# ORACLE İÇİN VERİ TABANI YÖNETİM ARACI
# VE
# PERFORMANS ANALİZİ

**ALPASLAN KILIÇKAYA**

**YÜKSEK LİSANS TEZİ**
**BİLGİSAYAR MÜHENDİSLİĞİ**

**TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**HAZİRAN 2008**
**ANKARA**

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Yücel ERCAN
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Ali YAZICI
Anabilim Dalı Başkanı

Alpaslan KILIÇKAYA tarafından hazırlanan ORACLE İÇİN VERİ TABANI YÖNETİM ARACI VE PERFORMANS ANALİZİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Prof. Dr. Ali YAZICI
Tez Danışmanı

Tez Jüri Üyeleri

Başkan :Doç. Dr. Elif Derya ÜBEYLİ _____

Üye : Prof. Dr. Ali YAZICI _____

Üye : Yrd. Doç. Dr. Osman ABUL _____

## TEZ BİLDİRİMİ

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

Alpaslan KILIÇKAYA

**Üniversitesi** : **TOBB Ekonomi ve Teknoloji Üniversitesi**
**Enstitüsü** : **Fen Bilimleri**
**Anabilim Dalı** : **Bilgisayar Mühendisliği**
**Tez Danışmanı** : **Prof. Dr. Ali YAZICI**
**Tez Türü ve Tarihi** : **Yüksek Lisans – Haziran 2008**

**Alpaslan KILIÇKAYA**

# ORACLE İÇİN VERİ TABANI YÖNETİM ARACI VE PERFORMANS ANALİZİ

## ÖZET

Bu tez çalışmasında, Oracle performans sorunları ve etkin çözümlerin nasıl olabileceğine değinilmiştir.

Bu tez kapsamında geliştirilen DBAExplorer programı Oracle VTYS için hazırlanmış bir veri tabanı yönetim aracıdır. Yazılan program, veri tabanına hızlı erişim sağlayan Borland Delphi 7.0 programlama dili ile geliştirilmiştir. Programın hazırlanmasında veri tabanı erişim yöntemleri incelenmiş, en hızlı ve etkin bağlanma şekilleri araştırılmıştır. Yapılan araştırmalar sonucunda en etkin performans yöntemleri ve SQL cümleleri bulunmuş ve hazırlanmıştır.

**Anahtar Kelimeler**: Oracle, Veri Tabanı, VTYS, Performans Analiz, Delphi 7.0, PL/SQL

**University** : **TOBB Economics and Technology University**
**Institute** : **Institute of Natural and Applied Sciences**
**Science Programme** : **Computer Engineering**
**Supervisor** : **Prof. Dr. Ali YAZICI**
**Degree Awarded and Date** : **M.Sc. – June 2008**

**Alpaslan KILIÇKAYA**

**DATABASE MANAGEMENT TOOL FOR ORACLE AND PERFORMANS ANALYZE**

## ABSTRACT

In this thesis, a database management tool for oracle RDMS was developed to provide effective solutions to some of the performance issues.

DBAExplorer Management Tool is designed for the Oracle RDMS. The tool is developed using Borland Delphi 7.0 programming environment which provides fast access to the Oracle Database. During the development of the program fastest and the most effective ways of database access techniques were examined and used. As a consequence of this study, the most effective techniques and SQL statements are determined for better performance and put into action.

**Keywords**: Oracle, Database, RDBMS, Performance Analysis, Delphi 7.0, PL/SQL

## TEŞEKKÜR

Geliştirmiş olduğum bu tez boyunca beni yönlendiren, değerli katkılarını benden hiç eksik etmeyen ve kendime örnek aldığım değerli hocam Sayın Prof. Dr. Ali YAZICI' ya teşekkürlerimi bir borç bilirim.

Hayatımda ve tez çalışmam sırasında, bana gerekli sabrı gösteren ve her zaman yanımda bulunan sevgili eşim Banu ve oğlum Arda'ya, beni yetiştirmek için büyük emek sarf eden aileme de teşekkür ederim.

# İÇİNDEKİLER

# ÇİZELGELERİN LİSTESİ

**Çizelgeler** **Sayfa**

# ŞEKİLLERİN LİSTESİ

**Şekiller** **Sayfa**

## KISALTMALAR

| Kısaltmalar | Açıklamalar |
|---|---|
| **VTYA** | Veri Tabanı Yönetim Aracı |
| **VTYS** | Veri Tabanı Yönetim Sistemi |
| **DDL** | Veri ve Veri Modeli Tanımlama Dili (Data Definition Language) |
| **DML** | Veri İşleme Dili (Data Manipulation Language) |
| **SQL** | Yapısal Sorgulama Dili (Structured Query Language) |
| **RDBMS** | İlişkisel Veri Tabanı Yönetim Sistemi (Relation Database Management Systems) |
| **ODAC** | Direkt Oracle Bağlantısı (Direct Oracle Access Component) |
| **BDE** | Borland Veri Tabanı Motoru (Borland Database Engine) |
| **VT** | Veri Tabanı |
| **VTY** | Veri Tabanı Yöneticisi |
| **PL/SQL** | Oracle firması tarafından geliştirilmiş SQL komutları |
| **DBA** | Veri Tabanı Yöneticisi (Database Administrator) |
| **MDI Form** | Çoklu Kullanıcı Arayüzü Dokümanı (Multiple Documents Interface) |
| **API** | Uygulama Geliştirme Arayüzü (Application Programming Interface) |
| **OLAP** | Çevrimiçi Analitik İşleme (Online Analytical Processing) |
| **OLTP** | Çevrimiçi Hareket İşleme (Online Transaction Processing) |
| **DBAExplorer** | Geliştirilen Veri Tabanı Yönetim Aracı |
| **SCRIPT** | SQL Komut Dizisi |

# BÖLÜM 1

## 1. GİRİŞ

Veri Tabanı Yönetim Sistemleri (VTYS) çok büyük veri tutan ve bu verileri yönetmek isteyen herkese hizmet etmek üzere geliştirilmiş programlardır. Başlıca kullanılan VTYS'ler Oracle, DB2, MS SQL Server, Sybase, Informix, MySQL, FireBird Postrage ve Access gibi sıralanabilir.

Günümüzde yaygın olarak kullanılan İlişkisel Veri Tabanı (Relational Database) yaklaşımı, verileri normalizasyon kuralları çerçevesinde tablolara ayırmayı, bu tablolar arasında bir birincil anahtar ve bir yabancı anahtar üstünden ilişki kurmayı öngörmektedir.

Her geçen gün bilginin çoğalmasıyla veri tabanlarının büyümesi, profesyonel yönetimi ve daha hızlı erişimi gerektirmektedir. Bu tez çalışması kapsamında bu konu ile ilgili araçlar ve çalışmalar incelenmiş, performans ve yönetim önerileri yapılmış, bu öneriler açık kaynak bir uygulamaya dönüştürülerek gerçekleştirilmiş ve alınan sonuçlar değerlendirilmiştir.

### 1.1. Çalışmanın Amacı

Bu tez kapsamında Oracle VTYS ve mimarisi, performans yönetimi ve çözümleri, analiz yapmaya yarayan SQL cümleleri, nesnesel programlama teknikleri, kullanılan araçlar ve özellikleri hakkında araştırmalar yapılmıştır. Yapılan araştırmalar sonucunda bulunan SQL çözümlerin ve kullanılan araçların belirli sorunları çözmeye yönelik olduğu genel kapsamı içermediği anlaşılmıştır. Buradan yola çıkarak performans ve istatistiksel bilgi almaya yaran SQL cümleleri toparlanmış, alınan sonuçlar incelenerek performans başarımını artırmaya yönelik bazı gelişmeler yapılmıştır. Bu kapsamda, öncelikle Oracle Veri Tabanı (VT) incelenmiş, çeşitli zamanlarda çeşitli yüklere tabi tutulmuş, değişik sistem parametreleri denenmiş,

oluşan çıktılar gözlenmiş, gerek duyulan yerlerde müdahale yapılarak nasıl tepki verdiği izlenmiştir. Diğer yandan Oracle VT kurulumu, PL/SQL cümle yapısı, VT bağlantı şekilleri, nesnesel programlama geliştirme teknikleri, programlama standartları ile ilgili birçok kaynak taranarak bilgi edinilmiştir. Oracle VTYS hakkında birçok kaynaktan bilgi elde edilerek VT yönetimi hakkında ileri düzey bilgi sahibi olunmuştur. Oracle VT'na ilişkin kurulumdan önce ve kurulumdan sonra kullanılan parametreler araştırılarak bilgi sahibi olunmuş, kurulumda kullanılması gereken optimum değerler araştırılarak öğrenilmiştir. Oracle Veri Tabanı Yönetim Araçları (VTYA) incelenmiş ve karşılaştırılmış, ortak özellikleri çıkartılmıştır. Edinilen sonuçlar karşılaştırıldığında bu araçların VT yönetimi ve performans incelemesini aynı anda desteklemediği ve yeterli olmadığı görülmüştür. Elde edilen birikimleri ve oluşturulan tasarımı gerçekleştirmek amacıyla açık kaynak bir araç geliştirilmiştir. Bahsedilen araştırma, tasarım, gerçekleme ve test aşamaları ile oluşturulan VTYA bu tez kapsamındadır.

Bu tez kapsamında yazılan VTYA, Oracle VT'nı yönetmek, son kullanıcılar için geliştirme ortamı sağlamak ve performansa yönelik iyileştirmeler yapmak amacıyla tasarlanmıştır. Bu tez kapsamında mevcut araçlardan farklı bir araç geliştirmenin nedenleri aşağıda maddeler halinde sunulmuştur:

- Nesnesel ön yüzü ile VT yönetimini sağlaması
- Kolay kullanımı ve kullanıcı dostu ara yüzü aracılığıyla VT uygulamalarının geliştirilmesini sağlaması
- Kurulum gerektirmeden Windows tabanlı işletim sistemlerinde hızlı bağlantı sağlaması
- Geliştirilebilir açık kaynak olması
- Performansa yönelik çözümler sunması

Sonuç olarak yaygın olarak kullanılan fakat pahalı olan VTYA alternatif olacak, endüstri ve akademik ortama katkıda bulunacak açık kaynak bir araç geliştirmek hedeflenmiştir.

İkinci bölümde; Oracle VTYS ve genel yapısına ilişkin bilgiler aktarılmıştır. Üçüncü bölümde; performans aşamaları, performans araçları ve kullanım şekilleri, son olarak da yapılan performans incelemeleri ve sonuçlarına değinilmiştir. Dördüncü bölümde; en popüler VTYA'nın özelliklerine ve karşılaştırılmasına yer verilmiştir. Bu bölümü takiben yeni bir VTYA neden ihtiyaç duyulduğu, kullanılan teknik alt yapı, kullanıcı ekran görüntüleri ve tasarımın nasıl gerçekleştiği anlatılmıştır. Altıncı bölümde geleceğe yönelik neler yapılabileceği vurgulanmış, son bölümde ise genel sonuçlar açıklanmış ve değerlendirme yapılmıştır.

# BÖLÜM 2

## 2. ORACLE VERİ TABANI YÖNETİM SİSTEMİNE BİR BAKIŞ

VT düzenli bilgiler topluluğudur. "*Veri Tabanı*" kelimesinin anlamı bilgisayar ortamında saklanan düzenli verilerle sınırlı olmamakla birlikte, daha çok bu anlamda kullanılmaktadır. Bilgisayar terminolojisinde ise, sistematik erişim imkânı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesi olarak tanımlanmıştır. Bir başka tanımı da, bir bilgisayarda sistematik şekilde saklanmış, programlarca işlenebilecek veri yığınıdır[6].

Bir VT'nı oluşturmak, saklamak, çoğaltmak, güncellemek ve yönetmek için kullanılan programlara VTYS adı verilir. VTYS özelliklerinin ve yapısının nasıl olması gerektiğini inceleyen alan Bilgi Bilimi'dir [6].

VT'da asıl önemli kavram, kayıt yığını ya da bilgi parçalarının tanımlanmasıdır. Bu tanıma şema adı verilir. Şema VT kullanılacak bilgi tanımlarının nasıl modelleneceğini gösterir. Buna Veri Modeli, yapılan işleme de Veri Modelleme denir. En yaygın olarak kullanılan, ilişkisel modeldir. Andrew Layman'ın (Microsoft) deyimiyle bu modelde veriler tablolarda saklanır. Tablolarda bulunan satırlar kayıtların kendisini, kolonlar ise bu kayıtları oluşturan bilgi parçalarının ne türden olduklarını belirtir.

VT Yazılımı ise verileri sistematik bir biçimde depolayan yazılımlara verilen isimdir. Birçok yazılım bilgi depolayabilir ama aradaki fark, VT'nın bu bilgiyi verimli ve hızlı bir şekilde yönetip değiştirebilmesidir. VT bilgi sisteminin kalbidir ve etkili kullanmakla değer kazanır. Bilgiye gerekli olduğu zaman ulaşabilmek esastır. İçeriği olmayan bir kütüphane ve bütün kitapların aynı kapağa sahip olduğunu düşündüğünüzde kütüphane kullanıcılarının ne kadar çok işi olacağını tahmin edebilirsiniz. Bir VT bir kütüphanenin mükemmel bir içerik sistemi olduğu gibi, aynı zamanda kütüphanenin kendisidir. İlişkisel Veri Tabanı Yönetim Sistemleri (İVTYS

- RDBMS) büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkânlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkânının sağlandığı programlardır.

**VT Yöneticisi**

Günümüzde VT sistemleri, bankacılıktan otomotiv sanayine, sağlık bilgi sistemlerinden şirket yönetimine, telekomünikasyon sistemlerinden hava taşımacılığına, çok geniş alanlarda kullanılan bilgisayar sistemlerinin alt yapısını oluşturmaktadır. VT fiziksel olarak bilgileri tutarken mantıksal bir sisteme de sahiptir. VT sistemlerinin kurulumu, yapılandırması, tasarımı, sorgulaması, güvenliği ve denetiminin karmaşık bir hal alması VT Yöneticiliği kavramının oluşmasına neden olmuştur. Bir VT yöneticisi; mantıksal veri modelleme, fiziksel VT tasarımı çıkararak VT oluşturma, sorgu yazma, kurulum ve ayarları yapma, güvenliği sağlama, VT'nın yönetimi ve bakımı sağlama, VT'nı denetleme işlerini üstlenir.

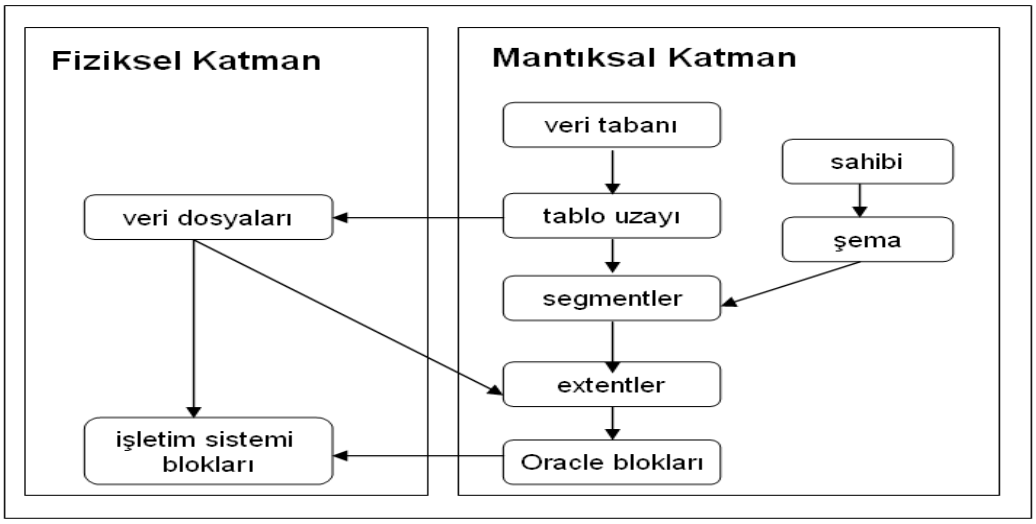
İVTYS büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkânlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkânının sağlandığı programlardır. Oracle VT da bir ilişkisel veri tabanı yönetim sistemidir. Oracle VTYS'nin özellikleri şunlardır [42]:

- Büyük miktarda veri tutabilmekte ve verilerin depolandığı alanları ayarlama imkânı vermektedir.
- Aynı anda çok sayıda kullanıcıya verilerin bütünlüğünü bozmadan hizmet verebilmektedir.
- Oracle VT 8 sürümü ile birlikte on binlerce kullanıcıya hizmet verebilmektedir.
- Hiç kapatılmadan uzun süre çalışabilmektedir.
- İşletim sistemi, veri erişim dilleri ve ağ iletişim protokolleri standartlarıyla uyumludur.

- Yetkisiz erişimleri engelleme ve kontrol edebilme imkânı sağlamaktadır.
- Bütünlüğü VT düzeyinde sağlayabilmektedir, böylece daha az kod yazılmaktadır.
- İstemci/Sunucu mimarisinin bütün avantajlarını kullanabilmektedir.

## 2.1. Oracle VT Mimarisi

Oracle VT fiziksel ve mantıksal olmak üzere iki katmandan oluşur [33]. Fiziksel ve mantıksal katman birbirinden ayrı olduğu için verinin fiziksel olarak saklanma şekli mantıksal yapıya erişimi etkilemez. Fiziksel katman, işletim sisteminden görünen kısımdır. Bunlar veri dosyası, kontrol dosyası ve log dosyasından oluşmaktadır. Mantıksal katman bir ya da daha fazla tablo uzayı, tablolar, görüntüler, sıralar, eşanlamlar, indisler, kümeler, VT bağlantıları, prosedürler, fonksiyonlar ve paketlerden oluşan şema nesnelerinden oluşmaktadır. Fiziksel katmana işletim sistemi tarafından erişebilmesine rağmen, mantıksal katmana ancak Oracle'a bağlanıp, SQL komutları çalıştırılarak erişilebilir. Fiziksel ve mantıksal katman arasındaki ilişki Şekil 2.1'de gösterilmiştir. Oracle VT kurulu herhangi bir makinede, SQL bilgisi olmayan bir kullanıcı, Oracle'ın sadece fiziksel katmanını görebilmektedir. Oracle VT'nın daki her nesnenin bir sahibi (kullanıcı olarak bahsedilir) vardır. Her kullanıcı bir veya daha fazla tablo uzayına sahip olabilir. Her nesne, ait olduğu kullanıcının herhangi bir tablo uzayında (mantıksal olarak) bulunur. Her tablo uzayı da, kendisine sahip olan kullanıcının nesnelerini tutmak için işletim sisteminde bir veya daha fazla veri dosyasına sahip olabilmektedir. Sonuç itibariyle, VT'nındaki her nesnenin bir kullanıcısı vardır ve bu nesneler mantıksal olarak o kullanıcının sahip olduğu tablo uzaylarının herhangi birinin (hangisi olduğu komutlarla öğrenilebilir) içerisinde, fiziksel olarak da o kullanıcının sahip olduğu tablo uzayının herhangi bir veri dosyasında bulunur. Fakat o veri dosyasının içerisine işletim sisteminden bu nesneyi bulmak için bakılamaz. Bu nesnenin sahibi ve mantıksal yeri Veri İşleme Dili (DML) komutları ile bulunabilmektedir [33].




**Şekil 2.1.** Fiziksel ve mantıksal katman arasındaki ilişki.

## 2.2. Fiziksel Katman

Fiziksel katman VT'nı oluşturan işletim sistemi dosyaları bulunur. Bir Oracle VT fiziksel olarak bir ya da daha fazla veri dosyası, iki ya da daha fazla log dosyası, bir ya da daha fazla kontrol dosyasından oluşur. Oracle VT'nda kullanılan fiziksel yapılar aşağıdaki bölümde özetlenmiştir.

### 2.2.1. Veri Dosyaları

Oracle VT bir veya daha fazla veri dosyası içerebilir. Veri dosyaları VT'ndaki tüm verileri tutan dosyalardır. Tablo, indis gibi mantıksal VT yapılarının içerisindeki veriler fiziksel olarak veri dosyalarında tutulurlar. Bir veri dosyası kendisi için ayrılan alan dolduğunda, kendi sahip olduğu alanı artırabilecek özelliklere sahiptir. Bir ya da daha fazla veri dosyası mantıksal bir VT depolama birimi olan bir tablo uzayını oluştururlar. Normal VT işlemleri boyunca bir veri dosyası içerisindeki veriler okunur ve Oracle için ayrılan belleğe getirilirler. Örneğin bir kullanıcının VT'ndaki bir tablonun verilerine erişmek istediğini varsayalım. Eğer istenilen veriler bellekte yer almıyorsa, ancak o zaman uygun veri dosyasından okunur ve belleğe getirilirler. Değişikliğe uğrayan veriler ya da yeni eklenen veriler veri dosyalarına hemen yazılmazlar. Sabit diske erişimi azaltmak ve böylece sistemin performansını

arttırmak için veriler bellek havuzunda tutulur ve gerektiğinde hepsi birden uygun veri dosyalarına kaydedilirler.

### 2.2.2. Log Dosyaları

Log dosyaları olarak bilinen bu dosyaların amacı veriler üzerinde yapılan tüm değişiklikleri kaydetmektir. Bir redo log, redo kayıtlarından oluşur. Redo logun ana görevi veri üzerinde yapılan tüm değişikliklerin kaydını tutmaktır. Eğer veri dosyalarına kalıcı olarak kaydedilmiş olan, değişikliğe uğramış kayıtlarda bir bozukluk olursa yapılan değişiklikler redo log dosyalarından sağlanabilir ve işlemler kaybolmaz. Birden fazla tekrarlanan bozukluk durumlarında redo log dosyalarının da bozulmasını engellemek için Oracle farklı diskler üzerinde redo log dosyalarının birden fazla kopyasının alınmasına olanak sağlar. Bir VT işlemi sırasında elektrik kesilirse, bellekteki veriler veri dosyalarına kaydedilmeyecek ve verilerin kaybolması durumuyla karşılaşılacaktır. Oracle VT tekrar açıldığında redo log dosyalarında yapılan son değişiklikler veri dosyalarına yansıtılarak verilerin kaybolması engellenir.

### 2.2.3. Kontrol Dosyaları

Oracle VT bir kontrol dosyasına sahiptir. Kontrol dosyasında VT'nın fiziksel yapısı, adı, veri dosyaları, log dosyalarının adı ve diskteki yeri, VT'nın oluşturulma tarihi gibi VT ile ilgili bilgileri tutar. Her VT oturumu açıldığında Oracle bu dosyayı kontrol ederek gerekli bilgileri alır. Eğer VT'nda fiziksel bir değişme olursa (yeni bir log dosyası ya da veri dosyası oluşturulması gibi), yapılan değişiklikler Oracle tarafından otomatik olarak kontrol dosyalarına yansıtılır. Kontrol dosyası VT'nın kurtarılmasında da kullanılır.

## 2.3. Mantıksal Katman

Oracle VT'nın mantıksal yapısı tablo uzaylarını, şema nesnelerini, veri bloklarını, genişlemeleri ve parçaları içerir. Oracle VT'nın mantıksal yapısı aşağıdaki bölümlerde özetlenmiştir.

**Şema ve Şema Nesneleri**

Şema nesneleri mantıksal veri depolama yapıları olarak bilinir. Bir şemanın sahibi bir VT kullanıcısıdır ve bu şema o kullanıcıyla aynı isme sahiptir. VT üzerinde kullanıcının belirli işleri yapabilmesi için tanımlanan bu yapılar tablolar, görüntüler, sıralar, eşanlamlar, indisler, kümeler, VT bağlantıları, prosedürler, fonksiyonlar, ve paketlerdir. Bir şema ise bu nesnelerin oluşturduğu gruptur. Buradaki önemli nokta şemalar ile tablo uzayları arasında bir bağlantı yoktur.

### 2.3.1. Tablo

Kullanıcılar tarafından ulaşılacak bütün veriler tablolar içerisinde saklanır. Her tablonun bir ismi vardır ve her biri bir "kayıt" olarak adlandırılan satırlar ile bu kayıtlardaki verilerin özelliklerini belirleyen kolonlardan oluşur. Her tablo bir ya da daha fazla kolondan oluşabilir. Kolon sayısı 256 ile sınırlıdır. Tablo kolonlarının bir adı ve veri tipi vardır. Bir tablo oluşturulduğunda Oracle verileri depolamak için bir tablo uzayı içerisinde bir veri segmenti ayırır.

### 2.3.2. Görüntü

Görüntü bir ya da birkaç tablodan, hazırlanan sorgular ile istenilen alanların alınmasıyla oluşturulan sanal bir tablodur. Bir görüntü üzerinde silme, güncelleme gibi işlemler yapılamaz. Görüntü oluşturulduğu tabloların sadece o anlık görüntüsüdür. Tablolar gibi görüntülerin de bazı kısıtlamalar olmakla birlikte verisi sorgulanabilir, değiştirilebilir, silinebilir ve yeni veri girilebilir.

### 2.3.3. Sıra

Sıra tablolardaki kayıtlara otomatik numara vermek için kullanılır. Sıra numarası VT tarafından otomatik olarak üretilir. Özellikle çok kullanıcılı ortamlarda tekil numara üretilmek için kullanılır. Sıra numaraları Oracle'da tanımlı 38 rakama kadar tamsayılardan oluşur. Bir sıra tanımlaması sıranın adını, artan ya da azalan olacağını, iki sayı arasındaki fark miktarını içerir. Oracle tüm sıra numarası tanımlarını SYSTEM tablo uzayının içerisindeki bir veri sözlüğü tablosuna kaydeder. Sıra numaraları tablolardan bağımsız olarak üretilir. Yani bir sıra numarası bir ya da daha çok tablo için kullanılabilir.

### 2.3.4. Eşanlam

Eşanlam bir tablo, görüntü, sıra, prosedür, fonksiyon ya da paket için "alias" olarak adlandırılan bir takma isimdir. Eşanlam bir takma isim olduğu için veri sözlüğü içerisindeki tanımının kapladığı yer haricinde, VT'nında yer kaplamaz. Eşanlamlar güvenlik ve daha rahat kod yazma amacıyla kullanılırlar. Bir eşanlam kullanıldığında ilgili nesnenin adı ve sahibi gizlenir ve SQL komutu içerisinde kullanımı kolaylaşır. Eşanlamlar genel ve özel olarak tanımlanabilirler. "Genel" olarak tanımlanan eşanlamlara tüm VT kullanıcıları erişebilir. "Özel" olarak tanımlanan eşanlamlara sadece ilgili nesnenin sahibi ve sahibi tarafından hak verilmiş bir başka kullanıcı erişebilir.

### 2.3.5. Prosedürler ve Fonksiyonlar

Prosedürler ve fonksiyonlar belirli işleri yapmak veya belirli bir problemi çözmek üzere bir araya getirilmiş PL/SQL ve SQL cümleleri kümeleridir. Bunlar derlenmiş bir şekilde VT'nında saklanır ve kullanıcılar ya da VT uygulamaları tarafından kullanılabilirler. Fonksiyonlar ve Prosedürler arasındaki tek fark, fonksiyonların çağırıldığında bir değeri geri dönmesidir. Birden fazla uygulama programı içerisinde

aynı işi yapan kodları sürekli yazmak yerine kayıtlı prosedür ve fonksiyonlar bir kere yazılır ve tüm uygulamalar tarafından kullanılır.

### 2.3.6. Paketler

Paketler birbiriyle ilişkili olan prosedürlerin, fonksiyonların, değişkenlerin ve diğer yapıların bir bütün haline getirildiği ve VT'nında saklandığı yapıdır. Bu global değişken tanımlanıp paket içindeki herhangi bir prosedürde çağrılabilme gibi ek fonksiyonaliteler sağlar. Ayrıca paketler bir bütün halinde bir kerede hata kontrolü yapılıp, derlenip ve sonra belleğe yüklendiği için performans artışı da sağlar.

### 2.3.7. VT Tetikleri

Tetikler bir tablo veya görüntüde değişiklik olduğunda, bir kullanıcı ya da VT sistemi aksiyonu gerçekleştiğinde, VT tarafından otomatik olarak çağırılan, PL/SQL, Java veya C prosedürleridir. VT tetikleri, VT yönetimi için çeşitli yollar sağlar. Örneğin, veri yaratmayı otomatize etmek, veri değişimini izlemek, karmaşık güvenlik yetkilendirilmesi sağlamak gibi amaçlar için kullanılabilir.

## 2.4. Veri Bütünlüğü

Veri bütünlük kısıtlamaları, bir VT'ndaki bilgiler ile ilgili iş kurallarını uygulatmak amacıyla tanımlanabilir. Bütünlük kısıtlaması, bir tablonun bir kolonu için bir kural tanımlamanın bildirimsel bir yöntemidir. Bir bütünlük kısıtlaması, Oracle VTYS tarafından VT'nın temel tablolarına geçersiz veri girişini engellemekte kullanılan bir mekanizmadır. Eğer bir ifade yürütümünün sonuçlarından herhangi biri bir bütünlük kısıtlamasını ihlal ederse, ifadeye ilişkin değişiklikler ortadan kaldırılır ve bir hata kodu döndürülür. Oracle VTYS tarafından desteklenen bütünlük kısıtlamaları aşağıda maddeler halinde açıklanmıştır.

- **CHECK**: Önceden belirtilmiş mantıksal bir ifadeyi sağlamayan değerin girilmesini engeller. Anahtarlar çeşitli veri bütünlüğü tanımlamalarının tanımlarında kullanılır. Bir anahtar bir veya daha fazla kolondan oluşabilir. Anahtarlar ilişkisel VT'larında farklı tabloların kolonları arasında ilişki kurmak için kullanılır.
- **NOT NULL**: Boş bırakılamaz alanlar için kullanılır ve alana mutlaka bir değer girilmelidir.
- **UNIQUE KEY**: Tekil anahtar, aynı kayıttan iki defa girilmesini önler. Bu kısıtlamaya sahip olan alan NOT NULL ile tanımlanmadıysa sadece NULL değeri alabilir.
- **PRIMARY KEY**: Bu kısıtlama bu alanın birincil anahtar olmasını sağlar. Çift kayıt girilemez ve boş bırakılamaz. Tablo için birincil anahtar kısıtlaması tanımlanırken kullanılan öznitelik veya öznitelik kümesidir. Birincil anahtarın değeri tablonun her bir satırı tek başına belirtmek için kullanılır. Her tablo için bir birincil anahtar tanımlanabilir
- **FOREIGN KEY**: Yabancı anahtar, bir tablonun bir kolondaki tüm değerlerin bağlı olduğu diğer bir ilişkili tablonun bir kolonunda bulunan anahtarlarla ilişki kurulmasıdır. Diğer tablodaki tekil veya birincil anahtara referans eder.

## 2.5. Veri Blokları, Uzantılar ve Segmentler

Veri blokları, uzantılar ve segmentlerden oluşan mantıksal saklama yapıları sayesinde Oracle VT'nın disk alanı üzerinde ayrıntılı bir kontrolü vardır.

- **Oracle Veri Blokları**: Veri blokları, Oracle VT'nda verinin saklandığı en küçük yapıdır. Bir veri bloğu, fiziksel VT alanında belirli sayıdaki bayt'a karşılık gelir. Standart blok büyüklüğü "*DB_BLOCK_SIZE*" başlangıç parametresiyle belirlenir. Bunun dışında en çok 5 adet blok büyüklüğü belirtilebilir.

- **Uzantılar**: Mantıksal katmanındaki bir sonraki seviye uzantılardır. Bir uzantı, belirli sayıdaki ardışık VT bloğundan oluşur, bir seferde alınır ve belirli bir tipteki bilgiyi tutmak için kullanılır.
- **Segmentler**: Uzantıların üzerindeki mantıksal depolama seviyesi segmentlerdir. Segment, belirli bir mantıksal yapı için tahsis edilmiş uzantılar kümesidir. Bir segmentin içindeki tüm uzantılar dolduğunda, Oracle dinamik olarak yeni yer tahsis eder. Başka bir deyişle, bir segmente ait bütün uzantılar dolduğunda, Oracle bu segment için yeni bir uzantı tahsis eder. Uzantılar gerek duyulduğunda tahsis edildiğinden, segmente ait uzantıların ardışık olma zorunlulukları yoktur.

  Dört çeşit segment vardır:

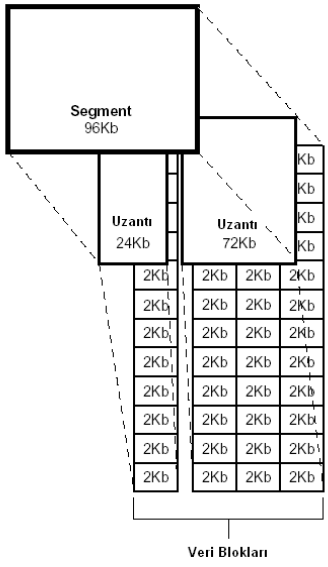
  - **Veri Segmenti**: Her kümelenmemiş tablonun bir segmenti vardır. Tablonun bütün verisi, veri segmentinin uzantılarında tutulur. Bölümlenmiş tablolarda, her parça için bir veri segmenti bulunur. Her küme için bir veri segmenti vardır. Kümedeki her tablonun verisi kümeye ait olan veri segmentinde tutulur.
  - **İndis Segmenti**: Her indisin kendi verisinin tutulduğu bir indis segment vardır. Bölümlenmiş bir indisin her bölümüne ait bir indis segment vardır.
  - **Geçici Segment**: Geçici segmentler bir SQL cümlesinin işini tamamlamak için geçici bir alana ihtiyaç duyulduğunda Oracle tarafından yaratılır. SQL cümlesinin çalıştırılması tamamlandığında geçici segment yeniden kullanılabilmesi için sisteme geri verilir.
  - **Geri Alma Segmenti**: Otomatik Geri Alma (*Automatic Undo Management*) modundayken, VT sunucusu tablo uzayları kullanarak geri alma alanını yönetir. Elle Geri Alma (*Manual Undo Management*) modundayken, VT yöneticisi tarafından geri alma bilgisini geçici olarak tutması için geri alma segmentleri yaratılır. Geri alma segmentlerdeki bilgi VT'nın kurtarılması sırasında kullanılır.

### 2.5.1. Veri Blokları

Oracle bir VT'nın veri dosyaları içindeki depo alanını veri blokları adı verilen birimlerle yönetir. VT tarafından kullanılan en küçük veri birimine veri bloğu denir. Çoğu Oracle VT sürümlerinde varsayılan veri blok uzunluğu 2 Kb, 4 Kb veya 8 Kb'dır. Standart blok uzunluğu başlangıç parametresi "*DB_BLOCK_SIZE*" ile belirlenir.

Buna karşılık, işletim sistemi seviyesinde bütün veriler bayt olarak depolanır. Her bir işletim sistemi bir blok büyüklüğüne sahiptir. Oracle, çoklu Oracle veri blokları içindeki veriden istekte bulunur, işletim sistemi bloklarından istekte bulunmaz. Oracle'ın kullandığı veya ayırdığı en küçük depo birimi Oracle veri bloğudur.

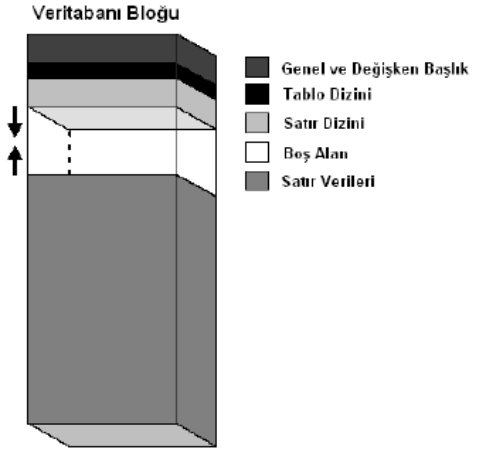
Oracle bir VT'ndaki bütün veriler için mantıksal VT alanı ayırır. Veri blokları, uzantılar ve segmentler VT alanı ayırma birimleridir. Şekil 2.2'de veri yapıları arasındaki ilişki gösterilmektedir [33].




**Şekil 2.2.** Segmentler, uzantılar ve veri blokları arasındaki ilişki.

**Veri Blok Formatı**

Veri Blokları Genel ve Değişken Başlık (*Common and Variable Header*), Tablo Dizini (*Table Dictionary*), Satır Dizini (*Row Dictionary*), Satır Verileri (*Row Data*) ve Boş Alan (*Free Space*) dan oluşur. Bir Veri Bloğunda sadece bir nesnenin verisi bulunabilir, aynı tablo uzayına ait olsa da birden fazla nesnenin verisi aynı veri bloğunda olamaz. Elbette bir nesnenin verisi birden fazla veri bloğuna dağılmış olabilir. Bu durumda Veri Blok Zincirleri oluşturulur. Şekil 2.3'de veri bloklarının biçimi gösterilmiştir [33].




**Şekil 2.3.** Veri blok formatı.

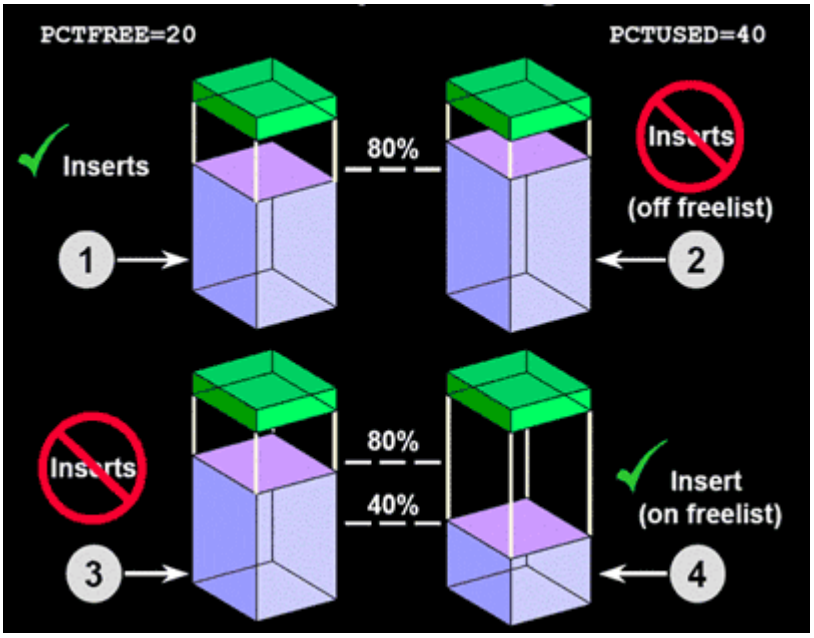
**PCTFREE ve PCTUSED**

Veri Bloklarını yönetmek için *PCTFREE* ve *PCTUSED* kullanılır. Bu değerler, tablo alanlarını el ile yönetmek için kullanılan iki alan yönetim parametresidir. Bunlar, belirli segmentteki bütün veri bloklarında bulunan satırlara eklemek ve güncellemek için boş alanın kullanımını denetleyebilmemizi sağlar.

*PCTFREE*, tablo üzerinde yapılan kayıt güncelleme işlemleri için, kayıt bloklarındaki ayrılacak rezervasyon yüzde değeridir ve 1–99 arasında bir tamsayıdır. 0 değeri girildiğinde rezervasyon işlemi yapılmaz. Varsayılan değer 10'dur. Yani güncelleme için her bloğun %10'unu rezerve eder. Geri kalan %90'nını ise yeni kayıt girişlerine ayırır.

*PCTUSED*, tablonun her veri bloğu için Oracle'ın koruduğu, kullanılmış alanın minimum yüzdesini verir ve 1-99 arasında bir tamsayıdır. Varsayılan değeri 40'dır. Verilen parametre değerinin altına düşen blok için, yeni kayıt girişi seçimliktir. *PCTFREE+PCTUSED <100* olmak zorundadır.

Bir Veri Blok'ta boşluk olması onun boş listede (*Free List*) olması anlamına gelmez. Aynı şekilde bir miktar veri olması da kullanılan listede (*Used List*) olması anlamına gelmez. Bir bloktan bir miktar veri silindiğinde veri miktarı belirtilen yüzdenin (ki bu *PCTUSED* parametresidir) altına düşünce eğer blok Used List'te ise Free List'e alınır. Aynı şekilde, veri eklendiğinde veri miktarı aynı yüzdenin üstüne çıkarsa ve blok Free List'te ise Used List'e alınır. *PCTFree* ile *PCTUsed* arasındaki ilişki şekil 2.4'de gösterilmiştir [33].




**Şekil 2.4.** PCTFree ve PCTUsed gösterimi.

Veri bloklarının büyük veya küçük olmasının bazı avantajları ve dezavantajları vardır. Bu avantaj ve dezavantajlar şunlardır:

- Küçük Blok Avantajı
    - Küçük bloklar, blok tartışmasını azaltır çünkü her blokta daha az satır vardır.

- Küçük bloklar, küçük satırlar için daha iyidir.
- Küçük bloklar rasgele okumalar için iyidir. VT gizli yerinin boyutu sınırlandırılabileceğinden, küçük blok boyutu, tampon gizli yerinin daha çok etkin kullanımını sağlar. Bu durum bellek kaynakları sınırlandırıldığında önemlidir.

- Küçük Blok Dezavantajı
  - Küçük bloklar göreceli olarak tepe üst düzeyine sahip olurlar.
  - Daha fazla blok okunmasına neden olurlar.
- Büyük Blok Avantajları
  - Büyük bloklar büyük satırlar için daha iyidir.
  - Büyük bloklar sıralı okumalar için daha iyidir.
  - Büyük bloklar, indis okumalar için performans sağlar. Büyük bloklar her bir blok içerisinde daha çok dizin girişi tutabilirler.

### 2.5.2. Uzantılar

Depolanan alanların ayrılması ile belli sayıda bitişik veri bloklarından meydana gelmiş VT mantıksal birimine uzantı denir. Segment bir veya birkaç uzantının bir araya gelmesinden meydana gelir. Segment içinde var olan alan tamamen kullanıldığında, Oracle bu segment için yeni uzantı ayırır.

**Uzantıların Hesaplanması**

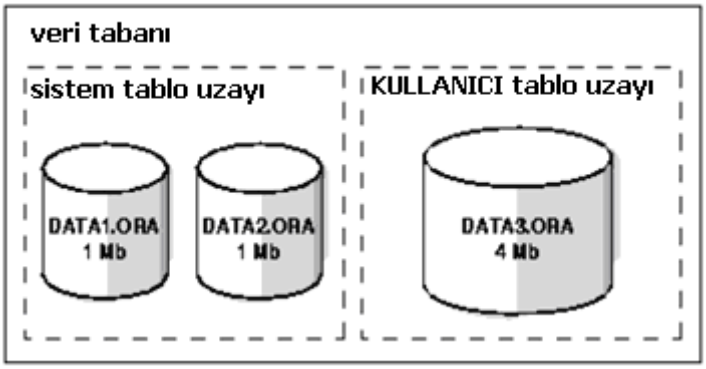
Uzantıların hesaplanmasında aşağıdaki maddeler göz önünde tutulur:

- Maksimum uzantı sayısı Veri Blok Büyüklüğüne bağlıdır.
- 2K blok için, maksimum uzantı = 121'dır.
- Bir uzantı dolduğunda, yeni uzantı ayrılır.
- Yeni uzantı büyüklüğü = *(Geçerli Uzantı Büyüklüğü)* * *(1 + PCTINCREASE/100)*'dür.
- Eğer *PCTINCREASE* büyüklüğü 0'a eşit ise uzantı aynı büyüklükte büyür.

## 2.6. Tablo Uzayı

Her VT, ilişkili mantıksal yapıların gruplanmasını sağlayan ve tablo uzayı olarak bilinen mantıksal depolama birimlerine bölünmüştür. Örneğin, tablo uzayı genelde yönetimsel işleri kolaylaştırmak için bütün uygulama nesnelerini birlikte gruplar. Tablo uzayı içindeki tüm mantıksal yapıları fiziksel olarak tutması için bir veya birden fazla veri dosyası yaratılır. Veri dosyalarının toplam büyüklüğü, tablo uzayının toplam depolama kapasitesini verir. Tablo uzaylarının toplam depolama kapasitesi, VT'nın toplam depolama kapasitesini verir. Şekil.2.5'de Veri tabanları, tablo uzayları ve veri dosyaları arasındaki ilişki gösterilmektedir.

Bir tablo uzayı çevirim içi veya çevirim dışı olabilir. Tablo uzayları, kullanıcıların içindeki veriye ulaşabilmesi için, genelde erişilebilir durumdadır. Fakat bazı durumlarda VT'nın bir kısmını erişilemez hale getirmek için tablo uzaylarının bir kısmı erişilemez hale getirilir. Bu birçok yönetimsel işlerin yapılmasını kolaylaştırır [33].




**Şekil 2.5.** Veri Tabanları, Tablo Uzayları ve Veri Dosyaları

Yukarıdaki şekil VT, tablo uzayı ve veri dosyaları arasındaki ilişkiyi açıklamaktadır. Buna göre:

- Bir VT bir ya da daha fazla tablo uzayına bölünmüştür.

- Tablo uzayı içerisindeki tüm mantıksal yapıları fiziksel olarak depolayabilmek için, her tablo uzayı bir ya da daha fazla veri dosyasına sahip olabilir.
- Tablo uzaylarının toplam kapasitesi, sahip oldukları veri dosyalarının toplam kapasitesine eşittir (Yukarıdaki şekil için SYSTEM tablo uzayı 2MB, USERS tablo uzayı 4MB).
- Tablo uzaylarının toplam kapasitesi VT'nın toplam kapasitesini belirler (6 MB).
- Bir tablo uzayı açık ya da kapalı olabilir. Tablo uzayı kapalı olduğunda bu tablo uzayının içerisindeki nesnelere erişilemez. Bir tablo uzayı yönetim amaçlı olarak kapalı duruma alınabilir.

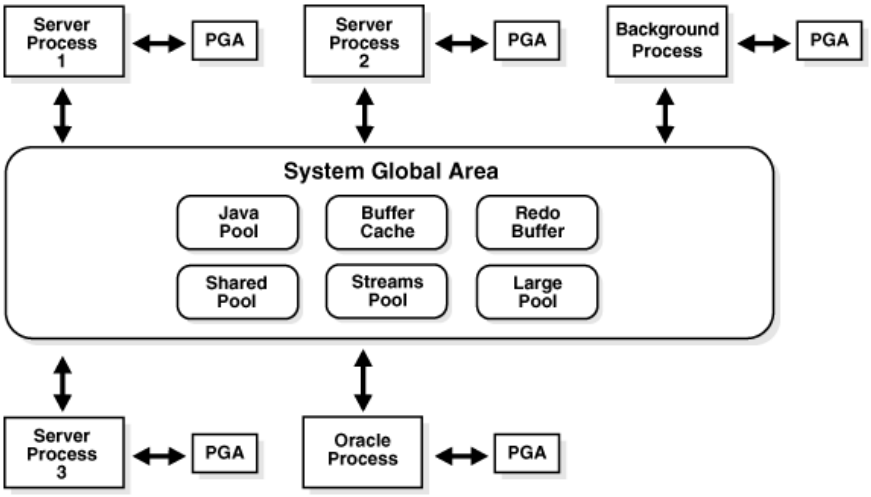
## 2.7. Bellek Yapısı

Oracle'da kullanılan tüm bellek yapıları, VT'nın oluşturulduğu bilgisayarın ana belleğinde yer almaktadır. Bu bölüm birden fazla kullanıcının aynı anda VT'na erişip işlemlerini gerçekleştirmesinin nasıl olduğunu anlamak açısından önemlidir. Bu bölümde anlatılacak olan mimari özellikler, Oracle sunucusunun şunları desteklemesini sağlamaktadır:

- Çok sayıda kullanıcının aynı anda tek bir VT'na ulaşması,
- Aynı anda çok sayıda kullanıcının ve uygulamanın bağlandığı bir VT'nın ihtiyaç duyduğu yüksek performansın sağlanması.

Aşağıda Oracle VT'nın temel bellek ve işlem yapısı anlatılmaktadır.

Oracle'ın bellek yapısı 3 bölümden oluşur: Sistem Global Alanı (SGA), Program Global Alanı (PGA) ve Kullanıcı Global Alanı (UGA). Bunların içinde de çeşitli bölümler vardır. Şekil 2.6'daki grafikte genel bellek yapısı gösterilmiştir [33].




**Şekil 2.6.** Genel bellek yapısı.

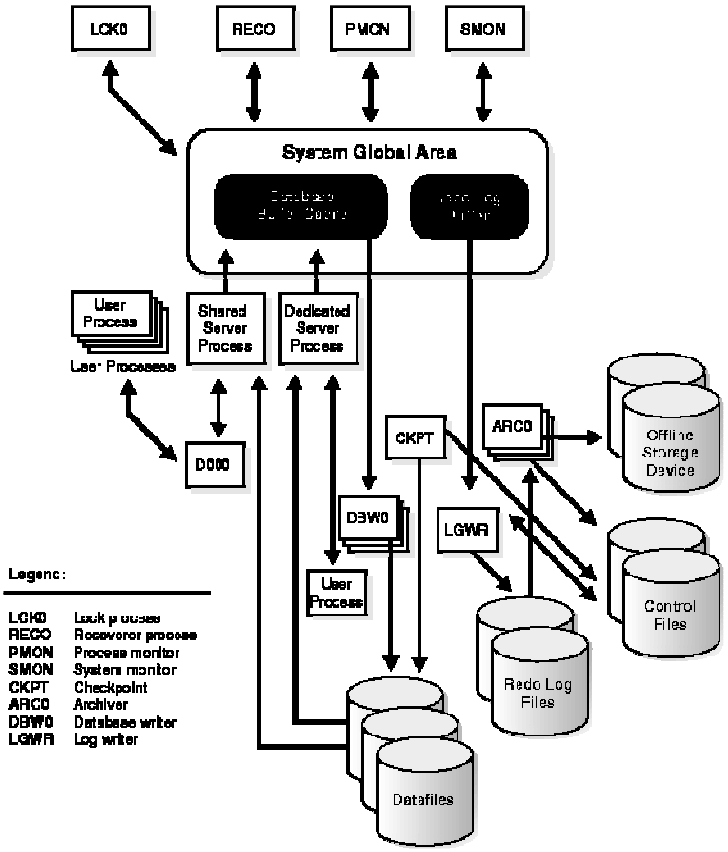
Oracle çeşitli işlemlerini yapabilmek için bellek yapıları oluşturur ve bunları kullanır. Örnek olarak, bellek çalışan program kodlarını ve kullanıcılar arasında paylaşılan verileri depolar. Oracle VT'da aşağıdaki gibi birkaç adet bellek yapısı mevcuttur:

- Sistem Global Alanı (*System Global Area*),
- VT Tamponları (*Database Buffers*),
- Redo Log Tamponları (*Redo Log Buffers*),
- Paylaşım Havuzu (*Shared Pool*).

### 2.7.1. Sistem Global Alanı (SGA - System Global Area)

SGA bir Oracle VT oturumu için gerekli verileri ve kontrol bilgilerini içeren paylaşımlı bellek bölgesidir. SGA ve Oracle arka plan işlemleri bir Oracle VT oturumunu oluşturur. VT oturumu başladığında Oracle SGA'yı oluşturur ve VT kapatıldığında yok eder. Her bir VT oturumunun kendine ait bir SGA'sı vardır. Oracle'a bağlanan kullanıcılar SGA içerisinde verileri paylaşımlı olarak kullanırlar. Yüksek performansın sağlanabilmesi için SGA'nın olabildiğince büyük olması gerekir. SGA büyük olursa bu alanda daha fazla bilgi depolanabilir ve sabit diske erişim sayısı azalır. SGA içerisinde depolanan bilgi VT tamponları, redo log tamponları ve paylaşım havuzunu içeren birkaç tip bellek yapısına bölünmüştür. Bu

alanlar sabit büyüklüktedir ve VT açılırken oluşturulurlar. Şekil 2.7 Oracle sunucusunun temel bellek ve işlem yapılarını göstermektedir [33].




**Şekil 2.7.** Oracle'ın bellek yapıları ve işlemleri

- **VT Tampon Belleği (Database Buffer Cache)**

  En son kullanılan veri blokları SGA içerisinde VT tamponu denilen yerde depolanır. Bu VT tamponlarının hepsi VT tampon belleğini oluşturur. VT tampon belleği değiştirilmiş ve değiştirilmemiş bilgileri içerir. Son kullanılan verilerin ya da çok kullanılan verilerin bellekte depolanması sayesinde sabit disk erişim işlemleri azalır.

- **Redo Log Tamponu (Redo Log Buffer)**

  SGA'nın redo log tamponu VT verileri üzerinde yapılan son değişiklikleri depolar. Redo log tamponunda depolanan değişiklik bilgileri VT kurtarma işlemlerinde gerekli olan redo log dosyalarına kaydedilirler.

- **Paylaşım Havuzu (Shared Pool)**

  Paylaşım havuzu SGA'nın paylaşımlı SQL alanları gibi bellek yapılarını içeren kısmıdır. Paylaşımlı SQL alanı VT'na girilen her farklı SQL komutunu işlemek için gereklidir. Her bir paylaşımlı SQL alanı aynı komutu işleyen birden fazla uygulama tarafından kullanılır. Burada amaç diğer kullanıcılar için daha fazla paylaşımlı bellek alanı bırakabilmektir

- **Geniş Havuz (Large Pool)**

  Geniş havuz SGA içerisinde isteğe bağlı bir alandır. Bu alan yedekleme, yapılan işlemleri geri yükleme, sunucunun giriş/çıkış işlemleri gibi işlemlerde daha geniş bellek ihtiyacı için kullanılan alandır.

- **Cümle Tanıtıcıları veya İmleçler**

  İmleç belirli bir cümle için tahsis edilmiş olan bellek için yaratılmış bir göstergedir. Uygulama geliştiriciler imleçleri kullanarak, SQL cümlesinin çalıştırılma aşamalarında daha fazla kontrol sağlayıp uygulama performansını arttırabilirler.

### 2.7.2. Program Genel Alanı (PGA- Program Global Area)

PGA sunucu işlemleri için veri ve kontrol bilgilerini tutan bellek tamponudur. Oracle bir sunucu işlemini başlattığında PGA otomatik olarak başlatılır. Genellikle PGA bir kullanıcı ya da bağlantı için ayrılan belleğe denir. Bu bellek üç bölümü içerir:

Birincisi yığın uzayı'dır (*Stack Space*). Yığın her bir bağlantıya ait değişkenlerin ve dizilerin yapılarını tutan bellektir.

İkincisi bağlantı bilgisi'dir (*Session Information*). Bağlantı bilgisi, "multi- threaded" sunucu olarak adlandırılan bir bilgisayarda çalışılmıyorsa PGA alanında, aksi halde SGA'da depolanır ("multi-threaded" uygulamalar aynı kod ve data segmenti

kullanıp, farklı program sayacı, kayıtlık ve yığın kullanan uygulamalar için kullanılır).

Üçüncüsü Özel SQL alanıdır. Bu alan farklı amaçla kullanılan bazı değişkenleri tutmak için kullanılır.

### 2.7.3. İşlem Yapısı (Process Architecture)

İşlem, işletim sistemlerinde, belli bir işi yapmak için bir adımlar dizisinin çalıştırılması ile oluşur. Bir işlemin çalışabilmesi için bellekte kendine özel bir ayrılmıştır. Oracle'da da iki tür işlem vardır. Kullanıcı işlemleri ve Oracle İşlemleri.

- **Kullanıcı İşlemleri**

  Kullanıcı işlemi bir uygulama ya da yazılımın çalıştırılmasını sağlamak için oluşturulur. Bu duruma örnek olarak *Enterprise Manager* uygulaması verilebilir. Kullanıcı işlemleri program ara yüzü (program interface) yoluyla sunucuyla iletişim işlemlerini sağlar.
  Program ara yüzü bir kullanıcı işlemlerinin sunucu ile iletişim kurmasında kullanılan mekanizmalar olarak bilinir.

- **Oracle İşlemleri**

  Oracle işlemleri diğer işlemler tarafından işlemin belli adımlarını gerçekleştirmesi için çağrılırlar. Oracle işlemleri de sunucu ve arka plan işlemleri olarak ikiye ayrılır.

  - **Sunucu İşlemleri**
    Oracle, VT'na bağlanan bir kullanıcının isteklerini gerçekleştirebilmek için sunucu işlemlerini başlatır. Örneğin bir kullanıcının, o an SGA'nın VT belleğinde yer almayan bir bilgiyi

sorgulaması, veri bloklarının veri dosyalarından okunup SGA'ya getirilmesini sağlayan sunucu işlemini başlatır. İstemci/Sunucu mimarili sistemlerde kullanıcı işlemleri ve sunucu işlemleri ayrı bilgisayarlarda çalıştırılır.

- **Arka Plan İşlemleri**

  Oracle yaratılan her VT için ayrı olarak bir dizi arka plan işlemleri oluşturur. Oracle, VT'na aynı anda birden fazla kullanıcının bağlandığı ve belli programları çalıştırdığında bu kullanıcı işlemlerini gerçekleştirmek için arka planda bazı işlemler gerçekleştirir. Her bir VT kendi arka planı işlemlerine sahiptir. Şimdi bu arka plan işlemlerinin neler olduğuna bakalım:

  - **VT Yazıcısı (Database Writer- DBWn)**

    VT yazıcısı, VT tampon belleğindeki değiştirilmiş veri bloklarını veri dosyalarına yazmakla görevlidir. Normalde tek bir VT yazım işlemi birçok sistemde yeterli olmasına rağmen, birden fazla yazım işlemi tanımlanabilir. Bu işlemlere DBW0,...,DBW9 şeklinde isim verilir. VT açılırken bu *DB_WRITER_PROCESSES* parametresi ile VT'na bildirilir.

    Bir kullanıcı yaptığı değişiklikleri "commit" komutuyla onayladığında VT yazıcısı bu değişiklikleri hemen veri dosyalarına kaydetmez. VT yazıcısı veri dosyalarına yazma işlemini kendi belirler ve ya SGA içerisine çok miktarda başka verilerin alınması gerektiği zaman ya da çok az VT tamponu kaldığı zaman yazma işlemini gerçekleştirir. Veri dosyalarına yazım işlemi en son kullanılan verilerden başlanarak gerçekleştirilir.

- **Log Yazıcısı (Log Writer LGWR)**

  Log yazıcısı SGA'nın redo log tamponundaki bilgileri diske kaydetmek için kullanılır. LGWR tampondaki bilgileri o an kullanımda olan bir redo log dosyasına sıra ile yazar. Bu yazma işlemi VT'nın sahip olduğu birden fazla redo log dosyasına da yapılabilir. Her Oracle anı için bir tane LGWR görevi vardır. Bir hareket (*transaction*) redo log dosyasına işlenmeden kaydedilmiş sayılmaz. Redo Log Tamponlarının yazılma koşulları kısaca şöyledir:

  - Onaylama (*Commit*) görüldüğünde,
  - Redo log tampon doluluğu eşik değerine ulaştığında,
  - DBWR kontrol noktası (*checkpoint*) için tampon blokların temizlemeye gerek duyarsa,
  - Zaman aşımı (Time-out) görülürse.

- **Değişme Noktası (Checkpoint- CKPT)**

  Genel anlamıyla CKPT'nin görevi, LGWR üzerindeki yükü azaltmaktır. Yani belirli zamanlarda SGA içerisindeki değişikliğe uğramış VT tamponları DBWn tarafından belleğe yazılır. Bu işlem değişme noktası işlemi olarak adlandırılır. Değişme noktası işlemi DBWn'e değişme anlarını haber vermekten ve VT'ndaki bütün veri dosyalarını ve kontrol dosyalarını yeni değişme noktasından haberdar etmek için güncellemekten sorumludur.

- **Sistem Analizi (System Monitor-SMON)**

  Sistem Analizi VT oturumu açılırken oturum için kurtarma yapar, yani kontrol dosyalarını kontrol ederek geri alınması gereken bir işlemin olup olmadığına bakar, eğer varsa geri alma işlemini gerçekleştirir. Birden fazla VT oturumunun olduğu ortamlarda SMON aynı zamanda bozulan sistemler

içinde ayrı ayrı kurtarma yapar. SMON aynı zamanda kullanılmayan geçici parçaları temizlemekte ve herhangi bir problemden dolayı bozulan işlemleri kurtarmaktadır. Bozulan işlemlerin sorgu komutları SMON tarafından tablo uzayı ve veri dosyası tekrar aktif hale getirildikten sonra kurtarılır. Son olarak SMON VT'nında daha fazla boş yer açılsın diye boş genişlemeleri birleştirmektedir. Özetlemek gerekirse SMON aşağıdaki işlemleri yerine getirir:

- Otomatik Oracle anı kurtarmayı gerçekleştirir.
- Geçici segment alanını geri elde eder.
- Kontrol dosyasının sürekliliğini sağlar.
- Sistemde kullanılabilir durumdaki serbest alanın kaydını tutar.

- **İşlem Analizi (Process Monitor-PMON)**

  İşlem Analizi herhangi bir kullanıcı işlemi bozulduğunda o işlemin kurtarılmasını yapmaktadır. PMON işlemin kullandığı belleği ve kaynakları temizlemekten sorumludur. PMON aynı zamanda dispatcher (ileriki bölümlerde anlatılmaktadır) ve sunucu işlemlerini kontrol eder ve kapandıklarında yeniden çalıştırır. Özetlemek gerekirse PMON aşağıdaki işlemleri gerçekleştirir.

  - Anormal bir şekilde kesilen bağlantıları temizler.
  - Onay (commit) verilmemiş değişiklikleri eski haline getirir (rollback).
  - İşletimi kesilen görevin tuttuğu kilitleri kaldırır.
  - Çakılan görev için ayrılan SGA kaynaklarını serbest bırakır.
  - Kilitlenmeleri (deadlock) otomatik olarak yakalar ve işlemi geri döndürerek (transaction rolling back) çözümler.

- **Yedekleyici (Archiver-ARCn)**

  Yedekleyici, görevi aslında seçimlik bir arka plan görevi olmasına rağmen, birçok sistem için özellikle tavsiye edilir. Eğer bu görev çalıştırılıyorsa VT ARCHIVELOG kipinde çalışıyor demektir.

  Yedekleyici o an kullanılmakta olan redo log dosyalarını, doldukları zaman yedek depolama birimlerine kopyalar. Tüm sistemler için bir ARC0 işleminin olması yeterli olsa da birden fazla işlem gerçekleştirilebilir. Bu LOG_ARCHIVE_MAX_PROCESSES parametresi ile belirlenir. ARCn işlemi VT ARCHIVELOG modda çalışırken kullanılır.

  Eğer Archivelog modda çalışılıyorsa:

  - Tablo uzaylarının (Tablespace) çevrim-içi yedeklenmesine,
  - Medya arızasından (failure) çevrim-içi kurtarmaya,
  - Günlük dosyalarının otomatik olarak arşivlenmesine izin verilir.
  - ARCH görevi, günlük dosyalarının kopyalarını, yerleri daha önce belirlenmiş disk ya da teyp birimleri üzerine çıkarır.

- **Geri Kurtarıcı (Recoverer-RECO)**

  Geri kurtarıcı dağıtık VT'nında sistem veya ağ hatalarından dolayı bekleyen işlemleri düzenler. Belli aralıklarla, yerel RECO uzaktaki VT'na bağlanıp yereldeki dağıtık işlemlerle ilgili Onay (*commit*) ve Geri Alma (*rollback*) işlemlerini yapar.

- **Dağıtıcı (Dispatcer-Dnnn)**

  Dağıtıcılar çoklu ortamlarda isteğe bağlı olarak çalıştırılmaktadırlar. Her iletişim protokolü için en az bir dağıtıcı işlemi (D000,....,Dnnn) oluşturulmaktadır. Her dağıtıcı işlemi kullanıcı işlemlerinden gelen istekleri sunucu işlemlerine yönlendirmekte ve gelen cevapları da uygun kullanıcılara tekrar döndürmekten sorumludur.

- **Kilit (Lock-LCKO)**

  Kilit işlemleri birden fazla VT oturumunun çalıştığı sistemlerde veri tabanları arasında gereken bir takım kilitleme işlemlerini gerçekleştirir.

- **İş Kuyruğu (Job Queue-SNPn)**

  Dağıtık VT uygulamalarında 38 adetten fazla (SNP0,...,SNP9, SNPA, ....SNPZ) iş kuyruğu işlemi tablo snapshot'larını otomatik olarak güncelleyebilir. Bu işlemler periyodik olarak başlatılır.

## 2.8. Veri Sözlüğü

Oracle veri sözlüğü, VT hakkında referans bilgisi içeren tablolar ve görüntüler kümesinden oluşur. Veri sözlüğü, SYS kullanıcı tarafından sahiplenen tablolar ve görüntülerden oluşur. VT'na salt-okunur bilgi sağlamak amacıyla kullanılır. Veri sözlüğü aşağıda belirtilen bilgileri de tutar:

- Oracle kullanıcılarının kullanıcı bilgisi
- Kullanıcılara verilen roller ve haklar
- Şema nesnelerinin adları ve tanımları
- Tablolar üzerindeki bütünlük sınırlamaları
- VT nesneleri için ayrılan alanlar
- Genel VT yapısı

- Yordamların ve tetikleyicilerin işlevleri

VT oluşturulurken Veri Sözlüğü de oluşturulur. Oluşturulan bilginin güncel kalabilmesi için belirli durumlarda Oracle, veri sözlüğünde gerekli değişiklikleri otomatik olarak yapar. VT yaptığı işleri kaydetmek ve onaylamak için veri sözlüğünü kullanır.
Örneğin; Bir kullanıcının belli bir tabloya, görüntüye veya herhangi bir şema nesnesine ulaşmak için yeterli yetkiye sahip olup olmadığını kontrol etmek için Oracle, veri sözlüğünü kullanır.

Veri sözlüğü içerisindeki bilgiler SQL komutlarıyla sorgulanıp görülebilir. SYS Kullanıcısı dışındaki hiçbir kullanıcı veri sözlüğünü değiştiremez, ekleme yapamaz ve kayıt silemez. VT oluşturulurken oluşturulan veri sözlüğü tabloları SYS adlı kullanıcıya aittir. Veri sözlüğü tabloları için görüntüler oluşturulmuştur. Tüm kullanıcılar hakları olduğu müddetçe bu görüntülerden bir ya da birkaçını sorgulayabilirler. Veri sözlüğü görüntüleri, başlarındaki ön eklerine göre üç gruba ayrılır:

**USER_xxx:** VT'na o an bağlı olan kullanıcının, sadece kendine ait olan nesneleri görüntüler.
**ALL_xxx:** VT'na o an bağlı olan kullanıcının, kendine ait olan ve başka kullanıcılar tarafından kendisine verilen haklara ait tüm nesneleri görüntüler.
**DBA_xxx:** DBA (VT yöneticisi) veya DBA hakkına sahip kullanıcıların görebileceği görüntülerdir.

Yukarıdaki anlatılanlara örnek vermek gerekirse;

**Örnek–1:** SCOTT kullanıcısının kendine ait olduğu bütün nesneleri görmesi için aşağıdaki SQL cümlesinin yazılması gerekir;

```
SELECT * FROM USER_OBJECTS
```

**Örnek–2:** SCOTT kullanıcısının ulaşabildiği bütün nesneleri görmesi için aşağıdaki SQL cümlesinin yazılması gerekir;

```
SELECT * FROM ALL_OBJECTS
```

## 2.9. Yedek Alma ve Tekrar Kullanma

Veri tabanları sistem ya da donanım hatası olasılığıyla karşı karşıya kalabilir. Oracle, verileri her ne kadar güvenli tutsa da, fiziksel sebeplerden (disk hataları, doğal afetler gibi), kullanıcı hatalarından veya benzer durumlardan dolayı yedeklere ihtiyaç duyulmaktadır. Yedek alma, VT Yöneticisinin yapacağı en önemli işlerden birisidir.

Oracle'da, yedek almayla ilgili değişik yöntemler vardır. VT yöneticisi, kendi VT'nın durumuna göre, yedek almayla ilgili bir veya birden fazla yönetimi kullanmaya karar vermelidir. Bu kararı vermede, VT yöneticisi; yedeğin kimin tarafından alınacağı, hangi medya aracılığı ile alınacağı, ne kadar sıklıkta alınacağı, alınan yedeklerin büyüklüğü, VT'nında küçük de olsa bir kayda tahammül olup olmadığı gibi çok değişik sebepleri göz önünde bulundurmak durumundadır.

### 2.9.1. Fiziksel Yedek Alma ve Kurtarma

Fiziksel yedek alma veri dosyalarının, redo log dosyalarının ve kontrol dosyalarının yedeklerinin alınması işlemidir. Bir VT *ARCHIVELOG* ve *NOARCHIVELOG* olmak üzere iki farklı modda çalışabilir. *ARCHIVELOG* modunda yapılan tüm işlemler redo log dosyalarına otomatik olarak kaydedilir. Bunun anlamı VT'ndaki değişikliklerin sürekli dosyalara kaydedilmesidir. Kaydedilen bu dosyaların hangileri olduğu ve nerede bulundukları gibi bilgiler "*init<SID>.ora*" dosyası içerisinde yer alır. *ARCHIVELOG* modda çalışan bir VT'nında veri kaybı söz konusu değildir. Eğer VT *NOARCHIVELOG* modda çalışıyorsa yapılan değişiklikler bir yere kaydedilmeyecektir. Bu yüzden sistemin bozulması durumunda ancak son alınan

yedekler geri getirilebilir. Yani son alınan yedekten sonra yapılan değişiklikler kaybolur. İşletim sistemi yedeği alma olarak da bilinen fiziksel yedek alma işleminin dezavantajı, yedek alma işlemi boyunca VT'nın kapatılması gereğidir.

#### 2.9.1.1. NOARCHIVELOG Modu

İşletim sistemi yedeği (NOARCHIVELOG MODE) alma, daha çok VT'ndaki verisi az olan ve VT'nın sürekli açık kalması zorunluluğu bulunmayan ve az da olsa veri kaybına tahammül edebilecek VT yöneticilerinin kullanacağı bir yöntemdir. Çünkü bu yöntemle yedek alırken VT kapatılmak zorundadır, bu yüzden sistem sürekli açık kalamaz. İkinci olarak, bütün log, kontrol ve veri dosyalarının teker teker yedeği alınacaktır ve bu işlem uzun sürebilir. Bu yüzden verisi az olan veri tabanlarında kullanılması önerilmektedir. Üçüncü olarak, bu sistemde herhangi bir göçme esnasında bir önceki yedek ile bir sonraki yedek arasında yapılan işlemler kaybolmaktadır, bunu dengelemek için iki yedeğin ara süresi kısıtlanabilir fakat bu sefer de performans kaybı ve sistemin sürekliliğinin kesilmesi durumu ortaya çıkmaktadır. Bu yöntemde en ideal zaman, geceleri günde bir defa yedek almadır. Bu sebeplerden dolayı, bu yöntemi kullanan VT yöneticisi az da olsa veri kaybına tahammül edebilecek durumda olmalıdır.

Yedek alma işlemi aşağıda adım adım anlatılmıştır. Oracle VT'na bağlantı SQL*Plus ile gerçekleştirilmiştir.

A. SQL*Plus ile bağlanılır.

```
sqlplus system/manager;
```

B. Yedeği alınacak dosyaların tespiti yapılır.

```
SQL> SPOOL YEDEK1;
SQL> SELECT NAME FROM V$CONTROLFILE
     UNION
     SELECT MEMBER FROM V$LOGFILE
     UNION
```

```
SELECT FILE_NAME FROM DBA_DATA_FILES;
SQL>  SPOOL OFF;
```

C. SQL Manager ile bağlanarak VT kapatılır.

```
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> SHUTDOWN IMMEDIATE;
SVRMGRK> EXIT;
```

D. Listener kapatılır.

```
lsnrctl stop
```

İşletim sistemindeki "*YEDEK1*" adlı dosyanın içerisindeki dosyalar ve "*init<sid>.ora*" dosyası işletim sisteminde bir başka yere veya bir medya birimine (data type, cd gibi) ya normal ya da herhangi bir sıkıştırma programı (compress benzeri) ile sıkıştırılarak kopyalanır (taşınmaz). Kopyalama işlemi bittikten sonra VT açılır.

E. SQL Manager ile bağlanarak VT açılır.

```
SVRMGRL > CONNECT INTERNAL;
SVRMGRL > STARTUP;
```

F. Listener Açılır.

```
lsnrctl start;
```

Alınan yedekten geri dönme işlemi aşağıda adım adım anlatılmıştır.

A. SQL Manager ile bağlanarak VT kapatılır.

```
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> SHUTDOWN IMMEDIATE;
SVRMGRK> EXIT;
```

B. Listener kapatılır.

```
lsnrctl stop;
```

C. İşletim sisteminden alınan yedekler tekrar eski yerlerine kopyalanır. Kopyalama işlemi bittikten sonra VT açılır.

D. SQL Manager ile bağlanarak VT açılır.

```
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> STARTUP;
```

E. Listener Açılır.

```
lsnrctl start;
```

#### 2.9.1.1. ARCHIVELOG Modu

İşletim sistemi yedeği alma (archivelog mode), VT'nın sürekli açık kalması zorunluluğu bulunan ve az da olsa veri kaybına tahammül edemeyecek VT yöneticilerinin kullanacağı bir yöntemdir.

Bu modda yapılan değişiklikler otomatik olarak redo log dosyalarına yazıldığı için dosyaları tek tek kopyalamak gerekmez. Fakat bu değişikliklerin kaydedildiği dosyaları yedeklemek yararlı olabilir. VT'nında bir bozukluk durumunda yapılacak iş VT'nı kapatıp "*Recover Database*" komutunu kullanmak ve sonra VT'nı yeniden açmaktır. "*Recover Database*" komutunu kullanmak için VT "*Mount*" modunda açılır.

Yedek alma işlemi aşağıda adım adım anlatılmıştır.

A. "*init <sid>.ora*" isimli dosyada aşağıdaki değişiklikler yapılmalıdır.

*LOG_ARCHIVE_START* parametresine "TRUE" değeri atanır.

*LOG_ARCHIVE_DEST* parametresine geri alma dosyalarının (redo log files) yedeklerinin alınacağı dizin adresi atanır.

B. Bu işlemler yapıldıktan sonra, VT'nında yapılan bütün değişikliklerin yazıldığı geri alma dosyaları (*redo log files*) VT tarafından her seferde otomatik olarak "LOG_ARCHIVE _DEST" parametresi ile belirtilen yere yedeği alınır. Geri alma dosyalarının (*redo log files*) işletim sistemindeki yeri, "*V$LOGFILE*" dinamik performans tablosunda bulunmaktadır."*LOG_ ARCHIVE_DEST*" parametresi ile belirtilen yere alınan yedekler belli aralıklarla eski olanlar bir medya birimine (data type, cd ,vb) alınır. Böylece, VT'nın yedeği alınmış olmaktadır.

Alınan yedekten geri dönme işlemi aşağıda adım adım anlatılmıştır. Oracle VT'na bağlantı SQL*Plus ile gerçekleştirilmiştir.

A. SQL Manager ile bağlanarak VT kapatılır.

```
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> SHUTDOWN IMMEDIATE;
SVRMGRK> EXIT;
```

B. Listener kapatılır.

```
lsnrctl stop;
```

C. İşletim sisteminden alınan yedekler tekrar eski yerlerine kopyalanır. Kopyalama işlemi bittikten sonra VT "MOUNT" modda açılarak günlük dosyalarının (redo log files) yedekleri teker teker geri getirilir.

D. SQL Manager ile bağlanılır.

```
SVRMGR>CONNECT INTERNAL;
SVRMGR>STARTUP MOUNT;
SVRMGR>RECOVER DATABASE;
```

Bu komuttan sonra, VT gerekli olan günlük dosyası yedeklerini söyler eğer yerleri değiştirilmediyse, VT dosyaları o adreste arar, bulur ve uygular.

E. SQL Manager ile bağlanarak VT açılır.

```
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> STARTUP;
```

F. Listener Açılır.

```
lsnrctl start;
```

### 2.9.2. Mantıksal Yedek Alma ve Geri Getirme

Oracle'da mantıksal yedek alma denince dışarı veri aktarma (*Export*) ve içeri veri alma (*Import*) anlaşılır. Bunlar Oracle firmasının geliştirdiği komut modunda çalışan yardımcı programlardır. Mantıksal yedek alma, daha çok VT'nın sürekli açık kalması zorunluluğu bulunan ve az da olsa veri kaybına tahammül edebilecek VT yöneticilerin kullanılacağı bir yöntemdir. Çünkü bu yöntemde yedek alırken VT kapatılmak zorunda değildir. Bu tür yedek alma VT nesnelerinin yedeklenmesi olduğu için mantıksal yedek olarak adlandırılır. Yani export ile log dosyalarının ya da kontrol dosyalarının yedeği alınmaz.

Mantıksal yedek almanın avantajları şunlardır:

- Tarihlere göre yedek almak daha kolaydır. Günlük dışarı veri aktarımları (export) VT'nın durumunu ve gelişmesini günlük olarak takip etmede en kolay yöntemdir.

- VT'nın yapısının verileri olmaksızın yedeği alınabilmektedir. Bu imkânla, herhangi bir uygulamaya ait tablo, indis, kısıtlama ve sıra gibi nesneler başka bir ortama kolayca alınabilir.

- Veri tabanları arasında veri kopyalama işlemi için kullanılabilmektedir. Oracle'ın değişik sürümleri arasında veri taşımak için kullanılabilmektedir.

#### 2.9.2.1. Dışarı Veri Aktarma

Dışarı veri aktarma, Oracle'ın mantıksal yedek alma işlemleri için geliştirdiği bir üründür. Veri aktarma, VT'ndaki verilerin anlık olarak görüntüsü işletim sistemi dosyası haline dönüştürülerek yedek alınması işlemine denmektedir. Dışarı aktarılan veriler ile alınan yedekler ancak İçeri veri alma işlemi (Import) ile gerçekleşir.
Dışarı veri aktarma seçenekleri:

A. Tablo Modu (*Table Mode*)'dur. Bu modda kullanıcılar kendi tablolarının yedeklerini alabilirler ya da hakkı olan kullanıcı bir başka kullanıcının tablolarının yedeklerini alabilir.

B. Kullanıcı Modu (*User Mode*)'dur. Bu modda bir kullanıcının nesnelerinin yedeği alınabilir.

C. Sonuncusu ise Tam VT Modu'dur (*Full Database Mode*). Bu modda tüm VT'nın yedeği alınabilir. *EXP_FULL_DATABASE* rolüne sahip olan kullanıcılar bu yedeği alabilir.

Dışarı veri aktarma programını çalıştırmak için komut satırına geçilmelidir. "*Exp.exe*" programı komut satırına yazılarak kullanılır.

**Komut Yapısı:**

```
EXP <anahtar> = (<değer>,<değer>,…)
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**Değer:** Anahtar'ın durumuna göre ilgili anahtara karşılık bir veya birden fazla olarak belirtilecek değerlerdir.

**Anahtar olarak kullanılacak ifadeler şunlardır:**

**USERID:** Kullanıcı adı ve şifresini belirtir. (*USERID = SCOTT/TIGER*)
**FULL:** Bütün VT'nın yedeğinin alınmasını belirtir. (*FULL = Y | FULL =N*)

**BUFFER:** Veri tamponunun büyüklüğünü belirtir. (*BUFFER = 2400* )

**OWNER:** Yedeği alınacak nesnelerin sahibini belirtir. (*OWNER = SCOTT*)

**FILE:** Yedek alınacak dosyanın işletim sistemindeki yerini belirtir. (*FILE = /disk1/yedek/yedek1.dmp*)

**TABLES:** Yedeği alınacak tablo veya tabloların listesini belirtir. (*TABLES = OGRENCI, OKUL*)

**COMPRESS:** Bir genişleme parçasına yedek alınmasını belirtir. (*COMPRESS= Y | COMPRESS= N*)

**RECORDLENGTH:** Giriş / Çıkış verisinin büyüklüğünü belirtir. (*FILERECORD=1000*)

**GRANTS:** Kullanıcının haklarının yedeğinin alınması gerektiğini belirtir. (*GRANTS = Y | GRANTS =N*)

**INCTYPE:** Artan veri aktarımı (export) alınmasını ifade eder. (*INCTYPE = COMPLETE | INCTYPE = INCREMENTAL*), şeklinde kullanılır.

**COMPLETE:** VT'nın tam olarak yedeğinin alınmasının belirtir.

**CUMULATIVE:** Bir önceki *CUMULATIVE* ya da *COMPLATE* veri aktarma işleminden sonra değişen tabloların yedeğinin alınmasını belirtir.

**INDEXES:** İndekslerin de ihracın içine alınmasını belirtir. (*INDEXES=Y | INDEXES = N*)

**RECORD:** INCREMENTAL ihracın VT'na kaydedilmesini belirtir. (*RECORD = Y | RECORD= N*)

**ROWS:** Tablonun kayıtlarının yedeğinin alınmasının belirtir.(*ROWS=Y|ROWS =N* )

**PARFILE:** Parametre dosyasını belirtir. (*PARFILE=/disk1/exp.par*)

**CONSTRAINTS:** Kısıtlamaların yedeğinin alınmasını belirtir. (*CONSTRAINTS= Y | CONSTRAINTS= N )*

**CONSISTENT**: İlişkili tabloların bütünlük içerisinde yedeğinin alınmasını belirtir. (*CONSISTENT= Y | CONSISTENT= N*)

**LOG:** Yedek işleminin sonucunun bir dosyaya alınmasını belirtir. (*LOG= /disk1/exp.log*)

**STATISTICS:** Dışarı aktarılan veri geri alınırken gerekli optimizasyonların yapılmasını belirtir. (*STATICS=COMPUTE | STATICS=ESTIMATE veya STATICS=NONE*)

**DIRECT:** Dışarı aktarılan kayıtların geri alma parçalarına (rollback segment) yazılmadan direk olarak dışarı aktarımın yapılmasını belirtir. (DIRECT=N | DIRECT=Y)

**Örnek – 1:** Bu örnek VT'nın tümünün yedeğini alır.

```
EXP system/manager FULL=Y FILE=SCOTT_yedek.dmp
```

**Örnek – 2:** SCOTT kullanıcısının tüm nesneleri ve verilerini SCOTT_yedek.dmp dosyasına alır.

```
EXP SCOTT/TIGER FILE =SCOTT_yedek.dmp
```

**Örnek – 3:** Bu örnek SCOTT kullanıcısının sadece "OGRENCI" ve "OKUL" tablolarının yedeğini alır.

```
EXP SCOTT/TIGER TABLES(OGRENCI, OKUL) ROWS=n
```

**Örnek – 4:** Bu örnek SCOTT kullanıcısının nesnelerinin yapısını içerisinde kayıt olmadan alır.

```
EXP SCOTT/TIGER FILE=SCOTT_yedek.DMP ROWS=N
```

### 2.9.2.2. İçeri Veri Alma İşlemi

İçeri veri alma işlemi, dışarı aktarılan veri yedeklerini geri almak için kullanılır. Veri alma komutu ile yedeklenmiş dosyanın tamamı ya da bir kısmını geri alınabilir.

**Komut Yapısı:**

```
IMP <anahtar> = (<değer>,<değer>,…)
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**Değer:** Anahtar'ın durumuna göre ilgili anahtara karşılık bir veya birden fazla olarak belirlenir.

**Anahtar olarak kullanılacak ifadeler şunlardır:**

**USERID:** Kullanıcı adı ve şifresini belirtir. (*USERID = SCOTT/TIGER*)
**FULL:** Bütün VT'nın yedeğinin geri alınmasını belirtir. (*FULL = Y | FULL =N*)
**BUFFER:** Veri tamponunun büyüklüğünü belirtir.(*BUFFER = 2400*)
**FROMUSER:** Dışarı aktarılan tabloların sahiplerini belirtir. (*FROMUSER=SCOTT, ALP*)
**TOUSER:** Verinin içeriye hangi kullanıcı veya kullanıcılara yapılacağını belirtir. (*TOUSER= SCOTT, ALP*)
**FILE:** İşletim sistemindeki yedek dosyasının yerini belirtir. (*FILE = /disk1 /yedek/yedek1.dmp*)
**SHOW:** İçeri alınacak veri dosyasının sadece listelemenin belirtildiği bölümdür. (*SHOW=Y | SHOW=N*)
**TABLES:** İçeri alınacak tablo veya tabloların listesini belirtir. (*TABLES = OGRENCI, OKUL*)
**IGNORE:** Nesneler, içeri aktarılmadan önce yaratılmaktadır. Eğer nesne var ise yaratılmaması için bu anahtarın belirtilmesi gerekmektedir. Eğer, Y (YES) olarak belirtirse nesne içeri aktarılmadan önce yaratılmaktadır. Standart değer, N(NO)'dur. (*IGNORE = Y | IGNORE = N*)
**COMMIT:** Her kayıt dizesi içeri aktarıldıktan sonra COMMIT edilmesini belirtir. (*COMMIT = Y | COMMIT = N*)
**RECORDLENGTH:** Giriş / Çıkış kaydının büyüklüğünün belirtildiği bölümdür. (*FILERECORD=1000*)
**GRANTS:** Kullanıcı haklarının da tekrar oluşturulması gerektiğini belirtir. (*GRANTS = Y | GRANTS =N*)
**INCTYPE:** Artan dışarı veri çıkma, alınan bir yedeğin içeri veri alınacağını belirtir. (*INCTYPE= SYSTEM |I NCTYPE= RESTORE*)
**INDEXES:** İndekslerin içeri alınmasını belirtir. (*INDEXES = Y | INDEXES= N*)

**RECORD:** INCREMENTAL dışarı veri çıkma VT'nında kaydedilmesinin belirtildiği bölümdür. (*RECORD = Y | RECORD= N*)

**ROWS:** Tablonun kayıtlarının içeri aktarılmasının belirtildiği bölümdür.(*ROWS=Y | ROWS=N*)

**PARFILE:** Parametre dosyasının belirtildiği bölümdür. (*PARFILE =/disk1/exp.par*)

**LOG:** Ekrandaki işlemlerin bir dosyaya alınmasını belirtir. (*LOG=/disk1/oracle/exp.log*)

**DESTROY:** VT'nı oluşturan ve dosyalarının tekrar kullanılmasının belirtir. Eğer, Y (YES) olarak belirtilirse veri dosyaları REUSE seçeneğiyle birlikte yaratılır ve sonradan tablespace'ın içi boşaltılırsa veri dosyası tekrar kullanılabilir. Standart değeri, N (NO)'dur. (*DESTROY = Y | DESTROY*=N)

**INDEXFILE:** İndeks yaratma komutlarının yer aldığı ayrı dosyayı belirtir. Standart değeri boştur. (*INDEXFILE= <dosya ismi>)*

**Örnek – 1:** Bu örnekte SCOTT kullanıcısının bütün nesneleri içeri aktarılacaktır.

```
IMP SCOTT/TIGER FILE= SCOTT_yedek.dmp
```

**Örnek – 2:** Bu örnekte "*SCOTT_yedek*" yedek dosyasından sadece "*OGRENCI*" tablosu içeri aktarılacaktır.

```
IMP SCOTT/TIGER FILE= SCOTT_yedek.dmp TABLES=OGRENCI IGNORE=Y
```

**Örnek – 3:** Bu örnek SCOTT kullanıcısının nesnelerini "*ALP*" kullanıcısına aktaran komuttur.

```
IMP SCOTT/TIGER FILE= SCOTT_yedek.dmp FOMUSER=SCOTT
TOUSER=ALP
```

# BÖLÜM 3

## 3. PERFORMANS ANALİZİ

Bilgisayar sistemlerinin hızlı gelişmesi, büyümesi ve karmaşıklaşması, internetin iş uygulamalarında daha fazla rol alması sistem performansını önemli hale getirmiştir. Performans problemleri özellikle bazı sistem kaynaklarının tüketildiği ya da bir kaynak üzerine yığılmaların olduğu durumlarda oluşur. Performans yönetimi böyle durumlarda devreye girer.

Oracle Performans Yönetimi ile Oracle performans problemlerinin sistematik olarak nasıl teşhis edilmesi gerektiği ve bu problemlerin nasıl çözülmesi gerektiği hakkında ipuçları vermesi hedeflenmiştir.

Performans izlemelerinde aşağıdaki durumlara dikkat edilmesi gerekmektedir [40]:

- Performans sorununun neden kaynaklandığının anlaşılması, ölçümlerin ve yapılandırma ayarlarının kontrol edilmesi
- İşletim sisteminin kontrol edilmesi
- Oracle istatistiklerinin toplanması
- Performans raporunun incelenmesi; Genel yaklaşım, VT sisteminin genel aktivitelerini içeren istatistikler, bellekler, disk erişimi ve Merkezi İşlem Birimi (MİB) kullanımı gibi
- Performans düşüşlerinin gruplandırılması
- Aşırı MİB tüketimi, yüksek disk erişimine neden olan SQL cümlelerinin bulunması
- Yüksek kaynak tüketen SQL cümlelerinin analiz edilmesi ve izlenmesi
- VT tasarımının genel incelenmesi

Oracle VT'nda performansı iyileştirmek veya teknik deyimiyle optimize etmek için yapılması gereken birçok işlem vardır. Bu bölümde VT kullanıcıları için performans optimizasyonu örnek SQL'lerle anlatılacaktır.

## 3.1. Optimizasyonun Sebepleri

Oracle VT diğer veri tabanlarına nazaran çok güçlü bir sistemdir. Fakat uygulamalara esneklik sağlamasının yanında, çok büyük veri işlemleri, çalışan uygulamalar ve uygulamalarda yazılan yanlış kodlar VT'nın performansını kötü yönde etkilemektedir. Bunların sonucunda VT optimizasyon ihtiyaçları ortaya çıkmaktadır.

VT'nın optimizasyonu hem fiziksel hem de mantıksal yapıdaki düzenlemelerle yapılmaktadır. VT'nın optimize edilmesini gerektiren başlıca sebepler şunlardır.

- Yanlış Tasarım

  VT ilk başta tasarlanırken yanlış tasarlanmış olabilir. Yani tablolar, aralarındaki ilişkiler, indisler ve diğer mantıksal nesneler iyi tasarlanamamış olabilir. Bununla beraber veri dosyalarının, kontrol dosyalarının yerleri, işletim sistemi ve ağ (network) gibi fiziksel nesneler de iyi tasarlanamamış olabilir.

- VT'nın Büyümesi

  Her geçen gün VT büyüdükçe yapılan tasarım tekrar gözden geçirilmelidir. Büyüme ile doğru orantılı olarak yeni gelen istekler ve gereksinimler gözden geçirilmelidir. Gerektiğinde yeni tasarımlara gidilmesi gerekmektedir.

- Değişen Gereksinimler

Mevcut uygulamalarda, yeni istekleri karşılamak için yeni eklentiler yapılmaktadır. Yapılan bu eklentiler uygulamaları hantallaştırmakta ve performans problemlerine sebep olmaktadır.

## 3.2. Performans Aşamaları

Performans optimizasyonu yapmak için gereken adımlar şunlardır:

- En uygun mantıksal tasarımı yapma
- En uygun fiziksel tasarım yapma
- Yapılan fiziksel tasarımı tekrar gözden geçirerek gerekirse yeniden tasarım yapma
- İyi tasarlanmış bir uygulama programı yapma
- Yapılan uygulamanın tasarımını tekrar gözden geçirerek gerekirse yeniden tasarım yapma
- VT parametrelerini yani; tampon alanı (cache buffer), VT blok büyüklüğü (db block size) ve paylaşılmış SQL alanı (shared pool area) değerlerini gözden geçirme
- Gerekirse işletim sistemi bellek yapılarını (unix'de Oracle kullanıcısının unlimited shell hakkı olmalıdır, swap alanı uygun büyüklükte olmalıdır, semaphor'lar sistem sorumlusu ile VTY tarafından ayarlanmalıdır) gözden geçirme

## 3.3. Oracle'ın Performans Silahları

Oracle performans yönetimin ana yöntemlerinden biri yüksek okuma ve yazmalara sahip SQL cümlelerinin izlenmesidir. SQL izleme yöntemleri arasında yer alan AUTOTRACE ve ÇALIŞMA PLANI (*EXPLAIN PLAN*) kullanımı, sorgular üzerinde varsayımsal çalışma planı oluşturmaktadır. TRACE ve *V$SQL_PLAN* ise, sorgu için uygulanan gerçek çalışma planını oluşturur. Bu nedenle TRACE ve *SQL$PLAN*'in diğer yöntemlere göre kesin sonuç ürettiği değerlendirmesi

yapılabilir. Bununla birlikte AUTOTRACE, gerek kullanım kolaylığı gerekse aynı oturum içinde izleme ve sorgu çalıştırma işleminin bir arada yapılabilmesi nedeniyle çoğu durumda tercih edilebilir [20,21,31,32,41,42].

### 3.3.1. SQL_TRACE VE TKPROF

SQL Trace belirtilen bir andan yine belirtilen bir ana kadar olan zaman dilimi içerisinde oturum veya VT seviyesinde gerçekleşen tüm olayların MİB de harcanan zaman, ayrıştır-çalıştır-getir aşamalarında geçen süre ve adet bilgileri, G/Ç miktarı gibi SQL' in performansı ile direk alakalı bilgilerin okunabildiği Oracle programıdır. TKPROF da alınan bu SQL Trace'in okunabilirliğini arttırmaya yönelik Oracle ürünüdür.

SQL Trace çalıştırmak için yapılması gereken işlemler aşağıda sıralanmıştır;

A. "i*nit.ora*"'daki *TIMED_SATISTICS* parametresine TRUE değeri atanarak yapılan işlemlerle ilgili zaman bilgisi alınır. Bu işlemin SQL cümlesi aşağıdaki gibidir.

```
ALTER SESSION SET TIMED_STATISTICS = TRUE;
```

B. "i*nit.ora*"'daki *USER_DUMP_DEST* parametresi iz dosyalarının kopyalanacağı dizin belirtilir. VT'nın iz dosyasını nereye attığını öğrenmek için aşağıdaki SQL cümlesi çalıştırılır;

```
SELECT VALUE FROM V$PARAMETER WHERE
NAME="user_dump_dest"
```

"*USER_DUMP_DEST*" bir "init.ora" parametresi olduğu için, SYSTEM seviyesinde aşağıdaki komutla değiştirilebilir:

```
ALTER SYSTEM SET user_dump_dest="d:sqltrace";
```

C. *MAX_DUMP_FILE_SIZE* parametresi iz dosyalarının olabileceği maksimum büyüklük olarak atanır. Bu işlemin SQL cümlesi aşağıdaki gibidir.

```
ALTER SESSION SET MAX_DUMP_FILE_SIZE = UNLIMITED;
```

D. "init.ora"'daki *SQL_TRACE* parametresine TRUE değeri atanarak yapılan işlemlerin iz dosyasına yazılma işlemi başlatılır. Bu işlemin SQL cümlesi aşağıdaki gibidir;

```
ALTER SESSION SET SQL_TRACE = TRUE;
```

E. Bu işlemlerden sonra iz almak istenen kod veya kodlar çalıştırılır.

F. İz alma işlemini durdurmak için aşağıdaki SQL cümlesi çalıştırılır.

```
ALTER SESSION SET SQL_TRACE = FALSE;
```

**Not**:
Bu işlemler VT yönetici tarafından başka kullanıcılar içinde yapılabilir.

```
DBMS_SESSION.SET_SQL_TRACE(TRUE);
DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION(sid,serial#,sql_trace
true/false);
```

Bu işlemler sonucunda "*USER_DUMP_DEST*" dizini altında "xxx.trc" adlı dosya oluşur. Bu dosyayı okuyabilmek için "*TKPROF*" Oracle programcığı çalıştırılır. Komut Windows işletim sisteminde aşağıdaki parametreler ile çalıştırılır:

*TKPROF ..\UDUMP\XXX.TRC D:\SQL_TRACE.TXT EXPLAIN=TEST/TEST SYS=YES WAITS= YES*

TKPROF kullandıktan sonra oluşan dosya içeriği aşağıdaki gibi gözükecektir :

```
EXPLAIN PLAN SET STATEMENT_ID='PLUS4294967295' FOR
SELECT * FROM TEST WHERE OBJECT_ID = 123;
```

| | Count | CPU | Elapsed | Disk | Query | Current | Rows |
|---|---|---|---|---|---|---|---|
| **Parse** | 1 | 0.17 | 0.35 | 0 | 0 | 0 | 0 |
| **Execute** | 1 | 0.00 | 0.01 | 0 | 0 | 0 | 0 |
| **Fetch** | 1 | 0.06 | 0.10 | 13 | 304 | 0 | 2987 |
| Misses in library cache during parse :1 | | | | | | | |
| Misses in library cache during execute:1 | | | | | | | |
| Optimizer mode :CHOOSE | | | | | | | |
| Parsing User id :7 | | | | | | | |

Yukarıda belirtilen iki zaman arasında oluşan istatistiksel bilgi gözükmektedir. Burada kullanılan parametrelerin açıklaması şu şekildedir.

- **COUNT**: Parse (SQL'in yapı kontrolü ), Execute (SQL'in çalıştırılması) ve Fetch (SQL sonucunu VT'ndan alıp getirme) işlemlerinin sayısıdır.
- **CPU**: Saniye birimi olarak her işlemin MİB zamanıdır.
- **ELAPSED**: Saniye birimi olarak her işlemin gerçekleşen süresidir.
- **DISK**: Fiziksel okumaları, veri dosyalarında okunan Oracle bloklarını gösterir.
- **QUERY**: VT tampon alanında bulunan geri alma tampon alanından (rollback buffer cache ) okunan tampon bilgisidir.
- **CURRENT**: VT blok kümeleri VT tampon alanından okunan o anki tamponlar (örnek INSERT, UPDATE, DELETE cümleleri)
- **ROWS**: SQL cümlesi sonucu dönen satır sayısını göstermektedir.

### 3.3.2. Plan Açıklama

Oracle VT'nında çalıştırılan her SQL sorgusu, bir çalıştırma planı (Execution Plan) doğrultusunda işletilir. Bu plan ile hangi indislere (varsa ve uygunsa) erişileceği,

hangi tip "join" işlemlerinin gerçekleştirileceğine karar verilir. Çalışma planı, bir yerden bir yere giderken izlenecek birçok yol arasında en hızlı ulaşımı sağlayacak güzergâhın seçilmesi olarak da düşünülebilir.

Bölüm 5.2.4.5'de VTYA'nda plan açıklamalarının nasıl kullanıldığı gösterilmiştir.

SQL iyileştirmede aşağıdakiler hedeflenmektedir:

- Karışık ve karmaşık alt sorgulardan kaçınarak gereksiz erişimleri engellemek
- Büyük tablolar üzerindeki gereksiz "full-table" erişimlerin indisli erişime dönüştürülmesi
- Küçük tablolar üzerindeki "full-table" erişimleri cache'lemek
- İndeks kullanımının optimum düzeyde olduğunu garantilemek
- Optimal JOIN teknikleri kullanarak sorunları çözmek

Plan açıklama çalışma şekli aşağıda anlatılmıştır.

İzleme işlemini gerçekleştirebilmek için ilk önce bir plan tablosu oluşturulmalıdır. Bu tablo genellikle *$ORACLE_HOME/rdbms/admin* dizini altına yer alan utlxplan.sql dosyası kullanılır. İzleme işlemi SQL*Plus kullanılarak veya bu işi yapan diğer araçlar ile yapılır.

Adım adım yapılması gereken işlemler aşağıda sıralanmıştır;

A. SQL*Plus açılır ve aşağıdaki komut çalıştırılır.

```
SQL> @utlxplan
```

B. Durumu izlenecek olan sorgu SQL*Plus'tan aşağıdaki şekilde çalıştırılır.

```
SQL> EXPLAIN PLAN SET STATEMENT_ID='OGRENCI' FOR
       SELECT OGRENCI_ADI
```

```
        FROM OGRENCI
       WHERE OGRENCI_NO = 123;
```

C. Çalıştırılan sorgunun Planı, aşağıdaki sorgu ile görüntülenir.

```
SQL> SELECT LPAD(' ',2*(LEVEL-1)) || OPERATION || ' '
     || OPTIONS || ' ' || OBJECT_NAME || ' ' ||
     DECODE(ID,0, 'COST = ' || POSITION) "QUERY PLAN"
FROM PLAN_TABLE
START WITH ID=0
AND STATEMENT_ID='OGRENCI'
CONNECT BY PRIOR ID=PARENT_ID
AND STATEMENT_ID='OGRENCI';
```

Bu işlem sonucu oluşan çıktı aşağıdaki gibidir.

| **Query Plan** |
|---|
| **SELECT** STATEMENT Cost = 30 |
| **TABLE ACCESS BY INDEX ROWID** OGRENCI |
| **INDEX RANGE SCAN** IDX_OGRENCI_NO |

Kullanılan plan tablosunun COST kolonunda sorgunun sisteme olan yükünün hesaplanmış değeri tutulmaktadır. Kullanılan en iyileştirici (optimizerin) çalışma yolunu değiştirerek (sorguya yardımcı ek kurallar koyarak, indis ekleyerek, indis kaldırarak, nesnelerin analizini yaparak) hesaplanan yükteki yükselmeler ve azalmalar gözlemlenir. Böylece sorgunun en uygun maliyeti veren çalıştırma yöntemi seçilir. Burada kullanılan parametrelerin açıklaması şu şekildedir [43].

- **FILTER:** Korelâsyon alt sorgusu gibi eşleşen kayıtları daha kaliteli bir hale getirmek için sorguda uygulanacak kriterdir.
- **FULL TABLE SCAN:** Tablo ilk kayıttan son kayda kadar taranmakta ve herhangi bir indis kullanılmamaktadır.
- **INDEX (UNIQUE):** SQL sorgusu belirli bir değeri aramak için her satır için ayrı tekil kayıt indis kullanmaktadır.

- **INDEX (RANGE SCAN):** SQL sorgusunda eşitsizlik ya da BETWEEN kıstası kullanılmaktadır.
- **HASH JOIN:** SQL sorgusundaki tablolar okunur ve hash-key olarak bilinen bir matematiksel hesaplama ile belleğe alınırlar.
- **MERGE JOIN:** SQL sorgusunda FROM cümleciğinde birden fazla tablo yer aldığı zaman bu birleştirme yöntemi kullanılır. Oracle, iki sonuç tablosunu birleşen kolonlar üzerinde bir araya getirerek sıralayacak ve sonra birleşen kolonlar yardımıyla sonuçları bir araya getirecektir.
- **NESTED LOOP:** Bu işlem, tabloları birleştirmenin bir başka yöntemidir. İç içe kullanılan döngü anlamına gelen yöntemde sistem paralel olarak birleştirilen indisler üzerinde döngü içinde ilerleyerek sonuca ulaşmaya çalışmaktır.

### 3.3.3. AUTOTRACE

AUTOTRACE özelliği, sorgunun Açıklama Planının yanı sıra TRACE ve TKPROF'da yer alan istatistiklerin çoğunu (fiziksel okuma sayısı ve toplam okuma gibi) verir. AUTOTRACE'in parametreleri şunlardır:

| | |
|---|---|
| SET AUTOTRACE ON | Sorguyu çalıştırılır, sonucu, çalışma planı ve istatistikleri görüntülenir. |
| SET AUTOTRACE ON EXP | Sadece çalışma planı görüntülenir. |
| SET AUTOTRACE ON STAT | Sadece istatistikler görüntülenir (sorgu çalıştırılır ama sonucu görüntülenmez). |
| SET AUTOTRACE TRACE | Sorgunun çalışma planı ve istatistikleri görüntülenir (sorgu çalıştırılır ama sonucu görüntülenmez). |

Adım adım yapılması gereken işlemler aşağıda sıralanmıştır:

A. SQL*Plus açılır ve aşağıdaki komut çalıştırılır.

```
SQL> SET AUTOTRACE ON
```

B. Durumu izlenecek olan sorgu SQL*Plus'tan aşağıdaki şekilde çalıştırılır.

```
SQL> SELECT COUNT(*) FROM OGRENCI
WHERE SOYADI = 'ALP'
```

C. Sırasıyla çıktılar aşağıdaki gibi olacaktır.

```
Count(*)
----------
9
```

```
Query Plan
----------
0 SELECT STATEMENT Optimizer=CHOOSE
1 0 SORT (AGGREGATE
2 1 INDEX (RANGE SCAN) OF 'IDX_OGRENCI01' (NON_UNIQUE)

Statistics
----------
0 recursive calls
0 db block gets
1 consistent gets
1 physical reads
0 redo size
223 bytes sent via SQL*Net to client
274 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
1 sorts (memory)
0 sorts (disks)
1 rows processed
```

### 3.4. Performans Çözüm Hedefleri

VTY için Oracle VT sürüm 8'den itibaren ileri seviye konuların gelişmesi başlamış oldu. VT ve kişisel SQL sorgular çok fazla değişmezken, bu sorgular ile geri alınan veri, kısımlara ayrılan tablolar ve cost-based en iyileştirici fonksiyonlar değişti ve

analiz edilmeye başlandı. Bu bölümde kişisel sorguların daha performanslı nasıl çalışacağı ve analiz edileceği gösterilmiştir [30].

Bölüm 5.2.4.9'da VTYA kullanılarak VT sistem parametrelerine (*V$PARAMETER* ) nasıl bakılacağı gösterilmiştir.

- **Hedef–1**: Oracle için yeterli bellek ayrılmış mı?

İlk hedefimiz donanımsal bellekten, Oracle için yeterli bir yer ayrılıp ayrılmadığını ve buna göre Oracle parametrelerinin düzenlenip düzenlenmediğini araştırmaktır. Bunun için, mevcut ayarlara, nelerin kurulu olduğuna ve anahtar parametrelerine bakılacaktır. (DB_BLOCK_BUFFERS, SHARED_POOL_SIZE, ve SORT_AREA_SIZE parametreleri).

- **Hedef–2**: Bellek içine veriyi koymak

İlk önce Oracle için yeteri kadar bellek ayrılmalıdır. Daha sonra odaklanması gerekilen nokta, çok önemli bilgilerin bellekte yer alması ve bunun güvence altına alınmasıdır.

- **Hedef–3**: Belleği veya Giriş/Çıkışı tıkayan SQL cümlelerini bulmak

En zor sorun, problemin nerede ve hangi zamanda oluştuğunu bulmaktır. V$SQLARIA'yı kullanarak, darboğazı kolayca tanımlayacak bir yöntem geliştirilmelidir.

- **Hedef - 4**: Problem SQL cümlelerinin optimizasyonu

Problem olan SQL cümlelerinin optimizasyonu başlıca bir eğitim gerektirmektedir. Bu yüzden iki bölüm üzerinde durulacaktır. Bunlar; a) Sistemin optimizasyonunu yapmadan önce bilinmesi gerekenler. b) Paralel Sorgu seçeneğini kullanarak genel bir optimizasyon yapılması.

### 3.4.1. Hedef – 1: Oracle İçin Yeterli Bellek Ayrılmış mı?

Eğer sistem 10 Giga Byte kullanıma uygun bellekte çalışıyor olsa bile, bu belleğin küçük bir parçası Oracle'a ayrılırsa bu durum bize yardımcı olmayacaktır. Oracle için bellek ayrımı "*initsid.ora*" dosyasında yapılır. Anahtar kelimelerin bazıları aşağıda listelenmiştir. Aşağıdaki bölümde bu parametrelerin hepsi anlatılmaktadır. "*V$PARAMETER*" dosyasına bakarak veya "*DBAExplorer*" programını kullanarak performansı etkileyen bu parametreleri görebiliriz.

"init.ora" parametre değerlerini öğrenmek için aşağıdaki SQL cümlesi çalıştırılır.

```
SELECT NAME, VALUE
  FROM V$PARAMETER
 WHERE NAME IN ('DB_BLOCK_BUFFERS', 'DB_BLOCK_SIZE',
'SHARED_POOL_SIZE', 'SORT_AREA_SIZE');
```

| NAME | VALUE |
|---|---|
| DB_BLOCK_BUFFERS | 4000 |
| DB_BLOCK_SİZE | 4096 |
| SHARED_POOL_SİZE | 7000000 |
| SORT_AREA_SİZE | 262144 |

**Veri Blok Tamponları (DB_BLOCK_BUFFERS)**

İlk olarak bakılacak "init.ora" parametresidir. Bu alan, bellekte depolama ve verinin işlemesi için kullanılan SGA alanıdır. Kullanıcılar bilgi istediğinde bilgi bellek içine koyulur. Eğer DB_BLOCK_BUFFERS parametresine çok küçük değer atanmış ise, en az kullanılan bilgi bellekten temizlenir. Eğer bellekten temizlenen bilgi bir sorgu ile tekrar çağırılırsa, diskten tekrar okuma yapılır. Bu durum MİB ve G/Ç kaynaklarının tüketilmesine neden olur. Eğer DB_BLOCK_BUFFERS parametresi çok küçük olursa, kullanıcılar çalışmak için yeterli belleğe sahip olamayacaklardır. Eğer DB_BLOCK_SIZE çok büyük olursa, sistem getir götür işlemleri yapmaktan belki duracaktır.

Veri Blok Tamponlarının yeteri kadar ayarlandığına karar vermek;

```
SELECT 1-(SUM(DECODE(NAME, 'PHYSICAL READS', VALUE,0))/
       (SUM(DECODE(NAME, 'DB BLOCK GETS', VALUE,0)) +
       (SUM(DECODE(NAME, 'CONSISTENT GETS', VALUE,0))))) * 100
       "READ HIT RATIO"
  FROM V$SYSSTAT;
```

```
READ HIT RATIO
---------------
98.415926
```

Değerlerin oranı %90-%95 olmasına rağmen, genellikle zayıf indisi işaret eder.

**SHARED_POOL_SIZE**

SHARED_POOL_SIZE, Paylaşılmış SQL ve PL/SQL deyimleri için kullanılacak alanın bayt olarak boyunu verir. *SHARED_POOL_SIZE* kullanan işlemler, prosedürler, paketler ve tetiklerdir. Bu kitaplık ve Veri Sözlüğü önbelleği için ayrılan bellektir. Eğer *SHARED_POL_SIZE* çok küçük ayarlanırsa o zaman *DB_BLOCK_BUFFERS*'ın avantajlarından tam yararlanılamaz.

Sözlük alanı kayıp oranına karar vermek;

```
SELECT SUM(GETS) "GETS", SUM(GETMISSES) "MISSES",
       (1 - (SUM(GETMİSSES) / (SUM(GETS)+
       SUM(GETMISSES))))*100 "HITRATE"
FROM V$ROWCACHE;
```

```
GETS       MISSES     HITRATE
------     -------    ----------
10233      508        95.270459
```

**Not:** Burada alınan sonuç iyi oran olur ve bu alana muhtemelen müdehale gerekmez.

Kitaplık önbellek vuruş oranına karar vermek;

```
SELECT SUM(PINS) EXECUTIONS, SUM(PINHITS) "EXECUTION HITS",
      ((SUM(PINHITS) / SUM(PINS)) * 100) PHITRAT,
      SUM(RELOADS) MISSES,
      ((SUM(PINS) / (SUM(PINS) + SUM(RELOADS))) * 100)  HITRAT
FROM V$LIBRARYCACHE;
```

| EXECUTIONS | EXECUTION HITS | PHITRAT | MISSES | HITRAT |
|---|---|---|---|---|
| 3,582 | 3,454 | 96.43 | 6 | 99.83 |

**Not**: Eğer vuruş oranı veya yeniden yükleme oranı yüksek olursa, "init.ora" dan *SHARED_POOL_SIZE* parametresi artırılır.

*SHARED_POOL_SIZE* için ne kadar bellek ayırmak gerekir;

```
COL VALUE FOR 999,999,999,999 HEADING "SHARED POOL SIZE"
COL BYTES FOR 999,999,999,999 HEADING "FREE BYTES"
SELECT TO_NUMBER(V$PARAMETER.VALUE) VALUE, V$SGASTAT.BYTES,
       (V$SGASTAT.BYTES/V$PARAMETER.VALUE)*100 "PERCENT FREE"
FROM V$SGASTAT, V$PARAMETER
WHERE V$SGASTAT.NAME = 'FREE MEMORY'
AND   V$PARAMETER.NAME = 'SHARED_POOL_SIZE;
```

| SHARED POOL | SIZE FREE BYTES | PERCENT FREE |
|---|---|---|
| 100,000,000 | 82,278,960 | 82.27896 |

**Not**: "ORA-4031" hatası, genellikle paylaşılan havuz (*SHARED_POOL*) boyundan dolayı oluşur. Bu hata, paylaşımsız havuzun parçalanmasından dolayı veya paylaşımsız havuzun yetersiz olmasından olabilir.

**Not**: Bellekten bilgiyi geri almak, diskten okumaktan daha hızlı olur. (sahip olunan belleğe göre yaklaşık 10.000 kat). Bu yüzden SGA'nın yeterince geniş olduğundan emin olmak gerekir.

**Geçici (Temporary) yerine bellekte sıralama;**

SORT_AREA_SIZE, "init.ora" parametresidir ve sıralama yapmak için bellekte ne kadar yer ayrılması gerektiğini belirtir. Bu alan ana bellekte her bir sıralama işlemi için ne kadar bellek ayrılması gerektiğini belirtir. Bu bellek, MTS veri tabanları için SGA ve NON-MTS (Multi-Thread Server) için PGA' nın UGA bölümünde bulunur. Eğer sıralama, bellekte gerçekleştirilemezse, geçici parçalar (Temporary Segment) diskte bilgileri tutar. SORT_ARES_SIZE değerini artırarak, disk sıralama (disk-sort) toplam sayısı azaltılabilir. Böylece disk okuma sayısı azalmış olur. Eğer diğer işlemler için az bellek bırakılırsa bu durum değiş/tokuşa neden olur. Geçici parçaları içeren durumlar: İndeks oluşturulması, Select, Order By, Distinct, Group By, Union, Unindexed Joins, Some Correlated Subqueries cümleleridir.

Geçici bölümlerin (segmentlerin) ilk değerleri en az SORT_AREA_SIZE kadar olmalıdır.
Birçok parametre oturum sahibi tarafından ayarlanabilir. Örneğin;

```
ALTER SESSION SET SORT_AREA_SIZE = 100000000;
```

Not: "init.ora" parametreleri değiştirilerek oturum değiştirilebilir. Bu hem geliştiriciler için hem de VT yöneticileri için iyi bir özelliktir. Sonuç olarak, eğer bu durum kısıtlanmaz ise, bir kullanıcı ALTER SESSION hakkı ile oturum bilgisini (SORT_AREA_SIZE) 100Mb' dan fazla yapabilir.

### 3.4.2. Hedef – 2: Bellek İçine Veriyi Koymak

İlk önce Oracle için yeterli bellek ayrılmalıdır. Çok önemli bilgilerin belleğe girdiğini, orada kaldığını ve garantili değiştirildiğine odaklanmalıdır. X$BH tablosunun ve "*alter table*" için "*cache*" parametresinin nasıl kullanıldığı araştırılacaktır.

**X$BH Tablosu**

Oracle'dan istenen verilerin, herhangi bir istenme durumunda daha hızlı getirilmesi için burada tutulur. Bunun için Oracle'ın X$ tabloları kullanılır.

```
SELECT STATE, COUNT(*)
FROM X$BH
GROUP BY STATE;
```

```
STATE      COUNT(*)
---------  ----------------
        0       371
        1       429
```

Yukarıdaki sonuca göre;

Toplam DB_BLOCK_BUFFERS = 800

Toplam kullanılan = 429

Toplam kullanılmayan = 371

Daha iyi bir SQL cümlesi şu şekilde olabilir;

```
SELECT     DECODE(STATE,0, 'FREE', 1,
           DECODE(LRBA_SEQ,0,'AVAILABLE','BEING USED'),
           3, 'BEING USED', STATE) "BLOCK STATUS", COUNT(*)
FROM X$BH
GROUP BY   DECODE(STATE,0,'FREE',1,DECODE(LRBA_SEQ,0,
           'AVAILABLE','BEING USED'),3, 'BEING USED', STATE);
```

```
BLOCK STATUS          COUNT(*)
------------          --------
AVAILABLE             779
BEING USED            154
FREE                  167
```

**Tablo Değiştirmede (Alter Table) "Cache" Parametresinin Kullanılması**

Eğer yeni tablo yaratılırken "*Cache*" parametresi kullanılırsa, "*Full Table Scan*" sonucu olarak "Most recently used" yerine "Least recently used" kullanılır.
Bu parametrenin kullanım şekilleri aşağıdaki örneklerde gösterilmiştir.

Bölüm 5.2.4.3'de VTYA kullanılarak tablo özelliklerinin nasıl değiştirileceği örneği gösterilmiştir.

**Örnek – 1:** Aşağıdaki örnekte bir tablo oluşturulurken "*Cache*" parametresi kullanılmıştır.

```
CREATE TABLE OGRENCI (COL1 NUMBER)
TABLESPACE USERS
CACHE;
```

**Örnek – 2:** Aşağıdaki örnekte bir tablo değiştirilirken "*Cache*" parametresi kullanılmıştır.

```
ALTER TABLE OGRENCI
CACHE;
```

**Örnek – 3:** Aşağıdaki örnekte "*Cache*" hint parametresi kullanılmıştır.

```
SELECT /*+ CACHE(OGRENCI) */ ADI, SOYADI
  FROM OGRENCI
 WHERE ADI = ALP';
```

**Örnek – 4:** Aşağıdaki örnekte "NOCACHE" hint parametresi kullanılmıştır.

```
SELECT /*+ FULL(OGRENCI)  NOCACHE(OGRENCI) */  ADI, SOYADI
  FROM OGRENCI
```

```
 WHERE ADI = ALP';
```

Bellek içerisinde PL/SQL nesne deyimlerini (Önbelleği) aynı zamanda ilişkilendirebilirsiniz.

Yeterli SHARED_POOL_SIZE değerini koruyamadığımız takdirde, bellekte çok önemli nesneleri (PINNED) tutmak önemli olabilir. Aşağıdaki örnek, DBMS_SHARED_POOL.KEEP işlemini kullanan bellekte PL/SQL nesne deyimlerinin nasıl ilişkilenmesi gerektiğini gösterir.

```
BEGIN
DBMS_SHARED_POOL.KEEP('PROCESS_DATE','P');
END;
```

**Not:**

Gün içerisinde yetersiz bellek hatalarından sakınmak için, VT başlamak üzereyken bellek içine PL/SQL nesneleri iliştirilir. PL/SQL nesne deyimleri için DBMS_SHARED_POOL-KEEP işlemleri kullanılır. Çok geniş olduğu için standart işlemlerin, VT başladıktan hemen sonra ilişkilendirildiğinden emin olunmalıdır.

Bütün paketler aynı zamanda ilişkilendirilebilir.

Sistemdeki bütün paketleri ilişkilendirmek için, aşağıdaki cümle çalıştırılır [11].

```
DECLARE
OWN VARCHAR2(100);
NAM VARCHAR2(100);
CURSOR PKGS IS
     SELECT     OWNER, OBJECT_NAME
     FROM       DBA_OBJECTS
     WHERE      OBJECT_TYPE = 'PACKAGE';
BEGIN
```

```
      OPEN PKGS;
      LOOP
      FETCH PKGS INTO OWN, NAM;
      EXIT WHEN PKGS%NOTFOUND;
      DBMS_SHARED_POOL.KEEP(OWN || '.' || NAM, 'P');
      END LOOP;
END;
```

Oracle ile yollanan yaygın problem paketi, "STANDART", "DBMS_STANDART" ve "*DIUTIL*"'i içerir.

VT başladığında bütün paketleri PL/SQL'de birleştiren DBMS_SHARED_POOL-KEEP işlemi kullanılır.

### 3.4.3. Hedef – 3: Belleği veya Giriş/Çıkışı Tıkayan SQL Cümlelerini Bulmak

Tek indis veya tek bir sorgu bütün sistemi durdurmaya yakın hale getirebilir. "*V$SQLAREA*"'yı kullanarak, sisteminizdeki problemli sorguları bulabilirsiniz. Aşağıdaki örnek, sorunlu SQL cümlesini nasıl bulmanız gerektiğini gösterir. Bu örnekte, disk okuması 10.000'den büyük olan sorgular aratılmaktadır. Eğer sisteminiz çok daha geniş ise, daha yüksek sayıda bu değeri ayarlamak gerekebilir.

Bölüm 5.2.4.9'da VTYA kullanılarak giriş/çıkışı tıkayan SQL cümlelerine nasıl bakılacağı gösterilmiştir.

Örnek – 1: Diskten fiziksel okuması en büyük olan SQL cümlesini sorgular;

```
SELECT      DISK_READS, SQL_TEXT
FROM        V$SQLAREA
WHERE       DISK_READS > 10000
ORDER       BY DISK_READS DESC;

DISK_READS      SQL_TEXT
----------      ----------------------------------------
```

```
12987          SELECT ORDER#,COLUMNS,TYPES FROM ORDERS
               WHERE SUBSTR(ORDERID,1,2)=:1
11131          SELECT CUSTID, CITY FROM CUSTOMER
               WHERE CITY = 'CHICAGO'
```

Örnek – 2: Diskten mantıksal okuması en büyük olan SQL Cümlesini sorgular;

```
SELECT      BUFFER_GETS, SQL_TEXT
FROM        V$SQLAREA
WHERE       BUFFER_GETS > 200000
ORDER BY    BUFFER_GETS DESC;
```

```
BUFFER_GETS     SQL_TEXT
-----------     ----------------------------------------
300219          SELECT ORDER#,CUST_NO, FROM ORDERS
                WHERE DIVISION = '1'
```

V$SQLTEXT görüntüsü ile bağlama:

V$SQLAREA, SQL metninin belli bir miktarını gösterdiği için, bu metnin tamamını V$SQLTEXT tablosuna bağlantı kurarak alabilirsiniz.

```
SELECT A.USER_NAME, B.DISK_READS, B.BUFFER_GETS,
       B.ROWS_PROCESSED, C.SQL_TEXT
FROM V$OPEN_CURSOR A, V$SQLAREA B, V$SQLTEXT C
WHERE A.USER_NAME = UPPER('&&USER')
  AND A.ADDRESS = C.ADDRESS
  AND A.ADDRESS = B.ADDRESS
ORDER BY   A.USER_NAME, A.ADDRESS, C.PIECE
```

```
USER NAME  DISK READS BUFFER GETS     ROWS PROCESSED  SQL_TEXT
---------  ---------- ------------    --------------  --------
ALP        2          2300            210
SELECT ITEMNO, CUSTNO FROM ITEMS WHERE CUSTNO = 'A101'
SCOTT      3          23213           7015
SELECT ITEMNO, CUSTNO FROM ITEMS WHERE STATE = 'IL'
```

```
USER       0          200              2
SELECT ITEMNO, CUSTNO FROM ITEMS WHERE ORDERNO = 131313
DENEME     32000      45541            7100
SELECT ITEMNO, CUSTNO FROM ITEMS WHERE NVL(ORDERNO,0) = 131313
OR NVL(ORDERNO,0) = 777777
```

### 3.4.4. Hedef – 4: Problem SQL Cümlelerinin Optimizasyonu

**A. Sistem performansını yapmadan önce bilinmesi gerekenler:**

İlk düşünülmesi gereken veri için neye ihtiyaç olduğudur. Verinin hacmi ve verinin dağılımı, kişisel sorguların nasıl en iyi şekle sokulacağını etkileyecektir. Denenecek çeşitli metotlara sahip olma ihtiyacı duyulur. Sorgu tiplerini kapsayacak birçok yaklaşımda bulunulmalıdır. Sadece bir metot veya bir optimizasyon yöntemi yeterli olmayacaktır. Sistemin nerede yavaşladığını bilmeye de ihtiyaç duyulacaktır. Çoğu VT yöneticisi veya geliştiriciler, sistemi kullananlara sormak yerine, problemi bulmak için çok fazla saat harcamaktadırlar. Bunun yerine kullanıcılara sormaları gerekmektedir, çünkü kullanıcılar bu bilgiyi neredeyse her zaman gönüllü vermekten mutlu olacaklardır. Aynı sistemde çalışan diğer geliştiriciler ile ağa aynı zamanda ihtiyaç duyulabileceği de unutulmamalıdır.

**B. Optimizasyonda "Key" işaretini kullanmak:**

Oracle'ın optimizasyonu kusursuz değildir, en iyi optimizasyon için kullanılabilecek ipuçları (*hint*) vardır. Kullanıcıya özel ayarlamaları gerektiren durumlar olacaktır. Bir sorgu bulunduğunda, Oracle'ın kişisel sorguları ayarlamak için sunduğu ipuçlarının avantajlarından yararlanılmalıdır. İpuçları için söz dizim kuralları aşağıda listelenmiştir. Akılda tutulması gereken önemli bir nokta, söz dizimi kuralları doğru veya yanlış olabilir, fakat bunun için hiçbir hata mesajı verilmez. Aynı zamanda ipuçları deyimlerinin sadece burada uygulandığını hatırlamak gerekir. Farklı

durumlarda, tamamen farklı deyimler için kendi ipuçlarınız ile işlem yapmak gerekir. Sorgu optimizasyon için en uygun SQL yardım cümlecikleri aşağıda verilmiştir.

**FULL ( Full Table Scan)**

Belirtilen tablo üzerinde tam tarama yapılmasını sağlar.

```
SELECT /*+ FULL(TABLE_NAME) */ COLUMN1, COLUMN2 ...
```

**INDEX (Index Search)**

Belirtilen tablo için indisi üzerinden tam tarama yapılmasını sağlar. Eğer indis adı verilmemişse, en iyileştirici tüm indisleri kontrol eder ve en uygun olanı ya da olanları plana dahil eder. Eğer doğrudan isimleri verilmiş ise, iyileştirici bu indisler içinde uygun olan ya da olanları seçer. Sadece tek bir indis verilmişse iyileştirici bu indisi kullanır.

```
SELECT /*+INDEX(TABLE_NAME INDEX_NAME1 INDEX_NAME2…)*/
COLUMN1,COLUMN2…
```

**ORDERED**

En iyileştirici "*FROM*" kalıbındaki tabloların belirtilen sırada (soldan sağa) birleştirilmesini sağlar. Raporlama ortamlarında bu yardım cümleciği çok yardımcı olabilir. Ne kadar çok tablo belirtilmişse bu yardım cümleciğinin faydası da o kadar artar.

```
SELECT /*+ ORDERED */ COLUMN1, COLUMN2...
FROM TABLE1, TABLE2
```

**ALL_ROWS**

En iyi işlem hacmi miktarı hedefi ile temel maliyet yaklaşımını seçer. Çalıştırılan sorguda tüm kayıtları getirmek için en az kaynak tüketimini sağlar.

```
SELECT /*+ ALL_ROWS */ COLUMN1, COLUMN2 ...
```

**FIRST_ROWS , FIRST_ROWS(N)**

Çağırılan sorgunun ilk n kaydını en kısa sürede döndürülecek şekilde optimizasyon sağlar. SQL cümlesindeki herhangi bir tablo için istatistik bilgisinin olması şart değildir, iyileştirici tarafından bu istatistikler hesaplanır.

```
SELECT /*+ FIRST_ROWS */ COLUMN1, COLUMN2 ...
SELECT /*+ FIRST_ROWS(100) */
```

*DELETE* ve *UPDATE* cümleleri, gruplama işlemi *(UNION, INTERSECT, MINUS, GROUP BY, DISTINCT, MAX, MIN, SUM* gibi) ya da *FOR UPDATE* kalıbı içeren sorgularda iyileştirici *FIRST_ROWS* yardım cümleciğini işleme almaz. Çünkü bu işlemlerde mutlaka tüm kayıtlara erişilmesi gerekeceğinden en hızlı cevap optimizasyonu yapılamaz.

**SQL Yardım Cümlecikleri (Hint) Örnekleri:**

İyileştirici İNDİS kullanırsa;

```
SELECT BLOCKS
  FROM CUST
 WHERE TABLE_NAME = 'EMP';
EXECUTION TIME - 1.1 SECONDS
```

CUST tablosunda “Full Table Scan” zorlamak için "FULL" yardımcı cümleciği kullanılır. Performansın şimdi düştüğüne dikkat edilmelidir. (Yardımcı cümlecikler her zaman iyi sonuç vermeyebilir.)

“Full Table Scan” kullanılması: Kötü Seçim;

```
SELECT /*+ FULL(CUST) */  BLOCKS
  FROM CUST
 WHERE TABLE_NAME = 'EMP';
```

*EXECUTION TIME – 75.1 SECONDS*

Bir SQL yan cümleciği yazarken hata yaparsak, bize hata/uyarı mesajı dönmez ve ihmal edilir. Aşağıdaki sorgu örneğinde , "+" işaretini (Önceki örnekte gösterildiği gibi, bu durumda ise bilerek) unutarak “FULL” yardımcı cümleciği yanlış belirtildi.

Geçersiz yardımcı cümle kullanımı; (+) işareti kullanılmamıştır,

```
SELECT /*FULL(CUST)*/ BLOCKS
  FROM CUST
 WHERE TABLE_NAME = 'EMP';
```

*EXECUTION TIME – 1.1 SECONDS*

### 3.5. SQL Optimizasyon Örnekleri

#### A. Sorgularda “OR” Kullanmak

SQL cümlelerinde çoklu “WHERE” koşulu kullanıldığında “OR” cümleciklerine dikkat etmek gerekir. Önceki sürümlerde, her bir SQL cümlesinde indisli bir kolon olması gerekirken, Oracle'ın sonraki sürümlerinde (V8+) indislerin birleşmesi performans için risklidir. Eğer çok sınırlama yok ise indisler ile deney yapmak gerekir.

İndis verilmiş alanlar: *EMPNO, ENAME ve DEPTNO*

```
SELECT ENAME, DEPTNO, CITY, DIVISION
  FROM EMP1
 WHERE EMPNO = 1
   OR ENAME = 'LONEY'
   OR DEPTNO = 10;
```

*EXECUTION TIME: 4400 SECONDS*

*EXECUTION PLAN:*

```
      TABLE ACCESS EMP1 FULL
```

**Çözüm**:

```
SELECT /*+ INDEX(EMP EMP11) */
        ENAME,  DEPTNO, CITY, DIVISION
 FROM EMP1
WHERE EMPNO = 1
   OR ENAME = 'LONEY'
   OR DEPTNO = 10;
```

*EXECUTION TIME: 280 SECONDS*

*EXECUTION PLAN:*

```
      TABLE ACCESS EMP1 ROWID
      TABLE ACCESS EMP11 INDEX RS
```

### B. Eşitsizlik Arasında Dağılım

Cost-Based en iyileştiricinin kapsamlı ve karışık bir çalışma prensibi bulunmaktadır. Kullanılacak olan en iyi yöntemi belirlenirken, çeşitli VT bilgileri (tablo boyutları, kayıt sayıları, verilerin dağılımı gibi) kullanılmaktadır.

Aşağıdaki verilere göre örnek incelendiğinde;
ORDER_LINE tablosu 10.000 satıra sahip ve kayıtlar 1 den 10.000 kadardır.
5000 den fazla kayıt (yaklaşık tablonun yarısı) ITEM numarası >9990'dan büyüktür.
ITEM_NO alanı indisli bir alandır.

Aşağıdaki örnekte iyileştirici indis kullanmayı seçer.

```
SELECT SIZE, ITEM_NO
  FROM ORDER_LINE
 WHERE ITEM_NO > 9990;
```

*EXECUTION TIME: 530 SECONDS*

Sorgu, verinin 50%'sini geri getirdiği için indis bastırılır.

Daha iyi bir sorgu aşağıdaki gibi olmalıdır.

```
SELECT /*+ FULL(ORDER_LINE) */  SIZE, ITEM_NO
 FROM ORDER_LINE
WHERE  ITEM_NO > 9990;
```

*EXECUTION TIME: 5 SECONDS*

### C. İç İçe Kullanılan Alt Sorgular

Birbirine bağlanmış tablo sorguları yerine iç içe kullanılan alt sorgular bazen daha iyi sonuç verebilir. Sadece belirli sorgularda, bu değişikliği yapmak için bazı kıstaslarla karşılaşılır. Bir hak bulunduğunda, bu tür hileler, daha yüksek performans sağlayacaktır. İçteki (Inner) ve dıştaki (Outer) tablo önemlidir. İçteki ve dıştaki tabloya iyileştiricinin (CBO) kendisi karar verir. Bu seçim yapıldıktan sonra seçilen "Outer" tablo "Driving Table" olarak adlandırılır.
Örnek olarak aşağıda gösterilen durumlar oluştuğu zaman sorgu değiştirilebilir.

Tablolar birbirine bağlı fakat sadece bir tablonun verisi dönüyor ise;

Sorgunun ilk hali;

```
SELECT A.COL1, A.COL2
  FROM TABLE1 A, TABLE2 B
```

```
 WHERE A.COL3 = VAR
   AND A.COL4 = B.COL1
   AND B.COL2 = VAR;
```

Yeni yazılan sorgu;

```
SELECT A.COL1, A.COL2
  FROM TABLE1 A
 WHERE A.COL3 = VAR
   AND EXISTS
       (SELECT 'X'
          FROM TABLE B
         WHERE A.COL4 = B.COL1
           AND B.COL2 = VAR);
```

Gerçek bir hayat örneği;

```
SELECT ORDER.ORDNO, ORDER.CUSTNO
  FROM ORDER_LINE OL, ORDER
 WHERE ORDER.ORDNO = OL.ORDNO
   AND ORDER.CUSTNO = 5
   AND OL.PRICE = 200;
```

*EXECUTION TIME: 240 MINUTES*

Çözüm;

```
SELECT ORDNO, CUSTNO
  FROM ORDER
 WHERE CUSTNO = 5
   AND EXISTS
      (SELECT 'X'
         FROM ORDER_LINE OL
        WHERE ORDER.ORDNO = OL.ORDNO
          AND OL.PRICE = 200);
```

*EXECUTION TIME: 9 SECONDS*

**D. Nested Loop Join**

İç içe olan döngüler genel olarak aşağıdaki gibi gösterilir:

```
FOR R1 IN (SELECT ROWS FROM TABLE_1 WHERE COLX = {VALUE}) LOOP
   FOR R2 IN (SELECT ROWS FROM TABLE_2 THAT MATCH CURRENT ROW FROM TABLE_1)
LOOP
        OUTPUT VALUES FROM CURRENT ROW OF TABLE_1 AND CURRENT ROW OF TABLE_2
   END LOOP
END LOOP
```

Dıştaki tablodaki her bir satır için, içteki tablodaki tüm satırlara erişim söz konusudur. Çalışma Planında (Execution Plan) bu sıra aşağıdaki gibi oluşur.

1. İç İçe Döngüler (Nested Loops)
2. Dıştaki Döngü (Outer Loop)
3. İçteki Döngü (Inner Loop)

Özellikle içteki döngüde ulaşım (Access Path) önemlidir. Çünkü dışarıdaki döngünün her satırı için gerçekleşmektedir. Örneğin her seferinde bir Full Table Scan işlemi yapılırsa büyük tablolar için performans sıkıntısına yol açacaktır.

Bu işlemi daha iyi anlatabilmek için aşağıda örnek verilmiştir.

Adım–1: İki tablo oluşturulacak ve bu tablolar analiz edilecek.

```
CREATE TABLE TABLO1 NOLOGGING AS
      SELECT OBJECT_ID, OBJECT_NAME, OBJECT_TYPE
      FROM USER_OBJECTS;
```

```
CREATE TABLE TABLO2 NOLOGGING AS
      SELECT OBJECT_ID, OBJECT_NAME, OBJECT_TYPE
      FROM USER_OBJECTS;
```

```
ANALYZE TABLE TABLO1 COMPUTE STATISTICS;
```

```
ANALYZE TABLE TABLO2 COMPUTE STATISTICS;
```

Adım–2: Yazılacak sorguya SQL yardım cümleciği "Hint use_nl(tableX)" parametresi koyularak en iyileştiriciye müdahale edilecek ve iç içe döngü oluşması sağlanacaktır.

```
SELECT /*+ USE_NL(T1 T2)*/  *
  FROM TABLO1, T2
 WHERE T1.OBJECT_ID = 1234
   AND T1.OBJECT_NAME = T2.OBJECT_NAME;
```

```
ELAPSED: 00:00:00.00
```

```
EXECUTION PLAN
| ID  | OPERATION           | NAME | ROWS  | BYTES | COST (%CPU)|
|   0 | SELECT STATEMENT    |      |     1 |    56 |    37   (3)|
|   1 |  NESTED LOOPS       |      |     1 |    56 |    37   (3)|
|*  2 |   TABLE ACCESS FULL |TABLO1|     1 |    28 |    19   (6)|
|*  3 |   TABLE ACCESS FULL |TABLO2|     1 |    28 |    18   (0)|
```

Yukarıdaki sorgu iç içe döngü olacak şekilde zorlandı. TABLO1 driving table olarak seçildi ve sonuçta "Cost 37" olarak hesaplandı. Dikkat edilirse TABLO1 ve TABLO2 için Access Path ve Full Table Scan durumu oluştu.

Adım–3: Bu durumu bertaraf etmek için TABLO2 tablosunda "OBJECT_NAME" alanı üzerinde bir indis oluşturulması gerekmektedir. Oluşturulan indis analiz edilerek iyileştirici için bilgi oluşturulacaktır.

```
CREATE INDEX T2_UNQ_IDX ON TABLO2(OBJECT_NAME);
```

```
ANALYZE INDEX T2_UNQ_IDX COMPUTE STATISTICS;
```

Adım - 4: Sorgu tekrar çalıştırılarak çalışma planı tablosuna bakılır.

```
SELECT /*+ USE_NL(T1 T2)*/  *
  FROM TABLO1, T2
 WHERE T1.OBJECT_ID = 1234
   AND T1.OBJECT_NAME = T2.OBJECT_NAME;
```

```
ELAPSED: 00:00:00.06
EXECUTION PLAN
| ID|OPERATION                    | NAME | ROWS | BYTES | COST (%CPU)|
|  0|SELECT STATEMENT             |      |    1 |    56 |    21   (5)|
|  1|TABLE ACCESS BY INDEX ROWID|TABLO2|    1 |    28 |    2    (0)|
|  2|NESTED LOOPS                 |      |    1 |    56 |    21   (5)|
|*3 |TABLE ACCESS FULL            |TABLO1|    1 |    28 |    19   (6)|
|*4 |INDEX RANGE SCAN             |T2_UNQ_IDX |1|       |     1   (0)|
```

Yukarıdaki sorgu sonucuna göre "Cost 21"olarak hesaplandı. Bu düşüşün en önemli sebebi içteki tabloda Full Table Scan yerine oluşturduğumuz indis üzerinden gidilmesi, Access Path olarak "INDEX RANGE SCAN" kullanılması oldu. Nested Loop Join özellikle az veri çekecek sorgularda ve içteki tablodaki veriye ulaşım maliyetinin düşük olduğu durumlarda iyi sonuç vermektedir.

### E. Hash Join

Birbirine bağlanan tablolardan küçük olanın verilerinin Hash mantığı ile belleğe çekilmesi, diğer tablonun sorgulanıp bellekte oluşturulan "Hash Table" ile karşılaştırılması esasına dayanır. Çekilecek veri büyük ise genelde uygun olan yöntemdir."*USE_HASH*" Hint'i ile iyileştiricinin "Hash Join" kullanması zorlanabilir.

Adım–1: İki tablo oluşturulacak ve bu tablolar analiz edilecek.

```
CREATE TABLE TABLO1 NOLOGGING AS
      SELECT OBJECT_ID, OBJECT_NAME, OBJECT_TYPE
      FROM USER_OBJECTS;
CREATE TABLE TABLO2 NOLOGGING AS
      SELECT OBJECT_ID, OBJECT_NAME, OBJECT_TYPE
      FROM USER_OBJECTS;
ANALYZE TABLE TABLO1 COMPUTE STATISTICS;
ANALYZE TABLE TABLO2 COMPUTE STATISTICS;
```

Adım-2: Aşağıdaki sorgu çalıştırıldığında iyileştirici direk olarak Hash Join kullanmayı tercih etti. Buradaki "Cost 38" olarak hesaplandı.

```
SELECT *
  FROM TABLO1, T2
 WHERE T1.OBJECT_ID = 1234
   AND T1.OBJECT_NAME = T2.OBJECT_NAME;
```

```
ELAPSED: 00:00:00.10
```

```
EXECUTION PLAN
| ID  | OPERATION           | NAME | ROWS  | BYTES | COST (%CPU)|
|  0  | SELECT STATEMENT    |      |  1192 | 66752 |    38   (6)|
|* 1  |  HASH JOIN          |      |  1192 | 66752 |    38   (6)|
|* 2  |   TABLE ACCESS FULL |TABLO1|   897 | 25116 |    19   (6)|
|  3  |   TABLE ACCESS FULL |TABLO2| 12478 |   341K|    18   (0)|
```

## F. Merge Join

Her bir tabloda koşula uygun verilerin sıralanması daha sonra da bu sıralı iki yapının birleştirilmesi esasına dayanır. Karşılaştırılacak veriler zaten sıralı ise "<, >, <=, >= " şeklindeki karşılaştırmalarda uygundur. Önce bir sıralama (sort) ardından da sıralı verinin "Merge" edilmesi söz konusudur."USE_MERGE" Hint'i ile en iyileştirici Merge Join yapmaya zorlanabilir.

Adım -1: İki tablo oluşturulacak ve bu tablolar analiz edilecek.

```
CREATE TABLE TABLO1 NOLOGGING AS
      SELECT OBJECT_ID, OBJECT_NAME, OBJECT_TYPE
      FROM USER_OBJECTS;
```

```
CREATE TABLE TABLO2 NOLOGGING AS
     SELECT OBJECT_ID, OBJECT_NAME, OBJECT_TYPE
      FROM USER_OBJECTS;
```

```
ANALYZE TABLE TABLO1 COMPUTE STATISTICS;
```

```
ANALYZE TABLE TABLO2 COMPUTE STATISTICS;
```

Adım–2: Yazılacak sorguya SQL yardım cümleciği “Hint use_merge(TABLO1 TABLO2)” parametresi koyularak iyileştiriciye müdahale edilecek ve iç içe döngü oluşması sağlanacaktır.

Buradaki “Cost 140” olarak hesaplandı.

```
SELECT /*+ use_merge(TABLO1 TABLO2) */ *
  FROM TABLO1, T2
 WHERE T1.OBJECT_ID = 1234
   AND T1.OBJECT_NAME = T2.OBJECT_NAME;
```

```
ELAPSED: 00:00:00.04
```

```
EXECUTION PLAN
| ID | OPERATION           | NAME |ROWS |BYTES | TEMPSPC|COST
(%CPU)|
|  0 | SELECT STATEMENT    |      |   1 |   56 |        |  140  (4) |
|  1 |  MERGE JOIN         |      |   1 |   56 |        |  140  (4) |
|  2 |   SORT JOIN         |      |   1 |   28 |        |   20  (10)|
|* 3 |    TABLE ACCESS FULL|TABLO1|   1 |   28 |        |   19   (6)|
|* 4 |   SORT JOIN         |      |12478| 341K | 1000K |  120   (3)|
|  5 |    TABLE ACCESS FULL|TABLO2|12478| 341K |        |   18   (0)|
```

# BÖLÜM 4

# 4. VERİ TABANI YÖNETİM ARAÇLARI

## 4.1. Genel Özellikler

SQL programlama ve geliştirme araçları, VT geliştiricilerine ve yöneticilerine, doğru ve güvenilir platform sağlayan yazılımlardır.

VT geliştiricileri ve yöneticileri sürekli olarak değişik olumsuz koşullarla yüz yüze gelirler ve bu tür durumların nadiren tamamını kontrolleri altına alabilirler. Bu olumsuz koşullar genel olarak yeni geliştirilen projelerin veya değiştirilen kodların sonucunda oluşur. Bu projeler gerçek ortama geçirildiğinde, her zaman beklenen sonucu vermeyebilir.

Bir SQL programlama ve yönetim aracında olması gereken başlıca özellikler aşağıda listelenmiştir.

### A. SQL Editör (SQL Editor)

SQL editörleri kullanıcılara SQL değiştirme ve çalıştırma olanağı sunmalıdır. Desteklenmesi gereken özellikler aşağıda belirtilmiştir:

- Kes, kopyala, yapıştır, geri al, ileri al, bul, değiştir gibi
- Blok cümle kaydırma, yazıcıdan çıktı alma, kaydetme ve büyük/küçük harf çevrimini sağlama
- Renkli kelime gösterme
- Otomatik tamamlama
- SQL sonucunu gösterme
- Commit ve Rollback toplu işlemlerini sağlamalıdır

**B. Nesne Gezgini (Object Browser)**

Geliştiriciler veya VT yöneticileri için bütün nesne bilgilerini göstermelidir. Desteklenmesi gereken özellikler aşağıda belirtilmiştir:

- Nesne ve nesne bilgilerinin niteliklerini gösterme
- Yeni nesne oluşturma
- Tetikleyicileri ve kısıtları aktif pasif yapabilme
- Geçersiz nesneleri derlenme
- Tablo verilerini değiştirme ve izlenme
- Nesnelerin izini ağaç yapısında gösterme (örneğin tablo, görüntü gibi referansları) gibi özellikleri desteklemelidir

**C. Oturum Gezgini (Session Browser)**

VT yöneticileri ve uygulama geliştiriciler, aktif oturumları ve bu oturumların geçerli aktivitelerini izleyebilmelidir. Kullanıcıların istatistiksel bilgilerini ve kaynak kullanımını kontrol edebilmeli, her oturumun için çalışan komutların kilidini çözebilmelidir.

**D. Kullanıcı Güvenlik Yönetimi (User Security Management)**

VT yöneticileri, kullanıcılar ile ilgili olarak, yaratma, silme, düzeltme, aktif veya pasif hale getirme işlemlerini yapabilmelidir. VT yöneticisi kullanıcılara, rol, sistem ayrıcalıkları, nesne ayrıcalıkları ve depolama kotası verebilmelidir.

**E. Hata Ayıklayıcı (Debugging)**

Paket, prosedür ve fonksiyonları derlerken hata ayıklayabilmelidir. Hata ayıklama esnasında; ileri veya geri gitme, hataya veya imlece kadar gitme, değişken ve yığın bilgisini gösterme gibi işlemleri yapabilmelidir. Herhangi bir programı değiştirmeden hata ayıklaması yapabilmelidir.

### F. Performans İzleme (Performance Monitoring)

VT kaynaklarının kullanım özetleri kolay anlaşılır grafiklerle gösterilebilmelidir. Aynı zamanda oturum faaliyetlerini, zaman özetini, son yapılan aktiviteleri, en çok işlem yapan oturum ve SQL cümlelerini gösterebilmelidir.

### G. Dışarıya Veri Aktarma/Dışarıdan Veri Alma İşlemleri (Data Import / Export)

Çeşitli veri türlerinden (text, xml, db ve excel gibi), belirlenen formatta veri alabilmeli ve aynı şekilde istenilen veri türünde veri çıkabilmelidir. Bu işlemleri kullanıcı ara yüzleri ile desteklemeli ve kullanıcıdan doğabilecek hataları engellemelidir.

### H. Çoklu Dil Desteği (Multi Languages)

Kullanılan program bütün kullanıcıları kapsamalı ve dil bağımsız olmalıdır. Program içerisinde standart diller bulunmalı yeni dillerin eklenmesini desteklemelidir.

## 4.2. Karşılaştırılan Programlar

Piyasada en yaygın olarak kullanılan VTYA araştırılmış ve belirlenen kıstaslara göre karşılaştırılmıştır. Bu bölümde fayda/maliyet analizi yapılmış ve en uygun aracın seçilmesi sağlanmıştır [3].

Aşağıdaki VTYA için oluşturulan puanlama tablolarında 1'den 5'e kadar değer kullanılmıştır. 5 en yüksek değer 1 ise en küçük değerdir.

### 4.2.1. Toad 8.6 (Quest Software)

Toad (www.toadsoft.com) kullanılan diğer VT araçları arasında geçmişi daha çok olan bir araçtır. Geliştirilmesi kişisel programcılıkla 1995 yılında başlamıştır. 1998 yılında Quest yazılım ürün geliştirmeye devam etmiş ve ilk resmi sürümü Toad 5, Ekim 1998 yılında çıkmıştır. Yaklaşık olarak her yıl iki ürün çıkartılmıştır. Temmuz 2005'te çıkan Toad 8.5 sürümü, JIT Debugging, CITRIX desteği, RAC ve gelişmiş 10g desteği sağlamaktadır.

Bu araç, geniş kullanıcı kitlesi ve popülerliğiyle, uygulama geliştiricilere yöneliktir. 1995 yılında geliştirilmeye başlanan bu ürün Oracle Geliştirme Araçlarına bir öncülük yapmıştır. Bugün pazarın büyümesi ve değişmesiyle beraber Toad, işlevsellik ve kalite açısından büyümekte ve devamını sürdürmektedir.

Toad, birden fazla VT için geliştirici ve yönetici aracıdır aynı zamanda çeşitli VT teknolojilerini desteklemekte ve buna göre şekil almaktadır. Desteklenen veri tabanları: Oracle, SQL Server, MySQL, ve DB2'dir.

Bütün platformları destekleyen tutarlı arabirimi ile DBA ve geliştiricilere sağlanan uygulama platformuna sahip olmanın pratikleri ile VT teknolojilerini harmanlamaya çalışan organizasyonun büyümesi ile dikkat çekmektedir.

TOAD'ın desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. Toad aynı zamanda 256 MB Hafıza ve 44 MB boş diske gereksinim duymaktadır.
Çizelge 4.1'de Toad 8.5 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.1.** Toad 8.5 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 4 |
| Geliştirici Faydası | 5 |

| | |
|---|---|
| İşlevsellik Ölçüsü | 5 |
| Kullanıcı Arayüzü | 5 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 4 |
| Doküman Kalitesi ve Kapsamı | 4 |
| Kullanılabilir Teknik Destek | 4 |
| Fiyat Performans Değeri | 4 |
| Diğer Araç/Sistem ile Entegrasyon | 5 |
| Genel Beğenilme | 4 |

### 4.2.2. KeepTool 7.2 (Tool for Oracle Database)

1996 Yılında Alman yazılım firması olan PSI'da 3 programcı, Oracle'da SQL yazmak için Delphi [45] arabirimini kullanmaya karar verdi. 1997 Yılında KeepTool (www.keeptool.com) GbR, daha sonra 2000 yılında KeepTool Gmbh geliştirildi. KeepTool, rakiplerini geride bırakmak ve yeni Oracle özelliklerini desteklemek için temel özelliklerini her yıl baştan gözden geçirmektedir.

KeepTool aracı, kullanıcı dostu ve güzel mimarisi, az kaynak harcayan yapısı ile, VT modelcileri, kod geliştiriciler ve VT yöneticileri için geliştirilmiştir.

2002 Yılında çıkan KeepTool 5.1 sürümü şu anki 7.2 sürümünden daha iyidir. Ürün başlıca gözden geçirme, kodlama, raporlama, izleme, koruma ve VT performansı içerir. Lisansa bağlı olarak ER diyagram ve PL/SQL hata ayıklayıcısı vardır.

KeepTool, VT'nından bağımsız olarak çeşitli dillerde kodlama imkânı da sunar. Kullanıcı şeması, seçilen nesneler veya bütün şema için HTML dokümantasyon üretir.

KeepTool'un mühendis takımında Oracle'a aşina benzer görevler alan uzman ve çıraklar görünür. KeepTool yönetilebilir basit bir kullanıma sahiptir. Aynı zamanda

araç çubuğu dinamik olarak değiştirilebilir, konuya uygun olarak seçilip ayarlanabilir.

KeepTool'un desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. KeepTool aynı zamanda 256 MB Hafıza ve 10 MB boş diske gereksinim duymaktadır.
Çizelge 4.2'de KeepTool 7.2 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.2.** KeepTool 7.2 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 5 |
| Geliştirici Faydası | 5 |
| İşlevsellik Ölçüsü | 5 |
| Kullanıcı Arayüzü | 5 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 4 |
| Doküman Kalitesi ve Kapsamı | 5 |
| Kullanılabilir Teknik Destek | 5 |
| Fiyat Performans Değeri | 5 |
| Diğer Araç/Sistem ile Entegrasyon | 5 |
| Genel Beğenilme | 5 |

### 4.2.3. SQLInsigth 3.0 (Isidian Technologies, Inc.)

Isidian Tech (www.isidian.com), 2000 yılında Oracle danışmanlığı ve üst düzey VT geliştiricileri için eğitim şirketi olarak başlamıştır. SQLInsigth Isidian Technologies şirketinin en önemli ürünüdür. Kendi müşterilerine Çalıştırma Planı ve performans ayarlamaları hakkında eğitim vermektedir. 2001 yılında ticari bir araç yapılmasına karar verilmiş ve SQLInsigth'ı geliştirmişlerdir. SQLInsigth geliştirilmeye devam edilmektedir. İleri ve acemi bütün geliştiricilere yönelik bir uygulamayı hedeflemiştir.

SQLInsigth, güzel kod yazımını ve SQL performans optimizasyonu hedeflemiştir. Bu editör, yaşamı daha kolay yapmak için tasarlanan özellikler açısından birinci sınıftır. Çalıştırma planı kullanımı ve kalitesi açısından şimdiye kadar geliştirilen en güzel araçlardandır. Her ne kadar otomatik ayarlama sihirbazına sahip olsa da en güzel ve zeki sürümü SQLInsigth 3.0 da gelmiştir. Bu akıllı sihirbaz, her hangi bir SQL cümlesinin, seçeneklere göre bütün performansını izleyebilir ve sonucu grafikler ile gösterebilir.

Kişisel olarak SQL performansına çok zaman harcanır fakat bu işlem SQLInsigth ile bu kadar zaman almayacağı görülmüştür.

SQLInsigth'ın desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. SQLInsigth aynı zamanda 64 MB Hafıza ve 30 MB boş diske gereksinim duymaktadır.
Çizelge 4.3'de SQLInsigth 3.0 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.3.** SQLInsigth 3.0 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 4 |
| Geliştirici Faydası | 5 |
| İşlevsellik Ölçüsü | 4 |
| Kullanıcı Arayüzü | 5 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 5 |
| Doküman Kalitesi ve Kapsamı | 5 |
| Kullanılabilir Teknik Destek | 4 |
| Fiyat Performans Değeri | 5 |
| Diğer Araç/Sistem ile Entegrasyon | 4 |
| Genel Beğenilme | 4 |

### 4.2.4. DBAConnect (DataSparc, Inc.)

DBAConnect, DataSparc (www.datasparc.com) firması tarafından Haziran 2002'de 1.0 sürümü çıkartıldı. İlk olarak "SQL Browser" adı ile SQL aracı olarak çıkarıldı, daha sonra bazı özellikler ve yenilikler katılarak adı DBAConnect olarak değiştirildi. DBAConnect, herhangi bir web gezgini ile Oracle VT'nın yönetimini sağlayan web tabanlı bir araçtır. DBAConnect kullanıcı oturumlarını kontrol eden ve yönetici komutlarını çalıştırmaya izin veren bir araçtır. DBAConnect, SQL cümlelerini geliştiricilerin çalıştırmasına olanak sağlar. DBAConnect hızlı performans ve sistem güvenliğini korumak için tasarlanmıştır.

DBAConnect kurulduğunda ve gerekli ayarlar yapıldığında, herhangi bir yerden uygun erişim ile Oracle VT'na bağlanabilir. Web tabanlı olduğundan, değişimler ve bakımı kolaylıkla yapılabilir. Müşteri lisans bakımı olduğu sürece, VT yöneticileri için sistem konusu, artırılmış sürüm desteği verilmektedir.

Diğer VT araçlarından farkı, platform bağımsız olması ve gelişmiş güvenlik özelliklerinin olmasıdır.

DBAConnect'in platform bağımsız bütün işletim sistemlerini desteklemektedir. DBAConnect aynı zamanda 128 MB Hafıza ve 30 MB boş diske gereksinim duymaktadır.
Çizelge 4.4'de DBAConnect 1.0 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.4.** DBAConnect 1.0 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 4 |
| Geliştirici Faydası | 3 |
| İşlevsellik Ölçüsü | 4 |
| Kullanıcı Arayüzü | 4 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 4 |
| Doküman Kalitesi ve Kapsamı | 3 |

| | |
|---|---|
| Kullanılabilir Teknik Destek | 4 |
| Fiyat Performans Değeri | 5 |
| Diğer Araç/Sistem ile Entegrasyon | 4 |
| Genel Beğenilme | 4 |

### 4.2.5. PL/SQL Developer 4.0.2 (Allround Automations)

Bir Hollanda yazılım şirketi olan Allround (www.allroundautomations.com) tarafından 1989 yılında kuruldu ve iyi geliştirilen kullanıcı dostu yazılım olarak anılmaktadır. Yönetimsel kalıptan gerçek zamanlı kontrol uygulamalarına değişen yazılım yaratıldı. PL/SQL için pazar araştırması yapıldığında, Delphi ve C++ ara yüzleri arasında benzer hiç bir nokta bulunmadığından PL/SQL 1997'de geliştirilmeye başlandı. PL/SQL Developer ekibi, aracın geliştirilmesine ve fiyat performans oranına odaklanmışlardır.

Allround firmasının otomasyonu olan PL/SQL Developer, genel kalıpların dışında ihtiyaç duyulan neredeyse her özelliğe sahip bir araçtır. Başka editör veya Oracle araçlarına gerek duymadan, Windows telnet veya SQL*Plus oturumu açabilir. Kullanım kılavuzunun tamamını aldığınızda, şimdiye kadar düşündüklerinizden daha fazlasını bulabilirsiniz. 2005'in Ocak ayına kadar yüksek fiyat artışına rağmen, Allround firması kabul edilebilir bir fiyata bu aracı sunmaya devam etmektedir, bu yüzden diğer pahalı rakiplerine karşı değerlidir.

PL/SQL Developer'ın desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. PL/SQL Developer aynı zamanda 15 MB Hafıza ve 30 MB boş diske gereksinim duymaktadır.
Çizelge 4.5'de PL/SQL Developer 4.0.2 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.5.** PL/SQL Developer 4.0.2 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 3 |

| | |
|---|---|
| Geliştirici Faydası | 5 |
| İşlevsellik Ölçüsü | 4 |
| Kullanıcı Arayüzü | 5 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 5 |
| Doküman Kalitesi ve Kapsamı | 5 |
| Kullanılabilir Teknik Destek | 5 |
| Fiyat Performans Değeri | 5 |
| Diğer Araç/Sistem ile Entegrasyon | 5 |
| Genel Beğenilme | 5 |

### 4.2.6. RapidSQL (Embarcadero)

1993 yılında SQL Server ve SYBASE veri tabanları için, grafiksel programlama çevresine sahip olan RapidSQL (www.embarcadero.com), Dec şirketinin amiral gemisi olarak tanıtıldı. Ocak 1998'de tam Oracle desteği eklendi. 1999 yılında DB2 eklendi. Şirketin başarı hikâyeleri oldukça çoktur.

RapidSQL çoklu veri tabanlarını destekleyen, kaynak kod kontrolünü sağlayan ve gelişmiş bir yardım kütüphanesi olan bir programdır. İlişkisel VT için geliştirilen ilk grafiksel programlama çevresine sahip olan RapidSQL, 1993'de tanıtıldı. Bu inceleme Nisan 2001'de yayınlanan 5.7 sürümü göz önüne alınarak yapılmıştır. Embarcdadero' nun hedefi, VT programcılarının seçimi olan bir geliştirme ortamı sağlamak ve yaklaşık olarak her alanda rekabet edecek bir ürün sunmaktır. Bu aracın sağlamlığından dolayı kullanan geliştirmecilerden şikâyet alınmamaktadır.

RapidSQL, çoklu veri tabanlarını desteklemekte, kaynak kod kontrol aracı içermekte, proje yönetimine yardımcı olmakta, VT nesneleri için rapor üretmekte ve mükemmel bir yardım ve destek sağlamaktadır.

RapidSQL'in desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. RapidSQL aynı zamanda 64 MB Hafıza ve 38 MB boş diske gereksinim duymaktadır.
Aşağıdaki çizelgede RapidSQL 5.7 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.6.** RapidSQL 5.7 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 3 |
| Geliştirici Faydası | 4 |
| İşlevsellik Ölçüsü | 5 |
| Kullanıcı Arayüzü | 5 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 5 |
| Doküman Kalitesi ve Kapsamı | 5 |
| Kullanılabilir Teknik Destek | 5 |
| Fiyat Performans Değeri | 4 |
| Diğer Araç/Sistem ile Entegrasyon | 4 |
| Genel Beğenilme | 4 |

### 4.2.7. DBTools 5.0.4 (SoftTree Technologies)

DBTools'un (http://www.softtreetech.com) 5.0 sürümü Şubat 2005'de yayınlanmıştır. Şu yeni parçaları içermektedir; LOB izleyici ve yükleyici, dosya yükleyici, VT boşluk uzmanı, VT kopyalama uzmanı, test veri üreticisi. Ek olarak, yeni VT hata izleyicisi ve hata paketleri, arka plan çalışan VT performans istatistik toplama işlemleri sunmaktadır.

DBTools aynı platformda bulunan 20 farklı parçadan oluşur. Parçalar, DBA ile çalışan bol miktarda idare eden ekstra özelliklere sahiptir. SoftTree programının hatalı bazı kenarları olmasına rağmen, bilgili verilen destek ve çalışan program sunar. DBTools aracının bir çok işlevsel ve parçalı olması kolayca büyümesini sağlar.

Kapsamlı bir settir. İçerisinde; PL/SQL editör ve hata ayıklayıcı, SQL ve uygulama profili, kullanıcı ve kapasite yönetimi, Oracle kapasite ve kaynak planlaması, örnek veri oluşturulması, görev planlaması yönetimi gibi özellikler bulunur.
DBTools'un desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. DBTools aynı zamanda 64 MB Hafıza ve 36 MB boş diske gereksinim duymaktadır.
Aşağıdaki çizelgede DBTools 5.0 sürümü için bir puanlama tablosu verilmiştir.

**Çizelge 4.7.** DBTools 5.0 Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 4 |
| Geliştirici Faydası | 2 |
| İşlevsellik Ölçüsü | 5 |
| Kullanıcı Arayüzü | 4 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 2 |
| Doküman Kalitesi ve Kapsamı | 3 |
| Kullanılabilir Teknik Destek | 4 |
| Fiyat Performans Değeri | 4 |
| Diğer Araç/Sistem ile Entegrasyon | 4 |
| Genel Beğenilme | 3 |

### 4.2.8. SmartDBA Cockpit (BMC)

BMC (www.bmc.com) yazılım, VTYA firması olarak bilinir. Oracle için askeri devriye olarak bilinir ve geçmişte başarılara sahiptir. Web DBA, web tabanlı VT yönetimine cevap vermektedir. BMC yazılımın DataOne çözümleri, bugünün dağıtılmış çevrelerinin bütün yaşam sürecini yönetecek yetenek ile VT profesyonelleri sağlar.

Web tarayıcısı ile web ara yüzü sayesinde Oracle VT'nı yönetmek ve yardımcı olmak için geliştirilen araçtır. İnteraktif olarak uzaktan VT ulaşımını ve yönetimini

sağlamaktadır. Oracle ve Microsoft SQL Server desteği sunmaktadır. Gelecek sürümlerde DB2 desteğini de sunacaktır.
Bu ürün VT yöneticilerinin gün gün çalışması için gereken bütün özelliklere sahiptir. Çok yenilikçi özellikleri; VT hakkında güncel bilgi toplar ve bu görüşler ile ilgili detaylı uyarlanabilir HTML raporlar üretir.

Ara yüz tutarlıdır ve sezgi yoluyla anlaşılabilir. Gerçek zamanlı durum izlemesi içerir, aynı zamanda, boşluk organizasyonu, bakım ve SQL performans ayarlaması sağlar.

Bu ürün DBA'ler için çok yararlı olmasının yanında ara yüzü kullanıcı dostudur. Pazarda birçok VT yönetimiyle ilgili araç vardır fakat onlar uzaktan yönetim kategorisinde değildir. Ürün Londra'da ve New York'da geliştirilmektedir, idare takımı ise Hindistan'dadır.

SmartDBA'in desteklediği işletim sistemleri 32Bit Windows ortamları (Windows 2000/NT/XP/2003) dır. SmartDBA'in desteklediği web servisleri Web sunucu 1.3.9 – 1.3.12 with Apache JServ 1.0, 1.1, Microsoft IIS'dir.
Aşağıdaki çizelgede SmartDBA için bir puanlama tablosu verilmiştir.

**Çizelge 4.8.** SmartDBA Puanlama Tablosu

| | |
|---|---|
| DBA Faydası | 5 |
| Geliştirici Faydası | 3 |
| İşlevsellik Ölçüsü | 4 |
| Kullanıcı Arayüzü | 5 |
| Yazılım Kalitesi/Doğruluk/Sağlamlık | 5 |
| Doküman Kalitesi ve Kapsamı | 4 |
| Kullanılabilir Teknik Destek | 5 |
| Fiyat Performans Değeri | 4 |
| Diğer Araç/Sistem ile Entegrasyon | 4 |
| Genel Beğenilme | 4 |

## 4.3. VTYA Karşılaştırma Tablosu

Bölüm 4.2'de seçilen popüler yönetim araçlarının detaylı karşılaştırması aşağıdaki tablolarda yapılmıştır[3]. Seçilen araçların karşılaştırılması şu şekilde yapılmaktadır. Bir aracın değerini bulmak için karşılaştırılan kıstasların ağırlık değeri ile araca verilen skor değerleri çarpılır. Her bölüm kendi içinde toplanarak her aracın toplam değeri bulunur. Genel toplam sonucu en büyük değeri alan araç birinci seçilir.

Değerlendirmede kullanılan ağırlık kıstasları aşağıdaki şekilde belirlenmiştir.

- Ağırlık Değeri 1: Çok kullanılmayacak olan fakat pastada payı olması gereken değer.
- Ağırlık Değeri 5: Olması gereken.
- Ağırlık Değeri 10: Olması zorunlu, verimliliği gösteren.

Esas noktalarda temel alınan skor değerleri aşağıdaki şekilde belirlenmiştir.

- Değer 0: İstenileni karşılamıyor.
- Değer 1: İstenileni Minimum karşılıyor.
- Değer 2: İstenileni Kısmen karşılıyor.
- Değer 3: İstenileni Çoğunlukla karşılıyor.
- Değer 4: İstenileni Tam karşılıyor.
- Değer 5: İstenileni fazlasıyla karşılıyor.

**Çizelge 4.9.** Arayüz özelliklerine göre karşılaştırma tablosu aşağıda gösterilmiştir.

| **Karşılaştırılan Kriterler**<br><br>**Arayüz Özellikleri** | **Ağırlık** | Rapid SQL 5.7 | | SQL Insight 3.0 | | Oracle SQL Dev 1.0 | | Toad 8.6 | | SQL Stn 5.0 | | SQL Nav. 3.0 | | PlSql Dev 7.0 | | Keep Tool 7.2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Standart Editör Opsiyonları; kes, kopyala, yapıştır, geri al, bul, sonrakini bul, değiştir, satıra git, içeriden başlama, ve yazdırma gibi | 10 | 4 | 40 | 5 | 50 | 3 | 30 | 4 | 40 | 5 | 50 | 4 | 40 | 4 | 40 | 4 | 40 |
| Kelime Vurgulama, ana dosya türlerini destekleme; sql, html, java, ora, ctl, sh &pl dosya tipleri ve EditPlus, MultiEdit, Ultra Edit gibi araç özelliklerini destekleme | 10 | 2 | 20 | 3 | 30 | 2 | 20 | 3 | 30 | 4 | 40 | 3 | 30 | 3 | 30 | 6 | 60 |
| Kesin, akıllı, görsel karşılaştırma ve farklı dosyaları birleştirme | 10 | 3 | 30 | 4 | 40 | 0 | 0 | 4 | 40 | 4 | 40 | 0 | 0 | 1 | 10 | 3 | 30 |
| Son çalışılan dizinleri ön belleğe alma | 10 | 4 | 40 | 4 | 40 | 0 | 0 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 |
| Ekranların kullanılabilirliği ve sezgi yolu ile çözülmesi | 10 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 6 | 60 | 5 | 50 |
| Doğru yanıt veren teknik destek | 10 | 5 | 50 | 4 | 40 | 3 | 30 | 5 | 50 | 1 | 10 | 4 | 40 | 5 | 50 | 4 | 40 |
| Kolay, bilgi verici, içerik hassas yardım | 10 | 5 | 50 | 3 | 30 | 3 | 30 | 5 | 50 | 2 | 20 | 3 | 30 | 5 | 50 | 3 | 30 |
| Gelişmiş, vazgeçilmez kod editör özellikleri: hepsini kapa, hepsini kaydet, otomatik kaydet, kitap işareti, kolon seçimi, dosya ekleme, dosya adından dosya açma vb | 10 | 2 | 20 | 3 | 30 | 2 | 20 | 3 | 30 | 3 | 30 | 3 | 30 | 4 | 40 | 3 | 30 |
| Otomatik tamamlama/ değiştirme kısa yolları | 10 | 2 | 20 | 5 | 50 | 2 | 20 | 5 | 50 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 |
| Proje bağlantıları bağlama yeteneği. Dizayn dokümanına bağlanma, kaynak veri dosyaları, To-Do listeleri, not karalama vb. | 5 | 0 | 0 | 4 | 20 | 0 | 0 | 5 | 25 | 0 | 0 | 0 | 0 | 4 | 20 | 0 | 0 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Belli dosyaları veya VT nesneleri açıldığında, eğer To-Do listeleri, proje parçalarını veya nesne isimlerini haritalayabilirse ek puan alır. | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 10 | 0 | 0 | 0 | 0 | 4 | 20 | 0 | 0 |
| Değiştirilebilir anahtar kelime sözlükleri | 5 | 0 | 0 | 5 | 25 | 0 | 0 | 4 | 20 | 3 | 15 | 0 | 0 | 4 | 20 | 4 | 20 |
| Alt dizin, tek dosya, çoklu dosya, dosya açma veya dizin yapısına bakma, arama ve değiştirme özelliği | 5 | 4 | 20 | 5 | 25 | 0 | 0 | 3 | 15 | 5 | 25 | 0 | 0 | 2 | 10 | 0 | 0 |
| Lider diğer SCC araçları ile iletişim kurma | 5 | 4 | 20 | 4 | 20 | 0 | 0 | 5 | 25 | 2 | 10 | 3 | 15 | 4 | 20 | 4 | 20 |
| İnternet kısa yolları için menü /pencere oluşturma. (özellikle yararlı Oracle grupları ve teknik web sayfaları için) | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 0 | 0 | 5 | 5 | 1 | 1 |
| Diğer desteklenen diller için, Değiştirilebilir kod formatlama ve otomatik güzelleştirme | 1 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Uzaktaki terminallere ftp ile ulaşım | 1 | 0 | 0 | 4 | 4 | 0 | 0 | 5 | 5 | 4 | 4 | 3 | 3 | 4 | 4 | 0 | 0 |
| **Alt Toplam (Ağırlık x Değer)** | | **350** | | **447** | | **190** | | **477** | | **368** | | **308** | | **460** | | **402** | |

**Çizelge 4.10.** VT programlama aracı olarak karşılaştırma tablosu aşağıda gösterilmiştir.

| **Karşılaştırılan Kriterler**<br><br>**VT Programlama Aracı Olarak** | **Ağırlık** | Rapid SQL 5.7 | | SQL Insight 3.0 | | Oracle SQL Dev 1.0 | | Toad 8.6 | | SQL Stn 5.0 | | SQL Nav. 3.0 | | PlSql Dev 7.0 | | Keep Tool 7.2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Özel komut dosyası çalıştırılmadan ve kurmadan tam işlevsellik | 10 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 1 | 10 | 4 | 40 | 4 | 40 | 4 | 40 |
| Tam özellikli, hızlı, filtre edilebilir, VT izleme | 10 | 3 | 30 | 2 | 20 | 3 | 30 | 4 | 40 | 4 | 40 | 3 | 30 | 4 | 40 | 5 | 50 |
| Tek örneklem (instance) ile tek bağlantıyı koruma | 5 | 0 | 0 | 4 | 20 | 4 | 20 | 4 | 20 | 4 | 20 | 4 | 20 | 4 | 20 | 4 | 20 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bir örneklem (instance) ile çoklu bağlantı sağlama, uzun çalışan DML veya derleme işlemlerinde programın kilitlenmesini önleme. | 10 | 4 | 40 | 0 | 0 | 0 | 0 | 4 | 40 | 0 | 0 | 0 | 0 | 5 | 50 | 2 | 20 |
| Çoklu örneklem ile tekil koşut zamanlı bağlantıları koruma. (her defasında el ile bağlanılmamalı) | 5 | 0 | 0 | 4 | 20 | 4 | 20 | 4 | 20 | 0 | 0 | 4 | 20 | 2 | 10 | 3 | 15 |
| Çoklu örneklem ile çoklu koşut zamanlı bağlantıları koruma. | 10 | 4 | 40 | 0 | 0 | 0 | 0 | 4 | 40 | 0 | 0 | 0 | 0 | 2 | 20 | 1 | 10 |
| Değiştirilebilir PL/SQL nesne kalıp taslağı (şirket standartlarına göre) | 10 | 3 | 30 | 4 | 40 | 0 | 0 | 4 | 40 | 3 | 30 | 1 | 10 | 5 | 50 | 4 | 40 |
| Yerel ağda bölüşülebilir, şık kod, blok/inşa, sorgu için uyarlanabilir şablonlar (hızlı takım kodu geliştirmek için) | 10 | 3 | 30 | 4 | 40 | 1 | 10 | 4 | 40 | 3 | 30 | 1 | 10 | 5 | 50 | 4 | 40 |
| VT fonksiyonları için sürükle bırak | 10 | 3 | 30 | 3 | 30 | 4 | 40 | 4 | 40 | 3 | 30 | 4 | 40 | 4 | 40 | 4 | 40 |
| Tümleşik SQL cümlesi ayarlama araçları | 10 | 2 | 20 | 6 | 60 | 2 | 20 | 5 | 50 | 5 | 50 | 4 | 40 | 3 | 30 | 4 | 40 |
| SQL cümlesi çalıştırma, seçili bölümü çalıştırma, tekli SQL DML çalıştırma, çoklu SQL cümlesi çalıştırma | 10 | 4 | 40 | 5 | 50 | 4 | 40 | 5 | 50 | 4 | 40 | 3 | 30 | 5 | 50 | 4 | 40 |
| Güçlü, görsel tablo editörü, sorgu ve tüm tabloyu değiştirme, kopyalayabilme, kaydetme, sıralama, yazma ve veri çıkma | 10 | 2 | 20 | 3 | 30 | 3 | 30 | 5 | 50 | 2 | 20 | 4 | 40 | 5 | 50 | 6 | 60 |
| Tabloları içeri atma etme ve sonuçları geçici taşıma panosuna çıkma (xls, tsv, xml) | 10 | 3 | 30 | 4 | 40 | 2 | 20 | 5 | 50 | 3 | 30 | 2 | 20 | 4 | 40 | 6 | 60 |
| Yedeklerin içinden tablo çıkarmaya yardım edici araçlar | 10 | 2 | 20 | 0 | 0 | 1 | 10 | 4 | 40 | 2 | 20 | 2 | 20 | 5 | 50 | 6 | 60 |
| Bütün VT nesnelerinin DDL'lerini kusursuz çıkarma. | 10 | 4 | 40 | 4 | 40 | 1 | 10 | 5 | 50 | 3 | 30 | 3 | 30 | 4 | 40 | 4 | 40 |
| VT'ndan bütün nesne türleri için, nesneye has mönü ve bu nesneler her nerede olursa kullanıcı ara yüzünün bulunması | 10 | 2 | 20 | 3 | 30 | 2 | 20 | 4 | 40 | 2 | 20 | 4 | 40 | 6 | 60 | 4 | 40 |
| VT'na bağlı olmadan, cümlecikleri değiştirme | 10 | 4 | 40 | 4 | 40 | 4 | 40 | 1 | 10 | 4 | 40 | 0 | 0 | 4 | 40 | 4 | 40 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VT nesnelerinin görsel karşılaştırılması (dosya ile dosyayı veya VT nesnesi ile VT nesnesini) | 10 | 3 | 30 | 4 | 40 | 0 | 0 | 5 | 50 | 4 | 40 | 0 | 0 | 3 | 30 | 2 | 20 |
| SQL için değiştirilebilir kod formatlayıcı / güzelleştirici. | 10 | 2 | 20 | 5 | 50 | 2 | 20 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 3 | 30 |
| Oturumlara, pencerelere ve VT'na bağlamadan, eski SQL cümlelerini kontrol etme. | 10 | 1 | 10 | 5 | 50 | 3 | 30 | 5 | 50 | 0 | 0 | 3 | 30 | 5 | 50 | 3 | 30 |
| VT'na gönderilen uzun çalışan DML'leri iptal edebilme. | 10 | 2 | 20 | 4 | 40 | 4 | 40 | 4 | 40 | 1 | 10 | 4 | 40 | 4 | 40 | 4 | 40 |
| Kolayca VT bağlantısı değiştirebilme ve SQL cümlelerini çeşitli şemalarda çalıştırabilme. | 10 | 6 | 60 | 4 | 40 | 4 | 40 | 4 | 40 | 2 | 20 | 4 | 40 | 4 | 40 | 2 | 20 |
| VT nesne tarayıcısından çoklu nesne seçme ve işlem yapma (analyze, drop vb) | 10 | 4 | 40 | 2 | 20 | 1 | 10 | 5 | 50 | 3 | 30 | 3 | 30 | 4 | 40 | 3 | 30 |
| Grafiksel olarak nesnelerin her iki tarafa bağımlılığının gösterilmesi (tablolar, nesneler, tetikleyiciler, görüntüler vb) | 10 | 3 | 30 | 3 | 30 | 1 | 10 | 4 | 40 | 4 | 40 | 2 | 20 | 4 | 40 | 4 | 40 |
| Nesneyi kapsayan her hangi bir hakları yakalayabilme, cümleciklere silme ve kaldırma ekleyebilme. | 10 | 3 | 30 | 4 | 40 | 0 | 0 | 4 | 40 | 5 | 50 | 2 | 20 | 4 | 40 | 5 | 50 |
| Grafiksel ızgara ve tarayıcılarla, ana/detay veri gösterimi. | 5 | 1 | 5 | 1 | 5 | 0 | 0 | 4 | 20 | 0 | 0 | 0 | 0 | 4 | 20 | 5 | 25 |
| Şema bazında tablo verilerini karşılaştırabilmeli. Tablo verilerini senkronize ederek birleştirebilmeli. | 5 | 1 | 5 | 0 | 0 | 0 | 0 | 5 | 25 | 0 | 0 | 0 | 0 | 3 | 15 | 0 | 0 |
| İsimlendirme, esnetik kurallar ve diğer ayarlanabilir eğilimleri kontrol edebilme. | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 25 | 0 | 0 | 0 | 0 | 2 | 10 | 0 | 0 |
| Kod bloklarını vurgulama. Açık pencerelerde seçilmiş tanımlayıcıların her kullanımda vurgulanması.. | 5 | 0 | 0 | 0 | 0 | 3 | 15 | 2 | 10 | 0 | 0 | 0 | 0 | 5 | 25 | 2 | 10 |
| Tersine tasarım DDL leri için ayarlanabilir formatlayıcı. | 5 | 3 | 15 | 2 | 10 | 1 | 5 | 3 | 15 | 0 | 0 | 0 | 0 | 2 | 10 | 3 | 15 |
| Çalıştırılırken veya yürütülürken imlecin etrafında bulunan SQL cümlelerini otomatik seçme. | 5 | 0 | 0 | 4 | 20 | 4 | 20 | 0 | 0 | 3 | 15 | 0 | 0 | 4 | 20 | 4 | 20 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Grafiksel modelleme aracı. Tersine mühendislik, alt kümeler, yazıcı, grup renkleri, SVG, JPG, EMF ve PDF çıktıları. | 5 | 0 | 0 | 3 | 15 | 4 | 20 | 3 | 15 | 0 | 0 | 0 | 0 | 5 | 25 | 4 | 20 |
| Bağlantı profillerinin organizasyonu, her bir VT örneği, kurunmuş şifre içerme. | 5 | 5 | 25 | 4 | 20 | 4 | 20 | 5 | 25 | 3 | 15 | 4 | 20 | 5 | 25 | 4 | 20 |
| Seçilen nesnelerin otomatik ortak DML'lerinin oluşturulması. | 5 | 0 | 0 | 4 | 20 | 1 | 5 | 5 | 25 | 0 | 0 | 0 | 0 | 4 | 20 | 3 | 15 |
| Fonksiyonların çalışma yöntemlerinin görsel oluşturulması. | 5 | 3 | 15 | 4 | 20 | 3 | 15 | 3 | 15 | 3 | 15 | 3 | 15 | 5 | 25 | 3 | 15 |
| VT nesnesi arama araçları. | 5 | 2 | 10 | 5 | 25 | 1 | 5 | 3 | 15 | 3 | 15 | 2 | 10 | 5 | 25 | 3 | 15 |
| VT tarayıcısı nesne sahibine göre filtre edebilmeli (tipi, adı, durumu da filtre seçenekleri olabilir). | 5 | 3 | 15 | 3 | 15 | 2 | 10 | 5 | 25 | 4 | 20 | 3 | 15 | 5 | 25 | 3 | 15 |
| VT nesneleri değişebilir, fakat tersine mühendislik nesnelerin varsayılan durumu salt okunurdur. | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 25 | 4 | 20 | 0 | 0 | 5 | 25 | 0 | 0 |
| Tablolar için test verilerinin oluşturulması için yardımcı özellikler. | 5 | 0 | 0 | 2 | 10 | 1 | 5 | 4 | 20 | 0 | 0 | 0 | 0 | 4 | 20 | 2 | 10 |
| VT'na bağlanırken hassas grafiksel gösterim. | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 20 | 0 | 0 | 0 | 0 | 4 | 20 | 5 | 25 |
| Diğer veri tabanlarının desteklenmesi (DB2, Sybase, SQL-Server, Informix vb) | 1 | 5 | 5 | 0 | 0 | 0 | 0 | 4 | 4 | 3 | 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| Var olan nesne kodları için dokümantasyon oluşturulması (Html, PDF). | 1 | 3 | 3 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 | 4 | 3 | 3 |
| Grafiksel harita veya grafiksel karmaşık sorgu. | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 4 | 4 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 |
| **Alt Toplam (Ağırlık x Değer)** | | **808** | | **971** | | **620** | | **1300** | | **743** | | **670** | | **1319** | | **1127** | |

**Çizelge 4.11.** Oracle PL/SQL özelliklerine göre karşılaştırma tablosu gösterilmiştir.

| **Karşılaştırılan Kriterler**<br><br>**Oracle / PLSQL Özellikleri** | **Ağırlık** | Rapid SQL 5.7 | | SQL Insight 3.0 | | Oracle SQL Dev 1.0 | | Toad 8.6 | | SQL Stn 5.0 | | SQL Nav. 3.0 | | PlSql Dev 7.0 | | Keep Tool 7.2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Prosedür, fonksiyon, tip gibi blokların PL/SQL test etmek için üretebilmelidir. LOG, UDT, nesne türleri, kayıt ve referans imleç parametrelerini ele alabilmeli. | 10 | 3 | 30 | 1 | 10 | 2 | 20 | 3 | 30 | 3 | 30 | 3 | 30 | 5 | 50 | 3 | 30 |
| Gerçek SQL çalıştırması olmadan, çalışma planı alınabilmelidir. Planın bazılarını veya bütününü yakalayabilmelidir. | 10 | 2 | 20 | 6 | 60 | 2 | 20 | 4 | 40 | 3 | 30 | 4 | 40 | 5 | 50 | 2 | 20 |
| SQL*PLUS' ın taklit edilmesi ve sıkı bağlantı. Eğer geliştirici hala SQL*Plus kullanıyor ise, bu özellik tam olarak karşılamamalı. | 10 | 1 | 10 | 4 | 40 | 3 | 30 | 5 | 50 | 1 | 10 | 3 | 30 | 5 | 50 | 3 | 30 |
| Çevrim dışı SQL ve PL/SQL söz dizim kontrolü. | 10 | 5 | 50 | 3 | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Hata yeri ve gösterime uygun hata mesajlarında, imleci doğru yere yerleştirerek hatanın kolay tanınmasını sağlama. | 10 | 3 | 30 | 4 | 40 | 2 | 20 | 5 | 50 | 3 | 30 | 4 | 40 | 5 | 50 | 4 | 40 |
| Karmaşık ve gelişmiş Oracle veri tiplerinin tablo kolon içeriklerini görüntüle ve değiştir. (Nested Tables, varrays, objects, CLOB, BLOB, XMLTYPE, BFILE, images) | 10 | 0 | 0 | 2 | 20 | 2 | 20 | 5 | 50 | 2 | 20 | 2 | 20 | 5 | 50 | 3 | 30 |
| Nesne merkezli yeni özellikler için tam destek ağlamak. Yeni anahtar kelimeler, yeni veri tipleri vb | 10 | 3 | 30 | 4 | 40 | 2 | 20 | 5 | 50 | 2 | 20 | 2 | 20 | 4 | 40 | 4 | 40 |
| Oracle'in profilindeki görsel ve grafiksel araçlar. | 10 | 4 | 40 | 4 | 40 | 0 | 0 | 5 | 50 | 0 | 0 | 0 | 0 | 4 | 40 | 3 | 30 |
| Özellikle Oracle 9i'deki SYS_REFCURSOR'un kuvvetli ve zayıf referans imleçlerinin gösterimini ele alma. | 10 | 1 | 10 | 1 | 10 | 0 | 0 | 3 | 30 | 2 | 20 | 1 | 10 | 5 | 50 | 1 | 10 |
| SQL ve PL/SQL kod analizi. | 5 | 3 | 15 | 5 | 25 | 2 | 10 | 5 | 25 | 4 | 20 | 4 | 20 | 4 | 20 | 0 | 0 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HTML sayfalarını ileri mühendislik ile gönderme. (HTMLDB, PL/SQL Server Pages, ve PL/SQL Web Toolkit (htp,htf) | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Web veya iç gösterim ile, derleme veya HTP/HTP çıktısının görünüşünü yakalama. | 5 | 3 | 15 | 4 | 20 | 3 | 15 | 0 | 0 | 4 | 20 | 4 | 20 | 4 | 20 | 0 | 0 | |
| Verilen bir şema içinde paketler arası topoloji/bağımlılığı analiz ve haritalamak için grafiksel araçlar. | 5 | 0 | 0 | 3 | 15 | 0 | 0 | 2 | 10 | 0 | 0 | 0 | 0 | 1 | 10 | 0 | 0 | |
| Seçili veya bütün geçersiz nesnelerin derlenmesi. Geçersiz nesneler arasında bağımlılığı olanların otomatik bulunması. | 5 | 2 | 10 | 0 | 0 | 0 | 0 | 3 | 15 | 0 | 0 | 0 | 0 | 5 | 25 | 4 | 20 | |
| Eşzamanlı olmayan özellikler; DBMS_AQ MADDESİ, DBMS_PIPE, DBMS_ALERT İÇİNE çengel ile yakalama, DBMS_APPLICATION_INFO aracılığıyla V$SESSION'DA değişiklik | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | |
| Oracle XMLDB ile çalışmaya yardım edecek araçlar. | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | |
| OID, Oracle Names Server, Tnsnames.ora, sqlnet.ora gibi Oracle bağlantı bilgilerinin gösterilmesi ve değiştirilmesi için editör. | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 4 | 0 | 0 | 0 | 0 | 3 | 3 | 1 | 1 | |
| Komut satırından, söz dizimi kontrol, Java derleme yetenekleri. | 1 | 2 | 2 | 3 | 3 | 1 | 1 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | |
| Bütün proje ve kaynak dosyalarını Oracle OFS den veya Oracle OFS ye açma ve kaydetme izin verme özelliği. (IFS kullanarak) | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 0 | 0 | |
| Durum veya oturum için otomatik izleme veya TKPROF kolayca açma. | 1 | 0 | 0 | 4 | 4 | 0 | 0 | 5 | 5 | 4 | 4 | 4 | 4 | 5 | 5 | 4 | 4 | |
| **Alt Toplam (Ağırlık x Değer)** | | **263** | | **357** | | **156** | | **414** | | **206** | | **236** | | **473** | | **262** | | |

**Çizelge 4.12.** Oracle PL/SQL Derleme yeteneklerinden beklenen özelliklerine göre karşılaştırma tablosu aşağıda gösterilmiştir.

| **Karşılaştırılan Kriterler**<br><br>**PL/SQL Derleme Yeteneklerinden Beklenenler** | **Ağırlık** | Rapid SQL 5.7 | | SQL Insight 3.0 | | Oracle SQL Dev 1.0 | | Toad 8.6 | | SQL Stn 5.0 | | SQL Nav. 3.0 | | PlSql Dev 7.0 | | Keep Tool 7.2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Derleme ve hata bağımlılığını otomatik ele alma. Nesnelerin hatalarını bulabilmek ve izlemek için, hata ayıklamaya eklemek. Derleme/hata ayıklama bağlı nesneler için kullanıcıya sorulması. | 10 | 1 | 10 | 1 | 10 | 2 | 20 | 2 | 20 | 2 | 20 | 2 | 20 | 2 | 20 | 2 | 20 |
| Tetikler ve saklanmış programlara kesme noktası (breakpoint) eklenmesi. Dosya temelli veya nesneler, hata ayıklamadan önce VT den çekilir. | 10 | 2 | 20 | 2 | 20 | 2 | 20 | 4 | 40 | 4 | 40 | 4 | 40 | 5 | 50 | 4 | 40 |
| Bütün standart hata ayıklayıcılarda; içeri, dışarı, üstünde, imlece kadar, durdurma, dinamik kesme noktası koyma/kaldırma, değişkenleri izleme/değiştirme Ek Puan: Eğer fare, değişkenin üstüne gelmesi ve değeri göstermesi. | 10 | 3 | 30 | 4 | 40 | 4 | 40 | 4 | 40 | 4 | 40 | 3 | 30 | 5 | 50 | 3.5 | 35 |
| Kapsam içerisinde, bütün parametreleri ve bütün değişkenleri görebilme ve değiştirebilme. RAW, String, Kullanıcı tanımlı nesne ve kayıtlar, Tablo bazlı kayıtlar, CLOB, BLOB, BFILE, XMLTYPE. | 10 | 3 | 30 | 3 | 30 | 2 | 20 | 0 | 0 | 3 | 30 | 2 | 20 | 5 | 50 | 3 | 30 |
| Çağrılan PL SQL nesnelerini derlemek için uygun, kendisinin üzerinde geliştirilen, hiçbir kontrolün olmadığı ön-uç uygulamaları ile kabul etme. (3.parti uygulamalar, JDBC, OCI vb) | 5 | 0 | 0 | 0 | 0 | 3 | 15 | 3 | 15 | 4 | 20 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tetikleri derlemek için tam destek (tetikleri n yerine), ve metotları yazmak | 5 | 0 | 0 | 4 | 20 | 4 | 20 | 4 | 20 | 3 | 15 | 0 | 0 | 4 | 20 | 4 | 20 |
| Programlar ve oturum için hata ayıklarken, DBMS_OTPUT programları ile gösterme. | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 25 | 0 | 0 | 2 | 10 | 4 | 20 |
| **Alt Toplam (Ağırlık x Değer)** | | **90** | | **120** | | **135** | | **135** | | **190** | | **110** | | **200** | | **165** | |

**Çizelge 4.13.** DBA ve uygulama geliştiriciye göre karşılaştırma tablosu gösterilmiştir

| **Karşılaştırılan Kriterler**<br><br>**DBA ye SR. Göre, Oracle programcıları** | **Ağırlık** | Rapid SQL 5.7 | | SQL Insight 3.0 | | Oracle SQL Dev 1.0 | | Toad 8.6 | | SQL Stn 5.0 | | SQL Nav. 3.0 | | PlSql Dev 7.0 | | Keep Tool 7.2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bütün Oracle nesnelerinin yaratılması ve bakımının yapılması (sil, oluştur, değiştir) | 10 | 2 | 20 | 2 | 20 | 3 | 30 | 5 | 50 | 2 | 20 | 2 | 20 | 2 | 20 | 5 | 50 |
| Oturumun gösterilmesi ve yönetilmesi (bütün v$Session bilgisi, aktif SQL, eski SQL, istatistik, kilit, oturum düşürme, otomatik yenileme). | 10 | 1 | 10 | 1 | 10 | 2 | 20 | 5 | 50 | 2 | 20 | 1 | 10 | 5 | 50 | 5 | 50 |
| Şemaları karşılaştıracak araç. | 10 | 3 | 30 | 2 | 20 | 0 | 0 | 3 | 30 | 2 | 20 | 0 | 0 | 3 | 30 | 2 | 20 |
| VT Yönetici görevleri, (export, import, Tablespace yönetimi, İndislerin yer değiştirilmesi, kullanıcılar, roller, haklar, analiz, tekrar derleme, kilit, bekleyen işler, oturumlar, uzantılar, SGA, server durumu, boşluk ataması, vb). | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 6 | 30 | 1 | 5 | 1 | 5 | 2 | 10 | 6 | 30 |
| DBA seviyesi performans aracı; v$(dictionary) viewer, PFILE ve SPFILE parametreleri, Pool optimizasyonu, TKPROF için ara yüz, vb. | 5 | 0 | 0 | 4 | 20 | 0 | 0 | 4 | 20 | 1 | 5 | 3 | 15 | 2 | 10 | 2 | 10 |
| Bazı tekil(OEM de bulunmayan) DBA merkezli, 10g yeni özellikleri (Datapunp, instant client, log mining, 10g program). | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 25 | 0 | 0 | 0 | 0 | 1 | 5 | 4 | 20 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bağlantıların kullanıcılar tarafından ele alınması, nesne tarayıcı, VT komut dosyalarının çalıştırılması. | 1 | 5 | 5 | 0 | 0 | 0 | 0 | 4 | 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| VT veri sözlüğü/ nesnelerinin raporlanması (HTML,PDF). | 1 | 5 | 5 | 2 | 2 | 2 | 2 | 5 | 5 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |
| **Alt Toplam (Ağırlık x Değer)** | | **75** | | **77** | | **57** | | **214** | | **74** | | **53** | | **129** | | **184** | |

**Çizelge 4.14.** Araçların karşılaştırma sonuçların gösterildiği tablo aşağıda verilmiştir.

| **Genel Toplam** | Rapid SQL 5.7 | SQL Insight 3.0 | Oracle SQL Dev 1.0 | Toad 8.6 | SQL Stn 5.0 | SQL Nav. 3.0 | Pl/Sql Dev 7.0 | Keep Tool 7.2 |
|---|---|---|---|---|---|---|---|---|
| Arayüz özellikleri | 350 | 447 | 190 | 477 | 368 | 308 | 460 | 402 |
| VT programlama aracı | 808 | 971 | 620 | 1300 | 743 | 670 | 1319 | 1127 |
| Oracle PL/SQL özellikleri | 263 | 357 | 156 | 414 | 206 | 236 | 473 | 262 |
| PL/SQL derleme yeteneklerinden beklenenler | 90 | 120 | 135 | 135 | 190 | 110 | 200 | 165 |
| DBA ve SR'ye göre Oracle programcıları | 75 | 77 | 57 | 214 | 74 | 53 | 129 | 184 |
| **Genel Toplam** | 1586 | 1972 | 1158 | 2540 | 1581 | 1377 | **2581** | 2140 |

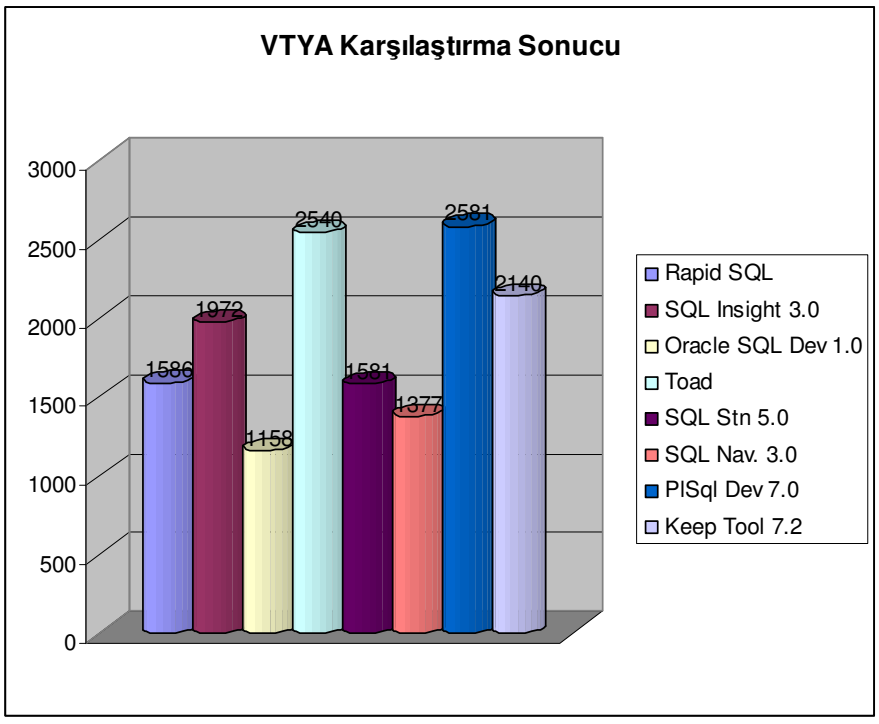
## Sonuç

Yukarıdaki tabloda en popüler Oracle VT yönetim programları karşılaştırılmıştır [3]. Bir Oracle Yönetim programında olması gereken özellikler bu tablolarda gösterilmiş ve değerlendirilmiştir. Tablo 4.14 ve şekil 4.1'de karşılaştırılan araçların değersel ve grafiksel sonuçları gösterilmektedir. Buna göre SQL Insigt 3.0, Toad 8.6, PL/SQL Dev 7.0 ve KeepTool 7.2 ortalamanın üzerinde puan almıştır. Programların en çok puanı arayüz ve PL/SQL özelliklerinden aldığı görünmektedir. Yapılan

değerlendirmeye göre Oracle firmasının aracı olan SQL Dev.1.0 1158 puan alarak sonuncu olmuştur. PL/SQL Developer 7.0 ise 2581 puan ile birinci seçilmiştir.
PL/SQL Developer aracının birinci olmasındaki etken, kullanıcı arayüzlerine ve PL/SQL özelliklerine ağırlık vermesi ve kullanıcıların ihtiyaç duyduğu neredeyse her özelliğe sahip olmasıdır. Bu özellikler bu aracı üst seviyeye taşımıştır.

SQL Dev. 1.0 aracının en düşük puanı almasındaki sebep ise genel bir araç yazmaya çalışmaları ve Oracle VT için gerekli işlevselliği sağlayamamasıdır.

Yapılan karşılaştırmada fiyat kıstası dikkate alınmamıştır [3].




**Şekil 4.1.** Araçların karşılaştırma sonuçların grafiksel gösterimi.

Bu karşılaştırma ile yeni bir program alınacak veya geliştirilecekse; nelere dikkat edilmesi gerektiği ve neleri kapsaması gerektiği gösterilmeye çalışılmıştır. Buradan alınan sonuçlar ile Bölüm 5'de bu tez kapsamında oluşturulan VTYA'nın karşılaştırması yapılacaktır.

# BÖLÜM 5

# 5. YENİ BİR VERİ TABANI YÖNETİM ARACI

## 5.1. Motivasyon

SQL Server, Oracle ve Sybase gibi birçok VTYS hem Çevrimiçi Analitik İşleme (OLAP) hem de Çevrimiçi Hareket İşleme (OLTP) sistemi olarak ayarlanabilir özellikleri bünyesinde barındırır. Ancak bir VT yöneticisinin ilgili ayarlamaları yapması gerekir. VT'nın kurulumu, yedeklenmesi, bakımı ve yenilenmesi gibi düzenli bakım gerektiren işlemlerinin gerçekleştirilmesi de bu ayarlamalar ile birlikte olur. Kullanıcı yönetimi ve yetkilendirme gibi işler "Veri Tabanı Yönetimi" olarak adlandırılmaktadır.

Bir VT programcısı; kullanılacak olan tabloların neler olacağına ve hangi ilişkiler ile birbirlerine bağlanacağına karar veren, artan kayıt sayılarına bağlı olarak uygulamanın hız problemini aşmak için gerekli indisleri tanımlayan, kısıtlamaların yetersiz kaldığı durumlarda tetikleyici gibi yapılarla veri bütünlüğünü sağlama gibi konulara hakim olan kişidir. Ancak bu kişinin bir uygulamayı ortaya koyabilmek için ayrı bir API ile VT'na erişmesi ve bu API'nin nasıl kullanıldığını da öğrenmesi gerekir. Bütün bunlar ciddi birikimler gerektirebilir.

Ülkemizde, programlama üstünde duran birçok kaynak ve yayın olmasına rağmen maalesef VT tasarımı, performans çalışması ve VT programlama hakkında fazla kaynak bulunmamaktadır.

Yeni bir VTYA geliştirilmesindeki temel amaç, uygulama geliştiricilerin ve VT yöneticilerinin kullanabileceği ortak açık kaynak olan bir Türkçe programın oluşturulmasıdır. Diğer önemli etken ise, piyasada bulunan diğer araçların hepsinin ücretli ve kapalı kutu olmasıdır. Bölüm 4'de en popüler araçlar için performans araştırması yapılmıştır. Bu araştırma ile küçük çaplı geliştirici ve VT yöneticileri için bu araçların ne kadar maliyetli olduğu ortaya konmuştur.

DBAExplorer'un geliştirilmesinde diğer önemli hedef ise üyelik gerektirmeyen web sayfası/form oluşturularak yetkin kişilerce her türlü Oracle desteğinin verilmesidir. Bu hizmet için "*www.DBAExplorer.com*" adlı sayfa satın alınmış ve geliştirilmeye başlanmıştır. Bu web sayfasının amacı, oluşturulan program ile birlikte kaynak kodlarının dağıtımı, Oracle soru ve sorunları olan kişilere en etkin yardımın sağlanmasıdır.

## 5.2. DBAExplorer'ın Tanıtımı

DBAExplorer bir VT sorgulama ve yönetim aracı olup Delphi [45] programlama dili ile geliştirilmiştir. Eğitim maksadıyla ya da kişisel maksatlarla kullanımı ücretsizdir. Ücretsiz kullanımda herhangi bir kısıtlama söz konusu değildir. Bu durum geliştirilen ürünü cazip kılmaktadır.

DBAExplorer sadece Oracle VTYS desteklemektedir ve Windows işletim sisteminde çalışmaktadır. Programa bağlanırken kullanılan Bağlantı (Login) seçeneği ile desteklenen VT sunucularını görebilirsiniz. Oracle VT sunucusuna Direkt Oracle Bağlantısı (Oracle Data Access Components – ODAC) [44] bileşeni veya Oracle bileşenleri ile bağlanmak mümkündür.

DBAExplorer Oracle VT sunucusu ile bütünleşik olarak çalışır. Yani aynı anda birden çok Oracle VT'na bağlanarak farklı oturumlarda kolay bir çalışma ortamı sağlar.

DBAExplorer ileri görsel özelliklere sahiptir. Bağlantı kurulan bir VT sunucusunda bulunan şema isimlerini ekrana getirir ve fare tıklaması ile istenilen şemanın seçilmesine imkân sağlar. Örneğin seçilen şema üzerinde bulunan tabloları ekrana getirir ve fare tıklaması ile rahatlıkla herhangi bir tablo işlemi yapılmasına olanak sağlar. Başka bir pencere açmadan seçilen tablonun kolonlarını ve veri tiplerini renkli olarak ekrana getirir. Aynı şekilde tablo, indis, prosedür, tetikleyici ve

görünüm gibi. VT nesneleri ile çalışma imkanı sağlar. Bu nesnelerin her türlü özelliklerini ve SQL veri ve veri modeli tanımlama dili (DDL) ve DML ifadelerini (CREATE, ALTER, DROP, SELECT, INSERT, UPDATE ve DELETE gibi) editör ortamına getirir.

VT, tablo, indis, tetikleyici ve görünüm gibi nesneleri oluşturmak ya da sistemden silmek oldukça kolaydır. Örneğin VT nesnesi oluşturmak için yeni bir pencere açılır. Açılan pencerede görsel olarak işlemleri gerçekleştirmek mümkündür ve nesneleri oluşturmadan SQL ifadesini ön izleme ile görmek mümkündür. Nesne silme işlemini gerçekleştirirken onay sorusu sorulur, ancak buna rağmen bu özelliği kullanırken dikkatli olmak gerekir.

DBAExplorer ile çalıştırılan sorgu ifadesi sonrası gelen sonuç kayıtları üzerinde çeşitli güncelleme, silme hatta yeni kayıt ekleme aktiviteleri yapabilmek ve en son durumu da VT'nında saklamak oldukça kolaydır.

DBAExplorer ile tabloda bulunan verileri dosyaya çıkma ya da tersi özelliği vardır. VT ER diyagramlarını çıkarmak, SQL ifadelerini otomatik olarak oluşturmak ve komut dizisi (script) üretmek mümkündür.

SQL Editörü sayesinde yeni prosedür yazılabilir veya SQL çalıştırılabilir. Zamandan kazanmak ve performans sağlamak amacıyla SQL çalışma sonucu sayfalı (paging) olarak ekrana gelmektedir. Yazılan her türlü SQL için analiz alınabilmektedir.

DBAExplorer bu tez kapsamında araştırılan performans çözümleri ile ilgili SQL cümlelerini belirli periyotlarda çalıştıran ve sonuçlarını ekranda gösteren ve aynı zamanda e-posta gönderebilen bir yapıya sahiptir.

DBAExplorer yazılımı geliştirme sürecinde, uygulama yazılımı içinde kullanılacak ve VT işlemlerini kolaylaştıracak kodlama işlemleri büyük önem taşımaktadır. Kodlama işlemi yapılırken kullanılacak alt programlardan yazılım diline, SQL cümlelerinden form yapısına kadar birçok alanda dikkat edilmesi gereken kurallar

bulunmaktadır. Bu kuralların bazıları standart yazılım geliştirme süreci kurallarını içermektedir. Bu kurallara ek olarak yazılım mühendisliği için geliştirmiş IEEE/EIA 12207 ve yazılım süreç iyileştirme yöntemleri (benimseme süreci, yeterli ve nitelikli kaynak yönetimi) gibi bazı yazılım mühendisliği standartlarında uyulmuştur.

### 5.2.1. Uygulamanın Geliştirileceği Ortamın Belirlenmesi

Uygulamanın geliştirileceği ortam belirlenirken, aşağıdaki kıstaslar göz önüne alınmıştır:

- Windows işletim sistemini kullanarak VT'na direk bağlantı sağlaması
- İşlemleri hızlandırmak amacıyla grafik ara yüzü olması
- VT'na uzaktan bağlantı kurmayı desteklemesi
- Arada hiçbir katman bulunmadan VT'na direk bağlanması
- Yüksek hızda çalışarak çoklu işlemlerde izlek (thread) kullanması
- Açık kaynak olması nedeniyle geliştirilen programlama dilin kabul görmüş olması

Bu kriterler göz önüne alındığında, uygulamayı geliştirmek üzere Delphi 7.0 programlama dili ve bileşenleri seçilmiştir. Oracle nesneleriyle ile ilgili modüller nesnesel programlama ile yazılmıştır. Aşağıdaki bölümde yazılan modüllerin nesne ilişkileri gösterilmiştir.

### 5.2.2. Teknik Alt Yapı

DBAExplorer programında yazılım dili olarak Delphi 7.0 kullanılmıştır. Delphi, gerek VT iletişimi gerekse hızlı kod geliştirme açısından çok performanslı bir araç olarak kendini kanıtlamıştır. Delphi 7.0, en kısa zamanda en az hatayla uygulama geliştirme imkanı sağlamıştır. Aynı zamanda programcıların işini kolaylaştıracak özellikleri de bulunmaktadır. Bu özelliklerin bazıları aşağıda sıralanmıştır:

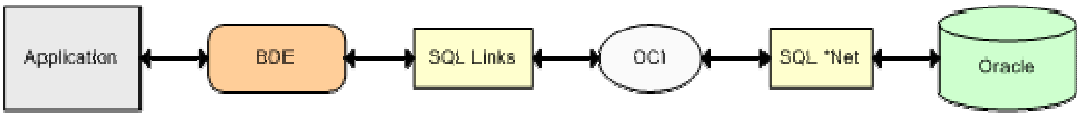
- Kod ve Tasarım sayfaları arasında kolay geçiş yapabilmesi

- Görsel tasarım yapabilmesi
- Anında söz dizimindeki hataları bildirmesi
- Kod yazarken aynı anda ekran tasarımlarının güncellenmesi
- Projede bulunan tüm dosyaların şablonunu gösterebilmesi
- Bütünleşmiş tasarım (build) ve derleme (compile) desteğinin olması

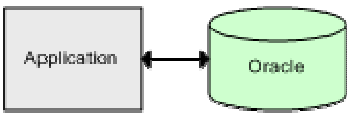
DBAExplorer'da VT erişimi için ODAC kullanılmıştır. Bu bileşenin kullanılmasındaki amaç; projenin kullanılacağı terminale Oracle istemci uygulamasını kurmadan direk bağlantıyı sağlamaktır. ODAC diğer bağlantı türlerine göre en yüksek performansı sunan bileşendir. ODAC ile Oracle'a bağlanmanın iki yolu vardır. İlki Oracle istemci kullanarak diğeri ise doğrudan TCP/IP üzerinden Oracle bağlantısıdır. Tez çalışmasında bu iki bağlantı şekli de desteklenmektedir.

Şekil 5.1 ve 5.2'de Delphi'nin çözümü olan Borland Database Engine (BDE) ile CoreLab firmasının çözümü olan ODAC arasındaki fark gösterilmiştir [3].

BDE ile Oracle VT'na ulaşmak için ara katmanlar kullanılmakta ve dolaylı bir bağlantı sağlanmaktadır. Aynı zamanda istemci makinede VT yazılımlarının kurulu olması gerekmektedir. ODAC ise TCP/IP protokolü kullanarak başka bir katman ve yazılım kullanmadan VT'na direk bağlana bilmektedir.




**Şekil 5.1.** Delphi ile Oracle bağlantısı.

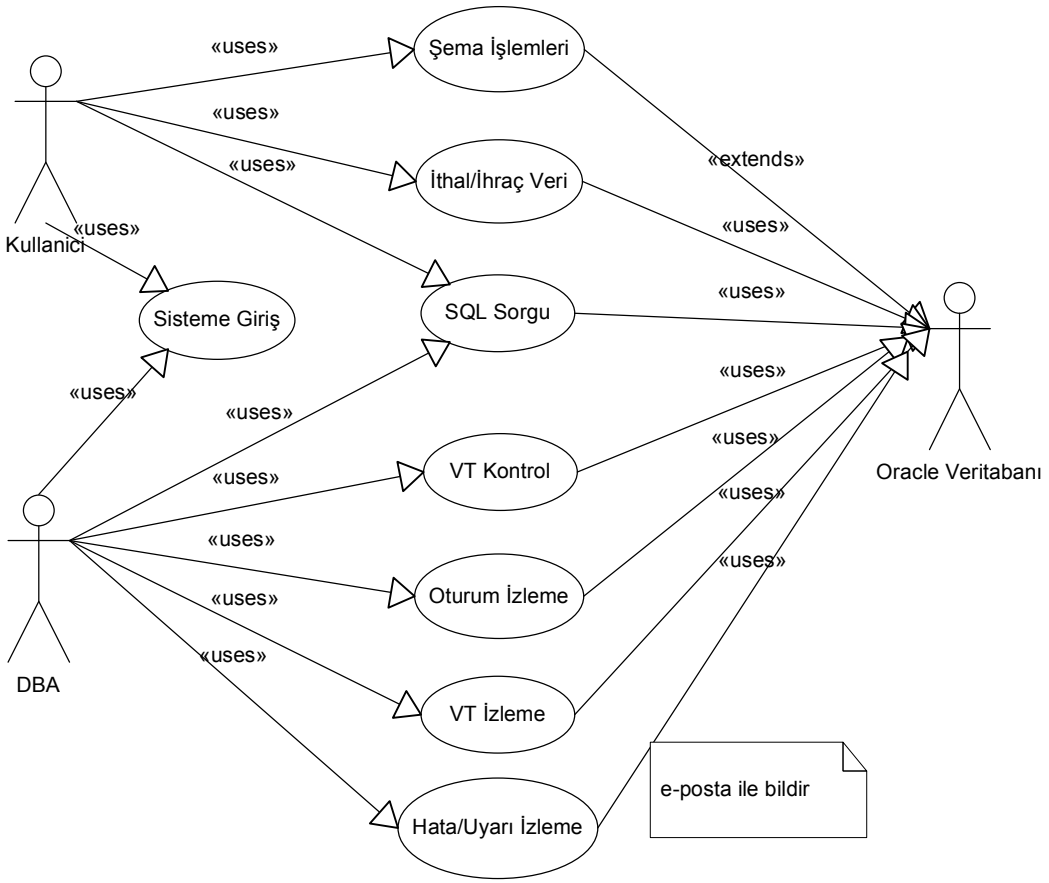



**Şekil 5.2.** Uygulamanın ODAC kullanarak VT'na bağlantısı.

### 5.2.3. Nesnesel İlişkiler

#### 5.2.3.1. Kullanım Durumu (Use Case)

Sistemin genel işleyişi ile ilgili kullanım durumu Şekil 5.3'de gösterilmiştir. Programın kullanışında "*Kullanıcı*" ve "*VTY*" diye iki profil vardır. Kullanıcı profili rutin VT işlemlerini (şema işlemleri, veri ithali-ihracı ve SQL sorgusu gibi işlemler) yapabilmektedir. VTY ise kullanıcı profilinin yaptığı işleri ve VT yönetim işlemlerini (VT kontrol, oturum izleme, yedek alma ve hataları izleme gibi) gerçekleştirmektedir. Bu iki profil VT üzerinden verilen haklar ve ayrıcalıklar ile belirlenmektedir. DBAExplorer'da hangi ekrana hangi kullanıcının gireceği kullanıcıya verilen bu haklar doğrultusunda sağlanmaktadır.

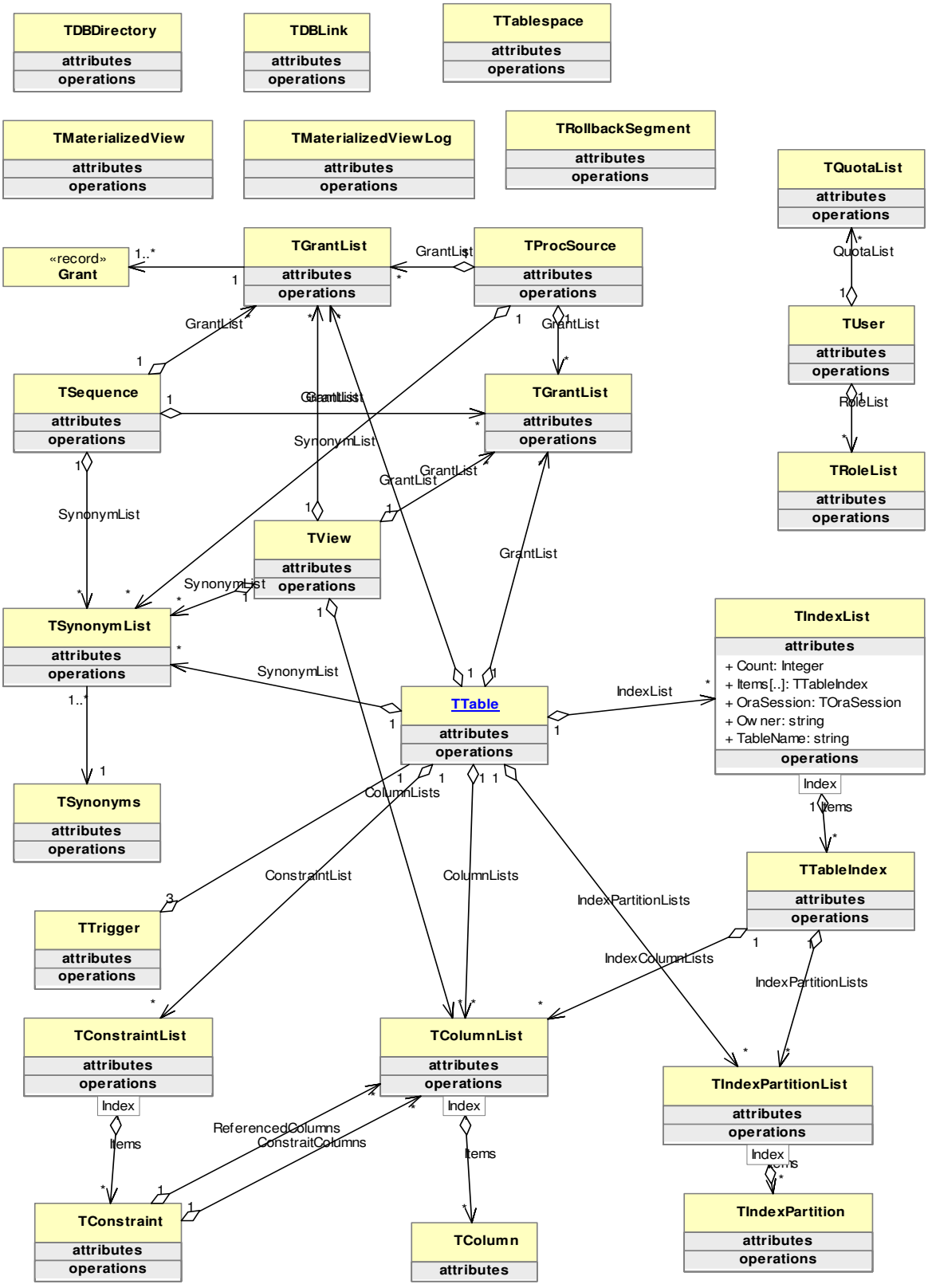



**Şekil 5.3.** DBAExplorer'ın genel kullanım durum diyagramı

#### 5.2.3.2. UML Diyagram

Şekil 5.4'de Oracle VT yapısının tutulduğu tablolar arasındaki ilişkiler gösterilmektedir. Bu tablolar arasındaki ilişkisi tez kapsamında geliştirilen programda oluşturulmuştur. Bu ilişkiler kullanılarak bir nesnenin SQL komut dizisi çıkartılabilmektedir. Örnek SQL komut dizisi tablolar için; tablonun yapısı, tabloya verilen haklar, tabloyu kullanan diğer nesneler ve tablonun fiziksel yeri gibi özelliklerdir.

Bir tablo nesnesinin (*owner, table_name, tablespace_name, status ve pct_free gibi*) özellikleri ve (*create, drop, alter ve getDdl gibi*) metotları vardır.
"*TTable*" nesnesinin SQL komut dizisini oluşturmak için "*getDDL*" metodu kullanılır. Bu metot görünüm listelerine, haklara, indislere, kolonlara ve kısıtlara bakarak istenilen tablonun SQL komut dizisini oluşturur.

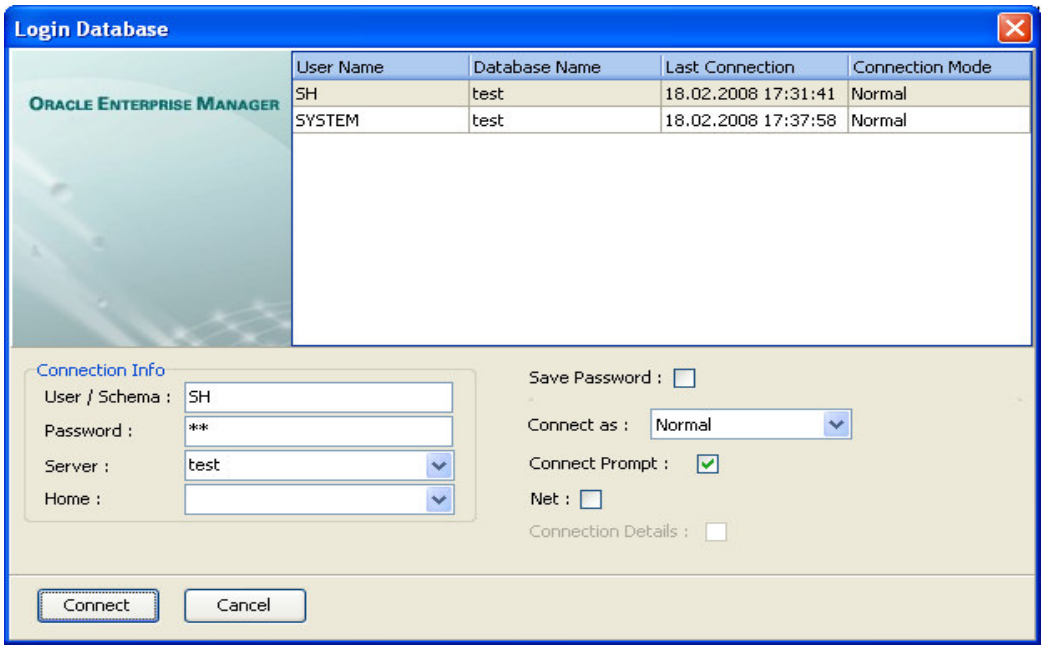



**Şekil 5.4.** VT nesneleri ile ilgili Delphi UML diyagramı

### 5.2.4. Kullanıcı Arayüzleri

#### 5.2.4.1. Sisteme Giriş

Sisteme giriş ekranı Oracle VTYA'na ilk giriş ekranıdır. Bu ekran kullanılarak hangi Oracle VT'na hangi Kullanıcı/Şifre ile girileceği belirtilir. Girilecek olan şifre ve kullanıcı adının VT'nında tanımlı olması gerekmektedir. Bir sonraki girişte daha önce girilen bilgiler otomatik liste şeklinde gelir. Oracle VT bağlantının doğrudan mı yoksa Oracle istemci üzerinden mi sağlanacağına burada karar verilir. Yeni bir VT bağlantısı oluşturulduğunda girilen şifre saklanabilir. Fakat bu şifre bilgisi kişisel bilgisayarda tutulacağı için bir güvenlik açığı oluşturacağı da unutulmamalıdır.

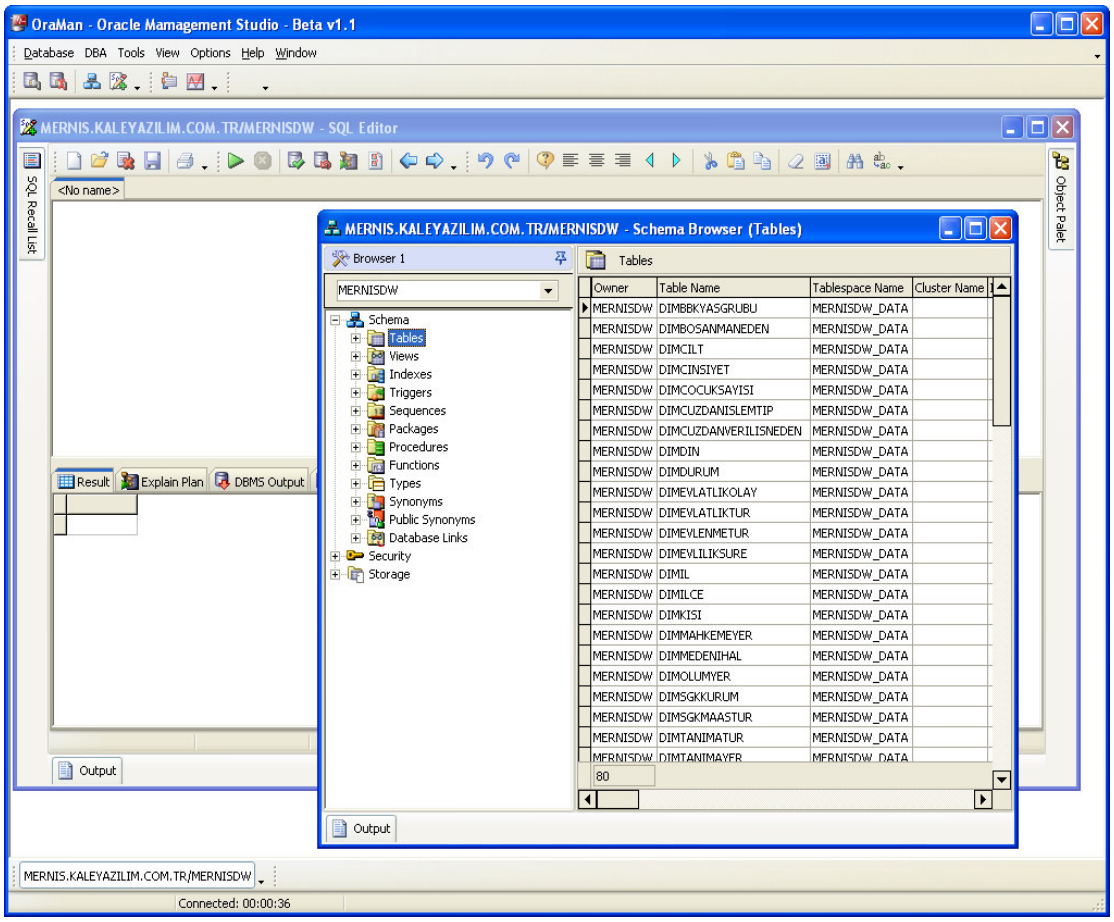



**Şekil 5.5.** Programa giriş ekran görünümü.

#### 5.2.4.2. Ana Ekran

Ana ekran DBAExplorer'un açılış ekranıdır. Bu ekran MDI yapısında olup açılan bütün pencereleri içerisinde gösterir. MDI yapısı iç içe ve birbirine benzer

pencerelerin açılmasını desteklemekte böylelikle gereksiz ekran kalabalığını ortadan kaldırmaktadır. Bu ekran birden fazla VT'na aynı anda bağlanmayı desteklemektedir. Bağlanılan VT isimleri ve ekranları bu ekranın altında görünmektedir. Hangi VT bağlantısı seçilmiş ise bundan sonra yapılacak olan işlemler seçilen oturumda geçerli olacaktır. Bir VT bağlantısının kapatılması ile bu bağlantıya sahip olan bütün ekranlar otomatik olarak kapatılmaktadır. Şekil 5.6'da ana ekran ve alt ekran görünümü gösterilmektedir.




**Şekil 5.6.** Program ana ekran görünümü.

### 5.2.4.3. Şema İzleyici

Şema İzleyici VT'na bağlandıktan sonra nesnelerin izlendiği ve nesneler ile ilgili her türlü işlemlerin yapıldığı ekrandır. Şema İzleyicisi ile verilen haklar doğrultusunda, diğer kullanıcıların nesneleri görülebilir ve müdahale edilebilir. Aynı oturuma ait

veya farklı oturumlar için birden fazla Şema İzleyicisi açılabilmektedir. Şema İzleyicisi ile bir nesneye tıkladığınızda ilgili nesnenin diğer özellikleri de görülebilir. Örneğin bir Tablo seçildiğinde, o tablonun tanımlamaları, kolonları, indisleri, tetikleyicileri, kısıtları, verilen haklar ve daha fazlası aynı ekranda görülebilir ve SQL komut dizileri çıkartılabilir. Şema İzleyici iki bölümden oluşmaktadır. Sol tarafta ağaç yapısı, ekranın sağında ise nesne özellikleri bulunmaktadır. Ekranın sağında, ana nesneye basıldığında alt nesne listesi, alt nesneye basıldığında ise sadece o nesnenin özelliği ve aktif/pasif durumları gelmektedir. Ekranın solunda bulunan nesne listesine farenin sağ tuşu ile tıklandığında, seçilen nesne ile ilgili hangi işlemlerin yapılabileceği görünmektedir. Şema İzleyici üzerinde işlem yapılırken oluşan her türlü hata ekranın altında ayrı bir pencerede takip edilebilmektedir. Şekil 5.7 ve 5.8'de şema izleyicisi ekran görünümleri gösterilmiştir.

Ayrıca bu ekranda aşağıdaki işlemler yapılabilmektedir.

- Tablo işlemleri:
    - Tablo oluşturma, değiştirme, silme işlemleri
    - Tablo verileri için; arama, silme, ekleme, filtreleme işlemleri
    - Kolon işlemleri (ekleme, silme)
    - İndeks işlemleri (ekleme, silme, analiz ve tekrar derleme)
    - Anahtar işlemleri (ekleme, silme, aktif/pasif yapma)
    - Tetikleyici işlemleri (ekleme, silme, aktif/pasif yapma)
    - Hak işlemleri (hak verme ve alma)
    - Eşanlam işlemleri (ekleme, silme)
    - Bölüntü (partition) işlemleri (ekleme, silme)
    - DDL oluşturma

- Görüntü işlemleri:
    - Görüntü oluşturma, değiştirme, silme işlemleri
    - Görüntü verileri için; arama, filtreleme işlemleri,
    - Tetikleyici işlemleri (ekleme, silme, aktif/pasif yapma)
    - Hak işlemleri (hak verme ve alma)
    - Eşanlam işlemleri (ekleme, silme)

    - Kullanan ve kullanılan nesneler listesi
- İndis işlemleri:
    - İndis oluşturma, değiştirme, silme işlemleri
    - Analiz işlemleri
    - Tekrar derleme işlemleri
    - Birleştirme (coalesce) işlemleri

- Tetikleyici işlemleri:
    - Tetikleyici oluşturma, değiştirme, silme işlemleri
    - Derleme işlemleri
    - Aktif/Pasif yapma işlemleri

- Sıra işlemleri:
    - Sıra oluşturma, değiştirme, silme işlemleri
    - Hak işlemleri (hak verme ve alma)
    - Eşanlam işlemleri (ekleme, silme)

- Paket işlemleri
    - Paket oluşturma, değiştirme, silme işlemleri
    - Hak işlemleri (hak verme ve alma)
    - Eşanlam işlemleri (ekleme, silme)
    - Kullanan ve kullanılan nesneler listesi
    - Derleme (compile) işlemleri

- Fonksiyon işlemleri:
    - Fonksiyon oluşturma, değiştirme, silme işlemleri
    - Hak işlemleri (hak verme ve alma)
    - Eşanlam işlemleri (ekleme, silme)
    - Kullanan ve kullanılan nesneler listesi
    - Derleme (compile) işlemleri

- Tip işlemleri:
    - Tip oluşturma, değiştirme, silme işlemleri
    - Hak işlemleri (hak verme ve alma)
    - Eşanlam işlemleri (ekleme, silme)
    - Kullanan ve kullanılan nesneler listesi
    - Derleme (compile) işlemleri

- Eşanlam işlemleri:
    - Eşanlam oluşturma, değiştirme, silme işlemleri

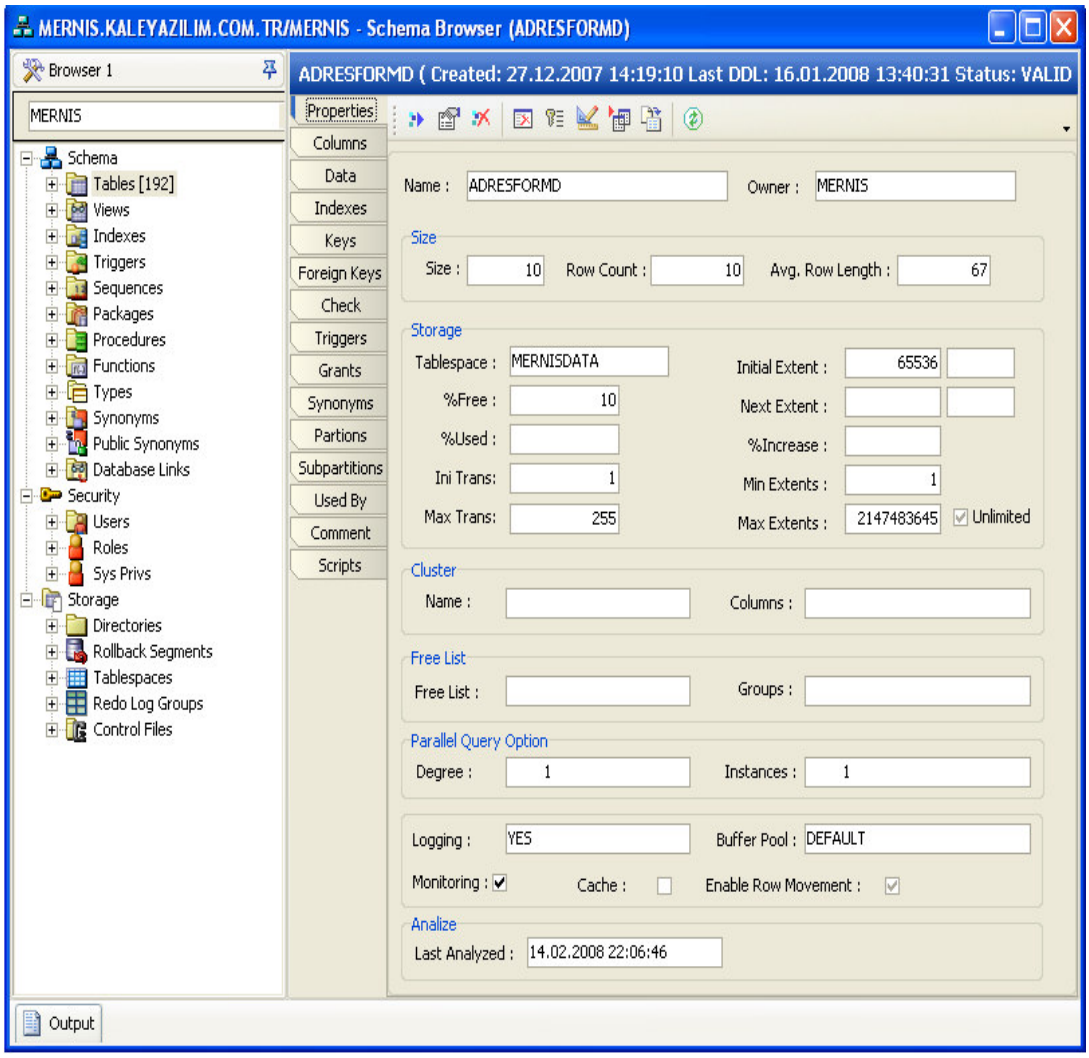
- VT Link işlemleri:
    - Link oluşturma, değiştirme, silme işlemleri

- Kullanıcı ve Rol işlemleri:
    - Kullanıcı oluşturma, değiştirme, silme işlemleri
    - Rol işlemleri (verme, alma)
    - Sistem ayrıcalığı (verme, alma)
    - Kota (verme, alma)
    - Şifre değiştirme işlemleri

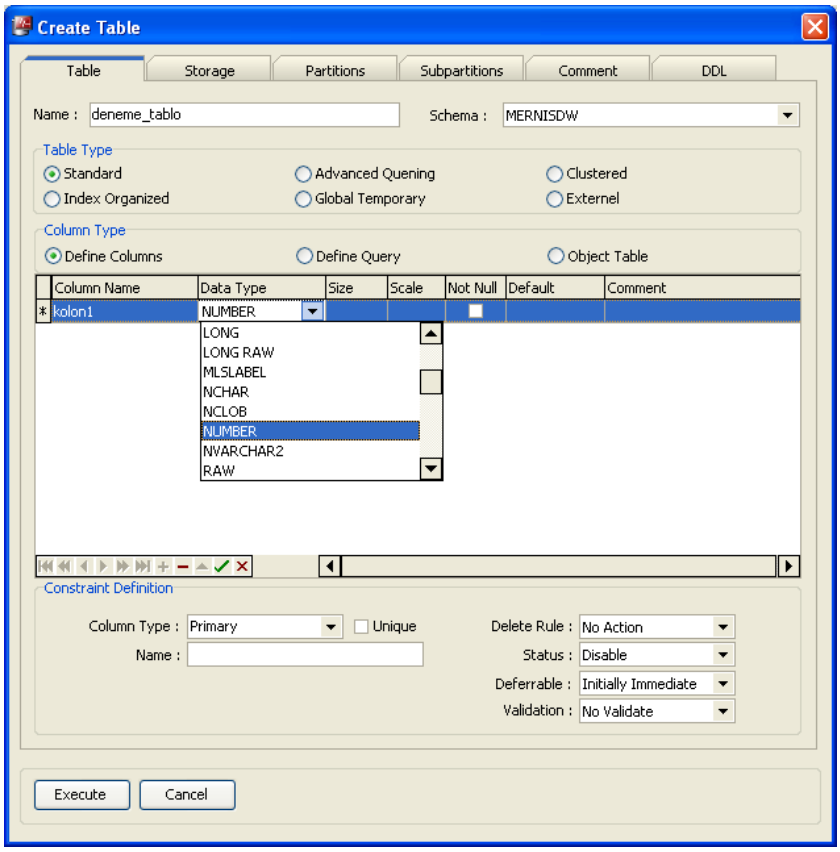
- Dizin işlemleri:
    - Dizin oluşturma, değiştirme, silme işlemleri

- Geri Alma Parçası işlemleri:
    - Geri alma parçası oluşturma, değiştirme, silme işlemleri

- Tablo Uzayı işlemleri:
    - Tablo boşluğu oluşturma, değiştirme, silme işlemleri



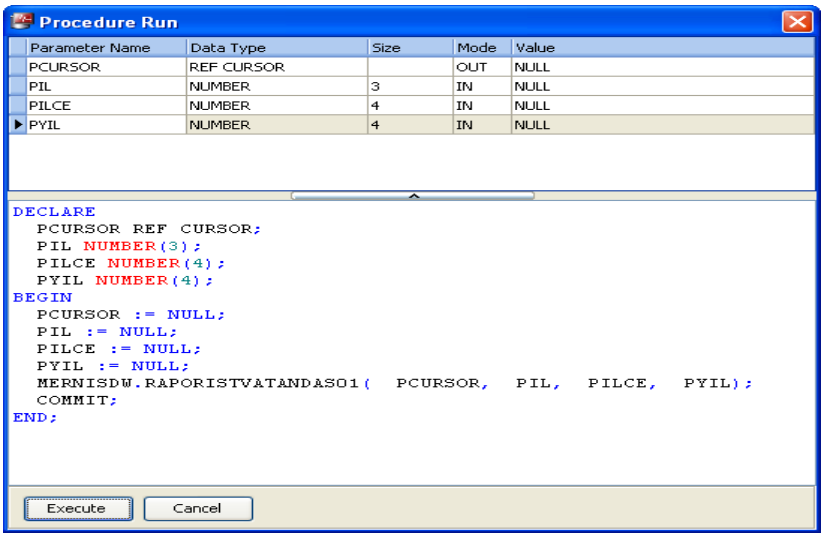

**Şekil 5.7.** Şema İzleyicisi ekran görünümü.




**Şekil 5.8.** Şema İzleyicisi üzerinden yeni bir tablo oluşturma ekran görünümü.

### 5.2.4.4. PL/SQL Prosedürleri, Fonksiyon ve Paket Çalıştırma

Prosedür, fonksiyon ve paketler, şema izleyicisi üzerinde ilgili nesneye sağ fare ile tıklayarak kolayca çalıştırılabilir. Açılan pencereye seçilen nesnenin parametreleri girilebilir. Bir prosedür çalıştırılmadan önce otomatik olarak derlenir ve hata var ise çalıştırılmaz. Çalıştırılan nesne arka planda çalışmaya devam eder ve işlem sonuçlandığında kullanıcıya gösterilir. Şekil 5.9'da prosedür çalıştırma ekranı gösterilmiştir.




**Şekil 5.9.** Ekran görünümü.

### 5.2.4.5. SQL Editörü

SQL Editörü VT sorgularının yazılabildiği ve test edilebildiği SQL çalışma ortamıdır. SQL editörü SQL, PL/SQL ve SQL*Plus komutlarının yazılmasını destekler. VT'na bağlandıktan sonra seçilen şemaya göre; yeni bir tablo oluşturma, tabloya kayıt ekleme, tetikleyici oluşturup düzenleme ve tablodan veri çekme gibi işlemler yapılabilir. VT sorgulama dili komutlarını renklendirme ve otomatik tamamlama özelliği ile kullanıcıların işlerini oldukça kolaylaştırır. Otomatik SQL üretme, SQL ifadesi yazarken akıllı editör yardımı ve çalıştırılan sorguların sıralanabilen sonuçlarını tablo ile göstermesi gibi özellikleri de vardır. Çalıştırılan SQL sonuçları, çalışma zamanı, alınan hatalar gibi bilgiler aynı ekranda gösterilmektedir. Şekil 5.10'da SQL editör görünümü gösterilmiştir.

Çalışma Planı tuşuna basılarak aktif olan SQL cümlesinin çalışma planı görülebilir. Çalışma planı, SQL performans düzenlemeleri için sık kullanılan bir özelliktir. Şekil 5.11'de çalışma planı örnek görünümü gösterilmiştir. Sürükle ve bırak teknolojisi ile tablo ve görünümler için daha hızlı kod yazılması sağlanır.

Çalıştırılan bütün komutların geçmişi kişisel bilgisayarlarda saklanabilir, içerisinde arama yapılabilir ve tekrar çağrılabilir.

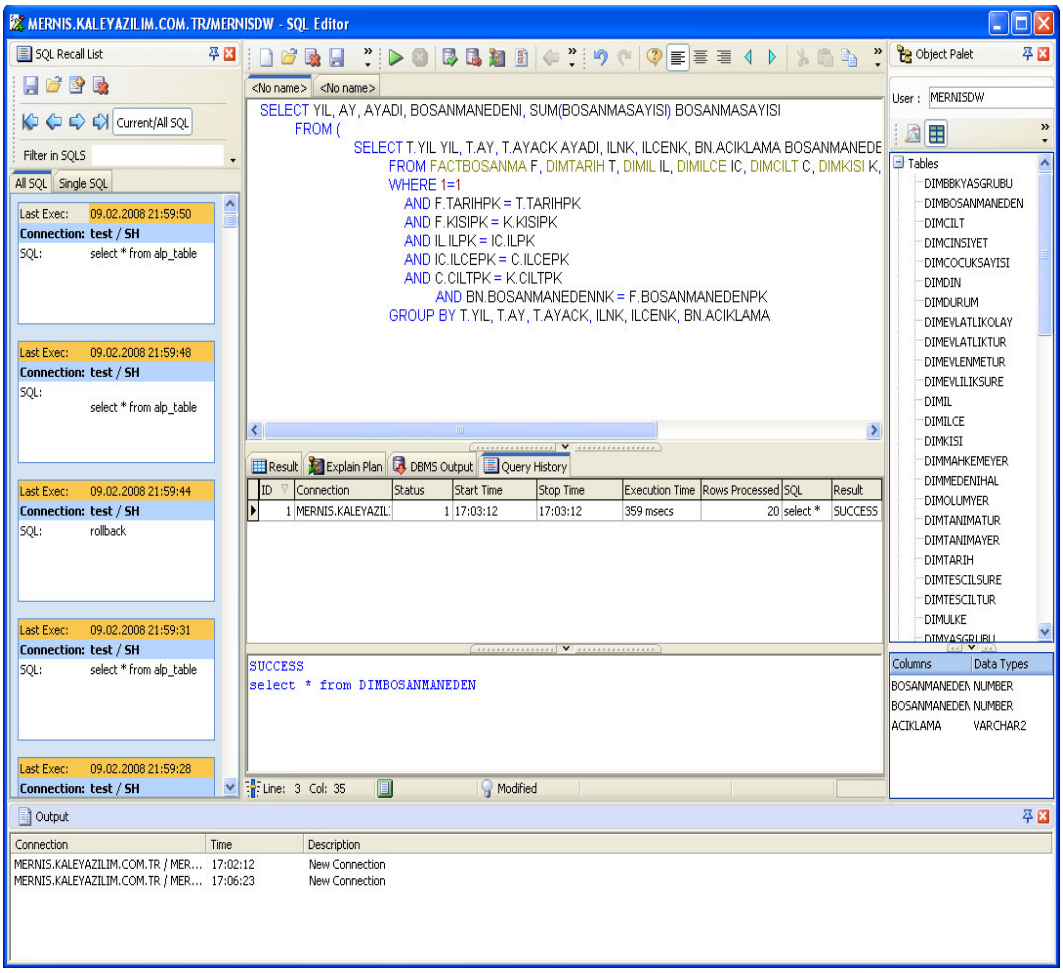
Çalıştırılan SQL hataların hangi satır ve kolonda olduğu bilgisi kullanıcıya gösterilmektedir. Alınan hata bilgilerinin tarihçesi detaylı bir şekilde tutulabilmektedir.

Bir SQL editöründe birden fazla çalışma sayfası açılabilmektedir. Editörde çalıştırılan SQL'ler arka planda is parçası (thread) olarak çalıştığından aynı anda bir den fazla SQL cümlesi çalıştırılabilir. Böylelikle kullanıcı bir işlemin bitmesini beklemek zorunda kalmamaktadır.

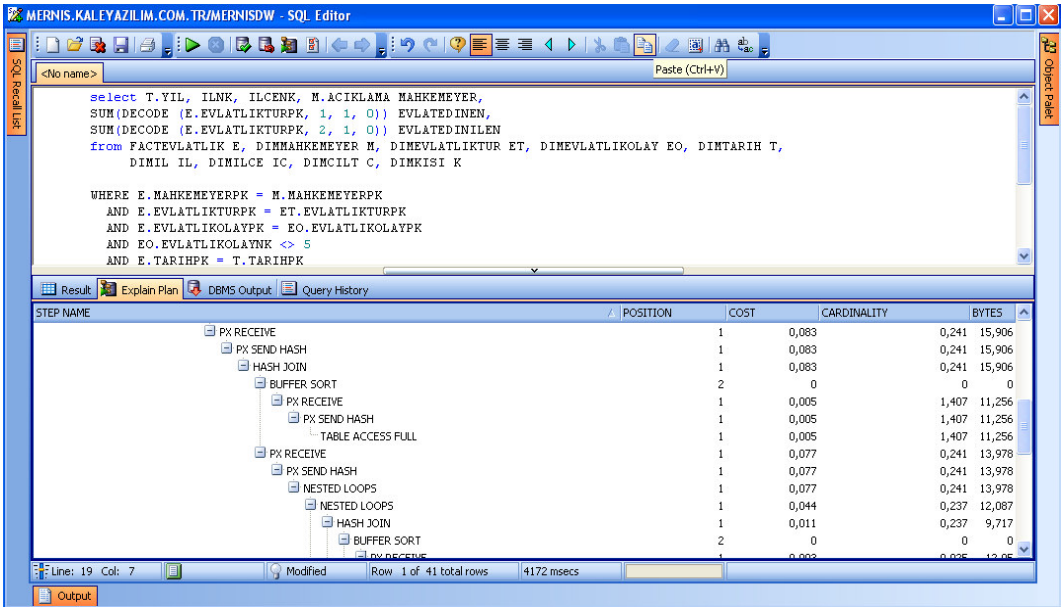
SQL editörü bir yazı editörünün sahip olması gereken minimum özelliklere sahiptir. Bu editör ile sağa sola dayama, kopyalama yapıştırma, hizalama ve arama değiştirme gibi işlemler oldukça kolaylıkla yapılabilmektedir.

Ayrıca SQL Editörü ile aşağıdaki işlemler yapılabilmektedir:

- Sorgu sonucunun görülmesi
- Geçmişe yönelik sorguların izlenmesi ve saklanması
- Sorgu sonucunun zamansal ölçümü
- Sorgu sonucu alınan hataların detayı
- Çalışma Planı sonuçları
- DBMS çıktısı
- Birden fazla editör açma
- Yazılan sorguyu dosyaya saklama veya dosyadan açma
- Commit/Rollback işlemleri
- SQL sorgularının hizalanması
- Sürükle bırak ile çalışan tablo ve görüntülerin listesi ve bunların alanları
- Kelime arama ve değiştirme




**Şekil 5.10.** SQL Editörü ana ekran görünümü.




**Şekil 5.11.** SQL Editörü içerisinden Çalışma Planı gösterim şekli.

#### 5.2.4.6. Oturum Bilgisi İzleme

Oturum bilgisi izleme ekranı VT'na bağlı oturumların izlendiği ve bu oturumlara müdahale edilmesini sağlayan ekrandır. Bu ekranı kullanabilmek için yönetici seviyesinde yetki gerekmektedir.

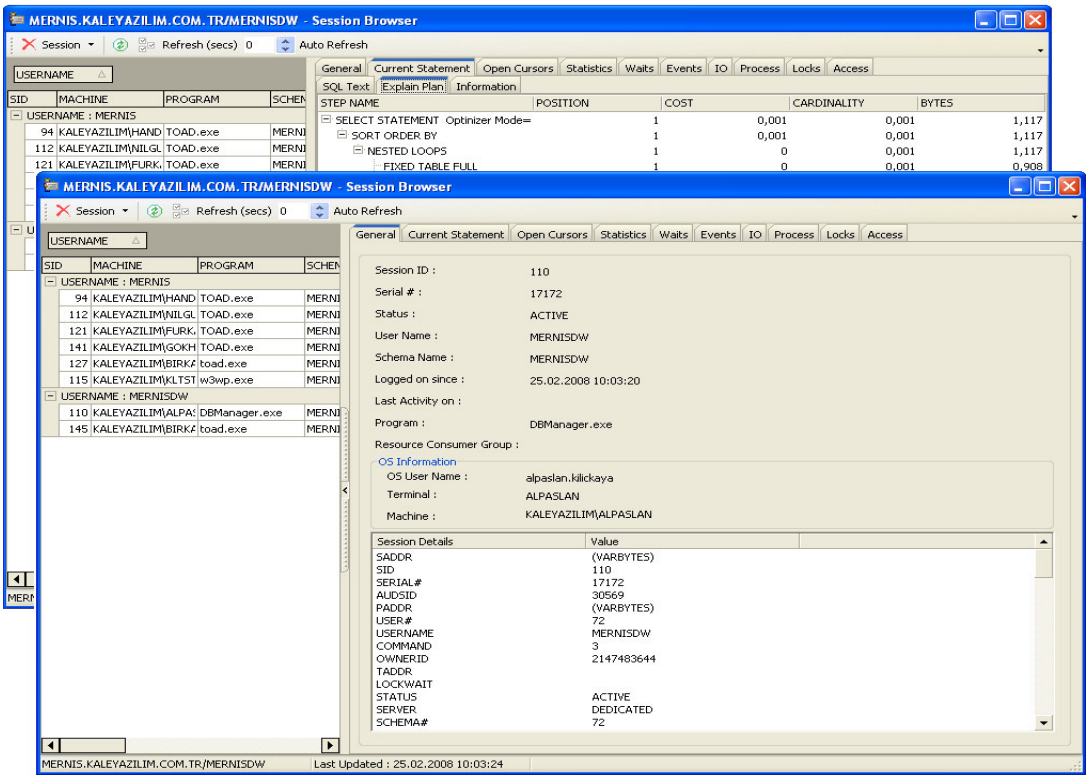
Oracle'a bağlı kullanıcıların hangi SQL cümlelerini çalıştırdığını, kullanıcıların aktif SQL cümlesini ve bunların çalışma planlarının gösterilmesini sağlar. Aynı zamanda Oracle'a bağlı kullanıcıların oturumunun kapatılmasını veya kilitlerinin kaldırılmasını sağlar.

Zaman bilgisi vererek bu ekranın belirli periyotlarda yenilenmesi sağlanabilir. Böylelikle kullanıcıların o anki işlemleri canlı olarak izlenebilir.

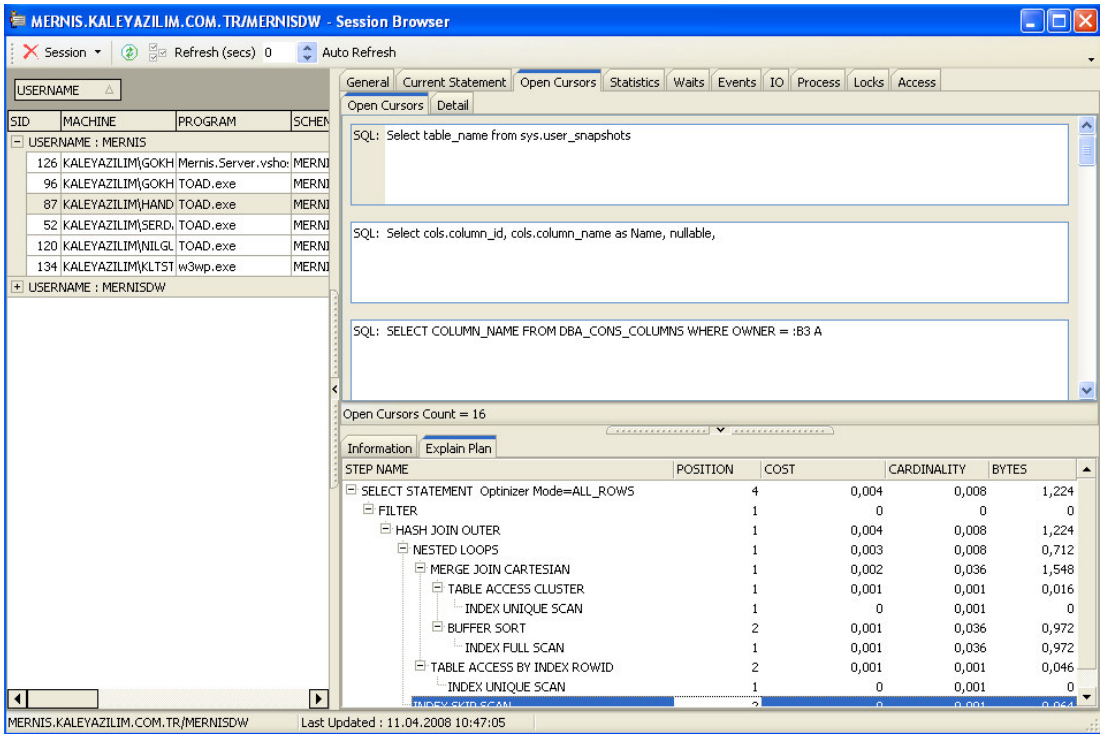
Ayrıca Oturum Bilgisi İzleme ile aşağıdaki işlemler yapılabilmektedir:

- Aktif oturumların izlenmesi
- Aktif oturumların çalıştırdığı SQL cümlelerini ve bu cümlelerin çalışma plan bilgilerinin izlenmesi
- Seçili oturuma ait, açık imleç bilgileri ve bu imleçlere ait SQL bilgilerinin izlenmesi
- Seçili oturumun istatistik bilgisinin alınması
- Seçili oturumun bekleme olaylarının izlenmesi
- Seçili oturumun G/Ç bilgilerinin izlenmesi
- Seçili oturumun aktif görevlerinin izlenmesi
- Seçili oturumun kilit koyduğu nesnelerin izlenmesi ve kilitlerin kaldırılması
- Seçili oturumun ulaştığı nesnelerin izlenmesi

Şekil 5.12 ve 5.13'de oturum izleyici için örnek ekranlar gösterilmiştir.




**Şekil 5.12.** Oturum Bilgisi İzleme ekran görünümü.

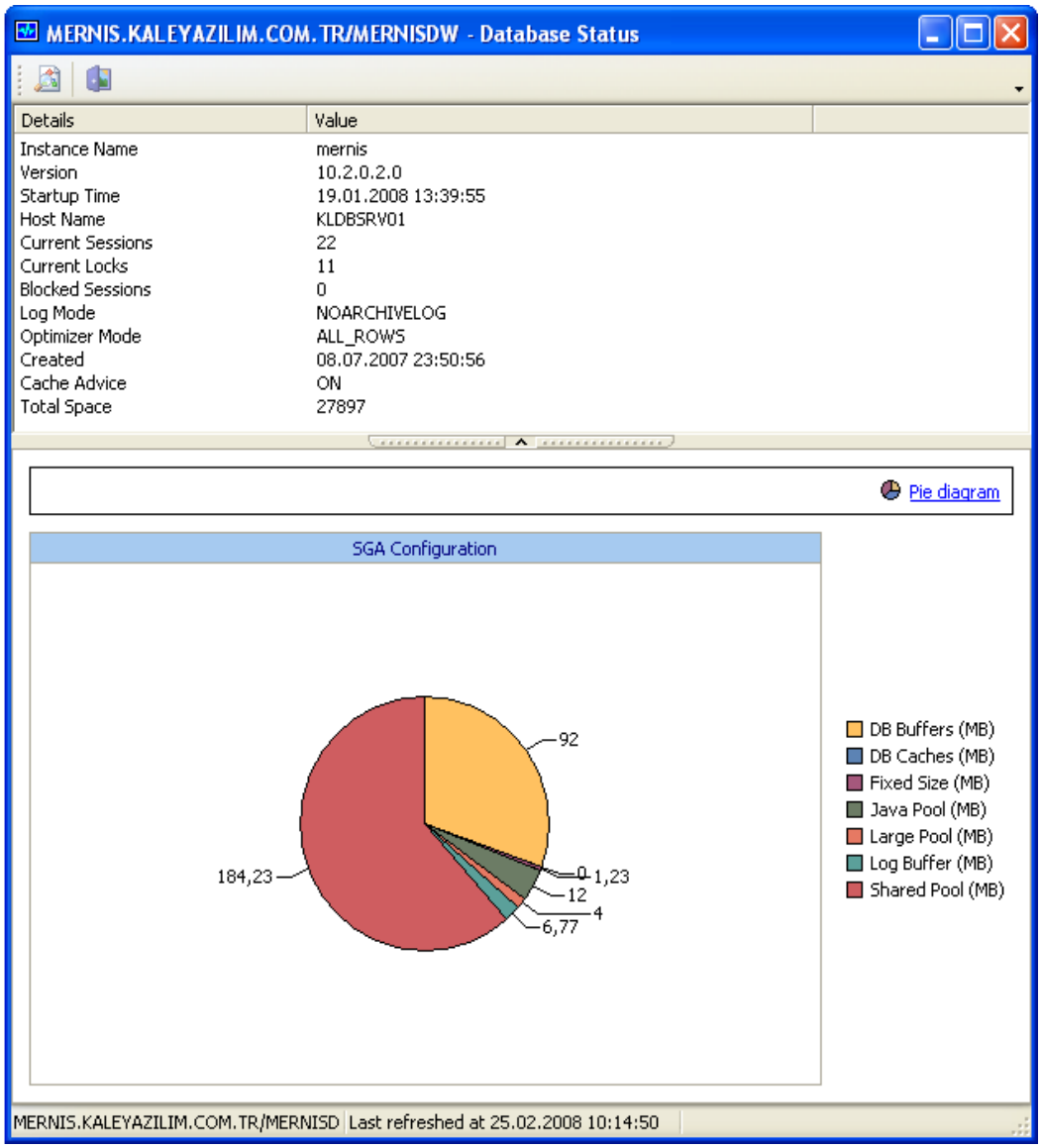



**Şekil 5.13.** Oturum İzleyici içerisinden Çalışma Planı gösterim şekli.

### 5.2.4.7. VT Durum İzleme

Durum izleme ekranı VT'nın aktif durum bilgisini gösteren ekrandır. Bu ekranı kullanabilmek için Yönetici seviyesinde yetkiye ihtiyaç vardır. Bu ekran VT oturumu başladığından kapatılana kadar olan SGA bilgisini gösterir. Bu ekranda herhangi bir müdahale söz konusu değildir. Şekil 5.14'de VT durum izleme ekranı gösterilmiştir.

Daha detaylı bilgi Bölüm 2.7.1 Sistem Global Alanı (SGA)'de yer almaktadır.




**Şekil 5.14.** VT durumu izleme ekran görünümü.

#### 5.2.4.8. VT Hata ve Uyarı İzleme

Hata ve uyarı izleme ekranı VT yöneticileri tarafından, VT durumunu izlemek amacıyla oluşturulan SQL cümlelerinin sonuçlarını, belirlenen periyotlarda listeleyen ve hata oluşması durumunda e-posta gönderen ekrandır. Bu ekranda VT ile ilgili durum ve hata bilgilerini veren SQL'ler tanınmakta ve bu SQL cümlelerinin sonucu dinamik olarak ele alınabilmektedir. Bu ekranı kullanabilmek için Yönetici seviyesinde yetkiye ihtiyaç vardır.

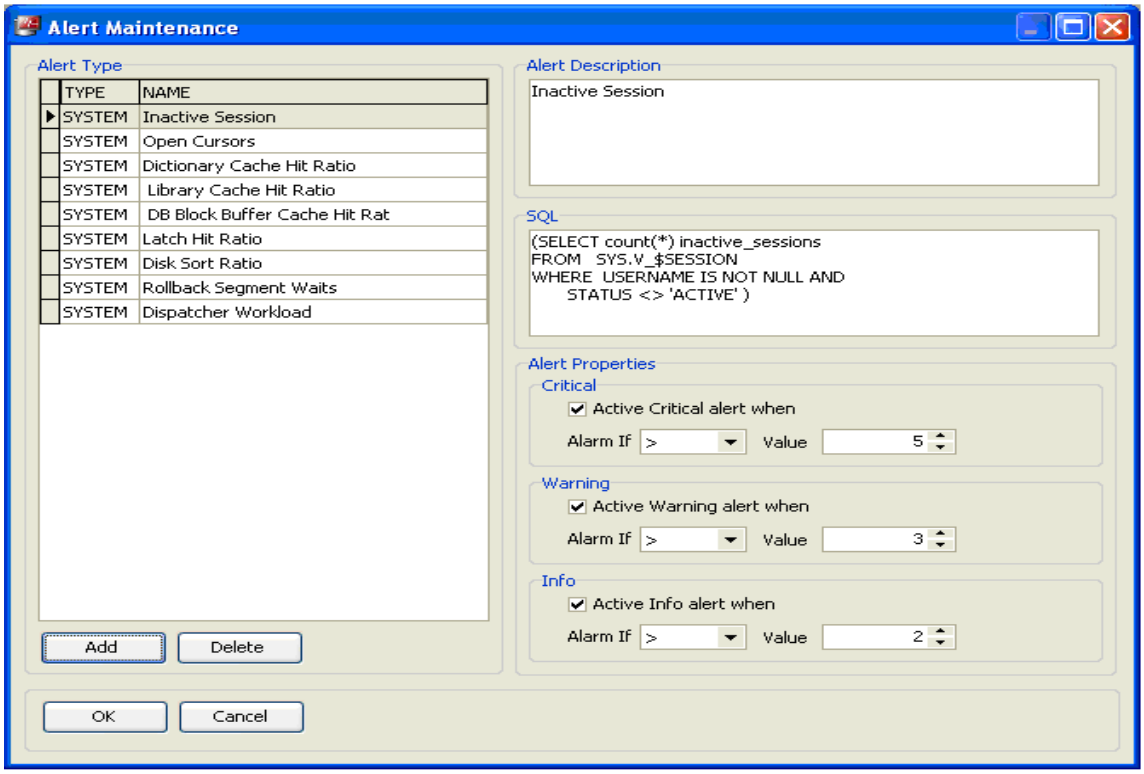
SQL cümlesi tanımlanırken tek satır ve tek kolon döndürmesi gerekmektedir. Dönen sonucun büyük / küçük değer karşılaştırması yapılabilmekte ve alınan sonuca göre değişik renkte gösterilmesi sağlanmaktadır. Böylelikle alınan sonucun kritik, uyarı veya bilgi amaçlı olup olmadığı anlaşılmaktadır.

Her VT farklı kurulum ve donanıma sahip olduğu için yazılacak olan SQL cümlelerinin sonuçları, bu durumlar dikkate alınarak ayarlanmalıdır. Alınan hata veya durum sonuçları Excel dosyası olarak yedeklenebilmektedir.

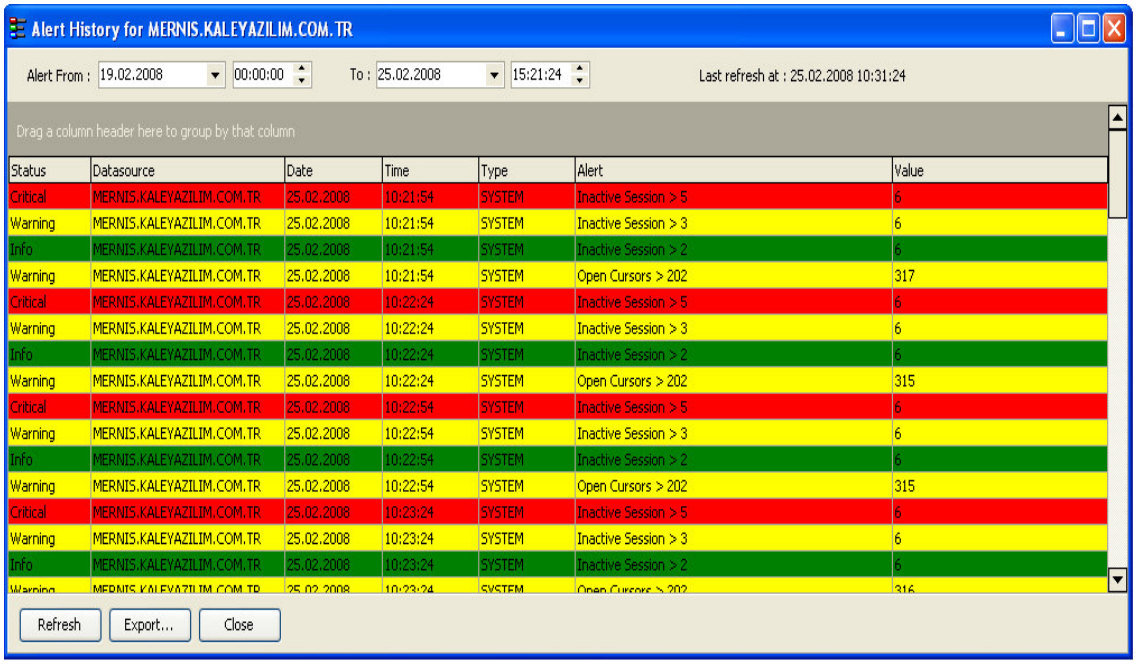
Bu tez kapsamında araştırılan ve geliştirilen performansa yönelik SQL cümleleri EK-B'de verilmiştir.

Zaman bilgisi vererek bu ekranın belirli periyotlarda yenilenmesi sağlanabilir. Böylelikle VT'nın o anki durumu canlı olarak izlenebilir.

Şekil 5.15 ve 5.16'da hata tanımlama ekranı ve hata izleme ekranı gösterilmiştir.




**Şekil 5.15.** Hata ve durum için SQL tanımlama ekran görünümü.




**Şekil 5.16.** Oluşan hataları izleme ekran görünümü.

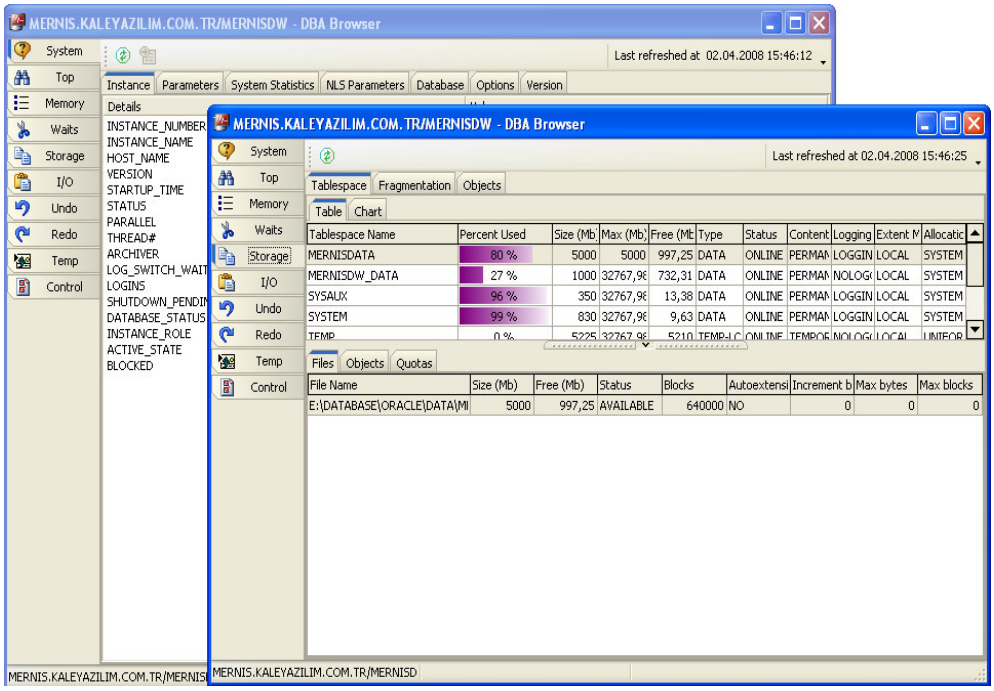
#### 5.2.4.9. VT Yöneticisi Ekranı

Yönetici ekranı ile VT tanımları, durumu ve gidişatı hakkında bilgiler alınabilmektedir. Bu ekranın amacı VT'nın şu anki durumunu incelemek ve oluşabilecek hataları önceden saptamaya çalışmaktır. Ekranda gösterilen bilgiler görsel öğelerle zenginleştirilmiştir. Böylelikle durumların karşılaştırılması ve anlaşılması daha kolay hale getirilmiştir. Bu ekrana sadece yönetici hakkı olan kullanıcılar bağlanabilir. Şekil 5.17, 5.18 ve 5.19'da VT yönetici ekran örnekleri gösterilmiştir.

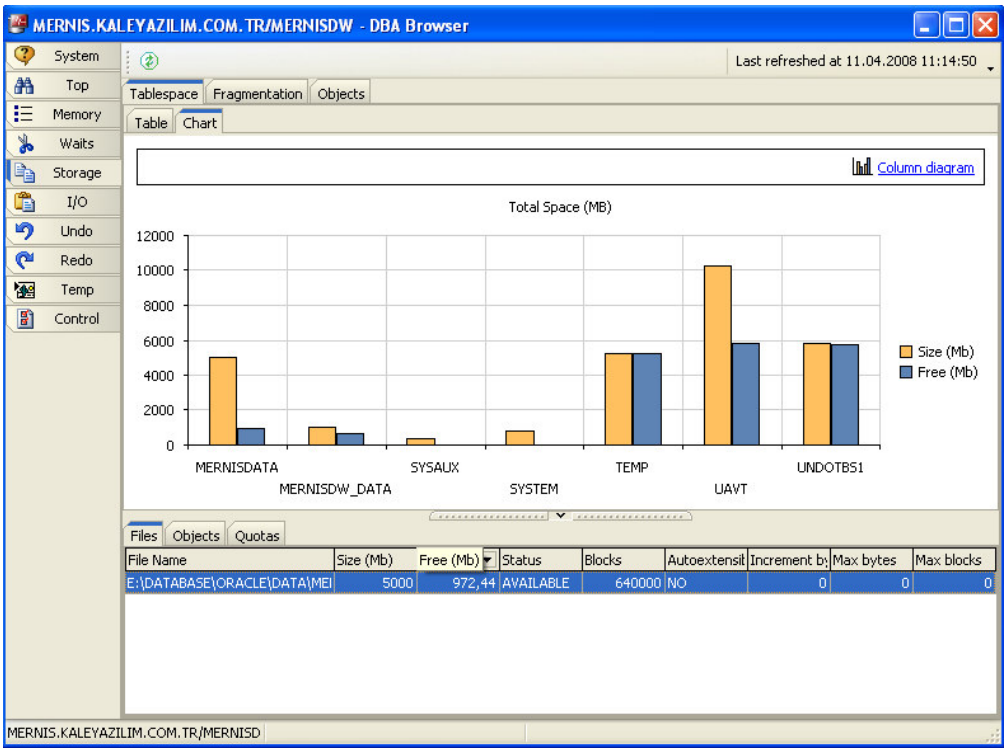
Bu ekranda aşağıdaki durum bilgileri incelenebilmektedir:

- Sistem Bilgisi İşlemleri
  - Instance, Parameters, System Statistic, NLS Parameters, Database, Options, Version,
- Top Session, Top SQL, Top Instance,
- Bellek Bilgileri
  - Buffer Pools, Buffer Wait Stats, Shared Pool Advice, PGA Advice, PGA Stats, Data Cache, Advice,
- Beklemeler
  - Events, Locks, Latches, Child Latches, Latch Holder,
- Saklama Bilgileri
  - Tablespace, Fragmentation, Objects,
- Veri Dosyaları
  - Data Files, SGA Stats,
- Geri Alma Bilgileri
  - Tablespace, Undo Stats,
- Redo Bilgileri
  - Groups, Log Switch History, Archived Logs,
- Geçici Dosyalar Bilgisi
  - Temp Files,

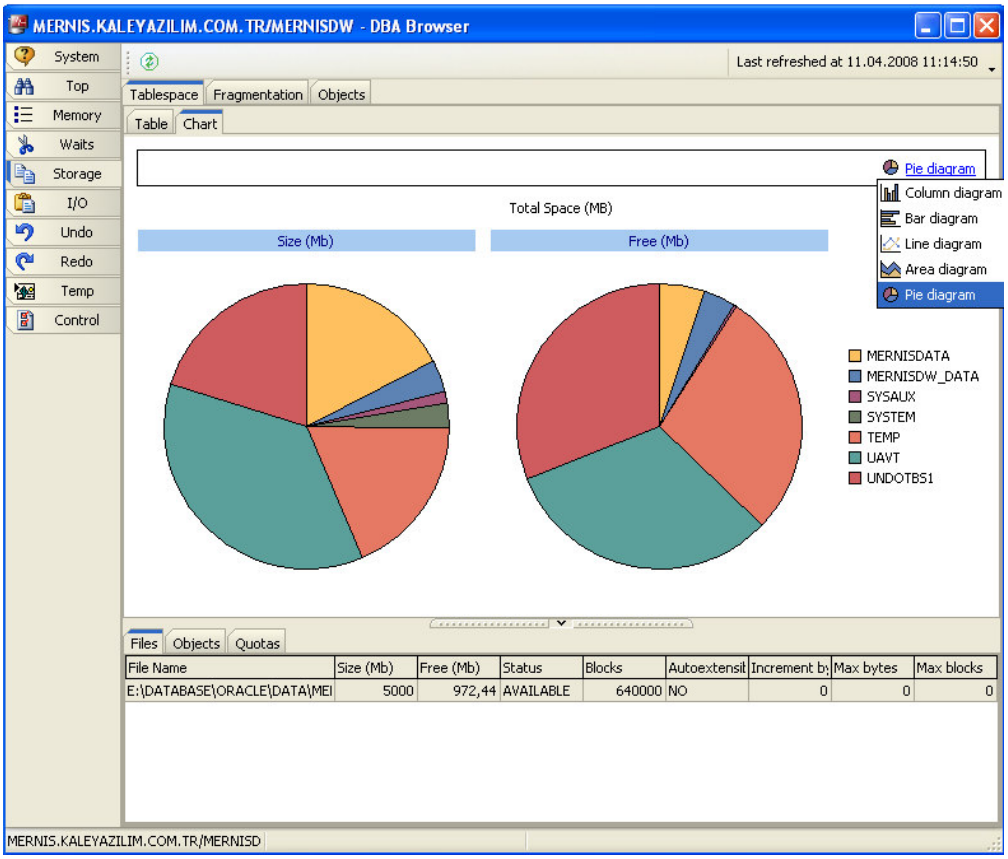
- Kontrol Dosyaları Bilgisi
    - Control Files.




**Şekil 5.17.** VT yöneticisi ekran görünümü



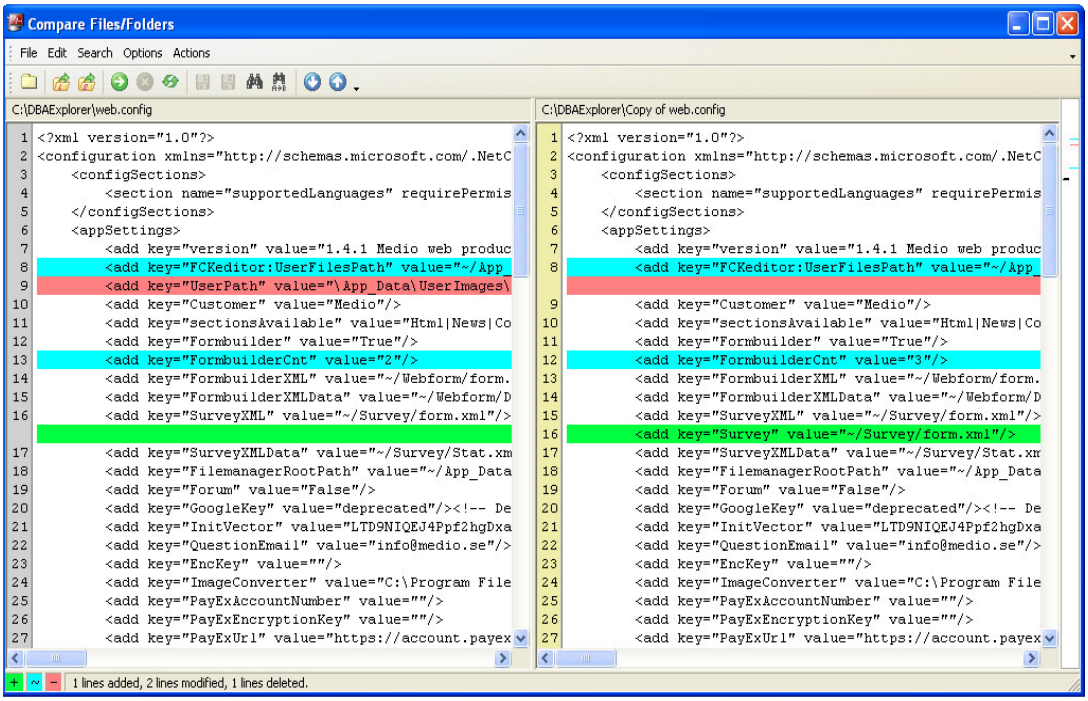

**Şekil 5.18.** VT yöneticisi grafiksel ekran görünümü




**Şekil 5.19.** Elde edilen veriler değişik şekillerde gösterilebilmektedir.

### 5.2.4.10. İki Dosya Karşılaştırma Ekranı

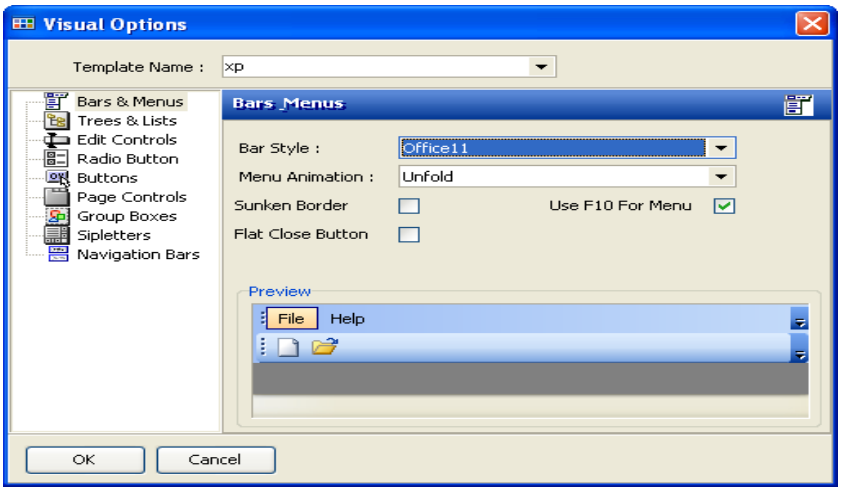
Dosya karşılaştırma ekranında, verilen iki dosya veya dizin arasındaki farklılıklar renkli olarak gösterilmektedir. Yeni eklenen satırlar yeşil, silinen satırlar kırmızı, değişen satırlar ise mavi olarak gösterilmektedir. SQL cümleleri, xml ve text dosyaları gibi farklı dosya türleri karşılaştırılabilmektedir. İstenildiğinde sadece fark bilgileri gösterilebilmekte aynı zamanda dosyalar üzerinde arama ve değiştirme işlemleri de yapılabilmektedir. Şekil 5.20'de örnek karşılaştırma ekranı görünümü gösterilmiştir.




**Şekil 5.20.** Dosya karşılaştırma ekran görünümü.

### 5.2.4.11. Görsel Seçenekler

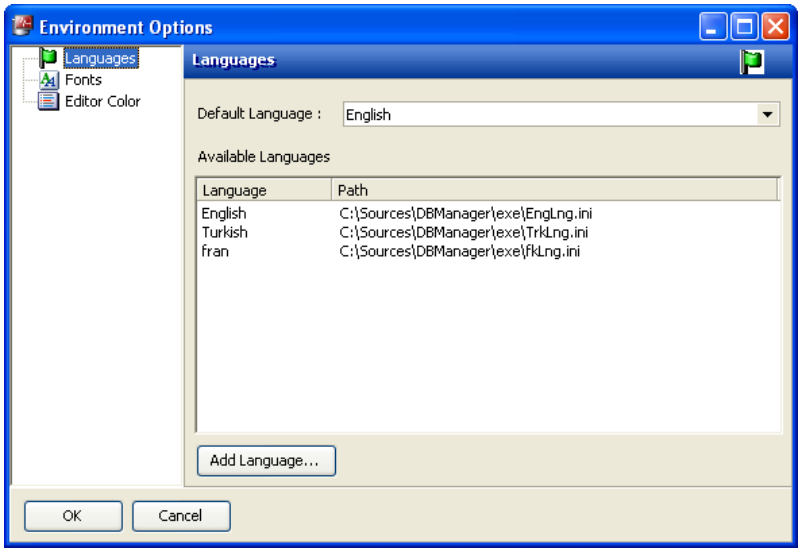
DBAExplorer da kullanılan ekran görüntülerini kullanıcılar istediği gibi değiştirilebilmektedir. Oluşturulan yeni görüntü saklanmakta ve program her çalıştığında bu görüntüler kullanılmaktadır. Böylelikle kullanıcılara görsel özgürlük tanınmış, alışmış oldukları ekran görüntüsü ile çalışma olanağı sunulmuştur. Şekil 5.21'de görsel seçenekler ekranı örneği gösterilmiştir.




**Şekil 5.21** Görsel Seçenekler ekran görünümü.

#### 5.2.4.12. Çoklu Dil Desteği

Çoklu dil desteği ekranı ile DBAExplorer'a birden fazla dil tanımı yapılabilmektedir. İsteğe bağlı olarak yeni diller de kullanıcılar tarafından tanımlanabilmektedir. DBAExplorer üzerinde hangi dil ile çalışılması gerektiği belirtilebilmektedir. Şekil 5.22'de çoklu dil tanımlama ekran görünümü gösterilmiştir.




**Şekil 5.22.** Çoklu dil tanımlama ekran görünümü.

### 5.3. VTYA Özellikleri

Nesneye yönelik programlama tekniklerindeki gelişmeyle, yazılım geliştirme araçlarının tümü, bileşen tabanlı kullanım ortamı sunmuştur. Bileşen oluşturma yazılım kodlama sürecini azalttığı gibi programlara işlevsellik, değişkenlik ve esneklik kazandırmaktadır. DBAExplorer'da nesneye yönelik programlama tekniği kullanılarak, VT bağlantısı ve üzerinde işlem yapma olanağı esnek hale getirilmiştir. Kullanım yerine göre bazen SQL cümlelerinin bazen ise hazır bileşen nesnelerinin kullanılması, sistem performansını artırmış ve kullanım kolaylığı sağlamıştır.

DBAExplorer'un en temel özelliği açık kaynak olmasıdır. Program www.DBAExplorer.com adresinden ve diğer açık kaynak adreslerinden

indirilebilecektir. DBAExplorer geliştirilirken, Developer Express firmasının Quantum Grid ve Core Lab[44] firmasının ODAC bileşenleri kullanılmıştır. Bu bileşenler ücretli olduğundan yayınlanamamaktadır (kullanılan bileşenler ilgili firma sayfalarından indirilebilir.).

DBAExplorer genel olarak VT Yöneticilerine hitap etmektedir. Bu nedenle programdaki bazı ekranlara sadece yönetici hakkı olan kullanıcılar girebilmektedir. Bu ekranlarla hedeflenen amaç; VT durumu ve gidişatı hakkında yöneticilere bilgi vermek ve basit yardımlar sağlamaktır. VT kurulum özellikleri ve kuruldukları donanımlar, farklı olabileceği için dinamik bir yapı kurulmuştur. Bu yapı ile dinamik SQL cümleleri çalıştırılmakta ve sonuçları e-posta olarak alınabilmektedir. Bu amaç ile performansa ve sistem iyileştirmesine yönelik SQL cümleleri web sayfasında yayınlanmakta ve yöneticilerin bunlara erişmesi sağlanmaktadır.

DBAExplorer çoklu dil desteği sağlamaktadır. Daha önce yazılan bütün profesyonel araçlar İngilizcedir. Bu konuda, kullanıcıların zorluk çektiği saptandığından VTYA ilk başta İngilizce ve Türkçe olarak çıkartılacaktır. Daha sonra diğer dillerin ilavesi yapılacaktır.

DBAExplorer Microsoft ODBC veya Oracle istemci kullanmadan direkt VT bağlantısı sağlayabilmektedir. Böylelikle hızlı ve performanslı bir iletişim kurulması sağlanmıştır. Buradaki diğer önemli husus ise, programın çalışması için bilgisayarlara başka bir uygulama yükleme gereğinin olmamasıdır. Örneğin; .Net projesi için Microsoft Framework veya Java uygulamaları için Java Runtime gibi programların kurulması gerekmemektedir.

DBAExplorer Windows ortamında Delphi ile yazıldığı için sadece Windows işletim sistemlerini desteklemektedir. İleride diğer işletim sistemlerinde çalışması hedeflenmiştir. DBAExplorer nesnesel tabanlı tasarlanmıştır. Buradaki amaç, DBAExplorer kullanacak veya geliştirecek olan diğer kullanıcılara yardımcı olmak ve sürüm yükseltmesini kolaylaştırmaktır.

## 5.4. Değerlendirme

VT kullanan veya bu konuda hizmet veren her firmanın, VTYA gibi bir programa ihtiyacı vardır. Böyle bir projenin geliştirilmesi belirli maliyetleri doğuracaktır. Bu tez kapsamında geliştirilen DBAExplorer'un değerlendirmesi değişik 3 kişi tarafından yapılmıştır. Değerlendirme kıstasları diğer programların ki ile aynıdır. Çizelge 5.1'de tez kapsamında geliştirilen DBAExplorer'un diğer araçlar ile karşılaştırma sonuçları gösterilmiştir.

Elde edilen sonuçlara göre "PL/SQL Derleme Yetenekleri" bölümünden düşük puan aldığı yani bu konuda yetersiz kaldığı görülmüştür. Bunun nedeni ise tez çalışması sırasında yeterli sürenin ayrılamadığıdır. Program şu anki durumu ile prosedür ve fonksiyonları derleyip oluşan hataları gösterebilmekte fakat hata ayıklama işlemini yapamamaktadır. Bu nedenle bu bölümün daha sonra yapılacak işler olarak planlanmıştır.

Diğer taraftan aldığı 1650 genel puanı ile ortalamada yer aldığı dikkati çekmektedir. Bu durum geliştirilen projenin işlevsel ve kullanıcılara faydalı olacağı anlamına gelmektedir.

DBAExplorer'un eksiklikleri tamamlandığında piyasadaki lider programlarla aynı seviyeye geleceği veya onları geçeceği hedeflenmiştir.

**Çizelge 5.1.** DBAExplorer karşılaştırma sonuçları

| **Genel Toplam** | DBAExplorer | Rapid SQL 5.7 | SQL Insight 3.0 | Oracle SQL Dev 1.0 | Toad 8.6 | SQL Stn 5.0 | SQL Nav. 3.0 | PlSql Dev 7.0 | Keep Tool 7.2 |
|---|---|---|---|---|---|---|---|---|---|
| Arayüz özellikleri | **230** | 350 | 447 | 190 | 477 | 368 | 308 | 460 | 402 |
| VT programlama aracı | **930** | 808 | 971 | 620 | 1300 | 743 | 670 | 1319 | 1127 |
| Oracle/PL SQL özellikleri | **290** | 263 | 357 | 156 | 414 | 206 | 236 | 473 | 262 |
| PL/SQL derleme yeteneklerinden beklenenler | **50** | 90 | 120 | 135 | 135 | 190 | 110 | 200 | 165 |
| DBA ve SR'ye göre Oracle programcıları | **150** | 75 | 77 | 57 | 214 | 74 | 53 | 129 | 184 |
| **Genel Toplam** | **1650** | 1586 | 1972 | 1158 | 2540 | 1581 | 1377 | 2581 | 2140 |

# BÖLÜM 6

## 6. SONRA YAPILACAKLAR

DBAExplorer yeni gelişmelere açık ve nesnesel yazıldığı için daha sonra eklentiler yapılabilir veya geliştirilebilir. Bununla birlikte zaman açısından bazı modüllerin sonra eklenmesi hedeflenmiş ve planlanmıştır. Bu plan çerçevesinde aşağıdaki modüllerin yapılması hedeflenmiştir.

- Çoklu Ortam: Tez kapsamında geliştirilen program şu anki durumu itibariyle sadece Windows işletim sistemlerinde çalışmaktadır. Diğer işletim sistemlerini kullanan kullanıcılar da göz önüne alındığında DBAExplorer'un alternatif işletim sistemlerini de desteklemesi hedeflenmiştir. Bunun için yeni yazılım araçları araştırılacak ve projenin bu ortamlarda da çalışması sağlanacaktır.

- Derleme: Oluşturulan programda, Oracle Prosedürleri, Paketleri, Fonksiyonları ve Tetikleyicileri oluşturulabilmekte fakat hata kontrolü yapılamamaktadır. DBAExplorer bir sonraki sürümünde bu eksikliklerin de ilave edilmesi hedeflenmiştir.

- Diğer Veri Tabanları: DBAExplorer sadece Oracle VT'nı desteklemektedir. Oracle VT'nın seçilme sebebi, diğer veri tabanlarından daha detaylı ve büyük olmasıdır. Daha sonraki sürümlerde ya diğer veri tabanları da desteklenecek ya da bu veri tabanlarını destekleyen başka projeler yazılacaktır.

# BÖLÜM 7

## 7. ÖZET VE SONUÇ

Yedi bölümden oluşan bu tez çalışması, VTYS olan Oracle'ın performansının iyileştirilmesi amacına yönelik olarak araştırmaların yapılmasını ve elde edilen sonuçlara göre bir yazılım projesinin geliştirilmesini içermektedir.

İkinci bölümde VT yönetim sistemlerine genel bir bakış yapılmıştır. Bu bölümde Oracle VT hakkında bilinmesi gereken en temel bilgiler sunulmuştur.

Üçüncü bölümde performans yönetiminin önemi anlatılmıştır. VT performans ölçütleri anlatılmış, performansı etkileyen unsurlara değinilmiştir. Ayrıca bu bölümde performansa yönelik örnekler ve küçük ipuçları verilmiştir.

Dördüncü bölümde diğer VTYA'ları anlatılmış ve bir karşılaştırma tablosu çıkarılmıştır. Bu tabloda en popüler araçlardan bahsedilmiş ve puanlama yapılarak en iyisinin bulunması sağlanmıştır.

Beşinci bolümde ise yeni bir VTYA niçin gerek duyulduğu ve kullanılan teknoloji anlatılmıştır. Bu bölümde yeni oluşturulan VTYA olan DBAExplorer'un nasıl oluşturulduğu, alt yapısı ve ekranlarından kısaca bahsedilmiştir.

VT kullanımının gün geçtikçe yaygınlaşması, ortaya çıkan performans sorunları ve bu sorunların çözümlenmesi günümüzde büyük önem arz etmektedir.

Bu araştırma ile yeni bir VTYA oluşturmanın, firmalara yüklü bir maliyet getireceği, yüksek işletme ve bakım-onarım giderlerinin olacağı, bunlara ek olarak bu teknolojiyi geliştirecek fazladan uzman personel istihdamının gerektiği ortaya konulmuştur.

Yapılan maliyet ve performans analizleri sonucunda, yeni bir VTYA'nın yazılması yerine açık kaynak bir programın kullanılmasının daha faydalı olduğu gösterilmiştir. Bu sonuçlar değerlendirildiğinde, orta ölçekli işletmeler için yeni bir araç yazmaktansa mevcut bir araç kullanmanın işletme giderlerini büyük bir ölçüde azaltacağı öngörülmüştür.

VT üzerinde karmaşık işlemlerin yapılması için yardımcı bir aracın kullanılmasının hata oranını en aza indirdiği saptanmıştır. VT yöneticileri veya uygulama geliştiriciler için böyle bir aracın kullanılması performansı arttırmış, hata oranını ve işlem zamanını azaltmıştır.

DBAExplorer IEEE/EIA 12207 yazılım yaşam döngüsüne uygun olarak geliştirilmiş olması avantaj sağlamıştır. DBAExplorer'un modüler bir yapıda tasarlanması ve geliştirilmeye açık olarak oluşturulması, diğer geliştiriciler tarafından müdahale edilmesini ve geliştirmesini kolaylaştırmıştır.

Delphi 7.0 kullanımı rahat, öğrenimi ve takibi kolay bir programlama dili olmasından dolayı seçilmiştir. Bu nedenle, DBAExplorer da herhangi bir değişiklik istendiğinde, kodlamayı yapacak kişiler farklı olsa bile bu durum herhangi bir zorluk çıkarmayacaktır.

Tez kapsamında Oracle optimizasyonu araştırılmış ve en etkin SQL cümleleri DBAExplorer'a ilave edilmiştir. Günümüzün gelişen koşulları ve yeni ihtiyaçlara göre bu SQL cümleleri değiştirilebilir veya yenileri ilave edilebilir durumdadır. Veri tabanlarının kurulduğu ortam ve parametreler farklı olacağından, programı kullanan kişi/yöneticilerin bu durumları göz önüne alması en iyi sonucu doğuracaktır.

## KAYNAKLAR


[1] "Oracle Whitepapers" erişim adresi: http://www.oraclewhitepapers.com, erişim tarihi: 01 Temmuz 2007

[2] "Oracle Magazine Online" erişim adresi: http://www.oracle.com/oramag/index.html, erişim tarihi: 01 Temmuz 2007

[3] "Compatitive products Oracle FAQ " erişim adresi: http://www.orafaq.com/tools/competitive, erişim tarihi: 02 Temmuz 2007

[4] "Oracle SQL Tutorial" erişim adresi: http://www.db.cs.ucdavis.edu/teaching/sqltutorial/, erişim tarihi: 20 Temmuz 2007

[5] "Yükseköğretim Kurulu Ulusal Tez Merkezi" erişim adresi: http://193.140.255.11/tezjic/tez.htm, erişim tarihi: 01 Ağustos 2007

[6] "Wikipedia, the free encyclopedia" erişim adresi: http://tr.wikipedia.org/wiki/Veri_taban%C4%B1 , erişim tarihi: 05 Ocak 2008

[7] "National Capital Oracle User Group" erişim adresi: http://www.natcapoug.org/presntn_downloads/NCAPprstn2003_ToolsOfTheTradeDBA+DEV_v1.html, erişim tarihi: 19 Şubat 2008

[8] "Bilgisayar Alemi" erişim adresi : http://www.bilgisayaralemi.com/content/oracle/bellek-yapilari.html, erişim tarihi: 20 Aralık 2007

[9] "Böteb Online Web Kütüphanesi" erişim adresi : http://www.mtuncel.com/oracle.htm, erişim tarihi: 25 Aralık 2007

[10] "Oracle: Covering today's Oracle topics" erişim adresi: http://searchoracle.techtarget.com/, erişim tarihi: 04 Ocak 2008

[11] "Oracle Consulting, Oracle Support and Oracle Training by BC" erişim adresi : http://www.dba-oracle.com/, erişim tarihi: 14 Ocak 2008

[12] "René Nyffenegger on Oracle" erişim adresi: http://www.adp-gmbh.ch/, erişim tarihi: 20 Ocak 2008

[13] "New York Oracle Users Group, Inc" erişim adresi : http://www.nyoug.org/meetings.htm#2007_December_General_Meeting, erişim tarihi: 5 Şubat 2008

[14] "Wikipedia, the free encyclopedia" erişim adresi: http://en.wikipedia.org/wiki/SQL_Programming_Tool, erişim tarihi: 5 Şubat 2008

[15] "National Capital Oracle User Group" erişim adresi : http://www.natcapoug.org/presntn_downloads/NCAPprstn2003_ToolsOfTheTradeDBA+DEV_v1.html, erişim tarihi: 10 Şubat 2008

[16] "Steve Rea's Oracle Tips, Tricks, and Scripts" erişim adresi: http://www.uaex.edu/srea/#Oracle_Database_Tuning, erişim tarihi:10 Şubat 2008

[17] "Larry Holder's DBA Page" erişim adresi: http://www.utm.edu/staff/lholder/dba/, erişim tarihi: 13 Şubat 2008

[18] "Oracle PL/SQL and SQL" erişim adresi: http://www.jusungyang.com/ORACLEfolder/PLSQL.html, erişim tarihi: 15 Şubat 2008

[19] Hermann Baer, Partitioning in Oracle Database 10g Release 2, May 2005
[20] Cary V. Millsap, Jeff Holt, Optimizing Oracle Performance , Eylül 2003
[21] Jonathan Lewis, Cost Based Oracle: Fundamentals Appress, 2006
[22] Gaja Krishna Vaidyanatha, Kirtikumar Deshpande, John A. Kostelac, Oracle Performance Tuning 101, ISBN 0-07-213145-4, 2001
[23] David Clement, The Oracle Disk I/O Mechanism, Eylül 2004
[24] David Clement, How is SQL Parsed ?, Eylül 2004
[25] Presented by Oracle, Oracle Database 11g Application Development, 2008-03-14
[26] Karl Dias, Mark Ramacher, Uri Shaft, Venkateshwaran Venkataramani, Graham Wood, Oracle Corporation, Automatic Performance Diagnosis and Tuning in Oracle, 2005
[27] Benoît Dageville, Mohamed Zait, SQL Memory Management in Oracle9i, Proceedings of the 28th international conference on Very Large Data Bases, VLDB 2002: 962-973, 2002
[28] Florian Haftmann, Donald Kossmann, Alexander Kreutz, Efficient Regression Tests for Database Applications, ISSN:1066-8888, 2007
[29] Kevin Loney, George Koch, Oracke9i The Complete Referans, 2002
[30] Richard J. Niemiec, Performance Tuning - Now You are the V8 Expert, Oracle Press (900 pages): ISBN 0-07-882434-6, 2008
[31] Global Bilgi, SQL Tuning El Kitabı, 2008
[32] Richard J. Niemiec, Performance Tuning for the Expert, (800) 755-TUSC, 2001
[33] Oracle Corparation, Oracle Veri Tabanı Yönetim Sistemine Giriş, 2008
[34] Sue Harper, Oracle SQL Developer for Database Developers, Haziran 2007
[35] Thomas B. Cox, Database Administration Maturity Model, 1999
[36] Nihat Demirli, Yüksel İnan, Borland Delphi 7, Ankara 2003
[37] Benoit Dageville, Dinesh Das, Karl Dias, Khaled Yagoub, Mohamed Zait, Muhamed Ziauddin, Automatic SQL Tuning in Oracle 10g, Proceeding of the 30th VLDB Conference, Totonto Canada 2004
[38] Oracle Corparation, Oracle White Paper, Performance Tuning using the SQL Access Advisor, http://otn.oracle.com, 2003
[39] Oracle Corparation, Oracle White Paper, Getting Started With Use Case Modeling, 2005
[40] "Oracle Expert Service" erişim adresi: http://www.oracle.com/global/tr/consulting/expert-services.html, erişim tarihi: 15 Ocak 2008
[41] "Oracle Database Performance Tuning Guide", erişim adresi: http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/ex_plan.htm#sthref1852, erişim tarihi: 20 Ocak 2008
[42] "Oracle Danışmanlık" erişim adresi: http://www.oracledanismanlik.com/belgeler/OraclePerformansIyilestirme_No1.html, erişim tarihi: 29 Ocak 2008
[43] "Oracle Enterprise Manager Database Tuning with the Oracle Tuning Pack" erişim adresi: http://download-east.oracle.com/docs/html/A86647_01/toc.htm, erişim tarihi: 20 Şubat 2008

[44] “Advanced Data Access Solutions” erişim adresi: http://www.crlab.com/, erişim tarihi: 10 Temmuz 2007
[45] “Borland The Open ALM Company” erişim adresi: http://www.borland.com, erişim tarihi: 01 Temmuz 2007

# EKLER

**EK A:** VT veri sözlüğü oluşturulurken kullanılan SQL komutları çalışma şekli ve parametreleri aşağıda verilmiştir.

## ➢ TABLO OLUŞTURMA (CREATE TABLE)

Tablo oluşturmak için kullanılan komuttur. Komut yapısı ve kullanım örneği aşağıda verilmiştir. Bir tablo oluşturabilmek için "*CREATE TABLE*" veya "*CREATE ANY TABLE*" sistem haklarına sahip olmak gerekir.

**Komut Yapısı:**

```
CREATE TABLE <tablo_adı>
(
<Kolon_adı1>          veri tipi,
< Kolon _adı2>        veri tipi
<...>
[<tablo kısıtlamaları>
      CONSTRAINT    <kısıtlama adı>
      [UNIQUE | PRIMARY KEY (<Kolon adı>, ...)]
      [FOREIGN KEY (<Kolon adı>, ...)]
       REFERANCES <tablo adı>(< Kolon adı>)
]
[PCTFREE         sayı]
[PCTUSED         sayı]
[INITRANS        sayı]
[MAXTRANS        sayı]
[TABLESPACE      <Tablo Uzayı İsmi>]
[STORAGE         <Kayıt Parametreleri>
      (INITIAL       sayı K|M
       NEXT          sayı K|M
      MINEXTENTS     sayı
      MAXEXTENTS     sayı | UNLIMITED
```

```
      PCTINCREASE    sayı
      FREELISTS      sayı
      FREELISTS GROUPS    sayı )]
 [RECOVERABLE | UNRECOVERABLE]
 [AS  <Sql Sorgusu>]
);
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**Tablo_adı:** Oluşturulacak olan tabloya verilecek addır.

**Kolon_adı:** Tablo kolonlarına verilen addır.

**Veri_tipi:** Kolonlara ait Oracle veri tipleri ve kullanıcı tarafından tanımlanan veri tipidir.

**Tablo Kısıtlamaları**: Veri bütünlüğünü sağlayan kısıtlamalardır. Kullanılan kısıtlamalar Bölüm 2.4 Veri Bütünlüğünde detaylı olarak anlatılmıştır.

**PCTFREE:** Tablonun her bloğunda sonradan yapılacak olan değişiklikler için ayrılan boş bölümdür.

**PCTUSED:** Bir tablonun bloklarında olması gereken en az doluluk oranıdır.

**INITRANS:** Her veri bloğunda olması gereken en az toplu işlem sayısını belirtir.

**MAXTRANS:** Her veri bloğunda olabilecek en fazla toplu işlem sayısını belirtir.

**Tablo Uzayı İsmi:** Tablonun hangi tablo uzayı içerisinde yaratılacağını belirten isimdir.

**RECOVERABLE**: VT ARCHIVELOG modda çalışırken, tablonun ismini belirtmeden yedekten indirilmesini belirten parametredir

**UNRECOIVERABLE:** VT ARCHIVELOG modda çalışırken, tabloyu yedekten otomatik olarak getirmemesini belirten parametredir.

**AS<Sql Sorgusu>**: Bir tablodan, içindeki kayıtlarla birlikte sadece belirtilen alanları alarak bir başka tablo oluşturma komutudur.

**STORAGE <Kayıt Parametreleri>:** Tablonun VT'ndaki kayıtla ilgili parametrelerin belirtildiği bölümdür. Parametreleri şu şekildedir;

**INITIAL:** Tablo yaratıldığında ilk aldığı genişlemenin büyüklüğünü belirten bölümdür. Eğer, sayının sonuna K konursa sonuç Kilobayt cinsinden, M konursa Megabayt cinsinden belirtilmiş olmaktadır.

**NEXT:** Tablo yaratıldıktan sonra alacağı genişlemelerin büyüklüğünü belirten bölümdür.

**MINEXTENTS**: Tablo yaratıldığında ilk olarak alacağı minimum genişleme sayısını belirtildiği bölümdür.

**MAXEXTENTS**: Bir tablonun ilk olarak aldığı genişlemede dâhil olmak üzere alabileceği en büyük genişleme sayısının belirtildiği bölümdür.

**PCTINCREASE:** Tablonun alacağı genişlemelerden bir sonrakinin bir öncekinden ne kadar büyük olacağını belirten bölümdür. Mesela, 50 yazılırsa, bir sonraki genişleme her zaman bir öncekinden %50 daha büyük olacaktır.

**FREELISTS**: Tablo için kullanılacak olan freelist gruplarının her birinde kaç tane boş liste olacağının belirtildiği bölümdür.

**FREELISTS GROUPS:** Bir tablo için kaç tane boş liste grubunun olacağının belirtildiği bölümdür.

**Tablo İsim Verme Kuralları:**

- İsimler harflerle başlamalıdır.(A-Z veya a-z)
- Büyük veya küçük harflerin tamamı kullanılabilir. (ADRES-Adres-adres-Adres)
- Özel karakterler başa gelmemek kaydıyla kullanabilir.($-%-_)
- En fazla 30 karakter olmalıdır.
- Bir başka tablonun ismi kullanılamaz.
- SQL'in kendi komutları isim olarak kullanılamaz.
- İki ayrı isim varsa "_" işareti ile birleştirilerek yazılabilir.

**Örnek - 1**:

```
CREATE TABLE OGRENCI
(
OGRNUM          NUMBER(11) NOT NULL,
AD              VARCHAR2(15),
```

```
SOYAD           VARCHAR2(15),
DOGYER_ID       NUMBER(2),
CONSTRAINT CST_OGRNUM PRIMARY KEY(OGRNUM)
CONSTRAINT CST_DOGYER FOREING KEY(DOGYER_ID) REFERENCES
IL(IL_ID)
);
```

Bu örnekte "*OGRENCI*" adında bir tablo oluştulmaktadır. Tablonun dört alanı vardır. Bunlardan biri özel anahtar (*Primary Key*) olarak tanımlanmıştır. Bu alanın değeri boş olamaz ve tabloda aynı iki değer bulunamaz. Bir tablo ile başka bir tablo arasında ilişki kurulacaksa bu ana tabloda *PRIMARY KEY* tanımıyla, diğer tabloda *FOREIGN KEY* tanımıyla yapılır. Yani doğum yeri alanı için buradaki tablo, ana tablo değildir. İl bilgilerinin tutulduğu başka bir tabloda olabilir. Böylece aynı il kodu birden fazla kayıtta yer alabilir. İlişki kurulan diğer tablonun adını yukarıdan çıkarabiliriz. Bu tablo "*IL*" ismindedir. Her il bu tabloda kayıtlıdır ve her birinin bir kodu vardır. Bu tabloda bir ilin kaydı iki kolonda yer alamaz. Bu yüzden "*IL_id*" alanı "*il*" tablosu için *PRIMARY KEY* olarak tanımlanmıştır.

**Örnek – 2:**

```
CREATE TABLE SINIF
(
SINIF_KOD           NUMBER(2),
SINIF_AD            VARCHAR2(20),
CONSTRAINT PK_SINIF PRIMARY KEY (SINIF_KOD) )
STORAGE             //Kayıt parametresi bölümünün başlangıcını göstermektedir.
(INITIAL  100K      //Tablonun yaratıldığında ilk alacağı parçanın büyüklüğünü
göstermektedir.
NEXT        10K     //Tablonun alacağı genişlemenin büyüklüğünü göstermektedir.
MINEXTENTS   2      //Tablo yaratıldığında ilk olarak alacağı en az parça + genişleme
sayısıdır.
MAXEXTENTS  10      //Tablonun büyümek için alabileceği en fazla genişleme sayısıdır.
PCTINCREASE 10      // Her sonraki genişlemenin bir öncekinden % kaç büyük olacağı
bilgisidir.
```

```
PCTFREE      40   // Sonradan yapılacak günleme işlemleri için boş kalması gereken alandır.
PCTUSED      40   // Her bloğun en az dolu olması gereken bölümünü göstermektedir. Eğer, doluluk miktarı bu sınırın altına düşerse, VT bu bloğa  kayıt eklemeye başlar.
);
```

## ➢ TABLO ÖZELLİĞİ DEĞİŞTİRME (ALTER TABLE)

Daha önceden oluşturulmuş bir tablonun özelliklerini değiştirmek için kullanılan komuttur.

**Komut Yapısı:**

```
ALTER TABLE <tablo_adı>
ADD | MODIFY | DROP (<kolon adı> veri tipi <kolon kısıtlaması>)
ENABLE ifade1
DISABLE ifade2
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**ADD**: Bir kolon veya kısıtlama eklemesini sağlayan komuttur.
**MODIFY**: Bir kolonun özelliklerini değiştirmeye yarayan komuttur.
**DROP**: Bir kolon veya kısıtlama silinmesini sağlayan komuttur.

ALTER komutuyla tablolara yeni bir alan ve kısıtlama eklenebilir, var olan alan ve kısıtlamaların durumu değiştirilebilir veya tablodan kısıtlamalar düşürülebilir. Fakat ALTER komutuyla var olan kolonlar düşürülemez. Bir alanın değeri büyültülebilir, fakat küçültülemez. Tablo alanlarında kayıt var ise alan veri türleri değiştirilemez. (Not: Alan içerisi boşaltılarak alan türünün değiştirilmesi sağlanabilir.)

**Örnek–1:** Mevcut tabloya yeni bir alan eklemek için kullanılan komut:

```
ALTER TABLE OGRENCI
ADD COLUMN (Bolum_id NUMBER);
```

**Örnek–2:** Mevcut bir tablo üzerinde, başka bir tabloya yeni bir referans vermek için kullanılan komut:

```
ALTER TABLE OGRENCI
ADD CONSTRAINT cst_Bolum_id
FOREIGN KEY(Bolum_id)
REFERENCES BOLUM(Bolum_id);
```

Yukarıdaki örnekte OGRENCI adlı tablonun yapısı değiştirilmiştir. İlk önce yeni bir alan (Bolum_id) eklenmiş daha sonra bu alan BOLUM tablosu ile eşlenmiştir.

**Örnek–3:** Mevcut tabloda kullanılan bir alanın boyutunu değiştirme komutu:

```
ALTER TABLE OGRENCI
MODIFY AD VARCHAR2(20);
```

Bu örnekte de OGRENCI tablosundaki mevcut AD alanının uzunluğu artırılmıştır.

### ➢ TABLO SİLME (DROP TABLE)

Daha önceden oluşturulmuş bir tablonun düşürülmesi için kullanılan komuttur.

**Komut Yapısı:**

```
DROP TABLE <tablo_adı> [CASCADE CONSTRAINTS]
```

Köşeli parantez içerisindeki tanım kullanılırsa ana-detay (master-detail) ilişkili tablolarda ana tablo düşürülünce detay tabloların da otomatik olarak düşürülmesi sağlanır. Eğer bu seçenek kullanılmazsa diğer tablolarla ilişkisi bulunan bir tablo silinemez. Ancak bu ilişkiler kaldırıldıktan sonra tablo silinebilir.

**Örnek:** Mevcut bir tablonun silinmesi için kullanılan komut:

```
DROP TABLE OGRENCI CASCADE CONSTRAINT;
```

- **TABLO ANALİZ ETME (ANALYZE TABLE)**

VT'nın performansının iyileştirilmesi için belli aralıklarla istatistiklerin alınması gerekmektedir. "ANALYZE" istatistik toplama komutudur. İstatistikler, tablo (table), indis (index) ve kümeler (cluster) için toplanmaktadır.

**Komut Yapısı:**

```
ANALYZE TABLE <tablo ismi>
[ESTIMATE STATISTICS [SAMPLE sayı ROWS I PERNCENT ] ]
[COMPUTE STATISTICS ]
[DELETE STATISTICS ]
[VALIDATE STRUCTURE [CASCADE ] ]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**ANALYZE TABLE:** Tablonun analiz edileceğini göstermektedir.

**ESTIMATE STATISTICS:** İstatistikleri yaklaşık olarak tahmin etmektedir ve kullanmak için veri sözlüğüne kaydetmektedir.

**SAMPLE:** İstatistik için örnek uzay olarak alınacak kayıtların belirtildiği bölümdür. Eğer, belirtilmezse Oracle istatistik için standart olarak 1064 kayıt almaktadır.

**ROWS:** Örnek uzay için tablodan kaç kayıt alınacağının belirtildiği bölümdür.

**PERCENT:** Örnek uzay için tablonun yüzde kaçının alınacağının belirtildiği bölümdür.

**COMPUTE STATISTICS:** İstatistikleri hesaplamaktadır ve kullanmak için veri sözlüğüne kaydetmektedir.

**DELETE STATISTICS:** Analiz sonucu olan istatistiklerin silinmesini ifade eden bölümdür.

**VALIDATE STRUCTURE:** Analiz edilen nesnenin yapısının doğru olduğunu teyit etmektedir.
**CASCADE:** Nesneye ait indislerin de nesne ile beraber analiz edilmesini sağlayan bölümdür.

**Örnek:**

```
ANALYZE TABLE OGRENCI
COMPUTE STATISTICS;
```

"*OGRENCI*" tablosunu istatistik toplama açısından analiz etme komutudur.

- **GÖRÜNTÜ OLUŞTURMA (CREATE VIEW)**

Görüntü oluşturmak için bir SELECT cümlesi kullanmak gerekir. Bir görüntü bir ya da daha fazla tablodan oluşturulabileceği gibi, bir başka görüntüden de oluşturulabilir.

**Komut Yapısı:**

```
CREATE [OR REPLACE] [FORCE | NONFORCE] VIEW <görüntü_adı>
( <alias>,...)
AS Alt sorgu (SELECT cümlesi)
[WITH [ CHECK OPTION | CHECK OPTION CONSTRAINT <kısıtlama
adı>]]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**OR REPLACE**: Eğer bu isimde bir görüntü varsa, onu düşürmeden (drop) yeniden aynı isimde yeni bir görüntü oluşturulabilir.
**FORCE**: Görüntüyü oluşturacak tablo olmasa dahi görüntünün oluşturulmasını sağlar.
**NONFORCE**: Görüntüyü oluşturacak tablo olmadan görüntünün oluşturulmasına izin vermez. Belirtilmese dahi standart olarak kullanılır.

**ALIAS**: Görüntüyü oluşturan SELECT ifadesindeki kolonlara görüntüde yeni bir isim vermeyi sağlar.

**Alt Sorgu**: Görüntüyü oluşturan SQL cümlesidir.

**WITH CHECK OPTION**: Görüntü ile yapılan ekleme ve değiştirme işlemlerinin görüntünün SQL cümlesindeki şarta uygunluğunu kontrol etmeyi sağlar.

**CONSTRAINT**: "WITH CHECK OPTION" ile belirtilen kontrolün bir kısıtlama olarak belirtilmesini sağlayan bölümdür.

**Örnek–1**:

```
CREATE OR REPLACE VIEW OGRENCI_LISTESI
AS
SELECT O.*, I.KOD, B.BOLUM_KOD
  FROM OGRENCI O, IL I, BOLUM B
 WHERE O.DOGUMYER_ID = IL.IL_ID
   AND O.BOLUM_ID = B.BOLUM_ID
ORDER BY O.AD, O.SOYAD;
```

Yukarıdaki örnekte üç farklı tablo birleştirilerek bir görüntü oluşturulmaktadır.

**Örnek – 2:**

```
CREATE OR REPLACE VIEW KOD_LISTESI
AS
SELECT KOD FROM IL
UNION ALL
SELECT KOD FROM BOLUM;
```

Yukarıdaki örnekte iki farklı tablo tek bir tabloymuş gibi bir görüntü oluşturulmaktadır.

### ➢ GÖRÜNTÜ DEĞİŞTİRME (ALTER VIEW)

Bir görüntüyü (view) yeniden derleme komutudur. Genel kullanımı aşağıdaki gibidir:

**Komut Yapısı:**

```
ALTER VIEW <görüntü_adı> COMPILE
```

**Örnek:**

```
ALTER VIEW KOD_LISTESI COMPILE;
```

### ➢ GÖRÜNTÜ SİLME (DROP VIEW)

Daha önceden oluşturulmuş bir görüntünün düşürülmesi için kullanılan komuttur.

**Komut Yapısı:**

```
DROP VIEW <görüntü_ismi>
```

**Örnek:**

```
DROP VIEW KOD_LISTESI;
```

### ➢ TABLO UZAYI OLUŞTURMA (CREATE TABLESPACE)

Daha önce anlatıldığı gibi tablo uzayı kullanıcılara ait olan nesnelerin VT'nında mantıksal olarak tutulduğu yere denmektedir. Bir tablo uzayı oluştururken, bu tablo uzayının verilerinin hangi veri dosyasına konulacağı ve bu dosyanın dizini ile büyüklüğü bildirilmelidir.

Tablo uzayı yaratmak için ,"CREATE TABLESPACE" sistem hakkına sahip olmak gerekmektedir ve" SYSTEM" Tablo Uzay'ında iki tane boş geri alma parçası (Rollback Segment) bulunmalıdır.

**Komut Yapısı:**

```
CREATE TABLESPACE <tablo_uzayı_ismi>
DATAFILE <dosya ismi>SIZE sayı K|M [AUTOEXTEND OFF |
{AUTIOEXTEND
ON | NEXT değer K|M  |  MAXSIZE UNLIMITED | değer K|M |}
[DEFAULT STORAGE kayıt parametreleri ]
[ONLINE | OFFLINE ]
[PERMANENT | TEMPORARY ]
[STORAGE     <Kayıt Parametreleri>]
            (INITIAL          sayı  K | M
             NEXT             sayı  K < M
             MINEXTENTS       sayı
             MAXEXTENTS       sayı | UNLIMITED
             PCTINCREASE      sayı )]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**Tablo Uzayı İsmi**: Oluşturulacak tablo uzayına verilecek addır.
**DATAFILE:** Tablo Uzayını oluşturan veri dosyasının tanımlandığı bölümdür.
**Dosya İsmi:** Veri Dosyası yolunun belirtildiği bölümdür.
**AUTOEXTEND OFF:** Data Dosyasının otomatik artışını iptal etmekte kullanılan komuttur.
**AUTOEXTEND ON:** Data Dosyasının otomatik artışını sağlayan komuttur.
**NEXT:** Data Dosyasını genişleme miktarı.
**MAXSIZE:** Data Dosyasının genişleyebileceği en son miktardır.
**UNLIMITED:** Data Dosyasının sınırsız büyüyebileceğini göstermektedir.
Belirtilmese dahi standart olarak seçilidir.

**DEFAULT STORAGE:** Kayıt parametrelerinin tanımladığı bölümdür.

**ONLINE**: Tablo Uzayını oluşturduktan hemen sonra aktif olacağını belirten bölümdür. Standart olarak seçilidir.

**OFFLINE:** Tablo Uzayını oluşturduktan hemen sonra pasif olacağını belirten bölümdür.

**PERMANENT:** Tablo Uzayının içinde tutulacak olan nesnelerin kalıcı olarak saklanacağını belirten bölümdür. Standart olarak seçilidir.

**TEMPORARY:** Tablo Uzayında geçici nesnelerin tutulacağının belirtildiği bölümdür.

**NOT:** Tablo Uzayını oluşturmak ve yönetmek için STORAGE MANAGER kullanılmasını tavsiye ederiz.

**STORAGE <Kayıt Parametreleri>:** Tablo Uzayı'nın VT'ndaki kayıtla ilgili parametrelerin belirtildiği bölümdür. Yapısı şu şekildedir:

**INITIAL:** Tablo Uzayı yaratıldığında ilk aldığı genişlemenin (extent) büyüklüğünü belirten bölümdür. Eğer sayının sonuna K konursa sonuç Kilobayt cinsinden, M konursa Megabayt cinsinden belirtilmiş olmaktadır.

**NEXT:** Tablo Uzayı yaratıldıktan sonra alacağı genişlemelerin (extent) büyüklüğünü belirten bölümdür.

**MINEXTENTS**: Tablespace yaratıldığında ilk olarak alacağı minimum genişleme sayısının belirtildiği bölümdür.

**MAXEXTENTS:** Bir Tablo Uzayı'nın ilk olarak aldığı genişleme de dâhil olmak üzere alabileceği en büyük genişleme sayısının belirtildiği bölümdür.

**PCTINCREASE:** Tablo Uzayı'nın alacağı genişlemelerden bir sonrakinin bir Öncekinden ne kadar büyük olacağını belirten bölümdür. Mesela, 50 yazılırsa, bir sonraki genişleme her zaman bir öncekinden%50 daha büyük olacaktır.

**Örnek**:

```
CREATE TABLESPACE TBS_OKUL
DATAFILE 'c:\oracle\data.dat' SIZE 10M
DEFAULT STORAGE (INITIAL 10K NEXT 50K
```

```
MINEXTENTS 1 MAXEXTENTS 999)
ONLINE;
```

*SIZE* bildirisi veri dosyasının diskte kaplayacağı yeri belirler. Burada *10M, 5K* gibi değerler girilebilir. *INITIAL* bildirisi tablo uzayı oluşturulduğunda, ilk alacağı genişleme'nin büyüklüğünü belirler. *Next* tablo uzayı oluşturulduktan sonra alacağı genişlemelerin büyüklüğünü belirler. *MINEXTENTS* tablo uzayı oluşturulduğunda ilk olarak alacağı minimum genişleme sayısının belirtildiği bölümdür. *MAXEXTENTS* bir tablo uzayının ilk olarak aldığı genişleme de dâhil olmak üzere alabileceği maksimum genişleme sayısının belirtildiği bölümdür.

## ➢ TABLO UZAYI DEĞİŞTİRME (ALTER TABLESPACE)

Bu komutla, veri dosyası eklenebilmekte, veri dosyasının ismi, kayıt parametreleri değiştirilebilir. Tablo uzayı açma/kapama işlemleri yapılabilir. Genel kullanımı aşağıdaki gibidir:

**Komut Yapısı:**

```
ALTER TABLESPACE < tablo_uzayı_ismi>
[ADD DATAFILE {<dosya bölümü>,..}
[AUTOEXTENT OFF|ON NEXT sayı K | M [MAXSIZE[UNLIMITED|sayı K |
M]]
[ONLINE]
[OFFLINE NORMAL|TEMPORARY|IMMEDIATE]
[BACKUP BEGIN|END ]
[READ ONLY|WRITE]
[PERMANENT|TEMPORARY]
[RENAME DATAFILE {<dosya ismi>,..} TO {<dosya ismi>,...}]
[STORAGE     <Kayıt Parametreleri>]
           (INITIAL          sayı  K | M
            NEXT             sayı  K < M
            MINEXTENTS       sayı
            MAXEXTENTS       sayı | UNLIMITED
```

```
                PCTINCREASE      sayı )]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**ADDDATAFILE**: Dosya bölümünde belirtilen dosyanın tablo uzayına eklendiği bölümdür.

**AUTOEXTEND ON**: Veri dosyasının otomatik olarak genişlemesini sağlamaktadır.

**AUTOEXTENT OFF**: Veri dosyasının otomatik olarak genişlemesini durdurmaktadır.

**NEXT:** Bir sonraki otomatik genişlemenin büyüklüğünü belirtmektedir.

**MAXSIZE**: Otomatik genişleme için ayrılan maksimum disk kapasitesini belirtmektedir.

**UNLIMITED:** Otomatik genişlemenin sınırsız olabileceğinin belirtildiği bölümdür.

**RENAME DATAFILE:** Tablo uzayının veri dosyalarının isimlerinin değiştirildiği bölümdür.

**COALESCE:** Tablo uzayına ait bütün veri dosyaları içerisindeki boş alanları birleştirip yan yana daha büyük boş alanların elde edilmesini sağlayan bölümdür.

**DEFAULT STORAGE:** Belirtilen parametreleri yeni kayıt parametreleri olarak kabul etmesinin belirtildiği bölümdür.

**ONLINE:** Tablo uzayını açık hale (kullanılabilir) getirmektedir.

**OFFLINE:** Tablo uzayı dosyasını kapalı(kullanılmaz) hale getirilmektedir.

**NORMAL:** Tablo uzayına ait bütün veri dosyalarını açarken yapılacak işlemler için bir işaret konmasının belirtildiği bölümdür. Bu işlem sonucu VT açılırken kurtarma işlemi yapmaya gerek kalmamaktadır.

**TEMPORARY**: Tablo uzayına ait bütün veri dosyalarını açarken yapılacak işlemler için bir işaret konmasının belirtildiği bölümdür. Bu işlem sonucu VT açılırken önceden kapalı olan veri dosyalarına kurtarma işlemi yapmaya gerek kalmaktadır.

**IMMEDIATE:** İşaret verme işleminin yapılmayacağını ve dosyaların sağlam kalacağını garantilemeyen bölümdür.

**BEGIN BACKUP:** Tablo uzayını oluşturan veri dosyalarında VT açık halde fiziksel yedek almanın başlamasını belirten bölümdür.

**END BACKUP:** Yedek alma işleminin bittiğini göstermektedir.

**READ ONLY:** Tablo uzayını salt okunur hale getirme bölümüdür.

**READ WRITE:** Tablo uzayını yazılabilir hale getirme bölümüdür.

**PERMANENT:** Tablo uzayını sabit (içindeki bilgiler kalıcı olan ) hale getirme bölümüdür.

**TEMPORARY:** Tablo uzayını geçici (içindeki bilgiler kalıcı olmayan ) hale getirme bölümüdür.

**Örnek**:

```
ALTER TABLESPACE TBS_OKUL
DATAFILE 'c:\oracle\data2.dat' SIZE 20M ;
```

- **TABLO UZAYI SİLME (DROP TABLESPACE)**

Tablo uzayı düşürme komutudur.

```
DROP TABLESPACE <tablo uzayı ismi>
[INCLUDING CONTENTS [ CASCADE CONSTRAINTS]]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**INCLUDING CONTENTS:** Tablo uzayının bütün içeriğinin de boşaltılacağını belirtir.

**CASCADE CONSTRAINTS:** Tablo uzayı içerisindeki tablolarla diğer tablo uzaylarındaki tablolarda olan bütün kısıtlamaların da düşürülmesini sağlar.

**Örnek:**

```
DROP TABLESPACE TBS_OKUL;
```

## ➢ YENİ KULLANICI OLUŞTURMA (CREATE USER)

Oracle da Kullanıcı VT nesnelerinin sahibidir. Kullanıcılar, nesneleri oluşturur, kullanır ve silerler. Oracle VT ilk kurulduğunda standart olarak üç kullanıcı tanımlanır. Bunlardan bir SYS kullanıcısıdır.

SYS kullanıcısı: Veri sözlüğünün sahibi olan kullanıcıdır. Tüm nesneleri oluşturma hakkına sahiptir ve diğer bütün kullanıcıların nesnelerine erişebilir. SYS kullanıcısının ilk şifresi "change_on_install" olarak belirlenmiştir.

SYSTEM kullanıcısı: SYSTEM kullanıcısı veri sözlüğünü kullanma hakkına sahiptir. Önemli nesneleri oluşturma hakkına da sahiptir. İlk şifresi "manager" olarak belirlenmiştir. Diğer kullanıcıların nesnelerine erişme hakkına da sahiptir.

SCOTT kullanıcısı: SCOTT kullanıcısı VT'na başlangıçta yüklenen demo tabloların sahibidir. Bu kullanıcının nesneleri kullanılarak SQL denemeleri yapılabilir.

"CREATE USER" komutunu SYS ve SYSTEM kullanıcıları standart olarak kullanabilir. Bu hak diğer kullanıcılara da verilebilir. Her kullanıcının nesnelerini tutmak için bir tablo uzayı oluşturmak sistemin performansı açısından gereklidir. Kullanıcı oluşturulurken bu tablo uzayı o kullanıcıya atanır.

**Komut Yapısı:**

```
CREATE USER <kullanıcı ismi> {IDENTIFIED BY <şifre>
|EXTERNALLY }
[DEFAULT TABLESPACE  <tablo uzayı ismi>]
[TEMPORARY TABLESPACE  < tablo uzayı ismi> ]
[QUOTA UNLIMITED | (sayı K|M ) ON < tablo uzayı ismi> ]
[PROFILE <parametre dosyası> ]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**Kullanıcı ismi**: Oluşturulacak olan kullanıcının ismidir.

**IDENTIFIED BY**: Oluşturulacak olan kullanıcıya verilecek şifredir.

**IDENTIFIED EXTERNALLY**: Oluşturulacak olan kullanıcıya verilecek şifrenin işletim sistemi tarafından belirleneceğini söyler.

**DEFAULT TABLESPACE**: Kullanıcını oluşturduğu nesnelerin yerini belirtir. Eğer belirtilmezse "SYSTEM" Tablo Uzayı kullanılır.

**TEMPORARY TABLESPACE:** Kullanıcının geçici parçalarının saklanacağı yeri ifade eder. Eğer belirtilmezse "SYSTEM" Tablo Uzayı kullanılır.

**QUOTA**: Kullanıcının belirtilen Tablo Uzayında kullanacağı yer miktarını belirtir.

**PROFILE**: Kullanıcının değişik özelliklerini belirten parametre dosyasıdır.

**Örnek:**

```
CREATE USER usr_alp
IDENTIFIED BY alpi
DEFAULT TABLESPACE tbs_ornek
QUOTA UNLIMITED ON tbs_ornek
```

Yukarıdaki örnekte *usr_alp* adında bir kullanıcı oluşturuluyor. Kullanıcının şifresi *IDENTIFIED BY* ile "*alpi*" olarak bildiriliyor. Kullanıcının kendi nesnelerini oluşturacağı tablo uzayı için ise "*tbs_ornek*" tablo uzayı bildiriliyor. Kullanıcının bu tablo uzayındaki tüm alanı kullanabileceği *QUOTA UNLIMITED* ile belirleniyor.

### ➢ KULLANICI BİLGİSİ DEĞİŞTİRME (ALTER USER)

Bir kullanıcının şifresini, standart tablo uzayını, geçici tablo uzayını, tablo uzayı kullanma haklarını ve standart rollerini değiştirme komutudur.

**Komut Yapısı:**

```
ALTER USER <kullanıcı ismi>
[ IDENTIFIED [ BY <şifre>] | [EXTERNALLY]]
```

```
[ DEFAULT TABLESPACE < tablo uzayı ismi>]
[ TEMPORARY  TABLESPACE < tablo uzayı ismi>]
[DEFAULT ROLE [ {<rol ismi> , ... }]| [ALL [EXCEPT (<rol
ismi>)]] | [NONE]]]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**IDENTIFIED BY:** Kullanıcının şifresinin değiştirildiği bölümdür.
**IDENTIFIED EXTERNALLY:** Kullanıcının işletim sisteminin şifresini kullanacağının belirtildiği bölümdür.
**DEFAULT TABLESPACE:** Kullanıcını oluşturduğu nesnelerin yerini belirtir. Eğer belirtilmezse "SYSTEM" tablo uzayını kullanır.
**TEMPORARY TABLESPACE:** Kullanıcının geçici parçalarının saklanacağı yeri ifade eder. Eğer belirtilmezse "SYSTEM" tablo uzayını kullanır.
**DEFAULT ROLE:** Kullanıcı için standart olarak rol tanımlama bölümüdür.
**ALL:** Kullanıcı için bütün rolleri standart olarak tanımlama bölümüdür.
**NONE:** Kullanıcı için standart olarak rol tanımlamama bölümüdür.

**Örnek:**

```
ALTER USER usr_alp IDENTIFIED BY 12345;
```

- **KULLANICI SİLME ( DROP USER)**

Oracle da mevcut bit kullanıcıyı düşürme komutudur.

**Komut Yapısı:**

```
DROP USER <kullanıcı ismi> [CASCADE]
```

**Kullanılan İfade Açıklaması:**

**CASCADE:** Kullanıcıya ait bütün nesnelerin de düşürüleceğini belirtir.

**Örnek:**

```
DROP USER usr_alp;
```

- **YENİ ROL OLUŞTURMA (CREATE ROLE)**

Rol, VT'ndaki hakların toplanmış halidir. Rollerle, VT Yöneticisi ( DBA - Database Administrator ) işini daha kolay gerçekleştirebilmektedir. Rol oluşturabilmek için "CREATE ROLE" sistem hakkına sahip olmak gerekmektedir.

**Komut Yapısı:**

```
CREATE ROLE <rol ismi>
[NOT IDENTIFIED | ( IDENTIFIED BY <şifre> | IDENTIFIED
EXTERNALLY)]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**NOT IDENTIFIED**: Kendisine bu rol hak olarak verildiğinde kullanıcı rolü alınca, aldığına dair onaylamasının gerek kalmadığı durumlarda kullanılır. Standart olarak seçilidir.
**IDENTIFIED BY**: Kendisine bu rol hak olarak verilen kullanıcı bu rollü aktif hale getirmesi için şifre belirtmesi gerektiği durumlarda kullanılır.
**IDENTIFIED EXTERNALLY**: Şifrenin işletim sistemi tarafından kontrol edildiği durumlarda kullanılır.

**Örnek - 1:**

Bu örnekte, Şifresi "*sifre*" olan "*sifre_kontrol*" isminde bir rol oluşturma.

```
CREATE ROLE sifre_kontrol IDENTIFIED BY sifre;
```

**Örnek – 2:**

```
CREATE ROLE tablo_izle;
GRANT SELECT, INSER, UPDATE ON tbl_DERS TO tablo_izle;
```

Yukarıdaki örnekte "*tablo_izle"* adında bir rol oluşturulmuştur. Daha sonra bu role "*tbl_DERS"* tablosu üzerinde listeleme işlemi yapma hakkı verilmiştir. Böylece bu rolün atandığı kullanıcı "*tbl_DERS"* tablosu üzerinde "SELECT, INSERT, UPDATE" komutlarını çalıştırabilecektir.

- **ROL BİLGİSİ DEĞİŞTİRME (ALTER ROLE)**

Bir role erişim haklarını değiştirme işlemlerinde kullanılan komuttur.

**Komut Yapısı:**

```
ALTER ROLE <rol ismi> [NOT IDENTIFIED ] | [IDENTIFEID [BY
<şifre>] |EXTERNALLY]]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**NOT IDENTIFIED**: Rolün şifresiz oluşturulmasını sağlamaktadır.
**IDENTIFIED:** Rol için şifrenin belirtildiği kısımdır.
**Örnek:**

"*sifre_kontrol*" isminde ve şifresi "*sifre*" olan rolün şifresini "*yeni_sifre*" şeklinde değiştirme işlemi.

```
ALTER ROLE sifre_kontrol IDENTIFIED BY yeni_sifre;
```

### ➢ ROL BİLGİSİ SİLME (DROP ROLE)

Bu komut ile Rol düşürme işlemi yapılır.

**Komut Yapısı:**

```
DROP ROLE <rol ismi>
```

**Örnek:**

```
DROP ROLE sifre_kontrol;
```

### ➢ İNDİS OLUŞTURMA (CREATE INDEX)

İndeks, tablodaki bilgilere daha hızlı bir erişim için kullanılan nesnelerdir. İndeks oluşturmak için "CREATE ANY INDEX" sistem haklarına sahip olmak gerekir.

**Komut Yapısı:**

```
CREATE [UNIQUE] INDEX <indis ismi>
ON {<tablo ismi>(<kolon isimleri>,...[ASC |DESC])}
[INITRANS    sayı]
[MAXTRANS    sayı]
[STORAGE    <kayıt parametreleri>
           (INITIAL        sayı K | M
            NEXT      sayı K | M
            MINEXTENTS    sayı
            MAXEXTENTS    sayı | UNLIMITED
           PCTINCREASE   sayı
           FREELIST GROUPS sayı )]
[TABLESPACE  <tablo uzayı ismi>]
[PCTFREE     sayı]
[NOSORT]
[RECOVERABLE | UNRECOVERABLE ]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**UNIQUE**: İndekslenecek alanın tekil olacağını belirten bölümümdür.

**İndeks İsmi**: Oluşturulacak indise verilecek isimdir.

**Tablo İsmi**: İndeksin koyulacağı tabloyu belirtir.

**Kolon İsimleri**: İndeksin oluşturulacağı kolonlardır.

**ASC:** İndeksin artan olacağını belirten bölümdür.

**DESC**: İndeksin azalan olacağını belirten bölümdür.

**INITRANS:** İndeks üzerinde aynı anda ilk olarak kaç tane işlem yapılacağının belirtildiği bölümdür.

**MAXTRANS**: İndeks üzerinde aynı anda en fazla kaç işlem yapılacağının belirtildiği bölümdür.

**TABLESPACE**: İndeksin hangi tablo uzayında üzerinde oluşturulacağının belirtildiği bölümdür.

**PCTFREE**: Sonradan yapılacak ekleme ve değiştirmeler için boş olarak ayrılacak yerin bloğa göre yüzdesinin belirtildiği kısımdır.

**NOSORT:** Kayıtlar VT'nında sıralı olarak yer alıyorsa, indis kayıtlarının tekrar sıralanmaması için kullanılan ifadedir.

**RECOVERABLE:** İndeks için yapılan işlemlerin geri alma ihtimaline kaydedilmesini belirten bölümdür.

**UNRECOVERABLE:** İndeks için yapılan işlemlerde geri alma işlemi olmayacağının belirtildiği bölümdür.

**STORAGE <Kayıt Parametreleri>:** İndeksin VT'ndaki kayıtla ilgili parametrelerin belirtildiği bölümdür. Yapısı şu şekildedir:

**INITIAL:** İndeks yaratıldığında ilk aldığı genişlemenin (extent) büyüklüğünü belirten bölümdür. Eğer sayının sonuna K konursa sonuç Kilobayt cinsinden, M konursa Megabayt cinsinden belirtilmiş olmaktadır.

**NEXT:** İndeks yaratıldıktan sonra alacağı genişlemelerin (extent) büyüklüğünü belirten bölümdür.

**MINEXTENTS**: İndeks yaratıldığında ilk olarak alacağı minimum genişleme sayısının belirtildiği bölümdür.

**MAXEXTENTS:** Bir İndeksin ilk olarak aldığı genişleme de dâhil olmak üzere alabileceği en büyük genişleme sayısının belirtildiği bölümdür.

**PCTINCREASE:** İndeksin alacağı genişlemelerden bir sonrakinin bir öncekinden ne kadar büyük olacağını belirten bölümdür. Mesela, 50 yazılırsa, bir sonraki genişleme her zaman bir öncekinden%50 daha büyük olacaktır.

**FREELIST GROUPS**: Bir indis için kaç tane boş liste grubunun olacağının belirtildiği bölümdür.

**Örnek**:

```
CREATE INDEX idx_ogrenci on OGRENCI(adi, soyadi);
```

Bu örnekte "*OGRENCI*" tablosunda "*adi, soyadi*" alanlarına göre indis oluşturma işlemi yapılmıştır.

## ➢ İNDİS BİLGİSİNİ DEĞİŞTİRME (ALTER INDEX)

Bir indisin kayıt parametrelerini değiştirme komutudur.

**Komut Yapısı:**

```
ALTER NDEX <indis ismi>
PCTFREE    sayı
INITRANS   sayı
MAXTRANS   sayı
STORAGE <kayıt parametreleri>
ALLOCATE EXTENT
SIZE sayı K | M
DATAFILE <dosya ismi>
INSTANCE  sayı
[DEALLOCATEUNUSED[KEEP sayı K | M]
REBUILT
```

```
PARALLEL sayı | NOPARALLEL
RECOVERABLE | UNRECOVERABLE
[TABLESPACE <tablo uzayı ismi>]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**İndeks İsmi:** Değiştirilecek olan indis isminin belirtildiği bölümdür.

**PCTFREE:** İndeksin her bloğunda sonradan yapılacak olan değişiklikler için ayrılan boş bölümdür.

**INITRANS:** Her veri bloğunda olması gereken en az toplu işlem sayısını belirtmektedir.

**MAXTRANS**: Her veri bloğunda olabilecek en fazla toplu işlem sayısını belirtmektedir.

**ALLOCATE EXTENT: İ**ndeks için genişleme (extent) ayırma bölümüdür.

**SIZE:** Tanımlanacak genişlemenin (extent) bayt cinsinden büyüklüğünün belirtildiği bölümdür

**DATFILE:** Tanımlanacak genişlemenin (extent) bulunacağı tablo uzayının belirtildiği bölümdür.

**INSTANCE:** Tanımlanacak genişlemenin (extent) belirtilen VT oturumu için geçerli olmasının belirtildiği bölümdür.

**DEALLOCATE UNUSED:** İndeksin sonunda boş kalan yerin bırakılmasını sağlamaktadır.

**KEEP:** Boş alanın bırakılmasından sonra ayrıca kalması istenilen bayt(byte) sayısının belirtildiği bölümdür.

**REBUILD**: İndeksin yeni genişlemeyi kullanarak yeniden yaratılmasını sağlamaktadır.

**PARALLEL**: Yeni indisin paralel olarak yaratılmasını sağlamaktadır.

**NOPARALEL**: Yeni indisin paralel işlem olmadan yaratılmasını sağlamaktadır.

**RECOVERABLE**: İndeks yaratılması işleminin günlük dosyasına yedek alınmasının belirtildiği bölümdür.

**UNRECOVERABLE:** İndeks yaratılması işleminin günlük dosyasına yedek alınmamasının belirtildiği bölümdür.

**TABLESPACE**: Yeni yaratılacak indisin kaydedileceği tablo uzayının belirtildiği bölümdür.

### ➢ İNDİS BİLGİSİNİ SİLME (DROP INDEX)

Bu komut ile İndeks düşürme işlemi yapılır.

**Komut Yapısı:**

```
DROP INDEX <indis ismi>
```

**Örnek:**

```
DROP INDEX idx_ogrenci;
```

### ➢ SIRA BİLGİSİ OLUŞTURMA (CREATE SEQUENCE)

Sıra bilgisi, sıralı olarak artan alanlar için VT'nında saklanan bir nesnedir. Sıra bilgisini oluşturabilmek için "CREATE SEQUENCE" veya "CREATE ANY SEQUENCE" haklarına sahip olmak gerekmektedir.

**Komut Yapısı:**

```
CREATE SEQUENCE <sıra ismi>
[INCREMENT BY     sayı]
[START WITH       sayı]
[MINVALUE         sayı | NOMINVALUE]
[MAXVALUE         sayı | NOMAXVALUE]
[CYCLE | NOCYCLE ]
[CACHE            sayı | NOCACHE]
[ORDER |  NOORDER]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**INCREMENT BY:** Sıra bilgisinin artış sırasını belirtir.
**MINVALUE:** Sıra bilgisinin alabileceği minimum değeri belirtir.
**NOMINVALUE:** Sıra bilgisinde artan sıralar için minimum değerin 1, azalan sıralar için ise minimum değerin $-10^{26}$ olduğu belirtir. Standart olarak seçilidir.
**MAXVALUE**: Sıra bilgisinin alabileceği maksimum değeri belirtir.
**NOMAXVALUE:** Sıra bilgisinde artan sıralar için minimum değerin $10^{27}$, azalan sıralar için ise minimum değerin -1 olduğu belirtir. Standart olarak seçilidir.
**START WITH:** Sıra bilgisinin başlangıç değerini belirtir.
**CYCLE:** Sıra bilgisinin minimum veya maksimum değerlerine ulaşınca sayı üretmeye devam edeceğini belirtir.
**NOCYCLE:** Sıra bilgisinin minimum veya maksimum değerlerine ulaşınca sayı üretmeye devam etmeyeceğini belirtir. Standart olarak seçilidir.
**CACHE**: Sıra bilgisine erişmek için belleğe alınan Sıra bilgisi miktarını gösterir.
**NOCACHE**: Belleğe, Sıra bilgisinin alınmayacağını belirtir.
**ORDER**: Sıra bilgisinin numaralarının sıralı olacağını belirtir.
**NOORDER**: Sıra bilgisi numaralarının sıralı olarak gelmeyeceğini belirtir. Standart olarak seçilidir.
**Örnek – 1:**

```
CREATE SEQUENCE orenci_seq
START WITH 1
INCREMENT BY 1
ORDER;
```

Yukarıdaki örnek, 1 den başlayıp 1'er 1'er artan "*ogrenci_seq*" isminde bir sırası numarası oluşturur.

### ➢ SIRA BİLGİSİNİ DEĞİŞTİRME (ALTER SEQUENCE)

Bir sıra için numara üretme işlemini yeniden tanımlama komutudur.

**Komut Yapısı:**

```
ALTER SEQUENCE <sıra ismi>
{INCREMENT BY sayı }
[MINVALUE      sayı | NOMINVALUE]
[MAXVALUE      sayı | NOMAXVALUE]
[CYCLE | NOCYCLE ]
[CACHE         sayı | NOCACHE]
[ORDER |  NOORDER]
```

**Örnek:**

```
ALTER SEQUENCE ogrenci_seq
MAXVALUE 2000;
```

Yukarıdaki örnek, ”ogrenci_seq” ismindeki sıranın alabileceği en büyük sayıyı 2000 yapar.

### ➢ SIRA BİLGİSİNİ SİLME (DROP SEQUENCE)

Bu komut ile Sıra bilgisi düşürme işlemi yapılır.

**Komut Yapısı:**

```
DROP SEQUENCE <sıra ismi>
```

**Örnek:**

```
DROP SEQUENCE ogrenci_seq;
```

### ➢ GERİ ALMA PARÇASI YARATMA (CREATE ROLLBACK SEGMENT)

Geri alma parçaları SELECT, INSERT, DELETE, UPDATE gibi komutlarla yapılan işlemlerin gerektiğinde geri alınabilmesi için VT’nında ayrılan alanlara denir.

**Komut Yapısı:**

```
CREATE [PUBLIC] ROLLBACK SEGMENT <geri alma parçası ismi >
[TABLESPACE        <tablo uzayı ismi>]
[STORAGE           <kayıt parametreleri >]
[OPTIMAL           [NULL | TO sayı K | M]
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**PUBLIC**: Geri alma parçasını bütün VT bölümlerinin (instance) kullanılabileceğini gösteren bölümdür.
Geri alma parçası ismi: Yaratılan geri alma parçasına verilen isimdir.
**TABLESPACE**: Geri alma parçasının hangi tablo uzayındaki yeri kullanılacağını belirten bölümdür.
**STORAGE**: Kayıt parametrelerinin belirtildiği bölümdür.
**OPTIMAL**: Geri alma parçası için ortalama bir değerin belirtildiği bölümdür. Geri alma parçası büyüdükten sonra, içindeki veriler kullanılmaz hale geldiğinde, Oracle geri alma parçasını bu uygun değer değere kadar küçültür.

**Örnek:**

"tbsp_deneme" tablo uzayını kullanan, kayıt parametrelerinden ilk parçanın büyüklüğü 10M, sonraki genişlemelerin büyüklüğü 1M, minimum genişleme sayısı 2, maksimum genişleme sayısı 121 ve ortalama değeri 30M olan ve ismi "rol_seg_deneme" olan bir geri alma parçası şöyle oluşturulabilir:

```
CREATE ROLLBACK SEGMENT rol_seg_deneme
TABLESPCE tbsp_deneme
STORAGE(
          INITIAL  10M
          NEXT      1M
          MINEXTENTS 2
```

```
        MAXEXTENTS    121
        OPTIMAL    30M);
```

- **HAK VERME (GRANT)**

Oracle da bulunun belirli tipteki nesne veya SQL cümlelerini çalıştırabilme ve başkalarının nesnelerine erişebilme olanağı verme işlemidir. Oracle' da iki çeşit hak vardır. Bunlar:

Sistem Hakları(System Privileges): VT ile ilgili ve kullanıcılara verilecek olan önceden tanımlanmış rollerdir.

Nesne Hakları(Object Privileges) :Kullanıcılara ait VT nesnelerine erişme hakkıdır.

Sistem hakları önceki bölümde anlatılmış ve liste olarak Ek-1'de gösterilmiştir. Nesne hakları 8(sekiz) çeşittir. Bunlar:

- **SELECT**: Tablo ve görüntülerin içindeki kayıtlara bakma hakkıdır.
- **INSERT** : Tablo ve görüntülere kayıt ekleme hakkıdır.
- **UPDATE**: Tablo ve görüntülerin içindeki hakları değiştirme hakkıdır.
- **DELETE**: Tablo ve görüntülerin içindeki hakları silme hakkıdır.
- **ALTER**: Nesnelerin yapısını değiştirme hakkıdır.
- **INDEX**: Tablo ve görüntülere indis tanımlayabilme hakkıdır.
- **EXECUTE**: Yordam ve fonksiyonları çalıştırma hakkıdır.
- **REFERENCES**: Yabancı anahtar tanımlayabilme hakkıdır.
- **ALL**: Yukarıdaki haklarının bütününün yerine geçen bir ifadedir.

**Komut Yapısı:**

```
GRANT nesne_hakları ON nesne_ismi TO kullanıcı_ismi
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**GRANT <nesne hakları>:** Kullanıcıya verilecek haklardır.

**ON <nesne ismi> :** Hakkın hangi nesne (tablo, görüntü, sıra, eşanlam, yordam, fonksiyon) ile ilgili olacağının belirtir.

**TO<kullanıcı ismi> :** Hakkın hangi kullanıcıya verileceğinin belirtildiği bölümdür.

**Örnek – 1:**

Bu kullanım şekli ile belirtilen kullanıcıya, belirtilen nesne üzerinde, belirtilen haklar verilmektedir.

```
Grant [hak][,.......],[rol][,...] To
[kullanıcı][,.....],[Rol][,...]
```

**Örnek – 2:**

```
GRANT SELECT ON SCOTT.OGRENCI TO ALP;
```

Bu örnekte "*SCOTT*" kullanıcısın sahip olduğu "*OGRECI*" tablosuna bakma hakkını "*ALP*" kullanıcısına vermektedir.

**Örnek – 3:**

```
GRANT EXECUTE ON SCOTT.NOT_HESAPLA TO ALP;
```

Bu örnekte "*SCOTT*" kullanıcısının "*NOT_HESAPLA*" isimli kaydedilmiş yordamını "*ALP*" kullanıcısına çalıştırma hakkı vermektedir.

- **HAK ALMA (REVOKE)**

Verilen sistem veya kullanıcı haklarının geri alınması için kullanılan komuttur.

**Komut Yapısı:**

```
REVORKE nesne_hakları ON nesne_ismi FROM kullanıcı_ismi;
```

Komutta kullanılan ifadeler aşağıda açıklanmıştır:

**REVOKE <nesne hakları> :** Kullanıcıdan alınacak haklardır.

**ON<nesne ismi>:** Alınacak hakkın hangi nesne ile ilgili olacağının belirtildiği bölümdür.

**FROM <kullanıcı ismi >:** Hakkın hangi kullanıcıdan alınacağının belirtildiği bölümdür.

**Örnek:**

```
REVOKE SELECT ON SCOTT.OGRENCI FROM ALP;
```

"*SCOTT*" kullanıcısının "*OGRENCI*" tablosuna "*ALP*" kullanıcısının bakma hakkını geri alma komutudur.

**EK B:** VT performansına yönelik geliştirilen SQL cümleleri ve sonuçları aşağıda verilmiştir.

SQL 1:

Eğer "BUFFER HIT RATIO" değeri 70'den büyük ise "init.ora" dosyasından "***db_block_buffers***" parametresi artırılması gerekmektedir.

```
SELECT a.VALUE + b.VALUE "logical_reads", c.VALUE "phys_reads",
       d.VALUE "phy_writes",
       ROUND (100 * ((a.VALUE + b.VALUE) - c.VALUE) / (a.VALUE +
       b.VALUE) ) "BUFFER HIT RATIO"
  FROM v$sysstat a, v$sysstat b, v$sysstat c, v$sysstat d
 WHERE a.statistic# = 37
   AND b.statistic# = 38
   AND c.statistic# = 39
   AND d.statistic# = 40;
```

```
logical_reads phys_reads phy_writes BUFFER HIT RATIO
------------- ---------- ---------- ----------------
      1764891 6.0798E+10    1017191         -3444775
```

SQL 2:

Eğer "DATA DICT CACHE HIT RATIO" değeri 90'dan büyük ise "init.ora" dosyasından "***shared_pool_size***" parametresi artırılması gerekmektedir.

```
SELECT SUM (gets) "Data Dict. Gets",
       SUM (getmisses) "Data Dict. cache misses",
       TRUNC ((1 - (SUM (getmisses) / SUM (gets))) * 100
             ) "DATA DICT CACHE HIT RATIO"
  FROM v$rowcache;
```

```
Data Dict. Gets Data Dict. cache misses DATA DICT CACHE HIT RATIO
--------------- ----------------------- -------------------------
       10627015                 1101048                        89
```

SQL 3:

Eğer "LIBRARY CACHE MISS RATIO" değeri %1'dan büyük ise "init.ora" dosyasından "***shared_pool_size***" parametresi artırılması gerekmektedir.

```
SELECT SUM (pins) "executions",
       SUM (reloads) "Cache misses while executing",
       (((SUM (reloads) / SUM (pins)))) "LIBRARY CACHE MISS RATIO"
  FROM v$librarycache;
```

SQL 4:

| “HIT RATIO” ve “PIN HIT RATIO” 70 değerinden büyük olmalıdır. |
|---|

```
SELECT namespace, TRUNC (gethitratio * 100) "Hit ratio",
       TRUNC (pinhitratio * 100) "pin hit ratio", reloads "reloads"
  FROM v$librarycache;
```

```
NAMESPACE         Hit ratio pin hit ratio    reloads
--------------- ---------- ------------- ----------
SQL AREA                 21            97      16917
TABLE/PROCEDURE          90            94      28734
BODY                     95            99        478
TRIGGER                  98            97        497
INDEX                    41            77        245
CLUSTER                  99            99         47
OBJECT                  100           100          0
PIPE                     98            99          0
JAVA SOURCE             100           100          0
JAVA RESOURCE           100           100          0
JAVA DATA                66            99          0
```

SQL 5:

| “REDO LOG” için gerekli olan boşluk değeri |
|---|

```
SELECT SUBSTR (NAME, 1, 30), VALUE
  FROM v$sysstat
 WHERE NAME = 'redo log space requests';
```

```
SUBSTR(NAME,1,30)                      VALUE
------------------------------ ----------
redo log space requests                377
```

SQL 6:

| Boş bellek değeri |
|---|

```
SELECT NAME, BYTES
  FROM v$sgastat
 WHERE NAME = 'free memory';
```

```
NAME                                 BYTES
-------------------------- ----------
free memory                     44821748
```

```
free memory                    3120304
free memory                     390144
```

SQL 7:

VT açıldığından ve kullanıcı bağlandığından beri çalıştırılan SQL değerleri toplamı

```
SELECT SUM (executions) "Total SQL run since startup",
       SUM (users_executing) "SQL executing now"
  FROM v$sqlarea;
```

```
Tot SQL run since startup SQL executing now
------------------------- -----------------
                  1404816                27
```

SQL 8:

Kilit ve kilitlenme durumlarının listesi

```
SELECT SUBSTR (username, 1, 12) "User", SUBSTR (lock_type, 1, 18)
"Lock Type",
       SUBSTR (mode_held, 1, 18) "Mode Held"
  FROM SYS.dba_lock a, v$session b
 WHERE lock_type NOT IN ('Media Recovery', 'Redo Thread')
   AND a.session_id = b.SID;
```

```
User         Lock Type          Mode Held
------------ ------------------ ------------------
             XR                 Null
             Control File       Row-S (SS)
             RS                 Row-S (SS)
MERNISDW     PS                 Share
REPOWNER     PL/SQL User Lock   Exclusive
REPOWNER     PL/SQL User Lock   Share
MERNISDW     PS                 Share
             Temp Segment       Row-X (SX)
MERNISDW     PS                 Share
MERNISDW     PS                 Share
REPOWNER     PL/SQL User Lock   Share
MERNISDW     PS                 Share
MERNISDW     PS                 Share
MERNISDW     PS                 Share
MERNISDW     PS                 Share
```

```
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     TO                  Row-X (SX)
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     PS                  Share
MERNISDW     TO                  Row-X (SX)
MERNISDW     Transaction         Exclusive
```

SQL 9:

Eğer “miss_ratio” veya “immediate_miss_ratio” 1 den büyük ise kilitlenme durumu vardır bu durumda “init.ora” dosyasından “*LOG_SMALL_ENTRY_MAX_SIZE*” parametresi azaltılması gerekmektedir.

```
SELECT    SUBSTR (l.NAME, 1, 30) NAME,
          (misses / (gets + .001)) * 100 "miss_ratio",
            (immediate_misses / (immediate_gets + .001)
            )
```

```
          * 100 "immediate_miss_ratio"
    FROM v$latch l, v$latchname LN
   WHERE l.latch# = LN.latch#
     AND (    (misses / (gets + .001)) * 100 > .2
          OR (immediate_misses / (immediate_gets + .001)) * 100 > .2
         )
ORDER BY l.NAME;
```

```
NAME                           miss_ratio immediate_miss_ratio
------------------------------ ---------- --------------------
active service list            ,.4494E+39                   0
client/application info        ,.5134E+40                   0
dummy allocation               4.,420E+38                   0
FOB s.o list latch             ,.2623E+40                   0
KTF sga latch                  ,.2959E+40           ,.0075E+42
library cache                  ,.0527E+40           2.,747E+39
loader state object freelist   ,.6459E+40                   0
Memory Management Latch        ,.2028E+40                   0
messages                       ,.3129E+40                   0
parameter table allocation man ,.4131E+40                   0
process queue reference        ,.0064E+42           8.,529E+39
query server freelists         2.,277E+39                   0
resmgr group change latch      2.,430E+39                   0
resmgr:free threads list       5.,868E+39                   0
session state list latch       6.,364E+38                   0
slave class create             1.1,61E+40                   0
SQL memory manager latch       ,.5102E+40                   0
user lock                      ,.7538E+40                   0
```

SQL 10:

Alınan sonuçtaki herhangi bir değer 1 den büyük ise veri için daha fazla “Rollback Segment” ihtiyaç duyulur.

```
SELECT   CLASS, COUNT
    FROM v$waitstat
   WHERE CLASS IN
            ('free list',
             'system undo header',
             'system undo block',
             'undo header',
             'undo block'
            )
```

```
GROUP BY CLASS, COUNT;
```

```
CLASS                    COUNT
------------------ ----------
undo header                27
system undo block           0
system undo header          0
free list                   0
undo block                 12
```

SQL 11:

Alınan sonuçlardan herhangi biri 0.01 den büyük ise “Roolback Segment” artırmak gerekir.

```
SELECT NAME, waits, gets, waits / gets "Ratio"
  FROM v$rollstat a, v$rollname b
 WHERE a.usn = b.usn;
```

```
NAME                      WAITS       GETS      Ratio
------------------------- ---------- ---------- ----------
SYSTEM                            0       1847          0
_SYSSMU1$                         1      51812 ,.0000E+43
_SYSSMU2$                         0      67028          0
_SYSSMU3$                         0      50381          0
_SYSSMU4$                         2      57912 ,.0000E+44
_SYSSMU5$                         2      41491 ,.0000E+44
_SYSSMU6$                         5      64980 ,.0001E+44
_SYSSMU7$                         2      56792 ,.0000E+43
_SYSSMU8$                         3      53986 ,.0001E+44
_SYSSMU9$                         2      60986 ,.0000E+44
_SYSSMU10$                        1      57212 ,.0000E+44
```

SQL 12:

Toplam bekleme zamanlarını görmek

```
SELECT SUBSTR (event, 1, 30) event, total_waits, total_timeouts,
average_wait
  FROM v$session_event
 WHERE average_wait > 0;
```

| EVENT | TOTAL_WAITS | TOTAL_TIMEOUTS | AVERAGE_WAIT |
|---|---|---|---|
| log file sync | 2 | 0 | ,04 |
| db file sequential read | 56 | 0 | ,44 |
| db file scattered read | 1 | 0 | 1,53 |
| SQL*Net more data to client | 26 | 0 | ,01 |
| SQL*Net message from client | 150 | 0 | 91,55 |
| SQL*Net break/reset to client | 4 | 0 | ,21 |
| cursor: pin S wait on X | 31 | 30 | 1,47 |
| PX Deq: Msg Fragment | 1 | 0 | ,12 |
| PX Deq: Execution Msg | 8082 | 8080 | 199,91 |
| events in waitclass Other | 3 | 2 | ,02 |
| events in waitclass Other | 1 | 0 | ,12 |
| events in waitclass Other | 2 | 0 | ,03 |
| events in waitclass Other | 2 | 0 | ,03 |
| SQL*Net message from client | 6 | 0 | 7,93 |
| cursor: pin S wait on X | 31 | 30 | 1,46 |
| PX Deq: Msg Fragment | 1 | 0 | ,08 |
| PX Deq: Execution Msg | 8082 | 8080 | 199,91 |
| db file sequential read | 1 | 0 | ,11 |
| cursor: pin S wait on X | 32 | 30 | 1,42 |
| PX Deq: Execution Msg | 8086 | 8083 | 199,88 |
| SQL*Net message from client | 168 | 0 | 13898,4 |

SQL 13:

Kuyruklar için ortalama bekleme, sıfırın yakınında olmalıdır.

```
SELECT paddr, TYPE "Queue type", queued "# queued", WAIT, totalq,
       DECODE (totalq, 0, 0, WAIT / totalq) "AVG WAIT"
  FROM v$queue;
```

| PADD | Queue type | # queued | WAIT | TOTALQ | AVG WAIT |
|---|---|---|---|---|---|
| 00 | COMMON | 0 | 0 | 0 | 0 |
| 2124 | DISPATCHER | 0 | 0 | 0 | 0 |

**EK C:** Tez kapsamında geliştirlen DBAExplorer programının kaynak kodları.

# ÖZGEÇMİŞ

## Kişisel Bilgiler


| | |
|---|---|
| Soyadı, adı | : KILIÇKAYA, Alpaslan |
| Uyruğu | : T.C. |
| Doğum tarihi ve yeri | : 16.01.1976 Kayseri |
| Medeni hali | : Evli |
| Telefon | : 0 (312) 363 18 43 |
| Faks | : |
| e-mail | : akilickaya@etu.edu.tr<br>alpaslankilickaya@hotmail.com |


## Eğitim

| Derece | Eğitim Birimi | Mezuniyet Tarihi |
|---|---|---|
| Lisans | Anadolu Üniversitesi İktisat Fakültesi Çalışma Ekonomisi ve Endüstri İlişkileri | 2005 |
| Ön Lisans | Erciyes Üniversitesi/Bilgisayar Prog. | 1996 |

## İş Deneyimi

| Yıl | Yer | Görev |
|---|---|---|
| 2008- | Kale Yazılım Ltd | Uzman |
| 2002-2008 | KoçSistem A.Ş. | Analiz Tasarım Uzmanı |
| 2000-2002 | Likom Yazılım A.Ş. | Yazılım Mühendisi |
| 1997-2000 | T.C. E.G.M. Polis Bakım ve Yardım San. | Bilgi İşlem Şefi |

## Yabancı Dil

İngilizce