YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TARİHİ YERLEŞİMLER İÇİN NESNEYE YÖNELİK VERİ TABANI VE VERİ MADENCİLİĞİ YÖNTEMLERİ KULLANARAK BİLGİ SİSTEMİ OLUŞTURULMASI VE UYGULAMASI

Ayşe BUHARALI

FBE Bilgisayar Mühendisliği Anabilim Dalında
Bilgisayar Mühendisliği Programında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd.Doç.Dr. A.Gökhan YAVUZ

İSTANBUL, 2004

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

ADT	Abstract Data Type
API	Application Programming Interface
CBS	Coğrafi Bilgi Sistemi
CRM	Customer Relationship Management
GIS	Geographic Information System
HIS	Historical Information System
OID	Object Identifier
OLAP	Online Analytical Processing
ORDBMS	Object Relational Database Management System
PMML	Predictive Model Markup Language
SQL	Structured Query Language
UDF	User Defined Function
UDT	User Defined Type
UML	Unified Modeling Language

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

İş dünyasında sadece rekabet değil gün geçtikçe müşterilere ait veriler de yığınlar halinde artmaktadır. Rakiplerin gerisinde kalmadan müşteri memnuniyetini arttırmak ve maliyetleri azaltmak amacıyla her müşterinin her hareketini izleyip kaydetmek zamanla içinden çıkılmaz veri yığınları oluşturmuştur. İşte bu yığınlardan karar alma süreçlerinde faydalanabilmek amacıyla anlamlı bilgi elde etmek için 1990'ların başında veri madenciliği teknikleri kullanılmaya başlanmıştır. İş dünyasında oldukça verimli sonuçların alındığı veri madenciliği üzerine çalışmalar günümüzde halen devam etmektedir.

Bu kadar faydalı olduğu düşünülen veri madenciliği, yüzyıllar boyunca çok farklı kültürlere ev sahipliği yapmış ülkemizde tarihi yapıların korunmasında kullanılamaz mı? Bu sorunun cevabı olumlu olmasına karşın araştırmalarımız sonucu bugüne kadar bu alanda hiçbir çalışmanın yapılmadığı ortaya çıkmıştır.

Araştırmalar sonucunda ayrıca ilişkisel veri tabanı yapısının daha önce yapılan çalışmalarda yetersiz kaldığı ve mimari alanda kullanılan GIS programlarının da nesneye yönelik veri tabanı alt yapısına sahip olduğu gözlenmiştir. Bu nedenlerden dolayı kültürel çevrelerdeki yapıların bilgileri nesneye yönelik veri tabanı yapısında tutulmaya karar verilmiştir.

Bu çalışmanın tez yürütücüsü sayın Yrd.Doç.Dr.A.Gökhan Yavuz'a yardımı ve değerli fikirleri için teşekkür ederim. Ayrıca bu çalışma boyunca bana yardımcı olan araştırma görevlisi arkadaşım Z.Cihan Tayşi'ye ve bilgisayar mühendisliğini bana sevdiren ve bu yolculukta bana hep rehberlik edip tanıştığımız günden itibaren her konuda destek olan sevgili hocam Prof.Dr.Oya Kalıpsız'a da teşekkürlerimi sunarım.

Sadece bu tez çalışmasında değil hayatımın her aşamasında aldığım kararlarda beni destekleyen ve sıkıntılı zamanlarımda kaprislerime katlanan annemin, babamın ve kardeşimin bana olan güvenini boşa çıkarmamak ümidiyle kendilerine çok teşekkür ederim.

ÖZET

Tarihi yerleşim envanterinin oluşturulmasına ilişkin gerçekleştirilen çalışmalarda toplanan verilerin çeşitliliği ve sayılarının fazlalığı bu verilerin dijital ortamda saklanması ve işlenmesindeki temel problemi oluşturur. Özellikle farklı dönemlerde birçok uygarlığa ev sahipliği yapan bölgelerde günümüze kalan mimari eserler, gerek yapı türleri gerekse yapısal özellikleri açısından büyük farklılıklar göstermektedirler. Dolayısıyla yapılan çalışmalarda ya mimari üslup açısından farklılık gösteren her bölge için ayrı bir veri tabanı yapısı oluşturulmakta ya da o bölgeye özgü mimari özelliklerin kaybolmasına göz yumarak daha önceden oluşturulan bir veri tabanı yapısına uygun olarak veriler saklanmaktadır.

Veri girişinin standartlaştırılmasındaki zorunluluk ilişkisel veri tabanı yapısının kısıtlarından kaynaklanmaktadır. Ancak daha önce de belirtildiği gibi mimari eserlerin çok çeşitli olması sonucu oluşturulan ilişkisel veri tabanları yeterli gelmemektedir. Halbuki son yıllarda bilgi teknolojisinde oldukça yaygın hale gelen nesneye yönelik yaklaşım, veri tabanı yapısına da yansımış ve ilişkisel yapının kısıtlarını ortadan kaldırmasına olanak sağlayan nesneye yönelik veri tabanı geliştirilmiştir. Bu çalışmada da nesneye yönelik veri tabanı yapısının özellikleri incelenerek mimari bilgilerin saklanmasındaki uygulama önerileri ortaya konacaktır.

Tarihi eser verilerinin türlerinin fazlalığı kadar sayılarının fazla oluşu da dijital ortamda saklanması açısından önem taşımaktadır. Bir veri ambarı niteliğindeki veri tabanlarında sadece tarihi yapıların bilgilerini tutmak karar verme aşamasında kullanıcıyı zorlayabilir. Çünkü mimari eserleri tarihten soyutlayarak incelemek mümkün değildir. O dönemin sosyal ve siyasal durumu, günlük hayat, bölgede meydana gelen savaş, doğal afetler, hastalıklar vb. gibi toplumlar üzerinde köklü etkileri olan olaylar araştırmacının varacağı sonucu etkilemektedir. Bütün bu bilgilerin de veri tabanında tutulacağı göz önüne alınırsa veri tabanının boyutu kolayca tahmin edebilir. Böyle geniş bir bilgi okyanusunda kullanıcının tek başına sonuçlara varması oldukça zorlaşmaktadır. Bu nedenle kültürel miras çalışmalarında iş hayatında özellikle de pazarlamada oldukça faydası olan veri madenciliği kullanılabilir. İşte bu çalışmada da veri madenciliğinin kültürel miras çalışmalarındaki yeri ve sağlayabileceği faydalar üzerinde durulacaktır.

Anahtar kelimeler: Veri Madenciliği, Tarihi Yerleşim, Nesneye Yönelik Veri Tabanı

ABSTRACT

The variety and the number of data that are collecting during the historical environment studies, cause the main problem of saving and processing of those data on digital media. Especially the historical architectures of the regions which hosted many civilizations at different periods indicate great differences at their building kinds and structured attributes. That's why a new database has been developed for every region. Otherwise some architectural attributes which are special to that region can be lost if that region's data are stored in the database which had been developed before for another region study.

The necessity of data entering standardization is the result of relational database constraints. Relational database isn't sufficient for the variety of architectural buildings. This problem can be solved by using object oriented database on the historical sites studies. The object oriented database application's offers are made for saving the architectural information on digital media at this study.

The enormous number of historical buildings is important as much as their variety at saving them on digital media. Saving only the historical buildings' attributes on the databases isn't sufficient for end users at the decision stage because it isn't possible to analyze the architectures of one region without the history of that region. The social and political lives, the wars, the disasters, the contagious diseases and other events, that influence the society, have an important fact at researcher's decisions. The size of such a database that collects all historical events' data will be enormous so it will be difficult for a researcher to analyze all these data without computer help. That's why a data mining method can be used at historical heritage studies. The place and benefits of data mining at the historical heritage studies are described at this study.

Keywords: Data Mining, Historical Heritage, Object Oriented Database

1. GİRİŞ

Tarihi yapıların korunmasına yönelik sürdürülen çalışmalarda bilgi sistemlerinin sağlayacağı katkı ve fayda düşünülerek bugüne kadar çeşitli uygulamalar geliştirilmiştir. Sadece tarihi değil tüm yapılara ait bilgilerle beraber dijital haritalar üzerinde kolayca çalışmasını sağlamak amacıyla geliştirilen Coğrafi Bilgi Sistemi - CBS (Geographic Information System - GIS) uygulamaları bu konuda önemli bir adımdır.

İlk geliştirilen CBS'ler ilişkisel veri tabanı modeline dayanmaktadır. Sistem geliştirilme sürecinde ve veri tabanı yönetiminde nesneye yönelik yaklaşımın yaygınlaşmasından CBS'ler de etkilenmiş ve yeni uygulamalar nesneye yönelik veri modeline göre geliştirilmeye başlanmıştır.

Coğrafi bilgi sistemi uygulamalarının, hem kullanımının bu alanda çalışanlara göre karmaşık olması hem de geniş veri tabanları üzerinde bilgi keşfi sürecinin zaman alması bakımından tam olarak kullanıcı ihtiyacını karşılayamamaktadır. Bu nedenlerle CBS uygulamaları ile uyumlu çalışan ilişkisel veri tabanı modeline dayalı çeşitli bilgi sistemleri geliştirilmiştir. Ancak ilişkisel veri modelinden kaynaklanan kısıtlar ve nesneye yönelik CBS uygulamalarının kullanımının artmasıyla yeni arayışlar başlamıştır. İşte bu doğrultuda bu çalışmada kullanıcının ihtiyaçlarını tam olarak karşılayacak tarihi yapılar için bilgi sistemi olan HIS (Historical Information System) geliştirilmiştir.

Nesneye yönelik veri tabanı modeline göre geliştirilen HIS'in üzerinde veri madenciliği uygulanarak bilgi keşfi sürecini kolaylaştırılmaya çalışılmıştır. HIS, tarihi yapılar alanında oluşturulan nesneye yönelik veri tabanı modeli olması ve bu modelde veri madenciliğinin uygulanması açılarından yeni bir bakış açısı getirmektedir. Özellikle iş dünyasında kullanılan veri madenciliği, HIS'de tarihi yapıların birbirleriyle ve tarihi olaylarla ilişkisini ortaya koymak için kullanılmıştır.

Bu çalışmada kullanılan nesne ilişkisel veri modelini daha iyi anlamak amacıyla birinci bölümde hiyerarşik modelden başlayan veri tabanı gelişim süreci anlatılmaktadır. İkinci bölümde ise yine çalışmada uygulanan veri madenciliğinin tanımı, gelişim süreci ve veri madenciliği sürecinde gerçekleştirilen adımlarda kullanılan metotların ve tekniklerin genel tanımı yapılmaktadır. Çalışmanın son bölümü olan üçüncü bölümde HIS'in tanımı ve alt yapısı anlatılmaktadır. Ayrıca HIS geliştirilirken kullanılan araçlar ve karşılaşılan problemlere önerilen çözümler yer almaktadır.

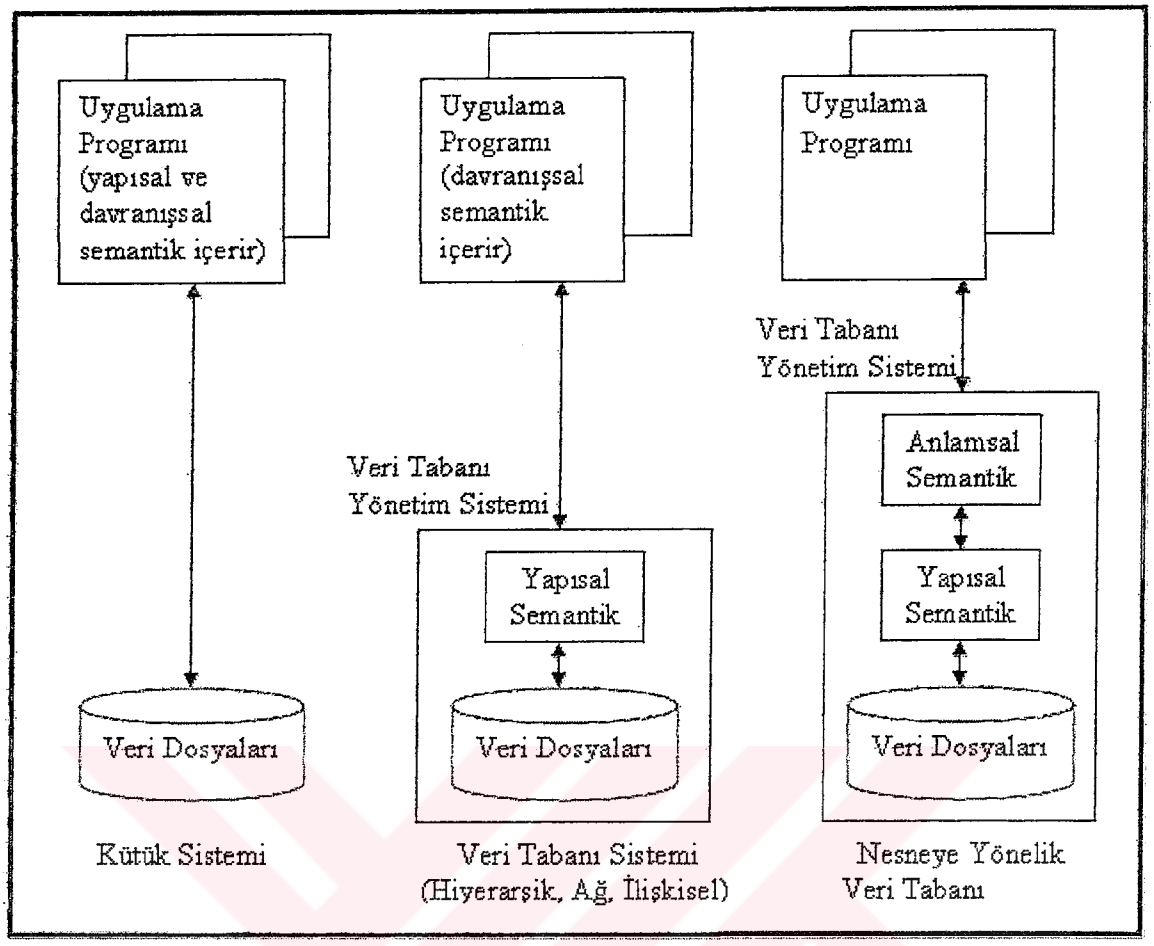
2. VERİ TABANI YAPILARI

Veri tabanı uygulamaları iş dünyasının hayat kanıdır (Blaha,1998). Her gelişmiş organizasyonun veri tabanı kendine özgüdür. Bu veri tabanlarına işin yapısına göre çeşitli kayıtlar yapılıp, işlemler takip edilir. Ayrıca sadece iş dünyasında değil insan geninin araştırılması gibi konularda da veri tabanlarına rastlamak mümkündür.

Veri tabanı uygulamalarında fazla miktardaki veriye farklı yollarla ulaşabilme imkanı sağlanırken birden fazla kullanıcının ve uygulamanın da eş zamanlı olarak çalışabilmesi koordine edilmelidir. Bu yaklaşımla veri tabanı sistemlerinin en önemli üç karakteristik özelliğini aşağıdaki şekilde özetlenebilir.

1. Fazla miktardaki verinin organizasyonunu sağlayıp en verimli şekilde bilgi haline dönüştürmek. Böylece verinin yeniden kullanımını ve güncelleştirilmesini sağlamak.
2. Veriler, farklı kullanıcılar tarafından paylaşımını sağlamak.
3. Bu amaçla veri paylaşımı sırasında ortaya çıkabilecek sorunları çözerek verinin doğruluğunu ve bütünlüğünü sağlamak.

Veri tabanının ilk kullanımından itibaren artan veri miktarıyla beraber kullanıcı ihtiyaçlarını da karşılamak üzere zaman içinde veri tabanı yapıları ve uygulamaları da gelişmiştir. Şekil 2.1'de de görüldüğü veri tabanı yapıları gelişmeden önce veriler kütükler halinde tutulmaktaydı. Ancak bu yapıda bütün iş, uygulama kısmında yapılmaktaydı. Kütüklerde yapılan en küçük bir değişiklik bile uygulama programının yeniden düzenlenmesinin gerektirmektedir. 1970'lerde ortaya çıkan veri tabanı sistemleri, sayesinde yapısal kısımda meydana gelen değişikliklerin uygulamayı etkilemesi önlenmiştir. Ancak daha ileride de anlatılacağı üzere bu veri tabanı modellerin kısıtlarından ötürü gerçek dünya nesnelerinin davranışsal modellemesi veri tabanı yönetimi tarafında değil uygulamada gerçekleştirilmektedir. Bu sorun, 1990'larda kullanılmaya başlanan nesneye yönelik veri tabanı sistemleri ile aşılmıştır.



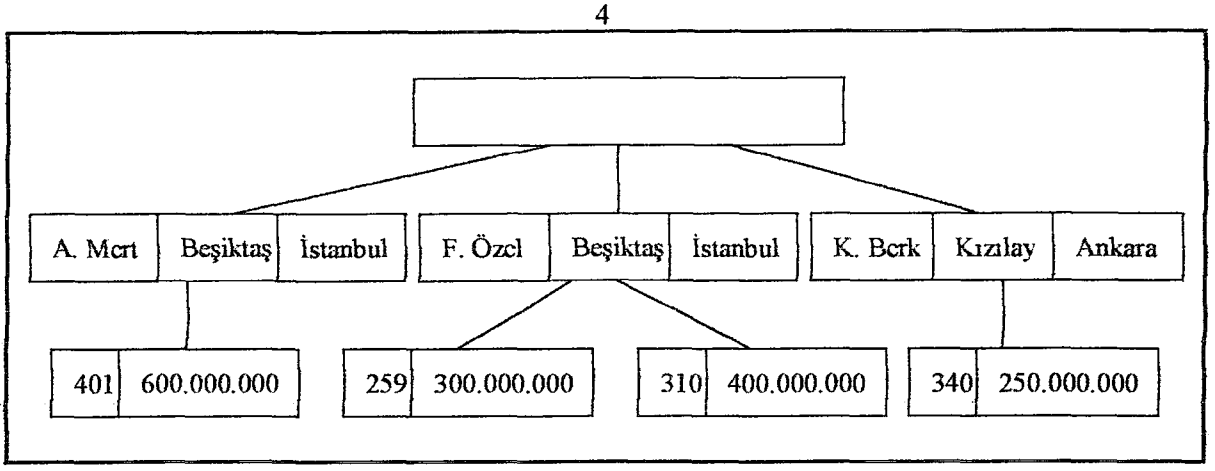
Şekil 2.1 Veri modellerinin gelişimi (Eaglestone ve Ridley, 1998)

Veri tabanı modelleri gelişim yıllarına göre şu sırada özetlenebilir. [1]

1. Hiyerarşik Veri Tabanı Modeli
2. Ağ Veri Tabanı Modeli
3. İlişkisel Veri Tabanı Modeli
4. Nesneye Yönelik Veri Tabanı Modeli

2.1 Hiyerarşik Veri Tabanı Modeli

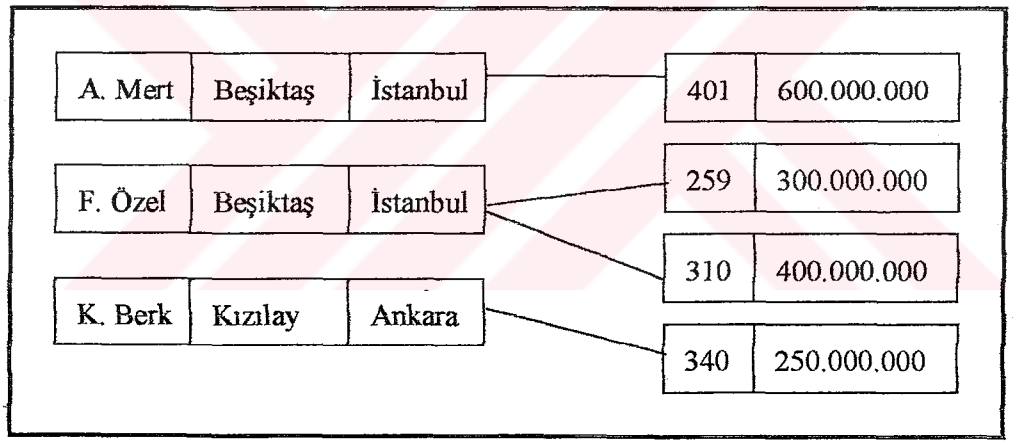
IBM'in Bilgi Yönetimi Sistemi (Information Management System - IMS) tarafından 1960'larda geliştirilen hiyerarşik model 1970'lere kadar kullanılmıştır. Şekil 2.2'de görüldüğü gibi hiyerarşik modelde kayıtlar, ağaç yapıları oluşturacak şekilde birbirlerine bağlanırlar.



Şekil 2.2 Hiyerarşik model (Kalıpsız, 1998)

2.2 Ağ Veri Tabanı Modeli

Ağ veri tabanı modelinin popüler olduğu zamanlar hiyerarşik modelle aynı zamana denk gelmektedir. Veri, her çocuk için birden fazla üst sınıf için modellenmektedir. Böylece ağ model çoka çok ilişkinin tanımlanmasına izin vermektedir. Ağ modelde kayıtlar, birbirlerine ağ yapıları içinde bağlanırlar (Şekil 2.3).



Şekil 2.3 Ağ model (Kalıpsız, 1998)

Hiyerarşik ve ağ modeller, kayıtları yerleştirme, okuma ve yazma için gerekli işlemlere sahiptirler. Ancak ilk nesil veri tabanı teknolojisi olarak adlandırılan bu modellerin bazı kısıtları vardır: (Eaglestone ve Ridley, 1998)

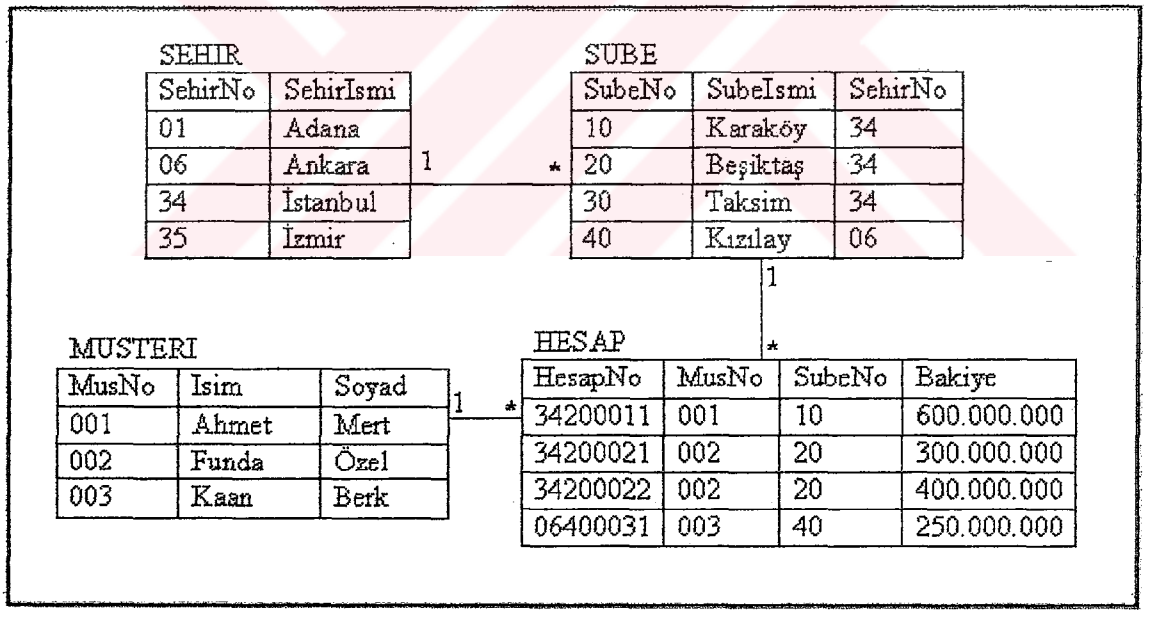
1. Verinin mantıksal yapısı ile fiziki uygulaması arasında belirli ayrımın olmaması sonucu programlarla veri tabanı yönetimi arasındaki ara birimler oldukça karmaşıktır. Ayrıca kayıtların birbirlerine nasıl bağlanacağı da genelde program kodunun içinde belirlenmektedir.

2. Veri tabanı dilleri bir seferde sadece bir kaydı kullanabilmektedirler. Bu nedenle veri tabanındaki kayıtlar arasında gezinebilmek için veri tabanı dili komutlarının yerleştirildiği programlama dilinin kullanılması gerekmektedir.
3. Hiyerarşik ve ağ modellerde teorik temel olmadığı için araştırmaya engel teşkil etmektedirler.

2.3 İlişkisel Veri Tabanı Modeli

1970'de geliştirilen ilişkisel modelin kullanımı 1985'den sonra artmıştır. İlişkisel veri tabanı yönetimi günümüzde de en yaygın kullanıma ve ürün yelpazesine sahiptir.

İlişkisel veri tabanı modelinde veriler tablolarda saklanır (Şekil 2.4). Bu tablolar üzerinde SQL (Structured Query Language) yardımıyla gerçekleştirilen işlemlerle istenilen bilgiye ulaşılmaktadır. Veri tabanında modellenmek istenen her gerçek nesne, ilişkisel modelde varlık (entity) ile tanımlanır. İlişkisel modelde varlığın nitelikleri (attributes) nesnenin özelliklerini temsil ederken varlıklar arasındaki birliktelikler ise ilişkiler (relationships) ile ifade edilmektedir.



Şekil 2.4 İlişkisel model

İlişkisel modelin gücü hakim olduğu kesin matematiksel ifadeden kaynaklanır. İlişkisel veri tabanındaki tablo, matematiksel ilişkiyi temsil ederken tabloları kullanan işlemler ise ilişkilere dayanan matematiksel işlemlere karşılık gelmektedirler. Böylece ilk kez veri tabanı sistemi,

ilişkisel modelle teorik bir temele dayandırılmıştır (Eaglestone ve Ridley, 1998). Bu özelliğinden ötürü ilişkisel model, veri tabanı teknolojisinin gelişmesinde önemli bir adım olmuş ve ikinci nesil veri tabanı teknoloji sürecini başlatmıştır.

İlişkisel modelin getirdiği diğer bir yenilik ise basitliktir. İlişkisel modelde bütün bilgiler basit mantıksal bir yapı olan tabloyla ifade edilebilmektedir. Tüm bilgiler tablolar sayesinde veri değerleri olarak kolaylıkla görülebilmektedir. Bu sayede veri tabanı dilleri, sütunlar ve satırlar üzerinde basit kes, kopyala ve yapıştır işlemlerine dayanmaktadır (Eaglestone ve Ridley, 1998).

Veri modelinin getirdiği yeniliklerinin yanı sıra iki önemli kısıdı bulunmaktadır (Eaglestone ve Ridley, 1998):

1. **Yapısal Kısıt :** Her ne kadar tüm bilgi, atomik değerler tablolarda kolayca ifade edilse de gerçek dünyada bulunan karmaşık yapıları tablolarla ifade etmek oldukça zordur. Karmaşık yapıları, ilişkisel modele oturturken birbirine ortak alanlarla bağlı birden fazla tablo kullanılır. Bunun sonucunda da uygulama sırasında bir varlık hakkında bilgi alabilmek için birden fazla tablodan verinin toplanıp düzenlenmesi gerekmektedir.
2. **Davranışsal Kısıt :** İlişkisel veri tabanı sadece yapısal anlamı destekler ama davranışsal anlamı desteklemez. Gerçek hayattaki nesnelerin sahip oldukları fonksiyonlar, veri tabanı yapısı içinde değil uygulamada gerçekleştirilir.

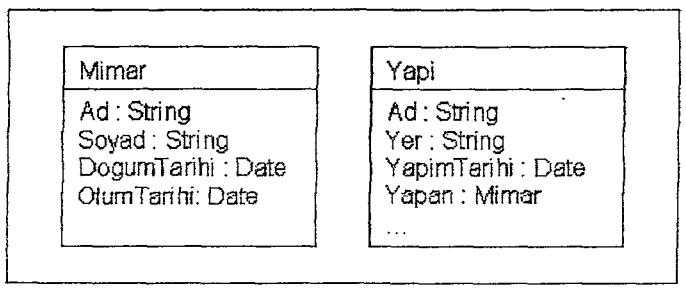
2.4 Nesneye Yönelik Veri Tabanı Modeli

Son on yılda ortaya çıkan nesneye yönelik kavramının kullanımı da hızla artmaktadır. Ancak ilk iki modelden (hiyerarşik ve ağ) farklı olarak nesneye yönelik modelin geliştirilmesi ilişkisel modelin gelişimini etkilememiştir. Halbuki ilişkisel modelin kullanımın yaygınlaşması sonucu hiyerarşik ve ağ modeller kullanılmaz olmuştur.

Nesneye yönelik modelde hareket noktası gerçek dünyadaki nesnedir. Gerçek dünyadaki nesne hiçbir değişikliği uğratılmadan aynen modellenir. Çünkü veri tabanındaki nesne de aynı gerçek dünyada olduğu gibi hem duruma (state) hem de davranışa (behaviour) sahiptir. Aynı özelliklere (attributes) ve metotlara (methods) sahip olan nesneler bir sınıfa (class) aittir. Dolayısıyla her nesne bir sınıfın örneğidir (instance).

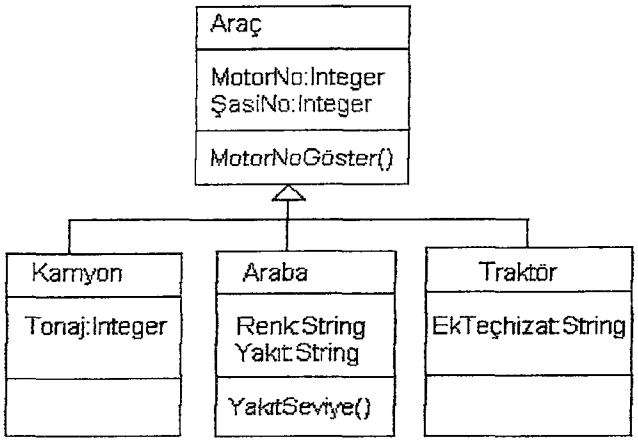
Bir sınıfa ait özelik, basit bir veri tipi olabileceği gibi, başka bir sınıf da olabilir. Basit veri

tipi, nesneye yönelik yazılım ortamı tarafından desteklenen bir veri tipidir. Şekil 2.5'deki mimar sınıfı basit veri tiplerine sahipken, yapı sınıfının bir özelliği mimar sınıf yapısındadır. Bu özellik kullanılarak bir nesne farklı nesneler tarafından da referans gösterebilmektedir. Böylece karmaşık özellikler, karmaşık nesneler (complex objects) yardımıyla tutulabilmektedir. Çalışmanın ileri bölümlerinde sınıflar arasındaki ilişkilerin nasıl gerçeklşetirildiği anlatılmaktadır.



Şekil 2.5 Sınıf özellikleri

Şekil 2.6'da da görüldüğü gibi nesneler sınıf hiyerarşisine (hierarchy) göre düzenlenirler. Aynı durum ve davranışa sahip nesneler daha üst seviyede bulunan üst sınıf (superclass) içinde gruplanırlar. Daha sonra genel durum ve/veya davranıştan farklı bir özelliğe sahip olan sınıflar özelleştirilerek alt sınıflar (subclass) oluşturulur. Kalıtım (inheritance) ile bir alt sınıf, kendine has özelliklerin yanı sıra hiyerarşik olarak kendi üstünde olan bütün sınıfların özelliklerini de taşımaktadır. Sınıflara ait özellikler alt sınıflardan üst sınıfa doğru gidildikçe genelleşir, üst sınıftan alt sınıflara doğru gidildikçe özelleşir. Bu nedenle bu ilişki tipine genelleştirme-özelleştirme (generalisation-specialisation) veya o'dur (is_a) adı verilir (Eaglestone ve Ridley, 1998). Örneğin Şekil 2.6'daki modele göre "Araba bir araçtır.", "Araç, arabanın genelleştirilmiş halidir." veya "Araba, aracın özelleştirilmiş halidir" şeklinde tanımlanabilir.



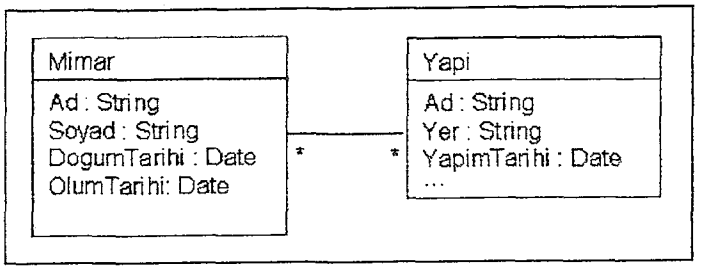
Şekil 2.6 Sınıf hiyerarşisi

Genelleştirme-özelleştirme ilişkisinde çoklu kalıtımla (multiple inheritance) bir alt sınıf, birden fazla üst sınıfa sahip olabilir. Buradaki en önemli sorun üst sınıfların aynı isme sahip farklı özellikleri varsa alt sınıfın nasıl davranacağıdır. Nesne veri modeli, basit olarak bu sorunun oluşmasına izin vermez. Dolayısıyla isim karmaşıklığı, veri tabanı tasarımcısının sorumluluğundadır (Eaglestone ve Ridley, 1998).

İlişkisel veri tabanındaki varlık ile nesneye yönelik veri tabanındaki nesne arasındaki en büyük fark sarmadır (encapsulation). Nesne, bu özellikle hem veri yapısını hem de metotları içermektedir.

Nesne sınıfları arasındaki ilişkiler, ilişkisel veri tabanı modelindeki gibi birebir, bireçok ve çoka çok olabilmektedir. Ancak bu ilişkiler ikincil anahtar ile değil daha ileride anlatılan referans gösterimiyle sağlanmaktadır.

En çok kullanılan ilişki türlerindeki gösterimler, "**0..1**", "**0..***", "**1..1**", "**1..***", "***" şeklindedir. "0", alt limitin seçimlik olduğunu gösterir. "1" alt limiti ise, sınıflar arasında en az bir tane bağlantı olması gerektiğini gösterir. Üst limit olarak "1" ise, iki sınıf arasında en fazla bir bağlantıya izin verileceğini göstermektedir. Birden fazla bağlantı için ise, üst limitin belli olmadığı anlamında "*" işareti kullanılır. Örneğin Şekil 2.7'deki Mimar sınıfı ile Yapı sınıfı arasında çoka çok ilişkisi vardır. Çünkü bir mimarın birde fazla yapısı, bir yapının da birden fazla mimarı olabilir.



Şekil 2.7 Sınıflar arası bağıntı

Nerneye yönelik veri tabanı modelinin getirdiği yeni yaklaşımlardan biri de sınıflar arasındaki parça-bütün ilişkisidir. İlişkinin gücüne göre **bileşim** (composition) ve **bütünleşme** (aggregation) olarak ikiye ayrılan parça-bütün ilişkisinde bütünü temsil eden bir sınıfla bu sınıfın bileşenleri temsil eden sınıf (ya da sınıflar) bulunur. Bu ilişkilerin daha ayrıntılı açıklaması, UML'de (Unified Modelling Language) sınıf kavramı altında anlatılmaktadır.

Nesneye yönelik modelin avantajları:

- **Veri Saklama :** Nesnenin iç yapısı kullanıcılardan saklanır. Böylece bir nesnenin istemcileri, sadece nesnenin sunduğu genel (public) servislerden faydalanabilirler, nesneye ait özel (private) servisler veya özellikler sadece nesne tarafından kullanılabilinir (Brown, 1991).
- **Modülarite :** Nesneye yönelik yaklaşım yapısı gereği birbiriyle çok iyi hareket eden ama birbirinden bağımsız nesnelerin tasarlanmasını desteklemektedir (Brown, 1991).
- **Yeniden Kullanım :** Sınıfların hiyerarşik yapıda oluşturulması sonucu bir sınıfta tanımlanan özelliklerin ve metotların bu sınıfın alt sınıflarında yeniden oluşturulmasına gerek yoktur. Çünkü kalıtım yoluyla üst sınıftaki genel durum değişkenleri ve metotları alt sınıflara aktarılmaktadır (Brown, 1991).
- **Hızlı Dolaşım :** Nesneler arasındaki kalıtım özelliği sayesinde dolaşım (navigational) sorgulamalar hafızadaki veriye ulaşım kadar hızlı gerçekleşmektedir. İlişkisel sistemlerde bir nesneden başka bir nesneye geçiş uygulama sırasında daha yavaş gerçekleşmektedir. [2]

Nesneye yönelik modelin dezavantajları ve kısıtları:

- **Düşük Verimlilik :** Veri tabanında modellenen sistemin verileri ve ilişkileri basit ise nesneye yönelik modelin kullanılması ilişkisel modele göre verimi daha düşük olacaktır. [3]
- **Standartlaşmama :** İlişkisel veri tabanı yönetiminin standartları nesneye yönelik veri

tabanı yönetime göre daha durağandır. [3]

- **Karmaşıklık :** İlişkisel modelin tasarımı ve kullanılan tablolar daha basit olduğu için veri tabanı yönetimi daha basittir. Nesneye yönelik modelde ise sistemi geliştirenlerin daha fazla teknik bilgiye ve deneyime ihtiyacı vardır.
- **Teorik Temel Eksikliği :** Nesneye yönelik sistemlerde matematiksel temel yoktur. Bunun sonucu olarak da sistemin analizinde, kavranmasında ve sonuç çıkarma aşamalarında kayıplar meydana gelebilmektedir. [2]

2.4.1 Nesne İlişkisel Veri Tabanı Modeli

1999 yılında geliştirilen nesne ilişkisel veri tabanı modeli nesneye yönelik özelliklerle ilişkisel modelin birleşimidir. Nesne ilişkisel veri tabanı modeli, ilişkisel veri tabanı modelinin basitliğiyle nesneye yönelik veri tabanı modelinin sunduğu nesne özelliklerini taşımaktadır.

Nesne ilişkisel model; nesneye yöneliğin sunduğu sarma (encapsulation), çok çeşitlilik (polymorphism) ve kalıtım özelliklerini geleneksel ilişkisel veri tabanı modelinin içine katmıştır. Nesne ilişkisel model; SQL'i desteklediği için ilişkisel, karmaşık veriyi desteklediği için nesneye yöneliktir (Chaudhri ve Zicari, 2001). Daha öncede belirtildiği üzere nesneye yönelik veri tabanı modeli, karmaşık yapıların oluşturulmasına olanak sağlarken beraberinde sorgulama karmaşıklığını da neden olmaktadır. Halbuki ilişkisel modelin sunduğu basitlik, sistemin geliştirilmesinde vazgeçilmez bir unsurdur. İşte bu açıdan bakıldığında nesne ilişkisel veri tabanı modeli, gelecekte daha çok tercih edileceği muhtemeldir. Bu üç veri tabanı modelin çeşitli açılardan karşılaştırılması Çizelge 2.1'de yer almaktadır. Bu tabloda adı geçen ve nesne ilişkisel veri tabanı model tarafından desteklenen SQL99 (SQL3) bir sonraki bölümde açıklanmaktadır.

Çizelge 2.1 Veri tabanı yönetim sistemlerinin karşılaştırılması [4]

Özellik	İlişkisel Veri Tabanı Yönetim Sistemi	Nesne İlişkisel Veri Tabanı Yönetim Sistemi	Nesneye Yönelik Veri Tabanı Yönetim Sistemi
Kimlik (identity)	Birincil anahtar	Birincil anahtar, Kimlik	Nesne kimliği
Nesne Referansı	Yabancı anahtar	Yabancı anahtar, Referans (REF)	Nesne kimliği
Veri Tipleri	Standart	(extensible) Standart	Herhangi biri
Tip Yapıları	SET, ortogonal olmayan TUPPLE	SET, TUPLE, orthogonal olmayan ARRAY	Hepsi
Şema Tanımı	SQL	SQL99	Java, C++, (Smalltalk), ODL
Veri Kullanımı	SQL	SQL99	Java, C++, (Smalltalk)
Sorgulama Dili	SQL	SQL99	OQL (Object Query Language)
Kalıtım	Yok	Basit	Basit ve birden fazla
Soyut Veri Tipi (abstract data type – ADT)	Yok	Mümkün	Var

2.4.2 Nesne İlişkisel Veri Tabanı Modeli ve SQL99

İlk kez ilişkisel veri tabanı geliştirildikten sonra veri tabanına erişebilmek için geliştirilen SQL, hiçbir firma tarafından sahiplenilmeyen açık bir standarttır (Kauffman, 2001). SQL, ANSI (the American National Standards Institute) tarafından tanımlanan ve yönetilen veri tabanı dilidir. ANSI-SQL Grubu zaman içinde üç tane SQL standardı yayınlamıştır:

- SQL89 (SQL1)
- SQL92 (SQL2)
- SQL99(SQL3)

Üç versiyon boyunca SQL'de çok büyük bir değişiklik olduğu söylenemez. Nesneye yönelik özellikleri ve OLAP'ı destekleyen SQL99, SQL92'nin bir üst versiyonudur. Nesne ilişkisel modelin tanımlandığı ANSI/ISO SQL99 standardı genel olarak aşağıdaki bölümlerden

oluşmaktadır (Elmasri ve Navathe, 2000) :

- SQL/Framework : genel giriş; dokümanların yapısı
- SQL/Foundation : standardın tanımı
- SQL/CLI (Call Level Interface) : çağrı seviyesi ara birimi
- SQL/PSM (Persistent Stored Modules) : sürekli saklı modüller
- SQL/Bindings: gömülü ve dinamik SQL

SQL99, yapısal tipler ve tablolar için basit kalıtımı desteklemektedir; Alt tip, üst tipin özelliklerini ve davranışını kalıtım yoluyla alır. Yapısal tablonun her satırı bir nesne örneğidir ve diğerlerinden **nesne tanımlayıcı** (object identifier – OID) alanı aracılığıyla farklılık gösterir. Tabloya yeni bir nesne kaydı girildiği zaman OID değeri sistem tarafından otomatik olarak üretilir. Bu kolonun tipi referans (reference - REF) tipidir. Bu nedenle her yapısal tablonun mutlaka OID değeri taşıyan bir kolonu olmalıdır. Her nesne tipi için farklı olan REF tipi, tipin kendisi değil aslında tiplerin yaratıcı metodudur (constructor). Referans tipi olarak tanımlanan özellik, referans edilen nesnenin OID'sidir. Böylece sınıflar arasındaki ilişkiler, REF tipi sayesinde ikincil anahtar (foreign key) kullanılmadan tanımlanmış olur (Marcos vd., 2001).

SQL99'da iki tip nesne vardır: (Elmasri ve Navathe, 2000)

- Satır tipi (row type)
- Soyut veri tipi (abstract data type – ADT)

Satır tipinde, tablolarda örnekler "tuple" şeklinde saklanır. Bu tipin tanımlanması için aşağıdaki SQL cümlesi sentaksı kullanılır.

CREATE ROW TYPE** <satır_tipi_ismi> **(** <bileşenlerin tanımı> **)

Bu tipe ait kayıtların tutulacağı tablonun SQL cümlesi sentaksı ise

***CREATE TABLE** <tablo_ismi> **OF TYPE** < satır_tipi_ismi>*

şeklindedir. Bu nesne tipi, karmaşık nesnelerin oluşturulmasına ve nesnenin işlevselliğini desteklemesine rağmen sarmayı (encapsulation) desteklemez.

ADT ile kullanıcı, davranışsal özelliklerini ve iç yapısını birlikte tanımlayabileceği tipler oluşturabilir. Ayrıca "UNDER" komutuyla nesne kalıtımı sağlanırken üst sınıfta tanımlanan bir fonksiyonun alt sınıfta yeniden tanımlanmasına (overloading) sağlar. Hiyerarşik yapıda en

baştaki sınıfı oluşturmak için genel olarak aşağıdaki SQL sentaksı kullanılır.

CREATE TYPE <tip_ismi> **AS**
(<bileşenlerin tanımı>,
<eşit (equal) ve daha az (less than) fonksiyonlarının tanımı>,
< fonksiyonların metotların tanımı>)

Bu sınıfın altında yer alan alt sınıflara ait tip tanımı ise

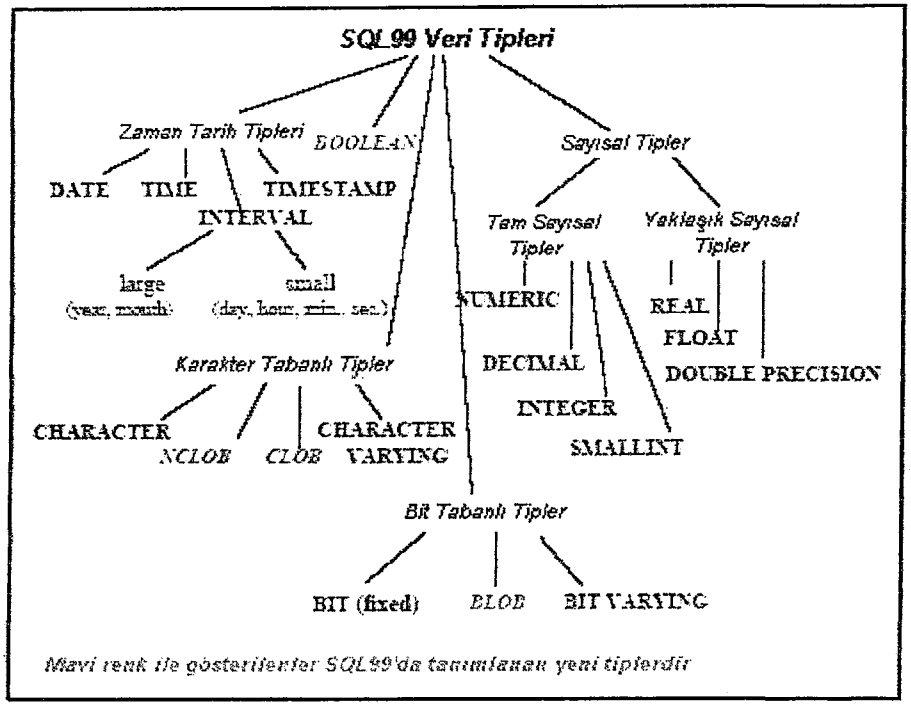
CREATE TYPE <altsinif_tipin_ismi> UNDER <üstsinif_tipin_ismi> AS
(<bileşenlerin tanımı>,
<eşit (equal) ve daha az (less than) fonksiyonlarının tanımı>,
< fonksiyonların metotların tanımı>)

şeklinde yapılır.

Bu çalışmanın ileri bölümlerinde anlatılan rekursif sorgulamayı destekleyen doğrusal özyineleme (linear recursion) üçüncü kuşak SQL'in getirdiği en önemli işlevlerden biridir.

SQL99, dağınık veri tabanları için **istemci modülü** (client module) anlayışını getirmiştir. İstemci modülü, SQL99 sentaksı ile tanımlabilen geçici tabloları oluşturup harici prosedürlere başvurulmasını sağlar.

SQL99'un getirdiği diğer bir yenilik de değişken tipleridir. SQL99, Şekil 2.8'de gösterilen geniş nesnelerin (large objects – LOB) kullanımını veri tabanı yönetimi sistemiyle harici dosyaları kullanmadan desteklenmesini sağlar. [4] Nesne ilişkisel model, sadece bu değişken tipleriyle sınırlı değildir; kullanıcı, veri tabanı yapısı içinde yeni değişken tiplere de tanımlayabilmektedir. SQL99, kullanıcı tanımlı veri tiplerinin (user defined data type – UDT), ayrık veri tiplerinin (district data type), toplama tiplerinin (collection type), kullanıcı tanımlı fonksiyonlara (user defined function – UDF) ve metotlara olanak sağlamaktadır. Böylece teorideki karmaşık nesne mantığı, uygulamaya geçirilmiş olmaktadır.



Şekil 2.8 SQL99 veri tipleri [4]

3. VERİ MADENCİLİĞİ

Büyük miktarlarda ve oldukça hızlı toplanan verilerin çeşitli analizler sonucunda anlamlı bilgilere dönüştürülmesi noktasında "veri madenciliği" süreci devreye girmektedir.

Veri madenciliği, veri ambarlarında tutulan çok çeşitli verilere dayanarak daha önce keşfedilmemiş bilgileri ortaya çıkarmak, bunları karar vermek ve eylem planını gerçekleştirmek için kullanma sürecidir.[5]

Gartner Group tarafından yapılan tanıma göre ise veri madenciliği, istatistik ve matematik tekniklerle birlikte örüntü tanıma (pattern recognition) teknolojilerini kullanarak, depolama ortamlarında saklanmış bulunan veri yığınlarının elenmesi ile anlamlı yeni korelasyon, örüntü ve eğilimlerinin keşfedilmesi sürecidir (Akpınar, 2000).

Veri madenciliği tanımları incelendiğinde, bu tanımlardan ortak olan unsurlardan ilki "çok fazla" miktarlarda verinin veri ambarlarında tutulması ikincisi ise bu verilerden "anlamlı" bilgiler elde edilmesidir.

3.1 Veri Madenciliğinin Gelişim Süreci

Çizelge 3.1'de de görüldüğü gibi daha fazla veriyi saklamak ve iş analistlerinin sorularına cevap vermek için veri tabanlarının kullanımının artmasının doğal sonucu olarak veri madenciliği de gelişmiştir. [6]

Geleneksel sorgu ve rapor araçları veri tabanında tam olarak ne olduğunu çıkarmak ve anlatmak için kullanılırlar. Kullanıcı, ilişki üzerine bir hipotez geliştirip veriler üzerinde yapılan sorgulamalarla bunu doğrulamaya çalışır. Örneğin analist, düşük gelirli ve yüksek borca sahip müşteriler kredi açısından risklidir hipotezini geliştirebilir. Bu aşamadan sonra analist, hipotezinin doğru olup olmadığını görmek için veri tabanı üzerinde çeşitli sorgulamalar yapar. Veri madenciliği ise bu hipotezin geliştirilmesinde kullanılabilir. Örneğin analist, aradığı modeli yapay sinir ağlarını kullanarak bulabilir. Ancak bunu gerçekleştirirken 30 yaşının üstünde düşük gelirli, borcu olan ama ev sahibi ve çocuklu müşteriler kredi açısından risksizdir hipotezini kendisi oluşturmak zorunda değildir.

Çizelge 3.1 Veri madenciliğinin gelişim süreci [6]

Gelişim Adımı	**İş Sorusu**	**Kullanılan Teknoloji**
Veri Toplama (1960'lar)	"Son beş yıl içindeki toplam gelirim ne kadar?"	Bilgisayarlar, teypler, diskler
Veri Erişimi (1980'ler)	"Geçen Mart ayında İngiltere'de gerçekleştirilen satışlar nelerdir?"	Daha fazla kapasiteye sahip daha hızlı ve ucuz bilgisayarlar, ilişkisel veri tabanları
Veri Ambarı ve Karar Destek	"Geçen Mart ayında İngiltere'de gerçekleştirilen satışlar nelerdir? Drill down to Boston"	Daha fazla kapasiteye sahip daha hızlı ve ucuz bilgisayarlar, OLAP, çok boyutlu veritabanları, veri ambarları
Veri Madenciliği	"Gelecek ay Boston'daki satışlar nasıl olacak? Neden?"	Daha fazla kapasiteye sahip daha hızlı ve ucuz bilgisayarlar, geliştirilmiş bilgisayar algoritmaları

Hızla artan rekabet koşullarıyla beraber birebir pazarlamaya geçiş süreci sonucunda farklı sektörlerde veri madenciliği uygulamalarının önemi daha artmıştır. Özellikle müşteri ilişkileri yönetimi (Customer Relationship Management – CRM) uygulamalarının temelinde veri madenciliği yer almaktadır. Veri madenciliği, sadece CRM'de değil maliyetleri azaltmak amacıyla risk yönetimi ve sahtecilik tespiti gibi alanlarda da kullanılmaktadır. Çizelge 3.2'nin dikey kolonunda veri madenciliğinin başlıca uygulandığı sektörler, yatay kolonununda da bu sektörlerde yer alan veri madenciliği uygulamalarının kullanıldığı alanlar yer almaktadır. Türkiye'de bu alanlarda veri madenciliği hizmeti sunan başlıca firmalar ve bu firmaların kullandığı uygulamalar EK 1'de tablo halinde özetlenmiştir.

Çizelge 3.2 Veri madenciliği uygulama alanları

Dikey	**Yatay**
• İnsan kaynakları • Finansal hizmetleri • Kamu • Perakende • Sigortacılık • Telekomunikasyon • Tıp • Ulaştırma	• Müşteri ilişkileri yönetimi • Müşteri kaybını önleme • Risk yönetimi • Pazar araştırması • Sahtecilik tespiti • Web analizleri

3.2 Veri Madenciliği Modelleri

Veri yığınının yapısına ve alınmak istenen sonuca göre veri madenciliğinde çeşitli modeller kullanılmaktadır. Bir modelin diğer modellere göre avantajları ve dezavantajları düşünülerek en uygun modelin ve bu modelin geliştirilmesinde kullanılacak tekniğin kararına sistem analisti karar vermelidir. Nitekim veri madenciliği tanımlarından da anlaşılacağı üzere veri madenciliği, kendi başına bir çözüm değil problemlerin tanımı için gerekli olan bilgiyi sağlayarak karar verme sürecini destekleyen bir araçtır.

Veri madenciliği modelleri, tahmini ve açıklayıcı modeller olarak iki grup altında toplamak mümkündür. Örneğin tarihi yapı kayıtları üzerine yapılan tahmini modelde

- *Deniz kenarında yosunlaşmanın yapılara ne kadar zararı olabilir?*
- *Bu yerleşim yerinde kale yerleşimi nasıl olabilir?*

benzeri sorulara cevap aranırken; açıklayıcı modellemede

- *Yapı plan tipi yerleşim yerine göre değişmektedir.*
- *Ticari yapıların yerleşiminde satılan ürünün tipi ve değeri önemli faktörlerdir.*

şeklinde eldeki kayıtlara üzerine açıklama yapılır (Buharalı, 2002).

Veri madenciliğinde kullanılan başlıca modeller aşağıda kısaca anlatılmaktadır.

Sınıflandırma, eldeki nesnelerin, konuların ve problemlerin ortak özelliklerine göre kategorize edilmesidir. Sınıflandırmada yeni sunulan verilerin özelliklerini inceleyerek bu verileri daha önceden belirlenmiş sınıflardan birine dahil edilir (Berry ve Linoff, 1997).

Sınıflandırmada kullanılan başlıca teknikler,[6]

- Karar Ağaçları (Decision Tree),
- Yapay Sinir Ağları (Artificial Neural Networks)
- Genetik Algoritmalar (Genetic Algorithms)
- En Yakın K Komşu (K-Nearest Neighbour)

Bu tekniklere ilişkin açıklamalar, "Veri Madenciliğinde Kullanılan Teknikler" bölümünde kısaca anlatılmaktadır.

Kümeleme, bir tür sınıflandırma olmasına rağmen kümeleme modelini sınıflandırmadan ayıran en belirgin özellik önceden belirlenmiş sınıfların olmamasıdır. Bu model uygulanmadan önce veri tabanındaki kayıtların hangi kümelere ayrılacağı veya kümelemenin hangi değişkenlere göre yapılacağı bilinmemektedir. [6]

Kümeleme analizinde kullanılan algoritmalar oldukça fazladır. Analistin probleme yaklaşımı ve eldeki veri tiplerine göre en uygun algoritma seçilmelidir. Kümeleme algoritmalarının çoğu iki görüşe dayanmaktadır: (Kantardzic, 2001)

- Hiyerarşik kümeleme
- İteratif kare-hata bölümlendirme kümeleme

Hiyerarşik kümelemede veriler ağaç yapısında ya da dendogram şeklinde gösterilecek şekilde organize edilir. İkinci algoritmada ise küme içindeki aralığı azaltacak kümeler arası aralığı arttıracak bölümleri bulmaktır.

K adet küme oluşturmak için kullanılan kümeleme metotları temel olarak üç gruba ayrılır; (Fayyad ve Stolorz, 1997)

1. **Ölçüt-Mesafe Tabanlı Metotlar :** Kullanılacak mesafe ölçütü belirlenir. Bu ölçüte göre her blokta birbirine en yakın (veya merkeze en yakın) durumlar tespit edilerek en iyi K bölüme ayrılmaya çalışılır.
2. **Model Tabanlı Metotlar :** Her küme için bir model hipotezi kurularak her kümeye en uygun model bulunmaya çalışılır. Örneğin k kümesi için M_k modeli geliştirildiyse modelin uygunluğu

$$\textbf{Prob}\,(M_k|\,D) = \textbf{Prob}(D|\,M_k)\,\frac{\textbf{Prob}\,(M_k)}{\textbf{Prob}(D)}$$

formülüyle hesaplanır. D verilerine ait önceden hesaplanmış olasılığı **Prob**(D), sabittir

ancak yapılan karşılaştırmanın amacına göre hesaba katılmayabilir. **Prob** (M_k) ise model için önceden hesaplanmış olasılık değeridir. Bu formülle olasılığı en yüksek olan model kabul edilir.

3. **Bölme Tabanlı Metotlar :** Temel olarak farklı bölümler numaralandırılır ve daha sonra belli kriterlere göre bu bölümler puanlanırlar. Yukarıda anlatılan diğer iki metot bu metodun özel durumları olarak değerlendirilebilinir. Yapay zekada kullanılan birçok teknik bu kategori altında değerlendirilir ve ad hoc puanlama fonksiyonlarından faydalanılır.

Regresyon, mevcut değerleri kullanarak diğer değerlerin ne olacağını tahmin etmeye çalışır. Regresyon metodunda genelde doğrusal regresyon gibi standart istatistik yöntemleri kullanılır. Ancak birçok problem önceki değerlere bağlı olarak doğrusal modellemeyle anlatılabilinecek kadar basit değildir. Bu nedenle daha mantıksal regresyon, karar ağaçları veya yapay ağlar gibi daha karmaşık teknikler de kullanılmaktadır. Aynı model tipleri, hem regresyon hem de sınıflandırma için kullanılabilirler. Örneğin CART (Classification and Regression Trees) karar ağacı hem sınıflandırma ağaçları (kategoriksel cevap değişkenlerini sınıflandırmak için kullanılırlar) hem de regresyon ağaçları (sürekli cevap değişkenlerinin tahmininde kullanılırlar) için kullanılabilinir.[7]

Doküman Madenciliği

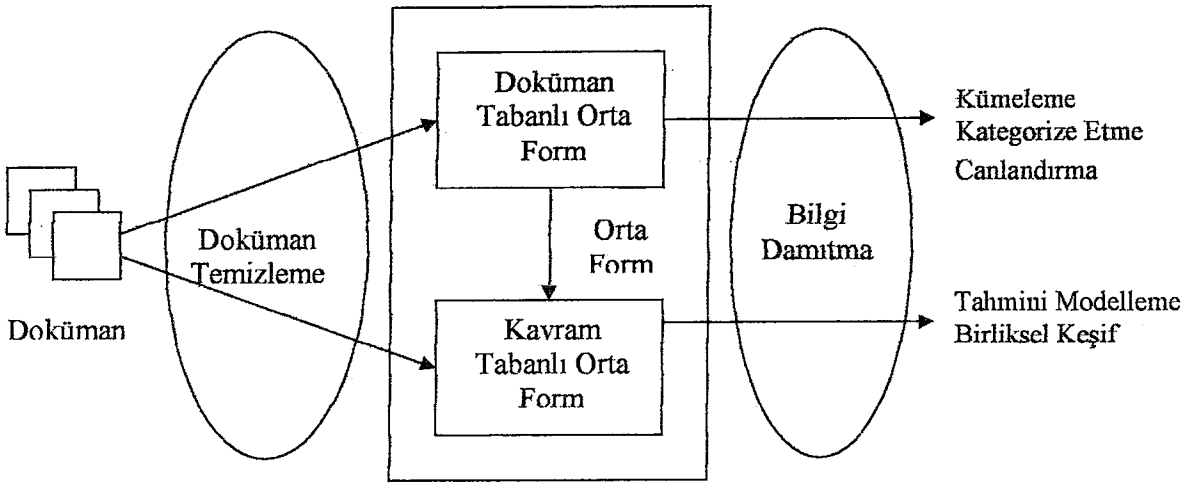
Kurumlarda saklanan dokümanların sayısı her geçen gün daha da artmaktadır. Bu hızlı artışın başlıca nedeni ise bilginin elektronik formda saklanmasının yaygınlaşmasıdır. Örnek olarak e-postaları, dijital kütüphaneleri ve web sayfalarını verebiliriz. Çoğu doküman veri tabanında tutulan bilginin yarı yapısal (semistructured) olması nedeniyle geniş metin verilerden yeni bilgiler keşfetmek amacıyla doküman madenciliği adı verilen özel bir teknik kullanılmaktadır (Kantardzic, 2001).

Doküman analizindeki başlıca amaçlar;

1. Çok geniş doküman koleksiyonun içeriğine genel bir bakış sağlayarak en verimli şekilde organize edilmesini sağlamak.
2. Dokümanlar ve doküman grupları arasındaki gizli yapıları belirlemek.
3. Benzer ya da ilişkisel bilginin bulunmasında verimliliği ve etkinliği arttırmak.
4. Arşivde tekrarlanan bilgi veya dokümanları saptamak.

Doküman madenciliği süreci Şekil 3.1'de grafiksel olarak tanımlanmıştır. İlk aşama olan

doküman temizlemede serbest formdaki dokümanlar seçilmiş orta forma (intermediate form – IF) dönüştürülür. Bilgi damıtma aşamasında ise orta formdan bilgi ve örüntüler çıkarılır. (Kantardzic, 2001)



Şekil 3.1 Doküman madenciliği çerçevesi (Kantardzic, 2001)

Orta form, kavramsal grafik gibi yarı yapısal olabileceği gibi ilişkisel veri gibi yapısal da olabilir. Farklı madencilik amaçlarına göre orta formun karmaşıklık derecesi değişkenlik göstermektedir. Orta düzeydeki formlar, doküman tabanlı ve kavram tabanlı olarak iki grupta toplanırlar. Doküman tabanlı orta formda yapılan madencilik çalışmasıyla dokümanlar arasındaki ilişkiler ve örüntüler belirlenir.

3.3 Veri Madenciliğinde Kullanılan Teknikler

Karar ağaçları, eldeki veriden oluşturulduktan sonra ağaç kökten yaprağa doğru inilerek kurallar (IF-THEN rules) yazılabilir. Bu şekilde kural çıkarma (rule extraction), veri madenciliği çalışmasının sonucunun geçerlenmesini sağlar (Alpaydın, 2000).

1980'lerden sonra yaygınlaşan yapay sinir ağlarında (artificial neural networks) amaç fonksiyon birbirine bağlı basit işlemci ünitelerinden oluşan bir ağ üzerine dağıtılmıştır. Yapay sinir ağlarında kullanılan öğrenme algoritmaları veriden üniteler arasındaki bağlantı ağırlıklarını hesaplar (Alpaydın, 2000).

Geleneksel küme kavramına göre herhangi bir nesne, bir kümeye ya aittir ya da ait değildir. Bulanık mantıkta ise nesnelerin kümeye aitliği derecelere bölünmektedir; herhangi bir

nesnenin bulanık kümeye aitliği 0.5'tir veya 0.8'dir veya 0.25'tir gibi deyimlerle sıkça karşılaşılmaktadır. Böylece nesnelerin bulanık kümeye aitliği üyelik dereceleri ile belirlenir ve bu dereceler de üyelik fonksiyonları ile karakterize edilir.

1975'te biyoloji alanında doğal genetik sistemlerin simulasyonunda kullanılmaya başlanan genetik algoritmalar, veri madenciliğinde çözümü zor olan optimizasyon problemlerinde kullanılmaktadır (Kantardzic, 2001). Genetik algoritmada başlangıçta birçok iterasyon arasından bir model seçilir ve daha sonra yeni modeller oluşturmak için mevcut modeller birleştirilir. Bu modeller arasından en iyisi bir sonraki iterasyonda girdi olacak şekilde uygunluk işlevine (fitness function) göre seçilir. Genetik algoritmada tüm alan aranmadığı için en iyi sonuç elde edilmeyebilir ama zor sorunlara uygun çözümler üretebilir.

Kümelemede oldukça sık kullanılan yöntemlerden biri olan en yakın k komşu algoritmasında örneklerle mevcut kümeler arasındaki farklar hesaplanır. Bu algoritmada yeni bir örneğin daha önceden kümelere yerleştirilmiş olan örneklerle arasındaki mesafeler hesaplanır. Artan şekilde sıralanan mesafe değerlerinden en küçük değere sahip K adet örnek seçilir. Oylama prensibi uygulanarak yeni örnek, içinde en fazla K adet seçilmiş örneğe yakın olan sınıfa eklenir (Kantardzic, 2001). Çizelge 3.3'de veri madenciliğinde kullanılan teknikler, sağladıkları avantajlara ve kısıtlarına göre karşılaştırılması yer almaktadır.

Çizelge 3.3 Veri madenciliği tekniklerinin karşılaştırılması (Brooks, 1997)

Teknoloji	Avantajlar	Kısıtlar
Kural Tabanlı Analiz	Kurallar okunabilir. Veriler tam ise "if ... then" kuralları veya karar ağaçları ile veri ilişkileri kolayca modellenebilir.	Çok sayıda kural varsa anlaşılması zordur. Verilerin güçlü kural tabanlı ilişkileri olmayabilir.
Yapay Sinir Ağları	Doğrusal olmayan ilişkiye sahip veriler için uygundur. Eksik veri değeri varsa bile çok iyi çalışır.	Sayısal olmayan verilerin sayısal veri değerlerine dönüştürülmesi gerekir. Her ne kadar son zamanlarda bazı araçlar alınan kararları açıklamaya çalışsa da bulunan ilişki açıklanamamaktadır.
Bulanık Mantık	Elde edilen sonuçları, istenen sonuca göre sıralayabilir.	Uygulama ve satıcı sayısının az olması
En Yakın K Komşu	Kümelerin belirlenmesinde başarılıdır.	Çok geniş hafızaya ihtiyaç duyar. Kayıtların eşleştirilmesinde duyarlılık düzeyinin fazla olması.
Genetik Algoritmalar	Doğrusal olmayan ilişkilere sahip verileri hesaba katarak tahmin problemlerinde başarılıdır. Eksik veri değeri varsa bile çok iyi çalışır.	Sayısal olmayan verilerin sayısal veri değerlerine dönüştürülmesi gerekir. Her ne kadar son zamanlarda bazı araçlar alınan kararları açıklamaya çalışsa da bulunan ilişki açıklanamamaktadır.
Geliştirilmiş Canlandırma	Yüksek sofistike üç boyutlu canlandırmalarda ilişkilerin bulunmasında kullanılır.	Diğer metotlara göre daha fazla kullanıcıya daha çok ihtiyaç vardır.

3.4 Bilgi Keşfi Süreci

Veri madenciliği kullanılsın ya da kullanılmasın bilgi keşfi süreci, problemin tanımıyla başlar. Tanımlanan probleme göre oluşturulan hipotez için gerekli veriler toplanıp derlenerek model oluşturulur. Bu model kullanarak üretilen hipotezin geçerliliği izlenir.Veri madenciliği, bu sürecin model oluşturma safhasında katılır.

Veri madenciliği olmadan yapılan bilgi keşfinde ise tüm sorumluluk analiste aittir. Analist, oluşturulan her hipotezi test eder, model oluşturur, tekrar test eder ve bu süreç iteratif şekilde devam eder. Görüldüğü üzere veri madenciliği sayesinde; (Buharalı, 2002)

- En uygun modeli bulmak için analist tarafından harcanan iş yükü azalmıştır.
- Daha az iş yüküyle çok daha verimli modeller oluşturulabilmektedir.

- Daha az zaman harcanarak daha fazla model geliştirilebilmektedir.
- Süreç, adım adım sistem tarafından otomatik olarak gerçekleştirildiği için analistin eskisi gibi çok fazla teknik bilgiye ihtiyacı yoktur.

Her ne kadar veri madenciliği, bilgi keşfini kolaylaştırdıysa da veri madenciliği aşamasına gelene kadar olan adımlar elde edilen bilginin kalitesi için önemlidir. Bunun için bilgi keşfindeki tüm aşamalar aşağıda kısaca anlatılmaktadır.

3.4.1 Problemin Tanımlanması

Veri madenciliği çalışmalarında başarılı olmanın ilk şartı, uygulamanın hangi amaç için yapılacağının açık bir şekilde tanımlanmasıdır. Amaç, problemin üzerine odaklanmış, açık bir dille ifade edilmiş ve elde edilecek sonuçların başarı düzeylerinin de nasıl ölçülebileceği tanımlanmalıdır. [6]

3.4.2 Verilerin Hazırlanması

Modelin kurulması aşamasında ortaya çıkacak sorunlar, bu aşamaya sık sık geri dönülmesine ve verilerin yeniden düzenlenmesine neden olmaktadır. Bu durum verilerin hazırlanması ve modelin kurulması aşamaları için, bir analistin bilgi keşfi sürecinin toplamı içerisinde enerji ve zamanının %50 - %85'ini harcamasına neden olmaktadır. [6]

Verilerin toplanmasıyla başlayan bu aşamada verinin doğrulanması, birleştirilmesi, temizlenmesi, seçilmesi ve son olarak modelde kullanılacak şekilde dönüştürülmesi gerekmektedir.

Veri toplanırken tanımlanan problemin çözümünde bilgi sağlayacak nitelikte olmasına dikkat edilmelidir. Dolayısıyla verinin toplanacağı kaynakların ve bu kaynakların güvenirliğinin önceden belirlenmesi ileriki aşamalarda ortaya çıkabilecek problemleri azaltacaktır.

Veri madenciliği çalışması sonucunda elde edilen bilgilerin doğruluğu ve güvenirliği büyük ölçüde verinin ne kadar sağlıklı olduğuna bağlıdır. Bu nedenle birden fazla kaynaktan toplanan verilerin birbiriyle uyumu sağlanmalıdır. Veri uyuşmazlığı, daha çok farklı veri giriş biçimlerinden ve farklı ölçü birimlerinden kaynaklanmaktadır. Dolayısıyla veri doğrulama aşamasında toplanan verilerin birbiriyle ne kadar uyumlu olduğu ortaya konmalıdır.

Veriler arasındaki uyumsuzluklar ve uyumsuzluk nedenleri belirlendikten sonra bu sorunlar mümkün olduğu ölçüde giderilerek veriler ortak bir veri tabanında toplanır. Sadece veri uyumsuzluğu değil kayıtlardaki eksik değerler de (missing values) gözden geçirilerek

tamamlanmaya çalışılmalıdır.

Veri toplanırken yanlış girilen veri değerleri, sonuçlarda yanlış sapmaya neden olabilir. Bu nedenle, aykırı değerler (outliers) dikkate alınarak analiz dışı tutulması durumunda sonuca olan etkisi hesaplanmalıdır. Bu hesaplama sonucunda uygun görüldüğü takdirde aykırı değerler veri tabanından silinmelidir.

Veri seçimi, kullanılacak modele bağlı olarak veri seçimi yapılır. Örneğin tahmin edici bir model için, bu aşamada bağımlı ve bağımsız değişkenlerin ve modelin eğitiminde kullanılacak veri kümesinin seçilmesi anlamını taşımaktadır. Veri ambarı modelleme bölümünde anlatılacağı üzere operasyonel veri tipleri analize katılmaz. Böylece diğer değişkenlerin modeldeki ağırlığının azalması engellenmiş olur.

Modelde kullanılan veri tabanının çok büyük olması durumunda tesadüfîliği bozmayacak şekilde örnekleme yapılması uygun olabilir. Günümüzde hesaplama olanakları ne kadar gelişmiş olursa olsun, çok büyük veri tabanları üzerinde çok sayıda modelin denenmesi zaman kısıdı nedeniyle mümkün olamamaktadır. Bu nedenle tüm veri tabanını kullanarak birkaç model denemek yerine, tesadüfî olarak örneklemmiş bir veri tabanı parçası üzerinde birçok modelin denenmesi ve bunlar arasında en güvenilir ve güçlü modelin seçilmesi daha uygun olacaktır.[6] Zaten günümüzde kullanılan veri madenciliği yazılımları da bu veri özetleme işini otomatik olarak kendileri yapmaktadır.

Verinin kalitesinden ve tamamlanmasından kaynaklanan veri analizindeki kısıtları azaltabilmek için son aşama olan veri dönüştürmede çeşitli teknikler kullanılmaktadır. Bu tekniklerin başında veri filtreleme, sıralama, redaksiyon ve gürültü modelleme gelmektedir (Famili vd.,1997). Veri filtrelemede, sapmış ve gürültülü veriler gibi istenmeyen veriler temizlenir. Veri sıralamada amaç tekrar kullanma ve analiz için tablolardaki verilerin düzenlenmesidir. Veri redaksiyonu, doküman veya sembolik veri tiplerinde özel bir nitelik için tekil bilgiyi temsil eden bir veya birden fazla karakterin önişlemesinde kullanılır. Son olarak gürültü modellemede ise Fourier transformu, Bayesian, maksimum benzerlik, korelasyon ve koveryans vb. yöntemlerden faydalanarak verinin toplanması sırasında dış etkenlerden kaynaklanan veride oluşan gürültü ortadan kaldırılmaya çalışılır.

3.4.3 Modelin Kurulması

Tanımlanan problem için en uygun modelin bulunabilmesi, mümkün olduğu kadar çok sayıda modelin kurularak denenmesi ile mümkündür. Bu nedenle öğrenme kümesi üzerinde L

değişik teknik kullanılarak L tane model oluşturulur. Sonra bu L model deneme kümesi üzerinde denenerek en başarılı olanı, yani deneme kümesi üzerindeki tahmin başarısı en yüksek olanı seçilir (Alpaydın, 2000).

Modelin kurulması, kullanılacak olan öğrenimin denetimli (supervised) veya denetimsiz (unsupervised) olmasına göre farklılık gösterir. Denetimli öğrenimde, sınıflar önceden belirlenen kriterlere göre ayrılarak, her sınıf için çeşitli örnekler verilir. Sistemin amacı verilen örneklere göre her bir sınıfa ilişkin özelliklerin bulunması ve bu özelliklerin kural cümleleri ile ifade edilmesidir. Denetimsiz öğrenimde ise kümeleme analizinde olduğu gibi ilgili örneklerin özelliklerini inceleyerek aralarındaki benzerliklere göre sınıfların tanımlanması amaçlanmaktadır. [6]

Modeli geliştirirken eldeki veriler öğrenme ve deneme verisi olarak ikiye ayrılır. Oluşturulan L tane model üzerinde öğrenme verileriyle çalışılarak L tane sonuç alınarak sistemin öğrenmesi sağlanır. Daha sonra öğrenen model üzerinde deneme verileri çalıştırılır. Alınan sonuç, hangi modelde istenen sonuca en yakınsa o model seçilir. Alınan hiçbir sonucun istenen sonuca yakın olmaması durumunda en başa dönülerek olası modeller yeniden kurulur. Bu süreç en iyi model bulunana kadar devam eder.

3.5 Veri Madenciliğinde Karşılaşılan Problemler

Veri madenciliği girdi olarak ham veriyi sağlamak üzere veri tabanlarına dayanır. Bu da veri tabanlarının dinamik, eksiksiz, geniş ve net veri içermemesi durumunda sorunlar doğurur. Diğer sorunlar da verinin konu ile uyumsuzluğundan doğabilir (Vahaplar ve İnceoğlu, 2001).

Sınıflandırmak gerekirse başlıca sorunlar şunlardır :

- **Sınırlı Bilgi :** Veri tabanları genel olarak veri madenciliği dışındaki amaçlar için tasarlanmışlardır. Bu yüzden, öğrenme görevini kolaylaştıracak bazı özellikler bulunmayabilir.
- **Gürültü ve Eksik Değerler :** Veri özellikleri ya da sınıflarındaki hatalara gürültü adı verilir. Veri tabanlarındaki eksik bilgi ve bu yanlışlardan dolayı veri madenciliği amacına tam olarak ulaşmayabilir. Bu bilgi yanlışlığı, ölçüm hatalarından ya da öznel yaklaşımdan olabilir.
- **Belirsizlik :** Yanlışlıkların şiddeti ve verideki gürültünün derecesi ile ilgilidir. Veri tahmini bir keşif sisteminde önemli bir husustur.

- **Güncelleme :** Veri tabanlarındaki bilgiler, veri eklendikçe ya da silindikçe değişebilir. Veri madenciliği perspektifinden bakıldığında, kuralların hala aynı kalıp kalmadığı ve istikrarlılığı problemi ortaya çıkar. Öğrenme sistemi, kimi verilerin zamanla değişmesine ve keşif sisteminin verinin zamansızlığına karşın zaman duyarlı olmalıdır.

3.6 Veri Ambarı

Veri tabanlarında artık çok yüksek miktarda verinin tutulmasıyla beraber veri ambarı (data warehouse) kavramı kullanılmaya başlanmıştır. Veri ambarı, farklı kaynaklardan elde edilen verinin kullanıcıların bir iş ya da karar sürecinde kullanabileceği şekilde sunulduğu bütünleşik bir veri saklama ortamıdır (Kalıpsız, 1998). İyi tasarlanmış bir veri ambarı, veri madenciliği sürecini çok kolaylaştırabilir. Verinin temizlenmiş ve eksik kısımlarının tamamlanmış olması veri madenciliği sürecinin daha sağlıklı olmasını sağlar.

Birçok kaynakta bulunan veriyi bir araya toplayarak veri madenciliği için katalizör görevi gören veri ambarında, verinin akışını ve son kullanıcıya kadar ulaşma sürecini yakından inceleyecek olursak, veri ambarının çok katmanlı yapısıyla karşılaşırız. Bu çok katmanlı yapıyı kısaca özetleyecek olursak (Berry ve Linoff, 1997):

- **Kaynak Sistemler :** Verinin ilk elden ve soyutlama seviyesinin en düşük olduğu kısımdır. Operasyonel anlamda yararlanılan verinin karar destek için kullanılması söz konusu değildir.
- **Veri Nakli ve Temizlenmesi :** Bu aşamada veriyi kaynak sistemlerden çıkararak veri ambarı ve analiz ortamına nakleden yazılımlar kullanılır.
- **Merkezi Depo :** Veri ambarının teknik olarak en gelişmiş kısmıdır. Veriyi içinde bulunduran çok büyük bir veri tabanıdır.
- **Meta Veri :** Meta veri, verinin fiziksel alt yapısını hazırlar. Analiz için gerekli olan kısımların ön plana çıkarılmasına ve indeksi tablo, alan sayılarının belirlenmesine çalışılır. Veriye kendisi hakkında bilgi sağlar. Çoğu zaman veri ambarı çerçevesinde göz ardı edilen bir konudur.
- **Datamart :** Bir işletmede aynı anda farklı bilgilere ihtiyaç duyan insanlar olacaktır. Verinin tümü veri ambarında olduğuna göre aynı anda bu veri ambarından farklı bilgiler sağlamak mümkün mü? Bu sorunun cevabı datamartdadır. Datamartlar bir bölüm için gerekli olan bilgiyi merkezileştirme özelliğine sahip bir sistemdir.
- **Operasyonel Geri Besleme :** Bu aşamaya kadar olan veri işleme sonuçlarının geri besleme olarak operasyonel sisteme verilmesi sürecidir. Veri madenciliğinin hayati

döngüsünü tamamlama yeteneğine sahip bir süreçtir. Bu nedenle oldukça önemlidir.

- **Son Kullanıcı :** Veri ambarın yapısı içindeki en önemli kısmıdır. Son kullanıcı, analizci, uygulama geliştirici veya işletmeci olabilir.

3.6.1 Veri Ambarı Modelleme

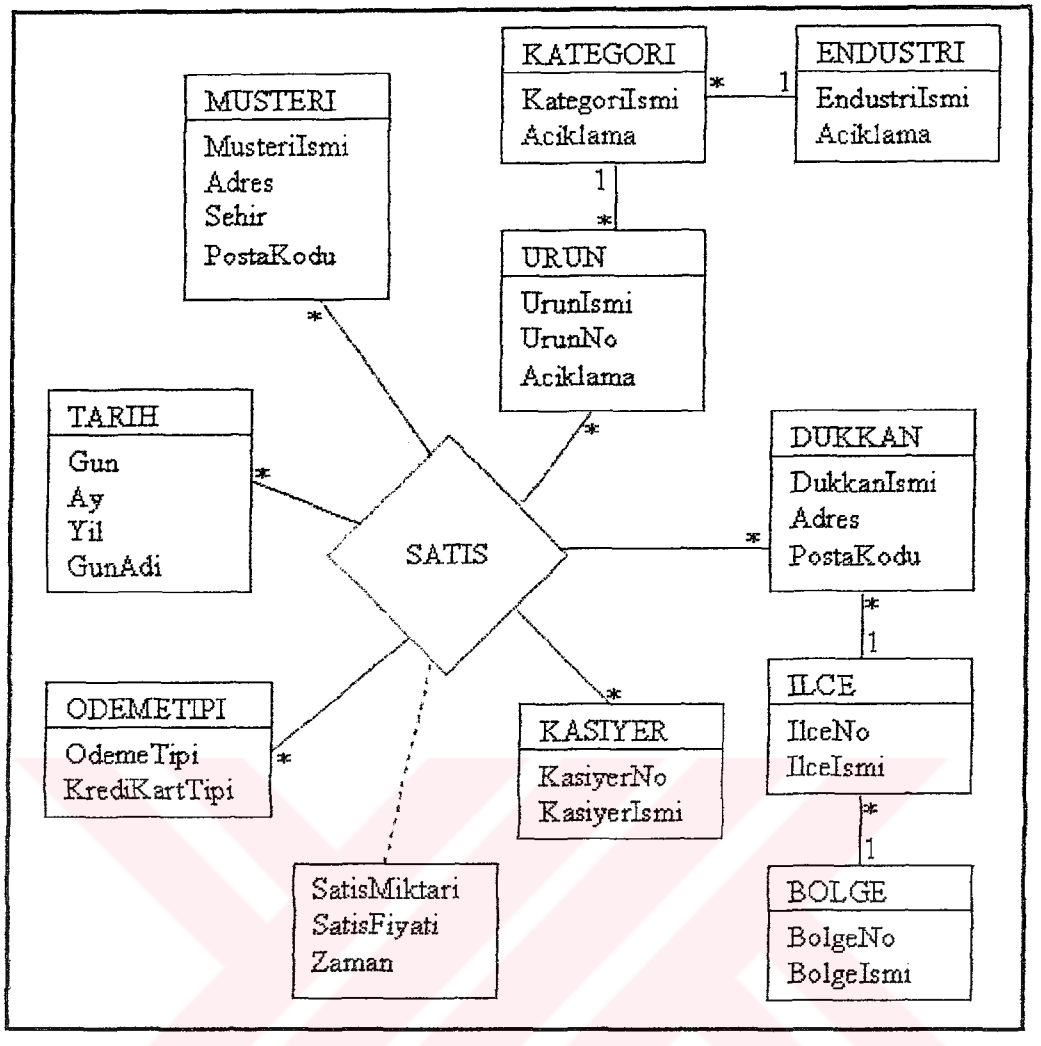
Veri ambarları, çeşitli boyutları olan olaylarla ilgilenir. Olay (fact), işin performansını ölçerken boyut (dimension) bu olayın temellerden birini sağlayan etkendir (Blaha,1998).

Modelleme, analitik ve operasyonel uygulamalar için farklılık göstermektedir. Operasyonel uygulamalar, ilişkilerle birbirine bağlı olan atomik varlıklarla ilgilenirler. Analitik uygulamalar ise daha basit ama daha çok tahmine dayalı bir yapıya sahiptirler.

Şekil 3.2'de bir markete ait veri ambarı modeli görülmektedir. "Satış", altı boyuta ("Tarih", "Müşteri", "Ürün", "Kasiyer", "Dükkan" ,"Ödeme Tipi") dayalı bir olaydır. Modelden anlaşılacağı üzere her sektörün birden fazla ürün kategorisi ve her kategoride de birden fazla ürün vardır. Benzer yapıda bir bölgenin birden fazla ilçesi her ilçenin de birden fazla dükkan olabilir.

Bir veri ambarının sadece karar almada yardımcı olabilecek verileri içermesine dikkat edilmelidir. Örneğin market veri ambarının kredi kartı numarası veya son kullanma tarihi gibi operasyonel verileri içermesine gerek yoktur.

Geniş organizasyonlara ait veri ambarlarında genelde 10 – 15 olay tablosu ile her olay tablosuna ait 5 – 15 boyut tablosu vardır. Eğer boyut sayısı azsa olması muhtemel sorgulamalar gözden geçirilip bu sayı arttırılmaya çalışılmalıdır. Tam aksi durumda ise bazı boyutların birbirine bağımlı olması durumu gözden geçirilmelidir. Bağımlı boyutlar birleştirilerek boyut sayısı azaltılabilir.



Şekil 3.2 Market veri ambarı

Her boyutun birden fazla özelliği olabilir. Bu özelliklerin bir kısmı tanımlayıcı iken geri kalanı tekrarlayıcıdır. Örneğin müşteri tablosunda "müşteriismi" ve "adres" tanımlayıcı özellikler, geri kalan özellikler ise tekrarlanan özelliklerdir. Böylece, veri ambarında bireysel satışlar incelenerek müşteri, şehir ve posta kodu bazında ayrı ayrı toplam satış miktarı bulunabilir.

3.7 OLAP Teknolojisi

İlişkisel veri tabanlarının yaygınlığı ve sonrasında ortaya çıkan veri ambarlarının gelişmesi ile beraber, verilere daha hızlı şekilde erişme ve çok boyutlu analiz ihtiyaçları, bilim adamlarını ve yazılım şirketlerini, daha farklı yapılar geliştirmeye itmiştir. Bu amaçla geliştirilen bir teknoloji olan OLAP (On-line Analytical Processing) 1993 yılında, Dr. E.F.Codd'un ortaya koyduğu kurallar çerçevesinde oluşmuştur. Bir veri yapısının OLAP olarak

nitelendirilebilmesi için 12 kural belirlenmiştir. [8]

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

OLAP ham veriyi, kullanıcının kavrayabileceği şekle dönüştürür. OLAP, tek başına yararlı bilgi oluşturmaz ve bir dönüştürücü etmen gibi davranarak, ham veriyi karar verme desteğinde kullanılmak üzere yönetimsel bilgiye çevirir.[9] OLAP'ın ortaya çıkardığı sonuçların daha da analiz edilmesi ve korelasyonunun alınmasıyla bazı görüşler sonucunda, gerçek, gerçek yararlı bilgiye dönüşüm sağlanmış olur.

Hem OLAP hem de veri madenciliği veri ambarı üzerinde çalıştıkları için çoğu zaman bu iki kavram birbirine karıştırılmaktadır. Halbuki OLAP ve veri madenciliği sadece birbirini tamamlayan ama birbirinden farklı uygulamalardır. Kullanıcı, kendi ortaya koyduğu hipotezi doğrulamak için OLAP aracını kullanır. Veri madenciliğinde ise tam aksi durum söz konusudur; hipotez üretmek için veri madenciliği aracı kullanılır. Veri madenciliğinde veri ambarından bilgi çıkarma sürecinde kullanıcının katkısının mümkün olduğunca az olmasına çalışılırken OLAP uygulamasında elde edilen sonuçların yorumlanması tamamıyla kullanıcının sorduğu sorulara bağlıdır. Ancak OLAP tekniği sayesinde SQL ile saatlerce hatta günlerce sürebilecek cevaplama zamanı dakikalara inmiştir. Ayrıca OLAP, veriye birden fazla açıdan bakma imkanı sağlayan bir küp şeklindedir.

Veri madenciliğinde istatistik, matematik, bulanık mantık ve yapay zeka gibi disiplinlerden faydalanılır. OLAP ise çok boyutlu analizi kolaylaştırmak üzere gerçekleştirilen işlevler dizisidir. Çok boyutlu analiz, boyutlar veya çeşitli sınıflar olarak gruplandırılan veri kullanma yeteneğidir (Berry ve Linoff, 1997). Bu yapısı sayesinde veri madenciliği, "ne?" sorusunun

arkasında yatan "niçin?" sorusundan başka, "neden o?" ve "başka ne?" gibi soruları da cevaplamaya çalışır.

3.8 Nesne İlişkisel Veri Tabanında Veri Madenciliği

Veri madenciliği uygulamaları, nesne ilişkisel veri tabanı sistemleriyle entegre edilerek nesne yöneliğin sunduğu kalıtım özelliğinden maksimum faydalanabilinir. Veri madenciliği ile nesne ilişkisel veri tabanı sistemlerinin entegrasyonu iki şekilde sağlanabilir. [13]

1. Veri tabanı şemasının kendi içinde veri madenciliğinin birleşmesi
2. Nesne ilişkisel veri tabanı sisteminde yer alan veri üzerinde veri madenciliği uygulamasına ait makina dilinin (host language) kullanılması

Birinci yöntemde, nesne ilişkisel veri tabanı üzerinde SQL kullanılarak geliştirilen öğrenen algoritmayla madencilik modeli kurulabilinir. Ancak bu durumda veri tabanı yönetim sistemi, madencilik modellerinin semantiklerinden habersiz olacaktır çünkü bu modeller aslında veri tabanı üzerinde değildir.

Birçok madencilik modeli geliştirilse bile bir kullanıcının ya da uygulamanın özelliklerine göre uygun modeli araması mümkün değildir. Çünkü mevcut veri tabanı yönetim sistemleri, bu modelleri paylaşma, yeniden kullanma ve yönetme kapasitesine sahip değilleridir. Benzer şekilde oluşturulan modellerden elde edilen tahminlerinin kıyaslanması gibi uygun bir mekanizma da mevcut değildir.

Geleneksel SQL, kolonlardaki meta verileri yakalamada yetersiz kaldığı için madencilik modelini en verimli şekilde kullanmak için bir kolonun özelliğinin (örneğin ayrık ya da sürekli olması gibi) ve kolonlar arası ilişkilerin tanımlanması gerekmektedir. Ancak nesne ilişkisel model, nesnelerin kalıtım özelliği sayesinde bu yapıya çok uygundur.

Bu yöntemde, veri madenciliği için kullanılan her durum bir nesne olarak tanımlanabilir. Durumun özellikleri doğrudan nesne özellikleri gibi tanımlanabilir. Bu özellikler arasındaki ilişki ise nesne referansları kullanma tekniğine uygulanarak saklanabilir. Nesne hem veriyi hem de davranışı birarada barındırdığı için geliştirilen veri madenciliği modelleri kavramsal olarak daha iyi anlaşılmaktadır.

Mevcut veri madenciliği teknolojisi, geleneksel ilişkisel veriden daha uzağa gitmemektedir. Çünkü resim, coğrafik harita, video filmler, ses kaydı vb. karmaşık veri tipleri üzerinde veri madenciliği yapılamamaktadır. Ancak nesne ilişkisel veri tabanı sistemleri sayesinde bu

problem kolayca çözülebilir. Bütün bu veri tipleri nesne olarak tanımlanması durumunda hem nesnenin fonksiyonları üzerinde veri madenciliği algoritmalarına olanak sağlanır hem de durum semantiği nesne içinde saklanır.

İkinci yöntemde, veri tabanı yöneticisinin denetimi için şemada ve sistemin alt yapısında gerekli değişikliklerin yapılması gerekmektedir. Makina dili kullanılarak nesne ilişkisel veri tabanı yönetim sisteminde (Object Relational Database Management System - ORDBMS) daha esnek bir veri madenciliği yaklaşımı kullanılır. Bu yapıyı gerçekleştirmek için önce nesne ilişkisel veri tabanından nesneleri almak sonra da nesnelerin davranışlarını kullanarak ilgilenilen asıl değerler uygulanabilir. Bu değerler, uygulanan veri madenciliği algoritmasının durumunu şekillendiren değerlerdir.

Yukarıda anlatılan sürece uygun olarak Oracle8i'de SQL sorgulaması çalıştırıldığında sonuç olarak nesneler elde edilir. Oracle8i'de nesneler ya bir özellik kolonu ya da bir nesne tablosu olarak saklanırlar. Çalışmanın dördüncü bölümünde nesnelerin Oracle8i'de nasıl saklandığı daha detaylı olarak anlatılmaktadır.

Nesnenin özellik olarak saklandığı durumda nesne, makina dilindeki nesne değişkenine atanır. Daha sonra nesne, durumuna göre kendi üyelik fonksiyonlarıyla sorgulanır. Nesneye herhangi bir veri madenciliği algoritması uygulanabilir. Örneğin kümeleme analizinde benzer, tahmin edilebilir karakteristik özelliklerine göre kayıtlar gruplandırılabilinir. Halbuki sıradan SQL sorgulamalarında bu karakteristik özellikler, saklı olduğu gibi sezgi yoluyla da gözlenemeyebilirler.

Nesnelerin, nesne tablosu olarak saklandığı durumda ise daha farklı bir yöntem uygulanır. Bu nesnelerin referans olaraka kullanılan tekil tanımlayıcıları (unique id) vardır. Oracle8i, değişken operatörü üzerinden satır nesnesini sonuç olarak döndüren gelişmiş SQL'i desteklemektedir. Ayrıca Oracle8i, metodları ve aşırı yükleme operatörlerinin uygulanmasına da olanak sağlar. Nitekim veri madenciliği algoritmaları çok fazla karşılaştırmaoperatörü kullandığı için aşırı yükleme özelliği çok önemlidir. Bu yöntemde bir nesne başka bir nesneye referans gösterebildiği için ulaşılmak istenen nesneye diğer nesneler üzerinden kolaylıkla erişilebilinmektedir. İstenilen nesne elde edildikten sonra bir önceki paragrafta anlatıldığı şekilde veri madenciliği uygulaması geliştirilir.

4. TARİHSEL BİLGİ SİSTEMİNİN GERÇEKLEŞTİRİLMESİ

Tarihi yapılar, yapıldıkları döneme ve içinde bulundukları coğrafi bölgeye göre oldukça farklı özelliklere sahiptirler. Dolayısıyla bu alanda yapılan çalışmalarda bir sonuca varmak için sadece yapıyı incelemek yeterli gelmemekte coğrafi konumla beraber tarihi olaylar da göz önüne alınmaktadır. Halbuki bilgisayar ortamında bugüne kadar geliştirilen bilgi sistemlerinde sadece yapısal bilgilerin tutulduğu görülmektedir. Tarihi restorasyon alanında çalışan araştırmacılara yardımcı olması amacıyla oluşturulan tarihsel bilgi sistemi (Historical Information System – HIS), tarihi yapıların özelliklerini ve tarihi olayların bilgilerini tutmaktadır.

4.1 Tarihsel Bilgi Sisteminin Teknik Gereksinimleri

Tarihi yapıların birbirinden farklı özellikler göstermesi nedeniyle çok esnek bir veri tabanı yapısına ihtiyaç duyulmaktadır. İlişkisel veri tabanının yapısından kaynaklanan kısıtlardan dolayı daha önce bu alan için yapılan bilgi sistemleri istenilen esnekliğe sahip değillerdir. Dolayısıyla kullanıcılar, veri girişinde zorlanmakta hatta eksik veri girişi yapmaktadırlar. Bu nedenle çalışmanın başında da belirtildiği üzere tarihsel bilgi sistemini nesneye yönelik veri tabanı yapısında geliştirilmesine karar verilmiştir.

HIS'in sadece yapısında değil analiz ve tasarım aşamasında da nesneye yönelik yaklaşım kullanılmıştır. Bu yaklaşıma uygun olarak modelleme aracı olarak UML, HIS veri tabanının oluşturulmasında IBM tarafından sunulan DB2 Universal Database ve bilgi keşfi sürecinde ise DB2'nun sunduğu Intelligent Miner kullanılmıştır. Kullanılan bu uygulamaların teknik özellikleri ve seçilme nedenleri çalışmanın bu kısmında açıklanmaktadır.

4.1.1 Nesneye Yönelik Modelleme Aracı

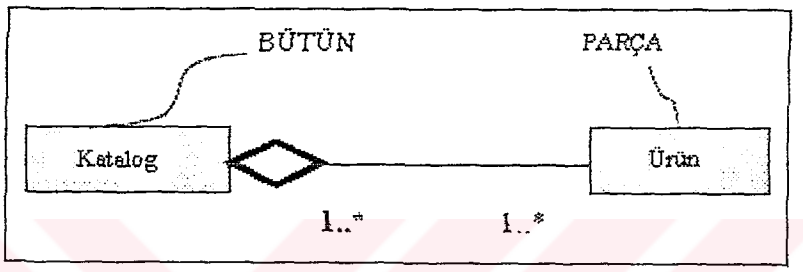
UML, Rumbaugh, Booch ve Jacobson'nun üç temel nesneye yönelik tasarımında yer alan elemanların biraraya getirilmesiyle oluşturulan işaretler sistemidir (Muller, 1999). UML'de yer alan diyagramlar sayesinde geliştirme aşamasındaki uygulamalar kolayca anlaşılabilir. Ancak her uygulamada diyagramların hepsini birden değil analizi yeterli biçimde tanımlayacak kadar diyagramın kullanılması yeterlidir. Çünkü bu diyagramlar birbiriyle ilişkili ancak bağımsız diyagramlardır. Bu bölümde UML'in sunduğu diyagramların genel olarak tanımı yapıldıktan sonra HIS'de nasıl kullanıldığı anlatılmaktadır.

Sınıf diyagramı (class diagram), insan, nesne gibi farklı yapıların birbiriyle ilişkisini gösterir. Başka bir deyişle, sistemin durağan yapısını gösterir. [10] Aynı zamanda her sınıfın özellikleri

ve işlemleri de bu diyagramda gösterilir. Bu açıdan baktığınızda sınıf diyagramı, varlık ilişki diyagramına (VAD) benzemektedir. VAD'de varlık tipi ile sınıf örtüşmektedir (Muller, 1999). Ancak aralarındaki en temel farklılık işlemlerin ve ilişkilerin modellenmesidir.

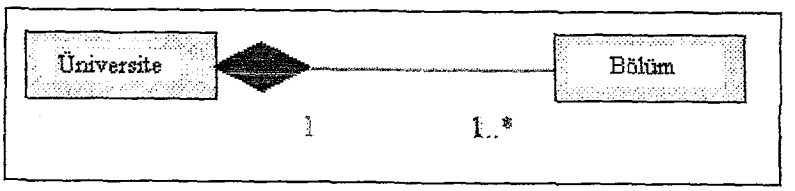
Sınıf diyagramını VAD'den ayıran en önemli fark, sınıflar arasındaki parça-bütün ilişkisinin gösterilmesidir. UML'de, güçlü ilişkiye **bileşim**, zayıf ilişkiye ise **bütünleşme** adı verilir.

Bütünleşme ilişkisine sahip sınıflar, ilişkide oldukları bütün sınıfları ile birlikte olabilecekleri gibi, tek başlarına da bir anlam ifade edebilirler. Şekil 4.1'de, bütünleşme ilişkisine sahip iki sınıf görülmektedir. Bir katalogda çeşitli ürünlerin tanımı olabileceği gibi, bir ürün hiç katalogda yer almadan satışa sunulabilir.



Şekil 4.1 Bütünleşme ilişkisi (Marcos vd., 2001)

Bileşim ilişkisinde ise bir sınıfın varolması için başka bir sınıfın olması gerekliliği şartını taşımaktadır. Bir nesne, üst sınıfın nesnesine bağlantısı olmadan yaratılamaz. Üst sınıfın nesnesi yok olduğunda, tüm alt sınıflara ait nesneler de yok olmalıdır. Şekil 4.2'de, bölüm ve üniversite sınıfları görülmektedir. Üniversite nesnesi olmadan, bölüm nesnesi tek başına olamaz. Üniversite nesnesi yok edildiğinde, ona bağlı tüm bölümler de yok edilmelidir.



Şekil 4.2 Bileşim ilişkisi (Marcos vd., 2001)

Bileşim ilişkisi, UML'de bağlantı çizgisinin bütün kısmının ucuna içi dolu bir dörtgen çizilerek gösterilir. Buna karşılık bütünleşme ilişkisi, içi boş dörtgen ile belirtilmektedir. Ayrıca her iki ilişki tipinde de ilişkinin derecesi "1", "1..*" vb. şeklinde gösterilebilinir. Ancak bu konuda incelenen kaynaklarda kesin bir zorlama olmadığı gözlenmiştir.

Kullanım senaryosu diyagramı (use case diagram), sistem tarafından sağlanan fonksiyonelliği gösterir. Kullanım senaryosu diyagramının asıl amacı, aktörlerin* süreçlerle ve farklı senaryolarla ilişkilerini göstererek sistemin fonksiyonel ihtiyaçlarını belirlemektir. [10]

İşbirliği diyagramı (collaboration diagram), mesajlar yollayan ve alan nesnelerin yapısal organizasyonları üzerinde duran bir tür etkileşim diyagramıdır. İşbirliği diyagramı, yazılım sisteminin statik yapısını ve dinamik davranışlarını birlikte gösterir. [11] Kullanım senaryosu diyagramında belirtilen her durum için işbirliği diyagramı çizilmelidir.

Ardışıl diyagram (sequence diagram), belli bir kullanım senaryosuna ve bir kullanım senaryosunun sadece bir bölümüne ait detaylı akışı gösterir. İşbirliği diyagramından farklı olarak daha detaylı bilgi içerir. İki boyutu olan ardışıl diyagramın dikey boyutunda zaman içinde oluşan mesaj ve çağrıların sırası gösterilirken, düşey boyutunda mesaj gönderilen nesne örnekleri gösterilir. [10]

Durum grafiği diyagramı (state chart diagram), bir sınıfın içinde bulunabileceği durumları ve bir durumdan diğer duruma nasıl geçileceğini modeller. Her sınıfın bir durumu vardır ama her sınıfın bir durum grafiği diyagramı olmayabilir. Sistemde sadece önemli durumlara sahip olan sınıflar için modelleme yapılması yeterlidir. [10]

Aktivite diyagramı (activity diagram), bir aktivite gerçekleştirilirken sistemdeki iki veya daha fazla sınıf nesnesi arasındaki kontrolün akışını gösterir. Aktivite diyagramları, üst düzey iş süreçlerinin veya alt düzey iç sınıfların faaliyetlerinin modellenmesinde kullanılabilinir. [10] Çünkü aktivite diyagramı, ne işbirliği ve ardışıl diyagramları kadar detaylı ne de kullanım senaryosu diyagramı kadar genel değildir.

4.1.2 Veri Tabanı Yönetim Sistemi

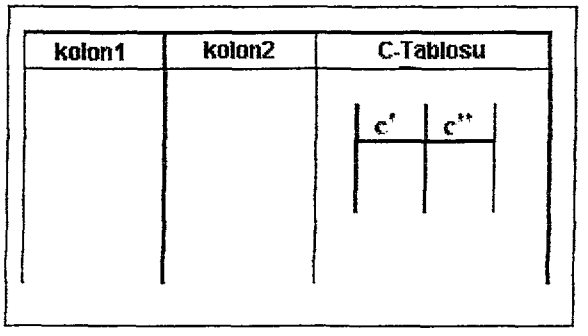
Çalışmanın amacına uygun olarak piyasada bulunan nesneye yönelik veri tabanı yönetim sistemleri incelenmiştir. En yaygın olarak kullanılan ORACLE ve DB2 arasından aşağıda anlatılan özellikleri nedeniyle HIS'in veri tabanı için IBM tarafından geliştirilen DB2 (Database 2) Universal Database Versiyon 8.1 kullanılmıştır.

Oracle8i, SQL99'da tanımlanan ARRAY'e karşılık gelen **VARRAYS** ve **içiçe tablolarla** (nested tablo) koleksiyonu (collection) desteklemeksine rağmen nesneye yöneliğin en önemli

---

* Sistemle ilişkili olan kişidir. UML'de aktörler çöp adamlarla temsil edilirler.

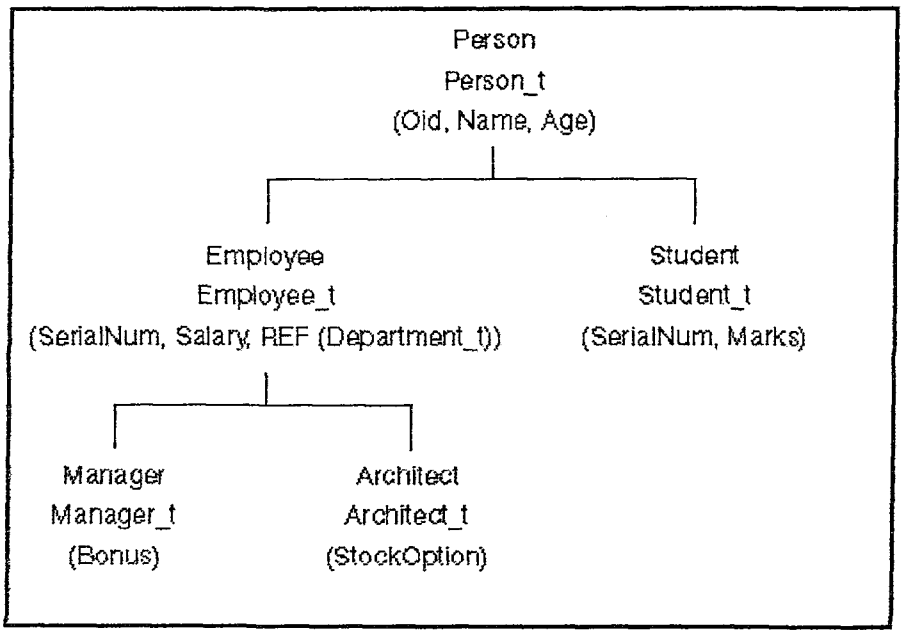
özelliklerinden olan kalıtımı desteklememektedir. İçiçe tablo, başka bir tablonun içine gömülüdür. Bir tabloyu veri tipi olarak tanımlayıp diğer tabloda da bir kolonun tipini bu tablo veri tipinde tanımlamak mümkündür (Şekil 4.3). SQL99 tarafından desteklenmeyen içiçe tablo, UML'de tanımlanan kümeleşmenin (aggregation) uygulamasıdır. Böylece bir nesnenin içine gömülen tablo tipindeki kolon ile başka bir nesne topluluğuna referans verilir (Marcos vd., 2001).



Şekil 4.3 Oracle8i'de içiçe tablo (Marcos vd., 2001)

İlişkisel veri tabanı modelini destekleyen IBM DB2, SQL99'un gelişimiyle beraber nesne ilişkisel veri tabanı modelini desteklemeye başlamıştır. İşletim sistemi platformundan bağımsız olan DB2, Oracle'ın aksine basit kalıtımı desteklemektedir. Ancak sınıflar arası çoklu ilişkiyi destekleyen VARRAYS gibi bir dizi veri tipine sahip değildir.

DB2'da kalıtımın takip edilebilmesi için hiyerarşi tablosu kullanılmaktadır. Bu tablo, hiyerarşide yer alan her tablonun her tekil kolonu için bir kolona sahiptir. Örneğin Şekil 4.4'de gösterilen hiyerarşiyi göz önüne alındığında hiyerarşi tablosu, "oid, name, age, serialnum, salary, marks, bonus, stockoption" kolonlarına sahip olacaktır. Kolonların sırası tabloların oluşturulma sırasına bağlıdır. Ayrıca tip tanımlayıcısı (type ID) için ekstra bir kolonda yer alır.[12]



Şekil 4.4 Sınıf hiyerarşisi [12]

Hiyerarşi tablosunda tip tanımlayıcılarının Person için 10, Employee için 25, Manager için 35, Architect için 45 ve Student için 100 olduğunu düşünelim. Bu sınıf yapısına göre kullanıcı tarafından girilen kayıtların hiyerarşi tablosundaki görünümü Çizelge 4.1'de verilmiştir.

Çizelge 4.1 Hiyerarşi tablosu [12]

(type)	Oid	Name	Age	SerialNum	Salary	Marks	Bonus	StockOption
10	a	Andrew	20	--	--	--	--	--
10	b	Bob	30	--	--	--	--	--
10	c	Cathy	25	--	--	--	--	--
25	d	Dennis	26	105	30000	--	--	--
25	e	Eva	31	83	45000	--	--	--
25	f	Franky	28	214	39000	--	--	--
100	g	Gordon	19	10245	--	90	--	--
100	h	Helen	20	10357	--	70	--	--
35	i	Iris	35	251	55000	--	12000	--
35	j	Christina	10	371	85000	--	25000	--
35	k	Ken	55	482	105000	--	48000	--
45	l	Leo	35	661	92000	--	--	20000
45	m	Brian	7	882	112000	--	--	30000

HIS çalışması kapsamında nesne ilişkisel modelinin kullanılmasındaki başlıca nedenlerden biri kalıtım özelliği olduğu için veri tabanı yönetim sistemi olarak DB2 seçilmiştir.

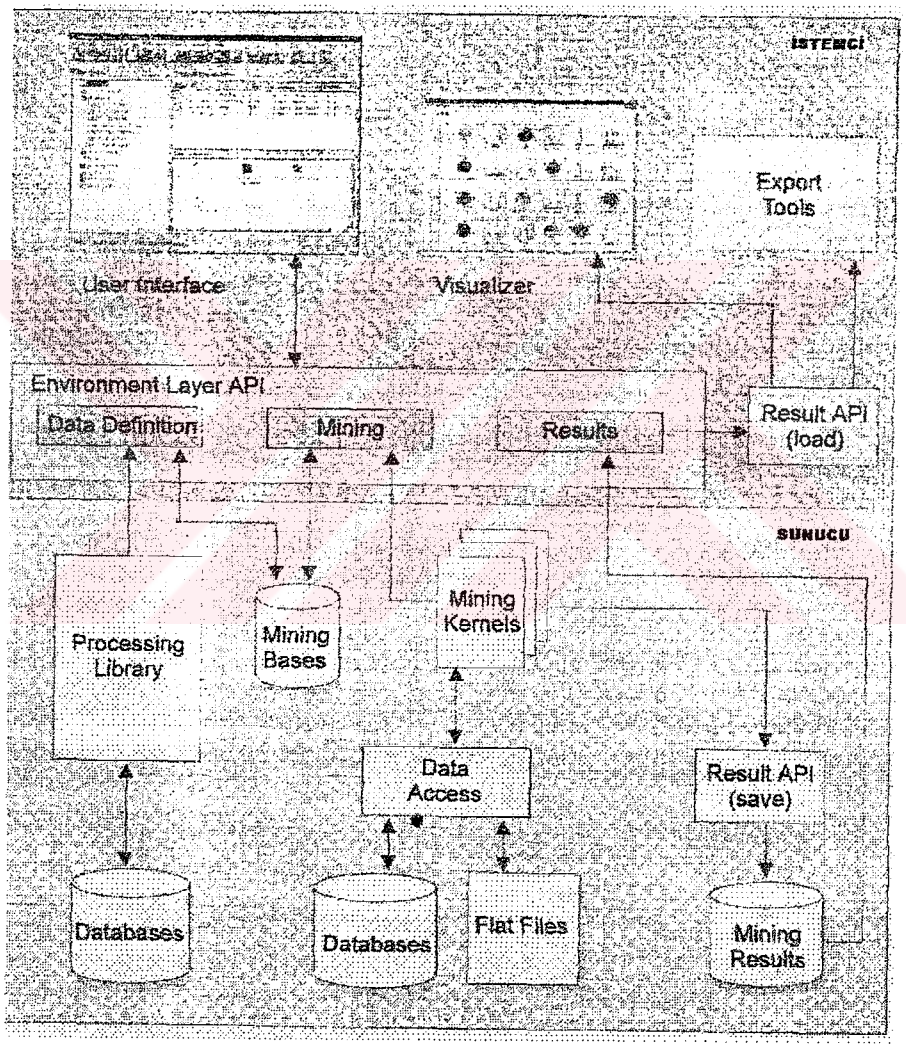
4.1.3 Bilgi Keşfi Aracı

Veri tabanı yönetim sistemi aracı olarak IBM DB2 ile uyumlu bilgi keşfi sürecinde kullanılan araçlar incelenmiştir. Yine IBM tarafından üretilen Intelligent Miner'ın, DB2 ile uyum problemi olmaması, ilişkilendirme, sınıflama, tahmin modelleme, gruplama, sıralı desen analizi, regresyon analizi gibi fonksiyonları içermesi ve kolay (user friendly) ara birimlerine

sahip olması nedeniyle uygulamada kullanılmıştır.

Intelligent Miner, ayrıca yapay sinir ağları algoritmaları, istatistik metotları, veri hazırlama ve görsel gösterim araçları gibi ek özellikler içerir. Ölçeklendirebilir olması ve IBM'in DB2 ilişkisel veritabanı sistemi ile sıkı birleşik olması diğerler veri madenciliği araçlarından ayıran en önemli özelliğidir.

Günümüzün en kapsamlı doküman madenciliği araçlarından olan IBM Intelligent Miner, kümeleme, özetleme, özellik belirleme (feature-extraction) vb. doküman analiz araçlarına ve doküman arama motoruna sahiptir (Kantardzic, 2001).



Şekil 4.5 Intelligent Miner'ın yapısı (IBM, 1999)

Şekil 4.5'de Intelligent Miner'ın yapısında istemci ve sunucu taraflarındaki bileşenler gösterilmektedir. Kullanıcı ara birimiyle (user interface) kullanıcı, grafik ortamda veri

madenciliği fonksiyonlarını tanımlayabilmektedir. Ortam katmanı (environment layer API), madenciliğin çalışmasını ve sonuçlarını kontrol eden bir grup API fonksiyonuna sahiptir. Bu katman bütün sunucu işletim sistemlerinde bulunmaktadır. Canlandırıcı (visualizer), madencilik veya istatistiki fonksiyonlar sonucu elde edilen sonuçları göstermek için kullanılan bir araçtır. Veri erişimi (data access), veri tabanlarına ve dosyalara erişimi sağlar. Süreç kütüphanesi (processing library), veri tabanına erişim fonksiyonlarını içerir. Madencilik tabanları (mining bases), veri madenciliği nesneleri topluluğudur. Bu nesneler, iş probleminin çözümünde kullanılırlar. Madencilik çekirdekleri (mining kernels), kullanıcı veri madenciliği veya istatistiki fonksiyonu çalıştırdığında ilgili algoritmalar çalıştırılmaya başlanır. Madencilik veya istatistiki fonksiyon sonucu elde edilen sonuçların istemci tarafında gösterimi, madencilik sonuçları (mining results), API sonucu (result API) ve ihraç araçları (export tools) sayesinde yapılmaktadır (IBM, 1999).

Intelligent Miner, dört araçtan oluşmaktadır: (IBM, 2002)

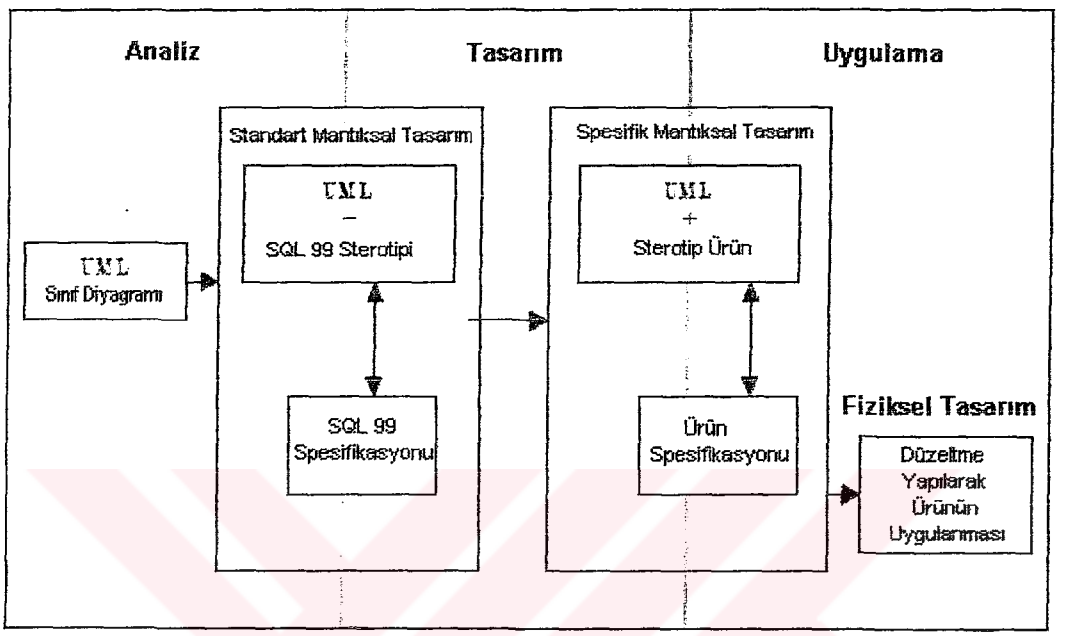
- **Intelligent Miner for Data;** veri madenciliği modellerinin test edilip oluşturulmasında kullanılır. Veri madenciliğinin veri hazırlama ve dönüştürme uygulamalarında içeren bu araç modelleri PMML* (Predictive Model Markup Language) standartında oluşturur. Bu uygulama, veri tabanı üzerinde çalıştığı gibi düz dosyalar üzerinde de çalışmaktadır.
- **Intelligent Miner for Modeling;** DB2'ya erişerek SQL uygulamalarıyla kümeleme, sınıflandırma ve birliktelik işlemlerini çalıştırır. Sonuçta oluşturulan PMML V2.0 formatındaki modeller, Scoring ve Visualizing araçları tarafından kullanılır.
- **Intelligent Miner for Visualizing;** Java görselleriyle analizde kullanılmak üzere veri modelleme sonuçlarını görüntülemek için kullanılır. Bu sonuçlar, uygulamarla Web ortamında da görüntülenebilinir.
- **Intelligent Miner for Scoring;** PMML modellerinin geniş veri tabanlarına veya basit durumlara uygulamak için kullanılır. Intelligent Miner for Data, skorlama işlemi için UDF ve UDM içeren SQL API'yi kullanır. Kullanılan PMML modeller, Intelligent Miner'ın bir ürünüyle geliştirildiği gibi başka araçlar tarafından da geliştirilebilinir.

---

* Predictive Model Markup Language (PMML), XML tabanlı dildir. Farklı firmaların geliştirdikleri tahmin edici modellerin kolayca paylaşımını sağlayan bir standarttır.

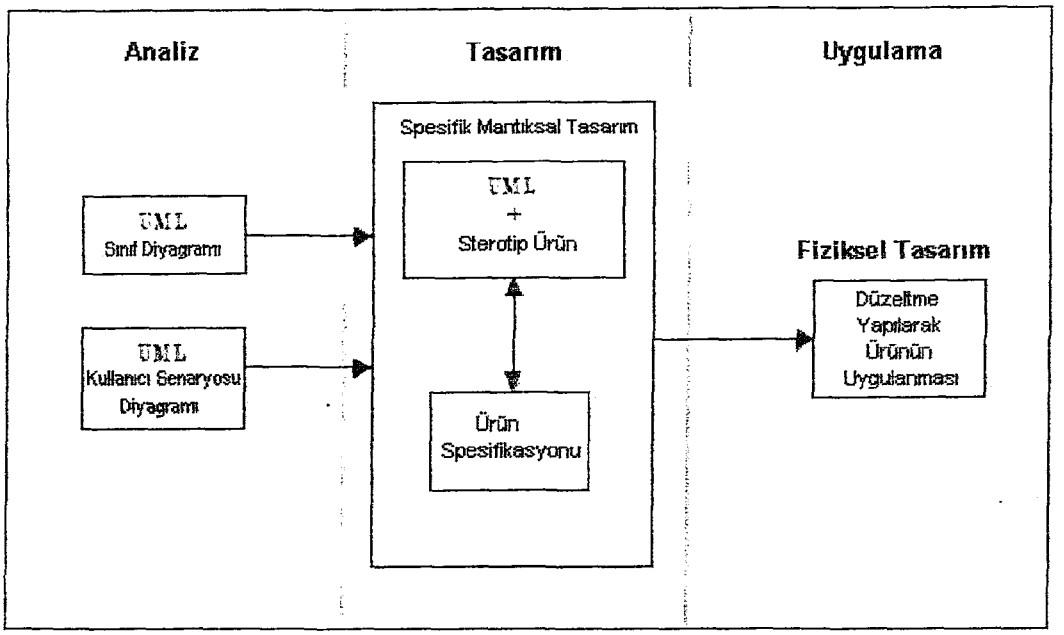
4.2 HIS'in Oluşturulması

Bertino, Marcos ve Caceres'in yaptığı çalışmalar sonucu ortaya konan nesne ilişkisel veri tabanı geliştirme metodolojisi Şekil 4.6'da gösterilmektedir. Analiz, tasarım ve uygulama olmak üzere üç ana aşama öneren metodolojide aşamalar arasındaki farklılıklar yapısal dizaynda olduğu gibi güçlü değildir. (Marcos vd., 2001)



Şekil 4.6 Nesne ilişkisel veri tabanı dizayn metodolojisi (Marcos vd.,2001)

HIS, bu metodoloji esas alınarak oluşturulan modele göre oluşturulmuştur. Şekil 4.7'de geliştirilen bu modelde farklı olarak analiz aşamasında sınıf diyagramına kullanım senaryosu diyagramı da eklenmiştir. Böylece sistemin ileride ara birimlerinin geliştirilmesinde kullanım senaryosu diyagramlarından faydalanabilinecektir.



Şekil 4.7 HIS'in oluşturulmasında kullanılan model

Temel alınan metodolojide tasarım aşaması standart ve spesifik olarak ikiye ayrılmıştır. Standart mantıksal tasarım aşamasında sınıflar, kullanılan ORDBMS'den bağımsız olarak SQL99 standartlarına göre oluşturulurken, spesifik mantıksal tasarım aşamasında kullanılan ORDBMS'in sunduğu sentaksa göre oluşturulmaktadır (Marcos vd.,2001). SQL99 standartları ile HIS'in ORDMS'i olarak kullanılan IBM DB2'nun SQL sentaksı arasında büyük farklılıklar olmadığı için oluşturulan modelin tasarım aşamasında bu ayrım yapılmamıştır (Şekil 4.8).

SQL 99

```
CREATE TYPE person AS
( firstname char(25),
  lastname char(25)) ;

CREATE TABLE persontable OF person
( firstname NOT NULL);
```

DB2

```
CREATE TYPE person AS
( firstname char(25),
  lastname char(25))
MODE DB2SQL

CREATE TABLE persontable OF person
( REF IS pid USER GENERATED,
  firstname WITH OPTIONS NOT NULL)
```

Şekil 4.8 SQL99 ile DB2 arasındaki farklılıklar

Son olarak uygulama aşamasında fiziksel dizayn gerçekleştirilirken uygulamanın ihtiyaçlarına göre daha verimli olmak amacıyla bir önceki aşamada tasarlanan sınıflar yeniden düzenlenir.

Örneğin DB2, dizi veri tipini desteklemediği için sınıflar arasındaki çoka çok ilişkiyi sağlamak amacıyla dizi yapısında yeni sınıflar oluşturulmuştur. Bu problemin çözümüne yönelik yapılan uygulama "HIS'in Gerçekleştirilmesinde Karşılaşılan Problemler ve Çözümler" bölümünde anlatılmıştır.

4.2.1 Analiz

Nesne ilişkisel veri tabanı modeline dayanarak geliştirilen HIS'in analiz aşamasında tarihi yapılarda karşılaşabilinecek sınıflar belirlenmiştir. Sınıflar tanımlanırken ileride kullanıcı ihtiyaçlarına cevap verebilmek için esnek bir yapı geliştirilmeye çalışılmıştır. Böylece kullanıcılar, istedikleri takdirde HIS veri tabanında bulunan temel sınıfların altında yeni alt sınıflar tanımlayabileceklerdir.

Tarihi yapılar üzerine yapılan araştırmalarda sadece yapıların özellikleri değil yapının içinde bulunduğu bölgenin özellikleri ile o bölgeyi etkileyen tarihi olaylar da dikkate alınmaktadır. Bu nedenle HIS veri tabanında tarihi olaylar da tutulmaktadır. Böylece bu çalışmanın çıkış noktasını oluşturan nesne ilişkisel veri tabanı üzerinde veri madenciliği yöntemlerinin uygulanması sürecinde tarihi yapıların hem kendi içlerinde hem de tarihi olaylarla ilişkileri ortaya konabilmektedir.

SmartDraw* programı kullanılarak çizilen HIS için oluşturulan sınıf diyagramı Şekil 4.9'da gösterilmiştir. Bu sınıf diyagramında insan (person), yapı (structure) ve tarihi olaylar (historical events) sınıfları temeli oluşturmaktadırlar. Sınıf diyagramındaki diğer sınıflar ise ya bu temel sınıfların alt sınıfları ya da bu sınıfların ilişkili olduğu sınıflardır. Ancak HIS'in yapısında çok fazla sınıf olduğu için sınıflar arasındaki çoklu ilişkileri oluşturmak için kullanılan dizi görevini gören ara sınıflar bu tabloda yer almamaktadır. Bu tipteki sınıfların yapısının nasıl oluşturulduğu ileride ayrıntılı olarak anlatılmıştır.

HIS veri tabanı, hem yapının yukarıda anlatıldığı gibi esnek olması hem de veri madenciliği sonucunda daha iyi sonuç alabilmek amacıyla tarihi yapı bilgisi detaylı olarak saklanacak şekilde tasarlanmıştır. Bu nedenlerle tarihi bilgi sisteminde yer alan nesneler, mümkün olduğunca alt sınıflara ayrılarak oluşturulmuştur. Mevcut sistem içindeki bir kavramın sınıf olarak mı yoksa sınıfın bir özelliği olarak mı kaydedilmesi konusunda karar vermek için aşağıdaki sorular kullanılmıştır:

---

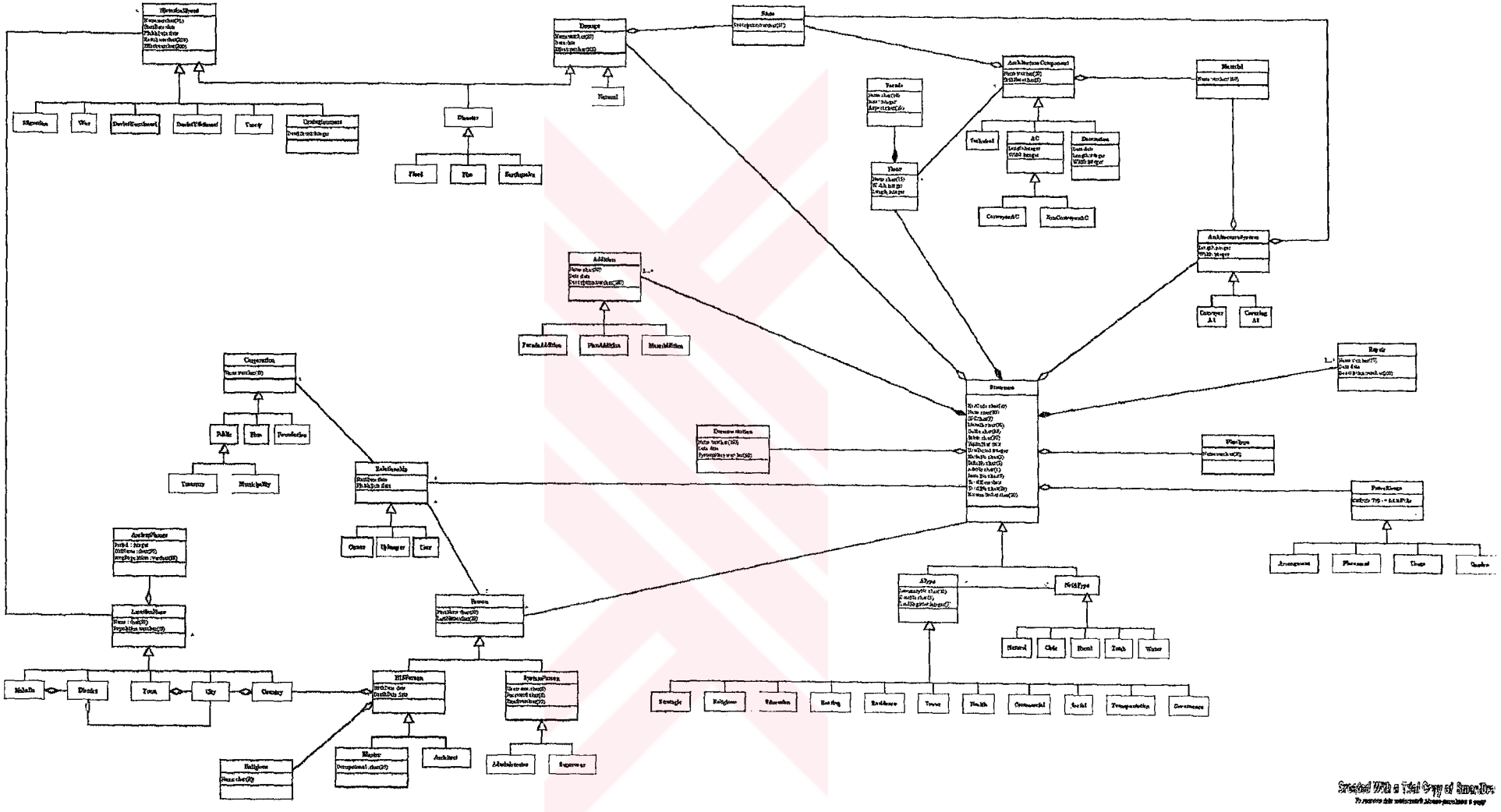
* Windows işletim sistemiyle uyumlu çalışan SmartDraw, proje tasarım aşamasında gerekli olan UML ve ER

1. Kavramla ilgili veri saklanması gerekiyor mu?
2. Değişik değerler alabilecek farklı özellikleri var mı?
3. Kavramdan birçok nesne türeyebilir mi?
4. Uygulamanın kapsama alanı içinde mi?

Örneğin HIS yapısında yapının adres bilgisindeki il, ilçe gibi yerleşim yerleri birer sınıf olarak tanımlanmansına karar verilmiştir. Çünkü yerleşim yerinin tipi, bugünkü adı, tarihi isimleri ve nüfus bilgileri sistemde tutulması gerekmektedir. Diğer nedense yerleşim yerinin tarihi olaylarla tarihi yapıların ortak noktası olmasıdır.

---

gibi tasarım diyagramlarının çizilmesinde kullanılan bir yazılım aracıdır.



Şekil 4.9 HIS sınıf diyagramı

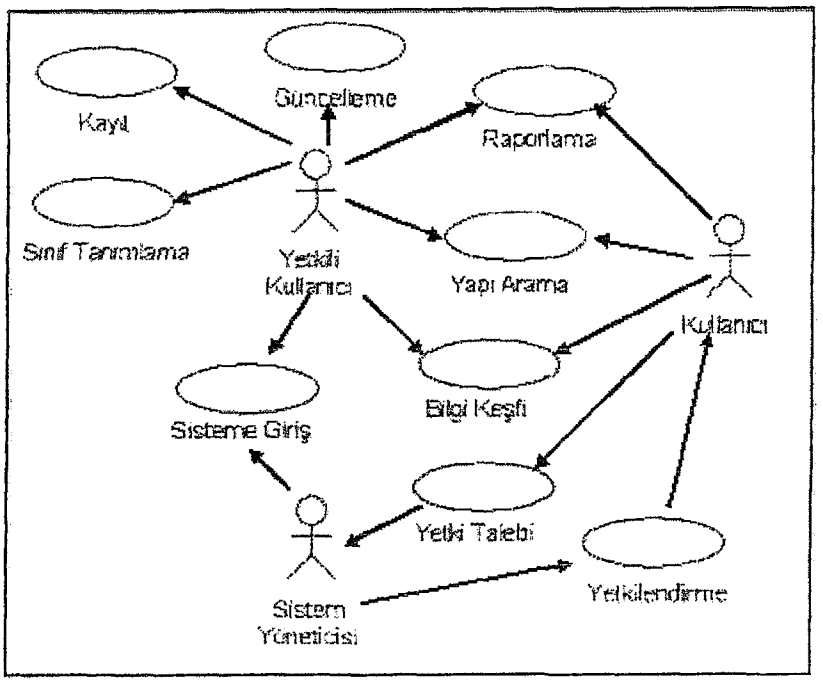
Analiz aşamasında gerçekleştirilen diğer işlem ise kullanıcı senaryolarının oluşturulmasıdır. HIS'de oluşabilecek senaryoları belirlemek için önce sistemin genel akışı aşağıdaki şekilde tanımlanmıştır:

1. Yetkili kullanıcı veya sistem yöneticisi sisteme giriş yapar.
2. Yetkili kullanıcı, bir sınıfa ait verileri sisteme girebilir.
3. Yetkili kullanıcı, mevcut bir yapıyı çeşitli kriterlere göre arayabilir.
4. Yetkili kullanıcı, kayıtlar üzerinde güncelleme yapabilir
5. Yetkili kullanıcı, çeşitli kriterlere göre raporlama yapabilir.
6. Yetkili kullanıcı, veri tabanında yeni bir sınıf tanımlayabilir.
7. Kullanıcı, mevcut bir yapıyı çeşitli kriterlere göre arayabilir.
8. Kullanıcı, çeşitli kriterlere göre raporlama yapabilir.
9. Kullanıcı, yetkili kullanıcı olmak için sistem yöneticisinden yetki isteyebilir.
10. Sistem yöneticisi, kullanıcıya yetki verebilir.
11. Yetkili kullanıcı, veri madenciliği tekniklerini kullanarak istediği özelliklere göre yapılar üzerinde bilgi keşfi sürecini gerçekleştirebilir.

Sistemin bu genel akışı göz önüne alındığında üç temel aktör;

- yetkili kullanıcı,
- kullanıcı
- sistem yöneticisi

ve dokuz temel kullanım senaryosu Şekil 4.10'da gösterilmiştir. Bu kullanım senaryolarının her birinin detaylarını içeren dokümantasyonlar ise EK 2'de verilmiştir.



Şekil 4.10 HIS kullanım senaryosu diyagramı

4.2.2 Tasarım

HIS nesne ilişkisel veri tabanında olması tasarlanan sınıflar ve bu sınıflara ait tablolar Şekil 4.8'de gösterilen örnekte ki gibi SQL99 formatında DB2 SQL sentaksına göre geliştirilmiştir. EK 3'de HIS nesne ilişkisel veri tabanında yer alan bütün sınıflara ve sınıf tablolarına ait SQL komutları bulunmaktadır. Sistemin tasarlanmasında kullanılan SmartDraw programının veri tabanı sistemiyle bağlantısı olmadığı için veri tabanını oluşturan EK 3'deki SQL komutları DB2'nun sunduğu Command Center isimli SQL ara biriminde tek tek yazılarak çalıştırılmıştır.

Çalışmanın teorik kısmında anlatıldığı üzere sınıfları birbirinden ayırt etmek için DB2 tarafından nesne tanımlayıcısı (OID) alanlarının tanımlanması zorunludur. Kullanıcı, sınıflara ait tabloları oluştururken henhangi bir isim tanımlanmadığı takdirde otomatik olarak oluşturmaktadır. Ancak HIS'de oluşturulan sınıfların SQL sorgulamalarında daha anlaşılır olması OID'ler tanımlanması için isim formatı oluşturulmuştur. OID alanının formatı; sınıfın ismi iki kelimeden oluşuyorsa kelimelerin ilk harfleri + "id"; sınıf tek kelimeden oluşuyorsa kelimenin ilk harfi + ilk harfinden sonraki ilk sesssiz harf + "id" şeklinde geliştirilmiştir.

Örnek : "parcelusage" tablosu için "p" + "u" + "id" = "**puid**"

"material" tablosu için "m" + "t" + "id" = "**mtid**"

HIS'in oluşturulması aşamasında geliştirilen diğer formatlarda da uygulamayı ve ileride bakımı kolaylaştırmak amacıyla geliştirilmitir. Birinci format, sınıflarla ve sınıflara ait tablolarla ilgilidir. Bir sınıfa ait tablonun ismi, sınıfın ismine "table" kelimesinin eklenmesi şeklinde oluşturulur.

Örnek : "material" sınıfına ait tablonun adı "**materialtable**" olarak verilmiştir.

İkinci format ise farklı sınıflar atlında aynı tip veriyi tutmak üzere oluşturulan alan adlarına yöneliktir. OID alanı formatında olduğu gibi bu formatın geliştirilme nedeni SQL sorgulamalarında kolaylığın sağlanmasıdır. Örneğin tarih bilgisi birçok sınıfta olması gereken bir alan olduğu için bu alanın ismi verilirken ait olduğu sınıfın ilk harfi alan adı önüne eklenir.

Örnek: "repair" sınıfında yer alan tarih alanın adı "r" + "date" = "**rdate**" olarak verilmiştir.

4.2.3 Uygulama

Çalışmanın teorik kısmında belirtildiği üzere bilgi keşfi süreci sonunda sağlıklı sonuçlara ulaşmak için uygulamada veri girişi üzerinde önemle durulmuştur. HIS'de yer alan yapı bilgileri, 2003'de Kütahya çevresinde YTÜ Mimarlık Fakültesi Restorasyon Anabilim Dalı tarafından yapılan araştırma sonuçlarından alınmıştır. Bu araştırma kapsamında incelenen yapıların bilgilerinin nasıl tutulduğunu göstermek amacıyla örnek dokümantasyon EK 4'de yer almaktadır. Kütahya'nın ait tarihi bilgiler ise internet ve yazılı tarihi kaynaklardan toparlanarak HIS'in yapısına uygun olarak veri girişi yapılmıştır.

Ancak EK 4'deki örnek dökümantasyonda da görüldüğü üzere yapılara ait verilerin HIS'te tasarlandığı kadar detaylı olmaması ve sadece bir bölgeye ait yapılara ait bilginin bulunması çalışma öncesinde amaçlanan detaylı sonuçlara ulaşılmasını engellemiştir.

4.2.3.1 HIS'in Hiyerarşik Yapısının Oluşturulması

DB2'da hiyerarşi tablosu, tip tablosuna ait hiyerarşinin uygulanması ile uyumlu çalışmaktadır. Hiyerarşik yapı içindeki kök tablo oluşturulduğu anda hiyerarşi tablosunda otomatik olarak sistem tarafından oluşturulur. Oluşan bu hiyerarşi tablosunun ismi kullanıcı tarafından belirtilmezse kök tablosunun ismi ile aynı şekilde türetilip sonuna yine sistem tarafından üretilen tekil son ek olarak eklenir.[12] Örneğin hiyerarşik yapıda kök olan PERSON tablosu oluşturulduğunda sistem tarafından PERSON_HIERARCHY adında bir hiyerarşi tablosu oluşturulmaktadır.

"list tables" SQL cümlesi çalıştırıldığında kullanıcı tarafından oluşturulan tüm tabloların ismi, tablo tipi ve oluşturulduğu zaman (timestamp) olarak listenir. Çizelge 4.2'de görüldüğü üzere veri tabanında PERSON isimli tip tablosunun ve PERSON_HIERARCHY isimli hiyerarşik tablonun oluşturma zamanları birebir aynıdır. Bu listede hiyerarşi tablo tipi de yer almasına karşın SQL cümlesiyle hiyerarşi tablosuna doğrudan ulaşmak mümkün değildir. Uygulama içinde sınıf hiyerarşisinden faydalanmak için kullanılan yöntem "Rekursif SQL ile Sınıf Hiyerarşinin Belirlenmesi" başlığı altında anlatılmaktadır.

Çizelge 4.2 DB2 Command Center'da "List Tables" SQL cümlesi çalıştırıldıktan sonra elde edilen sonuç

Table/View	Schema	Type	Creation Time
PERSON	HISDB	T	2004-05-17-13.30.19.444215
PERSON_HIERARCHY	HISDB	H	2004-05-17-13.30.19.444215

4.3 HIS'in Gerçekleştirilmesinde Karşılaşılan Sorunlar ve Çözümler

HIS'in hem veri tabanının geliştirilmesinde hem de uygulaması esnasında pek çok sorun ile karşılaşılmıştır. Bu bölümde bu çalışmadan sonra yapılacak benzer çalışmalara yardımcı olması düşüncesiyle karşılaşılan sorunlar ve imkanlar ölçüsünde bulunan çözüm önerileri anlatılacaktır.

4.3.1 Rekursif SQL ile Sınıf Hiyerarşinin Belirlenmesi

Daha öncede belirtildiği gibi DB2 her sınıf hiyerarşisi için bir hiyerarşi tablosu oluşturulmasına rağmen "select * from <tablename>" şeklinde bir SQL cümlesiyle iç yapısına ulaşılamamaktadır. Halbuki HIS'in amaçlarından biri olan esnek veri tabanı yapısının sağlanması için hiyerarşik yapının nasıl olduğunun bilinmesi gerekmektedir. Bu amaçla veri tabanı yapısında yer alan her sınıf için oluşturulan tablo, "hierarchytable" isimli tabloya kaydedilir. Bu tablonun yapısı,

```
CREATE TABLE hierarchytable
( tableid int,
  tablename char(20),
  supertableid int )
```

SQL komutu ile oluşturulmuştur.

Oluşturulan sınıf hiyerarşik yapı içinde kök ise bu durumda bir üst sınıfı olmadığı için

"hierarchytable" tablosunda o sınıfa ait tablo için "supertableid" alanına değer atanmaz. Oluşturulan alt sınıfa ait tablonun ait olduğu üst sınıfın tablosunun "tableid" değeri "supertableid" olacak şekilde kayıt yapılır.

Kök sınıfa ait tüm alt sınıfların tablo isimlerine ve her tablonun bu hiyerarşide kaçıncı seviyede olduğunu belirlemek içinse aşağıdaki rekursif SQL cümlesi çalıştırılması yeterlidir.

```
WITH temptab(tableid, tablename, supertableid, level) AS
( SELECT root.tableid, root.tablename, root.supertableid, 1
  FROM hierarchytable root
        WHERE root.tablename = 'person'
  UNION ALL
  SELECT sub.tableid, sub.tablename, sub.supertableid, super.level+1
  FROM hierarchytable sub, temptab super
  WHERE sub.supertableid = super.tableid
)
SELECT tablename FROM temptab
```

Bu SQL sorgusunu Şekil 4.9'da gösterilen HIS yapısı için çalıştırdığımızda elde edilen sonuç Çizelge 4.3'de verilmiştir.

Çizelge 4.3 Rekursif SQL sonucu elde edilen sınıf hiyerarşi

TABLEID	TABLENAME	SUPERTABLEID	LEVEL
10	Person	--	1
20	HISPerson	10	2
30	SystemPerson	10	2
40	Master	20	3
50	Architect	20	3
60	Administrator	30	3
70	Superuser	30	3

4.3.2 Dizi Yapısında Nesne Tanımlanması

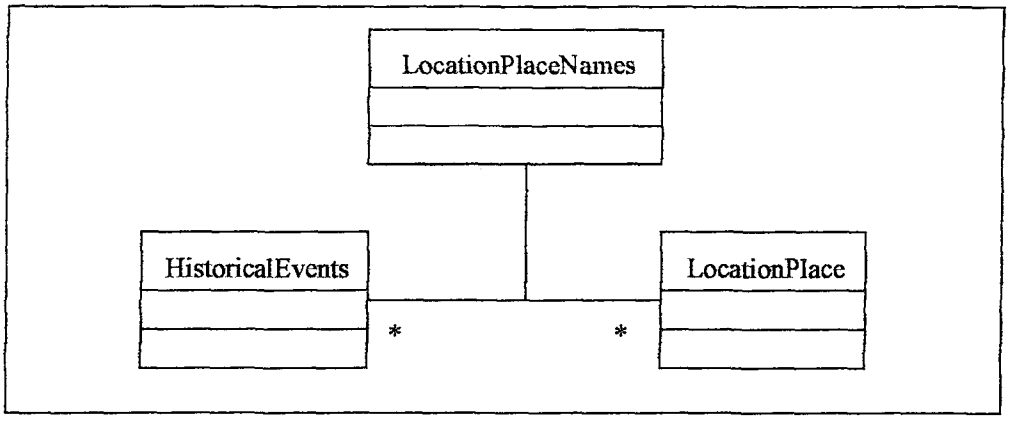
Daha önce de belirtildiği üzere nesne ilişkisel veri tabanı yönetim sisteminde (ORDBMS) kullanılmak SQL3 diğer adıyla SQL99 geliştirilmiştir. Literatüre göre SQL99, nesneler arasındaki n-m ilişkisini sağlamak üzere ARRAY yapısının aşağıda gösterildiği şekilde

kullanılmasına olanak sağlamaktadır.[4] Ancak benzer yapıyı tarihi olayları içeren "historcicalevent" nesnesi ile yerleşim yerlerini içeren "locationplace" nesnesi arasında uyguladığımızda DB2'nun ARRAY yapısını desteklemediği görülmüştür.

```
CREATE TYPE DepartmentType AS (
 departmentNo ThreeDigitNo,
 departmentName VARCHAR(25),
 budget PositiveReal,
 employeeNo SMALLINT,
 worksOnProduct REF(ProductType) ARRAY[20],
 REF IS SYSTEM GENERATED);

CREATE TYPE ProductType AS (
productNo INTEGER,
productName VARCHAR(25),
producttype VARCHAR(25),
costPerUnit PositiveReal,
netPrice PosittiveReal,
isProcessedBy REF(DepartmentType) ARRAY[5],
REF IS SYSTEM GENERATED);
```

Bir tarihi olay birden fazla yerleşim yerinde geçebileceği için "historicalevent" UDT'si için bir dizi tanımına ihtiyaç duyulmaktaydı. Aynı ilişki tipi HIS veri tabanında bir esere ait hasarlarda, onarımlarda vb. birçok yapıda yer almaktadır. Bu sorunu iki yöntemle çözülebilir; Birinci yöntemde ilişkisel veri tabanında olduğu gibi sınıflar arası ilişkiyi tutan yeni bir yapısal tipin oluşturulur. Örneğin bir tarihi olay, birden fazla yerleşim birimini etkileyebildiği gibi bir yerleşim biriminde de birden fazla tarihi olay meydana gelmiş olabilir. Bu durumda Şekil 4.11'de de görüldüğü gibi "LocationPlaceNames" adında bir bağlantı sınıfı tanımlanır. Bu sınıfta "HistoricalEvents" ve "LocationPlace" sınıflarının referans bilgileri tutulmaktadır.



Şekil 4.11 İki sınıf arasındaki bağlantı sınıfı

İkinci yöntemde ise dizi mantığıyla aynı yapısal tipi referans gösteren yeni bir yapısal tip (sınıf) tanımlanır. Örneğin "locaitonplacearray" yapısal tipi aslında "locationplace" yapısal tipine üç tane referans veren bir dizi mantığında aşağıdaki SQL komutuyla oluşturulur.

```
CREATE TYPE locationplace AS
( name char(50), population varchar(50) )
MODE DB2SQL

CREATE TYPE locationplacearray AS
( lp1 REF(locationplace),
  lp2 REF(locationplace),
  lp3 REF(locationplace) )
MODE DB2SQL
```

Daha sonra tarihi olayı içeren "historicalevent" tanımlı tipi aşağıdaki SQL komutuyla oluşturulur. Böylece bir tarih olay için üç farklı yerleşim yerinin bilgisi tutulabilmektedir. Tarihi olay başına tutulması istenen yerleşim yerinin sayısı arttırılmak istendiğinde ise ALTER SQL komutu ile hem dizi olarak tanımlanan tip hem de bu tipe ait tablo değiştirilebilinir. Veri tabanında yapılan bu değişiklik önceden girilen verileri de etkilemeyeceği için esneklikte sağlanmış olacaktır.

```
CREATE TYPE historicalevent AS
( name varchar(70),
  startdate date,
  finishdate date,
  startcentury decimal(4),
  finishcentury decimal(4),
  result varchar(200),
  effect varchar(200),
  locationplacenames REF(locationplacearray) )
  MODE DB2SQL
```

HIS veri tabanı tasarımı yapılırken ikinci yöntem kullanılmıştır. Çünkü birinci yöntemde bir tarihi olay üç yerleşim birimini etkilerse "LocationPlaceNames" tablosuna üç kayıt girilmesi gerekmektedir. İkinci yöntemde ise "LocationPlaceArray" sınıfı üç yerleşim birimine referans verdiği için bir kayıt yeterli olmaktadır. Böylece veri tabanında fazla kayıt sayısından dolayı artış önlenmiş olur. Ancak bu yöntemde de dizi mantığında olduğu gibi boyut tasarım aşamasında belirlendiği için uygulamada bu sınır dışına çıkılamamaktadır.

HIS veri tabanında tarihi olay için girilen verilere göre yerleşim yerlerinin isimlerine ise aşağıdaki sorgulama cümlesi ile kolayca ulaşılabilmektedir. Görüldüğü üzere "historicaleventtable" isimli tabloda "locationplacenames" özelliğiyle dizi yapısında tanımlanan "locationplacearray" tipine oradan da "lp1" özelliği üzerinden "locationplace" olarak tanımlanan yerleşim yerinin adına ulaşılmaktadır.

```
SELECT h.name, h.locationplanames->lp1->name
FROM historicaleventtable h
```

4.3.3 Sınıf Özelliklerinin Değiştirilmesi

DB2'da tanımlanan nesne yönelimli modelde sınıfa karşılık gelen yapısal tip oluşturulduktan sonra bu tipe yeni bir özelliğin eklenmesi ilişkisel modeldeki gibi kolayca gerçekleşmemektedir. İlişkisel modelde tablonun herhangi bir kayda sahip olup olmadığına bakmaksızın veri tabanı yöneticisi istediği zaman "ALTER" SQL komutuyla tablodan bir özellik çıkarabilir veya yeni bir özellik ekleyebilir. Hatta yeni özellik eklendiğinde tabloya daha önce girilen kayıtlar üzerinde düzeltme yapmaya gerek yoktur. Ancak yapısal tiplere ait tablolar üzerinde özellikler için bir değişiklik yapılamamaktadır. Çünkü bu tablolar daha önce de anlatıldığı üzere yapısal tiplere bağlıdırlar. Bu nedenle ilişkisel modeldekine benzer şekilde aşağıda gösterilen "ALTER" SQL notasyonuyla yapısal tip üzerinde değişiklik gerçekleştirilebilinir.

DB2'da oluşturulan yapısal tipe yeni özellik eklemek için kullanılan SQL komutu;

ALTER TYPE <tip-ismi>
ADD ATTRIBUTE <özellik-ismi>

DB2'da oluşturulan yapısal tipte var olan bir özelliği silmek için kullanılan SQL komutu;

ALTER TYPE <tip-ismi>
DROP ATTRIBUTE <özellik-ismi>

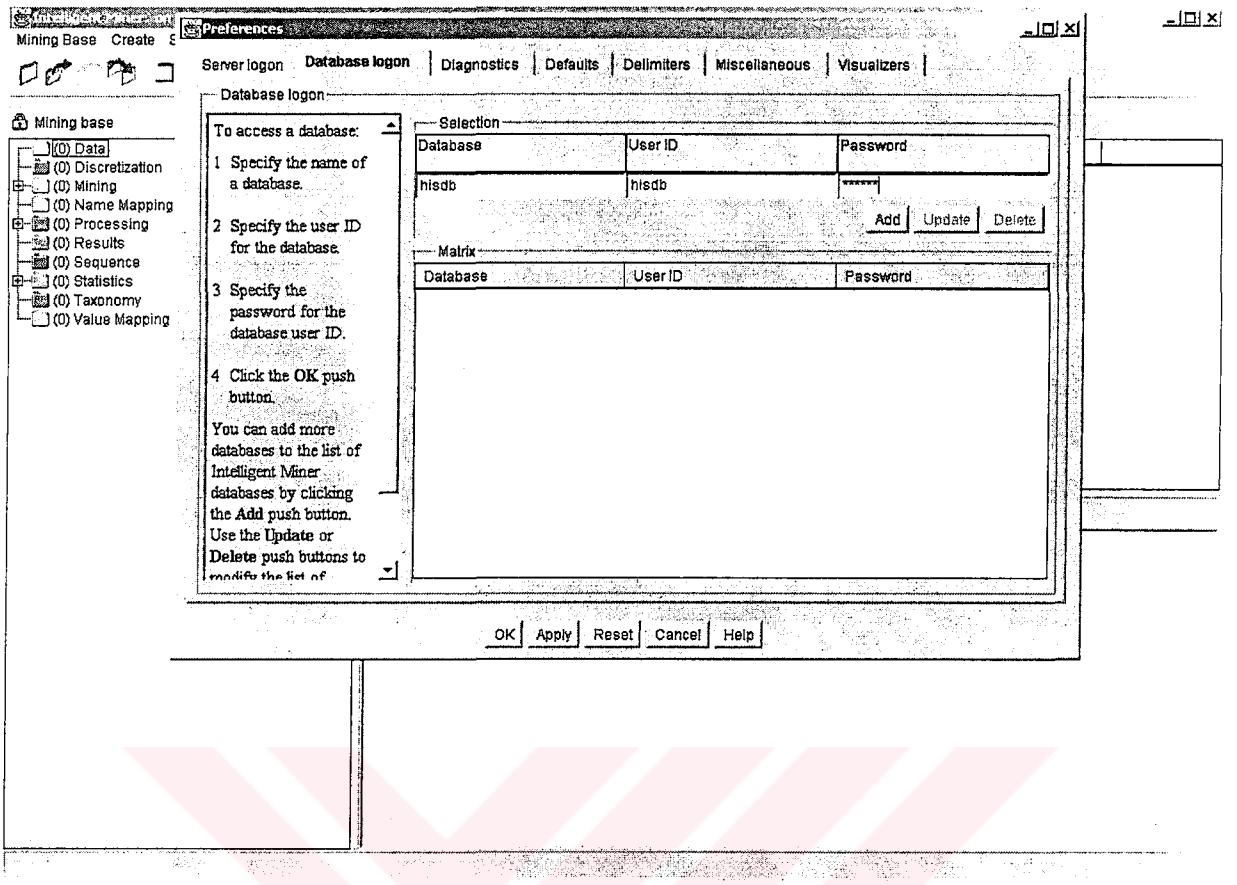
Yapısal tip üzerinde yapılması istenen değişiklikte dikkat edilmesi gereken en önemli unsur ise bu yapısal tipe veya bu yapısal tipin alt sınıflarına ait herhangi bir tablonun veri tabanında yer almaması gerektiğidir.[12] Aksi takdirde değişikliğin yapılmasına izin verilmemektedir. Bu nedenle mevcut bir yapısal tip üzerinde değişiklik yaparken aşağıdaki sıra takip edilmelidir.

1. Yapısal tipe ait mevcut tablo veri tabanında kaldırılması
2. Yapısal tip üzerinde istenen değişiklik ALTER komutuyla gerçekleştirilmesi
3. Yapısal tipe ait tablo yeniden oluşturulması

Benzer şekilde HIS veri tabanı oluşturulduktan bazı yapısal tiplerin özellikleri değiştirilmek istendiği için bu yapısal tiplere ait tablolar veri tabanından silindi. Özellik değişiklikleri yapıldıktan sonra tablolar yeniden oluşturuldu. Ancak hem silinen tablolarla beraber girilen kayıtların daha sonra tekrardan veri tabanına aktarılması hem de sadece üzerinde değişiklik yapılan yapısal tipe ait tablonun değil o tabloya bağımlı olan diğer tüm tabloların silinip yeniden oluşturulması oldukça fazla zaman kaybına neden olmuştur. Bu da göstermektedir ki DB2'da ilişkisel nesne modeli kullanılmadan önce veri tabanının yapısı hiçbir değişikliğe gerek kalmayacak şekilde en ince ayrıntısına kadar tasarlanmalıdır.

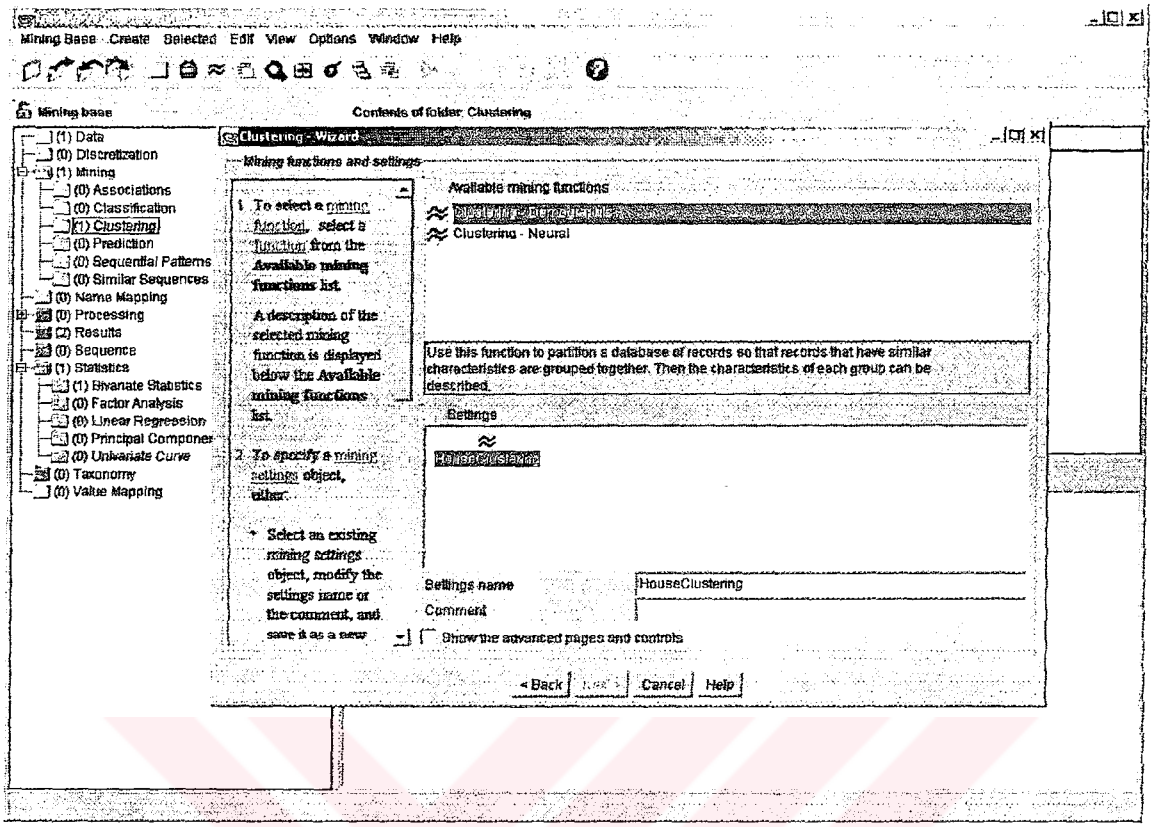
4.4 HIS'de Veri Madenciliği Uygulaması

HIS nesne ilişkisel veri tabanında kayıt sayısı az olduğu için veri madenciliği için sadece Intelligent Miner for Data (IM Data) aracının kullanılması uygun görülmüştür. Şekil 4.12'de görülen ara birimle IM Data'da çalışma alanı olan HIS veri tabanına bağlantı için gerekli olan düzenlemeler yapılmıştır. Bağlanılacak sunucunun tanımı "Server Logon", veri tabanının tanımı "Database Logon" sekmelerinde yapılır. "Diagnostics" sekmesinde çalışılacak tablolar, "Visualizers" sekmesinde ise sonuçların nasıl görüntüleneceği belirlenir. Örneğin HIS veri tabanında ev tablosu üzerinde kümeleme analizi yapılmak istendiğinde "Visaulizers"da sonuç tipi olarak uygulanan kümeleme tekniğine göre seçim yapılır.



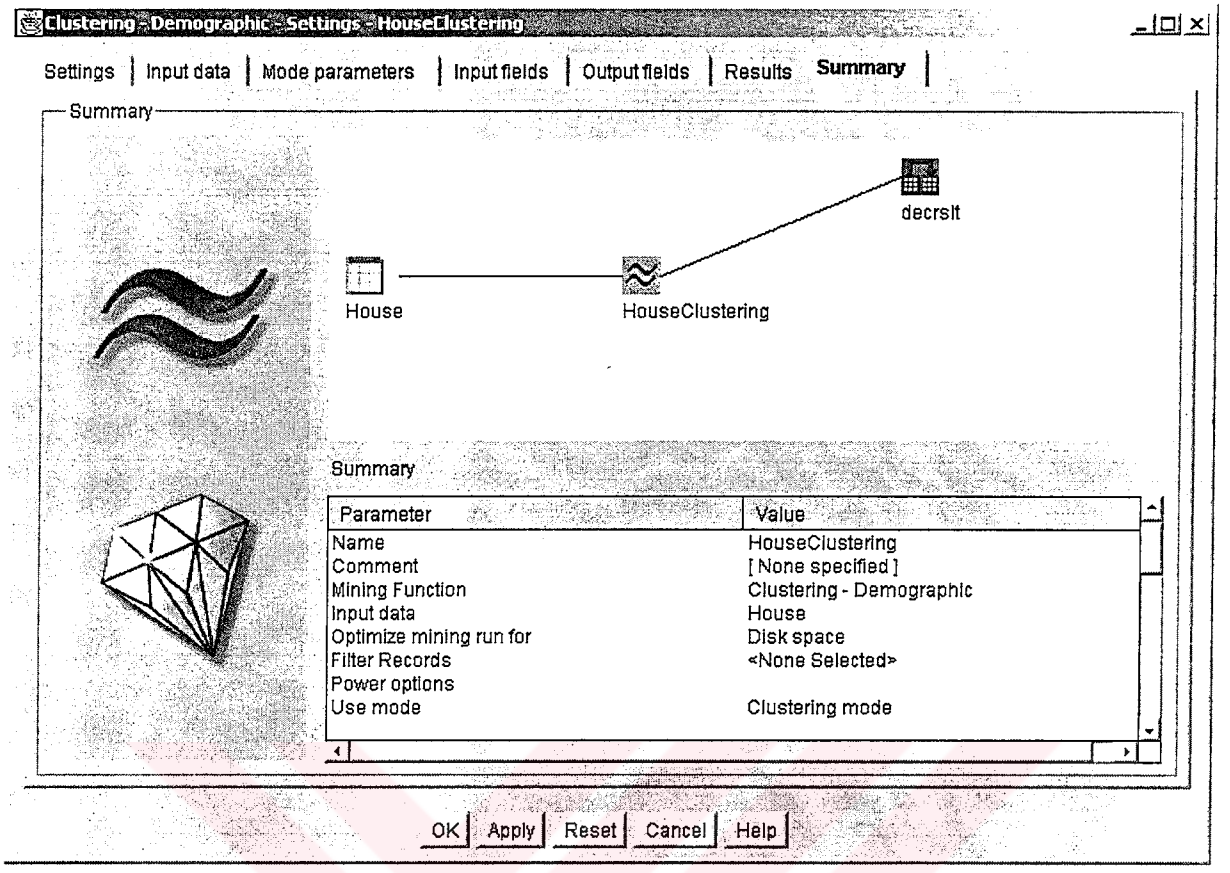
Şekil 4.12 Intelligent Miner for Data için çalışma alanı özelliklerinin belirlenmesi

HIS üzerinde uygulanan modellerden biri evlerin durumuna göre gruplandırılmasıdır. Bu amaçla IM Data'da demografik kümeleme analizi kullanılmasına karar verilmiştir. Öncelikle veri temizliği yapılmıştır. Veri temizliği yapılırken kayıt sayısı az olduğu için eksik veri içeren kayıtlar silinmemiştir. Eksik verilere karşılık gelen alanlar için "0" değeri atanmıştır. Model için gerekli tablolar ve modelde kullanılan alanlar seçilirerek "Data Create" işlemi tamamlanır. Şekil 4.13'de görülen ara birimde uygulanacak veri madenciliği tekniği seçilir. Bu örnekte "Clustering – Demographic" algoritması seçilerek bu modele "HouseClustering" adı verilmiştir. Bir sonraki aşamada bu modelde kullanılacak veri (data House) seçilip model için giriş ve çıkış alanları seçilir. Bir veri madenciliği modeli için sadece bir veri sınıfı seçilebildiği için nesneye ilişkisel veri tabanı modeli ile kurulan sınıf bağıntıları kullanılamamaktadır.



Şekil 4.13 Veri madenciliği metodunun oluşturulması

Oluşturulan veri madenciliği modeli kümeleme olduğu için ana ara birimin sol tarafında "Mining -> Clustering" altında saklanır. Bu model çalıştırıldığında elde edilen sonuç, en başta belirlenen çıkış görüntüleme verisi üzerinde saklanır. Şekil 4.14'de veri, model ve çıktı arasındaki ilişki gösterilmiştir. "house" kullanılan veri, "houseclustering" kullanılan model, "decrslt" ise model veri üzerinde çalıştıktan sonra elde edilen sonuçları gösteren sembollerdir.



Şekil 4.14 Veri, model ve çıktı arasındaki ilişki

IM Data'da model için tanımlanan veri tabanı belirlendikten sonra oluşturulacak veri için tablo seçimi kullanıcıya bırakılıyor. IM Data ile DB2 arasındaki başta kurulan bağlantı sayesinde seçilen tablodaki tüm alanlar ve değişken tipleri otomatik olarak IM Data'ya aktarılır. Ancak bu aşamada nesne ilişkisel veri tabanı modelden kaynaklanan iki problemle karşılaşılmaktadır;

1. Bir sınıf tablosundaki özellik başka bir sınıfa referansa gösteriyorsa IM Data, bu değişken tipini tanımadığı için modelleme ya da istatistiki hesaplama sürecinde bu değişkenleri kullanamıyor.

2. Veri girişi olarak kullanılan sınıf tablosu bir alt sınıf ise kalıtım özelliğiyle üstündeki sınıflara ait tüm özelliklere erişilebilinmektedir. Ancak üst sınıf, alt sınıflardaki özellikleri göremediği için aynı seviyedeki sınıflarda aynı veri madenciliği modeli her sınıf için ayrı ayrı çalıştırılması gerekmektedir.

Bu nedenlerden ötürü nesne ilişkisel veri tabanı modeli üzerinde uygulanan veri madenciliği, ilişkisel veri tabanında uygulanan madencilik metotlarından ileriye gidememiştir.

5. SONUÇLAR VE ÖNERİLER

HIS'in oluşturulmasında karşılaşılan en büyük sorunlardan biri veri tabanı yapısı oluşturulduktan sonra değişiklik yapılmasıdır. Veri tabanı, nesne ilişkisel olduğu için yapılması istenen en küçük bir değişiklik için bile neredeyse tüm veri tabanının yeniden oluşturulmak zorundadır. Bu nedenle uygulamaya geçmeden önce tasarım aşamasında veri tabanının en ufak detayına kadar tasarlanması gerekmektedir.

HIS yapısında yapıların sadece tarihi olaylara ilişkisi belirlenmeye çalışılmıştır. Ancak araştırmacılar, tarihi yapıların oluşumunda içinde bulundukları toplumun sosyal ve ekonomik değerlerinin de önemli olduğunu düşünmektedirler. Bu doğrultuda gelecek çalışmalarda veri tabanı yapısı bu bilgileri de tutacak şekilde geliştirilip bilgi keşfi için daha fazla parametreden faydalanılması sağlanabilir.

Çalışma süresi boyunca sadece belli bir bölgenin (Kütahya'nın Gediz bölgesine ait) yapı bilgilerine ulaşıldığı için bilgi keşfi süreci sonucunda coğrafi konum farklılıklarının yapılar üzerindeki etkisi ortaya konamamıştır. Çalışmanın devam ettirilmesi halinde farklı bölgelerin tarihi yapı verileri de sisteme girilerek coğrafi açıdan bir değerlendirme yapılması da mümkün olacaktır.

Bu çalışmanın amacı tarihi yapılar için nesne ilişkisel model oluşturarak bilgi keşfi sürecini geliştirmektir. Uygulamanın geliştirilmesinde kullanılan nesne ilişkisel veri tabanı yönetim sistemi DB2 ve veri madenciliği aracı olan IBM Intelligent Miner araçlarının sunduğu ara birimler yeterli olmuştur. Ancak bu çalışmanın araştırma alanı olan mimari restorasyon kullanıcıları tarafından kullanılması oldukça zordur. Bu nedenle ileride son kullanıcıların da HIS'den kolaylıkla faydalanabilinmesi için WebSphere gibi DB2 ile uyumlu olan bir yazılım aracı ile kullanıcı ara birimleri geliştirilebilinir.

Her ne kadar nesne ilişkisel veri tabanı modeli, nesneye yönelik modelin özelliklerin çoğunu içermektedir. Ancak DB2'da bir alt sınıfın birden fazla üst sınıfa sahip olacak şekilde karmaşık yapıda bir yapısal tipin (sınıfın) tanımlanamamaktadır. Nesne ilişkisel veri tabanı modelinin bu yapıyı destekleyecek şekilde geliştirilmesi için bir çalışma yapılabilir.

Saleem, M.A. ve Hameed, J.'nin yaptığı çalışmaya göre nesne ilişkisel veri tabanı sistemlerinde sayıca çok fazla nesne olması halinde veri madenciliği uygulandığında ilişkisel veri tabanı sistemlerine göre performans kaybı olduğu görülmüştür. Ancak HIS veri tabanında nesne sınıfı fazla olmasına rağmen kayıt sayısı çok olmadığı için bir performans kaybına

rastlanmamıştır. İleride bu konuda yapılacak çalışmalarda kayıt sayısının fazla tutularak performansın ne kadar etkilendiği ölçülebilir.

Tarihi yerleşimlerdeki yapıların çok farklı özelliklere sahip olması nedeniyle kullanıcıya yeni veri tipleri tanımlamasına olanak sağladığı için nesneye yönelik veri tabanı modelinin uygulanmasının faydalı olduğu gözlenmiştir. Ancak tarihi bilgi sistemi, bir perakende sektöründe müşterinin gösterdiği kadar davranışsal özelliklere sahip olmadığı için nesneye yönelik modelin sarma özelliğinden hiç faydalanılamamıştır. Bir uygulama için veri tabanı modelini seçerken uygulamanın çalışma alanı da dikkate alınmalıdır.

IBM Intelligent Miner, nesne ilişkisel veri tabanı modelini tanımaktadır. Ancak nesne ilişkisel veri tabanını, ilişkisel veri tabanından ayıran kalıtım, sarmal gibi özelliklerinden faydanılamamaktadır. İleride bu özelliklerin kullanılabildiği ve bir yapı tipi içinde başka bir kullanıcı yapı tipini (UDT) destekleyen veri madenciliği uygulamaları üzerine yapılacak çalışmaların, nesne ilişkisel veri tabanı modelinin kullanımını arttıracak unsurlardan biri olarak düşünülmektedir.

KAYNAKLAR

Akpınar, H. (Nisan 2000), "Veri Tabanlarında Bilgi Keşfi ve Veri Madenciliği", İ.Ü. İşletme Fakültesi Dergisi, C:29, S: 1.

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

Berry, M. ve Linoff, G. (1997), "Data Mining Techniques", John Wiley and Sons, USA.

Blaha, M.R. (1998), "A Manager's Guide to Database Technology", Prentice Hall, New Jersey.

Brooks, P. (1997), "Data Mining Today", DBMS, Miller Freeman, Inc.

Brown, W.A. (1991), "Object-Oriented Databases Applications in Software Engineering", McGraw-Hill, London.

Buharalı, A. (Aralık 2002), "The Role of Information Technologies on Historical Sites – New Methodologies", Re-evaluating Urban Heritage Documentation, Rehabilitation and Preservation of Architecture and History, Aralık 11-14, 2002, Beyrut.

Chaudhri, B. A. ve Zicari, R. (2001), "Succeding with Object Databases a Practical Look at Today's Implementations with Java and XML", Wiley Computer Publishing, New York.

Eaglestone, B. ve Ridley, M. (1998), "Object Databases: An Introduction", McGraw-Hill.

Elmasri, R. ve Navathe, B. S. (2000), "Fundamentals of Database Systems", 3rd Edition, Addison-Wesley.

Famili, A., Shen W., Weber, R. ve Simoudis, E. (1997), "Data Preprocessing and Intelligent Data Analysis", Intelligent Data Analysis 1, Elsevier Science B.V.

Fayyad, U. ve Stolorz, P. (1997), "Data Mining and KDD: Promise and Challenges", Future Generation Computer Systems 13, Elsevier Science B.V.

IBM (1999), "Using the Intelligent Miner for Data", International Business Machines Corporation, USA.

IBM (2002),"DB2 Intelligent Miner Modeling Administration and Programming Version 8.1", International Business Machines Corporation, USA.

Kalıpsız, O. (1998), "Bilgisayar Veri Tabanı Sistemleri", Der Yayınları, İstanbul.

Kantardzic, M. (2001), "Data Mining Concepts, Models, Methods, and Algorithms",IEEE Pres.

Kauffman, J. (2001), "Beginning SQL Programming", Wrex Press Limited.

Marcos, E., Vela, B., Cavero, J.M. ve Cáceres, P. (2001), "Aggregation and Composition in Object – Relational Database Design", Advances in Databases and Information Systems, 5th East-European Conference ADBIS' 2001, Eylül 25-28, 2001, Vilnius, Litvanya.

Muller, R.J. (1999), "Database Design for Smarties: Using UML for Data Modeling", Morgan Kaufmann Publishers, San Francisco.

Vahaplar, A. ve İnceoğlu, M.M. (Kasım 2001), "Veri Madenciliği ve Elektronik Ticaret", Türkiye'de İnternet Konferansları VII, İstanbul.

INTERNET KAYNAKLARI

[1] http://www.metu.edu.tr/~entary/giswp/vol1/cp4/4-3.gif

[2] http://www.andrew.cmu.edu/user/clord/Portfolio/Object-Oriented%20Database%20Survey.pdf

[3] http://www.comptechdoc.org/independent/database/basicdb/dataobject.html

[4] http://www-inf.fh-reutlingen.de/forschung/laux/SQL1999ooAspects.pdf

[5] http://idari.cu.edu.tr/sempozyum/bil38.htm

[6] http://www.eco.utexas.edu/~norman/BUS.FOR/ course.mat/Alex/#5

[7] http://www.twocrows.com

[8] http://www.danismend.com/konular/fihrist/bilgi_olap1.htm

[9] http://www.on-lineanalitikislemecouncil.org/research/whtpapco.html

[10] http://www-106.ibm.com/developerworks/rational/library/769.html

[11] http://zion.comu.edu.tr/~msahin/ppts/umlseminer.ppt

[12] http://webdocs.caspur.it/ibm/web/udb-6.1/db2d0/creatdb.htm

[13] http://citeseer.ist.psu.edu/cache/papers/cs/26771/http:zSzzSzmoazzamsaleem.s5.comzSz.zSzpaperszSzintegration.pdf/integration-of-data-mining.pdf

EKLER

Ek 1 Türkiye'de Veri Madenciliği Hizmeti Sunan Firmalardan Örnekler

Firma Adı	Kullanılan Uygulama	Uygulamanın Başlıca Özellikleri
Kratis	Kxen Analytical Framework; Similarity Systems	• Kxen • Veri hazırlama, dönüştürme • Sınıflama, birliktelik, regresyon analizi, örüntü tanıma, zaman serileri
SPSS Veri Madenciliği Çöz.	Clementine	• Integral Solutions LTD • Sinir ağları, sınıflama, karar ağaçları, görsel araçlar
Microsoft Türkiye	MS Analysis Services; ProClarity 4.0	• Microsoft • SQL Server 2000 ile entegrasyon • Birliktelik, regresyon,sıralı örüntü analizleri, gruplama, istatistiki metodlar
Siemens Business Services	SAP Business Intellience	• SAP • SAP ürün olmayan veriler üzerinde de çalışabilme • Birliktelik, regresyon,sıralı örüntü analizleri, gruplama, istatistiki metodlar, veri hazırlama, raporlama
IBM Türk	Intelligent Miner	• IBM • Birliktelik, regresyon,sıralı örüntü analizleri, gruplama, sinir ağları algoritmaları, istatistiki metodlar, veri hazırlama, veri görselleştirme araçları • DB2 ile sıkı entegrasyon
Nexum Boğaziçi	MIS DeltaMiner	• MISAG • İstatistiki analizler, tahmin, sınıflandırma, ABC analizi

Ek 2 HIS Kullanım Senaryoları Dokümantasyonu

Sisteme Giriş

Kısa Açıklama	Yetkili kullanıcı veya sistem yöneticisi, kullanıcı adı ve şifreyle sisteme giriş yapar.
Aktörler	Yetkili kullanıcı, sistem yöneticisi
Ön Koşullar	Yetkili kullanıcı veya sistem yöneticisi, tarayıcıda (browser) sisteme giriş sayfasını açmalı. Sistem, kullanıcı adının ve şifrenin doğruluğunu kontrol etmeli.
Ana Akış	Yetkili kullanıcı veya sistem yöneticisi, ilgili web sayfasında kullanıcı adını ve en az 6 karakterli şifresini girdikten sonrsa kontrolü yapan fonksiyonu "Tamam" tuşuna basarak çağırır. Sistem, kullanıcı adını ve şifreyi kontrol eder. Eğer doğruysa ilgili sayfaya girişi sağlar; yanlış ise hata mesajını kullanıcıya gösterir.
Alternatif Akış	Yok
Artçı Koşullar	Senaryo başarılı olduğunda yetkili kullanıcı veya sistem yöneticisi için ana menü oluşturulur. Aksi durumda sisteme giriş yapılamaz.

Kayıt

Kısa Açıklama	Yetkili kullanıcı, HIS veri tabanında yer almayan verilerin girişini yapar.
Aktörler	Yetkili kullanıcı
Ön Koşullar	Yetkili kullanıcı, kaydetmek istediği verilerin ait olduğu sınıfa ait web sayfasına giriş için seçim yapmalı. Bir kayda ait zorunlu alanlara ait değerler mutlaka girilmeli.
Ana Akış	Yetkili kullanıcı, kayıt sayfasında kaydın ait olduğu sınıfı seçer. Sistem, bu sınıfın özelliklerinin neler olduğunu veri tabanından çekerek giriş ara birimini oluşturur. Yetkili kullanıcı, sınıfın zorunlu ve/veya zorunlu olmayan özellikleri için değerleri girip veri tabanında kayıt işlemini gerçekleştirecek olan fonksiyonu çağırmak için "Tamam" tuşuna basar. Sistem, girilen değerlerin tiplerini, uzunluklarını vb. kriterleri kontrol eder. Hatalı veri varsa hata mesajını kullanıcıya gösterir yoksa kaydı gerçekleştirir. Kayıt yapıldıktan sonra kaydın başarıyla yapıldığı yetkili kullanıcıya mesaj yoluyla bildirilir.
Alternatif Akış	Yetkili kullanıcı, girdiği değerlerin tamamını değiştirmek için "Temizle" tuşuna basarak temizleme fonksiyonunu çağırır. Sistem, web sayfasındaki tüm değerleri silerek yeniden girilmesine olanak sağlar. Yetkili kullanıcı, eserin ait olduğı sınıfı yanlış seçmiş olduğunu düşünerek yeniden sınıf seçebilmek için "Geri" tuşuna basar. Sistem, sınıfların olduğu sayfaya geri döner ve giriş ekranını yeniden oluşturur.
Artçı Koşullar	Senaryo başarılı olduğunda veri tabanına kayıt yapılır.

Arama

Kısa Açıklama	Yetkili kullanıcı veya kullanıcı, bilgisine ulaşmak istediği kaydı HIS veri tabanında arar.
Aktörler	Yetkili kullanıcı, kullanıcı
Ön Koşullar	Ana menüden web sayfasına ulaşmak için "Arama" seçilmeli. Sunulan seçeneklerden arama kriterleri belirlenmeli. Sistem, kriterlere göre veri tabanında sorgulama yapıp uygun kayıtları onarlık listeler halinde sunmalı.
Ana Akış	Yetkili kullanıcı veya kullanıcı, arama ara birimine ulaştıktan sonra arayacağı kaydın özelliklerine göre istediği değerleri ve/veya değer aralıklarını girer. Sistem, seçilen özelliklere ve bu özelliklerin alabileceği değerlere göre dinamik bir sorgulama (SQL) hazırlayarak sorgulamayı başlatır. Yapılan sorgulama sonucunda uygun olan kayıtlar yetkili kullanıcıya/kullanıcıya gösterilmek onarlık kayıtlar halinde özetlenerek listelenir. Yetkili kullanıcı/kullanıcı, bu listeler arasında dolaşbilir veya listeden seçeceği kayda ait daha ayrıntılı bilgi edinebilir. Sistem, seçilen kayda ait tüm bilgileri detaylı olarak yeni bir ara birimde gösterir.
Alternatif Akış	Yetkili kullanıcı/kullanıcı, seçtiği özellikleri ve girdiği değerlerin tamamını değiştirmek için "Temizle" tuşuna basarak temizleme fonksiyonunu çağırır. Sistem, web sayfasında kullanıcı tarafından girilen tüm değerleri silerek yeniden tanımlamaya olanak sağlar. Yetkili kullanıcı/kullanıcı, yeni bir kayıt aramak için "Geri" tuşuna basarak tüm süreci tekrarlayabilir. Hiçbir kriter tanımlanmadan sorgulama fonksiyonu çağrılırsa sistem yetkili kullanıcıya/kullanıcıya uyarı mesajı verir.
Artçı Koşullar	Senaryo başarılı olduğunda yetkili kullanıcı/kullanıcı aradığı kayda en kısa sürede esnek bir sorgulama imkanıyla kavuşur.

Güncelleme

Kısa Açıklama	Yetkili kullanıcı, HIS veri tabanında bulunan bir kaydın bilgilerini değiştirerek güncelleme yapar.
Aktörler	Yetkili kullanıcı
Ön Koşullar	Yetkili kullanıcı, güncelleme web sayfasına ulaşmak için menüden seçim yapmalı. Güncellenmek istenen kayda ulaşmak için "Arama" senaryosu kullanılmalı.
Ana Akış	Yetkili kullanıcı, ana menüden güncelleme ekranını seçtikten sonra "Arama" senaryosundaki arama işlemi gerçekleştirilmek üzere sistem tarafından yetkili kullanıcıya gösterilir. Güncellenmek istenen kayıt sorgulama sonucu elde edilen listeden yetkili kullanıcı tarafından seçilir. Sistem, bu kayda ait tüm değerleri ara birimde görüntüler. Bu ara birim, bilgilerin değiştirilmesine izin verilecek şekilde tasarlanmıştır. Yetkili kullanıcı, gerekli değişiklikleri yaparak "Tamam" tuşuna basarak güncelleme fonksiyonu çağrılır. Sistem, "Kayıt" senaryosunda olduğu gibi ilgili alanların kontrollerini yapar. Girilen değerler kriterlere uygunsa veri tabanında göncelleme yapılır ve güncelleştirmenin başarıyla gerçekleştiği mesajı yetkili kullanıcıya sistem tarafından gösterilir. Aksi durumda hata mesajı gösterilir.
Alternatif Akış	Yetkili kullanıcı, girdiği değerlerin tamamını değiştirmek için "Temizle" tuşuna basarak temizleme fonksiyonunu çağırır. Sistem, web sayfasında kullanıcı tarafından girilen tüm değerleri silerek veri tabanındaki orjinal değerleri yerleştirir. Yetkili kullanıcı, yeni bir kayıt güncellemek için "Geri" tuşuna basarak tüm süreci tekrarlayabilir.
Artçı Koşullar	Senaryo başarılı olduğunda veri tabanındaki kayıt güncellenmiş olur.

Raporlama

Kısa Açıklama	Yetkili kullanıcı veya kullanıcı, HIS veri tabanındaki kayıtlara ait seçtiği kriter ve bu kriterlerin değerlerine göre basılmak üzere rapor alabilir.
Aktörler	Yetkili kullanıcı, kullanıcı
Ön Koşullar	Ana menüden web sayfasına ulaşmak için "Raporlama" seçilmeli. Sunulan seçeneklerden raporu oluşturacak dinamik sorgulama için kriterler belirlenmeli.
Ana Akış	Yetkili kullanıcı veya kullanıcı, "Arama" senaryosundakine benzer bir kriter belirleme ara birimini kullanarak istenen değer ve/veya değer aralıklarını girer. Sistem, seçilen özelliklere ve bu özelliklerin alabileceği değerlere göre dinamik bir sorgulama (SQL) hazırlayarak sorgulamayı başlatır. Sistem, sorgu sonucu elde edieln uygun kayıtları ayrı bir pencerede yazıcıdan basılacak formatta yetkili kullanıcıya/kullanıcıya gösterir. Yetkili kullanıcı/kullanıcı, yeni açılan pencerede "Yazdır" tuşuna basarak yazıcısından raporun tamamını alabilir.
Alternatif Akış	Yetkili kullanıcı/kullanıcı, önceden tasarlanan standart raporlardan birini seçerek sorgulama fonksiyonunu çağırır. Böylece sistem, dinamik bir sorgulama oluşturmadan seçilen rapor tipine ait sorgulamayı çalıştırıp sonuçları rapor halinde sunar. Yetkili kullanıcı/kullanıcı seçtiği özellikleri ve girdiği değerlerin tamamını değiştirmek için "Temizle" tuşuna basarak temizleme fonksiyonunu çağırır. Sistem, web sayfasında kullanıcı tarafından girilen tüm değerleri silerek yeniden tanımlamaya olanak sağlar. Yetkili kullanıcı/kullanıcı, yeni bir rapor oluşturmak için "Geri" tuşuna basarak tüm süreci tekrarlayabilir. Hiçbir kriter tanımlanmadan ya da standart bir rapor seçmeden sorgulama yapılmak istenirse sistem kullanıcıya uyarı mesajı verir.
Artçı Koşullar	Senaryo başarılı olduğunda veri tabanındaki kayıtlar rapor halinde basılacak şekilde düzenlenir.

Sınıf Tanımlama

Kısa Açıklama	Yetkili kullanıcı, HIS veri tabanında olmayan yeni bir alt sınıf oluşturur.
Aktörler	Yetkili kullanıcı
Ön Koşullar	Ana menüden web sayfasına ulaşmak için "Sınıf Tanımlama" seçilmeli. Yeni sınıfın ait olacağı üst sınıf listeden seçilmeli.
Ana Akış	Yetkili kullanıcı, ana menüden sınıf tanımlamayı seçtikten sonra sistem HIS veri tabanında bulunan üst (kök) sınıfları yetkili kullanıcıya listeler. Yetkili kullanıcı, bu hiyerarşik yapıya göre hangi sınıfın altında yeni bir sınıf oluşturmak istiyorsa o sınıfı seçer. Sistem, bu üst sınıfın özelliklerini ara birimde gösterir ve yetkili kullanıcı yeni sınıfı tanımlaması için ara birimi oluşturur. Yetkili kullanıcı, yeni oluşturacağı sınıfın adını ve özelliklerini tanımlayıp "Tamam" tuşuna basarak sınıf oluşturma fonksiyonunu çağırır. Sistem, girilen verilere ait gerekli kontrolleri yaptıktan sonra yen, sınıfı oluşturup başarı mesajını yetkili kullanıcıya gösterir. Sistem, yetkili kullanıcının girişlerinde hata varsa hata mesajı verir.
Alternatif Akış	Yetkili kullanıcı, yeni tanımlamak istediği sınıfa ait verilerin tamamını değiştirmek için "Temizle" tuşuna basarak temizleme fonksiyonunu çağırır. Sistem, web sayfasında kullanıcı tarafından girilen tüm değerleri silerek yeniden tanımlamaya olanak sağlar. Yetkili kullanıcı, üst sınıfı yanlış seçtiğini düşünerek yeniden sınıf seçmek için "Geri" tuşuna basarak tüm süreci tekrarlayabilir. Yetkili kullanıcı, sınıfın ismini girmeden "Tama" tuşuna basarsa yetkili kullanıcıya uyarı mesajı verilir.
Artçı Koşullar	Senaryo başarılı olduğunda veri tabanı yapısında yeni bir sınıf tanımlanmış olur.

Yetkilendirme

Kısa Açıklama	Kullanıcı, yetkili kullanıcı olmak için sistem yöneticisinden yetki ister. Sistem yöneticisi, gelen talebi değerlendirerek yetkilendirme yapar.
Aktörler	Kullanıcı, sistem yöneticisi
Ön Koşullar	Kullanıcı, gerekli kişisel bilgilerini doldurarak Web sayfası aracılığıyla sistem yönetcisine göndermeli.
Ana Akış	Kullanıcı, ana menüden "Yetki Talebi"ni seçtikten sonra ara birimde kendisiyle ilgili kişisel bilgilere ait alanları doldurup "Tamam" tuşuna basar. Sistem, alan kontrollerini yapar. Hatalı bir veri varsa hata mesajını kullanıcıya iletir. Herhangi bir hata yoksa başarı mesajı kullanıcıya iletilirken kullanıcının talebi kişisel bilgileriyle beraber sistem yöneticisine iletilir. Sistem yöneticisi, sisteme giriş yapıp "Yetki Talebi"ni menüden seçtiğinde gelen talepler listelenir. Sistem yöneticisi, uygun gördüklerini seçip "Tamam" tuşuna basar. Ssitem, seçilen kullanıcıları yetkili kullanıcı olarak veri tabanına kaydeder ve e-posta adreslerine kullanıcı adını ve şifreyi gönderir.
Alternatif Akış	Sistem yöneticisinin, onaylamadığı talepler silinerek talebin red edildiği yine kullanıcıların e-posta adresleri aracılığıyla duyurulur.
Artçı Koşullar	Senaryo başarılı olduğunda veri tabanı yapısında yeni bir yetkili kullanıcının kaydı yapılmış olur.

Bilgi Keşfi Süreci

Kısa Açıklama	Kullanıcı/yetkili kullanıcı, HIS veri tabanında tutulan yapılar arasında ve bu yapıların tarihi olaylarla ilişkisini belirlemeye çalışır.
Aktörler	Kullanıcı, yetkili kullanıcı
Ön Koşullar	Sistemde bilgi keşfi sürecinin sağlıklı yapılabilmesi için yapını bulunduğu bölgeye ait tarihi olayların kayıtları, HIS veri tabanında olmalı. Bilgi keşfi sürecinde veri madenciliğini uygulanabilmesi için üzerinde çalışılacak veriler temiz olmalı ve eksik olmamalı.
Ana Akış	Kullanıcı/yetkili kullanıcı, ana menüden "Bilgi Keşfi"ni seçmeli. Kullanıcı/yetkili kullanıcı ana menüden "Bilgi Keşfi"ni seçtikten sonra üserinde çalışma yapacağı alanı (yapıların sınıflandırılması, belli bir tarihi olayın yapılar üzerindeki etkisi vb. şeklinde) belirlemelidir. Ayrıca çalışacağı bölgeyi de seçebilmelidir. Sistem, yapılan seçimlere ve uygulanmak istenen yönteme göre gerekli fonksiyonları çalıştırarak elde edilen sonuçları kullanıcıya/yetkili kullanıcıya iletir.
Alternatif Akış	Kullanıcı/yetkili kullanıcı, yeniden başka alanlarda bilgi keşfini gerçekleştirebilmek için sonucu elde ettikten sonra "Geri" tuşuna basarak tüm süreci tekrarlayabilir.
Artçı Koşullar	Senaryo başarılı olduğunda kullanıcıya/yetkili kullanıcıya kara verme sürecinde yardımcı olunur.

Ek 3 HIS Veri Tabanımındaki Sınıfların ve İlgili Tabloların SQL Komutları

SINIF	SINIF TABLOSU
CREATE TYPE religious AS (name char(30)) MODE DB2SQL	CREATE TABLE religioustable OF religious (REF IS rlid USER GENERATED, name WITH OPTIONS NOT NULL)
CREATE TYPE plantype AS (name varchar(20)) MODE DB2SQL	CREATE TABLE plantypetable OF plantype (REF IS ptid USER GENERATED, name WITH OPTIONS NOT NULL)
CREATE TYPE parcelusage AS (description varchar(100)) MODE DB2SQL	CREATE TABLE parcelusagetable OF parcelusage (REF IS puid USER GENERATED, description WITH OPTIONS NOT NULL)
CREATE TYPE arrangement UNDER parcelusage AS (dummy char(1)) MODE DB2SQL	CREATE TABLE arrangementtable OF arrangement UNDER parcelusagetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE placement UNDER parcelusage AS (dummy char(1)) MODE DB2SQL	CREATE TABLE placementtable OF placement UNDER parcelusagetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE usage UNDER parcelusage AS (dummy char(1)) MODE DB2SQL	CREATE TABLE usagetable OF usage UNDER parcelusagetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE garden UNDER parcelusage AS (dummy char(1)) MODE DB2SQL	CREATE TABLE gardentable OF garden UNDER parcelusagetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE locationplace AS (name char(50), population varchar(50)) MODE DB2SQL	CREATE TABLE locationplacetable OF locationplace (REF IS lpid USER GENERATED, name WITH OPTIONS NOT NULL)
CREATE TYPE ancientname AS (oldname char(50), startcentury decimal(3), finishcentury decimal(3), avgpopulation varchar(50), locationplacename REF(locationplace)) MODE DB2SQL	CREATE TABLE ancientnametable OF ancientname (REF IS anid USER GENERATED, oldname WITH OPTIONS NOT NULL, locationplacename WITH OPTIONS SCOPE locationplacetable)
CREATE TYPE country UNDER locationplace AS (dummy char(1)) MODE DB2SQL	CREATE TABLE countrytable OF country UNDER locationplacetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE city UNDER locationplace AS (countryname REF(country)) MODE DB2SQL	CREATE TABLE citytable OF city UNDER locationplacetable INHERIT SELECT PRIVILEGES (countryname WITH OPTIONS SCOPE countrytable)
CREATE TYPE district UNDER locationplace	CREATE TABLE districttable OF district

AS (cityname REF(city)) MODE DB2SQL	UNDER locationplacetable INHERIT SELECT PRIVILEGES (cityname WITH OPTIONS SCOPE citytable)
CREATE TYPE town UNDER locationplace AS (cityname REF(city)) MODE DB2SQL	CREATE TABLE towntable OF town UNDER locationplacetable INHERIT SELECT PRIVILEGES (cityname WITH OPTIONS SCOPE citytable)
CREATE TYPE mahalle UNDER locationplace AS (districtname REF(district)) MODE DB2SQL	CREATE TABLE mahalletable OF mahalle UNDER locationplacetable INHERIT SELECT PRIVILEGES (districtname WITH OPTIONS SCOPE districttable)
CREATE TYPE person AS(firstname char(25), lastname char(25)) MODE DB2SQL	CREATE TABLE persontable OF person (REF IS pid USER GENERATED, firstname WITH OPTIONS NOT NULL)
CREATE TYPE systemperson UNDER person AS (username char(10), password char(6), email varchar(50)) MODE DB2SQL	CREATE TABLE systempersontable OF systemperson UNDER persontable INHERIT SELECT PRIVILEGES (username WITH OPTIONS NOT NULL, password WITH OPTIONS NOT NULL)
CREATE TYPE hisperson UNDERperson AS (birthyear decimal(4), birthcentury decimal(3),deathyear decimal(4), deathcentury decimal(3), religiousname REF(religious), countryname REF(country)) MODE DB2SQL	CREATE TABLE hispersontable OF hisperson UNDER persontable INHERIT SELECT PRIVILEGES (religiousname WITH OPTIONS SCOPE religioustable, countryname WITH OPTIONS SCOPE countrytable)
CREATE TYPE hispeople AS (hp1 REF(hisperson), hp2 REF(hisperson), hp3 REF(hisperson), hp4 REF(hisperson), hp5 REF(hisperson)) MODE DB2SQL	CREATE TABLE hispeopletable OF hispeople (REF IS hpid USER GENERATED, hp1 WITH OPTIONS SCOPE hispersontable, hp2 WITH OPTIONS SCOPE hispersontable, hp3 WITH OPTIONS SCOPE hispersontable, hp4 WITH OPTIONS SCOPE hispersontable, hp5 WITH OPTIONS SCOPE hispersontable)
CREATE TYPE administrator UNDER systemperson AS (dummy char(1)) MODE DB2SQL	CREATE TABLE administratortable OF administrator UNDER systempersontable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE superuser UNDER systemperson AS (dummy char(1)) MODE DB2SQL	CREATE TABLE superusertable OF superuser UNDER systempersontable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE architect UNDER hisperson AS (dummy char(1)) MODE DB2SQL	CREATE TABLE architecttable OF architect UNDER hispersontable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE master UNDER hisperson AS (occupation varchar(30))MODE DB2SQL	CREATE TABLE mastertable OF master UNDER hispersontable INHERIT SELECT PRIVILEGES
CREATE TYPE corporation AS (name varchar(50)) MODE DB2SQL	CREATE TABLE corporationtable OF corporation (REF IS crid USER GENERATED, name WITH OPTIONS NOT

	NULL)
CREATE TYPE public UNDER corporation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE publictable OF public UNDER corporationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE firm UNDER corporation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE firmtable OF firm UNDER corporationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE foundation UNDER corporation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE foundationtable OF foundation UNDER corporationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE treasury UNDER public AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE treasurytable OF treasury UNDER publictable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE municipality UNDER public AS (dummy2 char(1)) MODE DB2SQL	CREATE TABLE municipalitytable OF municipality UNDER publictable INHERIT SELECT PRIVILEGES (dummy2 WITH OPTIONS DEFAULT 'D')
CREATE TYPE ministry UNDER public AS (dummy3 char(1)) MODE DB2SQL	CREATE TABLE ministrytable OF ministry UNDER publictable INHERIT SELECT PRIVILEGES (dummy3 WITH OPTIONS DEFAULT 'D')
CREATE TYPE relationship AS (startdate date, finishdate date, startcentury decimal(3), finishcentury decimal(3), personname REF(person), corporationname REF(corporation)) MODE DB2SQL	CREATE TABLE relationshiptable OF relationship (REF IS rsid USER GENERATED, personname WITH OPTIONS SCOPE persontable, corporationname WITH OPTIONS SCOPE corporationtable)
CREATE TYPE owner UNDER relationship AS (dummy char(1)) MODE DB2SQL	CREATE TABLE ownertable OF owner UNDER relationshiptable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE upkeep UNDER relationship AS (dummy char(1)) MODE DB2SQL	CREATE TABLE upkeeptable OF upkeep UNDER relationshiptable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE user UNDER relationship AS (usagefunction varchar(50), usagetime varchar(50)) MODE DB2SQL	CREATE TABLE usertable OF user UNDER relationshiptable INHERIT SELECT PRIVILEGES
CREATE TYPE state AS (description varchar(50)) MODE DB2SQL	CREATE TABLE statetable OF state (REF IS stid USER GENERATED, description WITH OPTIONS NOT NULL)
CREATE TYPE material AS (name varchar(50)) MODE DB2SQL	CREATE TABLE materialtable OF material (REF IS mtid USER GENERATED, name

	WITH OPTIONS NOT NULL)
CREATE TYPE addition AS (name varchar(50), adate date, description varchar(200)) MODE DB2SQL	CREATE TABLE additiontable OF addition (REF IS adid USER GENERATED, name WITH OPTIONS NOT NULL)
CREATE TYPE facadeaddition UNDER addition AS (dummy char(1)) MODE DB2SQL	CREATE TABLE facadeadditiontable OF facadeaddition UNDER additiontable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE planaddition UNDER addition AS (dummy char(1)) MODE DB2SQL	CREATE TABLE planadditiontable OF planaddition UNDER additiontable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE massaddition UNDER addition AS (dummy char(1)) MODE DB2SQL	CREATE TABLE massadditiontable OF massaddition UNDER additiontable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE documentation AS(name varchar(100), date date, century decimal(3), systemplace varchar(100)) MODE DB2SQL	CREATE TABLE documentationtable OF documentation (REF IS dcid USER GENERATED, name WITH OPTIONS NOT NULL)
CREATE TYPE photo UNDER documentation AS (photo BLOB(51200)) MODE DB2SQL	CREATE TABLE phototable OF photo UNDER documentationtable INHERIT SELECT PRIVILEGES
CREATE TYPE image UNDER documentation AS (image BLOB(51200)) MODE DB2SQL	CREATE TABLE imagetable OF image UNDER documentationtable INHERIT SELECT PRIVILEGES
CREATE TYPE gravur UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE gravurtable OF gravur UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE kitabe UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE kitabetable OF kitabe UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE kroki UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE krokitable OF kroki UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE tasbaski UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE tasbaskitable OF tasbaski UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE report UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE reporttable OF report UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE project UNDER documentation AS (dummy char(1)) MODE	CREATE TABLE projecttable OF project UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH

DB2SQL	OPTIONS DEFAULT 'D')
CREATE TYPE vakfiye UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE vakfiyetable OF vakfiye UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE sign UNDER documentation AS (dummy char(1)) MODE DB2SQL	CREATE TABLE signtable OF sign UNDER documentationtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE publishing UNDER documentation AS (authors varchar(75), publisher varchar(50), publishplace REF(city), publishtype varchar(15)) MODE DB2SQL	CREATE TABLE publishingtable OF publishing UNDER documentationtable INHERIT SELECT PRIVILEGES (publishplace WITH OPTIONS SCOPE citytable)
CREATE TYPE ottomancourt UNDER documentation AS (place varchar(50), no varchar(15), topic varchar(200), placeoldname REF(ancientname), mahalle REF(mahalle), street varchar(50), structure REF(structure), owner varchar(50), buyer varchar(50), cost varchar(50), retailer varchar(50), retailcost varchar(50), people REF(hispeople)) MODE DB2SQL	CREATE TABLE ottomancourttable OF ottomancourt UNDER documentationtable INHERIT SELECT PRIVILEGES (placeoldname WITH OPTIONS SCOPE ancientnametable, structure WITH OPTIONS SCOPE structuretable, people WITH OPTIONS SCOPE hispeopletable)
CREATE TYPE locationplacenames AS (lp1 REF(locationplace), lp2 REF(locationplace), lp3 REF(locationplace)) MODE DB2SQL	CREATE TABLE locationplacenamestable OF locationplacenames (REF IS lpid USER GENERATED, lp1 WITH OPTIONS SCOPE locationplacetable, lp2 WITH OPTIONS SCOPE locationplacetable, lp3 WITH OPTIONS SCOPE locationplacetable)
CREATE TYPE historicalevent AS (name varchar(70), startdate date, finishdate date, startcentury decimal(3), finishcentury decimal(3), result varchar(200), effect varchar(200), locationplacenames REF(locationplacenames)) MODE DB2SQL	CREATE TABLE historicaleventtable OF historicalevent (REF IS heid USER GENERATED, locationplacenames WITH OPTIONS SCOPE locationplacenamestable)
CREATE TYPE migration UNDER historicalevent AS (dummy char(1))MODE DB2SQL	CREATE TABLE migrationtable OF migration UNDER historicaleventtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE war UNDER historicalevent AS (dummy char(1)) MODE DB2SQL	CREATE TABLE wartable OF war UNDER historicaleventtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE stateestablish UNDER historicalevent AS (dummy char(1))MODE DB2SQL	CREATE TABLE stateestablishtable OF stateestablish UNDER historicaleventtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE statedemolish UNDER historicalevent AS (dummy char(1)) MODE DB2SQL	CREATE TABLE statedemolishtable OF statedemolish UNDER historicaleventtable INHERIT SELECT PRIVILEGES (dummy

	WITH OPTIONS DEFAULT 'D')
CREATE TYPE treaty UNDER historicalevent AS (dummy char(1)) MODE DB2SQL	CREATE TABLE treatytable OF treaty UNDER historicaleventtable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE law UNDER historicalevent AS (lawmaker REF(ministry)) MODE DB2SQL	CREATE TABLE lawtable OF law UNDER historicaleventtable INHERIT SELECT PRIVILEGES (lawmaker WITH OPTIONS SCOPE ministrytable)
CREATE TYPE disaster UNDER historicalevent AS (deadcount integer) MODE DB2SQL	CREATE TABLE disastertable OF disaster UNDER historicaleventtable INHERIT SELECT PRIVILEGES
CREATE TYPE earthquake UNDER disaster AS (dummy char(1)) MODE DB2SQL	CREATE TABLE earthquaketable OF earthquake UNDER disastertable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE flood UNDER disaster AS (dummy char(1)) MODE DB2SQL	CREATE TABLE floodtable OF flood UNDER disastertable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE disease UNDER historicalevent AS (deadcount integer) MODE DB2SQL	CREATE TABLE diseasetable OF disease UNDER historicaleventtable INHERIT SELECT PRIVILEGES
CREATE TYPE fire UNDER disaster AS (dummy char(1)) MODE DB2SQL	CREATE TABLE firetable OF fire UNDER disastertable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE damage AS (name varchar(50), date date, century decimal(3), effect varchar(200), state REF(state), diseasename REF(disease)) MODE DB2SQL	CREATE TABLE damagetable OF damage (REF IS dmid USER GENERATED, name WITH OPTIONS NOT NULL, state WITH OPTIONS SCOPE statetable, diseasename WITH OPTIONS SCOPE diseasetable)
CREATE TYPE moss UNDER damage AS (dummy char(1)) MODE DB2SQL	CREATE TABLE mosstable OF moss UNDER damagetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE damagenames AS (d1 REF(damage), d2 REF(damage), d3 REF(damage), d4 REF(damage), d5 REF(damage)) MODE DB2SQL	CREATE TABLE damagenamestable OF damagenames (REF IS dnid USER GENERATED, d1 WITH OPTIONS SCOPE damagetable, d2 WITH OPTIONS SCOPE damagetable, d3 WITH OPTIONS SCOPE damagetable, d4 WITH OPTIONS SCOPE damagetable, d5 WITH OPTIONS SCOPE damagetable)
CREATE TYPE arccomponent AS (name varchar(50), gridplace char(5), state REF(state)) MODE DB2SQL	CREATE TABLE arccomponenttable OF arccomponent (REF IS acid USER GENERATED, name WITH OPTIONS NOT NULL, state WITH OPTIONS SCOPE statetable)

CREATE TYPE technicaltype AS (name varchar(75)) MODE DB2SQL	CREATE TABLE technicaltypetable OF technicaltype (REF IS tcid USER GENERATED)
CREATE TYPE technical UNDER arccomponent AS (type REF(technicaltype)) MODE DB2SQL	CREATE TABLE technicaltable OF technical UNDER arccomponenttable INHERIT SELECT PRIVILEGES (type WITH OPTIONS SCOPE technicaltypetable)
CREATE TYPE ac UNDER arccomponent AS (length integer, width integer, material REF(material)) MODE DB2SQL	CREATE TABLE actable OF ac UNDER arccomponenttable INHERIT SELECT PRIVILEGES (material WITH OPTIONS SCOPE materialtable)
CREATE TYPE conactype AS (name varchar(75)) MODE DB2SQL	CREATE TABLE conactypetable OF conactype (REF IS cid USER GENERATED)
CREATE TYPE conveyerac UNDER ac AS (type REF(conactype)) MODE DB2SQL	CREATE TABLE conveyeractable OF conveyerac UNDER actable INHERIT SELECT PRIVILEGES (type WITH OPTIONS SCOPE conactypetable)
CREATE TYPE nonconactype AS (name varchar(75)) MODE DB2SQL	CREATE TABLE nonconactypetable OF nonconactype (REF IS ncid USER GENERATED)
CREATE TYPE nonconveyerac UNDER ac AS (type REF(nonconactype)) MODE DB2SQL	CREATE TABLE nonconveyeractable OF nonconveyerac UNDER actable INHERIT SELECT PRIVILEGES (type WITH OPTIONS SCOPE nonconactypetable)
CREATE TYPE decorationtype AS (name varchar(75)) MODE DB2SQL	CREATE TABLE decorationtypetable OF decorationtype (REF IS dtid USER GENERATED)
CREATE TYPE decoration UNDER arccomponent AS (date date, century decimal(3), length integer, width integer, master REF(master), material REF(material), type ref(decorationtype)) MODE DB2SQL	CREATE TABLE decorationtable OF decoration UNDER arccomponenttable INHERIT SELECT PRIVILEGES (master WITH OPTIONS SCOPE mastertable, material WITH OPTIONS SCOPE materialtable, type WITH OPTIONS SCOPE decorationtypetable)
CREATE TYPE arcsystem AS (name varchar(50), length integer, width integer, material REF(material), state REF(state), type REF(conastype)) MODE DB2SQL	CREATE TABLE arcsystemtable OF arcsystem (REF IS asid USER GENERATED, name WITH OPTIONS NOT NULL, material WITH OPTIONS SCOPE materialtable, state WITH OPTIONS SCOPE statetable, type WITH OPTIONS SCOPE conastypetable)
CREATE TYPE conastype AS (name varchar(75)) MODE DB2SQL	CREATE TABLE conastypetable OF conastype (REF IS cid USER GENERATED)
CREATE TYPE coveringastype AS (name varchar(75)) MODE DB2SQL	CREATE TABLE coveringastypetable OF coveringastype (REF IS cid USER GENERATED)
CREATE TYPE covering UNDER arccomponent AS (type REF(coveringastype), material REF(material)) MODE DB2SQL	CREATE TABLE coveringtable OF covering UNDER arccomponenttable INHERIT SELECT PRIVILEGES (type WITH OPTIONS SCOPE coveringastypetable, material WITH OPTIONS SCOPE

	materialtable)
CREATE TYPE facade AS (name char(10), inner integer, aspect char(10)) MODE DB2SQL	CREATE TABLE facadetable OF facade (REF IS fcid USER GENERATED, name WITH OPTIONS NOT NULL)
CREATE TYPE floor AS (name char(15), width integer, length integer, as REF(arcsystem)) MODE DB2SQL	CREATE TABLE floortable OF floor (REF IS flid USER GENERATED, name WITH OPTIONS NOT NULL, as WITH OPTIONS SCOPE arcsystemtable)
CREATE TYPE acs AS (ac REF(arccomponent), floor REF(floor), facade REF(facade)) MODE DB2SQL NOT : elemanın hangi katta hangi cephede olduğu belirlenir.	CREATE TABLE acstable OF acs (REF IS acsid USER GENERATED, ac WITH OPTIONS SCOPE arccomponenttable, floor WITH OPTIONS SCOPE floortable, facade WITH OPTIONS SCOPE facadetable)
CREATE TYPE structure AS (keycode char(10), inventoryno char(10),name char(30), gps char(10), cityname REF(city), districtname REF(district),townname REF(town), mahallename REF(mahalle), registrationdate date, registrationno char(10), mapno char(5), paftano char(5), islandno char(5), parcelno char(5), protectiongroup varchar(30), damagedangerous varchar(30), publishing varchar(100), kuruldocument varchar(100)) MODE DB2SQL	CREATE TABLE structuretable OF structure (REF IS stid USER GENERATED, keycode WITH OPTIONS NOT NULL, name WITH OPTIONS NOT NULL, cityname WITH OPTIONS SCOPE citytable, districtname WITH OPTIONS SCOPE districttable, townname WITH OPTIONS SCOPE towntable, mahallename WITH OPTIONS SCOPE mahalletable)
CREATE TYPE atype UNDERstructure AS (avenue varchar(30), street varchar(30), doorno char(5), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), architect REF(architect), master REF(master), maker REF(person), owner REF(owner), user REF(user), upkeep REF(upkeep), username varchar(50), landregister decimal(1), newactivity varchar(50), oldactivity varchar(50), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage), parcelusageG REF(garden), conveyerstate REF(state), outerstate REF(state), innerstate REF(state), topstate REF(state)) MODE DB2SQL NOT: landregister,"1" ise tapu kaydı var demektir.	CREATE TABLE atypetable OF atype UNDER structuretable INHERIT SELECT PRIVILEGES (architect WITH OPTIONS SCOPE architecttable, master WITH OPTIONS SCOPE mastertable, maker WITH OPTIONS SCOPE persontable, owner WITH OPTIONS SCOPE ownertable, user WITH OPTIONS SCOPE usertable, upkeep WITH OPTIONS SCOPE upkeeptable, landregister WITH OPTIONS DEFAULT 0, plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, conveyerstate WITH OPTIONS SCOPE statetable, outerstate WITH OPTIONS SCOPE statetable, innerstate WITH OPTIONS SCOPE statetable, topstate WITH OPTIONS SCOPE statetable)
CREATE TYPE floors AS (floor REF(floor), structure REF(structure)) MODE DB2SQL	CREATE TABLE floorstable OF floors (REF IS flid USER GENERATED, floor WITH OPTIONS SCOPE floortable, structure WITH OPTIONS SCOPE structuretable)
CREATE TYPE damages AS (damage REF(damage), structure REF(structure))	CREATE TABLE damagestable OF damages (REF IS dmid USER GENERATED, damage

MODE DB2SQL	WITH OPTIONS SCOPE damagetable, structure WITH OPTIONS SCOPE structuretable)
CREATE TYPE additions AS (addition REF(addition), structure REF(structure)) MODE DB2SQL	CREATE TABLE additionstable OF additions (REF IS adid USER GENERATED, addition WITH OPTIONS SCOPE additiontable, structure WITH OPTIONS SCOPE structuretable)
CREATE TYPE technicals AS (technical REF(technical), structure REF(structure)) MODE DB2SQL	CREATE TABLE technicalstable OF technicals (REF IS tcid USER GENERATED, technical WITH OPTIONS SCOPE technicaltable, structure WITH OPTIONS SCOPE structuretable)
CREATE TYPE documentations AS (document REF(documentation), structure REF(structure)) MODE DB2SQL	CREATE TABLE documentationstable OF documentations (REF IS dcid USER GENERATED, document WITH OPTIONS SCOPE documentationtable, structure WITH OPTIONS SCOPE structuretable)
CREATE TYPE education UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE educationtable OF education UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE religiousstructure UNDER atype AS (religious REF(religious)) MODE DB2SQL	CREATE TABLE religiousstructuretable OF religiousstructure UNDER atypetable INHERIT SELECT PRIVILEGES (religious WITH OPTIONS SCOPE religioustable)
CREATE TYPE resting UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE restingtable OF resting UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE health UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE healthtable OF health UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE commercial UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE commercialtable OF commercial UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE social UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE socialtable OF social UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE governance UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE governancetable OF governance UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE residence UNDER atype AS (dummy char(1)) MODE DB2SQL	CREATE TABLE residencetable OF residence UNDER atypetable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS

	DEFAULT 'D')
CREATE TYPE bazilika UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE bazilikatable OF bazilika UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE cami UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE camitable OF cami UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE hanikah UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE hanikahtable OF hanikah UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE kilise UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE kilisetable OF kilise UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE manastir UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE manastirtable OF manastir UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE mescid UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE mescidtable OF mescid UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE namazgah UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE namazgahtable OF namazgah UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE sinegog UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE sinagogtable OF sinegog UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE sapel UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE sapeltable OF sapel UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE tapinak UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE tapinaktable OF tapinak UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE tekke UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE tekketable OF tekke UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE vaftizhane UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE vaftizhanetable OF vaftizhanebazilika UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS

	DEFAULT 'D')
CREATE TYPE zaviye UNDER religiousstructure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE zaviyetable OF zaviye UNDER religiousstructuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE belediye UNDER governance AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE belediyetable OF belediye UNDER governancetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE hukumetkonak UNDER governance AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE hukumetkonaktable OF hukumetkonak UNDER governancetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE mahkeme UNDER governance AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE mahkemetable OF mahkeme UNDER governancetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE bimarhane UNDER health AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE bimarhanetable OF bimarhane UNDER healthtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE hastahane UNDER health AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE hastahanetable OF hastahane UNDER healthtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE saglikocagi UNDER health AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE saglikocagitable OF saglikocagi UNDER healthtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE sifahane UNDER health AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE sifahanetable OF sifahane UNDER healthtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE dispanser UNDER health AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE dispansertable OF dispanser UNDER healthtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE arasta UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE arastatable OF arasta UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE atolye UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE atolyetable OF atolye UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE banka UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE bankatable OF banka UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')

CREATE TYPE carsi UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE carsitable OF carsi UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE degirmen UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE degirmentable OF degirmen UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE depo UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE depotable OF depo UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE dukkan UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE dukkantable OF dukkan UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE fabrika UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE fabrikatable OF fabrika UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE firin UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE firintable OF firin UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE gazhane UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE gazhanetable OF gazhane UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE hal UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE haltable OF hal UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE imalathanc UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE imalathanctablc OF imalathane UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE kahvehane UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE kahvehanetable OF kahvehane UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE lokanta UNDER commercial AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE lokantatable OF lokanta UNDER commercialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE asevi UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE asevitable OF asevi UNDER socialtablc INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE darulhadis UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE darulhadistable OF darulhadis UNDER socialtable INHERIT

	SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE imarethane UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE imarethanetable OF imarethane UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE itfaiye UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE itfaiyetable OF itfaiye UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE kutuphane UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE kutuphanetable OF kutuphane UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE muze UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE muzetable OF muze UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE rasathane UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE rasathanetable OF rasathane UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE sporsalonu UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE sporsalonutable OF sporsalonu UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE tabhane UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE tabhanetable OF tabhane UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE tiyatro UNDER social AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE tiyatrotable OF tiyatro UNDER socialtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE idadi UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE idaditable OF idadi UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE ilkokul UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE ilkokultable OF ilkokul UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE medrese UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE medresetable OF medrese UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE lise UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE lisetable OF lise UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS

	DEFAULT 'D')
CREATE TYPE rustiye UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE rustiyetable OF rustiye UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE sibyanmektebi UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE sibyanmektebitable OF sibyanmektebi UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE universite UNDER education AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE universitetable OF universite UNDER educationtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE han UNDER resting AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE hantable OF han UNDER restingtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE otel UNDER resting AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE oteltable OF otel UNDER restingtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE kervansaray UNDER resting AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE kervansaraytable OF kervansaray UNDER restingtable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE apartman UNDER residence AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE apartmantable OF apartman UNDER residencetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE ev UNDER residence AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE evtable OF ev UNDER residencetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE konak UNDER residence AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE konaktable OF konak UNDER residencetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE kosk UNDER residence AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE kosktable OF kosk UNDER residencetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE mustemilat UNDER residence AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE mustemilattable OF mustemilat UNDER residencetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE saray UNDER residence AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE saraytable OF saray UNDER residencetable INHERIT SELECT PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE yali UNDER residence AS	CREATE TABLE yalitable OF yali UNDER residencetable INHERIT SELECT

(dummy1 char(1)) MODE DB2SQL	PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE monument UNDER structure AS (avenue varchar(30), street varchar(30), doorno char(5), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), architect REF(architect), maker REF(person),owner REF(owner), user REF(user), upkeep REF(upkeep), username varchar(50), landregister decimal(1), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage), parcelusageG REF(garden), conveyerstate REF(state), outerstate REF(state), innerstate REF(state), topstate REF(state)) MODE DB2SQL	CREATE TABLE monumenttable OF monument UNDER structuretable INHERIT SELECT PRIVILEGES (architect WITH OPTIONS SCOPE architecttable, maker WITH OPTIONS SCOPE persontable, owner WITH OPTIONS SCOPE ownertable, user WITH OPTIONS SCOPE usertable, upkeep WITH OPTIONS SCOPE upkeeptable, landregister WITH OPTIONS DEFAULT 0, plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, conveyerstate WITH OPTIONS SCOPE statetable, outerstate WITH OPTIONS SCOPE statetable, innerstate WITH OPTIONS SCOPE statetable, topstate WITH OPTIONS SCOPE statetable)
CREATE TYPE strategic UNDER structure AS (avenue varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), architect REF(architect), master REF(master), maker REF(person), owner REF(owner), user REF(user), upkeep REF(upkeep), username varchar(50), landregister decimal(1), newactivity varchar(50), oldactivity varchar(50), conveyerstate REF(state), outerstate REF(state), topstate REF(state)) MODE DB2SQL	CREATE TABLE strategictable OF strategic UNDER structuretable INHERIT SELECT PRIVILEGES (architect WITH OPTIONS SCOPE architecttable, master WITH OPTIONS SCOPE mastertable, maker WITH OPTIONS SCOPE persontable, owner WITH OPTIONS SCOPE ownertable, user WITH OPTIONS SCOPE usertable, upkeep WITH OPTIONS SCOPE upkeeptable, landregister WITH OPTIONS DEFAULT 0, conveyerstate WITH OPTIONS SCOPE statetable, outerstate WITH OPTIONS SCOPE statetable, topstate WITH OPTIONS SCOPE statetable)
CREATE TYPE hapishane UNDER strategic AS (street varchar(30), doorno char(5), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage),parcelusageG REF(garden), innerstate REF(state)) MODE DB2SQL	CREATE TABLE hapishanetable OF hapishane UNDER strategictable INHERIT SELECT PRIVILEGES (plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, innerstate WITH OPTIONS SCOPE statetable)
CREATE TYPE karakol UNDER strategic AS (street varchar(30), doorno char(5), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage), parcelusageG REF(garden), innerstate REF(state)) MODE DB2SQL	CREATE TABLE karakoltable OF karakol UNDER strategictable INHERIT SELECT PRIVILEGES (plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH

	OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, innerstate WITH OPTIONS SCOPE statetable)
CREATE TYPE kisla UNDER strategic AS (street varchar(30), doorno char(5), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage), parcelusageG REF(garden), innerstate REF(state)) MODE DB2SQL	CREATE TABLE kislatable OF kisla UNDER strategictable INHERIT SELECT PRIVILEGES (plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, innerstate WITH OPTIONS SCOPE statetable)
CREATE TYPE kale UNDER strategic AS (street varchar(30), doorno char(5), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage),parcelusageG REF(garden), innerstate REF(state)) MODE DB2SQL	CREATE TABLE kaletable OF kale UNDER strategictable INHERIT SELECT PRIVILEGES (plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, innerstate WITH OPTIONS SCOPE statetable)
CREATE TYPE burc UNDER strategic AS (innerstate REF(state)) MODE DB2SQL	CREATE TABLE burctable OF burc UNDER strategictable INHERIT SELECT PRIVILEGES (innerstate WITH OPTIONS SCOPE statetable)
CREATE TYPE sur UNDER strategic AS (dummy char(1)) MODE DB2SQL	CREATE TABLE surtable OF sur UNDER strategictable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE surduvari UNDER strategic AS (dummy char(1)) MODE DB2SQL	CREATE TABLE surduvaritable OF surduvari UNDER strategictable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE tabya UNDER strategic AS (plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage),parcelusageG REF(garden), innerstate REF(state)) MODE DB2SQL	CREATE TABLE tabyatable OF tabya UNDER strategictable INHERIT SELECT PRIVILEGES (plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, innerstate WITH OPTIONS SCOPE statetable)
CREATE TYPE natural UNDER structure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE naturaltable OF natural UNDER structuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE agac UNDER natural AS (dummy1 char(1)) MODE DB2SQL	CREATE TABLE agactable OF agac UNDER naturaltable INHERIT SELECT

	PRIVILEGES (dummy1 WITH OPTIONS DEFAULT 'D')
CREATE TYPE bahce UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE bahcetable OF bahce UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE bostan UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE bostantable OF bostan UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE bag UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE bagtable OF bag UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE koru UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE korutable OF koru UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE kumsal UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE kumsaltable OF kumsal UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE orman UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE ormantable OF orman UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE park UNDER natural AS (milli decimal(1), landregister decimal(1)) MODE DB2SQL NOT: milli "1" ise ise millipark olduğunu gösterir.	CREATE TABLE parktable OF park UNDER naturaltable INHERIT SELECT PRIVILEGES (milli WITH OPTIONS DEFAULT '0', landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE tarla UNDER natural AS (landregister decimal(1)) MODE DB2SQL	CREATE TABLE tarlatable OF tarla UNDER naturaltable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0)
CREATE TYPE urban UNDER structure AS (dummy char(1)) MODE DB2SQL	CREATE TABLE atypetable OF atype UNDER structuretable INHERIT SELECT PRIVILEGES (dummy WITH OPTIONS DEFAULT 'D')
CREATE TYPE asansor UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), architect REF(architect), maker REF(person), owner REF(owner), user REF(user), upkeep REF(upkeep), username varchar(50), landregister decimal(1), newactivity varchar(50), oldactivity varchar(50), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP	CREATE TABLE asansortable OF asansor UNDER urbantable INHERIT SELECT PRIVILEGES (architect WITH OPTIONS SCOPE architecttable, maker WITH OPTIONS SCOPE persontable, owner WITH OPTIONS SCOPE ownertable, user WITH OPTIONS SCOPE usertable, upkeep WITH OPTIONS SCOPE upkeeptable, landregister WITH OPTIONS DEFAULT 0, plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE

REF(placement), parcelusageU REF(usage), parcelusageG REF(garden), Conveyerstate REF(state), outerstate REF(state), innerstate REF(state), topstate REF(state)) MODE DB2SQL NOT: landregister,"1" ise tapu kaydı var demektir.	arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, conveyerstate WITH OPTIONS SCOPE statetable, outerstate WITH OPTIONS SCOPE statetable, innerstate WITH OPTIONS SCOPE statetable, topstate WITH OPTIONS SCOPE statetable)
CREATE TYPE dikilitas UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), master REF(master), maker REF(person), landregister decimal(1)) MODE DB2SQL	CREATE TABLE dikilitastable OF dikilitas UNDER urbantable INHERIT SELECT PRIVILEGES (master WITH OPTIONS SCOPE mastertable, maker WITH OPTIONS SCOPE persontable, landregister WITH OPTIONS DEFAULT 0) NOT: landregister,"1" ise tapu kaydı var demektir.
CREATE TYPE heykel UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), master REF(master), maker REF(person), landregister decimal(1)) MODE DB2SQL	CREATE TABLE heykeltable OF heykel UNDER urbantable INHERIT SELECT PRIVILEGES (master WITH OPTIONS SCOPE mastertable, maker WITH OPTIONS SCOPE persontable, landregister WITH OPTIONS DEFAULT 0) NOT: landregister,"1" ise tapu kaydı var demektir.
CREATE TYPE meydan UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), landregister decimal(1)) MODE DB2SQL	CREATE TABLE meydantable OF meydan UNDER urbantable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0) NOT: landregister,"1" ise tapu kaydı var demektir.
CREATE TYPE nisantasi UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), landregister decimal(1)) MODE DB2SQL	CREATE TABLE nisantasitable OF nisantasi UNDER urbantable INHERIT SELECT PRIVILEGES (landregister WITH OPTIONS DEFAULT 0) NOT: landregister,"1" ise tapu kaydı var demektir.
CREATE TYPE sokaklambasi UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3)) MODE DB2SQL	CREATE TABLE sokaklambasitable OF sokaklambasi UNDER urbantable INHERIT SELECT PRIVILEGES
CREATE TYPE tabela UNDER urban AS (avenue varchar(30), street varchar(30)) MODE DB2SQL	CREATE TABLE tabelatable OF tabela UNDER urbantable INHERIT SELECT PRIVILEGES
CREATE TYPE levha UNDER urban AS (avenue varchar(30), street varchar(30)) MODE DB2SQL	CREATE TABLE levhatable OF levha UNDER urbantable INHERIT SELECT PRIVILEGES
CREATE TYPE yol UNDER urban AS	CREATE TABLE yoltable OF yol UNDER

(avenue varchar(30), street varchar(30)) MODE DB2SQL	urbantable INHERIT SELECT PRIVILEGES
CREATE TYPE yolkaplama UNDER urban AS (avenue varchar(30), street varchar(30)) MODE DB2SQL	CREATE TABLE yolkaplamatable OF yolkaplama UNDER urbantable INHERIT SELECT PRIVILEGES
CREATE TYPE saatkulesi UNDER urban AS (avenue varchar(30), doorno char(5), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), architect REF(architect), maker REF(person), owner REF(owner), user REF(user), upkeep REF(upkeep), username varchar(50), landregister decimal(1), newactivity varchar(50), oldactivity varchar(50), plantype REF(plantype), parcelusageA REF(arrangement), parcelusageP REF(placement), parcelusageU REF(usage), parcelusageG REF(garden), Conveyerstate REF(state), outerstate REF(state), innerstate REF(state), topstate REF(state)) MODE DB2SQL NOT: landregister,"1" ise tapu kaydı var demektir.	CREATE TABLE saatkulesitable OF saatkulesi UNDER urbantable INHERIT SELECT PRIVILEGES (architect WITH OPTIONS SCOPE architecttable, maker WITH OPTIONS SCOPE persontable, owner WITH OPTIONS SCOPE ownertable, user WITH OPTIONS SCOPE usertable, upkeep WITH OPTIONS SCOPE upkeeptable, landregister WITH OPTIONS DEFAULT 0, plantype WITH OPTIONS SCOPE plantypetable, parcelusageA WITH OPTIONS SCOPE arrangementtable, parcelusageP WITH OPTIONS SCOPE placementtable, parcelusageU WITH OPTIONS SCOPE usagetable, parcelusageG WITH OPTIONS SCOPE gardentable, conveyerstate WITH OPTIONS SCOPE statetable, outerstate WITH OPTIONS SCOPE statetable, innerstate WITH OPTIONS SCOPE statetable, topstate WITH OPTIONS SCOPE statetable)
CREATE TYPE teleferik UNDER urban AS (avenue varchar(30), street varchar(30), startyear decimal(4), startcentury decimal(3), finishyear decimal(4), finishcentury decimal(3), architect REF(architect), maker REF(person), owner REF(owner), user REF(user), upkeep REF(upkeep), username varchar(50), newactivity varchar(50), oldactivity varchar(50), Conveyerstate REF(state), outerstate REF(state), innerstate REF(state), topstate REF(state)) MODE DB2SQL	CREATE TABLE asansortable OF asansor UNDER urbantable INHERIT SELECT PRIVILEGES (architect WITH OPTIONS SCOPE architecttable, maker WITH OPTIONS SCOPE persontable, owner WITH OPTIONS SCOPE ownertable, user WITH OPTIONS SCOPE usertable, upkeep WITH OPTIONS SCOPE upkeeptable, conveyerstate WITH OPTIONS SCOPE statetable, outerstate WITH OPTIONS SCOPE statetable, innerstate WITH OPTIONS SCOPE statetable, topstate WITH OPTIONS SCOPE statetable)
CREATE TYPE repair AS (name varchar(50), rdate date, description varchar(200), structure REF(structure)) MODE DB2SQL	CREATE TABLE repairtable OF repair (REF IS rpid USER GENERATED, name WITH OPTIONS NOT NULL, structure WITH OPTIONS SCOPE structuretable)

Ek 4 HIS'de Kullanılan Yapı Bilgisi Dökümantasyonu Örneği

TÜRKİYE KENTSEL KÜLTÜR VARLIKLARI ENVANTERİ

ANIT	ENVANTER NO.	G34
	FOTOĞRAF NO.	67:8
	DİJİTAL FOTOĞRAF NO.	E4:32
YAPI ✓	SAYDAM NO.	
KENTSEL ÖĞE	RÖLÖVE NO.	
DOĞA ÖĞESİ	HARİTA NO.	
	GPS VERİLERİ	

YERLEŞME

İLİ	İLÇESİ	MAHALLE KURTULUŞ			
SOKAK VE KAPI NO. Kurşunlu cad No:73		PAFTA	ADA 117	PARSEL 7	

ANIT

ADI — GÜNÜMÜZDEKİ İŞLEVİ: Konut

ANITIN BULUNDUĞU BÖLGENİN NİTELİĞİ	KONUT ALANI	TİCARET ALANI	KONUT + TİCARET ALANI ✓	ENDÜSTRİ ALANI	MERKEZİ İŞ ALANI	DİNSEL ALAN	KAMU / HİZMET ALANI	BOŞ ALAN / DİĞER

YAPININ BÖLGE İÇİNDEKİ KONUMU: BİTİŞİK | AYRIK | KÖŞE

YAPI TÜRÜ

KONUT: SARAY | KONAK | YALI | EV ✓ | MÜSTEMİLAT

DİNSEL YAPILAR	TİCARET YAPILARI		EĞİTİM – KÜLTÜR-DİNLENCE YAPILARI	GÜVENLİK YAPILARI	SU YAPILARI	SAĞLIK YAPILARI	YÖNETİM/İLETİŞİM YAPILARI		ULAŞIM YAPILARI
CAMİ	KERVANSARAY / HAN	FABRİKA	OKUL	KALE	HAMAM	ŞİFAHANE	HÜKÜMET KONAĞI		İSTASYON
KİLİSE	ARASTA	ATÖLYE	SÜBYAN MEKTEBİ	SUR	ÇEŞME	BİMARHANE	BELEDİYE BİNASI		İSKELE
SİNAGOG	BEDESTEN	HAL BİNASI	KİTAPLIK	KIŞLA	SEBİL	HASTANE	İTFAİYE		LİMAN
MEDRESE	ÇARŞI	DEPO	MÜZE	KARAKOL	SARNIÇ	SAĞLIK OCAĞI	POSTAHANE		
MANASTIR	FIRIN	BÜRO/İŞ HANI	SPOR TESİSİ	HAPİSHANE	AYAZMA	DİSPANSER			
TÜRBE	DEĞİRMEN	BANKA	TİYATRO		SU KEMERİ				
TEKKE	DÜKKAN	OTEL							
DİĞER									

KENTSEL ÖĞE	İSKELE / RIHTIM	KÖPRÜ	YOL	MEYDAN	SAAT KULESİ	HEYKEL	SOKAK LAMBASI	TABELA/ LEVHA	HAVUZ	KUYU	MEZARLIK	DİĞER
DOĞA ÖĞESİ	ORMAN	KORU	BOSTAN	BAHÇE	PARK	AĞAÇ	KUMSAL	KAYALIK	DİĞER			

DÖNEM	ANTİK	ORTAÇAĞ	OSMANLI	CUMHURİYET
YAPIM TARİHİ / YÜZYIL Son dönem osmanlı 19. Yüzyıl başı?			YAZIT	VAR / YOK

MAL SAHİBİ				BAKIMINDAN SORUMLU KURULUŞ		
DEVLET	VAKIF	DİĞER	ÖZEL ✓	DEVLET	VAKIF	DİĞER ✓
YAPAN				YAPTIRAN		

YAPIYA GİRİLDİ | YAPIYA GİRİLMEDİ

YAPININ KAT ADEDİ	BODRUM	1 KATLI	2 KATLI ✓	ÇOK KATLI	TERAS	CİHANNÜMA	DİĞER

TAŞIYICI SİSTEM

HİMİŞ ✓	BETONARME KARKAS	ÇELİK KARKAS	YIĞMA, TUĞLA	YIĞMA, TAŞ	YIĞMA, TUĞLA-TAŞ ALMAŞIK	YIĞMA, KERPİÇ	KARMAŞIK SİSTEMLER	DİĞER

ÖRTÜ TÜRÜ	BEŞİK ÇATI	KIRMA ÇATI ✓	DÜZ ÇATI	BÖRİSEL		DİĞER
				KUBBE	TONOZ	
ÖRTÜ MALZEMESİ TÜRÜ	ALATURKA KİREMİT	MARSİLYA KİREMİTİ ✓	TAŞ	TOPRAK	BETON	ETERNİT / DİĞER

SAÇAK TÜRÜ	SAÇAKSIZ	DAR SAÇAK ✓	GENİŞ SAÇAK	DİĞER

KONUT PLAN TİPİ	İÇ SOFALI	DIŞ SOFALI	EYVANLI	AVLULU	BAHÇELİ			DİĞER
				AVLU / BAHÇE DUVARI ÖRGÜSÜ	TUĞLA	TAŞ	ALMAŞIK	DİĞER
				AVLU / BAHÇE DUVARI BİTİŞİ	HARPUŞTA		KİREMİT	DİĞER

ANA CEPHE

KAPLAMA MALZEMESİ		AHŞAP	TAŞ	TUĞLA	SIVA ✓	DİĞER

CEPHE ÖZELLİKLERİ	SUNDURMA	ÇIKMA ✓	TEK KATLI ÇIKMA ✓	ÇOK KATLI ÇIKMA	DİĞER	BALKON ✓	ÇIKMA ÜSTÜ BALKON		
		PARALEL ÇIKMA ✓	ÜÇGEN ÇIKMA	ÇOKGEN ÇIKMA		BALKON KORKULUĞU	TAŞ	METAL	DİĞER

PENCERE							GİRİŞ KAPISI				
KEMERLİ	GİYOTİN	ÇİFT KANATLI ✓	TEK KANATLI	TEPE	DİĞER		KEMERLİ	ÇİFT KANATLI ✓	TEK KANATLI	DİĞER	
PENCERE ÖĞESİ							GİRİŞ KAPISI MALZEMESİ	AHŞAP	METAL ✓	DİĞER	

KEPENK	PARMAKLIK	KAFES	RENKLİ CAM	DİĞER	YAPI ÖĞELERİ						
AHŞAP	AHŞAP				YANGIN DUYARI	BACA	EL BÖĞRÜNDE	KONSOL	ÇÖRTEN	KÜŞEVİ	DİĞER
METAL	METAL										
BEZEME					DEVŞİRME MALZEMESİ						

TEKNİK DONATI

ELEKTRİK ✓	SU ✓	KANALİZASYON ✓	ISITMA				TELEFON	DİĞER
			KALORİFER	SOBA ✓	GELENEKSEL			

SAĞLAMLIK DURUMU	HARAP	KÖTÜ	ORTA ✓	İYİ	ONARIM GÖRMÜŞ	TESCİLLİ	DİĞER
ÖZGÜNLÜK DURUMU		KÖTÜ ✓	ORTA	İYİ			

ARAZİDE HAZIRLAYANLAR	G. E	TARİH	16.07
ARAZİDE KONTROL EDENLER		TARİH	

ÖZGEÇMİŞ

Doğum tarihi	04.02.1976	
Doğum yeri	Ankara	
Lise	1987 - 1994	Kadıköy Anadolu Lisesi
Lisans	1995 - 1999	Yıldız Teknik Üniversitesi Elektrik Elektronik Fakültesi Bilgisayar Bilimleri ve Mühendisliği Bölümü
Yüksek Lisans	1999 - 2002	Yıldız Teknik Üniversitesi Sosyal Bilimler Enstitüsü İşletme Yönetimi Programı
Yüksek Lisans	2002 -	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği

Çalıştığı Kurumlar

1999 (6 ay)	Mikado Bilgisayar Ltd. Şti. Bilgisayar Mühendisi
1999 - 2004	YTÜ Bilgisayar Mühendisliği Bölümü Araştırma Görevlisi
2004 -	Turkcell İletişim Hizmetleri A.Ş. İş Analisti