YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VİSUAL C++'TA KURALLARA DAYALI ORACLE VERİ TABANLI SORGULAMALAR

Elektronik ve Haberleşme Müh. Bora GÖKÇE

F.B.E. Matematik Mühendisliği Anabilim Dalı Matematik Mühendisliği Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Prof. Dr. A. Bened

Tez Danışmanı : Yrd. Doç. Dr. Ayla ŞAYLI

Y. Doç. Dr. İbrahim Emiroğlu

İncelemiştir.
Y. Doç. Dr. Ayla ŞAYLI

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

@	Karşılaştırma operatör göstergesi
-->	Kurallarda sol ve sağ tarafların ayrıldığı gösterge

KISALTMA LİSTESİ

ADO	Aktif Veri Nesnesi (Activex Data Objects)
ANSI	Uluslararası Amerikan Standartları Enstitüsü (American National Standarts Institute)
DDL	Veri ve Veri Modeli Tanımlama Dili (Data Defination Language)
DML	Veri İşleme Dili (Data Manipulation Language)
ISO	Uluslararası Standart Organizasyonu (International Standart Organization)
KDD	Veri Bilgi Keşfi (Knowledge Discover in Databases)
ODBC	Açık Veritabanı Bağlantısı (Open Database Connectivity)
QBE	Örnekle Sorgu Dili (Query By Example)
QUEL	İngilizce Tabanlı Sorgulama Dili (Query English Language)
RDBMS	İlişkisel Veri Tabanı Yönetim Sistemi (Relational Database Management Systems)
SQL	Yapısal Sorgu Dili (Structured Query Language)
SQO	Anlamsal Sorgu İyileştirmesi (Semantic Query Optimization)
VBF	Veri Bilgi Keşfi
VTYS	Veri Tabanı Yönetim Sistemi

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bilgisayarlar her geçen gün hayatımıza daha çok girmekte ve her alanda kullanılmaktadırlar. Böylece elimizdeki verileri saklayabileceğimiz, istediğimizde bu verilere kolayca ulaşabileceğimiz, rahatça arama yapabileceğimiz ve paylaşabileceğimiz bir araç olmuştur. Depolanan verilere, kolaylıkla ulaşabilme, güncelleme, değiştirme, ekleme yapabilmek için veritabanı sistemleri geliştirilmiştir.

Bu tezde, genel olarak veritabanı kullanımı ve özellikleri, öğrenme metotları, kural oluşturma, veritabanı üzerinde çalıştırılan DML ifadelerinin performansını artırmak için öğrenilen kuralların DML'lerde kullanılması, bu kuralların kullanılması ile ortaya çıkan sonuçlar ve bu sonuçların değişik koşullarda karşılaştırılmaları anlatılmaktadır.

Bu tezin hazırlanmasında yardımlarını ve desteğini esirgemeyen değerli hocamız Sayın Yrd. Doç. Dr. Ayla Şaylı'ya çok teşekkür ederim.

ÖZET

Bugünün dünyasında, yaşam kalitemizi artırmak amacıyla bilgisayar kullanmak eğitim, bankacılık, sağlık, mühendislik, idari işler gibi bir çok sahada çok önemli bir rol oynar. Bilgisayarlar aynı işi otomatik olarak ve daha verimli yapacağından insan kaynaklı hatalar en aza indirgenir. Bilgisayarlar teşhis numaraları, banka hesapları ve barkodlar gibi pek çok önemli bilgileri saklarlar ve bu bilgiler birçok kez tekrar tekrar kullanılabilirler.

Veritabanı sistemi, temel olarak bilgisayar tabanlı veri saklama sistemidir. Bu sistemin amacı veri saklamak ve gerektiğinde veriye erişmektir. Bir veritabanı yönetim sistemi kullanıcılarının veriye erişebilmesi için bir sorgulama diline ihtiyaç duyar. Yapısal sorgulama dili olan SQL çoğu veritabanı sisteminde kullanılan en yaygın ve standartlaşmış sorgulama dilidir. SQL İngilizce tabanlı bir dildir. Create, select, update, delete, insert into, where, order..... gibi komutları vardır. Kayıtlar kümesinin en genel şekli tablodur. SQL çeşitli işler için komutlar sağlar. Bunlar veriyi sorgulama, güncelleme, kayıt eklenmesi, silinmesi, veri tabanı nesnelerinin düzeltilmesi ve silinmesi, veritabanı ve veritabanı nesnelerinin erişiminin kontrol edilmesidir.

Bu tezde, en son sorgu iyileştirme yaklaşımı olan Anlamsal Sorgu İyileştirme (SQO) yaklaşımına odaklanılmıştır. SQO, verilen bir sorguya alternatif sorgular inşa etmek için anlamsal bilgiler (kurallar) kullanır ve sonra alternatifler içinde optimum sorguyu, hesaplama maliyetlerine göre seçer. Daha da fazlası, bu alternatif sorgular anlamsal olarak aynı fakat sözdizimsel olarak farklıdırlar. Önemli olan alternatif sorguların hepsinin aynı cevap kümesine sahip olmasıdır.

Bir başka önemli husus ise bu kuralların ilk sorgudan itibaren türetilmesidir. Türetilen kurallar kullanılarak yeni kuralların elde edilmesi sağlanabilir. Kuralların kullanımında maksimum performans artışı, sorgunun veritabanına giriş yapılmadan sonucunun verildiği durumlarda sağlanır. Diğer performans kazanımı ise kurallar sayesinde sorgunun daha hızlı arama metotları ile yapılmasıdır. Örneğin birincil anahtar veya dizin içeren kuralların kullanılmasında performans artışı ispatlanmıştır. SQO yaklaşımı çeşitli bileşenlerden oluşur: Sorgunun ifade edilmesi, otomatik olarak sorguya dayalı öğrenme, sorgunun iyileştirilmesi ve öğrenilen bilgilerin, kısaca kuralların güncellenmesidir.

Tez içerisinde ise kuralların sorgulama performans artışının, DML'lerde de sağlanmasının araştırılması ve araştırma sonuncunda bu artışın ispatlanması yapılmaktadır.

Anahtar Kelimeler : Veritabanı, SQL, DDL, DML, SQO, kurallar, ADO, Oracle.

ABSTRACT

In today's world, using computers play as a serious role in almost every environment such as education, banking, health care, engineering, government, etc. in order to improve the quality of our life. This reduces human errors since computers can be used for the same work automatically and effectively. Computers keep vital information such as student identification numbers, bank accounts and barcodes... etc. so that this information can be used over and over.

A database system is basically a computerized record keeping system. The overall purpose of this system is to maintain information and to make that information available on demand. A Relational Database Management System (RDBMS) requires a query language to enable users to access and to store data. Structured query language (SQL) is the language used by most relational database systems as a standard query language. SQL is an English based language. It uses words such as create, select, update, delete, insert into, where, order....est. as part of its command set. SQL processes sets of records rather than a single record at a time. The most common form of a set of records is a table. SQL records commands for a variety of tasks including; querying data, inserting, updating, deleting rows in a table, creating, modifying and deleting database objects, controlling access to the database and database objects.

In this thesis, we focus on Semantic Query Optimization (SQO) which is the most recent query optimization approach. SQO uses semantic knowledge (rules) to construct alternative queries for a given query, and then select an optimum query between the alternatives according to their estimated costs. Moreover, these alternative queries can be semantically the same but syntactically different. Here the important issue is, all alternative queries must have the same response set.

Another important issue is that these rules are derived from the beginning of the first query. Those derived rules can be used to obtain new rules. Maximum performance while using rules takes place for the condition which the result of the query is given before accessing to the database. Another performance gain is that query can be executed by faster search methods via rules. For example performance rise is proved in the usage of primary key or usage of rules that have indexes. SQO approach has various components: Query expression, learning based on query automatically, query optimization and updating learned knowledge in other words rules.

In this thesis ,we are studying on the performance rising of query on rules to DMLs and proving the rise at the result of the study.

Keywords : Database, SQL, DDL, DML, SQO, rules, ADO, Oracle.

1 GİRİŞ

Bir veritabanı sistemi bilgisayarlı kayıt saklama sistemidir. Veritabanı birbiri ile ilişkili veriler topluluğudur. Veritabanı içinde bir veri deposu bulunur ve bu veri deposundaki verilerde zaman içinde oluşabilecek tüm değişiklikleri yaptırmayı sağlayan programlar grubudur. Bundan dolayı veritabanı üzerinde işlem yapma, veri modelini tanımlama gibi birçok işlemi barındırır. Veri modeli tanımlama, saklamak istediğimiz verilerin tiplerini, uzunluklarını, kısıtlamalarını ve nesne ilişkilerini içerir. Bir veritabanı üzerinde işlem yapma ise belirli verilerin sorgulanması, veriler üzerinde meydana gelen değişiklikleri yansıtmak üzere veri tabanının güncellenmesi ve verilerden rapor üretilmesi gibi eylemlerin gerçekleştirilmesidir.

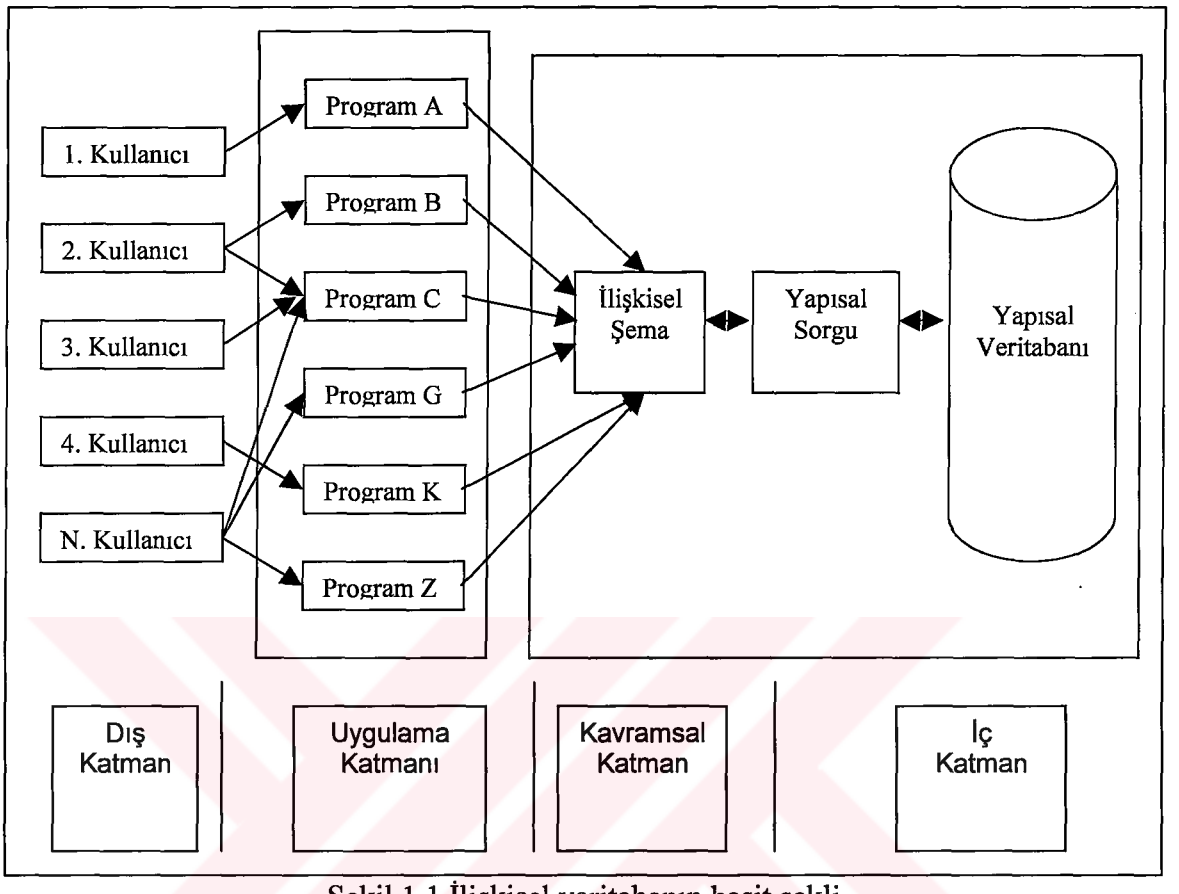
Veritabanı üzerinde veri saklama, saklanan verilerin artması ve genişlemesiyle uygulaması gün geçtikçe zorlaşan bir konu olmuştur. Kayıtların sayılarının büyümesi veritabanı üzerinde sorgulamayı, işlem yapmayı yavaşlatan ve performansı düşüren en büyük etkendir. Bunun için veri modelimizi uygun şekilde kurmalı ve verilecek sorguların performansını en iyi şekilde gerçekleştirmeliyiz.

Bu tezde veritabanı üzerinde çalıştırılan veri işleme komutlarının performansını arttırmanın yolları araştırılacaktır. Bunun için çeşitli sorgular veritabanı üzerinde çalıştırılıp bunlar üzerinden belirli kriterlere göre kurallar öğrenilir. Üretilen kurallar, sorgulama yapılması, veri işlenmesi gibi alanlarda, verilen koşullara uygulanır. Bu sayede veritabanı üzerinde yapılan işlemlerde zaman ve maliyet performansları hesaplanır. Hesaplanan değerler ise orijinal işlemlerin maliyeti ile karşılaştırılarak performans ortaya çıkarılır.

1.1 İlişkisel Veri Tabanı Sistemleri ve Yönetimi (RDBMS)

Veri yönetimi gelişen yapısı nedeniyle gerçek hayatta çok kolay uygulanan bir konu değildir. Veri saklama konusunda ilk araştırma (Codd 1970) tarafında yayınlanmıştır. Bu araştırma 2 * 2 metrikli tablosal formda nasıl tutulduğunu anlatır. Bu tablosal form matematikçiler açısından yeni bir buluş olmasa da bilgisayar mühendislerince verinin düzenli tutulması açısından önemli bir çözümdür. Bu araştırma RDBMS'nin başlangıcı olarak görülebilir. Fakat

RDBMS 70'lerin sonunda şekillendirilmiştir. (Cardes Blasgen ve Esuran 1977) Basitleştirilmiş bir RDBMS şöyle gösterilebilir.



Şekil 1.1 İlişkisel veritabanın basit şekli

Şekilde görülebileceği gibi, basitleştirilmiş RDBMS'nin 4 katmanı vardır. Dış katman, uygulama katmanı, kavramsal katman ve iç katmandır. Dış katman, tüm veritabanı kullanıcıları Kullanıcı1, Kullanıcı2.....KullanıcıN gibi tanımlanır. Uygulama katmanı, kullanıcılar tarafından istenen verilerin işlenmesini sağlayan yazılım programlarını içine alır. Bu programlar, veritabanından gelen verileri belirli şartlar altında sorgulama, güncelleme, silme veya ekleme işlemlerini yapar. (Korth and Silberschatz 91). Şekil 1.1'de bu uygulamalar A'dan Z'ye isimlendirilerek gösterilmiştir. Kavramsal katman, tutulan verilerin model detaylarını saklarken, tüm veritabanını tanımlar. İlişkisel şemalar bu katman için bir örnek olabilir. Kullanıcılar, kendi sorgularını yapabilmek için bu katmanı bilmelidirler. Son katman olan iç katman ise asıl depolama yapılarını ve depolanan verilerin değerlerini kapsar. Şekil 1.1'den görülebileceği gibi, değişik kullanıcılar aynı veya ayrı uygulama programlarını kullanıp, aynı anda ilişkisel şema tanımlamalarını kullanarak veritabanındaki aynı veriler

üzerinde sorgulama, değişiklik yapma, silme ve ekleme işlemleri yapmak isteyebilirler. Bu da veri tabanı yönetimini zorlaştırır.

RDBMS'de en geniş kullanılan sorgulama dili yapısal sorgu dili (SQL) dir. Veri ve veri modeli tanımlama kısmı ve veri işleme kısmı olarak SQL iki kısımdan oluşur. Temel olarak, veri ve veri modeli tanımlama dili (DDL) veri tabanında, veri şeması, tabloları, dizinleri, veri biçimlerini tanımlamayı sağlar. Veri işlemi dili (DML) ise verinin nasıl seçileceğini, güncelleneceği, silineceği ve ekleneceğini belirler. DML'in en çok bilinen iki çeşidi vardır. Bunlar 'İlişkisel Cebir' ve 'İlişkisel Hesap' dır. Kavramsal katman ise tanımları yapılan şema ve modelin içeriğini, depolanmış verinin iletişiminin nasıl sağlanması gerektiğini içinde saklayan alandır.

Sorgulama dillerinin temeli olarak ortaya atılan diller ilişkisel cebir ve hesaplama dilidir. Fakat bazı sorgulama dilleri DDL'lerde ki , bazı sorgulama dilleri de DML'lerdeki bazı işlemleri içermez. SQL bu nedenle ilişkisel cebir ve ilişkisel hesap ile karşılaştırıldığında en geniş anlamda DDL ve DML'leri içerir. Araştırmada ise DML'ler esas alınmış ve SELECT(Sorgulama) hariç, güncelleme(UPDATE), silme(DELETE) ve ekleme(INSERT INTO) işlemleri ele alınmış, türetilen bilginin kullanımı ile önceki halleri karşılaştırılıp performansları ölçülmüştür.

1.1.1 İlişkisel Veri Modeli

İlişkisel veri modeli, Codd'un 2*2 metrikli tablosal yapısında verileri nasıl sakladığı Çizelge 1.1'de örnek olarak gösterilmektedir. Bir ilişki (tablo) ilgili verinin tutulması için satır ve sütunlar içerir. Çizelge 1.1'deki örnekte 'MODEL' tablosunun ismi ve 'Sütun 1', 'Sütun 2'.....ve 'Sütun M' ise tablonun sütunlarıdır. 'MODEL' isimli tablo N tane kayıt ve M tane değişik sütuna sahiptir.

Çizelge 1.1 İlişkisel veri modeli

MODEL	**Sütun A**	**Sütün B**		**Sütun M**
Kayıt 1	A1	B1		M1
Kayıt 2	A2	B2		M2
KayıtN	AN	BN		MN

Codd aynı zamanda bir tabloyu aşağıdaki gibi tanımlamıştır.

MODEL(Sütun 1, Sütun 2, ...Sütun M)

Her tablo kayıt bazında bazı alanlar üzerinde tekil olmalıdır. Tekillik tablo üzerindeki birincil anahtar ile sağlanır. Tablo üzerindeki diğer sütunlar ise birincil anahtara bağlı olan kayıtlardır. Tablo üzerinde birincil anahtar sütunları 'SütunA' veya 'SütunA#' şeklinde gösterilir. Tablo üzerindeki kayıtlara birincil anahtarlar üzerinden rahatlıkla ulaşılabilir ve bu şekilde kayıtlar üzerinde arama işlemi çok daha hızlı yapılabilir. Kayıt arama işlemini anahtar alan dışındaki alanlar üzerinde de hızlı yapmak için o alanlar üzerine dizinler koyulabilir.

1.1.2 Sorgulama Dilleri

Sorgulama dilleri, kullanıcılarının veritabanı üzerinde verecekleri ifadeleri çalıştırma işleminin yapılmasıdır.

Sorulama dili olarak ilişkisel cebir ve ilişkisel hesaba ek olarak üç dil daha tanıtılmıştır. Bunlar yapısal sorgu dili (SQL), örnekle sorgu dili (QBE), İngilizce tabanlı sorgu dili (QUEL)dir. Daha önce belirtildiği gibi, ilişkisel cebir, ilişkisel hesap veritabanı sistem dilleri olarak da bilinir. Bu diller RDBMS'nin kullanılmasında iyi tanınmış olsalar da, kısıtlamalar vardır. Örneğin, ilişkisel cebir ardışık işlemler olarak inşa edilmiştir ve sisteme bağlı olan kullanıcılar yapacakları işlem sırasını belirtmek zorundadırlar. Fakat sistem kullanıcılarının işlemleri nasıl sıralandığını bilmediği düşünüldüğünde bu işlemi RDBMS'nin sorumluluğunda olmalıdır.

Veritabanı uygulamalarında en çok kullanılan sorgu dili SQL'dir. SQL, DDL(Tablo, görüntü, dizin.... oluşturma, yapısını değiştirme ve düşürme) ve DML(Sorgu, ekleme, silme, güncelleme) gibi ek özelliklere sahip ilişkisel veritabanı için IBM tarafından SYSTEM R RDBMS için tasarlanmış ve uygulanmıştır. SQL'in standart versiyonu, ANSI ve ISO'nun araştırması sonucu üretilmiştir. Programla dilleri (Örneğin: 'C', 'C++,' 'COBOL', 'PASCAL'.....vb.) için SQL komutlarının kullanılabilme özelliği vardır. Her ne kadar QUEL ilişkisel hesap temelleriyse de gömülmüş SQL ifadelerini içermez. QBE içinde aynı şey söz konusudur.

1.2 RDBMS ile İlgili Alanlar

RDBMS keşfedildiğinden beri, istatistik, olasılık ve mantık gibi değişik alanlar RDBMS'nin performansını geliştirmek için kullanılmıştır. İstatistik ve olasılık, sorgu işlemine yol göstermek amacıyla sistem katalogunda gerekli bilginin oluşturulması için kullanılır. Bu faktörler, istatistikler kullanılarak hesaplanırlar.

İstatistik ve olasılığa ek olarak, mantık fonksiyonları da RDBMS'nin geliştirilmesinde diğer bir önemli alandır. Mantık fonksiyonları veritabanı üzerinde yapılan işlemleri iyileştirme için, kural sunumu ve sorgu dönüşümünde çok kullanılır. (Minker 1996) tarafından yayınlanan araştırma veritabanı üzerinde mantık kullanımı konusunda geniş açıklamalar içerir. Bizim araştırmamızda SQO sistemlerin kural yapıları üzerindedir.

Servisler / işler için daha iyi gelecek planlar, zaman periyotlarındaki veri hareketlerine göre yapılabilir. Çok basit bir örnekle, son beş yılın içinde, bir araba firması martta 300 araba satmış olsun. Bir gelecek stok planı, firmanın martta en az 300 arabayı elinde bulundurmasını söyleyebilir. KDD bu fikre dayanmaktadır. Veri değişkenleri arasındaki benzerlikler ve farklılıklara dayanarak verinin analiz kabiliyetine sahiptir. Bu yüzden, RDBMS'nin sorgu işlemi, işlenen komutu hızlandırmak için KDD'nin yöntemlerinden yararlanır. Anlamsal sorgu iyileştirmesi(SQO) için kullanılan otomatik kural türetme yöntemleri, KDD yöntemleri olarak görülebilir. Çünkü kurallar veritabanındaki o anki sorgu ve veriyi kullanarak çıkarılırlar. Sonra bu kurallar, sorgu işlemi için kullanılır.

1.3 Sorgulama Tipleri

SQL tam açılımı ile Yapısal bir Sorgulama Dili(Structured Query Language)dir. RDBMS ile bağlantıya geçip gerek veri modelleri gerekse verilerin üzerinde işlem yapmayı sağlayan komut setlerinin tamamıdır.

1.3.1 Veri ve Veri Modeli Tanımlama Dilleri(DDL)

DDL komutları veri ve veri modeli tanımlama işlemleri yapmamızı sağlar. Bunlar

i. Yeni nesne oluşturmak, değiştirmek ve oluşturulan objeleri düşürmeye yarar.
ii. Hak vermek, ayrıcalıklara hak vermek iptal etme ve rol verme işlemleri yapılır.
iii. Tablo, dizin ve küme üzerinde bilgi analizi yapılır.
iv. Sistem özellikleri görülebilir ve değiştirilebilir.
v. Tanımlamalar hakkında açıklamalar yazılıp değiştirilebilir.

Yeni bir nesne oluşturmak için CREATE komutu uygun bir şekilde yazılarak oluşturulur. Bu tezde kullanılan komutlar anlatılmaktadır.

Create Database : Yeni bir veritabanı oluşturmak ve parametrelerini vermek için kullanılan komuttur.

Create Tablespace : Tablolar için alanlar oluşturan ve parametrelerini vermek için kullanılan komuttur.

Create User : Yeni bir kullanıcı oluşturmak, şifresini vermek ve parametrelerini vermek için kullanılan komuttur.

Create Table : Yeni bir tablo oluşturan, diğer nesnelerle ilişkisini belirten ve parametrelerini vermek için kullanılan komuttur.

Create Index : Tablolar üzerinde yeni bir dizin oluşturan komuttur.

Örneğin: Veri tabanında 'PRST0100' isimli bir tablo yapalım ve bu tablo içinde hasta no ve hasta adı bilgisini bulunduralım. Bu işlemi yapmak için aşağıdaki DDL ifadesini yazmak gerekir.

```
Create Table PRST0100 (Hasta_No Number(8), Hasta_Adi Varchar2(20))
```

1.3.2 Veri İşleme Dilleri (DML - Data Manipulation Language)

DML komutları, veriler üzerinde sorgulama, ve veriler üzerinde değiştirme, silme, ekleme işlemleri yapmamızı sağlar. Bunlar aşağıdaki belirtilmiştir.

Select : Bir veya birkaç tablo veya görüntüden veri çekmek, sorgulamak, istenen koşullardaki verileri bulmak ve gelen verileri istenilen şekilde göstermemizi sağlayan komutlardır.

Update : Bir tablodaki verilerin gerek tek kayıt ve sütun gerekse tümünü birden verilen sabitlerle veya değişkenlerle güncellenme işlemini yaptığımız komuttur.

Delete : Bir tablodaki verilerin kayıt bazında istenen koşullar çerçevesinde silinme işlemini yaptığımız komuttur.

Insert Into: Bir tabloya yeni satır veya satırlar ekleyebilmek için gerek sabit gerekse dinamik olarak veri girişi yaptığımız komuttur.

1.4 ORACLE Veri Tabanı Genel Özellikleri

RDBMS büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkanlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkanının sağlandığı yapıdır. Oracle veritabanı da bir RDBMS yapısındadır. Genel özellikleri aşağıdaki gibidir.

i. Büyük miktarda veri tutabilme ve verilerin depolandığı alanları ayarlayabilme imkanı sağlar.
ii. Aynı anda çok sayıda kullanıcıya birden verilerin bütünlüğünü bozmadan hizmet verebilmektedir.
iii. Günün 24 saati ve haftalar boyu hiç kapatılmadan çalışabilmektedir.
iv. İşletim sistemi, veri erişim dilleri ve ağ iletişim protokolleri standartlarıyla uyumludur.
v. Yetkisiz erişimleri engelleme ve kontrol edebilme imkanı sağlamaktadır.
vi. Bütünlüğü veritabanı düzeyinde sağlayabilmektedir, böylece daha az kod yazılmaktadır.

vii. İstemci/Sunucu mimarisinin bütün avantajlarını kullanabilmektedir.

Oracle ile ilk defa karşılaşan kullanıcılar genellikle Delphi, Visual C++, Visual Basic gibi görsel programlama dillerine benzeyen uygulamalarla karşılaşmayı umarlar. RDBMS bir programlama dili değildir. Bunun için Oracle tarafından geliştirilen ve Oracle'ın kendi uygulama geliştirme araçları içerisinde kullanılan bazı programlama dilleri vardır. Oracle ürünleri genellikle büyük çaplı veri kontrolünü gerektiren uygulamalarda kullanılır.

Oracle veritabanının, işletim sistemi tarafından bakıldığında, biri fiziksel diğeri mantıksal olmak üzere iki bölümü vardır. Fiziksel bölüm, işletim sisteminden görünen kısımdır. Bunlar Veri(Data) dosyası, Kontrol(Control) Dosyası ve Kütük(Log) dosyalarından oluşmaktadır. Mantıksal Bölüm, bir ya da daha fazla tablo alanı(Tablespace) ve tablolar(table), görüntüler(view), sıralar(sequence), eşanlamlar(synonym), dizinler(index), kümeler(cluster), veritabanı bağlantıları (database link), prosedürler(procedure), fonksiyonlar(function), ve paketlerden(package) oluşan şema nesnelerinden oluşmaktadır. Fiziksel bölüm işletim sistemi tarafından görülebilmesine rağmen, mantıksal bölüm ancak Oracle'a bağlanıp, SQL komutları çalıştırılarak görülebilmektedir. Yani, Oracle kurulu herhangi bir makinede, SQL bilgisi olmayan bir insan, Oracle'ın sadece fiziksel bölümünü görebilmektedir.

Oracle veritabanındaki her nesnenin bir sahibi vardır. Her kullanıcı bir veya daha fazla tablo uzayına sahip olabilir. Her nesne, ait olduğu kullanıcının herhangi bir nesne uzayında (mantıksal olarak) bulunur. Her nesne uzayı da, kendisine sahip olan kullanıcının nesnelerini tutmak için işletim sisteminde bir veya daha fazla veri dosyasına sahiptir.

Sonuç itibariyle, veritabanındaki her nesnenin bir kullanıcısı vardır ve bu nesneler mantıksal olarak o kullanıcının sahip olduğu nesne uzaylarının herhangi birinin içerisindedir, fiziksel olarak da o kullanıcının sahip olduğu nesne herhangi bir veri dosyasında bulunur. Fakat, o veri dosyasının içerisine işletim sistemi üzerinden bulunamaz. Bu nesnenin sahibi ve mantıksal yeri sadece 'DML' komutları ile bulunabilmektedir.

1.4.1 Fiziksel Bölüm

Fiziksel bölüm veritabanını oluşturan işletim sistemi dosyalarıdır. Bir Oracle veritabanı fiziksel olarak bir ya da daha fazla veri dosyası, iki ya da daha fazla kütük dosyası, bir ya da daha fazla kontrol dosyasından oluşur.

1.4.1.1 Veri Dosyaları

Veri dosyaları veri tabanındaki tüm verileri tutan dosyalardır. Tablo, dizin gibi mantıksal veritabanı yapılarının içerisindeki veriler fiziksel olarak veri dosyalarında tutulurlar. Bir veri dosyası kendisi için ayrılan alan dolduğunda, alanını artırabilecek özelliklere sahiptir.

Veritabanı işlemleri boyunca bir veri dosyası içerisindeki veriler okunur ve Oracle için ayrılan belleğe getirilir ve bu bellek üzerinden sorgu ve işlemler yapılır. Örneğin bir kullanıcı veritabanındaki bir tablo verilerine erişmek istediğinde önce belleğe bakılır eğer istenilen veriler bellekte yer almıyorsa, ancak o zaman uygun veri dosyasından okunur ve belleğe getirilirler.

Veriler üzerinde yapılan değişiklikler veya eklenen veriler veri dosyalarına hemen yazılmazlar. Bunun sebebi sabit diske erişimi azaltmak ve böylece sistemin performansını artırmaktır. Bunun için ise önce veriler bellek havuzunda tutulur ve gerektiğinde hepsi birden uygun veri dosyalarına kaydedilirler. Bunu Oracle üzerinde arka planda çalışan işlemleri yapar.

1.4.1.2 Kontrol Dosyaları

Tüm Oracle veritabanları kontrol dosyasına sahiptir. Bir kontrol dosyası veritabanı adı, veri dosyaları, kütük dosyalarının adı, diskteki yeri, veritabanının oluşturulma tarihi vb. veritabanı ile ilgili bilgileri tutar.

Her veritabanı oturumu açıldığında Oracle bu dosyayı kontrol ederek gerekli bilgileri alır. Eğer veritabanında fiziksel bir değişme olursa(yeni bir kütük dosyası ya da veri dosyası

oluşturulması gibi), yapılan değişiklikler Oracle tarafından otomatik olarak kontrol dosyalarına yansıtılır.

1.4.1.3 Kütük Dosyaları

Yeniden yapım kütük dosyaları(Redo Log) olarak bilinen bu dosyaların amacı veriler üzerinde yapılan tüm değişiklikleri kaydetmektir. Eğer veri dosyalarına kaydedilmiş olan, kayıtlarda bir bozukluk olursa yapılan değişiklikler kütük dosyalarından sağlanabilir ve yapılan işlemler bu sayede kaybolmaz. Birden fazla tekrarlanan bozukluk durumlarında kütük dosyalarının da bozulmasını engellemek için Oracle farklı diskler üzerinde bu dosyalarının birden fazla kopyasının alınmasına olanak sağlar.

Örneğin bir veritabanı işlemi sırasında elektrik kesilirse, bellekteki veriler veri dosyalarına kaydedilmeyecek ve verilerin kaybolması durumuyla karşılaşılacaktır. Oracle veritabanı tekrar açıldığında kütük dosyalarında yapılan son değişiklikler veri dosyalarına yansıtarak verilerin kaybolması engellenir.

1.4.2 Mantıksal Bölüm

Oracle veritabanının mantıksal yapısı tablo alanları(tablespaces), şema nesnelerini(schema objects), veri bloklarını(data blocks), genişlemeleri(extents) ve parçalarını(segments) içerir.

1.4.2.1 Tablo Alanı

Bir veritabanı, ilişkili mantıksal yapıların gruplaşmasını sağlayan ve tablo uzayı olarak bilinen mantıksal depolama ünitelerine bölünmüştür.

1.4.2.2 Veri Tabanı Şema Nesneleri

Şema nesneleri mantıksal veri depolama yapıları olarak bilinir. Veritabanı üzerinde kullanıcının belirli işleri yapabilmesi için tanımlanan bu yapılar tablolar, görüntüler, sayaçlar,

eşanlamlar, dizinler, kümeler, veritabanı bağlantıları, prosedürler, fonksiyonlar, ve paketlerdir. Bir şema ise bu nesnelerin oluşturduğu gruptur.

1.4.2.3 Küme

Aynı anda sorgulanan birden fazla tablonun bir arada kaydedilmesini sağlayan yapıdır. Bu yapı, beraber sorgu yapılan tablolarda hız kazanmak için çok önemlidir.

1.4.2.4 Dizin

Dizinler tablo ve kümeler için kullanılan veri tabanı nesneleridir. Dizinlerdeki amaç aranılan bir kayda daha hızlı erişimdir. Özellikle üzerinde çok arama yapılan alan veya alanlar üzerinde dizin oluşturmak çok etkilidir. Sadece ilgili kayıtların kayıt numaraları olarak adlandırılan rowid'ler alınarak sıralama yapılır. Dizin oluşturulduğunda ilgili tablonun kayıtları yer değiştirmez.

Bir tablo üzerinde oluşturulabilecek dizin sayısı sütunların kombinasyonları farklı olduğu müddetçe sınırsızdır. Aynı sütun kombinasyonlarının dizini sadece bir kez oluşturulabilir.

Dizinler mantıksal ve fiziksel olarak oluşturuldukları tablodan bağımsızdırlar. Eğer bir dizin silinirse, ilgili tablo zarar görmez, çalışmaya devam eder. Fakat dizin olmadığı için veriye erişim süresi artacaktır yani erişim yavaşlayacaktır.

Oracle, bir dizin oluşturulduğunda onu otomatik olarak kullanmaya başlar ve dizinin oluşturulduğu tabloda silme, güncelleme ve ekleme işlemleri yapıldığında dizine otomatik olarak değişim yansıtılır.

1.4.2.5 Rol

Oracle veritabanında her nesnenin ait olduğu bir kullanıcı vardır. Bir kullanıcı bir başka kullanıcının nesneleri üzerinde işlem yapmak isterse buna hakkı olması gerekir. Örneğin veritabanına bağlanma, tablo oluşturma, bir başkasına ait tablolarda sorgulama, veri

değiştirme, silme, ekleme, bir başkasının prosedürünü çalıştırmak için buna hakkının olması gerekir. Hakların verilmesiyle kullanıcıların bu işlemeleri gerçekleştirmeleri sağlanır. Hakların kullanıcılara atanması iki şekilde olabilir. Birinci olarak bir tabloya kayıt ekleme, kayıt silme vb. haklar bir kullanıcıya ya da kullanıcılara ayrı ayrı atanır. İkinci şekilde ise verilmek istenen haklar bir rol altında birleştirilir ve bu rol istenen kullanıcılara aktarılır.

Haklar 'sistem hakları' ve 'nesne hakları' olmak üzere ikiye ayrılır. Sistem hakları veritabanı ile ilgili olarak Oracle'da önceden tanımlanmış rollerdir. Oracle'da 60'tan fazla sistem hakkı tanımlanmıştır. Nesne hakları ise veri tabanı nesneleri üzerinde işlem yapma haklarıdır. 'Create Table', 'Create TableSpace', 'Drop Any Index' gibi haklar sistem haklarına örnek olarak verilebilir. Nesne hakları ise 8 çeşittir. Bunlar 'Select', 'Insert Into', 'Update', 'Delete', 'Alter', 'Index', 'Execute', 'References'dir. 'All' hakkı ise tüm hakları kapsar. Bu haklar sırayla kayıt seçme, kayıt güncelleme, kayıt silme, nesnelerin yapısını değiştirme, dizin oluşturma, alt program çalıştırma, yabancı anahtar tanımlayabilme işlemlerini içerirler.

1.4.2.6 Geri Alma Parçası

Oracle veritabanının güvenliği açısından 'Insert Into', 'Update', 'Delete' gibi işlemlerde yapılan değişikliklerin yedeğini almaktadır. Alınan yedeklerin konulduğu yerlere geri alma parçası denir. Kullanıcı veriler üzerinde yaptığı değişiklikleri aynı işlem içersinde geri alabilir. Bu işlemi 'Rollback' komutu ile yapar. Kullanıcı 'Commit' komutunu vererek bu değişiklikleri geri almaksızın yapmış olur. Her veritabanında bir ya da birkaç tane geri alma parçası olabilir.

1.4.2.7 Sayaç

Tablolardaki kayıtlar için otomatik sıra numarası verilmesi isteniyorsa sayaç nesnesi kullanılabilir. Bu sıra numarası veritabanı tarafından otomatik olarak üretilir. Özellikle çok kullanıcılı ortamlarda tekil olarak numara üretilmek istendiğinde çok kullanışlıdır. Birden fazla kullanıcı aynı anda böyle bir sayı üretmek isterse bunun program koduyla yapılması işlem hızını yavaşlatır. Çünkü bir kullanıcı diğeri işini bitirene kadar beklemek zorundadır. Sayaç nesnesi bu işi otomatik olarak ve çok seri bir şekilde başarır.

Sayaçlar Oracle'da tanımlı 38 haneye kadar tamsayılardan oluşturulabilir. Bir sıra tanımlaması sayacın adını, artan ya da azalan olacağını, iki sayı arasındaki fark miktarını içerir. Oracle tüm sayaç tanımlarını SYSTEM tablo alanının içerisindeki bir veri sözlüğü tablosuna kaydeder. SYSTEM tablo alanı sürekli çalışır durumda olduğu için tüm sayaçlarda aktiftir. Sayaçlar tablolardan bağımsız olarak üretilir. Yani bir sayaç bir ya da birkaç tablo için kullanılabilir.

1.4.2.8 Eşanlam

Eşanlam bir tablo, görüntü, sayaç, prosedür, fonksiyon ya da paket için 'alias' olarak adlandırılan bir takma isimdir. Eşanlam bir takma isim olduğu için sadece veri sözlüğü içerisinde yer kaplar. Eşanlamlar güvenlik ve daha rahat kod yazma amacıyla kullanılırlar. Bir eşanlam kullanıldığında ilgili nesnenin adı ve sahibi gizlenir ve SQL komutu içerisinde kullanımı kolaylaşır ve daha güvenli bir hale gelir.

Eşanlamlar genel(Public) ve özel(Private) olarak tanımlanabilirler. 'Genel' olarak tanımlanan eşanlamlara tüm veritabanı kullanıcıları erişebilir. 'Özel' olarak tanımlanan eşanlamlara sadece ilgili nesnenin sahibi ve sahibi tarafından hak verilmiş bir başka kullanıcı erişebilir.

Bir nesnenin adı değiştirilmek ya da silinmek istendiğinde, bu nesneyi kullanan tüm uygulama programları değiştirilmek zorundadır. Oysa ki bu nesnenin bir eşanlamı oluşturulursa ve uygulama programları bu eşanlamı kullanırsa, eşanlam üzerinde yapılacak değişikliklerle uygulama programlarının etkilenmesi önlenebilir.

Bir kullanıcı bir başka kullanıcının nesnesini kullanmak istediğinde "kullanıcı_adı.nesne_adı" şeklinde bir yazım kuralına uymak zorundadır ve eğer bir eşanlam tanımı yapılırsa tüm kullanıcılar direk eşanlam ismini kullanarak işlemlerini gerçekleştirebilirler.

1.4.2.9 Tablo

RDBMS'de veriler, tablolar içerisinde yer alır. Her tablo bir isimle tanımlanır ve bir ya da daha fazla sütuna sahip olabilir. Her sütunun bir adı ve veri tipi vardır. Bir tablo

oluşturulduğunda Oracle verileri depolamak için bir tablo alanı içerisinde bir veri parçası ayırır.

1.4.2.10 Görüntü

Görüntü bir ya da birkaç tablodan istenilen sütunların alınmasıyla oluşturulan sanal bir tablodur. Görüntü bu tablolar üzerinde gerçekleştirilen bir sorgu sonucu oluşturulur. Görüntüler üzerinde veri değiştirme yapılıp yapılmayacağı görüntü oluşturma sırasında belirlenir. Fakat veri değiştirme yani güncellemelerin görüntülerle yapılması sakıncalıdır. Özellikle iki ve daha fazla tablodan oluşan görüntülerde güncelleme yapılmamalı, tabanını teşkil eden tablolarda UPDATE komutu ile yapılmalıdır.

1.4.2.11 Veri Blokları, Genişlemeler, Parçalar

Oracle veritabanında verilerin depolandığı en küçük birim veri bloğu olarak adlandırılır. Bir veri bloğu veritabanının depolama alanı üzerindeki belli bir byte uzunluğuna karşılık gelir. Veri bloğunun uzunluğu veritabanı oluşturulurken belirlenir.

Veri bloklarının bir üst birimi genişleme olarak adlandırılır. Bir genişleme art arda olan belirli sayıda veri bloğundan oluşur.

Genişlemelerin bir üst birimi de parçalardır. Parçalar belli bir mantıksal yapı için ayrılmış bir dizi genişlemeden oluşurlar. Farklı amaçlar için kullanılan parçalar vardır. Bunlar veri parçaları(data segments), dizin parçaları(index segments), geri alma parçaları(rollback segments) ve geçici parçalardır(temporary segment).

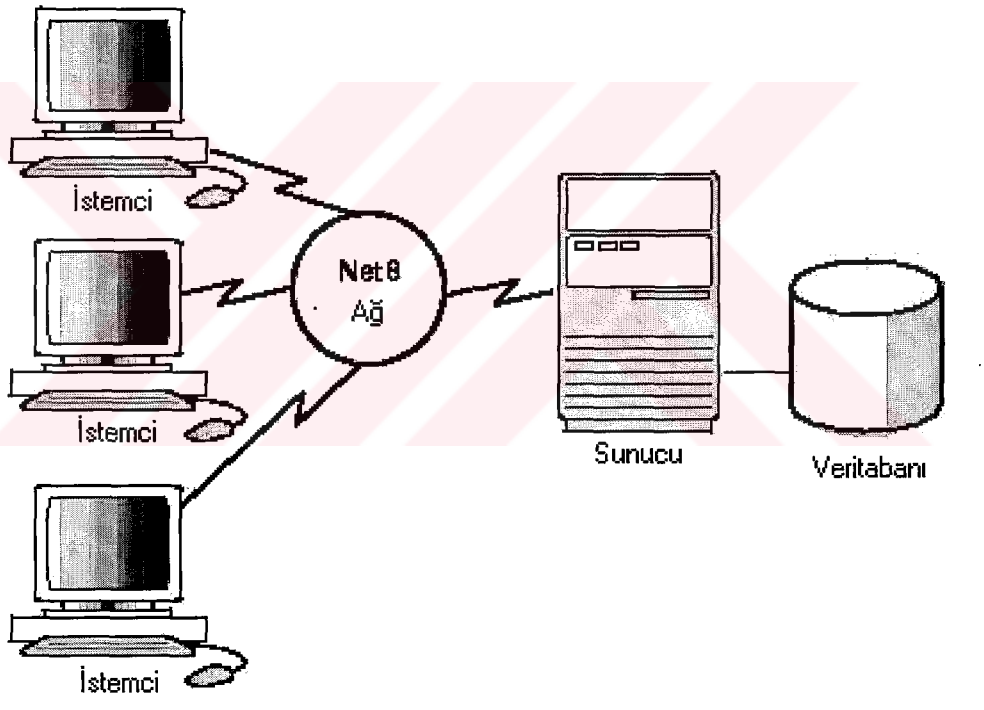
Bir tablo bir veri parçasından oluşur. Tablonun verileri bu parça içerisindeki genişlemelere kaydedilir. Küme(cluster)'lerde bir veri parçasından oluşur ve küme içerisindeki tüm tablolar o kümenin veri parçasında yer alır.

Bir veritabanında, veritabanı kurtarma işlemleri(database recovery) ve yapılan değişiklikleri geri alma işlemleri(rollback) için bir ya da daha fazla geri alma parçaları yer alır. Geçici

parçalar Oracle tarafından SQL komutları işletilirken ihtiyaç olduğunda oluşturulur ve SQL komutu işlemini bitirdiğinde bu parça tekrar sistemin kullanımına bırakılır.

1.4.3 Oracle'a Erişim

Oracle'da ağ üzerinde bir istemcinin sunucudaki veritabanına erişebilmesi için, sunucuda bir ağ servis adının(Net Service Name) ve bir dinleyicinin(Listener) oluşturulması gerekir. Veritabanına bağlantı kurması ve istemci-sunucu arasında veri alışverişinin sağlanabilmesi Oracle'ın bir ürünü olan Net8 ile sağlanır. Net8, Oracle'ı kullanacak olan ağdaki her bilgisayara kurularak ağ bağlantısı sağlandığında NET8 istemci ile sunucu arasında bir veri taşıyıcısı gibi işlem görür.



Şekil 1.2 Oracle'a erişim yapısı

1.4.4 Oracle'ın Çalışma Yapısı

1. Oracle veritabanı sunucu olarak adlandırılan bilgisayarda çalışıyor vaziyettedir.
2. Bir kullanıcı istemci bilgisayarda kullanıcı işlemlerini gerçekleştiren bir uygulama programını çalıştırmaktadır. İstemci bilgisayar sunucu bilgisayar ile bağlantısını uygun Net8 sürücüsünü kullanarak gerçekleştirir.

3. Sunucu bilgisayarda da uygun bir Net8 sürücüsü çalışıyor vaziyettedir. Sunucu uygulama programından gelen bağlantı isteğini tespit eder ve kullanıcı işlemine karşılık gelen sunucu işlemini oluşturur.
4. Sunucu işlemi komutu alır ve paylaşım havuzunda bu SQL komutuna benzeyen bir paylaşımlı SQL alanı olup olmadığına bakar. Eğer böyle bir alan bulunursa sunucu işlemi kullanıcının bu SQL cümlesini çalıştırma haklarını kontrol eder. Eğer böyle bir alan yoksa yeni bir paylaşımlı SQL alanı oluşturulur ve SQL komutu çalıştırılır.
5. Sunucu işlemi bu SQL komutu için gerekli verilerin bellekte olup olmadığına bakar. Eğer burada yoksa ilgili veri dosyasından verileri alıp belleğe getirir.
6. Sunucu işlemleri komutun gereklerine göre bellekteki verileri değiştirir. Daha sonra Oracle'ın arka planda çalışan işlemleri ile diske yazılır.
7. Kullanıcı bir SQL komutu çalıştırır ve yaptığı değişikliği "commit" eder, yani onaylar.
8. Eğer SQL komutunun çalıştırılması başarılı olduysa sunucu işlemi ağ üzerinden istemcideki uygulamaya mesaj gönderir. Eğer başarılı olmadıysa uygun hata mesajını gönderir.
9. Tüm bu işlemler yapılırken veritabanı sunucusu diğer kullanıcıların aynı ya da farklı veriler üzerindeki işlemlerini de yürütür.

1.5 İlişkisel Veri Tabanlarında Programlama Dillerinin Kullanımı

Programlama dillerinden veritabanlarına erişim için iki ayrı yol bulunmamaktadır. Bunlar Gömülmüş SQL üzerinden (Embedded SQL) ve Aktif Veri Objesi üzerinden (ADO - Activex Data Objects) kullanmaktır. Bu araştırmada ADO yapısı kullanıldığı daha ayrıntılı anlatılacaktır.

1.5.1 Gömülmüş SQL (Embedded SQL)

SQL'in programlama dilleri içerisinde kullanıldığı oluşuma Gömülmüş SQL denir. Bu diller PASCAL, COBOL, C/C++, JAVA..... olabilir. Uygulamalar dillerin özellikleri ile oluşturulur, dosyaları ise veritabanında tutulur.

Bu yapıda SQL ifadeleri diller içinde genellikle EXEC SQL komutunun devamı olarak verilir.

Örneğin: Burada <veritabanı> veritabanı ismi, <:host variable> sunucu değişkeni verilerek veri tabanı bağlantısı sağlanır. Kullanımı aşağıda verilmiştir:

EXEC SQL AT <dbname> <SELECT İfadesi>

veya

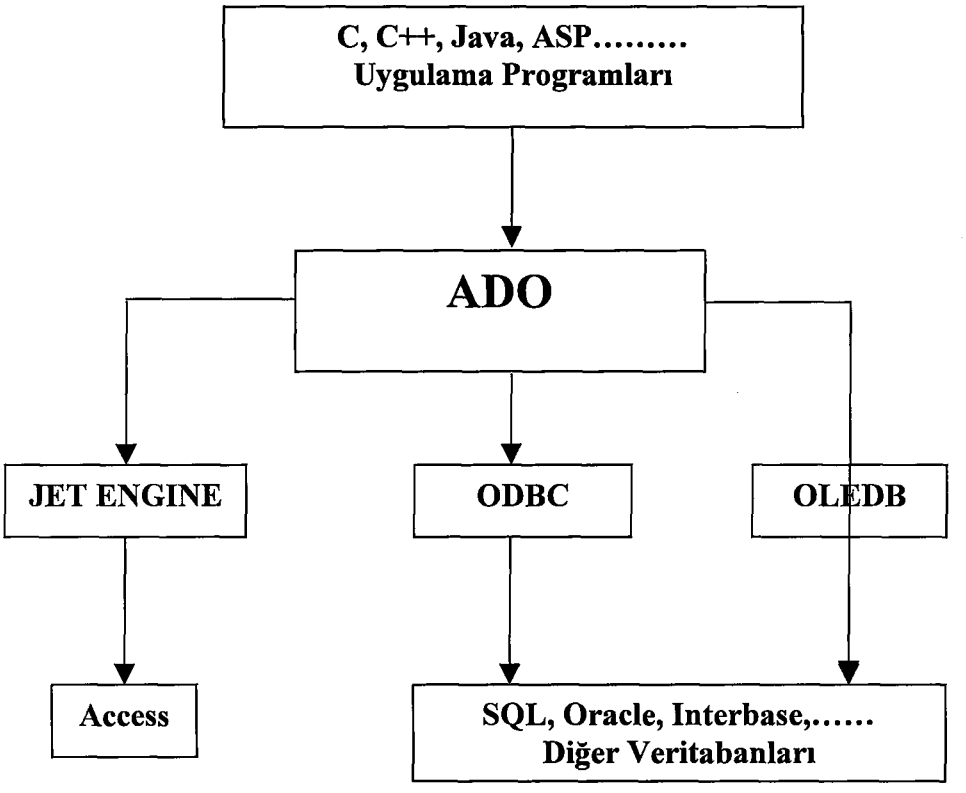
EXEC SQL AT <:host variable> <SELECT İfadesi>

Araştırmamızda Oracle 7.3.3'ün Visual C++ dilindeki gömüşmüş SQL komutları SELECT, UPDATE, DELETE, INSERT INTO ve veritabanı bağlantısı ve koparılması işlemlerinde kullanılmıştır.

1.5.2 Aktif Veri Nesneleri (ADO)

Microsoft Windows 95 ile standart olarak gelen aktif veri nesnesidir ve veritabanına erişim yapısıdır. ADO'da veritabanına erişim üç yolla gerçekleştirilir:

i. Microsoft Jet Engine,
ii. Açık Veritabanı bağlantısı (ODBC – Open Database Connectivity),
iii. Doğrudan Veritabanına erişim (OLEDB).



Şekil 1.3 ADO genel yapısı

Microsoft Jet Engine ya da kısaca jet, Microsoft Access dosyalarına erişmek için kullanılır. Bu dosyanın yerel bilgisayarda olması da zorunlu değildir, ağ üzerindeki başka bir bilgisayarda bulunan bir Microsoft Access dosyasına Jet aracılığıyla erişim yapmak mümkündür.

ODBC, yalnızca Access formundaki değil: dBase, Oracle, Paradox, Excel vs. biçimlerindeki veritabanlarına da ulaşabilmek için kullanılan bir arabirimdir. Bağlantı bu ara birim üzerinden geçerek veri tabanına gider.

OLEDB, veritabanlarına erişim yöntemlerinin üçüncüsü ise doğrudan erişim yöntemidir. Bu yolla, Jet ve ODBC gibi, aslında birer bürokrasi olan arabirimler ortadan kaldırılmakta ve veriye doğrudan erişim sağlanmaktadır. Yetki verilmiş olması halinde, yeryüzünün herhangi bir noktasındaki bir sunucunun veritabanına, OLEDB ile doğrudan erişim mümkündür.

Bu araştırma çalışması, Visual C++ programlama dilinden Oracle veritabanına bağlantı ADO-ODBC üzerinden yapılarak gerçekleştirilmiştir.

1.6 Visual C++ Programlama Dli

C++ bilindiği gibi programlama dünyasında en çok ilgi çeken C dilinden türemiştir. C dili 1970'li yılların başında AT&T Bell laboratuarlarında Dennis M. Ritchie tarafından yazıldı. Kendisinden önceki B dili üzerine kurulu bir yapı oluşturulmasından dolayı bu dil C olarak isimlendirilmiştir. UNIX işletim sisteminin yazımı için oluşturulan C dili başlangıçta sa- dece Bell laboratuarlarında kullanıldı. C dilinin programcılar tarafından tanınmasındaki en Önemli adım 1978 yılında Dennis M. Ritchie ve Brain W. Kernighan tarafından"C Programming Language" isimli kitabın yayınlanması olmuştur.

1.6.1 Temel Özellikleri

Dilin seviyesi : Diller C'ye kadar Yüksek Seviyeli ve Düşük Seviyeli olmak üzere iki gruba ayrılırdı. C görünüş olarak yüksek seviyeli bir dil olarak görünürken temelde işlevleri düşük seviyeli dillere çok yakındır. C dilinin geliştirilmesindeki amaç, C'nin düşük seviyeli dillerden Assembly'nin yerini almasıdır. İşlev hızı çoğu zaman Assembler'a yakındır.

Makineye bağımlılık ve taşınabilirlik : Assembly ile makineden bağımsız işletim sistemlerinin yazımı oldukça zordur. Bunun yerine tüm işletim sistemlerine kolaylıkla uyarlanabilecek bir dil olan C oluşturulmuştur.

Hız : C programları diğer dillere nazaran çoğu zaman daha hızlı çalışır. C programlarının hızlılığının ana sebebi, komutların çoğunun içinden işlemleri kontrol mekanizmalarının kaldırılmasıdır. Kontrol mekanizmaları programcının isteğine sunulmuştur.

Zengin komut kütüphanesi : C'de hemen hemen her işlem için bir komut mevcuttur. Ayrıca programcının kendi komut kütüphanelerini de oluşturulması mümkündür.

Nesne yönelimli : C++'ı klasik C dilinden farklı yapan Nesne Yönelimli Programlamasıdır. (Object Oriented Programming) C'nin sözdizimi kurallarıyla birlikte C++'sında desteklemesidir. Normalde C ile sadece yapısal programlama yaparken C++ dili ile hem yapısal hem de nesne yönelimli programlar yazılmaktadır.

1.6.2 Veritabanına Erişim

Visual C++ ile ADO üzerinden bir veritabanına bağlanmak için "msado15.dll" dosyasının C++ içerisinde çağırılması gerekmektedir. Bundan sonra bir bağlantı nesnesi oluşturularak o nesne üzerinden belirleyeceğimiz bağlantı şekli ve parametrelerle veritabanı bağlantı cümleciği verilerek bağlantı açılır.

C1) driver={Microsoft ODBC for Oracle};server=DENEME;uid=YTU;pwd=YTU

C2) Provider=MSDAORA;User ID=YTU;Password=YTU;Data Source=DENEME

C1 ve C2'de Visual C++'dan Oracle veritabanına bağlantı cümleleri verilmiştir. Burada C1 Windows ODBC üzerinden bağlantısı gerçekleştirirken, C2 Oracle'a direkt erişim sağlar. Veritabanı bağlantısı sadece bir kere açılır ve açık bağlantı üzerinden ODBC kullanılır veritabanı ile işimiz bittiğinde veritabanını açtığımız nesne ile veritabanı kapatılır.

2 SORGULAMA DİLLLERİ VE TİPLERİ (SQL)

SQL (Yapısal Sorgulama Dili - **S**tructured **Q**uery **L**anguage), RDBMS ile, iletişime geçmek için kullanılan komut setlerinin tamamına verilen isimdir.

Veritabanı veya tablo oluşturulması, kayıt ekleme veya silme işlemleri gibi bütün veritabanı işlevleri SQL yapıları ile gerçekleştirilir.

SQL'in yerine getirdiği işlemlerin bazı veritabanı sistemlerinde farklılık göstermesi, o veritabanı yönetim sistemini tasarlayan yazılım şirketinden kaynaklanmaktadır. Temel fonksiyonları yerine getiren komut yapıları her zaman için kalıcı ve bütün SQL veritabanlarında aynıdır. Daha öncede söz edildiği gibi SQL iki kısımdan oluşur: DDL ve DML'ler. Bu bölümde SQL'in bu kısımlarına daha içerikli olarak yer verilecektir.

2.1 Veri ve Veri Modeli Tanımlama Dili (DDL-Data Definition Language)

Nesne tanımlamaların veya tanımlanmış nesneler üzerinde tanımlamalarında değiştirme, düşürme ve ekleme yapan komutlar bütünüdür. Bunun için uygulama yapılan nesne üzerine paylaşımsız erişim hakkı gerektirir. Örneğin ALTER TABLE(Tablo Bilgileri Değiştirme) komutu o anda başka bir kullanıcı tarafından işlem yapılıyorsa tablonun kullanıldığını belirtir ve diğer kullanıcı için komutun çalışmasına başlanmaz. İlk kullanıcı işini bitirdikten sonra diğeri işlemini yapabilir.

Hak verme, kaldırmak, analiz, hesap işlemleri ve açıklama komutları uygulama yapılan nesne üzerinde paylaşımlı erişim gerektirmez. Örneğin diğer kullanıcılar tabloyu güncellerken aynı anda tabloyu analiz etmek mümkündür.

2.1.1 Veritabanın Oluşturması (CREATE DATABASE)

2.1.1.1 Amaç

Yeni bir veritabanını oluşturmak için kullanılır. Veritabanına erişecek olan bağlantı sayısı belirlenir. Veri dosyası, sistem dosyaları ve diğer parametreleri verilir.

Veri dosyaları ve sistem dosyalarının belirlenmesini sağlar.

Sistem dosyalarının ipinin belirlenmesini sağlar.

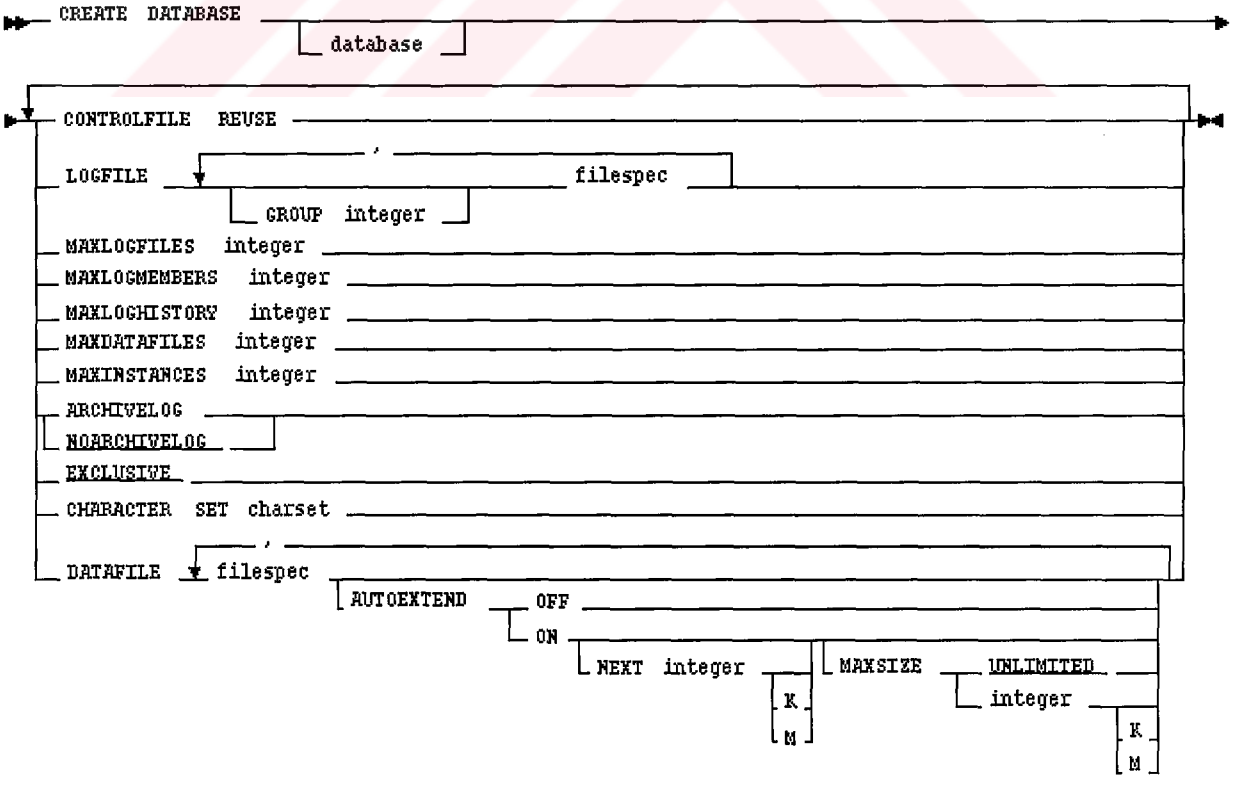
Uyarı: Bu komut kullanıldığında belirtilen dizin altındaki Oracle dosyaları ile aynı isimdeki dosyaları silerek kendi dosyalarını oluşturur.

2.1.1.2 Ön Gereksinimler

Veritabanı oluşturma komutunun kullanımı için, kullanıcının kullanıcı oluşturma 'CREATE USER' hakkına sahip olması gerekmektedir ve veritabanının çalışacağı alan belirlenmelidir.

2.1.1.3 Veritabanı Oluşturma Komutunun Kullanımı

Aşağıdaki şekilde veritabanı oluşturma komutunun kullanımı ve parametreleri görülmektedir. Veritabanı oluşturma komutunun kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanlar açıklanacaktır.



Şekil 2.1 Veritabanı oluşturma parametreleri

(Oracle7 Server SQL Ref. Man.)

Veritabanı Adı (Database) : Veritabanının isminin verildiği parametredir. Bu isim en fazla sekiz karakterli olabilir. Veritabanı oluşturulmasında veritabanı adı verilmeyebilir. Bu takdirde veritabanı ismi daha sonrada verilebilir. 2.1.1.5'de ki örnekte veritabanı adı 'TEST' olarak verilmiştir.

Kontrol Dosyaları (Controlfile Reuse) : Kontrol dosyalarının oluşturulmasını ve ilk kullanım için ayarlanmasını sağlar. Bu parametre sadece yeni bir veritabanı oluşturulurken verilebilir. Bunun için veritabanı oluşturulurken bu parametrenin verilmesi zorunludur.

Kütük Dosyaları (Logfile) : Bir veritabanına birden fazla kütük dosyası verilebilir. Her bir kütük dosyası için birden fazla üye dosyası oluşturulabilir ve kütük dosyaları birden fazla verildiğinde GROUP parametresi kullanılır. Aynı grup değerinde birden fazla kütük dosyası belirlenemez. MAXLOGFILE parametresi ile en fazla kütük sayısı verilebilir. 2.1.1.5'deki örnekte "log1.log" ve "log2.log" kütük dosyalarıdır.

Kısmı Durum (Exclusive) : Veritabanı oluşturulduktan sonra (mount)monte konumunda açılmasını sağlar. Bu biçim veritabanına yalnızca kendi (instance)örneğimizin erişmesini sağlar. Birden fazla örneğin erişimi için ilk olarak veritabanını oluşturulmalı, kapatılmalı ve (dismount) monte edilmemiş olarak açmalıyız ve paralel biçimde monte etmeliyiz.

Karakter Tipi (Character Set) : Veritabanında saklanacak olan verinin karakter tipini belirler. Veritabanı oluşturulduktan sonra karakter tipi değiştirilemez. Seçilen karakter tipini işletim sisteminin desteklemesi gerekmektedir. Karakter setinin seçiminde veritabanında saklanacak olan dilleri desteklemesine dikkat edilmesi gerekmektedir.

Veri Dosyası (Datafile) : Veritabanında kullanıcı verilerinin saklandığı dosyalardır. Bütün bu dosyalar sistem dosyalarının bir parçasını oluştururlar. Dosya ismi ve boyutu işletim sisteminin belirlediği şekilde olmalıdır. 2.1.1.5'de ki örnekte "C:\ORANT\ORADB\df1.dbf" ve "C:\ORANT\ORADB\df2.dbf" veri dosyalarıdır.

Otomatik Genişleme (Autoextend) : Veri dosyalarının otomatik olarak genişleyip genişlemeyeceği ayarlanır.

i. OFF : Veritabanı dosyalarında dosya büyümesi kullanıcı tarafından yapılacak demektir.

ii. ON : Veritabanı dosyalarında dosya büyümesi otomatik olarak yapılacak demektir.

NEXT: Genişlemenin her defasında ne kadar genişleyeceği belirlenir.

MAXSIZE : Genişlemelerin ulaşabileceği en büyük değerdir.

UNLIMITED : Genişlemelerin sınırsız olmasıdır.

2.1.1.5'de ki örnekte otomatik genişleme ayarlanmış ve en fazla 10 Megabyte olarak genişleme sınırı verilmiştir.

2.1.1.4 Örnek

Burada 'TEST' veritabanı adı 'log1.log' ve 'log2.log' veritabanı kütük dosyalarıdır ve boyutları 50 Kilobyte'dır. Veri dosyaları 'C:\ORANT\ORADB\df1.dbf' ve 'C:\ORANT\ORADB\df2.dbf''dir ve boyutları 10 Megabyte'dır. Dosya büyümesi sınırsız olarak ayarlanmıştır.

```
CREATE DATABASE TEST
        CONTROLFILE REUSE
        LOGFILE
                GROUP 1 ('log1.log') SIZE 50K,
                GROUP 2 ('log2.log') SIZE 50K
        DATAFILE
          'C:\ORANT\ORADB\df1.dbf' AUTOEXTEND ON
          'C:\ORANT\ORADB\df2.dbf' AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
```

2.1.2 Yeni Tablo Alanlarının Oluşturması (CREATE TABLESPACE)

2.1.2.1 Amaç

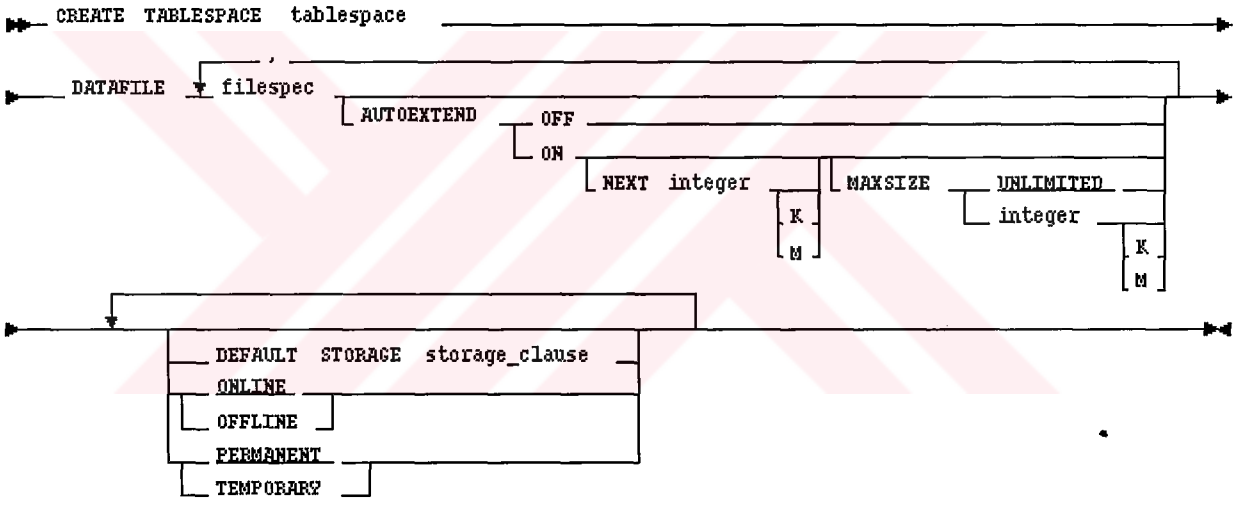
Tablolar için yeni veri alanları tanımlamak için kullanılır. Nesneler içerebilen alanlar için veritabanında yerlerinin ayrılmasıdır.

2.1.2.2 Ön Gereksinimler

Yeni tablo alanları oluşturma komutunun kullanımı için kullanıcının 'CREATE TABLESPACE' hakkına sahip olması gerekmektedir.

2.1.2.3 Yeni Tablo Alanları Oluşturma Komutunun Kullanımı

Aşağıdaki şekilde yeni tablo alanları oluşturma komutunun kullanımı ve parametreleri görülmektedir. Yeni tablo alanları oluşturma parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanlar açıklanacaktır.



Şekil 2.2 Yeni nesne alanları oluşturma parametreleri
(Oracle7 Server SQL Ref. Man.)

Yeni Tablo Alan Adı (Tablespace): Oluşturulacak olan yeni tablo alan adıdır. 2.1.2.5'de ki örnekte yeni tablo adı "tablespace_3" olarak verilmiştir.

Fiziksel Veri Dosyası (Datafile) : Oluşturulacak olan yeni tablo alanları için disk üzerindeki fiziksel veri dosyalarıdır. 2.1.2.5'de ki örnekte fiziksel veri dosyası "C:\ORANT\ORADB\tabspace_file3.dat" olarak verilmiştir.

Otomatik Genişleme (Autoextend) : Veri dosyaları dolduğunda otomatik olarak genişlemesini sağlar.

i. OFF: Nesne alanları dolduğunda alanların büyümesi kullanıcı tarafından yapılacak demektir.
ii. ON : Nesne alanları dolduğunda alanların büyümesi otomatik olarak yapılacak demektir.

2.1.2.5'de ki örnekte otomatik genişleme ayarlanmış ve en fazla genişleme sınırı 10 Megabyte olarak verilmiştir.

2.1.2.4 Örnek

Burada 'tabspace_3' yeni nesne alan adıdır. 'C:\ORANT\ORADB\tabspace_file3.dat' yeni nesne alanına ait fiziksel veri dosyasıdır. Boyu 500 Kilobyte olarak ayarlanmıştır. Boyu her genişlemede 500 Kilobyte genişler ve en fazla 10 Megabyte olacak şekilde ayarlanır.

```
CREATE TABLESPACE tablespace_3
       DATAFILE 'C:\ORANT\ORADB\tabspace_file3.dat' SIZE 500K REUSE
       AUTOEXTEND ON NEXT 500K MAXSIZX 10M
```

2.1.3 Kullanıcıların Oluşturması (CREATE USER)

2.1.3.1 Amaç

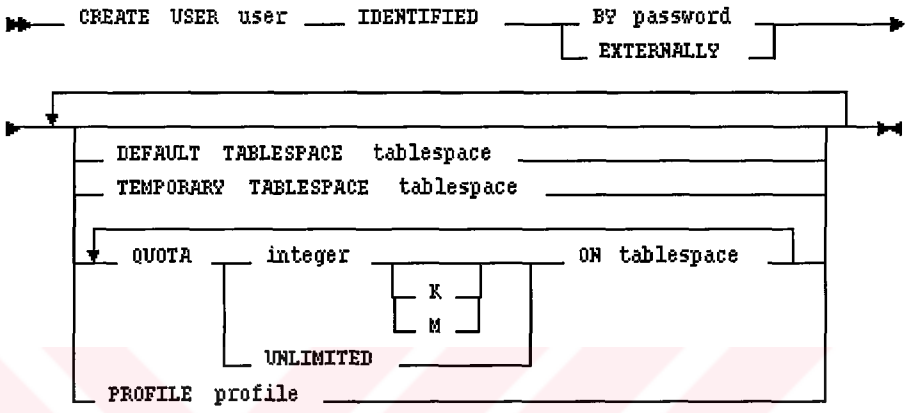
Veritabanı kullanıcısı oluşturmak ve veritabanına erişmek için kullanılır. Kullanıcıya belirli haklar ve yeni tablo oluşturacağı veri dosyalarının belirlenmesi de sağlanır.

2.1.3.2 Ön Gereksinimler

Kullanıcı oluşturma komutunun kullanımı için kullanıcının kullanıcı oluşturma 'CREATE USER' hakkına sahip olması gerekmektedir.

2.1.3.3 Kullanıcı Oluşturma Komutunun Kullanımı

Aşağıdaki şekilde kullanıcı oluşturma komutunun kullanımı ve parametreleri görülmektedir. Yeni kullanıcı oluşturma parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanlar açıklanacaktır.



Şekil 2.3 Kullanıcı oluşturma parametreleri
(Oracle7 Server SQL Ref. Man.)

Kullanıcı Adı (User) : Oluşturulacak olan kullanıcının adıdır. Bu isim veritabanının karakter setinin izin verdiği şekilde verilebilir. 2.1.3.5'deki örnekte kullanıcı adı "deneme" olarak verilmiştir.

Şifre Verme (By) : Kullanıcı şifresi bu parametre ile verilir. Şifre veritabanının karakter setinin izin verdiği şekilde olabilir. 2.1.3.5'deki örnekte "dene" kullanıcı şifresi olarak verilmiştir.

İşletim Sistemi Şifresi (Externally) : Bu parametre işletim sistemi seviyesindeki kullanıcıların veritabanına erişirken işletim sistemi şifrelerinin kullanımını sağlar.

Standart Tablo Alanı (Default Tablespace) : Oluşturulan kullanıcının tabloların oluşturacağı veri dosyasının otomatik tablo alanlarının belirlenmesi sağlanır. 2.1.3.5'deki örnekte "tables01" kullanıcının işlemleri için kullanacağı veri dosyası olarak verilmiştir.

2.1.3.4 Örnek

Bu örnekte 'deneme' kullanıcı adı, 'dene' ise şifresidir. 'tables01' ise bu kullanıcının kullanacağı tablo alanı olarak ayarlanır.

```
CREATE USER deneme
        IDENTIFIED BY dene
        DEFAULT TABLESPACE tables01
```

2.1.4 Tabloların Oluşturması (CREATE TABLE)

2.1.4.1 Amaç

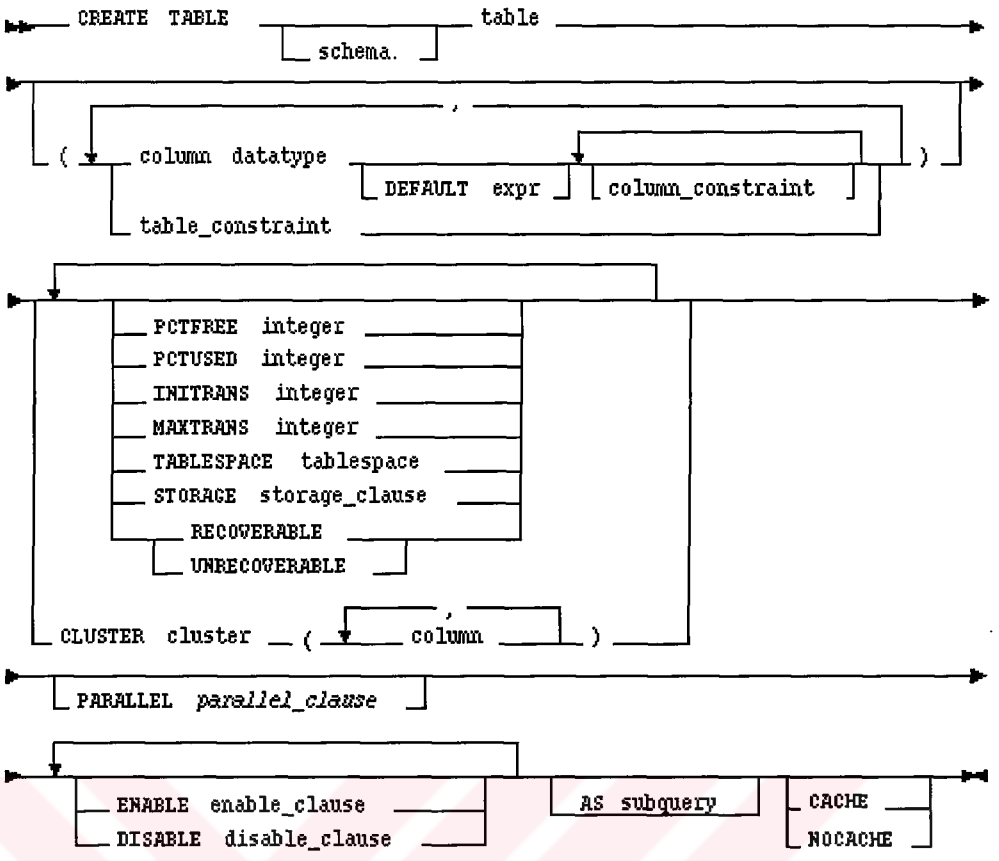
Veritabanı üzerinde tablo nesnesi oluşturmak için kullanılan komut dizisidir.

2.1.4.2 Ön Gereksinimler

Tablo oluşturma komutunun kullanımı için kullanıcının tablo oluşturma 'CREATE TABLE' hakkına sahip olması gerekmektedir. Başka bir kullanıcı şemasında tablo oluşturmak için ise diğer şemalarda tablo oluşturma 'CREATE ANY TABLE' hakkına sahip olması gerekmektedir.

2.1.4.3 Tablo Oluşturma Komutunun Parametreleri

Yeni tablo oluşturma parametrelerinin, kullanımı bir çok şekilde yapılabilmektedir. Komutun bir çok parametresi vardır. Bunlardan en çok kullanılanları açıklanacaktır.



Şekil 2.4 Yeni tablo oluşturma parametreleri

(Oracle7 Server SQL Ref. Man.)

Şema (Schema) : Tabloların hangi şema içerisinde olacağı belirlenir. 2.1.4.5'deki örnekte şema ismi "deneme" dir.

Tablo (Table) : Oluşturulacak olan tablonun adıdır. 2.1.4.5.'de ki örnekte tablo adı "PRST0100" olarak verilmiştir.

Sütun (Column) : Tablolar içerisindeki sütun adlarıdır. Bir tabloda en fazla 254 sütun olabilir.

Veri tipi (Datatype) : Sütunlarda saklanacak olan veri tipidir. Bu veri tipleri aşağıdaki gibidir.

Çizelge 2.1 Veri tipleri

Veri tipi (Datatype)	Açıklama
VARCHAR2(boyut)	Dizi tanımı için kullanılan veri tipidir. İçine yazıldığı veri büyüklüğü kadar veritabanında yer ayırır, en fazla alacağı değer verilmelidir. En küçük değeri 1 ve en büyük değeri 2000 olabilir.
NUMBER(p, s)	Sayı saklamak için kullanılan veri tipidir. Saklanacak olan verinin tam ve ondalıklı kısmı verilebilir. p tam kısmıdır 1 ile 38 arasında, s ondalıklı kısımdır –84 ile 127 arasındadır.
LONG	Karakter veri değişkeni büyüklüğü 2 Gigabyte veya 2^{31}-1 byte'dır.
DATE	Tarih ve saat veri tipini saklamak için kullanılır. MÖ 1 Ocak 4712' den MS 31 Aralık 4712 aralığı saklanabilir.
RAW(boyut)	Binary veri saklama tipidir. En büyük değeri 255'dir.
LONG RAW	RAW binary veri saklama tipinin 2 Gigabyte uzunluğunda olan veri tipidir.
ROWID	Tablodaki sütuna ait tekil adresin saklandığı veri tipidir. Verinin olduğu adresi işaret eder.
CHAR(boyut)	Dizi tanımı için kullanılan veri tipidir. Belirtildiği andaki boyut kadar yer ayrılır. En küçük değeri 1 ve en büyük değeri 255 olabilir.
MLSLABEL	İşletim sistemi isminin binary biçimde saklandığı veri tipidir.

Sütun Kısıtlama (Column Constraint) : Sütunlara bir kısıtlama koymak için kullanılır. Bu kısıtlama birincil anahtarları, sütunların tekilliğini, sütunlarda boş veri olup olmayacağı veya başka tablo üzerindeki alanlarla verinin bütünlüğünü korumak amacıyla kullanılır.

i. **Birincil Anahtar (Primary Key) :** Tablodaki bir veya birkaç sütunun birleşmesiyle oluşturulan birincil anahtardır. Birincil anahtar bileşkesi bir tabloda ancak bir kere bulunabilir ve tekrarlanamaz. Birincil anahtar bileşkesindeki tüm alanlar "not null" yani içerisinde veri olması zorunlu sütunlar olmak zorundadır. Birincil anahtar içinde LONG ve LONG RAW veri tipinde sütun olamaz. Her tabloda ancak bir tane birincil anahtar bileşkesi olabilir. Fakat tekilliğin sağlanması için bazı durumlarda tablonun birden fazla sütunu birleştirilerek, bileşim birincil anahtar olarak tanımlanır. Bu tip bileşimli anahtarlar ayrıca 'Bileşik Anahtar' olarak da isimlendirilir. 2.1.4.5'de ki örnekte Personel_No alanı Birincil anahtar olarak belirlenmiştir.

ii. **Yabancı Anahtar (Foreign Key) :** Tablodaki bir veya birkaç sütunun birleşmesiyle oluşturulan yabancı anahtardır. Yabancı anahtar ana tabloyla ona ait bir detay tablonun ilişkisini yapmak için veya bir tabloyla o tabloda bulunan bazı alanların

tanımının yapıldığı tablolar arasındaki ilişkiyi kurmak için yapılır. Bir başka deyişle bir tablonun diğer tabloyu referans etmesidir. Yabancı Anahtarı içeren tablo, referans eden diğer tablo ise referans edilebilir. Böylece tablolar arası ilişkiler sağlanır.

iii. **Tekil Anahtar (Unique Key) :** Tablodaki bir veya birkaç sütunun birleşmesiyle oluşturulan tekil anahtardır. Tekil anahtar bileşkesi tabloda en az bir kere bulunabilir, buda birincil anahtar üzerindedir. Fakat diğer sütunlarda da tekillik şartı verilebilir. Aynı alanda tekillik bir defa verilir, tekrarlamamayı sağlar. Tekil anahtar içinde LONG ve LONG RAW veri tipinde sütun olamaz.

iv. **Boş Olmayan (Not Null) :** Tabloda belirtilen alanın boş olmayacağının verilmesidir. Bu alan içerisinde bilgi bulunması zorunludur. Yani bu alan hükümsüz olamaz.

v. **Veri Sınırı Koymak (Check) :** Tablodaki bir alana verilecek değerlere bir sınır verme işlemidir. Bu değer sınırı, tablodaki diğer alanların ortak kullanımının bir sonucu hesaplanan bir değerde olabilir.

Tablo Alanı (Tablespace) : Oluşturulacak olan tablonun hangi tablo alanı içinde oluşacağı belirtir. 2.1.4.5'deki örnekte tablo 'tables01' tablespace'i üzerinde oluşturulmaktadır.

2.1.4.4 Örnek

Burada 'PRST0100' oluşturulacak olan tablonun adıdır. personel_no, personel_adi, personel_soyadi, personel_telno, personel_adres tablodaki sütun adları ve veri tipleri de bu alanlarla birlikte verilmektedir. 'pk_prst0100' tablodaki birincil anahtardır. 'tables01' ise tablonun oluşturulduğu alanının adıdır.

```
CREATE TABLE DENEME.PRST0100
       ( personel_no      NUMBER  CONSTRAINT pk_prst0100 PRIMARY KEY,
         personel_adi     VARCHAR2(30),
         personel_soyadi  VARCHAR2(30),
         personel_telno   VARCHAR2(15),
         personel_adres   VARCHAR2(100)
       )
       TABLESPACE tables01
```

2.1.5 Dizinlerin Oluşturması (CREATE INDEX)

2.1.5.1 Amaç

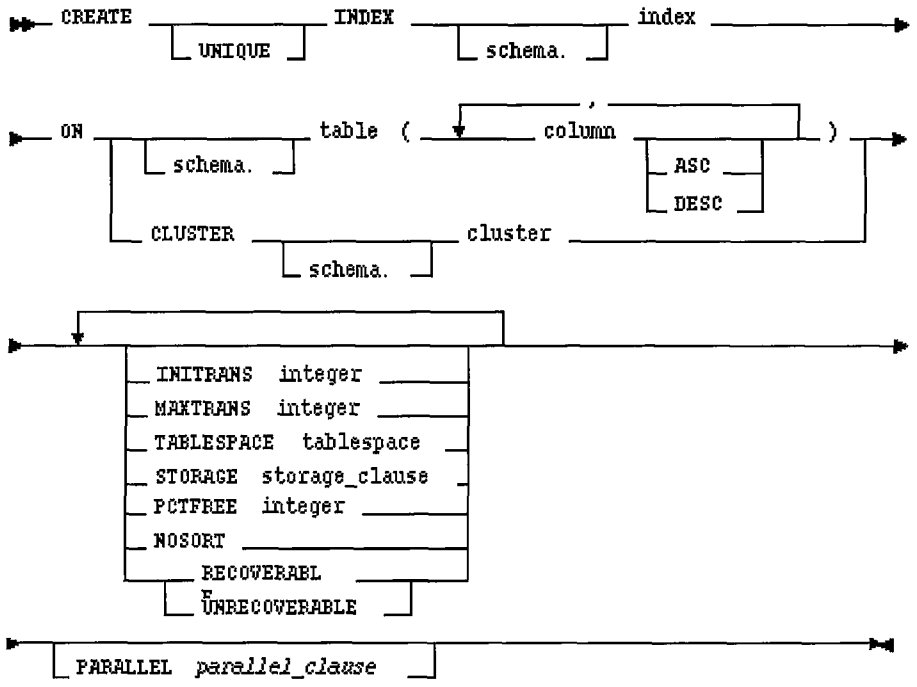
Tablo üzerindeki bir veya birden fazla sütuna dizin koymak için kullanılır.

2.1.5.2 Ön Gereksinimler

Dizin oluşturmak için, eğer kendi oluşturduğumuz tablolar üzerinde yapıyorsak, ekstra bir hakka sahip olmamıza gerek yoktur. Eğer başka bir kullanıcıya ait tablo üzerinde dizin oluşturmak istiyorsak 'CREATE ANY INDEX' hakkına sahip olmamız gerekmektedir.

2.1.5.3 Dizin Oluşturma Komutunun Kullanımı

Aşağıdaki şekilde CREATE INDEX komutunun kullanımı ve parametreleri görülmektedir. Yeni dizin oluşturma parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanlar açıklanacaktır.



Şekil 2.5 Yeni dizin oluşturma parametreleri

(Oracle7 Server SQL Ref. Man.)

Tekillik (Unique) : Oluşturacağımız dizinin sütun veya sütun bileşkesinde verilerinin tekil olup olmayacağını belirten parametredir.

Dizin (Index) : Oluşturulacak olan dizinin adıdır. 2.1.5.5'de ki örnekte dizin adı 'INX_Personel_Adi'dir.

Tablo (Table) : Dizinin hangi tablo için oluşturulacağı verilir. 2.1.5.5'de ki örnekte tablo adı 'PRST0100' olarak verilmiştir.

Sütun (Column) : Tablo içerisindeki sütün adıdır. Bir dizin en fazla 16 sütundan oluşabilir. LONG ve LONG RAW veri tipleri kullanılarak dizin verilemez. 2.1.5.5'de ki örnekte "Personel_Adi" dizinin oluşturulacağı sütun adıdır.

Sıralama Şekli (Asc Desc) : Oluşturulacak olan dizinin küçükten büyüğe mi yoksa büyükten küçüğe mi sıralanacağı belirlenir.

i. ASC : Dizinin küçükten büyüğe sıralanmasını sağlar.

ii. DESC: Dizinin büyükten küçüğe sıralanmasını sağlar.

2.1.5.4 Örnek

'INX_Personel_Adi' oluşturulacak olan dizinin adıdır. 'DENEME' veri tabanı şeması, PRST0100 ise tablo adıdır. Dizin 'Personel_Adi' sütunu üzerinde yapılmaktadır.

```
CREATE INDEX INX_Personel_Adi
      ON DENEME.PRST0100
      (
        Personel_Adi
      )
```

2.2 Veri İşleme Dili (DML - Data Manipulation Language)

DML komutları veriler üzerinde sorgulama, değiştirme, silme ve ekleme işlemleri yapmamızı sağlar. Bu bölümde belirtilen DML işlemleri incelenecektir.

Çizelge 2.2 Prst0100 tablosu veri bilgileri

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000001	Ali	Koşan	2123214435
00000002	Mehmet	Bali	2123323449
00000003	Zeynep	Çingir	2164328784
00000004	Mehmet	Kara	2163334862

Çizelge 2.3 Odmt0001 tablosu veri bilgileri

Hasta No	Ödeme Tutarı
00000001	125.000.000
00000002	542.251.000
00000004	86.000.000

2.2.1 Verilerin Sorgulaması (SELECT)

2.2.1.1 Amaç

Bir veya birkaç tablo veya görüntüden nesnelerin o anki durumlarını sorgulamak için kullanılır.

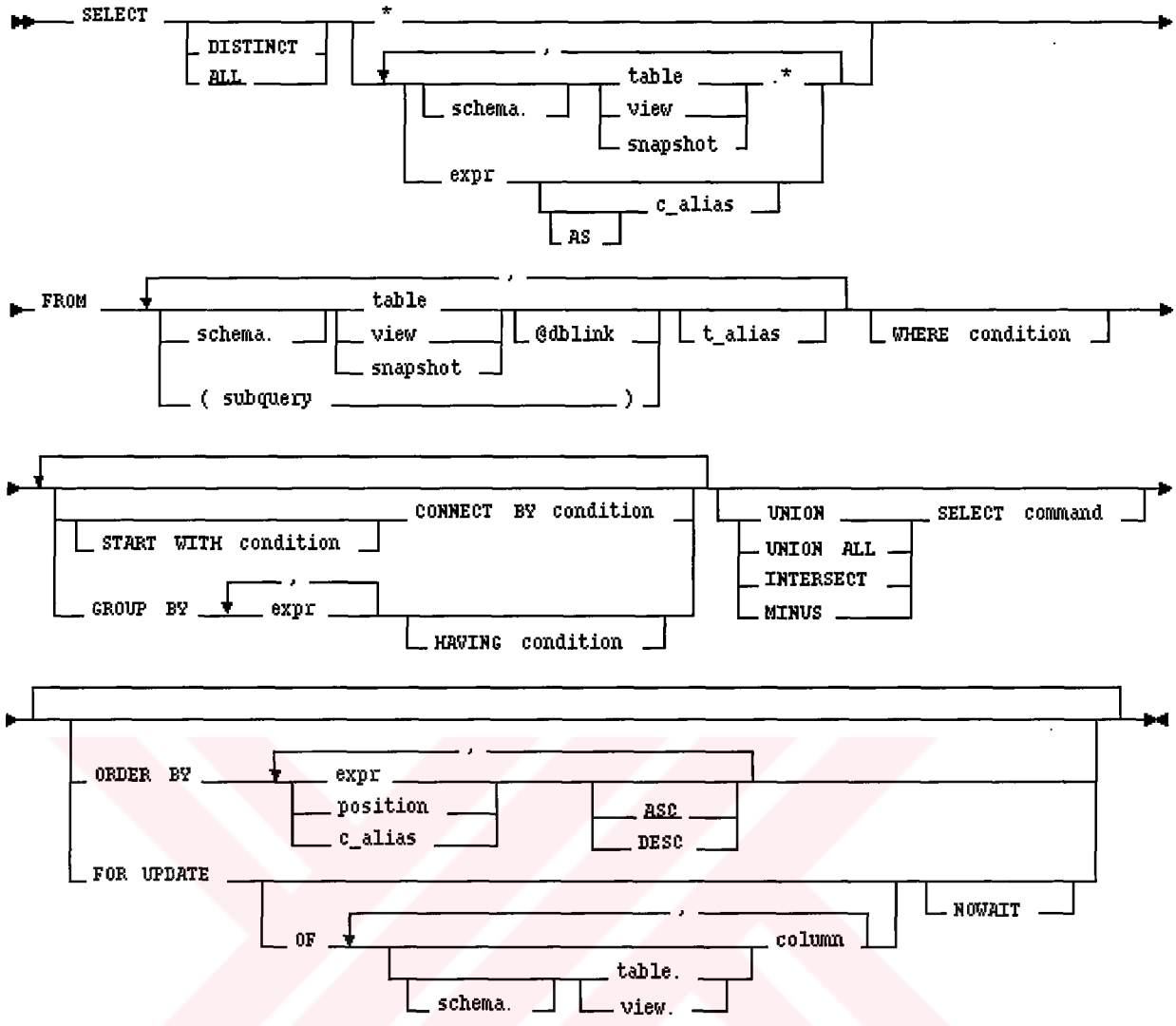
2.2.1.2 Ön Gereksinimler

Kendi şemamızda bulunan tablo veya görüntüden veri seçmek için bu sorgu 'SELECT' hakkına sahip olmak gerekmektedir.

Bir görüntünün tablolarından sütun seçmek için görüntüyü içeren şemaların sahibi tablolarda kullanıcının veri sorgulama hakkına sahip olması gerekir. Görüntü başka bir şema üzerinde ise o görüntü üzerinde veri sorgulama hakkınız olmalıdır. Herhangi bir tablo veya görüntüden veri sorgulamak için 'SELECT ANY TABLE' hakkına sahip olmak gerekmektedir.

2.2.1.3 Veri Sorgulamasının Kullanımı

Aşağıdaki şekilde veri sorgulama komutunun kullanımı ve parametreleri görülmektedir. Veri sorgulama komut parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Tablolardan veri çekerken, verileri istediğimiz gibi görüntüleme, listeleme ve işlem yaptırma şekilleri verir. Bunlardan en çok kullanılanlar açıklanacaktır.



Şekil 2.6 Veri sorgulama parametreleri

(Oracle7 Server SQL Ref. Man.)

Farklılık (Distinct) : Seçilen sütundaki kayıtların tekil olarak getirilmesidir. Yani çift kayıtlı satırların getirilen diğer satırlarla karşılaştırılması ve her kayıttan bir tane getirmemizi sağlar.

Örnek : Aşağıdaki ifadede eğer aynı hasta adından birden fazla varsa sadece ilk bulduğu ada ait olan kaydı getirir. Çizelge 2.2'de aşağıdaki sorgu çalıştırıldığında çizelge 2.4 oluşmaktadır:

```
Select Distinct Hasta_Adi
From Prst0100;
```

Çizelge 2.4 Sütundaki kayıtları tekil getiren sorgu sonucu

Hasta Adı
Ali
Mehmet
Zeynep

Tüm Kayıtlar (All) : Yapılan sorguda tüm kayıtların gelmesi demektir. Yani 'Distinct' komutunun tersidir. Standart kullanım şeklinde 'ALL' parametresi bulunmaktadır.

Örnek: Aşağıdaki ifadede eğer aynı hasta adından birden fazla varsa hepsinin gelmesini sağlar. Çizelge 2.2'de aşağıdaki sorgular çalıştırıldığında çizelge 2.5 oluşmaktadır.

```
Select All Hasta_Adi
 From Prst0100;
```

veya

```
Select Hasta_Adi
 From Prst0100;
```

Çizelge 2.5 Sütundaki tüm kayıtları getiren sorgu sonucu

Hasta Adı
Ali
Mehmet
Zeynep
Mehmet

Tüm Sütunlar (*): Tüm sütunların getirilmesidir. Yapılan sorguda tablodan veya görüntüden tüm sütunların gelmesi demektir. Sütun ismini vermeden veritabanı üzerinde sorgulanan tablo veya görüntüye ait tüm sütunları getirir.

Örnek: Aşağıdaki ifadede *, Prst0100 tablosundaki tüm sütunların getirilmesini sağlar. Çizelge 2.2'de bu sorgu çalıştırıldığında aynı sonuç bulunmaktadır.

```
Select *
 From Prst0100;
```

İsimlendirme (c_alias) : Sütunlara yeniden isim verme veya başlık vermektir. Yapılan sorguda gelen verilerin başlığının değiştirilerek istediğimiz bir isimle gösterilmesidir. Takma isim verilirken AS kelimesi isteğe bağlı olarak verilebilir. Verilen takma isim sorguda sadece gelen verilerin sütun sıralanmasında kullanılabilir.

Örnek: Aşağıdaki ifadede "Hasta_Adi" alanının isimlendirme ile sonucunda başlığın 'Ad' olarak gösterilmesini sağlar. Çizelge 2.2'de bu sorgu çalıştırıldığında, çizelge 2.6 oluşmaktadır.

```
Select Hasta_Adi Ad
 From Prst0100;
```

veya

```
Select Hasta_Adi As Ad
From Prst0100;
```

Çizelge 2.6 Sütun başlığı değiştirilmiş sorgu sonucu

Ad
Ali
Mehmet
Zeynep
Mehmet

Şema (Schema) : Şema demektir. Tablo, görüntü ve diğer nesnelerin bulunduğu yerdir.

Örnek: Aşağıdaki ifadede 'Ytu' şeması altında Prst0100 tablosunun seçilmesini sağlar. Çizelge 2.2'de bu sorgu çalıştırıldığında, tüm tablo getirildiği için sonuç çizelge 2.2'de bulunmaktadır.

```
Select *
 From Ytu.PRST0100;
```

Tablo veya Görüntü (Table or View) : Verilerin sorgulanacağı tablo veya görüntü adıdır.

Örnek: Aşağıdaki ifade Prst0100 tablosunun gösterilmesi sağlar. Çizelge 2.2'de bu sorgu çalıştırıldığında tüm tablo getirildiği için sonuç çizelge 2.2 bulunmaktadır.

```
Select *
 From Prst0100;
```

Başka Veritabanı Bağlantısı (Dblink) : Uzaktaki bir veri tabanında bulunan nesnelere yapılan veritabanı bağlantısının ismidir. Eğer dblink düşürülürse nesnelerin lokal veritabanında olduğu kabul edilir.

İç İçe DML Kullanımı (Subquery) : İstenen sorgudaki bir tablonun sütunu için sorgu kullanılarak çıkan değerleri tek tek veya bir grup olarak sütunlarla eşleştirmektir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda bulunan kayıtlar ile Odmt0001 hasta ödeme tablosunda var olan kayıtlarda ortak olanlar getirilmek istenmektedir. Çizelge 2.2'de bu komut çalıştırıldığında sonuç çizelge 2.7'de verilmektedir.

```
Select Hasta_No, Hasta_Ev_Tel
 From Prst0100
Where Hasta_No in (Select Hasta_No
                   From Odmt0001);
```

Çizelge 2.7 Sorgulamada subquery kullanılmış sorgu sonucu

Hasta No	Hasta Ev Tel
00000001	2123214435
00000002	2123323449
00000004	2163334862

Tablo İsimlendirme (T_Alias) : Tablo veya görüntülere yeniden isim veya takma isim vermektir. Genellikle sorgularda kullanılan alanlara verilen takma isimdir. Sorgularda bir yerde tanımlanan takma isim o sorgu içerisinde her yerde kullanılabilir.

Örnek: Aşağıdaki ifadede Prst0100 tablosuna 'p' tanımlaması yapılarak kullanılabilir. Çizelge 2.2'de bu sorgu çalıştırıldığında sonuç çizelge 2.2 gibi elde edilmektedir.

```
Select p.Hasta_No, p.Hasta_Adi, p.Hasta_Soyad, p.Hasta_Ev_Tel
From Prst0100 p;
```

Koşul (Where) : Sorguda kayıtların verilen koşul veya koşullarla sınırlandırılarak getirilmesidir. Verilen koşul doğru ise verileri getirir. Eğer bir koşul verilmezse tablo veya görüntü üzerinde tüm kayıtlar getirilir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda adı 'Mehmet' olan kayıtların getirilmesi sağlanır. Çizelge 2.2'de bu sorgu çalıştırıldığında sonuç çizelge 2.8 elde edilmektedir.

```
Select *
From Prst0100
Where Hasta_Adi = 'Mehmet';
```

Çizelge 2.8 Koşul kullanılmış sorgu sonucu

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000002	Mehmet	Bali	2123323449
00000004	Mehmet	Kara	2163334862

Gruplama (Group By) : İstenen kayıtların seçilen sütundaki değerlere göre gruplanarak getirilmesidir. Gruplama yapılırken grup içerisinde istenilen hesaplamalar yapılabilir.

Örnek: Aşağıdaki ifade Prst0100 tablosunda aynı isme sahip olan kayıtları gruplayarak gösterir ve Count(*) komutuyla kaçar tane olduğu da gösterilebilir. Çizelge 2.2'de bu sorgu çalıştırıldığındaki sonuç çizelge 2.9'da verilmiştir.

```
Select Hasta_Adi, Count(*) Adet
From Prst0100
Group By Hasta_Adi;
```

Çizelge 2.9 Grup yapılmış sorgu sonucu

Hasta Adı	Adet
Ali	1
Mehmet	2
Zeynep	1

Gruplar Üzerindeki Koşul (Having) : Sorguda gruplanmış kayıtların istenilen şekilde sınırlandırılmasıdır. İstenen koşul doğru ise verileri getirir. Buradaki amaç gruplamada ki hesaplamalar üzerinde sınırlama yapmaktır. Eğer bir koşul verilmezse tablo veya görüntü üzerinde tüm kayıtlar guplanarak getirilir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda ismi en az 2 defa geçen hasta isimlerini gruplanarak sayıları birlikte getirir. Çizelge 2.2'de bu sorgu çalıştırıldığında sonuç çizelge 2.10'da verilmiştir.

```
Select Hasta_Adi, Count(*)
From Prst0100
Group By Hasta_Adi
Having Count(*) >= 2;
```

Çizelge 2.10 Grup koşulu kullanılmış sorgu sonucu

Hasta Adı	Adet
Mehmet	2

Sıralama (Order By) : Yapılan sorguda gelen kayıtların istenilen sütuna göre sıralanarak getirilmesidir. ASC ve DESC sıralama şekillerini belirler. Eğer sıralama şekli belirtilmezse ASC olarak standart şekli otomatik olarak alınır.

Sıralama şekli:

i. ASC, küçükten büyüğe sıralar,

ii. DESC, büyükten küçüğe sıralar.

Örnek: Aşağıdaki ifadede Prst0100 tablosu getirilirken, Hasta_Adi sütununa göre alfabetik olarak sıralayarak getirir. Çizelge 2.2'de hasta adına göre büyükten küçüğe sıralama sorgusu çalıştırıldığında çizelge 2.11 sonucu elde edilir.

```
Select *
From Prst0100
Order By Hasta_Adi Desc;
```

Çizelge 2.11 Sıralama yapılmış sorgu sonucu

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000003	Zeynep	Çingir	2164328784
00000002	Mehmet	Bali	2123323449
00000004	Mehmet	Kara	2163334862
00000001	Ali	Koşan	2123214435

Kayıt Kilitleme (For Update) : Seçilen kayıtları başka kullanıcıların değişimine kapatılmasıdır. Bu işlem başka kullanıcıların seçilen tablodaki kayıt üzerinde sorgu yapma dışındaki işlemlerini durdurur.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda Hasta_No = '00000001' olan hasta kaydının bilgilerini sadece bizim güncelleyebilmemiz için seçmemiz sağlanır.

```
Select Hasta_Adi
From Prst0100
Where Hasta_No = '00000001'
For Update Prst0100;
```

Tablo veya Görüntüleri Birleştirmek (Join) : Tablolar veya görüntüler arasında sütun veya sütunlarla ilişki kurulmasıdır. Kurulan ilişkiler verilerin doğru gelmesi açısından çok önemlidir. İlişki genellikle ana nesneden detay nesneye veya bir nesneden bir tanım nesnesine yapılır ve veri getirimi gerçekleşir. Tablolarda ki nesnelere isimler takılarakda ilişkileri kullanabiliriz.

Örnek: Aşağıdaki ifade çizelge 2.2'de bulunan Prst0100 hasta tanım tablosu ile çizelge 2.10'da bulunan Odmt0101 hasta ödeme tablosu arasında ilişki kurarak ödeme yapacak olan hastaların hasta numaraları, adı, soyadı, telefon numarası ve ödeme tutarlarını görmek için aşağıdaki sorgu oluşturulabilir.

```
Select p.Hasta_No, p.Hasta_Adi, p.Hasta_Soyad, p.Hasta_Ev_Tel, o.Odeme_Tutar
    From Prst100 p, Odmt0101 o
    Where p.Hasta_No = o.Hasta_No;
```

Çizelge 2.12 Prtst0100 ve Odmt0101 tablolarının join sorgu sonucu

Hasta No	Ödeme Tutar
00000001	50,000,000
00000003	78,000,000
00000004	112,500,000

2.2.2 Verilerin Değiştirmesi (UPDATE)

2.2.2.1 Amaç

Tablo veya temel tablo görüntüsü içerisindeki verilerin değiştirilmesidir, başka bir deyişle güncelleştirmenin yapılamasıdır.

2.2.2.2 Ön Gereksinimler

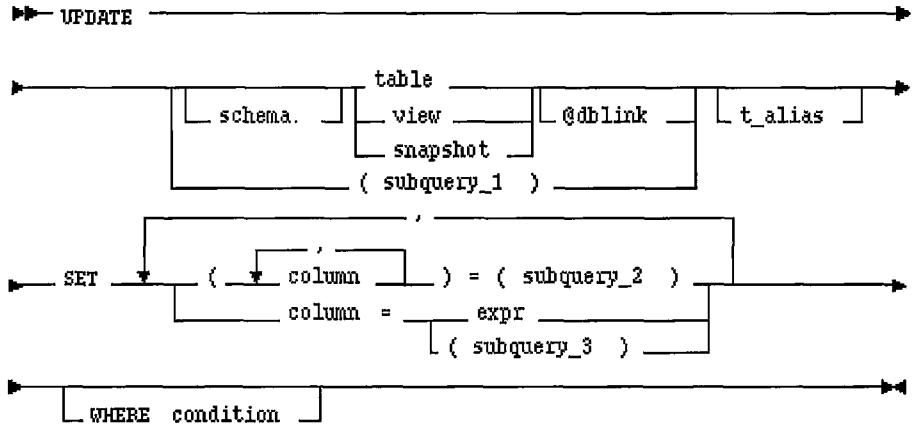
Şemamızda veya başka bir şemada bulunan tablo veya temel tablo görüntüsü üzerinde değişiklik yapma hakkına sahip olmamız yeterlidir. Kendi şemamızda bulunan tablo veya temel tablo görüntüsünde değişiklik yapmak için 'UPDATE' hakkına sahip olmak gerekmektedir.

Herhangi bir tablo veya görüntüde veri değiştirmek için 'UPDATE ANY TABLE' hakkına sahip olmak gerekmektedir.

Tablo veya temel tabloya ait bir görüntü verileri üzerinde değişiklik yaptığımızda bizim için bir işlem başlar, değişiklik yapılan nesne kilitlenir ve diğer kullanıcıların değiştirilmesine kapatılmış olur, işlem bitirene kadar diğer kullanıcılar bu değişiklileri göremez, işlemi bitirdiğimiz anda diğer kullanıcılar bu değişiklikleri görür ve nesne üzerindeki kilit kalkar.

2.2.2.3 Veri Değiştirme Komutunun Kullanımı

Aşağıdaki şekilde veri değiştirme komutunun kullanımı ve parametreleri görülmektedir. Veri değiştirme komut parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanları açıklanacaktır.



Şekil 2.7 Veri değiştirme parametreleri

(Oracle7 Server SQL Ref. Man.)

Değişiklik Alanı (Set) : Değişiklik yapılacak sütunları ve değişiklikleri belirtir. Sütunun alacağı değer verilir. Birden fazla sütuna aynı anda farklı değerler verilebilir. Değer olarak sabit bir değer veya bir sorgu yapılarak da sonuç değerleri ile değişiklik yapılabilir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda ki Hasta_No = '00000001' koşulunu karşılayan kayıdı Hasta_Adi = 'Zehra' olarak değiştirelim. Çizelge 2.2'de bu sorgu çalıştırıldığında çizelge 2.13 oluşmaktadır.

```
Update Prst0100
   Set Hasta_Adi = 'Zehra'
Where Hasta_No = '00000001';
```

Çizelge 2.13 Tablo veri değişikliği sonucu

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000001	Zehra	Koşan	2123214435
00000002	Mehmet	Bali	2123323449
00000003	Zeynep	Çingir	2164328784
00000004	Mehmet	Kara	2163334862

Koşul (Where) : Update ifadesinde değiştirilecek kayıtların istenilen şartı veya şartların verildiği kısımdır. İstenen koşul doğru ise verileri değiştirir. Eğer bir koşul verilmezse tablo veya görüntü üzerinde tüm kayıtlar değiştirir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda Hasta_No = '00000001' koşulunu karşılayan olan kayıtlarda hasta soyadı alanını 'Öztürk' ve hasta ev telefonunu '2123322122' olarak değiştirelim. Çizelge 2.2'de bu sorgu çalıştırıldığında çizelge 2.14 oluşmaktadır.

```
Update Prst0100
    Set Hasta_Soyadi = 'Öztürk', Hasta_Ev_Tel = '2123332122'
 Where Hasta_No = '00000001';
```

Çizelge 2.14 Tabloda birden fazla sütunun veri değişikliği

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000001	Ali	Öztürk	2123332122
00000002	Mehmet	Bali	2123323449
00000003	Zeynep	Çingir	2164328784
00000004	Mehmet	Kara	2163334862

İç İçe DML Kullanımı (Subquery) : Veri değiştirilecek olan tablo veya temel tabloya ait bir görüntünün sütun değerini bir sorgu kullanılarak çıkan değerleri tek tek veya bir grup olarak sütunların yeni değeri olarak verilmesidir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda bulunan hasta telefonları Prst0002(Hasta_No#, Tel1, Tel2, Tel3) hasta telefonları tablosundaki 'Tel1' alanı ile değiştirilmek istemektedir. Bunun için Prst0100 tablosunda bulunan hasta no alanı Prst0002 tablosundaki hasta no alanları arasında ilişki kurularak veri aktarımı yapılır.

```
Update prst0100 p1
    Set Hasta_Ev_Tel = (Select Tel1
                        From Prst0001 p2
                       Where p1.Hasta_No = p2.Hasta_No);
```

2.2.3 Verilerin Silinmesi (DELETE)

2.2.3.1 Amaç

Tablo veya temel tablo görüntüsü içerisindeki verilerin kayıt bazında silinmesidir.

2.2.3.2 Ön Gereksinimler

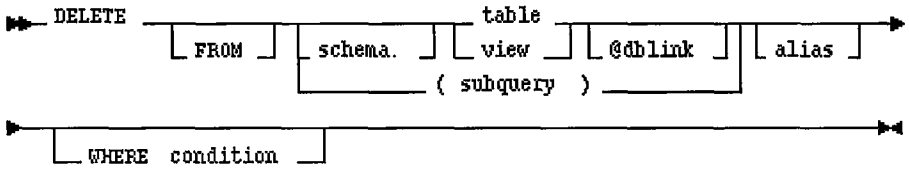
Şemamızda veya başka bir şemada bulunan tablo veya temel tablo görüntüsü üzerinde kayıt silme hakkına sahip olmamız yeterlidir. Başka bir deyişle 'DELETE' hakkına sahip olmamız gerekmektedir.

Herhangi bir tablo veya görüntüde kayıt silmek için 'DELETE ANY TABLE' hakkına sahip olmak gerekmektedir.

Tablo veya temel tabloya ait bir görüntü kayıtları üzerinde silme yaptığımızda bizim için bir işlem başlar, silme yapılan nesne kilitlenir ve diğer kullanıcıların değiştirilmesine kapatılmış olur, işlem bitirene kadar diğer kullanıcılar bu değişiklileri göremez, işlemi bitirdiğimiz anda diğer kullanıcılar bu değişiklikleri görür ve bu nesne üzerindeki kilit kalkar.

2.2.3.3 Veri Silme Komutunun Kullanılması

Aşağıdaki şekilde komut silme komutunun kullanımı ve parametreleri görülmektedir. Veri silme komutunun parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanları açıklanacaktır.



Şekil 2.8 Veri silme parametreleri

(Oracle7 Server SQL Ref. Man.)

Tablo Adı için (From) : Bu parametre işlem yapılacak tablo adının verilmesi için kullanılır. Tablodan bir veya birden fazla kayıt aynı anda silinebilir. Koşul olarak sabit bir sütun değeri veya bir sorgu sonucu verilebilir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda Hasta_No = '00000001' koşulunu karşılayacak olan kayıtlar siliniyor. Çizelge 2.2'de bu sorgu çalıştırıldığında çizelge 2.15 oluşmaktadır.

```
Delete Prst0100
Where Hasta_No = '00000001';

veya

Delete From Prst0100
Where Hasta_No = '00000001';

veya

Delete (Select *
        From Prst0100)
Where Hasta_No = '00000001';
```

Çizelge 2.15 Tabloda koşula göre kayıt silme sonucu

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000002	Mehmet	Bali	2123323449
00000003	Zeynep	Çingir	2164328784
00000004	Mehmet	Kara	2163334862

İç İçe DML Kullanımı (Subquery) : Veri silinecek olan tablo veya temel tabloya ait bir görüntünün koşulu içerisinde bir sorgu kullanılmasıdır.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda bulunan Prst1000(Veri modeli Prst0100 tablosu gibidir.) tablosunda hasta numarası '00000100' den büyük olan kayıtlarda hasta adı aynı olan kayıtları siler.

```
Delete Prst0100
 Where Hasta_Adi in (Select Hasta_Adi
                     From Prst1000
                     Where Hasta_No > '00000100');
```

2.2.4 Verilerin Eklenmesi (INSERT INTO)

2.2.4.1 Amaç

Tablo veya temel tablo görüntüsü içerisine yeni veri eklenmesidir.

2.2.4.2 Ön Gereksinimler

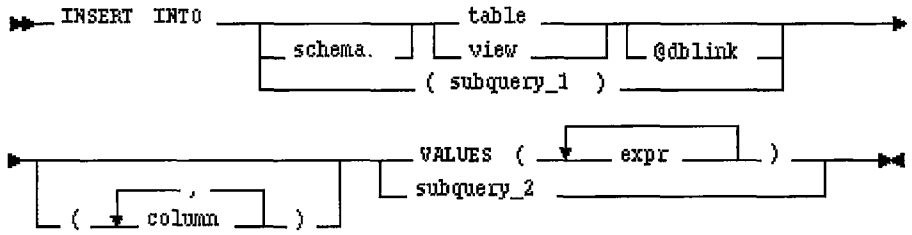
Şemamızda veya başka bir şemada bulunan tablo veya temel tablo görüntüsü üzerinde kayıt ekleme yapma hakkına sahip olmamız yeterlidir. Kendi şemamızda bulunan tablo veya temel tablo görüntüsünde kayıt eklemek için 'INSERT' hakkına sahip olmamız gerekmektedir.

Herhangi bir tablo veya görüntüde kayıt eklemek için 'INSERT ANY TABLE' hakkına sahip olmak gerekmektedir.

Tablo veya temel tabloya ait bir görüntüye kayıt ekleme yaptığımızda bizim için bir işlem başlar, ekleme yapılan nesne kilitlenir ve diğer kullanıcıların değiştirilmesine kapatılmış olur, işlem bitirene kadar diğer kullanıcılar bu değişiklileri göremez, işlem bittiği anda diğer kullanıcılar bu değişiklikleri görür ve bu nesne üzerindeki kilit kalkar.

2.2.4.3 Veri Ekleme Komutunun Kullanımı

Aşağıdaki şekilde veri ekleme komutunun kullanımı ve parametreleri görülmektedir. Veri ekleme komut parametrelerinin kullanımı bir çok şekilde yapılabilmektedir. Bunlardan en çok kullanılanlar açıklanacaktır.



Şekil 2.9 Veri ekleme parametreleri

(Oracle7 Server SQL Ref. Man.)

Değerler (Values) : Eklenecek olan kayıtın verilerinin belirtilmesidir.

Örnek: Aşağıdaki ifadede Prst0100 tablosunda hasta no su '00001111' adı 'Ayşe' soyadı 'Öztürk' olan bir hasta kaydı eklenmek isteniyor. Çizelge 2.2'de bu sorgu çalıştırıldığında çizelge 2.16 oluşmaktadır

```
Insert Into Prst0100 (Hasta_No, Hasta_Adi, Hasta_Soyad)
        Values ('00001111', 'Ayşe', 'Özden');
```

veya

```
Insert Into  (Select Hasta_No, Hasta_Adi, Hasta_Soyad From Prst0100)
        Values ('00001111', 'Ayşe', 'Özden');
```

Çizelge 2.16 Tabloya yeni kayıt ekleme sonucu

Hasta No	Hasta Adı	Hasta Soyadı	Hasta Ev Tel
00000001	Ali	Koşan	2123214435
00000002	Mehmet	Bali	2123323449
00000003	Zeynep	Çingir	2164328784
00000004	Mehmet	Kara	2163334862
00001111	Ayşe	Özden	

Burada kayıt eklenecek olan nesnedeki sütun isimleri verilmeden de yeni kayıt eklenebilir. Bu işlem için nesneye ait olan sütunlarının hepsi için bir değer belirtmek gerekir. Değer verilmeyecek sütun için hükümsüz(NULL) değeri verilmelidir.

```
Insert Into Prst0100
        Values ('00001111', 'Ayşe', 'Öztürk', Null);
```

İç İçe DML Kullanımı (Subquery) : Veri eklenecek olan tablo veya temel tabloya ait bir görüntünün koşulu içerisinde bir sorgunun sonuç değerlerinin kullanılmasıdır.

Örnek: Bu örnekte Prst0100 tablosuna Prst1000 Prst1000(Veri modeli Prst0100 tablosu gibidir.) tablosundan hasta no '00001000' olan kayıt aynen eklemektir.

```
Insert Into Prst0100
    Select *
     From Prst1000
    Where Hasta_No = '00001000';
```

3 ÖĞRENEN SİSTEMLER

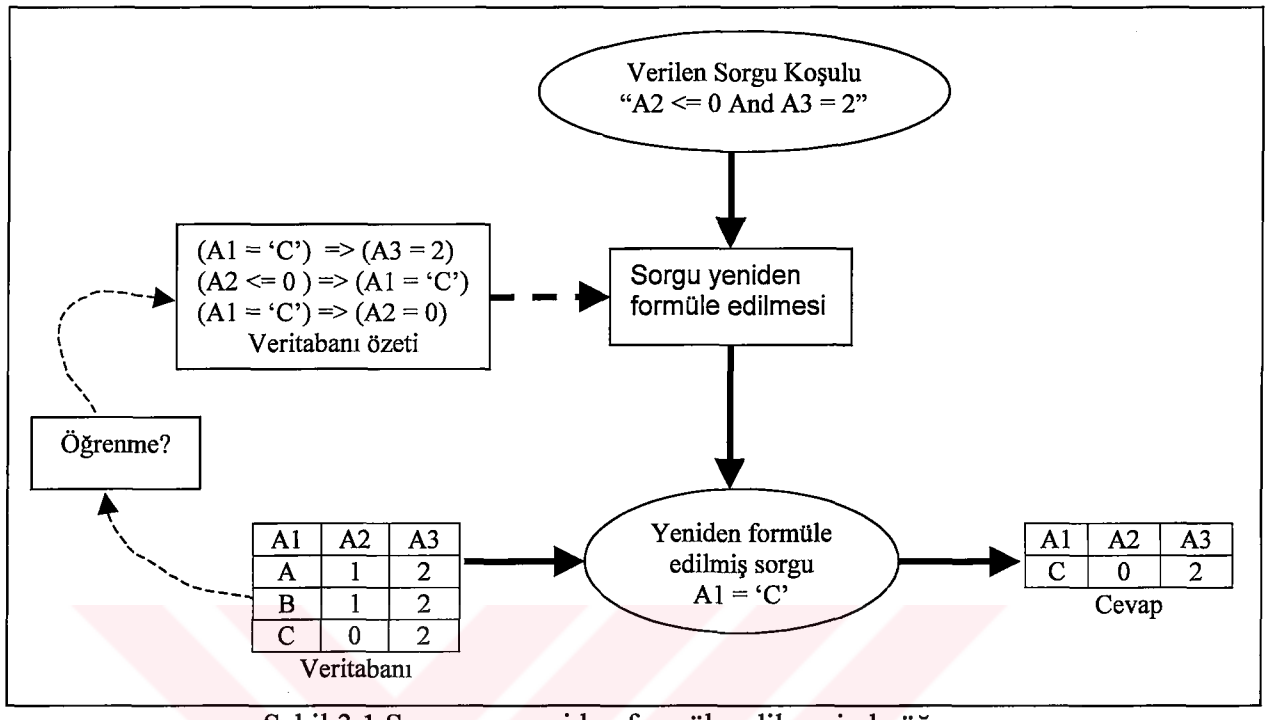
Bu bölümde SQL tabanlı yapılarda bulunan öğrenme metotları anlatılacaktır. SQO yaklaşımının, daha performanslı çalışması için sorgu iyileştirmesi yapılır. Bu işlem verilen sorguda bulunan koşula göre gerçekleştirilir. Sorgunun koşulu alınır ve sorgu sonuç setiyle öğrenme metotlarından biri kullanılarak sonuç verileriyle kurallar türetilir. Bu öğrenilen kurallar araştırmada DML ifadelerine uygulanır.

3.1 Bilgiden Bilgi Öğrenme

Sorgu iyileştirme yöntemleri, veri modelleri ve dillerinin (Ullman 88) ortaya çıkışından beri üzerinde çalışılan bir konudur. Çünkü genellikle, sorguların verimli uygulanması zordur. Sorgunun yeniden formüle edilmesi yaklaşımı, aynı zamanda anlamsal sorgu iyileştirme yaklaşımı (SQO) olarak da bilinir. İyileştirme yaklaşımına geleneksel sözdizimsel yaklaşımdan daha farklı yaklaşılır, şöyle ki sorguları iyileştirmek için veritabanlarının içerikleri hakkında daha zengin bir bilgi kümesi oluşumunu gerektirir. Anlamsal bilginin kullanımı maliyet indirimi sağlar.verilerden elde edilen sorgunun sonuç ve koşulu birlikte kullanılarak kurallar üretilir. Üretilen bu kurallar yeni sorgulara uygulanır. Kuralların içerdiği özelliklere göre çalışma performansı artırılır. Bu nedenle SQL, içinde bilgiden bilgi öğrenme sistemini içerir.

Şekil 3.1, sorgu işleme sisteminde sorgunun kurallar kullanılarak yeniden formüle edilmesini basitçe örneklemektedir. 3 örnekli bir veritabanımız olsun ve her bir örneğin 3 sütunu olsun. Bunlar: A1, A2 ve A3'dür. Bunlardan sadece A1 dizinlenmiştir. Daha önceden elde edilmiş veritabanı özetlerinin bir kümesi olsun. Veritabanı sistemimizde iki kısıtlamalı (constraint) bir sorgumuz olduğunu varsayalım. Sorgunun yeniden formüle edilmesi, sorgu işlenmesinin maliyetini düşürmek için verilen sorgunun daha ucuz ama öncekine denk olacak şekilde yeniden formüle edilmesi için veritabanı özetlerinin kullanılmasıdır. Bu örnekte, sorgunun yeniden formüle edilmesi verilen sorguyu tek kısıtlamalı bir sorguya dönüştürür. Değiştirilen koşul dizinlenmiş bir sütun üzerindedir. Böylece daha performanslı ve aynı kayıtları belirten koşul oluşturulmuş olur. Dizinlenmiş sütun üzerine bir koşul eklemenin dışında, sorgunun maliyetini yeniden formüle ederek azaltmak için başka bir çok yol vardır. Örneğin sorgunun

hükümsüz (boş küme) döndürebileceğini kurallardan bularak yapmak, veritabanına erişmeden gerçekleşeceği için performans kazancını en yüksek kılar.



Şekil 3.1 Sorgunun yeniden formüle edilmesinde öğrenme

Bu yaklaşımı sağlamak için yeniden formüle etmede çok verimli bir algoritmaya ihtiyacımız vardır. Bu konuyla ilgili verimli bir yeniden formüle etme algoritması geliştirilmiştir (Hsu ve Knoblock 1993). Şekil 3.1'deki kesikli çizgilerde de belirtildiği gibi, veritabanı özetlerini otomatik olarak öğrenen bir sistem istenilmektedir. Bu araştırmada, veri temelli ve sorguya dayalı bilgiden bilgi öğrenme ve öğrenilen bilgilerin (kuralların) yeni sorgular için kullanımı çeşitli yaklaşımlarla yapılabileceği anlatılmaktadır.

Yeniden formüle etmek için otomatik kural türetme fikri (Siegel 1988) tarafından önerilmiştir. Siegel'in yaklaşımında her ne kadar örnek sorgular kullanıyorsa da öğrenme aslında sabit bir Heuristic'ler kümesinden yürütülür. Heuristic'ler veritabanı yapısına ve uygulamasına göre tasarlanırlar. Bizim yaklaşımımız diğerlerinden şu yönden ayrılır; bizim yöntemimiz net Heuristic'lere dayanmaz. Örnek sorgular ve yeniden formüle etmede gerekli kuralları belirlemek için bunlarla elde ettiğimiz verilere odaklanmıştır. Bizim yaklaşımımızın amacı, veritabanı yapısı ve uygulamasına bağımlılığın en aza indirilmesi ve daha çok sorguların anlamsal yönlerini bilgilerle değiştirip yeniden formüle etme gerekliliklerine dayanır.

Sonuç olarak elimizdeki kurallardan yeni kurallar çıkarabilme ve bu kuralları kullanabilme performans için çok etkilidir. Kurallardan kural öğrenme işleminde oluşan kuralın veritabanı üzerinde doğruluğunun kesin olması gerekir, bu işlem için veritabanına erişerek veya elimizdeki kuralların veritabanı üzerinde kaç tane satırda bulunduğu bilgisini de kullanabiliriz.

Örneğin : Elimizde aşağıdaki gibi iki kural olsun

R1) Hasta_No = '00000053' --> Hasta_Ad = 'Zeki'
R1) Hasta_Tel = '3125245214' --> Hasta_Ad = 'Zeki'

Burada hasta numarası '00000053' olan hastanın adı 'Zeki' ve veritabanında 'Zeki' ismi tek ise hasta numarası '00000053' olan hastanın telefonu '3125245214' olarak bulunur.

R3) Hasta_No = '00000053' --> Hasta_Tel = '3125245214'

Bu şekilde R1 ve R2 kurallarından R3 kuralı elde edilmiş olur.

3.2 Öğrenme Metotları

SQO yaklaşımında, daha iyi bir sorgu uygulanması için, sorgu iyileştirme yapılır ve bunun için kurallar kullanılır. Bununla birlikte kurallar sorgu çalıştırılmadan önce türetilmelidirler. Kural türetmede ki ana düşünce, o anki sorgu ve verilen veritabanına bağlı olarak kuralları otomatik olarak öğrenmektedir. Kural türetmede, SQO yaklaşımının performansını etkileyen bir faktördür. Eğer bir kural türetilmediyse, sorgu iyileştirmede kullanılacak bir kural olmayacaktır ve iyileştirme yapılamayacaktır. Bu nedenle kuralların öğrenilmesi SQO'da çok önemlidir ve bir çok araştırmacı tarafından sorgu iyileştirmesinde kural kullanımın avantajları incelendiği gibi öğrenmede araştırılmıştır. (Hsu ve Knoblock 1993; Sayli ve Lowden 1996, 1997a-b).

3.2.1 Siegel Yöntemi

(Siegel 1988) tarafından, SQL yaklaşımına otomatik kural türetme için temel bir yöntem olarak sunulmuştur. Bu yöntem iki bölümlü bir işlem olarak görülebilir. İlk bölüm kuralların karakteristiklerini belirler. İkinci bölümde ise bu karakteristiklere sahip yeni kuralları türetmek için veritabanından sorgulama yapılır. Bu bölümler SQO yaklaşımında yararlı kurallar bulunmasını sağlar. Bu yöntemle öncelikle bir 'önerilen kural' fikri oluşturulur ve önerilen kuralın önceki durumu, sorgunun herhangi bir durumunu alarak oluşturulur. Sonuç sütunu, heuristics tarafından bulunur ve sonra yeni bir kural türetmek için veritabanına girilerek uygun bir koşul bulunabilir. Bu yöntem dört modülde açıklanmıştır: kural karakteristiklerini tanımlama, önerilen kuralların seçimi, sorgu üretilmesi ve kural yönetimidir. Bunlar bu bölümde açıklanmaktadır.

3.2.1.1 Kural Karakteristiklerini Tanımlama

Kural karakteristiklerini tanımlama yöntemin ilk modülüdür. Bu modülün amacı, işe yarar kurallar türetmek için sorgu iyileştiricisi tarafından kural türetme işleminin kontrol edilmesidir. Kuralları var olan kurallardan veya doğrudan veritabanından türetebilmek mümkündür. Yinede, eğer türetme işlemi sorgu iyileştiricisine bağlı değilse bu işlem SQO yaklaşımı için gereksiz kurallar bulmamıza neden olabilir. Bu modülde, SQO yaklaşımı beklenen maliyet tasarrufu ile birlikte bütün uyumsuz önerilen kuralları, kural türetme modülüne gönderir ve sonra bu modül tarafından incelenen önerilen kurallar, hesaplanan potansiyel maliyet tasarrufları ile birlikte önerilen kural listesine girer. Eğer bu kural listede mevcutsa birikmiş potansiyel maliyet tasarrufu bu önerilen kural için hesaplanır. Bundan sonra, belli bir eşiğin altındaki birikmiş potansiyel maliyet tasarruflu önerilen kurallar, önerilen kural listesinden silinmelidir. Aksi takdirde, bu önerilen kurallar yeni kurallar gibi algılanır.

3.2.1.2 Önerilen Kuralları Seçme

Yöntemin ikinci modülü önerilen kuralları seçmektir. Bu modül türetilebilen en iyi kuralları belirlemede kullanılır. Önerilen kuralların belirlenmesi uzmanlar için çok zor bir iştir. Bu

sebeple, gittikçe artan beklenen maliyet tasarruflarını hesaplama, önceki tecrübelere dayandırılır. Potansiyel değeri yeterince yüksek olan uyumlu önerilen kurallardan biri listede yer aldığında, önerilen kurallar işlem için seçilebilir. Bu işlem boyunca, daha düşük potansiyel değeri var olan kurallar, kurallar kümesinden silinecektir. Bu işlem kurallar kümesinin büyüklüğü ve kalitesinin belirlenmesinde kullanılabilir. Aynı amaçla, kural kümesinin en fazla değerine sınır koymak ve bir kuralın en az değerine sınır koymakta yararlı olabilir. Eğer kurallar bu en az değerin altında bir değere sahipse kurallar kümesinden silinecektir.

3.2.1.3 Sorgu Üretme

Sorgu üretme yöntemin üçüncü modülüdür. Bu modül, ikinci modül tarafından seçilen bir önerilen kural alır. Temel olarak, modül önerilen kuralı kullanarak bir sorgu şablonu üretir. Bundan sonra veritabanı üzerinde sorgu çalıştırılır. Bu işlem boyunca ortaya çıkabilecek birkaç durum söz konusudur:

i. Eğer bir önerilen kural bir koşulun çıkarılması ile bulunduysa sonucu ile ilgili bir koşul olacaktır. Bu koşul üretilen sorguda kullanılmaz.

ii. ii) Eğer önerilen kuralda, iki nesne ilişkisi arasında bir bağlantı her hangi bir sütun ile geliyorsa, bu ilişki koşulu kural oluşturulmasında kullanılmaz.

3.2.1.4 Kural Yönetimi

Kural yönetimi yöntemin son modülüdür. Bu modül, kurallar kümesine yeni bir kuralın eklenip eklenmeyeceğine karar verir. Modülde, türetilmiş kural, üretilen sorgunun cevabından ve önerilen kuraldan oluşturulur. Eğer cevap hükümsüzse, yeni kural olamaz ve önerilen kural çıkarılır. Eğer önerilen kural bir koşul çıkarılması ile oluşuyorsa, önerilen kuralın sonucu sorgunun cevabına göre kontrol edilmelidir. Bu durumda türetilmiş kuralın sonucu sorgunun olabilecek tüm cevaplarını kullanarak bulunabilir. Bu yöntem, olabilecek tüm kuralların türetilmesinde kullanılabilir. Yinede sadece gerekli kuralları bulabilmek, otomatik kural türetmedeki en önemli faktördür.

3.2.2 Knoblock Yöntemi

Öğrenme sistemlerindeki asıl amaç, veritabanındaki veri değerlerinin modellerine ve bunların önemine göre, kurallar türetmek, veritabanında verilen bir sorguda veritabanına mümkünse giriş yapmadan, mümkün değilse en kısa sürede sorgunun sonucunu bulmaktır. Knoblock yöntemi bu amaçla en çok kullanılan öğrenme yöntemidir.

(Piatetsky ve Saapiro 1991) tarafından yayınlanan, veritabanlarında bilgi keşfine dayalı araştırma, bir yarı uygun algoritma olan KID3 kullanılarak kuralları öğrenmeyi anlatır. Bilgi keşfi, kural-yarar ölçümleri ve tüm veri kümelerindeki örnek türetme kurallarının doğruluğu gibi diğer konuları da içerir. Bu konular, istatistiksel yöntemler kullanılarak tanımlanmışlardır. Kesin kurallar ve sağlam kurallar olarak iki gruba ayrılırlar. Kesin kurallar veri tabanında her zaman doğrudurlar ve SQO yaklaşımı için kullanılırlar, sağlam kurallar hemen hemen her zaman doğrudurlar. Bu yüzden kesin kuralların keşfi için KID3 algoritması kullanılır. Bir kesin kuralı öğrenmek için, önceki duruma uyan kayıtların aynı zamanda sonuç durumuna uyup uymadığı kontrol edilmelidir. KID3 algoritması A @ a B @ b şeklindeki kural için kullanılabilir. Burada 'a' ve 'b' sabitler ve @ karşılaştırma operatörlerinden { <, <=, >, >=, =, != } biridir. Algoritmanın ana fikri, her bir kaydı A ile değiştirmektir. Her bir değişen hücre, tüm kayıtların bir özetini saklamak için kullanılır. Eğer bir kayıt kullanılan bir hücreye dönüştürülürse, hücre özeti, özet ve kayıt arasındaki karşılaştırmaya göre güncellenir. Algoritma aşağıdaki gibidir.

Procedure KID3 (A, file)

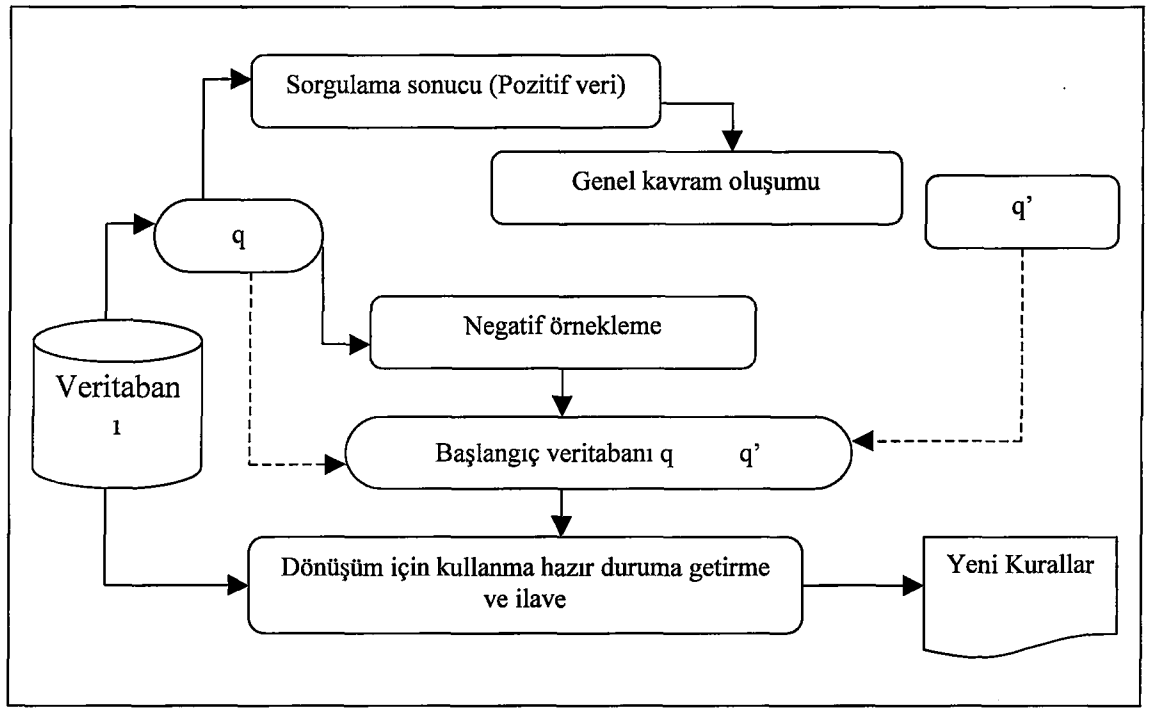
```
For Tuple in File
        Get Cell corresponding To Tuple.A
        If empty Cell Then
                Cell.Summary = Tuple, Cell.Count = 1;;Hücre başlangıç durumuna
gelir
        Else
                Cell.Count = Cell.Count + 1;; Kayıt değiştirme
                For field C in Cell.Summary
                        When Cell.Summary.C != NIL ;; Hücre özeti hükümsüzleştirilir.
                        Do<sütun tipi> Summary-Update(Cell.Summary.C, Tuple.C)
                End for
        End If
End For
```

KID3'deki 'Summary-Update(hücre, kayıt)' prosedürü kayıttaki C değeriyle karşılaştırarak o anki hücre özetini bir C alanı için değiştirir. Eğer alanın özeti çok genelse, özet değeri hükümsüzleştirilir ki bu da alanın öğrenmede kullanılmayacağı anlamına gelir. Dizi alanının tipi için, hücre özeti K değerlerine sahiptir. Burada K, alanın alabileceği değişik değerlerin en fazla sayısıdır. Bu maksimum sayıdan daha fazla K değeri varsa, özet hükümsüzleştirilir. Olağan değerler ve onların frekansları, kompleks özetler için K değerlerinin seçiminde kullanılabilir. 'NUMBER' alan tipi için, hücre, alanın en az ve en fazla değerlerine sahiptir. Kompleks özetler için ortalama değer ve standart sapma gibi değerler elde etmek mümkündür.

Bu aşamada, özetleri kullanarak kuralların nasıl yorumlandığını açıklamak gerekir. Bu hücre sayısı ve her bir hücre için A = a değerli tüm kayıtlarının özetleri kullanılarak yapılır. Her bir geçerli özet alanı C için, alanın tipini bulmak için bir kontrol yapılır. Eğer tip 'CHAR' veya 'VARCHAR2' ise C alanı şöyle yorumlanabilir. C = c1V.....Vck, burada c1,.....,ck ise C alanının değerleridir. Eğer tip 'NUMBER' ise, C alanı şöyle yorumlanabilir. c1 <= C <= c2, burada c1 C'nin en az c2 ise C'nin en fazla değeridir.

Bu algoritmayı değişik tipteki kurallar için genişletmek mümkündür. Örneğin önceki durum a1 <= A <= a2 şeklinde ise yine algoritma uygun kayıt sayısı veya veri dizinleme yapısı seçiciliği temeline dayanmaz. Yapılan tek kontrol, A üzerindeki durumla uyumlu tüm kayıtların, C üzerindeki duruma da uyup uymadığının bulunmasıdır. Bu yüzden tüm kuralların veritabanındaki etkilerini kontrol etmek için bir yoldur. Yani bazı kurallar, kurallar kümesinde tutulmaya değer olmayabilir. Bu gelecekte çok büyük bir kurallar kümesine sahip olmamıza ve verimsiz bir sorgulama işlemine yol açar. Her ne kadar bu araştırma, kuralları analiz etme için kural yararı ölçümlerini verse de, öğrenme işleminde kurallar türetme aşamasında daha seçici olması gereklidir.

Knoblock yöntemi ise, veritabanındaki asıl veri örneklerini kontrol ederek, verilen bir sorgunun durumlarını kullanır. Bunu, aday durumlar üzerinde dizinleme gibi özel veri yapılarını içeren en çok istenilen kuralları belirlemek için yapar. Yöntem, verilen bir sorgunun herhangi bir durumu için veritabanından gelen pozitif ve negatif örneklerle, bu durumun SQO yaklaşımında kaç kere oluştuğu temeline dayanır. Yöntem diğerlerine göre daha seçicidir ve buda en büyük avantajıdır.



Şekil 3.2 Veritabanını genelini öğrenme yöntemi

Knoblock sistemi iki bölümlü bir işlem olarak verilmiştir. İlk bölümde alternatif bir sorgu q', tümevarımlı öğrenme algoritmasıyla veritabanından oluşturulur. Bu alternatif sorgu verilen sorguyla aynı kayıtları gösterir yani anlamsal olarak aynıdır. Ama verilen sorgudan daha verimli ve ucuz bir şekilde işlenebilir. Bu algoritma 3.2.2.1'de anlatılmaktadır.

3.2.2.1 Alternatif Sorgu için Tümevarımla Öğrenme Algoritması

P pozitif veri, N negatif veri, S ise verilen veritabanı şemasıdır.

Adım 1 : q' hükümsüzleştirilir.

Adım 2 : Her bir sütunun (A), değer aralığı R ve P arasında olduğu bulunur.

Adım 3 : Her bir sütunun kazancı: gain(x) ve maliyeti: cost(x) hesaplanır.

Adım 4 : Eğer kazanç < 0 ise kullanılmaz

Adım 5 : Kazanç / Maliyet oranı en yüksek olan sütunu seçilir.

Adım 6 : q' = q U {x}; A = A – {x}; N = N – {n|Her n, x'ler hariç n'lerin toplamı N'den}

Adım 7 : Eğer N = Hükümsüz ise geriye q'nü döndür.

değilse Adım 3.'e git.

Where gain(x) = {n|Her n, n'lerin toplamı N ve x'ler elenir.}

cost(x) = E-cost * C-cost

= Değişen cost(x)*(P+N-gain(x)), eğer sütun dizin üzerinde ise

= Değişen cost(x)*(P+N), eğer sütun dizin üzerinde değilse

Bir koşul için kazanç, gain(x) fonksiyonu, durum tarafından elenen örneklerin sayısıdır. Durumun maliyeti, değer maliyeti, E-cost, ile değerlendirilecek örnek sayısının, C-cost, çarpımıyla bulunabilir. E-cost, değerin maliyetidir ve bir örneğin duruma uygun olup olmadığını belirler. Bu maliyet, verilen durum için karşılaştırma koşullarının sayısıyla, her bir karşılaştırmanın maliyetinin çarpımıyla bulunur. Karşılaştırma maliyeti, koşul veri tipine bağlıdır. Dizi tipi için, maliyet veri şemasındaki durum sütunun uzunluğunun tanımıdır. Bu sayı tipi için 2'dir. C-cost, koşulun dizinlenmiş bir sütun temelli olup olmadığıyla ilgilidir. Eğer koşul dizinlenmiş bir sütun temelliyse, C-cost koşulu karşılayan örneklerin sayısıdır. Yoksa, veritabanındaki örneklerin toplam sayısıdır. Bunun sebebi, veritabanındaki tüm örnekler koşula değer biçmek için kontrol edilmesindendir.

İkinci bölümde, veri tabanı başlangıç geneli iki aşamada işleme konur. Bu var olan kuralları formüle etmek için yapılır ki böylece kurallar verilen sorgunun yeniden formüle edilmesinde kullanılabilirler, q q': dönüşüm ve yarar. Dönüşüm aşamasında, veritabanı başlangıç geneli bir kriteri karşılamak için dönüştürülür, bu kriter kuralın sonuç tarafı ile orantılı bir aralığa sahip olmalıdır. Yarar aşamasında dönüştürülen kurallar başka bir kriteri karşılamak için basitleştirilir, bu kriter 'Kuralın önceki durum tarafının olabildiğince kısa olmasıdır.' Daha da fazlası önceki durum tarafını basitleştirmek için, önemsiz koşullar elenir ve bunun için bir algoritma kullanılır. Algoritmayla sonuç tarafındaki negatif örnekler aranır ve sonra eğer koşul sonuç tarafında en çok örneğe sahipse koşulu önceki durum tarafından seçer.

Tümevarımlı öğrenme algoritmasından görüleceği gibi, yöntem sıkça uygulanan ve düşük maliyetli yararlı kuralları öğrenmek için kullanılabilir. Öğrenme işlemi her ne kadar iyi tanımlanmış olsa da, sistem SQO yaklaşımının kural bakımı ve maliyet değeriyle sorgu iyileştirme gibi diğer bileşenlerine daha az hitap eder.

SQO yaklaşımı için Knoblock'un yöntemine benzer olarak (Lowden 1995; Lowden ve Lim 1995) tarafından anlamsal kurallar türetmek için ARDOR ortaya atılmıştır. ARDOR öğrenme zamanını azaltan örnek sorgular temeline dayanır. Bu yöntem sistem kullanıcılarının herhangi bir denetim ve müdahalesine gerek duymaz. Kural türetme işlemine bir ilavesi vardır. Bu

ilave, yeni bir kural türetildiğinde, kurallar kümesine eklenmeden önce yeni kuralın aynı önceki durumuna ve sonucuna sahip herhangi başka bir kuralın olup olmadığının kontrol edilmesidir. Eğer böyle bir kural varsa yeni kural ve var olan kural önceki koşul ve sonuç koşulunun sınırlarını değiştirmek için kıyaslanırlar. Bu sonuç kuralının düzgün sınırları olmasını sağlar. Yapılan inceleme aynı sonuçla biterse, fakat önceki koşul farklıysa ve var olan kuralın önceki koşulu yeni kuralın önceki koşulu kullanılarak genişletilebiliyorsa, yeni kural var olan kuralın yerini alır. Diğer bir durumda, eğer inceleme aynı önceki koşullarla fakat farklı sonuçlarla biterse ve var olan kuralın sonucunu yeni kuralın sonucuyla sınırlamak mümkünse, yeni kural var olan kuralın yerini alır. Örneğin iki var olan kural {R1, R2} ve iki türetilmiş kural {R1a, R1b} olsun.

R1) project >= 12 --> dcode = 'SALE', R2) dcode = 'SALE' --> project >= 12
R1a) project > 15 --> dcode = 'SALE', R1b) project >= 10 --> dcode = 'SALE'

R1a ve R1b, yeni kurallarıyla karşılaştırarak, R1'in sınırını değiştirecek bir kontrolün yapılmasına gerek duyulduğu görülebilir çünkü bu kurallar aynı kurallar sınıfına dahildirler. R1 kuralının önceki koşulunu R1a'ya değiştirirsek, kuralın sınırı daha küçülür. Bu yüzden, R1a kuralı pas geçilir. Ama R1'in önceki durumun sınırını R1b'ye değiştirmek sınırı genişletir. Bu yüzden, kural R1b, R1'in yerini alır.

İki türetilmiş yeni kural {R2a, R2b} düşünelim:

R2a) dcode = 'SALE' --> project > 15, R2b) dcode = 'SALE' --> project >= 10

Bu kuralın R2 kuralıyla aynı olduğu görülebilir. Sonuçlar açısından, R2b kuralı pas geçilir ve R2a kuralı R2'nin yerini almak için seçilir böylece sonucun sınırı en küçük değerine düşürülür. (project => 12'den project >= 15'ye kadar)

3.2.3 Öğrenme Metotlarının Avantajları ve Dezavantajları

Siegel yönteminin temel yaklaşımı (Siegel 1988) tarafından kural türetme, sorgu dönüşümü ve düşük maliyetli sorgu işlemini bulmaktır. Bu yöntemden, önerilen kuralları kullanarak bir çok dönüşüm türetmek mümkündür ve bunların bir kısmı sorgu iyileştirme işlemi için

kullanışlı olmayabilir. Bunun için kurallar kümesinin büyüklüğünü sınırlamak çok önemlidir. Başka bir nokta ise yöntem gerçek verilere değil gerçek verilerden oluşturulan verilere bağlıdır. Bu yüzden, yöntem veri şablonlarını kullanarak geliştirilmelidir. Böylece veritabanındaki gerçek veri değerlerinin üzerinde, kuralların etkileri belirlenebilir. Bu yüzden, bu tarz sorunlarla başa çıkabilmesi için temel yöntemi geliştirmek önemlidir. Bu geliştirmeler Siegel'in aynı araştırmasında üç farklı alanda olabilir. Bunlar, Kullanıcıya özel alan bilgisi, Heuristic bilgi ve istatistiksel artıştır.

3.2.3.1 Kullanıcıya Özel Alan Bilgisi

Otomatik kural türetme sabit bir işlem değildir, başka bir deyişle, çeşitli anlamsal bilgi ve istatistiklere bağlıdır. Temel yöntemde bilginin kullanımı tek bir alan tarafından sınırlandırılmıştır. Yine de, kullanıcıya özel alan ilgisi tarafından analiz edilen ve kullanılan bilgi için KDD ile ortaya atılan birkaç yöntem vardır. Bu alan bilgisi kural sınıflarının bir belirtisi olarak görülebilir ve bu belirtiye göre hangi kural sınıfının anlamlı ilişkiler içerdiğini belirlemek mümkündür.

3.2.3.2 Heuristic Bilgi

Temel yöntemi desteklemek amacıyla Heuristic bilgi SQO yaklaşımı için çok önemlidir. Yine de bu bilginin kullanımı da kurallar için sınırlandırılmıştır. Bu kurallar sadece değere bağlıdır. Başka bir değişle bağımlılık yoktur. Eğer ilişkilerin bağımlılıklarını kapsayan yeni bir Heuristic tanımlamak mümkün olursa, temel yöntem gerçekte kullanılan sorgular için daha verimli ve daha uygulanabilir olacaktır. Heuristic, keşfedilmiş Heuristic'lerden (Dizin tanıtma, tarama azaltması gibi) türetilemeyen farklı kural tiplerini bulmak için kullanılabilirler. Eklenen Heuristic'ler, referans kural sütunları, dinamik kural sütunları, statik kurallar (Özellikle ilişkinin anahtar sütunu için) ve kategori sütunları gibi yararlı kurarların türetilebilmesi için ilişkilerde ne tip sütunların olması gerektiğini keşfedecek şekilde geliştirilmelidir.

3.2.3.3 İstatistiksel Artış

İstatiksel artış, geçmiş sorguların işleme konulması ile toplanabilen istatistikleri (geçmiş bilgileri) kullanan kural türetme işleminin kalitesini artırmak için kullanılır. Siegel yöntemi, istatistiklerin elde edilmesi durumunda kural türetme yöntemindeki önerilen kurallar için potansiyel maliyet kazancının değerlendirilmesinde 3 faktörün belirlenmesi için bu istatistiklerin kullanılabileceğini işaret eder. Bu üç faktör türetilebilme, bakım ve önceki seçicilik faktörleridir. İstatistikleri kullanmadaki sebep eski türetmelerin performansı ile her hangi bir bilgiye sahip olmadan sadece şimdiki sorguyu kullanarak bu faktörlerin yöntemle keşfedilememesidir. Türetilebilme faktörü, kural sınıfındaki sütunların ilişkisine göre yapılır. Bakım faktörü kural kümesindeki bir kuralın bakımının ve bozuklukların kontrol edilmesinin maliyetiyle ilgilenir. Bu faktör sonradan kural türetişinin yararlılığı hakkında bir karar vermek için kullanılabilir. Önceki seçicilik faktörü, önerilen kural için önceki koşulunun seçiciliği ile alakalıdır. Eğer önceki koşul çok genişse, önerilen kuraldan gerçekten yararlı yeni bir kural türetmek pek mümkün değildir. Görüldüğü gibi, bu faktörlerin uzmanlar tarafından belirlenebilmesi çok kolay değildir. Çünkü bu faktörlerde işlenmiş sorguların geçmiş performanslarına ve bunların istatistiklerine bağlıdır.

KDD'de veri bağımlılıkları temelli yöntemleri kullanarak yeni kuralları öğrenmek mümkündür. Bu yöntemlerdeki ana fikir, istatistikler ve alan bilgisi aracılığıyla ilişkiler arasındaki veri bağımlılığını ölçmektir. Böylece sağlam kural sınıfları bulabilmek için gereksiz sütunların elenmesi sağlanır. Genellikle yöntemler tek bir kuralın yerine, bir kural sınıfında olmasının test edilmesi temeline dayanır. Çünkü bazı durumlarda kurallar aynı kural sınıfında olmalarına rağmen bir kural diğerlerinden daha verimli olabilir. Bu sebeple kural kümesinin verimli verimsiz kurallarla çok fazla büyümesi gerçekleşebilir. Buda SQO'nun performansını azaltır.

(Siegel 1988) Siegel tarafından sunulan, yöntemlere karşılık, (Hsu ve Knoblock 1993) tarafından sunulan, Knoblock temelli yöntem daha verimli kullanılabilir. Çünkü her hangi bir Heuristic ile sınırlandırılmamıştır. Yöntem birkaç yönden çok umut vericidir. Bunlardan biri, verilen sorgu yeniden formüle edilir ve sonra yeni kurallar türetmek için kullanılırlar. Bir başka umut verici yönü ise, veri örnekleri verilen sorgunun koşulları için bu yöntemle kontrol edilir ve her bir koşul için koşulla karşılanan pozitif örnekleri bulup bu örnekleri aday koşulların yapımında kullanılır. Sonra negatif örnekleri elemek için en güçlü aday koşulları

seçer. Bu eleme, veri tipleri, uzunlukları, dizin yapıları ve bunların oluşum sayılarına bağımlıdır.

(Lowden 1995; Lowden ve Lim 1995) tarafından sunulan SQO sistemi, birkaç ekle benzer bir öğrenme işlemi temeline dayanır. Öğrenme işlemi, bu araştırmanın konusu değilse de SQO yaklaşımının tüm sistemi için işlemi gerçekleştirilmesi ve kuralların öğrenilmesi zorunludur.

4 DML'LERE ÖĞRENİLEN BİLGİLERİN UYGULANMASI

DML'lerde öğrenilen bilgilerin uygulanması, veritabanı üzerinde Select(sorgulama), Update(değiştirme), Delete(silme) ve Insert Into(kayıt ekleme) komutlarında verilen koşul üzerinde yapılan iyileştirmedir.

Amaç verilen komutun performansının artırılması ve sistemde en kısa sürede verilen komutun sonucunun bulunmasıdır. Bu sayede işlemin gerçekleşme zamanı, performansı ve sisteme bağlı olunan zaman en düşük seviyeye çekilmeye çalışılır.

İyileştirilmenin kullanılabilmesi için verilen ifadede bir koşul olması gerekmektedir. Koşul verilmezse tüm tablo üzerinde işlem yapılmak istenildiği için kurallar uygulanamaz.

DML'lerde kuralların uygulanabilmesi için daha önceden oluşturulmuş olan kural dosyasından verilen koşula denk bir birincil anahtar, dizin veya normal bir alandan oluşan bir koşul istenen koşulun önüne eklenir. Önüne eklenmesindeki sebep veritabanı üzerinde yapılan sorgulamayı eklenen koşul üzerinde bir dizin varsa onun üzerinden yapmaktır ve bu sayede sorgulamayı hızlandırmak ve performansı artırmak amacı güdülür. Bu kazanç sorgulama yapılan ifadedeki koşul alanının tipine göre değişmektedir. DML ifadelerinin bilgi uygulamadan önceki hali İ1'de, sonrası ise İ2'de aşağıdaki gibi gösterilebilir:

İ1) <DML ifade>
"Where" <Koşul><Operatör><Değer>;

İ2) <DML ifade>
"Where"**<Eklenen Kural><Eklenen Operatör><Eklenen Değer>**
"and" <Koşul><Operatör><Değer>;

İ1'de bulunan DML ifadesinde ki koşul kurallar listesinde bulunarak koşula denk olan kural İ1'deki DML koşulunun önüne eklenerek İ2'deki DML ifadesi bulunur ve bu sayede kural eklenmiş olur. Yazınım olarak İ1'den farklı olan İ2 ifadesi anlamsal olarak aynıdır. Başka bir deyişle aynı sonucu verirler.

Bu tezde, DML'lerin iyileştirilmesindeki performans durumları aşağıda verilen durumlar ve koşullar göz önüne alınarak yapılmıştır.

i. Kullanılan kayıt sayıları : 100, 1000 ve 6000 adet kayıt
ii. Kullanılan DML ifadeleri : Update, Delete ve Insert Into komutları
iii. Koşul tipleri : Normal alan, birincil anahtar ve dizin alanları
iv. Eklenen kural tipleri : Normal alan, birincil anahtar ve dizin alanları
v. Koşullarda ve kurallarda kullanılan operatörler : '=', '>', '>=', '<=', '<', '!='

Bu bölümdeki ifadelerin kullanılmasında çizelge 4.1'de Prst0100 tablo yapısı ve çizelge 4.2'de örnek olarak bir kısmı verilen 'Prst0100' tablosu esas alınmıştır. (Prst0100 tablosunun tamamı Ek 1'dedir.)

Çizelge 4.1 Prst0100 tablo yapısı

Sütun Adı	Veri Durum	Veri Tipi	Veri Boyutu	Açıklama
PRS_PAT_NO#	NOT NULL	VARCHAR2	8	No
PRS_NAME_FAMILY		VARCHAR2	20	Ad
PRS_NAME_GIVEN		VARCHAR2	20	Soyad
PRS_BIRTH_DATE		DATE		Doğum Tar.
PRS_BIRTH_PLACE_CODE		VARCHAR2	2	Doğum Yer.
PRS_CITY_CODE		VARCHAR2	5	Adres Plaka
PRS_COUNTRY_CODE		VARCHAR2	5	Adres Ülke
PRS_PHONE_HOME		VARCHAR2	20	Ev Tel
PRS_PHONE_WORK		VARCHAR2	20	İş Tel
PRS_SEX		VARCHAR2	1	Cinsiyet
PRS_ZIP_CODE		VARCHAR2	5	Posta Kod

Çizelge 4.1'de bulunan PRST0100 tablosunun özellikleri

i. PRS_PAT_NO# alanı tabloda ki birincil anahtar alanıdır.
ii. PRS_PHONE_HOME alanı üzerinde tekil olmayan dizin bulunmaktadır.
iii. PRS_WORK_HOME alanı üzerinde tekil olmayan dizin bulunmaktadır.

Çizelge 4.2 Prst0100 tablo veri bilgisinin bir bölümü

Hasta No	Hasta Ad	Hasta Soyad	Doğum Tar.	Doğum Yer.	Ev Tel.	İş Tel.	Adres Plaka	Adres Ülke	Cins	Posta Kod
00000000	GOKAY	GÖK	01.01.1984	70	2163145		32	TR	E	81260
00000001	EMEL	ÖZYILDIZ	16.10.1976	34	11111		34	TR	E	81260
00000002	ZEHRA	YÜCEL	01.01.1984	6	3393541	4176304	34	TR	E	81260
00000003	ALİ	CEYHUN	03.01.2000	1	21312		21	TR	E	81260
00000004	MESUT	CEVİZ	01.01.1988	1	8123134		5	TR	E	81260
00000005	YILDIRIM	KÜMELİ	01.01.1967	34	6234366		34	AB	E	81260
00000006	GÖKÇE	KUŞCU	01.01.1966	1	8654371		5	TR	E	81260
00000007	MEHMET	ÇATAN	01.01.1933	1	4567456		6	TR	E	81260
00000008	BÜLENT	DENİZ	01.01.1976	34	2122122	2122234	3	TR	E	81260
00000009	CEM	HUYSUZ	01.01.1972	1	2164492		5	TR	E	81260
00000010	SEZER	GÜNEY	01.01.1980	1	.	2121234	6	TR	E	81260
00000011	GÖKÇE	SEZGİN	01.01.1977	35	1		7	TR	E	81260
00000012	GULER	CAKMAK	01.03.2000	34	3491458		8	TR	K	81260
00000013	FİLİZ	ŞAHİN	28.11.2000	34	5662452		9	TR	K	81260
00000014	NİZAMETTİN BARIŞ	DİKİLİTAŞ	31.08.1975	6	3802094		34	TR	E	81260
00000015	AYÇA	BAŞBUĞA	17.08.1976	34	0532		34	TR	K	81260
00000016	NİLGÜN	ARGA	26.06.1975	34	4250977		11	TR	K	81260
00000017	IŞIK	GECECİ	01.01.1959	36	3441070		35	TR	K	81260
00000018	GÜL ÖZMEN	SAYLIK	01.01.1966	44	3493025		35	TR	K	81260
00000019	SEMRA	NAMLI	23.11.1971	34	6247269		35	TR	K	81260
00000020	EMİNE	BÖYET	10.10.1959	42	3716069		35	TR	K	81260
00000021	AHMET	DİLSİZ	01.01.1938	35	3589045		35	TR	E	81260
00000022	EVREN	KAYMANLI	12.09.1980	35	8654383	879879	35	TR	K	81260
00000023	SAMİ	YILGÖREN	10.11.1926	35	347 8758		35	TR	E	81260
00000024	ZEHRA GÜLİN	ONUR	29.04.1962	6	3469277		34	TR	K	81260
00000025	HAKAN	GÜNERİ	01.01.1974	42	2122113	2121231	6	TR	K	81260

4.1 DML'lerde Öğrenilen Bilgilerin Kullanılması

DML'lerde kullanılacak olan kural bilgileri, veritabanı üzerinde yapılan sorgulama işlemleri ile elde edilirler. Bu sorguların ve kuralların otomatik olarak oluşturulması sistem tarafından yapılmaktadır. Kural bilgisi içinde koşul tablosu, koşul bilgisi, koşulu karşılayan kayıt sayısı, kural tablosu, kural bilgisi ve kuralı karşılayan kayıt sayısı bulunmaktadır. Koşulların kullanıldığı kural listesinin yapısı R0'da ki gibidir.

R0) <T1>.<S1><O1><D1>---<C1> --> <C2>---<T2>.<S2><O2><D2>

T1: Kuralın sol tarafını belirten tablo adıdır.

S1: Kuralın sol tarafını belirten sütun adıdır.

O1: Kuralın sol tarafında belirten operatör dür ('=', '>', '>=', '<=', '<', '!=')

D1: Kuralın sol tarafındaki sütuna karşılık gelen değeridir.

C1: T1'deki S1'in değerinin kaç satırda bulunduğudur.

T2: Kuralın sağ tarafını belirten tablo adıdır.

S2: Kuralın sağ tarafını belirten sütun adıdır.

O2: Kuralın sağ tarafından belirten operatördür ('=', '>', '>=', '<=', '<', '!=')

D2: Kuralın sağ tarafındaki sütuna karşılık gelen değeridir.

C2: T2'deki S2'in değerinin kaç satırda bulunduğudur.

Kurallarımızı R0'da belirtildiği biçimde oluştururuz. Çizelge 4.1 ve çizelge 4.2'de belirtilen Prst0100 tablosuna göre kurallarımızı oluştururuz. Oluşturduğumuz kuralların bir kısmı karışık olarak aşağıda verilmektedir. (Kuralların tamamı Ek 2'de bulunmaktadır.)

R1) PRST0100.PRS_BIRTH_PLACE_CODE='06'(4) --> (46) PRST0100.PRS_CITY_CODE='34'
R2) PRST0100.PRS_COUNTRY_CODE!='TR' (1) --> (1) PRST0100.PRS_PAT_NO='00000005'
R3) PRST0100.PRS_COUNTRY_CODE != 'TR' (1) --> (1) PRST0100.PRS_PHONE_HOME = '6234366'
R4) PRST0100.PRS_PHONE_HOME<'02122113452'(1) --> (11) PRST0100.PRS_PAT_NO<'00000011'
R5) PRST0100.PRS_COUNTRY_CODE != 'TR'(1) --> (1) PRST0100.PRS_PHONE_HOME='6234366'
R6) PRST0100.PRS_PAT_NO='00000035'(1) --> (16) PRST0100.PRS_CITY_CODE='35'
R7) PRST0100.PRS_CITY_CODE<'05' (1--> (1) PRST0100.PRS_PAT_NO='00000008'
R8) PRST0100.PRS_BIRTH_PLACE_CODE > '71'(1) --> (99) PRST0100.PRS_PAT_NO!='00000041'
R9) PRST0100.PRS_BİRTH_DATE = '10/11/1926' (1) --> (1) PRST0100.PRS_PAT_NO = '00000023'
R10) PRST0100.PRS_PHONE_HOME>'8654371'(1) --> (21) PRST0100.PRS_PAT_NO<'00000023'
R11) PRST0100.PRS_NAME_GIVEN>' ÜLKENCİLER' (1) --> (13) PRST0100.PRS_PAT_NO>='00000087'

Örneğin : R1'de verilen kurala göre kuralın sol tarafı, Prst0100 tablosunda bulunan Prs_Birth_Place_Code sütunu '06' değerine eşit ve Prst0100 tablosunda bu sütun koşuluyla 4 kayıt bulunmaktadır. Sağ tarafı, Prst0100 tablosunda bulunan Prs_City_Code sütunu '34' değerine eşit ve Prst0100 tablosunda bu sütun koşuluyla 46 tane kayıt vardır.

4.2 Verilerin Değiştirilmesinde Öğrenilen Bilgilerin Kullanılması

Update ifadesinin kullanımı bölüm 2.2.2'de anlatılmıştır. Bu bölümde UPDATE komutunun en genel kullanımı olan İ3 ifadesinde kuralların uygulanması esas alınacaktır.

İ3) Update <Tablo Adı>

Set <Değiştirilecek Sütun veya Sütunlar><Operatör><Değer veya Değerler>

Where <Koşul><Operatör><Değer>;

Örnek 1:

İ4) Update Prst0100

Set Prs_Sex = 'E', Prs_Name_Given = 'Koşan'

Where Prs_Country_Code != 'TR';

Bu bölümde Update DML'inin en genel kullanımı olan İ4 ifadesi şeklindeki kullanımı üzerinde kuralların kullanımı anlatılacaktır. İ4 ifadesinde "Prs_Country_Code != 'TR'" koşulunu kurallarda arayarak denk bir kural bulunmaya çalışılır. Kurallardan R2'de bu koşula göre bulunan kural bulunmaktadır.

R2) Prs_Country_Code != 'TR' (1) --> (1) Prs_Pat_No = '00000005'

R2 kuralında bulunan "Prs_Pat_No = '00000005'" kuralı İ4'ün koşul cümleciğinin önüne eklenir ve İ5 ifadesi oluşturulur.

Update işlemi yapıldığında yukarıda verilen koşula göre toplam 1 kayıt değişiyor ve update işleminin gerçekleşme süresi 0,38 saniyedir.

İ5) Update Prst0100

Set Prs_Sex = 'E', Prs_Name_Given = 'Koşan'

Where **Prs_Pat_No = '00000005' And** Prs_Country_Code != 'TR';

İ4 ifadesi veritabanı üzerinde belirtilen tablodaki kayıtların tümü üzerinden arama işlemi yaparken, İ5 ifadesi veritabanı üzerinde ki birincil anahtar üzerinden arama işlemi yaparak ifadenin performansı artırılmış olur.

Bu sayede normal bir alan üzerinde sorgulanan ifade artık bir birincil anahtar üzerindeki tekil bir dizin üzerine çekilmiş olur. Update işleminin gerçekleşme süresi ise 0,33 saniyeye düşer. Kazanç %13,5 olur.

Örnek 2 : İ4 ifadesine birincil anahtar üzerindeki bir koşul değilde dizin üzerindeki R4 koşulu eklenirse İ6 ifadesi oluşur.

R4) Prs_Country_Code != 'TR' (1) --> (1) Prs_Phone_Home = '6234366'

İ6) Update Prst0100
Set Prs_Sex = 'E', Prs_Name_Given = 'Koşan'
Where **Prs_Phone_Home = '6234366' And** Prs_Country_Code != 'TR';

Bu sayede normal bir alan üzerinde sorgulanan ifade artık bir dizin üzerine çekilmiş olur. Update işleminin gerçekleşme süresi ise 0,36 saniyeye düşer. Kazanç %5,2 olur.

4.2.1 Kullanım Sonuçları

Update ifadesine kuralların uygulanmasında kullandığımız tablolarda 100, 1000 ve 6000 kayıt olmasına göre kuralların performansı incelenmektedir. Çizelge 4.3'de görülen karşılaştırmalı tabloya göre kurallar eklendiğinde kazanç yüzdesi 100 kayıtta %3,82 iken kayıt sayısı 1000'e çıktığında %51,66'ya ve 6000 olduğunda %55,47'ye çıkmaktadır. Kazanç yüzdesinin kayıt sayısı ile doğru orantılı olarak arttığı görülmektedir.

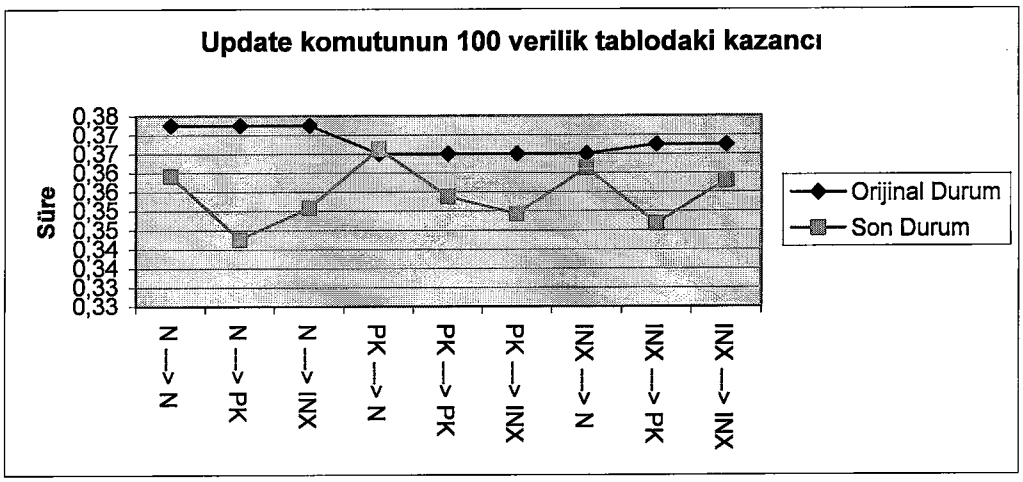
Çizelge 4.3'de kullanılan terimlerin açıklaması aşağıdaki gibidir.

Koşul : Verilen DML ifadesinde koşulun tipini gösterir.
Kural : Verilen DML ifadesinde ki koşula eklenen kuralın tipini gösterir.
N : Üzerinde birincil anahtar veya dizin bulunmayan bir sütunu göstermektedir.
PK : Üzerinde birincil anahtar bulunan bir sütunu göstermektedir.
INX : Üzerinde dizin bulunan bir sütunu göstermektedir.
Hepsi : Tüm veri cinslerini (N, PK, INX) içerin ortalamaları gösterir.
Orijinal süre : Verilen DML ifadesinde kural eklenmediğinde gerçekleşme süresidir.
Son süre : Verilen DML ifadesine kural eklendiğinde gerçekleşme süresidir.
Kazanç süre : Orijinal süre ile son süre arasındaki farkıdır.
Kazanç yüzde : Orijinal süre ile son süre arasındaki % farkı göstermektedir.

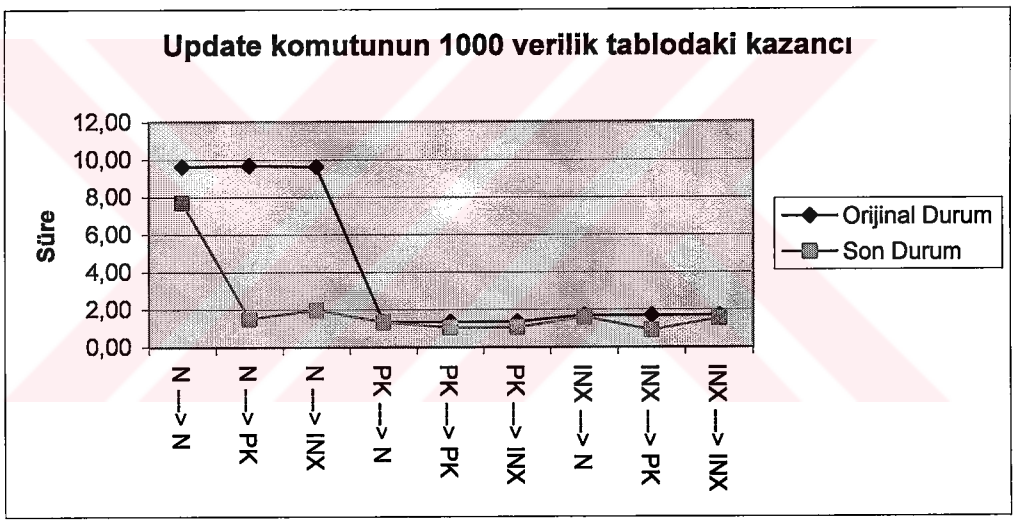
Çizelge 4.3 Update komutun koşul, kural ve veri sayısına göre performans tablosu

		100 Kayıt				1000 Kayıt				6000 Kayıt				100-1000-6000 Kayıt	
		Orijinal	Son	Kazanç		Orijinal	Son	Kazanç		Orijinal	Son	Kazanç		Genel	
Koşul	Kural	Süre	Süre	Süre	Yüzde	Süre	Süre	Süre	Yüzde	Süre	Süre	Süre	Yüzde	Kazanç Ort.	Yüzde
N	N	0,37	0,36	**0,01**	**3,5794**	9,61	7,69	**1,92**	**19,9792**	20,20	14,97	**5,23**	**25,8984**	N ---> N	**16,49**
N	PK	0,37	0,34	**0,03**	**8,0537**	9,67	1,49	**8,18**	**84,6048**	20,20	2,57	**17,62**	**87,2612**	N ---> PK	**59,97**
N	INX	0,37	0,35	**0,02**	**5,8166**	9,61	1,98	**7,63**	**79,4311**	20,20	2,97	**17,22**	**85,2787**	N ---> INX	**56,84**
PK	N	0,37	0,37	**0,00**	**-0,3425**	1,34	1,32	**0,02**	**1,8312**	5,40	5,60	**-0,20**	**-3,6711**	PK ---> N	**-0,73**
PK	PK	0,37	0,35	**0,01**	**3,0822**	1,34	1,04	**0,31**	**22,8430**	5,40	3,63	**1,77**	**32,8397**	PK ---> PK	**19,59**
PK	INX	0,37	0,35	**0,02**	**4,3379**	1,34	1,02	**0,32**	**23,9913**	5,40	2,97	**2,44**	**45,1026**	PK ---> INX	**24,48**
INX	N	0,37	0,36	**0,00**	**1,0274**	1,67	1,59	**0,08**	**4,9327**	4,69	4,38	**0,31**	**6,5813**	INX ---> N	**4,18**
INX	PK	0,37	0,35	**0,02**	**5,6689**	1,67	0,90	**0,78**	**46,3876**	4,69	1,51	**3,17**	**67,7161**	INX ---> PK	**39,92**
INX	INX	0,37	0,36	**0,01**	**2,6077**	1,67	1,52	**0,16**	**9,4170**	4,69	2,41	**2,28**	**48,6304**	INX ---> INX	**20,22**
N	Hepsi	0,37	0,35	**0,02**	**5,8166**	9,63	3,72	**5,91**	**61,3847**	20,20	6,84	**13,36**	**66,1461**	N ---> Hepsi	**44,43**
PK	Hepsi	0,37	0,36	**0,01**	**2,3592**	1,34	1,12	**0,22**	**16,2218**	5,40	4,07	**1,34**	**24,7571**	PK ---> Hepsi	**14,45**
INX	Hepsi	0,37	0,35	**0,01**	**3,3561**	1,67	1,24	**0,43**	**25,6602**	4,69	2,94	**1,74**	**37,1487**	INX ---> Hepsi	**21,44**
Hepsi	N	0,37	0,36	**0,01**	**1,4361**	4,21	3,53	**0,68**	**16,0561**	10,10	8,31	**1,78**	**17,6352**	Hepsi ---> N	**6,65**
Hepsi	PK	0,37	0,35	**0,02**	**5,6184**	4,23	1,14	**3,09**	**73,0271**	10,10	2,57	**7,52**	**74,5295**	Hepsi ---> PK	**39,83**
Hepsi	INX	0,37	0,35	**0,02**	**4,2609**	4,21	1,50	**2,70**	**64,2607**	10,10	2,78	**7,31**	**72,4424**	Hepsi ---> INX	**33,85**
Hepsi	Hepsi	0,37	0,35	**0,02**	**3,8205**	4,21	1,88	**2,33**	**51,6677**	10,10	3,99	**6,10**	**55,4735**	Hepsi ---> Hepsi	**34,65**

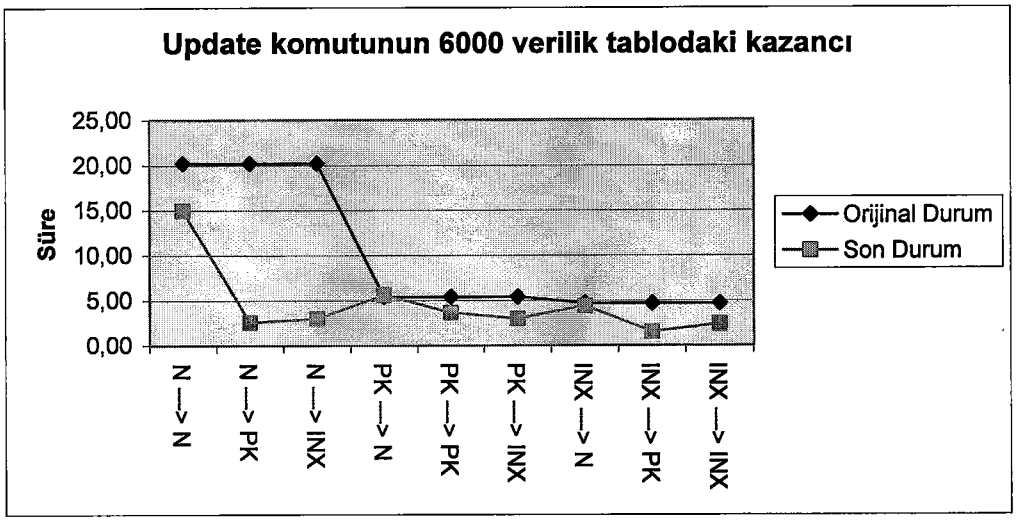
Çizelge 4.3'de verilen değerlere göre performans eğrileri Şekil 4.1, 4.2 ve 4.3'de verilmektedir.



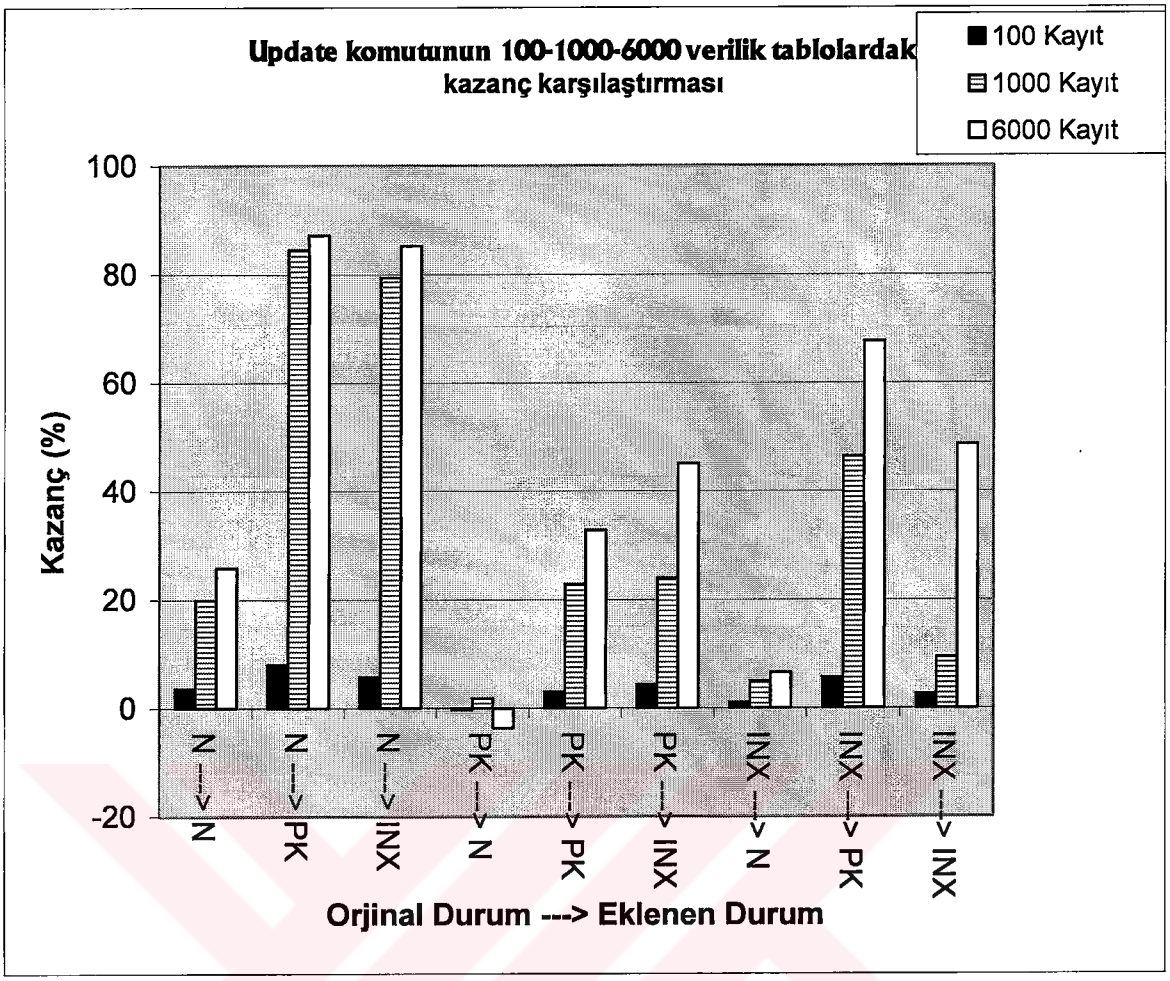
Şekil 4.1 Update komutu için 100 verilik tablodaki performans grafiği



Şekil 4.2 Update komutu için 1000 verilik tablodaki performans grafiği



Şekil 4.3 Update komutu için 6000 verilik tablodaki performans grafiği



Şekil 4.4 Update komutunun 100-1000-6000 verilik tablolardaki performans karşılaştırması

Şekillerdeki durumlardan okun sol taraftaki alan ifadede verilen koşul sağ tarafındaki alan ise eklenen kuraldır.

Şekillerde durum alanlarında belirtilen alan bilgileri şu şekildedir.

i. N, normal alan,
ii. PK, birincil anahtar üzerindeki alan,
iii. INX, dizin üzerindeki alan

Şekil 4.1, 4.2, 4.3'de Update komutunun tablodaki değişik veri sayılarına göre yapılmış incelemeler yapılmıştır ve bu değerlerden çizelge 4.4 oluşturulmuştur. Şekillerdeki sonuçlara göre en yüksek kazanç koşulu normal alandan oluşan ve kural olarak birincil anahtar veya dizin eklendiğinde oluşmaktadır.

4.3 Verilerin Silinmesinde Öğrenilen Bilgilerin Kullanılması

Delete ifadesinin kullanımı bölüm 2.2.3'de anlatılmıştır. Bu bölümde Delete komutunun en genel kullanımı olan İ4 ifadesinde kuralların uygulanması anlatılacaktır.

İ7) Delete <Tablo Adı>
Where <Koşul><Operatör><Değer>;

Örnek :

İ8) Delete Prst0100
Where Prs_City_Code < '05';

Bu bölümde Delete komutunun en genel kullanımı olan İ8 ifadesi şeklinde ki kullanımı üzerinde kuralların kullanımı anlatılacaktır. İ8 ifadesinde ki "Prs_City_Code <'05'" koşulunu kurallarda arayarak denk bir kural bulunmaya çalışılır. Kurallardan R7'de bu koşula göre bulunan kuraldır:

R7) Prs_City_Code <'05' (1) --> (1) Prs_Pat_No ='00000008'

R7 kuralında bulunan "Prs_Pat_No = '00000008'" kuralı İ9'ün koşul cümleciğinin önüne eklenir ve İ8 ifadesi oluşturulur.

Delete işlemi yapıldığında yukarıda verilen koşula göre toplam 1 kayıt değişiyor ve delete komutunun gerçekleşme süresi 0,46 saniyedir. Aşağıdaki kural kullanılarak yapılırsa

İ9) Delete Prst0100
Where **Prs_Pat_No ='00000008' And** Prs_City_Code < '05';

İ8 ifadesi veritabanı üzerinde belirtilen tablodaki kayıtların tümü üzerinden arama işlemi yaparken, İ9 ifadesi veritabanı üzerinde ki dizin üzerinden arama işlemi yaparak ifadenin performansı artırılmış olur.

Bu sayede bir dizin alan üzerinde sorgulanan ifade artık bir birincil anahtar üzerindeki tekil bir dizin üzerine çekilmiş olur. Delete işleminin gerçekleşme süresi ise 0,39 saniyeye düşer. Kazanç %15,20 olur.

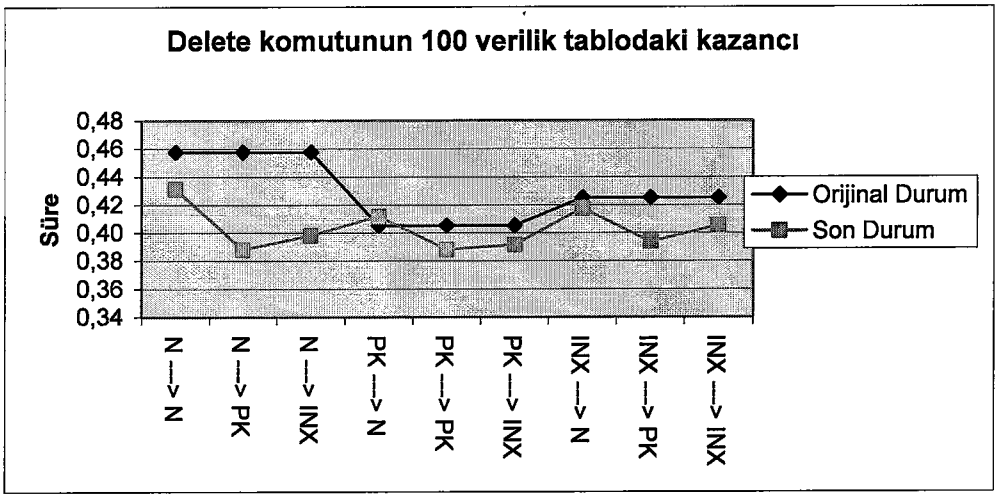
4.3.1 Kullanım Sonuçları

Delete komutunda kuralların uygulanmasında, kullandığımız tablolarda 100, 1000 ve 6000 kayıt olmasına göre kuralların performansı değişmektedir. Çizelge 4.4'de görülen karşılaştırmalı tabloya göre kurallar eklendiğinde kazanç yüzdesi 100 kayıtta %6,18 iken kayıt sayısı 1000'e çıktığında %35,77'ye ve 6000 olduğunda %44,83'ee çıkmaktadır. Kazanç yüzdesinin kayıt sayısı ile doğru orantılı olarak arttığı görülmektedir.

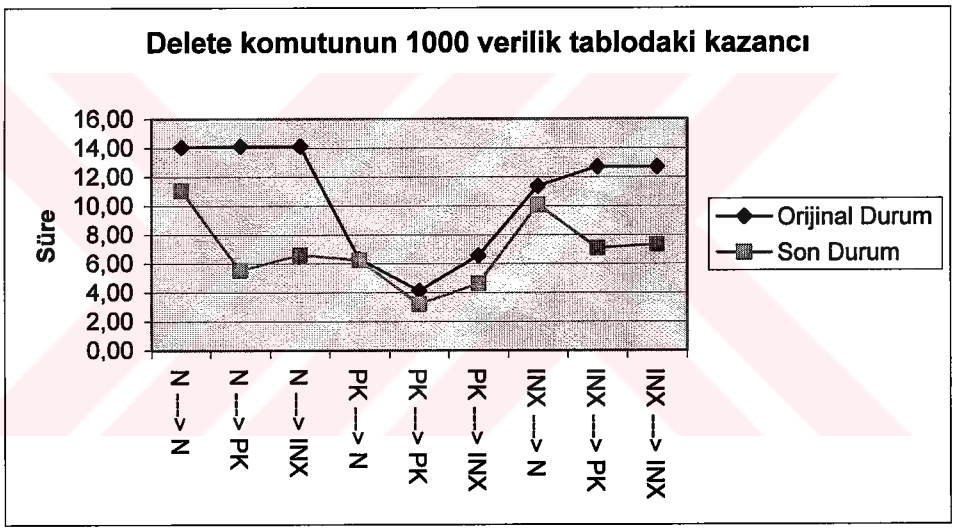
Çizelge 4.4 Delete komutun koşul, kural ve veri sayısına göre performans tablosu

		100 Kayıt				1000 Kayıt				6000 Kayıt				100-1000-6000 Kayıt	
		Orijinal	Son	Kazanç		Orijinal	Son	Kazanç		Orijinal	Son	Kazanç		Genel	
Koşul	Kural	Süre	Süre	Süre	Yüzde	Süre	Süre	Süre	Yüzde	Süre	Süre	Süre	Yüzde	Kazanç Ort.	Yüzde
N	N	0,46	0,43	**0,03**	**5,7377**	14,04	11,05	**3,00**	**21,3319**	29,15	22,55	**6,60**	**22,6501**	N ---> N	**16,57**
N	PK	0,46	0,39	**0,07**	**15,2095**	14,11	5,51	**8,60**	**60,9565**	28,43	7,02	**21,41**	**75,3090**	N ---> PK	**50,49**
N	INX	0,46	0,40	**0,06**	**13,0237**	14,11	6,57	**7,54**	**53,4355**	29,15	11,44	**17,71**	**60,7604**	N ---> INX	**42,41**
PK	N	0,41	0,41	**-0,01**	**-1,6461**	6,29	6,22	**0,07**	**1,1853**	8,96	8,92	**0,03**	**0,3861**	PK ---> N	**-0,02**
PK	PK	0,41	0,39	**0,02**	**4,2181**	4,12	3,18	**0,94**	**22,9066**	6,96	3,98	**2,98**	**42,8435**	PK ---> PK	**23,32**
PK	INX	0,41	0,39	**0,01**	**3,3951**	6,54	4,60	**1,94**	**29,7096**	8,96	5,40	**3,55**	**39,6642**	PK ---> INX	**24,26**
INX	N	0,43	0,42	**0,01**	**1,7647**	11,36	10,06	**1,30**	**11,4278**	18,11	15,17	**2,94**	**16,2364**	INX ---> N	**9,81**
INX	PK	0,43	0,39	**0,03**	**7,4510**	12,69	7,08	**5,61**	**44,2233**	20,06	9,51	**10,55**	**52,6053**	INX ---> PK	**34,76**
INX	INX	0,43	0,41	**0,02**	**4,7059**	12,69	7,32	**5,37**	**42,2989**	23,08	11,68	**11,41**	**49,4133**	INX ---> INX	**32,14**
N	Hepsi	0,46	0,41	**0,05**	**11,3236**	14,08	7,71	**6,38**	**45,2781**	28,91	13,67	**15,24**	**52,7192**	N ---> Hepsi	**36,49**
PK	Hepsi	0,41	0,40	**0,01**	**1,9890**	5,65	4,66	**0,99**	**17,4703**	8,29	6,10	**2,19**	**26,4080**	PK ---> Hepsi	**15,85**
INX	Hepsi	0,43	0,41	**0,02**	**4,6078**	12,02	8,57	**3,45**	**28,7326**	19,08	12,34	**6,75**	**35,3476**	INX ---> Hepsi	**25,57**
Hepsi	N	0,43	0,42	**0,01**	**2,1036**	10,56	9,11	**1,46**	**13,7820**	18,74	15,55	**3,19**	**17,0365**	Hepsi ---> N	**8,79**
Hepsi	PK	0,43	0,39	**0,04**	**9,1909**	10,30	5,25	**5,05**	**49,0174**	18,48	6,83	**11,65**	**63,0213**	Hepsi ---> PK	**36,19**
Hepsi	INX	0,43	0,40	**0,03**	**7,2492**	11,11	6,16	**4,95**	**44,5403**	20,40	9,51	**10,89**	**53,3918**	Hepsi ---> INX	**32,93**
Hepsi	Hepsi	0,43	0,40	**0,03**	**6,18**	10,66	6,84	**3,82**	**35,7799**	19,21	10,63	**8,58**	**44,4832**	Hepsi ---> Hepsi	**25,97**

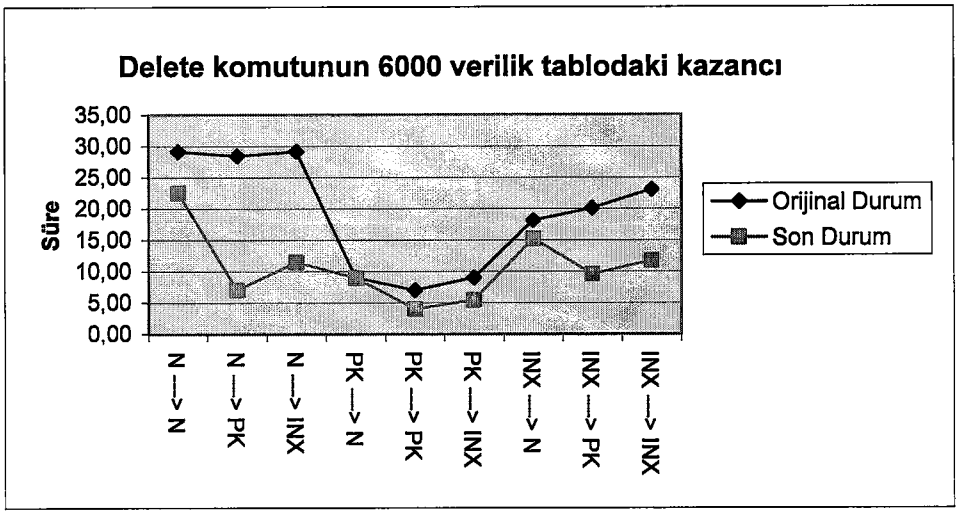
Çizelge 4.4'de verilen değerlere göre performans eğrileri Şekil 4.5, 4.6 ve 4.7'de verilmektedir.



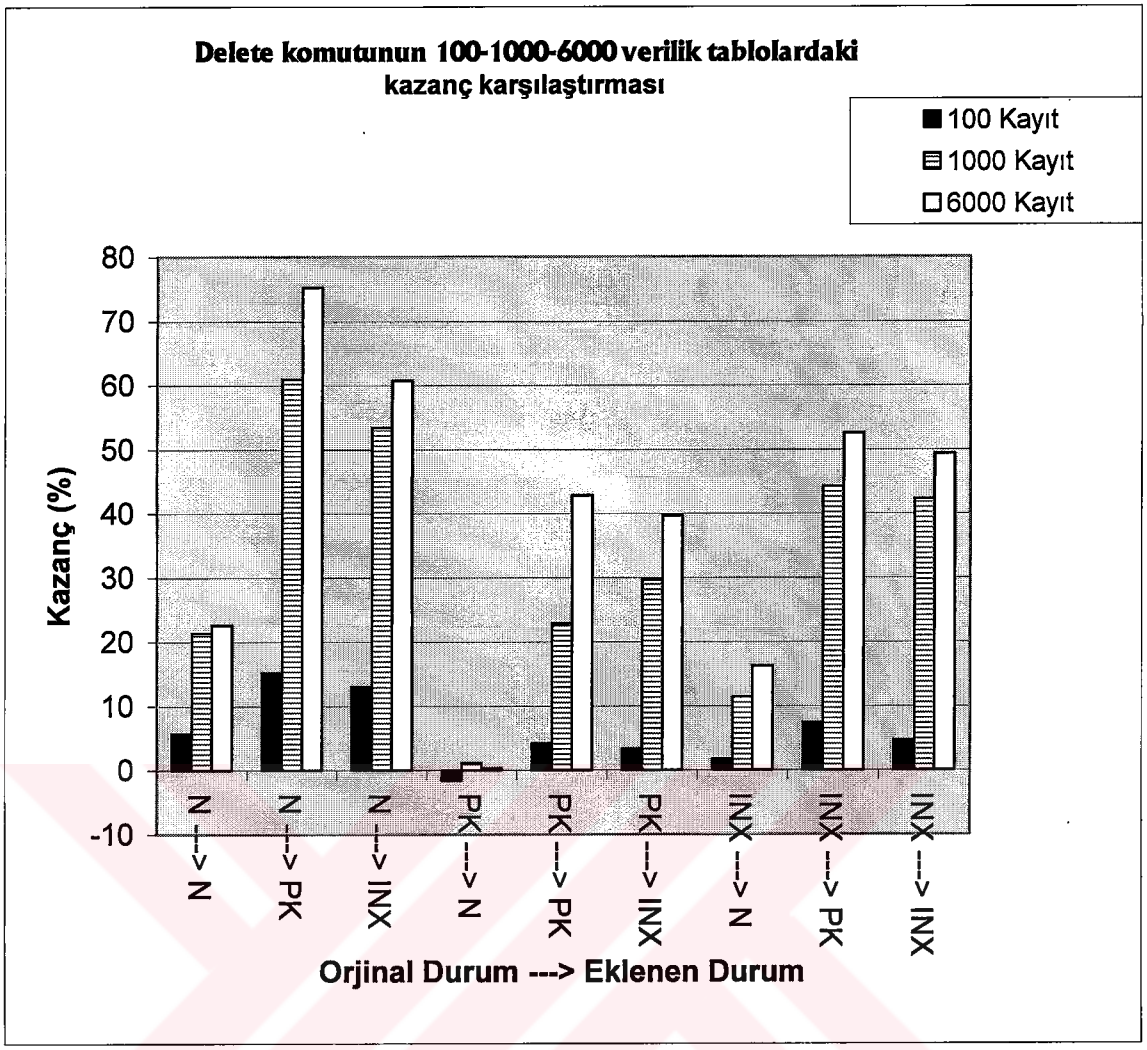
Şekil 4.5 Delete komutu için 100 verilik tablodaki performans grafiği



Şekil 4.6 Delete komutu için 1000 verilik tablodaki performans grafiği



Şekil 4.7 Delete komutu için 6000 verilik tablodaki performans grafiği



Şekil 4.8 Delete komutunun 100-1000-6000 verilik tablolardaki performans karşılaştırması

Şekil 4.5, 4.5, 4.7'de Delete komutunun tablodaki değişik veri sayılarına göre yapılmış incelemeleri yapılmıştır ve bu incelemelerin sonuç değerlerden çizelge 4.8 oluşturulmuştur. Şekillerdeki sonuçlara göre en yüksek kazanç koşulu normal alandan oluşan ve kural olarak birincil anahtar veya dizin eklendiğinde oluşmaktadır.

4.4 Verilerin Eklenmesinde Öğrenilen Bilgilerin Kullanılması

Insert Into komutunun kullanımı bölüm 2.2.4'de anlatılmıştır. Insert Into komutu üzerinde kural kullanımında bir koşul belirtmek için başka bir tablo üzerinden kayıt eklenmesi gerekmektedir. Tablomuzda birincil anahtar bulunduğundan tablonun kendi üzerine kayıt eklemesi birincil anahtarların kısıtlamalarına karşı olacağı için başka bir tablo üzerinden bu işlemi yapmalıyız ve kuralları veri çekilen tablo üzerinde yaparak uygulamalıyız. Bu bölümde

insert into komutunun en genel kullanımlarından olan İ11 ifadesinde kuralların uygulanması anlatılacaktır. Komutun genel ifadesi aşağıda İ10'da verilmiştir.

İ10) Insert Into <Eklenecek Tablo Adı>
Select <Sütunlar>
From <Veri Çekilen Tablo Adı>
Where <Koşul><Operatör><Değer>;

Örnek :

İ11) Insert Into Prst1000
Select * From Prst0100
Where Prs_Birth_Date = '10/11/1926';

Bu bölümde Update DML'inin en genel kullanımı olan İ11 ifadesi şeklinde kullanımı üzerinde kuralların kullanımı anlatılacaktır. İ11 ifadesinde "Prs_Birth_Date = '10/11/1926'" koşulunu kurallarda arayarak denk bir kural bulunmaya çalışılır. Kurallardan R9'da bu koşula göre bulunan kuraldır.

R9) Prs_Birth_Date = '10/11/1926' (1) --> (1) Prs_Pat_No = '00000023'

R9 kuralında bulunan "Prs_Pat_No = '00000023'" İ11'in koşul cümleciğinin önüne eklenir ve İ12 ifadesi oluşturulur.

Kayıt ekleme işlemi yapıldığında yukarıda verilen koşula göre toplam 1 kayıt değişiyor ve ekleme işleminin gerçekleşme süresi 0,44 saniyedir.

İ12) Update Prst0100
Set Prs_Sex = 'E', Prs_Name_Given = 'Koşan'
Where **Prs_Pat_No = '00000023' And** Prs_Birth_Date = '10/11/1926';

Bu sayede, İ11 ifadesi veritabanı üzerinde belirtilen tablodaki kayıtların tümü üzerinden arama işlemi yaparken, İ12 ifadesi veritabanı üzerinde ki birincil anahtar üzerinden arama işlemi yaparak ifadenin performansı artırılmış olur.

Bu sayede dizin bir alan üzerinde sorgulanan ifade artık bir birincil anahtar üzerindeki tekil bir dizin üzerine çekilmiş olur. Update işleminin gerçekleşme süresi ise 0,38 saniyeye düşer. Kazanç %13,16 olur.

4.4.1 Kullanım Sonuçları

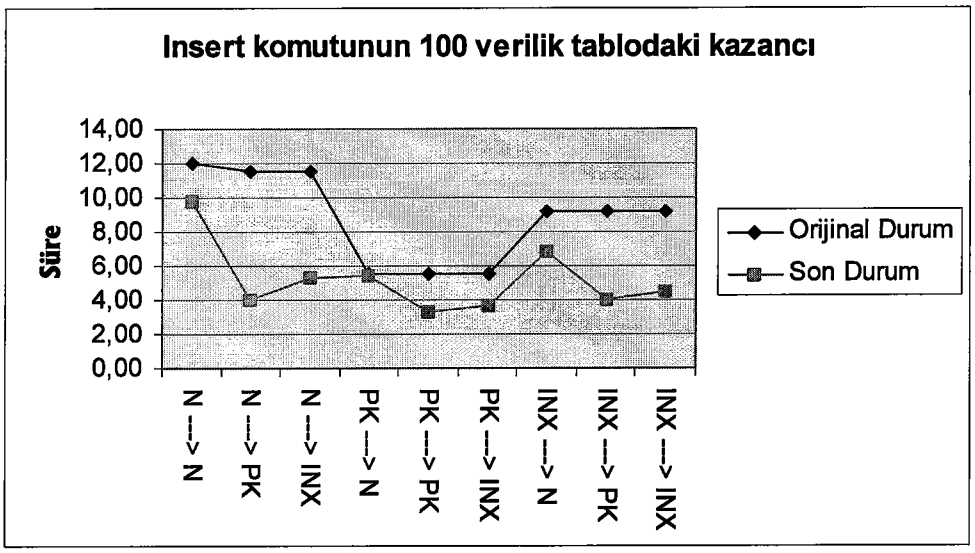
Insert Into komutu üzerinde kural kullanımında bir tablodan diğer bir tabloya kayıt ekleneceği için 2 ayrı veri büyüklüğüne sahip tablo kullanılarak bunlar üzerinde kuralların kullanımı gerçekleştirildi. Burada 100 kayıtlık tabloya 1000 kayıtlık tablodan kayıt eklenirken, 1000 kayıtlık tablodan 100 kayıtlık tabloya kayıt eklendi. Bundan dolayı 100 kayıtlık tablo üzerinde yapılan işlemlerin gerçekleşme süresi 1000 kayıtlık olan tablodakine göre daha fazla bulunmuştur.

Insert Into komutuna kuralların uygulanmasında kullandığımız tablolarda 100 ve 1000 kayıt olmasına göre kuralların performansı değişmektedir. Çizelge 4.5'de görülen karşılaştırmalı tabloya göre kurallar eklendiğinde kazanç yüzdesi 100 kayıtta %41.10 iken kayıt sayısı 1000'e çıktığında %4,61'e olmaktadır. Kazanç yüzdesi, gerçekleşme süresi ile doğru orantılı olarak arttığı görülmektedir. 100 kayıtlık tabloda gerçekleşen Insert Into komutlarında 1000 kayıtlı tablo üzerinden geldiğinden koşulların gerçekleşme süresi artmaktadır ve kurallar burada daha önemli bir hale gelerek daha verimli işlev görürler.

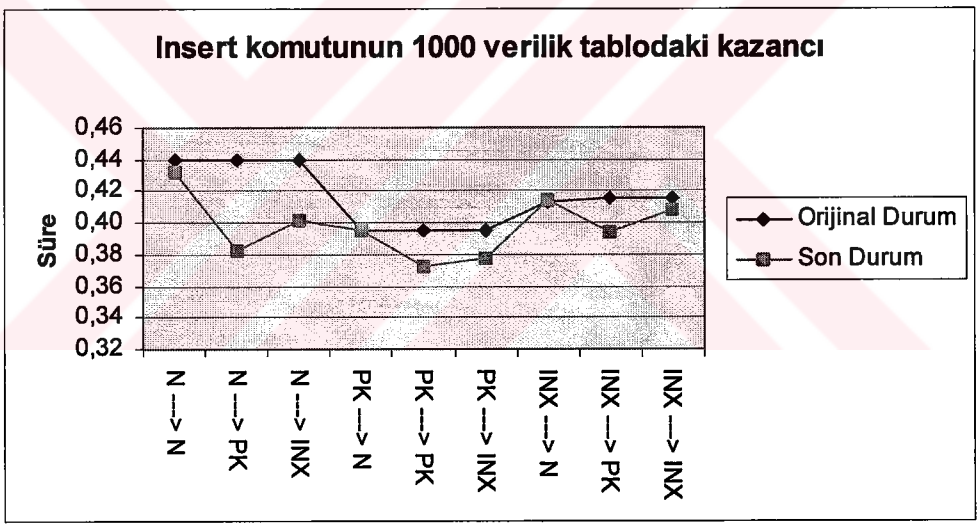
Çizelge 4.5 Insert Into komutun koşul, kural ve veri sayısına göre performans tablosu

		100 Kayıt				1000 Kayıt				100-1000 Kayıt	
		Orijinal	Son	Kazanç		Orijinal	Son	Kazanç		Genel	
Koşul	Kural	Süre	Süre	Süre	Yüzde	Süre	Süre	Süre	Yüzde	Kazanç Ort.	Yüzde
N	N	12,02	9,76	**2,26**	**18,7890**	0,44	0,43	**0,01**	**1,8939**	N ---> N	**10,34**
N	PK	11,58	4,03	**7,55**	**65,2382**	0,44	0,38	**0,06**	**13,1629**	N ---> PK	**39,20**
N	INX	11,57	5,24	**6,33**	**54,7223**	0,44	0,40	**0,04**	**8,8068**	N ---> INX	**31,76**
PK	N	5,50	5,45	**0,05**	**0,9439**	0,40	0,39	**0,00**	**0,1055**	PK ---> N	**0,52**
PK	PK	5,50	3,31	**2,19**	**39,8788**	0,40	0,37	**0,02**	**5,9072**	PK ---> PK	**22,89**
PK	INX	5,50	3,69	**1,81**	**32,9015**	0,40	0,38	**0,02**	**4,5359**	PK ---> INX	**18,72**
INX	N	9,14	6,80	**2,34**	**25,6130**	0,41	0,41	**0,00**	**-0,5051**	INX ---> N	**12,55**
INX	PK	9,14	3,99	**5,16**	**56,3850**	0,42	0,39	**0,02**	**5,2209**	INX ---> PK	**30,80**
INX	INX	9,14	4,43	**4,71**	**51,5268**	0,42	0,41	**0,01**	**1,6064**	INX ---> INX	**26,57**
N	Hepsi	11,72	6,34	**5,38**	**45,9071**	0,44	0,41	**0,04**	**7,9545**	N ---> Hepsi	**27,10**
PK	Hepsi	5,50	4,15	**1,35**	**24,5747**	0,40	0,38	**0,01**	**3,5162**	PK ---> Hepsi	**14,05**
INX	Hepsi	9,14	5,39	**3,75**	**40,9990**	0,41	0,40	**0,01**	**2,3666**	INX ---> Hepsi	**23,31**
Hepsi	N	8,89	7,34	**1,55**	**17,4476**	0,42	0,41	**0,00**	**0,5344**	Hepsi ---> N	**7,81**
Hepsi	PK	8,74	3,77	**4,97**	**56,8326**	0,42	0,38	**0,03**	**8,2333**	Hepsi ---> PK	**30,97**
Hepsi	INX	8,74	4,45	**4,28**	**49,0287**	0,42	0,40	**0,02**	**5,0667**	Hepsi ---> INX	**25,68**
Hepsi	Hepsi	8,79	5,19	**3,60**	**41,1029**	0,42	0,40	**0,02**	**4,6115**	Hepsi ---> Hepsi	**21,48**

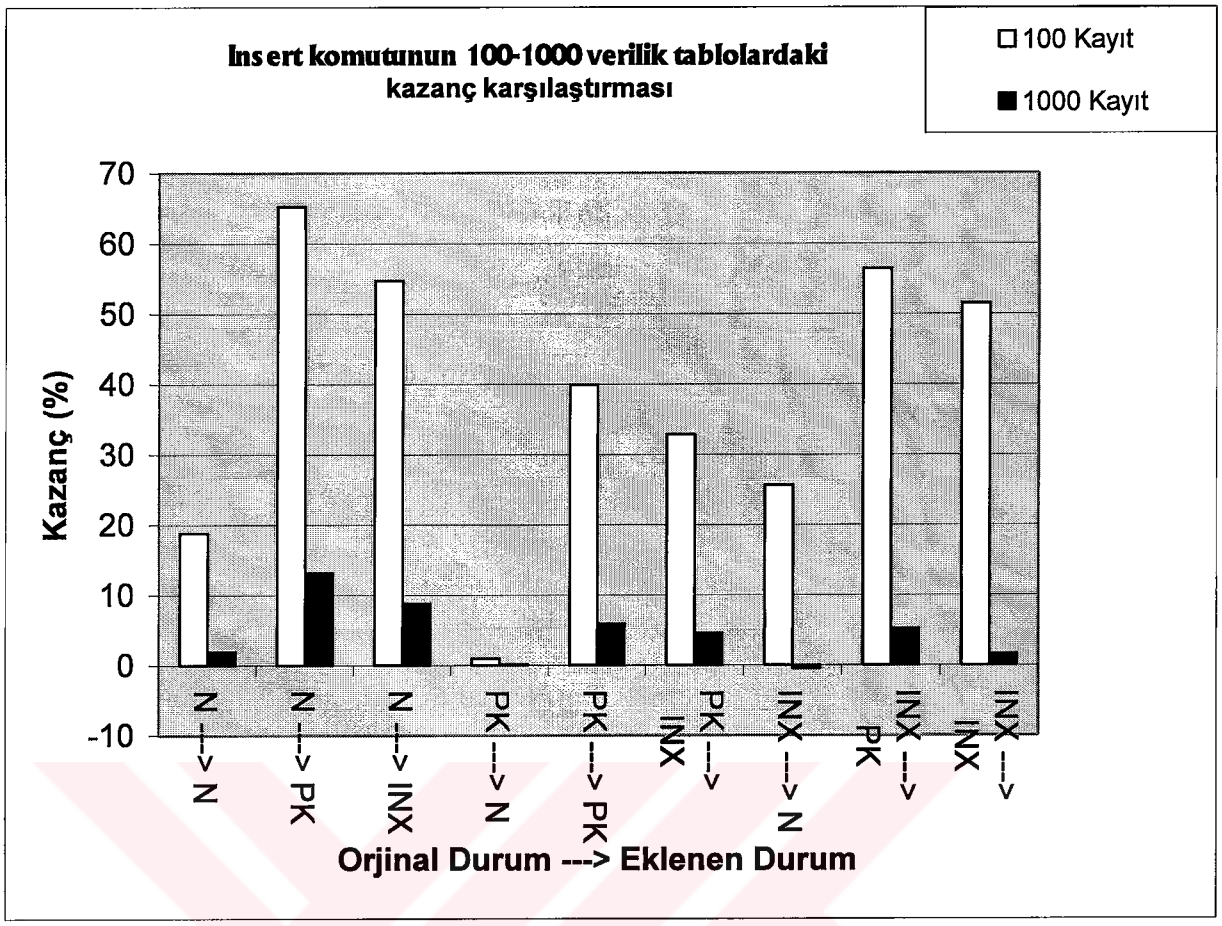
Çizelge 4.5'de verilen değerlere göre performans eğrileri Şekil 4.9, 4.10'da verilmektedir.



Şekil 4.9 Insert Into komutu için 100 verilik tablodaki performans grafiği



Şekil 4.10 Insert Into komutu için 1000 verilik tablodaki performans grafiği



Şekil 4.11 Insert Into komutunun 100-1000 verilik tablolardaki performans karşılaştırması

Şekil 4.9, 4.10'da Insert Into komutunun tablodaki değişik veri sayılarına göre yapılmış incelemelerdir ve inceleme sonuç değerlerinden çizelge 4.11 oluşturulmuştur. Şekillerdeki sonuçlara göre en yüksek kazanç koşulu normal alandan oluşan ve kural olarak birincil anahtar veya dizin eklendiğinde oluşmaktadır.

4.5 Genel Sonuçlar

Bu bölümde, bölüm 4.2.1, 4.3.1 ve 4.4.1 de alınan sonuçlara göre ortalamalar alınarak kuralların sisteme genel etkisi ele alınacaktır.

Aşağıdaki listede verilen koşullar ve bu koşullara eklenen alanlara göre aradaki kazancın yüzdesi verilmektedir.

Çizelge 4.6'da ki değerler:

i. 100, 1000 ve 6000 kayıt bulunan tablolar üzerinden
ii. Update, Delete ve Insert Into komutlarıyla
iii. normal alan, birincil anahtar ve dizin üzerindeki alanlarla
iv. =, >, >=, <=, <, != operatörleri ile yapılan sorguları, verilen ve eklenen koşula göre gruplanarak elde edilmiştir.

Çizelge 4.6 Verilen koşul tiplerine ve eklenen koşul tiplerine göre kazanç durumu

Verilen Koşul	**Eklenen Koşul**	**Ortalama Kazanç (%)**
Normal Alan	Normal Alan	16,53
Normal Alan	Birincil Anahtar	55,23
Normal Alan	Dizin	49,62
Birincil Anahtar	Normal Alan	-0,38
Birincil Anahtar	Birincil Anahtar	24,46
Birincil Anahtar	Dizin	21,37
Dizin	Normal Alan	1,00
Dizin	Birincil Anahtar	37,34
Dizin	Dizin	26,18

Bu alanları istenen koşul göre gruplarsak aşağıdaki liste oluşur:

Çizelge 4.7 Verilen koşul tiplerine göre eklenen koşula göre kazanç durumu

Verilen Koşul	**Eklenen Koşul**	**Ortalama Kazanç (%)**
Normal Alan	Tüm Alanlar	40,46
Birincil Anahtar	Tüm Alanlar	15,15
Dizin	Tüm Alanlar	21,51

Bu alanları eklenen koşullara göre gruplarsak aşağıdaki liste oluşur:

Çizelge 4.8 Verilen koşula eklenen koşul tiplere göre kazanç durumu

Verilen Koşul	**Eklenen Koşul**	**Ortalama Kazanç (%)**
Tüm Alanlar	Normal Alan	5,72
Tüm Alanlar	Birincil Anahtar	39,01
Tüm Alanlar	Dizin	32,39

Hesaplanan değerlere göre:

a) Verilen koşullardan normal alanlı koşullarda kazancı en yüksektir. Bunun sebebi ise bu alan üzerinde anahtar veya dizin olmadığı için sorgulama tüm tablo bilgilerini tarayarak yapılmaktadır ve bundan dolayı daha uzun sürmektedir. Bir anahtar eklenerek sorgulandığında bir tekil dizin olduğu için tablo sadece gerektiği kısmıyla taranmaktadır. Bir dizin eklenerek sorgulandığında ise dizin üzerinde sorgulanma yapıldığı için yine tablo sadece gerektiği kısmıyla taranmaktadır ve bu sayede sorgulama hızlandırılmıştır.

b) Eklenen koşullarda birincil anahtar alanlı koşullarda kazanç en yüksektir. Bunun sebebi sorgulama tekil dizin üzerinde olduğundan tablo sadece gerektiği kısmıyla taranmaktadır.

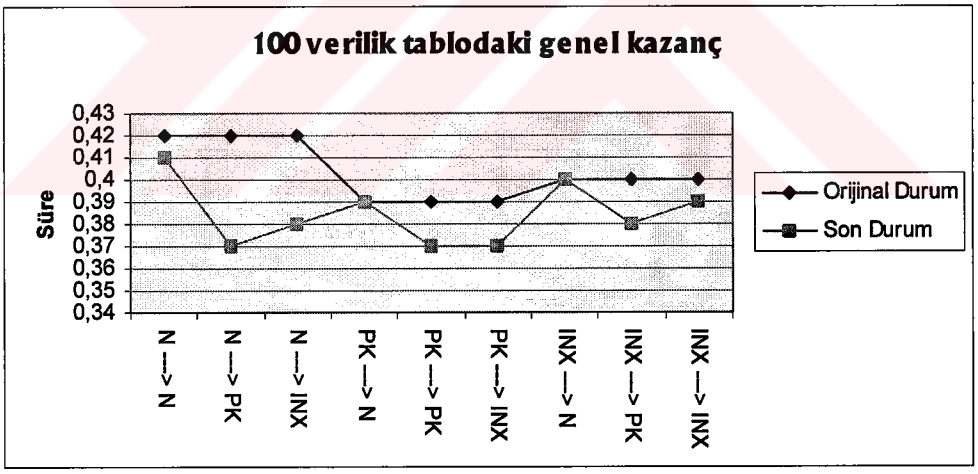
Buradan da anlaşılacağı gibi eklenen koşulun bir dizin üzerinde olması özelliklede tekil bir dizinin üzerinde olması her zaman daha yüksek kazanç sağlar. Fakat birincil anahtar üzerine veya bir dizin üzerine normal alan eklenmesi zaten dizin üzerinde çalışan bir yapı olduğu için sisteme performans kazancı göz ardı edilebilecek kadar azdır bazen de hiç yoktur.

Yapılan sorgularda kullanılan tablolarda bulunan kayıt sayıları performans oranları ile doğru orantılıdır. Aşağıdaki tabloda bu değerler verilmektedir.

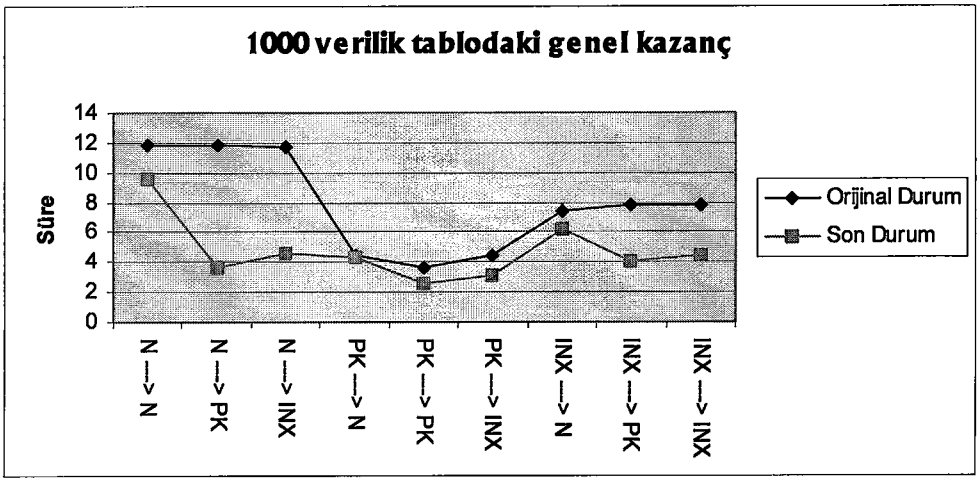
Çizelge 4.9 Genel koşul tipi ve kuralların kayıt bazlı performans durumu

Verilen Koşul	Eklenen Koşul	100 Kayıt			1000 Kayıt			6000 Kayıt		
		Orjinal Süre	Son Süre	Kazanç %	Orjinal Süre	Son Süre	Kazanç %	Orjinal Süre	Son Süre	Kazanç %
Normal	Normal	0,42	0,41	**3,773**	11,89	9,5	**20,1108**	24,67	18,76	**23,9796**
Normal	B. Anahtar	0,42	0,37	**12,4016**	11,78	3,67	**68,8259**	24,31	4,8	**80,2739**
Normal	Dizin	0,42	0,38	**9,4488**	11,76	4,59	**60,9379**	24,67	7,21	**70,7955**
B. Anahtar	Normal	0,39	0,39	**-0,6438**	4,38	4,33	**1,1502**	7,18	7,26	**-1,1403**
B. Anahtar	B. Anahtar	0,39	0,37	**4,4349**	3,65	2,51	**31,4139**	6,18	3,8	**38,4709**
B. Anahtar	Dizin	0,39	0,37	**4,0773**	4,46	3,1	**30,4476**	7,18	4,19	**41,7102**
Dizin	Normal	0,40	0,4	**0,7623**	7,39	6,15	**16,7869**	11,4	9,77	**14,252**
Dizin	B. Anahtar	0,40	0,38	**6,1422**	7,83	3,99	**49,1083**	12,37	5,51	**55,4669**
Dizin	Dizin	0,40	0,39	**3,0021**	7,83	4,42	**43,5486**	13,88	7,04	**49,2812**

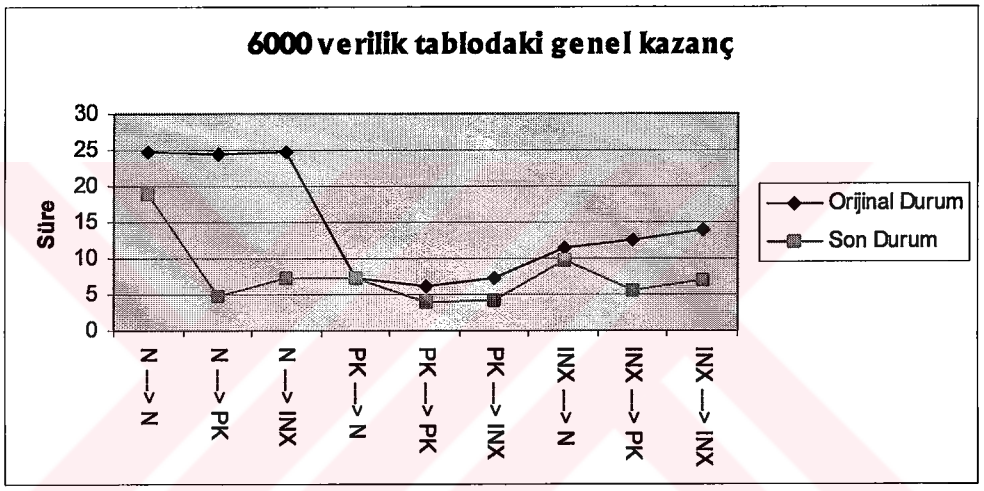
Örneğin : 100 kayıt üzerinden istenen koşulda normal bir alan varsa, eklenen koşulda birincil anahtarsa, iyileştirme yapılmadan önce gerçekleşme süresi 0,42 saniye, iyileştirmeden sonra 0,37 saniyede gerçekleşir ve kazanç 0,05 saniyedir. 6000 kayıt üzerinde ise süre 24,31 saniyeden 4,8 saniyeye iniyor burada kazanç ise 19,51 saniyedir. Bu da bize sistemdeki kayıt sayısı arttıkça performansın oranının da arttığını göstermektedir.



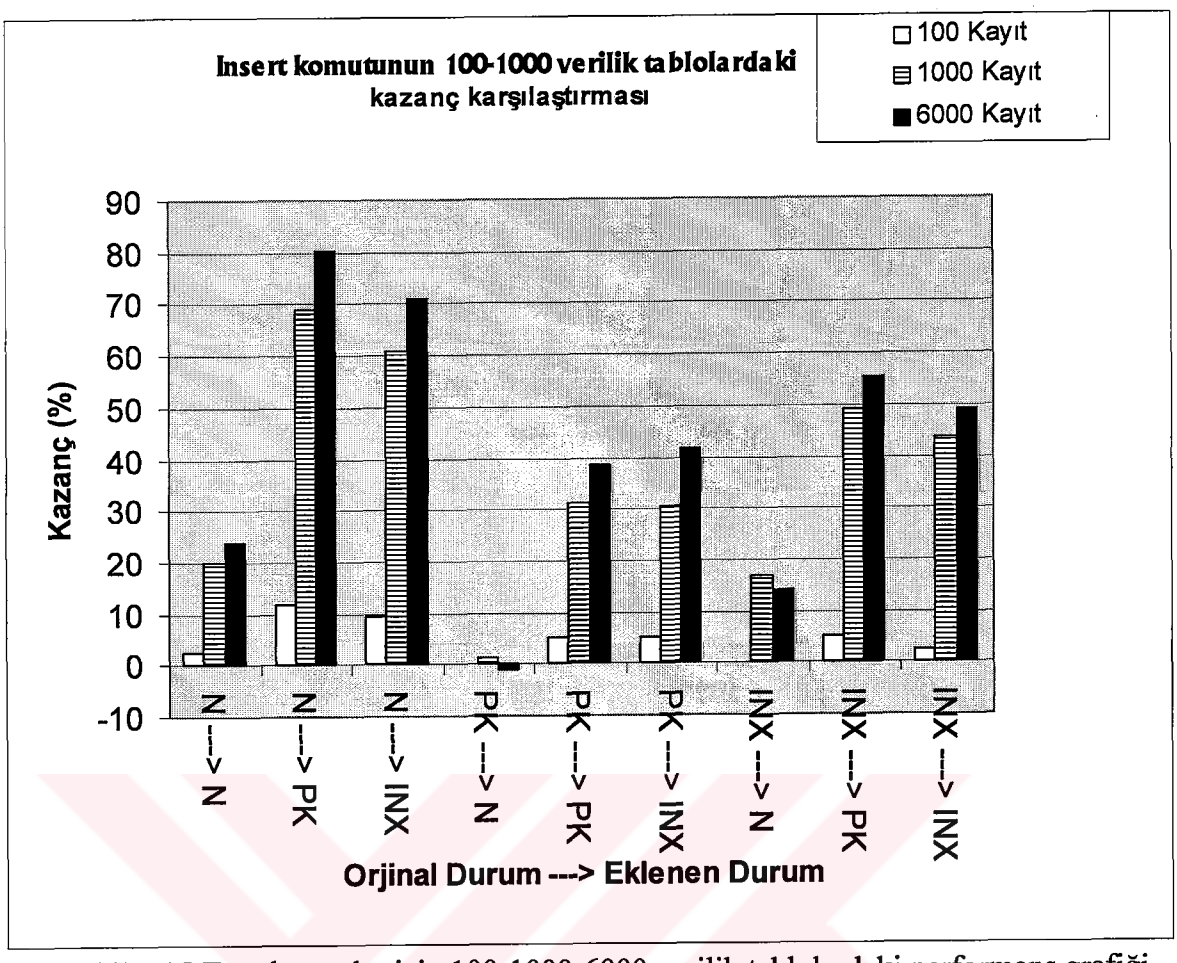
Şekil 4.12 Tüm komutlar için 100 verilik tablodaki performans grafiği



Şekil 4.13 Tüm komutlar için 1000 verilik tablodaki performans grafiği



Şekil 4.14 Tüm komutlar için 6000 verilik tablodaki performans grafiği



Şekil 4.15 Tüm komutlar için 100-1000-6000 verilik tablolardaki performans grafiği

Bu çizelge ve grafiklere göre şu sonuçlar çıkarılabilir.

1) Kayıt sayısının artması ile performans ve kazanç doğru orantılı olarak artmaktadır.

2) Kural olarak birincil anahtar eklenmesi, kazancı artıran en yüksek durumdur. Dizin eklenmesi, kazancı artıran diğer bir durumdur. Normal bir alan eklenmesi ise en düşük hatta kimi durumlarda negatif kazanç sağlayan bir durumdur.

3) Normal alan durumuna kural eklenmesi, kazancın en yüksek durumudur. Dizin durumuna kural eklenmesi daha az bir kazanç sağlar. Birincil anahtar durumuna kural eklendiğinde ise kimi durumlarda negatif kazanç sağlanır.

4) 2 ve 3. maddelerdeki duruma göre normal bir alan koşuluna birincil anahtar ve dizin ile oluşan kural eklenmesi bize en yüksek kazanç veren durumlardır.

5 Sonuç

5.1 Ulaşılan Hedefler

i. Visual C++'dan ADO nesnesi ile ODBC üzerinden Oracle veritabanına bağlantı sağlanması.
ii. Visual C++'dan bir sorgu gönderilerek cevabın ifade edilmesi. (Select)
iii. Visual C++'da verilen diğer bir DML ifadesinin çalıştırılması. (Update, Delete, Insert Into)
iv. Verilen DML ifadesinin komut ve parametre olarak ayrılması.
v. Veritabanına ilk bağlantı sağlanıldığında veritabanın modeli fiziksel bir dosya üzerine alınması ve sistem tablolarının kullanılması.
vi. Verilen DML ifadesinde hata olup olmadığının kontrol edilmesi.
vii. 100 – 1000 ve 6000 verilik gerçek verili tabloların kullanılması.
viii. Knoblock öğrenme tekniğine göre veritabanı üzerinde otomatik olarak sorgular çalıştırılarak kuralların öğrenilmesi.
ix. 100 – 1000 – 6000 verilik tablolar için ayrı ayrı kural dosyalarının oluşturulması.
x. DML'lere kuralların uygulanması. (DML'lerin en genel hali için yapıldı. Bölüm 4.2, 4.3, 4.4'de en genel hali belirtilmiştir.)
xi. DML komutu çalıştırıldığında verilen koşulun kurallar içerisinde bulunması sağlanarak koşulun önüne eklenmesi sağlandı.
xii. Çalıştırılan DML komutunun kural eklenmeden önceki ve sonraki hali bulunarak etkisinin ölçülmesi ve uygulamamızın gerçekleşme zamanlarının bulunması.
xiii. Bulunan değerle ışığında kazanılan sürelerin yüzde olarak değerlerin ve grafiklerin elde edilmesi.

5.2 Araştırmanın İleriye Dönük Hedefleri

i. Kuralların, DML'lerin en genel hali dışında diğer durumları içinde uygulanabilmesi.

- Select <Sütun Adı>
  From <Tablo Adı>
  Where <Sütun_Adı> in (Select <Sütun Adı>
  From <İlişik Tablo_Adı>);

Örnek :

Aşağıdaki örnekte Prst0100(Hasta Tanım) tablosunda bulunan hastalardan Prst0101(Hasta Ücret Tipi) tablosundaki ücret tipi '3' olan hastaları getiren DML ifade bulunmaktadır.

```
Select * From Prst0100
Where Prs_Pat_No in (Select Prs_Pat_No
                     From Prst0101
                     Where Prs_Price_Type = '3');
```

- Update <Tablo Adı>
  Set <Sütun_Adı> = <Değer>
  Where < Sütun_Adı > in (Select <Sütun Adı>
  From <İlişik Tablo_Adı>);

Örnek :

Aşağıdaki örnekte Prst0100(Hasta Tanım) tablosunda bulunan hastalardan Prst0101(Hasta Ücret Tipi) tablosundaki ücret tipi '3' olan hastaların PRst0100 tablosundaki ücret durum alanını 'O' olarak değiştiren DML ifade bulunmaktadır.

```
Update Prst0100
    Set Prs_Price_State = 'O'
Where Prs_Pat_No in (Select Prs_Pat_No
```

From Prst0101
Where Prs_Price_Type = '3');

- Delete <Tablo Adı>
  Where < Sütun_Adı > in (Select <Sütun Adı>
  From <İlişik Tablo_Adı>);

Örnek :

Aşağıdaki örnekte Prst0100(Hasta Tanım) tablosunda bulunan hastalardan Prst0101(Hasta Ücret Tipi) tablosundaki ücret tipi '8' olan hastaların silinmesini sağlayan DML ifade bulunmaktadır.

```
Delete Prst0100
Where Prs_Pat_No in (Select Prs_Pat_No
                     From Prst0101
                     Where Prs_Price_Type = '8');
```

- Insert Into <Tablo Adı>
  Select <Sütun Adı>
  From <Kayıt Alınacak Tablo Adı>
  Where <Sütun_Adı> in (Select <Sütun Adı>
  From<Kayıt Alınacak İlişik Tablo_Adı>);

Örnek :

Aşağıdaki örnekte Prst0100(Hasta Tanım) tablosunda bulunan hastalardan Prst0101(Hasta Ücret Tipi) tablosundaki ücret tipi '3' olan hastaları Prst0102 tablosuna ekleyen DML ifade bulunmaktadır.

```
Insert Into Prst0102
Select *
From Prst0100
Where Prs_Pat_No in (Select Prs_Pat_No
                     From Prst0101
                     Where Prs_Price_Type = '3');
```

ii. Kuralların, nesne ilişkileri içeren(join) DML ifadelerin için de kullanılabilmesi.

- Select <Sütun Adı>
  From <1. Tablo Adı>, <2. Tablo Adı>.....
  Where <Tablolar Arası İlişkiler> and <Verilen_Koşul>;

Örnek :

Aşağıdaki örnekte Prst0100 (Hasta tanım) tablosu ile Tnmt0101(İl tanımları) bulunduğu tablo arasında ilişki kurularak Prst0100'de bulunan hastaların il kodu '34'den büyük olanların il isimlerini getiren DML ifade bulunmaktadır.

```
Select Tnm_City_Name
 From Prst0100 p1, Tnmt0101 t1
Where p1.Prs_City_Code = t1.Tnm_City_Code
  And t1.Prs_City_Code > '34';
```

KAYNAKLAR

Dennis M. R. ve Brain W K. (1978), C Programming Language.

Elmasri R. ve Navathe S.B. (1994), Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company, 1994

Korth H.F. ve Silberschartz A.(1991), Database System Concepts. McGraw-Hill Book Company.

Kuijk V. (1991), Semantic query optimization in distributed database systems. Doktora Tezi, Twente Üniversitesi.

Lowden B.G.T. ve Lim K.Y. (1995), A data-driven semantic optimizer. In Proceeding of ISCISX Konferans Notları, Türkiye.

Sayli A. ve Lowden B.G.T. (1996), The use of statistics in semantic query optimization. Thirteenth European Meeting on Cybernetics and Systems Research, Vienna, 991-996

Sayli A. ve Lowden B.G.T. (1997)a, A fast transformation method to semantic query optimization, International Database Engineering and Application Symposium, Canada.

Sayli A. ve Lowden B.G.T. (1997)b, Reducing rule effectiveness in SQ Transformation, Symposium on Methodologies for Intelligent Systems, North California.

Sayli A. (1998), Semantic Query Optimization in RDBMS, Doktora Tezi, Essex Üniversitesi.

Oracle (1993), Oracle7 Server SQL Ref. Man.

Cardenas A.F. (1975), Analysis and performance of inverted database structures, Communications of the ACM, 18: 253-263.

Codd E.F. (1970), A. relational model of data for large shared data banks, Communication of the ACM, 13:377-387.

Hsu C. ve Knoblock C.A. (1993), Learning database abstractions for query reformulation Knowledge Discovery in Database Workshop, 276-290.

Minker J. (1996) - Logic and database, In ACTA Informatica, 227-253.

Piatetsky G. ve Shapiro (1991), Discovery, analysis and presentation of strong rules. Knowledge Discovery in Database, 229-248.

Siegel M.D. (1988), Automatic rule derivation for semantic query optimization, 2. International Conference on Expert Database Systems, 669-698.

EKLER

Ek 1 PRST0100 Tablo veri bilgileri (100 Kayıtlık)

Çizelge Ek1.1 Prst0100 tablo verileri

Hasta No	Hasta Ad	Hasta Soyad	Doğum Tar.	Doğ Yer.	Ev Tel.	İş Tel.	İl	Ülke	Cin.	Posta Kod
00000000	GOKAY	GÖK	01.01.1984	70	2163145		32	TR	E	81260
00000001	EMEL	ÖZYILDIZ	16.10.1976	34	11111		34	TR	E	81260
00000002	ZEHRA	YÜCEL	01.01.1984	6	3393541	4176304	34	TR	E	81260
00000003	ALİ	CEYHUN	03.01.2000	1	21312		21	TR	E	81260
00000004	MESUT	CEVİZ	01.01.1988	1	8123134		5	TR	E	81260
00000005	YILDIRIM	KÜMELİ	01.01.1967	34	6234366		34	AB	E	81260
00000006	GÖKÇE	KUŞCU	01.01.1966	1	8654371		5	TR	E	81260
00000007	MEHMET	ÇATAN	01.01.1933	1	4567456		6	TR	E	81260
00000008	BÜLENT	DENİZ	01.01.1976	34	2122122	2122234	3	TR	E	81260
00000009	CEM	HUYSUZ	01.01.1972	1	2164492		5	TR	E	81260
00000010	SEZER	GÜNEY	01.01.1980	1	.	2121234	6	TR	E	81260
00000011	GÖKÇE	SEZGİN	01.01.1977	35	1		7	TR	E	81260
00000012	GULER	CAKMAK	01.03.2000	34	3491458		8	TR	K	81260
00000013	FİLİZ	ŞAHİN	28.11.2000	34	5662452		9	TR	K	81260
00000014	NİZAMETTİN BARIŞ	DİKİLİTAŞ	31.08.1975	6	3802094		34	TR	E	81260
00000015	AYÇA	BAŞBUĞA	17.08.1976	34	0532		34	TR	K	81260
00000016	NİLGÜN	ARGA	26.06.1975	34	4250977		11	TR	K	81260
00000017	IŞIK	GECECİ	01.01.1959	36	3441070		35	TR	K	81260
00000018	GÜL ÖZMEN	SAYLIK	01.01.1966	44	3493025		35	TR	K	81260
00000019	SEMRA	NAMLI	23.11.1971	34	6247269		35	TR	K	81260
00000020	EMİNE	BÖYET	10.10.1959	42	3716069		35	TR	K	81260
00000021	AHMET	DİLSİZ	01.01.1938	35	3589045		35	TR	E	81260
00000022	EVREN	KAYMANLI	12.09.1980	35	8654383	879879	35	TR	K	81260
00000023	SAMİ	YILGÖREN	10.11.1926	35	347 8758		35	TR	E	81260
00000024	ZEHRA GÜLİN	ONUR	29.04.1962	6	3469277		34	TR	K	81260
00000025	HAKAN	GÜNERİ	01.01.1974	42	2122113	2121231	6	TR	K	81260
00000026	AYTEN	GÜÇLÜ	14.12.1936	34	3459863		35	TR	K	81260
00000027	NERMİN	TURAN	04.06.1974	34	49125 46		35	TR	K	81260
00000028	DENEME	RUNTIME	31.01.2000	34	7231234		35	TR	K	81260
00000029	ÖZLEM	ALTAŞ	01.01.1977	32	3252456		35	TR	K	81260
00000030	MİNE	ALTAŞ	25.01.1964	34	3254556	2154556	34	TR	K	81260
00000031	ZEYNEP	MEMİYA	01.01.1944	3	2154565	456525	34	TR	K	81260
00000032	BURCU	BAYCAN	01.01.1972	34	3264515	3254565	35	TR	K	81260
00000033	HANİFE	CİVELEK	01.01.1938	34	3474641	2122001	35	TR	K	81260
00000034	DERYA	ÖZSÖZGÜN	01.01.1964	34	2154566	4521213	35	TR	K	81260
00000035	BİRAY	AKARTUNA	01.01.1965	34	4566211		35	TR	E	
00000036	ADİL	OKUR	01.01.1971	34	3265245	2653542	35	TR	E	81260
00000037	UMUT	GÜNEY	01.01.1991	34	3256566	1252245	6	TR	E	81260
00000038	KEZBAN NESLİHAN	OKUŞ	29.05.1961	34	3377476		6	TR	K	81260
00000039	AYSUN	EREN	31.05.1971	34	3057378		6	TR	K	81260
00000040	MELEK	KALEM	15.09.1980	59	4912647	4492222	60	TR	K	81260
00000041	MİNA	EKER	25.05.1998	34	3257495		6	TR	K	81260
00000042	NEŞET	BÜYÜKYILMAZ	01.01.1954	34	3464418		6	TR	E	81260
00000043	MELİHA	KARAGÖZ	26.01.1973	60	2126553		60	TR	K	81260
00000044	BELGİN	ÜNSAL	25.12.1972	34	5324152		7	TR	K	81260
00000045	AYŞEGÜL	ÖZDUMAN	01.01.1981	34	2163342		8	TR	K	81260

00000046	ENVER	DURMAZ	01.01.1924	10	3370727		34	TR	E	81260
00000047	ELİF	ÖZTURAN	01.01.1980	72	5665973	4492222	34	TR	K	81260
00000048	MELTEM	AÇAN ABAT	08.06.1967	34	4893701	4178034	9	TR	K	81260
00000049	İLKAY	ÇİNER	07.01.1977	34	3526799		11	TR	K	81260
00000050	GÜLNAZ	ORDU	29.01.1975	34	4929227		12	TR	K	81260
00000051	FUNDA	ÇAĞDAŞ	01.01.1975	34	3287391	4492222	16	TR	K	81260
00000052	RAHMİ	ARSLAN	03.09.1973	43	5323219		25	TR	E	81260
00000053	EGE	GÜRSEL	24.05.1995	34	3336584		24	TR	K	81260
00000054	HÜLYA	MUSLU	01.01.1978	34	3437282	4492222	27	TR	K	81260
00000055	ARZU	KULAK	01.01.1981	63	4566354		28	TR	K	81260
00000056	ZEHRA	AKÇASU	20.10.1972	34	3369926		29	TR	K	81260
00000057	AYLİN	MATRACI	21.12.1980	32	4492222		40	TR	K	81260
00000058	SİBEL	KESKİN	01.01.1979	1	3786897		41	TR	E	81260
00000059	DENİZ	CAN	01.01.1979	1	4492222		42	TR	E	81260
00000060	ELÇİN	DEMİR	01.01.1980	1	2222222		42	TR	K	81260
00000061	TEMEL	SADIK	01.01.1955	47	4257896	2475263	50	TR	E	81260
00000062	HAKAN	MUSLU	01.01.1973	34	4663413	7576763	42	TR	E	81260
00000063	SERDAR	ORTAÇ	01.01.1978	34	4686810	5635463	43	TR	E	81260
00000064	MUSTAFA	KÖSELER	17.09.1979	45	3361902	5424849	34	TR	E	81260
00000065	ORHAN	COŞKUN	01.01.1975	58	4493542	4492222	34	TR	E	81260
00000066	GÜL	EFİLOĞLU	01.01.1980	34	3369874	4492222	34	TR	K	81260
00000067	DAMLA	YÜCEL	01.01.1994	6	3599539		34	TR	K	81260
00000068	ŞAHİKA	NANTO	01.01.1971	60	4163872		34	TR	K	81260
00000069	ERHAN	SÖNMEZOCAK	01.01.1990	34	1234545	7851325	34	TR	E	81260
00000070	SADİ	MUSLU	01.01.1954	34	1235467	5536546	34	TR	E	81260
00000071	METE	GÜLTEKİN	01.01.1990	34	1643630	1346463	34	TR	E	81260
00000072	SÜHEYLA	MUSLU	01.01.1966	34	4746330	424341	34	TR	E	81260
00000073	KADİR	ELGÜN	23.11.1971	34	2164146		34	TR	K	81260
00000074	KAAN	MUSLU	01.01.1999	34	2334561	456317	34	TR	E	81260
00000075	HÜSEYİN	YAZICI	01.01.1971	58	4545665	1564565	34	TR	E	81260
00000076	HÜLYA	AKDENİZ	01.01.1978	34	4754410		34	TR	E	81260
00000077	TÜLİN	KURUOĞLU	07.05.1966	26	3954066		34	TR	K	81260
00000078	SAADET	ÇETİN	10.04.1976	34	3867974		34	TR	K	81260
00000079	ABDULLAH	YAZICI	01.01.1979	1	6545645		34	TR	E	81260
00000080	HEVES	AKIN	01.01.1978	67	4565656		34	TR	E	81260
00000081	ESRA	DEVİREN	10.04.1970	58	4284632		34	TR	K	81260
00000082	PINAR	EKŞİOĞLU	02.02.1978	34	3344743		34	TR	K	81260
00000083	DOĞUKAN	GEMİCİ	01.01.1990	34	3020560		34	TR	E	81260
00000084	ÖZLEM	ÖZGENEL	16.12.1975	34	4591895	4284650	34	TR	K	81260
00000085	GÜLEN	ALTUGAY	01.01.1931	34	3376168		34	TR	K	81260
00000086	NESRİN	KALAY	01.01.1978	34	1245124	4364641	34	TR	E	81260
00000087	ZEYNEP	ÜLKENCİLER	08.04.1975	34	3484660		34	TR	K	81260
00000088	ERGÜN	KALAY	01.01.1945	1	4763471	6263547	34	TR	E	81260
00000089	SİBEL	KARAKUŞ	01.01.1995	1	4565645	6454565	34	TR	E	81260
00000090	RÜKSAN	KALAY	01.01.1965	5	1245700	124578	34	TR	E	81260
00000091	ZEYNEP	ERBERK	19.12.1972	34	2163027		34	TR	K	81260
00000092	KAAN	ÖZDUMAN	01.01.1983	34	3342217		34	TR	E	81260
00000093	ATİLLA	KIZILALP	01.01.1948	1	1421466	5566655	34	TR	E	81260
00000094	TÜRKAN	SAYKAL	01.01.1959	34	3342217		34	TR	K	81260
00000095	IŞIL	ÖZKAN	01.04.1962	36	3441070	212	34	TR	K	81260
00000096	SERPİL	KESKİN	01.01.1984	2	4492222		34	TR	K	81260
00000097	FETİ	ÇELİKSU	01.01.1929	7	1233452	2525246	34	TR	E	81260
00000098	NALAN	ÇELİKSU	01.01.1960	1	1345631	5525452	34	TR	E	81260
00000099	TEYFİK	AK	01.01.1975	1	1241466	5548816	34	TR	E	81260

Ek 2 PRST0100 Tablosu için kullanılan kurallar (100 Kayıtlık)

R1) PRST0100.PRS_BIRTH_PLACE_CODE='06'---4---46---PRST0100.PRS_CITY_CODE='34'
R2) PRST0100.PRS_BIRTH_PLACE_CODE='06'---4---25---PRST0100.PRS_CITY_CODE>'33'
R3) PRST0100.PRS_BIRTH_PLACE_CODE='06'---4---71---PRST0100.PRS_CITY_CODE>='34'
R4) PRST0100.PRS_BIRTH_PLACE_CODE='06'---4---29---PRST0100.PRS_CITY_CODE<'35'
R5) PRST0100.PRS_BIRTH_PLACE_CODE='06'---4---75---PRST0100.PRS_CITY_CODE<='34'
R6) PRST0100.PRS_BIRTH_PLACE_CODE='06'---4---54---PRST0100.PRS_CITY_CODE!='33'
R7) PRST0100.PRS_BIRTH_PLACE_CODE>'06'---78---80---PRST0100.PRS_COUNTRY_CODE='TR'
R8) PRST0100.PRS_BIRTH_PLACE_CODE>'06'---78---88---PRST0100.PRS_NAME_GIVEN>'BAYCAN'
R9) PRST0100.PRS_BIRTH_PLACE_CODE>'06'---78---89---PRST0100.PRS_NAME_GIVEN>='BAYCAN'
R10) PRST0100.PRS_BIRTH_PLACE_CODE>'06'---78---79---PRST0100.PRS_NAME_GIVEN<'TURAN'
R11) PRST0100.PRS_BIRTH_PLACE_CODE>'06'---78---80---PRST0100.PRS_NAME_GIVEN<='TURAN'
R12) PRST0100.PRS_BIRTH_PLACE_CODE>'06'---78---99---PRST0100.PRS_SMOKER_Y!='E'
R13) PRST0100.PRS_BIRTH_PLACE_CODE<'06'---18---46---PRST0100.PRS_CITY_CODE='34'
R14) PRST0100.PRS_BIRTH_PLACE_CODE<'06'---18---25---PRST0100.PRS_CITY_CODE>'33'
R15) PRST0100.PRS_BIRTH_PLACE_CODE<'06'---18---71---PRST0100.PRS_CITY_CODE>='34'
R16) PRST0100.PRS_BIRTH_PLACE_CODE<'06'---18---29---PRST0100.PRS_CITY_CODE<'35'
R17) PRST0100.PRS_BIRTH_PLACE_CODE<'06'---18---75---PRST0100.PRS_CITY_CODE<='34'
R18) PRST0100.PRS_BIRTH_PLACE_CODE<'06'---18---54---PRST0100.PRS_CITY_CODE!='33'
R19) PRST0100.PRS_BIRTH_PLACE_CODE!='06'---98---99---PRST0100.PRS_SMOKER_Y=''
R20) PRST0100.PRS_BIRTH_PLACE_CODE!='06'---98---99---PRST0100.PRS_NAME_GIVEN>'AK'
R21) PRST0100.PRS_BIRTH_PLACE_CODE!='06'---98---100---PRST0100.PRS_NAME_GIVEN>='AK'
R22) PRST0100.PRS_BIRTH_PLACE_CODE!='06'---98---98---PRST0100.PRS_CITY_CODE<'50'
R23) PRST0100.PRS_BIRTH_PLACE_CODE!='06'---98---99---PRST0100.PRS_CITY_CODE<='50'
R24) PRST0100.PRS_BIRTH_PLACE_CODE!='06'---98---99---PRST0100.PRS_SMOKER_Y!='E'
R25) PRST0100.PRS_BIRTH_DATE='10/11/1926'---1---1---PRST0100.PRS_PAT_NO='00000023'
R26) PRST0100.PRS_BIRTH_DATE='10/11/1926'---1---77---PRST0100.PRS_PAT_NO>'00000022'
R27) PRST0100.PRS_BIRTH_DATE='10/11/1926'---1---77---PRST0100.PRS_PAT_NO>='00000023'
R28) PRST0100.PRS_BIRTH_DATE='10/11/1926'---1---24---PRST0100.PRS_PAT_NO<'00000024'
R29) PRST0100.PRS_BIRTH_DATE='10/11/1926'---1---24---PRST0100.PRS_PAT_NO<='00000023'
R30) PRST0100.PRS_BIRTH_DATE='10/11/1926'---1---99---PRST0100.PRS_PAT_NO!='00000022'

R31) PRST0100.PRS_BIRTH_PLACE_CODE>'71'---1---1---PRST0100.PRS_PAT_NO='00000042'
R32) PRST0100.PRS_BIRTH_PLACE_CODE>'71'---1---58---PRST0100.PRS_PAT_NO>'00000041'
R33) PRST0100.PRS_BIRTH_PLACE_CODE>'71'---1---58---PRST0100.PRS_PAT_NO>='00000042'
R34) PRST0100.PRS_BIRTH_PLACE_CODE>'71'---1---43---PRST0100.PRS_PAT_NO<'00000043'
R35) PRST0100.PRS_BIRTH_PLACE_CODE>'71'---1---43---PRST0100.PRS_PAT_NO<='00000042'
R36) PRST0100.PRS_BIRTH_PLACE_CODE>'71'---1---99---PRST0100.PRS_PAT_NO!='00000041'
R37) PRST0100.PRS_CITY_CODE<'05'---1---1---PRST0100.PRS_PAT_NO='00000008'
R38) PRST0100.PRS_CITY_CODE<'05'---1---91---PRST0100.PRS_PAT_NO>'00000007'
R39) PRST0100.PRS_CITY_CODE<'05'---1---91---PRST0100.PRS_PAT_NO>='00000008'
R40) PRST0100.PRS_CITY_CODE<'05'---1---9---PRST0100.PRS_PAT_NO<'00000009'
R41) PRST0100.PRS_CITY_CODE<'05'---1---9---PRST0100.PRS_PAT_NO<='00000008'
R42) PRST0100.PRS_CITY_CODE<'05'---1---99---PRST0100.PRS_PAT_NO!='00000007'
R43) PRST0100.PRS_COUNTRY_CODE!='TR'---1---1---PRST0100.PRS_PAT_NO='00000005'
R44) PRST0100.PRS_COUNTRY_CODE!='TR'---1---94---PRST0100.PRS_PAT_NO>'00000004'
R45) PRST0100.PRS_COUNTRY_CODE!='TR'---1---94---PRST0100.PRS_PAT_NO>='00000005'
R46) PRST0100.PRS_COUNTRY_CODE!='TR'---1---6---PRST0100.PRS_PAT_NO<'00000006'
R47) PRST0100.PRS_COUNTRY_CODE!='TR'---1---6---PRST0100.PRS_PAT_NO<='00000005'
R48) PRST0100.PRS_COUNTRY_CODE!='TR'---1---99---PRST0100.PRS_PAT_NO!='00000004'
R49) PRST0100.PRS_NAME_FAMILY='ŞAHİKA'---1---1---PRST0100.PRS_PHONE_HOME='4163872'
R50) PRST0100.PRS_NAME_FAMILY='ŞAHİKA'---1---31---PRST0100.PRS_PHONE_HOME>'3954066'
R51) PRST0100.PRS_NAME_FAMILY='ŞAHİKA'---1---31---PRST0100.PRS_PHONE_HOME>='4163872'
R52) PRST0100.PRS_NAME_FAMILY='ŞAHİKA'---1---69---PRST0100.PRS_PHONE_HOME='425097'
R53) PRST0100.PRS_NAME_FAMILY='ŞAHİKA'---1---69---PRST0100.PRS_PHONE_HOME<='4163872'
R54) PRST0100.PRS_NAME_FAMILY='ŞAHİKA'---1---99---PRST0100.PRS_PHONE_HOME!=''
R55) PRST0100.PRS_BIRTH_DATE>'01/03/2000'---1---1---PRST0100.PRS_PHONE_HOME='3491458'
R56) PRST0100.PRS_BIRTH_DATE>'01/03/2000'---1---44---PRST0100.PRS_PHONE_HOME>'3484660'
R57) PRST0100.PRS_BIRTH_DATE>'01/03/2000'---1---44---PRST0100.PRS_PHONE_HOME>='3491458'
R58) PRST0100.PRS_BIRTH_DATE>'01/03/2000'---1---56---PRST0100.PRS_PHONE_HOME<'3493025'
R59) PRST0100.PRS_BIRTH_DATE>'01/03/2000'---1---56---PRST0100.PRS_PHONE_HOME<='3491458'
R60) PRST0100.PRS_BIRTH_DATE>'01/03/2000'---1---99---PRST0100.PRS_PHONE_HOME!='3484660'
R61) PRST0100.PRS_BIRTH_DATE>'10/11/1926'---1---1---PRST0100.PRS_PHONE_HOME='3478758'
R62) PRST0100.PRS_BIRTH_DATE>'10/11/1926'---1---45---PRST0100.PRS_PHONE_HOME>'3474641'
R63) PRST0100.PRS_BIRTH_DATE>'10/11/1926'---1---45---PRST0100.PRS_PHONE_HOME>='3478758'

R64) PRST0100.PRS_BIRTH_DATE>'10/11/1926'---1---55---PRST0100.PRS_PHONE_HOME='3484660'
R65) PRST0100.PRS_BIRTH_DATE>'10/11/1926'---1---55---PRST0100.PRS_PHONE_HOME<='3478758'
R66) PRST0100.PRS_BIRTH_DATE>'10/11/1926'---1---99---PRST0100.PRS_PHONE_HOME!='3474641'
R67) PRST0100.PRS_COUNTRY_CODE!='TR'---1---1---PRST0100.PRS_PHONE_HOME='6234366'
R68) PRST0100.PRS_COUNTRY_CODE!='TR'---1---6---PRST0100.PRS_PHONE_HOME>'5665973'
R69) PRST0100.PRS_COUNTRY_CODE!='TR'---1---6---PRST0100.PRS_PHONE_HOME>='6234366'
R70) PRST0100.PRS_COUNTRY_CODE!='TR'---1---94---PRST0100.PRS_PHONE_HOME<'6247269'
R71) PRST0100.PRS_COUNTRY_CODE!='TR'---1---94---PRST0100.PRS_PHONE_HOME<='6234366'
R72) PRST0100.PRS_COUNTRY_CODE!='TR'---1---99---PRST0100.PRS_PHONE_HOME!='6247269'
R73) PRST0100.PRS_PAT_NO='00000035'---1---16---PRST0100.PRS_CITY_CODE='35'
R74) PRST0100.PRS_PAT_NO='00000035'---1---25---PRST0100.PRS_CITY_CODE>'34'
R75) PRST0100.PRS_PAT_NO='00000035'---1---25---PRST0100.PRS_CITY_CODE>='35'
R76) PRST0100.PRS_PAT_NO='00000035'---1---91---PRST0100.PRS_CITY_CODE<'36'
R77) PRST0100.PRS_PAT_NO='00000035'---1---91---PRST0100.PRS_CITY_CODE<='35'
R78) PRST0100.PRS_PAT_NO='00000035'---1---99---PRST0100.PRS_CITY_CODE!='36'
R79) PRST0100.PRS_PAT_NO>'00000090'---9---99---PRST0100.PRS_COUNTRY_CODE='TR'
R80) PRST0100.PRS_PAT_NO>'00000090'---9---29---PRST0100.PRS_CITY_CODE>'32'
R81) PRST0100.PRS_PAT_NO>'00000090'---9---71---PRST0100.PRS_CITY_CODE>='34'
R82) PRST0100.PRS_PAT_NO>'00000090'---9---75---PRST0100.PRS_CITY_CODE<'35'
R83) PRST0100.PRS_PAT_NO>'00000090'---9---75---PRST0100.PRS_CITY_CODE<='34'
R84) PRST0100.PRS_PAT_NO>'00000090'---9---85---PRST0100.PRS_CITY_CODE!='35'
R85) PRST0100.PRS_PAT_NO<'00000010'---10---47---PRST0100.PRS_SEX='E'
R86) PRST0100.PRS_PAT_NO<'00000010'---10---85---PRST0100.PRS_BIRTH_PLACE_CODE>'01'
R87) PRST0100.PRS_PAT_NO<'00000010'---10---99---PRST0100.PRS_CITY_CODE>='04'
R88) PRST0100.PRS_PAT_NO<'00000010'---10---75---PRST0100.PRS_CITY_CODE<'35'
R89) PRST0100.PRS_PAT_NO<'00000010'---10---75---PRST0100.PRS_CITY_CODE<='34'
R90) PRST0100.PRS_PAT_NO<'00000010'---10---85---PRST0100.PRS_CITY_CODE!='35'
R91) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE='81260'
R92) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE>''
R93) PRST0100.PRS_PAT_NO!='00000005'---99---100---PRST0100.PRS_ZIP_CODE>=''
R94) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE<'81260'
R95) PRST0100.PRS_PAT_NO!='00000005'---99---100---PRST0100.PRS_ZIP_CODE<='81260'
R96) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE!=''

R97) PRST0100.PRS_PAT_NO='00000050'---1---47---PRST0100.PRS_SEX='K'
R98) PRST0100.PRS_PAT_NO='00000050'---1---53---PRST0100.PRS_SEX>'E'
R99) PRST0100.PRS_PAT_NO='00000050'---1---100---PRST0100.PRS_SEX>='E'
R100) PRST0100.PRS_PAT_NO='00000050'---1---99---PRST0100.PRS_SEX<'X'
R101) PRST0100.PRS_PAT_NO='00000050'---1---99---PRST0100.PRS_SEX<='K'
R102) PRST0100.PRS_PAT_NO='00000050'---1---53---PRST0100.PRS_SEX!='E'
R103) PRST0100.PRS_PAT_NO>'00000096'---3---47---PRST0100.PRS_SEX='K'
R104) PRST0100.PRS_PAT_NO>'00000096'---3---53---PRST0100.PRS_SEX>'E'
R105) PRST0100.PRS_PAT_NO>'00000096'---3---100---PRST0100.PRS_SEX>='E'
R106) PRST0100.PRS_PAT_NO>'00000096'---3---99---PRST0100.PRS_SEX<'X'
R107) PRST0100.PRS_PAT_NO>'00000096'---3---99---PRST0100.PRS_SEX<='K'
R108) PRST0100.PRS_PAT_NO>'00000096'---3---53---PRST0100.PRS_SEX!='E'
R109) PRST0100.PRS_PAT_NO<'00000012'---12---47---PRST0100.PRS_SEX='K'
R110) PRST0100.PRS_PAT_NO<'00000012'---12---53---PRST0100.PRS_SEX>'E'
R111) PRST0100.PRS_PAT_NO<'00000012'---12---100---PRST0100.PRS_SEX>='E'
R112) PRST0100.PRS_PAT_NO<'00000012'---12---99---PRST0100.PRS_SEX<'X'
R113) PRST0100.PRS_PAT_NO<'00000012'---12---99---PRST0100.PRS_SEX<='K'
R114) PRST0100.PRS_PAT_NO<'00000012'---12---53---PRST0100.PRS_SEX!='E'
R115) PRST0100.PRS_SEX!='E'---53---99---PRST0100.PRS_COUNTRY_CODE='TR'
R116) PRST0100.PRS_SEX!='E'---53'---1---100---PRST0100.PRS_COUNTRY_CODE>"
R117) PRST0100.PRS_SEX!='E'---53'---1---99---PRST0100.PRS_COUNTRY_CODE>='TR'
R118) PRST0100.PRS_SEX!='E'---53'---1---100---PRST0100.PRS_SEX<'X'
R119) PRST0100.PRS_SEX!='E'---53'---1---99---PRST0100.PRS_COUNTEY_CODE<='TR'
R120) PRST0100.PRS_SEX!='E'---53---1---100---PRST0100.PRS_COUNTRY_CODE!="
R121) PRST0100.PRS_PAT_NO='000000057'---1---3---PRST0100.PRS_PHONE_HOME='4492222'
R122) PRST0100.PRS_PAT_NO='000000057'---1---29---PRST0100.PRS_PHONE_HOME>'4284632'
R123) PRST0100.PRS_PAT_NO='000000057'---1---29---PRST0100.PRS_PHONE_HOME>='4492222'
R124) PRST0100.PRS_PAT_NO='000000057'---1---75---PRST0100.PRS_PHONE_HOME<'4492592'
R125) PRST0100.PRS_PAT_NO='000000057'---1---75---PRST0100.PRS_PHONE_HOME<='4492222'
R126) PRST0100.PRS_PAT_NO='000000057'---1---97---PRST0100.PRS_PHONE_HOME!="
R127) PRST0100.PRS_PAT_NO>'00000098'---1---1---PRST0100.PRS_PHONE_HOME='1241466'
R128) PRST0100.PRS_PAT_NO>'00000098'---1---83---PRST0100.PRS_PHONE_HOME>'1235467'
R129) PRST0100.PRS_PAT_NO>'00000098'---1---83---PRST0100.PRS_PHONE_HOME>='1241466'

R130) PRST0100.PRS_PAT_NO>'00000098'---1---18---PRST0100.PRS_PHONE_HOME<='12451244'
R131) PRST0100.PRS_PAT_NO>'00000098'---1---18---PRST0100.PRS_PHONE_HOME<='1241466'
R132) PRST0100.PRS_PAT_NO>'00000098'---1---97---PRST0100.PRS_PHONE_HOME!='4492222'
R133) PRST0100.PRS_PAT_NO<'00000001'---1---1---PRST0100.PRS_PHONE_HOME='02163145'
R134) PRST0100.PRS_PAT_NO<'00000001'---1---92---PRST0100.PRS_PHONE_HOME>'02163026'
R135) PRST0100.PRS_PAT_NO<'00000001'---1---92---PRST0100.PRS_PHONE_HOME>='02163145'
R136) PRST0100.PRS_PAT_NO<'00000001'---1---8---PRST0100.PRS_PHONE_HOME<'02163342217'
R137) PRST0100.PRS_PAT_NO<'00000001'---1---8---PRST0100.PRS_PHONE_HOME<='02163145'
R138) PRST0100.PRS_PAT_NO<'00000001'---1---3---PRST0100.PRS_PHONE_HOME!='4492222'
R139) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE='81260'
R140) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE>"
R141) PRST0100.PRS_PAT_NO!='00000005'---99---100---PRST0100.PRS_ZIP_CODE>="
R142) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE<'81260'
R143) PRST0100.PRS_PAT_NO!='00000005'---99---100---PRST0100.PRS_ZIP_CODE<='81260'
R144) PRST0100.PRS_PAT_NO!='00000005'---99---99---PRST0100.PRS_ZIP_CODE!="
R145) PRST0100.PRS_PHONE_HOME='3441070'---2---2---PRST0100.PRS_BIRTH_PLACE_CODE='06'
R146) PRST0100.PRS_PHONE_HOME='3441070'---2---71---PRST0100.PRS_CITY_CODE>'33'
R147) PRST0100.PRS_PHONE_HOME='3441070'---2---71---PRST0100.PRS_CITY_CODE>='34'
R148) PRST0100.PRS_PHONE_HOME='3441070'---2---91---PRST0100.PRS_CITY_CODE<'36'
R149) PRST0100.PRS_PHONE_HOME='3441070'---2---91---PRST0100.PRS_CITY_CODE<='35'
R150) PRST0100.PRS_PHONE_HOME='3441070'---2---15---PRST0100.PRS_CITY_CODE!='01'
R151) PRST0100.PRS_PHONE_HOME>'654564546'---2---3---PRST0100.PRS_CITY_CODE='05'
R152) PRST0100.PRS_PHONE_HOME>'654564546'---2---71---PRST0100.PRS_BIRTH_DATE>'01/01/1965'
R153) PRST0100.PRS_PHONE_HOME>'654564546'---2---71---PRST0100.PRS_BIRTH_DATE>='01/01/1966'
R154) PRST0100.PRS_PHONE_HOME>'654564546'---2---4---PRST0100.PRS_CITY_CODE<'06'
R155) PRST0100.PRS_PHONE_HOME>'654564546'---2---4---PRST0100.PRS_CITY_CODE=<'05'
R156) PRST0100.PRS_PHONE_HOME>'654564546'---2---47---PRST0100.PRS_SEX!='K'
R157) PRST0100.PRS_PHONE_HOME<'02122122121'---2---8---PRST0100.PRS_CITY_CODE='06'
R158) PRST0100.PRS_PHONE_HOME<'02122122121'---2---65---PRST0100.PRS_BIRTH_DATE>'03/09/1973'
R159) PRST0100.PRS_PHONE_HOME<'02122122121'---2---65---PRST0100.PRS_BIRTH_DATE>='01/01/1974'
R160) PRST0100.PRS_PHONE_HOME<'02122122121'---2---12---PRST0100.PRS_CITY_CODE<'07'
R161) PRST0100.PRS_PHONE_HOME<'02122122121'---2---12---PRST0100.PRS_CITY_CODE<='06'
R162) PRST0100.PRS_PHONE_HOME<'02122122121'---2---34---PRST0100.PRS_CITY_CODE!='34'

R163) PRST0100.PRS_PHONE_HOME!='6234366'---99---99---PRST0100.PRS_COUNTRY_CODE='TR'
R164) PRST0100.PRS_PHONE_HOME!='6234366'---99---99---PRST0100.PRS_COUNTRY_CODE>'AB'
R165) PRST0100.PRS_PHONE_HOME!='6234366'---99---100---PRST0100.PRS_COUNTRY_CODE>='AB'
R166) PRST0100.PRS_PHONE_HOME!='6234366'---99---99---PRST0100.PRS_SMOKER_Y<'E'
R167) PRST0100.PRS_PHONE_HOME!='6234366'---99---100---PRST0100.PRS_COUNTRY_CODE<='TR'
R168) PRST0100.PRS_PHONE_HOME!='6234366'---99---99---PRST0100.PRS_COUNTRY_CODE!=''
R169) PRST0100.PRS_PHONE_HOME='02122122121'---1---1---PRST0100.PRS_PAT_NO='00000008'
R170) PRST0100.PRS_PHONE_HOME='02122122121'---1---92---PRST0100.PRS_PAT_NO>'00000008'
R171) PRST0100.PRS_PHONE_HOME='02122122121'---1---92---PRST0100.PRS_PAT_NO>='00000008'
R172) PRST0100.PRS_PHONE_HOME='02122122121'---1---8---PRST0100.PRS_PAT_NO<'00000009'
R173) PRST0100.PRS_PHONE_HOME='02122122121'---1---8---PRST0100.PRS_PAT_NO<='00000008'
R174) PRST0100.PRS_PHONE_HOME='02122122121'---1---99---PRST0100.PRS_PAT_NO!='00000007'
R175) PRST0100.PRS_PHONE_HOME>'8654371'---1---1---PRST0100.PRS_PAT_NO='00000022'
R176) PRST0100.PRS_PHONE_HOME>'8654371'---1---80---PRST0100.PRS_PAT_NO>'00000021'
R177) PRST0100.PRS_PHONE_HOME>'8654371'---1---80---PRST0100.PRS_PAT_NO>='00000022'
R178) PRST0100.PRS_PHONE_HOME>'8654371'---1---21---PRST0100.PRS_PAT_NO<'00000023'
R179) PRST0100.PRS_PHONE_HOME>'8654371'---1---21---PRST0100.PRS_PAT_NO<='00000022'
R180) PRST0100.PRS_PHONE_HOME>'8654371'---1---99---PRST0100.PRS_PAT_NO!='00000021'
R181) PRST0100.PRS_PHONE_HOME<'02122113452'---1---1---PRST0100.PRS_PAT_NO='00000010'
R182) PRST0100.PRS_PHONE_HOME<'02122113452'---1---90---PRST0100.PRS_PAT_NO>'00000009'
R183) PRST0100.PRS_PHONE_HOME<'02122113452'---1---90---PRST0100.PRS_PAT_NO>='00000010'
R184) PRST0100.PRS_PHONE_HOME<'02122113452'---1---11---PRST0100.PRS_PAT_NO<'00000011'
R185) PRST0100.PRS_PHONE_HOME<'02122113452'---1---11---PRST0100.PRS_PAT_NO<='00000009'
R186) PRST0100.PRS_PHONE_HOME<'02122113452'---1---99---PRST0100.PRS_PAT_NO!='00000009'
R187) PRST0100.PRS_ZIP_CODE=''---1---1---PRST0100.PRS_PAT_NO='00000035'
R188) PRST0100.PRS_ZIP_CODE=''---1---65---PRST0100.PRS_PAT_NO>'00000034'
R189) PRST0100.PRS_ZIP_CODE=''---1---65---PRST0100.PRS_PAT_NO>='00000035'
R190) PRST0100.PRS_ZIP_CODE=''---1---36---PRST0100.PRS_PAT_NO<'00000036'
R191) PRST0100.PRS_ZIP_CODE=''---1---36---PRST0100.PRS_PAT_NO<='00000035'
R192) PRST0100.PRS_ZIP_CODE=''---1---99---PRST0100.PRS_PAT_NO!='00000034'
R193) PRST0100.PRS_PHONE_HOME='02122122121'---1---1---PRST0100.PRS_PHONE_WORK='02122234'
R194) PRST0100.PRS_PHONE_HOME='02122122121'---1---99---PRST0100.PRS_PHONE_WORK>'02122001'
R195) PRST0100.PRS_PHONE_HOME='02122122121'---1---99---PRST0100.PRS_PHONE_WORK>='02122234'

R196) PRST0100.PRS_PHONE_HOME='02122122121'---1---3---PRST0100.PRS_PHONE_WORK<'05424849'
R197) PRST0100.PRS_PHONE_HOME='02122122121'---1---3---PRST0100.PRS_PHONE_WORK<='02122234'
R198) PRST0100.PRS_PHONE_HOME='02122122121'---1---60---PRST0100.PRS_PHONE_WORK!=''
R199) PRST0100.PRS_PHONE_HOME>'8654371'---1---60---PRST0100.PRS_PHONE_WORK=''
R200) PRST0100.PRS_PHONE_HOME>'8654371'---1---61---PRST0100.PRS_PHONE_WORK<'785132586'
R201) PRST0100.PRS_PHONE_HOME>'8654371'---1---61---PRST0100.PRS_PHONE_WORK>='879879'
R202) PRST0100.PRS_PHONE_HOME>'8654371'---1---39---PRST0100.PRS_PHONE_WORK<''
R203) PRST0100.PRS_PHONE_HOME>'8654371'---1---39---PRST0100.PRS_PHONE_WORK<='879879'
R204) PRST0100.PRS_PHONE_HOME>'8654371'---1---60---PRST0100.PRS_PHONE_WORK!=''
R205) PRST0100.PRS_PHONE_HOME<'02122113452'---1---1---PRST0100.PRS_PHONE_WORK='.'
R206) PRST0100.PRS_PHONE_HOME<'02122113452'---1---38---PRST0100.PRS_PHONE_WORK>'0212121'
R207) PRST0100.PRS_PHONE_HOME<'02122113452'---1---39---PRST0100.PRS_PHONE_WORK>='0212123'
R208) PRST0100.PRS_PHONE_HOME<'02122113452'---1---3---PRST0100.PRS_PHONE_WORK<'021220001'
R209) PRST0100.PRS_PHONE_HOME<'02122113452'---1---3---PRST0100.PRS_PHONE_WORK<='02121231'
R210) PRST0100.PRS_PHONE_HOME<'02122113452'---1---60---PRST0100.PRS_PHONE_WORK!=''
R211) PRST0100.PRS_PHONE_HOME!='45662112'---99---99---PRST0100.PRS_ZIP_CODE='81260'
R212) PRST0100.PRS_PHONE_HOME!='45662112'---99---99---PRST0100.PRS_ZIP_CODE>''
R213) PRST0100.PRS_PHONE_HOME!='45662112'---99---100---PRST0100.PRS_ZIP_CODE>=''
R214) PRST0100.PRS_PHONE_HOME!='45662112'---99---99---PRST0100.PRS_ZIP_CODE<'81260'
R215) PRST0100.PRS_PHONE_HOME!='45662112'---99---100---PRST0100.PRS_ZIP_CODE<='81260'
R216) PRST0100.PRS_PHONE_HOME!='45662112'---99---99---PRST0100.PRS_ZIP_CODE!=''

Ek 3 PRST0100 Tablosunda kuralların kullanımındaki DML'ler (100 Kayıtlık)

İ1) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_PLACE_CODE='06'
İ2) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_PLACE_CODE>'06'
İ3) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_PLACE_CODE<'06'
İ4) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_PLACE_CODE!='06'
İ5) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_DATE='10/11/1926'
İ6) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_PLACE_CODE > '71'
İ7) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_CITY_CODE < '05'
İ8) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_COUNTRY_CODE != 'TR'
İ9) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_NAME_FAMILY='ŞAHİKA'
İ10) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_DATE>'01/03/2000'
İ11) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_BIRTH_DATE>'10/11/1926'
İ12) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_COUNTRY_CODE != 'TR'
İ13) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO='00000035'
İ14) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO>'00000090'
İ15) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO<'00000010'
İ16) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO!='00000005'
İ17) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO='00000050'
İ18) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO>'00000096'
İ19) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO<'00000012'
İ20) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_SEX!='E'
İ21) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO='00000057'
İ22) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO>'00000098'
İ23) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO<'00000001'
İ24) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PAT_NO!='00000005'
İ25) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME='3441070'
İ26) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME>'654564546'
İ27) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME<'02122122121'
İ28) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME!='6234366'
İ29) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME='02122122121'
İ30) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME>'8654371'

İ31) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME<'02122113452'
İ32) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_ZIP_CODE=''
İ33) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME='02122122121'
İ34) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME>'8654371'
İ35) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME<'02122113452'
İ36) UPDATE PRST0100 SET PRS_COUNTRY_CODE = 'TR' WHERE PRS_PHONE_HOME!='45662112'
İ37) DELETE PRST0100 PRS_BIRTH_PLACE_CODE='06'
İ38) DELETE PRST0100 PRS_BIRTH_PLACE_CODE>'06'
İ39) DELETE PRST0100 PRS_BIRTH_PLACE_CODE<'06'
İ40) DELETE PRST0100 PRS_BIRTH_PLACE_CODE!='06'
İ41) DELETE PRST0100 PRS_BIRTH_DATE='10/11/1926'
İ42) DELETE PRST0100 PRS_BIRTH_PLACE_CODE > '71'
İ43) DELETE PRST0100 PRS_CITY_CODE < '05'
İ44) DELETE PRST0100 PRS_COUNTRY_CODE != 'TR'
İ45) DELETE PRST0100 PRS_NAME_FAMILY='ŞAHİKA'
İ46) DELETE PRST0100 PRS_BIRTH_DATE>'01/03/2000'
İ47) DELETE PRST0100 PRS_BIRTH_DATE>'10/11/1926'
İ48) DELETE PRST0100 PRS_COUNTRY_CODE != 'TR'
İ49) DELETE PRST0100 PRS_PAT_NO='00000035'
İ50) DELETE PRST0100 PRS_PAT_NO>'00000090'
İ51) DELETE PRST0100 PRS_PAT_NO<'00000010'
İ52) DELETE PRST0100 PRS_PAT_NO!='00000005'
İ53) DELETE PRST0100 PRS_PAT_NO='00000050'
İ54) DELETE PRST0100 PRS_PAT_NO>'00000050'
İ55) DELETE PRST0100 PRS_PAT_NO<'00000050'
İ56) DELETE PRST0100 PRS_PAT_NO!='00000050'
İ57) DELETE PRST0100 PRS_PAT_NO='000000057'
İ58) DELETE PRST0100 PRS_PAT_NO>'00000098'
İ59) DELETE PRST0100 PRS_PAT_NO<'00000001'
İ60) DELETE PRST0100 PRS_PAT_NO!='00000005'
İ61) DELETE PRST0100 PRS_PHONE_HOME='3441070'
İ62) DELETE PRST0100 PRS_PHONE_HOME>'654564546'
İ63) DELETE PRST0100 PRS_PHONE_HOME<'02122122121'

İ64) DELETE PRST0100 PRS_PHONE_HOME!='6234366'
İ65) DELETE PRST0100 PRS_PHONE_HOME='02122122121'
İ66) DELETE PRST0100 PRS_PHONE_HOME>'8654371'
İ67) DELETE PRST0100 PRS_PHONE_HOME<'02122113452'
İ68) DELETE PRST0100 PRS_ZIP_CODE=''
İ69) DELETE PRST0100 PRS_PHONE_HOME='02122122121'
İ70) DELETE PRST0100 PRS_PHONE_HOME>'8654371'
İ71) DELETE PRST0100 PRS_PHONE_HOME<'02122113452'
İ72) DELETE PRST0100 PRS_PHONE_HOME!='45662112'
İ73) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_PLACE_CODE='06'
İ74) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_PLACE_CODE>'06'
İ75) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_PLACE_CODE<'06'
İ76) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_PLACE_CODE!='06'
İ77) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_DATE='10/11/1926'
İ78) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_PLACE_CODE > '71'
İ79) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_CITY_CODE < '05'
İ80) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_COUNTRY_CODE != 'TR'
İ81) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_NAME_FAMILY='ŞAHİKA'
İ82) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_DATE>'01/03/2000'
İ83) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_BIRTH_DATE>'10/11/1926'
İ84) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_COUNTRY_CODE != 'TR'
İ85) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO='00000035'
İ86) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO>'00000090'
İ87) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO<'00000010'
İ88) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO!='00000005'
İ89) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO='00000050'
İ90) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO>'00000050'
İ91) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO<'00000050'
İ92) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO!='00000050'
İ93) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO='000000057'
İ94) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO>'00000098'
İ95) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO<'00000001'
İ96) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PAT_NO!='00000005'

İ97) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME='3441070'
İ98) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME>'654564546'
İ99) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME<'02122122121'
İ100) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME!='6234366'
İ101) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME='02122122121'
İ102) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME>'8654371'
İ103) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME<'02122113452'
İ104) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_ZIP_CODE=''
İ105) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME='02122122121'
İ106) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME>'8654371'
İ107) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME<'02122113452'
İ108) INSERT INTO PRST1000 SELECT * FROM PRST0100 WHERE PRS_PHONE_HOME!='45662112'

ÖZGEÇMİŞ

Doğum Tarihi	08.10.1977	
Doğum Yeri	İstanbul	
Lise	1990-1993	Prof. Faik Somer Lisesi
Lisans	1995-1999	Yıldız Teknik Üniversitesi Elektrik – Elektronik Fak. Elektronik ve Haberleşme Müh.
Yüksek Lisans	2000-2003	Yıldız Teknik Üniversitesi Matematik Mühendisliği Anabilim Dalı Matematik Mühendisliği Programı

Çalıştığı Kurumlar

2000-2000	Gordion Bilgi Hizmetleri Veritabanı ve Uygulama Geliştiricisi
2000-2001	Netaş Telekom Test Mühendisi
2001-2002	Datasel Bilgi Sistemleri A.Ş. Veritabanı ve Yazılım Uzmanı