T.C.
ANKARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
KÜTÜPHANECİLİK ANABİLİM DALI

VERİTABANI KURAMI VE BİR KÜTÜPHANE UYGULAMASI

Doktoro Tezi

Tülay FENERCİ

Tez Danışmanı

Prof. Dr. Mustafa AKBULUT

Ankara, 1995

A.Ü. SOSYAL BİLİMLER ENSTİTÜSÜ

DOKTORA TEZ SAVUNMA TUTANAĞI

A.Ü. Sosyal Bilimler Enstitüsü ...KÜTÜPHANECİLİK.....
Anabilim Dalı yüksek lisans DOKTORA öğrencisi ..TÜLAY FENERCİ..... in
" VERİTABANI KURAMI VE BİR KÜTÜPHANE UYGULAMASI
....." başlıklı tezini değerlendirmek üzere görevlendirilen jürimiz, Prof.Dr.Necmeddin Sefercioğlu..... başkanlığında, ...ÇARŞAMBA..... günü saat ....15:30.....'da A.Ü. DTL VE TARİH-COĞRAFYA..... Fakültesi'nde toplandı.

Tezin;

a) PEKİYİ..... derece ile başarılı sayılmasına,
b) düzeltilmek üzere geri verilmesine,
c) reddine

Oybirliği/Oyçokluğu ile karar verildi.

Üye
Prof.Dr. Necmeddin Sefercioğlu

Üye
Prof.Dr. Mustafa Akbulut

Üye
Prof.Dr. Bengü Çapar

ÖNSÖZ

Günümüzde nitelik bireyler açısından bilgi sahibi olma ve bilgiye erişim, kuruluşlar açısından da bilgiyi üretim, hizmet ve pazarlama alanlarında kullanabilme becerisi ile ölçülmektedir. Bilginin bu bağlamda gösterdiği yükseliş bilgisayar teknolojisinin yazılım ve donanım alanındaki gelişimi ve getirdiği sistem yaklaşımının etkisiyle olmuştur. Sistem yaklaşımı kuruluşları açık birer sistem olarak ele alındığından bu noktada girdi, çıktı, işlem süreci ve bu süreç içinde bilginin ekonomik bir değer olarak değerlendirilmesi gündeme gelmiştir. Bilginin para, hammadde ve insangücü gibi bir kaynak olarak ele alınması ve bu yönde gelişen bilinçlenme bilginin organizasyonu ve yönetimi konusunda çalışmaların yoğunluk kazanmasını sağlamıştır. Bu noktada ön plana çıkan bilgi uzmanlarının, bilginin verimli bir biçimde yönetimi için öncelikle iyi bir tasarım becerisine sahip olmaları gerekliliği doğmuştur. İşte bu nedenle tezin ana temasını verimli tasarım yöntemi olan NIAM oluşturmaktadır. Çünkü inancımız gelecekte bilgi hizmeti veren kuruluşlar kadar iş dünyasında da böylesi becerilerle donatılmış bilgi uzmanlarının yönlendirici bir görevi üstleneceğidir.

Çalışmanın başlangıcından sonuçlandırılmasına değin geçen uzun ve zorlu zaman diliminde bana gösterdiği sabır ve verdiği destekden ötürü eşime ve aileme, önemli katkılarından dolayı hocam Prof. Dr. Mustafa Akbulut'a ve değerli Bölüm öğretim üyelerine, yardımlarını ve dostluklarını esirgemeyen sevgili mesai arkadaşlarıma sonsuz teşekkürlerimi sunmayı bir borç bilirim.

Tülay FENERCİ

İÇİNDEKİLER

Sayfa

TABLO LİSTESİ

ŞEKİL LİSTESİ

I. BÖLÜM

1.1 GİRİŞ

Bilginin organize edilmesi ve bilgiye erişim stratejilerinin belirlenmesi konusunda uzun yıllar geleneksel yöntemlerle çalışan bilgi uzmanları teknolojinin bilgi merkezlerine girmesiyle yeni teknolojiyi tanıma, anlama ve uygulamaya koyma zorunluluğu ile karşı karşıya kalmışlardır. Bu durum onlar için bir zorunluluk olarak ortaya çıkmıştır. Çünkü yeni teknoloji hız, doğruluk ve kapsamlı bilgiye erişim gibi geniş olanaklar sunmaktadır. Söz konusu olanaklar hizmetin verimliliği için büyük önem taşımaktadır. Ancak bilginin organizasyon dünyasında etkinliğinin artması veri yığınlarının verimli bir biçimde uygulamalara yönelik olarak düzenlenmesini gerekli kılmış ve bunun doğal sonucu olarak bilgi uzmanlarının etkinlik alanı genişlemiştir. Bilgi uzmanları özellikle enformasyon sistemlerinin tasarımında büyük bir güç olmak durumundadır. Bu görevi üstlenebilmek için de bilginin organizaşyonu ve bilgiye erişim stratejileri konularındaki deneyimlerini enformasyon sistemi tasarımı alanına aktaracak altyapıya sahip olmaları gerekmektedir. Bunun sağlanması da verilerin bir model uyarınca yapılandırılması, onlara erişimi somut olarak gerçekleştiren veritabanı yönetim sistemlerinin yapısı ve işleyişi hakkında bilgi sahibi olmalarına bağlıdır.

1.2 AMAÇ VE VARSAYIM

Veritabanı yönetim sistemi oluşturulacak enformasyon sisteminin vazgeçilmez, temel bir öğesidir. Kapsamlı ve karmaşık bir yazılım olan veritabanı yönetim sistemi ilişkilendirilerek yüklenen verileri işlemek ve çeşitli bileşimlerini oluşturmak suretiyle bilgi üretir. Son yıllarda veritabanı sistemleri konusunda yapılan çalışmalar ve üretim ilişkisel modele dayalı sistemler üzerinde yoğunlaşmıştır.

Amacımız veritabanı sistemini yapı ve işleyişi açısından inceleyerek kapasitesi hakkında bilgi vermek ve sistemin desteklediği modelleri inceleyerek verimliliği yakalamak bakımından bunların birbirlerine olan üstünlüklerini ortaya koymaktadır. Böylelikle bu alandaki eğilimin neden ilişkisel model yönünde olduğu ortaya çıkacaktır.

Bunun yanında ilişkisel modeli oluşturmada kullanılan ve uygulamaya aktarımı diğer yöntemlere oranla son derece basit olan NIAM tekniğini bir kütüphane işlemi aracılığıyla tanıtmak ve incelemektir.

Varsayım ise; "şimdiye kadar veritabanı yönetim sistemlerinin bilgi merkezleri açısından sahip oldukları kaynakların niteliği gözönünde bulundurularak hep bibliyografik verinin depolanması ve bunlara erişim işlevini yerine getirdiği, oysa ilişkisel modelin yalnız bibliyografik denetim değil bilgi merkezinin tüm işlemleri için kullanılabilen, esnek bir model olduğu ve bunun doğruluğunun kullanılan NIAM tekniği ile açıkça gösterileceği" yolundadır.

1.3 KAPSAM VE DÜZEN

"Veritabanı kuramı ve bir kütüphane uygulaması" başlıklı çalışma iki aşamalı olarak ele alınmıştır. Birinci aşamada veritabanı yönetim sisteminin yapısı, işleyişi, gelişimi ve bu gelişim doğrultusunda ortaya çıkan modeller değerlendirilmiştir. Modellerin seçiminde ise bunların gelişim çizgisi yönünde piyasadaki ticari ürünlerde en çok kullanılan örnekleri bağlamında bir sınırlama yapılmıştır.

İkinci aşama enformasyon sistemi tasarımı konusunda en çok benimsenen ilişkisel modele uygun tasarım tekniklerinden yine en yaygın biçimde kullanılanları seçilmiştir. Benimsenen tasarım tekniğinin kütüphane işlemlerine uyarlanmasında ise kapsam, üniversite kütüphanelerindeki sipariş işlemi ile sınırlandırılmıştır. İlgili Bölüm'de ayrıntılarıyla belirtilen sınırlama nedeni ise, üniversite kütüphanelerinin ülkemizde bilgisayar uygulamalarına geçişte öncü bir rol üstlenmiş olmalarıdır. Bunun yanında sipariş işlemi tekdüze niteliği ile küçük ayrıntılar dışında bütün bu kuruluşlar genelinde benzer sürece sahip bir işlemdir.

Çalışma VI bölümden oluşmaktadır.

I. Bölüm; giriş bölümü olup bu bölümde amaç, kapsam ve düzen, yöntem ve terimler ele alınmıştır.

II. Bölüm'de veritabanı gelişimi terim ve tarihçe açısından ele alınmıştır. Ayrıca veritabanının öğeleri, işleyişi ve kuruluşlar içerisinde uygulamaya konma aşamaları incelenmiştir.

III. Bölüm veritabanı yönetim sistemlerindeki düzeyleri, mantıksal yapıyı oluşturan modelleri ve veritabanından bilgi sağlama sürecini gerçekleştiren dillerden SQL'i tanıtmaktadır.

IV. Bölüm'de ise ilişkisel modele uygun sistem tasarımını gerçekleştirmede kullanılan geleneksel bir yöntem olan normalizasyon ve yeni bir teknik olan NIAM incelenmiştir. Söz konusu tekniğin kullanılması ile oluşturulan sipariş işlemine yönelik tasarım şemaları ise ekler kısmında verilmiştir.

V. Bölüm'de, veritabanı sistemlerinde verilerin ikincil belleklerde düzenlenmesi ve verilere en düşük maliyet ve hızla erişimin gerçekleştirilmesindeki alternatifler sunulmuştur.

VI. Bölüm ise sonuç ve önerilerden oluşmaktadır.

1.4 YÖNTEM

Araştırmanın ilk aşamasında araştırma konusu ile ilgili "varolan belgelerin saptanması ve incelenerek gerekli verilerin toplanmasında"[1] **belgesel tarama yöntemi kullanılmıştır.** Daha sonra çalışmanın ana bölümlerinin oluşturulmasında "....olayların, objelerin, varlıkların, kurumların, grupların ve çeşitli alanların ne olduğunu belirlemeye, açıklamaya çalışan"[2] **betimleme yöntemin**'den yararlanılmıştır. Ayrıca çalışmanın 4.2. Bölüm'ünde tanıtılan NIAM tekniği kullanılarak üniversite kütüphanesindeki sipariş işlemi için kavramsal

1. Niyazi Karasar, Bilimsel Araştırma Yöntemi, (Ankara: Bilim Kitabevi, 1986), 36 s.
2. Saim Kaptan, Bilimsel Araştırma ve İstatistik Teknikleri. (Ankara: (yayl. yok), 1993), 59 s.

şema oluşturulmuş ve bu şemanın ilişkisel şemaya dönüştürülmesi gerçekleştirilmiştir. Şemanın gerçekleştirilebilmesi için öncelikle Bilkent Üniversitesi, Gazi Üniversitesi, Orta Doğu Teknik Üniversitesi kütüphaneleri ve Hacettepe Merkez Tıp Kütüphanesi'nde kullanılan sipariş formları ve sipariş işlemindeki süreçler incelenmiştir. İşlemin aşamalarında gereksinim duyulan girdi/çıktı bilginin ne olacağı yani tasarlanan sistemde işlemin her aşamasında sistem-kullanıcı diyaloğunun nasıl gerçekleşeceği planlanmıştır. Girdi formlar yukarıda anılan kütüphanelerde yürütülmekte olan işlemin tüm özelliklerini yansıtan ortak bir biçim olarak tasarlanmıştır. Bu formlar doğrultusunda belirlenen varlık tipleri, onların özellikleri ve aralarındaki ilişkiler NIAM tekniğinin aşamaları uyarınca kavramsal şemaya dönüştürülmüş ve bu sürecin sonunda yine Nijssen'in önerdiği optimum normal biçim uyarınca ilişkisel tablolar ortaya çıkarılmıştır.

1.5 TERMİNOLOJİ

Çalışmada, çoğunluğu yabancı dilde veya yabancı dilden türkçeye çevrilmiş olan kaynaklar kullanılmıştır. Terimlerin karşılıkları aranırken türkçe kaynaklar (belge/yayın ve sözlük)daki kullanımları da dikkate alınmıştır. Ancak kimi terimlerin karşılıklarında türkçe kaynaklarda da bir birlik olmadığı görülmüştür. Böylesi durumlarda yaygın kullanımı olan karşılıkların veya araştırmacının önerdiği karşılıkların kullanılması yoluna gidilmiştir. Araştırmacının bölüm öğretim üyeleri ve bilgisayar uzmanları ile yaptığı görüşmelere dayanarak önerdiği türkçe karşılıklar şöyle sıralanabilir:

Değer (Value/Label): Veritabanını kuramsal açıdan inceleyen kaynaklarda "değer" veri değeri olan "data value" karşılığında kullanılmaktadır. Bununla beraber Nijssen eserinde "değer" için değişik bir anlam katmadığı halde "label" sözcüğünü kullanmaktadır."Label"ın sözlük anlamı etiket veya nitelendirici isim'dir. Gerek sözlük anlamı gerekse Nijssen'in yüklediği anlam veritabanı açısından bir farklılık göstermediğinden terim birliği sağlamak amacıyla çalışmada "label" karşılığı olarak "değer" sözcüğü kullanılmıştır.

Hazırlıksız Sorular (Ad hoc Queries): SQL gibi dördüncü nesil dillerde, soruların yöntemsel dillerden farklı olarak anında oluşturulup sisteme yöneltilmesidir. Yöntemsel dillerin kullanıldığı veritabanı sistemlerinde veri bağımsızlığı bulunmadığından veriye erişim için verinin yerini bilme zorunluluğu vardır. Yöntemsel dil veriye nasıl ulaşılacağının belirlendiği bir süreci kullanır. Oysa veri bağımsızlığının sağlandığı ANSI/SPARC mimari yapısına sahip veritabanlarında "neyin nasıl yapılacağı" yerine "nelerin istendiği" önem taşır. Bu durumda verinin yerini bilme zorunluluğu olmadığı için soruların yapılandırılmasmında bu yönde bir sınırlılık yoktur.O nedenle bu tür sorular için "hazırlıksız sorular" karşılığı önerilmiştir.

Niteleme Öğesi (Reference Scheme): Bu terime hiçbir türkçe belge ya da terim sözlüğünde rastlanmamıştır. Önce sözcüğü oluşturan öğeler tek tek ele alınarak anlamları irdelenmiştir. Buna göre sözlüklerde "reference" için yönlendirme, gönderme,"scheme" içinse düzen, plan; sınıflandırma cetveli anlamları verildiği görülmüştür. Sözcüğün bütün olarak ele alınmasında yukarıda değinilen anlamların yetersiz

yetersiz kaldığı ve veritabanı kuramcıları tarafından bu terimin varlığına ait değişken olarak ele alındığı gözönünde bulundurularak "reference scheme" için niteleme öğesi karşılığı önerilmektedir. Çünkü Nijssen'in ortaya attığı bu terim veritabanı kuramcılarının ve kendisinin kullandığı "attribute" yani "özellik" sözcüğü ile aynı işlevi görmektedir.

Özellik (Attribute): "Attribute" ilişkisel veritabanlarında bir varlığı tanımlayan ve sütun başlıkları olarak ifade edilen herbir değişkenin adıdır. Bu değişkenler aynı zamanda varlığı tanımlayan özellikler olduğundan "attribute" karşılığı olarak "özellik" sözcüğü önerilmektedir.

Sıra (Tuple): Bu sözcüğe hiçbir dil sözlüğünde rastlanmamıştır. Ancak terim, ilişkisel veritabanı tablolarında bir varlığa ya da bir kayda karşılık gelen sıraları ifade etmektedir. O nedenle "tuple" karşılığı olarak "sıra" sözcüğü kullanılmıştır.

Söyleşi Evreni (Universe of Discourse): Nijssen'in ortaya attığı bu terime, konuyla ilgili olarak incelenen yabancı dildeki hiçbir belge/yayın ve terminolojik kaynakta rastlanmamıştır. Sözcüğe türkçe bir karşılık bulmak için yine sözcük içinde yeralan öğelerin ayrı ayrı ele alınması yöntemi izlenmiştir. Buna göre "universe" için "evren","discourse" için "karşılıklı konuşma" açıklaması yapıldığı görülmüştür.

Nijssen'in sözcüğe yüklediği anlam ve sözlükler de yapılan açıklamalar dikkate alınarak terim için "söyleşi evreni" karşılığının kullanılmasının doğru olacağı düşünülmüştür.

Veri Kataloğu (Data Dictionary): Türkçe'deki tam karşılığı veri sözlüğü'dür. Ancak sözlüğün işlevi dikkate alındığında görülür ki sözlük, sözcüklerin anlamlarını vermekte ve onları açıklamaktadır. Ancak katalog, kapsamı içindeki birimleri tanıtmanın yanında bunların yerlerini de gösterir. Veri kataloğunun işlevi de veri yapılarını tanımlamak ve ait oldukları kütükleri belirtmek olduğundan çalışmada "veri kataloğu" karşılığı benimsenmiştir.

Yalın Gerçek (Elementary (Fact): Yine hiçbir türkçe ve yabancı dildeki kaynakta izine rastlayamadığımız bu terim için sözcüklerin tek tek ele alınması yoluyla karşılık bulunmaya çalışılmıştır. Burada "elementary" sözcüğü gerçeğin en arınmış ve bölünmez halini ifade ettiği için türkçedeki "yalın" sözcüğü ile karşılanmıştır.

Yöntemsel Dil (Procedural Language): Türkçe kaynaklarda prosedürel dil olarak kullanılmaktadır. Ancak prosedürün programcılık açısından taşıdığı anlam değerlendirildiğinde prosedürün bir işlemin çözümlenme süreci olarak ele alındığı görülür. Yani çözümlemede izlenen yöntemi ifade etmektedir. Dolayısıyla prosedürel dil yerine yöntemsel dil kullanılması yeğlenmiştir.

1.6 KAYNAKLAR

Çalışmada kullanılan belge ve yayınlara erişimi sağlayan ikincil kaynaklar türkçe ve yabancı dilde olmak üzere iki grupta toplanmıştır.

Türkçe belge ve yayınlara erişim için **Türkiye Bibliyografyası**[3], **Türkiye Makaleler Bibliyografyası**[4] ve **Türk Kütüphaneciler Derneği Bülteni Dizini**[5] gibi kaynaklardan yararlanılmıştır.

Yabancı dildeki belge ve yayınların saptanmasında ise LISA veritabanı, Science Citation Index veritabanı ve YÖK Dokümantasyon ve Bilgi Tarama Merkezi aracılığıyla Dialog veritabanı kullanılarak tarama yapılmıştır. Bunun yanında ODTÜ ve Bilkent Üniversitesi Kütüphaneleri ile Amerikan Kültür Heyeti Kütüphanesi ve Norveç Kütüphanecilik Okulu Kütüphanesi katalogları taranmıştır.

İkincil kaynaklar aracılığı ile yapılan taramalarda veritabanı kuramı konusunda bilgi sağladığımız kaynakların çoğunlukla kitap türünde yoğunlaştığı gözlenmiştir. Makaleler ise daha çok konuyla ilgili tartışmalara veya çok genel bilgilere yer vermektedirler.O nedenle çalışmada ağırlıkla kitap türü yayınlardan yararlanılmıştır.

Çalışmanın III.Bölüm'ünde McFadden'in **Database Management**[6], Pratt'ın **Database Systems; management and design** [7], Date'in **An Introduction to Database Systems**[8] adlı yapıtları, IV.Bölüm'de Kroenke'nin **Database Processing; fundamentals, design, implementation**[9] ve Nijssen'in **Conceptual Schema and Relational**

---

3. Türkiye Bibliyografyası. Ankara: Maarif Vekaleti, 1949-
4. Türkiye Makaleler Bibliyografyası. Ankara: Milli Kütüphane, 1952-
5. Dizin; Türk Kütüphaneciler Derneği Bülteni/Türk Kütüphaneciliği, 1952-1992. Ankara: Kütüphaneler Genel Müdürlüğü, 1993.
6. Fred R. Mc Fadden-Jeffrey A. Hoffer; Database Management. Menlo Park, Calif: Benjamin Commings Pub. Com., 1985.
7. Philip J. Pratt-Joseph J. Adamski, Database Systems; management and design. Boston: South-Western Pub. Com., 1991.
8. J.C. Date, An Introduction to Database Systems. vol: 1, 4th ed. Reading Massachusetts, Calif: Addison Wesley, 1985.
9. David Kroenke-Kathleen A. Dolan. Database Processing; fundamentals, design, implementation. 3rd ed. Chicago: Science Research Associates, 1990.

Database design; a fact oriented approach[10] adlı yapıtları, V. Bölüm'de ise Folk'un **File Structures**[11] ve Holvik'in **Databaseteori**[12] adlı yapıtları temel kaynak olarak kullanılmıştır.

Bu kaynakların temel kaynak olarak seçilme nedeni, konuya sistematik bir biçimde, ayrıntılı olarak elealmaları ve diğer kaynakların sürekli kullandığımız bu yayınlara gönderme yapmalarıdır. O nedenle yinelemeye kaçan kaynakların kullanımından kaçınılmıştır.

Ayrıca çalışmanın sonunda yeralan Sözlük bölümünde gerek terimlerin açıklanmasında gerekse karşılık bulma çalışmalarında **Bilgi İşlem Terimleri Sözlüğü**[13],**Ansiklopedik Bilgi İşlem Terimleri Sözlüğü**[14], **Mikrosoft's Computer Dictionary, the comprehensive standart for business, school, library and home**[15] ve **Dictionary of Information Technology**[16] adlı terminolojik kaynaklardan yararlanılmıştır.

---

10. Gerardus Maria Nijssen-Terence Aldan Halpin, Conceptual Schema and Relational Database Design; a fact oriented approach. New York: Prentice Hall, 1989.
11. Michael J. Folk-Bill Zoellick, File Structures. 2nd ed. Reading Massachusetts, Calif: Addison Wesley, 1992.
12. Helge Holvik, Databaseteori. Oslo: Statens Bibliotek og Informasjonshogskole, 1991.
13. Gazi Güder, Bilgi İşlem Terimleri Sözlüğü. İstanbul: Birsen Yayınevi, 1986.
14. Ansiklopedik Bilgi İşlem Terimleri Sözlüğü. Hazl: Özlem Meltem Kurtaran, Faruk Çubukçu. İstanbul: Türkmen Kitabevi, 1991.
15. Microsoft's Computer Dictionary; the comprehensive standart for business, school, library and home. Redmond, Washington: Microsoft Press, 1991.
16. Dennis Longley-Michael Shain, Dictionary of Information Technology. 2nd ed. London: Mc Millan Press, 1985.

II. BÖLÜM
VERİTABANI ve GELİŞİMİ

2.1. VERİTABANI TERİMİ

Toplumlar bugünkü uygarlık düzeyini bilginin kaydedilmesi ile başlayan ve bilgiye yönelik olarak gerçekleştirilen bir dizi etkinliğin sonucunda yakalamışlardır, denilebilir. Özellikle bilgisayar ve iletişim teknolojisindeki gelişmelerin mal ve hizmet üretimine yansıması ile başlayan hareketlilik, bilginin önemini artırmıştır. Bu bağlamda bilgi, yalnız doğrudan bilgi üretimini gerçekleştiren ya da bilgi hizmeti veren kuruluşlar için temel bir öğe olmaktan çıkıp her türlü kurum, kuruluş ve işletme için vazgeçilmez bir kaynak durumuna gelmiştir.

Mal ve hizmet üretiminde söz konusu teknolojinin kullanılması verimliliği büyük ölçüde artırmış ve ekonomide meydana gelen hızlı değişimler karşısında kuruluşlara hareket kolaylığı sağlamıştır.

Kuruluşların bilgisayara dayalı bir sistemi bünyelerinde oluşturmalarının ilk ve önemli evresi, mevcut sistemdeki aksaklıkları ve geleceğe yönelik bilgi gereksinimlerini saptamaya yarayan sistem analizinin gerçekleştirilmesidir. Sistem analizinde yapılması gereken ilk işlem planlanan enformasyon sistemi için gerekli verilerin toplanmasıdır.

"Veri, insan ya da makina tarafından iletişim, açıklama ve işlem amacıyla herhangi bir amaç, konu, durum, koşul, fikir ya da diğer unsurları açıklamak için kullanılan sayılar, harfler ve simgeleri belirtmek üzere kullanılan genel bir terim olarak tanımlanmaktadır"[1].

Aynı zamanda veriyi varlıklara ilişkin değişkenler olarak da düşünebiliriz. Bu değişkenler, veri nesnelerinin nitelenmesinde bize yardımcı olur. Nesneler, kendilerini niteleyen diğer nesneler ile olan benzerliklerini ya da ayrılıklarını ortaya koyan veriler ışığında ilişkilendirilir ve sınıflanır. Bu yolla biraraya getirilen ve özellikleri doğrultusunda sınıflanan veriler aslında gerçek yaşamı yansıtan bir modeli de ortaya koymuş olurlar. Kavram oluşumu ile benzerlik gösteren model, aynı zamanda kuruluşlar için gereksinmeler doğrultusunda ve problem durumlarını çözmek amacıyla oluşturulan veritabanlarını da biçimlendiren temel olgudur.

Bilginin yeniden kullanım amacıyla kaydedilmesi ve düzenlenmesi için insanoğlu bugüne kadar birçok yöntem geliştirmiştir. Bu bağlamda veritabanı da kuruluşların kendi çalışma alanlarındaki yoğun bilgi birikimini denetim altında tutmak ve işlerinde kullanmak amacıyla geliştirdiği araçlardan biridir.

"Veritabanı" tabiri ilk kez ABD ordusunda 1963'te yapılan "Bilgisayar Merkezli Veritabanı Geliştirilmesi ve Yönetimi" adlı sempozyumda kullanılmıştır.

---

1. Ansiklopedik Bilgisayar Terimleri Sözlüğü. Hazl: Özlem Meltem Kurtaran-Fuat Çubukçu, İstanbul: Türkmen Kitabevi, 1992. 117 s.

Anılan sempozyumda yapılan tanıma göre:

1. Veritabanı, bir kütükler kümesi'dir;

2. Kütük, sıralanmış bir giriş öğeleri topluluğudur;

3. Giriş öğesi ise bir anahtar veya anahtarlar ile verilerden meydana gelir [2].

Veritabanı konusunun kuramsal altyapısının ortaya konmasına katkıda bulunmuş uzmanlardan J.C. Date, "Veri Tabanı Sistemlerine Giriş" (An Introduction to Database Systems) adlı kitabında veritabanını şöyle tanımlıyor: "veritabanı sistemi, temel olarak bütün amacı kaydetme ve bilgi sağlama olan bir kayıt tutma sistemidir"[3] .Date'in bu tanımı veritabanına çok genel bir anlam yüklemekte olup onun yapısı ve işlevlerine ilişkin açıklayıcı bir ifadeye yer vermemektedir. Kendisi ile yapılan bir görüşmede ise J.C. Date, veritabanını tanımlarken "sisteme o anda oluşturulup yöneltilen soruları yanıtlayabilme becerisine sahip kolay kullanılacak biçimde düzenlenmiş veri topluluğu"[4] ifadesini kullanmaktadır. Date, daha ayrıntılı olarak yaptığı bu ikinci tanımlamada özellikle,

1. Kullanıcının sistemle karşılıklı etkileşim halinde olduğu ve

2. veritabanı yönetim sisteminin sahip olduğu bilgi erişim stratejisi ile yalnız veritabanı yöneticisi ve bu alandaki uzmanların değil herkesin kolaylıkla yararlanabileceği bir yapıda oluşturulması gerektiği gibi iki temel işlevi vurgulamak istemiştir.

---

2. John Anthony Jones, Database in Theoy and Practice. London: Kogan Page, 1986, 33 s.
3. J.C. Date, An Introduction to Database Systems. vol: 1, 4th ed. Reading Massachusetts. Calif: Addison Wesley, 1985, 10 s.
4. Eliot B. Koffman, Pascal; problem solving and program design. 4th ed. Reading Massachusetts. Calif: Addison Wesley, 1992, 523 s.

Diğer taraftan Korth ve Silberschatz, veritabanı yönetim sistemi için "birbiri ile ilişkili veriler ve bu verilere erişimi sağlayan bir dizi programdan oluşur"[5] demektedir.

Sitansu S. Mittra ise, veritabanını "uygulama programları aracılığı ile işlerlik kazanan farklı kullanıcılar için farklı görünümler verebilen, birbiri ile ilişkili verilerin topluluğu" [6] olarak tanımlamaktadır.

Korth ve Silberschatz veritabanı yönetim sisteminin yapısına değinirken daha çok kütük yönetim sistemine benzer bir tanım getirmektedir. Bununla beraber her üç yazar, veritabanı terimine "yalnız bilgisayarla işlerlik kazanan veri topluluğu" anlamını yüklediğinden tanımları, Date'in tanımına oranla daha sınırlı kalmaktadır.

Daniel Martin, veritabanı için "iyi tanımlanmış bir konudaki ayrıntılı, tekrarlayıcı olmayan ve yapılandırılmış bilgi (enformasyon) topluluğudur",[7] dedikten sonra söz konusu tanıma açıklık getirmektedir.

Burada "iyi tanımlanmış konu" ibaresi veri kataloğunda tanımlanmış veri kümesi; "ayrıntılı" sözü ise veritabanının bir konu alanı ile ilgili tüm veri parçalarını içermesi, yinelemenin olmaması ve bu yolla veri tutarlılığını büyük ölçüde sağlaması anlamında kullanılmış-

5. Henry F. Korth-Abraham Silberschatz, Database System Concepts. Singapore: Mc Graw Hill, 1986, 1 s.
6. Sitansu S. Mittra, Principles of Relational Database Systems. Englewood Cliffs, N. 3: Prentice Hall, 1991, 4 s.
7. Daniel Martin, Advanced Database Techniques. Cambridge: MIT Press, 1986, 5 s.

tır. Veritabanı için "bilgi topluluğu" ifadesini kullanmakla Martin, bilgi ile veri arasındaki ayrımı gözardı etmiştir.

Hülya Polat'ın yaptığı tanıma göre ise veritabanı "bir kuruluşun değişik uygulamalarında kullanılmak üzere gereksiz yinelemelerden arınmış olarak merkezi denetimle bellekte tutulan ilişkili veriler bütünü"[8]dür. Polat burada veritabanının bütünleşik ve merkezi denetime sahip olma özelliklerine değinmekle önemli bir ayrıntıyı vurgulamıştır.

Ancak yukarıda verilen her iki tanımda kullanılan "bir kuruluşun uygulamaları..." ve "ayrıntılı konu alanı..." gibi ifadeler, veritabanında bir arada bulunması gereken olguları belirtmektedir. O nedenle tanımda her iki ifadeye de yer verilmesi gerekmektedir.

Koffman, konuya daha teknik bir açıdan yaklaşarak, bir tanım vermekten öte, şöyle bir açıklama yapar: "veritabanı bilgisayar belleğine ya da disk kütüğüne yüklenen bilgiyi içeren kayıt adındaki alt bölümlerden oluşur"[9].

Bratbergensen ise yaptığı iki ayrı tanımda veritabanı ile veritabanı sistemi arasındaki çarpıcı ayrımı özet olarak şöyle vermektedir. O'na göre "veritabanı", sınırlandırılmış bilgi gereksinimi için düzenlenmiş veri topluluğudur. "Veri tabanı sistemi" ise veritabanının oluşturulması, düzenlenmesi, depolama, erişim, güvenlik gibi birtakım işlevleri olan bir yazılım paketidir [10].

---

8. Hülya Polat, "Veri Tabanı Zamanı" Bakış 1: 1 (1986), 4 s.
9. Koffman, a.g.e., 486 s.
10. Kjetil Bratbergensen-Knut Hofstad-Kjell Wibe, Filsystemer og Databaser. 8 opplag.Trondheim: RUNIT Institutt for Databehandling, 1983, V-1 s.

Yukarıda konunun belli başlı uzmanlarının "veritabanı" terimiyle ilgili olarak yapmış olduğu tanımlar verilmeye çalışılmıştır. Konu, araştırmacının yaptığı bir tanımla toparlanacak olursa, "Veritabanı", bir konu alanında ya da bir uygulamaya yönelik olarak, bir model uyarınca oluşturulmuş, ayrıntılı, gereksiz yinelemelerden arınmış ilişkili veriler topluluğudur, denilebilir. Donanım, yazılım ve sorgulama dilleri alanlarında meydana gelen gelişmelerden etkilenen veritabanları veri güvenliğini sağlama, yanlışların düzeltilmesi ve kapsamlı sorgulama gibi ek özellikler kazanarak bütünleşik bir yazılım paketine yani veritabanı yönetim sistemine dönüşmüştür.

Literatürde terimle ilgili olarak yapılan tanımlamalarda ve yukarıda verilen kimi alıntılarda "veritabanı" teriminin sistemin geçirdiği tüm gelişmeleri de içeren kapsamlı bir terim olarak halen yaygın biçimde kullanıldığı gözlenmektedir.

2.2. TARİHÇE

Bilgisayarın kuruluşlardaki geleneksel kayıt tutma etkinliğinin yerini alması ve kullanımının giderek yaygınlaşması iletişim, işlem ve üretim alanlarında bilginin ön plana çıkmasına ve öneminin giderek artmasına neden olmuştur. Fife'nin bu düşünceyi destekler yöndeki, "eğer rakibinizin bir veritabanı varsa sizin de bu yönde bir girişimde bulunmanız yararlı olur, aksi halde rakibinizin gerisinde kalabilirsiniz" cümlesiyle belirtilen görüşü [11] aynı zamanda veritabanı yönetim sistemlerinin gelişmesini motive eden kullanıcı talebini de ifade etmektedir.

---

11. Dennis W. Fife-W. Terry Hardgrave-Donald R. Deutsch, Database Concepts. Cincinnati, Ohio: South-Western Pub, 1986, 10 s.

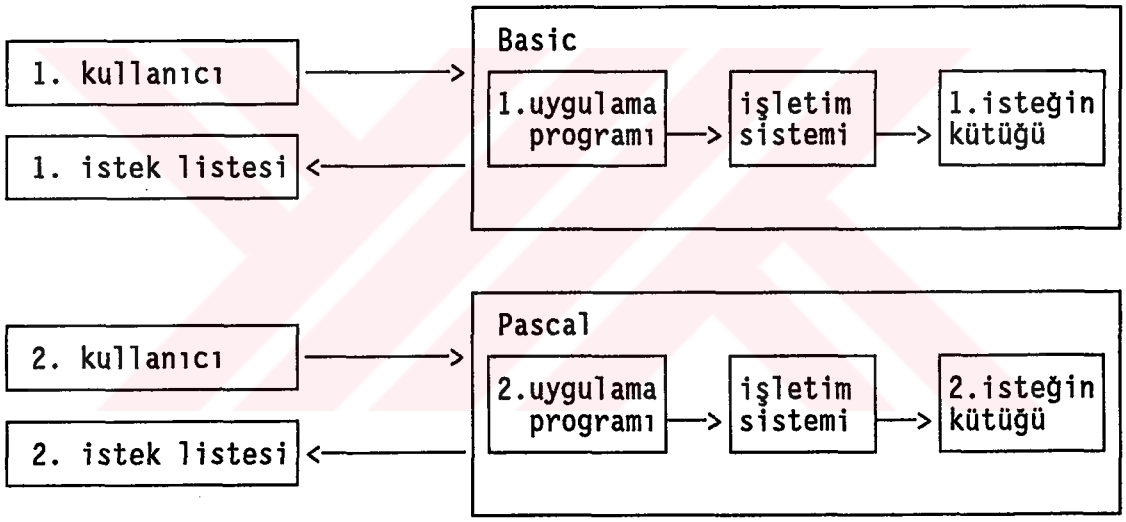
Bilgisayara dayalı kayıt tutma etkinliği 1960'lı yılların başında ilk kez kütük yönetim sistemi biçiminde ortaya çıkmıştır.

Kütük yönetim sistemi kullanıcı gereksinimleri doğrultusunda hazırlanan programlar ile o programların verilerini sağladığı kütüklerden oluşmaktaydı. Program ve kütükler farklı zamanlarda farklı programcılar tarafından oluşturulabilmekteydi. O nedenle kütükler farklı formatlarda yapılandırılabiliyor ve programlar farklı programlama dillerinde yazılabiliyordu. Dolayısıyla her programın veri tanımlaması kendi bünyesinde yapılıyordu. Bunun sonucu olarak kimi veri yapıları birden fazla program içinde yinelenebilmekte ve programlar yalnız çağrı yapabildikleri kütüklerdeki verileri görüp işleyebildikleri için özellikle güncelleme işlemlerinde farklı kütüklerdeki veriler ele alınamamamaktaydı. Bunun sonucu olarak çoklu kütükler içindeki veri yinelemesi veri kaynaklarının bütünü üzerinde tutarsızlığa yol açmaktaydı.

Veri yinelemesinin neden olduğu bir diğer sorun, depolama alanında yarattığı yer sıkıntısıydı. Aynı zamanda bir işlem için birçok kez ayrı kütüğe giriş yapma gereği işlemin uzun sürede tamamlanmasına yol açmaktaydı. Çeşitli kombinasyonları gerektiren kullanıcı soruları karşısında sistemin, önceden biçimlendirilmiş kütükler ve uygulama programları dışında bilgi üretememesi de bilgi erişim açısından olumsuzluk yaratıyordu.

Örneğin: belli bir tarihten sonraki siparişleri öğrenmek isteyen bir kütüphanecinin, program yalnız sipariş edilen eserler listesi-

ni üretebildiği için sorusuna yanıt alabilecek listeyi elde etmesi olanaklı değildi. Burada kütüphanecinin biri, programcıya bu yönde bir program yazdırmak diğeri, listeyi alıp o liste üzerinde tarih belirlemesini elle yapmak gibi iki seçeneği vardı. Ancak her iki seçenek de zaman ve emek açısından verimli birer yol değildi. "Verinin birçok kütük içerisinde yeralması ve kütüklerin farklı formatlarda bulunabilmesinden dolayı yeni uygulama programlarının yazımı da güçleşmekteydi"[12]. Konuya açıklık getirmesi bakımından kütük yönetim sistemlerinin işleyişini bir şema ile şöyle gösterebiliriz [13].



Şekil 2.1: Kütük Yönetim Sistemi

Yukarıda verilen şemadan da anlaşılacağı gibi kütük yönetim sistemlerinde farklı uygulama programları için farklı kütüklerin bulunması güncelleme işlemlerinin tek kütükten bir kez yapılmasına olanak vermemektedir. Oysa merkezi tek bir kütüğün bulunması işlemlerde kolaylık ve zaman tasarrufu sağlayacaktır.

12. Korth, a.g.e., 3 s.
13. Mittra, a.g.e., 2 s.

Bilgisayar çağının ilk evrelerinde kütük yönetim sistemlerinin ortaya koyduğu ve daha önce değinilen olumsuzlukların yanı sıra donanım kapasitesinin bütünüyle kullanılmaması ve bunun gibi diğer başka sorunlar da bulunmaktaydı. Örneğin makina dilinde yazılmış başlangıçtaki yazılımlar üç adımlı veri işleme döngüsünü (veri girişi, işlenmesi ve çıktı olarak görünümü) tamamlamadan yeni bir komutu devreye sokamıyordu. Bunun yanısıra bir uygulama programı işlevini tamamlamadan yeni bir program uygulamaya geçirilemiyordu.

İlk döneme özgü bütün bu sorunların çözümü için başlatılan çalışmalar, özellikle ana bellekte yer alan ve bilgisayar donanımının yönetimi ile uygulama programlarının yürütülmesi işlevini yerine getiren işletim sistemlerinin geliştirilmesi ve bu yolla veri kaynaklarının yönetimindeki verimsizliğin giderilmesi yönünde yoğunluk kazanmıştır. Söz konusu evrede üreticilerin bilincinde olduğu bir diğer olgu, sistemin kullanıcılara esneklik ve seçenek sunması talebinin pazara hakim olduğuydu. Bu bağlamda üreticilerin kullanıcılara sunduğu iki seçenekten biri "anadil" diğeri "kendi içinde yetebilen" bir sistemdir. **Anadil** seçeneği kullanıcıya veri yönetimini ve erişimini sağlayan komutları içeren yüksek bir programlama dili sunar. Söz konusu dil aynı zamanda veritabanı uygulamalarını yürütmekte yararlanılan aritmatik işlem fonksiyonlarını da kapsamaktadır [14]. **Kendi içinde yetebilen sistem** seçeneği, sorgu ve güncelleme komutları ile beslenmiş bir dil sağlamaktadır. Veritabanı işleme programları bütünüyle veritabanı yönetim sistemi dilinde yazılmıştır [15].

---

14. Fife, a.g.e., 10 s.
15. Aynı, 10 s.; İclâl Yıldırım, Veritabanı Yönetim Sistemi Yayınlanmamış Yüksek Lisans Tezi, İstanbul: İstanbul Üniversitesi, 1990, 7-8 ss.

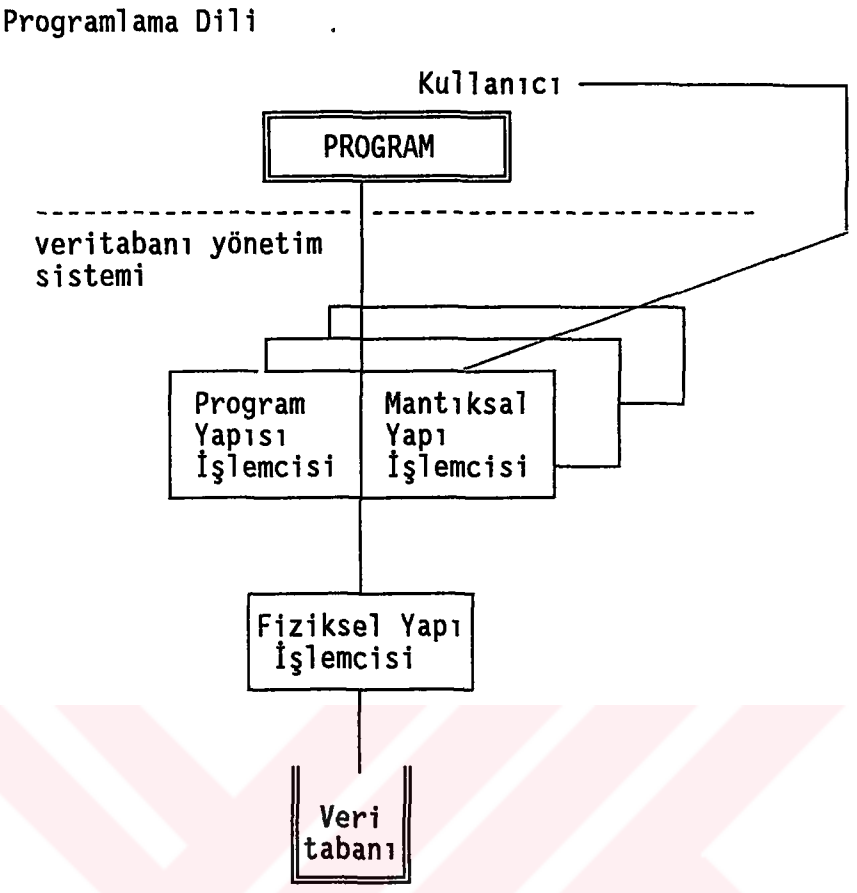
Yine bu döneme özgü bir sorun donanım ve yazılımın yüksek maliyette olmasıydı ve o nedenle bu tür sistemler ancak ana bilgisayarlar ve büyük, kapsamlı uygulamalar için sağlanabilmekteydi.

1970'li yıllara gelindiğinde disk depolama maliyeti azaldı ve tasarımda önemli gelişmeler kaydedildi. Örneğin programlama dilleri ve veritabanı yönetim sistemine özgü fonksiyonlar birbirinden ayrıldı. Böylelikle neyin nasıl yapılacağının açıklandığı Pascal, Cobol gibi yüksek düzeyli diller yerine yapılması gerekenin belirtildiği dördüncü nesil diller gündeme geldi. "Dördüncü nesil dil" ifadesi genellikle veritabanı dilleri için kullanılmaktadır. Bu diller büyük ve karmaşık veritabanlarına yönelik işlemleri gerçekleştirmek üzere hazırlanmışlardır. Veritabanı dillerinin yüksek düzeyde etkileşim, ilişkisel bütünlük,[16] veri ismi ile erişim,[17] bütünleşik veri kataloğu, dinamik iyileştirme,[18] veri güvenliği, tam otomatik onarım [19] ilişkisel etkileşimli arabirime sahip olma gibi özellikleri bulunmaktadır. Bu türün sağladığı olanaklardan biri de her seferinde birden fazla kayda ulaşılabilmesi olmuştur.

Veritabanı yönetim sisteminde fiziksel ve mantıksal düzeylerin ayrılması, bilgisayar çağının olgunluk evresi olarak nitelendirilmektedir. Bu ayrım aşağıdaki şemada açıklanmaktadır [20].

---

16. İlişkisel bütünlük: Tablolar halinde ifade edilen veriler ve bunlara ilişkisel cebirin uygulanabilmesi.
17. Veri ismi ile erişim: Verinin fiziksel konumunu bilme zorunluluğu olmadan içeriği ile sağlanan erişim.
18. İyileştirme: Bir program veya sistemin belli ölçütler (bellek kullanımı, hız vs.) çerçevesinde en etkin ve verimli çalışacak şekilde tasarlanmasıdır.
19. Onarma: Bir arıza sonrasında sistemi çalışır konuma getirmek için yapılan işlemlerdir.
20. Fife, a.g.e., 13 s.



Şekil 2.2: Programlama dili ve veritabanı yönetim sistemi fonksiyonlarının birbirinden ayrılışı

Veritabanı yönetim sistemlerinin fiziksel ve mantıksal düzeylerinin birbirinden ayrıldığı olgunluk evresinde daha önceden bilinen "veri modeli" kavramı söz konusu sistemlerin önemli bir özelliği olarak değerlendirilmiştir.

Veri modeli kavramının ortaya çıkışı ve ardından uygulamaya konması ABD Başkanı J.F. Kennedy'nin uzaya bir insan göndermek amacıyla başlattığı Apollo Projesi sonucunda olmuştur.

Kütük yönetim sisteminin yapısından kaynaklanan sorunların üstesinden gelecek bir sistem geliştirme isteği söz konusu proje bağlamında IBM firmasına iletilmiş ve bunun üzerine IBM, "Genelleştirilmiş

Güncelleme Erişim Yöntemi" (Generalized Update Access Method) adıyla bilinen sistemi 1964'te üretime sokmuştur. Ürünün daha başka alanlarda da kullanılabileceğinin keşfedilmesinin ardından IBM, halka yönelik DL/1 (Data Language/1) adlı sistemi geliştirmiştir. Gerçekte ürün, veritabanı yönetim sistemlerinin ilk örneklerinden biri olup öncü nitelik taşımaktadır [21].

Hiyerarşik modele dayalı bu ilk ürünlerde veri ilişkileri tümüyle ifade edilememekte yalnız sahip-üye (owner-member) ilişkisi ile sınırlı kalmaktadır. Çünkü bu ilişkilendirme bağlamında erişim kök düğümden başlayarak düğümlere doğru gerçekleşmektedir. Bu bakımdan veriye erişim için verinin yerini bilme zorunluluğunun bulunması gibi olumsuz özellikleri nedeniyle söz konusu sistemler yeni modellere dayalı sistemlerin doğmasına neden olmuşlardır.

Öte yandan 1963'te General Electric (sonraki adıyla Honeywell Information Systems) Charles Bachman tarafından geliştirilmiş olan bütünleşik veri sistemlerini pazarlamaya başlamıştır. Bunlar daha sonraki ağ veri modeline dayanan veritabanı yönetim sistemlerine öncülük etmiştir.

1960'lı yılların sonlarında "Veri Sistem Dilleri Konferansı" bünyesinde bir alt komite oluşturulmuştur. Birçok donanım ve yazılım satan firmalardan, üniversitelerden başlıca kullanıcı ve üreticilerden oluşan temsilcilerin katıldığı ve daha ziyade bir gönüllüler komitesi niteliğindeki "Veritabanı Komitesi"nin esas amaç ve görevi, Cobol

---

21. Philip Pratt-Joseph Adamski, Database Systems; management and design. Boston: South-Western Pub, 1991, 29 s.

programlama dilinde yapılması gereken değişiklikleri tartışmak ve sonuçları bir rapor halinde sunmaktı. Sonuçta Cobol dilinin akılcı bir biçimde geliştirilmesi ve çoklu kütük sistemleri ile birlikte işleyebilme özelliğini kazanması gerekliliği ortaya çıkmıştır [22].

IBM komitede temsil edilmesine rağmen ortaya konan standartları onaylamamış ve bu modeli temel alan bir veritabanı yönetim sistemi üretmemiştir. Ancak donanım satan ve yazılım üreten birçok firma, IBM bilgisayarları da dahil olmak üzere birçok bilgisayarda kullanılabilen yazılımlar üretmiştir. Kendisi Veritabanı Komitesi'nde yeralmamasına rağmen, Komitede Honeywell'den birçok temsilci bulunması ve bütünleşik veri sisteminin zamanın en gelişmiş veritabanı yönetim sistemi olması nedeniyle, **Bütünleşik Veri Sistemini** geliştiren Charles Bachman'ın fikirleri Komiteyi oldukça etkilemiştir.

CODASYL (Conference on Data System Languages) adı ile anılan ağ veri modeli mantığının kullanıcılar tarafından kolay anlaşılmaması ve ilişkilerin kolay kavranmaması bu sistemin olumsuz bir yanı olarak değerlendirilmektedir. Bununla beraber yaygın kullanım alanı bulmuştur. Ingres ve Image bu modeli temel alan sistemlere örnek gösterebilir.

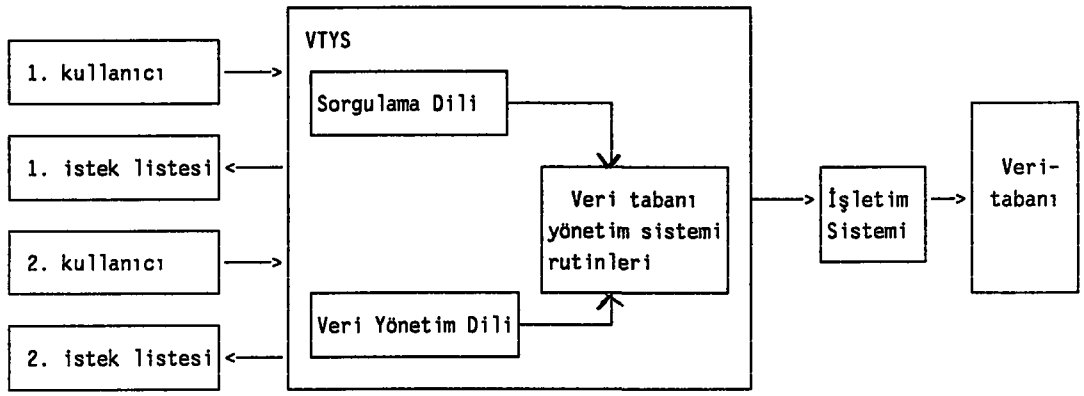
1970 yılında ise E.F. Codd tarafından ilişkisel model ortaya atılmıştır [23]. Bu modele dayalı sistemler kullanıcı için daha çok veri miktarına ulaşma olanağı vermektedir. Bunun yanında soruların

---

22. Fred R. Mc Fadden-Jeffrey A. Hoffer, Databased Management. Menlo Park, Calif: The Benjamin Commings Pub. Com., 1985, 418 s.
23. E.F. Codd, A Relational Model of Data for Large Shared Data Banks CACM, 13:6 1970, 378 s.; The Relational Model for Database Management. versiyon: 2, Reading Massachusetts, Calif: Addison Wesley, 1990, 1 s.

anında oluşturulup makina ortamına aktarılabilmesi, kullanıcıya makina ile karşılıklı etkileşim olanağı sağlamıştır. Önceki modellerde gösterici veri yapısından yararlanılarak veriler arasında bağlantı kurulurken ilişkisel modelde söz konusu bağlantılar mantıksal göstericiler ile sağlanmıştır. Bu durum özellikle araştırıcı kesime kapsamlı sorgulama olanağı tanıyarak çok büyük yarar getirmiştir. Ancak fazla miktarda veri tekrarının bulunması bu model için olumsuz bir nitelik olarak değerlendirilmektedir.

Kütük yönetim sisteminin işleyişine daha önce bir çizimle açıklık getirilmeye çalışılmıştı. Buna karşılık veritabanı yönetim sisteminin verileri nasıl bütünleşik olarak merkezi bir denetimle yönetebildiğini aşağıdaki gibi bir diğer çizimle somut biçimde ortaya koymak olanaklıdır. Böylelikle kütük yönetim sisteminden bütünleşik sistemlere geçişi kavramak daha kolay olacaktır.



Şekil 2.3: Veritabanı Yönetim Sistemi[24]

24. Mittra, a.g.e., 3 s.

2.3. VERİTABANI YÖNETİM SİSTEMİNİN ÖĞELERİ

Veritabanı yönetim sistemi karmaşık ve kapsamlı bir program olup sekiz temel öğeden meydana gelmektedir. Bunların ilk ikisi **"veri tanımla dili"** (DDL: Data Definition Language) ve **"veri yönetim dili"** (DML: Data Manipulation Language)dir [25]. Veri tanımlama dili ilişki veya kaydın yapısını tanımlamada kullanılır. Örneğin bir kayıt tipinin ya da bir tablonun alanlarında kullanılan veri yapılarının neler olduğu ve bunlar için öngörülen uzunluklar belirtilirken veri tanımlama dilinden yararlanılır.

Konu daha somut bir biçimde açıklanacak olursa, örneğin: İlişkisel modelde SQL (Structured Query Language) aracılığı ile bir öğrenciye ait verilerin yeraldığı **öğrenci tablosu** yaratılmak istendiğinde tablonun oluşumu şöyle formüle edilebilir:

```
CREATE TABLE ÖĞR.

Öğr. no.: number (8)
Öğr. no.: char (20)
Öğr. no.: char (10)
```

Burada **öğrenci tablosu** için öğrencinin numarası, adı ve adresi olmak üzere üç alan saptanmıştır. Ardından bu alanların sayısal ya da karakter veri yapısında olduğu ve alanın uzunluğu belirtilmiştir.

---

25. Emin D. Aydın, Veritabanı. İstanbul: Evrim Basım Yayın, 1980, 55-58 ss.

Veri yönetim dili ise erişim ve güncelleme olmak üzere başlıca iki işlevi yerine getiren bir grup komuttan oluşmaktadır. Find, Store, Delete, Insert gibi komutlarla gerçekleştirilebilen işlemler şöyle özetlenebilir:

Erişim İşlemleri:

1. bir ilişkiden seçim (selection) yapılabilir;
2. seçim sonucu azalan ya da artan değerler doğrultusunda sıralanabilir;
3. ortak özellikler aracılığı ile birleştirilen ilişkilerden seçim yapılabilir;
4. mantık belirteçleri kullanılarak seçim yapılabilir;
5. matematiksel karşılaştırma işaretleri kullanılmak suretiyle seçim yapılabilir;
6. DISTINCT, GROUP BY ve HAVING gibi özel birtakım kalıplar yardımıyla seçim yapılabilir [26].

Güncelleme İşlemleri:

1. yalnız bir sıranın güncellenmesi;
2. birden fazla sıranın güncellenmesi;
3. bir alt soru ile güncelleme;
4. birden fazla ilişkinin güncellenmesi;
5. tek sıraya veri girme;
6. birden fazla sıraya veri girme;
7. tek bir sıradan veri silme;
8. birden fazla sıra veya ilişkiden veri silme [27].

---

26. Aynı, 35 s.
27. Aynı, 37 s.

Veritabanı yönetim sisteminin bir diğer temel öğesi "sorgulama dili"dir. DML ve DDL genelde veri altdili olarak bilinmektedir. Sorgulama dili ise bu veri alt dilinin bir altkümesidir. Teknik olarak sorgulama dili kullanıcıdan gelen soruları yanıtlamak amacıyla kullanılmaktadır.

Kullanıcı sorgulama dilini kullanırken soruyu formüle etmek için kimi ölçütlere özen göstermek zorundadır. Bu ölçütlerin içermesi gereken bilgiler şöyle sıralanabilir:

1. seçilecek özelliklerin adı;
2. seçilen özellikleri içeren ilişkilerin adı;
3. seçim koşulu gibi [28].

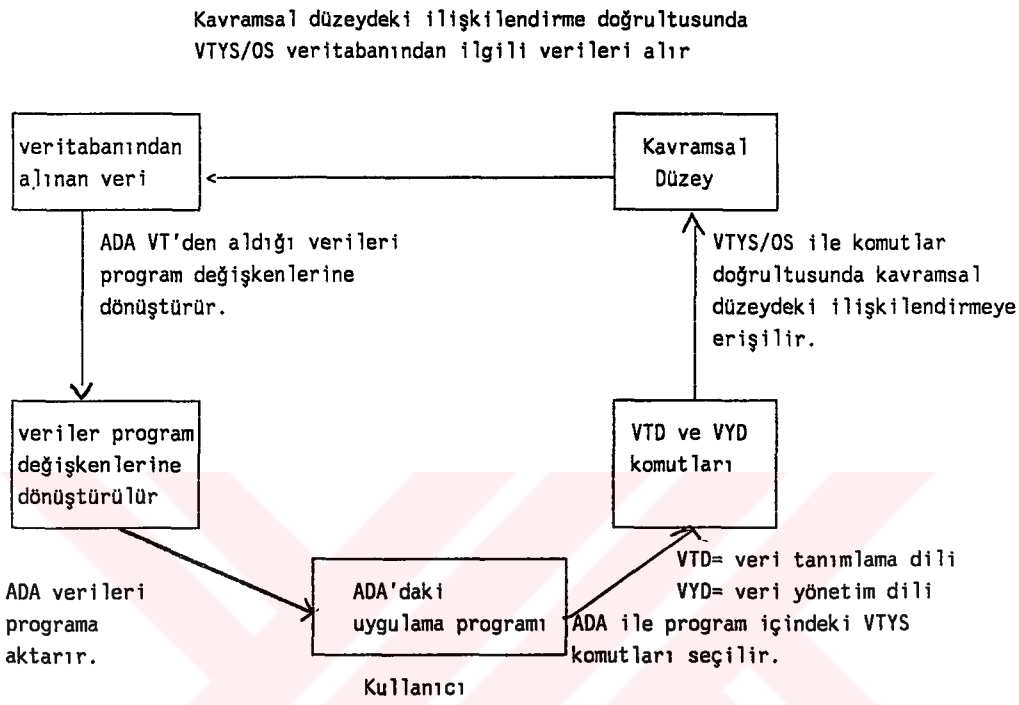
Soru aynı zamanda birden fazla seçim ifadesi de içerebilmektedir.

Veritabanı yönetim sisteminin bir başka öğesi olan **"etkileşimli anadil arabirimi"**, etkileşimli soru dilinden daha kapsamlı fonksiyonlardan meydana gelir. Yöntemsel dilin sözdizimini kullanmaktadır. Sorgulama dilinin yetersiz kaldığı durumlarda karmaşık raporların üretimi için kullanılan uygulama programları, veritabanı yönetim sistemi komutlarını da içermektedir. Böylelikle uygulama programındaki komutlar doğrultusunda arabirim, veritabanı yönetim sistemi (VTYS)'nin gerekli verileri veritabanından sağlayıp uygulama programına aktarmasına yardımcı olur.

---

28. Aynı, 40 s.

Aşağıdaki şekilde anadil arabirimi (ADA)'nin uygulama programında oynadığı rolü görebiliriz [29].



Şekil 2.4: Anadil arabiriminin uygulama programındaki rolü.

Veritabanı yönetim sistemi yazılımının bir diğer parçası olan "**rapor yazıcı**", istenilen biçimlerde rapor üretir. Rapor yazıcının genel özellikleri şöyle sıralanabilir.

1. sütun başlığı, rapordaki her özellik için tanımlayıcı başlık;
2. rapor başlığı, sayfa ve tarih= rapor için tanımlayıcı başlık, sayfa sayısı ve raporun üretildiği tarih;
3. dipnot, her sayfanın sonuna yazılan bir başlık ya da açıklama;
4. alt kapsamlar ve denetim aralıkları, rapordaki sıraların gruplara ayrılarak her grup sonunda toplamların alınması ve belirtilmesi;

29. Aynı, 45 s.

5. sayfa kapsamı, her sayfada basılan sıra sayısının belirtilmesi [30].

"Grafik üreticisi", kullanıcıya iş grafikleri üretme olanağı verir. Rapor yazıcıyla benzer işlevleri olmasına karşılık ayrıldığı nokta, raporu çizim biçiminde üretebilmesidir.

Veritabanı yönetim sistemini meydana getiren bir diğer öğe, **"yöntemsel dil"**dir. VYD, karmaşık raporları üretebilmek için, kimi zaman yöntemsel programlama dili içerir. Bu dil, bir problemin çözüm işleminin nasıl yapılacağını açıklayan, bilgisayarınkinden bağımsız bir dildir. Kullanıcı yöntemsel VYD'ni kullanabilmek amacıyla, komutlar içeren bir kütük oluşturur ve daha sonra bu kütük rapor üretiminde kullanılır [31].

Veritabanı yönetim sisteminin bir diğer öğesi **"veri kataloğu"**dur. Veri kataloğu veritabanı sistemini meydana getiren tüm veri tabanları ile ilgili veri nitelemelerini içerir. Veri kataloğunun içeriği veri üzerine veri olarak da değerlendirilebilir, çünkü sistemdeki tüm kütük ve programlar ile ilgili verilerin nitelemesini ve yerlerini gösterir bilgiyi içermektedir.

2.4. İŞLEYİŞ

Veritabanı yönetim sisteminde bilgi akışını, ilişkisel veritabanı yönetim sistemi ve SQL sorgulama dilini kullanarak şöyle örnekleyebiliriz.

---

30. Aynı, 42 s.
31. Aynı, 40 s.

"SELECT Belg. no., Başlık FROM Belge" SQL'deki yazılım biçimi ile ifade edilen böyle bir soruda kullanıcı, Belge kütüğünden (tablosundan) belge numarası ve başlık alanlarını kapsayan bir belge listesi talep etmektedir.

Buna göre:

1. Kullanıcı ilk adım olarak soruyu makinaya girer;

2. Veritabanı yönetim sistemi soruyu çözümler, veri tanımlama dili aracılığı ile belge numarası ve başlık gibi istenen alanların veri kataloğu ile uyumunu inceler ve bu alanları içeren kayıtların disk üzerindeki fiziksel konumunu saptar.

3. Kütük işletim sistemini harekete geçirerek verilerin hangi kütüklerde bulunduğunu saptar.

4. Veritabanı yönetim sistemi veri yönetimine ilişkin SELECT, FROM gibi komutları makinanın komutlarına uygun olarak çevirir ve denetimi işletim sistemine bırakır,

5. İşletim sistemi kayıtlara erişerek bunları ara belleğe (buffer) taşır,

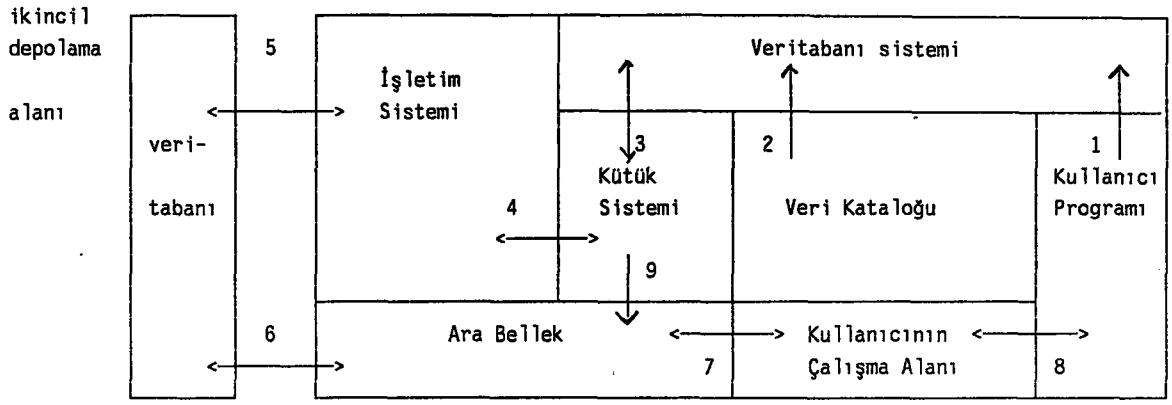
6. Veritabanı yönetim sistemi ara bellekten verileri alarak kullanıcının çalışma alanına ulaştırır,

7. Veritabanı yönetim sistemi son olarak istenen belge listesini ekranda görüntüler.

Bilgi akışı bir çizimle daha somut bir biçimde ortaya koyulabilir [32].

---

32. Bratbergensen, a.g.e., V-5 s.



Şekil 2.5: VeriTabanı yönetim sisteminde veri akışı

2.5. VERİTABANININ KURULUŞLAR İÇERİSİNDEKİ YER VE ÖNEMİ

Kuruluşların, ister mal ister hizmet üretsinler, rekabet dünyasında etkili ve verimli olabilmelerinde en iyi çözüm sistem yaklaşımıdır. Ortaya çıkışı pek de yeni olmayan bu kavram bağlamında kimliği ne olursa olsun herşey sistem olarak düşünülmektedir. Bu doğrultuda sistem, "aralarında belirli ilişkiler bulunan aynı zamanda çevre ile de ilişkisi olan bir veya daha fazla amaca, hedefe veya sonuca ulaşmak üzere birlikte hareket eden fiziksel veya kavramsal birçok bileşenden oluşan bir bütün"(33) olarak tanımlanmaktadır.

Yukarıda verilen tanımdan yola çıkarak kuruluşları açık birer sistem biçiminde nitelemek olanaklıdır.

Çünkü örgüt yapısını oluşturan birimler ve söz konusu yapının dışında kalan diğer sistemler birbirleri ile sürekli etkileşim halindedir. Günümüzde değişimler, iletişim ve bilgisayar teknolojisinin

33. Bengü Çapar, "Bilgi İşlemlerinin Yönetiminde Sistem Yaklaşımı ve Sistem Analizi" Jale Baysal'a Armağan. Yay. haz. Hasan S. Keseroğlu. İstanbul: İstanbul Üniversitesi, Mart 1993, 51-52 s.

gösterdiği gelişim doğrultusunda son derece hızlı bir biçimde gerçekleşmektedir. Bu bakımdan kuruluşlar açısından sağlanacak başarının temel etkenlerinden biri kuruluşun hiyerarşik örgütsel yapısı içinde işlem birimlerinden karar verme mekanizmalarına kadar özellikle, bilgi birikiminin varlığı ve bilgi dolaşımının zamanında hızlı ve güvenilir biçimde gerçekleştirilmesidir. Söz konusu sistemin varlığı, kuruluşlar içerisinde kullanıcıların işlemleri gerçekleştirmesi ve her düzeyde gereksinmeler doğrultusunda rapor üretebilmelerini sağlamanın yanında "yöneticilerin almak zorunda oldukları kararlarda gerekli olan işletme bilgilerini istedikleri anda öğrenebilmelerine ve yöneticilerin olayların değerlendirilmesine daha fazla zaman ayırabilmelerine de olanak tanımaktadır"(34).

Bunun yanında tarih boyunca kuruluşların işlemlerinin en önemli parçasını oluşturan veri, bilgisayar teknolojisinin ortaya çıkışıyla bir kaynak olarak da değerlendirilmeye başlanmıştır. Bu yaklaşım aynı zamanda tıpkı yaygın olarak bilinen insan, para ve materyal kaynakları gibi verinin de benzer birtakım özelliklere sahip olduğunun anlaşılmasına yol açmıştır. Yaygın olarak bilinen bütün bu kaynaklar maliyet ve değer özelliklerini paylaşırlar (35).

Kuruluşun bütün kaynaklarını yönetmek için gerekli olan bilgi, verilerden türetilmektedir. Veri, anlaşılabilmeli, korunabilmeli, da-

---

34. Ataç Soysal, "Yönetimde Bilgisayarlar ve Günümüz Endüstriyel İşletmelerinde Bilgisayarın Yeri", Bilgisayar Destekli Yönetim Sistemleri: Seminer. İstanbul: İstanbul Teknik Üniversitesi İşletme Fakültesi Endüstri Mühendisliği Bölümü, 1989, 5 s., Hakan Baykut, "Bilgisayar Teknolojisi ve Örgütsel Değişim", Bakış. 3:9 1988, 10 s.
35. W. Leong Hong Belkıs-Bernard K. Playman, Data Dictionary; Directory Systems, Administration, Implementation and Usage. New York: John Wiley and Sons, 1982, 2 s.

ğıtılabilmeli, işlenebilmeli ve bütünleşik olarak ele alınabilmelidir. Verinin anlaşılmasında verinin doğasını, karakteristiklerini, nasıl ve ne için kullanılacağını ve nereden sağlandığını bilmek gereklidir. Böylece o veri bir kaynak olarak etkin bir biçimde yönetilebilir. Veritabanı yönetim sistemi de bu bağlamda veriyi kullanan ve işleyen bir sistemdir. Bunun doğal sonucu olarak da veritabanı yönetim sistemlerinin kuruluşlar içerisinde önemli ve etkili bir yer işgal ettiği söylenebilir.

2.5.1. Veritabanı Planlaması

Daha önce açık birer sistem olarak nitelediğimiz kuruluşlar, dış dünya ile sürekli etkileşim halinde olduğundan içinde bulundukları koşulları değiştirmeye ve yakın gelecekte karşılaşabilecekleri yeni durumları belirlemeye yönelik yeni düzenlemeler yapmak zorundadırlar. Daha bilimsel bir deyişle, kuruluşlar sistem analizi yaparak mevcut sorunları çözer ve yeni hedeflere ulaşırlar. Sistemlerin zamanla büyüyerek karmaşık hale gelmesi ve yönetimdeki gelişmeler sistem analizini önemli kılmaktadır.

Sistemdeki mevcut aksaklıkların saptanması veya sistemin daha etkin ve verimli kılınması için sorunların belirlenmesi, bu yolla yapılacak çalışmaların ilk aşamasıdır. Amaçlar doğrultusunda gereksinimlerin saptanması ve verilerin toplanması, toplanan verilerin ilişkilendirilerek yeni sisteme yönelik tasarımın gerçekleştirilmesi ikinci aşama olarak nitelendirilebilir [36]. Örneğin uygulamaların kime ne

---

36. Zafer Kurdakal, "Bilgi Bankasına Girecek Verilerin Ön Analizi" Elektrik Mühendisliği, 15: 176/177, 1971, 37-38 ss.
"Veritabanı Yönetimi" Bilgisayar, 8:52, 1985, 70 s.

kazandıracağı, ne kadara mal olacağı, sonuçların yaklaşık ne kadar bir süre sonunda elde edileceği, kuruluş içerisinde bilgi akışının nasıl gerçekleşeceği, kullanıcıların yeni uygulamalardan nasıl etkileneceği gibi soruların dikkate alınması gerekir.

Kuruluşlar yeni bir problem çözme sürecine girdiklerinde şu üç temel işlevi gerçekleştirmek durumundadırlar.

1. dış ve iç çevrenin tanımlanması;
2. dış ve iç çevreyi oluşturan öğelerin tanımlanması;
3. bütün öğeler arası bağıntıların tanımlanması [37].

Aynı zamanda veri analizi süreci olarak da nitelendirilebilecek bu sürecin giderek önem kazanma nedenlerini Curtis kısaca şöyle özetlemektedir [38]:

1. bütünleşik veritabanlarının kullanımı artmaktadır;
2. veri, bir girdi/çıktı ürün olmaktan çok bir kaynak olarak ele alınmaktadır;
3. verinin işlenmesi zaman içerisinde değişime uğrasa da veri yapısı sabit tutulmaktadır.

Söz konusu süreç içerisinde veri birikimi çok fazla değişikliğe uğramasa da veriye erişim ve veriyi işleme süreci yenilenmeyi zorunlu kılmaktadır. Bu bakımdan veritabanının veri yönetiminde kullanı-

37. Mete Tanrıkut, "Sistem Analizi" Bakış, 2:6, 1987, 9 s., "Veritabanı Yönetimi" Bilgisayar, 8:52, 1985, 70 s.
38. Graham Curtis, Business Information Systems; analysis, design and practice. Washington: Addison Wesley, 1989, 433 s.

lan dil ve uygulama programlarından bağımsız olması önem taşır. O nedenle veritabanının dikkatlice planlanması zorunludur. Veritabanının planlanması ve bir proje biçiminde uygulamaya konması sistem analizi ve sistem tasarımı ile benzer bir yöntem izler.

Planlama için öncelikle mevcut bilgi gereksinimi doğru saptanmalı ve gelecekte ortaya çıkabilecek yeni koşullar gözönünde bulundurulmalıdır. Bu amaçla yönetim, üretim, pazarlama ve destek hizmetler gibi kuruluşun işlevsel birimlerince anlaşılıp desteklenecek stratejik bir model oluşturulur.

Veritabanı zaman içerisinde aşamalı olarak bir plan dahilinde yapılandırılmalıdır. Böyle bir plan için üç aşama saptanmıştır: **stratejik, taktik ve işlem denetimi** (39).

Planın stratejik kısmında kuruluşun amaç ve hedefleri, gereksinim duyulan kaynakların kullanımı ve dağılımı ile ilgili olarak saptanan politikalar yer alır.

Planın taktik kesiminde bir önceki aşamadaki saptamaların yerine getirilip getirilmediği yolunda denetim yapılır. Örneğin: kaynaklar sağlanmış mı?, verimli kullanılıyor mu?, hedeflere ulaşılabilmiş mi?, gibi.

İşlemlerin denetiminde ise belirli iş ve görevlerin yerine getirilip getirilmediğine ilişkin saptamalar yapılır.

---

39. Mc Fadden, a.g.e., 57 s.

Veritabanı planlaması genel planın bir alt kesiti biçiminde uzun vadeli olarak yapılmalı ve resmi bir proje niteliğinde gerçekleştirilmelidir.

Proje grubunda yer alacak birey sayısı kuruluşun büyüklüğüne ve karmaşıklığına göre saptanmalı ve proje grubu yalnız kuruluş çalışanları ile değil kullanıcı kesimi ile de görüşmelerde bulunmalıdır.

Proje grubu raporu kuruluşun bilgileme sistemi yöneticisine ya da bir üst düzey yöneticiye sunmalıdır.

Bir kütüphanenin hizmetlerini bilgisayara dayalı olarak gerçekleştirme yolunda yapılacak planlama söz konusu olduğunda planın aşamaları aşağıdaki gibi belirlenebilir:

1. Proje grubu öncelikle el yordamı ile yürütülen hizmetlerin neler olduğunu, bu hizmetlerin verilmesinde karşılaşılan sorunları saptar. Örneğin yönetime ilişkin sıradan işlerin zaman, emek ve parasal açıdan bütçeye getirdiği yük bilgi sağlama açısından kapsamlı bilginin varolan kataloglar ve düzenleme ile kullanıcıya zamanında sunulamaması problem olarak düşünülebilir.

Bunun ardından veritabanının getireceği yarar belirlenir. Örneğin yönetime ilişkin işlerin bir veritabanı yönetim sistemi ile gerçekleştirilmesinin sonucu, kurum çalışanları ile ilgili anında ve tam bilgi elde etmek olacaktır. Karmaşık bütçe ve istatistik hesaplarının daha kısa zamanda kolaylıkla ve doğru yapılması sağlanacaktır. Kısa zamanda karmaşık raporlar üretilebilecektir.

Diğer taraftan bir veritabanı yönetim sistemine dayalı olarak verilecek bilgi sağlama hizmetinde, kullanıcı gereksinimine uygun olarak, kapsamlı bilgiye değişik uçlardan erişmek veya türetmek kısa zamanda gerçekleştirilebilecektir.

2. Kütüphanede gerçekleştirilen çeşitli hizmetler için oluşturulacak veritabanının yönetimini gerçekleştirecek bir grubun belirlenmesi ve söz konusu grubun işlev ve sorumluluklarının tanımlanması;

3. Kütüphanedeki okuyucu ve teknik hizmetler kapsamında yapılan bilgi sağlama (güncel duyuru, seçmeli bilgi yayımı, kullanıcı sorularının yanıtlanması gibi) kataloglama, sınıflama, ödünç verme gibi hizmet türlerinin saptanması ve buna bağlı olarak anılan hizmetlerde rol alan temel varlıkların (materyal ve kullanıcı gibi) belirlenmesi;

4. Hizmetlerin verilmesi sırasında gerçekleştirilen işlemlere yönelik varlık tiplerinin oluşturulması, varlıklar arası benzerliklerin ortaya konması, yani model oluşturma. Örneğin kullanıcı, personel, yayıncı, kitap, süreli yayın vb. gibi, varlık tiplerinin ve özelliklerinin belirlenmesi (söz gelişi kullanıcı için adı, soyadı, üye no, adresi) ya da bir başka deyişle bu varlıklara ilişkin değişkenlerin saptanması ve benzer özellikler doğrultusunda sınıflanmaları;

5. Gerekiyorsa veritabanının fiziksel olarak bölümlenmesi;

6. Veritabanının uygulamaya konma sürecinde izlenecek zaman

7. Rapor ve değerlendirmelerin kütüphanenin bağlı bulunduğu üst yönetime sunulması.

2.5.2. Veritabanı Yönetim Sisteminin Sağladığı Olanaklar

Verilerin, veritabanı yönetim sistemi gibi kapsamlı bir yazılımla bütünleşik olarak tek merkezden yönetimi kuruluşlar ve çalışanları açısından birçok olanaklar sağlar. Söz konusu avantajlar şöyle sıralanabilir:

1. **Ekonomi:** İşlemlerin bir merkezden yürütülmesi o işlemler için tek tek harcanan zaman, emek, donanım ve paradan tasarruf sağlar. Aynı zamanda veritabanı sisteminde gerçekleşecek en küçük bir yenilik tüm kullanıcılara bu gelişmeden yararlanma olanağı sunar.

2. **Aynı Verilerden Farklı Bilgi Sağlama:** Temel amacı verileri işleyerek bilgiye dönüştürmek olan bilgisayar sisteminde eğer veriler ayrı kütükler içerisinde depolanmışlar ise her kullanıcının, veri sistemde yeralmasına rağmen, ona erişmesi mümkün olmayabilir. Oysa ortak/merkezi tek bir veritabanında eğer kullanıcı söz konusu verilerden yararlanma yetkesine sahipse istediği bilgiyi çıktı olarak elde edebilir [40].

3. **Veri Paylaşımı:** Veriler yetkin kişiler arasında paylaşılmıştır. Veri yönetim yetkesinin paylaşımı ile kullanıcı veriye erişebilse de veri üzerinde her işlemi yapamaz.

4. **Çelişkili Gereksinimlerin Dengelenmesi:** Veritabanının bütünüyle işlevsel çalışabilmesi için kuruluşlar içerisinde veritabanından sorumlu bir kişi ya da grubun bulunması gerekir. Böyle bir grup ya da kişiye "Veritabanı Yöneticisi" denmektedir. Söz konusu grup ya da kişi sürekli olarak kuruluşun tümüne yararlı olabilecek durumları göz önünde bulundurur. Bu grup aynı zamanda kuruluş içerisinde veriye yönelik bütün işlemleri standart hale getirir.

5. **Yineleme Denetimi:** Program ve kütükler zaman içerisinde farklı kişiler tarafından yüklenebileceğinden, veri yinelemesi meydana gelebilir. Örneğin bir kütüphanede hem kitaplar hem plaklar ödünç verildiğinde ödünç alan kullanıcıların isimlerinin iki ayrı kütükte yer alması mümkündür. Bu tür veri yinelemesine veritabanının işlemesinde kolaylık sağlamak bakımından, sınırlı ölçüde yer verilir.

6. **Tutarlılık:** Veri tutarsızlığı özellikle verinin birkaç kez giriş öğesi olarak kullanılmasından kaynaklanır. Bu bakımdan güncelleme işlemleri sırasında veriyle ilgili değişikliğin tüm giriş öğelerine yansıtılması gerekmektedir. Aksi halde tutarsızlık meydana gelir.

7. **Veri Güvenliği:** Kullanıcıların tüm verilere erişmesi ve bunlar üzerinde işlem yapması söz konusu değildir. Erişim ve işlem yapma yetkisi sınırlandırılarak veri güvenliği sağlanmış olur.

8. **Veri Bütünlüğü:** Veritabanındaki veriler arasında çelişki olmamalıdır. Bir veriyle ilgili farklı değerlerin bulunması veri doğ-

ruluğunu yok eder. Bu açıdan merkezi denetim bulunması sağlıklı sonuçların alınmasını sağlar [41].

9. **Veri Bağımsızlığı:** Programların veritabanı yapısından bağımsız olması anlamına gelir. Böyle bir durumda veritabanı yapısında meydana gelecek bir değişiklik karşısında uygulama programları bundan etkilenmez dolayısıyla programlarda bir değişiklik yapma zorunluluğu doğmaz [42].

Veritabanı yönetim sisteminin sahip olduğu ve yukarıda sözü edilen özelliklerle olanaklar yanında, kimi dezavantajları da bulunmaktadır.

1. Veritabanı yönetim sisteminin oluşum ve hizmete geçiş süreci pahalıdır. Veritabanı tasarımı için kuruluşun öncelikle varlıklar ile ilgili olarak tutmak istediği verilerin incelemesi yapılır. Bunlar arasındaki ilişki ve bağlantılar saptanır. Bu bakımdan pahalı ve zaman alan bir süreçtir. Oysa kütüğe dayalı sistemlerde yeni uygulama gereksinimleri ortaya çıktıkça tasarım işi gerçekleştiğinden bu işlem daha kolay olup, maliyet, uygulamaların ortaya çıkış zamanına dağılmıştır [43];

---

41. Ender Başer, "Bütünlüğün Veritabanı Sistemlerindeki Önemi ve Kıyaslanması" Bilgisayar Araştırma ve Uygulama Merkezi Dergisi. 8:1, 1985, 35-36 ss.
42. Pratt, a.g.e., 32 s.; Ethem Refi Kök, "Data Yığını Yönetim Sistemleri" İstanbul Teknik Üniversitesi Elektronik Hesap Bilimleri Enstitüsü Elektronik Bilgi İşleme Ulusal Sempozyumu II, 25-26, 4, 1977, İstanbul: İstanbul Teknik Üniversitesi, 1977, 166-167 s.
43. Curtis, a.g.e., 164 s.

2. Bütünleşik bir sistem olduğu için sistemin herhangi bir noktasında meydana gelebilecek bir aksama sistemin işlerliğini bozabilir;

3. Sistem kapsamlı ve karmaşık olduğundan gelişmiş bir donanım kullanımını gerektirmektedir.

4. Veritabanı yönetim sistemlerinde erişim zamanı kütük yönetim sistemine oranla daha yavaştır. Veritabanının fiziksel olarak oluşturulmasında erişimi yavaşlatan çok sayıda gösterici kullanılabilir. O nedenle veritabanınına erişim zamanı kütük okumaya göre fazladır.

Bu bölümde veritabanı yönetim sisteminin kavramsal ve fiziksel yapısı verilerin organizasyonu, veri akışı, sistemi oluşturan öğeler ve bu öğelerin işlevleri açısından ayrıntılarıyla ele alınmıştır.

Bunun yanısıra veritabanı yönetim sisteminin ortaya çıkışı, gelişmesi ve kurumsal yaşamda kazandığı önem ile planlanma süreci anlatılmıştır.

III. bölümde ise veritabanı yönetim sisteminin temelini oluşturan veri modelleri ve söz konusu modelleri örnek alan kimi sistemler, bu modellere işlerlik kazandıran işlem dilleri ile beraber tanıtılacaktır.

III. BÖLÜM
VERİTABANI ve YAPISI

3.1. VERİ DÜZEYLERİ

Veritabanı yönetim sistemleri için geliştirilen ve Amerikan Ulusal Standartlar Enstitüsü'nce bir standart olarak onaylanan üç şemalı model[1] veri bağımsızlığını gerçekleştirerek söz konusu sistemler için önemli avantajlar sağlamıştır.

Veri bağımsızlığı Doğaç'a göre[2] "veri modeli ve karşılığındaki veri altdili ile sağlanmaktadır. Veri modeli veritabanının kullanıcı tarafından görünümü veri altdili ise kullanıcının kendi çalışma alanı ile veri modeli arasında alış verişini sağlayan dildir."

Oysa Date[3] veri modelini kavramsal model karşılığında kullanmakta ve kavramsal modeli veritabanının "kullanıcı tarafından görünümü" olarak değil veritabanının içerdiği tüm bilginin temsil edildiği düzey olarak tanımlamaktadır.

Standartın onaylanmasından önce oluşturulmuş veritabanı yönetim sistemlerinde, verinin ikincil belleğe depolanması sırasında gözetilen düzen ve veriye erişim yolu, uygulama programlarının gereksinim-

---

1. ANSI/SPARC: Amerikan Ulusal Standartlar Enstitüsü/Standartlar, Planlama ve Gereksinimler Komitesi, 1970'li Yıllarda ANSI/X3/SPARC veritabanı yönetim sistemleri ve bunların arabirimleri için üç şemalı genel bir mimari yapı önermiştir. Taner M. Özsu, "Dağıtımlı VeriTabanı Sistemleri" Bilişim. 19-20, 1985, 32 s.
2. Asuman Doğaç, "Veri Temeli Sistemlerine Kısa Bir Bakış" Türkiye Bilişim Derneği Dergisi. 6:11, 1977, 92 s.
3. J.C. Date, An Introduction to Database Systems. Vol:1, 4th ed. Reading Massachusetts: Addison Wesley, 1985, 16 s.

leri doğrultusunda belirlenmekte idi. Dolayısıyla depolama yapısını ve erişim yolunu değiştirmeden uygulama programlarında herhangi bir değişiklik yapmak olanaksızdı. Bunun yanında veri bağımsızlığının bulunmadığı sistemlerde aynı verinin farklı uygulamalar için değişik görünümlerini elde etmek de olanaklı değildir.

Örneğin: A ve B gibi iki farklı uygulama programının her ikisinin de verilerini sağladıkları kütüklerde "sipariş edilen yayının fiyatı"na ilişkin ortak bir alanın bulunduğunu düşünelim. Ancak A programı yayın fiyatını ondalıklı sayı, B programı ise yayın fiyatını tam sayı biçiminde vermekte olsun. Yinelemenin önlenmesi bakımından her iki kütüğün de birleştirilmesi düşünüldüğünde ortaya hangi sayı sisteminin benimseneceği sorunu çıkmaktadır. Bu durumda programlardan birinin öngördüğü sayı sistemi benimsenirken diğer programın gereksinimine uygun dönüşümü sağlayacak bir süreç veritabanı yönetim sisteminde oluşturulur. Böylelikle verinin depolanış biçimi ile uygulama programındaki tanımı arasında fark bulunmasına rağmen her iki öğenin birbirini etkilemesi önlenmiş olur.

Veri bağımsızlığının bulunmadığı sistemlerde veritabanı yöneticisinin değişen gereksinimler (uygulama önceliklerinin değişmesi, yeni tip depolama gereçlerinin ortaya çıkması gibi) doğrultusunda uygulama programlarına dokunmadan depolama yapısı veya erişim stratejisinde değişiklik yapma özgürlüğü yoktur [4].

Yukarıda anılan sorunlar nedeniyle veritabanı yönetim sistemlerinde veri bağımsızlığının sağlanması temel amaç olmuştur.

---

4. Date, a.g.e., 9 s.

Veri bağımsızlığı fiziksel ve mantıksal veri bağımsızlığı olmak üzere iki kademede ele alınmaktadır.

1. **Fiziksel Veri Bağımsızlığı:** İç model tanımlamasının diğer iki düzeyden etkilenmeyecek biçimde olmasıdır. Örneğin: kütük düzenlemelerinde yapılacak değişikliklerin uygulama programlarını etkilememesidir.

2. **Mantıksal Veri Bağımsızlığı:** Kavramsal düzeyde yapılacak bir değişikliğin dış düzeyi etkilememesidir. Örneğin: Kavramsal düzeye yeni bir varlık eklenmesi ya da bir varlığı niteleyen özelliklere bir yenisinin eklenmesi sistemde önceden yeralmış kullanıcı görünümleri veya uygulama programlarında bir değişikliğe neden olmamalıdır [5].

ANSI/SPARC'da veritabanı yönetim sistemleri için belirlenen üç düzeyli mimari yapı ise şöyledir:

1. **Kavramsal Düzey:** Veritabanının orta düzeyidir. Veritabanında yeralan varlıklara ilişkin tüm veriler ve aralarındaki bağıntıları gösterir. Varlıklara ilişkin değişkenlerin biraraya getirilmesi ve aralarında kurulan ilişkiler doğrultusunda sınıflanmaları, kavram oluşumu ile benzerlik göstermektedir. Bu nedenle veritabanının mantıksal düzeyi olarak da adlandırılır. Mantıksal yapı kavramsal şemada tarif edilir. Şema, bunun yanında yetke denetimi ve doğruluk denetimi gibi özellikleri de içerir.

---

5. Helge Holvik, Databaseteori. (Oslo: Statens Bibliotek og Informasjonshogskole, 1991), 14 s.

2. **Dış Düzey:** Veritabanının en üst ve soyut düzeyidir. Kullanıcılar veritabanında temsil edilen tüm verilere değil, bir sorgulama dili (örneğin: SQL) aracılığı ile formüle ettikleri sorular ışığında yalnız kendi gereksinimlerini karşılayan verilere erişim sağlar. Böylelikle kullanıcılar veritabanının mantıksal yapısını daha kolay kavrar ve gereksiz veri yığını ile karşılaşmamış olur.

Bunun yanı sıra görünümler aracılığı ile erişim ve işlem yapma açısından getirilen sınırlılık, veri güvenliğini de sağlamaktadır.

Daniel Martin, "görünüm"ü tek ve çok düzeyli olmak üzere iki kategoriye ayırmaktadır.

"1. **Tek Düzeyli Görünüm:**

a) seleksiyon kullanarak sıraların alt kümesinden;

b) soyut özelliklerden (sıraların diğer özellikleri kullanılarak gerçekleştirilen aritmetik işlemler)

c) Özelliklerin alt kümesini oluşturan projeksiyondan türetilmektedir.

2. **Çok Düzeyli Görünüm:** Birden fazla ilişkiden türetilir. İlişki sayısı $İ_1$'den $İ_n$'e kadar istenilen sayıda olabilir. Görünüm N-1 adet birleşim ile gerçekleştirilir.

Verilen bir değere göre her D düzeyinde $İ_1$ gibi bir ilişkiye uygulanan S1 seleksiyonu sonucu sıraların alt kümesi ortaya çıkar. P1 projeksiyonu da $İ_1$ ilişkisinin özellikleri üzerine sınırlama getirir.

Daha sonra $İ_1$ ilişkisinin altkümesi biçiminde ortaya çıkan sıraların herbiri $İ_2$ ilişkisindeki karşılıkları ile birleşim kuralları uyarınca birleştirilir.

İkinci seleksiyon $İ_2$ ilişkisindeki kimi sıraları P_2 tarafından yapılan sınırlama doğrultusunda seçer ve ortaya çıkan ilişkiye enlemesine bazı soyut özellikler ilave edilir ve işlemler bu doğrultuda ilişki sayısına göre sürer gider [6].

Görünüm ile güncelleme de gerçekleştirilebilmektedir. Ancak güncellemenin yapılabilmesi iki koşula bağlıdır [7].

1. taban tablosunun tüm alanlarını içeriyor olması;
2. yalnız taban tabloları üzerine yapılıyor olması.

Veritabanında yapı değişikliği söz konusu olduğunda ya da veritabanının yeniden düzenlenmesi halinde görünüm kavramsal düzeyden etkilenebilir. Tabloya yeni bir alanın eklenmesi ile bir genişleme meydana gelmesi söz konusu olduğunda veritabanının yapısı değiştirilir[8]. Görünümün sağladığı avantajlar ise şöyle özetlenebilir:

"1. mantıksal veri bağımsızlığı;
2. veritabanının yetkili kişilerce kullanımına olanak verilerek veri güvenliğinin sağlanması,

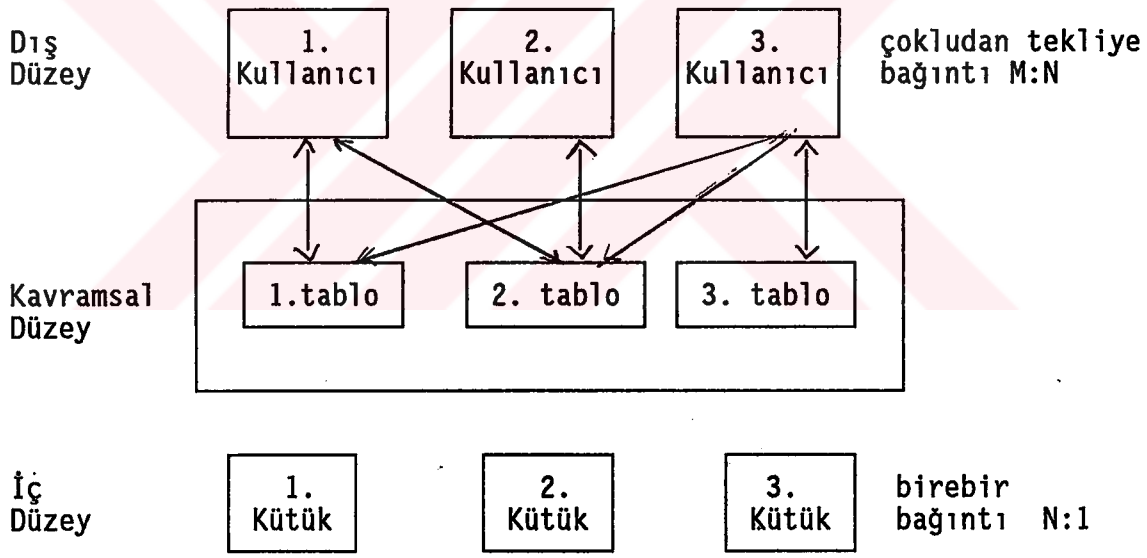
---

6. Daniel Martin, Advanced Database Techniques. London: MIT Press, 1986, 31 s.
7. Sitansu S. Mittra, Principles of Relational Database Systems. Englewood Cliffs, New Jersey: Prentice Hall, 1991, 60 s.
8. Mittra, a.g.e., 60 s.

3. kullanıcı isteğine göre verinin biçimlendirilebilmesi ve erişim"[9].

3. **İç Düzey:** Veritabanının fiziksel olarak temsil edildiği düzeydir. Kavramsal şemada yapılan veri tanımlamalarının veri yapılarının veri tanımlama dili aracılığı ile dış depolama alanı olan disk kütükleri üzerindeki gerçek yerlerinin gösterildiği düzeydir.

Söz konusu üç düzeyi ilişkisel model açısından bir şema ile gösterecek olursak:



Şekil 3.1: Veri düzeyleri

Mc Fadden kütük düzenlemelerini ve adresleme tekniklerini tanıtan bir düzeyin daha varlığını ortaya koymakta ve bunu fiziksel veri animasyonu olarak tanımlamaktadır [10].

---

9. Aynı, 61 s.
10. Fred R. Mc Fadden-Jeffrey A. Hoffer. Database Management. Menlo Park, Calif: The Benjamin Cummings Pub. Com., 1985, 50 s.

3.2. VERİ MODELLERİ VE ÖRNEK SİSTEMLER

Model, gerçek dünya nesneleri ve bunlar arasındaki bağlantıların bir temsilidir.

"Bir veri modeli ise organizasyonun içinde varlık, olay, etkinlik ve bunlara ait benzerliklerle ilgili verilerin soyut temsilidir "[11].

Veritabanı yönetim sistemlerinin temel dayanağı olan veri modeli, verinin organize edildiği yapı ve bu yapı üzerine gerçekleştirilen işlemler kümesini kapsar. Bir başka deyişle veritabanı yönetim sistemindeki mantıksal yapıyı tarif eder, anlamlılığı sağlar ve tutarlılık sınırlılıklarını belirler. Gereksinim duyulan verileri tanımlamak ve uygulama açısından verileri anlaşılır kılmak amacıyla oluşturulan veri modelleri genelde üç ana sınıf altında ele alınmaktadır.

"a) nesneye dayalı mantıksal modeller;
b) kayda dayalı mantıksal modeller;
c) fiziksel veri modelleri"[12].

a) **Nesneye Dayalı Mantıksal Modeller:** Veri ile ilgili açıklamalar kavramsal ve dış düzey olmak üzere iki düzeyde yapılır. Esnek bir yapıya sahip olup veri sınırlılıklarını tam olarak ifade ederler.

---

11. Aynı, 50 s. Nejat Tuğcu, "Veri Modellerinde Veri Tanım Dilleri" Ulusal Bilişim Kurultayı 2, 18-20,12, 1978 Ankara, Bilişim 78 Bildiriler. Ankara, Meteksan, 1978, 75 s.
12. Henry F. Korth-Abraham Silberschatz, Database System Concepts. Sangapore: Mc Graw Hill, 1986, 6 s.

Söz konusu türe örnek oluşturan ve yaygın olarak bilinen kimi modeller şöyledir:

varlık ilişki modeli;
nesne uyarlı model;
binear model;
anlamsal veri modeli;
infolojikal model;
fonksiyonel veri modeli [13].

b) **Kayda Dayalı Model:** Veriyi kavramsal ve dış düzeylerde açıklar. Ancak nesneye dayalı modele karşılık veritabanının tüm mantıksal yapısını belirler ve yürütmeye ilişkin yüksek düzeyde açıklamalar getirir. Söz konusu model, çeşitli tipteki sabit uzunluğa sahip kayıtlardan oluşur. Kayıtların sabit oranda alanlardan ve alanların sabit uzunluklardan meydana gelmesi veritabanının fiziksel düzeyindeki yürütmeyi kolaylaştırır. Bunların daha zengin yapıya sahip olanları fiziksel düzeyde değişken uzunluktaki kayıtlardan oluşanıdır. Bu türün en bilinen örnekleri ilişkisel, hiyerarşik ve ağ modelleridir[14]. Günümüzde tüm ticari sistemler söz konusu üç modeli temel aldığından bir sonraki kesimde bu modeller üzerinde durulacaktır.

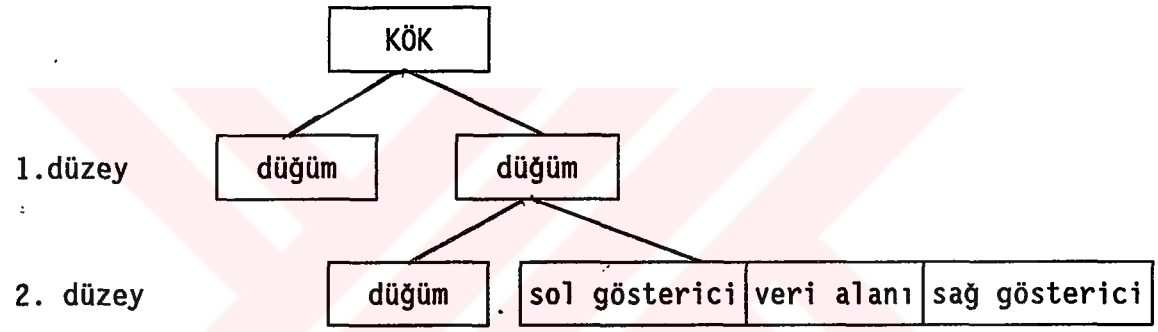
c) **Fiziksel Veri Modelleri:** Verinin en düşük düzeyde temsil edildiği modellerdir. Bu türde çok az örnek vardır. En yaygın olarak bilinen modeller, birleştirici model ve bellek çerçevesidir [15].

---

13. Aynı, 6 s.
14. Aynı, 8 s.
15. Aynı, 10 s.

3.2.1. Hiyerarşik Model

En eski veri modeli olmasına rağmen halen kullanılagelen hiyerarşik model "ağaç" veri yapısı üzerine kurulmuştur. "Ağaç veri yapısı", çok bağlantılı listenin özel bir türüdür. En üstte "kök" yeralır. Kökten çıkan sağ ve sol dalların ucunda "düğüm" bulunur. Düğüm, bir "veri alanı" ve "iki gösterici"[16] alanından oluşan bir kayıt tipidir.



Şekil 3.2: Ağaç veri yapısı

Kökten başlayacak en uç düğüme kadar giden ve düzeyleri de gösteren uzunluk, ağacın derinliğidir. Her düğüm kendinden sonra gelen ağaçlar için bir kök görevi yapar. Eğer bir düğümün alt ağaçları boş ise söz konusu düğüm "yaprak" olarak adlandırılır. "Eğer K1 gibi bir kayıt tipinden K2 gibi diğer bir kayıt tipine bir ilişki varsa K1 sahip, K2'de onun üyesi olur "[17].

Daha önce de belirttiğimiz gibi hiyerarşik modelde varlıklar kayıt tipi veri yapısına sahip düğümler olarak ifade edilir. Kayıt ti-

16. Gösterici: Dinamik bir veri yapısıdır. Veri nesnesinin bellekteki adresini gösterir. Yeni kayıt tipi eklenmesi ve silinmesi sırasında kaydın tam yerine yerleştirilmesi göstericiler ile son derece kolay olmaktadır. O nedenle listeye esneklik kazandırır.
17. Naphtali Rishe, Database Design Fundemantals; a structural Introduction to databases and a structured application design methodology. Englewood Cliffs, N.J.: Prentice Hall, 1988, 261 s.

pi, veri yapısının farklı veri yapılarından oluşmuş birçok değişkeni birarada ifade etme özelliği bulunmaktadır. Örneğin: Pascal programlama dilinde kayıt tipi şöyle ifade edilmektedir.

```
Type
Record type = record
              idlist : type1;
              idlist : type2;
              end;
```

Şekil 3.3: Kayıt tipi

Farklı veri yapılarında olmalarına karşılık tek bir varlığa ait birbiri ile ilgili veriler tek bir kayıt tipinde bütünlük kazanmıştır. Bunların aldığı değere ise "kayıt olgusu" denir. Kök dışındaki bütün düğümler bağımlı kayıt tipi olarak adlandırılmaktadır. Veriye erişim için genelde izlenen yol sırasıyla kök, sağ düğüm ve sol düğümdür [18].Öncelikle anahtar erişim tekniği ile köke erişilir. Doğru kökün bulunmasının ardından bağımlı kayıt tipine erişilir. Söz konusu özellik ilişkisel veri modeli ile kıyaslandığında hiyerarşik model için bir zayıflık göstergesidir. Ancak kullanıcının her zaman verinin yerini bilmesi söz konusu olamayacağından veriye erişim yolları önceden programcı tarafından belirlenmektedir. Kayıt tipleri ve bunlar arasındaki sahip-üye ilişkileri kavramsal şemada tanımlanır. Ancak veriye ulaşmak için erişim yollarının önceden belirlenmesi soruların anında oluşturulup veritabanına yöneltilemeyeceği anlamına gelir. Bir

18. Buna Inorder traverse tree denir. Ayrıca pre ve post order biçimleri de vardır.

başka deyişle erişim yolu bir ölçüde veritabanına yöneltilecek soruları da belirlemektedir.

İlişkisel veritabanındaki "görünüm"e karşılık hiyerarşik modelde "mantıksal veritabanı" bulunur ki burada kayıt tiplerinin bütün görünümleri yeralır. Mantıksal veritabanı oluşum süreci kısaca şöyle özetlenebilir.

1. Mantıksal veritabanı (MVT) ile fiziksel veritabanı (FVT) kökleri aynıdır;
2. MVT kayıt tipleri, FVT kayıt tiplerinin bir alt kümesidir.
3. Hiçbir sahip-üye ilişkisi bozulmadan yeni bir kayıt tipi daha önce yazılmış bir kayıt tipinin üyesi olarak herhangi bir noktada eklenemez;
4. FVT'ndaki kök hariç her kayıt tipi MVT'ndan çıkarılabilir. FVT'den bir kayıt çıkarıldığı zaman ona bağlı diğer bütün kayıt tipleri de çıkarılır [19].

3.2.1.1. Bir Hiyerarşik VeriTabanı Sistemi: IMS

Veritabanı yönetim sistemleri önceleri hiyerarşik veri modeli uyarınca oluşturulmuşlardır. Ancak zaman içerisinde veritabanı gereksinimleri daha iyi anlaşıldıkça çeşitli veri yapılarını kullanan yeni modeller geliştirilmiştir. Bununla beraber halen günümüzde hiyerarşik modelin kullanıldığı kuruluşlar bulunmaktadır.

---

19. Mittra, a.g.e., 174 s.

IBM'in IMS (Information Management System) ve INTEL'in system 2000'ni bugün yaygın biçimde kullanılan hiyerarşik veri sistemlerine örnek teşkil etmektedir.

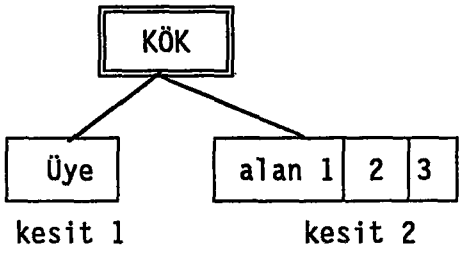
System 2000/80 temelde hiyerarşik bir yapıya sahip olmasına rağmen içerdiği bazı komutlar nedeniyle ilişkisel ve ağ modellerinin bazı özelliklerini taşımaktadır. Bu bakımdan bütünüyle hiyerarşik bir modele sahip olmadığı için burada IMS'nin tanıtımı tercih edilmiştir.

IMS, 1960'ların ortalarında uzay ve havacılık sanayiinin gereksinimlerine bir çözüm olarak ortaya çıkmıştır. IBM'nin dev bilgisayarında[20] kullanılabilen IMS'nin sonraki yıllarda birçok türü üretilmiştir. Sistemi geliştirenler doğadaki hiyerarşik yapıyı gözleyerek fiziksel olarak bu yapıyı oluşturmaya çalışmışlarsa da bu noktada başarı sağlayamamışlar ve çalışmalarını mantıksal hiyerarşik yapı oluşturma yönünde sürdürmüşlerdir [21].

IMS, fiziksel ve mantıksal olmak üzere iki temel olgu üzerine kurulmuştur. Buna göre fiziksel kısım, fiziksel veritabanı kaydı üzerine oturtulmuştur. Bu ana yapı ya da kayıt, hiyerarşik biçimde kök ve ona bağlı kesitlerden meydana gelmektedir. Söz konusu kesitler ise birbiri ile ilişkili alanlardan ibarettir. Bir şema ile göstermek gerekirse:

---

20. Dev bilgisayar: Dağıtılmış bir bilgisayar sisteminde merkezde bulunan oldukça yüksek işlem gücüne sahip ana bilgisayar ve ona bağlı çevre bilgisayarlardır. Sözcüğün taşıdığı bir diğer anlam, merkezi işlemci, büyük sistemlerin merkezi işlem birimi.
21. Mc Fadden, a.g.e., 386 s. Ethem Refi Kök, "Data Yığını Yönetim Sistemleri" İstanbul Teknik Üniversitesi Elektronik Hesap Bilimleri Enstitüsü, Elektronik Bilgi İşleme Ulusal Sempozyumu 2, 25-26, 4, 1977, İstanbul: İstanbul Teknik Üniversitesi, 1977, 170 s.



Şekil 3.4: Ağaç yapısındaki hiyerarşik model

Fiziksel veritabanı kayıt tipi, tüm kesit ve alanları ile birlikte **Veritabanı Nitelemesi** (DBD: Database description) bölümünde tanımlanır.

Veritabanı nitelemesi (DBD: Database Description)

İsim : Söz konusu veritabanının ismi (Örneğin sipariş veri tabanı için SIPVT gibi)

Kesit: (SEGM: Segment)

İsim : SIPVT'nda kaydın uzunluğu, düğümün niteliği (sahip ya da üye düğüm oluşu gibi)

Alan (Field)

İsim : alanın adı (örneğin yayın adı) gibi bilgilere yer verilmektedir.

Mantıksal veritabanı, kullanıcı gereksinimleri göz önünde bulundurularak oluşturulmuş görünümlerdir. Fiziksel olarak var olmayan ancak soyut biçimde nitelenen kısımdır. Burada kullanıcı gereksinimleri doğrultusunda **program iletişim bloğu** adı altında bir bölüm oluşturulur [22].

22. Aynı, 389, s.

Program iletişim bloğunda tip= veritabanı, adı = veri tabanı adı, anahtar uzunluğu gibi bilgiler yeralır.

Mantıksal veritabanında kesitler **duyarlı kesit** olarak adlandırılır. Duyarlı kesitin tanımlanmasında kullanıcının her kesit için yapabileceği işlemleri belirleyen bir bölüm yer alır. Bu yolla veri güvenliği ve bağımsızlığı sağlanabilmektedir.

Her kullanıcının bir ya da daha fazla iletişim bloğu olabilir. Yukarıda anılan bölümlerin tümü **program belirleme bloğu** (PBB) altında toplanır ve sistem kütüphanesi (veri kataloğu)'ne yüklenir. Kullanıcı programı işlerlik kazandığında kontrol programı PBB'yi sistem kütüphanesinden alarak uygulamaya koyar.

IMS'de dört tür erişim yöntemi kullanılmaktadır. Bunlar sırayısla;

A- Sıradüzen ardışık erişim yöntemi (Hierarchical Sequential Access Method: HSAM)

B- Sıradüzen dizinsel ardışık erişim yöntemi (Hierarchical Indexed Sequential Access Method: HISAM)

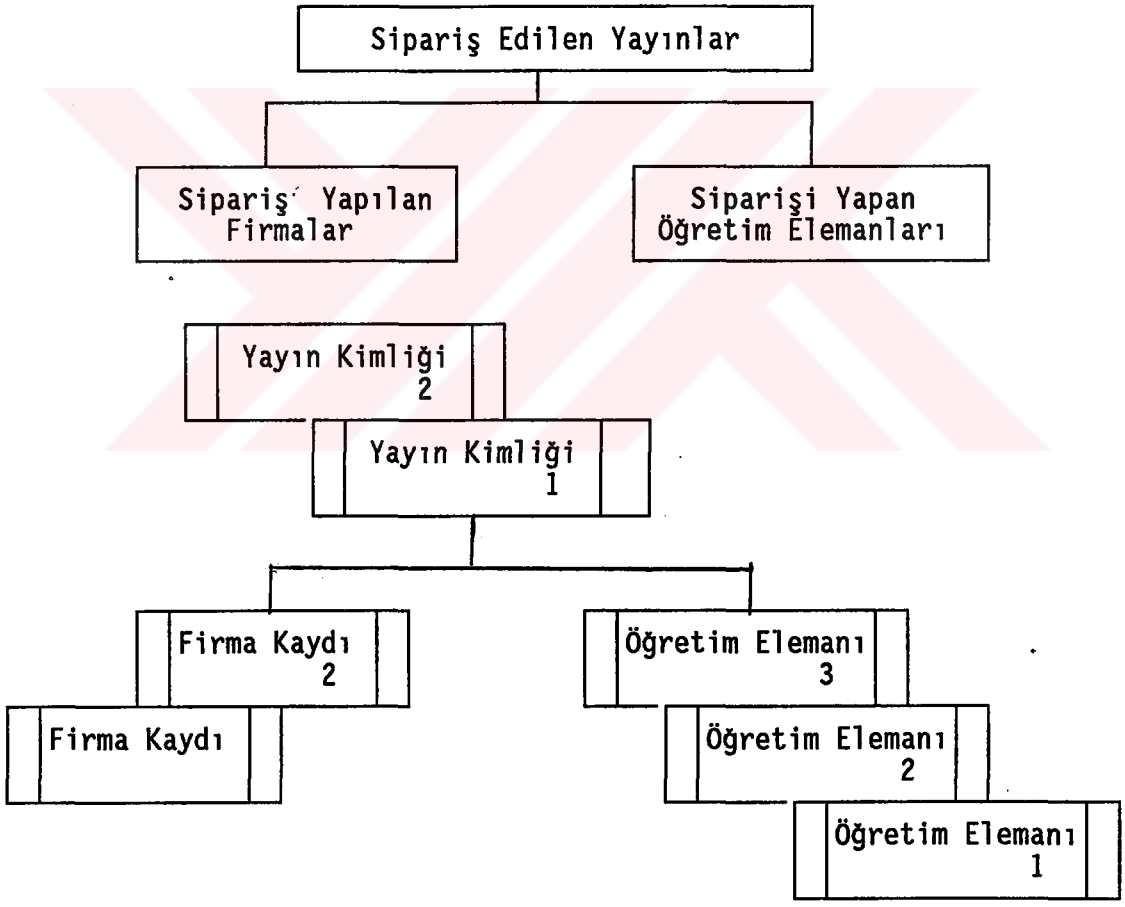
C- Sıradüzen doğrudan erişim yöntemi (Hierarchical Direct Access Method: HDAM)

D- Sıradüzen dizinsel doğrudan erişim yöntemi (Hierarchical Indexed Direct Method: HIDAM)

Söz konusu yöntemleri kısaca özetlemek gerekirse,

A- HSAM : Fiziksel veritabanı kaydındaki kesitler, yani kök ve ona bağlı diğer kesitler bir sıra izleyecek biçimde yerleştirilmişlerdir. En basit veri yapısına sahip olma avantajına rağmen yeni kesit ekleme veya eskisini silmede güçlük yaratır. Daha çok güncellemenin en az düzeyde olduğu arşiv kütükleri için kullanılmaktadır (23).

Hiyerarşik model temel alınarak oluşturulmuş basit bir sipariş veritabanı ve ona bağlı HSAM yapısı şöyledir.



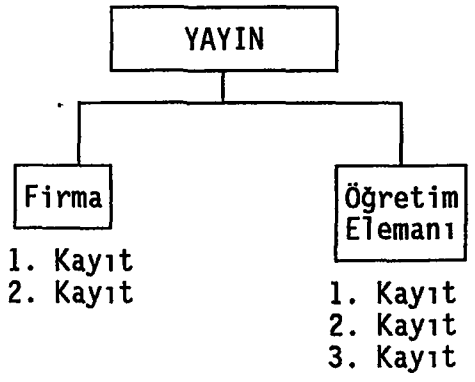
23. Aynı, 394, s.

Hiyerarşik yapıda temsil edilen mantıksal yapının fiziksel olarak kütükleri temsili ise aşağıdaki gibidir:

Yayın kimliği Kayıt 1	Yayın Kimliği Kayıt 2	Firma Kayıt 1	Firma Kayıt 2	Öğretim Elemanı Kayıt 1	Öğretim Elemanı Kayıt 2	Öğretim Elemanı Kayıt 3

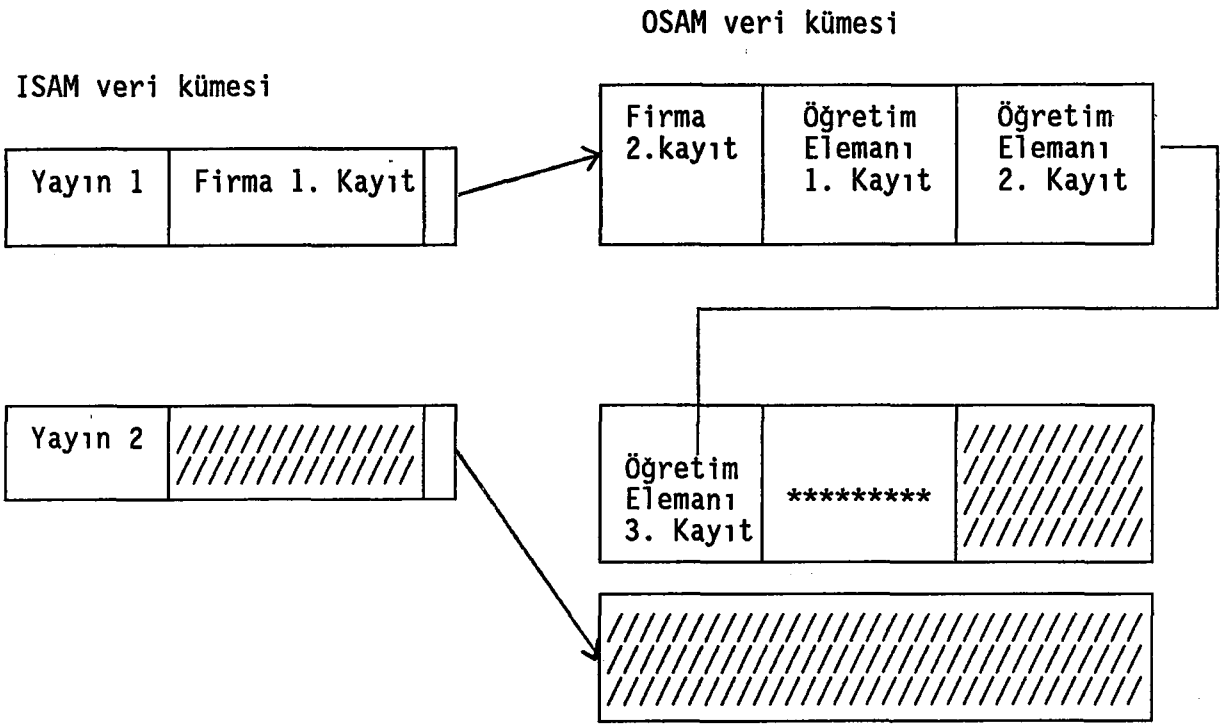
Şekil: 3.5 HSAM yapısı

B- HISAM: Hem bir sıra izleyerek hem de doğrudan erişim gerçekleştirilebilir. Doğrudan erişim için iki veri kümesi oluşturulur. Veritabanına veri yüklemesi yapılırken her kök kesit için bir ISAM (Indexed Sequential Access Method) kaydı oluşturulur. ISAM'da kayıt uzunluğu sabittir. O nedenle kayıt alanına sığmayan kesitler yine hiyerarşik düzende OSAM (Overflow Sequential Access Method)'a kaydedilir. Aralarındaki bağıntıyı göstermek için ISAM'daki kök kesitin son alanına yani göstericiye OSAM'daki kaydın yerini belirleyen adres bilgisi kaydedilir [24].



Şekil 3.6: HISAM'ın yapısı

24. Aynı, 385, s.



Şekil 3.7: HISAM'daki Düzenleme

C- HDAM: Kök kesite rastgele erişim yöntemi ile doğrudan ulaşır. Kök sabit uzunluktaki kayıt alanlarına yerleştirilir [25].Rastgele erişim algoritması kök kesit olgusunun bulunduğu yeri gösteren bir **bağıntı adresi** üretir. Kök kesite bağlı üye kesitlere de göstericiler aracılığı ile erişilir. Kök, bir kayıt anahtarı olduğunda rastgele erişim işlevi aşağıdaki gibi görüntülenebilir.

Şekil 3.8: Rastgele Erişim

25. Aynı, 396, s.

Rastgele erişim fonksiyonu iki ayrı kök kesit için aynı adresi ürettiği zaman ikinci kök yeterli depolama alanına sahip bir sonraki uygun kayda yerleştirilir ve birinci kök kesitin sonuna ikinci kökün yerini gösteren bir gönderme yapılır. Doğrudan erişim sağlandığı için erişim zamanı kısadır.

D- HIDAM: Kök kesite erişimi dizin ile sağlar. Kök ve ona bağlı kesitler HDAM'da olduğu gibi göstericiler ile birbirine bağlanmıştır. Hem ardışık hem rastgele erişim olanağı sağlanması bakımından en çok tercih edilen erişim yöntemidir.

IMS'de veri yönetimi Data Language/1 ile gerçekleştirilir. DL/1 Cobol ile birlikte kullanılan komutları içerir. DL/1'in işlem komutları ile kesit eklenmesi, silinmesi ve yer değiştirmesi yapılabilir. Bunun yanında doğrudan sırasal erişim sağlayan komutlar da bulunmaktadır.

Genel özellikleri ile tanıtmaya çalıştığımız IMS'ye daha sonra iki önemli özellik daha kazandırılmıştır. Birincisi, kesitin iki ayrı köke bağlı olarak mantıksal veritabanında ifade edilebilmesi diğeri ise ikincil dizindir. Ancak konumuz dışında olduğu için bu noktada ayrıntıya girilmeyecektir.

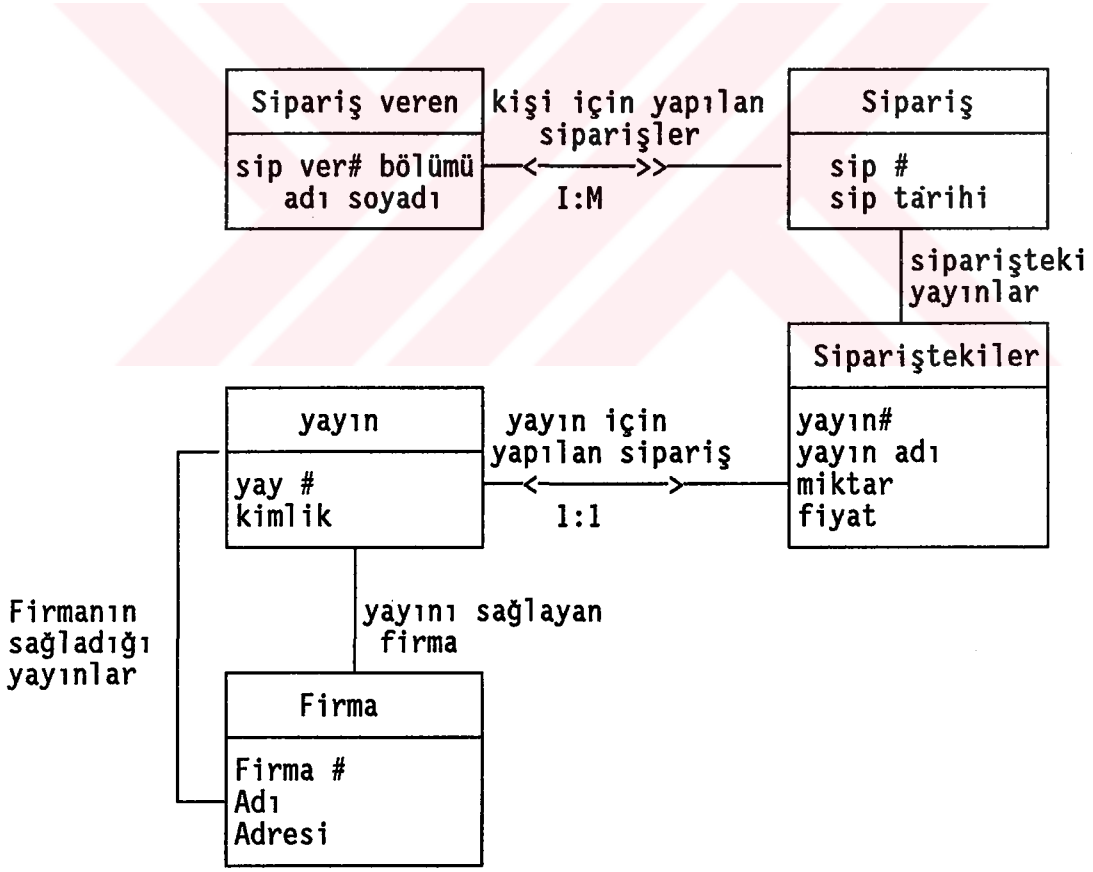
3.2.2. Ağ Veri Modeli

Model, veriyi bir kayıt kümesi ve ilgili kayıt tipleri arasındaki ilişkiler olarak ele almaktadır. Ağ modeli ile küme kavramı geti-

rilmiştir. Küme, sahipten üyeye doğru yönlenmiş bir ilişkinin ifadesidir. İki kayıt tipi arasında dinamik ve öncü olmak üzere istenilen sayıda küme tanımlanabilir.

Üç tip ağ modeli bulunmaktadır[26]:

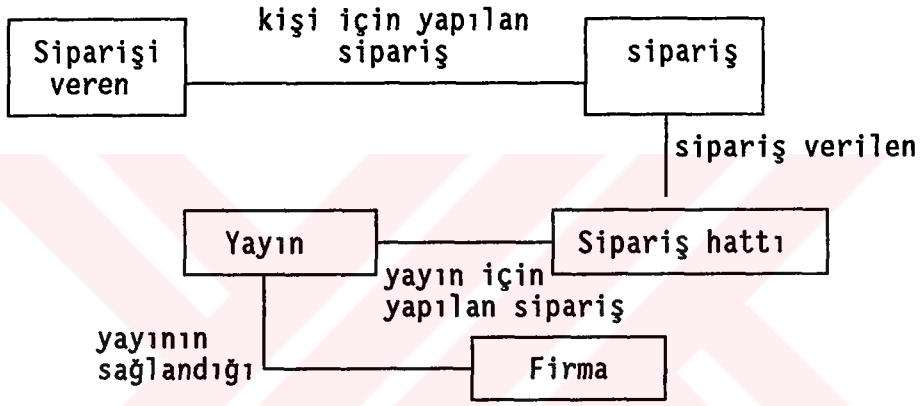
1. **Basit Ağ Modeli:** Codasyl Veritabanı Grubu'nun çalışmalarının etkisiyle en yaygın biçimde kullanılan tiptir. Çoklu ilişkiler (M:N)[27] bu tipte doğrudan veri tabanında tanımlanamamaktadır. Codosyl ağ veri modeli de bu yapının en çarpıcı örneğidir. Basit bir şema ile açıklanacak olursa:



Şekil 3.9: Basit Ağ Veri Modeli

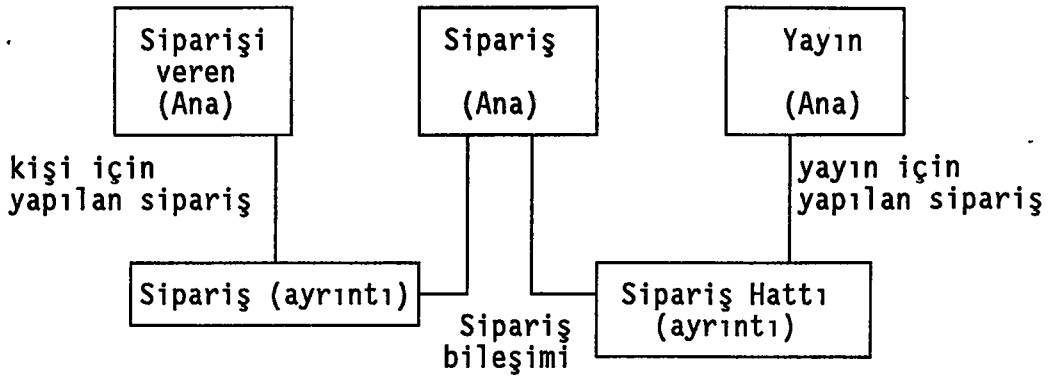
26. Aynı, 176, s.
27. Çokludan çokluya ilişkinin matematiksel ifadesidir.

2. **Karmaşık Ağ Veri Modeli:** İki kayıt tipi arasında çoklu ilişkiler (M:N) tanımlanabilmektedir. Bir üye kaydın birden fazla sahibi olma özelliği vardır. Bu amaçla bağlantı kayıt tipi kullanılır. Kesişim kayıt tipi kullanılmadığı zaman dairesel bağlantı sağlanır. Bu tür ilişki tanımının benimsendiği modellerde iki yönlü (M:N) ilişkisi tanımlanabilmektedir. Ancak bu da sistemde bütünlük sorunu yaratır. Az kullanılan bu tipin en belirgin örneği, Micro Database system III'tür.



Şekil 3.10: Karmaşık ağ veri modeli

3. **Sınırlı Ağ Veri Modeli:** Bütün kayıt tiplerini ana ve ayrıntı kayıt tipi olmak üzere ayrı birer kayıt kümesinde toplar. Bütün ilişkiler ana kayıt tipinden ayrıntı kayıt tipine doğru tanımlanmıştır. Veritabanına ana kayıt tipinden girilir ve rastgele erişim anahtarı aracılığı ile doğrudan ulaşım sağlanır. Hawlett Packard'ın image ve Total'ın VTS bu tipe örnek oluşturur.



Şekil 3.11: Sınırlı ağ veri modeli

Burada "sipariş" hem "siparişi veren"nin ayrıntı kaydı hem de "sipariş hattı"nın ana kaydı olamayacağından "sipariş ayrıntı kaydı" oluşturulmuştur.

Ağ veri modelinde ilişkilerin tanımlanması: Ağ modelinde ilişkiler genelde (I:M) ilişki olarak tanımlanır. Çoklu ilişkilerin doğrudan tanımlanması olanaklı değildir. İkiden fazla ilişkinin tanımlanması bağlantı veya kesişim kaydı aracılığı ile dolaylı olarak gerçekleştirilir.

Kütüklerarası ilişkiler açısından kavramsal ve dış düzeylerde "döngüsel" biçim izlenebilir. Yani aynı kayıt tipinden çıkan ve yine aynı kayıt tipine giren bir biçim söz konusudur. İç düzeyde ise "döngü" iki ayrı biçimde gösterilebilir:

"1.kayıt tipi içinde bir alanla ilgili olarak ikincil dizin oluşturulması;

2. başka bir kayıt tipi oluşturularak bundan eski kayıt tipine bir I:M ilişkisinin gerçekleştirilmesidir" (28).

28. Mc Fadden, age., 179.s.

Kütüklerarası döngünün bir başka biçimi kayıtların kütük içinde mantıksal bir sıra izlemek suretiyle birbirine bağlanmasıdır. Bu tür sıralama özellikle çıktı almada elverişlidir.

Ağ veri modelinin tüm türlerinde ve özellikle sınırlı ağ veri modelinde temsil edilebilen bir başka ilişki, "dairesel" ilişki tipidir. Kütüklerarasında erişim yolunu kısaltmasına rağmen veri yinelemesine yol açar. Ağ veri modelinde temsil edilebilen bir diğer ilişki tipi "sınıf" ilişkisidir. Söz konusu ilişki tipinde kayıt tipleri tek tek ele alınabildiği gibi bir küme içinde toplu olarak da düşünülebilir. Burada göz önünde bulundurulan nokta kullanıcının yaklaşımıdır.

Ağ veri modelinde "kavramsal düzey"in tanımlandığı "şema"da veriler ile ilgili şu bilgilere yer verilir.

"a) her kayıt tipi, isim ve alabileceği karakter sayısı belirlenerek tanımlanır;

b) her kayıt tipi içinde bulunan alanlar isim, veri tipi ve karakter sayısı olarak tanımlanır;

c) küme tipi, ismi, sahip mi, üye mi olduğu ve tipi belirtilmek suretiyle tanımlanır "[29].

Ağ veri modelinde dış düzeye karşılık gelen "alt şema"da alt küme tanımlamaları ise şöyle yapılır:

a) şemada kullanılan kayıt tipi;

b) şemada kullanılan kayıt tipleri içinde yeralan veri tipleri;

---

29. Mittra, a.g.e., 176.s.

c) kullanılan kayıt tiplerini birbirine bağlayan küme tipleri belirlenir [30].

Ağ modelinde erişim ve güncelleme için kullanılan komutlar bir başka şemada yeralır. Kullanıcının söz konusu yönetim komutlarından yararlanabilmesi için ilgili şemayı gözden geçirmesi gerekmektedir.

3.2.2.1 Codasyl Ağ Veri Modeli

Codasyl ağ veri modeli Veri Sistem Dilleri Konferansı'nda oluşturulan VeriTabanı Komitesi'nin çalışmaları sonucu ortaya çıkmıştır. Başlangıçta görevi, Cobol programlama dilinde yapılması düşünülen değişiklikleri ele almak olan komite daha sonra Cobol dilinin aynı anda birçok kütüğü kapsar biçimde verileri işleyebilmesini sağlayacak yönde geliştirilmesi için çalışmalarını yoğunlaştırmıştır.

Komite, ilki 1969 ve ikincisi gözden geçirilmiş olarak 1971'de[31] olmak üzere iki rapor yayınlamıştır. Söz konusu raporlarda veri işleme ve tanımlama dillerine ilişkin nitelemeler yapılırken raporun 1978 ve 1981'de yeniden gözden geçirilmiş basımlarında birkaç ağ veri modelinin genel özelliklerine değinilmiştir. Ağ modelinde veri tanımlaması "Şema"da veri tanımlama dili (VTD) aracılığı ile yapılır. Şemada kayıt tipleri ve kayıt tipleri arasındaki bağlar, sahip-üye ilişkileri tanımlanır ve şema ikincil depolama alanına yüklenir.

---

30. Aynı, 177 s.
31. Database Task Group of CODASYL Programming Language Committee, Report, Nisan 1971. Tamer M.Özsu-Behçet Sarıkaya-E. Atalay Özarahan, "Veri Tabanı Bilgisayarları İçin Yüksek Düzeyli Kullanım Dilleri" Ulusal Bilişim Kurultayı II, 18-20.12. 1978; Bilişim 78 Bildiriler. Ankara: Meteksan, 1978, 61-62 ss.

Şema'nın yapısı üç ana kesitten meydana gelmektedir [32].

1. **Kesit fiziksel depolama alanı:** ALAN (Area)dır. Burada veri değerleri bulunur. Veri bağımsızlığını sağlamak için raporun 1981 basımında bu bölüm kaldırılmıştır.

2. **Kesit,** bütün kayıt tipleri, kütükler ve veri içeriklerinin tanımlandığı kısımdır;

3. **Kesit,** kayıtlararası ilişkilerin yani kümelerin tanımlandığı kısımdır.

Şemayı tanımlayan veri tanımlama dili üç öğeden oluşmaktadır. Bunlar sırasıyla Location Mode, Set Selection ve Set Membership kalıplarıdır. Veritabanında varlıklar kayıt tipi olarak temsil edilmektedir. Kayıt tiplerinin aldıkları değerlere göre ortaya çıkan kayıt olguları ise veritabanının bütününü oluşturmaktadır. Kayıt olgularının disk üzerindeki fiziksel konumunu belirlemede Location mode[33] kalıbı kullanılır. Kayıt olgusunun fiziksel olarak bulunduğu yeri belirlemek için iki yöntem kullanılmaktadır.

1. CALC LOCATION MODE: Ana anahtar esas alınarak rastgele erişim tekniği ile kayda ulaşılması,

2. VIA LOCATION MODE: Kayda kayıtlar arasındaki ilişki aracılığı ile erişilmesi. Bu amaçla kayıtlar olabildiğince bağlı bulunduk-

---

32. Mc Fadden, age., 424 s.
33. Aynı, 429 s.

ları sahip kaydın yakınına yerleştirilmektedir. Location mode halen birçok ticari sistemde kullanılmakla beraber onun yerine daha genel ve mantıksal KEY IS[34] kalıbı getirilmiştir. KEY IS kalıbına göre her kayıt tipinin bir ya da daha fazla tek ya da bileşik ana ya da ikincil anahtarı bulunabilmektedir. Bu anahtarlar azalan ya da artan mantıksal bir düzenlemeye sahiptir. KEY IS kalıbı rastgele erişim veya dizinleme gibi bir anahtar erişim yöntemi yaratır. Kayıt tipini meydana getiren tüm alanlar şemada tanımlanmaktadır. Kayıt tipi alan içermiyorsa buna bağlantı kaydı adı verilir. Şemada yalnız kayıt tipinde yeralan alanların tanımlanması yapılmaz aynı zamanda veritabanındaki biçimleri de belirlenir. Veritabanının bir parçasını oluşturan söz konusu biçim kalıpları şöyledir [35]:

Picture : Veri elemanlarının ekrandaki görünüm biçimini tanımlar, sayısal veya sözel biçimleri destekler.

Type : Daha verimli, depolama biçimleri yaratmak için kullanılır. Depolamadaki uzunluk belirlemelerini yapar.

Occurs : Type ve Picture kalıpları ile birlikte kullanılabilir. Tekrarlanan veri gruplarını belirtmek için yararlanılır.

Codasyl ile birlikte soyut veri alanı kavramı ortaya atılmıştır. Soyut veri alanı kayıtta varmış gibi görünen ancak fiziksel olarak söz konusu kayıt olgusunda yeralmayan alandır. **Source** ve **Result** kalıpları da gerçek ve soyut veri alanlarını tanımlamak ve aralarındaki ayrımı belirlemek amacıyla kullanılmaktadır. **Source** kalıbı bir kay-

34. Aynı, 430 s.
35. Aynı, 431 s.

dın herhangi bir veri alanı tanımlaması ile birlikte kullanıldığında o veri alanına ait değerin, kaydın küme içinde bağlı olduğu sahip kaydın belli bir alanı ile aynı değere sahip olduğunu göstermektedir. Bu bakımdan source kalıbının veritabanında bütünlüğü sağlayıcı bir rolü vardır.

Result kalıbı ise gerçek ve soyut alanları belirleyen Actual ve Virtuel kalıpları ile birlikte kullanılmaktadır. Söz konusu kalıp aracığılı ile bir kaydın sahip olduğu alanlar kullanılarak yeni bir alan türetilebilmekte ya da hesaplanabilmektedir. **Result** ve **Actual** kalıpları birlikte kullanıldığında türetilen değer kayıt içinde oluşturulup yine kayıt içinde bir alana depolanır. Kalıp **Virtuel** kalıbı ile birlikte kullanıldığında ise yeni alana ilişkin değerler her kayıt olgusuna erişildikçe türetilir ya da hesaplanır. Kayıt olgusu içinde yeralıyormuş gibi görünse de aslında yalnız kullanıcının çalışma alanında yeralır.

Yukarıda anılan bütün bu kalıpların yanısıra Codasyl ağ veri modeli yönetim sisteminde **Check** ve **Coding** adlı iki kalıp daha kullanılmaktadır (36).

Check kalıbı, yeni bir veri değeri girildiğinde ya da değer değiştirildiğinde geçerlilik/doğruluk denetimi yapar.

Coding kalıbı ise veri değerinin kodlanması veya deşifresinde ve ne yapılması gerekliliği konusunda veritabanını bilgilendirir. İşlevi, yer kazanmaktır.

---

36. Aynı, 433.s.

Codasyl ağ veri modelinde ilişkiler, küme biçiminde tanımlanmaktadır [37].

Daha önce de belirttiğimiz gibi Codasyl ağ veri modeli ile küme kavramı geliştirilmiştir. Küme, sahip kayıt tipinden bir ya da daha fazla üye kayıt tipine doğru yönlendirilmiş bir ilişkidir. Codasyl modelinde kümeler I:M ilişkisini tanımlayabilmekte ancak M:N ilişkisini doğrudan tanımlayamamaktadır.

Veri tanımlama şemasında I:M ilişki, döngüsel olarak ifade edilebilir. Tek bir kayıt tipi söz konusu olduğunda sahip de üye de aynı kayıt tipinden ibarettir. Döngü, aynı kayıt tipi üzerinde gerçekleşir. I:M ilişkisi, iki kayıt tipi kullanılarak ya da ikincil anahtar ile ifade edilebilir.

Küme tek ya da birden fazla kayda sahip olabilmektedir. Tek tip kayıt söz konusu olduğunda o kayıt tipine ilişkin tüm olgular ortak bir **sahip** altında sınıflanmaktadır. Bu amaçla Codasyl'de **ORDER IS SORTED** kalıbı kullanılmaktadır. Küme içinde birden fazla kayıt tipinin bulunması halinde ise bir sahip kayıt tipi altında farklı tipte birçok üye kayıt sınıflanabilmektedir. Her küme içinde bir ya da daha fazla üye bulunabilmektedir. Söz konusu üye kayıtlar belirlenen bir özellik (kayıt ismi veya anahtar) uyarınca sınıflanabilmektedir.

Kayıt tipleri arasında birden fazla ilişki tanımlanabilir. Ancak Codasyl ağ modeli basit ağ modeli tipinde olduğu için M:N ilişki-

---

37. Aynı, 434 s.

lerin doğrudan veritabanında ifadesi olanaklı değildir. Bu amaçla "bağlantı kayıt tipi" kullanılır. **Bağlantı kayıt tipine** ilişkin kayıt olgusu yoktur. Ancak yine M:N ilişkiyi ifade edebilmek amacıyla kullanılan bir diğer kayıt tipi yani "kesişim" kayıt tipinde, ortak özellikler gözönünde bulundurularak oluşturulduğu için kayıt olgusu yer almaktadır.

Codasyl ağ veri modelinde kümeler, daha önce de belirtildiği üzere **dinamik** ve **öncü** olarak iki biçimde nitelenir(38). **Dinamik**, kümenin üye kayıt tipinin bulunmadığı ancak üye kayıt tiplerinin belirlenen sahip kayıt tipine bağlanabileceği anlamına gelir. **Öncü** kümede veritabanı yönetim sistemi küme içinde iki yönlü hareket olanağı yaratır. Bir üye kaydın girdisi yapıldıktan sonra o kaydın bağlanacağı sahip kayıt belirlenirken **SET SELECTION** kalıbı kullanılır. Söz konusu kalıp veritabanının şematik denetiminin sağlanmasında ve bütünlük sınırlılıklarının saptanmasında önemli rol oynar.

SEARCH KEY IS kalıbı ise küme içinde sahip olgudan üye kayıt olgularına belirli bir anahtar mantığı ile erişim olanağı yaratır. Sahip kayıtta oluşturulan işlevsel bir anahtar dizin aracılığı ile tüm üye kayıtlara gönderme yapılır.

Veritabanına ilişkin tüm nitelemeleri içeren **Şema**'dan birbirinden farklı gereksinmeleri olan kullanıcıların tümüyle yararlanması söz konusu değildir. Bu bakımdan kullanıcıların kendi gereksinimleri doğrultusunda veritabanı görünümleri elde edilebilmesi amacıyla **alt şemalar** oluşturulur. Bunun yanında veri güvenliğinin sağlanması ya da

---

38. Aynı, 440.s.

herhangi bir kullanıcı hatası yüzünden veritabanının zarar görmemesi amacıyla **altşema** da kullanıcının etkinliğini sınırlandırmak olanaklıdır. **Altşema** aynı zamanda bir altküme niteliğine sahiptir ve bu özelliği nedeniyle veri bağımsızlığını sağlar. **Altşema** bir program aracılığı ile kullanıldığı için şema tanımlama dili kullanılan programlama diline bağımlıdır. Ancak **altşema** tanımlaması uygulama programından bağımsız olarak **altşema kütüphanesinde**[39] yapılabilmektedir.

Codasyl ağ veri modelinde Cobol veri yönetim komutları kullanılmaktadır. Mc Fadden söz konusu komutları erişim, niteleme ve denetim olmak üzere üç katagoride sınıflamakta ve işlevlerini bir tablo halinde açıklamaktadır [40].

Tablo 3.1:

ERİŞİM	
FİND	Kaydı veritabanına yerleştirir
GET	Kaydı çalışma alanına ulaştırır
OBTAIN	Önceki iki komutu birleştirir

Yukarıda anılan tüm komutlar aracılığı ile tek kayda, duplike kayda, küme içinde önceki ve sonraki kayıtlara ve üye kaydın sahibine erişme olanağı vardır.

---

39. Uygulama programının gereksindiği verilerin veri bağımsızlığını sağlamak amacıyla tanımlandığı yer. İlişkisel modeldeki veri kataloğuna karşılık gelen şema.
40. Mc Fadden, a.g.e., 455 s.

Tablo 3.2:

NİTELEME	
STORE	Yeni bir kaydı veritabanına girer Otomatik olarak üyesi bulunduğu kayda bağlar
MODIFY	Kayda ilişkin mevcut değerleri değiştirir
CONNECT	Bir küme olgusuna mevcut üye kaydı bağlar
DISCONNECT	Küme olgusundan üye kaydın ayrılmasını sağlar
RECONNECT	Önce kaydın bağlantısını koparır sonra onu yeni bir küme olgusuna bağlar
ERASE	Veritabanından kaydı siler.

Tablo 3.3: 1,2 ve 3. tablolar cobol veri yönetim dili komutlarını ve işlevlerini göstermektedir.

DENETİM	
COMMIT	Bu komutun işlediği andan itibaren güncellemeyi yapar.
ROLLBACK	Commit komutundan sonra yapılmış güncellemelerin ardından yine veritabanını commit komutu için hazırhale getirir.
KEEP	Veritabanı kayıtlarına güncel erişim denetimleri yapar.

Cobol veri yönetim dili yanı sıra ağ veri modeline dayalı veritabanları için geliştirilmiş doğal dil ve sorgulama dilleri bulunmaktadır. Bunlardan Cullinet'in geliştirdiği **OnLine Query: OLQ** (Çevrimci Sorgulama) kullanıcı sorusunu anında yanıtlama becerisine sahiptir. Yine Cullinet tarafından geliştirilen doğal dil işlemcisi **OnLine**

English: OLE (Çevrimiçi İngilizce) günlük dilin sözcükleri ile ifade edilmiş bir soruyu analiz etme ve uygun veriye erişebilme olanağı sağlar[41]. Ancak bu tür dillerde erişim ve rapor üretme

Ancak bu tür dillerde erişim ve rapor üretme olanakları sınırlıdır. Genelde güncelleme yoktur. Ayrıca sorgulama dilinde yöneltilen cümlelerin analizi ve makina komutlarına dönüştürülmesi süreci zaman aldığından yöntemsel dilin yanında yavaş kalmaktadır.

3.2.3. İlişkisel Model

İlişkisel model 1970'de Dr. E.E. Codd tarafından geniş paylaşımlı veri bankaları için önerilmiştir. Geniş çaplı veri yığınlarını işleme olanağı vermesi nedeniyle ilgi görmüştür. Matematiksel ilişkiler kuramına dayanmaktadır. Söz konusu kuramda ilişki şöyle tanımlanmaktadır. D_1, D_2... Dn kümeleri verildiğinde (bu kümelerin ayrı ayrı olmaları gerekmez) eğer R d_1'in D_1'e, d_2'nin D_2'ye ... dn'nin Dn'e ait olduğu < d_1,d_2....dn> sıralanmış n'li elemanların bir kümesi ise R bu n'li küme üzerinde tanımlanmış bir ilişkidir. n değerine R'nin derecesi denmektedir [42].

1970'li yıllar boyunca kuramsal gelişimini sürdüren modelin ilk uygulamaya konması IBM firması tarafından üretilen Sistem R ile gerçekleşmiştir.

---

41. Aynı, 463. s.
42. Doğaç, a.g.m., 92.s. Nejat Tuğcu, "Veri Modellerinde Veri Tanım Dilleri" Ulusal Bilişim Kurultayı II, 18-20,12 1978, Ankara; Bilişim 78, Bildiriler. (Ankara, Meteksan, 1978), 76.s.

1980'li yıllarda dev ve mikro bilgisayarlarda kullanılmak üzere ticari sistemlerin de üretimi başlamıştır. Söz konusu sistemlere örnek vermek gerekirse:

Tablo 3.4: İlişkisel veritabanı yönetim sistemleri

Paket Program	Satıcı Firma
dBase II	Ashton Tate
SQL/DS	IBM
Oracle	Relations Software
Ingres	Relational Technology
RIM	Boeing Commercial Aeroplane Company
Knowledge Man	Micro Database Systems
Query By Example	IBM
ADABAS	Software AG

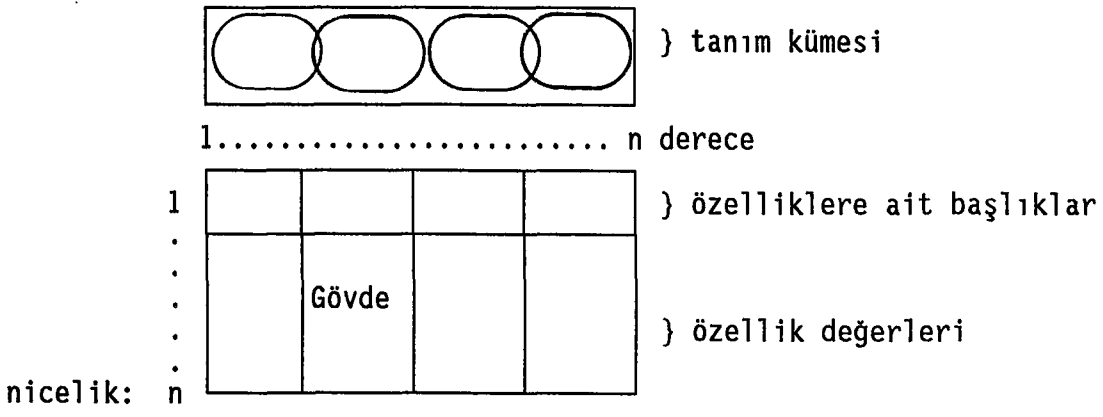
İlişkisel modelin yapısını tablolar oluşturmaktadır. Tablolar ya da bir başka deyişle ilişkiler farklı tipteki birçok varlığın temsilcisidir. Burada varlık kişi, yer ve eylemi ifade etmek için kullanılmaktadır. Varlık ve onun tanımlayıcı özelliklerini iki boyutlu matrisler halinde biraraya getiren tablolarda sıra ve sütunların belirli bir düzeni yoktur. Bu özelliğinden ötürü ilişkisel modelin tabloları genel anlamıyla bilinen tablolardan ayrılır.

Tabloların sütunları varlığa ilişkin özellikleri ifade eden değişkenlerdir. Bu değişkenlerin aldıkları değerler parçalanamazlar.

Sütunları meydana getiren değişkenler için bir değer aralığı tanımlanır ve bu değer aralığı söz konusu değişken için tanım kümesini belirler. Tablolarda herbir özelliğin ayrı tek bir adı vardır. Küme tanımı uyarınca yinelenemezler ancak değişken adlar altında aynı tanım kümesine sahip olabilirler. Tabloda bir varlığın tanımlayıcısı olan özelliklerin sayısı (1....n) o tablonun ya da ilişkinin derecesini belirler.

İlişkisel modelde bir varlığın temsilcisi olan tablonun herbir sırası varlık kümesinin elemanlarını oluşturur. Herbir sıra bir kaydın özellikler kümesi biçiminde ifadesini sağlar. Sıraların belirli bir sıralama uyarınca düzenlenme zorunluluğu yoktur. Tablodaki sıra sayısı ilişkinin nicelik belirleyicisidir. Sıra kümesi zamana bağımlı olarak değişebilir. Örneğin güncelleme sonucu yeni bir sıranın eklenmesi ya da silinmesi gibi [43].

İlişkisel modelde tablonun yapısını bir şema ile şöyle gösterebiliriz [44]:



Şekil 3.12: İlişkisel veri modelinde tablo yapısı

43. Date, a.g.e., 22.s.
44. Halvik, a.g.e., 24.s.

Date, ilişkisel modelin yapısını oluşturan tüm kavramların tanımlarını verdikten sonra ilişkisel modele ilişkin olarak ortaya çıkan özellikleri şöyle özetler [45].

1. bir ilişkide hiçbir zaman hiçbir iki sıra birbirinin aynı olamaz. İlişkinin bir küme olduğu gerçeğinden yola çıkılırsa kümede eleman tekrarı yoktur;

2. ilişkide sıraların bir düzeni yoktur. Çünkü sıralar da bir kümenin elemanlarıdır ve küme içinde elemanlar bir düzene bağlı değildir;

3. ilişkide özellikler düzenli değildir. Çünkü özellikler de birer küme elamanıdır;

4. bütün özellik değerleri parçalanamaz bir niteliğe sahiptir. Yani bütün ilişkiler normal biçimdedir. (en azından birinci normal biçimdedir). Daha yüksek düzeyli normal biçimler ilişkisel cebir yolu ile sağlanır.

İlişkisel modelde ana anahtar tek başına bir varlığı tanımlayabilen bir **özelliğe** karşılık gelir ve tek ya da bileşik olan bir yapıya sahiptir. Yani ana anahtar tek bir **özellikten** veya bir grup **özellikten** meydana gelen bileşik bir yapıya sahip olabilir. Ana anahtar aday anahtarlar arasından belirlenir. Bir ilişkiye ilişkin özellikler kümesi elemanlarının aday anahtar olabilmesi zamana bağlı olmayan şu iki koşulu gerektirir [46].

1. teklik: ilişkideki iki ayrı sıra hiçbir özellik için aynı değere sahip olamaz;

---

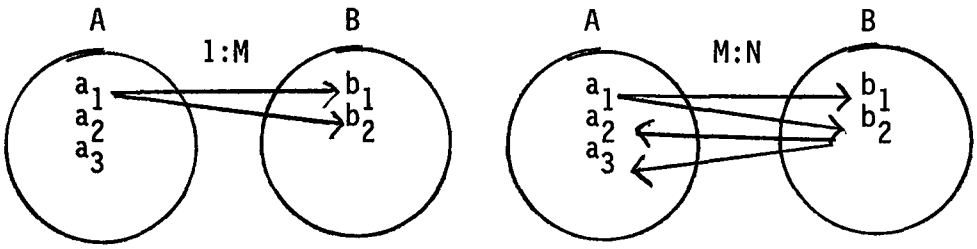
45. Date, a.g.e., s.24.s.
46. Aynı, 25.s.

2. minimalite: özellikler kümesinin hiçbir elemanı teklik niteliği bozulmadan kümeden çıkarılamaz.

Ana anahtarın belirlenmesinin ardından geride kalan diğer aday anahtarlar **alternatif anahtar** durumuna gelir. Anahtarlararası bağımlılığa ise fonksiyonel bağımlılık denmektedir. Buna göre ana anahtar dışındaki tüm özellikler ana anahtarın tümüne ya da bir bölümüne bağımlı olmalı ve ana anahtar aracılığı ile belirlenebilmelidir.

İlişkiler/tablolar arası bağıntı ortak özellikler veya bir tablodaki ana anahtara karşılık diğer bir tablodaki **ilişki anahtarı** yardımıyla sağlanabilmektedir. Bu durumda söz konusu anahtarlar **mantıksal gösterici** işlevini yerine getirir. İlişkisel model işte bu özelliği ile ağ modelindeki **gösterici veri yapısının** dezavantajlarını ortadan kaldırır. Çünkü bu durumda kullanıcı için veriye ulaşmada erişim yolunu bilme zorunluluğu yoktur.

İlişkisel modelde bağıntılar 1:M ve M:N biçiminde ifade edilebilmektedir. Bu durumda A gibi bir varlık kümesinin herbir elemanının B gibi diğer bir varlık kümesinde birden fazla görünümü bulunuyor ise bu ilişki tekliden çokluya yani 1:M ilişkisidir. Buna karşılık A kümesinin herbir elemanı için birçok B elemanı varsa herbir B elemanı için yine birçok A elemanı bulunuyor ise bu ilişki, çokludan çokluya yani M:N ilişkisidir. Bir şema ile aşağıdaki gibi ifade etmek olanaklıdır.



Veritabanındaki bir sütun değeri bilinmiyor ise veya henüz saptanmamış ise buna sıfır değeri[47] denir.

Tabloda meydana gelen bu tür boşluklar özellikle matematiksel hesaplarda yanlış sonuçların ortaya çıkmasına neden olur. Diğer yandan bir sütun değeri sayısal anlamda sıfır ise **birleşim** işlemine göre o değere karşılık gelen diğer bir tablonun sütununda da **boş** anlamında sıfır değeri bulunuyor ise tabloların birleştirilmesinde yine sorun yaşanabilir. Verilen bir koşulun doğruluğunun saptanmasında da bu tür değerler önem taşır. Çünkü doğruluk için ya her iki koşulun da doğru olması ya da "OR" işlemcisi kullanılarak bir tarafın durumunun gözönünde bulundurulması gerekir. Bu amaçla sistemde önceden bir doğruluk tablosu oluşturulur.

Sıfır değerinin sorun yaratma nedeni bilinmezliktir. Bu durumun çözüme kavuşturulması amacıyla sıfır değeri alamayacak sütunların önceden saptaması yapılır ve bunun için SQL'de bulunan **NOT NULL** kalıbı kullanılır. Örneğin ana anahtarın sıfır değeri alması söz konusu değildir ve **NOT NULL** kalıbı bunun belirtilmesini sağlar.

---

47. Philip Pratt-Joseph Adamski, Database Systems; Management and design. (Boston: South Western Pub. Com. 1991), 146.s.

Gerçekte fiziksel olarak bulunmayan ancak verilerin mantıksal birleşimi sonucunda ortaya çıkan soyut tablolar ilişkisel modeldeki "görünüm tabloları"dır. Bir kullanıcı sorusu aracılığı ile veritabanına açılan pencereden görünüm olarak da nitelendirilebilecek **görünüm tabloları** geçici tablolardır. Bu bakımdan da taban tablolarından farklıdır.

Görünüm tabloları kullanıcı sorusunun yöneltilmesi ile biçimlenir. Ancak aynı tür soru sıkça yinelendiğinde taban tablosu olarak tanımlanabilir. Görünümün veritabanı içinde sağladığı yararları erişim ve günleme açısından iki katagoride ele almak olanaklıdır. Buna göre **erişim açısından** sağladığı avantajlar şöyledir.

1. Kavramsal şema ile dış şema arasında veri bağımsızlığını sağlar. Kavramsal şemaya veri eklendiğinde ya da çıkarıldığında veya ilişkiler yeniden tanımlandığında görünüm tabloları bu durumdan etkilenmez ve mevcut veriler ile yeniden tanımlanabilir. Ancak kullanıcının aynı görünüm tablosuna önceki biçimiyle ulaşabilmesini sağlamak amacıyla yöneltilen soruyu yeniden tanımlamak gerekir;

2. Görünüm, kullanıcıların farklı gereksinimleri sonucunda ortaya çıktığı için aynı verilerin çeşitli bileşimlerini elde etme olanağı verir;

3. Kullanıcının veritabanını kavraması son derece kolaylaşır. Çünkü görünüm tabloları veritabanının dikey altkümesini verdiği için kullanıcı tüm veritabanına oranla daha az veri miktarı ile karşı kar-

şıyadır. Bunun yanında görünüm tablosundaki veriler birden fazla tablodan sağlanmasına rağmen tek bir tablo olarak kullanıcının önüne gelir.

4. Veritabanının güvenliği söz konusu olduğunda kullanıcının kimi sütunlara erişimi önlenebilir. Getirilen sınırlama nedeniyle veriler görünüm tablolarında yer almayacağından veri güvenliği de sağlanmış olur.

Görünümün güncelleme açısından sağladığı avantajları, görünüm tablosunun yapısını da gözönünde bulundurarak şöyle sıralayabiliriz:

Sıra ve sütunların altkümesi biçimindeki bir görünüm tablosu üzerinde güncelleme yapabilme görünüm tablosunun taban tablosu ile aynı anahtara sahip olma koşuluna bağlıdır. Bu tür görünüm tablosuna yeni bir sıra eklenirken sistemin taban tablosunda olup da görünüm tablosunda yer almayan sütunlar için hangi değerin girilmesi gerektiği belirlenmelidir. Böyle bir durum söz konusu olduğu zaman görünüm tablosunda yeralmayan taban tablosu sütunları sıfır (null) değeri alır.

Şema ile örneklersek:

Tablo 3.5:

Taban Tablosu

S1	S2	S3	S4	S5	S6	S7
	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0		0 0 0 0 0	

Görünüm Tablosu

S1	S5	S7

Görünüm tablosunda yeralmayan bir sıra girilmek istendiğinde aynı sıra taban tablosunda da bulunuyor ise sistem girdi yapmayı reddeder. Bunun yanında yapılan bir değişiklik ya da silme işlemi taban tablosunda da gerçekleşir.

İlişkisel modelde erişim yolunu oluşturan dizinler ayrı birer kütük olup iki sütundan meydana gelmiştir. İlk sütunda bir özellik ikincisinde kaydın veritabanı içerisindeki yer numarası bulunur. Amaç, belirli bir koşula uygun olarak bir ya da daha fazla sütuna erişmek olan dizinlerde o koşulu doğrulayan bir ya da birden fazla kayıt bulunabilmektedir.

Dış depolama alanında yeralan ve sistemle ilgili tüm bilgileri içeren "sistem kataloğu"nda tablolar ile ilgili olarak şu bilgiler bulunmaktadır (48):

a) tablo adı;
b) tabloyu oluşturanın adı;
c) sütun sayısı.

Sütunlar için;

a) sütun adı;
b) tablo adı;
c) sütun tipi yani veri tipi.

Veritabanında oluşturulacak dizinler için;

a) sütun adı;
b) tablo adı.

---

48. Date, a.g.e., 168.s.

İlişkisel modelin mimarı olarak Codd, verinin anlamını daha iyi yakalayabilmek için **veri bütünlüğüne** ilişkin iki kural ortaya koymuştur.

1. **Varlık Bütünlüğü:** Ana anahtar ilişkisel modelde bir sırayı tek başına tanımlamaya yeterli olduğundan **sıfır değeri** alamaz. Böylelikle her varlığın bir kimliği olması özelliğini güvence altına almış olur [49].

Date, varlık bütünlüğü kuralının geçerliliğine açıklık getirmektedir;

a. taban ilişkileri gerçek dünyadaki varlıklara karşılık gelmektedir;

b. varlıklar, tanımları uyarınca gerçek dünyada birbirlerinden ayırd edilebilmektedir. Yani kendi türlerinde tek bir kimliğe sahiptir;

c. ana anahtar, ilişkisel modelde tek bir kimliğe sahip olma işlevini yerine getirmektedir;

d. böylelikle ana anahtar değerinin **sıfır** olması bu anlamda bir çelişki yaratacaktır. Yani kimliği olmayan bir varlığın bulunması söz konusu değildir [50].

2. **Göndermede Bütünlük:** İlişkilerarası bağıntının ortak özellik uyarınca sağlanması esasına dayanır. Bu amaçla bir tablodaki ana anahtar değeri diğer tablodaki ilişki anahtarı değerine eşit olmalı ya da ilişki anahtar konumundaki her özellik değeri **sıfır** olmalıdır.

---

49. E.F. Codd, The Relational Model for Database Management. Versiyon:2 Reading Massachuset: Addison Wesley, 1990, 177 s.
50. Date, a.g.e., 8.s.

Date konuyu şöyle formüle eder: I_1 ve I_2 birer ilişki/tablodur. I_1'in ana anahtarı AA ve I_2'nin ilişki anahtarı Y_A'dır. Yani I_1 (AA) ve I_2 (YA)'dır. Bu durumda AA ve YA değerlerinin birbirine eşit ya da sıfır olması zorunludur.

İlişkisel modelin yaratıcısı E.F. Codd ve model üzerine önemli katkıları olan J.C. Date ilişkisel modele göre oluşturulmuş sistemlerin model açısından değerlendirmek üzere bazı ölçütler belirlemişlerdir. Codd'un somutlaştırdığı 13 ölçüt şöyledir [51]:

1. **İlişkisel Beceriler:** İlişkisel bir veritabanı yönetim sistemi veriyi bütünüyle görebilme becerisine sahip olmalıdır;
2. **Bilgi:** Veritabanı ile ilgili bilgi (tablo başlığı, dizin bilgisi) de dahil olmak üzere bütün veriler tablolardaki değerler olarak temsil edilmelidir;
3. **Erişim:** Kullanıcılar, tablo adını vermek koşulu ile istedikleri verilere erişebilmelidir;
4. **Sıfır Değerleri:** Bir ilişkisel veritabanı sıfırlı değerleri tutmak zorundadır;
5. **Katalog:** Kullanıcı sistem kataloğuna da aynı ilişkisel özelliklerden yararlanarak erişebilmelidir;
6. **Veri Altdilleri:** Bir ilişkisel veritabanı yönetim sisteminin veri tanımı, görünüm tanımı, veri yönetimi, bütünlük sınırlılığı, yetkinlik ve mantıksal geçişleri gerçekleştirebilen en az bir dile sahip olması gerekir;
7. **Güncelleme:** Görünüm tabloları üzerinde güncelleme yapılabilmelidir;

---

51. Pratt, a.g.e., 176.s.

8. **Yüksek Düzeyli Güncelleme:** Veritabanı yönetim sisteminde basit bir komut ile sıralar üzerinde yüksek düzeyli güncelleme yapılabilmelidir;
9. **Fiziksel Veri Bağımsızlığı:** İlişkisel bir veritabanı yönetim sisteminde fiziksel depolama ve erişim açısından yapılacak değişiklikler kullanıcıları veya uygulama programlarını etkilememelidir;
10. **Mantıksal Veri Bağımsızlığı:** Aynı şekilde mantıksal yapıda meydana gelecek değişiklikler de kullanıcıları ve uygulama programlarını etkilememelidir;
11. **Bütünlük:** İkincil veri dili bütünlük sınırlılığına sahip olmalı söz konusu sınırlılığa katalogda yer verilmelidir;
12. **Dağıtım:** Dağıtık veritabanı, veritabanının işlevsel parçalar halinde bilgisayar ağı içerisinde değişik bilgisayarlara yüklenerek veritabanı yönetimini sağlayan ilişkisel veritabanı yönetim sistemi tarafından kullanıcılara uzaktan erişim olanağı sunması;
13. **Bütünlüğün Korunması:** Her düşük düzeyli dil ilişkisel bir veritabanı yönetim sisteminde yüksek düzeyli bir dille oluşturulmuş bütünlük sınırlılığını zedelememelidir;

Date ise ilişkiselliği belirleyen dört kategori saptamıştır. Buna göre:

1. Tablo: Bu tür bir sistemde kullanıcının kavrayabildiği tek yapı tablodur. Ancak Codd'un tanımındaki gibi sistem **seleksiyon, projeksiyon** ve **birleştirme** işlemcilerini desteklememektedir. Tersine çevrilmiş kütük sistemi buna örnek oluşturur.

2. **En Alt Düzeyde İlişkisel Olma:** Yukarıda adı geçen işlemciler ilişkisel cebirin yegane işlemleri değildir. Toplam sekiz işlemci daha bulunmaktadır. Ancak en alt düzeyde ilişkisel olmanın koşulu tablo yapısına sahip olma ve söz konusu üç işlemi gerçekleştirebilmektir [52].

3. **Tam İlişkisellik:** Tablo yapısı ile birlikte ilişkisel cebirin tüm işlemlerinin gerçekleştirilebilmesi.

4. **Bütünüyle İlişkisellik:** Tablo yapısı, ilişkisel cebirin tüm işlemleri ve iki bütünlük kuralını destekleyen tüm sistemlerdir. Bu tanıma uyan bütünüyle ilişkisel bir sistem henüz gerçekleşmemiştir. Çünkü sistemlerin çoğu **gönderme bütünlüğünde** başarısızlığa uğramaktadır [53].

İlişkisel modelin diğer modellere oranla getirdiği birçok avantaj bulunmaktadır. Bu avantajları Curtis[54] şöyle özetler:

1. Dizinleme, depolama ayrıntıları veya ağ ve hiyerarşik modellerde görülen sıralama gibi mantıksal olmayan kavramları içermezler;

2. Veri parçalarına erişimi sağlayan mantıksal olmayan yollara gerek duymazlar. Bu veri parçalarına kurallar ve koşullar bazında erişilir. O nedenle ilişkisel model mantıksal bir veri modeli olup depolamadan bağımsızdır;

---

52. Aynı, 173 s.
53. Aynı, 174 s.
54. Graham Curtis, Business Information Systems; analysis, design and practice. Washington: Addison Wesley, 1989, 184 s.

3. İlişkisel veritabanı sistemleri büyük veri gruplarının basit işlemler ile işlenmesini sağlar.

Sağladığı **uygulama bağımsızlığı, mantıksal anahtar göstericiler, normalizasyon kuramı** ve **yüksek düzeyli programlama dilleri** gibi özellikleri ile ilişkisel modeller, sahip oldukları düz kütük yapılarıyla anlaşılmaları ve erişimleri kolay modellerdir. Ancak veri tekrarına neden oldukları için depolama alanı ve çoklu güncelleme işlemleri açısından olumsuz yönleri bulunmaktadır.

İlişkisel veritabanı satın alınırken gözönünde bulundurulması gereken bazı noktalar vardır. Örneğin satıcı firmanın ürünü destekleyici, güncelleştirici ve ek ürünleri sağlayıcı türden yardımcı hizmetler sunması; ürünün gelişimi ve denenmesi gibi. Ürünün gelişimi söz konusu olduğunda ise sistemin veri kataloğu, kullanıcı hizmet birimi, iyileştirme, taşınabilirlik ve SQL diline sahip olması beklenmektedir. Ayrıca veritabanı kapasitesi, depolama yapısı, yedekleme, eş zamanlılık kontrolünde sahip olduğu özellikler de önem taşımaktadır [55].

3.2.3.1. Sistem R

Yöneltilen rastgele sorulara yanıtlar üreten ve ilişkisel cebir ve matematik kullanmak suretiyle önceden biçimsel belirlemeleri yapılmış çıktılar üreten, bunun yanında sıradan güncelleme işlemlerini yapan Sistem R ilişkisel bir veri yönetimi sistemidir.

---

55. Fatma Yüksel, "İlişkisel Veri Tabanı Seçiminde Kriterler" Bilgisayar 115 (1991), 156.s.

IBM'in San Jose Araştırma Laboratuvarı'nda 1978-79 yılları arasında Codd'un bu kavramı ortaya atmasından sonra geliştirilen ilk ilişkisel sistemdir. Bu sistemde uç kullanıcının sisteme erişimi SQL (Yapısal Sorgulama Dili) aracılığı ile gerçekleştirilir. Sistem R, bir veritabanı yönetim sisteminin tüm olanaklarına sahip olup aynı anda birçok kullanıcıya yararlanma olanağı sunar. Veri ve dizin kütüklerini iç düzeyde depolamak amacıyla kesit (segment) kullanılır. Depolanmış veritabanı, mantıksal olarak bölümlenmiş birbirinden bağımsız kesitler kümesinden meydana gelir. Kesitler ise sayısı sürekli değişen "sayfa"lardan oluşmuştur. **Sayfa**, bilgisayarın belleği ile veritabanı arasında bir transfer birimidir. Transfer birimi olan sayfanın sabit bir boyutta olması geçici belleğin yönetimini kolaylaştırır. Çünkü bu durum birden fazla kullanıcının bilgisayardan aynı anda yararlanması olanağını verir. Kesitler aynı zamanda depolama için ayrılacak alanların denetimini de sağlar. Sistem R'deki kesitler üç tiptir.

a) **Kullanıcıya Açık Kesit:** Birçok kullanıcının aynı anda erişebildiği paylaşılan verileri içerir;

b) **Özel Kesit:** Tek bir kullanıcının yararlandığı yani paylaşılmayan verilerin kesiti;

c) **Geçici Kesit:** Paylaşılmayan ve sistem hata yaptığında yeniden elde edilemeyen verilerin kesiti.

İlişkisel bir veritabanı yönetim sisteminde iki temel işlev vardır:

1. İlişki yapısını oluşturmak,

2. Veriyi bu ilişkiye yükleyerek daha sonra erişim sağlamak: Bu iki işlev veritabanı yönetim sisteminde veri tanımlama dili ve veri yönetim dili ile gerçekleştirilir. Sistem R'nin kullandığı sorgulama dili SQL'in alt dillerini oluşturan söz konusu dillere ilişkin komutlar SQL'in anlatıldığı 3.3.3. bölümünde ele alınacaktır.

3.3. SORGULAMA DİLLERİ

3.3.1. İlişkisel Cebir

İlişkisel cebir kullanıcıya, gereksinim duyduğu verilere nasıl erişeceğini bilme zorunluluğu bulunmadan ulaşma olanağı tanıyan yöntemsel olmayan bir dildir.

Yöntemsel dile oranla daha az programlama gerektirmesi açısından avantaj sağlar. Çünkü burada kaynak programı makina diline çeviren bir "üretici program" bulunmaktadır. O nedenle bir programın makine içindeki işleyişi sırasında gerçekleşen aşamalardan bağımsızdır. Onun yerine aşağıdaki şu üç yolu izler:

1. okunacak kayıt ve kütüklerin belirlenmesi;
2. girilen veri üzerine hangi matematiksel ve mantıksal işlemlerin gerçekleştirileceği ve
3. çıktının biçimlendirilmesi [56].

---

56. Mittra, a.g.e., 73 s.

Bütün bunların yanında esnek olmayışı yöntemsel olmayan dillerin dezavantajıdır. Çünkü girdi/çıktı işlemleri ve belleğin en alt düzeyde kullanımı gibi makina işlevleri denetlenememektedir.

İlişkisel model küme kavramı ve onun daha gelişmiş biçimi olan ilişkisel cebir üzerine kurulmuştur.

Codd ilişkisel modelin dayandığı ilişkisel cebir için sekiz işlem belirlemiştir. Sırasıyla bu işlemler şöyle sıralanabilir:

1. bileşim (union);
2. kesişim (intersection);
3. ayrım (difference);
4. bölme (division);
5. seleksiyon (selection);
6. projeksiyon (projection);
7. birleştirme (join);
8. dekart (Descarters) çarpımı (cartesian product).

İlişkisel modelin temel olgusu küme, elemanlardan meydana gelmektedir. Örneğin yayın numarası, yayın adı, teslim süresi ve yer gibi özelliklerden meydana gelen bir yayınevi tablosunda bulunan "yer" özelliği bir küme oluşturmaktadır. Özellik için belirlenen değerler kümesi ise o özelliğin tanım kümesini oluşturur. Yer= (Ankara, İstanbul) Burada "yer" küme, ankara ve istanbul da "yer" kümesinin elemanlarıdır.

Yukarıda verilen örnekten yola çıkarak genel bir küme ifadesi üretecek olursak : K = $\{e_1, e_2 \ldots\ldots e_n\}$'dir.

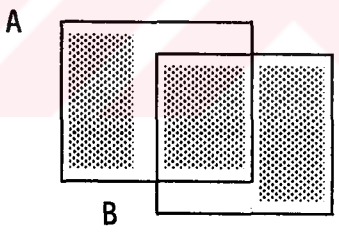
Cebirsel ifade işlemci, işleme tabi tutulan ve çözüm olmak üzere üç temel öğeden meydana gelmektedir. Kümeler birer ilişki olduklarına göre bileşim işlemlerine girebilmeleri için kümelerin aynı dereceden ve aynı tanım kümesine sahip olmaları gerekmektedir.

Bileşim (AUB) = A bileşim B işlemi sonucunda ortaya çıkan kümenin elemanları hem A hem de B'nin elemanları toplamıdır.

$AUB = \{ x | x \in A \text{ veya } x \in B \}$

$A = \{ e_1, e_2, e_3 \}$, $B = \{e_1, e_4\}$

$AUB = \{ e_1, e_2, e_3, e_4 \}$



Kümeye ait elemanlar belirtilirken örneğin ankara e {Ankara, İstanbul } ya da genel biçimiyle $e_n \in K$ şeklinde ifadeler kullanmak olanaklıdır.

Bunun yanında özelliklerden meydana gelen "yayınevi" tablosu da bir kümedir. "Yayınevi", yayın no, yay. adı teslim süresi ve yer olmak üzere dört elemanı bulunan bir kümedir.

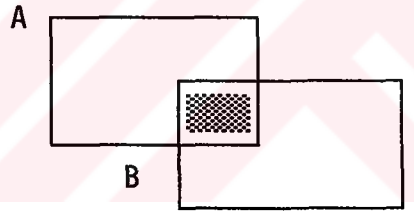
Gerçek dünyanın nesneler ve bunlar arasındaki benzerlikler üzerine kurulu olduğu yolundaki temel görüşten hareketle benzer nesnelerin oluşturduğu topluluğa o nesnelerin kümesi demek olanaklıdır. Buna göre bir kütüphanedeki belgeler, ödünç alan kullanıcılar, personel birer nesne ya da varlık olarak ele alınabilir. Burada söz konusu varlıkları biraraya getiren, onların sistem gereksinimi doğrultusunda sahip oldukları özellikleridir.

Kesişim (A∩B) = A kesişim B işlemi sonucunda ortaya çıkan küme elemanları hem A hem de B'de bulunan ortak elemanların tümüdür.

$A \cap B = \{ x | x \in A \text{ ve } x \in B \}$

$A = \{e_1, e_2, e_3\}$, $B = \{ e_1, e_4 \}$

$A \cap B = \{ e_1 \}$

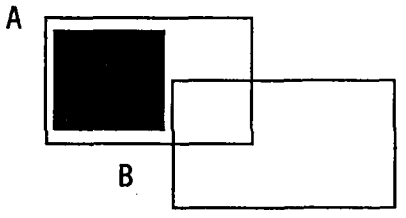


Ayrım (A-B): A ayrım B işlemi sonucunda ortaya çıkan kümenin elemanları A'da olup da B'de bulunmayan elemanların tümüdür.

$A-B = \{ x | x \in A, x \notin B \}$

$A = \{ e_1, e_2, e_3 \}$, $B = \{ e_1, e_2 \}$

$A-B = \{ e_3 \}$

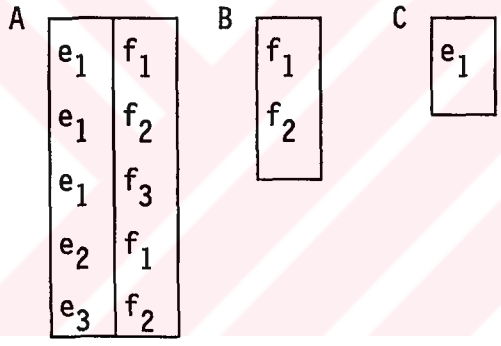


Bölme (A - B) Bölen, bölünen ve katsayı olmak üzere üç temel öğenin varlığını zorunlu kılar. Bu işlemci her iki işlem elemanının da bazı özelliklere sahip olmasını gerektirmektedir. Bölenin tüm özelliklerinin bölünenin bir özelliği ile aynı tanım kümesine sahip olması zorunludur. Katsayı yani bölümün sonucu ise her iki işlem elemanında da bulunmayan elemanların kümesini oluşturur.

$A = \{ a_1 \ldots\ldots a_m, a_{m+1} \ldots\ldots\ldots a_{m+n} \} = \{ (x,y) | x \quad x, y \quad x)$

$B = \{ b_1 \ldots\ldots\ldots b_n \} = \{ y | y \quad Y)$

$C = \{ c_1 \ldots\ldots\ldots c_m \} = \{ x | x \quad A_x, \{x,y) \quad A, A_y \quad B_y \}$



Bölünen Bölen Katsayı/Sonuç

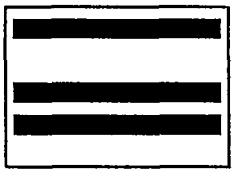
Seleksiyon: Predikat ile A gibi bir ilişki üzerine yapılan seleksiyonun sonucu İ gibi bir ilişki olup 'nun sağladığı koşullara uyan A elemanlarını içerir.

İ = predikat (ilişki)

İ = yer = Ankara (yayınevi)

Predikat= ,<,>, = >,< =, AND, OR ve NOT gibi mantık işlemcileridir.

A



Projeksiyon: A gibi bir ilişki üzerine yapılan işlemin sonucu İ gibi bir özellikler listesi olan A'nın özellikleri içinden kimi elemanları içeren yeni bir ilişkidir.

$$İ = \Pi_{\text{özellik adı}} (<\text{ilişki}>)$$

$$İ = \Pi_{\text{yer}}(\text{yayınevi})$$

A

Birleşim ({X}): $İ_1$ ve $İ_2$ dereceleri m ve n olan iki ilişki ise ve Θ da matematiksel herhangi bir karşılaştırma işlemcisi ise $İ_1$'in a_1 özelliği ve $İ_2$'nin b_1 özelliği $İ_1$ üzerine yapılan birleşim işlemi sonucunda ortaya çıkan küme $İ_1$'in S_1 sırası ile $İ_2$'nin S_2 sırasının Θ'ya uyan koşul uyarınca birleşmesinden meydana gelir.

Birleşimi dekart çarpımı ve seleksiyon cinsinden ifade etmek de olanaklıdır. Bir tablodaki bir sütun değerine karşılık diğer bir tabloda aynı sütun değeri bulunmuyor ise birleşim gerçekleşemez. Bunun yanında bir sıranın birden fazla bir tabloda yeralması da söz konusu değildir.

A

e_1	f_1
e_2	f_1
e_3	f_2

B

f_1	g_1
f_2	g_2
f_3	g_3

(AXB)

e_1	f_1	g_1
e_2	f_1	g_1
e_3	f_2	g_2

Tabloların bu tür birleşimine "doğal birleşim" denir. Birleşmenin bu yolla yapılmasına karşılık yaratılan yeni tabloda birleşmeyi sağlayan sütunun her iki kopyasının da aynen yaratılması durumunda buna "eşit birleşme" denir. Bir diğer birleşim tipi "dış birleşme"dir. Doğal birleşme ile büyük ölçüde benzerlik gösterir. Ancak doğal birleşmede orjinal ilişkideki sıraların diğer bir ilişkide karşılığı bulunmadığı zaman birleşim gerçekleştirilemezken "dış birleşme"de bu tür sıralar tabloya ilave edilerek bulunamayan karşılık değerleri "sıfırlı" değerler biçiminde ifade edilebilmektedir (57).

Kartezyan Çarpımı (AXB): A ve B gibi iki küme arasında gerçekleşen kartezyan çarpımı sonucunda ortaya çıkan İ kümesi A ve B'nin tüm elemanlarının kombinasyonu sonucunda oluşmaktadır.

$A = \{ e_1, e_2, e_3 \}$, $B = \{ f_1, f_2 \}$

$İ = AXB = \{e_1, f_1\}, \{e_1, f_2\}, \{e_2, f_1\}, \{e_2, f_2\}, \{e_3, f_1\}, \{e_3, f_2\}$

A

e_1
e_2
e_3

B

f_1
f_2

(AXB)

e_1	f_1
e_1	f_2
e_2	f_1
e_2	f_2
e_3	f_1
e_3	f_2

57. Pratt, a.g.e., 121.s.

3.3.2. İlişkisel Matematik

İlişkisel matematik veri yönetim dilinin formüle edilmesinde bir alternatif oluşturmaktadır. Ancak soruyu formüle etme biçimi ilişkisel cebire oranla farklıdır. İlişkisel cebir yoluyla formüle edilen soruda veritabanından istenilen verilerin hangi işlemler uyarınca elde edileceği ve işlemlerin sırasının ne olacağı belirtilir. Bu noktada ilişkisel matematik ilişkisel cebirden ayrılmaktadır. Yöntemsel bir dil olmayan ilişkisel matematik bilgiyi sağlamada belirli bir yöntem belirtmeden gereksinim duyulan bilgiyi tarif eder. O nedenle veri yönetiminde ilişkisel cebiri esas alan diller, kurala dayalı diller ilişkisel matematiği kullanan diller ise niteleyici diller olarak tanımlanmaktadır.

Ulaşılmak istenen bilgi için kurulan denklemin soruda belirli bir düzen içerisinde formüle edilme özelliğinden dolayı Korth ve Silberschatz gibi kimi bilim adamları ilişkisel cebiri yöntemsel bir dil olarak tanımlamaktadır [58]. Buna karşılık Mittra, ilişkisel matematiğin DO-WHILE ve IF THEN ELSE gibi yöntemsel dile özgü yapıları içermemesi nedeniyle ancak düşük düzeyli yöntemsel bir dil olarak kabul edilebileceğini savunmaktadır [59].

İlişkisel matematik terimi, matematikteki mantık işlemleri için kullanılmaktadır. Mantık işlemleri için kullanılması fikrini ilk kez 1967'de J.L., Kullins ortaya atmıştır. Daha sonra Codd, özellikle ilişkisel veritabanlarında bu dilin uygulamasını yaparak dile özgü temel kuralları formüle etmiştir. İlişkisel matematikte kullanılan işlem ve nitelemeler şöyle sıralanabilir.

---

58. Korth, a.g.e., 75-76. ss.
59. Mittra, a.g.e., 96.s.

1. NOT ($\neg$) kalıbı ile ifade edilen olumsuzluk işlemi;
2. AND ($\wedge$) kalıbı ile ifade edilen bileşim işlemi;
3. OR (V) kalıbı ile ifade edilen ayrıştırma işlemi;
4. IF THEN (——>) kalıbı ile ifade edilen gerektirme işlemi;
5. IFF (<——>) kalıbı ile ifade edilen denklik işlemi;
6. FORALL ($\forall$X) kalıbı ile ifade edilen evrensel niteleme;
7. EXISTS ($\exists$X) kalıbı ile ifade edilen varoluşsal niteleme.

İlişkisel matematik "sıra ilişkisel" ve "tanım kümesine dayalı ilişkisel" olmak üzere iki tiptir.

Sıraya Dayalı İlişkisel Matematik: İlişkisel modelde bir ilişkinin yani bir tablonun herbir sırası bir varlığa ilişkin olguları temsil eder. Diğer bir deyişle herbir sıra o ilişkinin değişkenidir. **Sıra ilişkisel** matematiğe özgü S/P(S) ifadesinde S, I ilişkisinin bir sırası yani bir değişkenidir. K(S) ise değişkenin doğru olduğu değerleri içeren sıraların kümesidir. K(S), değişkenin yöneltilen soru uyarınca alacağı değerleri içeren kümeyi oluşturmada kullanılan formül biçiminde ele alınabilir. Söz konusu formül işlemciler ve öğeciklerden meydana gelir. Öğecikler formülde üç tipte bulunur:

1. I(S): S, I ilişkisinin bir sırasıdır;

2. s(i)θu(j):s ve u sıra değişkenlerinin (örneğin farklı iki sıra), i ve j özellik değerlerinin, ise ilişkisel matematikteki herhangi bir karşılaştırma (>, < gibi) işlemcisinin ifadesidir.

3. S(i)θa: ifadesinde ise a sabit bir değerin ifadesidir.

Tanım Kümesine Dayalı İlişkisel Matematik: Sınırlı ve serbest olma gibi özellikleri bulunan tanım kümesi, bir değişkendir. Eğer tanım kümesi elemanlarının tümü sırasıyla değer olarak alınıyor ise **serbest**, değişkenin değeri bir özellik-sıra kombinasyonundaki değere karşılık geliyor ise **sınırlı tanım kümesi** olarak nitelendirilir. Tanım kümesine dayalı ilişkisel matematikte **öğecikler** aşağıdaki şu biçimlerde bulunabilir:

a) $I(ö_1.....ö_n)$: Burada I ilişkisi n. dereceden bir ilişkidir. Herbir özellik yani ö, sabit bir değer ya da bir tanım kümesi değişkenidir.

b) $ö_1 \sigma ö_2$ gibi bir ifade de $ö_1$ ve $ö_2$ sabit birer değer veya tanım kümesi değişkenidir. ise (<,>=, gibi) matematiksel bir işlemcidir. $ö_1$ ve $ö_2$'nin anlamı, "$ö_1 \sigma ö_2$" ifadesini doğru kılacak $ö_1$ ve $ö_2$ değerlerinin bulunduğudur.

3.3.3. Sorgulama Dili: SQL

Kullanıcının veritabanından istediği veriyi elde etme süreci **sorgulama** olarak adlandırılmaktadır. Söz konusu süreçte bir **komut** kümesinden yararlanılır. Veriye erişim yanında ekranda görüntülenmesini de sağlayan komut kümesi aynı zamanda veri yönetim dilinin bir altkümesidir.

Veritabanından bilgi sağlama yani sorgulamada çeşitli yaklaşımlar vardır. İlişkisel cebir, ilişkisel matematik, sıraya dayalı

ilişkisel matematik, hibrid yapıya sahip diller ve doğal dil. Söz konusu yaklaşımlara dayanan kimi diller ise QQL (Quick Query Language), QUEL (Query Language), QBE (Query By Example), SQUAR (Specifying Queries As Relational Expressions) ve SQL (Structural Query Language) gibi dillerdir. Bunların yanında doğal dil yaklaşımını temel alan OLQ (OnLine Query) ve OLE (OnLine English) gibi diller de bulunmaktadır.

Yöntemsel olmayan bu dillerin sağladığı avantajlar şöyle sıralanabilir:

1. tek kayıt yerine bir kayıt kümesine erişim sağlama;
2. soruyu hızlı yanıtlama;
3. tam bir sözdizimi ve gramere sahip olma.

Doğal dil yaklaşımını benimseyen dillerin getirdiği yarar ise kullanıcının doğal dilde yönelttiği soruyu yapılandırılmış soru biçimine dönüştürerek anlamlı çıktı üretebilme olarak özetlenebilir.

Yöntemsel olmayan dillerin sağladığı söz konusu avantajlar dikkate alınırsa **program yazma**'nın gereksiz olacağı düşünülebilir. Ancak yöntemsel olmayan dillerin erişim, rapor yazma ve güncelleme konularında yetersiz kaldığı noktalar vardır. Örneğin doğal dil yaklaşımında doğal dildeki cümlelerin kayıt düzeyindeki komutlara dönüştürülmesi işlem sürecini uzatmaktadır. Önceden yöntemsel dille yazılmış bir rutin aynı türden başka bir erişimi daha hızlı yapabilmektedir. Anında niteleyip sisteme soru yöneltmek ve bir kereye mahsus raporlar üretmek gerektiğinde yöntemsel olmayan diller iyi sonuç vermektedir. Ancak ha-

reketsel işlemlerde[60] ve sıklıkla yinelenen ve hızlı yanıt gerektiren soruları, bilgisayarın iş yükünü artırdığından yöntemsel diller veri modelinin işlenmesine daha elverişli olan dillerdir.

Yöntemsel olmayan diller içinden SQL'in burada örnek olarak seçilmesinin nedeni, yöntemsel olmayan veritabanı sorgulama dilleri için bir endüstri standartı olarak kabul edilmesidir. 1982 yılında ANSI veritabanı komitesi (X3H2)'ne standart ilişkisel bir dil geliştirme görevi verilmiş ve X3H2'nin önerisi 1986'da ANSI tarafından onaylanmıştır. Bu dil, IBM'in ürettiği System R için geliştirilen SQL'den başkası değildi. SEQUEL adıyla gelişmeye başlamışsa da donanım SEQUEL'den ayırdedilebilmesi için buna SQL adı verilmiştir. Hem veri tanımlama hem de veri yönetim dilidir. Ulusal Standartlar Enstitüsü tarafından standart bir dil olarak onaylanan SQL için 1987 yılında yapılan bir saptamaya göre, 60'ın üzerinde ticari firma SQL'e dayalı veritabanı yönetim sistemi satmaktadır [61].

İlişkisel veritabanında gerçekleştirilecek işlemlerin ifadesinde kullanılan SQL, yöntemsel bir dil olmadığı ve karmaşık becerilere sahip bulunduğu için kullanıcı belli istekleri belirgin bir biçimde ifade etmede veya kimi ifadeleri anlamada zorluk çekebilmektedir.

SQL'in sözdiziminin biçimsel tanımlamaları konusunda önemli ilerlemeler kaydedilmesine rağmen anlamsal biçim tanımı henüz gelişimini tamamlayamamıştır.

---

60. Uyarmalı İşlem: Bir merkezi işlemcinin uzak yerlerden gelen mesajlarla eyleme geçmesi ve bu yerlere uzun yanıtlar yollaması.
61. Mittra, a.g.e., 139 s.

SQL soruları genelde SELECT cümlesinin nasıl geliştiği konusunda kavramsal açıklama yapan bir algoritma ile tarif edilmektedir. Böyle bir açıklama SELECT ile başlayan cümlenin **anlamsal tanımı** görevini yapar. Son yıllarda çalışmalar SQL'in eş düzeydeki ilişkisel cebir cümlelerine çevrimi konusunda yoğunluk kazanmıştır. Bu amaçla, özellikle anlamlılığın biçimsel tanımını oluşturmada, **özellik grameri**'nin etkin bir yöntem olduğu savunulmaktadır[62].Özellik grameri tarif edildiği dilin sözdizimi ve anlama dayalı biçimsel tanımı olarak düşünülmektedir. Bu konuda SQL'den (ilişkisel cebirde olduğu gibi) iyi tanımlanmış bir yapıya biçimsel bir ulaşım gerçekleştiğinde başarılı olunacağına inanılmaktadır.

Özellik grameri ile ilişkisel cebir ifadesi herhangi bir veri diline çevrilebilir. Bu da heterojen veritabanı sistemlerinin karşılıklı bağlanabilmesini sağlayacaktır. Multibase, Series Delta ve Mermaid gibi sistemler bu tür karşılıklı bağlantıyı başarıyla gerçekleştirmişlerdir. Ancak bunu kodlamaya başvuran çeviricilerle yeni bir sorgu dilini desteklemek suretiyle başarmışlardır. Özellik gramerinin dağıtık veritabanlarında kullanılması ile aynı veritabanı makinasında birçok veri dilini kullanma olanağı doğmaktadır. Özellik grameri dil için belirgin bir biçimde ifade edilebildiği ve gramer dili komut kümesi olarak tanımlanabildiği sürece yeni veri dili eskisinin yerini alabilmektedir. Dolayısıyla **özellik grameri** SQL cümlelerinin ilişkisel cebire çevriminde anlamsal biçim tanımını sağlayıcı bir yöntem olarak önerilmektedir.

---

62. Michael Frame-Mehdi Owrang, "SQL Translation Using an Attribute Grammer" Information Sciences. 71:3 (1993), 285 s.

SQL sorgulama dilinin veritabanına yönelik işlemleri yapacak komutları ve bunların işlevleri şöyle açıklanabilir: SQL'de veritabanı ya da yeni bir ilişki yaratmak için CREATE TABLE komutu kullanılır. Örneğin: CREATE TABLE tablo adı, sütun adları..... . Tablo adını belirleme kuralları SQL versiyonlarına göre değişmekle birlikte genel kurallar şöyle özetlenebilir.

- İsim 18 karakterden fazla olamaz;
- İsim bir harf ile başlamalıdır;
- İsim harf, sayı ve noktalama işareti içerebilir;
- İsimde ara boşluklar bulunmaz.

Veri tanımlamada kullanılan veri yapıları ise decimal, char, integer, smallint ve date gibidir.

SQL'de bir tablonun tüm sütunlarına kimi sütunlarına veya tablodaki belli bir değere erişmeye **basit erişim** denir. Bu amaçla SELECT, FROM ve WHERE kalıpları kullanılır.

Örneğin: SELECT görmek istediğimiz sütun adları FROM sütunların yeraldığı tablo adları WHERE erişilmek istenen veriler için belirlenen koşullar.

Eğer bir tablodaki tüm sütunlar görülmek isteniyor ise bu istek sütun adlarını tek tek yazmak yerine '*' işareti kullanılarak ifade edilebilir.

Karmaşık erişim durumunda AND, OR ve NOT işlemcileri kullanılarak birden fazla basit erişim koşulu birleştirilebilmektedir.

İlişkisel modelde görülen önemli bir özellik tablolarda sütunların bir sıra izlememesi ve verilerin giriş sırasına göre tablolarda yeralmasıdır. Ancak sıralamanın önem taşıdığı durumlarda bu işlev ORDER BY kalıbı ile yerine getirilir.

İşleve dayalı bilgi üretiminde, örneğin toplama ve ortalama alma gibi kullanılan fonksiyonlar COUNT, SUM, AVG, MAX, MİN'dur. Tek tek toplumlar alınmak istendiğinde GROUP BY kalıbı kullanılır. Bu aynı değere sahip parçaların tek bir grup altında toplanmasını sağlar. Gruplar için belli koşullar doğrultusunda sonuç elde edilmek istendiğinde ise, HAVİNG kalıbı kullanılır. Çünkü WHERE yalnız sıralara ilişkin koşulları belirleyebilmektedir. Bunun dışında; Join, union gibi ilişkisel cebir işlemcileri de kullanılır. SQL'de varolan veri üzerine değişiklik yapmak için "UPDATE alan adı SET alan adı: değer WHERE koşul" sözdizimi kullanılır. Veri ilavesi ve silme için sırasıyla INSERT ve DELETE komutları kullanılmaktadır.

SQL'de varolan tablolar kullanılarak yeni tablolar üretme olanağı da vardır.

III. Bölüm boyunca veri tabanlarının oluşturulmasında temel amaç olan modellemenin ne olduğu, bu alanda benimsenen ve ticari sistemlerde büyük ölçüde yeğlenen üç temel model ve bunların özellikleri, yine bu üç modelin ilk ticari ürünleri hakkında tanıtıcı bilgiler ve-

rilmiştir. Model, oluşturulacak sistemin verimli kullanımını sağlayan ve kullanılacağı birim için bilginin tasarımının yapıldığı mantıksal bir oluşumdur.

Bu oluşumdan yararlanmayı sağlayan ise kullanıcının mantığını kolayca kavrayacağı, kullanıcı uyarlı diller vardır. Veritabanından istenen bilginin çıkarılmasında veya üretilmesinde etkin bir görev üstlenmiş olan bu dillerden SQL bölüm kapsamına alınmış ve özellikleriyle tanıtılmıştır. IV. Bölümde ise Model oluşturmada kullanılan tekniklerden normalizasyon ve Nijssens Information Analysis Methodology adıyla bilinen NIAM tekniği anlatılacaktır. Böylelikle tasarımın gerçekleştirilmesinde, veri tabanında tutarlılığın ve verimliliğin sağlanmasında bilimsel bir yöntem izlenmiş olacaktır.

IV. BÖLÜM
VERITABANI TASARIM YÖNTEMLERİ

4.1 NORMALİZASYON

Normalizasyon veritabanı tasarımını verimli kılan bir araçtır. Bir varlık ile ilgili bir gerçeğin ideal olarak tek bir yerde görünmesi ilkesine dayanır. Amaçları şöyle sıralanabilir:

1. veritabanının bütünlüğünü artırmak;
2. veritabanında yineleme ve tutarsızlığı en aza indirgemek;
3. verilerin işlem dışında silinmesini önlemek;
4. istenen bilginin temsil edilebilme olasılığını artırmak.

1,2 ve 3. normal biçimleri 1972 de codd tarafından ortaya atılan normalizasyon kuramı "normal biçim" kavramı üzerine kurulmuştur. Daha sonraları Codd'un tanımlaması dışında olan diğer normal biçimler için Boyce, Fagin [1], Ullmann, Aho ve Beeri gibi bilim adamları birçok kural geliştirmiştir. Bir **normal biçim** ilişkinin sahip olduğu sınırlılıklara paralel ele alınmışsa yani ilişki ancak birtakım sınırlılıklara sahipse belirli bir normal biçimdedir. Örneğin bir ilişki tanımlandığı anda 1.normal biçimdedir veya parçalanamaz, ayrışamaz bir niteliğe sahip olduğunda INB'dedir.

---

1. R. Fağın, "A Normal Form for Databases That is Based on Domains and Keys" ACM Transactions on Database Systems. 6 September,1981,388 s.

Normal biçimler şöyle sıralanabilir :

a) 1.normal biçim : INB,

b) 2.normal biçim : 2NB

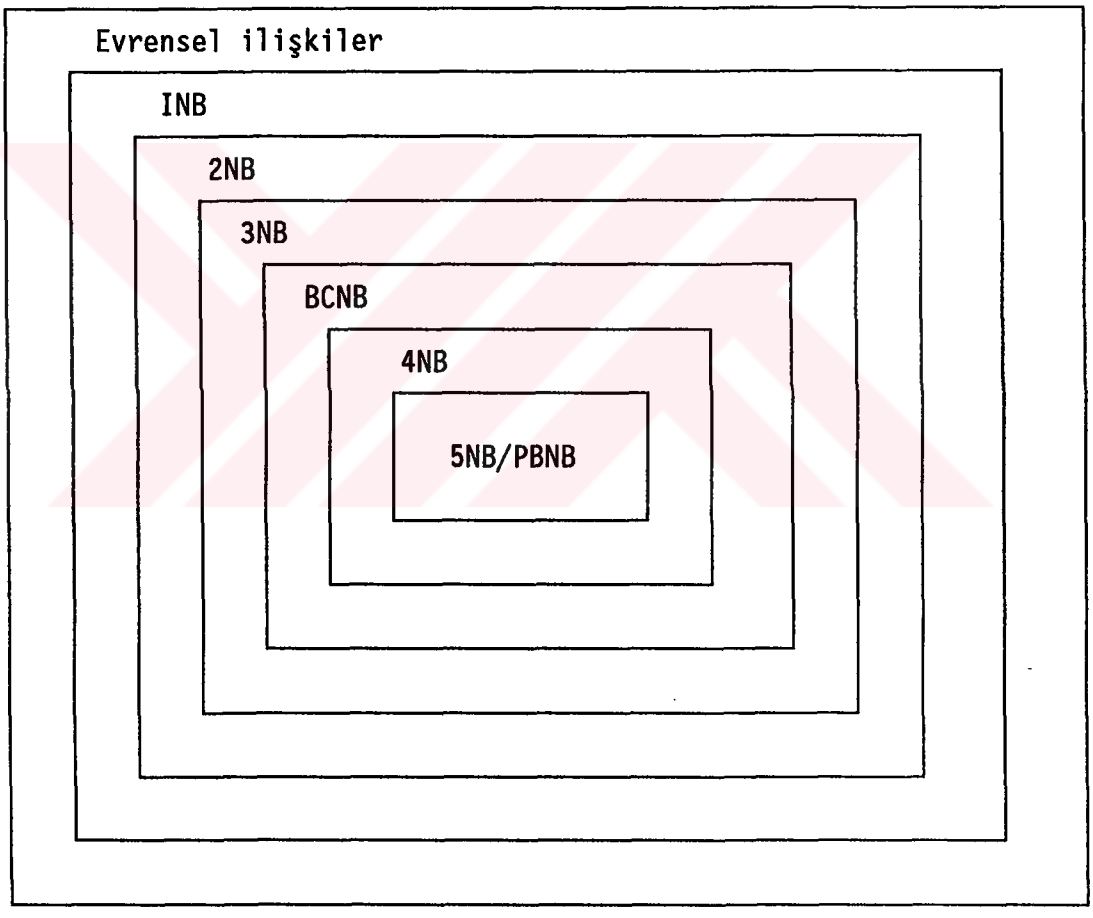
c) 3.normal biçim : 3NB

d) Boyce-Codd normal biçim: BCNB,

e) 4.Normal biçim : 4NB

f) 5.normal biçim : 5NB veya

projeksiyon-birleşim normal biçim : PBNB



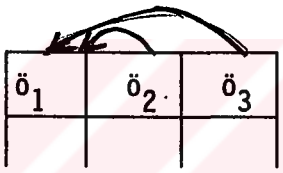
Şekil 4.1: Normal biçimlerin ayrışarak olgunluk kazanma süreci[2]

2. J.C. Date, An Introduction to Database Systems. Vol:1, 4th ed. Reading Massachusetts: Addison Wesley, 1985, 363 s.

Normalizasyon kuramını incelemeden önce bu kuramın dayandığı temel kavramlardan biri olan "fonksiyonel bağımlılığı" tanımlamak gerekmektedir.

Fonksiyonel bağımlılık ilişkisel bir veritabanında bulunan varlıklara uygulanan bir grup sınırlamadır [3].

I ilişkisinin $ö_2$ özelliği I'nın $ö_1$ özelliğine ancak ve ancak I'deki her $ö_1$ değeri belli bir $ö_1$ değerini belirleyebiliyorsa **fonksiyonel olarak bağımlıdır.** Bir başka deyişle eğer $ö_1$ özelliği I ilişkisinin bir aday veya ana anahtarı ise o zaman I'nin bütün diğer özellikleri fonksiyonel olarak $ö_1$'e bağımlı olmalıdır.



$I.ö_1 \longrightarrow I.ö_2$ yani $ö_1$, $ö_2$'yi fonksiyonel olarak belirleyen bir rol oynar. $ö_1$ ve $ö_2$, $ö_1$ ve $ö_2$, gibi değerlerden oluşan birer küme ise matematiksel olarak $f:ö_1 \longrightarrow ö_2$ biçimde ifade edilir. Fonkisyon $ö_1$ in $ö_2$'deki görüntüsünü verir.

Tanımlanması gereken bir diğer bağımlılık **"tam fonksiyonel bağımlılık"**tır. Burada I ilişkisinin $ö_2$ özelliği yine I ilişkisinin $ö_1$ özelliğine tam fonksiyonel olarak bağımlıdır. $I(ö_1) \longrightarrow I(ö_2)$ eğer $ö_2$, $(ö_1)$'e fonksiyonel olarak bağımlı ise ve $ö_1$'in hiçbir alt kümesine

3. Sitansu S. Mittra, Principles of Relational Database Systems. Englewood Cliffs: N.J.: Prentice Hall, 1991., 145 s.

fonksiyonel olarak bağımlı değilse $ö_2$, $ö_1$'e **tam fonksiyonel bağımlıdır** denir [4].

Verimli ve tutarlı mantıksal bir veritabanı tasarımı için fonksiyonel bağımlılığı olan özelliklerin kümesini saptamak zorunludur. 1974'de W.W.Armstrong kendi adıyla anılan bir grup belit (axiom)ortaya atmıştır.

Buna göre:

s : fonksiyonel bağımlılığı olan özelliklerin kümesine verir,

S : maktıksal olarak s'den çıkarılan bütün fonksiyonel bağımlılığı olan özelliklerin kümesidir [5].

Armstrong'un fonksiyonel bağımlılığı bulunan özelliklerin kümesini belirleme olanağı veren bütünleyici kuralları söyledir:

1. **Esneklik (reflexity):** X özellikler kümesi olduğunda eğer Y, X'in bir altkümesi ise yani YCX ise o zaman X ——>Y olur.

2. **Ekli Dizey (augmentation):** eğer X——>Y ise W de bir özellikler kümesi ise W X ——> W Y olur.

3. **Geçişlilik (transitivity):** eğer X——>Y ise Y——>2 ise o zaman X——>2'dir.

4. **Bileşim (union) :** eğer X——>Y ve Y—>2 ise o zaman X—>Y 2 dir.

5. **Ayrışma (decomposition):** eğer X—> Y 2 ise X—> Y ve X—>2 dir.

---

4. Date, a.g.e., 365 s.
5. Mittra, a.g.e., 147 s.

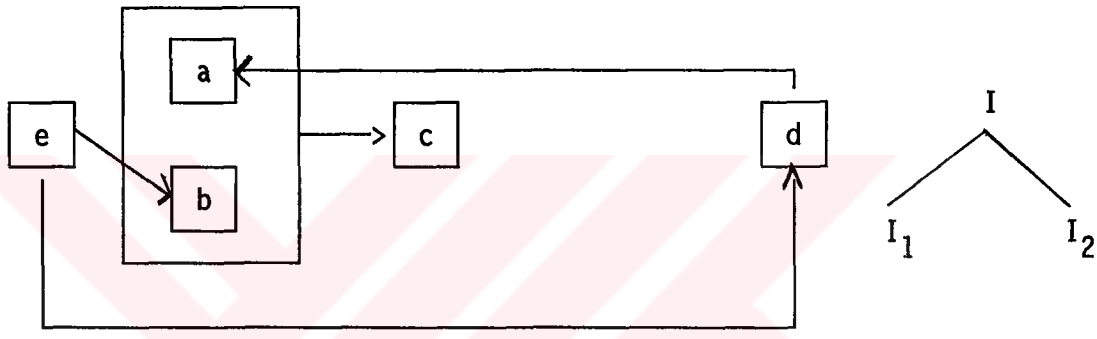
6. **Yapaylık (pseudatansivity):** eğer X——> Y ve W∧Y——>2 ise a zaman X∧W——>2'dir [6].

Codd'un ortaya attığı ilk üç normal biçim ise şöyledir:

1. **INB :** Bir ilişki, eğer tanım kümesi yalnız ayrışmaz/parçalanamaz değerler içeriyorsa normal biçimdedir [7].

A—>B : A ve B iki ayrı özellik kümesi ise her A değeri için bir B değeri vardır.

Örnek:



I=(e,a,b,c,d)

e , diğer tüm özellikleri belirleme becerisinden dolayı ana anahtar. İlişki INB'de.

2. **2NB:** Bir ilişki ancak ve ancak 1.normal biçimde ise ve anahtar özellikler ana anahtarın tümüne tam fonksiyonel bağımlı ise 2NB'dedir. 2NB'de geçişli bağımlılık vardır. 1NB'de geçişli bağımlılık vardır. Bir ilişki INB'de olup da 2 NB'de değilse projeksiyon kullanılarak ayrıştırılabilir ve birleştirme işlemi ile tekrar biraraya getirelerek bilgi kaybı olup olmadığı kontrol edilebilir. Yani ayrışma bilgi kaybı olmadan gerçekleşmelidir. Söz konusu durum Heath kuramı olarak da bilinmektedir [8].

---

6. Aynı, 147 s.
7. Date, a.g.e., 367 s.
8. Aynı, 370 s.

Örnek :

Sipariş = (**Yayın Adı, Sipver Böl,** Fiyat)

	Yayın Adı	Sipver Böl.	Fiyat
1	Rönesans'ta Tiyatro	Sanat Tarihi	300.000
2	Rönesans'ta Tiyatro	Tiyatro	300.000
3	Mikelanj Dönemi	Sanat Tarihi	250.000

Burada **Rönesans'ta Tiyatro** adlı eserin fiyatı değiştirilmek istendiğinde güncelleme 1. ve 2. sıralarda beraber yapılmalıdır. Aksi halde tutarsızlık meydana gelecektir. 2NB'de anahtar olmayan özellik ilişki anahtarı (Yayın Adı, Sipver Bol)nın tümüne bağımlıdır. Ancak burada yineleme söz konusu olduğunda (Yayın Adı, Sipver Böl) ve (Yayınadı, fiyatı) ilişkiyi ikiye bölmek olanaklıdır).

3. **3NB** : Bir ilişki 3NB'dedir ancak ve ancak ilişkide anahtar olmayan özellikler.

a) birbirine fonksiyonel bağımlı değil ise,

b) yalnız ilişkinin ana anahtarına bağımlı

ise 3NB'dedir [9].

I_1 (a,b,c)

ab ——> c'yi belirliyor
ilişki BCNB'de

I_2 (e,d,a,b)

e ——> d'yi
d ——> a'yı
e ——> b'yi belirliyor
İlişki 2NB'de

I_2

I_{21}

I_{21} (e,b,d)

İlişki BCNB'de

I_{22}

I_{22} (d,a)

İlişki BCNB'de

9. Aynı, 366 s.

4. BCNB : Codd'un 3NB için vermiş olduğu tanımın yetersiz kalan bazı yanları bulunmaktadır. Örneğin, Codd'un tanımı aşağıdaki şu esasları yeterince karşılayamamaktadır.

1. bir ilişkinin bileşik aday anahtarlara sahip olması ve

2. bu aday anahtarların en az bir ortak özelliğinin bulunması gibi durumlarda yetersiz kalmaktadır. Ortaya konan yeni tanım 3NB'den daha güçlü yeni bir biçim getirdiği için bunun yeni bir adla anılması gereği uygun görülmüş ve BCNB olarak nitelendirilmiştir. BCNB tanımını vermeden önce "belirteç" (determinant) terimine açıklık getirmek gerekir. Belirteç, kendisi dışındaki diğer özelliklerin en az bir tanesinin kendisine tam fonksiyonel bağımlı olduğu bir özelliktir. Buna göre bir ilişki, ancak ve ancak herbir belirteci bir aday anahtar olduğu zaman BCNB dedir.

Örnek :

I_2 (a,b.d,e) INB

I_2

I_{21} I_{22}

I_{21} (e,d,a) I_{22} (e,b) BCNB'

e ⟶ d geçişli
d ⟶ a bağımlılık 2NB

I_{21}

I_{211} I_{22}

I_{21} (e,d) I_{22} (d,a)

e ⟶ d BCNB d ⟶ a BCNB'

5. **4NB** : I, özellikleri $ö_1$.. $ö_i$.. $ö_j$.....$ö_n$ olan bir ilişkidir. $ö_i$.. $ö_j$ özellikler kümesi içinde herhangi iki özelliktir. Bu iki özellik arasında **çok değerli bağımlılığın** olduğu düşünülürse yani $ö_i$ özellik kümesinin herhangi bir değeri için $ö_j$ özelliği de sıfır komesi veya daha fazla eş değerden oluşan bir küme bulunuyor ise (yani $ö_i$ deki değer kümesi hiçbir şekilde I ilişkisinin altkümesinin özellik değerlerine bağlanmamışsa) $ö_j$'nin $ö_i$ üzerinde **çok değerli bağımlılığı** vardır [10].

Date ise çok değerli bağımlılığı şöyle tanımlamaktadır:

A,B ve C gibi özellikleri olan bir I ilişkisinde I.A —>—>I.B çok değerliliği ancak ve ancak B kümesi değerlerinin I'deki (A değeri, C değeri) çiftine uyumluluğu A değerinin C değerinden bağımsız olmasına bağlıdır. Genelde A,B ve C bileşik bir özellik olabilir [11].

Örnek:

DÖK

Ders	Öğretim Elemanı	Konu

Burada DÖK ilişkisinde çok değerli bağımlılık vardır. İlişki ancak DÖ ve DK şeklinde projeksiyon ile 4NB'deki iki ilişkiye dönüştürülebilir.

Bunun yanı sıra çok değerli bağımlılık ancak I'nin en az üç özelliğinin bulunması halinde gerçekleşir. Fonksiyonel bağımlılık da aynı zamanda çok değerli bağımlılıktır. Burada belli bir belirteç de-

10. Mittra, a.g.e., 152 s.
11. Date, a.g.e., 366 s.

ğeri ile uyumlu bağımlı değerler kümesi, gerçek tek bir değerdir. Böylelikle her fonksiyonel bağımlılık aynı zamanda çok değerli bağımlılıktır. Ancak bunun tersi geçerli değildir [12].

Sonuç olarak bir ilişki, ancak ve ancak A ——> ——> B gibi I ilişkisinde bir çok değerlilik varsa 4NB'dedir. O zaman I'nin bütün özellikleri A'ya fonksiyonel bağımlıdır. Bir başka deyişle I'deki bağımlılıklar K ——> X (yani aday anahtardan bir özelliğe doğru olan fonksiyonel bağımlılık) e doğrudur.

6. **5NB**: 4NB'e kadar bir ilişkinin projeksiyon ile iki ayrı ilişkiye bilgi kaybına uğramadan ayrışabildiğini gördük. Ancak kimi ilişkiler üç ve daha fazla ilişkiye ayrıştığında bilgi kaybına uğramamaktadır. Bu olgu (n >2) ilk kez Aho, Beeri ve Ullman tarafından ele alınmıştır.

I ilişkisi ayrıştıktan sonra tekrar birleşim ile elde edilemesektedir. I = I_1 join I_2. Böyle bir durumda I = I_1 join I_2 .... join I_n n >2 biçiminde ayrışma gerçekleşir. Bazı bilim adamları 5NB'i **projeksiyon birleşim normal biçimi** olarak da adlandırmaktadır [13].

5NB'i tanımlayacak olursak : Bir ilişki eğer I ilişkisindeki her birleşim bağımlılığı o ilişkinin aday anahtarlarınca gerçekleştiriliyor ise 5NB'dedir.

---

12. Aynı, 364 s.
13. Mittra, a.g.e., 153 s.

Date, başarılı bir ayrıma özgü kuralları şöyle sıralamaktadır:

1. fonksiyonel bağımlılığı olmayan özellikleri elemek için INB'deki orijinal ilişkinin projeksiyonu alınır. Bu durum 2NB'deki ilişkiyi üretir;

2. 2NB'deki ilişkilerin projeksiyonu alınarak geçici bağımlılıklar elenir ve 3NB elde edilir;

3. 3NB projeksiyonu alınarak halen mevcut olan fonksiyonel bağımlılıklar elenir ve belirtecin aday anahtar olmadığı durumlar ortadan kaldırılır ve BCNB elde edilir;

4. BCNB projeksiyon alınarak fonksiyonel bağımlılığı olmayan çok değerli bağımlılıklar ortadan kaldırılır ve 4NB elde edilir;

5. 4NB'deki ilişkilerin projeksiyonu alınarak aday anahtar ile gerçekleşmeyen herhangi bir birleşim bağımlılığı ortadan kaldırılır ve 5NB elde edilir.

Sonuç olarak Date, 5NB tanımını şöyle vermektedir. Bir ilişki ancak ve ancak I'deki her birleşim bağımlılığı I'nin aday anahtarlarının bir sonucu olduğunda 5NB'dedir [14].

Normalizasyon ilişkilerin ayrışma yolu ile bir üst normal biçime kavuşturulmasında ve bu yolla veri bütünlüğünün sağlanmasında her

ne kadar verimli bir yol ise de nitelemeye ilişkin sorunları tümüyle ortadan kaldırmayi başaramamıştır. Örneğin veri üzerine yapılacak girdi, iptal gibi değişikliklerin yarattığı nitelemeye ilişkin sorunları bir tablo yardımıyla özetleyebiliriz.

Her öğrencinin bir etkinlikte bulunduğu ve söz konusu etkinlik için belli bir ücret ödendiğine ilişkin bilgiden yola çıkılarak oluşturulan aşağıdaki tabloda 125 no'lu öğrenci kaydının iptal edilmesi durumunda 125 no'lu öğrencinin jimnastik yaptığı ve bu etkinlik için 250.000 TL ücret ödemek gerektiğine dair bilgi kaybolmuş olur.

Etkinlik

Öğr no	Etkinlik	Ücret (bin)
100	Tenis	500
125	Jimnastik	250
150	Voleybol	300
200	Tenis	500

Buna karşılık yüzme etkinliği için 350.000 Tl ücret ödemek gerektiği yolundaki bilgi bir öğrenci bu etkinlik için başvurmadan tabloya girilememektedir. Girdi bağlamında nitelemeye ilişkin olarak karşımıza çıkan söz konusu sorun tablonun iki ayrı tabloya bölünmesi ile çözülebilir. Yeni ilişkiler (öğr.no,etkinlik) ve (etkinlik, ücret) biçiminde tanımlanır.

Öğr-Etk.

Öğrn.no.	Etkinlik
100	Tenis
125	Jimnastik
150	Voleybol
200	Tenis

Etk-Üct.

Etkinlik	Ücret
Tenis	500
Jimnastik	250
Voleybol	300

Bu durumda 125 no'lu öğrencinin kaydı silindiğinde jimnastik etkinliğinin bedelinin 250.000 Tl olduğuna dair bilgi kaybolmamış olur. Bunun yanı sıra yüzme etkinliği ve ilgili bedeli **Etk-Üct** tablosuna anında girmek de olanaklıdır.

Ancak ilişkinin bölünmesi de kimi zaman sorun yaratabilir. Örneğin **Etk-Üct** ilişkisinde yer almayan bir etkinlik ile ilgilenen bir öğrencinin kaydı **Öğr-Etk** tablosuna girilmeli midir ? **Öğr-Etk** tablosuna yeni bir kaydın girdisini yapmak için **Etk-Üct** tablosunda etkinlik, ücret değer çiftinin önceden yeralması gibi bir sınırlılık getirebilir. Bu tür sınırlılıklar ilişkiler arası sınırlılık olarak adlandırılır ve bunlara ilişkin bilgiye veri kataloğunda yer verilir.

Normalizasyon için şu ana kadar tanımları verilen normal biçimler henüz tüm niteleme sorunlarını çözebilmiş değildir. Buna karşılık Fagin, 1981'de ortaya attığı **Tanım kümesi/Anahtar Normal Biçim: TKANB** ile nitelemeye özgü sorunların çözülebileğini iddia etmektedir. Ancak ilişkinin **TKANB**'e dönüştürülmesine yönelik herkes tarafından kabul edilmiş kesin bir biçim oluşturulamamıştır. Fagin normal biçimi tanımlamadan önce söz konusu biçime özgü üç temel kavramı şöyle açıklamaktadır.

1. **Sınırlılık:** Özelliklerin bizim doğru ya da yanlış olarak değerlendirebileceğimiz net ve değişmez değerleri üzerine oluşturulmuş tüm kurallardır. Fonksiyonel bağımlılık, çok değerli bağımlılık örnek gösterilebilir. Ancak Fagin burada geniş bir tanım yapmasına rağmen zamana bağımlı sınırlılıkları kapsam dışında bırakmıştır [15].

15. David Kroenke-Kathleen A. Dolan, Database Processing; fundamentals, design, implementation. Chicago: Science Research Associates, 1983, 149 s.

2. **Anahtar** : Tek başına bir sıranın tanımlayıcısıdır;

3. **Tanım kümesi:** Özellik için belirlenmiş değer kümesidir. Fagin tanım kümesi değerlerini fiziksel ve mantıksal olmak üzere iki grupta ele almaktadır. Uygun bir biçimde gruplar oluşturma ve bunları ilişkiye dönüştürmenin amacı verileri kullanıcı gereksinimleri doğrultusunda gruplamak ve kural dışı nitelemeleri elemektir. Buna göre Fagin **TKANB**'yi şöyle tanımlar : eğer bir ilişkinin sınırlılıkları tanım kümesi ve anahtar tanımlamalarının bir sonucu ise a ilişki **TKANB**'dedir. TKANB kuramında ilişki oluşturma algoritması yoktur. Onun yerine tasarım kılavuzu vardır. Buna **ilişki yaratma kılavuzu** denir [16].

Kroenke özellik ilişkilerini aşağıdaki tablo ile şöyle özetler:

Tablo 4.1: Özellik İlişkileri [17]

	Bire bir	Çokludan tekliye	Çokludan Çokluya
İlişki tanımı	R(A,B)	S(C,D)	T(E,F)
Bağımlılıklar	A——>B B——>A	C——>D D—>C	E——>F F—>E
Anahtar	A veya B	C	(E.F)
Başka bir özellik ekleme kuralı	A veya B——>C	C——>E	(E,F)——>G

4.2 NIAM TEKNİĞİ

Yakın zamana kadar kütüphane otomasyon sistemleri bilgi erişim ve makinaya bağımlı basit kütük sistemleri olarak ele alınmaktaydı.

16. Aynı, 56 s.
17. Aynı, 156 s.

Ancak kütüphane otomasyonu üzerine çalışan kütüphaneci ve mühendislerin kullanılagelen sistemlerin kütüphanenin tüm işlemlerini kapsayamadığı yönünde şikayetleri vardı. Sistem tasarım sürecinde erişim hızı ve depolama alanı gibi fiziksel koşulların üzerinde fazlaca durulması nedeniyle sistemler esneklik ve genişlik becerilerinden yoksundu. Dahası sistem sentezi yapılmadığı için toplu veri işleme ve veri dönüşümüne bağlı olarak sistem performansı azalmaktaydı. Ayrıca hacim ve çeşitlilik açısından hızla artan kütüphane işlemlerinde, veri yönetimi konusunda sorunlar ortaya çıkıyordu. Örneğin ağ kullanımı, veri biçiminde standardizasyon, kütüphanelerarası ödünç verme, yayıncılar ile iletişim gibi o nedenle kütüphaneler otomasyona geçişte ticari veritabanı yönetim sistemlerinden yararlanma eğilimi gösterdiler. Böyle bir sistem,tasarım sürecini ve gelişimini basitleştirmek ve dördüncü nesil dillerin kullanımına olanak tanımak gibi avantajlar sağlamaktadır. Bunun yanı sıra ortak bilgi kaynağı olan veri kataloğunun bulunması, depolama alanının verimli kullanımını, toplu erişimi ve tutarlık denetiminin merkezi olarak yapılabilmesini olanaklı kılmaktadır. Ayrıca ilişkisel veritabanı yönetim sistemleri dev bilgisayarlardan kişisel bilgisayarlara kadar her çaplı bilgisayarda kullanılabilmektedir. İlişkisel veritabanı yönetim sistemlerinin performans sorunları üzerine birçok araştırıcı ve uzmanın yaptığı çalışma ve tartışmalar sözkonusu sistemlerin gelişimine ve kütüphane otomasyonu için kullanımına önemli katkılar sağlamıştır. Nishimoto ve Shoji uygulamaya yönelik sistemlerin tasarımında uzmanların bibliyografik verileri ilişkisel sistemlere şimdiye kadar uyarlamamalarına neden olarak INB varsayımını göstermektedir [18]. Bu varsayıma göre tablo sütunlarında yeralan tüm

18. Hidaki Nishimoto-Uro Shoji, "Complex View Support for Library Database System" Information Processing and Management, 25:5, 1989, 515 s.

değerler ayrışamaz niteliktedir. Bu koşulu sağlayan tablolar INB'dedir. Buna karşılık Nishimoto ve Shoji standart bir ilişkisel veritabanı yönetim sistemi için normal biçim varsayımına uymayan veri biçimlerini destekleyen **karmaşık görünüm arabirimini** öngörmektedir [19]. Öngörülen sistemin amacı kütüphane otomasyon sistemi için daha düşük maleyitle, esnek ve normal biçime uymayan koşulların çözülebilirliğini sağlamaktır. Söz konusu arabirin kullanılarak normal biçime sahip olmayan tabloların uygulama yöntemiyle INB'e eşit tablolara dönüştürülebileceği savunulmaktadır.

İlişkisel veritabanı yönetim sisteminin getirdiği ve bir önceki Bölüm'de anılan avantajları ile kütüphane otomasyon projelerinde yukarıda anılan problemlere çözüm olarak görülmeleri,ilişkisel veritabanı sistemlerinin kütüphanecilik alanında da giderek yaygınlaştığının göstergesidir. O nedenle bu bölümde ilişkisel veritabanı sistemlerinde tasarım sürecinde kullanılan, sisteme uyarlanması ve anlaşılması son derece kolay alan NIAM tekniği tanıtılacaktır. Ancak tekniğin tanıtımı ve uyarlamasına girmeden önce konu kapsamında yapılan sınırlılığın nedenlerini açıklamak yararlı olacaktır. Bu bağlamda kütüphanelerde hizmetler geleneksel anlamda **okuyucu** ve **teknik hizmetler** olmak üzere iki katagoride ele alınmaktadır. Kütüphanecinin okuyucu ile yüzyüze geldiği tüm hizmetler, okuyucu hizmetleri ve bunun dışında kalan diğer bütün hizmetler ise teknik hizmetler olarak değerlendirilmektedir. Teknik hizmetler için belirlenmiş alt hizmet birimleri ise;

a) sağlama,

b) kataloglama,

---

19. Aynı, 516 s.

c) süreli yayınlar,

d) bağış ve değişim,

e) cilt şeklinde gruplandırılmıştır.

Ancak kütüphanelerin türlerine ve büyüklüklerine göre bu gruplama farklılık gösterebilmektedir. Örneğin kimi kütüphanelerde cilt veya cilde hazırlama işlemleri süreli yayınlar bölümünde ele alınırken (20) bağış ve değişim işlemleri sağlama bölümü içinde gerçekleştirilir (21). Bu bağlamda teknik hizmetler için üç temel alt hizmet biriminden söz etmek olanaklıdır. Bunlar sırasıyla sağlama, kataloglama ve süreli yayınlardır.

Çalışmanın bu kısmında bir kütüphanedeki teknik hizmetler bölümünün alt hizmet birimlerinden birinde gerçekleştirilen bir işlem için kavramsal şema oluşturulması amaçlanmaktadır. Söz konusu amaç için bir üniversite kütüphanesinde "sağlama" alt hizmet birimi içerisinde yapılmakta olan **sipariş** işlemi seçilmiştir. Burada kütüphane türü olarak üniversite kütüphanesinin seçilme nedeni, hizmetlerin bilgisayara dayalı olarak verilmesi çabalarının, donanım ve yazılıma yönelik altyapı çalışmaları biçiminde öncelikle üniversite ve büyük kurum kütüphanelerinde görülmesidir. Çünkü üniversiteler araştırma ve eğitim işlevlerini yerine getiren kuruluşlardır ve üniversite kütüphanelerinin araştırıcı ve öğrenciden oluşan geniş bir kullanıcı kitlesi bulunmaktadır. Bu kitlenin en büyük özelliği de güncel ve doğru bilgiye hızlı bir biçimde erişme arzusudur. Yayın ve bilgideki sürekli ve

---

20. Marty Bloomberg-G. Edward Evans, Kütüphane Teknisyenleri için Teknik Hizmetlere Giriş. Çev: Nilüfer Tuncer, Ankara: Türk Kütüphaneciler Derneği, 1989, 6 s.
21. Aynı, 7 s.

kapsamlı artış ile bilgiye olan yoğun istek ve isteklerdeki çeşitlilik öncelikle üniversite kütüphanelerinde hizmetlerin bilgisayara aktarılmasını gerekli kılmış ve motive etmiştir. Sipariş işleminin seçilme nedeni ise tekdüze bir işlem olması bakımından tüm üniversite kütüphanelerinde uygulamaya konulabilirliğidir.

Sağlama bölümüne yayınlar öncelikle üç yoldan gelir :

a) satınalma,
b) bağış,
c) değişim.

Bağış ve değişim kütüphane için derme geliştirme konusunda izlenecek temel yol değildir. Çünkü kütüphaneler türlerine göre nitelenen kullanıcı kitlesinin bilgi gereksinimlerini karşılamakla yükümlüdür ve bu doğrultuda dermelerini geliştirir. Bu bakımdan satınalma derme geliştirmenin en tutarlı ve sağlıklı yoludur. Bununla beraber kütüphaneler zaman zaman bağış ve değişim yoluyla da dermelerine kaynak sağlar. Ancak özellikle bağış yoluyla kütüphaneye ne zaman hangi nitelikte yayın geleceğini önceden bilmek olanaklı değildir. Kütüphaneler kendi kullanıcılarının gereksinimlerini karşılayamayacak nitelikte olmasına rağmen bağışlanan yayınları değişim amacıyla kabul edebilir. Bağış ve değişim yolu ile kaynak sağlamadaki belirsizlikler ve standart yokluğu nedeniyle firmalar bu konuda bir yazılım geliştirmemişlerdir. O nedenle anılan sağlama yöntemleri konu dışında tutulmuştur. Bunun yanı sıra birçok kütüphane süreli yayın sağlama işini süreli yayınlar bölümünün sorumluluğuna verdiğinden bu konudaki istisnalar

dikkate alınmamıştır. Özet olarak veritabanı tasarımı üniversite kütüphanesindeki sağlama bölümü bünyesinde gerçekleştirilen sipariş işlemi üzerine yapılacaktır.

"Mantıksal veri modeli oluşturma, veritabanı tasarım sürecine girdi olacak bilgi yapı ve kurallarının açıkca temsil edilmesi tekniğidir. Mantıksal veri modeli oluşturmanın can alıcı noktası verinin kuruluş için **değerli bir kaynak** olduğunun kabul edilmesidir. Mantıksal veri modeli oluşturma bir teknik olmasının yanı sıra aynı zamanda verinin varlığı, bağımsızlığı, nasıl erişildiği, o'na kimin erişeceği ve erişimin bilgisayarla yapılıp yapılmadığı gerçeğinin farkına varılıp belgelendiği bir felsefedir"(22).

Mantıksal tasarım için kullanılmakta olan birçok yaygın yöntem bulunmaktadır. Bir önceki kısımda anlatılan normalizasyon buna örnek olarak gösterilebilir. Ancak burada **gerçek uyarlı bir yöntem olan Nijssen's Information Analysis Methodology: NIAM tekniği kullanılacaktır.** Çünkü NIAM tekniği;

a) tek bir veri yapısını yani gerçek tipini kullanırken birçok sınırlılık tipini de ortaya koyar,

b) yeni gerçek tiplerini türetme olanağı sağlar,

c) görünümü anlam taşıyan ve değer yüklemenin kolay olduğu çizimler oluşturur,

---

22. Candice C. Fleming-Barbara Von Halle, Handbook of Relational Database Design. Reading Massachusetts: Addison Wesley, 1989, 9 s.

d) insan iletişimini ilişkisel tablo yapısına dönüştüren ve gerçekler arası ilişkileri doğal dil aracılığı ile ifade edebilen bir süreci sağlar.

Daha önce de belirtildiği gibi veritabanı yönetim sistemi kavramsal, iç ve dış olmak üzere üç düzeyden oluşmaktadır. Veritabanındaki veri yapılarının fiziksel olarak depolama alanındaki konumları ve bunlara erişim yollarının belirtildiği düzey, iç düzey ve kullanıcıların gereksinimleri doğrultusunda sistemin çeşitli görünümler sunduğu düzey de dış düzeydir. Kavramsal düzey ise sistem ile kullanıcı arasında gerçekleşecek **söyleşi evreni** çerçevesinde tüm olasılıklar düşünülerek oluşturulmuş veritabanının mantıksal yapısıdır.

Burada ortaya atılan **söyleşi evreni** kavramı sistem ile kullanıcı arasında iletişimin sağlıklı ve anlaşılır olabilmesi için bu eylemde yer alan her iki öğenin de sınırlandırılmış bir konu üzerinde aynı dili kullanmasını veya aynı dili kullanmasa bile sözcüklerin aynı anlamı taşımasını öngörmektedir.. Bu bakımdan **söyleşi evreni** sistem ile gerçekleşmesi düşünülen karşılıklı konuşmanın tasarım ya da planına karşılık gelmektedir. Aynı zamanda söyleşi evreni sistemde kavramsal düzeyin de belirleyicisidir. Yani söyleşi evreninde yer alması düşünülen tüm nesneler, oynadıkları roller ve nesne-rol ilişkisinde ortaya çıkan sınırlılıklar tıpkı dilin gramer yapısında olduğu gibi belirlenir. Dolayısıyla söyleşi evreni, sistem ile kullanıcı arasında gerçekleşmesi planlanan karşılıklı konuşmanın mantıksal yapısını oluşturur. Nijssen buradan yola çıkarak söyleşi evreninin sistemin kavramsal düzeyinde tanımlandığını ve kavramsal şemanın bir cümleler kümesi

olarak nitelenebileceğini söylemektedir. Böylelikle Nijssen veri tabanı ile kavramsal şemanın birbiri ile aynı olduğunu savunmakta ve her ikisini de bilgi tabanı (knowledge base) olarak tanımlamaktadır [23]. Nijssen, bu tanımlamayı **meta** ilkesine dayanarak yapmaktadır. **Meta** ilkesi, inceleme konusu olan nesneyi "meta" olarak nitelemektedir. Veri tabanı için **doğru** olarak kabul edilen tüm durumlar **meta şema** ile bütünüyle belirtilebilmekte ve kavramsal meta dili kullanıcıya çok çeşitli kavramsal şemalar üretme olanağı tanımaktadır. Çoğu veritabanı sistemi "kapalı dünya varsayımı"na (closed world assumption) dayalı olarak çalışmakta ve iletişim evreni ile ilgili tüm bilgi (enformasyon) bilgi tabanı (knowledge base)ndan elde edilebilmektedir. Yani veritabanına yüklenmemiş bir bilgiyi sistem yok kabul eder ve bunun üzerine bir değerlendirme yapamaz.

Bir enformasyon sistemi geliştirme sorumluluğu ile karşı karşıya kalındığıda bu sürecin dört aşamada çözümlenebileceğini görüyoruz:

1. problemin tanımı;
2. bir plan geliştirilmesi,
3. planın uygulanması;
4. sonucun değerlendirilmesi,

Yukarıda anılan ilk iki aşama sistemin hangi işlemler için geliştirileceği veya hange gereksinimleri karşılamak üzere yapılandırılacağını gösteren çözümleme sürecidir. Tanımlanan problem ışığında ge-

23. Gerardus Maria Nijssen-Terence Aldan Halphin, Conceptual Schema and Relational Database Design; a fact oriented approach. New York: Prentice Hall, 1989, 11 s.

liştirilecek plan, sistemde kavramsal düzeyde kavramsal şema biçiminde gerçekleştirilir. Çünkü amaç söyleşi evreninin tanımlanmasıdır.

Kavramsal şema tasarımı için Nijssen ve Folkenberg bir yöntem gelişltirmiştir. Folkenberg'in dilbilimsi Filmore'un çalışmalarından etkilenerek önerdiği ve Nijssen ile birlikte geliştirdiği bu yöntem, şema tasarımını doğal dil cümlelerine dayandırmaktadır. Nijssen ise bu yönteme belirgin örneklerden yola çıkarak iyi tanımlanmış bir yöntem geliştirmek suretiyle katkıda bulunmuştur [24].

Şema tasarımı için dokuz adım belirlenmiştir. Söz konusu aşamalar şöyle sıralanabilir:

1. Bilgi sağlanacak örneklerin yalın gerçeklere dönüştürülmesi ve nitelik denetiminin yapılması;

2. Kavramsal şemanın ilk taslağının çizilmesi ve nicelik denetiminin yapılması;

3. Yinelenen varlık tiplerinin ve ortak rollerin elenmesi ve türetilen her gerçek tipinin tanımının yapılması;

4. Her gerçek tipine teklik sınırlılığının getirilmesi;

5. Gerçek tiplerinin doğru dereceye sahip olup olmadığının denetlenmesi;

6. Varlık tipi, zorunlu rol, alttip ve olgu sıklığı sınırlılıklarının eklenmesi;

7. Her varlığın tanımlanabilirliğinin denetlenmesi;

8. Eşitlik, dışlama (exclusion), altküme ve diğer sınırlılıkların eklenmesi;

---

24. Aynı, 31 s.

9. Orijinal örnekler ile kavramsal şema tutarlılığının denetlenmesi ve yineleme ile bütünlüğün sağlanması [25].

Kavramsal Şema Tasarımında Birinci Adım:

Kavramsal şema tasarımı iki koşul için geliştirilmektedir:

1. Önceden bilgisayar ya da elle yapılan bir uygulama için,

2. Henüz kurulmamış bir sistem için

Birinci koşul söz konusu olduğunda kullanılmakta olan girdi ve çıktı formlarından yararlanılır. Bu formlar rapor, tablo, grafik yada metin biçimde olabilir. Sistem için öngörülen söyleşi evrenine ilişkin gerekli bilgiler anılan formlardan elde edilebilir.

Ancak çıktı formlar her zaman yeterli bilgiyi sağlamada olumlu sonuç vermesine rağmen girdi formlar için aynı şeyi söylemek olanaklı değildir. Bu bakımdan en iyi yöntem bilgi kaybını önlemek açısından her iki formun da birlikte değerlendirilmesidir.

Ortada henüz bir sistemin olmadığı ve kavramsal şemanın ilk kez geliştirileceği ikinci koşulda kullanıcının bilgisine başvurulur. Kullanıcıdan spesifik örnekler ortaya koyması istenir. Söz konusu örnekler daha sonra yalın gerçeklere dönüştürülür ve veritabanı, kavramsal düzeyde bu yalın gerçekler çerçevesinde meydana getirilir.

---

25. Aynı, 32 s.

Söyleşi evreni bizim gerçek dünyadaki uyglamamızın sistemdeki tasarımı olduğuna göre ve söyleşi evreni uygulamada yeralan nesneler ve bu nesnelerin oynadığı rollerin bir kümesi biçiminde nitelenebildiğine göre, bu durumda "yalın gerçek" nesnenin oynadığı belirgin rol (ya da sahip olduğu özellik) ya da söyleşi evrenindeki cümlelerin "doğru" olarak değerlendirilmesi anlamına gelmektedir. "Yalın" sözcüğü gerçeğin daha küçük parçalara bölünemeyeceğini ifade etmek için kullanılır. Bu bakımdan yalın gerçekler bağlaç içermezler.

Örnek: Nutuk iki cilttir; gibi bir ifade "Nutuk" isimli kitabın (nesnenin) iki cilde sahip olduğu yalın gerçeğini ortaya koymaktadır ve bu yalın gerçeğin parçalanması söz konusu değildir.

Söyleşi evreni çerçevesinde tanımlanan ve sistem tarafından "doğru" olarak kabul edilen yalın gerçekler dışında "olumsuzluk" içeren durumlar **kapalı dünya varsayımı**'ndan dolayı sisteme dahil edilmez. Varlıklar ile ilgili tüm bilgi (information) bilgi tabanından (knowledge base) sağlanabilir. Bu bakımdan varlıklar ile ilgisi bulunmadığı düşünülen bilgiler aynı zamanda son derece kapsamlı oldukları için sisteme yüklenmezler. Burada sözü edilen **varlık**, bir nesne, bir insan veya bir olgudur. Bir varlık tipinin üyesi olan varlıklar, veri tabanında terimler ile ifade edilir. Söyleşi evreninde her varlığın belirgin bir rol oynadığı gerçeğinden yola çıkarsak her varlığın bulunduğu eylem bir çıkarım (predikant) olarak değerlendirilir. En basit çıkarım, tekli çıkarımdır. Bu durumda nesnenin oynadığı rol onun sahip olduğu bir özellik olarak da değerlendirilmektedir. Çıkarımı ifade etmek için niteliyici bir kalıp

kullanmak olanaklıdır. Söz konusu kalıp içinde nesne (..) iki nokta ile temsil edilir.

Örneğin : ".. iki cilttir" tekli bir çıkarımdır. Çıkarımın derecesi >n'e kadar olabilir. Derece >2 olduğunda çıkarım, çoklu olarak nitelendirilir. Tekli çıkarımda daha önce de beliirttiğimiz gibi eylem bir özellik olarak ele alınabilmesine karşılık çoklu çıkarımda varlıkların oynadığı roller "ilişki" (relation) biçiminde değerlendirilir.

Doğal dilde varlıklara yönelmenin iki yolu bulunmaktadır:

1. belirli bir nesneyi ifade etmek için bir isim ya da değerin kullanılması,

2. nitelemeyi sağlayacak belirli bir özelliğin kullanılması.

Yukarıda sözü edilen yollardan hangisi kullanılırsa kullanılsın sistemde herhangi bir çelişki ya da kargaşaya yol açmamak için değerler ile varlıklar arasında hangi niteleme öğeleri aracılığıyla ilgi kurulduğu ve varlıkların hangi varlık tipinin üyesi olduğu belirtilmelidir.

Örnek: Elimizde yazarları farklı iki kitabın olduğunu düşünelim. Kitaplardan birinin yazarı "Kemal Tahir", diğerinin yazarı "Yaşar Kemal" olsun. bu durumda "Kemal, İnce Memed'in yazarıdır;" gibi bir çıkarımda İnce Memed'in hangi Kemal'in eseri olduğunu anlamak güçtür. O nedenle "Kemal" değeri ile varlığı yani kişi'yi biraraya getire-

cek,ilişkilendirecek ve diğer kişi ile ayrımını verecek bir niteleme öğesi belirlenmelidir.

Çıkarımda belirsizlik nedeni ile karmaşa yaratacak bir diğer konu Ince Memed'in bir kitap mı, yoksa bir makale mi olduğudur.

Böylesi durumlar **varlık** ile **değer** arasında ilgi kurarak çözümlenir.Söz konusu "ilgi" ya da "bağıntı" **niteleme öğesi** aracılığı ile kurulur.

Ancak isim ve soyadı benzerliğinden dolayı karşılaşabileceğimiz sorunları/karmaşayı ortadan kaldırmak için en iyi çözüm, kişileri isim-soyad olmak üzere tam isimleri ile nitelemektir.

Böylece kişinin tam isminin dikkate alındığını gösteren **yazaradı** niteleme öğesi ile başlığı ifade eden **eseradı** niteleme öğesini kullanmak olanaklıdır.

Dolayısıyla "Yazaradı "Yaşar Kemal" olan KİŞİ, Eseradı "İnce Memed" olan ESER'in yazarıdır", şeklinde bir çıkarım oluşturulabilir.

Tablo biçiminde konuya şöyle açıklık getirmek olanaklıdır.

Tablo 4.1: Varlık, niteleme öğesi ve değer arasındaki ilişki

Yarlık Tipi	KİŞİ	ESER
Niteleme Öğesi	Yazaradı	Eseradı
Değer	"Yaşar Kemal"	"İnce Memed"

Varlık Niteleme Tablosu

Yalın gerçek, bir önerme biçiminin veya mantıksal bir çıkarımın değerlendirilmesi olarak düşünülebilir. Bir başka deyişle "**yalın gerçek**" bir önerme sonucunun "doğru" ya da "yanlış" olduğunu ortaya koyar. Yukarıdaki örnekten de anlaşılacağı gibi bir çıkarım birden fazla varlık hatta bir dizi varlık içerebilir ancak bu durumda varlıkların oynadığı rollerin karışmaması bakımından sıraları önem taşır. Örneğin "yazar-eser" ilişkisinin ifade edildiği bir tabloda;

Tablo 4.2: Varlık niteleme tablosu

Yazaradı	Eseradı	Niteleme Öğesi
Yaşar Kemal	İnce Memed	
Orhan Pamuk	Kara Kitap	
Azra Erhat	Mavi Yolculuk	Varlık Tipleri
Pınar Kür	Küçük Oyuncu	
—	—	
—	—	
—	—	
—	—	

verilmek istenen bilgiyi, (tablonun birinci sırasını kullanarak) yalın gerçek biçiminde şöyle ifade edebiliriz.

1. Yazaradı "Yaşar Kemal" olan KİŞİ, Eseradı "İnce Memed" olan ESER'in **yazarıdır**.

2. Eseradı "İnce Memed" olan ESER, Yazaradı "Yaşar Kemal" olan KİŞİ'nin **eseridir**.

Burada KİŞİ ve ESER varlık tiplerini, yazaradı ve eseradı niteleme öğelerini, "Kemal" ve "İnce Memed" niteleme öğelerinin aldığı değerleri belirtir.

"Yazarıdır" ve "eseridir" sözcükleri ise eylem yüklenmiş olan çıkarımları ifade eder.

Benimsenen temel yaklaşım varlıkların dikkatlice tanımlanması ve oynadıkları rollerin açıkca ifade edilmesidir. Bu yaklaşımı, dilin doğal yapısı içinde uygulamaya dönüştürmek ise ana amaçtır. O nedenle doğal dilde ifade edilen ve yukarıda örneği verilen türde gerçek tiplerinin isimlendirilmesi gerekir. Eğer gerçek tipi yerine mantıksal çıkarımlar kullanılacaksa yukarıda anılan 1. ve 2. cümlelerde olduğu gibi "yazarıdır" ya da "eseridir" çıkarımlarından biri seçilmelidir.

Yine, yukarıdaki örnekte olduğu gibi çıkarımlar için farklı isimler kullanılacak olursa o zaman bu isimler, rolleri belirlemek için **gerçek tipi ismi** olarak da ele alınabilir.

Bir "gerçek tipi" için söz konusu olan tüm varlık tipleri farklı ise varlık tiplerinin ismi roller için kullanılabilir [26].

Gerçek, aynı zamanda (varlık, rol) çiftinden oluşan bir küme olarak da isimlendirilebilir. Örneğin, G $((V_1,r_1), (V_2,r_2)...(V_n,r_n))$

Yazan ((Yazaradı "Yaşar Kemal" olan KİŞİ, yazarıdır)
Eser ((Eseradı "İnce Memed" olan ESER, eseridir)

Bir "rol" bir varlık tipinin bir ilişki içindeki katılımını gösterir. Tekli ilişkide varlık tek bir rol oynar ve "rol"ün de tek (unique) olması zorunludur.

---

26. Aynı, 42 s.

Kimi zaman nitelemelerin de "varlık" olarak kabul edildiği durumlar vardır.

Örneğin : 1. ISBN, bir yayın numarasıdır.

2. "ISBN" sekiz rakkamdan oluşur.

Burada 1. cümlede "ISBN" yayına ilişkin bir numarının ismi yani bir değerdir. (sipariş no'su, yer no'su gibi yayına ilişkin numaralardan biridir)

2. cümlede ise "ISBN"nin kendisi bir varlık olarak ele alınmıştır.

Yalın Gerçeklerin Nitelik Kontrolünün Yapılması: Nitelik kontrolünün yapılması için şu iki sorunun yanıtlanması gerekmektedir:

1. Varlıklar iyi tanımlanmış mıdır?
2. Gerçekler, bilgi kaybına uğramadan daha küçük gerçeklere ayrılabilir mi?

Burada üzerinde önemle durulması gereken konu gerçek tiplerinin unutulmamasıdır. O nedenle bir "örnek gerçek tablosu" çizilir. Böyle bir tablonun üst kısmında gerçek tipleri; varlık tipi, niteleme öğeleri ve roller olarak verilir.

Bu durumda tablonun yalnız üst kısmı kavramsal şema ile ilgilidir. Tablonun alt kısmı ise veritabanına yüklenmiş değerlerin kümesini vermektedir.

Örnek 1: Eseradı "Information Transfer" olan ESER, 08.03.1995 tarihinde, bölüm adı "Kütüphanecilik" olan BÖLÜM tarafından sipariş edilmiştir.

..,..,.. tarafından sipariş edilmiştir.

Sipariş Tablosu

Tablo 4.3 : Kavramsal şemada yeralacak alanların tablo aracılığı ile saptanması

ESER	TARİH	BÖLÜM
Eseradı	Sip. tar.	Bölümadı
Information Transfer	08.02.95	Kütüphanecilik

Daha önce de belirttiğimiz gibi yalın gerçek, daha küçük parçalara ayrışmamalıdır. Cümle içinde yeralan bağlaçlar ayrışabilirliğin göstergesidir. Cümlede herhangi bir bağlaç kullanılmamasına karşılık cümle iki yalın gerçek biçiminde ifade edilebilir. Buna göre:

2a) "Eseradı "Information Transfer" olan ESER, 08.03.1995 TARİHİNDE sipariş edilmiştir.

..,..,.. sipariş edilmiştir.

2b) "Eseradı "Information Transfer" olan ESER, Bölümadı "Kütüphanecilik" olan BÖLÜM tarafından sipariş edilmiştir.

..,..,.. tarafından sipariş edilmiştir.

Tablo 2a

ESER	TARİH
Eseradı	Sip. tar.

—> Varlık tipleri <-

—> Niteleme Öğeleri <-

—> Değerler <-

Tablo 2b

ESER	BÖLÜM
Eseradı	Bölümadı

2. Tablonun 2a ve 2b biçiminde ayrışması halinde 2a ve 2b cümlelerinden de anlaşılacağı gibi herhangi bir bilgi kaybı olmamaktadır. Bu durumda herhangi bir bağlaç kullanılmamasına rağmen 2. cümle ayrıştırılabilir.

Ancak kimi durumlarda tablonun ayrışması halinde bilgi kaybı meydana gelebilmektedir.

Örnek 2: Eseradı "An Introduction to Database Systems" olan ESER'in 2. basımının Yayın Tarihi 1977 dir. (Aynı eserin başka basımları da bulunmaktadır. Hangi basımın sipariş edileceğini bilmek ve bibliyoğrafik doğrulamayı yapmak önem taşır) Yalın gerçeği tablo biçiminde ifade edersek,

Tablo 3

Eseradı	Basım	Yayın Tarihi
An. Int....	2.	1977
An. Int....	4.	1985

Tablo ayrıştığında;

Tablo 3a

Eseradı	Basım
An. Int....	2.
An. Int....	4.

Tablo 3b

Basım	Yayın Tarihi
2.	1977
4.	1985

3a. Eseradı "An Introduction to Database systems@ olan ESER'in 2. basımı vardır.

3b. 2. basımın yayın tarihi 1977'dir.

Yukarıdaki ayrışma dikkatlice incelendiğinde görülür. 1. tablodan eserin kaç basıma sahip olduğunu anlamak olanaklıyken 2. tablodan **hangi eserin** basımlarına ilişkin tarihlerin verildiğini çıkarmak olanaklı değildir. Dolayısıyla burada bir bilgi kaybı söz konusudur.

Tablonun bir başka biçimde ayrışması da olanaklıdır.

Örneğin:

Tablo 3aa

Eseradı	Basım
An. Int....	2.
An. Int....	4.

Tablo 3bb

Eseradı	Yayın Tarihi
An. Int....	1977
An. Int....	1985

Tablo 3bb de esere ilişkin iki yayın tarihi görülmektedir. Ancak bu tarihlerin hangi eserin basımına ait olduğunu anlamak olanaklı değildir. Dolayısıyla bu durumda da bilgi kaybı söz konusudur.

2. ADIM ILK TASLAK

Eserin sipariş edildiği firmanın gösterildiği bir tablodan yararlanarak temel gerçekleri doğal dilde şöyle ifade edebiliriz:

Tablo 4.

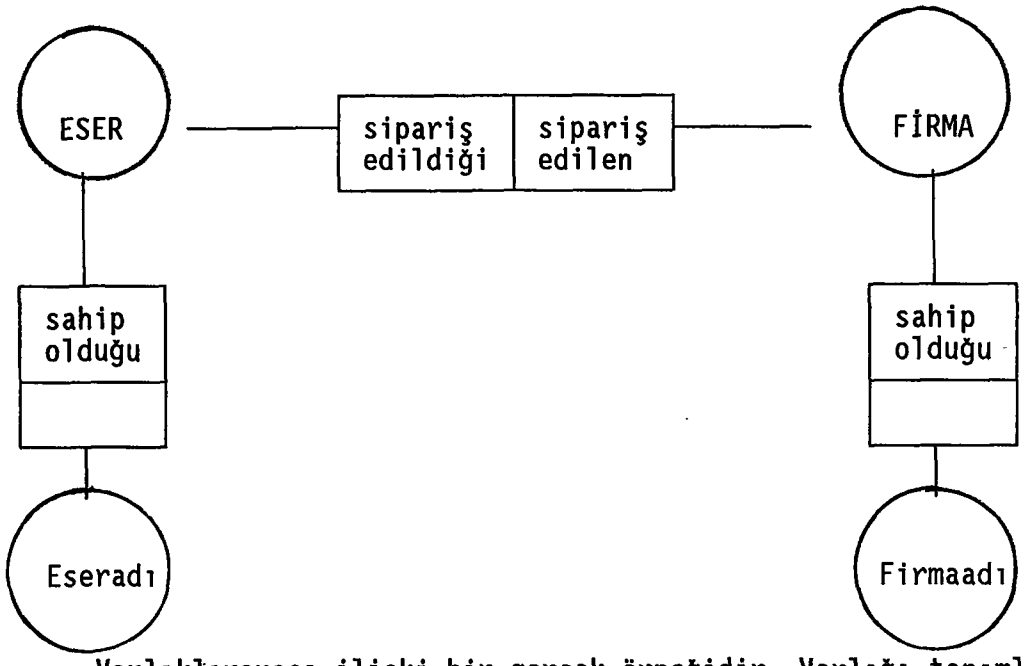
ESER	FİRMA
Eseradı	Firmaadı
E_1	F_1
E_2	F_2
E_3	F_3

1. Eseradı "E_1" olan ESER, Firmaadı "F_1" olan FİRMA'ya sipariş edilmiştir.

2. Eseradı "E_2" olan ESER, Firmaadı "F_1" olan FİRMA'ya sipariş edilmiştir.

3. Eseradı "E_3" olan ESER, Firmaadı "F_2" olan FİRMA'ya sipariş edilmiştir.

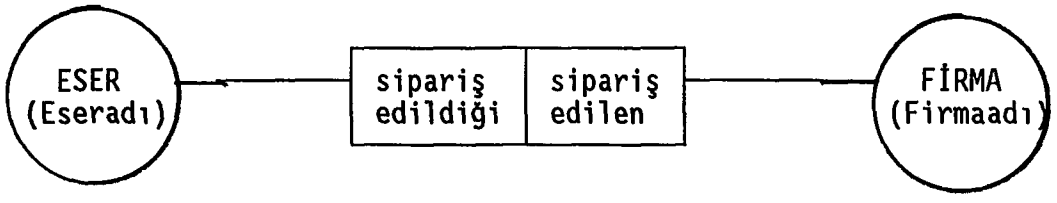
Yukarıda ifade edilen yalın gerçeklerden yola çıkarak bir kavramsal şema oluşturmamız gerekmektedir. Ancak daha önce çizeceğimiz bir şema diagramı varlıklar ve niteleme öğeleri arasındaki ilişkiyi daha net görmemizi sağlayacaktır. Böyle bir çizimde varlıklar daire biçiminde ki yuvarlaklar ile roller ise dikdörtgen biçimindeki kutucuklar ile ifade edilmektedir.



Varlıklararası ilişki bir gerçek örneğidir. Varlığı tanımlamak için kullanılan niteleme öğesi aynı zamanda o varlığın bir özelliğini yansıtır.

Yukarıdaki çizimde bir gerçek tipi ve iki niteleme öğesi vardır.

Söyleşi evreninde her eserin bir adı vardır ve herbir eseradı da yalnız bir esere karşılık gelir. Aynı durum firma için de söz konusudur. Bu durumda niteleme öğesi varlık ile değer arasında birebir 1:1 ilişki sağladığı için niteleme öğesi parantez () içinde belirtilebilir.



Yukarıda örneği görülen bir "kavramsal şema diagramı" iki amaca hizmet eder.

1. İletişim evreninin basit ve açık bir görünümünü verir.

2. Bilgi tabanı (knowledgebase) için bilgisayarın kavramsal olarak üstesinden gelebileceği biçimsel bir belirginlik sağlar.

Gerçek tipi, eğer aynı varlık tarafından oynanan rollerden ibaretse homojen,ayrı varlık tiplerini içeriyorsa heterojen olarak adlandırılır. Kavramsal şema çiziminde aynı anlamı veren farklı kavramsal şema diagramlarına dönüşüm sağlanabilir. Dönüşüm yollarından biri yuvalama'dır. Yuvalama, birbiri ile ilgisi olma zorunluğu bulunmayan, iç içe geçmiş parçalar anlamına gelir, temel farkı nesneler arası ilişkiyi de bir varlık gibi ele almasıdır. İç içe geçmiş gerçek tipleri ilgili roller etrafına çizilen bir elips ile ifade edilir ve rol, nesnel bir durum kazanır. Bu durumda;

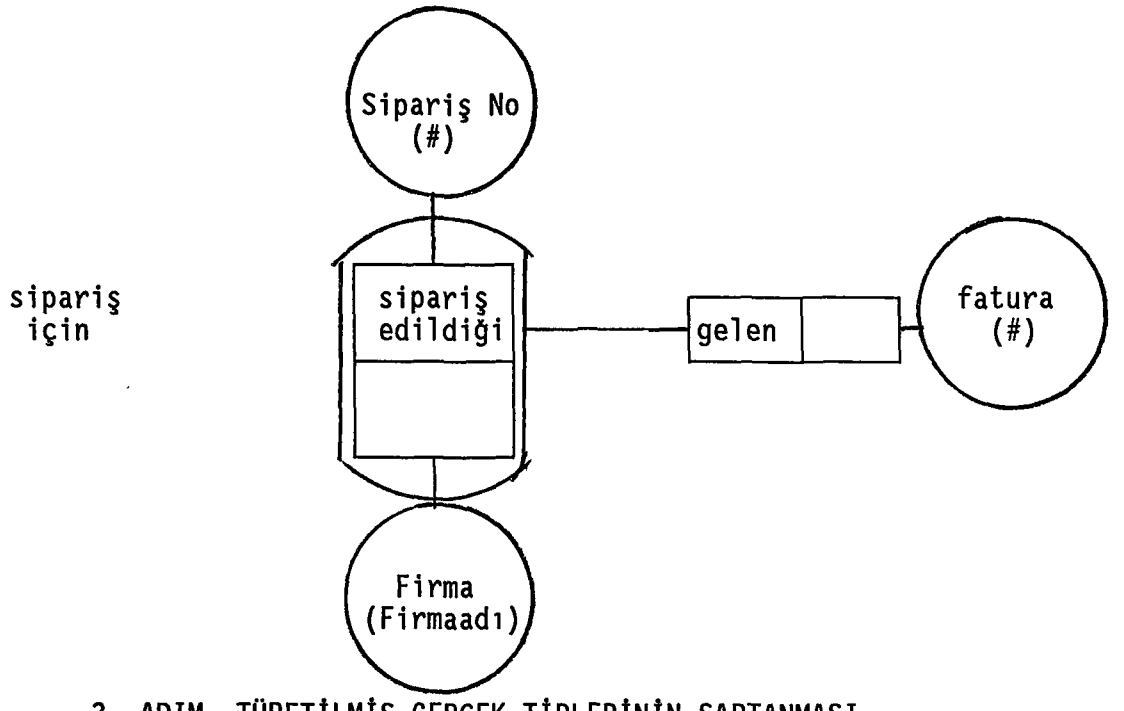
Nesneleri üç grupta toplayabiliriz:

1. Söyleşi evreninde konuşma konusu yaptığımız, basit nesneler,

2. Varlıklarla ilgili olup, veritabanında yeralan sözcüklerle ifade edilen nesneler,

3. Nesne niteliği kazandırılan nesnelerarası ilişkiden oluşan nesneler.

Örnek: Bir yayının sipariş numarası alarak bir firmaya sipariş edilmesi ve o numara ile siparişi yapılan firmadan gelen faturaya ilişkin bilgi bütününü kavramsal şema çiziminde yuvalama kullanarak şöyle ifade edebiliriz.



3. ADIM TÜRETİLMİŞ GERÇEK TİPLERİNİN SAPTANMASI

Bu aşamada gereksiz varlık tipleri ayıklanır ve varlıklar uygun varlık tipleri altında gruplandırılır. Varlık tipleri saptanırken ortak özelliklerin bulunmamasına yani karşılıklı teklik (mutually exclusive) ilkesine uymaya özen gösterilir. Dolayısıyla birbirini kapsayan varlık tiplerinin bulunmaması gerekir. 3. adımın uygulanması sırasında üzerinde durulması gereken noktaları özetlemek gerekirse:

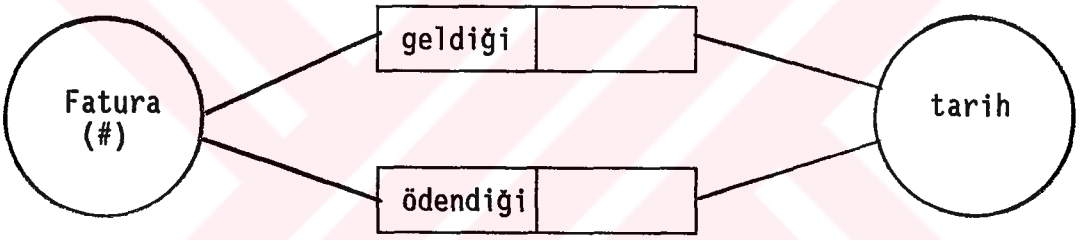
1. Bir varlık, iki ayrı varlık tipinin üyesi olabilir mi? Eğer olabiliyor ise varlık tipleri birleştirilmelidir.

2. İki ayrı tipteki varlık anlamlı biçimde karşılaştırılabiliyor mu? (Örn: oranların hesaplanması)
   Eğer böyle ise, varlık tipleri tek bir varlık tipinde birleştirilebilir.

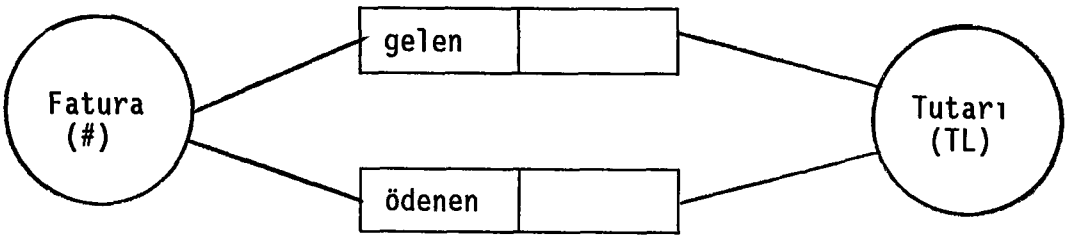
3. İki farklı varlık tipinin tüm elemanları aynı rolü oynayabiliyor mu? Eğer öyleyse varlık tipleri birleştirilebilir ve gerekiyorsa bu ayrımı korumak için bir başka varlık tipi eklenebilir.

4. Bir gerçek tipi diğerlerinden türetilebiliyor mu?
   Eğer öyle ise gerçek tipi '*' ile işaretlenir ve türetme kuralı belirtilir.(27)

Örnek 1 : "Gelen fatura", "ödenen fatura" gibi aynı varlık tipine ait ayrı niteleme öğeleri söz konusu olduğunda ve bunların aynı birime sahip olması durumunda şema diagramı şöyle çizilebilir;

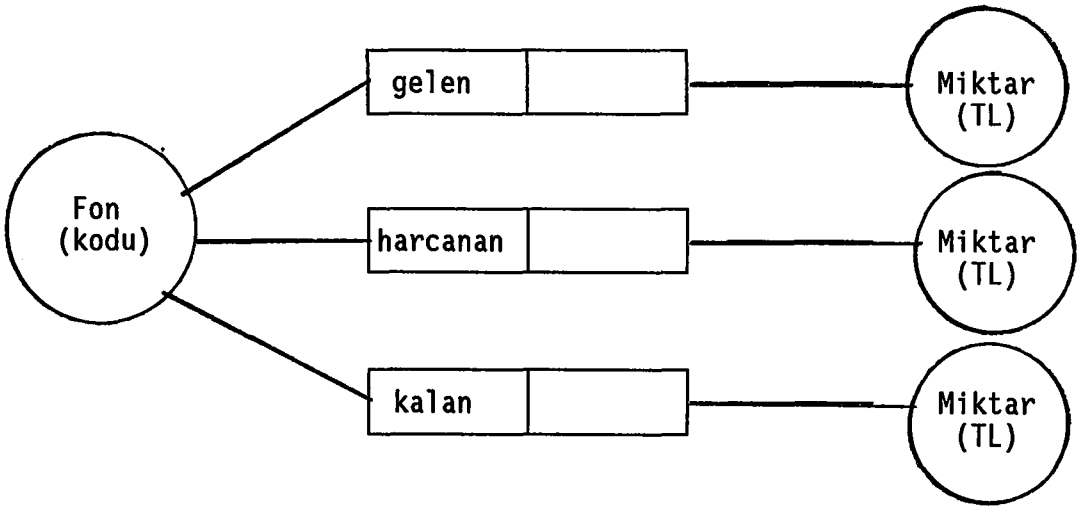


Bu aynı zamanda fatura tutarı için de yapılabilir. Bu durumda şema diagramı şöyle bir görünüm sergiler;



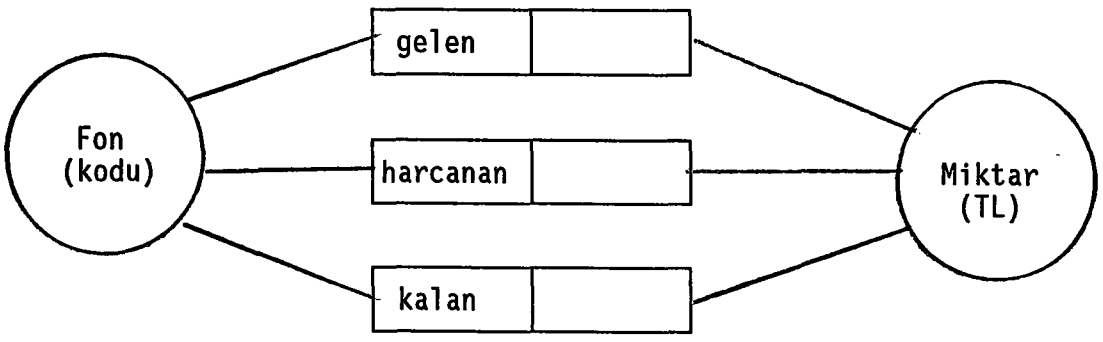
27. Aynı, 66 s.

Örnek 2 : Türetme gerçek tipi söz konusu olduğunda;



Görülüyor ki roller şema çiziminiin sol tarafında aynı özelliğe ve birime sahiptir. Dolayısıyla söz konusu özellikler birleştirilmelidir. Bu durumda şema diagramı şöyle bir görünüm kazanır.

F'de kalan miktar K'dir. Eğer F'ye gelen miktar G ve F'den harcanan miktar H ve K = (G-H) ise. Şema diagramında türetme kuralı şemanın altına yazılabilir. diagramda kolay görülebilmesi içinde ilgili rolün yanına "*" konarak belirgin hale getirilir.

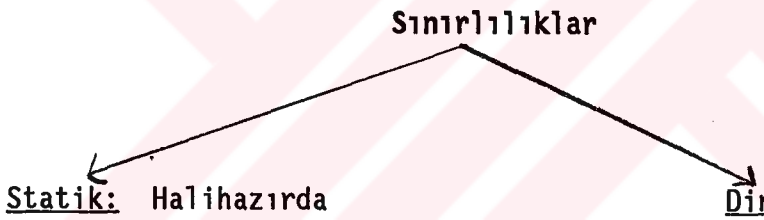


"F'de kalan miktar K'dir. Eğer F'ye gelen miktar G ve F'den harcanan miktar H ve K = (G-H) ise" Şema diagramında türetme kuralı şemanın altına yazılabilir. diagramda kolay görülebilmesi içinde ilgili rolün yanına "*" konarak belirgin hale getirilir.

4. ADIM TEKLİK SINIRLILIĞI

Bu aşamaya kadar veritabanına yüklenmesi planlanan ve bunlardan türetilebilecek gerçek tiplerinin belirlenmesine çalışılmıştı. Bu noktada ise gerçek tipleri üzerine olan sınırlılıkların saptanması konusu üzerinde durulacaktır. Nijssen sınırlılıkları;

1) Statik ve
2) Dinamik olmak üzere iki kategoride ele almıştır [28].



Statik: Halihazırda şemada bulunan sınırlılıklar

Dinamik : Doğal dilde veya geçiş(bağlantı) tablolarında ifade edilebilen sınırlılıklar.

Statik Sınırlılıklar :

Veri tabanının zaman içindeki değişimleri de gözönünde bulundurularak veri tabanının olası her durumuna uygulanabilen sınırlılıklardır. Statik sınırlılıklar, gerçek tipine ait bütün populasyon için uygulanır [29].

---

28. Aynı, 66 s.
29. Aynı, 66 s.

Örneğin : G1 Yayın, Bölüm tarafından sipariş edilir.

S1 Yayın en fazla bir Bölüm tarafından sipariş edilir.

Zaman içinde enformasyon sisteminde veritabanı durumları değişebilir. Örneğin, youn bir bölüm tarafından sipariş edilmişken ve bu bilgi veritabanına girilmişken bir başka Bölüm yine aynı yayının siparişi için talepte bulunabilir. Böyle bir durumda eğer kuruluşun politikası uyarınca S1 deki gibi bir sınırlılık belirlenmişse sistem söz konusu sınırlılık uyarınca bu yeni bilgiyi reddeder.

Gerçek Tipinde Tetlik Sınırlılığı

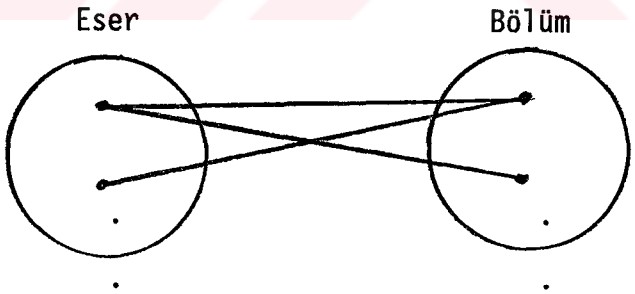
Tekli gerçek tipinde tetlik sınırlılığı söz konusu olduğunda örneğin yayın adına göre düzenlenmiş bir siparişlekiler listesinde güncelleme yapılırken daha önce listede yeralan bir eser yanlışlıkla yeniden listeye girilebilir. Böyle bir durumda küme kuralları gereğince veritabanının eleman sayısında herhangi bir artış gözlenmez. Ancak aynı bilgi veritabanında ayrı fiziksel konumlarda yeralır. Dolayısıyla veri tabanında yineleme meydana gelir. Bu durumda veritabanı **gerçekler kümesi** yerine **gerçekler torbası** (bag) olarak görülür. Torba (bag) nın kümeden farkı, eleman tekrarının belirtilmesidir. Örneğin; (a) ve (a,a) kümeleri aynı iken (a) ve (a,a) torbaları farklıdır. Bununla beraber kavramsal şema tablolarında yeralan herbir sıranın bir yalın gerçeğe karşılık geldiği ve hiçbir sıranın birbirinin aynı olması kuralı anımsandığında yukarıdaki gibi bir yinelemenin söz konusu kurala aykırı olduğu görülür. Veritabanında yinelemenin getirdiği olumsuzluklar şöyle sıralanabilir:

1. eğer veri, veritabanında birden fazla yerde bulunuyorsa verinin silinmesi durumunda silme işlemi verinin bulunduğu tüm konumlarda gerçekleştirilmelidir. Özellikle geniş veritabanlarında böyle bir durum oldukça fazla zaman kaybına yolaçar,

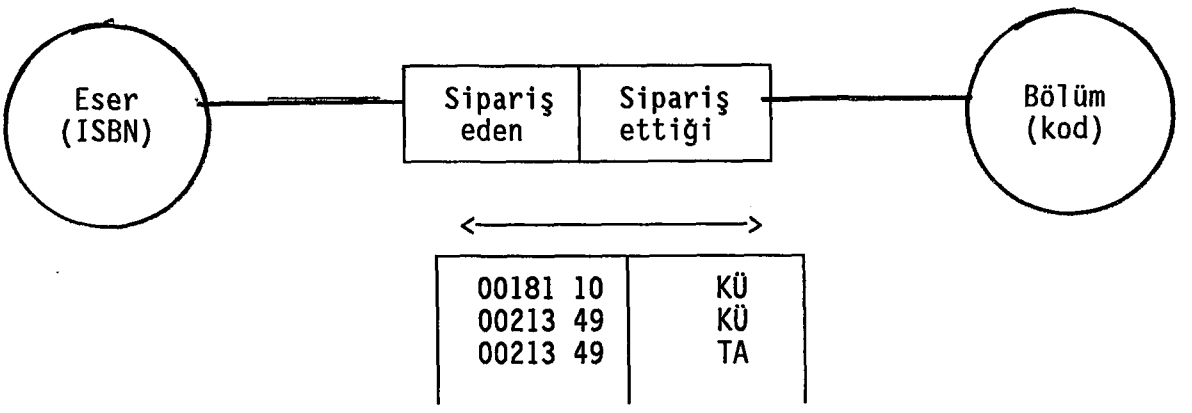
2. bunun yanı sıra verinin birden fazla yerde bulunması depolama alanının sınırlılığı düşünüldüğünde yer kaybına neden olur.

Ancak yinelemenin anılan olumsuzluklarına karşılık bilgi erişimini verimli kılmak ve sistemin hızlı çalışmasını sağlamak gibi yararları da vardır.

İkili gerçek tiplerinde teklik: Gerçek tablosunda her sıranın birkez yeralması zorunluluğu çokludan çokluya gibi kimi ilişki türlerinde sorun yeralır. Örneğin, eğer eser birden fazla bölüm tarafından sipariş ediliyor ise;

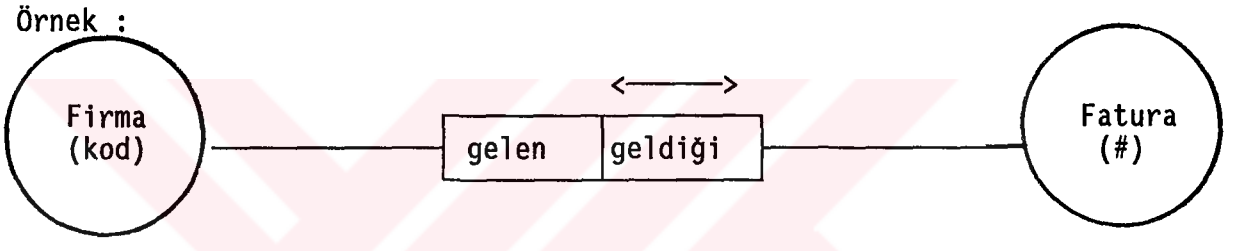


yukarıdaki olgu diagramında da görüldüğü gibi teklik ancak eser-bölüm birlikteliğinde söz konusu olabilmektedir. Bu durumda kavramsal şema diagramında eser-bölüm ilişkisinde teklik şöyle belirtilir:



Çokludan tekliye ilişki türünde örneğin bir firmadan birden fazla fatura gelebilir ancak bir faturanın birden fazla firmadan gelmesi sözkonusu değildir.

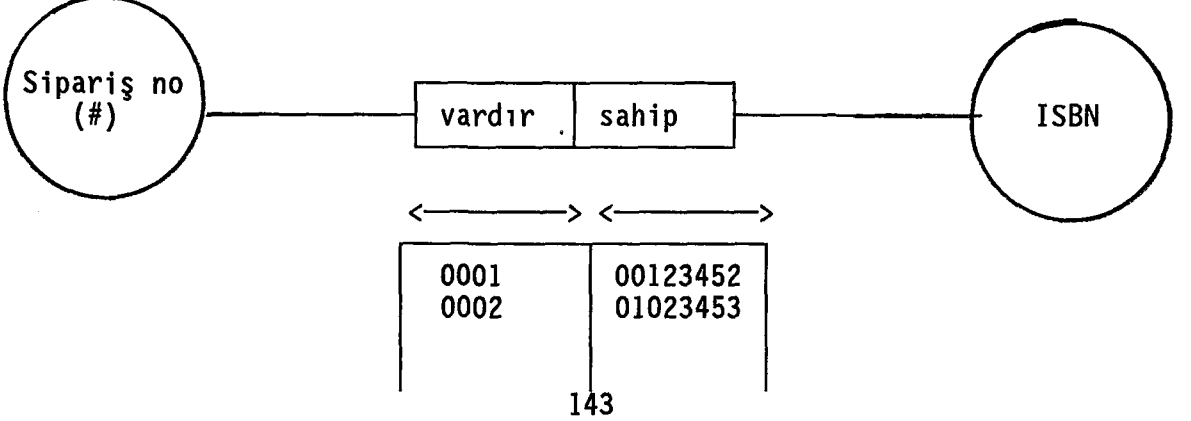
Örnek :



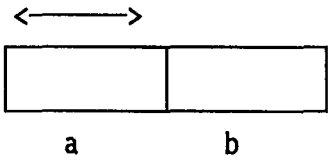
Bu durumda teklik ilkesinin geçerli olduğu sütuna söz konusu işaret konulur.

Tekliden tekliğe ilişki türünde teklik işareti her iki sütuna da ayrı ayrı konmalıdır.

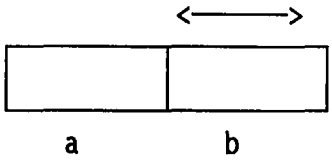
Örnek: Her yayının yalnız bir ISBN numarası olur ve her yayın yalnız bir sipariş numarasına sahiptir. Bu durumda şema diagramı şöyledir:



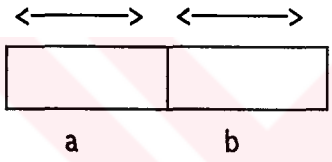
İkili gerçek tiplerindeki dört olası teklik durumu kısaca şöyle özetlenebilir:

a) 

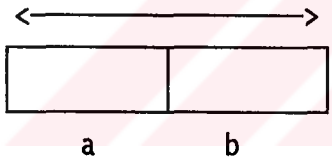
"a'da hiçbir dublikeye izin verilmez. Her "a" R ilişkisinde bir "b" ye gider".

b) 

"b'de hiçbir dublikeye izin verilmez. Her "b" R ilişkisinde bir "a" ya gider."

c) 

"sınırlılıklar a ve b için geçerlidir."



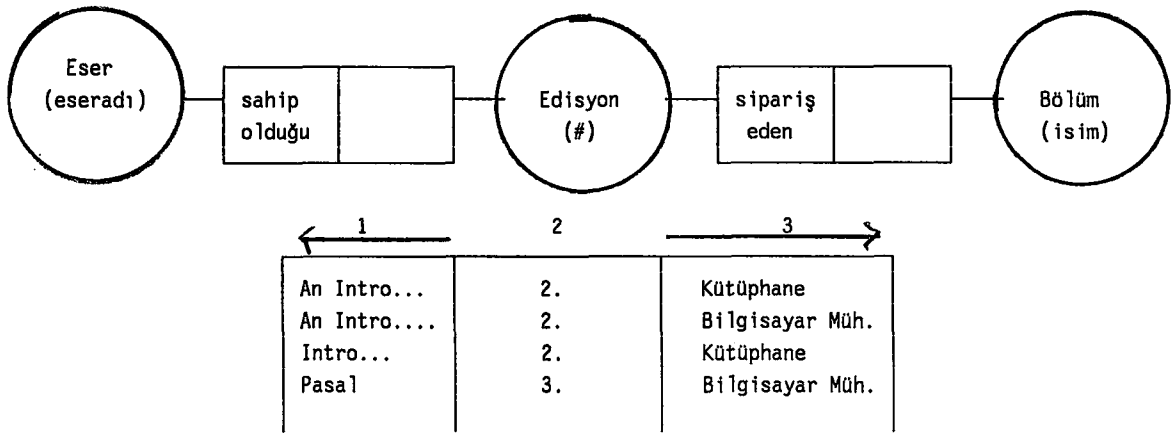
"ab sırasının tekrarlanması söz konusu değildir. Her 'a' R ilişkisinde birden fazla b'ye gidebilir ya da tersi söz konusudur"(30).

Teklik sınırlılığında benimsenen temel ilke sınırlılığın, gerçek yaşamdaki gerçek tipine uygulanacak kadar güçlü olmasıdır.

İkiden fazla Gerçek Tipinde Sınırlılık : Üçlü gerçek tipi söz konusu olduğunda sınırlılığın varlığını saptamak için öncelikle sutunlar tek tek ele alınır. Eğer sütünların herbirinde en az bir "değer" birden fazla miktarda bulunuyor ise a zaman sütun çiftlerine bakılır.

---

30. Aynı, 75 s.



An Intro...	2.	Kütüphane
An Intro....	2.	Bilgisayar Müh.
Intro...	2.	Kütüphane
Pasal	3.	Bilgisayar Müh.

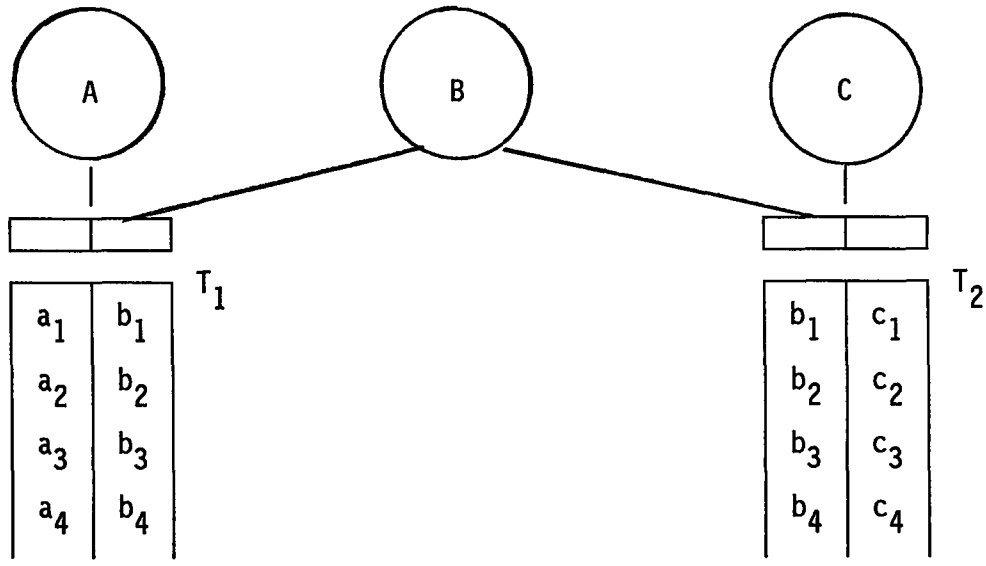
Yukarıdaki örnekte her sütunda en az bir değerin yinelendiği görülmektedir. Bu durumda 1 ve 2 no'lu sütun çifti ele alındığında {(1,2) = (eseradı, edisyon)} 1. ve 2. sıraların birbirinin aynı olduğu görülür. Aynı durum (1,3) = (edisyon, bölüm) sütun çiftinin 1.ve 3. sıraları için geçerlidir. yinelemenin bulunmadığı tek sütun çiftinin 1 ve 3 no'lu sütun çifti olduğu görülür. Dolayısıyla teklik anılan sütun çifti için geçerlidir.

Birleşim ve Yuvalamaya İlişkin Sınırlılıklar

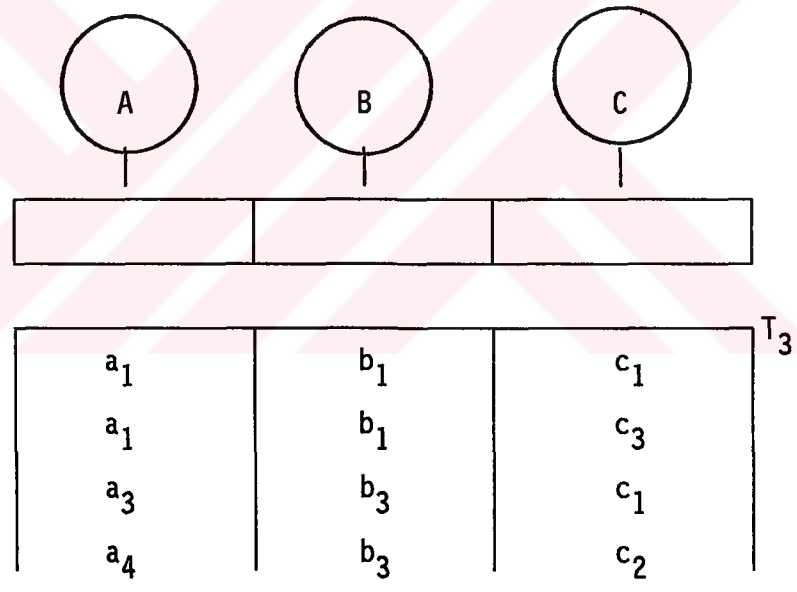
İki tablonun birleştirilerek yeni bir tablo oluşturulması için kullanılan üç tip birleştirme yöntemi vardır.

1. içten doğal birleşim,
2. dıştan doğal birleşim,
3. eşit doğal birleşim.

T_3 gibi yeni bir tablo oluşturmak için T_1 ve T_2 gibi iki tabloyu birleşltirmek sözkonusu olduğunda ve "içten doğal birleşim" yöntemi kullanıldığında T_1 ve T_2 tabloları ortak bir sütun aracılığı ile birleştirilir. Bir çizimle göstermek gerekirse;



B ortak sütunundaki değerlere karşılık gelen a ve c değerleri tek bir sıra oluşturacak biçimde birleşerek aşağıdaki yeni T_3 tablosunu meydana getirir (31).



Burada dikkati çeken konu T_3 tablosundaki yeralan b sütunun daha önceki T_1 ve T_2 tablolarındaki b sütunlarının **kesişiminden** oluşmasıdır.

---

31. Aynı, 83 s.

Bir diğer birleşim tipi **"dıştan doğal birleşim"**dir. Bu yolla yapılan birleşimde ortak sütunun tüm değerleri dikkate alınır. T_1 ve T_2 tablolarında yeralan ortak sütuna karşılık gelen tüm değerler T_4 tablosunda da bulunur. Ortak sütun değerine karşılık T_1 veya T_2 tablolarında herhangi bir değer yoksa eksik olan değerler için örneğin "?" işareti kullanılır. Dolayısıyla ortaya çıkan T_4 tablosunda bulunan ortak sütun değerleri T_1 ve T_2 tablolarındaki ortak sütun değerlerinin **bileşim** kümesini verir [32].

T_4

a_1	b_1	c_1
a_1	b_1	c_3
a_2	b_2	?
a_3	b_3	c_2
a_4	b_3	c_2
?	b_4	c_4

Gerçek tipinin doğru dereceye ulaşıp ulaşmadığını ve ayrışmaların bilgi kaybına neden olmadan yapılabilirliğini denetlemek gereklidir. Bu amaçla tablolar projeksiyon yöntemi ile önce ayrıştırılarak daha sonra birleşim yöntemiyle yeniden biraraya getirilir ve yeni örneklerin ortaya çıkıp çıkmadığı denetlenir.

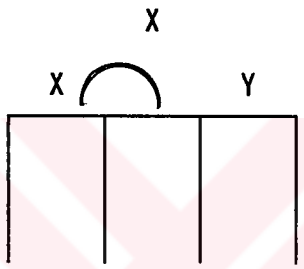
Bir gerçek tipinde teklik sınırlılığı aynı zamanda "anahtar"ın yani erişim ucunun saptanmasında bize yardımcı olur. Böylelikle üzerinde teklik sınırlılığı bulunan rol, aynı zamanda o gerçek tipinin anahtarı durumuna geir. Anahtarın uzunluğu gerçek tipindeki rollerin sayısı hakkında bilgi verir. Anahtar uzunluğunun 1 olması halinde bu, **basit**

32. Aynı, 85 s.

anahtardır. Anahtar uzunluğu > 1 olduğunda bileşik anahtarlardan söz edilir, Anahtarın uzunluğu gerçek tipinin ayrışabilirliğini de denetler. Örneğin gerçek tipi n-1 uzunlukta bir anahtara sahipse n.derecedeki gerçek tipi ayrışabilir. Anahtar uzunluğunun en az 1 olma zorunluluğu dikkate alınırsa anılan kural anahtar uzunluğu > 3 olan gerçek tipleri için geçerlidir.

Geleneksel veritabanı tasarımında ayrışabilirliğin denetlenmesi için **fonksiyonel bağımlılık :** FB saptaması yapılır.

Örnek :



X, tek bir sütun ya da birden fazla sütuna karşılık gelen bileşik bir sütun olabilir. Eğer her X değerine karşılık bir Y değeri bulunuyor ise o zaman Y fonksiyonel olarak X'e bağlımlıdır.

Fonksiyonel bağımlılığın saptanmasında söyleşi evrenini ve oradaki sınırlılıkları iyi bilmek gerekir. FB'nin varlığı hakkında karar verilirken kümedeki eleman sayısının belirginliği önem taşır. FB uyarınca yapılacak ayrışmaya yönelik kurala göre "n.dereceden bir gerçek tipinde uzunluğu (n-1)den küçük olan (uzunluk < (n-1) bir anahtar varsa sözü edilen gerçek tipi tablosunda **anahtar**'a fonksiyonel olarak bağımlı en az iki sütun bulunur. O zaman tablo, herbiri anahtarı içeren iki ayrı tabloya ayrışır. Tablolar daha sonra birleştirilerek orijinal tablonun yeniden elde edilebilirliği denenir [33].

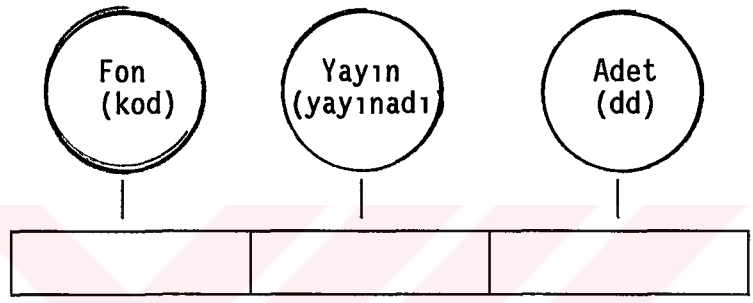
---

33. Aynı, 98 s.

Diğer Sınırlılıklar

Varlık tipi sınırlaması : Kavramsal şemada güncelleme ve erişim işlemlerinde kullanılan temel veri yapısı "gerçek tipi"dir. Gerçek tipi için söz konusu olan bütün roller ve o rollere karşılık gelen sütun değerlerinin tamamı rollerin popolasyonunu dolayısıyla rol popülasyonlarının toplamı da gerçek tipi popülasyonunu verir.

Örnek:



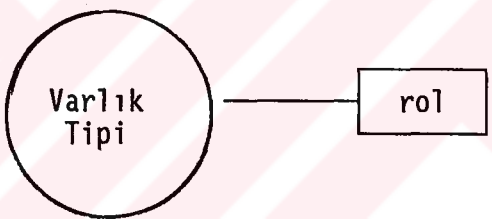
Burada sipariş edilen eserler, bunlardan kaç adet sipariş yapılacağı ve bu siparişlerin hangi fondan karşılanacağı yalın gerçekleri elde edilebilir. Sipariş edilecek yayınlar için herhangi bir sınırlama getirmek olanaklı değildir. Burada ancak sözel bir sınırlama getirilebilir. Örneğin yayın adının en fazla kaç karakterden oluşacağı saptabilir ve <c karakter sayısı > biçiminde ifade edilir. Bazı durumlarda ise alfa-sayısal diziler karşımıza çıkar. o zaman sınırlama <a-add>gibi sözel ve sayısal dizi bileşimi biçiminde verilir.

Kümedeki elemanların tümü aynı zamanda o kümenin "genişliği" hakkında bilgi verir. Örneğin ödemelerin yapılacağı fonlar için belirlenen kodların oluşturduğu küme, { } içinde verilir ve kodların tümü kümenin genişliğini ifade eder.bunun yanı sıra sipariş edilecek eser-

lerin sayısı sınırlandırılabilir. Bilgi merkezi politikası doğrultusunda yapılacak sınırlama [ ] içinde örneğin [1..10] biçiminde ifade edilir [34].

Zorunlu ve Seçimli Roller: Veritabanı tamlolarında kimi zaman bazı sütun değererinin, ya o an elde edilemediği için ya da anılan alan ile ilgili değer bulunmadığı için kaydedilmediği görülür. Bu durumda "değer"in kaydedilmesi yolunda bir zorunluluğun bulunmadığı anlaşılır. **Zorunluluk** veritabanının tüm durumları için ve varlık tipine bağlı olarak rol popülasyonunun tüm elemanları için değerin mutlaka kaydedilmesi gerekliliğidir; aksi halde **seçimli rol** olmamaktadır. Zorunluluk şema çiziminde rolün bağlı olduğu varlık tipine bir "nokta" koymak suretiyle ifade edilir.

Örnek

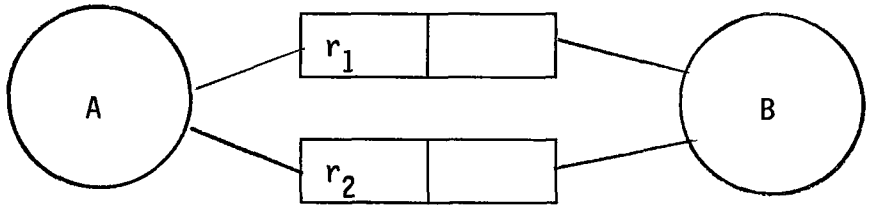


Rolün bağlı bulunduğu varlık tipinde zorunluluğu belirlemek için konan "nokta" işareti, ikili gerçek tipi söz konusu olduğunda her iki rol için de konması halinde bu aynı değerlerin veritabanında her iki rol için de görülebileceği anlamını taşır. İşaretin her iki rol için ayrı ayrı konması halinde ise bu, her iki rol için ayrı ayrı değer kaydetme zorunluluğunun bulunduğu anlamını taşır.

---

34. Aynı, 113 s.

Örnek



A'nın değer kümesindeki elemanlar r_1 ve r_2 rolleri için de görülebilir. Ancak her B elemanı için r_1 ve r_2 değerleri ayrı ayrı kaydedilmek zorundadır.

Zorunlu role göre gerçek dünya ile bilgi tabanı arasındaki ilişki şöyle açıklanabilir. Gerçek yaşamda bir rol seçimli ise bilgi tabanında da seçimlidir. Aynı şekilde bir rol bilgi tabanında zorunlu ise gerçek yaşamda da zorunludur ancak tersi geçerli değildir (35).

Rolün seçimli ya da zorunlu olduğu konusunda bir karara varmadan önce rolün gerçek yaşamda da zorunlu ya da seçimli olduğunu araştırmak gerekir. Zorunluluğun bulunmadığı durumlarda rolün seçimli olma nedenleri incelenir.

Türtilmiş gerçek tipleri söz konusu olduğunda bunlar türetildikleri ve veritabanına doğrudan kaydedilmedikleri için seçimli olarak belirlenirler.

Sipariş işlemlerinin ilk aşaması kullanıcı isteklerinin belirlenmesidir. Bu amaçla kullanıcıların satınalınmasını istediği yayınlar ile ilgili bazı temel bilgiler vermesi istenir.

---

35. Aynı, 117 s.

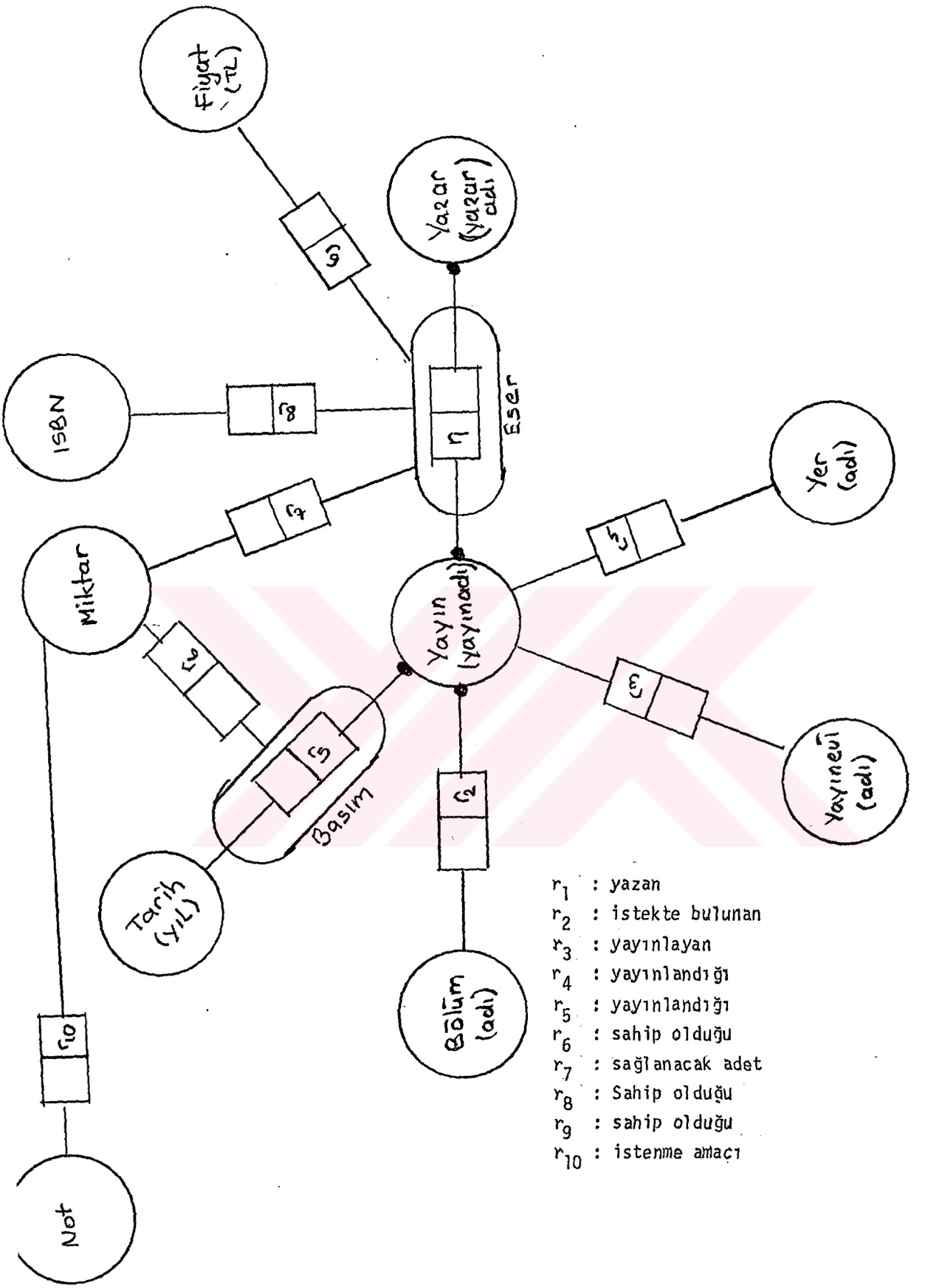
Örnek: Aşağıda örneği verilen türde bir girdi form söz konusu olduğunda;

Yazar(soyad-ad)		ISBN
Yayın adı		
Yayın yeri	Yayınlayan	Yayın tarihi
Cilt sayısı	Basım	fiyat
İsteyen Bölüm		
Not		

kullanıcının doldurması zorunlu alanlar yazar kimliği ve yayınadı alanlarıdır. Bunun yanında kullanıcı hangi bölüm adına istekte bulunduğunu da belirtmek zorundadır. Anılan zorunlu roller ve onların dışında kalan seçimli roller şema diagramında gösterilmiştir (sf. 153).

Küme ve Altkümeler: Küme, iyi tanımlanmış soyut ya da somut olabilen birimler topluluğudur. Birimlere aynı zamanda küme elemanı denir. Küme elemanları toplu olarak daha önce de belirtildiği gibi küme genişliğini ifade eder [36]. **Genişlik kanunu** kısaca şöyle tanımlayabiliriz. "A ve B iki küme ise A ve B ancak ve ancak aynı elemanlara sahip olduğunda (A=B) eşittir "[37]. Kümeler elemanları uyarınca oluştuklarından dolayı küme tanımlamanın basit yollarından biri ($\perp$) küme elemanlarının tek tek belirtilmesidir. Örneğin simgesel olarak $A = a_1, a_2$ biçiminde ifade edilir. Genişlik kanununa göre küme elemanlarının tekrar konması ile küme değişikliğe uğramaz ve genellikle küme elemanları sayılırken yineleme söz konusu değildir. Ancak yineleme özellikle yapılıyor ise bu durumda torba (bag) olan söz etmek gerekir. **Torba** ve **küme** ayrımını belirtmek içinse sırasıyla { } ve işaret-

36. Aynı, 121 s.
37. Aynı, 122 s.


Fiyat (TL)
Yazar (yazar adı)
ISBN
Eser
Yer (adı)
Miktar
Yayın (yayınadı)
Basım
Tarih (yıl)
Bölüm (adı)
Yayınevi (adı)
Not
r1 : yazan
r2 : istekte bulunan
r3 : yayınlayan
r4 : yayınlandığı
r5 : yayınlandığı
r6 : sahip olduğu
r7 : sağlanacak adet
r8 : Sahip olduğu
r9 : sahip olduğu
r10 : istenme amacı

leri kullanılır. Ayrımın belirtilmesi özellikle istatiki hesaplamalar açısından önem taşımaktadır [38]

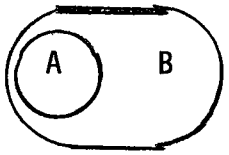
Örneğin {3,6,6,} kümesinde "ortalama 4,5 " iken (3,6,4) gibi bir torbada 'ortalama 5" tir[39]. Küme genişlik kanuna göre küme elemanlarının belli bir sıra izlemesi zorunlu değildir. Ancak elemanlar özellikle sıralı olarak belirtiliyor ise o zaman küme, **sıralı küme** adını alır. Torba, sıralı olduğunda <> işareti kullanılır.

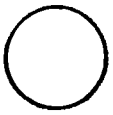
Küme tanımlamanın bir başka yolu da (2) kümenin oluşturulma nedeninin gözönünde bulundurularak biçimlendirilmesidir. Bir başka deyişle kümeye dahil olacak elemanların saptanan koşullar doğrultusunda belirlenmesidir. Örneğin 4 < N > 1 (N:doğal sayılar) gibi belirlenen bi koşul sonucu ortaya çıkan küme. Küme elemanlarının sayısı, o kümenin niceliğini belirler. Örneğin A kümesinin a_1 ,a_2 ve a_3 gibi üç elemanı varsa (A= {a_1 , a_2 ,a_3 }) A kümesinin niceliği 3'tür.

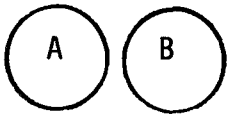
Üst küme = A, B'nin üst kümesidir ancak ve ancak B, A'nın alt kümesi ise ve A, B'nin üst kümesi ise[40]; Örneğin : {1,2,3} {1,3} kümeler arası ilişkinin açıkca ve kolaylıkla görülmesini sağlayan çizimler Isviçreli matematikçi Leonhard Euler tarafından ortaya atılan **Hypothetical Euler** çizimleridir. Bu çizimlere göre küme ilişkileri şöyle özetlenebilir [41].

---

38. Aynı, 122, s.
39. Aynı, 122, s.
40. Aynı, 123, s.
41. Aynı, 124, s.

1. 

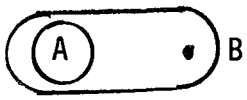
veya
A  B

2. 

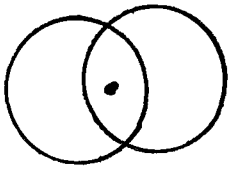
$A \neq B$

3. 

$A \subseteq B$ altküme

4. 

$A \subset B$ alt küme

5. 

A,B örten

Kümeler üzerine yapılan işlemler "=, =, $\subseteq$, C , ve $\supset, \supseteq$" işlemcileri ile gerçekleştirilir.

Alt küme = 1) A ve B gibi iki kümede, eğer A'nın tüm elemanları aynı zamanda B'nin de elemanı ise o zaman A,B'nin alt kümesidir. "A B" diyoruz.

Örneğin: {1,3} $\subseteq$ {1,2,3}

2) A ve B gibi iki kümede A $\subseteq$ B olabilmesi ancak ve ancak A'nın hiçbir elemanının B'de bulunmamasına bağlıdır.

Örneğin: { } veya $\emptyset \subseteq$ {1,2,3}

3) Her küme, aynı zamanda kendisinin alt kümesidir.

Örneğin: {1,3} $\subseteq$ {1,3}

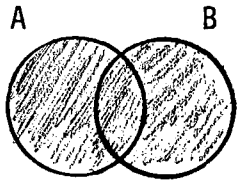
Ast altküme = (proper subset) A,B'nin ast altkümesidir, ancak ve ancak A,B'nin alt kümesi olduğunda ve A,B'ye A,B eşit olmadığında

Örneğin: {1,3} $\subset$ {1,2,3}

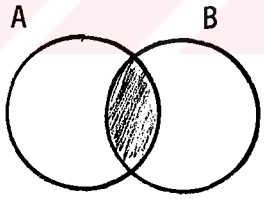
Kümeler arası ilişkilerini göstermek ve küme oluşturmak için birçok işlem ve çizim bulunmaktadır.

Örneğin: Venn diagrams bu amaçla kullanılabilen çizimlerdendir. Anılan çizimler doğrultusunda gerçekleştirilen kimi küme işlemleri şöyledir:

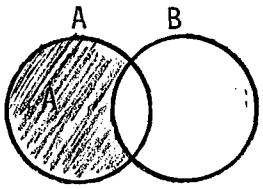
$A \cup B$ = {A ve B kümelerinin eleman toplamını verir}



$A \cap B$ = {A ve B kümelerinin ortak elemanlarını verir}



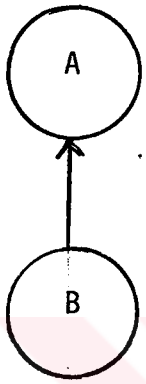
$A - B$ = {B'nin hiçbir elemanının bulunmadığı A kümesi}



Bu çizimler küme sayısının az olduğu durumlar için uygundur.

Ancak konumuz açısından önem taşıyan nokta, küme mantığını temel alarak kavramsal şema çiziminde alttipleri oluşturmaktır.

NIAM'da hiçbir nesne tipi tekrarlanmadığı için hiçbir zaman iki küme birbirine eşit (A=B) olmaz. Ancak A ve B gibi iki küme birbirinin **alt** ya da **üst tipi** olabilir. Her iki küme arasındaki ilişki ise bir ok ile gösterilir.



Bununla beraber "ast" nitelemesi NIAM'da popülasyon karşılaştırmaları için kullanılmaz. Çünkü veritabanının herhangi bir durumu için pop (A) = pop'(B) olabilir [42].

Alttipler aynı zamanda bir grafik özelliği de yansıtır. Bu durumda alttipler grafiğin "düğüm"lerini oluşturur. Söz konusu grafiklerde yön gösteren okların kullanılması durumunda anılan grafikler **"yönlendirilmiş grafik"** adını alır. Hiçbir "tip" birbirinin aynı olamayacağı için tip'e dayalı grafiklerde **döngü** ya da **daire** oluşumu söz konusu değildir.

Dolayısıyla NIAM'a göre oluşturulan şemalarda alttip ve tip düzenlemeleri **"döngü"** ya da **"daire"**nin bulunmadığı grafik özelliklerini taşır.

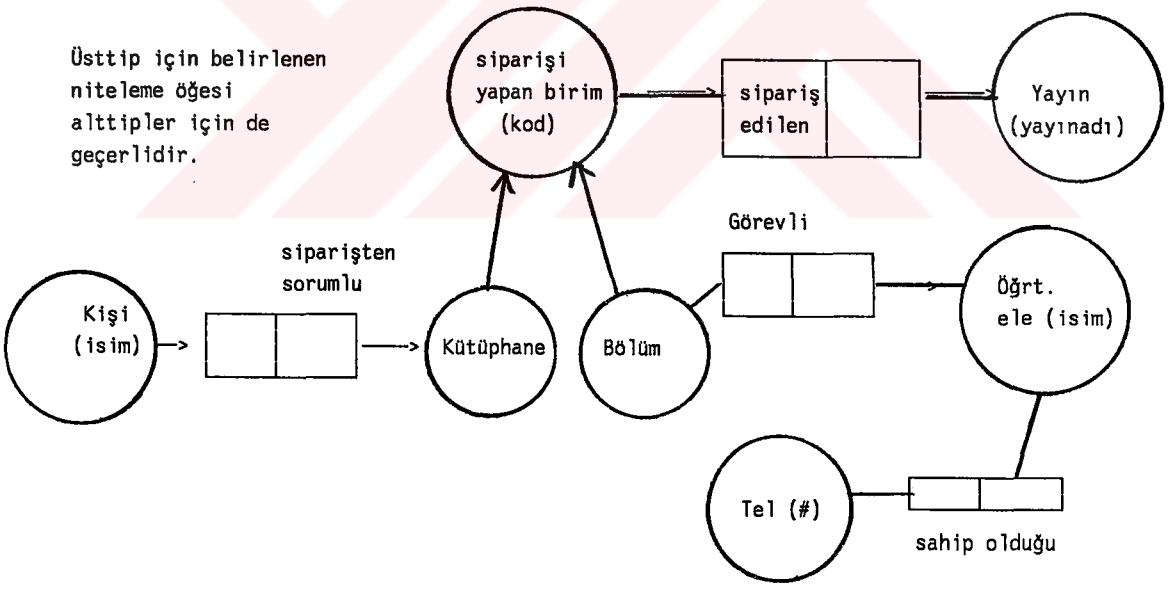
"Şemada, eğer bir varlık tipi başka hiçbir varlık tipinin ast alttipi değilse adı geçen varlık tipine "basit varlık tipi" denir[43].

---

42. Aynı, 126 s.
43. Aynı, 127 s.

Alttip kümelemesi sonucunda ortaya çıkan alttipler yeniden bir araya getirildiğinde yeni bileşimler söz konusu olduğunda orijinal üst tip tekrar elde ediliyor ise üsttip "ayrıntılı" olarak değerlendirilir.(44) Şemada alttip varlığını ortaya çıkarmak için şöyle bir süreç izlenir. "Bütün seçimli roller yalnız varlık tipine bağlı iyi tanımlanmış ast alttip için kaydedilmiş ise o zaman alttip kendisine bağlı rol ile belirlenir, aksi halde olduğu gibi bırakılır "(45).

Burada"iyi tanımlanmış" nitelemesi, sistemde alttipin tamamiyle üsttipe bağlı olan başka roller cinsinden tanımlanmasını ifade etmektedir. Siparişi yapan birimin hangi yayınları sipariş ettiğini bilmek istediğimizde aşağıdaki gibi bir şema çizimi gerçekleştirmek olanaklıdır.



Bu şemaya göre siparişi yapan birimler için bir alttip oluşturulurken yalnız "sipariş yapan birimler" için alttip gerçekleştiril-

44. Aynı, 131 s.
45. Aynı, 132 s.

miştir. Çünkü "birim alttipleri" varlık tiplerine bağlanabilen, en az bir rol bulunmaktadır. Alttip belirleme sürecinde çıktı raporlardan yeterince bilgi edinilemediği durumlarda kullanılan bir başka yöntem **matris analiz** yöntemidir. O nedenle kavramsal şemada birbirinden ayrı birçok alttip grafiği bulunabilir. Her alttip grafiği tam olarak bir "basit varlık tipi"nden oluşmaktadır. Buna **ortak üst tip** veya **grafik başı** denir. Alttip grafiklerinin yalnız bir "baş"ı vardır. Daha önceden de bildiğimiz gibi "basit varlık tipleri" karşılıklı dışlamalıdır. Yani hiç ortak elemanları yoktur. NIAM'da alttipler veri tabanına kaydedilen şeyler üzerine gerçek dünyadaki doğal sınıflamanın bir görüntüsünü vermek yerine sınırlılıkları ifade etmek için kullanılır.(46)

NIAM'da alttip belirlemesi yapmak için alttipe bağlı bir rolün kaydedilmiş olması gerekmektedir. Alttip bölümlemesinin yapılmasının ardından alttip üyelerinin belirlenmesi söz konusu olan alttipin bağlı olduğu üsttip tarafından oynanan en az bir rol dikkate alınarak gerçekleştirilir.(47)

Anılan matris analiz yönteminden biri ve en çok kullanılanı "bölümleme ayrıntı matrisi"dir. Boylamasına ve enlemesine genişleyen dikdörtgen biçimindeki elemanlar dizisi(48) olan matriste alttip oluşturma süreci şöyle tanımlanabilir.

1. A için bir "bölümleme ayrıntı matrisi" oluşturulur. A'nın tüm elemanları için sütun başlıklarını dikkate alarak kaydedilmiş ay-

---

46. Aynı, 127 s.
47. Aynı, 131 s.
48. Tor Gulliksen, Matematikk i Praksis. 2 utgave. Oslo: Universitetsforlaget, 1991, 271 s.

rıntılar listelenir. A **kayıt desenine** göre gruplara bölünür ve bu gruplar sıra başlıkları olarak listelenir. Her kayıt için kayıt deseni girilir. (1=kaydedilen ayrıntı için, 0= kaydedilmeyen ayrıntı için)

2. Alttip matrisi ve ona bağlı ayrıntılar çizilir. Farklı sütun desenleri grafiğin düğümleri gibi A,B vb. biçimde isimlendirilir. Düğümler arasındaki alttip ilişkileri belirlenir. (X,Y'nin alttip düğümüdür ancak ve ancak her sıra için X=1, Y=1 olduğunda) Alttip grafiği çizilir ve geçişli yaylar (transitivarc) ayıklanır. Her düğümde o sütunun başlıklarını belirleyen roller gösterilir.

3. Alttipler için anlamlı cümle ve tanımlar getirilir. Her alttip bağlı bulunduğu üsttipe ait roller cinsinden belirlenir.

4. Kavramsal şema tanımlanır. Şema diagramında ayrıntılar, alttip tanımlamaları listelenerek doldurulur.

Bölümleme ayrıntı matrisi sürecini daha anlaşılır kılmak için şöyle bir örnek kullanılabilir.

Örnek : 1. adım= siparişi yapan birimin hangi siparişleri yaptığını gösteren bir tablo için öncelikle bir bölümleme ayrıntı matrisi oluşturulur ve ayrıntılar sütun başlıklarına göre listelenir.

Birimin (Adı)	Yayının (Adı)	Öğretim Elemanı (Adı)	Kütüphaneci (Adı)	Telefon (#)
B_1	Y_1	-	K_1	-
B_2	Y_2	$Ö_1$	-	T_1
B_3	Y_3	$Ö_2$	-	T_2
B_4	Y_4	$Ö_3$	-	T_3
B_5	Y_5	$Ö_4$	K_2	T_4
B_1	Y_6			

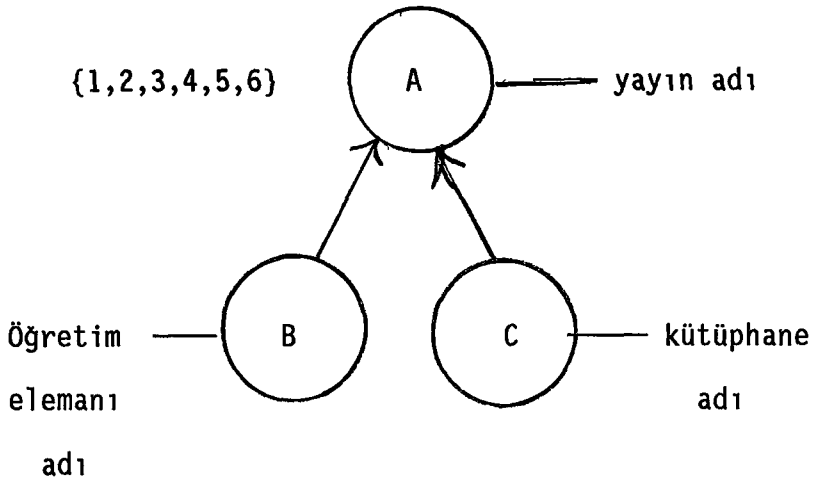
2. adım = Kayıt desenine göre grup bölümlemesi yapılır. Bu amaçla her sütuna kayıt deseni girilir.

	Birimin Adı	Yayının Adı	Öğretim Elemanı Adı	Kütüphaneci Adı	Telefon (#)	
1	1	1	0	1	0	G_1
2	1	1	1	0	1	G_2
3	1	1	1	0	1	G_2
4	1	1	1	0	1	G_2
5	1	1	1	0	1	G_2
6	1	1	0	1	0	G_1

Yukarıdaki tabloya göre 1 ve 6. sıralar aynı kayıt desenine sahip olduğu için aynı grup altında toplanır ve aynı yöntem diğer sıralara da uygulanır.

3. adım = Farklı sütun desenleri grafiğin düğümleri gibi A,B vb. biçimde isimlendirilir ve şekil de görüldüğü gibi düğümler arası ilişkiler belirlenir. Daha sonra düğümlere bağlı roller grafik üzerinde gösterilir.

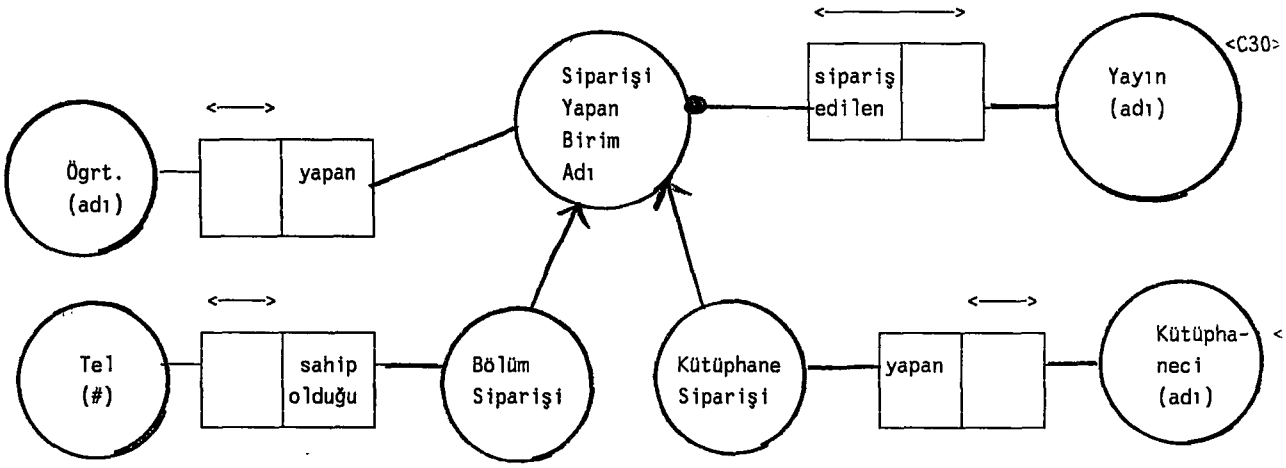
G_1	1	1	0	1	0
G_2	1	1	1	0	1
	A		B	C	B



Bölüm siparişi = öğretim elemanlarının yaptığı siparişler

Kütüphane siparişi= Kütüphanecinin yaptığı siparişler

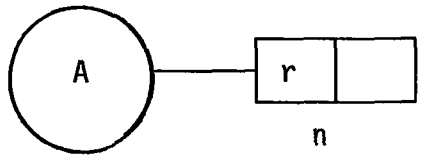
4. adım = Kavramsal şemanın oluşturduğu aşamadır. Buna göre kavramsal şema çizimi şöyle biçimlenir.



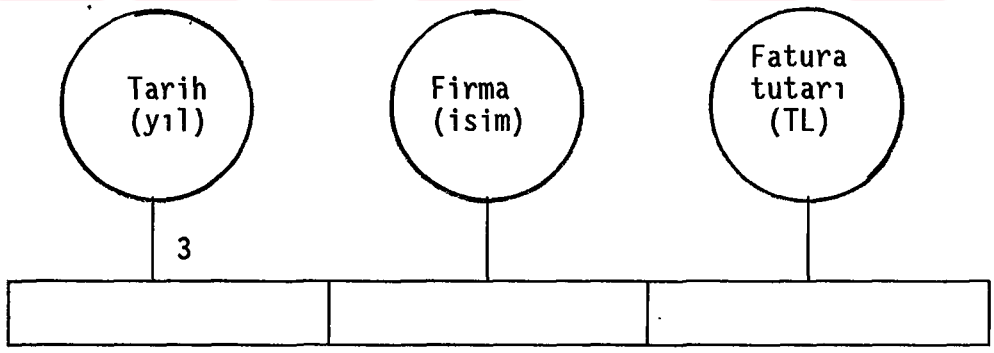
Olgu Sıklığı Sınırlılığı: Bir varlık tipi kendisine bağlı rol uyarınca bir tabloda birçok kez yinelenebilir. Niteleme öğesi değeri-

nin ilgili sütunda kaç kez görüneceğini belirleyen sınırlılıklar bulunmaktadır. Söz konusu sınırlılıklara "olgu sıklığı sınırlılığı" denmektedir.

Örnek:



Yukarıdaki şekle bakıldığında A varlık tipine bağlı r rolünün yeraldığı sütunda değerlerin n kez görülebileceği anlaşılır. n=1 olduğunda bu, teklik sınırlılığına karşılık gelmektedir. O nedenle n=1 durumu şema çizimi üzerinde gösterilmez. Olgu sıklığı sınırlılığı kimi küçük gerçek tiplerinin bütün örnekleri ile ilgili gerçekleri kaydetme gereksiniminin bulunduğu durumlar için söz konusudur. Örneğin, bir kütüphanenin üç firma aracılığı ile yayın siparişinde bulunduğunu varsayalım. Kütüphanenin her yıl söz konusu firmalara yaptığı ödeme tutarını gösteren şema çizimi ve tablo aşağıdaki gibi olacaktır.



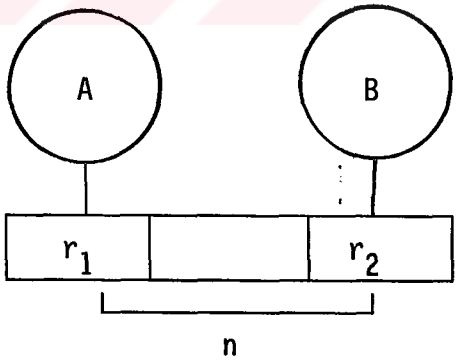
.. yılında, ..	yapılan siparişin,	tutarı..... dır.
1987	EBSCO	150
1987	SWISS	300
1987	IEE	250

Yukarıdaki şema çiziminde tarih ile ilgili sütunda yeralan 3 sayısı anılan sütundaki değerlerin herbir firma için 3 kez yineleneceği anlamını taşımaktadır. Burada varlık tipi sınırlılığı yalnız 3 adet firma olduğunu ve teklik sınırlılığı da her (yıl, firma) bileşiminin tabloda bir kez görüneceğini belirtir. Dolayısıyla adı geçen üç tip sınırlılık her yıl için anılan üç firma ile ilgili fatura tutarının kaydedilmesi gerekliliğini ortaya koyar.

Böyle bir tablonun güncelleşmesinde belki sorun olarak nitelendirilebilecek durum, bir yıla ait değeri girmek gerektiğinde o yıla ait diğer değerlerin de girilmesi zorunluluğunun bulunmasıdır.

Kimi durumlarda olgu sıklığı sınırlılığı saptanırken gerçek tipinin iki ya da daha fazla rolü dikkate alınır. Olgu sıklığı sınırlılığının söz konusu olduğu roller birbirine bir yay aracılığı ile bağlanır ve sınırlılığı belirten sayı yazılır.

Örnek:



Alt ve üst sınıra sahip belli bir aralığın söz konusu olduğu durumlarda aralık n...m biçiminde belirtilir. Ancak aralık kapsamına almak istediğimiz sayılar varsa bunlar aralarına "," koymak suretiyle kapsam dışı bırakılırlar.

Niteleme Öğelerinin Alacakları Değerlere İlişkin Düzenlemeler: Niteleme değeri, gerçek tipinde bir sütuna yapılan "girdi"ye karşılık kullanılmaktadır. Gerçek tipi içinde söz konusu değerin sayısal ya da sözel olduğuna dair ayrım tırnak işareti kullanılarak yapılır. Yani değerin tirnak içinde verilmesi onun sözel bir değer olduğunu gösterir.

Sözel nesneler ile sözel olmayan nesneler arasındaki ayrım kadar önem taşıyan bir diğer konu sözdizgileri uzunluğunun ve sayısal değer aralığı sınırının saptanmasıdır. Bu amaçla söz dizgileri için {cn}, sayısal aralıklar için [1..m] ifade biçimi kullanılır. (Burada n: karakter uzunluğunu, c: değerin karakterlerden meydana gelen sözdizgisi olduğunu belirtmek amacıyla kullanılır).

Sayısal aralıklarla, söz konusu aralığın dışında tutulmak istenen ara sayılar kapsam dışında bırakılarak boşluk "," ile ifade edilir.

Bunun yanı sıra söz dizgisi ya da sayısal biçimde ifade edilen değerler üzerine işlem yapılmak istendiğinde **soyut veri tipi** (abstract data type) gibi yeni bir veri yapısı ortaya konur. Bu veri yapısı temelde iyi tanımlanmış işlemlerle değerler kümesidir [49]. Söz konusu veri yapısal sıralama ya da toplama gibi söz dizgisi veya sayısal değerler ile ilgili işlemler yapmak için kullanılmaktadır.

---

49. Aynı, 160 s.

Kavramsal Şemada 7. Adım: Varlık Tanımlamalarına İlişkin Düzenlemeler

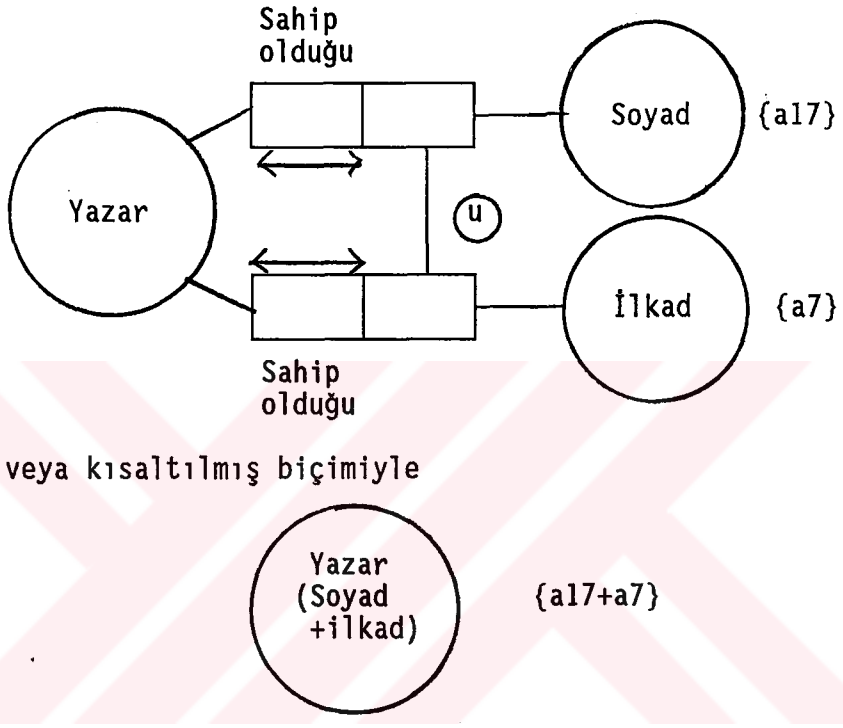
Varlık ile değer arasında birebir ilişki sağlayan niteleme öğelerini belirtmek için daha önce de olduğu gibi parantez kullanılmaktadır. Bunun yanısıra örneğin isim gibi kimi niteleme öğelerinde sözel sınırlama da yapılabilir. Sözel sınırlamada harf dizisinin uzunluğu yanında kesme, tire gibi işaretler de belirtilir.

Gerçek dünyadaki varlıklar bilgi tabanında yalın terimler ile ifade edilmektedir. Ancak niteleme öğelerine karşılık olarak kullanılan söz konusu terimlerin alacakları değerlerin yalnız bir varlığa ait olması esası vardır. Aksi halde değer o gerçek tipi için tanımlayıcı özelliğini kaybeder.

Örneğin "yazaradı" niteleme öğesinde yazaradı yalnız yazarın soyadından oluşuyorsa, aynı soyada sahip yazarlar arasında bir ayrım yapmak olanak dışıdır. Çünkü bir değil birden fazla varlığa karşılık gelebilmektedir. O nedenle yazaradını daha tanımlayıcı kılmak amacıyla söz konusu niteleme öğesi ile diğer niteleme öğeleri veya gerçek tipleri üzerine tanımlayıcılık sağlanana değin çeşitli bileşimler denenir.

Ancak varlığa ilişkin niteleme öğelerinin tanımlayıcı olması sağlanırken gerçek tiplerine ilişkin olarak kullanıcıdan gelebilecek çeşitli istekler de gözönünde bulundurulmalıdır. Örneğin "yazaradı" niteleme öğesi yazarın soyadı ve ilk adı olarak tam ad biçiminde daha spesifik bir hale getirilir. Fakat kullanıcı sisteme "soyadı .... olan

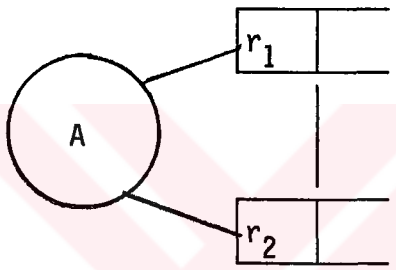
yazarları listele" gibi bir soru yönelttiğinde sistemin böyle bir soruyu yanıtlayabilmesi için ismin tamamı içinden soyadını çekebileceği yönde bir tanımlamanın yapılması gerekir. Bu durumda şema çizimi teklik ilkesi korunmak suretiyle aşağıdaki gibi gerçekleştirilebilir.



Varlık nitelemesinde karşılaşılan bir diğer sorun varlığa ilişkin bir niteleme öğesine (isim) ait birden fazla değerin (lakab, takma ad vb) bulunmasıdır. Böyle durumlarda bir niteleme öğesi standart tanımlayıcı olarak kabul edilir, diğerleri ise sıradan gerçekler biçiminde ele alınır. Aynı yöntem birden fazla 1:1 tanımlamanın bulunduğu gerçek tiplerine uygulanır.

Eşitlik, Dışlama ve Alttip Sınırlılıkları : Eşitlik sınırlılığı, bir çıktı tablosundaki bir değerin kaydedilmesinin bir başka değe-

rin girilmesine bağımlı olduğu durumlar için geçerlidir. Şema çiziminde söz konusu durumun belirtilmesi için çift yönlü ok kullanılır. Mantıksal işlemlerde "ancak ve ancak" anlamını taşıyan "çift yönlü ok"un kullanımı aynı anlamın şema çizimine de aktarılması amacını taşımaktadır. Eşitlik sınırlılığı ilke olarak şöyle formüle edilebilir: r_1 ve r_2 A gibi bir varlık tipi tarafından oynanan iki ayrı roldür. r_1 ve r_2 genelde farklı gerçek tiplerinde görünürse de amaç,söz konusu rollerin aynı gerçek tipi içinde yeralması olanağını yaratmaktadır.



Veri tabanının tüm durumları için pop (r_1) = pop (r_2) Her a için r_1 <=====> r_2 kaydedildiğinde kaydedilir.

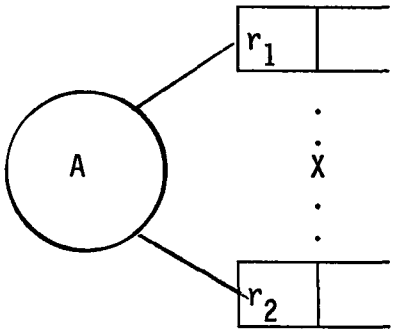
Birleşik gerçek tipleri cinsinden eşitlik sınırlılığı r_1 sütununda yeralan değer kümesinin r_2 sütunundaki değer kümesi ile eşit olduğu anlamına gelir. (Küme içinde dublikelerin bulunması önemli değildir [50] Rollerin zorunluluk sınırlılığına sahip olması durumunda eşitlik kendiliğinden sağlanacağı için (<———>) çift yönlü ok şema diagramında gösterilmez.

Dışlama sınırlılığı; bir tabloya bir sütuna ait bir değerin girilmesi halinde diğer sütuna ait bir başka değerin girilmemesi gibi bir koşul ortaya konduğunda anılan koşul şema diagramında rolleri birbirine bağlayan noktalı bir çizgi ve "X" işareti yardımıyla belirtilir.

---

50. Aynı, 173 s.

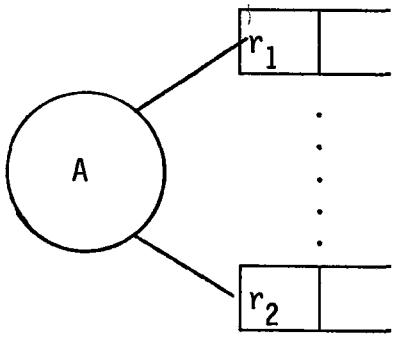
Örnek



Veritabanının tüm durumları için pop (r_1) n pop (r_2) = { }.Hiçbir a hem r_1 hem de r_2 sütunu için birlikte kaydedilmez (51).

Altküme Sınırlılığı : İki role ait kümelerin karşılaştırılmasına olanak veren altküme sınırlılığı, bir türetme kuralından çok bir veritabanı sınırlılığıdır.

Altküme sınırlılığı güçlü bir alttipleme yapabilliyor ise şema çiziminde gösterilmez ve alttipleme gerçekleştirilir. Böyle durumlarda üsttipin oynadığı role göre alttip = üsttip koşulu sağlandığında şema çizimi üzerinde altküme sınırlılığı belirtilir. Altküme sınırlılığı kısaca aşağıdaki gibi formüle edilebilir.



Veritabanının tüm durumları için

pop (r_2) $\subseteq$ pop (r_1)

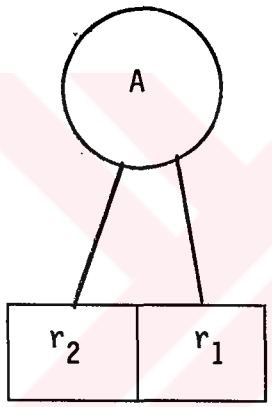
Her a için r_2 kaydedilmişse r_1 de kaydedilmiştir (61).

51. Aynı, 177 s.

Veritabanında her iki rolün de aynı varlık tipi tarafından oynandığı durumlar söz konusu olabilmektedir. Homojen ikililer olarak tanımlanan böylesi durumlarda gerçek tablosunun sıra kümeleri dikkate alınır.

Bir başka deyişle sıra kümesinde küme elemanları arasında ilişkiler tiplerine göre tanımlanır. Anılan ilişki tipleri **yansımalı bağıntı, bakışımlı bağlantı** ve **geçişli bağıntı** olmak üzere üç tiptir.

Yansımalı Bağıntı : Bir ilişki tipinin bir varlık tipi üzerinde yansımalı olmasıdır.



R,A üzerinde yansımalıdır, ancak ve ancak A'daki tüm x'ler için xRx ise [52].

Bakışımlı Bağıntı: Bir E kümesi üzerinde her x,y E için (x,y) ve B 'dir. Ancak ve ancak (y,x) B koşulu gerçekleyen B ikili bağıntısıdır [53].

Geçişli Bağıntı : Bir küme üzerinde (x,y) B, (y,z) B'dir; anca ve ancak (x,z) B ise [54].

---

52. Aynı, 184 s.
53. Doğan Çoker-Timur Karaçay, Matematik Terimleri Sözlüğü. Ankara: Türk Dil Kurumu, 1983, 24 s.
54. Aynı, 140 s.

Bir ilişkinin her üç özelliğe de sahip olması halinde ilişki tipi "RST" veya "eşitlik ilişkisi" olarak adlandırılır.

Yukarıda anılan ilişki tipleri ilişkiyi belirledikleri için sınırlılık olarak nitelendirilebilir. Ancak **gerçekler** bir başka gerçeği saptama olanağı yaratıyor ise bu özelliğinden dolaylı anılan ilişki tiplerini bir sınırlılıktan öte türetme kuralı biçiminde değerlendirmek daha doğrudur (55).

Sistemin anılan ilişki tiplerini tanıyabilmesi açısından "reflex (R) "sym (R)" ve "trans (R)" simgeleri kullanılır (56).

Diğer Sınırlılıklar

Kavramsal şema diagramı üzerinde şimdiye kadar birçok sınırlılık gösterildi. Ancak şema üzerinde ifade edilemeyen ya da şemayı daha karmaşık hale getirebilecek daha başka sınırlılıkların bulunması olasıdır. Bu durumda şemanın insanlar tarafından rahatlıkla anlaşılabilmesi ilkesini gözönünde bulundurarak bu tür sınırlılıkların ifadesinde başka yollar izlenir. Örneğin şemayı okuyanın kolay anlayabileceği, küme kuramı ve mantık işlemlerine dayalı notasyonların kullanılması ya da doğal dilde şema diagramı altında ifade edilmesi. Bir başka yol yine anlaşılırlık ilkesi gözönünde bulundurularak kavramsal yapıyı tamamlayan sınırlılıkların formüle edilmesidir.

---

55. Nijssen, a.g.e., 184 s.
56. Aynı, 186 s.

Şema diagramının bütünüyle tamamlanıp tamamlanmadığını anlamak için son kez popülasyon ve yineleme denetimi yapılır. Bu amaçla şema diagramının orjinal örneklerle tutarlılığını anlamak için şema tabloları örneklerle doldurularak popülasyon denetimi yapılır. Bu yolla yinelemenin varlığı ve popülasyonun önceden belirlenen sınırlılıkları zedeleyip zedelemediği irdelenir.

Yineleme çok çeşitli biçimlerde ortaya çıkmaktadır. Örneğin üçlü bir gerçek tipinin iki varlık tipine karşılık gelen ikili bir gerçek tipi bir başka yerde depolanmış ise buna depolama sınırlılığı denir ve çözüm için şemaya eşitlik, altküme ve dışlama sınırlılıklarının getirilmesi gerekmektedir. Bunun dışında türetilen gerçek tiplerinin de ayrı birer gerçek tipi olarak ifade edilmesi durumu ortaya çıkabilir. O zaman tüketilen gerçek tipini soru sorudukça hesaplanacağı ya da ayrı bir gerçek tipi olarak depolanacağı yolunda bir kural belirleme gereği vardır. Buna denetimli yineleme denir. Ayrıca yinelemenin kaçınılmaz olduğu durumlar da bulunmaktadır. Ancak bu tür yinelemelerin güncellemede herhangi bir sorun yaratmayacaklarından dolayı şemada bulunmalarında bir sakınca yoktur. Çünkü yineleme bilgi erişim açısından bazen oldukça yararlıdır. O nedenle belli oranda yineleme iyi sonuçlar verebilir.

Kavramsal Şemanın İlişkisel Şemaya Aktarımı

Günümüzde kavramsal şemanın sisteme doğrudan aktarılmasına olanak veren pek çok yüksek düzeyli dil bulunmaktadır. Örneğin Reference and IDea Language = RIDL, yalın gerçekleri, sınırlılıkları ve

alttipleri şema cümlelerini kullanarak doğrudan tanımlayabilmektedir. Ancak bu tür dillere erişim son derece sınırlıdır. O nedenle ilişkisel veritabanı sistemlerinde yaygın biçimde kullanılan dördüncü nesil dillerin örneğin SQL'in kullanımı daha uygundur. Çünkü SQL her çaplı bilgisayarda kullanılabilen standart bir dildir. Tüm işletim sistemleri ile uyumludur. Olmadığı durumlarda firmalar SQL arabirimini sisteme dahil edebilmektedir.

Kavramsal şemanın ilişkisel şemaya aktarımı aslında söyleşi evreninin ilişkisel veritabanı sistemlerindeki uygun yapıya dönüştürülmesidir.

Kavramsal şemadaki yalın gerçekler ilişkisel tabloların sıralarını oluşturur. Ne var ki ilişkisel tablonun bir sırasında birden fazla gerçeği sunmak olanaklıdır. Bunun yanı sıra ilişkisel tablolardan yöneltilen sorular uyarınca çeşitli bileşimler meydana getirilerek bilgi kolaylıkla elde edilebilir. Aynı zamanda bilgi istenen biçimlerde çıktı rapor olarak da görüntülenebilir.

İlişkisel tabloya aktarımda zorunlu ve seçimli rollerin ifadesinde sütunların boş değer alıp almadıkları gözönünde bulundurulur ve ilgili sütunun yanına niteleme kodu konur.

Örneğin, Tablo = [1.Z sütun MA (veya) Notnull, 2.sütun OP] Burada MA: zorunlu, OP:seçimli rolleri, Notnull de boş olmayan değerleri ifade etmek için kullanılmıştır.

Kavramsal şemanın ilişkisel şemaya aktarımında iki ana aşama bulunmaktadır. Tablo tanımlarının oluşturulması ve sınırlılıklar ile türetme kurallarının yürütülmesi için yöntemlerin yazılması.

Kavramsal şemadaki gerçek tiplerinin tablo tipleri biçiminde gruplanarak ilişkisel şema biçimine dönüştürülmesine optimum normal biçim : ONB algoritması denmektedir. Algoritmanın amacı basit, güvenli ve verimli bir tasarım gerçekleştirmektir. ONB'nin bir tablo tasarımı için ortaya koyduğu **ölçütler** şöyledir:

1. Tekrarlanmayan özellikler
2. Yineleme ve güncelleme problemlerinin olmaması
3. En az tablo sayısı

ONB algoritması, geleneksel normalizasyon kuramında 5NB'e karşılık gelen tabloları ilk aşamada üretebilmektedir. Bu algoritmaya göre her **gerçek tipi** bir **tablo'ya** dönüştürülmektedir. Rollerin sütun başlıklarının yerini aldığı tablonun her sırası böylelikle bir gerçek tipini yansıtır. Gerçek tipindeki anahtarlar teklik sınırlılıkları uyarınca tablonun anahtarlarını oluşturur ve ilişkisel tablo için bir ana anahtar belirlenir.

Yalın ana anahtara sahip tablolarda tablo, anahtarın belirlediği varlık tipinin özelliklerini, ana anahtarın bileşik bir yapıya sahip olduğu durumlarda ise tablo varlıklar arasındaki ilişkileri yansıtır.

Nijssen'in kavramsal şemadan ilişkisel şemaya aktarımı kolaylaştırmak ve şemanın niteliğini artırmak için ortaya attığı ONB algoritması aşamaları ile şöyle özetlenebilir [57].

"1. Tek anahtarı olmayan her gerçek tipi için ayrı bir tablo yaratılması ve gerçek tipinin en kısa anahtarının o tablonun ana anahtarı olarak seçilmesi;

2. Ortak nesne tipine bağlı ana anahtarlı gerçek tiplerinin aynı tabloda gruplanması;

3. Geride kalan her gerçek tipi için ayrı bir tablo oluşturarak gerçek tipi anahtarının tablonun ana anahtarı olarak belirlenmesi."

Tabloların gruplanmasının ardından tablo ve sütunları için anlamlı isimler bulunur, anahtarların altı çizilir ve seçmeli roller işaretlenir.

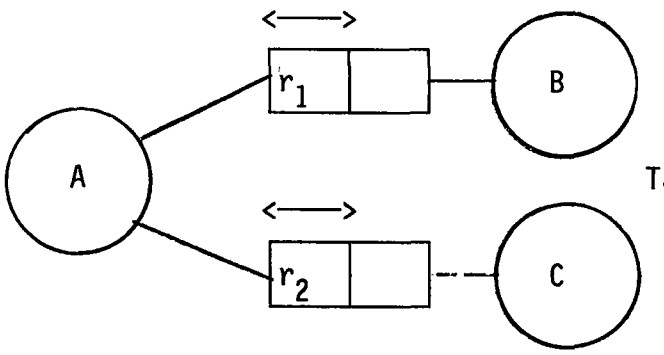
ONB'in birinci adımında tek bir anahtarı olmayan gerçek tipleri ayrı bir tabloya alınmaktadır. Yani çokludan-çokluya ikili bir gerçek tipi veya derecesi > 2 olduğunda gerçek tipleri ayrı bir tabloda gruplanır. Bir başka biçimde ifade etmek gerekirse en kısa anahtarının bileşik bir yapıya sahip olduğu gerçek tiplerinin başka bir tabloda gruplandırılmasıdır.

---

57. Nijssen, a.g.e., 255 s.

ONB'nin ikinci adımında bir varlık tipine ait birden fazla anahtar bulunuyor ise bunlar tek bir tabloda toplanır ve ortak varlık tipi kullanılarak tablonun ana anahtarı belirlenir.

Örnek :



Tablo = [A, B, C]

ONB'de üçüncü adım son derece yalın ve basittir. Geriye kalan tek anahtarlı gerçek tiplerinin tablolar halinde gruplanmasından ibarettir.

Nijssen ONB algoritmasını kavramsal şemanın ilişkisel şemaya aktarımını kolaylaştıracak ve verimliliğini artıracak bir yöntem olarak ortaya atmıştır. Gerçekten de uygulamaya geçildiğinde bu yöntemin pratikliği ve yararları görülmektedir.

Şema oluşturmada birinci adım daha önce de değinildiği gibi soru-yanıt sürecinin yalın gerçekler cinsinden ifade edilmesidir. Bu amaçla ilimizdeki birçok üniversite kütüphanesinin sipariş formları incelenmiş ve aşağıda örnekleri verilen ortak bir biçim belirlenmeye çalışılmıştır.

Sipariş Formu

Sipariş no :	ISBN :
Yayınadı :	Fiyat:
Yazaradı :	Adet :
Yayınlayan :	
Yayın yeri :	Sipariş edilen :
Yayın tarihi:	Firma kodu :
Basım :	Sipariş tarihi :
Cilt :	sipariş eden
	kodu :

Sipariş Eden Formu

Sipariş eden	Tel:
kodu :	
Adı soyadı :	
Ünvanı :	
bölümü :	

Sipariş Edilen Firma Formu

Firma No :			
Firma adı :			
Adresi :	1	2	3
	Tel:	Tel:	Tel:
İndirim oranı		Notlar:	
Teslim süresi			

Fatura Formu

Fatura No :	
Firma No :	
Geliş tarihi :	Kayıt tarihi:
Tutar :	
Ödendiği Fon :	

Görüldüğü gibi burada **sipariş eden, yayın,** firma ve fatura olmak üzere dört varlık tipi belirlenmiş ve bunlara ait olup bilgi uzmanının gereksindiği tüm veriler formlarda yeralmıştır. Söz konusu formlardan yararlanılarak çıkarılan yalın gerçekler ise söyledir.

1. Yazarın (yazaradı) (yayınadı) adlı Yayınının Basımı,(sayı), Cilt (sayı) olarak Yayınevi (yayınevi adı) tarafından yılında Yerde (şehiradı) yayınlanmıştır.

   Yazar(yazaradı) Yayın(yayınadı) Basım(sayı) Cilt(sayı) Yayınevi (yayıneviadı) Yıl Yer (yeradı)

2. Yayının (yayınadı) ISBN'i (sayı) dır.
   Yayın (yayınadı) ISBN (#)

3. Sipariş numaralı (no) Yayın (yayınadı) Fiyatı (TL) olup Firma (kod) ya sipariş edilmiştir.
   Sipno (no) Yayın (yayınadı) Fiyat Firma (kod)

4. Sipariş numarası (no) olan
   Yayın (yayınadı) Kullanıcı (kod) tarafından sipariş edilmiştir.
   Yayın (yayınadı) Sipno (no) kullanıcı (kod)

5. Siparişi veren kullancının (kod) İsim, Ünvanı, Çalıştığı Bölüm (kod) Telefon (#) dur.
   Kullanıcı (kod) İsim, Ünvan Bölüm Tel.

6. Sipariş verilen Firma (kod)nın İsmi (Firmaadı) bulunduğu şehir (İsim) deki Adresi telefonu (#)
   Firma (kod) nın tanıdığı indirim oranı (%), verdiği teslim süresi (gün)

7. Firma (Kod)nın tanıdığı indirim oranı (%), verdiği teslim süresi (gün)
   Firma (kod) İnd.or (%) tes.sür (gün)

8. Firma (kod) dan gelen Fatura (no) olup geliş tarihi (gün, ay, yıl) kayıt tarihi (gün, ay, yıl) dır.
   Firma (kod) Fatura (no) gel.tar.(gün, ay, yıl) Kay.Tar. (gün, ay, yıl)

9. Fatura (no)'nın ödendiği Fon (kod)
   Fatura (no) Fon (kod)

10. Fon(kod) da bulunan miktar (TL), fondan ödenen miktar (TL) fonda kalan miktar (TL) Fon (kod) bul.mik(TL) öd.mik (TL) Kal.mik. (TL)

Bunun yanı sıra ikili gerçek tipi cinsinden yalın gerçekleri aşağıdaki gibi ifade etmek olanaklıdır.

1. Yayın (yayınadı) **yazan** Yazar (yazaradı)
2. Yayın (yayınadı) **yayınlayan** Yayınevi (ismi)

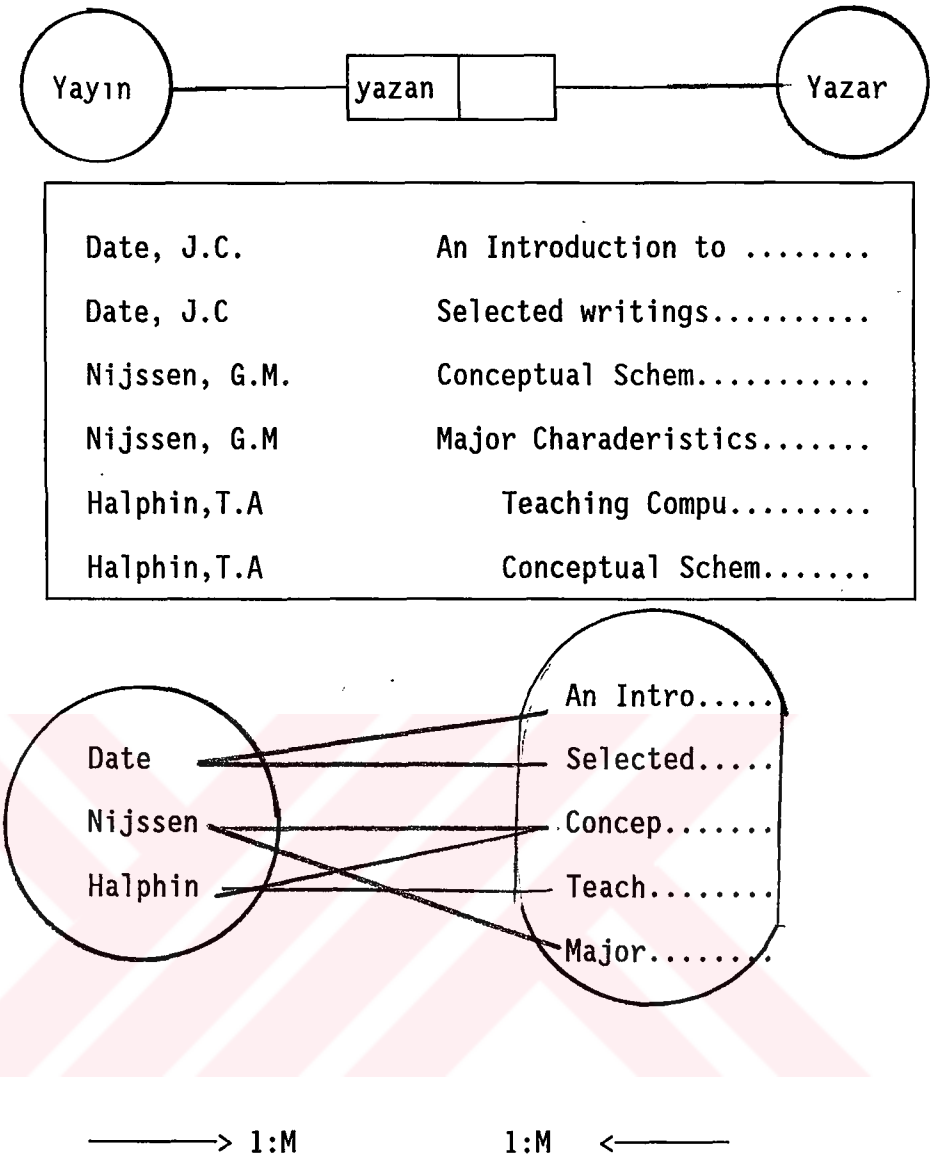
3. Yayın (yayınadı) **yayınlandığı** Şehir (isim)
4. Yayın (yayınadı) **yayınladığı** Tarih (yıl)
5. Yayın (yayınadı) **sahip olduğu** Basım (sayı)
6. Yayın (yayınadı) **sahip olduğu** ISBN
7. Yayın (yayınadı) **sahip olduğu** Cilt (sayı)
8. Yayın (yayınadı) **sahip olduğu** Sipariş no (no)
9. Sipariş no (no) **sahip yayının** Fiyatı (TL)
10. Sipariş no (no)'lu yayının **sipariş edildiği** Firma (kod)
11. Sipariş no (no)'lu yayını **sipariş eden** Kişi (isim)
12. Firma (kod) nın **sahip olduğu** Firma adı
13. Firma (kod) nın **bulunduğu** Adres
14. Firma (kod) nın **bulunduğu** şehir (isim)
15. Firma (kod) nın **gönderdiği** Fatura (no)
16. Firma (kod) **sahip olduğu** telefon (no)
17. Fatura (no) nun sahip olduğu Tutar (TL)
18. Fatura (no) nun **gönderildiği** Tarih (gün,ay,yıl)
19. Fatura (no) nun **kaydedildiği** Tarih (gün,ay,yıl)
20. Kişi(isim) nin **sahip olduğu** ünvan ( )
21. Kişi (isim) nin **görev yaptığı** Bölüm (isim)
22. Kişi (isim) **sahip olduğu** Telefon (no)
23. Firma (kod) **verdiği** Teslim Süresi (gün)
24. Firma (kod) **tanıdığı** İndirim Oranı (%)
25. Yayın (sip no) Firma (kod) **sipariş edildiği** Tarih
26. Yayın (sip no) ile Firma (kod)ye **sipariş edildiği** adet (sayı)

Burada listelenen yalın gerçekler kullanıcı ile makina arasındaki diyaloğu yansıtmaktadır. Ancak dikkat edilecek olursa kullanılan

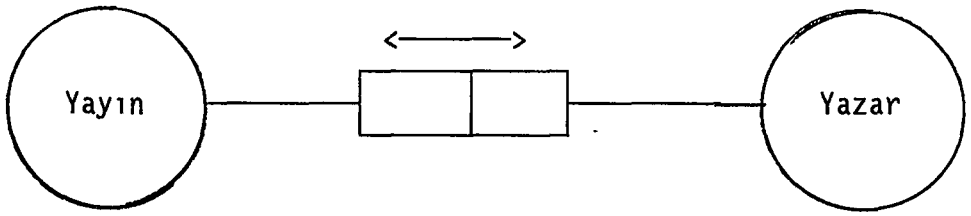
sözdizimi Türkçenin sözdizim kurallarına bütünüyle uymamaktadır. Bu durum ingilizce ile türkçenin dil yapısıdaki farklılıktan kaynaklanmaktadır. Nevarki şema diagramında varlık-rol ilişkisini çizim kuralları çerçevesinde gösterebilmenin de başka bir yolu bulunamamıştır. O nedenle Türkçeye en yakın şema diagramına da en uygun ifade biçiminin yukarıdaki yalın gerçeklerde verilen biçim olduğu düşünülmektedir. Şema diyagramını yalın gerçeklerden yola çıkarak somut hale dönüştürme, tekniğin ikinci aşamasıdır. Bu bağlamda oluşturulan diagram Ek:1 de verilmiştir.

Üçüncü aşama gereksiz varlık tiplerinin ve rollerin elenerek türetilmiş gerçek tiplerinin belirlenmesidir. Şema diagramında türetilmiş gerçek tipi, **fon** ve **miktar** arasındaki **fondan ödenen** ve **fona gelen** ilişkilerinden türetilen **fonda kalan** ilişkisidir. Böylelikle FK= FG - FÖ formülü ile diagramda ilgili rol kutusu altına bir yıldız işareti koymak suretiyle türetilmiş gerçek belirtilmiştir.

Bilindiği gibi dördüncü aşamada teklik sınırlılıkları saptanır. Bu sınırlılıklar yine şema diağramında rol kutucuklarının üzerine çift yönlü oklar konulmak suretiyle gösterilmiştir. Söz konusu oklar ve teklik sınırlılığında küme kavramına dayalı özellikler, daha önce ayrıntılarıyla anlatılmıştır. Ancak burada teklik sınırlılığının saptanmasında izlenen yol birkaç örnek ile desteklenecektir.



Görüldüğü gibi bir yazar birden fazla yayına sahip olabilmekte veya bir yayın birden fazla yazar tarafından yazılabilmektedir. Bu durumda her iki varlık tipi arasındaki ilişki çokludan çokluya ilişki tipindedir. Söz konusu ilişki <———> iki yönlü bir ok ile her iki rolü de kapsayacak biçimde gösterilir.



Teklik sınırlılığı aynı zamanda ifade ettiği ilişki tipi için ana anahtarın saptanmasına da yardımcı olur. Bu durumda yazarın belli bir eserine erişmek, ancak her iki sütün değeri birlikte alındığında olanaklıdır.

Daha sonraki aşamalarda sırasıyla zorunlu ve seçimli roller saptanır alttipler belirlenir, dışlama ve eşitlik sınırlılıkları şema diagramı üzerinde gösterilir. Söz konusu sınırlılıklar diagram üzerinde varolduğu ve uygun olduğu ölçüde Ek:2'de gösterilmiştir. Son olarak ise popülasyon ve yineleme denetimi yapılır.

Şemanın tamamlanmasının ardından yapılması gereken işlem kavramsal şemanın ilişkisel şemaya dönüştürülmesidir. Bu amaçla Nijssen'in önerdiği optimum normal biçim algoritması kullanılmıştır.

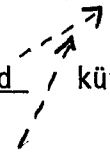
Eser = [Yay.adı Yaz.adı , Yer, Yıl, Yayınevi, Cilt OP, Basım OP,
Adet, Fiyat, Sipno]

Sipariş Arama = [Sipno, ISBN OP, yay.adı yaz.adı, Siptar OP]

Sipariş = [Sip.no, Firma kod, Sip,tar]

Siparişi Yapan

Birim =[Kod, Sip no]



Kütüphane =[kod / kütüphane]

Bölüm = [kod, kişi, ünvan, tel]

Sipariş Edilen

Firma =[kod, firma adı, fat.no, ind OP., Tes sür OP]

Firma Şube =[kod yer, adres, tel]

Gelen Fatura =[no, gel,tar., kay.tar. öd.fon]

Fon =[kod, gel.para, öd.para. kal.para]

Ödeme = [Fon kod, firmaadı Fat.no öd.para]

Fatura Arama [fat.no, firma kod]

Enformasyon sistemlerinin tasarımı konusunun sistem programcılığından ayrı bir konu ve söz konusu sistemlerde verinin taşıdığı anlamın temel çıkış noktası olduğunun anlaşılması ancak 1980'li yıllarda gerçekleşmiştir. 1960'lı ve 70'li yıllar boyunca çalışmalarının odak noktasını donanım ilkeleri, çeviri dili,işletim sistemleri ve derleyici oluşturma teknikleri meydana getirmekteydi. Bütün bu çalışma ve araştırmaları da matematik ve mühendislik alanlarındaki uzmanlar yürütüyordu. Dolayısıyla enformasyon sistemlerinde etkin rol oynayan bu alanın uzmanları ikinci plana itilmişlerdi. Dahası 1970'li yıllarda geliştirilen yapılandırılmış programlama, enformasyon sistemleri

için'de eşdeğer bir olgu olarak kabul ediliyordu. Yapılandırılmış programlama tekniklerinin enformasyon sistem tasarımına uyarlanmasındaki yetersizliğe bu yaklaşımın yanlış olduğuna dair "görüş" ilk kez Uluslararası Bilgi İşleme Federasyonu (International Information Processing Federation) un Enformasyon Sistemleri Yöntemleri anlayış için genel bir çerçeve (Information Systems Methodologies: a framework for understanding) adlı 1988'de yaptığı bir çalışmada ortaya atıldı. "Görüş"ün vurguladığı temel nokta bilgisayar alanında yapılan çalışmaların **veri** ya da **bilgi** uyarlı değil **algoritma** ya da **süreç** uyarlı olduğuydu. Ancak bu **görüş** yapılandırılmış programcılığın kazandığı başarının büyüsüne kapılan bilgisayarcılar arasında ilgi görmedi. Bununla beraber daha sonraki yıllarda kullanıcıların veriyi işleyen ve yöneten süreçlerden çok günlük yaşamlarındaki işlerinde verinin örneklerine aşina oldukları anlaşıldı ve tasarımın başlangıç noktasının veri örnekleri olması gerekliliği ortaya çıktı. Söz konusu yaklaşım aynı zamanda kullanıcı açısından verinin doğal dil yapısına uygun olarak ele alınması gerekliliğini ortaya koydu.

Günümüzde yaygın biçimde kullanılan birçok enformasyon sistemi tasarım yöntemi bulunmaktadır. Bunlardan başlıcaları şöyle sıralanabilir:

1. Veri Akış Diagramları (Data Flow Diagrams: DFD): Enformasyon sistemlerinde süreç, (process)leri belirlemede kullanılan ve geniş bir kullanım alanı bulunan grafiğe dayalı bir dildir. Süreçlerin hiyerarşik olarak daha küçük alt süreçlere ayrışması temel yöntemdir. Ancak kimi sakıncaları bulunmaktadır. Örneğin depolanan veri yalnız adı ile tanım-

landığından içeriğini belirleyen bir ipucu yoktur. Bu durum yanlış anlamaya yol açabilir. Oysa depolanan her verinin içerdiği bilgi yapısını formüle ederek tanımlamak yararlıdır. Bir diğer sorun, süreçlerin ayrışmasında izlenecek yolu gösteren ilkelerden yoksundur (58).

2. **ISAC,** İsveç'te geliştirilen süreç uyarlı bir yöntemdir. Temelde hiyerarşik ayrışmaya dayanıyorsa da **etkinlik analizi** olarak adlandırılmaktadır. Veri işlenişinde yetersiz kalmaktadır (59).

3. **Normalizasyon,** ilişkisel veritabanı tasarımında kullanılan en popüler yöntemdir. Başlangıç noktası olarak **evrensel ilişkiyi** alır. **Evrensel ilişki** bir uygulamadaki bütün gerçeklerin doğal birleşimini içeren ilişkidir. Kullanıcının bu sürece fonksiyonel bağımlılık, çok değerli bağımlılık ve birleşim bağımlılığı gibi üç tip sınırlılık getirmesi istenir. Anılan sınırlılıklar evrensel ilişkinin ayrışma yolu ile arındırılmasında kullanılır. Ancak bu yöndeki deneyimler evrensel ilişkinin başlangıç noktası olarak ele alınmasının pratik ilişkisel veritabanı sistemleri için kapsamlı oluşundan ötürü iyi sonuç vermemiştir. Dahası veri toplamadaki güçlük ve kullanıcıların söz konusu sınırlılıkları sağlayamamasından dolayı tam bir normalizasyon gerçekleştirmek oldukça güçtür.

---

58. Nijssens, a.g.e., 312 s.
59. Aynı, 313 s.

Normalizasyon tablo tasarımında kullanılan bir yöntemden çok tasarlanmış tablonun niteliğini ölçmede uygun bir yöntemdir[60].

4. **ER**, 1976'da Chen tarafından ortaya atılan **varlık** ve **ilişki**'yi temel alan popüler bir veritabanı tasarım yöntemidir. Bu yöntemde tasarımcılar bir gerçeğin ne zaman bir varlığa ait bir özellik ve ne zaman bir ilişki olarak ele alınacağı sorunu ile karşı karşıyadır. ER çizimleri popülasyona dayalı doğrulama tekniklerinde kullanılmak üzere tasarlanmıştır. Ancak NIAM'a oranla sınırlılıkların ifadesinde yetersiz kalmaktadır [61].

NIAM tasarım yöntemi yukarıda anılan diğer yöntemlere oranla daha basit ve verimlidir. Çünkü kullanıcının yabancı olmadığı örnekleri doğal dilde ifade edebilir; tasarım süresince bir sonraki adımda ne yapılması gerektiğini gösteren ipuçlarını içerir; niteleme denetimi ya da doğrulamanın sürecin sonunda değil tasarım sürecinde yapılmasını sağlar ve kullanılan diagram tekniği ile doğal dildeki örnekleri şema üzerinden kolayca okuma olanağı verir [62].

---

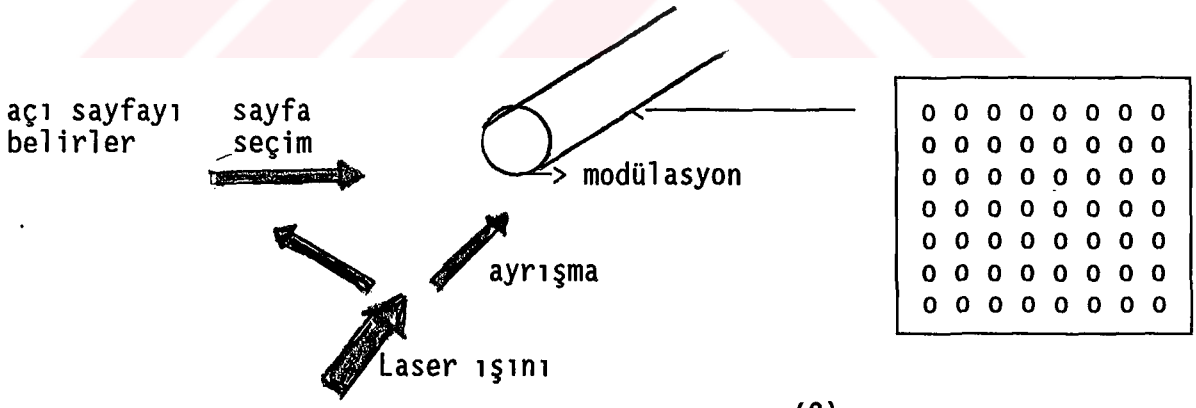
60. Aynı, 313 s.
61. Aynı, 314 s.
62. Aynı, 315 s.

V. BÖLÜM
DEPOLAMA VE ERİŞİM

5.1 DEPOLAMA

Depolama teknolojisinin gelişim düzeyini belirleyen üç temel ölçüt bulunmaktadır. Depolama alanı, erişim hızı ve fiyat. Önceleri, özellikle depolama kapasitesi 100MB'ın üzerinde olan diskler söz konusu olduğunda depolama maliyeti oldukça yüksekti. Ancak bu alandaki pazarın büyümesi ve üretim teknolojisinin gelişmesi depolama alanı açısından maliyetin düşmesine neden olmuştur. Disk'te gözlenen fiyat düşüşü CD ROM, manyeto-optik depolama ve holografik depolama gibi alertentaif tenolojilerde teknik sınırlılıklar nedeniyle beklenen düzeyde gerçekleşmemiştir. Söz konusu teknoloji ürünlerinden CD ROM da veri, laser ışını aracılığı ile ince bir metal üzerinde delikler açılmak suretiyle kaydedilir. Yaklaşık 660MB veri depolayabilen CD ROM'un ana plağının depoloma maliyeti kopyalama maliyetinin çok üzerindedir. CD ROM'larda uluslararası düzeyde standart sağlanabilmiştir. CD ROM'lara alternatif WORM'lar üretilmiştir. 30 yıl kullanım garantisi bulunan WORM'ların CD ROM'a göre avantajı üzerine yazılabilir olmasıdır. WORM disklerde de okuma ve yazma için lazer ışını kullanılmaktadır. Bir başka teknoloji veri üzerine takrar yazılabilen manyeto-optik disklerdir. Manyetik ve optik disk ilkelerinin birleştirildiği bir üründür. Depolama alanı diskte olduğu gibi manyetize edilebilmektedir. Veri, belli bir sıcaklıkta örneğin 150°C de plak üzerine gönderilen laser ışını aracılığı ile kaydedilmektedir. Holografik depolamada boyutları

5x5x2 ya da 1x1x1 olan kristal küp kullanılmaktadır. Çok fazla sayıda parçalara bölünen kristalin her parçası da bir sayfaya karşılık gelen alanlara bölünmektedir. Her sayfaya 32 KB'lik veri depolanabildiği ve parçaların 500 sayfaya bölünebildiği deneysel olarak gösterilmiştir. dolayısıyla bir kristal 40.000 MB ya da 40 GB veri depolayabilmektedir. Gelecekte ise bu miktarın 100 GB'a çıkması planlanmaktadır (1). Söz konusu depolama tekniğine göre her parçaya lazer ışını belli bir açıyla gönderilir. Veri kaydı yapılacağı zaman laser ışını veri ve adres kısmı olmak üzere iki parçaya ayrıştırılır. **Veri kısmı**, binit matrisine uygun biçimde module edilmekte ve **adres kısmı** parçadaki belli bir sayfaya gelecek şekilde açılandırılmaktadır. Adres kısmı kristale yazılacağı zaman her iki kısım birlikte gönderilir. 1'e module edilmiş ışık kristaldeki iç yapıyı o noktada değiştirirken 0'a module edilmiş ışığın herhangi bir etkisi bulunmamaktadır. Okuma için ise ışının yalnız veri kısmı kullanılır ve parçanın arka kısmından yansıyan ışın deseni kişi tarafından algılanarak okuma gerçekleştirilir.



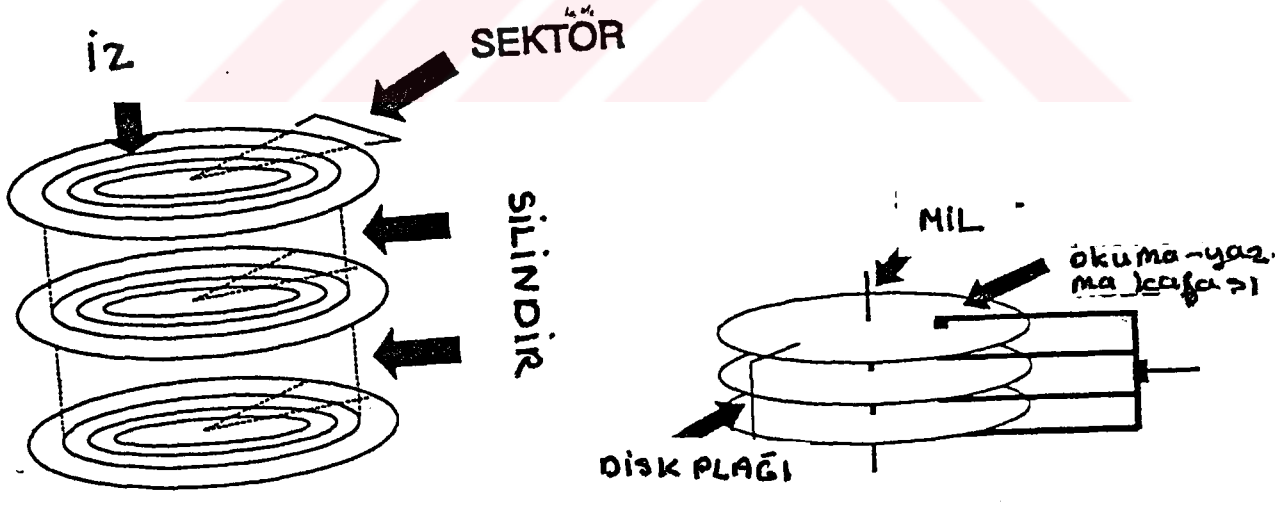
Şekil 5.1: Holografik depolama tekniği (2)

1. Helge Holvik, Databaseteori. Oslo: Statens Bibliotek og Informasjonshogskole, 1991, 165 s.
2. Aynı, 166 s.

Yukarıda anılan teknolojilerle üretilen depolama ortamları disk teknolojisine alternatif olmaya devam etmektedir.

Disk yüklü miktarda bilgi taşıyabildiği, çabuk enşim sağladığı ve depolama ile erişimde düşük maliyeti olduğu için halen en cok kullanılan depolama gerecidir.

Disk bir mile bağlı birçok disk plağından meydana gelmektedir. Mlin belli bir dönme hızı vardır (dakikada 3600 dönüş) Disk plakları arasında bir kola bağlı olarak hareket eden okuma/yazma kafası bulunmaktadır. Her disk plağında izler (tracks) vardır. Plak aynı zamanda bir pasta dilimini andıran sektörlere bölünmüştür. Sektör, disk plağı üzerindeki en küçük adres birimidir. Blok ya da sayfa ise bir ya da daha fazla sektörden oluşmaktadır. Aynı pozisyondaki izler ise silindir olarak adlandırılmaktadır.



Şekil 5.2: Diskin yapısı (3)

3. Aynı, 157 s.

Disk üzerinde yer alan bir doğa erişim zamanı hesaplanırken gözönünde bulundurulan noktalar şöyle özetlenebilir:

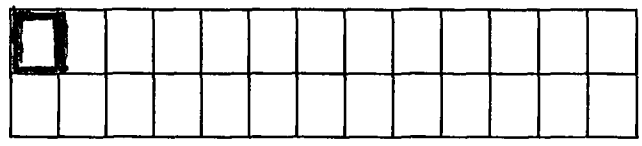
- kolun hareket zamanı,
- başlık seçim zamanı,
- görünüm zamanı,
- aktarım zamanı.

Disk, fiziksel olarak **disk kotrolü** ile denetlenir. **Disk kontrolü** program rutinleri ile gerçekleştirilmektedir. Söz konusu program rutinleri **disk yönetim sistemi** olarak adlandırılmaktadır. Anılan rutinler, disk üzerinde hangi verinin nerede olduğu gibi bilgilere gerek duyar. Bu tür bilgiler işletim sisteminin veri rehberinden sağlanmaktadır. Disk rehberinde işletim sistemi rehberinden farklı tipte bilgiler yer alır. Örneğin, kütük adı, büyüklüğü, kütükteki verilerin disk adresi gibi. Disk üzerinde belli bir miktar sayfa, sayfa içinde de bir ya da daha fazla sayıda blok bulunmaktadır. Disk üzerindeki herbir kütük ise bir sayfa kümesinden oluşmaktadır.

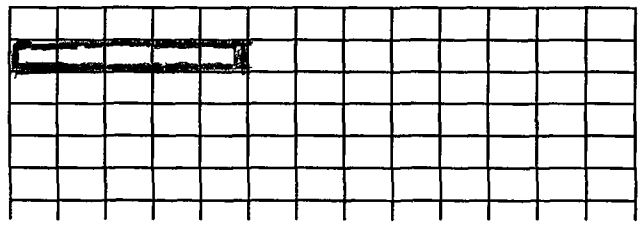
Örnek

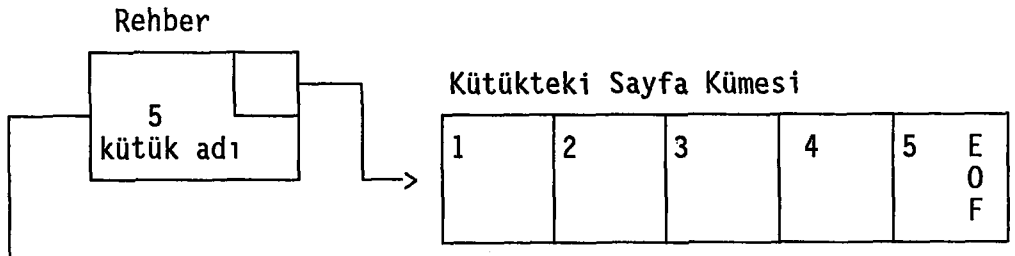
5 sayfadan oluşan bir kütük örneği

Rehberdeki sayfa kümesi



Kütük sayfalar

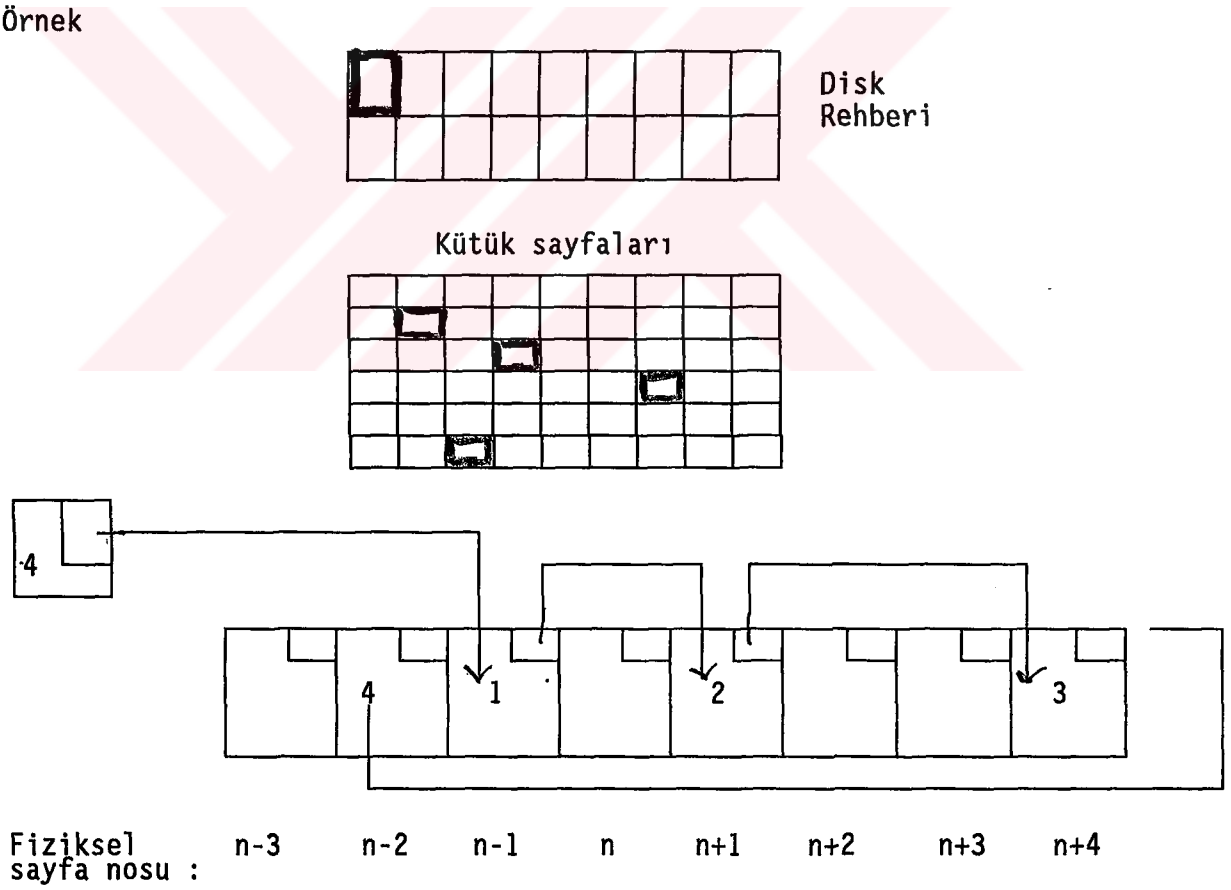




Şekil 5.3: Disk rehberi ile disk üzerindeki kütükler arasındaki ilişki(4)

Dinamik kütük yapısı söz konusu olduğunda kütüğün kapasitesini sayfa eklemek ya da çıkarmak suretiyle değiştirmek olanaklıdır. Kütük sayfaları dağınık biçimde bulunmalarına karşılık göstericiler yardımıyla birbirlerine bağlanmışlardır.

Örnek



Şekil 5.4: Dinamik disk rehberi yapısı (5)

4. Aynı, 115 s.
5. Aynı, 117 s.

Birinci dereceden verilerin söz konusu edildiği temel depolama yapıları iki ana birimden oluşur.

1. Doğrudan yapılandırılmış kütükler,
2. Sıralı kütükler

1. **Doğrudan Yapılandırılmış Kütükler** : Burada önceden adreslenen depolama ortamında her anahtar değerine karşılık bir adres bulunması ilkesi gözetilir. Anahtar değerlerinin tek bir adrese karşılık gelecek biçimde dağılması için kaydın bulunduğu adreste karşılık birden fazla kaydın olmaması gözönünde bulundurulan önemli noktalardır.

Bir kaydın adresini belirleme amacıyla aşağıdaki formul geliştirilmiştir. Buna göre:

d_{alt} : en düşük anahtar değeri,
$d_{üst}$: en yüksek anahtar değeri,
n : depolama alanının alabileceği kayıt sayısı,
u : kayıt uzunluğu,
k : güncel anahtar değeri,
adr : kaydın başladığı yeri gösteren kütüğün başladığı yer ile bağlantılı adres.

$$adr = \frac{(k-d_{alt}).u(n-1)}{(d_{üst} - d_{alt})} \quad (6)$$

6. Kjetil Bratsbergensen-Knut Hofstad-Kjell Wibe, Filsystemer og Databaser. 7. opplag (Trondheim; Institutt for Databehanling, 1983, III-6 s.

Örneğin en fazla 100 kayıt alabilecek bir kütüğe anahtar değeri 3010 ile 4000 arasında değişen bir grup kaydın depolanması söz konusu olduğunda ve her kayıt için 3 sözcüğe gereksinim duyulduğunda **3070 anahtar değerine** sahip kaydın adresi formule göre şöyle hesaplanır.

$$adr = \frac{(3070-3010).3.(100-1)}{(4000-3010)} = 18$$

0	3	6	9	12	15	(18)	21	24	adres
3010	3030	3040				(3070)			anahtar

2. **Sıralı Kütükler:** Bu tür kütük yapıları **seri** ve **sınıflanmış** kütükler olmak üzere iki ana grupta ele alınmaktadır. **Seri** kütüklerde kayıtlar herhangi bir düzen gözetilmeksizin depolanmışlardır. O nedenle kayıtların kütük içerisindeki yerleri bize onlara erişim ile ilgili bilgi vermez. Bu bakımdan belli bir kayda erişmek için kütükteki kayıtlara tek tek bakmak gerekmektedir. Bu tür aramaya çizgisel arama denir. Böyle bir aramada tek tek karşılaştırma yapma zorunluluğu bulunduğundan tarama uzunluğu ortalama kayıt sayısı ile doğrudan bağlantılıdır [7].

$$\text{Tarama uzunluğu} = \frac{(n+1)}{2}$$

(burada n, kayıt sayısıdır)

Sınıflanmış kütükler ise a) anahtara göre, b) frekansa göre sınıflanmış olmak üzere iki tiptir. **Anahtara göre sınıflanmış** sıralı kütükler de anahtar değerleri artan veya azalan yönde düzenlenmişler-

7. Aynı, III-7 s.

dir. Burada doğrudan yapılandırılmış kütüklerde olduğu gibi anahtar ile adresi arasında doğrudan bir bağlantı yoktur. Bu bakımdan kayıt girişi yapılırken veya kaydın iptali karşısında birtakım sorunlar ortaya çıkabilir.

Anahtara göre sınıflanmış sıralı kütüklerde kaydın iptali iki biçimde gerçekleştirilir [8].

1. Silinen kayıt alanının boş olduğunu vurgulamak için özel bir işaretle alan belirlenir.

Örnek :

1	2	3	4	5	6	7	8	9	10	{ kaydın yeri
5	11	13	15	31	50	59	61	66	69	{ kayıt anahtarı
						X X X X				veri

2. Kütükten silinen kayıt alanının kendisinden sonra gelen kayıtlarla doldurulmak suretiyle boş tarafa doğru kaydırılması.

1	2	3	4	5	6	7	8	9	10
5	11	13	15	31	50	61	66	69	

8. Aynı, III-9 s.

Anahtara göre sınıflanmış sırasal kütüklerde yeni bir kaydın kütük adresi belirlenirken ortalama n/2 kaydın kaydırılması gerekmektedir. Bu yöntem özellikle kayıt sayısının fazla olduğu durumlarda zaman alan bir uygulamadır. Ancak bu sorun kayıtlar arasında belli aralıklarla bırakılarak boşluklar yardımı ile çözümlenebilir. Bu tür kütüklerde tarama uzunluğu taramanın hangi ortamda yapıldığı ile bağlantılıdır. Tarama ana bellekte yapılıyor ise tarama uzunluğu karşılaştırma sayısı ile ölçülmektedir. Tarama ikincil depolama alanında gerçekleştiriliyor ise tarama uzunluğu erişim kollarının sayısı ile ifade edilmektedir. Çok fazla kayıt içeren kütüklerde tarama zamanını kısaltmak amacıyla birtakım yöntemler geliştirilmiştir. Bu yöntemlerden biri olan ikili taramada, tarama iki adımda gerçekleştirilir.

1. Taramaya en uygun kaydın bulunması ile başlanır,

2. Aranan anahtar değeri ile bulunan değerin karşılaştırması yapılır. Değerin büyük ya da küçüklüğüne göre tarama yönü saptanarak tarama işlemi o yönde doğru anahtar değeri bulunana değin sürdürülür Bu durumda tarama uzunluğu şöyle hesaplanır:

1. bölünme	n/2 kayıt daha
2. bölünme	
k. bölünme	$n/2_k$ kayıt daha
a. bölünme	$n/2_{a+1}$ kayıt daha

Bölünme işlemi sırasında anahtar karşılaştırması yapılarak benzerlik sınanıyor ise ortalama tarama uzunluğu $a = \frac{n+1}{n} \log_2 (n)-1$ formülü ile hesaplanır [9].

9. Aynı, III-13 s.

Basamaklı Arama , gelişmiş çizgisel bir arama yöntemidir. Burada kütük n/k kadar gruba bölünür. Tarama öncelikle gruplarda anahtarların karşılaştırılması yoluyla yapılır. Karşılaştırma işlemi daha büyük bir kayıt anahtarı bulununcaya kadar sürer ve doğru aralık bulunduğunda grup içinde çizgisel tarama gerçekleştirilir.

1. adım için

tarama uzunluğu = n/k iken

2. adımda tarama uzunluğu

$$a = \frac{\frac{n}{k} + 1}{2} + \frac{k+1}{2} = \frac{n}{2^k} + \frac{k}{2} = 1'dir \quad (10)$$

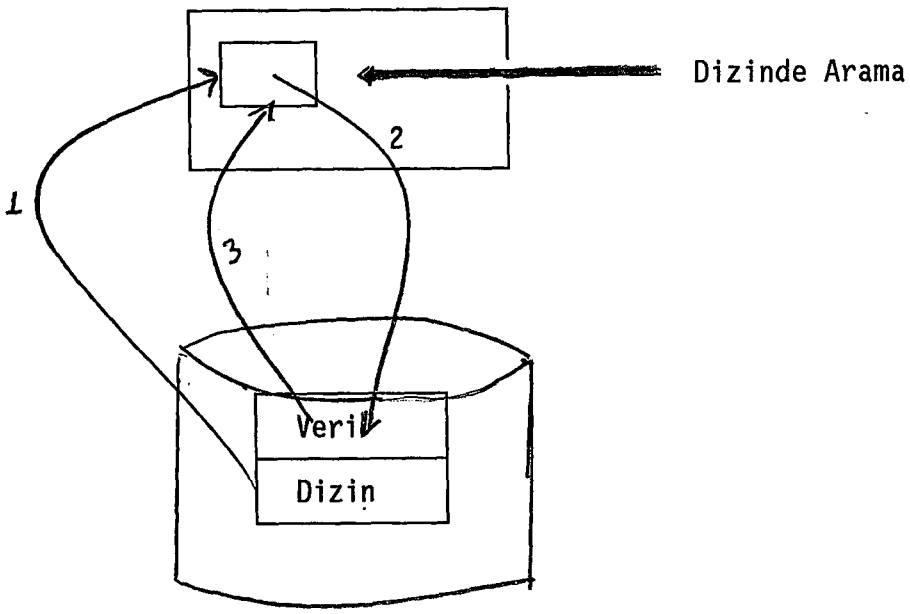
5.2 ERİŞİM YOLLARI

Bu bölümde iki erişim yolu ele alınacaktır. Dizinleme ve rastgele erişim.

Dizinleme : Dizin bir veri yapısı olup iki alandan oluşmaktadır. Alanlardan birinde tüm özellik değerleri veya bunların bir kısmı, diğerinde ise özellik değerlerinin bağlı olduğu ilgili kayıtların adresleri yeralmaktadır. Kayıt sayısının az olduğu durumlarda program dizin kütüğünü ikincil depolama alanından alarak makinanın belleğine getirir ve tarama işlemini burada gerçekleştirir. Ancak kayıt sayısı çok olduğu zaman tüm kütüğün belleğe alınıp okunması güçleşir. O nedenle dizini çeşitli düzeylerde oluşturmak bir çözüm olarak getirilmiştir.

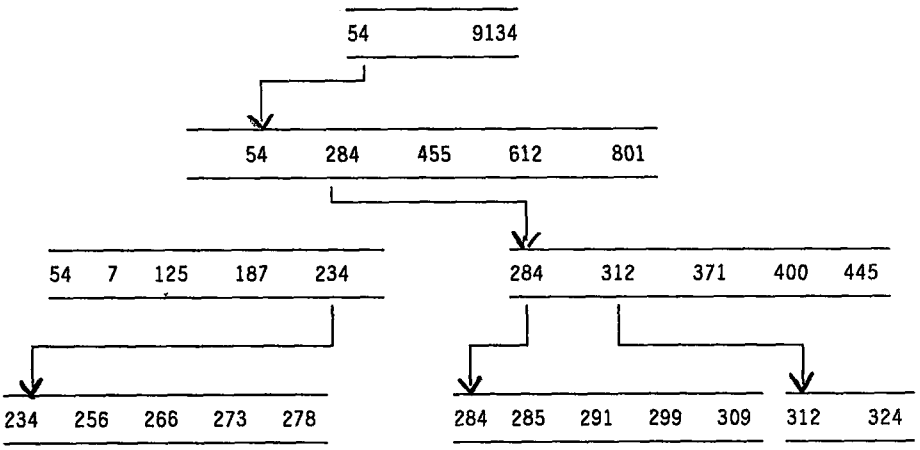
---

10. Aynı, 9 s.



Şekil 5.5: Dizinde arama (11)

Dizinlenmiş Sıralı Kütük (Indexed Sequential File): Değerlerin fiziksel olarak sıralı biçimde düzenlendiği kütüklerdir. Tüm dizin değerlerinin belleğe alınıp okunması güç olduğundan dizin kütüğü bloklara bölünür ve bloklar halinde belleğe alınıp okunur. Örneğin "291" anahtar değeri arandığında şöyle bir yol izlenmektedir.



11. Halvik, a.g.e., 177 s.

Program aramaya dizin ağacının kökünden başlar. Öncelikle arama değeri olan 294'ü 54 ile, daha sonra 934 ile karşılaştırır. 54 < 291 < 934 olduğundan program 54 değerinden bir alt düzeydeki dizin bloğunu gösteren göstericiyi izler. Bu tür arama göstericinin arama değeri olan "291"'i bulmasına değin sürer. Doğru bloğun bulunmasının ardından ikili arama ile doğru veriye ulaşılır. Bloğa yeni bir anahtar değerinin girilmesi blokta yer olduğu sürece kolaydır. Blok içinde yapılacak düzenleme ile anahtar, bloktaki yerini alır. Ancak blok dolu olduğu zaman yeni bir anahtar değeri girmek durumunda dizin ağacının yeniden düzenlenmesi gerekir. Böyle bir durumu çözümlemek için **taşma (overflow) blok yöntemi** kullanılır. Yeni girilecek kayıt ayrı bir bloğa geçirilir. En etkili yol bu tür bloğun diskteki her silindir üzerine yerleştirilmesidir. Böylelikle program yandaki bloğa geçerken okuma/yazma kafası hareket ettirilmemiş olur.

Dizinlenmiş sıralı kütüklerde yeni anahtar girileceği zaman girdi yapmanın bir yolu **bloğun ikiye bölünmesidir.** Böyle bir durumda program öncelikle girdinin yapılacağı bloğu bulur. Eğer blok dolu ise yeni değer bölünen bloğa girilir ve blok numarası bir artar. (önceki blok numarası x ise sonraki x + 1 olur) Değerlerin bir kısmı orijinal blokta diğerleri bölünen yeni blokta yeralır. Blokların ayrışması kayıt girdisinin hangi sıklıkla yapılacağına ve kayıt sayısına bağlıdır.

Dizinde bulunan iptal edilmiş değerler belli aralıklarla güncellenmelidir. Disk üzerinde yer kazanılması açısından yapılacak bu işlem önem taşır. Bunun yanında dizin ağacının yüksekliği azalmış dolayısıyla erişim zamanı kısalmış olur.

Depolama alanının verimli kullanımı açısından boş bloklar biraraya getirilir. Programlar söz konusu işlemi gerçekleştiren mekanizmaları oluşturur. Bu amaçla blokta bulunması gereken en az değer miktarı önceden saptanır.

B-Tree: Büyük kütük sistemlerinde veriye erişimde hız kazanılması ve başlık maliyetinin en aza indirgenmesi çabaları sonucu ortaya çıkmış bir düzenleme biçimidir. Bellekte tutulması güç büyük dizinlerin oluşturulmasında ve veriye erişimde verimliliğin sağlanmasında etkili olması nedeniyle veritabanı sistemlerinde standart bir dizin düzeni olarak kullanılmaktadır. "B"nin neye karşılık geldiği yaratıcıları olan Bayer ve MC Creight tarafından açıklanmamakla beraber "balunced", "broad" veya "bushy" karşılığı kullanıldığı sanılmaktadır [12].

Kapsamlı dizinler için en büyük problem, ancak ikincil bellekte tutulabilmeleri ve bu ortamda erişimin yavaş olmasıdır. Daha net ifade etmek gerekirse;

1. İkili arama çok sayıda arama gerektirmektedir. Disk üzerinde bir anahtarın aranması, farklı izlere bakmak anlamına gelir. Birden fazla yerde arama zorunluluğunun bulunması maliyeti artırmaktadır. Oysa amaç arama sayısını azaltmak, maliyeti düşürmektir.

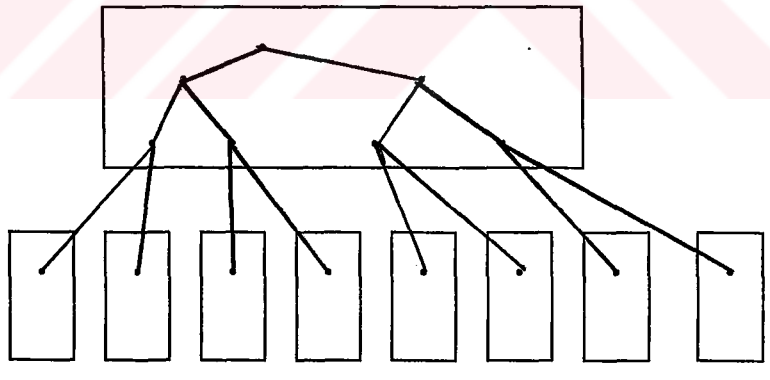
2. İkili arama yapabilmenin koşulu dizinin düzenli tutulmasıdır. Dizine anahtarı girerken ya da silerken dizin düzenlemesinde büyük çapta yenilikler yapmaktan ziyade küçük değişiklerle bu işlemlerin gerçekleştirilmesi yeğlenmektedir. Aksi halde maliyet artmaktadır.

---

12. Michael J. Folk-Bill Zoellick, File Structures. 2nd ed. Reading Massachusetts; Calif: Addison Wesley, 1992, 336 s.

B-tree adındanda anlaşılacağı üzere ağaç veri yapısındadır. Her elemanı bir düğüm, ondan çıkan kolların ucunda yeralan düğümler de onun çocuklarıdır. Düğümün iki alanından biri olan **değer alanı** sabit uzunlukta olabilir. **Diğer alan** düğüme bağlı başka kayıtlara ulaşmayı sağlayan **ilişkili kayıt numaraları** (relative record number: RRN) içerir.

B-tree'de erişim zamanını kısaltmak için ağaç yüksekliğini indirgemenin bir yolu yapıyı sayfalara bölerek bu sayfaları disk üzerindeki bloklara depolamaktır. Böylelikle arama uzunluğu azaltılmış olur. Tamamiyle dengeli bir ağaçta N anahtar içinden bir anahtarı bulma uzunluğu $\log_2(N+1)$'dir. Sayfa açısından gereken arama sayısı ile $\log_{k+1}(N+1)$'dir [13]. Burada k, sayfadaki anahtar sayısını belirlemektedir. Bir düğüme yüklenebilen azami gösterici sayısına B-tree'nin düzeni denir.



Şekil 5.6: Sayfalara Bölünmüş B-tree Örneği [14]

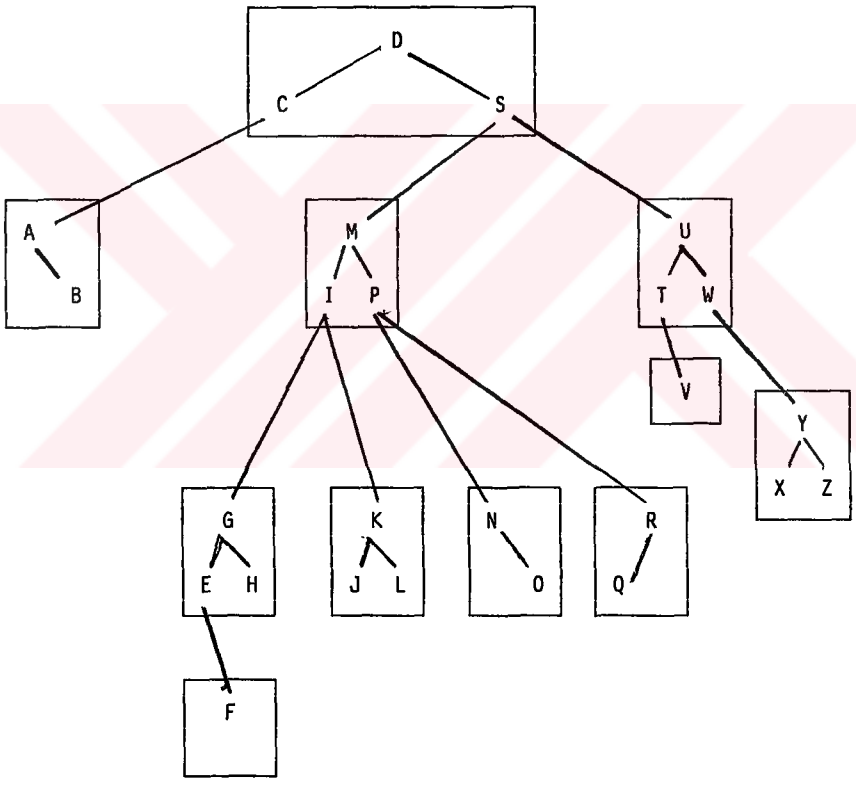
Bir ağacın sayfalara bölünmesi stratejisi disk gibi ikincil depolama gereçlerinin karakteristiklerine uymaktadır. Ağacın yapılan-

13. Aynı, 344 s.
14. Aynı, 344 s.

dırılmasında iki yola çıkış noktası vardır. Birincisi sıralı bir anahtar dizisinin, ikincisi anahtarların geldikçe gelişi güzel düzende ağaca girilmesidir. Kök'ten düğümlere doğru anahtarların girilmesi durumunda kimi sorunlar karşımıza çıkar. Örneğin ;

C S D T A M P I B W N G U R K E H O L J V Q 2 F X V

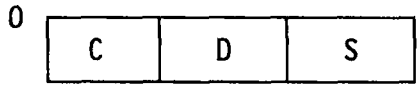
gibi bir anahtar dizisini sayfalara bölünmüş bir ağaç yapısına dönüştürmek istediğimizde aşağıdaki gibi bir görünüm elde edilir ([15]).



Anahtarların geliş sırasına göre sayfalandırılmış ikili ağaca yerleştirilmesinde ve her sayfanın en fazla üç anahtar alabilmesi dikkate alındığında yukarıdaki örnekten de anlaşılacağı gibi c ve d anah-

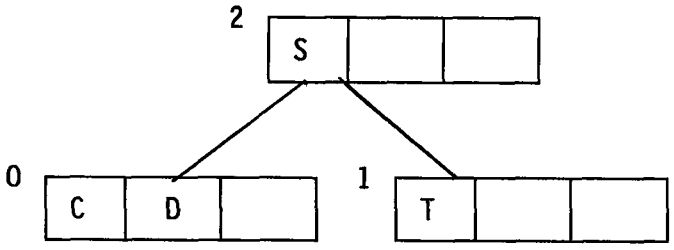
15. Aynı, 345 s.

tarları ardarda geldiğinden ve aynı sayfada yeraldığında ağacı dengede tutmak güçleşmektedir. Böyle durumlarda dengeyi sağlamak için sayfa içinde anahtarların yeniden düzenlenmesi olanaklıdır. Ancak ağaç içinde sayfa düzenlemesi yapmak son derece güçtür. Bu sorunların çözümü için önerilen bir yol ağacın aşağıdan yukarıya doğru yapılandırılmasıdır. Bu durumda aynı anahtar dizisi ağaca şöyle dönüştürülür.

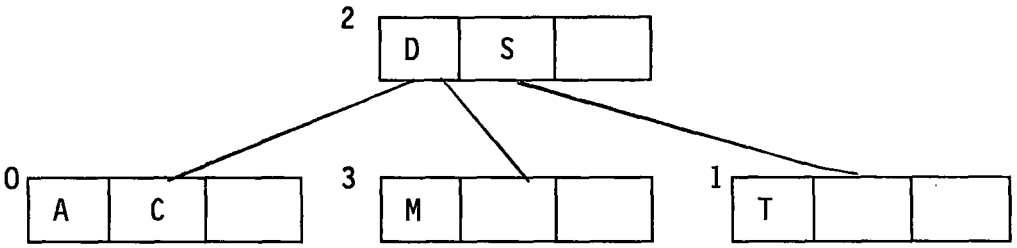


Geliş sırasına göre ilk üç anahtar ancak üç anahtar içerebilen kök sayfasına girilir. Her sayfada anahtar sayısının bir fazlası oranında anahtardan sonra gelen diğer anahtarları işaret eden göstericiler bulunmaktadır. Göstericilerin sayısı bu durumda 4, sayfanın düzeni de 4'tür. Yukarıdaki "0" no'lu sayfaya "T" gibi yeni bir anahtar girildiğinde sayfa dolu olduğu için iki ayrı sayfaya bölünür. Bu durumda

"0" ve "1" no'lu sayfalara enşim sağlamak için bir üst düzeye yeni bir sayfa açmak gerekir ve "S" anahtarı yeni sayfaya kaydırılır.

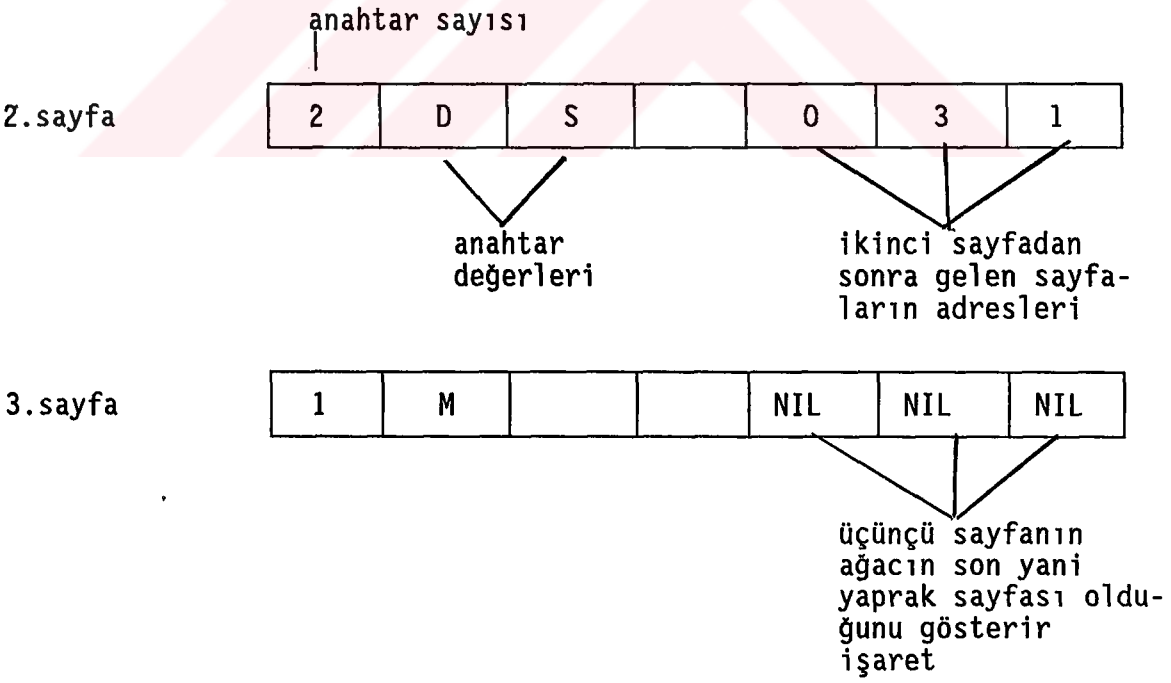


"A" anahtarı girileceği zaman ise "A" anahtarı "C" den önce geldiği için "O" no'lu sayfaya yerleştirilir. Bir sonraki anahtar olan "M" girileceği zaman "M", "D"'dan sonra "S"'den önce geldiği için "O" no'lu sayfaya yerleştirilir ve bu sayfa tekrar bölünür.



Görüldüğü gibi bu yöntemle ağacın dengesi de korunmaktadır. Böyle bir ağaçta tarama sayfa içindeki göstericiler ile yapılmaktadır. Bu göstericiler sayfadan sonra gelen diğer sayfaların adreslerini verir (16).

Örnek :



16. Aynı, 353 s.

B-tree'nin özellikleri kısaca şöyle sıralanabilir:

1. Her sayfanın kendinden sonra gelen en fazla m kadar düğümü vardır;

2. Kök ve yaprak sayfaları hariç her sayfanın kendinden sonra gelen en az m/2 kadar düğümü vardır;

3. Kök'ün kendinden sonra gelen en az iki düğümü vardır;

4. Bir yapraklar aynı düzeyde olur;

5. Bütün yaprak sayfası en az m/2-1 anahtar içerir m-1 den fazla anahtar içermez (-1) değeri ağacın sonuna yani yaprak düğüme ulaşıldığını gösteren simgedir.

2. **Rastgele Erişim (Hashing)** : Rastgele erişimi sağlayan ve yaygın olarak hashing diye adlandırılan fonksiyon tıpkı uçaklardaki kara kutu gibidir. Kara kutuya giren anahtar değerleri birer adres olarak buradan çıkarlar. Böylelikle anahtar ile ilişkili olduğu kayıt adresi arasında doğrudan bir bağlantı kurulmaktadır [17]. rastgele erişimin dizinlemeden farkı adreslerin fonksiyon uyarınca gelişi güzel dağılımı ve iki ayrı anahtarın aynı adrese sahip olabilmesidir. Fonksiyonun farklı iki anahtarı aynı adrese göndermesi burada bir sorun olarak ortaya çıkmaktadır. Aynı harflere sahip olmalarına karşılık farklı kimi anahtarların aynı adrese sahip olmaları ile **çarpışma** mey-

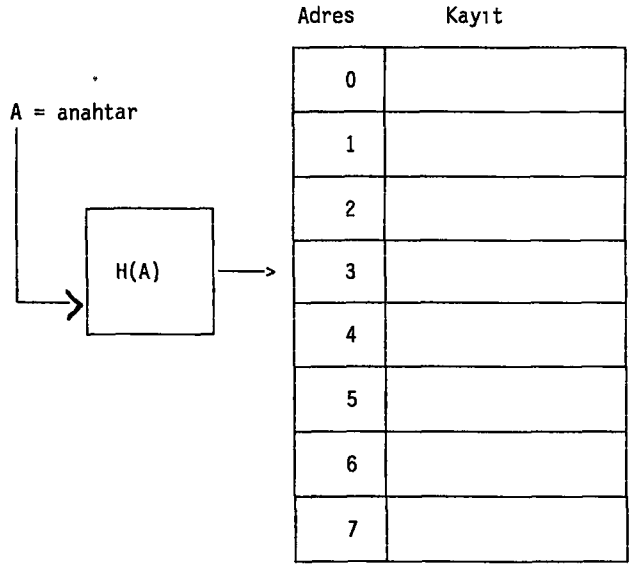
17. Aynı, 447 s.

dana gelir. Çarpışmayı önlemek için bazı çözüm yolları ortaya atılmıştır. Örneğin elde edilen adresler arasında kayıtların gelişigüzel dağılımını sağlayacak bir algoritma ile belli adresler atrafında çok sayıda kaydın birikimi önlenmiş olur. Bir diğer çözüm yolu ekstra bellek kullanımıdır. Burada az sayıda kayıt bulunmasına karşılık çok fazla adresin varlığı soruna çözüm getiriyormuş gibi gözükse de bu durum yer kaybına neden olmaktadır. Bir adrese birden fazla kaydın yerleştirilmesi ise bir başka çözüm olarak üretilmiştir. Fiziksel olarak her kayıt yeri ancak tek bir kayıt tutabilmektedir. Her kütük adresinin birden fazla kaydı tutması olanaklıdır. Bu durumda örneğin 512 baytlık bir fiziksel kütüğe, kayıt uzunluğu 80 bayt olan 6 kayıt depolanabilmektedir.

Rastgele erişim algoritmasında genelde üç temel aşama vardır:

1. Anahtarın sayısal ifadesi (eğer anahtar bir sözdizisi ise onun ASCIIl kodu alınarak sayıya dönüştürülür)
2. Sayının katlarını almak ve toplamak,
3. Adres alanına bölmek,

Rastgele erişim fonksiyonu bir çizimle şöyle açıklanabilir.



Rastgele erişim fonksiyonu kayıtlara hızlı ve doğrudan erişim olanağı sağlama gibi dezavantajlarının yanında çarpışma ve taşma gibi çözümü üzerinde halen çalışılan olumsuzlukları da bulunmaktadır.

Bölüm kapsamında veritabanı kayıtlarının depolanması ve yeniden erişimi için izlenen yollar, alanın verimli kullanımı, maliyetin azaltılması ve hızlı erişim açısından ele alınmıştır. Böylelikle bir veritabanı oluşum süreci veri analizinden başlayarak tasarım, depolama ve erişime kadar tüm evreleri ile bu aşamalarda izlenen yöntemler açısından incelenmiştir.

BÖLÜM VI

SONUÇ VE ÖNERİLER

Şimdiye kadar bilgi merkezlerinde çalışan meslektaşlar için veritabanı sistemi, elektronik ortama belli formatta aktarılmış bibliyografik kimlikler topluluğu ve bu kimliklere erişimi sağlayacak anahtar sözcük dizinleri ile programları ifade etmekteydi. Öte yandan iş dünyasında veritabanı sistemi, işlemlerin çözümlenme sürecine yönelik programlar ve bu programların verilerini sağladığı kütükler olarak algılanmaktaydı. O nedenle uzun yıllar bilgi merkezlerinin otomasyona geçişinde katalog verilerinin bilgisayara aktarılması ve bunlara erişim konusuna öncelik verildi. İş dünyasında ise kapsamlı programların üretilmesi önem taşımaktaydı ve bilgisayar uzmanları bu süreç içerisinde söz sahibi olmuşlardı. Bu konudaki çalışmalarını ise kuruluşların bünyesinde oluşturulan bilgi işlem merkezlerinde yürütmekteydiler. Kurum ve kuruluşların işlemlerinin bilgisayar ortamına aktarılarak girdi, çıktı ve işlem evrelerinin nasıl olması gerektiği konusundaki karar yetkisi de onlara aitti. Yine bu döneme özgü bir özellik veri bağımsızlığı, veri bütünlüğü ve veri güvenliği gibi konuların henüz çözülememiş olmasıydı. Ancak bütünleşik veritabanı sistemlerinin ortaya çıkışı ve gelişimi, yukarıda anılan sorunların çözümlenmesi ve söz konusu sistemlere model arayışı giderek kullanıcıyı ve veri analizi konusunu ön plana çıkardı.

Verinin toplanması, ilişkilendirilmesi ve bu bağlamda bir model uyarınca organizasyonu ile kullanıcı uyarlı dördüncü nesil dille-

rin gelişimi, sistemi kullanacaklara erişim için verinin yerini bilme zorunluluğu olmadan kapsamlı bilgiye erişim olanakları sundu. Özellikle 1970'li yıllardan sonra büyük bir gelişim ivmesi gösteren ilişkisel veritabanı yönetim sistemleri mantıksal yapısının kolay kavranır olması ve kullanıcıya bir seferde birden fazla kaydı görme olanağı tanımasıyla yaygın bir kullanım alanı buldu. Tezimizin II. ve III. Bölümlerinde verilen bilgiler dikkatlice değerlendirildiğinde bu alandaki gelişim ve değişimin yukarıda sunulan görüşleri destekler nitelikte olduğu görülecektir. Özellikle ticari ürünlerde yeğlenen üç (hiyerarşik, ağ, ilişkisel) model içinden ilişkisel modelin giderek baskın bir güç haline gelmesi, üç modelin özellikleriyle ve onları temel almış üç ticari ürünün tanıtımı ile ortaya konmuştur.

İlişkisel veritabanı sistemlerine uygun, doğal dil yaklaşımlı NIAM tasarım tekniği ise tasarımın yani sistemin mantıksal yapısının bu tekniği bilenler tarafından kolaylıkla gerçekleştirilebileceğini kanıtlamaktadır. Çalışmada popüler, geleneksel bir yöntem olan normalizasyon ve kullanıcının dile bağlı olarak düşünce mekanizmasının işleyişine dayanan bir teknik olan NIAM'ın birlikte incelenme nedeni her iki yaklaşımın birbirine olan üstünlüklerini ve zayıflıklarını ortaya koymaktadır. Dolayısıyla normalizasyonda tabloların ayrışma yöntemi ile arınır hale gelmesi ve aday anahtarlararası fonksiyonel bağımlılıkların saptanarak ana anahtarın belirlenme sürecinin NIAM'a oranla çok daha karmaşık olduğu gösterilmiştir. Ancak burada karşılaşılan sorun NIAM'ın Türkçenin yapısına uygun hale getirilmesidir. NIAM'da benimsene şema yapısı İngilizcenin gramer yapısına uygundur. Bu bakımdan doğal dilde oluşturulan yalın gerçeklerin şema diyagramına dönüş-

türülmesi kolaydır. Oysa Türkçede eylem ve eylemi gerçekleştiren öznenin şemanın yapısına uygun hale getirilmesi güçtür. Bu güçlük bütünüyle Türkçenin yapısına uygun olmasada anlamı yitirmeden özne ve eylemin yerini değiştirmek suretiyle aşılmıştır. Bunun yanısıra Türkçenin yapısına uygun olarak ifade edilen yalın gerçeklerin şema diyagramına dönüştürülmesinde tamlamalar ve ekler gözardı edilerek varlık ve rol'ün somut biçimde ortaya çıkarılmasına çalışılmıştır.

İster başlı başına bilgi sağlayan bir merkez isterse bir kuruluş bünyesindeki bilgi merkezi olsun bu tür birimlerin meslekten olan çalışanları böyle bir teknik ile donatıldıklarında enformasyon sistemi tasarımını gerçekleştirebilirler. Özellikle bir kuruluşun bünyesinde etkinlik gösteren merkezler böylelikle yalnız kuruluşun gereksinim duyduğu bilgileri iç ve dış kaynaklardan sağlamakla kalmaz, aynı zamanda kuruluşun birimleri için girdi/çıktı bilginin analizini gerçekleştirebilirler. Bu da bilgi merkezinin işlevlerine yeni bir boyut kazandıracaktır. Yöneticiler tarafından işlevleri doğru algılanamayan bu nedenle edilgen bir konuma sokulan hatta gerekliliği bile tartışılır kılınan enformasyon merkezlerinin statüsü bütünüyle değişecektir. Dolayısıyla tasarımı gerçekleştirecek olan bilgi uzmanlarının görev ve sorumluluklarının yeniden değerlendirilmesini gerekli kılacaktır. Ülkemizde, alanımıza özgü olarak ilk kez böyle bir çalışma yapma girişiminde bulunma nedenimiz konunun enformasyon sistemlerinin altyapısını oluşturduğu ve dolayısıyla bilgi uzmanlarının da bu konuyu girmeleri gerektiği yönünde taşıdığımız inançtır.

Günümüzde bilgi uzmanı (kütüphaneci, enformasyon uzmanı, dökümantalist ve arşivist) yetiştiren kurumlar, üniversite bünyesindeki

fakültelerin kütüphanecilik ya da arşivcilik bölümleridir. Arşivcilik Marmara Üniversitesi'nde başlı başına bir bölüm biçiminde örgütlenmişse de Ankara, Hacettepe ve İstanbul Üniversitelerinde arşiv eğitimi kütüphanecilik ve dokümantasyon, enformasyon anabilim dalı olarak etkinlik göstermektedir. Söz konusu eğitim birimlerinden yetişen insanlar ise üniversite, kurum ve kuruluş, halk ve çocuk, okul vb. türdeki bilgi merkezlerinde istihdam edilmektedir. Ancak söz konusu kurum ve kuruluşların ülkemizdeki düzeyi ve iyi hizmet üretimi için, içinde bulundukları koşullar ve gelişim olanakları değerlendirildiğinde aynı tip insan yetiştirmenin verimli istihdamı olumsuz yönde etkileyeceği düşünülmektedir. Çünkü içinde bulunduğumuz koşullarda herbir tür (kütüphane, arşiv, enformasyon) için gerekli insan güçünün sağlanması pek de olanaklı değildir. Kütüphane, enformasyon ve arşiv alanları için insangücü yetiştirilirken her üç alanın da birbirinden soyutlanmaması gereği unutulmadan ancak teknolojik gelişmeyi yakalamış kuruluşların gereksindiği insangücünün sağlanması zorunluluğu da gözardı edilmeden özellikle lisansüstü programlarda yukarıda sunulan becerilerle donatılmış insangücü yetiştirilmesi yönünde çalışmalar yapılmasının uygun olduğu düşünülmektedir. Yönetim, tasarım ve bilgi erişim gibi konuların yoğunlukla verileceği uzmanlığa yönelik programlarla alanımız için lider ve danışman niteliğine sahip insan gücü yetiştirilebilecektir. Böylelikle daha önce de dediğimiz gibi bilgi uzmanı ve bilgi merkezi kavramları özellikle iş dünyasında değişecek yeni bir bakış açısı sağlanmış olacaktır. Bunun yanında meslek elemanlarının diğer meslekler-den alanımıza yapılan sızmalar engellenmesini kolaylaştıracak ve rekabet gücünü artıracaktır.

Çalışmanın başlangıcında varsayımımız veritabanı yönetim sistemlerinin şimdiye kadar bilgi merkezleri açısından sahip oldukları kaynakların niteliği gözönünde bulundurularak hep bibliyografik verilerin depolanması ve bunlara erişim işlevini yerine getirdiği, oysa ilişkisel modelin yalnız bibliyografik denetim değil bilgi merkezlerinin tüm işlemleri için kullanılabilen esnek bir model olduğu ve bunun doğruluğunun kullanılan NIAM tekniği ile gösterileceği yönünde oluşturulmuştu. Çalışmanın sonunda kütüphanedeki sipariş işlemi için oluşturulan kavramsal şema ve bu şemanın ilişkisel şemaya dönüştürülmesi süreci varsayımımızın doğruluğunu kanıtlamıştır.

KAYNAKÇA

Açıklamalı Bilgisayar Terimleri Sözlüğü, haz. Filiz Yalçıner, Fikret Şahin. (İstanbul: Fono Açık Öğretim Kurumu, 1993) 328

Aydın, Emin D, Veritabanı. İstanbul: Evrim Basım Yayın, 1980.

Başer, Ender. "Bütünlüğün Veritabanı Sistemlerindeki Önemi ve Kıyaslanması". Bilgisayar Araştırma ve Uygulama Merkezi Dergisi.8:1 (1985). 35-42

Baykut, Hakan. "Bilgisayar Teknolojisi ve Örgütsel Değişim." Bakış. 3:9 (1988) 10-11

Belkıs, W. Leong Hong-Bernard K. Playman, Data Dicrionary; directory systems, administration implemantation and usage. New York: John Wiley and Sons, 1982

Bergensen, Kjetil-Knut Hofstad-Kjell Wibe, Filsystemer og Databaser. 7. opplag (Trondheim: RUNIT Institutt for Databehandling, 1983) VII-28

Bilişim Terimleri Sözlüğü, haz: Aydın Köksal. (Ankara, Türk Dil Kurumu, 1981) 126

Bloomberg, Marty-G., Edward Evans. çev: Nilüfer Tuncer Kütüphane Teknisyenleri İçin Kütüphane Hizmetlerine Giriş. (Ankara: Türk Kütüphaneciler Derneği, 1989) 412

Codd, E.F., "A Relational Model of Data or Large Shared Data Banks." CACM. 13:6 (1970)

Codd, E.F., The Relational Model for Database Management. versiyon: 2 Reading Massachusets Calif: Addison Wesley, 1990.

Curtis, Graham. Business Information Systems; analysis, design and practice. Washington: Addison Wesley, 1989

Çapar, Bengü. "Bilgi İşletmelerinin Yönetiminde Sistem Yaklaşımı ve Sistem Analizi". Jale Baysal'a Armağan. Yay. haz. Hasan S. Keseroğlu (İstanbul: İstanbul Üniversitesi, Mart 1993) 51-71

Database Task Group of CODASYL Programming Language Committea. Report. Nisan 1971

Date, J.C., An Introduction to database systems. Vol. 1. 4th ed. Reading Massachusetts: Addison Wesley, 1985

Date, J.C., Relational Database; selected writings based on appendix A (pp 265-275) of A Guide to DB2 (Reading Massachusetts, Calif: Addison Wesley, 1984)

Dizin; Türk Kütüphaneciler Derneği Bülteni/Türk Kütüphaneciliği, 1952-1992. Ankara; Kütüphaneler Genel Müdürlüğü, 1993.

Doğaç, Asuman. "Veri Temeli Sistemlerine Kısa Bir Bakış". Türkiye Bilişim Derneği Dergisi. 6:11 (1977) 91-96

Fagin, R. "A Normal Form for Databases That is Bade on Domains and Keys." ACM Transactions on Database Systems. 6 (September 1981)

Fazlı, Can. "Veritabanı Yönetimi". Bilgisayar 9:62 (1986) 65-71

Fife, Dennis W,W. Terry Hardgrave Donald R. Deutsch. Database Concepts. (Cincinnati, Ohio=South Western Pub., 1986) 294

Folk, Michael J.-Bill Zoellick File Structure: 2nd ed. Reading Massachusetts, Calif: Addison Wesley, 1992

Frame, Michael-Mehdi Owrang. "SQL Translation Using An Attribute Grammer". Information Sciences. 71: 3 (1993) 267-287

Gulliksen, Tor. Matematiki Praksis. 2. utgave (Oslo : Universitets, forlaget, 1991) 382

Güder, Gazi. Bilgi İşlem Terimleri Sözlüğü. (İstanbul: Birsen Yayınevi, 1986) 555

Holvik, Helge, Databaseteori. (Oslo, Statents Bibliotek og Informasjonshogskole, 1991) 276

Jones, John Anthony. Database in Theory and Practice. (London: Kogan Page, 1986) 324

Kaptan, Saim. Bilimsel Araştırma ve İstatistik Teknikleri. Ankara: [Yayl yok], 1993

Karasar, Niyazi. Bilimsel Araştırma Yöntemi. 3. bs. Ankara: Bilim Kitabevi, 1986

Koffman, Eliot B. Pascal; problem solving and program design. 4th ed (Reading Massachusetts, Calif: Addison Wesley, 1992) 862

Korth, Henry-F. Abraham Silberschatz, Database System Concepts. (Singapore: Mc Graw Hill, 1986) 546

Kök, Ethem Refi. "Data Yığını Yönetim Sistemleri". İstanbul Teknik Üniversitesi Elektronik Hesap Bilimleri Enstitüsü Elektronik Bilgi İşleme Ulusal Sempozyumu II, 25-26,4,1977 (İstanbul: İstanbul Teknik Üniversitesi, 1977) 165-173

Kroenke, David Kathleen A. Dolan. Database Processing; fundamentals, design, implementation. 3rd ed. (Chicago, Science Research Associates, 1990) 687

Kurtaran, Özlem Meltem Fuat Çubukçu. Ansiklopedik Bilgisayar Terimleri Sözlüğü. (İstanbul: Türkmen Kitabevi, 1991) 367

Longley, Dennis Michael Shain. Dictionary of Information Technology. 2nd ed. (London: Mc Millian, 1985) 382

Martin, Daniel. Advanced Database Techniques. (Cambridge: MIT press, 1986) 377

Matematik Terimleri Sözlüğü. Haz: Doğan Çoker-Timur Karaçay- (Ankara: Türk Dil Kurumu, 1983) 502

Mc Fadden, Fred R. - Jeffrey A. Hoffer. Database Management. (Menlo Park, Calif: The Benjamin Commings Pub.Com., 1985) 723

Microsoft Press Computer Dictionary; the comprehensive standart for business, school, library and home. (Redmond, Washington: Microsoft, 1991) 392

Mittra, Sinatsu S. Principles of Relational Database Systems. (Englewood cliffs, NJ.: Prentice Hall, 1991) 323

Nijssen, Gerardus Maria Terence Aidan Halphing Conceptual Schema and Relational Database Design; a fact oriented approach. (New York, Prentice Hall, 1989) 343

Nishimoto, Hideki-Ura Shoji. "Complex View Support for a Library Database System." Information Processing and Management. 25:5 (1989) 515-525

Özsu, Tamer M.-Behçet Sarıkaya E. Atalay Özkarahan. "Veritabanı bilgisayarları için yüksek düzeyli kullanım dilleri". Ulusal Bilişim Kurultayı II, 18-20 12, 1978. Bilişim 78 bildiriler. (Ankara: Meteksan, 1978) 81-89

Pratt, Philip J-Joseph J. Adamski. Database Systems; management and design. (Boston: South Western Pub., 1991) 757

Rishe, Naphtali, Database Design Fundemantals; a structured introduction to databases and a structural application design methodology. (Englewood Cliffs, N.J.: Prentice Hall, 1988) 421

Soysal, Ataç. "Yönetimde Bilgisayarlar ve Günümüz Endüstriyel İşletmelerinde Bilgisayarın Yeri". Bilgisayar Destekli Yönetim Sistemleri: Seminer (İstanbul: İstanbul Teknik Üniversitesi İşletme Fakültesi Endüstri Mühendisliği Bölümü, 1989) 5-17

Tanrıkut, Mete. "Sistem Analizi". Bakış 2:6 (1987) 8-12

Tuğcu, Nejat. "Veri Modellerinde Veri Tanım Dilleri". Ulusal Bilişim Kurultayı II Ankara 18-20,12, 1978, Bilişim 78 Bildiriler. (Ankara: Meteksan, 1978) 75-80.

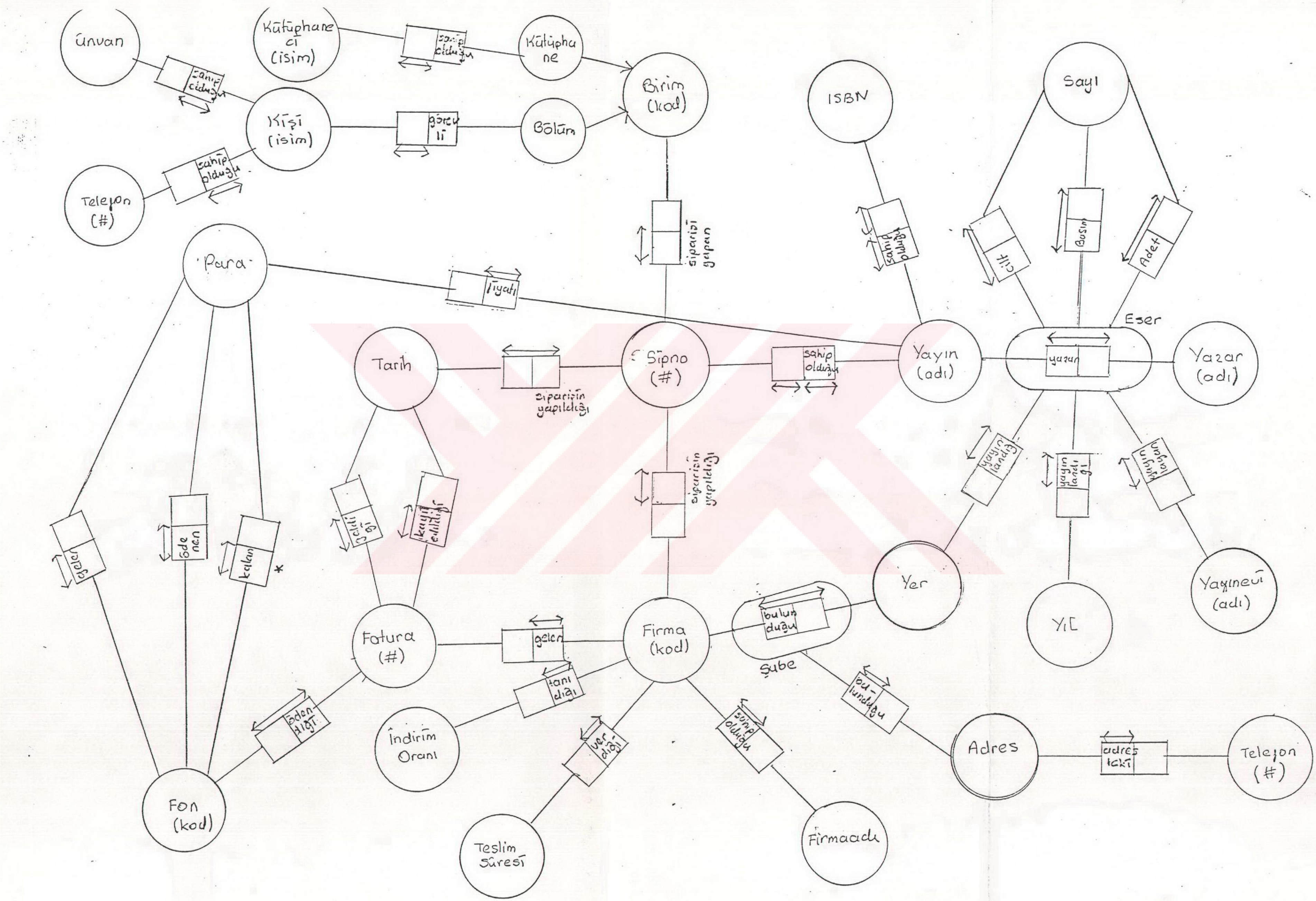
Türkiye Bibliyografyası. Ankara: Maarif Vekaleti, 1949-

Türkiye Makaleler Bibliyografyası. Ankara: Milli Kütüphane, 1952-

"Veritabanı Yönetimi" Bilgisayar 8:52 (1985) 67-74

EK I

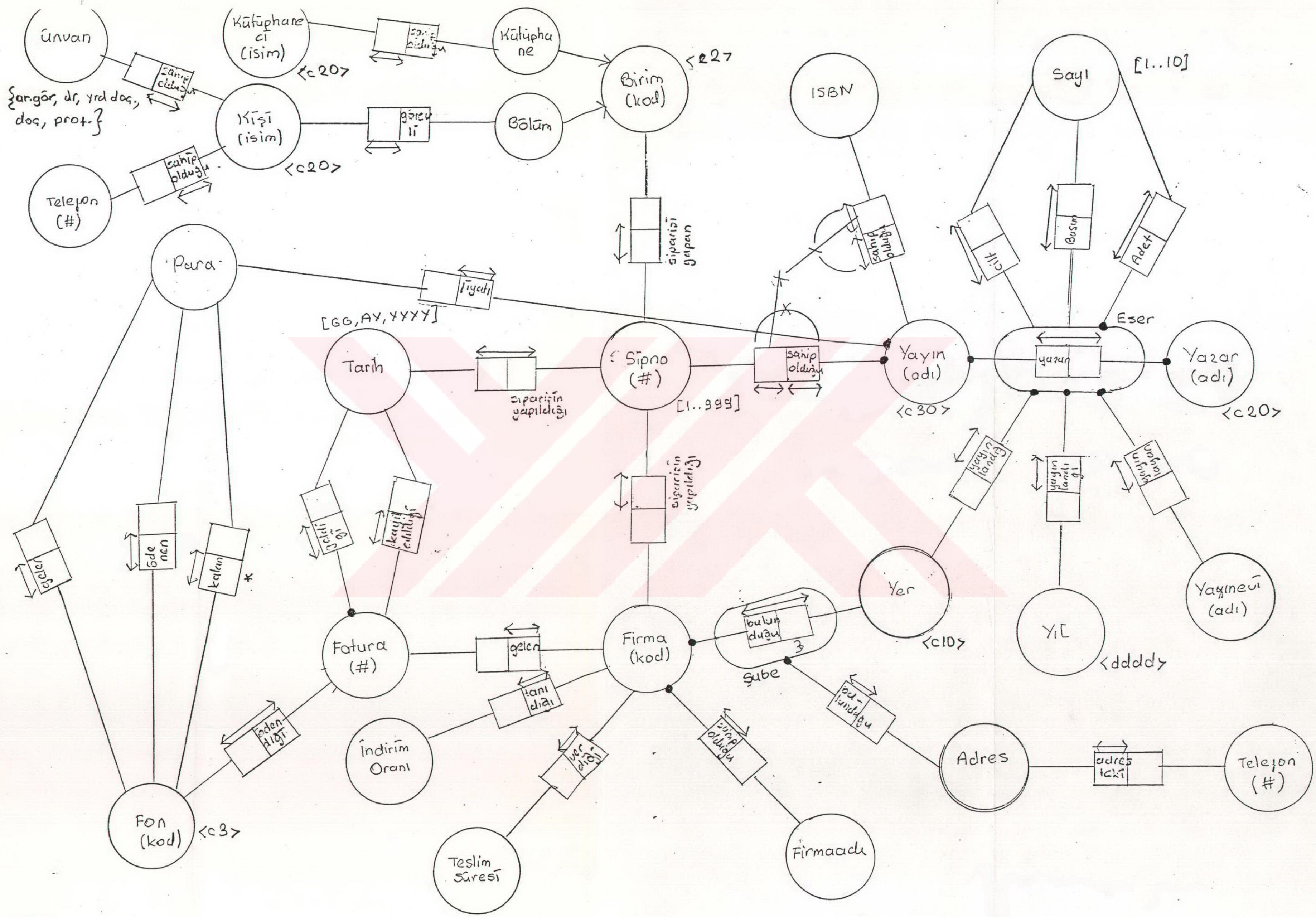
SİPARİŞ ŞEMASINDA TEKLİK SINIRLILIKLARI



Ünvan
Kütüphaneci (isim)
sahip olduğu
Kütüphane
Birim (kod)
Kişi (isim)
görevli
Bölüm
Telefon (#)
Para
Fiyatı
siparişi geçen
ISBN
Sayı
Cilt
Basım
Adet
Eser
Tarih
siparişin yapıldığı
Sipno (#)
Yayın (adı)
yazarı
Yazar (adı)
yayınlandığı
Yer
Yıl
Yayınevi (adı)
gelen
ödenen
kalan
geldiği
kayıt edildiği
Fatura (#)
Firma (kod)
bulunduğu
Şube
tanıdığı
ödendiği
İndirim Oranı
verdiği
Adres
adresteki
Fon (kod)
Teslim Süresi
Firmaadı

EK II

SİPARİŞ ŞEMASINDA DİĞER SINIRLILIKLAR



Ünvan
{arşgör, dr, yrd doç, doç, prof.}
sahip olduğu
Kütüphaneci (isim)
<c20>
Kütüphane
Birim (kod)
<c2>
Kişi (isim)
<c20>
görevli
Bölüm
Telefon (#)
sahip olduğu
Para
fiyatı
ISBN
Sayı
[1..10]
cilt
Basım
Adet
siparişi geçen
sahip olduğu
[GG, AY, YYYY]
Tarih
Sipno (#)
[1..999]
siparişin yapıldığı
sahip olduğu
Yayın (adı)
<c30>
Eser
yazarı
Yazar (adı)
<c20>
yayınlandığı
yayınlandığı
yayınlandığı
geldiği
kayıt edildiği
siparişin yapıldığı
gelen
ödenen
kalan
Yer
<c10>
Yıl
<dddd>
Yayınevi (adı)
Fatura (#)
gelen
Firma (kod)
bulunduğu
Şube
tanıdığı
ödendiği
bulunduğu
sahip olduğu
verdiği
İndirim Oranı
Adres
adresteki
Telefon (#)
Fon (kod)
<c3>
Teslim Süresi
Firmaadı

EK III

SÖZLÜK

SÖZLÜK

Aday Anahtar (Candidate Key): Bir kütükteki kayıtlara erişmek için kullanılan veri alanı; uluslararası veritabanlarında ana anahtar özelliklerini taşıyan anahtar.

Ağ Veri Modeli (Network Data Model): Bir veri tabanı yapısı olup tüm birimleri birbirine bağlayan bir modele sahiptir; birbirleriyle göstericiler aracılığı ile bağlanan kayıt kümeleridir.

Ağaç Veri Yapısı (Tree Data Structure): Bir döngü oluşturmaksızın birbirine bağlanmış düğümlerin meydana getirdiği veri yapısı; kök olarak adlandırılan başlangıç düğümü diğer bütün düğümlere giden yolu belirler, son düğümler yaprak olarak adlandırılır ve yolun sonuna gelindiğini belirtir.

Alternatif Anahtar (Alternat ve Key): Aday anahtarlar içinden belirlenen ana anahtarın bir yana ayrılmasının ardından geride kalan diğer anahtarlardır.

Ana Anahtar (Primary Key): Tek ya da bir grup anahtar olup bir ilişki sırasını tek başına tanımlayan özelliktir; veri tabanlarında bir varlığı tek başına tanımlayabilen anahtardır; kayıt derlemesi ya da bir kütükte yeralan belirli bir kayda erişmek için kullanılan kayıt içerisindeki belli bir alana eşit olan anahtar değer.

Anadil (Host Language): Programlamada kullanılan yüksek düzeyli dil olup içinde veri tabanı yönetim sistemine ait işlem komutlarına yer verir.

Ara Bellek (Buffer): Bilgisayar ve çevre birimleri arasındaki veri iletişiminde verilerin geçici olarak saklandığı bellek alanı; iki farklı birim arasındaki hız ve gerilim farklarının birbirine etkisini önler; girdi/çıktı işlemlerinde verimin okunması ya da yazılması için ayrılmış bellek alanıdır.

Arabirim (Interface): Elektronikte birbirine bağlı iki gereç arasındaki ortak sınırdır veya birimler arasında geçiş yapan sinyallerin biçim ve tipini belirlemek amacıyla tanımlanmış öğelerdir; bilgi işlem sistemleri arasında veya tek bir sistemin parçaları arasındaki ortak kullanımlı sınırdır. Kullanıcının programlarla iletişimini sağlayan arabirimler olduğu gibi bilgisayar birimleri arasındaki uyumlu çalışmayı sağlayanları da vardır.

Bütünleşik Veritabanı Yönetim Sistemi (Integrated Database Management System): Veritabanlarında özellikle codasyl tarafından belirlenen geniş veritabanlarının kullanımını sağlayan sistemdir; bütünleşik bir düzene uygun olarak geliştirilmiş ilgili alt sistemlerin oluşturduğu ağdır. Orjinal verinin kopya edilmesine gerek kalmadan kaynak verilerin tek bir açıklaması kullanılır ve birbiri ile ilgili sistemler birleştirilir.

Boş Değer (Null Value): Değere sahip bulunmayan bir karakter kodudur. Sözel olarak "hiçbir şey" anlamına gelen karakterdir. Tanınabilirlik anlamında bilgisayarda yer tutması ve bir karakter olarak gönderilip alınması bağlamında gerçek olmasına rağmen, sıfır değeri ekranda ve kağıt üzerinde hiçbir şey göstermez.

Çeviri Dili (Assembly Language): Düşük düzeyli bir programlama dilidir. Burada her cümle doğrudan basit bir makina komutuna karşılık gelir. Bu dil işlemciye bağlıdır. Programcı çeviri dilinde programı yazdıktan sonra makina diline çeviren kodu kullanır. Makina üzerinde tam bir kontrol sağlar. Ancak her makina için uyumlu olmadığından makina tipine göre her programın yeniden yazılması gerekir. Denetim ve hız bakımından yüksek düzeyli dillere oranla daha iyi sonuç verir. Çünkü burada kullanılan dil ile **doğrudan** makine diliyle etkileşim sağlanır.

Çevresel Gereçler (Pheripherial Equipment): Disk sürücüsü, yazıcı, modem vb. gibi bilgisayara bağlı olup onun işletim sistemi uyarınca kullanıp denetlenen gereçler için kullanılan genel addır.

Çok Bağlantılı Liste (Multi Linked List): Ağaç veri yapısıyla aynıdır. Çünkü burada liste boyunca ileri ve geri olmak üzere iki yönde de hareket olanağı sağlayan gösterici alanına sahip düğümler vardır.

Değer (Label, Value): Programlamada ve uygulamalarda bir değişkene tahsis edilen miktardır. Değer sayısal veya sözel olabilir, bellekte saklanan belli bir **sabit veya değişken niceliktir.**

Dış Düzey/Görünümler (External Level/Views): Veritabanlarında programcı ya da kullanıcının görüş açısına göre veri yapısı; Üç düzeyli bir veri tabanı modelinde belirli bir uygulamanın gerektirdiği veri uyarınca tanımlanmış veritabanı görünümü.

Disk Yönetim Sistemi (Disc Management System): Disk sürücü veya sürücülerinin denetimi, doğru iz veya sektörün seçimi veya kaza sonucu veri kaybının önlenmesi gibi işlemleri gerçekleştiren yazılım-donanım birimi.

Doğal Dil (Natural Language): İnsanoğlunun konuştuğu dillerden herhangi birinin bir programlama ya da makina diline karşılık gelmesi, kuralları önceden belirlenen kullanımı değil, halihazırdaki kullanımı açıklayan ve yansıtan bir dildir.

Etkileşimli Arabirim (Interactive Interface): Bilgisayar alanında kullanıcı ile makina arasında konuşma tarzında sürüp giden, diyaloğu sağlayan birim.

Gösterici (Pointer): Bir dizin, bir kayıt veya ilgili kaydın içerdiği adresi gösteren veri yapısı; bellekte saklanan bir bilgi parçasının adresini içeren veri elemanı. Pek çok programlama dilinde bir veri tipi olarak tanımlanır ve özellikle liste işlemlerinde kullanılır. Verinin kendisinden çok bellekteki konumunu gösterir.

Grafik Üreticisi (Graphic Generator): Özel bir mikro işlemci olup bazı video adaptörlerine monte edilmiştir. Çizgi biçiminde veya içi doldurulmuş olarak grafik görüntüleri verir. Grafik üretimini bilgisayardan aldığı komutlar doğrultusunda ama bilgisayardan bağımsız olarak yapar.
Hazırlıksız Sorular (Adhoc queries): Kullanıcıdan gelen istek doğrultusunda ve yalnız o istek için o anda formüle edilen sorulardır.

Hiyerarşik VeriTabanı (Hierarchical Database): Kayıtların birbiri ile ilişkilendirilmesini (1.....n) düzeninde veren veritabanı olup burada

kayıtlar birbirleriyle ağaç yapısında ilişkilendirilmişlerdir. Aralarındaki ilişkinin bir ağaç ve dalları biçiminde yapılandırıldığı kayıtların topluluğudur. Genelde büyük bilgisayarlar için kullanılır. Genelden ayrıntıya doğru birbirini izler biçimde mantıksal olarak bölünebilen (veya) parçalara ayrılabilen bilginin düzenlenmesi için uygun bir modeldir.

İç Düzey/Şema (Internal Level): Üç şemalı bir mimari yapıyı benimsemiş bir veri modelinde veritabanını meydana getiren fiziksel kütüklerdeki bilginin görünümüdür. Kütük adları, konumları, erişim yöntemi ve potansiyel veri türetmeyi içerir. Bütün veritabanının en düşük düzeyde temsilidir. Çeşitli tipteki kayıtların çeşitli görünümlerini verir.

İlişki Anahtarı (Foreign Key): Bir tablodaki bir sütundur. Bu sütunun sıfır olmayan değerlerinin aynı zamanda bir başka tablonun ana anahtarında görünmesi yolu ile tablolar arası bağıntının kurulmasıdır.

İlişkisel Cebir (Relational Algebra): İlişkisel veritabanlarında ilişkilerin yönetimi için sağlanmış işlemciler kümesidir. Genelde seçim, projeksiyon, çarpım, bileşim, kesişim, ayrım ve bölme gibi işlemcileri kapsar. İlişkisel cebir aracılığı ile veritabanında varolan ilişkilerden yeni ilişkileri oluşturacak yöntemler geliştirilir.

İlişkisel Matematik (Relational Calculus): İlişkisel veritabanlarında kullanıcının verinin yönetimi için gereksinim duyulan yanıtlar kümesini belirleyen bir dildir.

İlişkisel VeriTabanı (Relational Database): İlişkiler topluluğundan oluşan veritabanı ilişkilerinin yönetimi ve yeniden konfigürasyonun veritabanı yönetim sistemi tarafından yapıldığı ve verinin yüksek düzeyde ve esneklikte kullanılabilmesinin sağlandığı veritabanı.

İşlemci (Processor): Veriler üzerine aritmetiksel, mantıksal, biçimlendirme vb. gibi işlemler yapabilme yeteneği olan yazılım sistemi veya donanım aygıtıdır.

Kapalı Dünya Varsayımı (Closed World Assumption) Söyleşi evreni içinde taslağı çizilen sistem - kullanıcı diyaloğu kapsamında olan ve veri

tabanına kaydedilmiş tüm bilgileri ifade eder. Yani sisteme dahil olan ve sistem dışında kalan bilgilerin ayrımını verir. Çünkü gerçek yaşama ait herşeyin sistemde tutulması olanaksızdır.

Kavramsal Düzey (Conceptual Level): Verilerin gerçek dünya modeline benzer bir yapıda ilişkilendirilerek söyleşi evreni sınırları içinde mantıklı bir modelin oluşturduğu düzey.

Kayıt Tipi (Record Type): Veri işlem alanında birbiriyle ilişkili veriler topluluğunun oluşturduğu birim veri yapısı.

Kütük (File): Manyetik şerit, disk, disket gibi saklama birimlerinde bulunan organize edilmiş ilişkili veriler topluluğudur. Yapısal bakımdan sıralı, doğrudan erişimli ve dizinli olmak üzere, kullanım amaçlarına göre de çalışma kütüğü, ana kütük ve yedek kütük olmak üzere gruplara ayrılır. Bir kütük veri topluluğunu içerebildiği gibi bir program da olabilir.

Kütük Yönetim Sistemi (File Management System): Kütük yönetim sistemleri programlama sistemi türlerindendir. Kullanıcı tarafından kütük karakteristiklerini ve ilişkilerini rapor, içerik ve biçimlerini belirtmek üzere kütük yükleme, rapor üretme ve güncelleştirme gibi yöntemler sağlarlar. Çok az veya hiç proğramlama deneyimi olmayan kişiler tarafından kullanılmak üzere tasarımlanmışlardır.

Makina Dili (Machine Language): Ana bellekte saklanabilen, çağrılabilen ve bilgisayar tarafından çalıştırılabilen ikili komutlardır. Makina dili en düşük düzeyli dil olup bilgiyi elektriğin açık veya kapalı durumlarına karşılık gelen 0 ve 1'ler ile yansıtır. Bilgisayar doğrudan yalnızca makina dilini kullanabilir.

Merkezi İşlem Birimi (CPU: Central Processing Unit): Bellekte yüklü bilgiyi yöneten ve bilgiye yönelik işlemleri gerçekleştiren birim. Aynı zamanda bilgiye bellekten erişim de sağlar. Bu bilgi veri ve veri yönetimini sağlayan komutlar olabilir. CPU aynı zamanda veri yönetim sonuçlarını, tekrar kullanmak üzere yine belleğe yükler. Merkezi işlem biriminin içerisinde kontrol birimi, aritmetik-mantık birimi ve kayıt birimi bulunmaktadır.

Niteleme Öğesi (Reference Scheeme): Nesneler dünyası ile değerler dünyası arasındaki köprüdür. Varlık ile varlığa ilişkin değerlerin birleşmesini sağlayan bağlantı öğeleridir.

Normalizasyon (Normalization): İlişkisel veritabanında bilginin yapılandırılması için kullanılan yaklaşımdır. Bu yaklaşım yineleme ve tutarsızlığı önlemenin yanında bilginin verimli bir biçimde depolanması ve güncelleşmesini sağlar. Birden fazla normalleştirme düzeyi olup herbiri bir öncekinin daha arınmış (rafine olmuş) halidir.

Olgu Sıklığı Sınırlaması (Occurance Frequency Constraint): Bir rolü oynayan varlık tipinin tablo da kaç kez görünmesi gerektiğini belirleyen sınırlılıktır. Bir gerçek sütunundaki değerlerin kaç kez o sütunda görünebileceğini belirler.

Özellik (Attribute): Kayıt formatı, mantıksal kayıt uzunluğu, blok büyüklüğü gibi herhangi bir kütüğün sahip olduğu özelliklerden biridir. Bilgisayar sistemlerinin pek çoğunda kütük yönetiminde gereksinim duyulduklarından dolayı nitelikler kütük isimleriyle birlikte kaydedilir; veri tabanlarında bir varlıkla ilgili bilgi içeren alan örneğin: bir personel veri tabanında ev adresi "çalışan" varlık tipinin bir özelliği'dir.

Rastgele Erişim Fonksiyonu (Hash Function): Verilerin belli bir formül çerçevesinde üretilen koda göre saklanmasını ve erişimin bu kod üzerinden doğrudan yapılabilmesini sağlayan algoritma, arama süresini azaltan ve erişimin yüksek hızla gerçekleştirildiği bir algoritmadır; bilgisayar alanında bir kütüğün kayıtlarına depolama alanı tahsis etme yöntemi. Kayıt anahtarına, yer adresi üretmek üzere uygulanan bir algoritma.

Sınırlılık (Constraint): Bilgisayar alanında, bir problemin kabul edilebilir çözümleri üzerindeki sınırlamadır. Statik ya da dinamik olmak üzere gerçek tipi popülasyonu üzerine uygulanan sınırlılıklardır. Bunlara aynı zamanda doğrulama kuralları da denir. Çünkü bu sınırlılıklar veritabanı popülasyonunun geçerli olup olmadığını saptar.

Sıra (Tuple): Birbiriyle ilişkili değerler kümesi, ilişkisel bir veritabanı tablosundaki bir sıra olarak depolanan ilişkili değerler kümesi. İlişkisel olmayan kütüklerdeki kayıt veri tipi ile aynıdır.

Söyleşi Evreni (Universe of Discourse): Makina ile kullanıcı diyaloğunun en verimli düzeyde sağlanabilmesi için oluşturulmuş karşılıklı konuşma taslağıdır. Taslağın verimli ve etkili olabilmesi için her iki tarafın da aynı dili kullanması ya da sözcüklerin her iki taraf için de aynı anlamı taşıması gerekir.

SQL (Structured Query Language): Bir veritabanı altdilidir. Sorgulama, **güncelleme** ve ilişkisel veritabanlarının yönetiminde kullanılır. IBM firmasının araştırma projesi olan SEQUEL'den türetilmiştir. 1970'lerde bir standart olarak veri tabanı ürünleri için kabul edilmiştir. Pascal ve C gibi bir programlama dili olmamasına rağmen SQL, etkileşimli soruların formule edilmesinde veya bir uygulama içine komutlar biçiminde yerleştirilerek veri işlenmesinde kullanılabilir. **uzman ve sıradan kullanıcılar** için tasarlanmıştır.

Tablo (Table): ilişkisel veritabanlarında sıra ve sütun olarak karakterize edilen bir veri yapısıdır. Programlama alanında tek anahtar ile tanımlanan ve ilişkili değerleri içeren giriş öğelerinin listesidir.

Tanım Kümesi (Domain Set): Bir (G.A.B) bağıntısı için birinci iz düşüm gönderimi altında G bağıntı çizgisinin görüntüsü, veritabanında bir varlık tipini niteleyen özelliklerin herbirinin alabileceği değer aralığını belirleyen küme.

Varlık (Entity): Veritabanlarında hakkında bilgi depolanan nesne veya olgulardır, veritabanı tasarımında hakkında veri toplanabilen ilgi odağı, nesnedir.

Veri (Data): İnsan veya makina tarafından iletişim, açıklama veya işlem amaçlarıyla konu, durum, koşul, amaç, fikir veya diğer unsarları açıklamak için tüm veya bir kısım sayıları, harfleri ve simgeleri belirtmek üzere kullanılan genel terimdir.

Veri Bağımsızlığı (Data Independence: Uygulama programlarında değişiklik yapma gereği olmadan veritabanının yapısını değiştirebilme özelliği; verinin onu yöneten programlardan ayrılması.

Veri Bütünlüğü (Data Integrity): Veri doğruluğu. Veritabanının geçerli bilgi içermesidir. Aynı gerçekle ilgili iki verinin tutarsızlığı veri doğruluğunu yok eder, veritabanı sistemlerinde veri bütünlüğünün sürdürülmesi tek tek alanların içeriklerinin doğrulanması, alan değerlerinin birbirlerine göre doğrulanması, bir kütükteki veya bir tablodaki veriler ile karşılaştırılarak doğrulanması ve veritabanının her geçiş için doğru ve başarılı bir biçimde güncellenmesi anlamına gelir.

Veri Güvenliği (Data Security): Veriler üzerinde otorite sahibi olan kişilerin işlem yapabilmesi ve verilere ancak ulaşma izni olan kişilerin ulaşabilmesi; bilerek veya bilmeden yetkin olmayan kişilerin veri aktarımı yapması, nitelemesi ya da veriye zarar vermesi gibi durumlara karşı bilgisayarda ve iletişim sistemlerinde verinin korunması yöntemlerini inceleyen bilim.

Veri Kataloğu (Data Dictionary): Bir veritabanı sistemini meydana getiren bütün veri tabanları ile igili verileri içeren veritabanı; **veriye ilişkin veriler**; çeşitli şemaları ve kütük özelliklerini ve bunların yerlerine dair bilgiyi içerir. Tam bir veri kataloğu aynı zamanda hangi programların hangi veriyi kullandığını ve hangi kullanıcıların hangi raporlarla ilgilendiğini ve gösterilen bilgiyi içerebilir.

Veri Modeli (Data Model): Veritabanı yönetim sistemi için bir sınıflama şemasıdır. Veritabanı yönetim sisteminin iki önemli öğesini yapı ve işlemleri gösterir; veritabanı yönetim sistemi tarafından desteklenen soyut varlığı oluşturan birbiriyle ilişkili nesne tipleri, bütünlük kuralları ve işlemler topluluğudur.

VeriTabanı (Database): Sınırlandırılmış bilgi gereksinimi için düzenlenmiş veriler topluluğudur; verilerin biraraya gelmesi; belirli bir tipteki alanları içeren kayıtlardan oluşan kütük ile tarama, sıralama ve yeni kombinasyonlar yaratma gibi birtakım işlemlerin bütünü.

VeriTabanı Yönetim Sistemi (Database Management System): Veritabanının oluşturulması, düzenlenmesi, depolama, erişim, güvenlik gibi birtakım işlevleri olan bir yazılım paketidir.

Veri Tanımlama Dili (Data Definition Language DML): Veri tabanı yapısı ile veritabanı yönetim sistemi arasında iletişimi sağlamak amacıyla kullanılan bir dildir; veritabanı yönetim sisteminin bir parçası olup tüm özelliklerini tanımlamada kullanılır. Özellikle kayıtların fiziksel uzunlukları, alan tanımlamaları, anahtar tanımlamaları, kütük konumları ve depolama stratejisi gibi.

Veri Yönetim Dili (Data Manipulation Language): Veri tabanındaki verinin yönetimini gerçekleştiren dildir; veri tabanlarında verinin veri tabanı ile uygulama programı arasında transferini yöneten ve bir programcı tarafından kullanılan dildir; Genelde bir veri tabanı yönetim sisteminin bir parçası olan bir dildir. Veri girişi, güncelleme, sorgulama işlemlerinde kullanılır. DML genelde rapor üretimi ve matematiksel, istatistiki hesaplamaları da içeren özelliklere sahip olabilir.

Yalın Gerçek (Elementary Fact): Bir varlık ya da o varlığın oynadığı rol, ya da sahip olduğu özelliktir. **Temel** nitelemesi gerçeğin daha küçük parçalara bölünemeyeceği anlamına gelir.

Yöntemsel Dil (Procedural Language): Bazen veri yönetim dili karmaşık raporları üretebilmek için yöntemsel programlama dili içerir. Bu dil, bir problemin çözüm işleminin nasıl yapılacağını açıklayan, bilgisayarlardan bağımsız bir dildir; bir probleme ilişkin çözümün, herbiri yöntemsel olarak ifade edilen adımlar ve bunların belli sırada çağrılmasıyla gerçekleştirilmesini gerektiren programlama dili. (Pascal, algol, cobol ve fortran gibi)

Yüksek Düzeyli Programlama Dili (High Level Programming Language): Herbir komutunun veya durum ve komut cümlesinin makina dilindeki birçok komuta karşı geldiği bir dildir. Yüksek düzeyli diller kullanıcılara alışık oldukları bir tarzda yazma olanağı sağlarlar.

ÖZET

Enformasyon teknolojisine paralel olarak gelişen bütünleşik sistemler ve sorgulama dilleri kuruluşlarına yeni ufuklar açtılar. İşlemlerin yürütülmesinden yönetime ve karar düzeyine kadar her aşamada ve her konuda insanlara büyük olanaklar sundular. Bilgi merkezleri de bu gelişimden etkilenen kuruluşlar arasında yeraldı. Geleneksel anlamda bibliyografik denetim ve erişim etkinliğini temel anlan anlayış veritabanı yönetim sistemleri ile değişime uğradı. Özellikle kapsamlı, karmaşık bilgi bileşimlerini kullanıcıya sunabilen ilişkisel modele dayalı veritabanı yönetim sistemleri ve modelin oluşumunda kullanılan yeni teknikler gündeme geldi. Sistem oluşumunda tasarımın önemi anlaşıldı. Bu bağlamda bilgi merkezi ve bilgi uzmanı kavramlarını yeniden gözden geçirmek gerekti. Bu çizgisel gelişim doğrultusunda çalışmada veritabanı yönetim sistemi yapı, işleyiş, tarihsel gelişim açısından ele alındı. Sistemin mantıksal oluşumunu gerçekleştiren tasarım teknikleri incelenerek bu tekniklerden biri bir kütüphane işlemine uyarlandı. Görevi bilgi ile kullanıcı arasında köprü kurmak olan bilgi uzmanının tasarım aşamasında nedenli önemli bir rol üstlendiği dolaylı yoldan bu çalışma ile ortaya kondu.

SUMMARY

Integrated systems and query languages which developed parallel to information technology have provided a wide horizon for the organizations. They offered comprehensive facilities at a variety of levels which include executing operations, management, and decision steps. Information centers, also, took place among those organizations which were effected by this improvement. In traditional sense, understanding based on bibliographical control and retrieval has changed with the database management systems. Especially, database management systems based on relational model offered user a comprehensive, complex and combination of existing information and new techniques concerning semantic data modelling currently became available in design procedures. Soon it is accepted that system design is quite important. Besides this, information center and information specilist, concepts were to be revieved once more.

According to this increasing curve of advancement database management system is examined in structural, operational and historical aspects. Design techniques which involve logical formation of system is excimined and one of these techniques is adapted to one of library operations. The information specilist whose main responsibility is to establish a mutual connection between user and information has taken over a very important role at design stage and this is proved indirectly in this work.