**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# VERİ TABANI YÖNETİM SİSTEMLERİNDE MEKANSAL VERİLERİN ÇOK KULLANICILI ORTAMDA DEPOLANMASI VE ÖZGEÇMİŞLERİNİN İZLENMESİ

**DOKTORA TEZİ**

**Y. Müh. Ahmet KARABURUN**

**Anabilim Dalı : JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ**

**Programı : JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ**

**OCAK 2006**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# VERİ TABANI YÖNETİM SİSTEMLERİNDE MEKANSAL VERİLERİN ÇOK KULLANICILI ORTAMDA DEPOLANMASI VE ÖZGEÇMİŞLERİNİN İZLENMESİ

**DOKTORA TEZİ**

**Y. Müh. Ahmet KARABURUN**
**(501992203)**

**Tezin Enstitüye Verildiği Tarih : 18 Ocak 2006**
**Tezin Savunulduğu Tarih : 01 Mart 2006**

| | |
|---|---|
| **Tez Danışmanı :** | **Prof.Dr. Gönül TOZ** |
| **Diğer Jüri Üyeleri** | **Prof.Dr. Doğan UÇAR (İ.T.Ü.)** |
| | **Prof.Dr. Handan TÜRKOĞLU (İ.T.Ü.)** |
| | **Prof.Dr. Ayhan ALKIŞ (Y.T.Ü.)** |
| | **Doç.Dr. Gül BATUK (Y.T.Ü.)** |

**OCAK 2006**

## ÖNSÖZ

İTÜ Fen Bilimleri Enstitüsü Jeodezi ve Fotogrametri Anabilim Dalı, Jeodezi ve Fotogrametri Mühendisliği Programı'nda gerçekleştirilen doktora çalışmam sırasında desteğini hiçbir zaman esirgemeyen, bilgi ve deneyimleri ile beni yönlediren tez danışmanım sayın Prof. Dr. Gönül TOZ'a, sonsuz teşekkürlerimi sunarım.

Çalışmam sırasında bana zaman ayıran ve destekleyen Jeodezi ve Fotogrametri Bölüm Başkanı sayın Prof. Dr. Doğan UÇAR'a ve Şehir ve Bölge Planlama Bölümü Başkanı sayın Prof. Dr. Handan TÜRKOĞLU'na çok teşekkür ederim.

Ayrıca bu günlere gelmemi sağlayan aileme ve çalışmam sırasında emeği geçen, ilgi ve anlayışıyla beni her zaman destekleyen eşim Müh. Rabia KARABURUN'a sonsuz teşekkür ve sevgilerimi sunarım.

**Ocak 2006** **Ahmet KARABURUN**

**İÇİNDEKİLER**

**KISALTMALAR**

| | |
|---|---|
| **SQL** | : Structured Query Language |
| **OO** | : Object Oriented |
| **IP** | : Internet Protocol |
| **TCP/IP** | : Transmission Control Protocol/Internet Protocol |
| **ISO/TC** | : International Organization for Standardization Technical Committee |
| **OGC** | : Open Geospatial Consortium |
| **UTM** | : Universal Transversal Mercator |
| **BLOB** | : Binary Large Object |
| **CBS** | : Coğrafi Bilgi Sistemleri |
| **TGIS** | : Temporal Geography Information Systems |
| **VTYS** | : Veri Tabanı Yönetim Sistemi |
| **API** | : Application Programming Interface |
| **ESRI** | : Environmental Systems Reaseach Institute |
| **IDE** | : Integrated Development Environment |
| **COM** | : Component Object Model |
| **DLL** | : Dynamic Link Library |
| **UML** | : Unified Modelling Language |

**TABLO LİSTESİ**

## ŞEKİL LİSTESİ

## VERİ TABANI YÖNETİM SİSTEMLERİNDE MEKANSAL VERİLERİN ÇOK KULLANICILI ORTAMDA DEPOLANMASI VE ÖZGEÇMİŞLERİNİN İZLENMESİ


### ÖZET

Veri tabanı yönetim sistemi yazılımlarının coğrafi bilgi sistemlerinde artan kullanımı bu yazılımların sağlamış olduğu diğer faydaların da sistem içerisinde değerlendirilmesi fikrini oluşturmuştur. Bu faydalardan bir tanesi kullanıcılara çok kullanıcı ortamın sunulması, bir diğer ise verilerin özgeçmişlerinin takibidir.

Coğrafi bilgi sisteminin veri üretim ve yönetim aşamasında birden fazla kullanıcının aynı anda aynı veri üzerinde güncelleme yapma ihtiyacı olabilmektedir. Bu durumda kullanılan veri tabanı yönetim sisteminin kullanıcılarına verilerin birden fazla kopyasını almadan eş zamanlı güncelleme imkanı sağlaması gerekmektedir.

Coğrafi bilgi sistemlerinde mekansal bilgi çoğunlukla konum ve öznitelik bilgisi bileşenlerine ayrılmış ve zaman bileşeni sisteme dahil edilmemiştir. Zaman bilgisinin de coğrafi bilgi sistemine dahil edilmesi coğrafi bilgi sisteminin temsil ettiği gerçek dünyaya daha yakınlaştıracak ve mekansal planlama ve karar vermede daha güvenli bir kullanıma imkan verecektir. Böylece veri tabanlarında özgeçmişlerin modellenmesi; geçmiş bir tarihteki veri tabanı içeriğini veya bir objenin zaman içerisindeki değişiminin izlenmesine ve görüntülenmesine imkan sağlayacaktır.

Verilerin özgeçmişlerinin takibi ve çok kullanıcılı güncelleme imkanının sağlanması için veri modeli seçimi çok önemli olmaktadır. Bu veri modeli veri tabanının hacmini artırmamalı ve mekansal verileri depolarken karmaşık algoritmalar kullanmamalıdır. Özgeçmiş takibi ve çok kullanıcılı ortam için tek veri modelinin kullanılması sistemin kolay kullanılabilirliğini ve çalışma hızını olumlu etkileyecektir.

Mekansal veri tabanları üzerinde bu amaca hizmet etmek için farklı uygulamalar geliştirilmiştir. Bu uygulamalardan en önemlisi mekansal veri sunucusu olarak adlandırılan ve veri tabanı yönetim sistemleri üzerinde mekansal verilerin depolanmasına imkan sağlayan yazılımlardır. Bu çalışmada veri tabanı yönetim sistemleri incelenerek çok kullanıcılı ortam ve verilerin özgeçmişlerinin izlenmesi için bu yazılımların sunmuş olduğu en uygun yöntemin belirlenmesi amaçlanmıştır.

Çalışmanın ilk bölümünde veri tabanı yönetim sistemlerinde mekansal verilerin çok kullanıcılı ortamda depolanması ve özgeçmişlerinin takip edilmesi gerekliliğine neden ihtiyaç duyulduğu belirtilmektedir.

İkinci bölümde veri tabanı yönetim sistemleri ve esasları ile veri tabanı yönetim sistemlerinin kullanmış oldukları veri modellerine değinilmiş ve veri tabanı tasarımında kullanılan araçların önemleri vurgulanmıştır.

Üçüncü bölümde, mekansal verinin veri tabanı yönetim sistemlerinde depolama esasları özetlenmiş, mekansal verilerin depolama yöntemleri ve mekansal veri sunucuları incelenirken bu konudaki uluslararası standartlar vurgulanmıştır.

Dördüncü bölümde çok kullanıcılı ortam için kullanılan yöntem olan versiyon uygulaması örnek şekil ve tablolar ile açıklanmış ve versiyon senaryosu örnek bir güncelleme işlemi ile gösterilmişdir.

Beşinci bölümde mekansal verilerin özgeçmişlerinin takibi için kullanılan kavramsal veri modellerine değinilmiş ve özgeçmiş sorgulamaları içerisinde kullanılan zaman kavramı ayrıntılı bir biçimde ele alınmıştır.

Altıncı bölümde mekansal verilerin çok kullanıcılı ortamda güncellenmesi ve özgeçmişlerinin izlenmesi amacıyla bu tez çalışmasında geliştirilen özgün uygulama sunulmuştur.

Sonuç ve öneriler bölümünde ise coğrafi bilgi sistemlerinde veri tabanı yönetim sistemlerinin önemi bir kez daha vurgulanmıştır. Mekansal verilerin çok kullanıcılı ortamda depolanması ve özgeçmişlerinin izlenebilmesi için kullanıcılacak veri tabanı yönetim sistemlerinin ve coğrafi bilgi sistemi yazılımlarının sahip olması gereken özellikler ortaya konmuştur.

**STORING SPATIAL DATA IN MULTIUSER LEVEL AND TRACKING HISTORY IN DATABASE MANAGEMENT SYSTEMS**


## SUMMARY

The increased use of database management systems in geographical information systems technologies directed user to benefit the other advantages of database management systems. One of these advantage is to support multiuser level and the other is tracking history of data.

Users may be required to edit same data at the same time at any stage of geographical information of systems. So database management systems of geographical information systems must support multiuser concurrent editing without creating multiple copies of the data.

Spatial information is commonly broken into components of space and attribute in geographical information systems and time component wasn't included in systems. Including time component will bring a geographical information systems more close to the real world which it represents, and enable a more reliable use of the data spatial planning and decision making. Thus modeling history in a database allows visualizing the database's contents at a particular time in the past or visualizing how a particular object changed over time.

It is very important to select data model for tracking history of data and multiuser level . This data model musn't increase the size of database and musn't use complex storing algorithms to store spatial data. Many different applications were developed on database management systems to achieve this aim. One of these applications is developing a software called spatial database engine which allows storing spatial data in database management systems.The aim of this study is to research the database management systems and spatial database engines and to decide most convenient method that they provide.

The first chapter of this study explains the necessity for storing the spatial data in multiuser level and tracking history in database management systems.

The second part includes basic of database management systems and its data models.

The third part summarizes fundamentals of storing spatial data in database management systems. It includes the methods of storing spatial data and basics of spatial database engine.

The fourth part includes version application and its principles with sample figures and tables for multiuser level. A sample versioning scenario is shown using a sample editing process.

The fifth section includes tracking history of spatial data, its data model, and historical queries.

In the sixth part, introduces the used software for application that allows multiuser editing and historical queries.

It was mentioned about the reasons of using the database management systems in geographical information systems once again in the results and suggestion section. In order to store spatial data in database management systems in multiuser level and tracking history, technical specifications provided by database management systems and geographical information systems software are discussed.

# 1. GİRİŞ VE ÇALIŞMANIN AMACI

Coğrafi bilgi sistemlerinin kullanım alanı genişledikçe bilgisayarlarda dosya-bazlı olarak saklanan mekânsal verilerin gerek güncelleme ve gerekse veri yönetimi açısından beklentilere cevap verememesi kullanıcıları farklı çözüm arayışlarına yöneltmiştir. Uzunca bir süredir çoğu bilgi sistemlerinde başarı ile kullanılan ve artık veri depolama ve yönetimi açısından en uygun ortam olarak kabul edilen veri tabanı yönetim sistemlerinde mekânsal verilerin de depolanabilmesi coğrafi bilgi sistemlerinin gelişiminde büyük rol oynamıştır (Oracle,2002) (Adler, 2001).

Son yıllarda veri tabanları üzerinde mekansal verileri depolamaya imkan sağlayan mekansal veri sunucu yazılımlarının üretilmesi ve ile karmaşık veriler depolanmaya başlanmıştır. Böylece veriler üzerinde farklı sorgulamalara imkan verilmiş ve sağladığı kullanım kolaylığı nedeni ile coğrafi bilgi sistemlerinde geniş bir uygulama alanı bulmaya başlamıştır. Artık bugün çoğu coğrafi bilgi sistemi projelerinde kullanıcılar ilişkisel veri tabanının sağlamış olduğu;

- Mekânsal veriler ve bunların öznitelik bilgilerinin tek bir yerde depolanması
- Standart veri yönetim uygulamaları, veri yedeklerinin alınması ve dönüştürülmesi
- Performansın kullanıcı sayısından ve veri hacminden bağımsız olması
- Personel ve donanım değişiklerinin sistemi çok fazla etkilememesi
- Sistem arızalandığında yedekleme ünitelerinin devreye girmesi
- Kullanıcı/sunucu ve internet mekanizmalarını desteklemesi

imkânları nedeni ile mevcut verilerini veri tabanı yönetim sistemleri üzerinde depolamaya ve yönetmeye başlamıştır.

Veri tabanı yönetim sistemlerinin sağlamış olduğu bu imkânlar nedeniyle coğrafi bilgi sisteminden beklenen özellikler artmış, verilerin güncellenmesi ve yönetilmesi ile ilgili olarak farklı yöntemler aranmaya başlanmıştır.

Bu beklentilerden bir tanesi, çok kullanıcılı ortam olarak adlandırılan ve mekânsal veriye birden fazla kullanıcının ulaşarak güncelleme yapmasına imkân sağlanmasıdır (Zeiler, 2002). Bir diğeri ise güncellenen verilerin özgeçmişlerinin sistem içerisinde depolanarak daha sonraki bir zaman dilimi içerisinde bu verilere ulaşılarak analizler yapılması konusudur.

Çok kullanıcılı ortam ifadesi ele alınırsa, coğrafi bilgi sistemlerinde sistem içerisindeki bir kullanıcının bir katman üzerinde çalışma yaparken bir başka kullanıcının da aynı katman üzerinde çalışma yapmasına imkân vermelidir. Bu özellik sistem içerisindeki tüm kullanıcıların güncel veriye ulaşmalarına ve üzerinde çalışma yapmalarına imkân sağlamaktadır. Bu özellik coğrafi bilgi sistemi projelerinin hızlı bir şekilde hayata geçirilmesi açısından önemli bir yere sahip olmaktadır. Özellikle Büyükşehirlerin Belediyelerinin coğrafi bilgi sistemleri içerisinde tek katman halinde tutulması gereken bina verilerinin üzerinde sistem içerisindeki tüm kullanıcılar hiçbir kısıtlama olmadan çalışabilmelidir.

Bir diğer hususu ise, coğrafi bilgi sistemlerinde depolanan verilerin zaman içerisinde geçirmiş oldukları değişikliklerin izlenmesi gereken uygulamalarda verilerin belli tarih veya tarihler aralığındaki durumlarının görüntülenebilmesidir. Verilen önceki zaman dilimlerine ait durumları daha sonra yapılacak çalışmaları yönlendirecektir.

Bir parselin görmüş olduğu değişikliklerin ele alındığı mülkiyet uygulamalarında, yerleşim alanındaki planların yerleşim alanının büyümesi ile görmüş olduğu değişikliklerin izlendiği imar uygulamalarında veya belli bir özelliğe sahip olan alanların ( yeşil alanlar, havza alanları ) zaman içerisinde gördükleri değişimlerin izlendiği çevre uygulamalarında verilerin özgeçmişlerinin saklanması ve istenildiği anda bu verilere ulaşılabilmesi gerekmektedir.

Coğrafi bilgi sistemlerinde ilişkisel veri tabanlarında depolanan verilerin çok kullanıcılı ortamda kullanılmaları ve verilerin özgeçmişlerinin tutulması bilgi sisteminin çok yönlü bir sistem olmasında önemli bir yere sahiptir. Sistem içerisinde depolanan mekânsal verilerin değişimlerinin tarihsel süreçlerinin izlenmesi ve bir katman üzerinde birden fazla kullanıcının çalışabilmesi, her bir kullanıcının yapmış olduğu değişikliklerin bir sistem yöneticisinin denetiminde güncel veri olarak

depolanması için kullanılan veri tabanlarında bir dizi yöntemler izlenmesi gerekmektedir.

Bu çalışmada ilişkisel veri tabanı yönetim sistemleri analiz edilerek verilerin çok kullanıcılı ortamda depolanması ve özgeçmişlerinin izlenmesi için bir yöntem belirlenmiş ve bu yöntemin kullanıldığı özgün bir uygulama sunulmuştur.

## 2. VERİ TABANI

Veri tabanı, birbiriyle ilişkili veriler topluluğudur; veri tabanı verilerin yanında aynı zamanda veriler arasındaki ilişkileri de saklar. Veri tabanının tanımlanması, veri tabanını oluşturan verilerin tip ve uzunluklarının belirlenmesini kapsamaktadır. Veri tabanının oluşturulması ise, veri tabanı yönetim sistemi kontrolü altında, bir depolama ortamına verilerin yüklenmesini ifade eder. Bir veri tabanı üzerinde işlem yapma kavramı ise, belirli bir verinin sorgulanması, veriler üzerinde meydana gelen değişiklikleri yansıtmak üzere veri tabanının güncellenmesi ve verilerden rapor üretilmesi gibi eylemlerin gerçekleştirilmesidir. (Yomralıoğlu, 2000)

Veri tabanı yönetim sistemleri zamanla çok daha fazla özellikler kazanmaktadırlar.

- Belirli bir tarzda organize edilmiş bilgi koleksiyonudur.
- Veri tabanının içindeki tablolar ise veri alanlarından oluşur.
- Kitaplıklar, uygulamalar ve yardımcı programların birleşmesinden oluşur.
- Verilerin saklanması ve yönetilmesi ile ilgili konulardaki ayrıntılardan veri tabanı yöneticilerini kurtarır.
- Kayıtların güncellenmesi ve kayıtlar üzerinde araştırma yapılması da mümkündür.

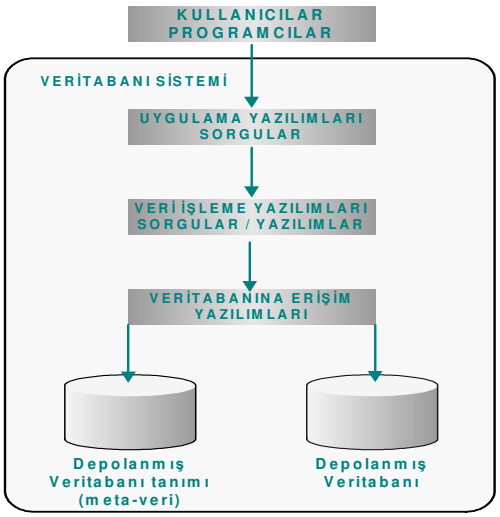
### 2.1. Veri tabanı Yönetim Sistemi

Veri tabanı Yönetim Sistemi, verilerin farklı programlar tarafından kullanılmasını sağlayan programlardır ( Laurini ve Thompson, 1992) . Veri tabanı yönetim sistemi çeşitli uygulamalar için veri tabanlarını tanımlama, oluşturma ve üzerlerinde işlem yapma yeteneği olan, genel amaçlı bir yazılım sistemidir.

Bir veri tabanı yönetim sisteminin temel işlevi, veri yönetimini uygulamalardan ayıran yazılım-veri bağımsızlığını sağlamaktır. Yazılım-veri bağımsızlığı, veri tabanlarının oluşum sürecinin en önemli etkenini oluşturmaktadır. Her bir programın kendi verisini ayrıca organize etmesi yerine, veri sadece veri tabanında, farklı yazılımların kullanımına olanak sağlayacak ortak bir tanım altında tutulur. Bu

durumda uygulama yazılımlarının, gereksinim duydukları veriyi veri tabanından çekebilme ve veriyi veri tabanına koyabilmeleri için veri tabanı yapısını bilmeleri gerekir.

Veri tabanının tanımı sistemin veri sözlüğünde bulunur. Veri sözlüğü, veri dosyalarının yapısı, veri elemanlarının tipi ve depolama formatı ile veri üzerindeki kısıtlayıcılar gibi bilgileri içerir. Veri sözlüğünde yer alan bilgiler meta-veri olarak adlandırılır (Şekil 2.1). Veri yönetiminin uygulamalardan ayrılması ve böylece veri paylaşımının, yani farklı programların aynı veriyi kullanabilmelerinin sağlanması için, bir veri tabanı yönetim sisteminin veri tabanına eş zamanlı giriş, veri fazlalığının kontrolü, veri bütünlüğü ve veri tabanının güvenliğinin sağlanması gibi fonksiyonları yerine getirmesi gerekir.




**Şekil 2.1:** İlişkisel veri tabanı bileşenleri

Veri tabanına eşzamanlı giriş, farklı uygulamaların aynı anda veri tabanına girebilmesi ve veri tabanında değişiklikler yapabilmesi açısından son derece önemlidir. Veri bütünlüğü, verinin veri tabanındaki doğruluğu ve tutarlılığını ifade eder. Veri tabanının güvenliği, veri tabanına yetkisiz girişlerin önlenmesi ile ilgilidir ve veri tabanının bütünlüğü açısından hayati önem taşır.

## 2.2. Veri Tabanı Tasarımı

Bir veri tabanı tasarımı veri tabanının konusunu oluşturan gerçeğin, veri tabanının oluşturuluş gereksinimi ve beklentileri çerçevesinde soyutlanarak veri tabanına aktarılmasını kapsar. Oluşturulacak veri tabanı farklı kullanıcıların gereksinimlerini karşılayabilmelidir. Bu nedenle veri tabanı tasarımında ilk adım, olası veri tabanı kullanıcı gereksinimlerinin belirlenmesidir. Söz konusu gereksinimler, veri tabanında yer alacak veri tiplerini ve verinin fiziksel olarak depolanması için kullanılacak olan veri yapılarını belirler. Gerçeğin veri tabanındaki sayısal temsili, onun belli bir perspektiften bir modeli olup, bir veri tabanı sisteminde gerek kullanıcılar ve gerekse bilgisayar tarafından anlaşılabilecek bir tarzda tanımlanması gerekir.

Geleneksel veri tabanı tasarımı kavramsal düzeyden fiziksel düzeye doğrudur. Kavramsal tasarımda, gereksinimlere göre kavramsal şema belirlenir. Kavramsal şema, fiziksel depolama yapılarının ayrıntılarına girmeden, varlıklar, veri tipleri, varlıklar arasındaki ilişki tipleri ve kısıtlayıcılar üzerinde yoğunlaşır. Kavramsal bir veri modelinde tanımlı bir şema genellikle doğrudan gerçekleştirilemez. Bu nedenle, geleneksel veri tabanı tasarımında kavramsal tasarımdan sonraki adım genellikle veri tabanı yönetim sisteminin seçimidir.

Fiziksel tasarım safhasında, verinin en yüksek verim için veri tabanında fiziksel olarak nasıl organize edilmesi gerektiği belirlenir. Depolama yapıları, kayıt formatları, kayıt alanları, veri tabanına giriş yol ve yöntemleri ile veri tabanının fiziksel gerçekleştirimini ilgilendiren diğer bütün detaylar tanımlanır. Bunun için, genellikle veri yapıları olarak bilinen, fiziksel veri modelleri kullanılır.

### 2.2.1. Veri Tabanı Tasarım Araçları

Yazılım teknolojisinin gelişmesi ve donanımlar ile iç içe girdiği büyük ağ sistemlerinde yazılacak uygulamaların daha anlaşılır bir şekilde geliştirilmesini sağlamak için standart modelleme ve analiz araçlarına ihtiyaç duyulmaktadır. Geliştirilecek uygulamaların analiz ve tasarım aşamasında yapılacak modelleme ne kadar anlaşılır olur ise ilerde ortaya çıkacak bir çok problemin şimdiden önüne geçilmiş olur.

Bu amaçlara hizmet etmek için geliştirilmiş CASE (Computer Aided Software Engineering ) araçları bulunmaktadır. Bu araçlara örnek olarak Microsoft Visio ve Rational Rose yazılımlarını örnek verebiliriz. Bu yazılımlar ile UML (Unified Modelling Language) olarak adlandırılan görüntüsel diyagramlar ve iş akış şemaları üretilebilmektedir (ISO, 2004) . UML modellerde geliştilecek uygulama içerisinde her bir parça ve bu parçalar arasındaki ilişkiler diyagramlar halinde ile gösterilir.

#### 2.2.1.1. UML Diyagramları

Nesneler arasında ilişki kurmak için UML bir takım grafiksel elemanlara sahiptir. Bu grafiksel elemanlar diyagram olarak adlandırılmaktadır. UML temel olarak dokuz diyagram türünden oluşur. Bir sistemin geliştirilmesi analiz, tasarım, uygulama geliştirme ve projenin uygulanması aşamalarından geçmektedir. İlk iki aşamada UML diyagramlar büyük ölçüde rol oynar. UML diyagramları bir programlama dili olmayıp bir diyagram çizme ve veriler arasındaki ilişkileri görmek için bir modelleme dilidir.

- **Sınıf (Class) Diyagram**

Gerçek dünyada aynı özelliklere sahip eşyalar araba, masa, bilgisayar şeklinde gruplandırıldığı gibi yazılım geliştirme ortamında benzer özellik ve fiillere sahip nesneler gruplandırılmaktadır. Bunlara "Sınıf(Class) “denir. UML' de bir sınıf dikdörtgen ile gösterilir (Şekil 2.2). Dikdörtgenin en üstünde sınıfın adı vardır.


**Şekil 2.2 :**Sınıf (Class) diyagram örneği

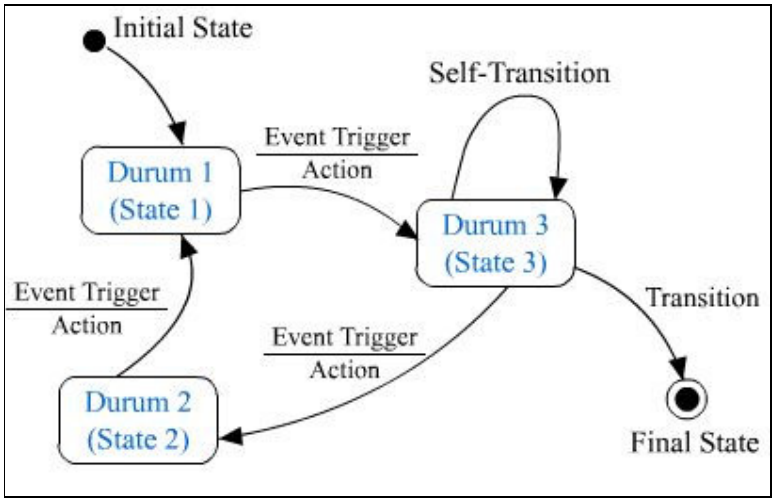
İşlevler özelliklerden farklı olarak birtakım bilgilere ihtiyaç duyabilir, veya birtakım bilgileri dışarıya verebilir, ya da bunların hiçbirini yapmaz. Şekil 2.2' deki işlev1, birtakım işler yapar bunun için dışardan bir bilgiye ihtiyaç duymaz ve dışarıya bir bilgi vermez, işlev2, işini yapabilmek için birtakım bilgilere ihtiyaç duyar. Örneğin, işlev2, eğer bir toplama işlemi yapıyorsa toplanacak elemanları ister. İşlev3 ise iş yaptıktan sonra işinin sonucunu sınıfın dışına verir.

- **Durum (State) Diyagramı**

Nesnelerin herhangi bir zaman içindeki durumunu gösteren diyagramlardır. Durum (State) diyagramları, bir nesnenin geçerli olduğu zaman dilimi boyunca sahip olacağı durumları modelleyen diyagramlar şeklinde tanımlanmaktadır. Örneğin, Ali nesnesi insan sınıfının gerçek bir örneği olsun. Ali 'nin doğması, büyümesi, gençliği ve ölmesi durum diyagramlarıyla gösterilir.

Bir başka örnek olan lamba ele alındığında, lambanın yaşam süresince 3 farklı durum söz konusu olmaktadır. Birinci durum lambanın kapalı olması, ikinci durum lambanın açık olması son durum da lambanın düşük enerji harcayarak yanması olmaktadır. Bu üç durum arasında geçişler mümkündür. Bu geçişleri sağlayan ise çeşitli olaylardır. Örneğin lambayı kapalı durumdan açık duruma getirmek için lambanın anahtarını çevirmek gerekir. Aynı şekilde lambayı düşük enerjili halde yanan durumundan açık durumuna getirmek için lamba anahtarını yarım çevirmek gerekir. Lamba nesnesi durum değiştirirken yaptığı işlere "aksiyon(action)", lambanın durum değiştirmesini sağlayan mekanizmaya ise "olay(event)" denilmektedir. Nesnelerin bu şekilde durumlarını modellemek için UML'deki

standart durum diyagramları kullanılmaktadır. Durum diyagramları bir nesneye ait dinamik modellemeyi içermektedir. Dinamik modellemeden kastedilen nesnenin ömrü boyunca gireceği durumları belirtmesidir (Şekil 2.3 ).

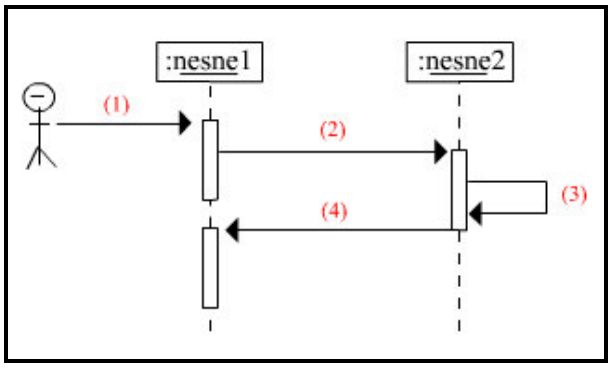



**Şekil 2.3 :** Durum (State) diyagram örneği

- **Ardışıklık (Sequence) Diyagram**

Sınıf diyagramları statik bilgiyi modeller. Oysa gerçek zamanlı sistemlerde zaman içinde değişen hareketler bu diyagramlarla gösterilemez. Bu tür zamanla değişen durumları belirtmek için Ardışıklık (Sequence) diyagramları kullanılır.

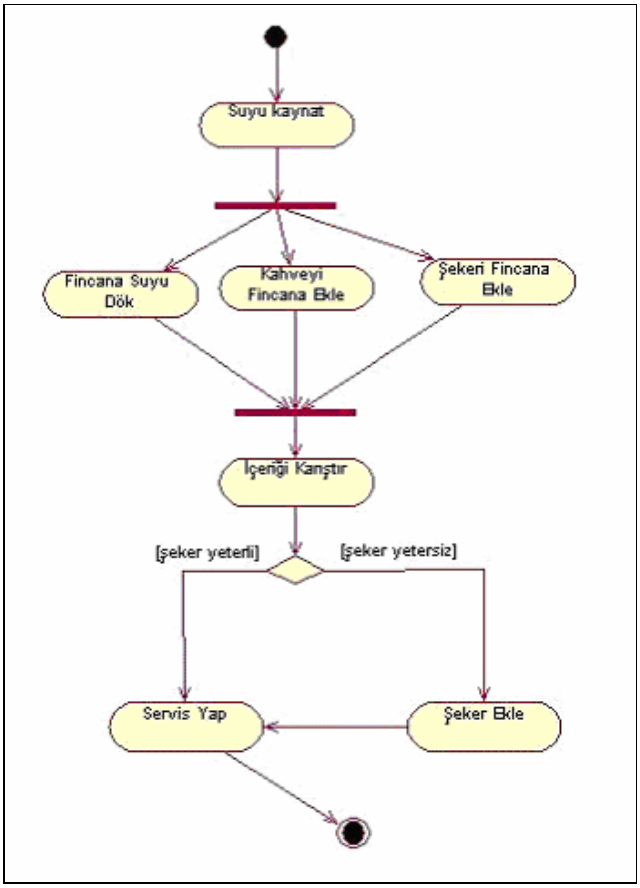
"Sequence" kelime anlamı olarak "birbirini takip eden, ardışıl olan, peşi sıra" anlamlarına gelmektedir. UML diyagramı olarak ise nesnelerin peşi sıra etkileşimde bulunmalarını ve nesnelerin zaman boyutunda birbirleri ile ilişkiye girmeleri anlamında kullanılmaktadır. Durum diyagramlarının aksine ardışık (sequence) diyagramları nesnelerin birbirleriyle haberleşmesini zaman boyutunda ele almaktadır. Bir diyagramın zaman bağlı olması, nesnelerin gerçekleştirdikleri aktivitelerin peşisıra gerçekleşmesi ve bu peşisıralığın belirlenen zaman dilimleri içerisinde meydana gelmesidir (Şekil 2.4).




**Şekil 2.4 :** Basit bir ardışık (sequence) diyagram örneği

- **Aktivite (Activity) Diyagram**

Bir nesnesinin durumu zamanla kullanıcı tarafından ya da nesnenin kendi içindeki işlevleri tarafından değişebilir. Bu değişim sırası Activite (Activity) diyagramlarıyla gösterilir. Algoritma tasarlanırken kullanılan akış diyagramlarının (flowchart) UML uyumlu hali olarak ifade edilmektedir (Şekil 2.5) .




**Şekil 2.5 :** Aktivite (activity) diyagram örneği

## 2.3. Veri tabanı Modeli

Veri tabanı modeli, gerçek dünyadaki varlıklar, olaylar, etkinlikler ve bunlar arasındaki ilişkiler hakkındaki verilerin temsil edildiği şeklidir. Varlıklar ve olayların kendi aralarında üç temel ilişkisi mevcuttur. Bunlar ;

Bire-bir / 1:1 (one-to-one)

Bire-çok / 1:M (one-to-many)

Çoka-çok / M:N (many-to-many)

Örneğin; kişi kimlik varlık sınıfı ile kişi ev adres bilgilerini içeren varlık sınıfı arasında 1:1 ilişki; bina varlık sınıfı ile daireler varlık sınıfı arasında 1:M ilişki; her bir parsel varlık sınıfı ile malik varlık sınıfı arasında M:N ilişki vardır.

Verilerin bilgisayar ortamında organizasyonu için uygun veri yapılarına ihtiyaç duyulmaktadır. Bir veri tabanı sisteminin kavramsal düzeydeki yani lojik şemada verilerin nasıl modelleneceği üzerinde klasik olarak kabul gören başlıca 4 farklı veri tabanı modeli vardır :

- Hiyerarşik veri modeli
- Ağ veri modeli
- İlişkisel veri modeli
- Nesne yönelimli veri modeli
- Nesne ilişkisel veri modeli

### 2.3.1. Hiyerarşik veri modeli

Hiyerarşik veri tabanı modeli ağaç modeli ile gösterilmektedir (Tsichritzis ve diğerleri 1982). Ağacın kökünden, gövde, dal ve yapraklarına doğru olan dallanma yapısı, veri tabanı için verinin modellenmesine yansıtılır. Bu model için sahip-üye, ya da ebeveyn-çocuk ilişkisi benzetmesi de yapılmaktadır. Başlangıçta bir kök, ve buna bağlı alt-dallanmalar modelin yapısını belirler. Dolayısıyla bir alt veri seti daima bir üst seviyedeki veri seti ile ilişkilidir. Bire-çok (1:M) ilişkisi açık bir şekilde görülürken çoka-çok (M:N) ilişkisi sağlanamaz. Aynı özellikleri taşıyan detaylar, dallanma durumunda tekrarlanabilmektedir. Bu da veri tekrarını dolayısıyla bellek sorununu gündeme getirir. Hiyerarşik veri modeli, hiyerarşik yapıdaki verilerin

güncellenmesi ve genişletilmesi için oldukça etkili olmasına karşın, hiyerarşik olmayan veya karmaşık ilişkilere sahip veriler için uygun değildir. Bu nedenle coğrafi bilgi sisteminde tercih edilen bir yaklaşım değildir.

### 2.3.2. Ağ veri modeli

Ağ veri tabanı modeli Bachman diyagramı olarak adlandırılan diyagramlar ile gösterilmektedir. Verilerin topolojik özellikleri bu diyagramlar üzerinde modellenmektedir (Elmasri ve Navathe, 1989). Ağ veri modelinde varlıklar arasındaki ilişkiler çoka-çok (M:N) şeklindeki bir yapıya sahiptirler. Varlıklar hiyerarşik modelde olduğu gibi organize edilirler. Burada farklı olan nokta, alt düzeydeki bir varlığın birden fazla üst düzeydeki varlık ya da varlıklar ile bağlantılı olmasıdır. İlişkiler ve bağlantıların önceden belirlenmesi halinde, ağ veri modelleri çok esnek ve kullanışlı bir yapıya sahiptir. Bu sistemlerde veri tekrarı en aza indirgenerek, veri çokluğundan kaçınılıp mevcut verinin çok daha fazla kullanımı sağlanır Ancak mevcut veri tabanının genişletilmesi, karmaşık bir durum ortaya çıkardığından oldukça zordur. Dolayısıyla ağ veri modeli en karmaşık modeldir denilebilir.

### 2.3.3. İlişkisel veri modeli

Bu veri modelinde objeler ile bunlar arasındaki ilişkiler ön plana çıkmaktadır. İlişkisel veri modelinin temelinde veriler arasındaki doğal ilişki yatar. Yani varlıklar arasında bire-bir (1:1) ilişki vardır. Veriler tablolar şeklinde düzenlenir. Tablolardaki her bir satır, varlığa ait bilgi içerirken, her kolon da varlıklara ait öznitelik bilgilerini içerir ve aynı satırda yer alan tüm öznitelik değerleri anahtarlar vasıtasıyla birbiriyle ilişkilendirilir. Anahtarlar vasıtasıyla tablolar arasında ilişki kurulurken, veri sorgulamasında bu ilişkilerden yararlanarak farklı tablolar arasında bağlantı sağlanıp, diğer tablolardaki verilere kolayca erişilir. Bu ilişki kuralları genel olarak SQL yazılım dilinde kodlanır.

İlişkisel veri modeli esnek yapıda bir model olarak kabul edilir. Kullanıcı tarafından yapılacak bütün sorgulama isteklerini karşılayabilecek şekildedir. Kullanıma başlandığı yıllarda coğrafi bilgi sistemleri için alternatifi olmayan bir veri tabanı modeli olarak tanımlanmıştır (Healey, 1991). Ancak nesneye yönelimli veri

modelinin zamanla geliştirilmesi ile bu fikir etkisini kaybetmeye başlamıştır. Veri ekleme veya veri çıkarma oldukça kolaydır, çünkü bu işlem tabloların sadece ilgili satırlarında gerçekleşmektedir.

### 2.3.4. Nesne yönelimli veri modeli

Karmaşık yapıdaki veri modellerinin eşleştirilmesindeki zorluklar yüksek seviyeli veri modellerinin oluşturulmasını gündeme getirmiştir. Yüksek seviyeli veri modeli için nesne yönelimli yazılım dilleri kullanılmıştır (Atkinson ve diğerleri , 1987) . Nesne yönelim (OO-Object Oriented) kavramları, ilişkisel veri tabanında veri tekrarının azaltılması ve verilere sıralı erişimdeki sorunların giderilmesi düşünceleriyle ortaya çıkmıştır. İlişkisel veri yapısında her bir varlık bir kayıtla ifade edilirken, öznitelikler ve bunlara bağlı değerler arasındaki mantıksal ilişkiler açıklanır. Nesne yönelimli yapılarda ise, veri, bağımsız objelerin bir kümesi biçiminde tanımlanır. Bunlar doğal yapıya göre benzer olguya sahip gruplar halinde organize edilirler. Farklı objeler ve sınıflar arasındaki ilişkiler belirgin bağlantılarla kurulur. Bir objenin karakteristik yapısı o objenin özniteliklerine göre ve bunlar arasındaki davranışları gösteren bir dizi işlem seti ile veri tabanında tanımlanır. Bu veriler, tanımlayıcı bir kod numarası ile veri tabanında tanımlanan bir obje içerisinde toplanırlar. Karakteristik değerlerin değişimine karşın, obje daima aynı kalır. Örneğin, bir "bina" objesi zaman içerisinde yapısal değişime uğramış olsa ya da kullanımı değişse, veri tabanında objenin sahip olduğu tanımlayıcı bağımsız kod numarası değişmeyecektir.

İlişkisel yaklaşımda her bir tablo arasındaki bağlantı bir tablodan diğerine tekrarlanan veriler ile sağlanmaktadır. Nesneye yaklaşımda ise veriler sadece bir kez veri tabanında bulunur ve veriye gösterge ile erişim hızı arttığı gibi komutların veri tabanı içerisinde tüm yönlerde geçişi de oldukça esnektir. Ancak, nesneye yaklaşımı günümüz coğrafi bilgi sistemi uygulamalarında arzu edilen düzeyde değildir. Çok az sayıda nesne yaklaşımlı veri tabanı bulunması ve coğrafi bilgi sistemi fonksiyonlarını yeterince destekleyememesi bunun başlıca nedenleridir.

### 2.3.5. Nesne ilişkisel veri modeli

Nesne ilişkisel veri modeli nesne yönelimli ve ilişkisel veri modeli arasındaki ortak özellikleri yansıtan, ilişkisel veri modelinin genişletilerek nesne yönelimli veri modeli ile entegre eden bir veri modelidir ve genişletilmiş ilişkisel veri modeli olarak da adlandırılır (Elmasri ve Navathe ,2000). İlişkisel model verileri tablolar halinde organize etmektedir. İlişkisel veri modeli matemaktiksel bir kurgu üzerine inşa edilmesine rağmen gerçek dünyayı modelleme bakımından sınırlı imkan sunmaktadır (Cooper R, 1995) . Nesneye yönelimli model teknikleri objelerin dinamik ve statik özelliklerini sunmaya imkan sağlamaktadır. Bu iki veri modelinin avantajlarından yararlanmak için son zamanlarda yapılan çalışmalar nesne ilişkisel model olarak adlandırılan model üzerine odaklanmıştır.

Günümüzdeki çoğu veri tabanı yönetim sistemleri nesne yönelimli özellikleri mevcut ilişkisel veri tabanı fonksiyonları ile birleştirmeye başlamışlardır. Oracle firması çıkarmış olduğu VTYS yazılımı olan "Oracle 8" ve ESRI firması "Geodatabase" veri modeli ile ilk uygulamaları sunmuştur.

Nesne ilişkisel veri modelleri objelerin davranışlarının veri tabanlarında temsil edilmesine imkan sağlamaktadır. Obje davranışları terimi objelerin görünen davranışlarının diğer hareketler ile olan etkileşimlerini tanımlamaktadır. Gerçek dünyada örnek olarak nehirlerin sürekli yukarı yönlü değilde aşağıya yönlü bir hareket içinde olması, iki kadastral parselin bir biri üzerine binmemesi hareketlerini gösterebiliriz. Obje hareketlerinin veri tabanı yönetim sistemlerinde temsil edilmeleri coğafi bilgi sistemlerinde spesifik uygulamaların gerçekleştirilmesine imkan sağlamıştır.

## 2.4. Verilere Erişim

İlişkisel veri tabanlarına erişim SQL (Structured Query Language) olarak adlandırılan standart arayüzler ile sağlanmaktadır ( Melton 1990) . SQL arayüzü ile kullanıcılar yaptıkları sorgulamalar ile verilere erişebilmektedirler SQL veri tipleri integer, smallint, character, decimal, numeric, real,float, double, date olmaktadır.

SQL tarafından kullanılan operatörler den bazıları ; SELECT, INSERT, UPDATE, DELETE, JOIN, UNION, karşılaştırmalar (=, >, >=, >, <, <=, <, [NOT], LIKE, IS [NOT] NULL, AND, OR, NOT, istatistikler (COUNT,SUM, AVG, MAX, MIN) sıralamalar (GROUP, ORDER) olmaktadır. Tipik bir SQL sorgulama ifadesi ise aşağıdaki gibi olmaktadır.

SELECT öznitelikler
FROM tablolar
WHERE koşullar
HAVING istatistiki koşullar
GROUP BY gruplandırma özniteliği
ORDER BY sıralama özniteliği

### 2.5. Verilerin güvenliği

Veri tabanı yönetim sistemleri depoladıkları verileri hatalı kullanımlara karşı korumaktadır. Veri tabanları veri güvenliğini sağlamak için kullanıcıların ve verilerin farklı seviyelerde belirlendiği çok seviyeli bir kontrol mekanizması sunmaktadır. Bu kontrol mekanizması her bir kullanıcının kendi kullanıcı adı ve şifresi ile kendisi için önceden belirlenmiş seviyesindeki ve bu seviyeden düşük seviyedeki verilere erişmesini sağlamaktadır.

# 3. VERİ TABANI YÖNETİM SİSTEMLERİNDE MEKANSAL VERİLERİN DEPOLANMASI

Coğrafi bilgi sistemleri içerisinde veri tabanlarının kullanılmasına yönelik araştırmalar, içerisinde mekansal verilerin depolama ve erişim algoritmalarının geliştirildiği mekansal veri tabanları merkezinde yoğunlaşmıştır (Samet, 1989). Bu yoğunlaşmanın nedeni ise veri tabanlarının çoğu mekansal verilerin farklı tiplerinin desteklenmesi için elverişli bir model sunamamasıdır (Egenhofer 1992). Bu nedenle araştırmacılar dikkatlerini nesne ilişkisel veya kural bazlı sistemlerin geliştirilmesine yöneltmiştir (Medeiros ve Pires, 1994).

Bilindiği üzere kurulan ilk bilgi sistemlerinde veriler dosya bazlı olarak saklanmakta (Gutig, 1994 ) ve bu saklama işlemi gereksiz verilerin hacimlerinin artmasına ve verilerin farklı uygulamalar tarafından kullanılmasına imkan vermemekteydi (Elmasri ve Navathe 2000). Dosya bazlı bilgi sistemlerinin veri ile ilgili tüm tanımlamaların uygulama programlarının içinde olması ve verilere ulaşım ve yönetim açısından kontrol olmaması gibi iki önemli dezavantajı bulunmaktadır (Begg ve Connolly, 2002) .

Zaman içerisinde kaydedilen gelişmeler ile bilgi sistemleri için hiyerarşik ve network veri modeline sahip veri tabanı yönetim sistemleri ortaya konmuştur. Bu sistemler dosya bazlı sisteminin çoğu problemlerini ortadan kaldırırken bunun yanında verilere erişimin karmaşık özel programları ile olması gibi yeni problemler ortaya çıkarmıştır. Sonunda ilişkisel veri tabanı modeli geliştirilerek bilgi sistemlerinde yerini alması sağlandı. Bu sayede gereksiz verilerin azaltılması, verilerin paylaşımı, güvenliği gibi konularda kullanıcılarına kolaylıklar getirmiştir ( Hoffer ve diğerleri 2004).

İlişkisel veri tabanları sayı, karakter, tarih ve metin tipindeki verileri yönetmesine rağmen mekansal veriler ve resimler gibi karmaşık yapıdaki verileri yönetmede aynı yetenekleri gösterememektedir (Brown ve diğerleri 1999) . İlişkisel veri tabanlarında karmaşık objelerin depolanması birbirleriyle ilişkili satırlar ile mümkün olduğundan bu objelerin sorgulanması veri tabanı performansını olumsuz yönde etkilemektedir

(Cevero ve diğerleri 2001) . Bu olumsuzlukların giderilmesi için nesne yönelimli ilişkisel veri tabanlarının oluşturulmasına zemin hazırlamıştır.

Coğrafi bilgi sistemleri nesne yönelimli veri tabanı yönetim sistemlerinin en başarılı örneklerinden biridir. Nesne yönelimli veri tabanı yönetim sistemleri mekansal veri sunucusu olarak adlandırılan programlar sağlamaktadır ( Goodchild ve diğerleri 2001) ve bu yazılımlar sayesinde kullanıcılar mekansal veri tipleri üzerinde işlem yapabilmekte ve bu verileri yönetebilmektedir (Chawla ve diğerleri 1999). Bu yazılımlarda SQL özelliği yeni fonksiyon ve işlemler ile zenginleştirilmiştir ve ayrıca veri tabanlarına ise mekansal verilerin etkili bir biçimde yönetimi için gerekli özellikler eklenmiştir ( Rigaux ve diğerleri 2002 ). Bu özellikler kullanıcılara geometrileri karşılaştırma ve mekansal analizler yapma gibi imkanlar sağlamaktadır.

### 3.1. Mekansal Veri Sunucuları

Veri tabanı yönetim sistemlerinde mekânsal veriler karakteristik özellikleri, boyutları ve uzaydaki konumu belli olan objeler olarak tanımlanmaktadır (ISO, 1998). Veri tabanı yönetim sistemlerinde mekansal verilerin depolanması standartlara uygun bir depolama olmamaktadır. Örneğin bir yol verisi yolu oluşturan noktaların koordinatları bir tablo ile temsil edilebilir, fakat yolu tek bir obje olarak göstermek mümkün olmamaktadır.

Bu sınırlamaları ortadan kaldırmak için nesneye yönelimli verirabanları mekansal veri sunucusu adı verilen özel programlar ile kabiliyetleri artırılmıştır. Mekansal veri sunucuları veri tabanı yönetim sistemlerinde mekansal veriler için üç temel fonksiyon sağlamaktadır ( Chawla ve Shekhar 2003);

- Mekansal olmayan verilere ek olarak mekansal verilerin de yönetimi sağlamak
- Veri modellerinde mekansal veri tiplerini desteklemek ve bu verilere sorgulamalar ile erişebilmek
- Verilerin indekslenme mekanizmaları ile mekansal işlemler için gerekli algoritmalarını sağlamak

### 3.1.1. İndeksleme

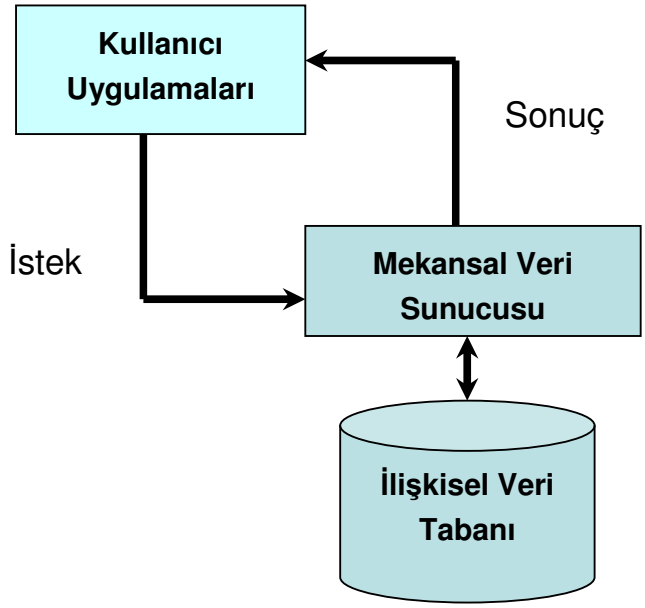
Mekansal veri sunucularının sağladığı önemli özelliklerden biri olan indeksleme mekanizması kullanıcılara sorgu kriterlerine uyan verileri hızlı bir şekilde sunmaya imkan sağlamaktadır. İndeks dosyası bir kitabın sonunda yer alan ve kitapta geçen önemli terimlerin alfabetik listesini ve kitapta terimin geçtiği sayfaları belirten bölümü gibi düşünülebilir. Bu sayede indeks içerisinde yer alan terimlerin kullanıldığı cümle veya cümleler ilgili sayfa numaraları kullanılarak kolayca bulunabilir. Eğer böyle bir mekanizma olmasaydı bu terimin kitapta geçtiği yerler, ancak kitabın baştan itibaren kelime kelime araştırılmasıyla bulunabilecekti. İndeksler terimin hangi sayfada geçtiğini kesin olarak belirtirler ve başka da hiçbir bilgi içermezler. Veri dosyalarındaki indeks kavramı da yukarıda tanımı yapılan bir kitaba ait indeks gibidir ( Banger G, 1998).

Bir veri dosyasına ait indeksler, bu dosyaya ait tek bir indeks dosyasında saklanırlar. İndeks dosyasında, terimin adı (kolon adı) ve bu terimin (kolonun) ana dosyadaki yerini işaret eden bir belirteç bulunur. Bu belirteçlere göre söz konusu kaydın ana dosya boyunca diskin hangi bloğunda yer aldığı belirlenir ve belirlenen bu yerden aranan bilgi kolayca bulunur.

İndeks dosyaları iki veya üç kayıt kolonundan oluşur. İlk alanda aynı türden verilerin sıralandığı anahtar kolon (key field), ikincil alanda ise kaydın diskteki başka bir deyişle ana dosyadaki adresini içeren belirteç bulunur. Bunlar sıralı verilerden oluşan kolona veri dosyasının birincil anahtarı (primary key) denir. Veri dosyasına ait her kayıt için indeks dosyasında bir indeks kaydı vardır. Parsellerle ilgili verilerin depolandığı bir veri tabanında parsel numarasına adına göre sıralanmış bir indeks düşünülürse sadece “parsel ” kolonundan oluşan indeks anahtarının tek anlamlı bir kayıt belirtmesi mümkün olmayabilir. Çünkü numaraları aynı olan fakat ada numaraları farklı olan parseller tek anahtar kolonla ayırt edilemezler. Bu gibi durumlarda ikincil anahtarlar kullanılır. Bu örnek de ikincil anahtar “ada numarası” kolonu olur. Veri tabanı yönetim sistemleri kullanıcılara verileri sunarken bu indeks yapısını kullanmaktadır (Begg ve Connolly, 2002) .

### 3.1.2. Verilere Erişim

Mekansal veri sunucu yazılımları sayesinde veri tabanı yönetim sistemlerinde mekansal bilgilerin uzmanlık gerektirmeyecek bir dizi işlemler sonunda kolaylıkla depolanması ve yönetilmesine imkan sağlanmıştır. Bu yazılımlar veri tabanları ve kullanıcı uygulama yazılımları arasında bir arayüz işlevi yapmaktadır (Şekil 3.1).




**Şekil 3.1 :** Mekansal veri sunucusu

Mekansal veri sunucuları iletişim için bilgisayar ağı ( network ) üzerinde TCP/IP protokolü kullanmaktadır. TCP/IP protokolü ağ üzerinde bir bilgisayarın tanımlanması, konumlandırılması ve bilgi alışverişi için kullanılan bir iletişim protokolüdür. TCP data aktarımı için IP ise bilgisayarın konumlandırılması ve verinin ulaşımı için yön tayininde kullanılmaktadır.

Mekansal veri sunucuları istemci / sunucu mimarisine sahiptir. Kullanıcı sunucuya istekler gönderir, sunucu almış olduğu bu isteğin sonuçlarını oluşturur ve kullanıcıya gönderir.

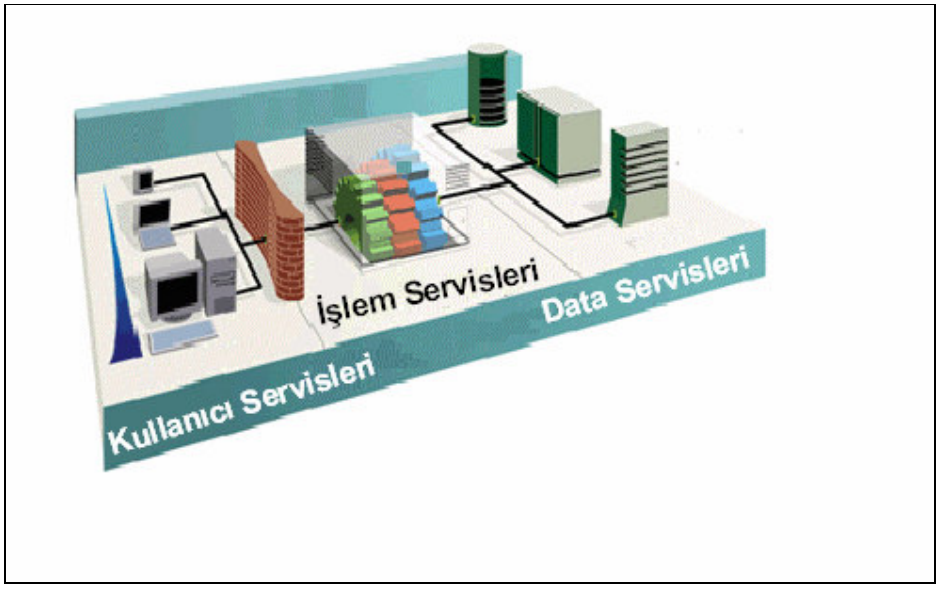
### 3.1.3. Çok Katlı Mimari

Veri, uygulama ve kullanıcı işlemlerinin birbirlerinden kesin hatlarla ayrıldığı sistemler çok katlı mimari olarak adlandırılır (Şekil 3.2) . Burada bağımsız olarak ayrılan işlemler ise;

**Veri Servisleri (Data Services) :** Veri tabanı üzerinde gerekli operasyonların tanımlandığı katmandır. Genelde tek bir obje içinde toplanabilirler. Veri tabanına bağlantıyı yapıp veri akışını sağlamak bu servisin görevidir. Ancak dağıtık bir veri tabanı uygulaması yapılıyorsa bu durumda her veri tabanı ve bu veri tabanı üzerindeki temel veya özel fonksiyonlar (query : sorgulama, insert : kayıt girişi, delete : kayıt silme, update : kayıt güncelleme) için ayrı ayrı tanımlanabilir.

**İşlem Servisleri (Busines Services) :** Bu bölümde işlemler içeriklerine göre tanımlanır ve birbirinlerinden ayrılır. Ancak bir işlem kendisinden farklı bir işlem ile birlikte değerlendirilebilir. Örneğin bir işletmenin malzemelerin kayıtlarını yapan bir programda farklı malzemelerin bir arayüzde sisteme kaydedilmesi şeklinde olabilir.

**Kullanıcı Servisleri (User Services) :** Kullanıcı tarafında gerçeklenecek olan bir takım işlemler için hazırlanan objeler bu bölümde yer alır ve basit işlemler bu servislerde bulunur.




**Şekil 3.2 :** Çok katlı mimari

Fiziksel olarak data servisleri veri tabanının bulunduğu bilgisayarda yer alır ve hiçbir servis gidip veri tabanından doğrudan veri çekmez. İşlem servisleri ise iş yükünün paylaştırılması açısından bir uygulama sunucusu (application server) olarak tabir edilen ayrı bir bilgisayarda veya performans dikkate alınarak veri tabanının olduğu bilgisayar veya kullanıcı bilgisayarında yer alabilir. Kullanıcı servisleri ise kullanıcı bilgisayarında veya uygulama sunucusu adı verilen bilgisayarda yer alır.

Data servisleri ve işlem servisleri birer DLL dosyası olabilirken, kullanıcı servisleri yapılacak işe göre bir EXE, ASP, Java Apleti veya Activex olabilir. Uç birim gerekli fonksiyonlar için işlem servislerine bağlanır. İşlem servisleri veri işlemek için data servislerine bağlanır. İşlemler bu şekilde yürütülür.

Bunca katmanın neden kullanıldığı sorusuna cevap olarak aynı şeyi tekrar tekrar yazmaktan kurtulmak ve işlemleri basitleştirmek olarak verilebilir. Ancak bunun önemli bir başka faydası bu üç servis tam olarak tanımlandıktan sonra kullanıcı tarafında hangi arabirimin kullandığının çok önemli olmamasıdır. Kullanıcı tarafı "Thin Client" tabir edilen sadece veri alıp ekrana veri gösteren bir arabirim haline gelmektedir. Bu şekliyle herhangi bir programlama dili veya araç ile bu kısmı yazmak oldukça kolaylaşmaktadır. Diğer taraftan bu servisler ayrı ayrı tanımlandığından ister dağıtık ister tek bir makine üzerinde olsun sorunsuz çalışabilmektedirler.

### 3.1.4. Standartlar

1980' lerden 2000' lere kadar mekansal verilerin depolanması ile ilgili pek çok standart yayınlanmış olmasına rağmen bu standartlar arasında bir uyuşum bulunmamaktadır. 1995 yılında kurulan ISO/TC 211 (International Organization for Standardization Technical Committee) uluslar arası organizasyonu uyulması gereken mekansal veri standartlarını belirlerken, OGC (Open Geospatial Consortium) ise mekansal verinin yönetimi ile ilgili çalışmalar yapmıştır. 1999 dan itibaren ise ortak hareket etme kararı almışlardır (Östensen, 2001 ) . Bu oluşumun amacı farklı coğrafi bilgi sistemleri arasında veri alışverişini ve veri modellerinin standartlarını ortaya koymaktadır ( Rigaux ve diğerleri 2002 ). ISO/TC 211 komitesi mekansal verilerin standardizasyonundan sorumlu kuruluş olarak hareket etmektedir (Carson G, 2000).

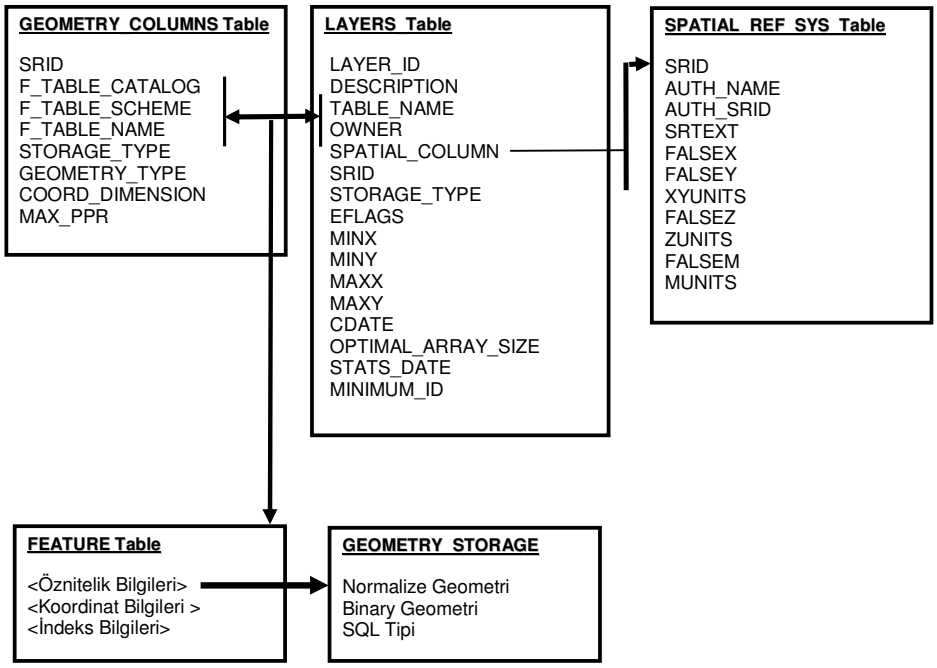
OGC konsorsiyumu 1994 yılında kar amacı güdülmeden coğrafi bilgi sistemi tekniklerinin birbiri ile uyumunu geliştirmek isteyen kuruluşların katkısı ile kurulmuştur. OGC yazılım özellikleri ile ilgilenirken, ISO TC/211 ise veri standartlarının belirlenmesi konusunda yoğunlaşmıştır.

### 3.2. Verilerin Depolandığı Tablolar

Coğrafi veri tabanı özelliğine sahip veri tabanlarında katmanlar mekansal veri sunucuları tarafından yönetilmektedir. Bu sunucuların yönetmiş olduğu bilgiler ;

- Katman sahibi, tablo ve kolon adı
- Projeksiyon bilgisi (Koordinat Referans Sistemi)
- Geometri tipi
- İndeksleme parametreleri
- Tablo oluşturma parametreleri
- Katman tanımları

Olmaktadır ve bu bilgiler veri tabanı yönetim sistemlerinde tablolar halinde yönetilmektedir (OGIS, 1999). Veri tabanlarında mekansal bilgilerin depolanması için kullanılan tabloları ve bu tabloların birbirleri ile olan ilişkileri Şekil 3.3' de gösterilmektedir.




**Şekil 3.3 :** Veri tabanlarında mekansal veri tabloları ve ilişkileri ( OGC, 1999)

Katman bilgileri veri tabanlarında Layers, Spatial_Ref_Sys ve Geometry_Columns tabloları olarak adlandırılan üç ana tablo tarafından yönetilmektedir (OGIS, 1999). Layers tablosu her bir mekânsal obje için bir satır kaydetmektedir. Uygulamalar mevcut mekânsal bilgi kaynaklarını öğrenmek için layer özelliklerini incelemektedirler. Layer özellikleri mekansal veri sunucuları tarafından mekânsal kolonu kontrol etmek, geometri değerlerini indekslemek ve ortak veri tabanı tablolarını yönetmek için kullanılmaktadır.

Geometry_Columns tablosu geometri tipi için bir satır kaydeder. Geometri kolon tablosunda her bir satır bir projeksiyon bilgisi içerir. Yeni bir geometri tablosu oluşturulduğunda kolon adı ve projeksiyon bilgisinin Id değeri Geometry_Columns tablosuna eklenir.

Projeksiyon bilgisi Spatial_Ref_Sys tablosunda saklanır. Projeksiyon bilgisi bir geometri için koordinat sistemini tanımlar ve geometri için kullanılan nümerik koordinat değerlerini anlamlandırır.

### 3.2.1. Layers Tablosu

Veri tabanlarında depolanan her bir katman için oluşturulan Layers tablosunun içerdiği kolon bilgileri ve bunların veri tipleri aşağıdaki Tablo 3.1'de gösterilmektedir (OGIS, 1999).

**Tablo 3.1 :** Layers tablosu

| **Kolon Adı** | **Veri tipi** |
|---|---|
| Layer_Id | Integer |
| Description | Varchar |
| Table_Name | Varchar |
| Owner | Varchar |
| Spatial_Column | Varchar |
| Srid | Integer |
| Storage_Type | Integer |
| Gsize | Float |
| Gsize2 | Float |
| Gsize3 | Float |
| Minx | Float |
| Miny | Float |
| Maxx | Float |
| Maxy | Float |
| Cdate | Integer |
| Layer_Config | Varchar |
| Optimal_Array_Size | Integer |
| Minimum_Id | Integer |

**Layer_id :** Mekansal veri sunucusu tarafından verilen numara.

**Description :** Katman açıklama bilgisi (opsiyonel ).

**Table_Name :** Geometri kolunun içeren tablonun adı.

**Owner :** Öznitelik (business) tablosuna sahip olan kullanıcının adı .

**Spatial_Column :** Geometri kolonu olan obje tablosundaki kolonun adı.

**Srid :** Koordinat geometri tablosu için kullanılan projeksiyon sisteminin ID değeri.

**Storage_Type :** Geometri tablosu için kullanılan depolama tipi.

0 = Normalize geometrik yöntem ile veri depolama

1 = Binary geometri yöntemi ile veri depolama

2 = SQL tipi geometri yöntemi ile veri depolama

**Gsize,Gsiz2, Gsize3 :** Geometri grid indeks değerleri

**Minx, Miny, Maxx, Maxy :** Mekânsal verinin kapsadığı alanın koordinat değerleri.

**Cdate :** Katmanın oluşturulduğu tarih.

**Optimal_Array_Size :** Verilerin veri tabanından kullanıcı uygulamasına hızlı aktarılması için kullanılan veri boyutu.

**Layer_Config :** Tablo oluşturma parametrelerinin anahtar kelimeleri

**Mınımum_id :** Katman için minimum obje Id değeri.

### 3.2.2. Spatial_Ref_Sys Tablosu

Katmanların sahip oldukları projeksiyon bilgilerinin depolandığı Spatial_Ref_Sys tablosu ve içerdiği kolon bilgileri ve bunların veri tipleri Tablo 3.2' de gösterilmektedir.

**Tablo 3.2 :** Spatial_Ref_Sys tablosu

| Kolon Adı | Veri tipi |
|---|---|
| Srid | Integer |
| Description | Varchar |
| Auth_name | Varchar |
| Auth_srid | Varchar |
| Falsex | Float |
| Falsey | Float |
| XYunits | Float |
| Falsez | Float |
| Zunits | Float |
| Falsem | Float |
| Munits | Float |
| Srtext | Varchar |

**Srid :** Veri tabanında bir projeksiyon sistemine karşılık gelen bir tamsayı .

**Description :** Projeksiyon bilgisinin kısa tanımı (örn: UTM Zone 17).

**Auth_Name :** Projeksiyon sisteminin standart adı

**Auth_Srid :** AUTH_NAME alanında belirtilen Projeksiyon sisteminin Id değeri.

**Falsex, Falsey :** Projeksiyon sisteminde x ve y değerlerine eklenen öteleme değerleri.

**XYunits :** Projeksiyon sisteminin ölçek çarpanı.

**Falsez :** z değerlerine eklenen öteleme değeri.

**Zunits :** z değerlerini dönüştürmek için kullanılan ölçek faktörü.

**Falsem :** Uzunluk değerlerini dönüştürmek için kullanılan öteleme değeri.

**Munits :** Uzunluk değerlerini dönüştürmek için kullanılan uzunluk ölçek faktörü

**Srtext :** Projeksiyon sisteminin bilinen adı.

### 3.2.3. Geometry_Columns Tablosu Örneği

Her bir geometri tipinin bir satır ile kaydedildiği Geometry_Columns tablosu ve içerdiği kolon bilgileri ve bunların veri tipleri Tablo 3.3'de gösterilmektedir (OGIS, 1999).

**Tablo 3.3 :** Geometry_Columns tablosu

| Kolon Adı | Veri Tipi |
|---|---|
| F_table_catalog | Varchar |
| F_table_schema | Varchar |
| F_table_name | Varchar |
| F_geometry_column | Varchar |
| Srid | Integer |
| Storage_type | Integer |
| Geometry_type | Integer |

**F_table_catalog, F_table_schema, F_table_name :** Geometri kolonunu içeren obje tablosunun adı.

**F_geometry_column :** Obje tablosunda geometri kolonunun adı.

**Srid :** Koordinat geometrisi için kullanılan projeksiyon sisteminin Id değeri.

**Storage_type** : Geometri kolonu için kullanılan depolama birimi.

0 = Normalize geometri

1 = Binary geometri

2 = SQL Tipi

**Geometry_type :** Bu kolonda depolanan geometri değelerinin tipleri.

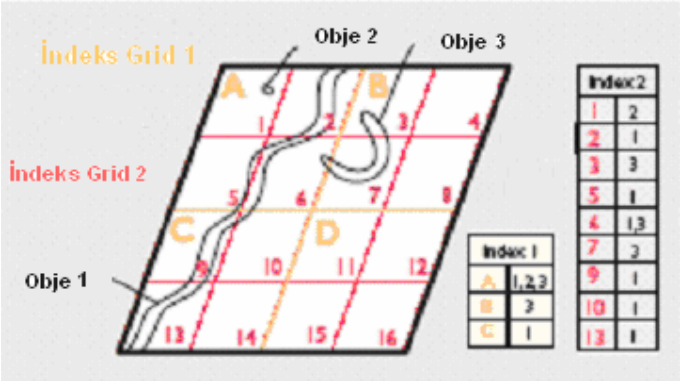
0 = Geometri
1 = Nokta
2 = Eğri
3 = Çizgi
4 = Yüzey
5 = Poligon
6 = Koleksiyon (Obje Topluluğu)
7 = Çoklu Nokta
8 = Çoklu Eğri
9 = Çoklu Çizgi
10 = Çoklu Yüzey
11 = Çoklu Poligon

### 3.2.4. İş Tablosu , F Tablosu ve S Tabloları

Mekansal Veri Sunucuları katmanlar için veri tabanında başka tablolar daha açmaktadır. Bu tablolardan bir tanesi katmanların sahip olduğu öznitelik bilgilerinin depolandığı iş tablosudur (Business Table). Bu tabloda her bir satır sahip olduğu ve tekrarlanmayan bir numarası (Id) bir objeye karşılık gelmektedir (OGIS, 1999).

Bir diğer tablo olan F Tablosu olarak adlandırılan tabloda ise katmanda yer alan objelerin koordinat bilgileri binary formatda depolanır. Bu tablo Geometri Tablosu olarak da adlandırılmaktadır. F tablosu ile iş tablosu arasında bire bir ilişki (one to one ) bulunmaktadır.

Son olarak depolanan tablo ise katmanların projeksiyon bilgilerinin depolandığı S tablosudur. S Tablosu objelerin yapılan sorgulamalar sonucu hızlı bir şekilde getirilmesi için gerekli olan grid bilgilerini depolamaktadır. İndeks mekanizması sayesinde bu tabloda oluşturulan grid sayısı kadar kayıt bulunmaktadır (Şekil 3.4).



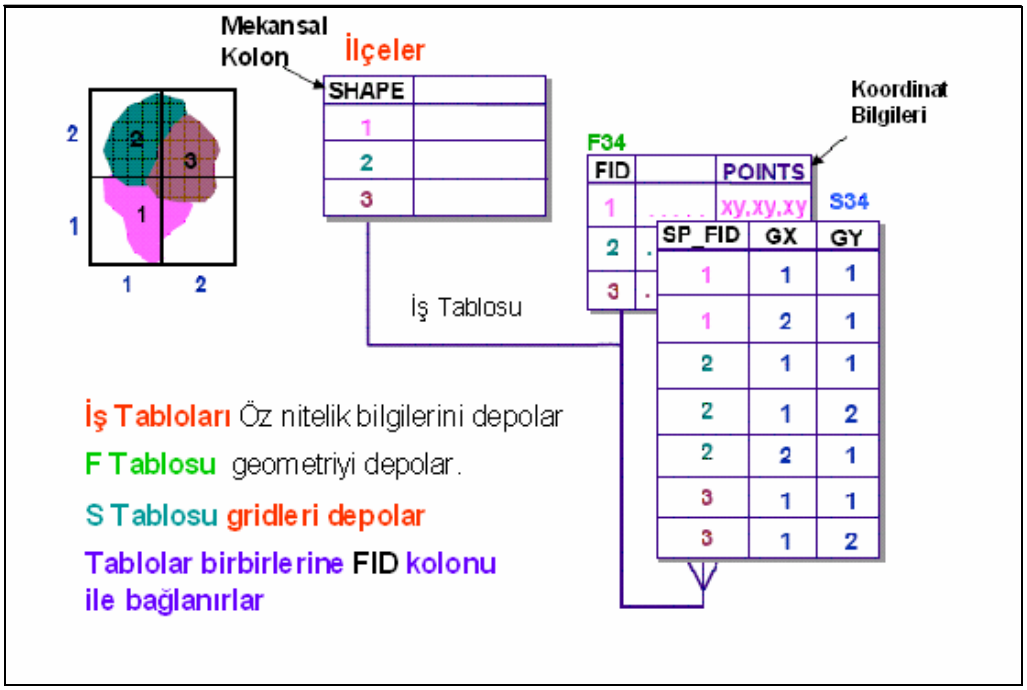

**Şekil 3.4 :** İndeks mekanizmasının geometrik görünümü

F ve S tablosundaki kayıtlar birbirleri ile tekrarlanmayan Id değerleri ile bağlantılı durumdadır. Bu bağlantı bire çok ( one to many ) bağlantı şeklindedir. Mekansal veri sunucuları katmana herhangi bir obje eklendiğinde aralarındaki bu bağlantıyı yönetmektedir.

Kullanıcı herhangi bir katmanı çalışma arayüzüne eklemek istediğinde mekansal veri sunucusu öznitelik bilgilerini iş tablosundan ve geometrik bilgileri ise F tablosundan getirmekte ve bu işlemler kullanıcılar tarafından fark edilmemektedir. Çalışma sırasında tüm bağlantılar mekansal veri sunucusu tarafından sağlanmaktadır.

S tablosu grid bilgilerini ve bu gridler içerisinde kalan objelerin tekrarlanmayan Id değerlerini içerdiğinden kullanıcı tarafından bir sorgulama yapıldığında mekansal veri sunucusu söz konusu verilerin Id değerlerinin hangi grid içerisinde bulunduğunu belirler ve hızlı bir şekilde verinin bulunduğu grid veya gridlerin bulunduğu alanı görüntüler. F tablosu ile S Tablosu arasında bire çok şeklinde bir bağlantı vardır (Şekil 3.5).

Şekil 3.5' de görüldüğü gibi mekansal indeks uygulanmış bir katmanın çevresi gridler tarafından kaplanır.




**Şekil 3.5 :** İş tablosu, F tablosu ve S tablolarının birbirleri ile ilişkileri

### 3.3. Mekansal Verilerin Depolanma Yöntemleri

Mekansal veri sunucuları mekansal verileri tablolar halinde depolamakta, bu tablolardaki her bir satır bir objeye karşılık gelmekte ve sütunlar ise objelerin öznitelik bilgilerini içermektedir (Rigaux ve diğerleri, 2002). Mekansal bilgiler ve veri tabanlarındaki karşılıkları Tablo 3.4' de gösterilmiştir. Bu öznitelik bilgileri tarih, açıklama ve sayılar olabilmektedir ve objelerin koordinat bilgileri ayrı bir kolonda depolanmaktadır. Veri tabanlarında tablolardaki herhangi bir satırdaki istenilen bilgilere erişilmesine imkan sağlayan SQL arayüzü bulunmaktadır. SQL ifadeleri birden fazla tablolarda bulunan bilgileri, tablolar arasında ilişkileri sağlayan ortak kolonları kullanarak verilere erişim sağlamaktadır.

Mekansal veri sunucuları veri tabanlarının yapılarını değiştirmemekte veri tabanında tablolar halinde depolanmış mekansal verilerin yönetilmesini sağlamaktadır.

**Tablo 3.4 :** Mekansal bilgiler ve veri tabanı karşılıkları

| Mekansal Bilgi | Veri tabanı Elemanı |
|---|---|
| Obje | Satır |
| Öznitelik | Kolon |
| Katman | Tablo |

Mekansal veri sunucuları veri tabanı tarafından katmanların geometrileri için sağlanan fiziksel depolama yeteneklerini kullanmaktadır. Mekânsal verilerin veri tabanlarında depolanabilmeleri için günümüzde 3 farklı yöntem kullanılmaktadır (OGIS, 1999).

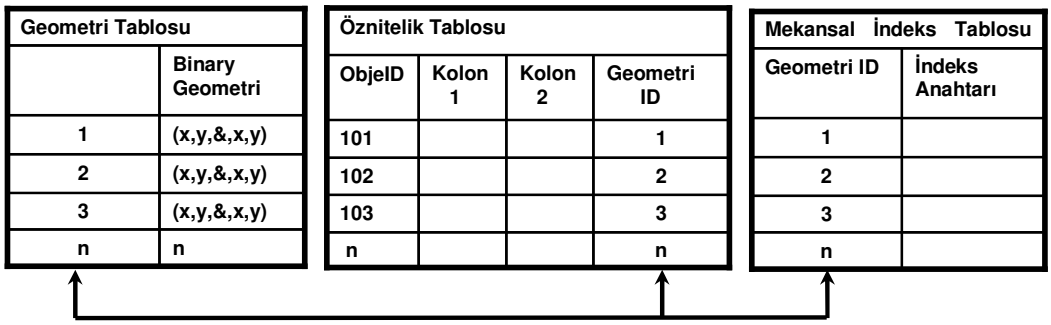
- Normalleştirilmiş Geometri Depolama Yöntemi
- Binary Geometri Depolama Yöntemi
- SQL tip sistemlerin genişletildiği Geometrik yöntemler

Veri tabanları üzerinde çalışan mekansal veri sunucuları bu depolama yöntemlerini kullanarak mekansal verileri depolamak depolamaktadır.

Mekansal veri sunucu yazılımları üzerinde çalıştığı veri tabanının veri depolama sistemlerini kullandıkları gibi veri tabanları üzerinde kendisinden farklı bir yöntem ile de verilerinin depolanmasını sağlayabilmektedir.

### 3.3.1. Binary Geometri Yöntemi

Mekânsal verinin yönetimi için 3 şemadan biri olan binary geometri yönteminde bir objenin koordinat değerleri BLOB denilen binary objeler halinde kaydedilir.BLOB ayrı bir Geometri tablosunda yönetilir. Bu geometriye ulaşabilmek için ilgili tablolardaki Geometri Id kolonları kullanılmaktadır. Şekil 3.6'de binary geometri şemasındaki İş Tablosu (Business Table) ile Geometri tablo arasındaki ilişki tanımlanmaktadır.




**Şekil 3.6 :** Geometri, iş ve mekansal indeks tabloları arasındaki ilişki

Binary geometri tablosunda yer alan kolon adı ve bu kolonlarda depolanan veri tipleri Tablo 3.5'de gösterilmiştir.

**Tablo 3.5 :** Binary geometri tablosu

| **Kolon Adı** | **Veri Tipi** |
|---|---|
| Fid | Integer |
| Numofpts | Integer |
| Entity | Smallint |
| Eminx | Float |
| Eminy | Float |
| Emaxx | Float |
| Emaxy | Float |
| Eminz | Float |
| Emaxz | Float |
| Min_measure | Float |
| Max_measure | Float |
| Len | Float |
| Points | <Blob Datatype> |

**Fid :** Geometrinin Id değeri

**Numofpts :** Geometrideki koordinat değerlerinin sayısı

**Entity :** Geometrinin gösterim şekli

**Eminx :** Minimum X koordinat değeri .

**Eminy :** Minimum Y koordinat değeri

**Emaxx :** Maksimum X koordinat değeri

**Emaxy :** Maximum Y koordinat değeri

**Eminz :** Minimum Z koordinat değeri

**Emaxz :** Maksimum Z koordinat değeri

**Min_measure :** Minimum M koordinat değeri

**Max_measure :** Maksimum M koordinat değeri

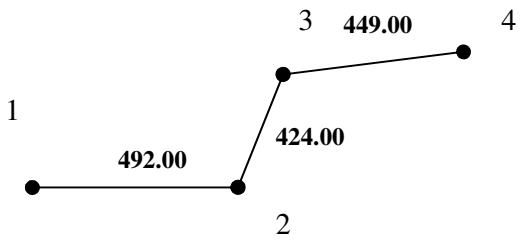
**Area :** Geometrinin hesaplanmış alanı

**Len :** Geometrinin hesaplanmış çevre uzunluğu

**Points :** Geometrinin koordinat değerleri

Tablo 3.5' de M olarak nitelendirilen değerler çizgi tipindeki objelerin uzunluk değerlerini olarak ele alınmaktadır (Şekil 3.7). M değeri her bir noktanın oluşan çizgi boyunca başlangıç noktasına göre olan uzaklığını göstermektedir.

| No | X | Y | M |
|---|---|---|---|
| 1 | 315.95 | 649.72 | 0.00 |
| 2 | 1923.08 | 649.72 | 492.00 |
| 3 | 1999.61 | 2027.26 | 916.00 |
| 4 | 3453.68 | 2103.80 | 1365.00 |




**Şekil 3.7 :** M değerlerinin gösterimi

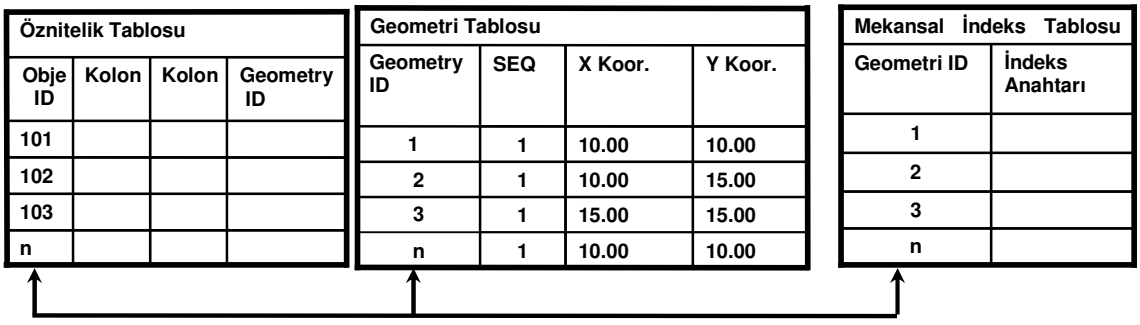
### 3.3.2. Normalize Edilmiş Geometri Yöntemi

Normalize edilmiş geometri şemasında objenin koordinat değerleri bağımsız koordinat değerleri şeklinde depolanır. Geometrik koordinatlar veri tabanının ayrı bir geometri tablosunda nümerik veri tipleri olarak depolanır. Öznitelik tablosundan (Business table) Geometriye ulaşım Gometry Id veya GID değerleri ile ilişki kurularak sağlanmaktadır.

Normalize edilmiş şemalarda geometrik objenin tüm koordinatları geometri tablosunda tek bir satırda depolanmaz. Çok sayıda koordinat değerleri içeren objeler topluluğu için geometri düzgün bir sırada olsun diye bir sıralama değeri (Seq) ile çoklu satırlarda halinde tutulur. Ek olarak geometrik objenin çoklu parçaları için tanımı için bir Eseq değeri kullanılır. Geometri tipi (nokta, çizgi, poligon) Etype değeri olarak kaydedilir.

Böylece geometrinin normalize olarak depolanması durumunda tek bir obje çoklu satırlar halinde kaydedilmekte ve bu çoklu satırlar koordinat değerlerini de içermektedir.

Şekil 3.8 normalize geometrideki geometri tablosu ile öznitelik tablo arasındaki ilişkiyi göstermektedir. Bu örnekte sadece bir koordinat çifti geometri tablosundaki bir satırda depolanmaktadır. Normalize geometri tablosunda yer alan kolon adı ve bu kolonlarda depolanan veri tipleri Tablo 3.7'de gösterilmiştir.




**Şekil 3.8 :** Normalize geometri yöntemi

Mekansal veriler normalize yöntemle depolandıklarında aşağıdaki kurallar uygulanmaktadır.

- Geometrik objeler birden fazla parçaya sahip olabilir. Eseq değeri bu parçaları tanımlamak için kullanılır.
- Bir obje birden fazla obje (satırdan) oluşabilir.Satırlar ve bunların sıralamaları bir sırlama değeri ile tanımlanır.
- Objenin koordinat değerlerinin tamamı bir satırda kaydedilebilir.
- Poligonlar içerisinde boşluklar bulunabilir, boşluk geometrideki yeni bir parçadır. bulunan boşluklar .
- Kullanılmayan koordinat çiftlerinin değerleri (hem X hem Y değeri ) Null (değersiz) olarak sabitlenmelidir. Bu koordinat listesinin sonunun bilinmesi açısından tek yöntemdir.
- Bir sonraki satırda devam eden geometrik objenin satırdaki son noktası ikinci satırdaki ilk noktası olmalıdır.
- Geometrideki parça sayısı için veya satır sayısı için herhangi bir limit yoktur.

**Tablo 3.6 :** Normalize geometri tablosu

| **Kolon Adı** | **Veri Tipi** |
|---|---|
| Gid | Number |
| Eseq | Number |
| Seq | Number |
| X1 | Number |
| Y1 | Number |
| X<Max_Ppr> | Number |
| Y<Max_Ppr> | Number |

**Gid :** Geometrinin ID değeri

**Eseq :** Geometri içindeki çoklu parçaların tanımlar

**Etype :** Geometrinin tipini tanımlar

1 = Nokta

2 = Çizgi

3 = Kapalı Alan

**Seq :** Geometrik parçaların sıra numarasını tanımlar

**X1 :** İlk noktasının koordinat değeri, **Y1 :** İlk noktanın koordinat değeri

**X<Max_Ppr> :** Satırdaki son noktanın ilk koordinatı

**Y<Max_Ppr> :** Satırdaki son noktanın ikinci koordinatı

### 3.3.3. SQL Geometri Yöntemi

SQL geometri yöntemi ile depolanmış mekânsal verilerde verinin koordinatları sadece SQL komutları ile erişilebilen SQL objeleri tarafından yönetilmektedir. SQL geometri yöntemi ile veri depolayan veri tabanlarında objenin koordinat değerleri tek satırda depolanmaktadır.

Tablo 3.7'de tabloda geometrinin nasıl tutulduğu örnek olarak gösterilmektedir. Bu tabloda indeksleme yapılmamıştır çünkü indeksleme veri tabanı otomatik olarak yapılmaktadır.

**Tablo 3.7 :** SQL geometri tablosu

| **Business Table** | | | | |
|---|---|---|---|---|
| Feature-ID | Column 1 | Column 2 | Column N | Geometry_Type |
| 101 | | | | (x,y,&,x,y) |
| 102 | | | | (x,y,&,x,y) |
| 103 | | | | (x,y,&,x,y) |
| & | | | | (x,y,&,x,y) |

## 3.4. Mevcut Veri Tabanları ve Mekansal Veri Depolama Yöntemleri

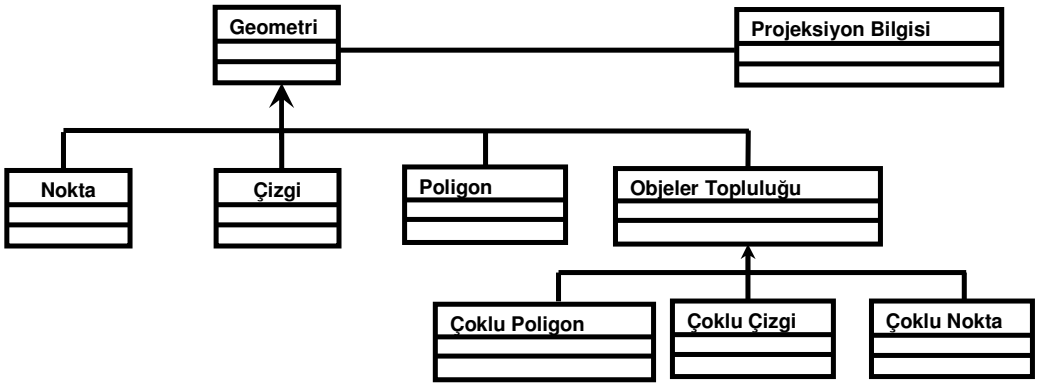
Günümüzde veri tabanı yönetim sistemleri olarak IBM DB2, Informix, Microsoft SQL Server ve Oracle yazılımları kullanılmaktadır. Bu yazılımların mekansal verileri depolayabilmek için izledikleri yöntemler Tablo 3.8'de verilmektedir.

**Tablo 3.8 :** Günümüz VTYS'leri ve mekansal veri depolama yöntemleri

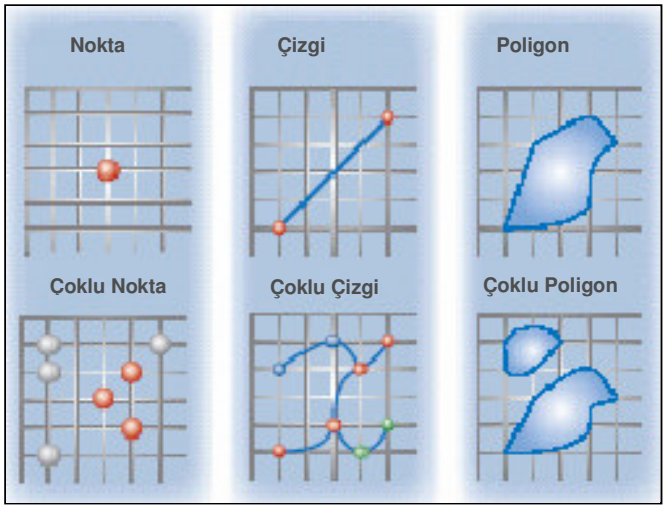
| VTYS | Geometri Depolama |
|---|---|
| IBM DB2 | SQL Geometri Yöntemi |
| Informix | SQL Geometri Yöntemi |
| Microsoft SQL Server | Binary Yöntemi |
| Oracle | SQL Geometri Tipi |
| | Normalize Geometri Yöntemi |

### 3.5. Geometri Tipleri

SQL dili kullanılarak ulaşılabilen veri tabanlarında bir geometri objesinin alt tipleri tanımlanmıştır (OGIS, 1999) (Şekil 3.9) (Şekil 3.10).




**Şekil 3.9 :** Geometrik veri tipleri ve birbirleri ile ilişkileri




**Şekil 3.10 :** Geometri veri tipleri

Şekil 3.9 'da ifade edilen çoklu nokta, çizgi ve poligon ifadeleri birden fazla mekansal objenin öznitelik tablosunda bir adet satır bilgisi kaydedilmesi anlamına gelmektedir. Birden fazla obje için bir adet öznitelik bilgisi olması şeklinde de ifade edilebilir. Bu objelerden bir tanesi seçildiğinde çoklu grup içerisinde yer alan diğer objelerde seçilmiş olur.

### 3.6. SQL Geometri Fonksiyonları

Yapısal Sorgulama Dili ( Structured Query Language- SQL) veri tabanı sorgulamaları ve uygulamaları için endüstriyel bir standarttır. Mekansal veri sunucuları veri tabanındaki bilgilere SQL ile ulaşmaktadır. SQL Fonksiyonları veri tipleri ile çalışmaktadır. Örneğin bir geometri tipinin alan veya uzunluk değerini geri döndüren fonksiyonlara sahip olabilirler. Geometrik tipler üzerinde işlem yapan fonksiyonların başka işlevleri de olabilmektedir. Örneğin iki geometrinin kesişim veya birleşimini hesaplayarak yeni bir geometrik obje gibi geriye döndürebilir. Örneğin belli bir poligon ile (mahalle1 ) kesişen tüm parselleri getir:

```
SELECT Parcel.no, Parcel.kodu
FROM Parsel
WHERE Intersects(Parcel.Geometry, Mahalle1)
```

Mekansal fonksiyonlar SELECT işleminde olduğu gibi SQL dili ile tamamen entegre edilmişlerdir. INSERT ifadesinin kullanıldığı bir SQL cümlesi ise aşağıdaki gibi olmaktadır:

```
INSERT INTO Sehirler (isim, Geometry)
VALUES ( Yalova , PolygonFromText(POLYGON ((x y, x y, x y, ..., x y)), 14))
```

# 4. VERİLERİN ÇOK KULLANICILI ORTAMDA DEPOLANMASI (VERSİYON UYGULAMASI)

Verilerin çok kullanıcılı ortamda tutulması sistem içerisindeki tüm kullanıcıların güncel veriye ulaşmalarına ve üzerinde çalışma yapmalarına imkan sağlamaktadır. Bu işlem veri tabanı kopyalarının yönetimi ile mümkün olmaktadır. Bu kopyalar veri tabanının güncellenmesi sonucu oluşan yeni durumları olmaktadır (Easterfield ve diğerleri,1990). Bu özellik pek çok CBS projesi için vazgeçilmez niteliklerinden biri haline gelmiştir. Çok kullanıcılı ortam için veriler, versiyon denilen uygulamalar ile sağlanmaktadır.

## 4.1. Versiyon

Veri tabanlarında versiyon; objelerin farklı durumlarının depolanması işlemidir (Medeiros ve Jomier 1994). Nesneye yönelimli veri tabanlarında "Obje Versiyonu" ve "Veri tabanı Versiyonu" olmak üzere iki farklı versiyon yaklaşımı bulunmaktadır. Obje versiyon modelinde versiyonlar objeler için tanımlanmaktadır ve obje versiyonları zamanın bir fonksiyonu olarak sıralandırılmaktadır (Kim ve Lochovsky 1992).

Veri tabanı versiyonunda ise veri tabanının tümü için bir versiyon oluşturulmaktadır. Oluşturulan her bir versiyona türetildiği üst versiyonun mantıksal kopyası aktarılmaktadır. Böylece veri tabanı versiyonu bir ağaç şeklinde tanımlanmaktadır. Bu versiyon yaklaşımında sadece değişiklik görmüş objeler kaydedilmektedir. Veri tabanı versiyon yaklaşımında objeler veri tabanında izole edilmez. Bu değişikliklerin tamamının izlenmesi ile veri tabanındaki bütünlük sağlanmaktadır (Cellary ve Jomier 1992).

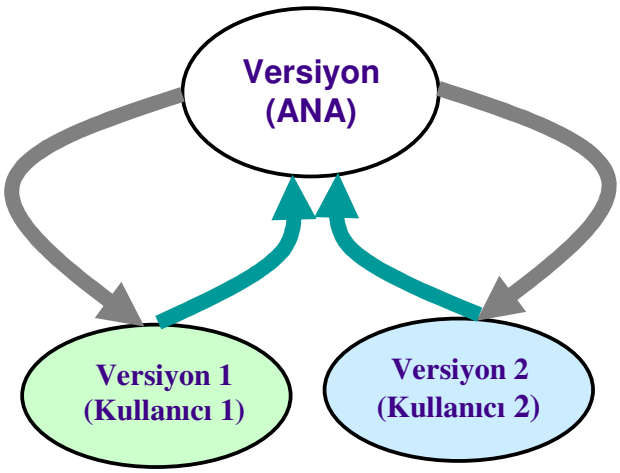
Veri tabanı versiyonunda verilerin değiştirilme, güncellenme ve silinme esnasında oluşan yeni durumları kaydedilmektedir. Verilerin görmüş oldukları değişimler Delta Tabloları olarak adlandırılan tablolara kaydedilmektedir (Esri, 2004). Delta tabloları

verilerin eş zamanlı olarak kullanıcılar tarafından işlenmesine yardımcı oldukları gibi verilerin özgeçmişlerinin de takip edilmesine imkan sağlamaktadır. Güncelleme ve özgeçmiş izleme işlemlerinin gerçekleştirilmesi esnasında mekansal veri sunucuları önemli rol oynamaktadır.

Sistem, içerisinde eş zamanlı güncelleme yapacak kullanıcıları birer versiyon gibi ele almaktadır. Böylece her bir kullanıcı için bir versiyon tanımlaması yapılır ve bu versiyonlar sadece tanımlı oldukları kullanıcılar tarafından belli yetkiler çerçevesinde güncellenir.

Verilerin en son hallerinin bulunduğu versiyon ana versiyon olarak tanımlanır ve diğer versiyonlar bu ana versiyondan türetilir. Ana versiyondan yeni versiyonlar türetildiğinde mekansal veri sunucuları bu ana versiyonun mantıksal kopyalarını tanımlanan yeni versiyonlara gönderir. Yapılan kopyalama işlemi verinin birden fazla yerde tutulması anlamına gelmemektedir.

Ana versiyon hariç oluşturulan her bir versiyonun bir üst versiyonu bulunmaktadır. Bu versiyondan alt versiyonlar üretmek mümkün olmaktadır (Şekil 4.1).

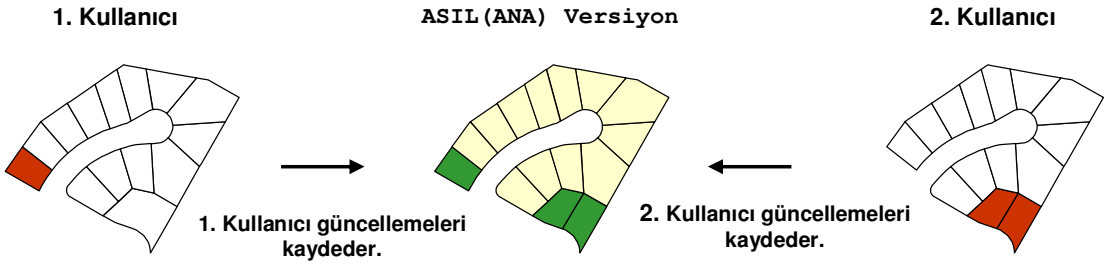



**Şekil 4.1 :** Versiyon

Örneğin; veri tabanınında Versiyon 1 ve Versiyon 2 diye adlandırılan iki ayrı versiyon olsun. Versiyon 1' de yapmış olduğu değişiklikleri ana versiyona gönderdiğinde artık Ana versiyon verinin yeni halini yansıtacaktır. Aynı şekilde

Versiyon 2 de yapmış olduğu değişiklikleri Ana versiyona gönderdiğinde yine Ana versiyon yapılan bu değişiklikleri yansıtacak biçimde değişecektir (Şekil 4.2).

Her bir versiyon sadece yetkili bir kullanıcı tarafından veri tabanından kaldırılabilir. Eğer bir versiyon üst bir versiyondan üretilmiş ise bu versiyonu silmek için öncelikle kendisinden türetilmiş versiyonların da silinmesi gerekmektedir. Versiyonlar korumalı ve korumasız olabilirler. Korumalı versiyonlar sadece versiyonu oluşturan kullanıcıya açık ve diğer kullanıcılara sınırlı olarak açık olmakta ve korumasız versiyonlar ise tüm kullanıcılar tarafından güncellenebilmektedir.




**Şekil 4.2 :** Versiyon uygulaması

### 4.1.1. Versiyon Uygulamasının Prensipleri

Gelişmiş bir VTYS büyük hacimli verilerin üretim, güncellemesi işlemleri için kullanıcılara destek sağlamaktadır. VTYS kullanıcılarına, aynı anda verilerin üzerinde çalışma ihtiyacı nedeniyle mekansal verilerin kopyaları oluşturulmadan çok kullanıcı ortamda çalışabilmeleri için gerekli araçları sağlamaktadır. Bu özellik sağlanırken veri tabanına yapılan değişikliklerin geri alınması veya yeniden yapılması, yayınlanan veri tabanını etkilemeden alternatif uygulama tasarımlarının ve veri modellerinin geliştirilmesi ve test edilmesi gibi işlemleri mümkün kılmalıdır. Veri tabanının zaman içerisinde obje veya katmanların nasıl değiştiklerinin izlenmesine imkan sağlamalıdır. Bu amaç için işbitimleri kavramı kullanılmaktadır.

Veri tabanlarının temel fikirlerinden biri işbitim (transaction) süreleridir. İşbitim operasyonel bir görevin tamamlanması için yapılan işlemler anlamına gelmektedir. İşbitimler verilere eş zamanlı erişimin sağlanması ve bu esnada verilerin yönetimi için kullanılmaktadır (Gray ve Reutet, 1993). İşbitim şeması " begin (başla)" ,

"commit (onayla)" ve "abort (çık)" olarak adlandırılan üç standart işlemden oluşmaktadır. İşbitim "begin" işlemi ile başlar ve "commit" ifadesi ile bitirilirse veri üzerinde yapılan işlemler artık kalıcı olur. İşbitimin herhangi bir zamanında "abort" işlemi ile çıkılırsa verinin işbitime başlamadan önceki hali geri depolanır (Ramakrishnan ve Gehrke, 2000).

Veri tabanı transaction işlemleri ve çok kullanıcı kontrol mekanizmalarının büyük bölümü ilişkisel veri tabanı modeli için geliştirilmiştir. (Bernstein ve diğerleri, 1987). Sonraları ise ilişkisel veri tabanı için geliştirilmiş olan transaction modelinin kapsamı nesneye yönelim veri tabanları içinde genişletilmiştir (Herlihy, 1990).

Veri tabanı fonksiyonelliğinde anahtar kelime işbitim kavramı olmaktadır. İşbitim (transaction) veri üzerinde gerçekleşen operasyonlarda bir operasyonun bitimini belirten işlemlerin mantıksal bir grubu olmaktadır.Veri tabanı iş bitimleri kısa veya uzun olabilmektedir (Gray ve Reutet, 1993).

#### 4.1.1.1. Kısa İşbitimler ( Short Transactions)

Kısa iş bitimleri saniyeler mertebesinde bitirilen güncelleme işleri olmaktadır. Örnek olarak bankacılık uygulamalarındaki iş bitimleri verilebilir. İşbitim esnasında güncellenecek olan kayıt kilitlenir. Bu kilitleme mekanizması farklı kullanıcılar tarafından işlemlerin doğru sırada yapılmasını sağlayarak verinin doğruluğunu garanti etmektedir. Kilitleme mekanizması veri tabanı içinde bir kullanıcı tarafından yönetilmektedir. Kısa iş bitim sonlanmasıyla kilit mekanizması ortadan kalkar.

Kısa iş bitim mekanizması kritik işlemler için gayet iyi çalışıyor olmasına rağmen mekansal verilerin güncelleme işlemleri için uygun olmamaktadır. Böylesi durumlarda mekansal veriler güncellenirken güncelleme işlemi birkaç dakikadan fazla süreceğinden kısa iş bitim kayıt kilitleme mekanizması diğer kullanıcıların bu verilere ulaşmasını engelleyecektir. Mekansal verilerin ilişkileri ve bağlantıları daha esnek bir güncelleme yaklaşımı gerektirmektedir.

Örneğin; altyapı verileri ile çalışan iki kullanıcı olsun. Kullanıcılardan biri elektrik kabloları ile güncelleme yaparken bir diğer ise elektrik direkleri üzerinde güncelleme yapacaktır. Birinci kullanıcının yapacağı değişiklik direklerinin konumunu

değiştireceğinden ikinci kullanıcı ilk kullanıcının yapacağı değişikliklerin bitmesini bekleyecektir.

Birbirleriyle ilişkili katmanlarda yapılan çalışmalarda oluşturulan kilit mekanizmaları ortadan kalkmadan ikinci iş bitimleri olamayacağından kısa işbitimleri mekansal verilerin güncellenmesi için faydalı olmamaktadır (Zeiler , 2002).

#### 4.1.1.2. Uzun İşbitimleri (Long Transaction)

Veri tabanı üzerinde kısa işbitim (short transaction) uygulamalarında veriye ulaşım işlem bitene kadar yasaklandığından uzun süren işlemler için uzun işbitim (long transaction) uygulaması kullanılmaktadır. Böylelikle kullanıcıların eş zamanlı olarak aynı verileri güncellemelerine imkan sağlanmaktadır. Bir başka deyişle kullanıcılar veriyi güncelleyebilmek için kendilerinden önce başlanan güncellemelerin bitmesini beklememektedirler.

Her bir veri tabanı iş bitimi için bir değişim olayı oluşturulmaktadır. Bu iş bitimleri veri tabanındaki değişiklikleri etkileyen bir projenin, bir iş akışının bitirilmesi işlemleri olabilir. Bir parsel katmanında yeni bir parsel oluşturulduğunda, değiştirildiğinde ve silindiğinde bir özgeçmiş versiyonu oluşturulur.

Coğrafi verilerin değişiklikleri birden fazla iş bitimlerini içerebilmektedir. Bir parsel örneği ele alınır ise;

1. Parsel objesini iki parçaya ayır
2. Ayırma çizgisi boyunca iki ayrı sınır çizgisi çiz
3. Oluşan iki parsele birer yeni numara ver
4. Her bir parselin hat açılış kapanışlarının yeniden oluştur

Yukarıdaki işlemler yapılırken veri tabanında dört ya da daha fazla iş bitimleri oluşturulur. Bu işlemlerden her birini sayısal güncelleme birimi şeklinde ifade edilmektedir. Veri tabanında tarihsel bir hareket oluşursa bu hareketin oluşumunun kaydı bir birim olarak ele alınır.

### 4.1.2. Versiyon Türleri

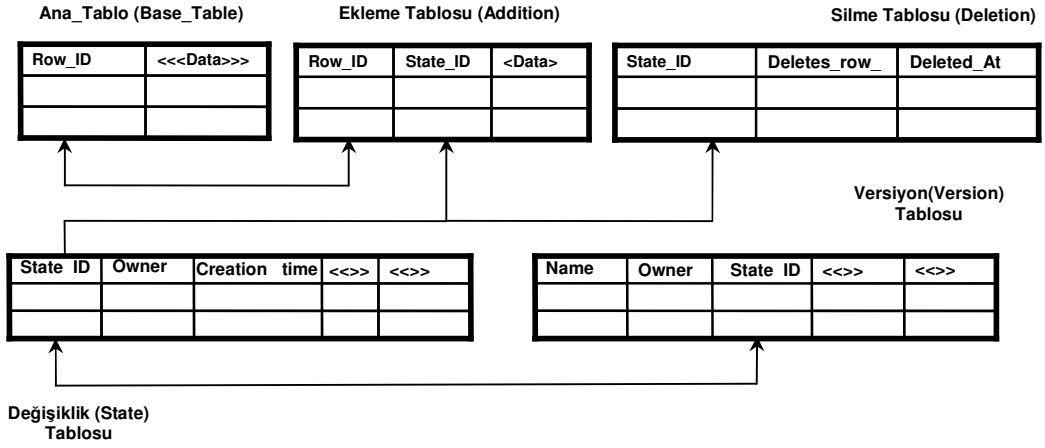
Veri tabanları üzerinde oluşturulan versiyonlar amaçlarına göre 3 gruba ayrılmaktadır.

- **Özel Versiyon ( Private Version ) :** Bu tip versiyonlarda sadece versiyonu oluşturan kişiler değişiklik yapabilir.
- **Genel Versiyon (Public Version ) :** Versiyonu verinin sahibi tarafından görme değiştirme hakkı verilen kullanıcıların erişip hakları doğrultusunda kullanmalarına imkan vardır.
- **Sınırlı Versiyon (Protected Version) :** Verileri sistemdeki tüm kullanıcılar görebilir ancak veriler üzerinde herhangi bir değişiklik yapamazlar.

Yukarıdaki özelliklere sahip olan versiyonlar mekansal veri sunucu yazılımları tarafından verilerin güncel halinin tutulduğu ana versiyonlardan üretilmektedir.

## 4.2. Versiyon Sistem Tabloları

Versiyonlar veri tabanlarında tablolar halinde mekansal veri sunucu yazılımları tarafından yönetilmektedir (Şekil 4.3).




**Şekil 4.3 :** Versiyon sistem tabloları ve birbirleri ile ilişkileri

Bu tablolara ek olarak değişikliklerin soyağacının izlendiği Değişiklik_Soyağacı (State_Lineage) tablosu da bulunmaktadır. Bu tablo opsiyonel olup State tablosundaki verilere hızlı ulaşmak için Değişiklik (State) tablosundan türetilmektedir.

Versiyon ve Değişiklik tabloları versiyon uygulamalarının sistem tabloları olarak ele alınmaktadır. Tablo 4.1'de versiyon sistem tabloları ve açıklamaları verilmiştir.

**Tablo 4.1 :** Versiyon tabloları

| Sistem Tablosu | Açıklama |
|---|---|
| Versiyonlar (Versions) | Her bir versiyon, adı, sahibi ve ilişkilendirildiği veri tabanı durumları (state) ile tanımlıdır. Bu tablo kullanıcılara sunulan mevcut versiyonları göstermektedir. |
| Değişiklikler (States) | Bu tablo veri tabanında yapılan değişiklikleri yönetmektedir. Her bir değişiklik için oluşturulmaya başlama zamanı, bitiş zamanı, önceki değişiklik Id değeri ve değişik sahibi gibi bilgileri içermektedir. Değişiklik soyağacı tablosu ile ilişkilendirilerek verilerin özgeçmişleri oluşturulmaktadır. |
| Değişiklik_Soyagacı (State_Lineages) | Bu tablo değişikliklerin özgeçmişlerini tutmaktadır. |

### 4.2.1. Versiyonlar (Versions) Tablosu

Versiyon tabloları veri tabanın sahip olduğu versiyon bilgilerini tanımlamaktadır. Bu versiyonlar uygulamalar tarafından veri tabanının spesifik veri tabanı durumlarına erişmek için kullanılırlar. Versiyonların adları ve sahip oldukları kimlik numaraları farklıdır. Yeni bir versiyon oluşturulduğunda Ana veri bu versiyona aktarılmaktadır. Birinci State_Id değeri 0 olarak değiştirilmektedir. Versiyon tablosunun içerdiği bilgiler Tablo 4.2'de verilmektedir.

**Tablo 4.2 :** Versiyon tablosu

| Kolon Adı | Açıklama |
|---|---|
| Name | Versiyonun adı |
| Owner | Versiyonun sahibi |
| Version_ID | Versiyonun kimlik numarası |
| Status | Versiyonun ortak kullanıma açık yada kapalı olduğu bilgisi |
| State_ID | Versiyonun gösterdiği veri tabanı durumunun ID numarası |
| Description | Versiyonun opsiyonel tanımlama bilgisi |
| Parent_Name | Bu versiyonun üretildiği bir üst versiyon |
| Parent_Owner | Bu versiyonun üretildiği bir üst versiyonun |
| Parent_Version_ID | Bu versiyonun üretildiği bir üst versiyonun ID kimlik numarası |
| Creation_Time | Versiyonun üretildiği tarih bilgisi |

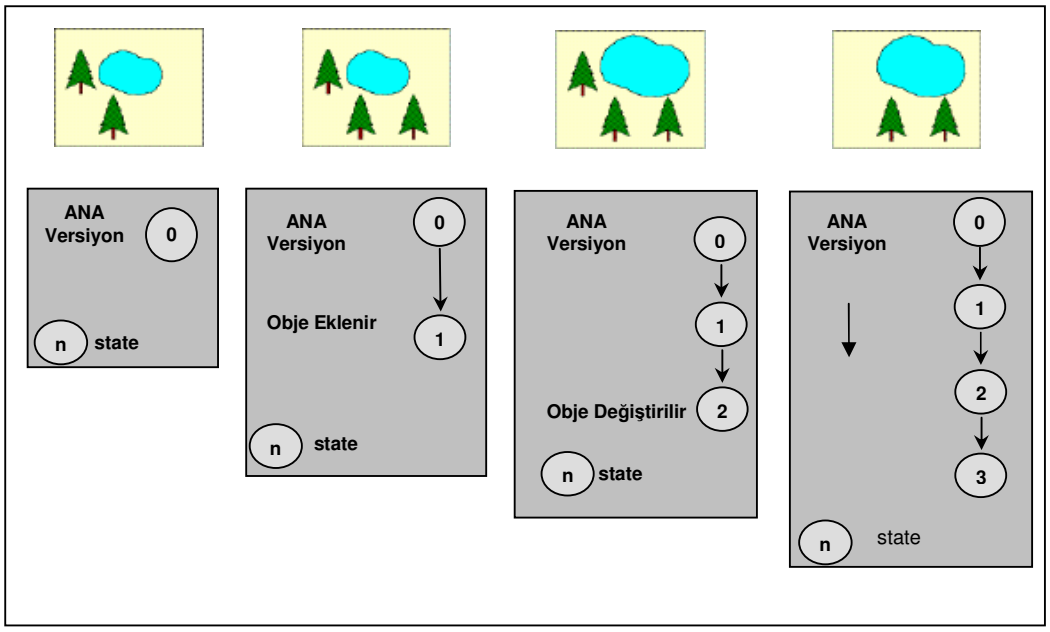
### 4.2.2. Değişiklikler (States) Tablosu

Veri üzerinde yapılan değişikliklerin izlenebilmesi için mekansal veri sunucuları bu değişikliklerin depolandığı State tablolarını kullanmaktadır. State tablosu zaman içerisinde oluşturulan verilerin geçirdikleri tüm değişiklik bilgilerini kaydetmektedir. Kullanıcı silme, değiştirme ve ekleme gibi bir değişiklik yaptığında bu işlemlerin tamamı state tablosuna kaydedilir. Bu kayıt işlemi değişiklik için tekrarlanmayan bir Id değeri ile yapılmaktadır. State tablosunun içerdiği bilgiler Tablo 4.3'de verilmektedir.

**Tablo 4.3 :** Değişimler (States) tablosu

| Kolon Adı | Açıklama |
|---|---|
| State_ID | Yazılım tarafından yapılan her bir güncelleme için verilen bir ID değeri |
| Owner | Bu güncellemeyi yapan kullanıcı |
| Creation_Time | Güncellemenin yapıldığı tarih ve saat bilgisi |
| Closing_Time | Güncellemenin kapandığı tarih ve saat bilgisi |
| Parent_State_ID | Yapılan güncellemenin üst güncelleme ID değeri |
| Lineage_Name | States_Lineages tablosunda depolanan güncellemenin soyağacı adı |

Değişiklik esnasında Closing_Time kolonu herhangi bir değer içermez bu değişiklikler saklandığında Closing_Time kolonu değişikliğin kaydedildiği zamanı kaydetmektedir. Parent_State_ID değeri ise yapılan her bir değişikliklerden önce hangi değişimlerin yapıldığını göstermektedir. Böylece değişimlerin sırasını takip etmek mümkün olmaktadır (Şekil 4.4).




**Şekil 4.4 :** Değişikliklerin izlenmesi

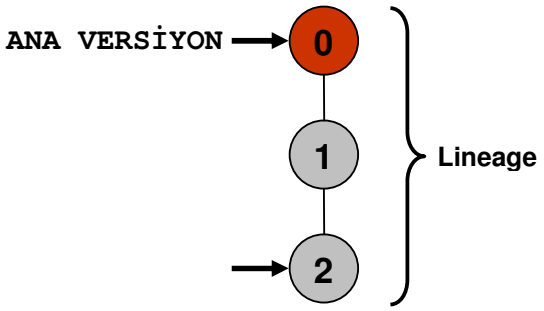
### 4.2.3. Değişiklik_ Soyağacı (State_Lineages) Tablosu

State_Lineages tablosu değişimlerin bir nevi soyağaçlarını depolamaktadır. Böylelikle değişikliklerin sıraları ayrı bir tabloda tutulmuş olur. Bu tablo sorgulamalarda hızlı sonuç almak için State tablosundan üretilmektedir. State_Lineages tablosunun içerdiği bilgiler Tablo 4.4'de verilmektedir.

**Tablo 4.4 :** Değişiklik_Soyağacı (State_Lineage) tablosu

| Kolon Adı | Açıklama |
|---|---|
| Lineage_Name | Değişikliğin yapıldığı State değeri |
| Lineage_ID | Kayıt Numarası |

State tablosunda olduğu gibi değişimlerin sırası state_lineage tablosunda da tutulabilmektedir (Şekil 4.5). Değişim izlemek için bu tablolarda yapılan sorgulamalar sonucu bilgiler alınıp State tablosu ile ilişkilendirilir.




**Şekil 4.5 :** State_Lineage tablosunda değişikliklerin izlenmesi

### 4.2.4. Mvtables_Modified (Değişiklik İzleme) Tablosu

MVTABLES_MODIFIED tablosu değişiklikler esnasında etkilenen tabloların kayıtlarını tutmaktadır. Bu tabloda yer alan bilgiler versiyonlarda yapılan değişikliklerin uyuşumsuzluklarının kontrol edilmesi için kullanılmaktadır. Bu tabloda yer alan bilgiler Tablo 4.5'de verilmektedir.

**Tablo 4.5 :** MVTables_Modified tablosu

| Kolon Adı | Açıklama |
|---|---|
| State_ID | Tablonun değiştirildiği State değeri |
| Registration_ID | Değiştirilen tablonun kayıt değeri |

## 4.3. Versiyon Tabloları

Veri tabanında versiyonlardaki verilerin değişimleri versiyon tabloları – delta tablolar olarak adlandırılan tablolarda yönetilir. Bu tablolar veri tabanındaki silinen ve eklenen kayıtları tutarlar. Mekansal veri sunucuları verilerin depolandığı tabloları orijinal hallerinde bırakarak sadece yapılan değişiklikleri delta tablosu olarak adlandırılan Ekleme (Addition) ve Silme (Deletion) tablolarında depolar.

Versiyon uygulaması içeren her bir katman için mekansal veri sunucusu tarafından ekleme ve silme tabloları oluşturulur. Versiyon tabloları ve kullanım amaçları Tablo 4.6'de gösterilmiştir.

**Tablo 4.6 :** Versiyon tabloları

| Veri Tablosu | Amaç |
|---|---|
| <Versioned View Name> | Herhangi bir zaman diliminde versiyonun durumunu gösteren ve ekleme ve silme tablolarından türetilen tablo |
| <Versioned Base Table Name> | Versiyon ana tablosu |
| A<Registration_Id> | Ana tabloya eklenen ve değitirilen objelerin kayıtlarının tutulduğu tablo. |
| D<Registration_Id> | Ana tablodan silinen objelerin bilgilerinin tutulduğu tablo. |

### 4.3.1. Versiyon Sanal (Version View) Tablosu

Version View tablosu opsiyonel bir tablo olup veri tabanında yapılacak sorgulamalar ile versiyonun belli bir zaman dilimindeki halini göstermek için kullanılır.

### 4.3.2. Versiyon Katman (Version Base) Tablosu

Version Base tablosu ilişkisel veri tabanında çok kullanıcılı ortama konu olacak tablodur. Bu tablo üzerinde mekansal veri sunucuları tarafından kontrol edilen bir kolon bulunmaktadır.

### 4.3.3. Ekleme (Additions) Tablosu

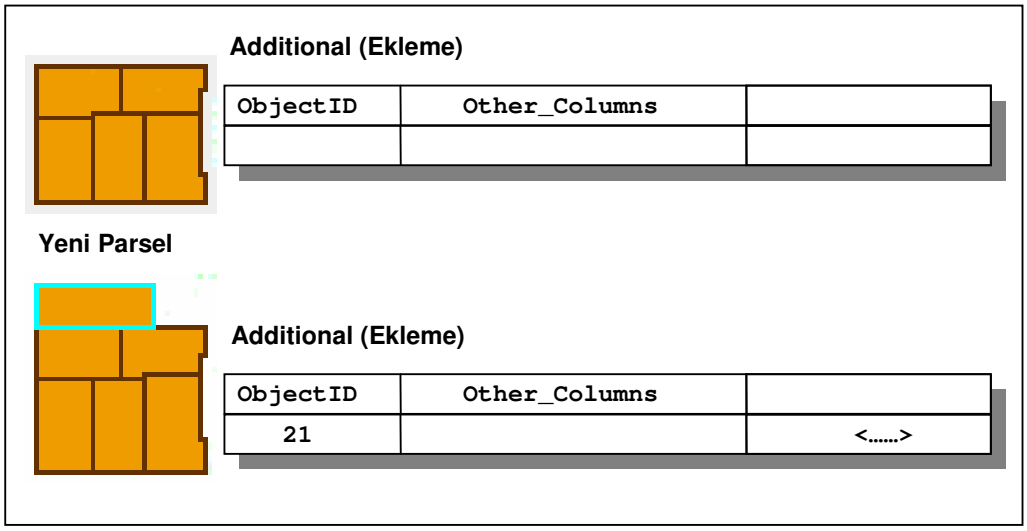
Ekleme (Additions) tablosu katman tablosu üzerinde yapılan eklemeleri ve değişiklikleri depolamaktadır (Şekil 4.6). Bu tabloya her bir kayıt eklendikçe bu değişimin sahip olduğu State_ID değeri de eklenmekte ve böylece değişikliğin hangi kullanıcı tarafından yapıldığı bilgisi tutulmaktadır. Bu bir anlamda ana versiyonda tutulan kayıtların bir anlamda bir kopyasının da bu tablolarda tutulması anlamına gelmektedir. Ekleme (Additions) tablosunun içeriği Tablo 4.7'de verilmiştir.

**Tablo 4.7 :** Ekleme tablosu

A (n)

| Kolon Adı | Açıklama |
|---|---|
| State_ID | Objenin ait olduğu değişiklik değeri |
| <row_ID> | Satırın tekrarlanmayan kayıt değeri |
| <other columns> | Yeni objenin öz nitelik bilgileri |

n : Kayıt Numarası




**Şekil 4.6 :** Ekleme (Addition) tablosuna veri eklenmesi

### 4.3.4. Silme (Deletions) Tablosu

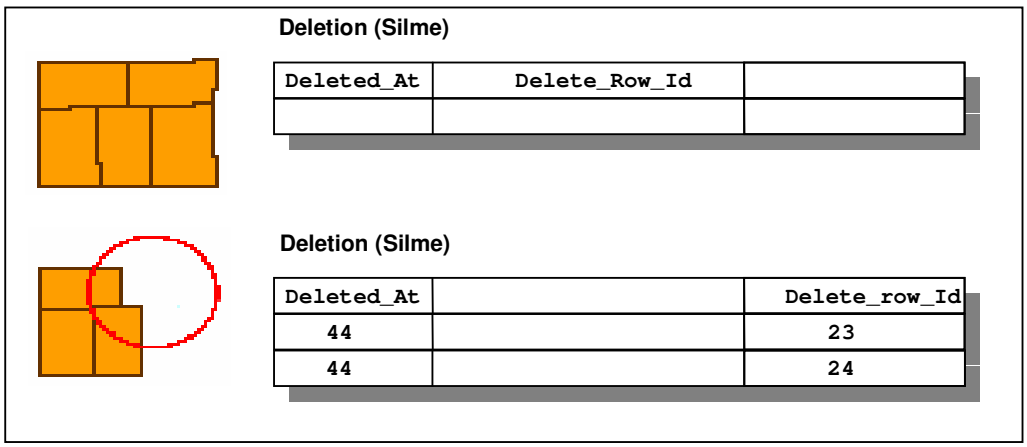
Versiyon uygulamasına konu olan katman üzerinde silinen ve değiştirilen objelerin kaydedildiği tablo Silme (Deletions) Tablosu olarak isimlendirilmektedir. Bu tablolar katmanlardan hangi objelerin silindiğinin belirlenmesi için yapılan sorgulama işlemlerinde kullanılmaktadır (Şekil 4.7). Silme (Deletions) tablosunun içeriği Tablo 4.8'de verilmiştir.

**Tablo 4.8 :** Silme (Deletion) tablosu

D (n)

| Kolon Adı | Açıklama |
|---|---|
| State_ID | Objenin silindiği State ID değeri. |
| Deletes_Row_ID | Silinen veya güncellenen obje için tekrarlanmayan ID değeri |
| Deleted_At | Objenin silindiği versiyondaki State ID değeri. |

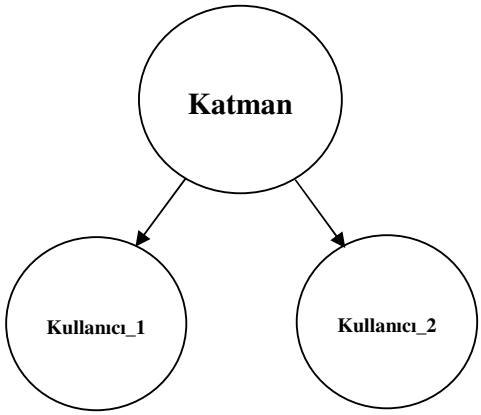
n : Kayıt Numarası




**Şekil 4.7 :** Silme (Deletion) tablosundan veri silinmesi

### 4.4. Versiyon Senaryoları

Aşağıdaki örnek iki kullanıcının bağımsız bir şekilde ortak bir veri tabanını nasıl güncelleştirdiklerini açıklamaktadır. Verilerin son hallerinin bulunduğu bir versiyon oluşturulmakta ve bu versiyona Katman ismi verilmektedir. Kullanıcılar katman olarak adlandırılan versiyondan Kullanıcı_1 ve Kullanıcı_2 alt versiyonların oluşturmaktadır (Şekil 4.8).




**Şekil 4.8 :** Versiyon ağacı

Alt versiyonlardan yapılan değişiklik olayları ve bu değişiklik olaylarının ana ve alt versiyondaki tabloları nasıl etkiledikleri incelenmektedir.

**Değişim 1:**

Kullanıcı_1 katman tablosuna erişerek bu tablonun kullanıcı_1 isimli bir versiyonunda versiyonda 100 ve 101 numaralı iki obje oluşturuyor. Kullanıcı_1 kendisinden versiyon üretmiş bu değişiklikleri katman tablosuna gönderiyor. Katman tablosunda bu veriler ile uyuşumsuzluk yaratan herhangi bir veri olmadığından katman tablosuna artık bu veriler kaydedilmiş oluyor. Dolayısıyla katman tablosu artık kullanıcı_1 versiyonundaki durumu gösteriyor.

**Değişim 2:**

Kullanıcı_2 kullanıcı 1 versiyonundaki değişikliklerin yer aldığı kendi versiyonunda 103 numaralı objeyi eklemekte ve 101 numaralı objede ise değişiklik yapmaktadır.

**Değişim 3 :**

Kullanıcı_2 kendi versiyonunda güncellemeye devam ederken kullanıcı_1 102 numaralı bir objeyi eklemekte ve 100 numaralı obje üzerinde değişiklik yapmaktadır. Kullanıcı_1 bu değişiklikleri katman tablosuna göndermekte ve katman tablosunda herhangi bir uyuşumsuzluk tespit edilmediği için artık Kullanıcı_1 tarafından yapılan değişiklikler gösterilmektedir.

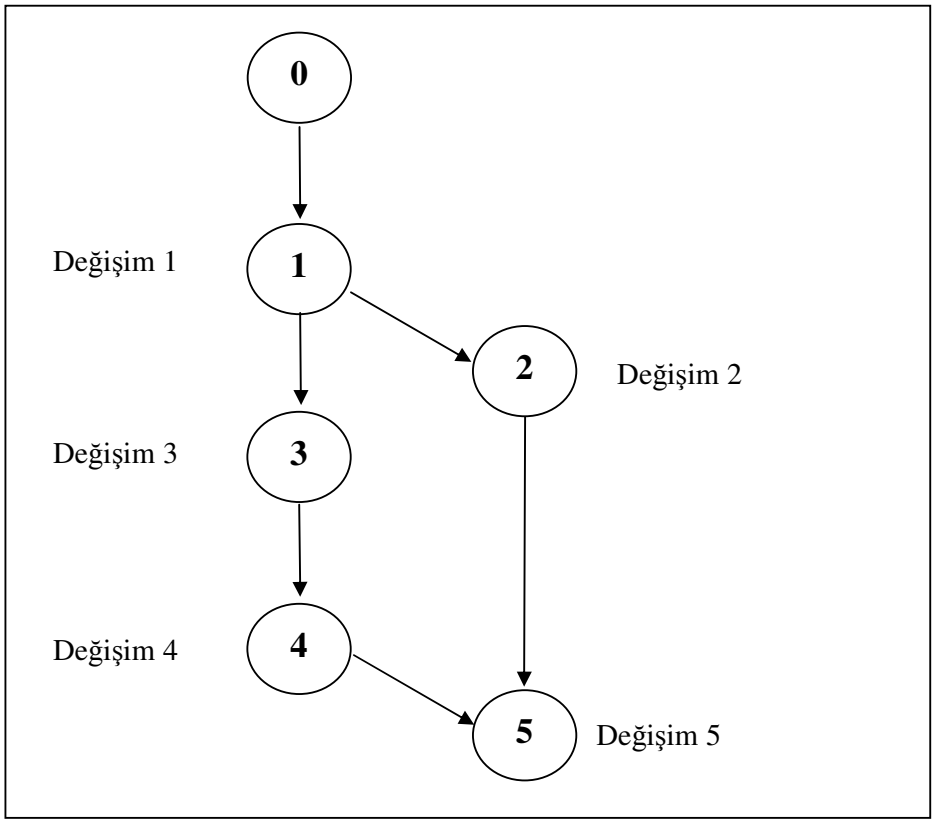
**Değişim 4:**

Kullanıcı_1 101 numaralı objeyi değiştirerek 100 numaralı objeyi silmekte ve bu değişiklikleri katman tablosuna göndermektedir.

**Değişim 5:**

Kullanıcı_2 güncelleme işlemlerini bitirmekte ve yapmış olduğu bu güncellemeleri katman tablosuna gönderdiğinde 100 ve 101 numaralı objelerin değiştirilmiş olduğunu görmektedir. Kullanıcı_2 katman tablosunda yer alan 101 numaralı objeyi kabul etmekte ve 100 numaralı objeyi ise kendi versiyonundaki obje olarak tabloya kaydetmektedir.

Bu ard arda gerçekleşen işlemlerin diyagramı Şekil 4.9' da gösterilmiştir.




**Şekil 4.9 :** Versiyon senaryosu diyagramı

Veriler üzerinde yapılan değişikler State tablolarında depolandıklarından bu değişimlerin State tabloları üzerinde ne gibi değişikler yaptığı incelenmektedir.

**State 0 :**

Veri tabanında Katman tablosu ana versiyon olarak bulunmaktadır.

**State 1 :**

Kullanıcı_1 isimli bir versiyon oluşturmakta ve 100 ve 101 numaralı objeleri eklemektedir. Bu değişiklikleri ise Katman tablosuna (State 0) göndermektedir. Katman tablosu artık Kullanıcı_1 versiyondaki değişiklikleri yansıtacak bir biçimde değişmiş olmaktadır.

**State 2 :**

Kullanıcı_2 state1 aşamasında yer alan tablo katmanından Kullanıcı_2 versiyonunu oluşturarak 103 numaralı bir obje eklemekte ve 101 numaralı obje üzerinde değişiklik yapmaktadır.

**State 3 :**

Kullanıcı_1 state1 aşamasındaki katman tablosunda değişikliğe devam etmekte ve 102 numaralı objeyi ekleyerek 100 numaralı obje üzerinde değişiklik yapmaktadır. Bu değişiklikleri Katman tablosuna göndererek katman tablosunun durumunu State 3 aşamasına getimektedir.

**State 4 :**

Kullanıcı_1 tarafından State 3 aşamasına getirilen katman tablosunda değişikliğe devam etmekte ve 101 numaralı objeyi değiştirerek 100 numaralı objeyi silmekte ve bu değişiklikleri katman tablosuna aktararak tablonun State 4 aşamasına gelmesini sağlamaktadır.

**State 5 :**

Kullanıcı kendi versiyonunda State 2 aşamasında bulunan yapmış olduğu State 4 aşamasında bulunan katman tablosuna göndermek istediğinde State 4 aşamasında

bulunan katman tablosundaki uyuşumsuzlukları fark etmekte ve 101 numaralı objenin State 4 aşamasındaki durumunu kabul ederken 100 numaralı objenin ise kendi yapmış olduğu değişikliği yansıtmasını sağlamaktadır.

Bu değişim işlemleri tablolar halinde gösterilmek istendiğinde ;

**Değişim 1 :**

Değişim 1 olayından sonra Version-State Tablosu Tablo 4.9'de gösterilmiştir. .

**Tablo 4.9 :** Değişim 1 olayından sonra version-state tablosu

| Versiyon | State |
|---|---|
| KATMAN | 1 |

Katman versiyonundaki tablonun son durumu. (100 ve 101 numaralı objeler eklenmiştir.) Tablo 4.10'de gösterilmiştir.

**Tablo 4.10 :** Katman versiyonundaki tablonun son hali

| OID | State | Veri |
|---|---|---|
| 100 | 1 | a |
| 101 | 1 | b |

**Değişim 2 :**

Değişim 2 olayından sonra Version-State tablosu Tablo 4.11 ' de gösterilmiştir.

**Tablo 4.11 :** Değişim 2 olayından sonra version-state tablosu

| Version | State |
|---|---|
| KATMAN | 1 |
| Kullanıcı_2 | 2 |

Kullanıcı_ 2 versiyonundaki katman tablosunun son durumu (103 numaralı obje eklenmiş ve 101 numaralı obje değiştirilmiştir) Tablo 4.12'de gösterilmiştir.

**Tablo 4.12 :** Değişim 2 olayından sonra kullanıcı_2 katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 1 | A |
| 101 | 2 | Bb |
| 103 | 2 | d |

Katman Versiyonun değişikliklerden sonraki durumu Tablo 4.13'de gösterilmiştir.

**Tablo 4.13 :** Değişim 2 olayından sonra katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 1 | a |
| 101 | 1 | b |

**Değişim 3:**

Değişim 3 olayından sonra Versiyon State Tablosu Tablo 4.14' da gösterilmiştir.

**Tablo 4.14 :** Değişim 3 olayından sonra version-state tablosu

| Version | State |
|---|---|
| KATMAN | 3 |
| Kullanıcı_2 | 2 |
| Kullanıcı_1 | 3 |

Değişim 3 olayından sonra sonra Kullanıcı 2 versiyonundaki katman tablosu (herhangi bir değişiklik yok ) Tablo 4.15'de gösterilmiştir.

**Tablo 4.15 :** Değişim 3 olayından sonra kullanıcı_2 katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 1 | a |
| 101 | 2 | bb |
| 103 | 2 | d |

3 nolu değişiklikden sonra Kullanıcı 1 versiyonundaki Katman Tablosu (102 nolu obje eklendi, 100 nolu obje değiştirildi ) Tablo 4.16'de gösterilmiştir.

**Tablo 4.16 :** Değişim 3 olayından sonra kullanıcı_1 katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 3 | aa |
| 101 | 1 | b |
| 102 | 3 | c |

3 Nolu değişiklikden sonra Katman versiyondaki tablonun son durumu (102 nolu obje eklendi, 100 nolu obje değiştirildi) Tablo 4.17'de gösterilmiştir.

**Tablo 4.17 :** Değişim 3 olayından sonra katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 3 | Aa |
| 101 | 1 | b |
| 102 | 3 | c |

**Değişim 4 :**

Değişim 4 olayından sonra Version- State Tablosu Tablo 4.18'da gösterilmiştir.

**Tablo 4.18 :** Değişim 4 olayından sonra version-state tablosu

| Version | State |
|---|---|
| KATMAN | 4 |
| Kullanıcı_2 | 2 |
| Kullanıcı_1 | 4 |

Kullanıcı_2 versiyondaki katman tablosu için son durum (Herhangi bir değişiklik yok ) Tablo 4.19'de gösterilmiştir.

**Tablo 4.19 :** Değişim 4 olayından sona kullanıcı_2 katman tablosu

| OID | State | Data |
|---|---|---|
| 100 | 1 | a |
| 101 | 2 | bb |
| 103 | 2 | d |

Kullanıcı_1 versiyonundaki katman tablosunun son durumu ( 101 nolu obje değiştirilmiş ve 100 nolu obje silinmiştir ) Tablo 4.20'de gösterilmiştir.

**Tablo 4.20 :** Değişim 4 olayından sonra kullanıcı_1 katman tablosu

| OID | State | Data |
|---|---|---|
| 100 | 4 | <delete> |
| 101 | 4 | bbb |
| 102 | 3 | c |

Katman versiyonundaki tablonun son durumu ( 101 nolu obje değişmiş ve 100 nolu obje silinmiştir. ) Tablo 4.21'de gösterilmiştir.

**Tablo 4.21 :** Değişim 4 olayından sonra katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 4 | <delete> |
| 101 | 4 | bbb |
| 102 | 3 | c |

**Değişim 5:**

Değişim 5 olayından sonra Versin-State tablosunun son durumu Tablo 4.22'de gösterilmiştir.

**Tablo 4.22 :** Değişim 5 olayından sonra version-state tablosu

| Version | State |
|---|---|
| KATMAN | 5 |
| Kullanıcı_2 | 5 |
| Kullanıcı_1 | 4 |

Kullanıcı_2 versiyonundaki katman tablosunun son durumu ( Katman versiyonundaki tablo ile Kullanıcı_2 versiyonundaki tablo 101 nolu obje Katman versiyonundan ve 100 nolu obje ise kullanıcı_2 versiyonundan alınarak birleştirilmiştir) Tablo 4.23'de gösterilmiştir.

**Tablo 4.23 :** Değişim 5 olayından sonra kullanıcı_2 katman tablosu

| OID | State | Data |
|---|---|---|
| 100 | 5 | a |
| 101 | 4 | bbb |
| 102 | 3 | c |
| 103 | 2 | d |

Kullanıcı_1 versiyonundaki katman tablosunun son durumu ( Herhangi bir değişim yok) Tablo 4.24'de verilmiştir.

**Tablo 4.24 :** Değişim 5 olayından sonra kullanıcı_1 katman tablosu

| OID | State | Data |
|---|---|---|
| 100 | 4 | <delete> |
| 101 | 4 | bbb |
| 102 | 3 | c |

Katman versiyonundaki tablonun son durumu ( Katman versiyonu ile Kullanıcı_2 versiyonu 101 nolu obje Katman veriyonunda ve 100 nolu obje ise Kullanıcı_2 versiyonundan alınarak birleştirilmiştir. ) Tablo 4.25'de gösterilmiştir.

**Tablo 4.25 :** Değişim 5 olayından sonra katman versiyonu

| OID | State | Data |
|---|---|---|
| 100 | 5 | a |
| 101 | 4 | bbb |
| 102 | 3 | c |
| 103 | 2 | d |

# 5. MEKANSAL VERİLERİN ÖZGEÇMİŞLERİNİN İZLENMESİ

Veri modellerinin gelişmesi ile birlikte artık coğrafi bilgi sistemlerinde mekansal veriler, geometri, öznitelik ve zaman bileşenleri ile ifade edilebilmektedir (Chrisman, 1997). Kullanılan yazılımlardaki geometri ve zaman bileşenlerinin kaydedilmesi özelliğine bağlı olarak zaman bilgisinin de sisteme dahil edilmesi coğrafi bilgi sistemlerinin bir sonraki aşaması olarak görülmektedir. Coğrafi bilgi sistemleri içerisinde zaman bilgisinin de depolanması ile birlikte Zaman Boyutlu Coğrafi Bilgi Sistemleri (TGIS- Temporal Geography Information Systems) kavramı ortaya çıkmış ve verilerin belirli tarihlerdeki durumlarının erişilmesi sağlanmıştır (Şekil 5.1). Zaman bilgisinin sisteme dahil edilmesi aşağıda belirtilen iki yöntem ile mümkün olmuştur (Yuan, 1996);

- Coğrafi bilgi sistemi veri modelinin zaman bileşenini içerecek şekilde genişletilmesi
- Coğrafi bilgi sisteminde kullanılan veri tabanlarının zaman verilerinin depolanmasına ve sorgulanmasına imkan verecek şekilde genişletilmesi

## 5.1. Zamanı İçeren Veri Modelleri

Mekansal verilerin modellenmesi veri tabanı içeriğinin belli bir zaman dilimindeki durumunun ne olduğunun görüntülenmesi ve veri tabanında kaydedilen bilgilerin zaman içerisinde nasıl bir değişim gösterdiklerinin izlenmesine ve analiz edilmesine imkan vermektedir (Yuan, 1996). Coğrafi bilgi sistemlerinde zaman bilgisinin depolandığı dört adet veri modeli bulunmaktadır. Veri modelleri; kurallar, işlemler ve nesne türlerini tanımlamaktadır (Ott and Swiaczny, 2001).

Bunlar;

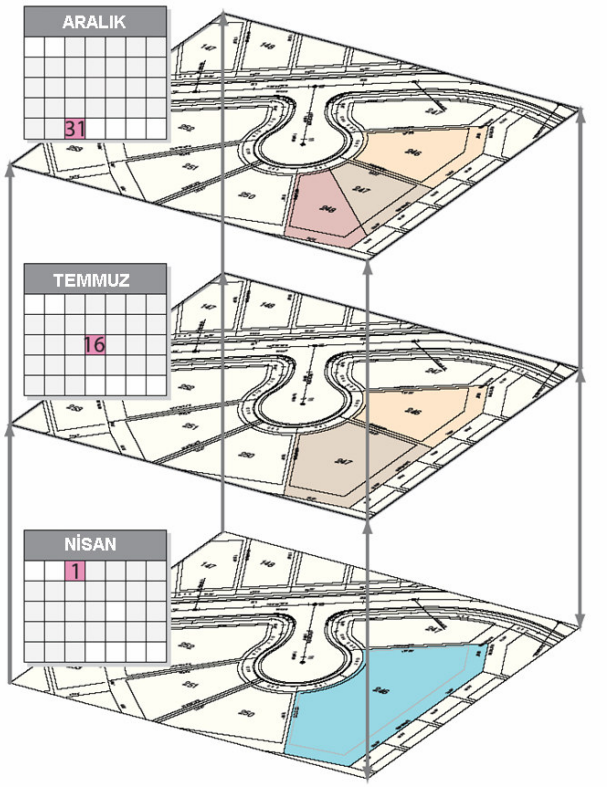
- **Basit anlık durum modeli:** Zaman bileşeninin coğrafi bilgi sistemi içerisinde kaydedildiği en basit veri modelidir (Langran, 1992). Her güncelleme işleminden sonra veri tabanının bir kopyası alınmaktadır (Rafaat H. ve diğerleri, 1994).

Verilerin zaman içerisindeki değişimlerini izlemek için her bir anlık durum modelinin incelenmesi gibi bir dezavantajı bulunmaktadır .

- **Güncelleme modeli:** Bu veri modelinde veri tabanının tamamı yerine sadece güncellenmiş obje kaydedilir. Değişen objeler versiyonlar içerisinde depolanmaktadır.

- **Mekan-zaman karma modeli:** Objelerin eski durumları ile yeni durumları aynı katman içerisinde zaman bilgisi ile birlikte kaydedilerek yapılan sorgulamalar ile verilerin istenilen tarihteki durumlarına erişilmektedir ( Worboys, 1994) . Bu model güncelleme modeli ile benzerlikler göstermektedir.

- **3 ve 4 boyutlu karmaşık model:** Bu modelde mekansal objelere zaman yeni bir boyut olarak eklenmektedir. Mekansal objelerin konumlarının zaman boyutu ile gösterilmesi yazılım mühendisliğindeki gelişmelere bağlı olmaktadır.




**Şekil 5.1 :** Coğrafi bilgi sisteminde verilerin zaman içerisindeki değişimlerinin izlenmesi

Veri tabanları üzerinde mekansal verilerin özgeçmişlerinin depolanması güncelleme işlemine tabi olan verilerin versiyonlarda depolanması veya veri tabanındaki ilgili tablolarında aktif oldukları tarihlerin belirlenerek tutulması ile olmaktadır.

Bu yöntemlerin kullanıldığı veri tabanı tablolarında her bir kayıt satırı objenin görmüş olduğu değişiklikleri izleyebilmek için bir ilk tarih ve son tarih diye adlandırılan sütün bilgilerine sahiptir. Yapısal sorgulama dili (SQL) sorgulamaları ilk ve son tarih sütunlarında depolanan bilgileri kullanarak yapılan sorgulamalarda bir objeye yapılan değişiklikler belirlenebilmektedir.

Mekansal objelerin özgeçmişlerinin gelişen teknoloji de göz önüne alınarak hızlı ve etkin bir biçimde izlenebilmesi için versiyon modeli kullanmak mümkün olmaktadır. Bölüm 4.1' de belirtildiği gibi bir versiyon veri tabanının belli bir zaman dilimindeki durumunu sunmaktadır. Böylelikle versiyon modelinde objelerin değişmiş , silinmiş durumları için veri tabanlarında ayrı katmanlar üretmeye gerek kalmadan tarihsel süreçlerini depolamak mümkün olmaktadır.

### 5.2. Değişim Olayları

Veri tabanlarında değişim veri tabanındaki bir kaydın bir tarih veya durumdan, bir başka tarih veya duruma hareket etmesi şeklindedir. Mekansal verilerin depolandığı veri tabanlarında bu değişim verinin geometrisi, topolojik özellikleri ve öznitelik bilgilerin de olmak üzere üç şekilde oluşmaktadır (Abraham ve Roddick, 1999).

Örneğin bir yerleşim yerinin nüfusunun 1950'den 2000'e kadar belirli aralıklarla nüfus bilgilerinin depolandığı bir veri tabanında değişim olayı nüfus olmaktadır.

Değişim olayları verilerin özgeçmişlerinin depolandığı veri tabanlarında verilerin yorumlanması ve verilerin anlamlandırılmasında çok önemli yere sahiptir. Değişim olayları arasındaki sıklık kavramı önemli olmaktadır. Özgeçmiş verileri özel veri tabanı işbitimleri halinde kaydedilir veya bu veriler geniş zaman aralığında özetlenir

### 5.3. Zaman Kavramı

Zaman bilgisi mekansal objelerin depolandığı VTYS' de ayrı bir öznitelik bilgisi olarak görülmektedir (Montgomery, 1995) . Özgeçmiş verilerinin depolandığı veri tabanlarında zaman ; olay zamanı, gözlem zamanı ve veri tabanı zamanı olmak üzere üç farklı zaman kavramı bulunmaktadır (Guptill ve Morrison, 1995).

Olay zamanı, değişimin oluştuğu gerçek zamandır. Gözlem zamanı, bu değişimin fark edildiği zamandır. Veri tabanı zamanı ise değişimin veri tabanına kaydedildiği zamandır. Bir değişim olayı 23 Mayıs'ta olmuş ve bu değişim 25 Mayıs'ta fark edilmiş ve sisteme 26 Mayıs'ta girilmiş olabilir.

### 5.4. Özgeçmiş Sorgulama Yöntemleri

Özgeçmiş bilgilerini tutan veri tabanlarında verilerin depolama yöntemleri ve verilerin güncellendiği coğrafi bilgi sistemi programları verilerin özgeçmişlerine ilişkin yapılacak sorgulamalar genellikle aşağıda belirtildiği gibi 3 farklı şekilde olmaktadır (Esri ,2002) .

**" Bir T zamanında verilerin durumu "**

Veri tabanındaki geçici değişimlerine bağlı olarak belirtilen bir tarihdeki verinin durumunu gösterilmektedir. Eğer bahsedilen tarihte veri yoksa ise gösterim mümkün olmayacaktır.

**"Zaman içinde bir B objesinin değişimi "**

Bu işlem genelde bir dizi sorgulamayı gerektirmektedir. Objenin seçilmesi ile işlem başlar bundan sonra bir diyalog kutusu ile ileri veya geri bir tarihe doğru verinin yaşadığı tarih arasında sorgulama yapılır.

**"Bir T Zamanında B objesinin durumu "**

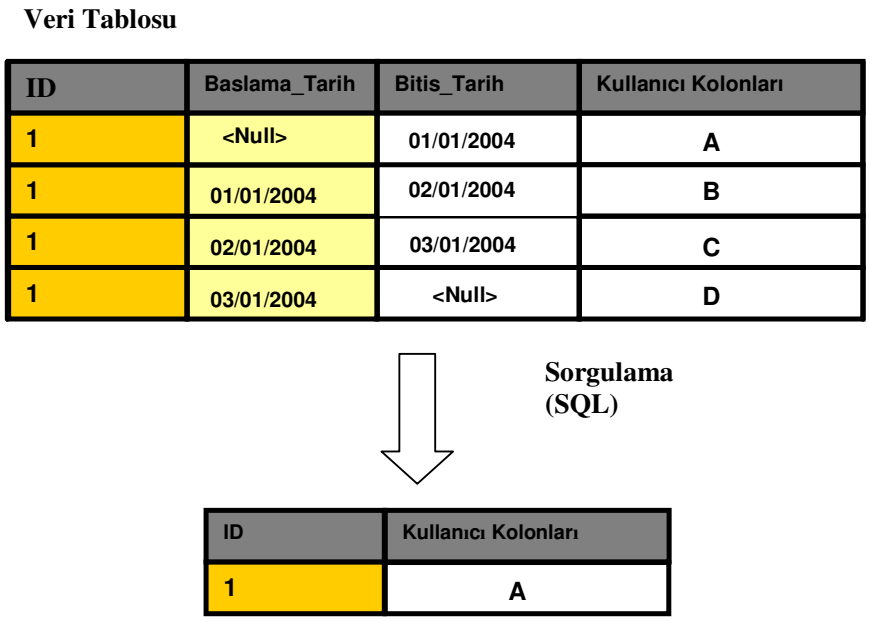
Yapılacak sorgulama kullanıcı tarafından belirtilen andaki konuma yöneliktir.

Veri tabanı yönetim sistemlerinde depolanan veriler için kullanılan işbitim yöntemlerine göre yapılacak özgeçmiş sorgulamaları değişiklik göstermektedir. Kısa

işbitimlerinin kullanıldığı yöntemler ile yapılacak olan özgeçmiş sorgulamaları uzun işbitimlerin kullanıldığı mekansal objeler ile yapılan çalışmalara göre farklılık göstermektedir.

### 5.4.1. Kısa İşbitim (Short Transaction) Uygulamalarında Özgeçmiş Sorgulama

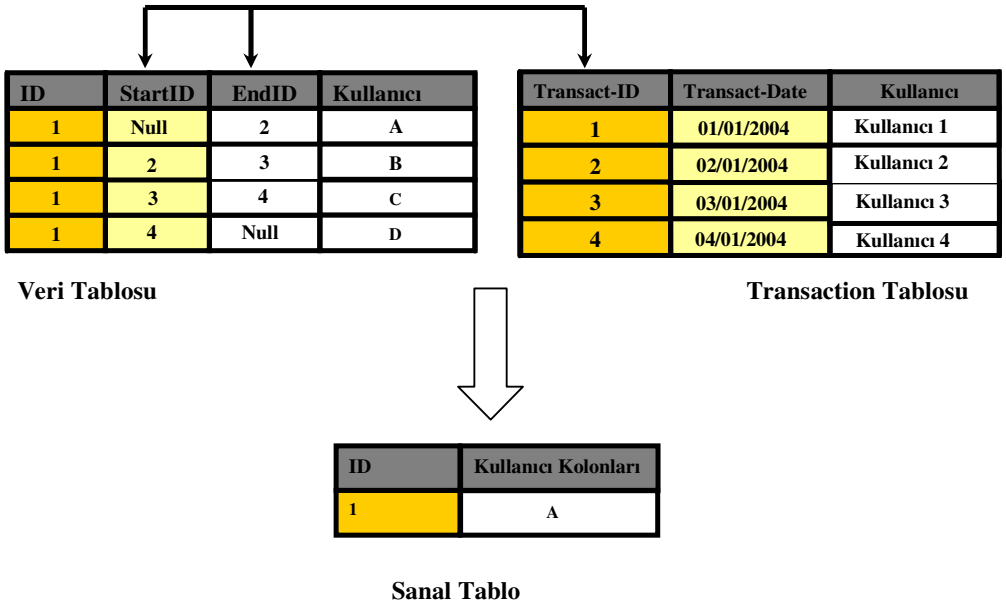
Özgeçmiş sorgulamaları veri tabanında kullanılan işbitim yöntemlerine göre değişiklik göstermektedir. Kısa işbitimlerinin kullanıldığı yöntemlerde veri tabanındaki tablolarda her bir kayıt için başlangıç ve bitiş tarihi olarak adlandırılan kolonlar bulunmaktadır. Başlangıç tarihi verinin aktif hale geldiği tarih , bitiş kolonunda ise verinin pasif hale geçtiği tarih depolanmaktadır. Geliştirilecek özel bir uygulama ile verilerin depolandığı tabloya özel sorgulamalar gönderilerek istenilen bir zaman dilimindeki veriye ulaşmak mümkün olmaktadır (Şekil 6.1) .




**Şekil 5.2 :** Kısa işbitimleri için özgeçmiş izleme

### 5.4.2. Uzun İşbitim (Long Transaction) Uygulamalarında Özgeçmiş Sorgulama

Mekansal verilerin uzun işbitim uygulamalarına konu olması nedeniyle özgeçmiş sorgulamaları için uzun işbitimlerin depolandığı tabloların yönetilmesi gerekli olmaktadır. Verilerin depolandığı tabloda her bir obje bir satır ile ifade edilmekte ve bu satırlarda uzun işbitimleri için bir başlangıç ve bitiş değerleri bulunmaktadır. Bu tablo daha sonra uzun işbitim tablosu ile ilişkilendirilerek herhangi bir zaman dilimindeki verinin durumu gösterilmektedir (Şekil 6.2).




**Şekil 5.3 :** Uzun işbitimleri için özgeçmiş izleme

### 5.4.3. Versiyon Uygulaması ile Özgeçmiş Sorgulamaları

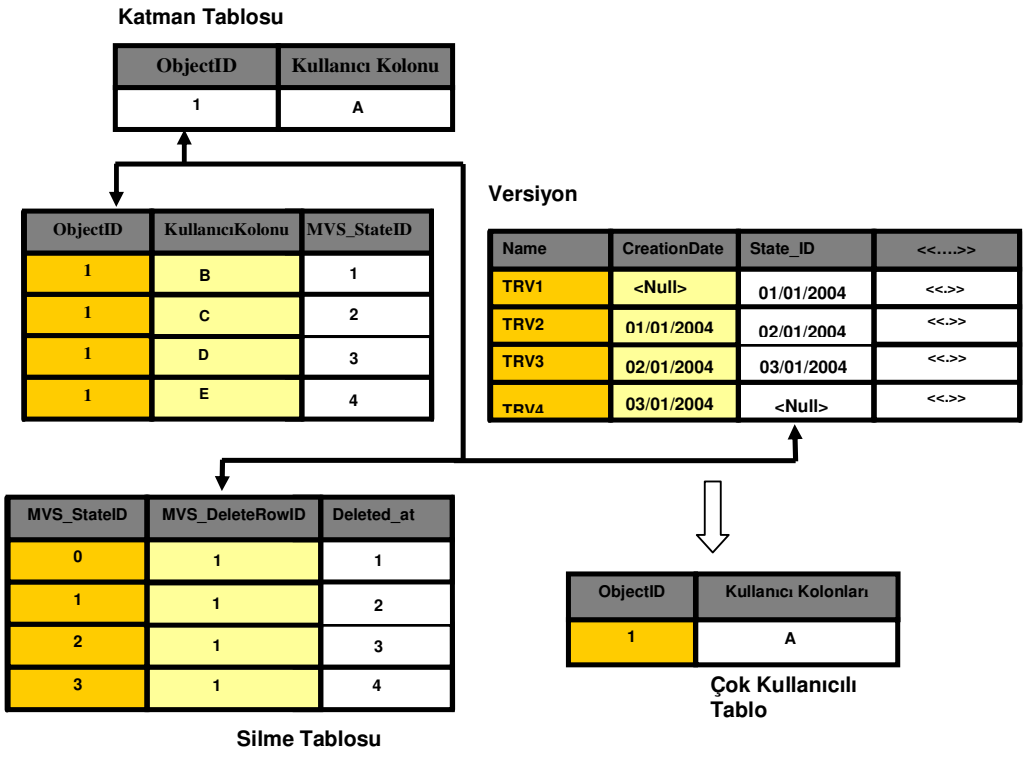
Versiyon özelliğine sahip bir veri tabanı bünyesinde birden fazla versiyon barındırmaktadır. Her bir versiyonun sahibi, tanımı, erişim tipi ve üretildiği üst versiyon bilgileri bulunmaktadır. Versiyonlar başka bir versiyon tarafından yapılan değişikliklerden etkilenmemektedir. Genel olarak her bir versiyon veri tabanında depolanmış verilerin belli bir tarihteki durumunu göstermektedir. Versiyon özelliğine sahip bir veri tabanında "Ana Versiyon " olarak adlandırılan bir versiyon bulunmaktadır. Bu ana versiyon aynı zamanda versiyon ağacının kökü olmaktadır. Ana versiyon hariç diğer tüm versiyonların üretildiği bir üst versiyon bulunmaktadır.

Versiyonların yönetiminde State tablosu çok önemli bir yere sahiptir. Bilindiği üzere versiyon özelliğine sahip bir veri tabanında objeler üzerinde yapılan her bir değişiklik bir State değeri alarak State tablosunda depolanmaktadır.

Versiyon özelliğine sahip her bir katman ekleme ve silme tablosu olarak adlandırılan ve değişime uğrayan objelerin depolandığı tablolara sahiptir. Herhangi bir obje değişikliğe uğradığında değişikliğe uğramamış hali yine veri tabanında saklanmaktadır. Değişiklikleri yöneten bir tablo olan State Tablosu, analiz edilmesi sonucunda objelerin kim tarafından ne zaman ve nasıl değiştirildiği öğrenilmektedir. Verinin kimler tarafından değiştirildiği bilgi State tablosunun versiyon tablosu ile olan ilişkisinden elde edilmektedir.

Verilerin versiyonlarda saklanması verilerin kopyalarının alındığı anlamına gelmemektedir. Verilerin üzerinde yapılan değişiklikler versiyonlarda kaydedilmekte, diğer veriler ise Ana versiyonda bulunmaktadır. Alt versiyonlar Ana Versiyonda bulunan verilerin mantıksal kopyaları üzerinde çalışma yapmaktadır.

Şekil 5.4'de Özgeçmiş sorgulamaları yapılabilmesi için Katman Tablosu, Ekleme Tablosu, Versiyon Tablosu ve Silme Tablosu arasındaki nasıl bir ilişki olduğu ve geliştirilecek olan sorgulamalar ile verinin belli bir tarihteki durumuna nasıl ulaşıldığını anlatılmaktadır .




**Şekil 5.4 :** Versiyon uygulamalarında özgeçmiş sorgulama modeli

Katman tablosu ile silme ve ekleme tabloları arasındaki ilişki objelerin sahip olduğu ObjectID değeri ile sağlanmaktadır. Silme ve Ekleme tablolarının versiyon tablosu ile ilişkileri ise her bir değişiklik işlemi için atanan StateId değerleri ile sağlanmaktadır. Yapılacak sorgulama ifadesi önce belirlenen tarih aralığında ilgili katmanda hangi kullanıcı versiyonun çalışma yaptığı belirlenir. Daha sonra StateID değerleri kullanılarak katmanlardaki güncellenen verilere erişilmektedir.

Versiyon modeli ile transaction tablo modeli karşılaştırıldığında transaction tablosundaki her bir satır versiyon modelinde bir versiyona karşılık geldiği görülmektedir. Versiyon modelinde bir obje katman tablosunda yer almakta ve bu obje ile yapılan silme ve ekleme işlemleri Ekleme ve Silme Tablolarında kaydedilmektedir. Bu tablolar versiyonlar ile ilişkilendirilmektedir. Böylece her hangi bir zaman dilimindeki versiyonun durumuna coğrafi bilgi sistemi uygulama yazılımları veya SQL sorgulama dili ile oluşturulacak uygulamalar ile erişmek mümkün olmaktadır.

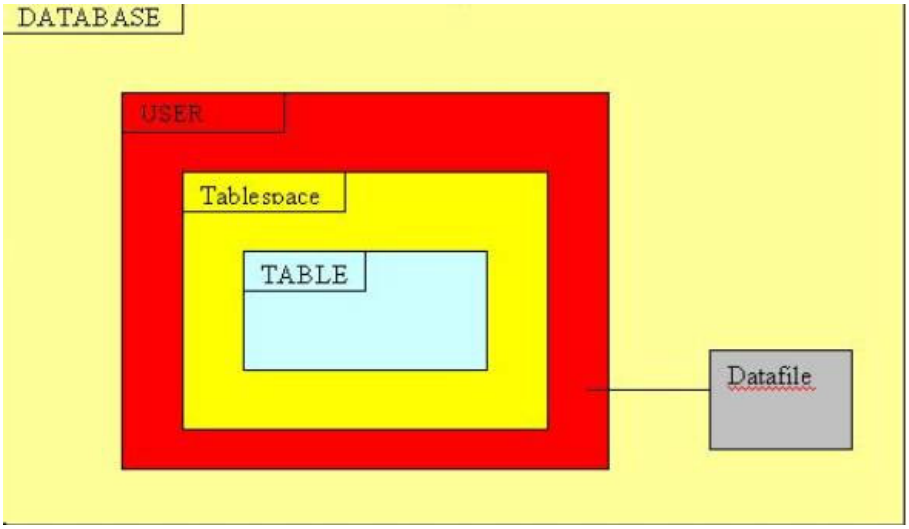
# 6. ÇALIŞMADA KULLANILAN YAZILIMLAR VE ÖZELLİKLERİ

Bu bölümde mekansal verilerin çok kullanıcılı ortamda depolanması ve özgeçmişlerinin takip edilmesi için geliştirilecek uygulamalar ve bu uygulamaların ihtiyaç duyduğu veri tabanı yönetim sistemi, mekansal veri sunucu yazılımı ve yazılım geliştirme platformu detaylı olarak anlatılmaktadır. Bölüm 6.2 de belirtilen sorgulama tipleri geliştirilen uygulama yazılımı ile birlikte detaylı olarak ele alınmaktadır.

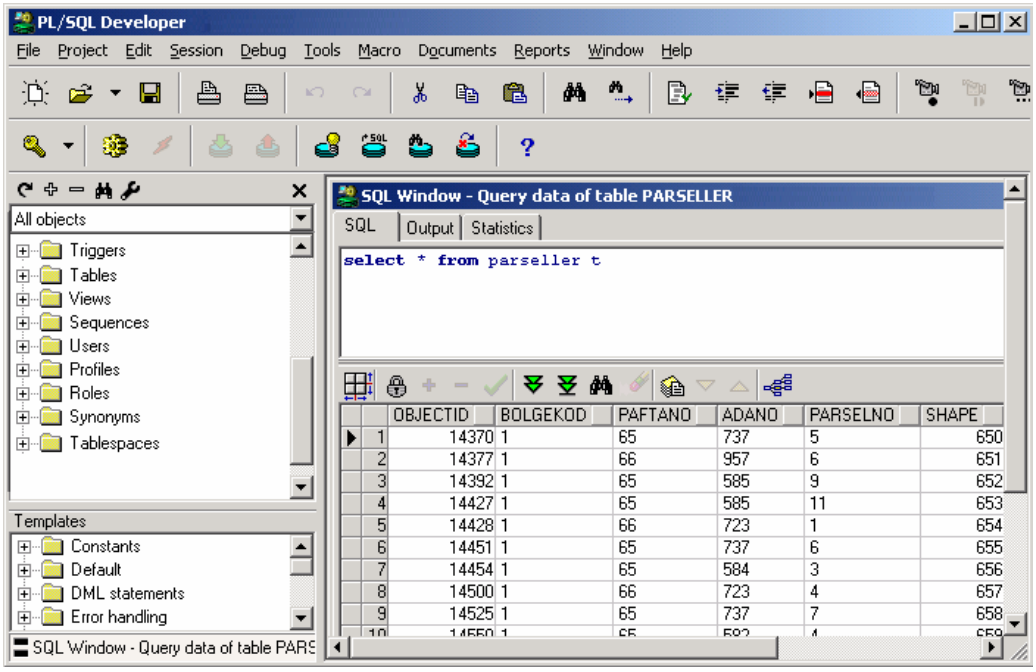
## 6.1. Oracle Veri Tabanı Yönetim Sistemi

Uygulama yazılımının ihtiyaç duyduğu verilerin depolanacağı veri tabanı olarak yaygın olarak kullanımda olması nedeni ile Oracle 9 ilişkisel veri tabanı yazılımı tercih edilmiştir. Geliştirilen sorgulama yazılımı diğer ilişkisel veri tabanlarında Arcsde mekansal veri sunucu yazılımının kullanılması durumunda çalışabilecektir.

Oracle veri tabanı yönetim sistemi fiziksel bölüm ve mantıksal bölüm olmak üzere iki bölümden oluşmaktadır. Fiziksel bölüm, işletim sistemini kullanarak görülebilen dosyalardır. Yani Oracle kurulduğunda fiziksel dosyalardan oluşur. Bunlar data, control ve logfile dosyalarıdır. Mantıksal bölüm ise, Table, Tablespace, İndex, View, User, Sequence gibi veri tabanı nesnelerinden oluşmaktadır. Bu nesnelere erişim sadece Oracle veri tabanına bağlanıp onun üzerinden SQL cümlecikleri ile gerçekleştirilebilir. Oracle veri tabanının yapısı Şekil 6.1' de ve Oracle nesneleri Şekil 6.2' de verilmiştir.




**Şekil 6.1 :** Oracle Veri Tabanı Yapısı




**Şekil 6.2 :** Oracle nesneleri

## 6.2. Oracle Nesneleri

### 6.2.1. İndeks Tanımı

Tablolarda aranan kayıta daha çabuk erişebilmek için tanımlanan nesnelerdir. İndeks tanımlamaları, tablo içinde en fazla aranan alanlara göre yapılır. Örneğin bir kişiye ait bilgiler o kişeye ait Id ve Ad alanları üzerinde yapılacaksa arama işlemini hızlandırmak için Id ve Ad alanlarına göre bir index tanımlamak gerekecektir.

İndeksler herhangi bir tablonun bir kolonu üzerinde tanımlanabileceği gibi birden fazla kolonu üzerinde de tanımlanabilir. İndeks tanımlaması yapıldığı zaman, indeks alanındaki bilgiler sıralanır ve her bir kaydın numarası alınır ve yeni segmentte kaydedilir.

### 6.2.2. Role (Rol) Tanımı

Oracle veri tabanında nesneler kullanıcılara aittir. Bir kullanıcı sahip olduğu nesneyi başka bir kullanıcının kullanımına sunabilir. Bunun için o kullanıcıya hak vermesi gerekir. Birkaç tane kullanıcı olduğu zaman bu işlem pek problem olmamaktadır. Ancak kullanıcı ve nesne sayısı arttıkça bu işlemi gerçekleştirmek oldukça zorlaşmaktadır. İşte roller bu noktada işlemleri kolaylaştırmaktadır. Rol, veri tabanındaki hakların gruplandırılmış halidir. Örneğin, kullanıcılar adlı bir tabloda birden fazla kullanıcının sadece select, insert, ve update işlemleri yapması gerekiyor. Bu işlemi geçekleştirmek için her bir kullanıcıya teker teker bu hakları atamak gerekir. 100 tane kullanıcı için bu işlemi yapıldığı düşünülürse bu işlemin gerçekleşmesi çok vakit alacaktır. Oysa bu hakları içeren bir role tanımı oluşturulursa işlemi 4 kat daha az zaman harcayarak yapmış oluruz.

### 6.2.3. System Privileges Tanımı

Veri tabanı ile ilgili ve kullanıcılara verilebilecek önceden tanımlanmış olan haklardır. Bu haklar SYSTEM ve SYS kullanıcıları tarafından diğer kullanıcılara verilebilmektedir.

### 6.2.4. Sequence Tanımı

Özellikle tekrarlanmayan bir değer olması gereken alanlarda otomatik olarak artan numara verilmek istendiği zamanlarda kullanılan nesnelerdir.

### 6.2.5. Table (Tablo)

Table, veri tabanında bilgilerin saklandığı alanlardır. Table'lar satır ve sütunlardan oluşmaktadır. Satır ve sütunların kesiştikleri alanlar da "field" olarak adlandırılmaktadır.

### 6.2.6. Tablespace

Tablespace, kullanıcıların sahip olduğu nesnelerin mantıksal olarak tutulduğu yerlerdir. Oracle kurulduğunda, standart olarak dört tane tablespace oluşturulur.

**System** **:** Sistem nesnlerinin saklandığı alandır.
**Temporar_Data** **:** Veri tabanı parçalarının geçici olarak saklandığı yerdir.
**Rollback_Data** **:** Her alanın yedeklerinin saklandığı geri alma parçalarının bulunduğu yerdir.
**User_Data** **:** Kullanıcıların nesneleri için oluşturulan tablespace'dir.

### 6.2.7. User

Oracle kuruduğunda, standart olarak 3 tane kullanıcı otomatik olarak oluşturulur. Bu kullanıcılar şunlardır:

Sys : Sistem kullanıcısıdır. Standart olarak veri tabanındaki bütün kullanıcılara ait bütün nesneleri kullanma hakkına sahiptir. Yani Oracle'ın en yetkili kullanıcısıdır.

System : Bu da SYS gibi bir sistem kullanıcısıdır. SYS'nin sahip olduğu bütün haklara sahiptir.

### 6.2.8. View ( Görüntü)

Görüntü (View), bir ya da birden fazla tablodaki bilgilerin sadece gösterildiği bir penceredir. View adından da anlaşılabileceği gibi tablodaki kayıtların aslı değildir,

sadece o anlık bir gösterimdir. View ile ilgili olarak, veri tabanında sadece görüntüyü oluşturan SQL cümlesi yer kaplamaktadır. View, tablolardan sadece sorgulama yapma imkanı (güvenlik açısından) sağlamaktadır.

### 6.2.9. Tetikleyici ( Trigger )

Trigger'lar, bir tablo üzerinde gerçekleştirilen işlemlere bağlı olarak aktif hale gelen ve yapılan işleme bağlı olarak yeni bir işlem gerçekleştiren SQL ifadeleridir. Genel kullanım amaçları şunlardır :

- Veri tabanı üzerinde denetimi sağlamak,
- Geçersiz işlemleri engellemek,
- Sütün değerlerinin otomatik olarak girilmesini sağlamak.

### 6.2.10. Oracle Spatial Veri Sunucusu

Oracle Veri tabanında mekansal verilerin depolanması için kullanılan mekansal veri sunucusudur. Oracle yazılımının tam sürümü ile birlikte kullanılan bir ürünüdür. Kullanıcılara mekansal verilerin depolanması, uygulamalara aktarılması, güncellenmesi ve sorgulanması için gerekli fonksiyonları sağlamaktadır. Temel olarak dört bileşenden oluşmaktadır.

- Mekansal verilerin depolandığı tablolar
- Mekansal indeksleme mekanizması
- Mekansal analizlerin ( çakıştırma, tampon bölge oluşturma, kesişim alma vb.) yapılmasına imkan sağlayan fonksiyonlar
- Mekansal verilerin depolandığı tablolar için yönetimsel araçlar

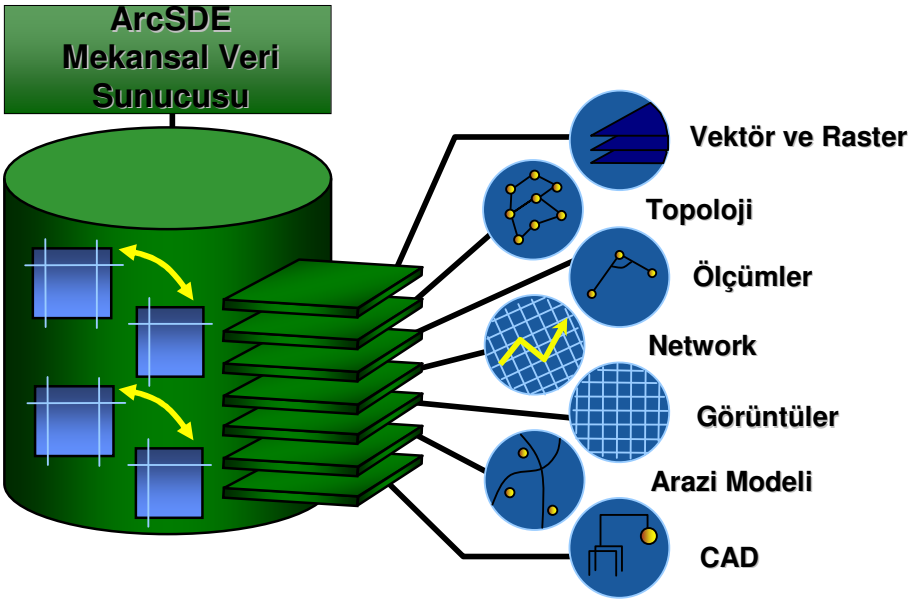
Mekansal verilerin çok kullanıcılı ortamda güncellenmesini uzun işbitim yöntemi ile sağlamakta ve Oracle Workspace Manager yazılımı sayesinde versiyon yöntemi ile çalışılmasına imkan vermektedir.Oracle Spatial yazılımı ile yönetilen mekansal veriler yine bu yazılımın sağlamış olduğu Mapviewer ile görüntülenebilmektedir. Oracle Spatial kullanıcıların kendi fonksiyonlarını yazabilmeleri için gerekli olan Java API desteğini içermektedir.

### 6.3. Mekansal Veri Sunucu Yazılımı (Arcsde- Arc Spatial Database Engine)

Bilindiği üzere mekansal verileri veri tabanları üzerinde depolayabilmek için mekansal veri sunucu yazılımlarına ihtiyaç duyulmaktadır. Bu nedenle Oracle veri tabanı üzerinde mekansal verileri depolayabilmek için mekansal veri sunucu yazılımı olarak ESRI firmasının geliştirmiş olduğu Arcsde yazılımı tercih edilmiştir (Şekil 6.3).

Arcsde çok kullanıcılı coğrafi bilgi sistemlerinde anahtar bir rol üstlenmekte ve Oracle, Oracle with Spatial, Microsoft SQL Server, IBM DB2 ve Informix gibi ilişkisel veri tabanı yönetim sistemleri ile çalışabilmektedir (West, 2001). Arcsde coğrafi bilgi sistemi yazılımları ile ilişkili veri tabanı yönetim sistemleri arasında ağ geçidi görevi yapar. Mekansal verilerin hızlı bir şekilde kullanıcılara ve uygulamalara sunumunu yapmakta ve uzun işbitim ve versiyon uygulamaları için gelişmiş servisler sunmaktadır.

Veri tabanı yönetim sistemleri, mekansal veriler için bir depolama mekanizması gibi kullanılmaktadır. Veri tabanı yönetim sistemleri mekansal verinin anlamını tam olarak tanımlamaz. Arcsde yüksek düzeyde veri entegrasyonu ve bilgi işleme fonksiyonlarını sağlayan çok katlı mimari (uygulama ve depolama ) üzerinde inşa edilmiş bir veri sunucusudur. Arcsde aynı zamanda, veri tabanı yönetim sistemleri temelinde bulunan yeteneklere bakmadan bütün bir mekansal işlevsellik sağlar ve bu mantığı bütün veri tabanları üzerinde uygular. Arcsde, veritbanı içerisinde SQL üzerinden erişilebilir olan her veri tipini kullanarak, veri tabanı tabloları içerisinde mekansal depolama yönetimi sağlar.Arcsde, özel uygulamalar için temelde bulunan mekansal tablolara tam erişim saglayarak açık olarak yayınlanan bir istemci kütüphanesine de olanak tanır. API, C ve Java gibi yazılım dilleri kullanılarak özel uygulamalar geliştirmeye uygundur. Bu esneklik daha bütün ve açık bir yapı, ölçeklendirilebilir çözümler ve kullanıcılar için daha fazla seçenek anlamına gelmektedir.

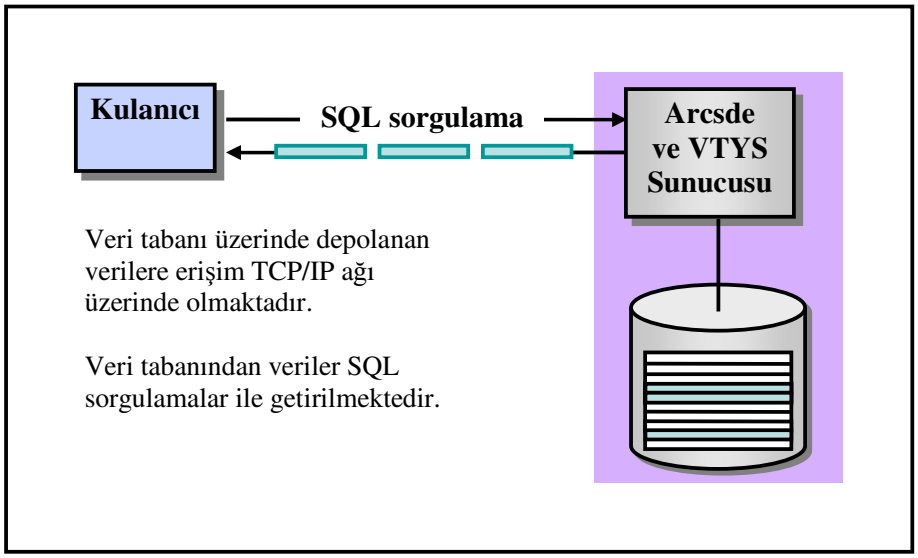



**Şekil 6.3 :** Arcsde mekansal veri sunucusu

### 6.3.1. Arcsde Mimari Yapısı

Arcsde yazılımı istemci/sunumcu (client/server) mimarisi ile üretilmiştir. Bu mimaride kullanıcılar veya uygulamalar sunumcuya istekler gönderir ve sunumcular bu isteği alır, işlemi yaparak sonuçları kullanıcıya gönderirler (Şekil 6.4). Arcsde veri tabanı yazılımı ile aynı bilgisayara kurulabilmektedir.

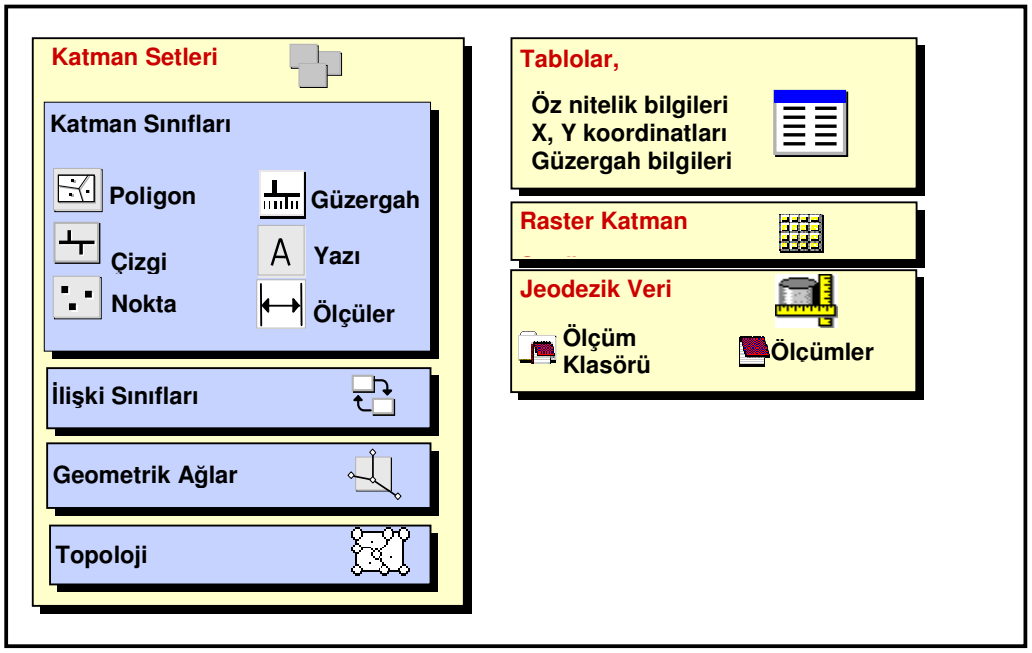
Arcsde kullanıcılara verileri " Veri Tamponlama –Data Buffering" denilen bir yöntem ile göndermektedir. Bu yöntemde veri geniş parçalar halinde kullanıcıya gönderilir . Bu özellik eğer çok sayıda obje ile çalışılıyorsa hız açısından önemli olmaktadır.Arcsde ortak işlem yapma diye adlandırılan bir yöntemle çalışarak işlemin hızlı olabilmesi için kullanıcı veya sunucu bilgisayarda yapılmasına imkan sağlar. Böylece sunucuda depolanmış verilere erişmeden önceden kullanıcı bilgisayarına aktarılmış veriler üzerinde bazı analizler (çakıştırma, kesiştirme vb.) yapılabilmektedir.




**Şekil 6.4 :** Arcsde ile veri tabanına erişim

### 6.3.2. Arcsde Katman Yapısı

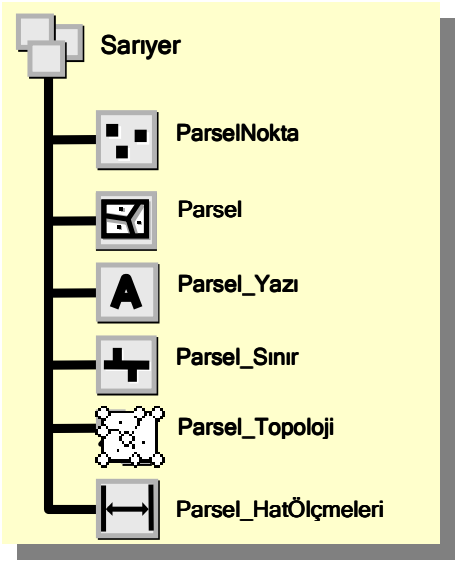
Arcsde yazılımı ile veri tabanları üzerinde katmanlar depolanırken farklı bir yöntem uygulanmaktadır.Arcsde kurulumu tamamlanmış bir veri tabanı yönetim sistemi artık "Mekansal Veri Tabanı - Geodatabase" olarak adlandırılmaktadır. Katmanlar mekansal veri tabanları içerisindeki ana dizin üzerinde oluşturulabileceği gibi katman seti olara adlandırılan gruplar içerisinde de depolanabilmektedir (Şekil 6.5). Tablolar ise mekansal veri tabanı içerisindeki ana dizin üzerinde oluşturulmaktadır. Katmanların birbirleri ile olan veya tablolar ile olan ilişkileri için mekansal veri tabanı içerisinde ilişki sınıfı oluşturmak mümkündür (Woo ve diğerleri, 1999) .




**Şekil 6.5 :** Arcsde katman yapısı

### 6.3.2.1. Katman Setleri

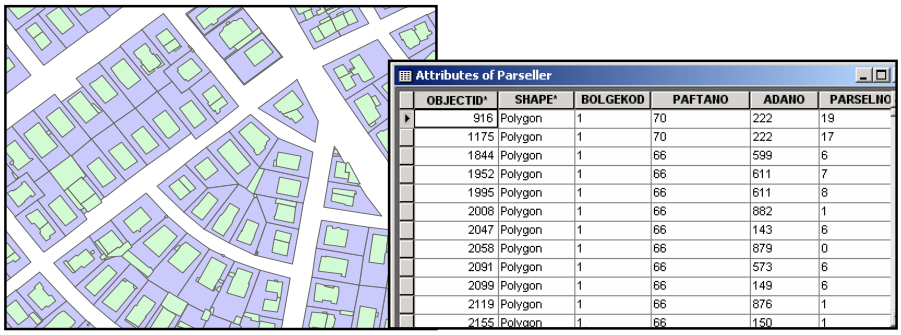
Veri setleri katman gibi mekansal objelerin ve katmanlar ile geometrik bilgi içermeyen tablolar arasında kurulan ilişki sınıflarının depolandığı kümeler olmaktadır (Şekil 6.6). Her bir katman seti bir projeksiyon sistemine sahiptir ve böylece içerisinde depolanan katmanlar da aynı projeksiyon bilgisine sahip olmak zorundadır. Katmanlar arasında topolojik kurallar ve geometrik ağlar kurulmak istenirse bu katmanların katman setleri içerisinde oluşturulması gerekmektedir.




**Şekil 6.6 :** Arcsde katman setleri

### 6.3.2.2. Katman Sınıfları (Feature Class)

Katman sınıfı (Feature Class) aynı tür geometrik özellik ve projeksiyon bilgisine sahip mekansal objeler topluluğudur. Katman sınıflarında yer alan objelerin öznitelik bilgileri öznitelik tablosu olarak isimlendirilen tablolarda depolanmaktadır (Şekil 6.7). Bir obje öznitelik tablosunda bir satıra karşılık gelmektedir. Katmanlar Arcsde içerisinde bağımsız bir şekilde veya katman setleri içerisinde de oluşturulabilmektedir. Katmanların projeksiyon bilgisi eğer katmanlar herhangi bir katman seti içerisinde oluşturulmuş ise bu katman setinden alınmaktadır. Bağımsız bir şekilde oluşturulmuş ise projeksiyon bilgisinin ayrıca tanımlanması gerekmektedir.



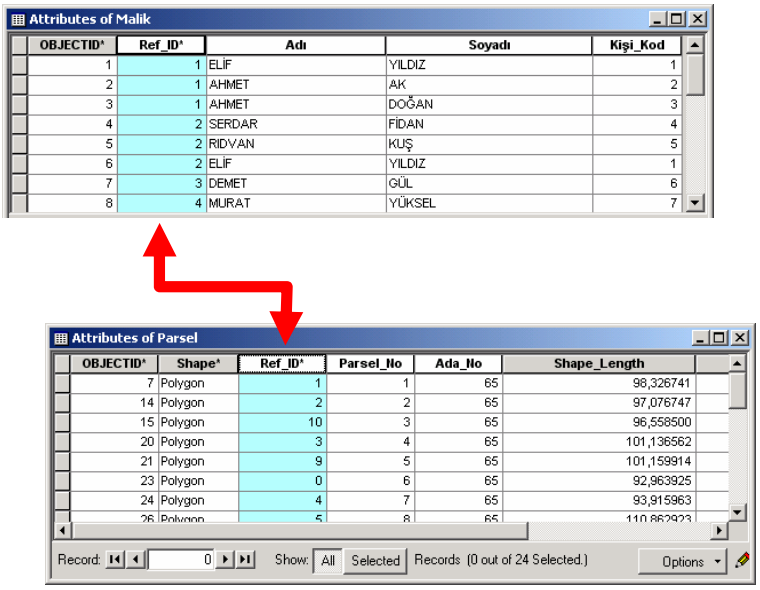

**Şekil 6.7 :** Arcsde katman sınıfı

### 6.3.2.3. İlişki Sınıfları

İlişki sınıfı veri tabanındaki iki veya daha fazla tablo arasındaki ilişkiyi tanımlamaktadır. Bu tablolar katman sınıfları olabileceği gibi sadece öznitelik bilgisi içeren tablolarda olabilmektedir. Bir ilişki sınıfı veri tabanında oluşturulur ise veri tabanındaki diğer nesneler gibi kalıcı olmaktadır. Veri tabanlarında kurulan ilişkilere benzer olarak Arcsde yazılımı ile aşağıdaki ilişkiler oluşturabilmektedir (Şekil 6.8).

- Bire-bir / 1:1 (one-to-one)
- Bire-çok / 1:M (one-to-many)
- Çoka-çok / M:N (many-to-many)

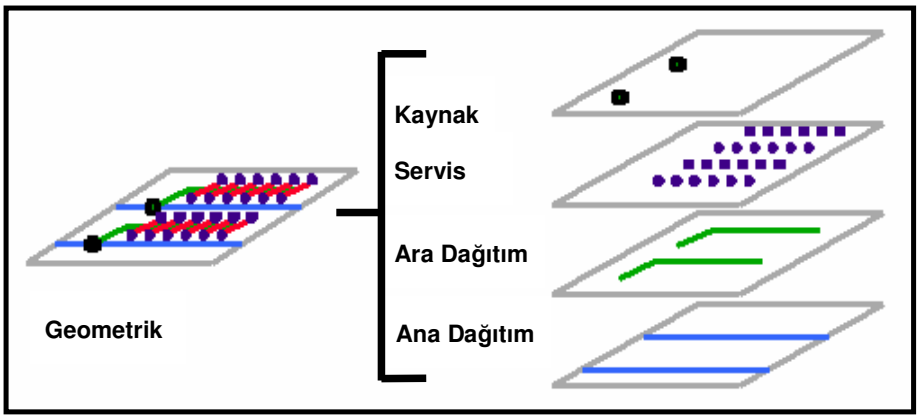
Arcsde yazılımı ile istenirse ilişki sınıfı aralarında ilişki kurulan objelerden herhangi birinin veri tabanında silinmesi durumunda diğer objenin de silinmesine imkan sağlamaktadır. Örnek olarak bir elektrik direğinin silinmesi durumunda üzerindeki lamba objesinin de ilgili tablodan silinmesi işlemi verilebilir. Bu gibi durumlarda silinme işlemi ilişki sınıfı tarafından otomatik olarak yapılmaktadır.




**Şekil 6.8 :** Arcsde ilişki sınıfı

### 6.3.2.4. Geometrik Ağlar

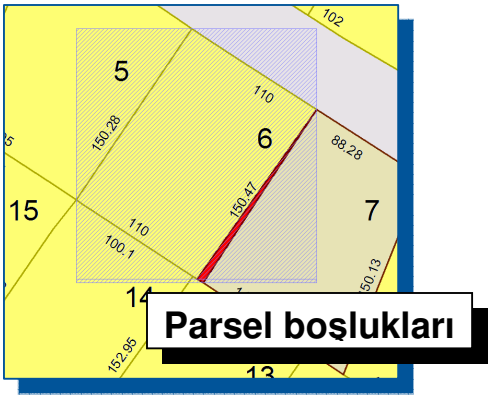
İnsan hareketleri, mal veya herhangi bir hizmetin taşınması veya dağıtılması, herhangi bir kaynak veya enerjinin dağıtımı önceden tanımlanmış bir ağ üzerinde oluşmaktadır. Bu tip hareketleri harita ortamında modelleyebilmek için bazı katmanlar içerdikleri objeler nedeniyle kullanılmaktadır. Bu nedenle bu katmanlarda yer alan objelerin birbirleri ile bağlantılarının ayrı bir biçimde ele alınması gerekmektedir. Bu bağlantılar sayesinde objeler üzerinde iletişim ve akış modellemelerine imkan sağlamaları için geometrik ağlar kurulmaktadır. Geometrik ağlarda nokta ve çizgilerden oluşan katmanlar bağlantı noktası ve ağ çizgisi olarak isimlendirilen katmanlara dönüştürülmektedir. Böylece bu objelerden oluşan katmanlar üzerinde akış izlenmesi yapılarak servis noktaları geometrik ağ üzerinde hangi kaynaklardan beslendiği belirlenebilmekte bu ağlar üzerinde herhangi bir sorun olması durumunda sorunun olduğu noktalar hızlı bir şekilde belirlenebilmektedir.




**Şekil 6.9 :** Arcsde geometrik ağ

### 6.3.2.5. Topoloji

Katmanlar üzerinde güncelleme çalışmaları yapılırken belli bazı kurallara göre yapılması istenildiğinde topolojik kurallar oluşturabilmektedir. Örneğin; parsel katmanında yeni parseller üretirken parsellerin birbirlerine komşu olmaları gerektiği ve aralarında herhangi bir boşluk kalmasın isteniyorsa bu doğrultuda bir kural oluşturularak bu kurala uymayan objeler üretildiğinde kullanıcının program tarafından uyarılması mümkün olmaktadır (Şekil 6.10).




**Şekil 6.10 :** Arcsde topoloji sınıfı

Topolojik kurallara örnek olarak aşağıdaki maddeleri verilebilir.

- Noktalar kapalı alan içerisinde yer almalı
- Kapalı alan objeleri birbirlerini kesmemeli ve aralarında boşluk olmamalı
- Çizgiler kapalı alanları kesmemeli
- Çizgiler aynı zamanda kapalı alanların sınırları olmalı

Bu kurallar Arcsde üzerinde oluşturulmakta ve böylelikle bu kuralların kalıcı olması sağlanmaktadır. Kullanıcılar güncelleme esnasında bu kurallara uymayan objeleri bulma ve bu hataları düzeltme gibi işlemler yapmaktadır.

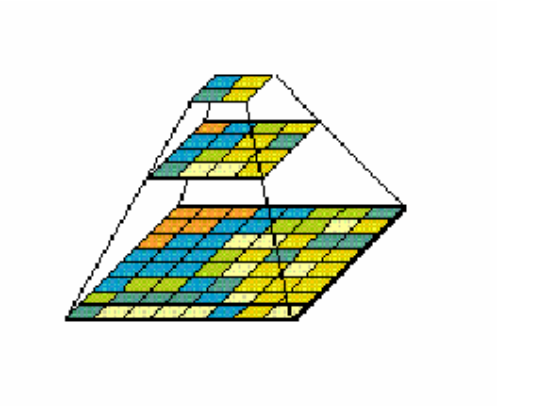
#### 6.3.2.6. Tablo

Arcsde yazılımı ile veri tabanında farklı tablolar oluşturmak mümkün olmaktadır. Bu tablolara veri tabanı üzerinde veya herhangi bir coğrafi bilgi sistemi yazılımı ile erişmek mümkündür.

Arcsde yazılımının veri tabanında oluşturmuş olduğu tablolar ve kullanıcı tarafından oluşturulmuş tablolar olmak üzere iki türde olmaktadır. Arcsde sistem tabloları çoğunlukla kullanıcılar tarafından güncellenmeyip versiyon ve özgeçmiş uygulamaları için kullanılmaktadır.

#### 6.3.2.7. Raster Katman Sınıfları

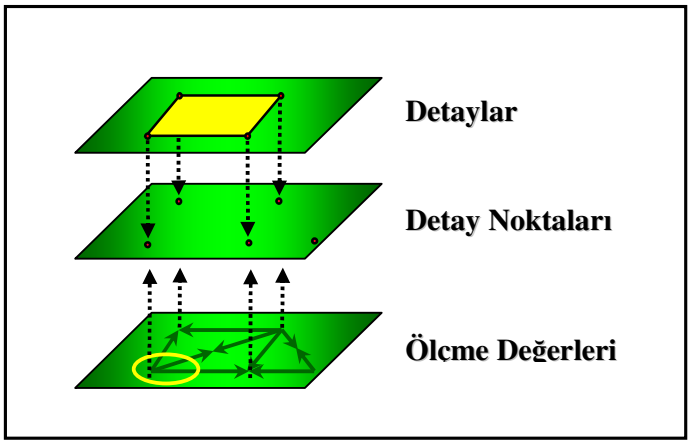
Arcsde yazılımı ile veri tabanlarında raster katmanlarında depolanması mümkün olmaktadır. Arcsde yazılım tarafından desteklenen raster formatlar bmp, jpeg, tiff ve img formatlarıdır. Kullanıcı uygulamalarına raster katmanları gönderirken piramit oluşturma denilen bir yöntem ile ilgili katmanı önce düşük çözünürlüklü bir biçimde göndermektedir. Kullanıcı katman üzerinde büyütme işlemi yapması durumunda çözünürlüğü artırmaktadır. Bu işlem ise yüksek çözünürlük ve hacimli katmanların hızlı bir şekilde görüntülenmesine imkan sağlamaktadır (Şekil 6.11).



**Şekil 6.11 :** Raster katmanların piramitleme işlemi ile kullanıcılara sunulması

#### 6.3.2.8. Jeodezik Veri Katmanları

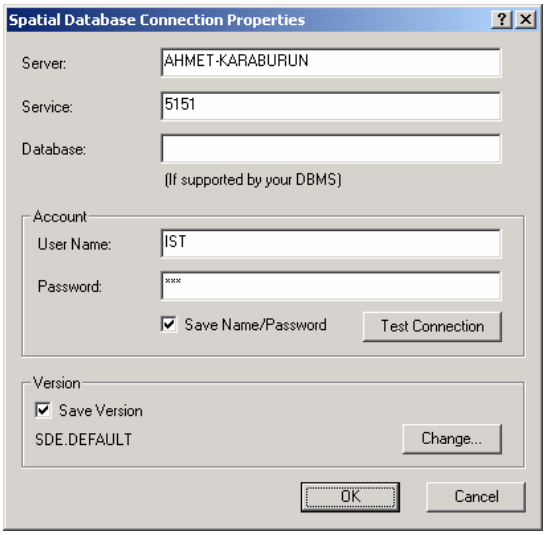
Arcsde yazılımı ile yapılan jeodezik ölçmeler sonucu üretilen detaylar bu ölçmeler ile ilişkilendirilebilmekte ve böylece her bir objenin hangi jeodezik ölçme sonucu elde edildiği bilgisine erişmek mümkün olmaktadır. (Şekil 6.12). Jeodezik ölçmelerin veri tabanlarında depolanması istendiği anda ölçme anının ekran üzerinde oluşturulması anlamına gelmektedir. Ham ölçme değerleri, dengeleme hesapları ve üretilmiş koordinatlar arcsde tarafından yönetilen katmanlar depolanmaktadır.




**Şekil 6.12 :** Arcsde jeodezik veri katmanları

### 6.3.3. Arcsde Verilere Erişim Güvenliği

Arcsde yazılımı ile veri tabanlarına erişebilme belli güvenlik kriterleri ile sağlanmaktadır. Kullanıcılar Arcsde ile verilere önceden veri tabanı yönetim sistemi içerisinde tanımlanmış olan kullanıcı bilgilerini kullanarak erişebilmektedir. Bu erişim işlemi Şekil 6.13' de gösterilen bir arayüz ile sağlanmaktadır. Bu arayüz üzerinde erişilmek istenen sunucu bilgisayar ismi, erişim işleminin yapılacağı port id değeri, erişilecek olan veri tabanı ismi, kullanıcı ismi ve kullanıcı şifresi bilgilere girilerek veri tabanına erişebilmektedir. Böylece yetkisiz kullanıcıların verilere erişimi engellenmiş olmaktadır.




**Şekil 6.13 :** Veri tabanı yönetim sistemine erişim

**Server :** VTYS kurulu olduğu sunucu bilgisayar

**Service :** VYTS ile haberleşmenin yapılacağı port numarası

**Database :** Erişilmek istenen VTYS ismi

**User Name :** Erişim yetkisine sahip kullanıcı ismi

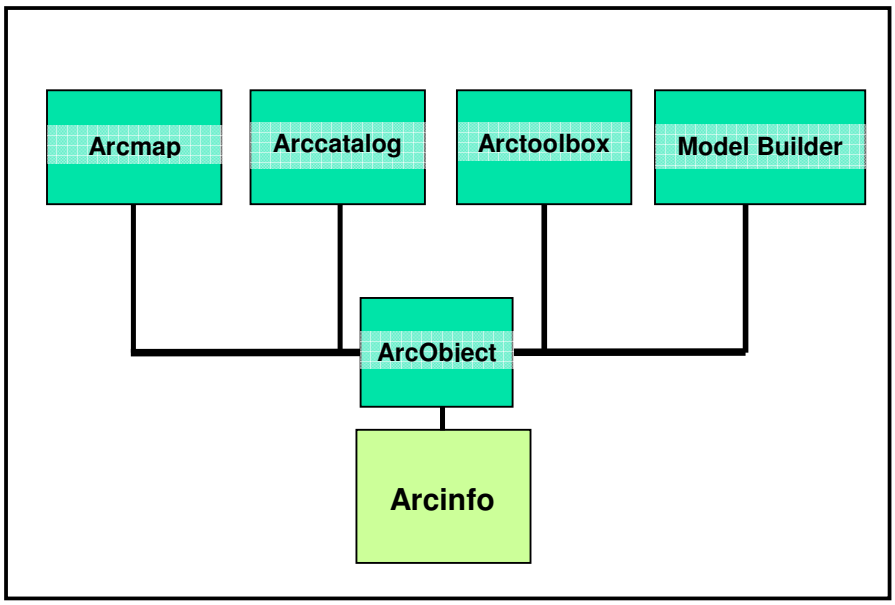
**Password:** Erişim yetkisine sahip kullanıcının şifresi

### 6.4. Coğrafi Bilgi Sistemi Yazılımı (Arcinfo 9.0)

Arcinfo yazılımı Arcmap, Arccatalog ,Arctoolbox ve Model Builder arayüzlerinden oluşan , coğrafi analizler, veri güncelleme, veri yönetimi ve görüntüleme işlemlerinin gerçekleştirilmesini sağlayan bir cografi bilgi sistemi yazılımıdır. (Şekil 6.14). Yazılım Amerikan ESRI firması tarafından üretilmiş olup Arcgis ürün ailesinin bir üyesidir.

Arcsde vasıtası ile veri tabanında depolanmış olan verilere erişebilmekte ve bu veriler üzerinde kullanıcı yetkileri doğrultusunda güncelleme yapılmasına imkan sağlamaktadır.

Arcinfo yazılımının yetenekleri ek programlar ile artırılabilmektedir. Ayrıca kullanıcılar yazılım geliştirme arayüzleri olan Visual Basic, .Net, Java ve Visual C++ ile geliştirmiş oldukları özel uygulamaları Arcinfo üzerinde çalıştırabilmektedirler.




**Şekil 6.14 :** Arcinfo yazılımının arayüzleri

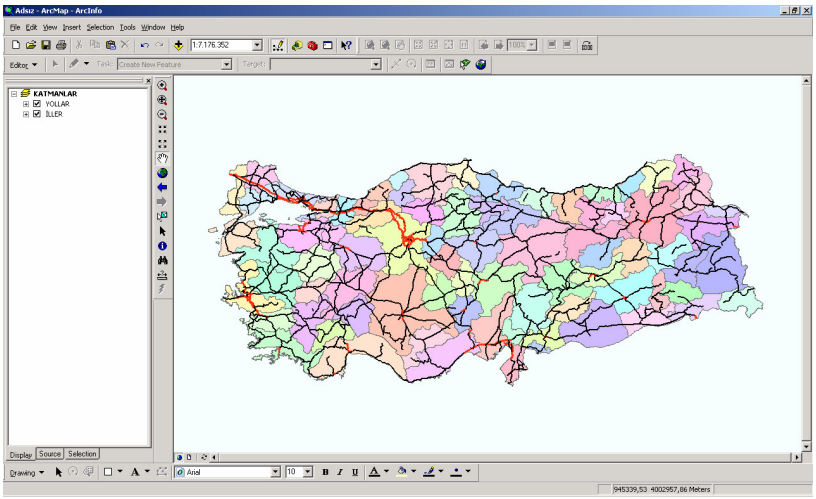
Arcinfo yazılımı ile çalışabilen ek modüllerden bazıları ise Tablo 6.1'de da gösterilmiştir.

**Tablo 6.1 :** Arcinfo Yazılımı Ek Modülleri

| Ek Modül | Açıklama |
|---|---|
| 3D Analyst | Üç-Boyutlu görüntüleme ve analiz araçları |
| Spatial Analyst | Yüzey oluşturma ve raster analiz araçları |
| Geostatistical Analyst | Yüzey modelleri için gelişmiş istatiksel analiz araçları |
| Arcpress | Kaliteli çıktıları çizicilerden hızlı bir şekilde alma araçları |
| Survey Analyst | Jeodezik ölçmelerin yönetildiği araçlar |
| Tracking Analyst | Zaman bazlı verilerin gösterim araçları |
| Arcscan | Raster verilerin vektör verilere dönüşüm araçları |

### 6.4.1. Arcmap Arayüzü

Arcmap arayüzü Arcinfo yazılımının verilerin analizi, güncellenmesi ve çıktıların alınması işlemler için temel uygulaması olmaktadır. Arcmap arayüzü kullanıcıya Veri Penceresi (Data View) ve Çıktı Penceresi (Layout View)gibi iki ayrı pencere sunmaktadır ( Minami ve diğerleri, 1999) . Veri üretim ve analiz işlemleri veri penceresinde yapılmakta çıktılar ise çıktı penceresinde üretilmektedir. Arayüzün sol tarafında ise Arcmap ortamında eklenmiş katmanların bilgilerini sunan bir içerik penceresi bulunmaktadır (Şekil 6.15). Katmanların renk ve işaretleştirme çalışmaları bu içerik penceresinde yapılmaktadır.




**Şekil 6.15 :** Arcmap Arayüzü

Arcmap arayüzünde kullanıcıların görüntüleme, veri üretme, veri sorgulama ve çıktı alma fonksiyonlarının yer aldığı araçlar bulunmaktadır. Bu araçlar ile gerçekleştirilen işlemlerden bazıları aşağıda belirtilmiştir.

- Coğrafi bilgiler için çeşitli bilgi sınıflarına göre istenilen aralık ve özelliklere göre (tek yada sınıflanmış halde) tematik haritalama, öznitelik ve mekansal sorgulama ve analiz yapabilmektedir.

- Yazılım en son kaydetmeye kadar sınırsız sayıda geri alma ve geri alınan işlemleri yeniden yapabilmektedir.

- Harita ve öznitelik veri tabloları arasında çift taraflı ilişki kurulabilmektedir.

- Yazılımın çıktı hazırlama ortamına (Layout), harita, tablo, grafik (bar, pasta, çizgi) ve lejant yerleştirilebilmekte, fontlar istenilen renkte ve büyüklükte kulanılabilmeli, yazılar gerekli açı değeri verilerek döndürülebilmektedir.

- Farklı projeksiyonlara sahip dosyalar aynı harita penceresinde üst üste açılabilmekte ve koordinatlar istenilen projeksiyonda ve birimde görüntülenebilmektedir.

- Katmanların görüntülenme aralığı kullanıcı tarafından verilen ölçek aralığına göre ayarlanabilmektedir.

- Bir veya birden fazla dökuman veya resim dosyasını tek bir mekansal veriye ilişkilendirebilmekte ve bu veri fare ile seçildiğinde kullanıcının isteğine göre linklenen dokuman veya resim dosyaları görüntülenebilmektedir.

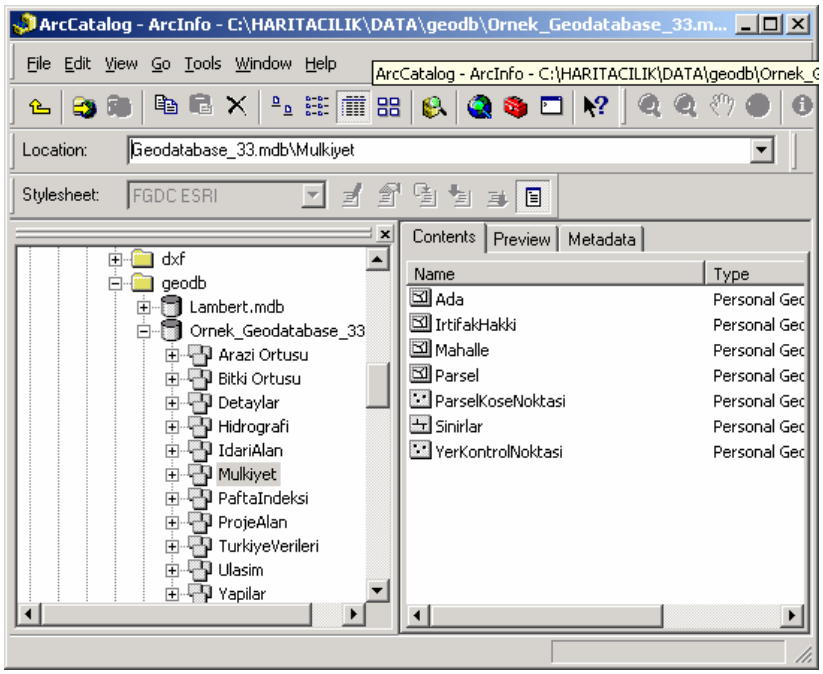
- Kullanıcı tarafından tanımlanan koordinatlara göre veya tanımlanan açı ve uzunluk değerlerine gore nesne oluşturabilmelidir Bir çizgiye dik veya parallel çizgileri otomatik olarak çizebilmelidir. Poligonları otomatik kapatabilmelidir. Üç noktaya göre eğri veya tanjant eğrisi oluşturabilmelidir. Bir çizgiyi belirtilen aralıklara gore eşit parçalara otomatik olarak bölebilmelidir. Çizgi-nokta, çizgi-çizgi, çizgi-poligon kesişimlerine göre yeni nesneler oluşturabilmelidir

- Çizgi katmanlarında yer alan her bir obje için köşe noktalarının dinamik bölümleme özelliği sayesinde x, y ve z koordinatlarının yanı sıra dördüncü boyut olarak zaman veya uzunluk değeri depolanabilmektedir.

- Ağ analizleri (rota bulma, en kısa yol bulma , akış izleme) yapabilmektedir.

### 6.4.2. Arccatalog Arayüzü

Arccatalog arayüzü katman ve tabloların oluşturulması ve yönetilmesi işlemlerinin yapıldığı arayüz olmaktadır. Microsoft Windows işletim sisteminde yer alan Windows gezgini uygulamasına benzer bir arayüze sahiptir (Vienneau A. ve diğerleri, 1999). (Şekil 6.16) . Bu arayüz ile yapılan uygulamalar ise;




**Şekil 6.16 :** Arccatalog arayüzü

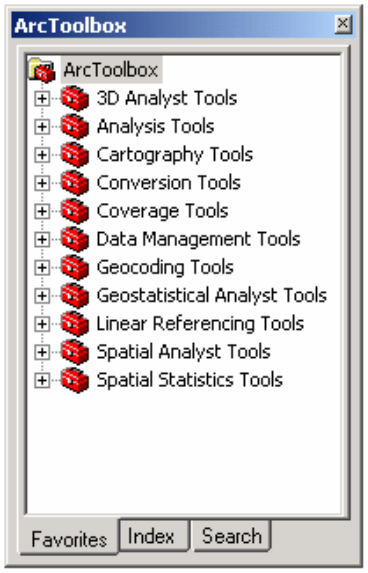
- İlişkisel veri tabanlarına bağlatı kurulması
- Arcsde yazılımı vasıtası ile ilişkisel veri tabanları üzerinde veya lokal olarak herhangi bir klasörde Arcgis ürün ailesinin kullanmış olduğu dosya tiplerinin oluşturulması ve yönetilmesi işlemleri
- Katmanların ve tabloların öngörünüm seçeneği ile görüntülenmesi
- Drag,drop Özelligi (ArcMap ve ArcToolBox için)
- Katmanların projeksiyon sistemlerinin oluşturulması

- Geometrik ağların ve topoloji kurallarının oluşturulması

### 6.4.3. Arctoolbox Arayüzü

Arctoolbox arayüzü ile özellikle veri analiz ve dönüşüm işlemleri yapılmaktadır. Bahsedilen bu işlemlerin yapılmasına imkan sağlayan fonksiyonlar kullanıcıların kolaylıkla erişimi için bu arayüzde toplanmış durumdadır (Şekil 6.17). Bu fonksiyonlar ;

- Veri Yönetim İşlemleri
- Veri Dönüşüm İşlemleri
- Vektörel Analizler
- Adresleme
- İstatiksel Analizler
- 3 Boyutlu Analizler
- Raster Analizleri

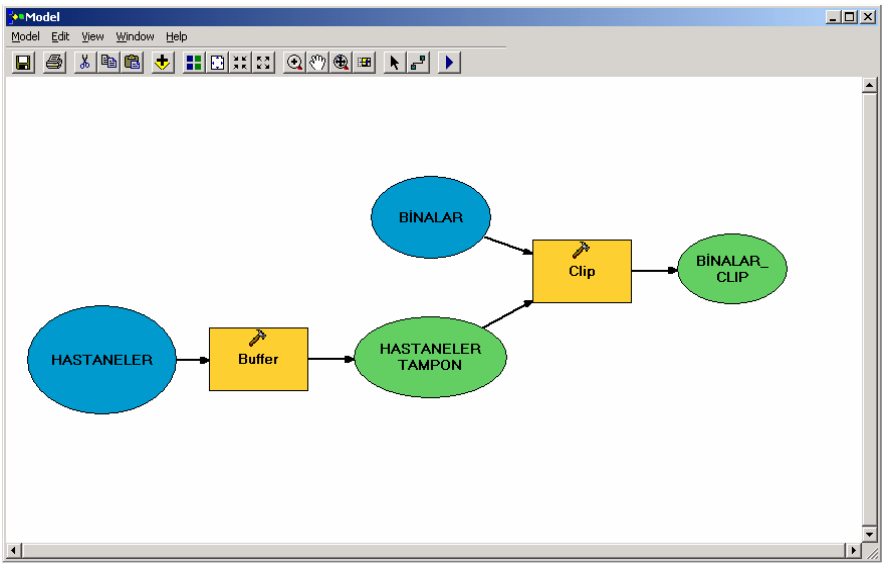



**Şekil 6.17 :** Arctoolbox arayüzü

Arctoolbox arayüzüne Arcmap ve Arccatalog arayüzlerinde bulunan kısayol butonları ile ulaşmak mümkündür.

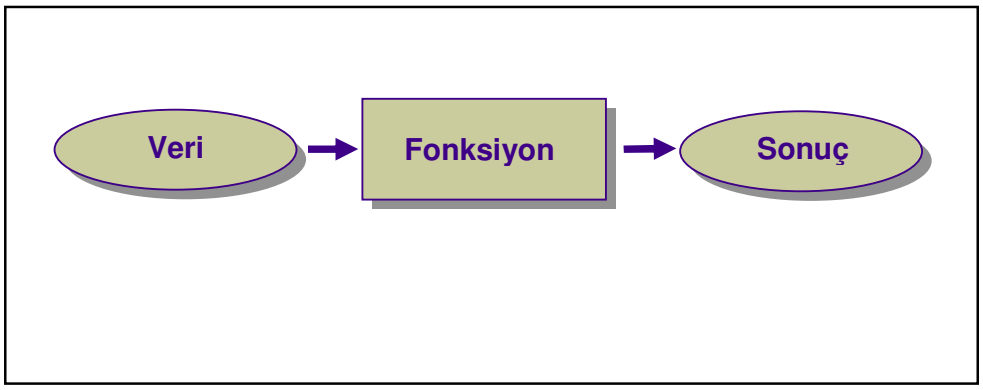
### 6.4.4. Model Builder Arayüzü

Model Builder arayüzü kompleks yapıdaki veri üretimi, analizi aşamalarından oluşan iş akışlarının modellenmesi ve yönetilmesine imkan sağlayan arayüzdür (Şekil 6.18). Kullanıclar bu arayüz üzerinde verileri, işlem adımlarını ve sonuç ürünleri aşamalandırmakta ve bu işlemler için geliştirmiş oldukları özel uygulamalarınıda kullanabilmektedir ( McCoy, 2003).




**Şekil 6.18 :** Model Builder arayüzü

Her bir işlem adımı girdi verileri, verilere uygulanacak fonksiyon ve çıktı verileri parametrelerinden oluşmaktadır (Şekil 6.19). Model builder arayüzüne Arctoolbox arayüzü gibi Arccatalog ve Arcmap arayüzlerinden erişmek mümkündür. Üretilen modeller kaydedilebilmekte ve sonrada kullanılabilmektedir.



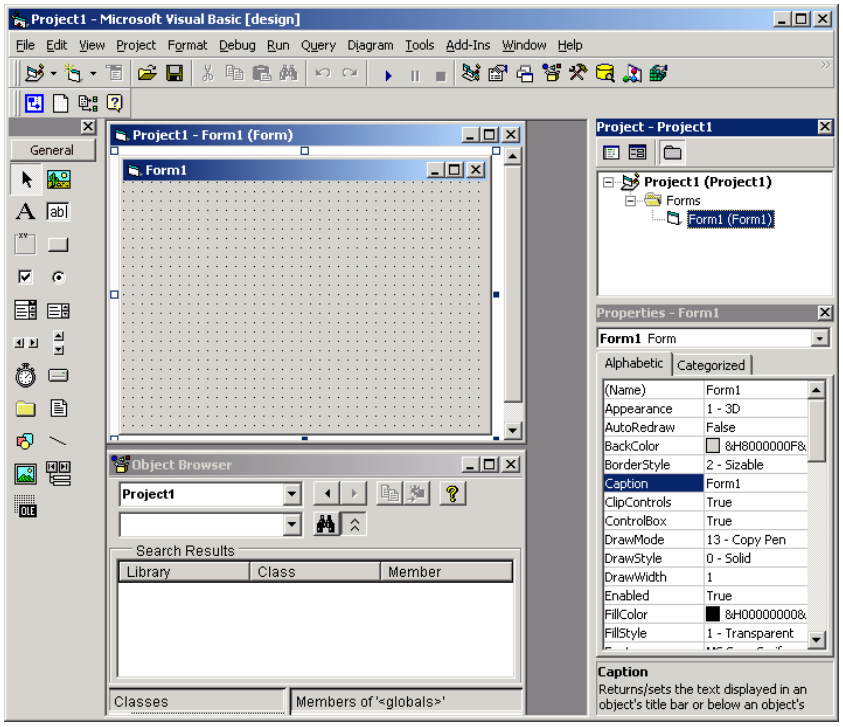

**Şekil 6.19 :** Model Builder parametreleri

### 6.4.5. MS Visual Basic Yazılımı

Microsoft firması tarafından geliştirilen Visual Basic içerisinde uygulamaların temel yapı taşları form ve denetimleri görsel olarak tasarlandığı grafik bir ortam sunan yazılım geliştirme platformudur.Visual Basic uygulama geliştiricilerin işlerini kolaylaştıracak proje, formlar, sınıf nesneleri, şablonlar ve denetimler gibi araçlar sağlamaktadır. ( Şekil 6.20 )


**Şekil 6.20 :** Visual Basic projesi

Visual Basic kullanıcılarına menü çubuğu, araç çubuğu, proje gezgini, özellikler penceresi, araç kutusu, form düzenleyici ve nesne gezgini gibi ana bileşenlerden oluşmaktadır (Şekil 6.21). Kullanıcılar oluşturdukları formlar üzerinde araç çubuğunda yer alan objeleri formlar üzerine yerleştirerek kendilerine özgü arayüzler oluşturarak bu objelerin içerisine kendi geliştirecekleri kodlarla özgün programlar üretebilmektedirler. Bu bileşenlere Entegre Geliştirme Ortamı ( IDE- Integrated Development Environment) olarak isimlendirilmektedir. Visual Basic ile geliştirilmiş programlar kendi başına çalışabilen programlar olabileceği başka programlar üzerinde de çalışabilmektedir.



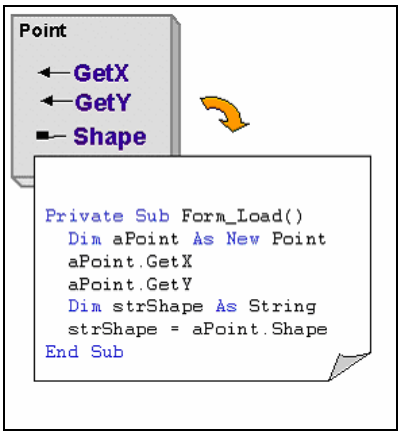

**Şekil 6.21 :** Visual Basic bileşenleri

### 6.4.6. COM Teknolojisi

COM (Component Object Model) Microsoft firması tarafından geliştirilmiş ve önceden geliştirilmiş program objelerinin yeniden farklı programlar üzerinde kullanılabilmesine imkan sağlayan bir teknolojidir. COM yazılım geliştirme ortamı olmayıp Visual Basic,Visual C ++, Visual J + + ve Dephi ile geliştirilmiş programlar olmaktadır. Böylece herhangi bir programla dili kullanılarak geliştirilmiş objeler farklı programlama dillerinde kullanılabilmektedir. Her defasında yeniden bir çok şeyi kodlamak yerine önceden yazılmış nesnelerin kullanılması uygulama geliştirme işlemlerini oldukça hızlandıracaktır. COM bir yazılım dili olmayıp farklı yazılım dilleri ile uyumlu programların geliştirilmesine imkan sağlayan teknolojidir (Grimes ve diğerleri, 1999).

COM arayüz bazlı programlama tekniği üzerine inşa edilmiştir. Bu arayüzler kullanıcıların COM objelerinin kaynak kodlarına ihtiyaç duymadan kendi uygulamalarında kullanmalarına imkan sağlamaktadır. Böylece bu arayüzler kullanıcılar ile kaynak kodlar arasında bir izolasyon sağlamaktadır. Şekil 6.22' de nokta objesi ve kullanılması bir örnek üzerinde anlatılmaktadır.

Burada point objesinin GetX() ve GetY() methodları noktanın koordinatlarını ihtiva fonksiyonlar içerebilir.

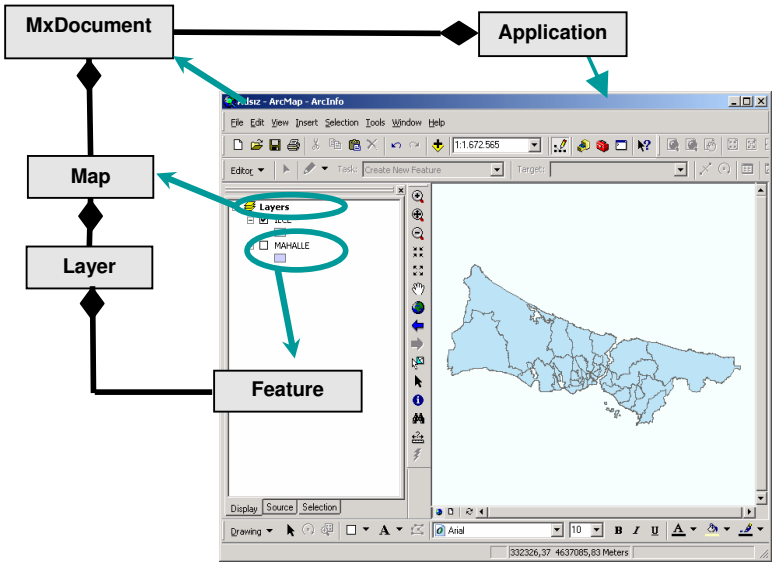



**Şekil 6.22 :** Point Objesinin kullanımı

### 6.4.7. ArcObjects Uygulama Geliştirme Objeleri

Arcobjects ESRI firması tarafından COM - Component Object Model teknolojisi ile üretilmiş objeler topluluğu olmaktadır. Uygulama geliştiriciler herhangi bir yazılım geliştirme platformunda Arcobjects objelerine erişip bu objeler ile özel uygulamalar geliştirerek bu uygulamları Arcinfo yazılımın Arcmap, Arccatalog, Arctoolbox arayüzleri üzerinde çalıştırabilme imkanına sahiptir (Zeiler, 2002).

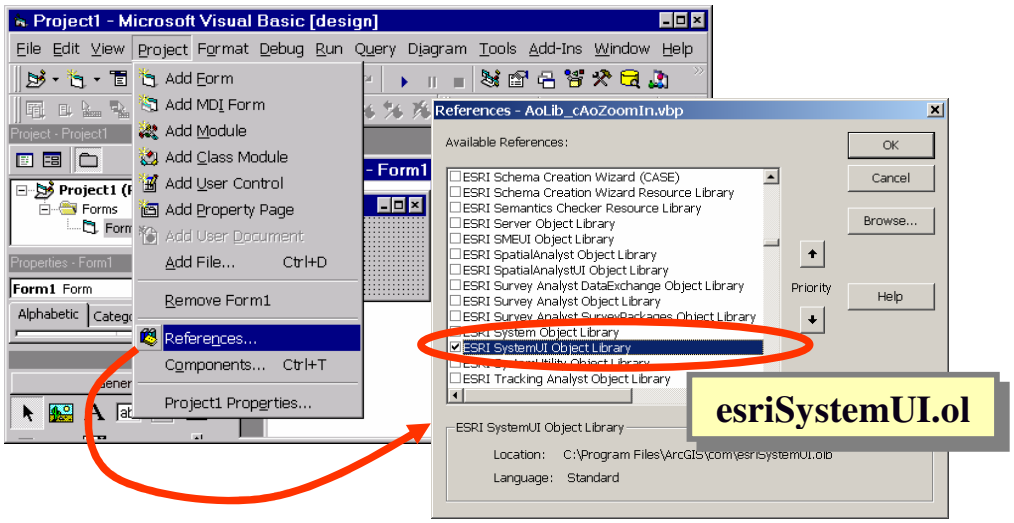
Arcinfo yazılımı üzerindeki her bir uygulama tamamen Arcobjects objelerinden oluşmuştur. (Şekil 6.23).




**Şekil 6.23 :** Arcmap üzerindeki arcobjects objeleri

### 6.4.7.1. ArcObjects Objelerine Erişim

yazılım geliştiriciler kullandıkları yazılım geliştirme platformlarında Arcobjects objelerine erişebilmek için Arcinfo yazılımını kurmaları gerekmektedir. Bu işlem yazılım geliştirme platformlarında COM objelerini referans alma olarak adlandırılır ve MS Visual Basic için bir örnek gösterilmiştir (Şekil 6.24).




**Şekil 6.24 :** ArcObjects objelerine Erişim

Bu işlem ile Arcobjects objeleri uygulama yazılımın herhangi bir aşamasında kullanılabilecek duruma gelmektedir. Aşağıda Arcobjects objelerinden nokta ve çizgi objelerinin kullanılmasına ilişkin örnek bulunmaktadır.Arcobjects objeleri önce değişken olarak tanımlanır ve daha sonra bu objelerin birer kopyaları oluşturulur;

```
Dim Nokta  As esriGeometry.IPoint
Dim Cizgi As esriGeometry.ILine
Set Nokta = New esriGeometry.Point
Set Cizgi = New esriGeometry.Line
```

# 7. UYGULAMA YAZILIMI

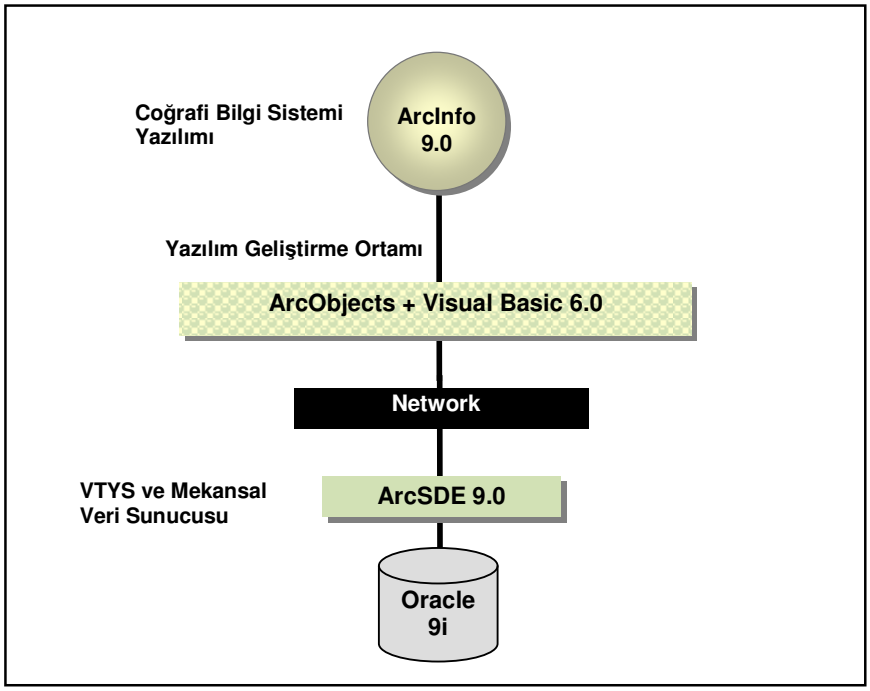
Coğrafi bilgi sistemi kullanıcıları çoğunlukla verilerinin zaman içerisinde geçirmiş oldukları değişiklikleri izlemek istemektedirler. Veri tabanlarının bu işlem için uygun bir biçimde modellenmesi veri tabanının belli bir zaman diliminde nasıl göründüğünün analiz edilmesine ve görüntülenmesine imkan sağlar.

Versiyon özelliğine sahip veri tabanlarında özgeçmiş izlenmesi veri tabanının bütününde veya veriler bazında mümkün olmaktadır. Bu tip veri tabanlarında istenen herhangi bir zaman diliminde verinin görüntülenebilmesi için ihtiyaç duyulan tüm bilgiler mevcuttur. Burada önemli olan nokta verilerin kesinlikle silinmeyip tüm güncellemelerin ekleme ve silme tablolarında depolanmasıdır. Bu yöntem bize versiyon özelliğinin neden özgeçmiş uygulamaları için en uygun yöntemlerden biri olduğu konusunda fikir vermektedir.

Mekansal verilerin çok kullanıcılı ortamda depolanması ve özgeçmişlerinin izlenmesi için Arcinfo üzerinde çalışan bir uygulama sunulmuştur. Bu program Oracle ilişkisel veri tabanında depolanmış verilere Arcsde vasıtası ile erişmekte, versiyonlar üreterek her bir versiyonda güncellenmiş verilerin özgeçmişlerinin izlenmesine imkan sağlamaktadır. Böylece mekansal veriler çok kullanıcılı ortamda depolanırken aynı zamanda özgeçmişlerinin izlenmesi de mümkün olmaktadır.

## 7.1. Uygulama Yazılımının Mimari Yapısı

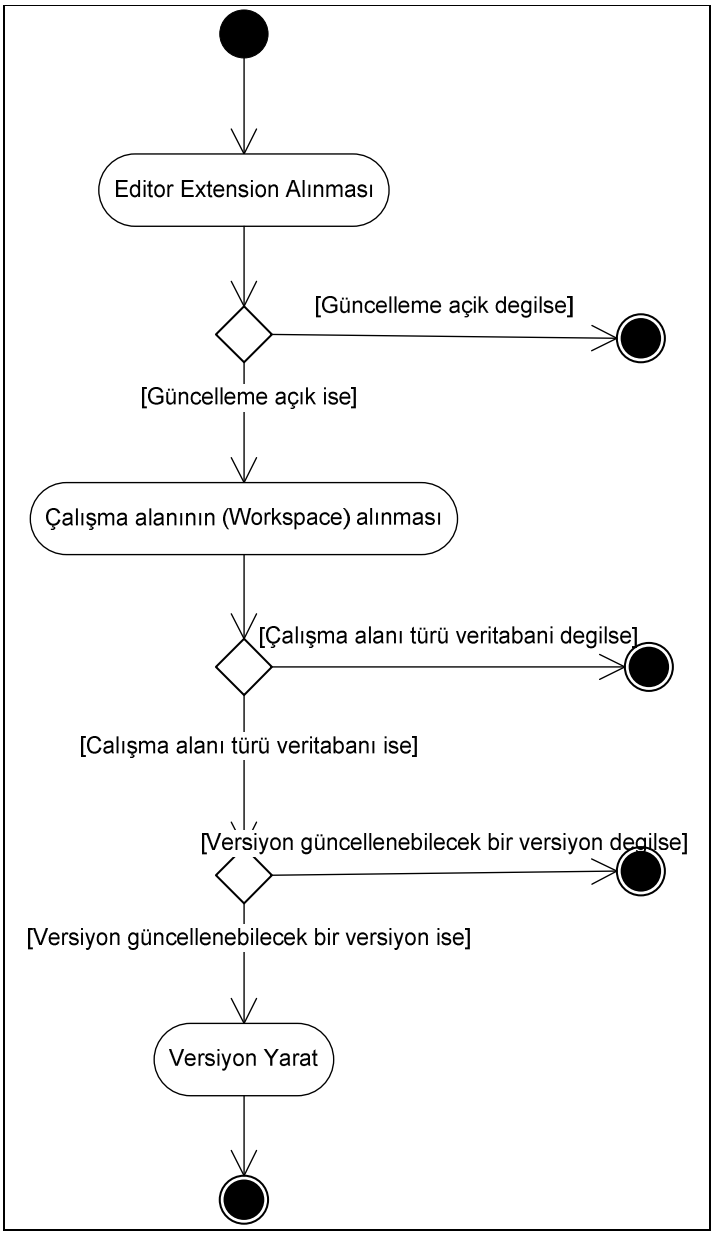
Uygulama yazılımı için kullanılan yazılımlar Şekil 7.1' de gösterilmiştir. Burada kilit rolü Arcinfo ve Arcsde yazılımı oynamaktadır. Arcinfo yazılımı ile Arcsde mekansal veri sunucu üzerinden belirlenmiş güvenlik kurallarına göre Oracle veri tabanına erişmekte ve burada depolanmış olan mekansal veriler ile çalışmaktadır ve tüm bu işlemler bilgisayar ağı üzerinde gerçekleşmektedir.



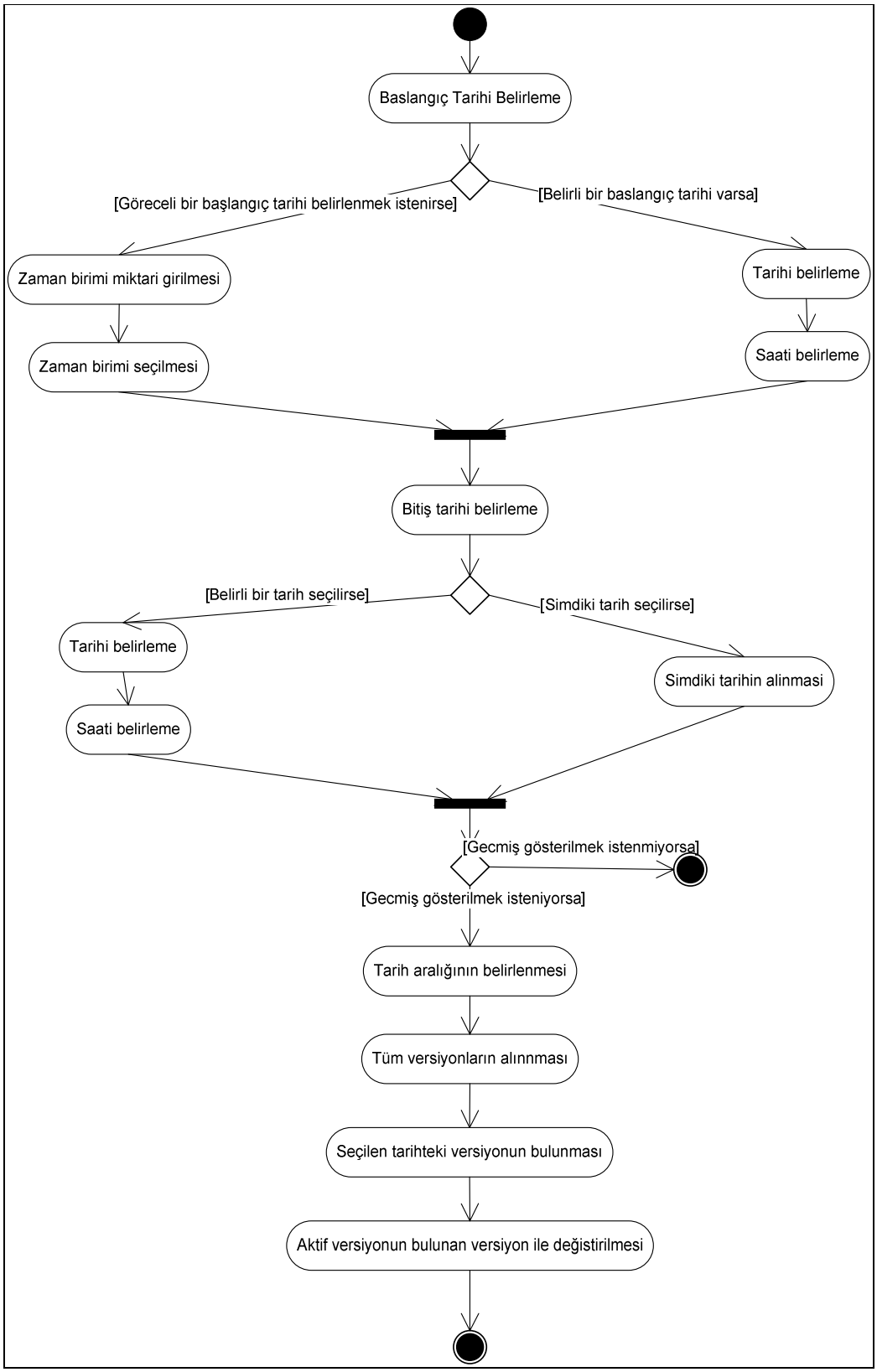

**Şekil 7.1 :** Uygulama yazılımı mimari yapı

### 7.1.1. Uygulama Yazılımı İş Akış Diyagramları

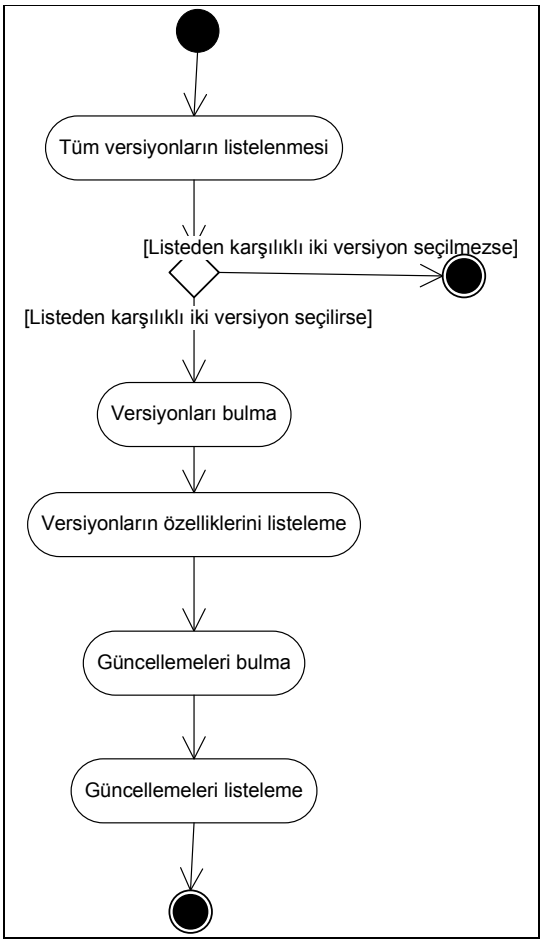
Özgün uygulamanın içerisinde yer alan işlemlerin UML aktivite diyagramları Şekil 7.2, Şekil 7.3, Şekil 7.4 ve Şekil 7.5' de gösterilmiştir.



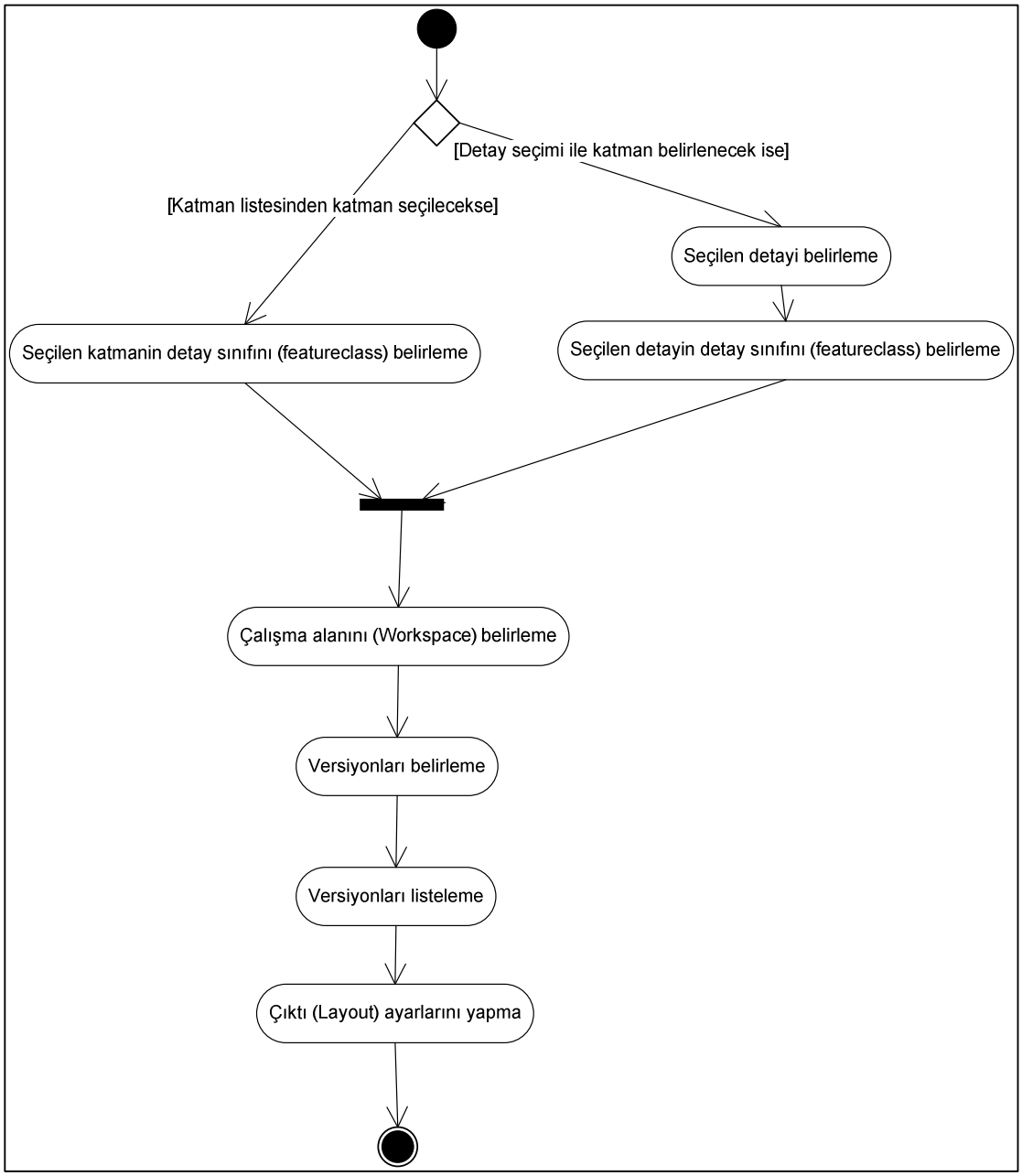

**Şekil 7.2 :** Versiyon oluşturma UML aktivite diyagramı




**Şekil 7.3 :** Özgeçmiş sorgulama kriterleri için UML aktivite diyagramı




**Şekil 7.4 :** Değişikliklerin izlenmesi işleminin UML aktivite diyagramı



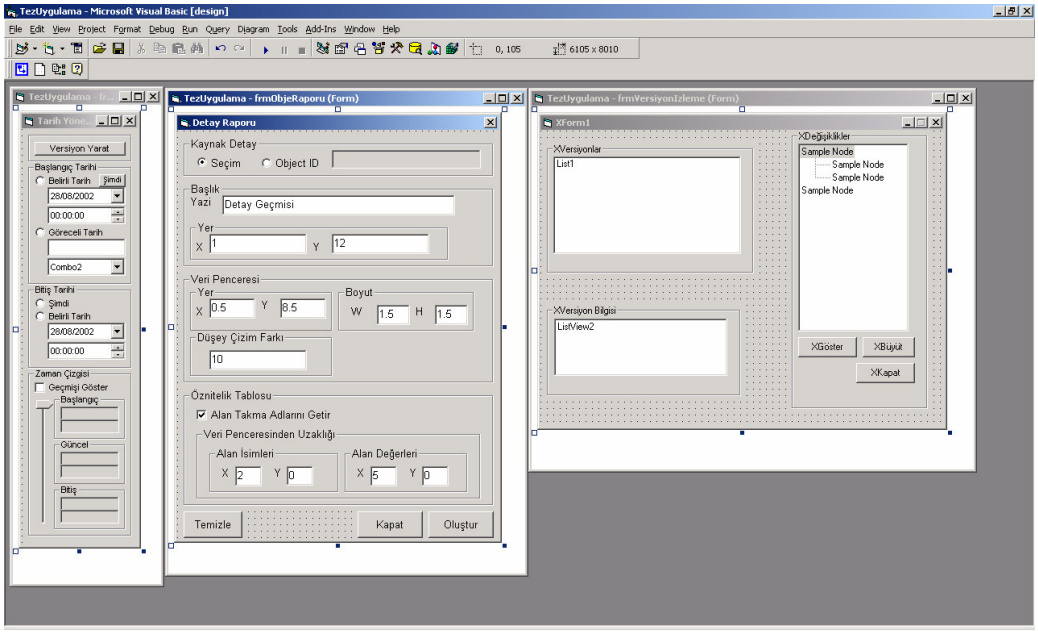

**Şekil 7.5 :** Değişiklikler için çıktı alma işleminin UML aktivite diyagramı

## 7.2. Uygulama Yazılımı Geliştirilmesi

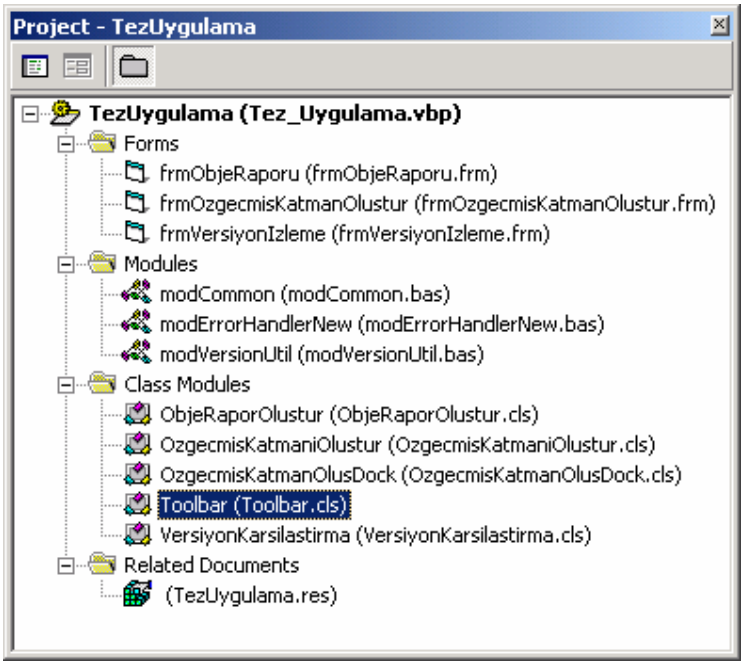
MS Visual Basic ortamının sunmuş olduğu proje seçeneklerinden ActiveX DLL seçeneği kullanılarak TezUygulama.vbp isimli proje oluşturulmuştur. Proje içerisine formlar oluşturulup bu formlar üzerine uygulama için ihtiyaç duyulan denetimler eklenmiştir (Şekil 7.6).

İhtiyaç duyulan sınıf (class) ve modüller (module) için tasarım yapılarak ihtiyaç duyulacak işlemler için gerekli çalışma yapılarak söz konusu bileşenler üretilmiştir ( Şekil 7.7).

Tasarım aşamasından sonra MS Visual Basic standartlarına uygun bir biçimde kaynak kodların üretilmesi safhasına geçilmiştir. Kod yazma işlemleri bittikten sonra gerekli derleme işlemi yapılarak uygulamanın Arcinfo ortamına alınabilmesi için TezUygulama.dll isimli dll (Dynamic Link Library) dosyası oluşturulmuştur. Uygulamanın kaynak kodları ve ilgili dll doyası ekde sunulmuştur (Ek 1).



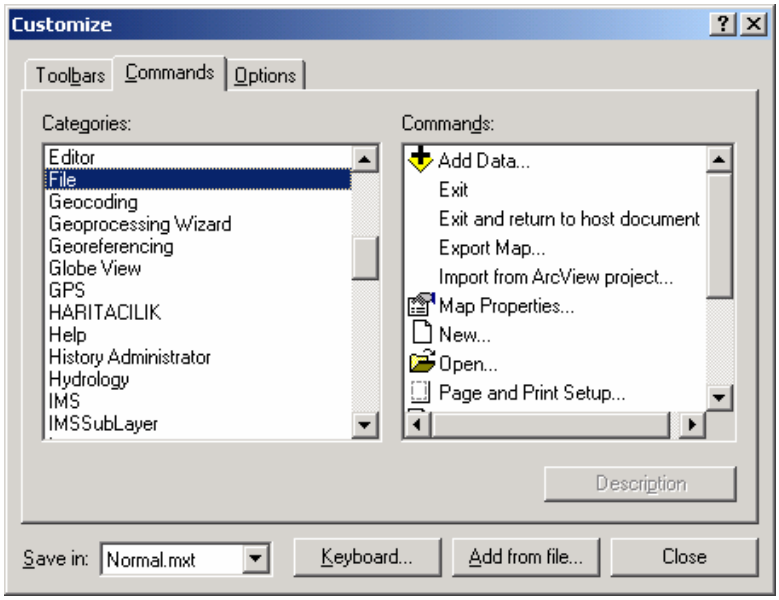

**Şekil 7.6 :** Uygulama yazılımı arayüzleri




**Şekil 7.7 :** Uygulama yazılımı bileşenleri

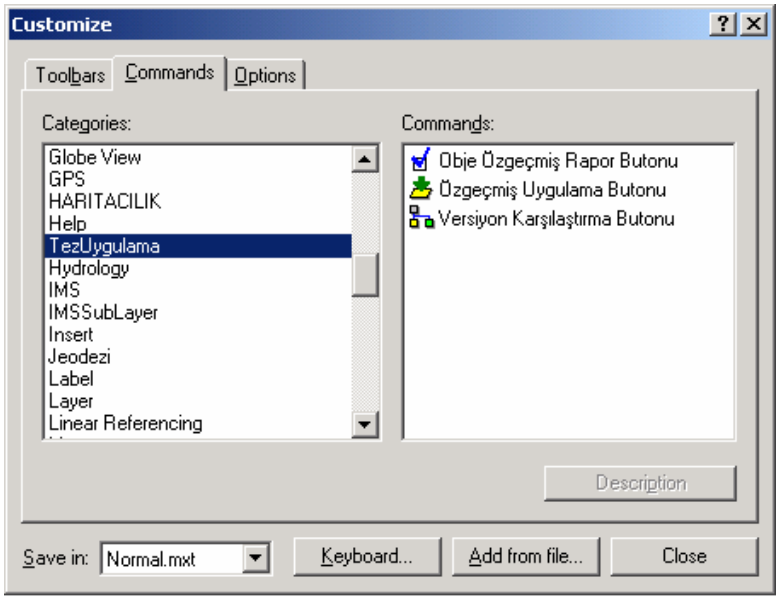
### 7.3. Uygulama Yazılımının Arcinfo Ortamına Aktarılması

Uygulama yazılımın Arcinfo ortamına alınması işlemi için Arcmap arayüzünde bulunan Tools menüsünden Customize seçeneği seçilir (Şekil 7.8). Customize arayüzü üzerinde bulunan Add from file seçeneği tıklanarak TezUygulama.dll isimli dosya ilgili klasörden seçilirek dll dosyasında bulunan arayüzler Arcmap ortamına alınmaktadır.




**Şekil 7.8 :** Uygulama yazılımının arcmap ortamına aktarılması

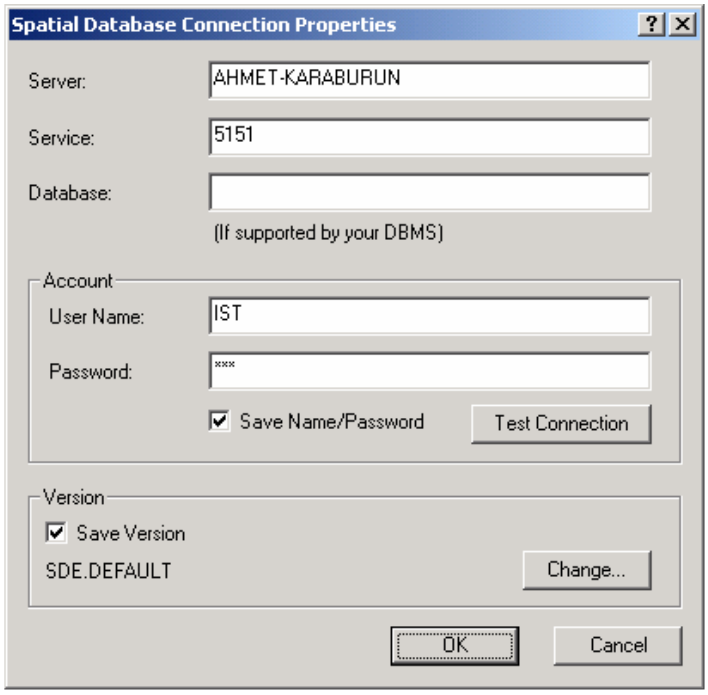
Tezuygulama.dll dosyası eklendiğinde Commands bölümünde Categories bölümüne TezUygulama ve TezUygulama içerisinde yer alan Obje Özgeçmiş Raporu Butonu, Özgeçmiş Uygulama Butonu ve Versiyon Karşılaştırma Butonu bileşenlerini gelecektir (Şekil 7.9).



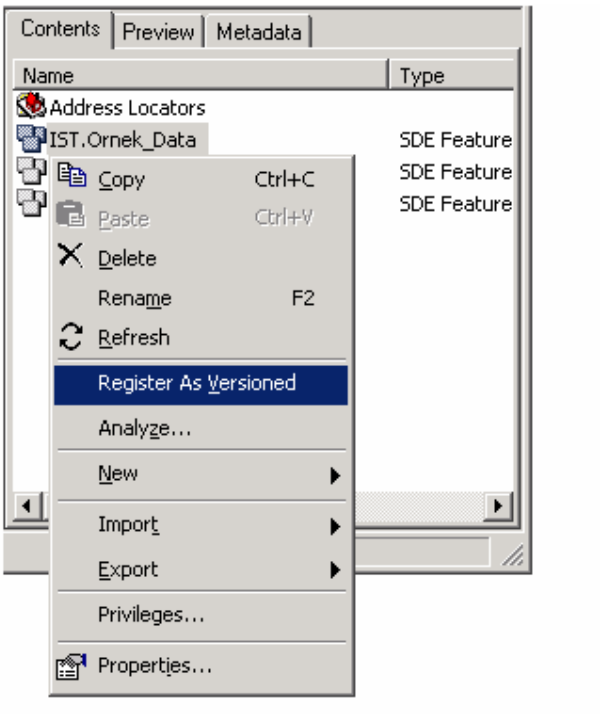

**Şekil 7.9 :** Uygulama yazılımı bileşenleri

### 7.4. Uygulama Yazılımının Kullanılması

Veri tabanlarında depolanmış katmanlar üzerinde çalışma yapabilmek için bu katmanlar üzerinde yapılacak güncellemelerin depolanabilmesi için silme ve ekleme tablolarının (delta tabloları) oluşturulması gerekmektedir. Bu işlemin yapılması için Arccatalog arayüzü kullanılmaktadır. Öncelikle veri tabanına gerekli kullanıcı adı ve şifreleri kullanılarak erişim sağlanır (Şekil 7.10). Arccatalog ortamında veri tabanındaki ilgili katman veya katmanlar seçilerek mouse sağ tuşu ile erişilecek arayüzde bulunan Register As Versioned işlemi yapılır (Şekil 7.11). Bu işlemden sonra ilgili katmanlar için silme ve ekleme tablolarının açılması işlemi yapılacağından artık Arcmap ortamında uygulama yazılımının çalıştırılması mümkün olmaktadır.



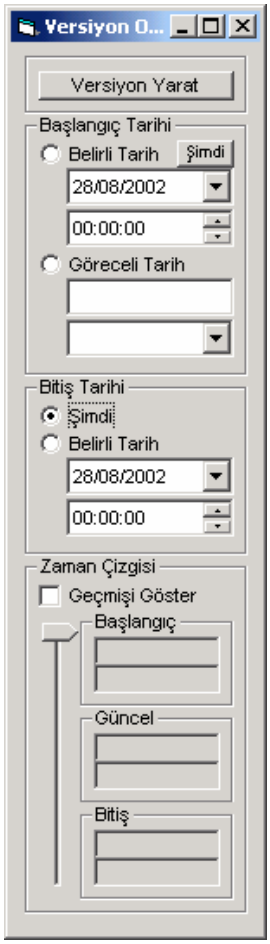

**Şekil 7.10 :** Veri tabanına erişim




**Şekil 7.11 :** Versiyon uygulaması

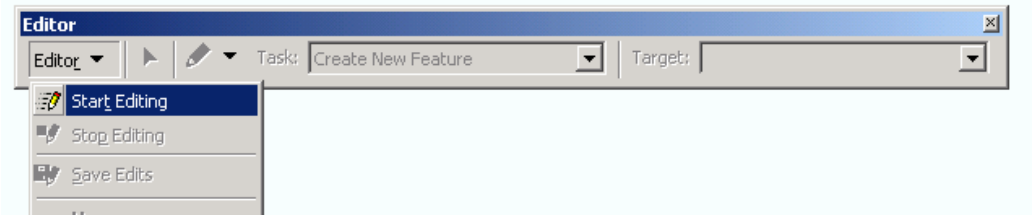
### 7.4.1. Versiyonların Oluşturulması

Arcmap ortamında bulunan özgeçmiş uygulama butonuna basılarak versiyonların oluşturulduğu ve katmanların özgeçmişlerin izlendiği arayüz aktif hale getirilmektedir (Şekil 7.12) .




**Şekil 7.12 :** Versiyon oluşturma ve özgeçmiş izleme arayüzü

Arayüzde bulunan Versiyon Yarat komutu kullanılmadan önce Arcmap arayüzünde bulunan Editor araç çubuğu ile güncelleme işlemlerine başlanması gerekmektedir. (Şekil 7.13)




**Şekil 7.13 :** Arcmap arayüzünde güncelleme işlemlerinin başlatılması

Güncelleme işlemleri başlatıldıktan hemen sonra "Versiyon Oluşturma Ve Özgeçmiş İzleme " arayüzünde bulunan "Versiyon Yarat" butonuna basılarak güncellemelerin kaydedileceği versiyon yaratılır. Arcmap üzerinde katmanlar üzerinde gerekli güncellemeler yapılarak işlemler Editor araç çubuğundan bulunan Save Edits komutu

kullanılarak saklama işlemi yapılır. Bu işlemleri adımlandırmak istersek aşağıdaki gibi bir sıralama oluşacaktır.

1. Katmanlar üzerinde güncelleme işlemine başlamadan önce yapılacak bu güncellemelerin özgeçmişlerinin takip edilmesi için bir versiyona aktarılması gerekmektedir. Bu amaç için **"Versiyon Yarat"** butonu ile güncellemelerin kaydedileceği versiyon yaratılır.

2. Katmanlar üzerinde gerekli güncelleme işlemi yapıldıktan sonra **"Editor"** araç çubuğunda yer alan **"Save Edits"** seçeneği ile güncellemeler kaydedilir. Bu işlem yapılan güncellemeleri yaratılan versiyonda kaydedecektir.

3. Güncellemelerin kaydedilmesinden sonra tekrar **"Versiyon Oluşturma Ve Özgeçmiş İzleme "** arayüzünde bulunan "Versiyon Yarat" butonuna basılarak yeni bir versiyon daha oluşturulur. Artık yapılacak güncellemeler bu versiyonda kaydedilecektir.

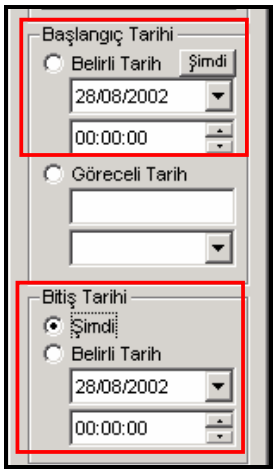
4. Güncellemeler yapılarak işlemler 2. Madde de belirtildiği gibi kaydedilir.

5. İstenildiği kadar versiyon oluşturularak en son aşamada güncellemeler kaydedilerek yine **"Editor"** araç çubuğunda yer alan **"Stop Editing"** seçeneği ile güncelleme ortamında çıkılır.

Burada dikkat edilmesi gereken en önemli nokta **"Save Edits"** işlemi ile yaratılan versiyonda güncelleme işlemleri bittiğinde artık yeni güncellemelerin özgeçmişlerinin takip edilebilmesi için yeni versiyon yaratılması gerektiğidir. Yukarıda belirtilen maddeler takip edilerek sonsuz sayıda versiyon yaratmak ve yapılan güncellemeleri bu versiyonlarda kaydetmek mümkündür.

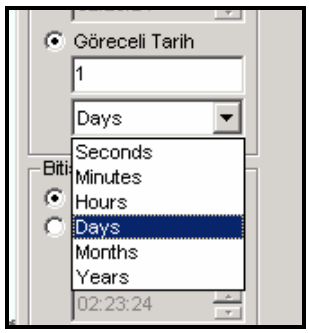
### 7.4.2. Verilerin Özgeçmişlerin İzlenmesi

Güncelleme ve versiyon yaratma işlemleri bitirildikten sonra oluşturulan verilerin özgeçmişlerin izlenmesi mümkün olacaktır. Bu işlem için "Versiyon Oluşturma Ve Özgeçmiş İzleme " arayüzünde bulunan Başlangıç Tarih ve Bitiş Tarih bölümlerinde bulunan alanlara özgeçmişlerin izlenmesi istenen tarihler girilerek izleme yapılacaktır (Şekil 7.14).



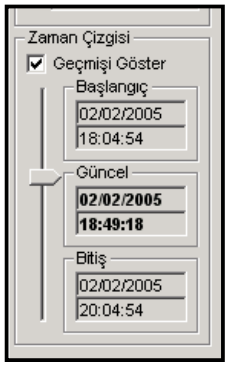

**Şekil 7.14 :** Özgeçmiş izleme bölümleri

Başlangıç zamanı Belirli Tarih alanında belirtileceği gibi bunun yerine yine arayüzde bulunan göreceli tarih seçeneği de kullanılabilir. Göreceli tarih seçeneğinde ise izleme başlangıç zamanını o anki zamandan ne kadar olması gerektiğini belirtmek mümkündür( Örneğin 5 dakika önce, 1 saat önce 1 ay önce vb.) (Şekil 7.15).



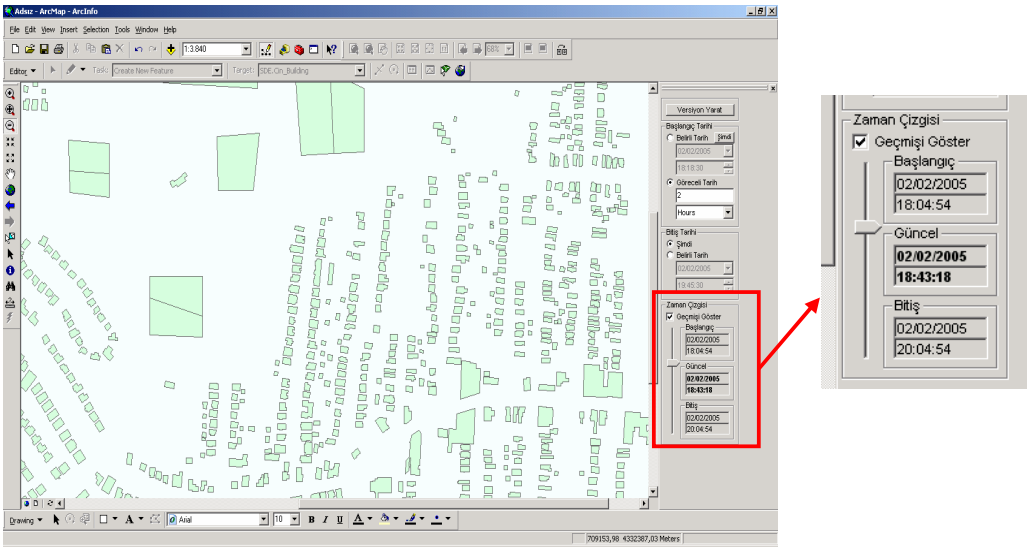

**Şekil 7.15 :** Özgeçmiş izleme zaman aralıkları

İzleme tarih bilgileri arayüze girildikten sonra zaman çizgisi bölümünde Geçmiş göster seçeneği aktif hale getirilerek zaman çizelgesi hareket ettirilerek Arcmap arayüzünde zaman çizgisi hangi tarihi gösteriyorsa o tarihteki verilerin durumunu görmek mümkün olacaktır (Şekil 7.16 ).

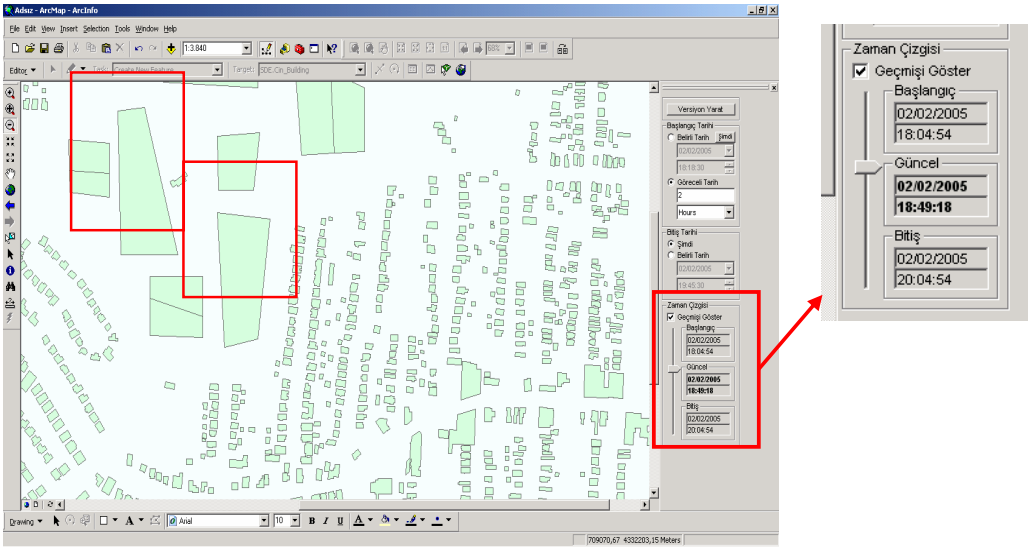



**Şekil 7.16 :** Zaman çizelgesi

Şekil 7.17 ve Şekil 7.18' de özgeçmiş izleme işlemi için bir örnek verilmiştir. Bu örnekte gerekli güncelleme işlemleri yapılarak özgeçmiş takip edilebilmektedir. Şekil 7.17' de arayüzün güncel bölümünde yer alan zamanda herhangi bir değişiklik yok iken zaman çizelgesinin hareket ettirilmesi durumunda ilgili katmana Şekil 7.18'de kırmızı ile işaretlenmiş iki obje haritaya eklenmektedir.



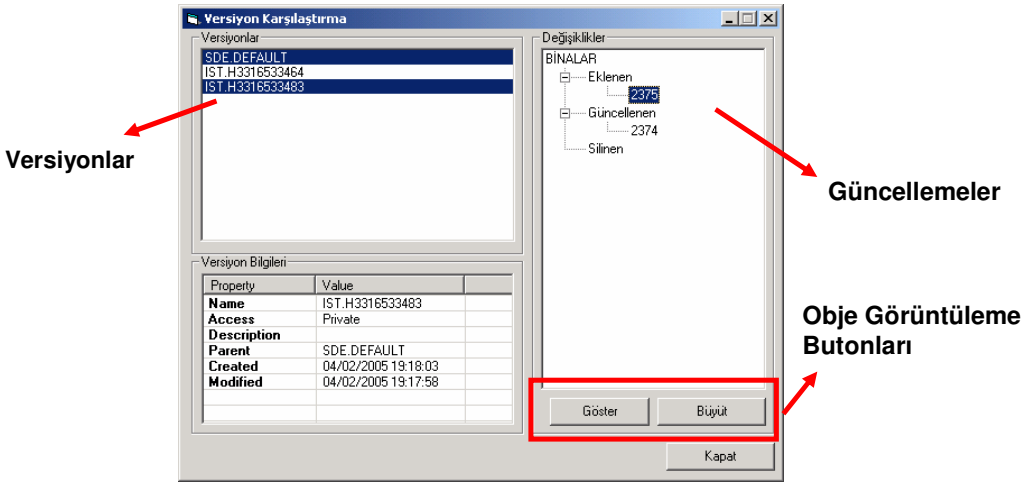

**Şekil 7.17 :** 02/02/2005 tarih ve 18:42:18 saat itibari ile verinin durumu




**Şekil 7.18 :** 02/02/2005 tarih ve 18:49:18 saat itibari ile verinin durumu

### 7.4.3. Versiyonların Karşılaştırılması

Versiyon Karşılaştırma arayüzü ile oluşturulmuş versiyonlar üzerinde ilgili katmanlarda ne tip işlemlerin yapıldığını izlenmesi mümkün olmaktadır. Armap ortamında versiyon karşılaştırma butonuna basılarak arayüze erişilmektedir. (Şekil 7.19).



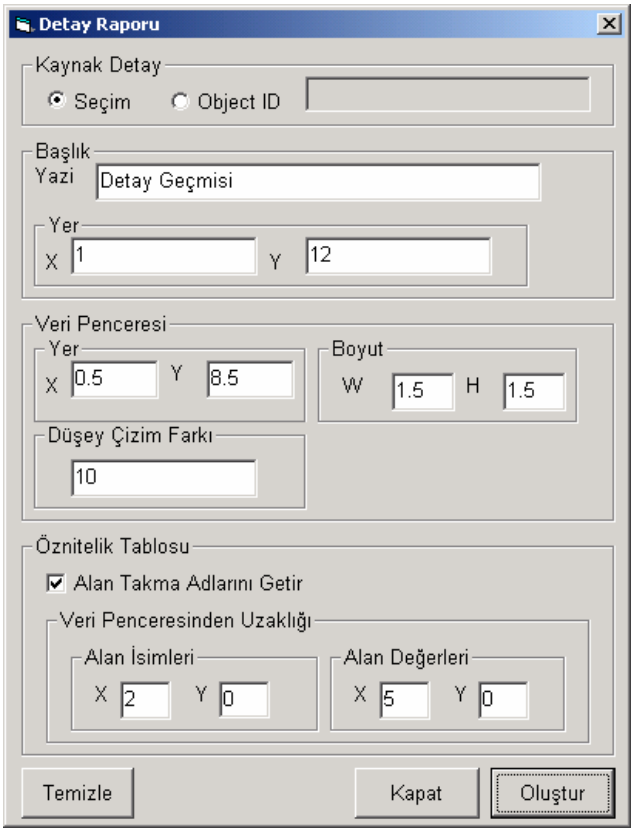

**Şekil 7.19 :** Versiyon karşılaştırma arayüzü

Arayüzün sol tarafında Versiyonlar bölümünde oluşturulan versiyonlar listelenmekte ve sağ tarafında bulunan değişiklikler bölümünde ise her bir versiyonda yapılan güncelleme işlemleri ve bu güncelleme işlemlerinin olduğu verilerin objeId değerleri

gösterilmektedir. Arayüzünde alt bölümünde ise değişiklik bölümünde seçilen verilerin arcmap arayüzünde görüntülenebilmesi için göster ve büyüt butonları yer almaktadır. Sol alt köşe de ise versiyonlar bölümünde seçili olan versiyonlara ait bilgiler listelenmektedir.

### 7.4.4. Verilerin Özgeçmiş Raporlarının Alınması

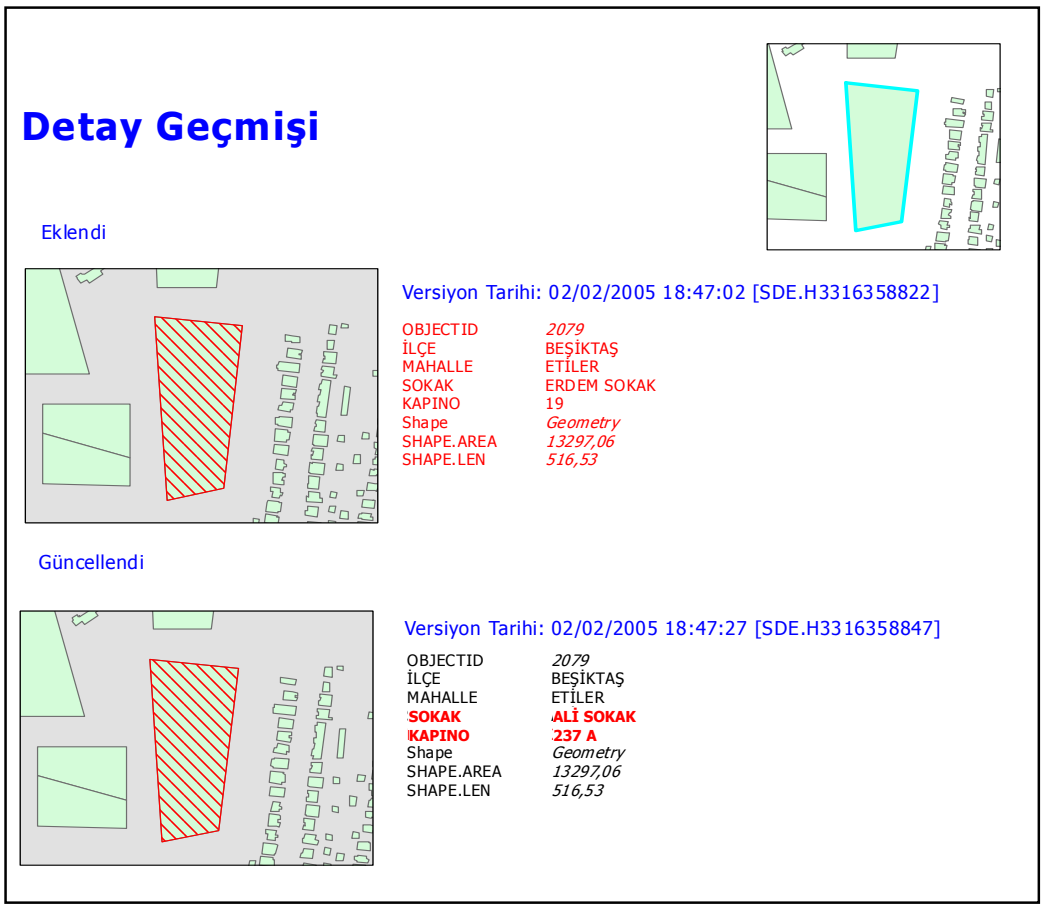
Katmanlar da yer alan herhangi bir objenin ne gibi değişiklikler geçirdiğinin gösterildiği raporları almak için bu arayüz kullanılmaktadır. Bu işlem için armap ortamında bir obje seçilerek çıktı alma bölümü olan Layout View bölümüne geçilir. Obje Özgeçmiş Rapor butonuna basılarak seçili objenin özgeçmiş raporunun alınması için kullanılan Detay Raporu arayüzü aktif hale getirilmektedir (Şekil 7.20).




**Şekil 7.20 :** Detay raporu arayüzü

Detay raporu arayüzünde alınacak raporun başlığı ve konumu kaynak detay ve başlık bölümünde bulunan alanlara gilrilmektedir. Verinin çıktı üzerindeki konumu ise Veri penceresi bölümünde belirlenmektedir. Verilerin öznitelik tablosunda yer alan kon

isimleri için takma adlar var ise bu takma adların kullanılması için Alan Takma Adlarını getir seçeneği aktif hale getirilmektedir. Öznitelik bilgilerinde yer alan alan kolon isimleri ve bunların konumları ise Veri penceresinden uzaklığı bölümüne girilmektedir (Şekil 7.21).

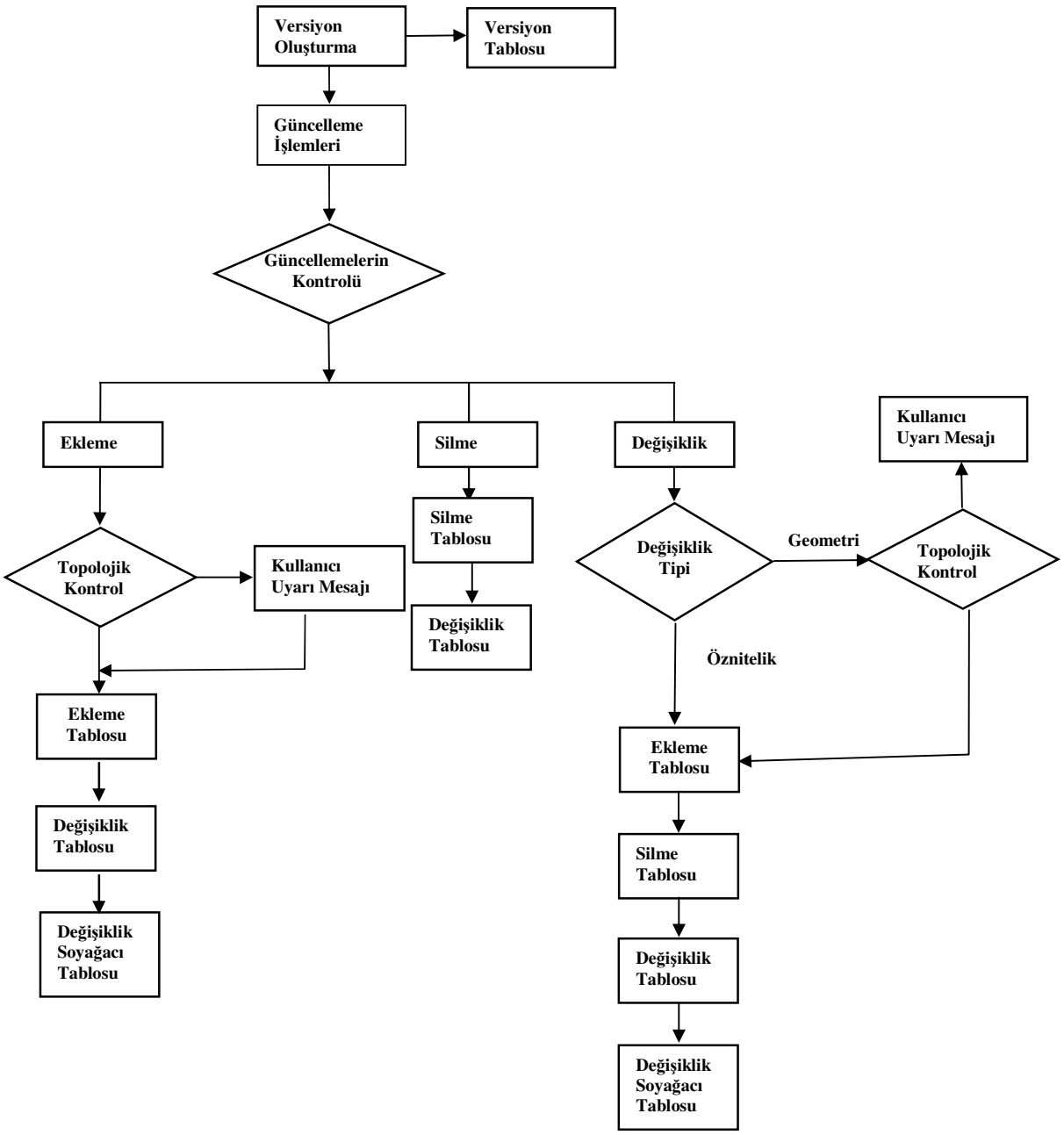



**Şekil 7.21 :** Detay özgeçmiş raporu

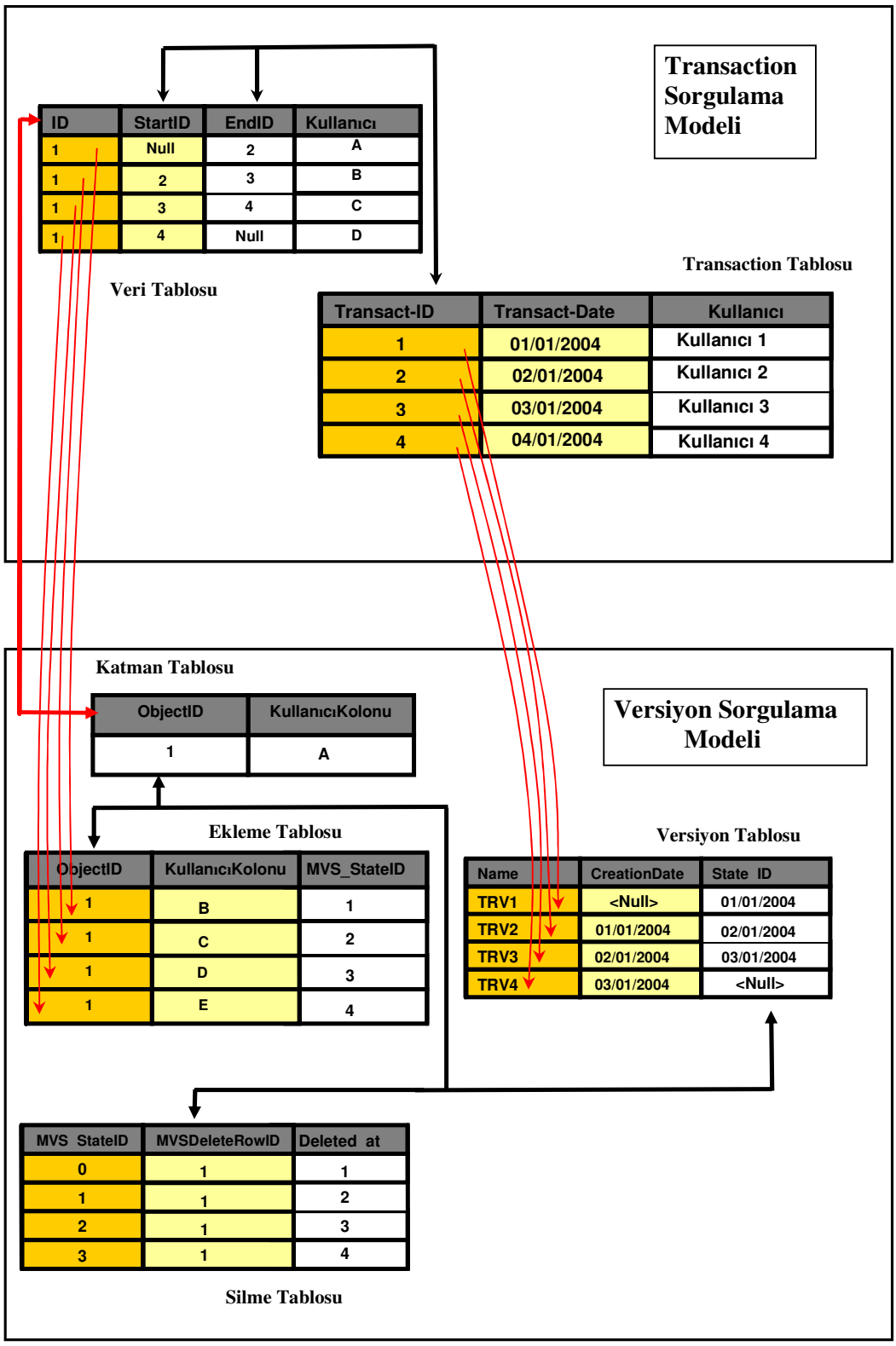
## 7.5. Uygulama Yazılımında Kullanılan Çok Kullanıcılı Ortam ve Özgeçmiş Sorgulama Yönteminin Diğer Yöntemlerle Karşılaştırılması

Veri tabanı yönetim sistemlerinde mekansal verilerin depolanmadığı tablolarda çok kullanıcılı ortam kısa işbitim olarak adlandırılan yöntemler ile sağlanmaktadır. Bu yöntemde veriler güncelleme işlemine tabi olması ile korunmaya alınmakta ve diğer kullanıcıların erişiminin engellenmektedir. Bu nedenle mekansal veriler için güncelleme işlemine alınan verilerin korunmasın gerek kalmayan uzun işbitim yöntemleri kullanılmaktadır.

Kısa iş bitim yöntemi mekansal veriler için kesinlikle uygun bir yöntem olmaması nedeniyle çalışma içerisinde herhangi bir karşılaştırma işlemine tabi tutulamamıştır. Uzun işbitim yönteminden esinlenerek ortaya konan versiyon yönteminin çalışma algoritması Şekil 7.22' de gösterilmektedir. Uzun işbitim yöntemi ve versiyon yönteminin birbirleri ile karşılaştırılması ise Şekil 7.23'da gösterilmektedir.




**Şekil 7.22 :** Versiyon yönteminin çalışma algoritması




**Şekil 7.23 :** Transaction sorgulama modeli ile versiyon modelinin karşılaştırılması

Versiyon modeli ile uzun işbitim (long transaction) tablo modeli karşılaştırıldığında transaction tablosundaki her bir satır versiyon modelinde bir versiyona karşılık geldiği görülmektedir. Versiyon modelinde bir obje katman tablosunda yer almakta ve bu obje ile yapılan silme ve ekleme işlemleri ekleme ve silme tablolarında kaydedilmektedir. Bu tablolar daha sonra versiyonlar ile ilişkilendirilmektedir. Böylece her hangi bir zaman dilimindeki versiyonun durumuna coğrafi bilgi sistemi uygulama yazılımları veya SQL sorgulama dili ile oluşturulacak uygulamalar ile erişmek mümkün olmaktadır.

Uzun işbitim yöntemlerinde ise yöntemlerinde verilerin özgeçmişlerinin izlenebilmesi için her katmanın tablosunda verinin oluşturulduğu başlangıç tarihi ile verinin silindiği silinme tarihi olan bitiş tarihi isimli sütünlar oluşturulmaktadır. Bu sütünlarda verilerin güncellendiği zaman bilgileri kaydedildiğinden yapılan sorgulamalar ile verilerin ögeçmileri takip edilmektedir. Silinen ve güncellenen veriler ana veriler ile aynı katman tablosunda depolandığında her güncelleme işlemi tablonunun hacmini artıracağından belli bir süre sonunda verinin kullanıcılara sunumunda performans problemleri ortaya çıkmaktadır.

Yapılan güncellemelerin kimler tarafından yapıldığı bilgisi uzun işbitim numaraları ile takip edilmektedir. Bu numaralar katman tablosunda depolandığından yapılan güncellemelerin sayısının artması benzer sorun kimlerin neyi güncellediğinin belirlenmesi konusunda da yaşanmaktadır.

Bu çalışmada uzun işbitim yönteminin belirtilen özellklerinden dolayı özgün uygulama yazılımında uzun işbitim yöntemi yerine versiyon yöntemi kullanılmıştır. Böylece aşağıda belirtilmiş olan kolaylıkların sağlandığı gözlenmiştir;

- Versiyon yönteminde versiyonların oluşturulma işlemi kullanıcılar tarafından özelleştirilebilmesi imkanı sayesinde uygulamanın herhangi bir aşamasında versiyon oluşturulabilmiştir.

- Ana verilerin farklı versiyonlarının oluşturulması yapılacak güncelleme işlemlerinin iş akış şemalarının oluşturulmasına imkan sağlamıştır.

- Herhangi bir katmanda silinen, değiştirilen ve eklenen verilerin tamamı farklı tablolarda depolandığından veri tabanına gönderilmiş olan sorgulamalar uzun

işbitim yöntemindeki veri tablosuna oranla yerine daha az sayıda verinin depolanmış olduğu ekleme ve silme tabloları kullanılmış böylelikle sorgu sonucu oluşan veriler daha kısa sürede görüntülenmiştir. Böylece verilerin yönetimi daha da kolaylaşmıştır.

- Versiyon yöntemi nesne ilişksel bir model olması nedeniyle geliştirilen uygulama yazılımları ile etkileşimi kolay olması, görüntüleme arayüzleri üzerinde obje bazlı özgeçmiş izlenmesine imkanı sağlanmıştır.

- Nesne ilişkisel bir model olması sayesinde herhangi bir versiyonda izlenen özgeçmişlerinin geometrik ve öznitelik bilgilerinin çıktılarının alınması sağlanmıştır.

# 8. SONUÇ VE ÖNERİLER

Coğrafi bilgi sistemlerinin gelişmesi ile sistem içerisinde kullanılan verilerin hacimlerinin artması ve bu verilerin bilgisayarlarda dosya-bazlı olarak saklanması verilerin üretimi ve yönetimi noktasında kullanıcılara ciddi sorunlar yaşatmaya başlamıştır.

Coğrafi bilgi sistemi kullanımının hızla artış göstermesi veri üretimi, depolanması ve analiz edilmesi aşamalarında veri tabanlarının kullanılmasını gündeme getirmiştir. Özellikle coğrafi bilgi sistemlerinin ihtiyaç duyduğu mekansal verilerin uygun standartlarda iş bölümü ve koordinasyon anlayışı içinde üretilmesi ve böylelikle verilerde tekrarlanmayı, işgücü ve kaynak kaybının engellemesi için veri tabanları veri depolama aracı olarak kullanılmaya başlanmıştır.

Veri tabanlarında depolanmış verilerin yönetiminin uzmanlık isteyen bir konu olması coğrafi bilgi sistemlerin de veri tabanlarının kullanım hızını yavaşlatmış ve bu konuda kullanıcılara veri tabanlarında kolaylıkla çalışabilmelerini sağlayacak yazılımların geliştirilmesi fikrini ortaya çıkarmıştır. Bu amaç için mekansal veri sunucuları olarak adlandırılan ve mekansal verileri veri tabanlarında çok fazla uzmanlık gerektirmeyen işlemlerin sonucunda depolanmasını ve yönetimini sağlayan yazılımlar üretilmiştir.

Mekansal veri sunucu yazılımlarının kullanımı coğrafi bilgi sistemlerinin gelişimi açısından büyük bir adım olmuştur. Veri tabanı yazılımlarının üreticileri kendi yazılımları için mekansal veri sunucuları geliştirirken bazı coğrafi bilgi sistemi yazılımı üreticileri ise en yaygın kullanılan veri tabanları üzerinde çalışabilecek mekansal veri sunucu yazılımlarını piyasaya sürmüştür.

Bu yazılımların kullanılmaya başlanması ile CBS için uzun vadeli modellemelerin yapılabilmesi ve veri güncellenmesi noktasında bazı sınırlamaların ortadan kaldırılmasına zemin hazırlamıştır. Bu sınırlamalar mekansal verilerin aynı anda

birden fazla kullanıcı tarafından güncellenebilmesi ve güncellemeler esnasında verilerin özgeçmişlerinin kaydedilmesi şeklinde olmaktadır.

Mekansal veri sunucu yazılımları ile güncel verinin veri tabanında tek bir tabloda tutulması, her bir kullanıcıya bu tabloların birer kopyaları anlamına gelen versiyonlarının gönderilmesi işlemi çok kullanıcılı güncelleme işlemlerine imkan sağlamıştır. Versiyon özelliği ile güncelleme işlemleri sonunda oluşan verilerin bir kontrol mekanizmasından geçerek en güncel halinin saklandığı tabloya kaydedilmektedir.

Verilerin versiyonlarının üretilmesinin sağlamış olduğu en önemli özellik güncellenen verilerin güncellemeden önceki durumlarının ilgili tablolarda saklanması ve veri tabanında silinmemesidir. Bu özellik uygun bir şekilde modellenebilirse güncelleme işlemlerine tabi olan mekansal verilerin özgeçmişlerinin takip edilmesine olanak sağlayabilmektedir.

Coğrafi bilgi sistemlerinde kaydedilen tüm bu gelişmeler ile kullanıcılar kurmuş oldukları sistemler içerisindeki verinin üretimi, yönetimi ve analizi açısından uzun vadeli planları ve buna bağlı olarak iş akışları oluşturabilir.

Yaşanan bu gelişmelere rağmen maalesef ülkemizde geliştirilmiş olan coğrafi bilgi sistemi uygulamalarında yukarıda bahsedilen özelliklere sahip örnekleri görmek pek mümkün olmamaktadır. Büyükşehir belediyelerinin ve bakanlıkların kurmuş oldukları coğrafi bilgi sistemlerinde veri tabanları sadece veri depolama aracı olarak kullanılmakta diğer kurumlar tarafından kurulan sistemlerde ise veriler kullanıcıların bilgisayarlarında dosya bazlı olarak saklanmaktadır.İlişkisel veri tabanı kullanılan coğrafi bilgi sistemlerinde ise çok kullanıcılı ortam için herhangi bir çalışma olmayıp buna bağlı olarak da verilerin özgeçmişleri izlenmemektedir. Verilerin özgeçmişleri güncelleme tarihindeki durumları ile dosya bazlı olarak saklanmaktadır.

Bu çalışma içerisinde coğrafi bilgi sistemlerinin optimum bir şekilde kullanılabilmesi için veri tabanlarında mekansal verilerin depolanma yöntemleri incelenmiş örnek alınan bir mekansal veri sunucu yazılımı ile de bu verilerin yönetimi analiz edilmiştir. Özellikle versiyon işlemi ayrıntılı bir şekilde incelenmiş, örnekleri ile ortaya konmuştur. Çalışmanın sonunda çok kullanıcılı güncelleme ve özgeçmiş takibi işlemleri için bir uygulama yazılımı geliştirilmiştir.

Uzun işbitim yönteminin çalışma esasları ortaya konarak versiyon yöntemi ile karşılaştırılmıştır. Karşılaştırma sonucunda versiyon yönteminde yapılan her güncelleme ilgili tablolarda depolandığından güncelleme sonucu oluşan veri hacminin uzun işbitim yöntemine nazaran daha az olduğu gözlenmiştir.

Versiyon yöntemi nesne ilişkisel yöntem yöntem olması yapılan güncellemelerin ile ilgili raporların alınmasını sağlamıştır. Bu raporlar harita arayüzü üzerinde bilgi kutuları ile güncellemelerin gösterilmesi ve bu güncellemelerin çıktılarının alınması şeklinde olmaktadır.

Çalışmada sunulan özgün uygulamada sağlamış olduğu kolaylıklar nedeni ile versiyon yöntemi kullanılmış ve bu yöntem ile çok kullanıcılı ortam ve özgeçmiş sorgulamalar için gerekli algortimalar ortaya konmuştur. Söz konusu uygulama çalışma içerisinde sunulan algoritmalar kullanılmak suretiyle farklı veri tabanı yönetim sistemleri üzerinde rahatlıkla uygulanabilir.

Mekansal verilerin bu sistemler üzerinde depolandıktan sonra versiyonlara verilerin kopyalarının değil mantıksal kopyalarının sunulması çok önemlidir. Aksi takdirde verilerin kopyaları oluşturulursa veri tabanındaki hacim artacağından çalışma hızı ve performansı düşecektir.

İdeal anlamdaki bir coğrafi bilgi sistemi içerisinde verilerin çok kullanıcılı ortamda yönetilmesi ve özgeçmişlerinin izlenmesi sistemin vazgeçilmez unsurlarından olmaktadır. Aksi takdirde coğrafi bilgi sistemlerinden beklenen verilerin en güncel haline ulaşma, verilerin zaman içerisindeki değişimlerini izleme noktasında sorunlar oluşacak ve bunun sonucunda ise kaynak ve vakit kaybına neden olacaktır.

## KAYNAKLAR


**Abraham, T. and Roddick, F. J.,** 1999. *Survey of Spatio-Temporal Databases Geoinformatica,* Vol.3, No. 1, pp. 61-99.

**Adler D W.,** 2001. IBM DB2 Spatial Extender, Spatial Data Within the RDBMS, *In Proc. of VLDB*, pp. 687-690.

**Atkinson M., Bancilhon F., Witt D., Dittrich K., Maier S. and Zdonik S.,** 1989 The Object-Oriented Database System Manifesto, *Proceedings of the 1st Intl. Conference on Deductive and Object-Oriented Databases (DOOD'89)*, Dec. 1989, pp. 40-57, Kyoto, Japan.

**Banger G.,** 1998. Başbakanlık Yönetim Bilişim Sistemi, Başbakanlık Kamu-Net Üst Kurulu, Ankara.

**Begg.C, and Connolly T.,** 2002. Database Systems, A Practical Guide to Design, Implementation and Management, 3.Edition Addison Wesley, Boston, USA.

**Bernstein P A. , V. Hadzilacos V. and Goodman N.,** 1987 Concurrency Control and Recovery in Database Systems, Addison Wesley, Boston, USA.

**Browne G.J.,StoneBrker M.,and Moore D.,** 1999. Object-relational DBMSs tracking the next great wave, 2.Edition, .Morgan Kaufmann Publishers San Fransisco, USA.

**Carson G.,** 2000. Spatial Standardization, *Computer Graphics*, Volume 34, Number 3, A publication of ACM SIGGRAPH, New York, USA.

**Cavero J.M., Marcos E.and Vela B.,** 2001. Extending UML for Objects,Relational Database Design, *In Proceesings of the UML 2001,* The Unified Modelling Language, Modelling Languages, Concepts and Tools, Springer-Verlag Heidelberg, 225-239, Toronto, Canada.

**Cellary W.and Jomier G.,** 1992. Consistency of Versions in Object-Oriented Databases, Building An Object-Oriented Database System, Morgan Kaufmann, San Fransisco, California, USA.

**Chawla S. and Shekhar S.,** 2003. Spatial Databases, A Tour, Pearson Education Inc, New Jersey, USA.

**Chawla S., Fetterer A., Liu X., Lu.C., Shekhar S. and Ravada S.,** 1999. Spatial Databses, Accomplishments and Research Needs. *IEEE Transactions on Knowledge and Data Engineering* ,Volume 11, No 1, pp. 45- 55.

**Chrisman, N.,** 1997. Exploring Geographic Information Systems, John Wiley & Sons Inc. New York, USA.

**Easterfield M., Newell R. And Theriault D.,** 1990. Version Management in GIS, Application and Techniques, *First European Conference on Geographical Information Systems*, pp 288-297 Amsterdam, The Netherlands.

**Egenhofer M.,** 1992. Why not SQL, *International Journal of Geographical Information Systems*, Volume 6, No 2, pp. 71-86, 1992 Taylor and Francis, UK.

**Elmasri R. and Navathe S B.,** 1989. Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company, California, USA.

**Elmasri R. and Navathe S B.,** 2000. Fundamentals of Database Systems, 3.nd Edition, Addison-Wesley, Massachusetts, USA.

**ESRI, Inc.,** 1999. Arcsde Developer Help, California, USA.

**ESRI, Inc.,** 2002. Modeling and Using History in ArcGIS, California, USA.

**ESRI, Inc.,** 2004. Versioning, An Esri Technical Paper, California, USA.

**Goodchild M.F. Longley P.A, Maguire D.J. and Rhind D.W.,** 2001. Geograhic Information Systems and Science. John Wiley & Sons, Ltd. Chishester UK.

**Gray, J. and Reuter, A.** 1993. Transaction Processing, Concepts and Techniques, Morgan Kaufmann Publishers, San Mateo, California, USA.

**Grimes, R., Templeman, J., Stockton, A., Watson, K. and Reilly, G.,** 1999. Beginning ATL COM Pogramming, Wrox Pres, UK.

**Guptill, C. S., and Morrison, J. L.,** 1995. Elements of Spatial Data Quality Temporal Information. Pergamon Pres, Oxford, UK.

**Gutig R.,** 1994. An Introduction To Spatial Databases Systems. *Journal On Very Large Database* Springer-Verlag, New York Volume 3 Issue 4. pp.357- 399.

**Healey, R.G.,** 1991. Database Management Systems In Geographical Information Systems, Principles and Applications, Longman, Newyork, USA.

**Herlihy M.,** 1990 Apologizing Versus Asking Permission, Optimistic Concurrency Control for Abstract Data Types, *Association for Computing Machinery Transactions on Database Systems*, Vol. 15, No. 1, March 1990, pp. 96-124.

**Hoffer J.A., Prescott M.B. and McFadden F.R.,** 2004. Modern Database Management. Pearson Prentice Hall, New Jersey, USA.

**ISO-1998.** Technical Committee 211, Working Group 1, *International Organization for Standardization*, CD 15046- 1.2, Geographic Information, Part 1, Reference Model, ISO/TC 211 N 623.

**ISO-2004.** International Electrotechnical Commission, 19501, International Organization for Standardization Information technology, Open Distributed Processing, Unified Modeling Language (UML) Version 1.4.2

**Kim W and Lochovsky F.H.,** 1992. Object Oriented Concepts, Databases And Applications. Acedemic Pres, NewYork, USA.

**Langran, G.,** 1992. Time in Geographic Information Systems, Taylor and Francis Ltd. UK.

**Laurini R. and Thompson D.,** 1992. Fundementals of Spatial Information Systems, Acedemic Pres, San Diego CA, USA.

**McCoy, J.,** 2003. Geoprocessing in ArcGIS, Environmental System Research Institute Inc. Press, California , USA.

**Medeiros C.B. and Jomier G.,** 1994. Using Version in GIS*, Proc. of Database and Expert Systems Applications Conference (DEXA'94),* Athens, Greece.

**Medeiros C.B., and Pires F.,** 1994. Databases for GIS. *Association for Computing Machinery ACM Sigmond Record*, Volume 23 No 1, pp. 107-115.

**Melton J.,** 1990. SQL2, The SEQUEL An Emerging Standard, *Database Programming and Design*, November, pp 24-32.

**Minami M., Sakala M. And Wrightsel J.,** 1999. Using ArcMap, Environmental System Research Institute Inc. Press, Redlands , California, USA.

**Montgomery L.,** 1995. Temporal Geographic Information Systems Technology and Requirements, Where we are Today, *MS Thesis*, The Ohio State University, Columbus, USA.

**Muller, R. J.,** 1999, Database Design for Smarties, Using UML for Data Modeling. Academic Pres, California, USA

**OGIS-1999.** Open GIS Simple Features Specification for SQL , Open GIS Consortium, USA.

**Oracle, Inc.,** 2002. Oracle Spatial User's Guide and Reference, Release 9.2, Oracle Corporation USA.

**Ott, T. and Swiaczny, F.,** 2001. Time-Integrative Geographic Information Systems, Management And Analysis Of Spatio-Temporal Data, Springer, Berlin, Germany.

**Östensen O.,** 2001. The Expending Agenda of Geographic Information Standarts, ISO Bulletin, July , pp. 16-21.

**Raafat, H., Yang, Z. and Gauthier,D.,** 1994. Relational Spatial Topologies for Historical Geographical Information, *International Journal of Geographical Information Systems,* Vol.8, No.2, pp.163-173.

**Ramakrishnan, R. and Gehrke. J.,** 2000. Database Management Systems, McGraw Hill Higher Educcation Books,

**Rigaux P., Scholl M., and Voisard A.,** 2002. Spatial Databases with application to GIS. Morgan Kaufmann Publishers, San Fransisco, USA.

**Rigaux P., Scholl M.and Voisard A.,** 2002. Spatial Databases With Application to GIS Academic Press, San Diego, USA.

**Samet H.,** 1989. The Design and Analysis of spatial Data Structures. Addison-Wesley, Boston, USA.

**Tsichritzis D C. and Lochovsky F H.,** 1982. Data Models Prentice Hall, Inc., USA.

**Vienneau A, Bailey J., Harlow M, and Woo S.,** 1999, Using ArcCatalog, Environmental System Research Institute Inc. Press, San Diego, USA.

**West, R.,** 2001. Understanding ArcSDE, Environmental System Research Institute Inc. Pres, Redlands, USA.

**Woo S., Perencsik A., Booth B., Crosier S., Clark J.and MacDonald A.,** 1999. Building a Geodatabase, Environmental System Research Institute Inc. Press, Redlands, California, USA.

**Worboys, M.,** 1994. A Unified Model for Spatial and Temporal Information*, The Computer Journal,* Vol.37, No.1.

**Yomralıoğlu, T**., 2000.Coğrafi Bilgi Sistemleri, Temel Kavramlar ve Uygulamalar, Trabzon.

**Yuan, M.,** 1996. Temporal GIS and Spatio-Temporal Modeling, *Proceedings of the 3rd International Conference/Workshop on Integrating GIS and Environmental Modeling,* January 21-25, Santa Fe, New Mexico, USA.

**Zeiler, M**., 1999. Modelling Our World, Environmental System Research Institute Inc. Press, Redlands ,California, USA.

**Zeiler, M.,** 2002. Exploring ArcObjects, Environmental System Research Institute Inc. Press, Redlands, California, USA.

**EKLER**

**EK 1 :** Uygulama Programı

# ÖZGEÇMİŞ

1973 yılında Malatya'da doğdu. İlköğrenimi Ziya Gökalp İlkokulunda, ortaöğrenimini Kubilay Ortaokulunda ve lise öğrenimini Tapu ve Kadastro Meslek Lisesinde yaptı. 1990 yılında İTÜ İnşaat Fakültesi Jeodezi ve Fotogrametri Mühendisliği Bölümü'nde başladığı lisans eğitimini 1994 yılında tamamlayarak yüksek lisans eğitimine başladı. 1994-1998 yıllları arasında İTÜ Fen Bilimleri Enstitüsü Jeodezi ve Fotogrametri Mühendisliği Jeodezi ve Fotogrametri Anabilim Dalı'nda yüksek lisans eğitimini tamamladıktan sonra Yüksek Mühendis ünvanını aldı. Halen özel bir firmada proje yürütücüsü olarak çalışmaktadır.