablolar arası ilişkiler sayesinde anahtar kelime arama detaylandırılmıştır. Hatta dış anahtar →

birincil anahtar ilişkisi ile elde edilemeyecek sonuçlara ulaşılmıştır. Örneğin Uye tablosunun herhangi bir tablo ile dış anahtar → birincil anahtar ilişkisi yokken benzersiz alan olarak nitelendirebileceğimiz TckimlikNo alanı üzerinden tanımlanmış bir ilişki ile Uye tablosunun diğer tablolar ile ilişkisi ortaya çıkarılabilir.

Benzersiz alan olarak tanımlayabileceğimiz alanlar Uye tablosu için TcKimlikNo, Email ve Telefon olurken, Fatura tablosu için TckimlikNo ve Telefon, Calisan tablosu için de TckimlikNo, Email ve telefon alanları olabilir. Bu alanlar sayesinde tablolar arası farklı ilişkiler elde edilerek anahtar kelime arama sonuçları zenginleştirilmiştir.

6.4. Uygulama

Önerilen yöntemin denenebilmesi için Java programlama dili kullanılarak bir uygulama geliştirilmiştir. Aşağıdaki şekilde uygulamanın kullanıcı ara yüzü yer almaktadır.



Şekil 6.3. Kullanıcı ara yüzü

Uygulama kullanıcı tarafından girilen “Anahtar Kelimeler” alanına girilen kelimeler için bağlantı kurduğu veri tabanı üzerinde arama yapar ve sonuçları bir ağaç yapısı halinde kullanıcıya gösterir. Oluşan kullanıcı ara yüzü Şekil 6.3’de gösterildiği gibidir. Ağaç yapısında yer alan verinin biçimi aşağıdaki gibidir:

<Seviye> – <İlişki Türü> :: <TabloAdı> → <Kayıt>

- *Seviye:* Kaydın köke olan uzaklığını gösterir. Kök kaydın seviyesi “0”dır.

- *İlişki Türü:* Doğrudan ya da dolaylı olabilir.

- Doğrudan ilişki, bu kaydın bağlı olduğu kayda dış anahtar → birincil anahtar ilişkisi ile bağlı olduğunu gösterir.

- Dolaylı ilişki, bu kaydın benzersiz alanlar üzerinden bir ilişkisi olduğunu gösteri.

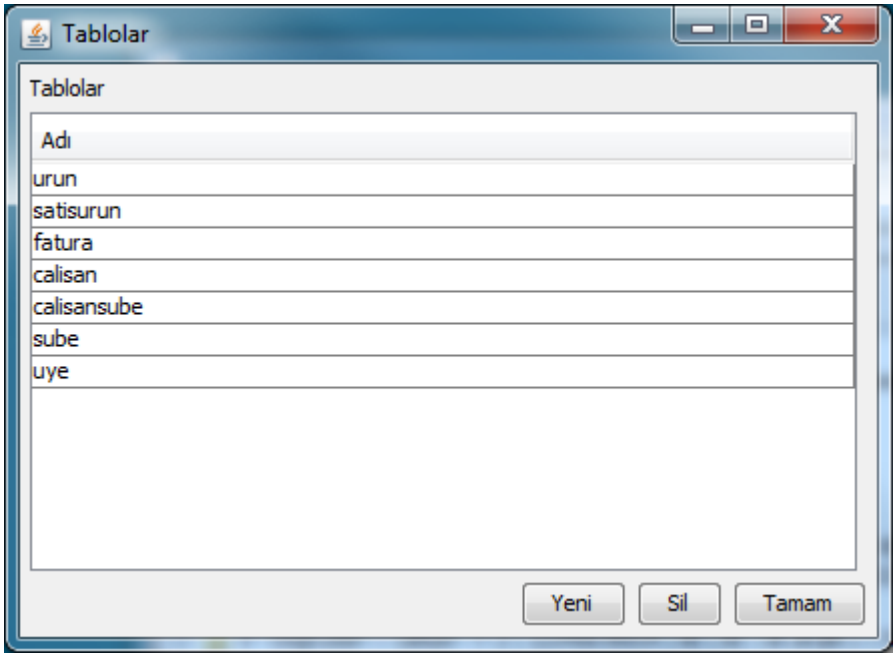
- *Tablo Adı:* Kaydın geçtiği tablonun adını gösterir.

- *Kayıt:* Bulunan kaydı gösterir. Bu gösterimde kaydın bütün alanları yer almaktadır ve her bir alan birbirlerinden "|" karakteri ile ayrılmıştır.

Uygulama öncelikle aranmak istenilen anahtar kelimeyi veri tabanın tümünde arar ve kelimeyi içeren kayıtları içeren kayıtları oluşturacağı ağaç yapısına yerleştirir. Bu kayıtlar oluşturulacak ağacın kökleri oldukları için seviyeleri "0" olarak işaretlenir. Örnek ekran görüntüsünde kullanıcı "ali" kelimesini aratmıştır. Uygulama veri tabanında bulduğu içinde "ali" geçen kayıtları oluşturduğu ağacın köküne yerleştirmiştir. Bu aşamadan sonra her bir kaydın önce doğrudan ilişkileri daha sonra dolaylı ilişkileri araştırılır. Bu şekilde önerilen yöntemin sağladığı iyileştirme daha net ve tarafsız bir şekilde gösterilmiş olur.

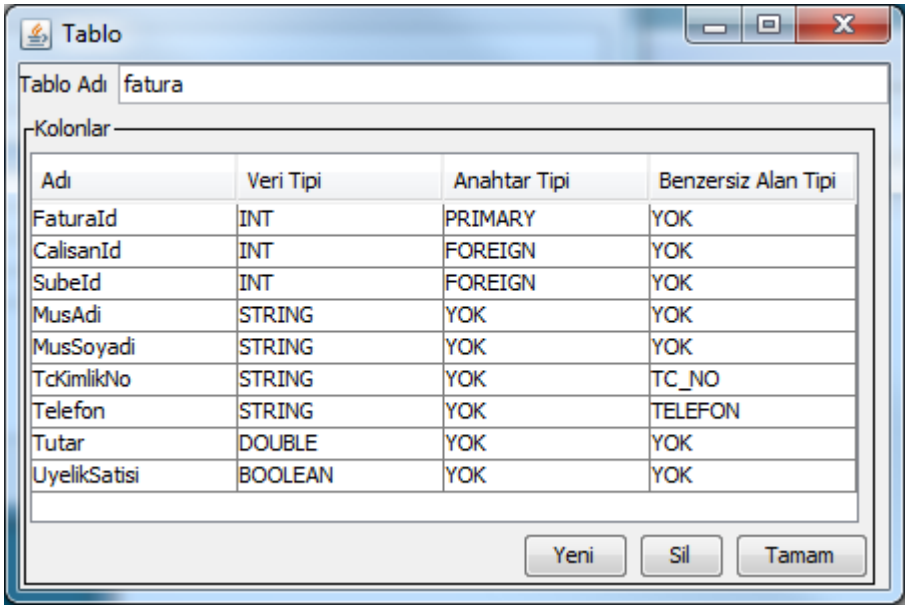
6.4.1. Veri tabanı şemasının tanımlanması

Uygulama arama işlemlerini gerçekleştirebilmek için veri tabanı yapısını ve benzersiz alanların hangileri olduğunu bilgisini kurulum sırasında kullanıcıdan almaktadır. Bu bilgiyi almak için "Ayarlar" menüsü altındaki "Şema Tanımlama" işlemi kullanılır. Kullanıcı üzerinde arama yapmak istediği veri tabanı şemasını bu menü yardımı ile tanımlar. Öncelikle, Şekil 6.4'de gösterilen ekran ile şemada hangi tabloların yer aldığı bilgisi girilir.



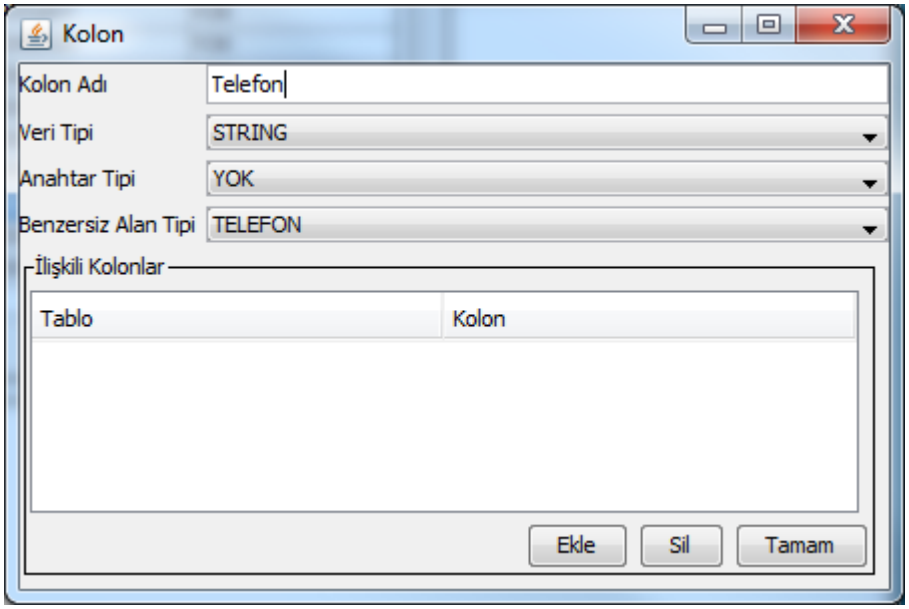
Şekil 6.4. Uygulamada veri tabanı tabloları listesi ekranı

Daha sonra Şekil 6.5'deki ekran ile her bir tablo için o tabloda hangi alanların olduğu tanımlanır.



Şekil 6.5. Uygulamada veri tabanı tablosunun alanları listesi ekranı

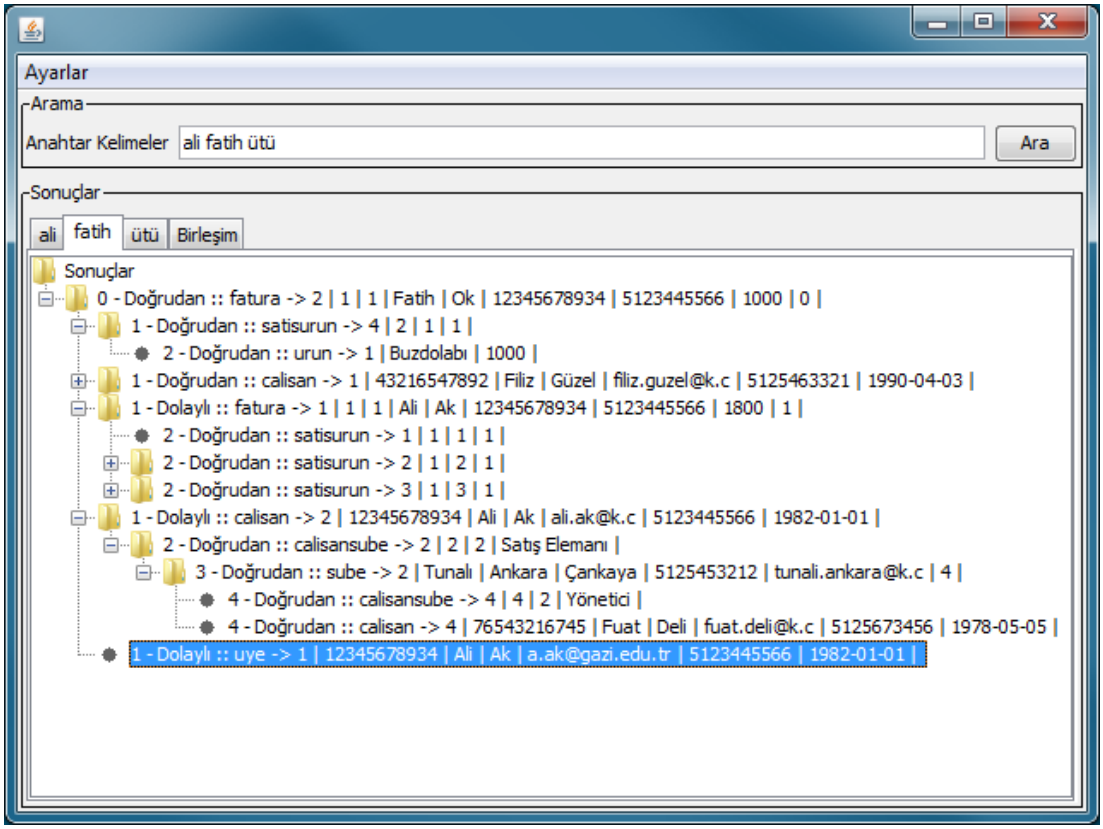
Alan tanımlama işlemi sırasında alanın adı, veri tipi, anahtar tipi Şekil 6.6'daki ekran ile tanımlanır.



Şekil 6.6. Uygulamada veri tabanı tablo alanlarının tanımlanması ekranı

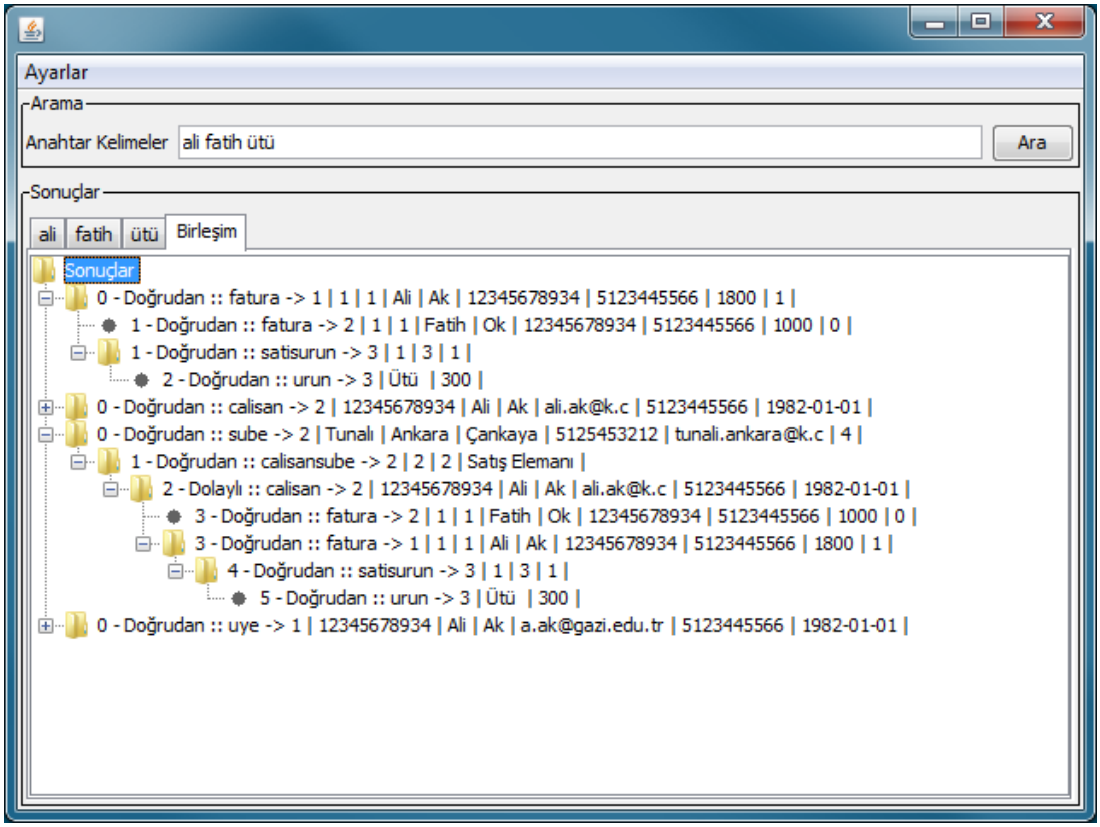
6.4.2. Anahtar kelime aratma

Kullanıcı aratmak istediği kelimeleri aralarında boşluk bırakarak “Anahtar Kelimeler” alanına girer. Uygulama her bir anahtar kelimeyi önce tek tek veri tabanında arar ve sonuçları ayrı ağaçlar altında kullanıcıya gösterir. Daha sonra bu sonuçlar birleştirerek kullanıcıya sunulur. Kullanıcının anahtar kelimeleri girdiği ve her bir anahtar kelime için elde ettiği sonuçları görüntülediği uygulama ekranı Şekil 6.7’de gösterildiği gibidir.



Şekil 6.7. Girilen anahtar kelimelerden biri için uygulama sonuç ekranı

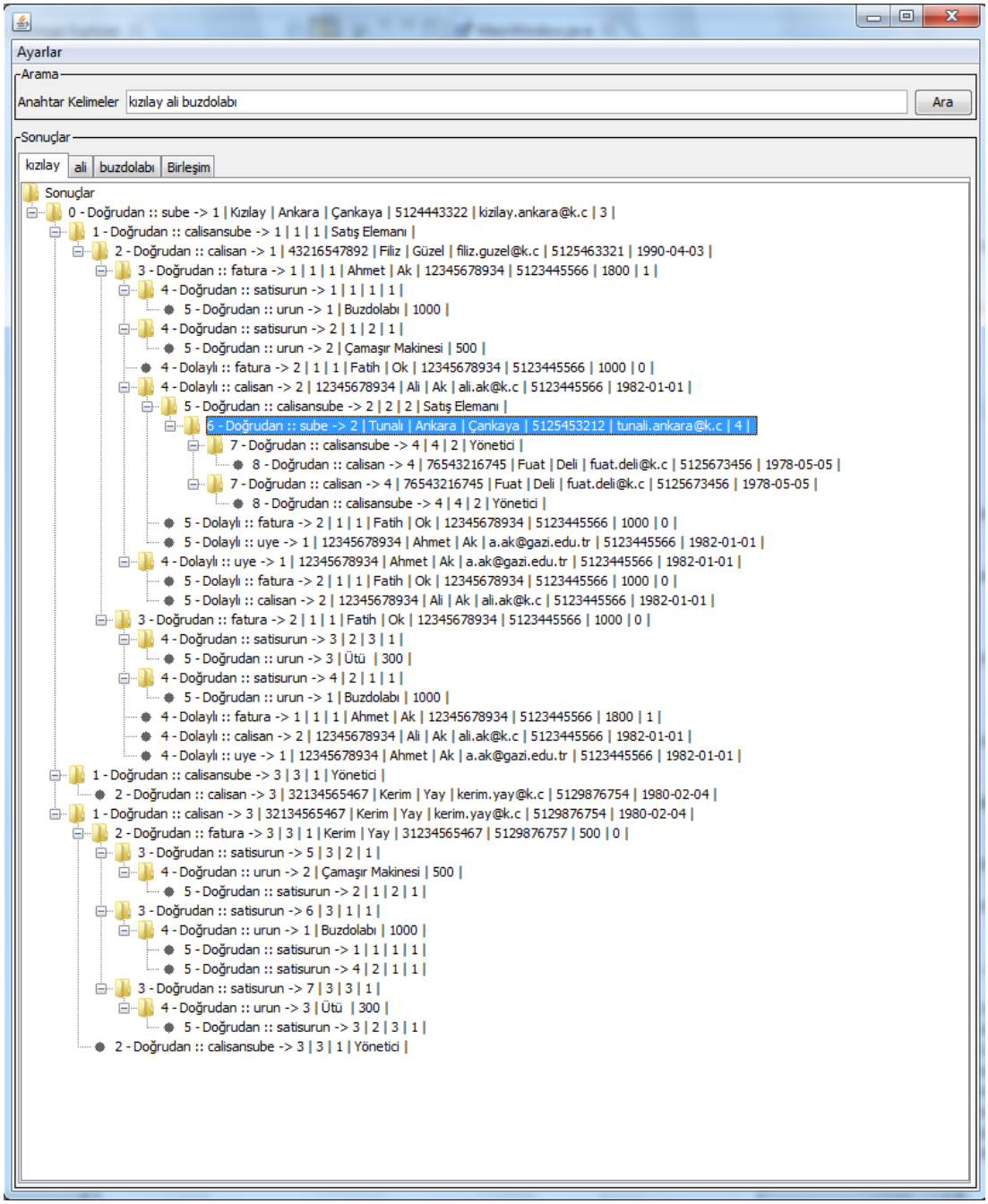
Kullanıcının girdiği iki veya daha fazla anahtar kelimeyi eçeren sonuçlar uygulamanın Şekil 6.8'deki birleşim ekranında görüntülenmektedir.



Şekil 6.8. Girilen tüm anahtar kelimeler için birleşim uygulama sonuç ekranı

6.5. Test Senaryosu

Önerilen metodun faydalarının anlaşılması için “*kızılay*”, “*ali*” ve “*buzdolabı*” anahtar kelimeleri kullanılarak bir deney senaryosu oluşturulmuştur. Bu senaryoda, kullanıcı "kızılay" şubesinde "ali" adındaki bir kişiye satılan ya da "ali" adılı bir kişi tarafından satılan bir "buzdolabı" olup olmadığını öğrenmek istemektedir. Bu amaçla uygulama öncelikle girilen anahtar kelimeleri veri tabanı içerisinde bağımsız olarak aramakta daha sonra bulduğu sonuçları birleştirmektedir.



Şekil 6.9. "*Kızılay*" anahtar kelimesi için uygulama sonuç ekranı

Şekil 6.9'da "*kızılay*" anahtar kelimesi için bulunan sonuçlar görülmektedir. Sonuçlar incelendiğinde uygulama "*kızılay*" anahtar kelimesinin geçtiği tek tablo ve kayıt olan Sube tablosundaki kayda ulaşmıştır. Bu kayıt R kümesinin elemanı olan R_0'ı ifade etmektedir. Bu durumda R_0 kümesi de $R_0 = \{r_{01}\}$ şeklinde tek elemanlıdır. Veri tabanında tanımlı dış anahtar → birincil anahtar

ilişkilerinin ve veri tabanı yöneticisi tarafından tanımlanmış olan ilişkiler kullanılarak Şekil 6.9'da görülen T_{01} ağaç yapısı oluşmaktadır. T_{01} ağaç yapısında görüldüğü gibi Sube tablosunda bulunan "*kızılay*" anahtar kelimesi dış anahtar →birincil anahtar ilişkisinden birinci seviyede 3 farklı kayda ulaşmıştır.

İlk bulunan Sube tablosundaki kayıttan Calısansube tablosundaki "*Satış Elemanı*" kaydına bu kayıttan da Calısan tablosundaki "*Filiz Güzel*" kaydına ulaşılmıştır. Bir sonraki ilişkide ise yani 3. seviyede 2 farklı kayda ulaşılmıştır. Bu kayıtlar Fatura tablosunun "*Ahmet Ak*" ve "*Fatih Ok*" kayıtlarıdır. "*Ahmet Ak*" kaydına bakıldığında bu kayda ait 4. seviyeden 2'si doğrudan 3'ü dolaylı ilişki olarak tanımlanan 5 kayıt mevcuttur.

2 doğrudan ilişki SatısUrun tablosu ile kurulmuştur ve bu iki kayıt da Urun tablosundaki "*buzdolabı*" ve "*çamaşır makinesi*" kayıtları ile doğrudan ilişkilidir. Bu aşamaya kadar olan doğrudan ilişkiler değerlendirildiğinde Filiz Güzel'in Kızılay şubesinde çalışan bir satış elemanı olduğu; buzdolabı ve çamaşır makinesi alan Ahmet Ak adlı kişinin faturasını düzenlediği bilgisine ulaşılır. Doğrudan ilişkilerin kullanılması ile CalısanSube tablosundan sadece bu bilgilere ulaşılabilir.

Fakat 4. seviyede oluşmuş 3 dolaylı ilişkiden daha fazla bilgiye ulaşılabildiği görülür. 4. seviyede oluşmuş ilk dolaylı ilişki Fatura tablosundaki "*Fatih Ok*" kaydıdır. Bu kayıt ile ilişkili kayıtlar aynı ağaç içerisinde daha önceden bulunduğu için ağaç devam etmemiştir. Bu dolaylı ilişki *tckimlikno* ve *telefon* alanları üzerinden kurulmuştur. Dolaylı ilişki "*Fatih Ok*" ve "*Ahmet Ak*" kayıtları arasındaki ilişkiyi ortaya çıkarmıştır.

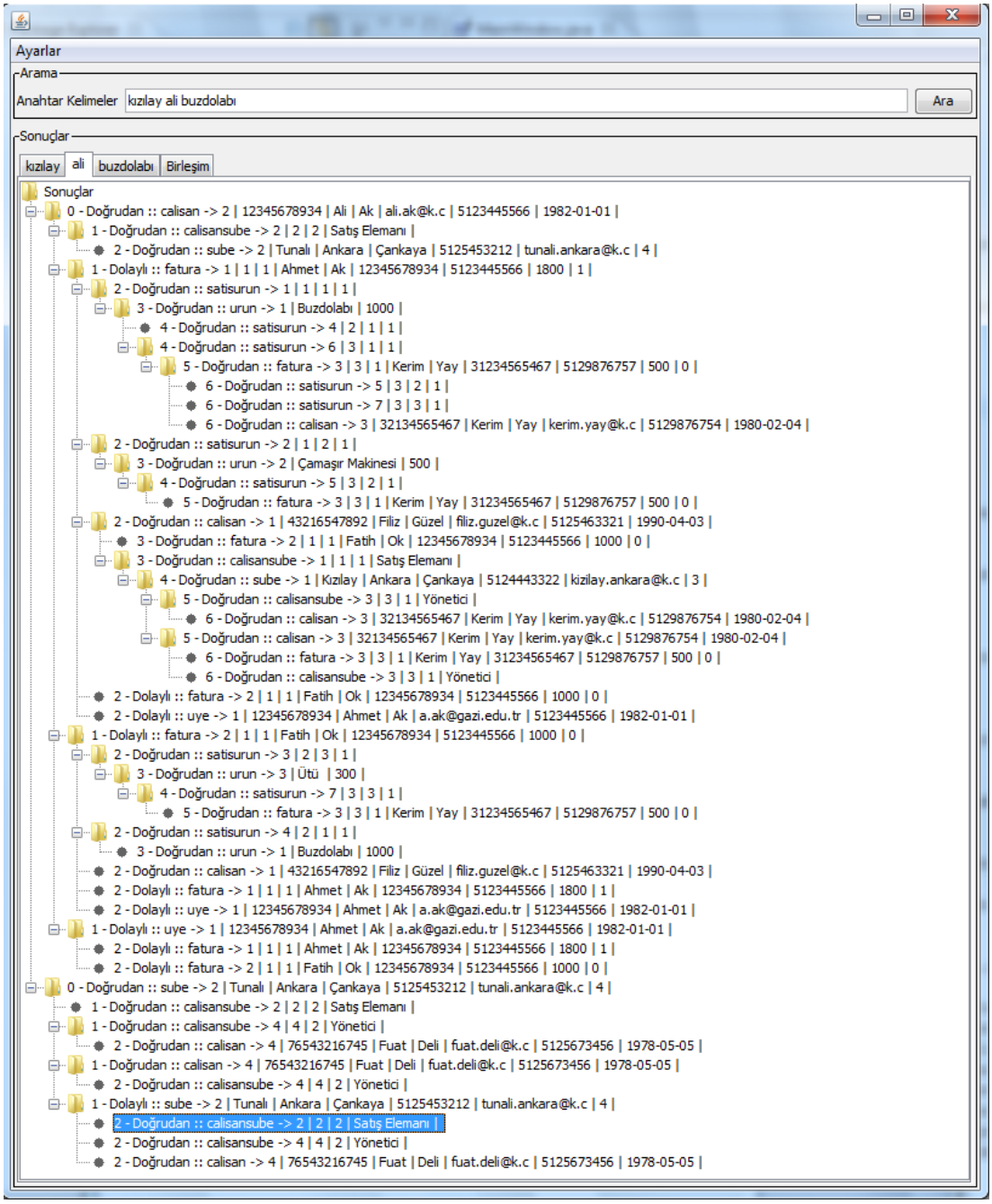
İkinci dolaylı ilişki Calısan tablosundaki "*Ali Ak*" kaydı ile gerçekleşmiştir. Kurulan bu dolaylı ilişki *tckimlikno* ve *telefon* alanları üzerinden kurulmuştur. Dolaylı ilişki ile ulaşılan bu kayıt doğrudan ilişki olarak tanımlanan veri tabanı dış anahtar →birincil anahtar ilişkileri kullanılarak CalisanSube ve Sube

tablolarına erişilmiştir. Dolaylı ilişkinin kullanılması "*Ahmet Ak*" kaydındaki *tckimlikno* ve *telefon* alanlarındaki bilginin Ali Ak'a ait olduğu ve Ali Ak'ın da Tunalı şubesinde satış elemanı olduğu bilgisini ortaya çıkarmıştır. Bu kayıt üzerinden ilerlendiğinde tekrar CalisanSube ve Calisan tablolarına erişilmiş ve Tunalı şubesinin yöneticisi bilgisine erişimi sağlamıştır.

3. dolaylı ilişki ise aynı *telefon* ve *tckimlikno* alan değerlerini içeren bir fatura kaydının daha olduğunu "*Fatih Ok*" kaydını ortaya çıkarmıştır.

Dolaylı ilişkilerin kullanılması ile arama sonucu genişletilmiş, doğrudan ilişkiden ulaşılamayacak sonuçlar elde edilmiştir. Eğer dolaylı ilişki kullanılmasaydı 4. seviyede sadece 2 ilişki elde edilecekti ve dolaylı ilişkilerin çıkarmış olduğu sonuçlar elde edilemeyecekti. Dolaylı ilişkinin de kullanılması ile 4. seviye ilişki sayısı 5 olmuştur.

Aynı şekilde bütün seviyeler veri tabanında tanımlı dış anahtar → birincil anahtar ilişkisi ve veri tabanı yöneticisi tarafından tanımlanmış olan ilişkiler kullanılarak en uç yaprağa kadar gider.

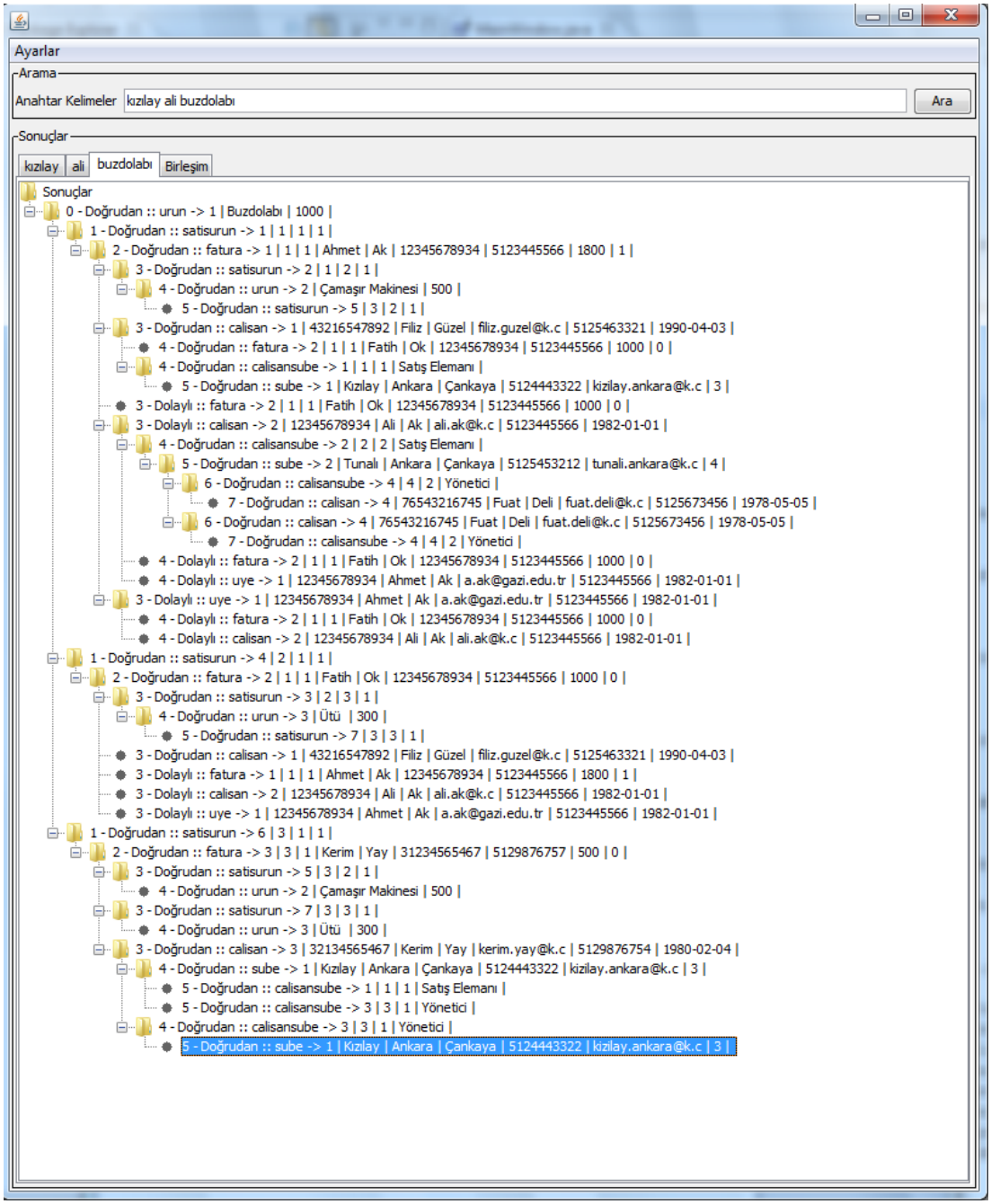


Şekil 6.10. "*Ali*" anahtar kelimesi için uygulama sonuç ekranı

"*Ali*" anahtar kelimesi için veri tabanında gerçekleştirilen arama sonucu bulunan kayıtlar Şekil 6.10'da gösterildiği gibidir. Görüldüğü gibi daha ilk seviyelerde dolaylı ilişki aramayı genişletmiştir. "*Ali*" anahtar kelimesi Calısan ve Sube tablolarının birer kaydında bulunmuş ve Calısan tablosundaki kayıt ile ilişkili diğer kayıtlara ulaşılmak istenildiğinde 1. seviyeden 1 doğrudan 3

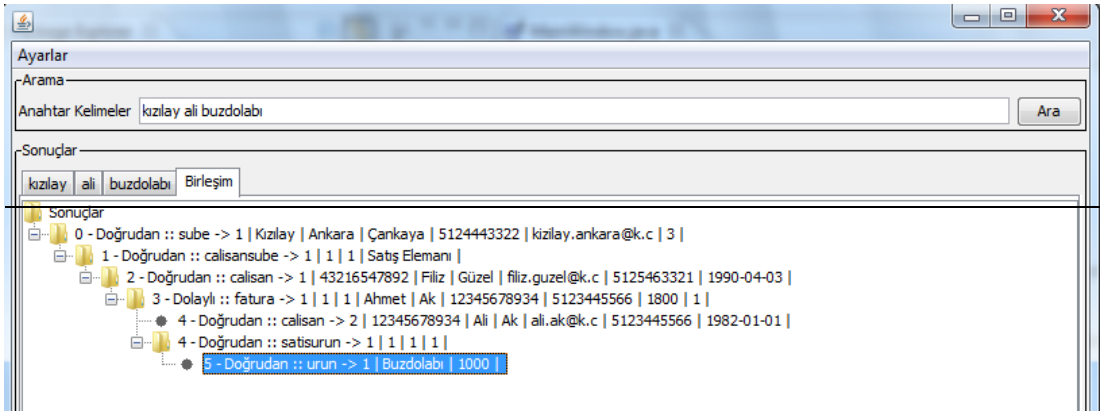
dolaylı ilişkiye ulaşılmıştır yine aynı şekilde Sube tablosundaki kayıt ile ilişkili diğer kayıtlara ulaşılmak istenildiğinde de 1. seviyeden 2 doğrudan 1 dolaylı ilişkiye erişilmiştir.

Eğer "*Ali*" anahtar kelimesini aramada dolaylı ilişki kullanılmamış olsa idi müşteri "Fatih"in ve üye "Ahmet"in çalışan "Ali" ile olan ilişkisi ortaya çıkmayacak ve Calısan tablosundan ulaşılan 1. seviye ilişki sayısı 4 değil 1 olacak, Sube tablosundan ulaşılan 1. seviye ilişki sayısı da 3 değil 2 olacaktı.



Şekil 6.11. "*Buzdolabı*" anahtar kelimesi için uygulama sonuç ekranı

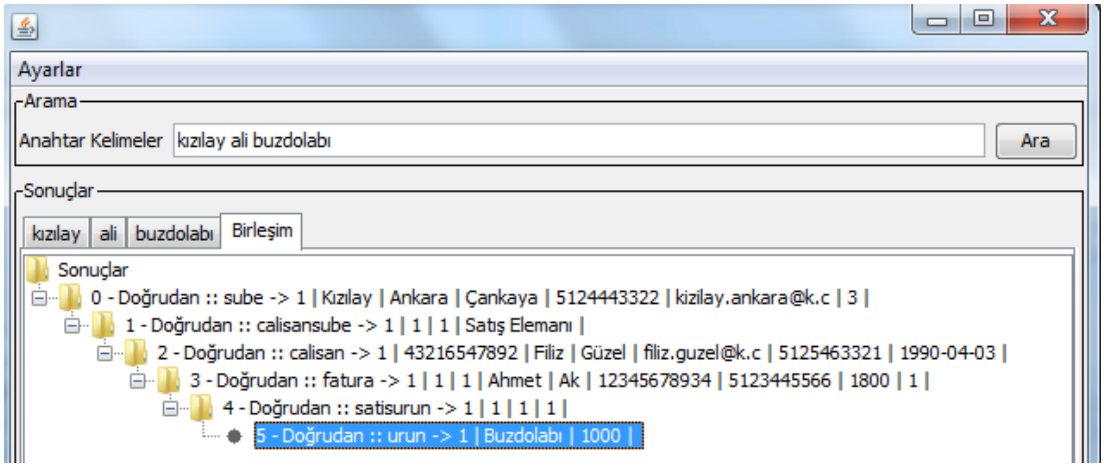
Şekil 6.11 de "*buzdolabı*" anahtar kelimesi için bulunan sonuçları göstermektedir. Şekil 6.11'de görüldüğü gibi arama sonucunda kurulan ilişkilerde dolaylı ve doğrudan ilişkiler mevcuttur. Şekil 6.11'de görüldüğü gibi dolaylı ilişkinin kullanılması bulunan ilişkili kayıt sayısını artırmış bu sayede arama genişletilmiştir.



Şekil 6.12. "*Kızılay*", "*ali*", "*buzdolabı*" anahtar kelimeleri için uygulama sonuç ekranı

Şekil 6.12 kullanıcı tarafından girilmiş olan "*kızılay*", "*ali*" ve "*buzdolabı*" anahtar kelimeleri için ortak arama sonucunu göstermektedir. Şekil 6.12'de de görüldüğü gibi sonuç ekranını doğrudan ve dolaylı ilişkiler oluşturmaktadır. Bu ekranda gösterilen doğrudan ve dolaylı ilişkiler her bir anahtar kelimenin aranması ile elde edilen ilişkileri ifade etmektedir. Sonuçlardan da anlaşılacağı gibi gösterilen sonuçlar 3 anahtar kelime için de ortak kayıtlardır.

Sonuç kayıtları değerlendirildiğinde Sube tablosunun "*kızılay*" kaydının kök yaprağı oluşturduğu görülmektedir. Bu kayıt üzerinden sırası ile CalisanSube, Calisan ve Fatura tablolarına erişilen kayıtlar mevcuttur. Bu ilişkilerden CalisanSube ve Calisan tablolarına doğrudan ilişki ile ulaşılmışken Fatura tablosuna dolaylı ilişki ile ulaşılmıştır.



Şekil 6.13. "Kızılay", "ali", "buzdolabı" anahtar kelimeleri için dolaylı ilişki kullanılmadan bulunan uygulama sonuç ekranı

Eğer dolaylı ilişki kullanılmamış olsaydı Şekil 6.13'deki sonuçlar elde edilecekti. Elde edilen sonuçlar incelendiğinde dolaylı ilişkinin de kullanılması ile elde edilen Şekil 6.12'deki sonuç ekranındaki Calısan tablosundaki "Ali Ak" kaydı mevcut değildir. Çünkü Calisan tablosundaki "Ali Ak" kaydı ile Fatura tablosundaki "Ahmet Ak" kaydının ilişkisi benzersiz alan olarak tanımlanan TcKimlikNo alanı sayesinde ortaya çıkmıştır. Bu nedenle dolaylı ilişkinin kullanılmaması "Ali Ak" kaydı ile "Ahmet Ak" kaydı arasındaki ilişkiyi ortaya çıkaramamıştır. Elde edilen sonuç ekranı kullanıcının ilgilendiği bir sonuç olmasına rağmen eksik bilgi vermektedir. Dolaylı ilişkinin kullanılması bu eksikliği ortadan kaldırmıştır.

7. SONUÇ

Tüm çalışma süreci boyunca birçok ilişkisel veri tabanlarında anahtar kelime arama yöntemi ayrıntılı olarak incelenmiştir. Literatürdeki çalışmalardan faydalanmak için kullanılan yöntemlerin çalışma mantıkları, oluşturulmuş indeks tabloları incelenmiştir.

İlişkisel veri tabanlarında anahtar kelime arama uygulamalarının başarısı literatür çalışmaları bölümünde ayrıntılı olarak verilmiştir. Literatürdeki çalışmalar hedeflediği amaca ulaşmış fakat kullanılan yöntemler avantaj ve dezavantajlara sahiptirler.

Çalışmaların bazıları kendi ilişkisel veri tabanlarında kendi indeks tablolarını oluştururken bazıları veri tabanının indeks tablolarını kullanmışlardır. Kendi indeks tablolarını kullanmasında tüm kelimeler için indeks oluşturulmasından dolayı aramanın daha hızlı olması gibi bir avantajı varken oluşturulmuş olan indeks tablolarının güncel tutulma ihtiyacı bir dezavantaj olarak değerlendirilmektedir.

Çalışmaların bazılarında ise veri tabanının kendi indeks tabloları kullanılmıştır. Bu yöntemin kullanılması tüm alanlarda indeks oluşturulmuş olsa bile tüm kelimeler üzerinde bir indeks tanımlanmamış olması aramayı yavaşlatmıştır. Bunun yanında veri tabanın indeks tablolarının kullanılması indeks tablolarının güncelliğinin kontrol edilmesi gibi bir ihtiyacı ortadan kaldırmıştır. Bu da anahtar kelime arama uygulamaları için bir avantaj oluşturmaktadır.

Önerilen yöntemde arama hızı bir kriter olarak düşünülmediğinden uygulama anahtar kelime aramada veri tabanı indeks tablolarını kullanmıştır.

Çalışmalarda kullanılan dış anahtar → birincil anahtar ilişkisi veri tabanındaki ilişkilerin çoğunu ortaya koyduğundan başka bir ilişki tanımlanma ihtiyacı

duyulmamıştır. Fakat bazı veri tabanlarında sadece dış anahtar → birincil anahtar ilişkisi ilişkilerin tamamını ortaya çıkaramamakta ve aynı zamanda kayıtlar arası ilişkiyi eksik bırakmaktadır. Bu tip veri tabanlarında diğer alanlar üzerinden ek ilişkilerin tanımlanması ilişkisel veri tabanlarında anahtar kelime arama kabiliyetini arttırmaktadır.

İlişkisel veri tabanlarında anahtar kelime arama çalışmaları veri tabanına bağımlılık gerektirdiğinden diğer veri tabanı yapılarından farklı yapıya sahip veri tabanları için farklı uygulamaların gerçekleştirilmesi gerekecektir.

KAYNAKLAR

1. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S., "Keyword searching and browsing in databases using BANKS", ***18th International Conference on Data Engineering***, San Jose, 431 - 440 (2002).

2. Kacholia, V., Pandit, S., Chakrabarti S., Sudarshan, S., Desai, R., Karambelkar, H., "Bidirectional Expansion For Keyword Search on Graph Databases", ***Very Large Data Bases***, 505-516 (2005).

3. He, H., Wang, H., Yang, J., Yu, P.S., "BLINKS: Ranked keyword searches on graphs", ***International Conference on Management of Data***,New York, 305 – 316 (2007).

4. Hristidis, V., Papakonstantinou, Y., "DISCOVERY: Keyword search in relational Databases", ***Very Large Databases***, 670 – 681 (2002).

5. Liu, F., Yu, C., Meng, W., Chowdhury, A. "Effective keyword search in relational databases", ***International Conference on Management of Data***, 563 (2006)

6. Qin, L., Yu, J.X., Chang, L., Tao, Y., "Querying communities in relational databases", ***International Conference on Data Engineering***, Shanghai, 724 – 735 (2009).

7. İnternet : www.computer.org, "RSearch: Enhancing keyword Search in relational databases using nearly duplicate records". http://sites.computer.org/debull/A10mar/yuge-paper.pdf (2010)

8. Agraval, S., Chaudhuri, S., Das, G., "DBXplorer: A system for keywod – based search over relational databases", ***International Conference on Data Engineering***, San Jose, 5 (2002).

9. Wang, S., Zhang, K.L., "Searching databases with keywords", ***Journal of Computer Science and Technology***, 20(1):55 - 62 (2005).

10. Park, J., Lee, S., "Keyword search in relational databases", ***Knowledge and Information Systems***, 26(2):175 - 193 (2011).

11. Hulgeri A., Bhalotia G., Nakhe C., Chakrabarti S., Sudarshan S., "Keyword Search in Databases", ***IEEE Data Engineering Bulletin***, (2001).

12. Haam, D., Lee, K.Y., Kim, M.H., "Keyword search on relational databases using keyword query interpretation", ***5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)***, Seoul, 957 – 967 (2010).

13. Wang, W., Lin, X.,Luo, Y. “Keyword Search on Relational Databases”, ***IFIP International Conference on Network and Parallel Computing Workshops***, Liaoning, 7 - 10 (2007).

14. Balmin, A., Hristidis, V., Papakonstantinou, Y., “ObjectRank: Authority-based keyword search in databases”, ***Very Large Data Bases***, 564 - 575 (2004).

15. Dalvi, B.B., Kshirsagar, M., Sudarshan, S., “Keyword search on external memory data graphs”, ***Proceedings of the Very Large Data Bases Endowment 1***, 1189 - 1204 (2008).

16. Goldman, R., Shivakumar, N., “Proximity Search in Databases”, ***Proceeding of the 24th Very Latge Data Bases Conference***, New York, 26 – 37 (1998).

17. Bahmani, A.H., Naghibzadeh, M., Bahmani, B., “Automatic database normalization and primary key generation”, ***IEEE Canadian Conference on Electrical and Computer Engineering***, Canadian, 11 - 16 (2008).

18. Russell, J., “Oracle8i: Application Developer’s Guide – Fundamentals Release 2”, **Oracle Corporation,** USA, 5.1 – 20 (1999).

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: DEMİRCİOĞLU, Serap
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 29.06.1982, Denizli
Medeni hali	: Evli
Telefon	: 0 (312) 411 2127
e-mail	: karadag.serap@hotmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Kocaeli Üniversitesi/Bilgisayar Mühendisliği	2004
Lise	Denizli Anafartalar Lisesi	2000

İş Deneyimi

Yıl	Yer	Görev
2007-Halen	Kara Kuvvetleri Komutanlığı	Bilgisayar Mühendisi
2005-2007	Turkuaz Tekstil	Yazılım Uzmanı

Yabancı Dil

İngilizce