ÇEVRE VE ŞEHİRCİLİK BAKANLIĞI MERKEZİ ORACLE VERİTABANLARININ PERFORMANS VE GÜVENLİK YÖNÜNDEN İNCELENMESİ VE İYİLEŞTİRME ÖNERİLERİ

UZMANLIK TEZİ

HAZIRLAYAN: RAŞİT ÖZCAN

ANKARA-2017

T.C.

ÇEVRE VE ŞEHİRCİLİK BAKANLIĞI

ÇEVRE VE ŞEHİRCİLİK BAKANLIĞI MERKEZİ ORACLE VERİTABANLARININ PERFORMANS VE GÜVENLİK YÖNÜNDEN İNCELENMESİ VE İYİLEŞTİRME ÖNERİLERİ

Tez Hazırlayanın Adı Soyadı	:	Raşit ÖZCAN
Tez Danışmanın Adı Soyadı	:	Cem Burak URAL
Birim Amirinin Adı Soyadı	:	Ömer ALAN

Raşit Özcan tarafından hazırlanan "Çevre ve Şehircilik Bakanlığı Merkezi Oracle Veritabanlarının Performans ve Güvenlik Yönünden İncelenmesi ve İyileştirme Önerileri" adlı bu tezin Çevre ve Şehircilik Uzmanlık tezi olarak uygun olduğunu onaylarım.

Çevre ve Şehircilik Uzmanı,
Cem Burak URAL
Tez Danışmanı

Bu çalışma, tez savunma komisyonumuz tarafından Çevre ve Şehircilik Uzmanlık tezi olarak kabul edilmiştir.

Başkan : Genel Müdür V., Ömer, ALAN

Üye : Genel Müdür Yrd. V., Recep, YILDIRIM

Üye : Daire Başkanı V., İskender, ERMİŞ

Üye : Daire Başkanı V., Sibel, ASLAN

Üye : Çevre ve Şeh. Uzmanı, Cem Burak URAL

Bu tez, Çevre ve Şehircilik Uzmanlığı Tez Hazırlama Yönergesi'ne uygundur.

İÇİNDEKİLER

ÖZET

ÇEVRE ve ŞEHİRCİLİK BAKANLIĞI	
Tezin Adı	Çevre Şehircilik Bakanlığı Merkezi Oracle Veritabanlarının Performans ve Güvenlik Yönünden İncelenmesi ve İyileştirme Önerileri
Türü	Çevre ve Şehircilik Bakanlığı Uzmanlık Tezi
Yazar	Raşit ÖZCAN
Teslim Tarihi	06.03.2017
Anahtar Kelimeler	Veritabanı, performans, güvenlik
Tez Danışmanı	Cem Burak URAL
Sayfa Adedi	140

Özet

Veritabanı yönetim sistemleri büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkanlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkanının sağlandığı programlardır. Veritabanları büyük miktardaki kurumsal verileri işleyerek çok sayıda uygulamanın ve kullanıcının hizmetine sunar. Büyük miktarlardaki verilerin hızlı ve güvenli bir biçimde gereksinim duyulan bilgiye dönüştürülmesi veritabanlarının en önemli özelliklerinden birisidir. Bilişim projelerinde, veritabanlarında performans ve güvenlikte aksaklık olmaması, kurumun itibarı ve projenin başarısı için hayati önem taşımaktadır.

Bu tez çalışması içerisinde bakanlığımızda en yaygın kullanılan ve en büyük veritabanları olan Merkezi Oracle Veritabanları (CSBRAC) ele alınmış olup, kurum veritabanlarının performans parametreleri ve güvenlik kriterleri açısından incelenmesi, potansiyel güvenlik risklerinin belirlenmesi ve performans iyileştirmesi önerileri üretilerek kurumun vermiş olduğu hizmet kalitesinin artırılması hedeflenmektedir.

ABSTRACT

MINISTRY OF ENVIRONMENT AND URBANIZATION	
Thesis	Investigation and Improvement Suggestions for Ministry of Environment and Urbanization Central Oracle Databases
Type	Ministry of Environment and Urbanization Expertise Thesis
Author	Raşit ÖZCAN
Submission Date	06.03.2017
Key Words	Database, performance, security
Advisor	Cem Burak URAL
Total Page	140

Abstract

Database management systems are programs in which large amounts of data can be held securely, fast access to information is provided, information can be kept in integrity, and information access is available to multiple users at the same time. Databases serve large numbers of applications and users by processing large amounts of enterprise data. One of the most important features of databases is the conversion of large amounts of data into the required information quickly and safely. The lack of performance and security flaws in the databases of the information projects is of vital importance for the institution's reputation and the success of the project.

Central Oracle Database (CSBRAC), which is one of the most widely used and largest databases in our ministry, is examined in this thesis study. It is aimed to examine the institutional databases in terms of performance parameters and security criteria, to identify potential security risks and to improve the service quality that the institution has given by producing performance improvement proposals.

TEŞEKKÜR

Tez çalışması boyunca desteğini esirgemeyen başta Eşim Gözde Özcan'a, Daire Başkanım İskender Ermiş'e, tez danışmanım Cem Burak Ural'a ve mesai arkadaşlarıma teşekkür eder, tezin Bakanlığımız ve diğer meslektaşlarım için faydalı olmasını temenni ederim.

(İmza)

Raşit ÖZCAN

TABLO LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR

ADDM : Automatic Database Diagnostic Monitor
ASM : Automatic Storage Management
AES : Advanced Encryption Standard
ANSI : American National Standards Institute
API : Application Programming Interface
ASO : Advanced Security Option
ASSM : Automatic Segment Space Management
AWR : Automatic Workload Repository
BLOB : Binary Large Objects
CBO : Cost Based Optimizer
CLOB : Character Large Object
CP : Check Point
CPU : Central Processing Unit
DBA : Database Administrator
DDL : Data Defination Language
DES : Data Encryption Standart
DML : Data Manupulation Language
DNS : Domain Name Service
DOP : Degree Of Parallelism
DSS : Decision Support System
DV : Database Vault
DoS : Denial of Service
EM : Enterprise Manager
FGA : Fine-Grained Auditing
FTP File : Transfer Protocol
G/Ç : Giriş/Çıkış
HTTP : Hyper Text Transfer Protocol
IP : Internet Protocol
LOB : Large Objects
LRU : Least Recent Used
MPS : Massively Parallel Systems
NCLOB : National Character Large Object
NDV : Number of Distinct Values
OLAP : Online Analytical Processing
OLTP : Online Transaction Processing
PGA : Program Genel Alanı
PL/SQL : Procedural Language Extension of SQL
PSP : Parallel Service Process
QC : Query Coordinator
RAC : Real Application Clusters
RDA : Remote Diagnostics Agent
RR : Round Robin
SCAN : Single Client Access Name
SCN : System Change Number
SGA : Sistem Genel Alanı
SID : Security Identifier

SQL : Structured Query Language
SPM : Sql Plan Management
SSL : Secure Sockets Layer
TCP : Transmission Control Protocol
TDE : Transparent Data Encryption
TNS : Transparent Network Substrate
VLDB : Very Large Database
VPD : Virtual Private Database
VTYS : Veritabanı yönetim sistemi
UGA : User Global Area

GİRİŞ

Veritabanı Yönetim Sistemleri yazılım projelerinde kullanılan verileri saklamak, doğru ve etkin şekilde sunmak için kullanılan sistemlerdir. İlişkisel veritabanı yönetim sistemleri büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkanlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkanının sağlandığı programlardır. Veritabanları büyük miktarda bilginin girilmesi, depolanması ve alınmasına izin veren çeşitli işlevler sunar ve bu bilgilerin nasıl organize edildiğini yönetmek için yollar sağlar.[31] Veritabanları büyük miktardaki kurumsal verileri işleyerek çok sayıda uygulamanın ve kullanıcının hizmetine sunar. Büyük miktarlardaki verilerin hızlı ve güvenli bir biçimde gereksinim duyulan bilgiye dönüştürülmesi veritabanlarının en önemli özelliklerinden birisidir.

Günümüzde herhangi bir veritabanı yönetim sisteminin kullanılmadığı bilişim projelerine artık rastlanmamaktadır. Kurumlar verdikleri hizmetleri özellikle son yıllarda artan bir yatırımla bilişim alt yapısına taşımaktadır. Bu da bilişimin temel taşı olan verilerin saklandığı ortam olan veritabanlarının önemini artırmaktadır. Günümüzde veri tabanı sistemleri bankacılıktan otomotiv sanayisine, sağlık bilgi sistemlerinden şirket yönetimine, telekomünikasyon sistemlerinden hava taşımacılığına, çok geniş alanlarda kullanılan bilgisayar sistemlerinin alt yapısını oluşturmaktadır.

Bakanlığımız sorumluluk alanında bulunan görev ve süreçleri yönetmek için kullanılan yazılım projelerinde de veriler kurum bünyesinde barındırılan merkezi veritabanlarında tutulmakta ve oradan servis edilmesi sağlanmaktadır. Coğrafi uygulamalar başta olma üzere, artan ve yaygınlaşan proje sayısıyla birlikte zamanla büyüyen veri hacmi, kurum veritabanlarının yönetimini hassasiyetle ele alınması ve zaafiyet gösterilmemesi gereken bir konu haline getirmiştir. Yazılım projelerinin kurum kaynakları açısından maliyeti de göz önüne alındığında, üretilen verilerin güvenlik ve performans kriterleri açısından başarılı bir şekilde tutulması ve servis edilmesi bakanlığımızın vermiş olduğu hizmet memnuniyeti başta olmak üzere birçok açıdan önemlidir. Bu tezle, kurum veritabanlarının performans parametreleri ve güvenlik kriterleri açısından incelenmesi, potansiyel güvenlik risklerinin

belirlenmesi ve performans iyileştirmesi önerileri üretilerek kurumun vermiş olduğu hizmet kalitesinin artırılması hedeflenmektedir.

Bu tez; giriş ile sonuç bölümleri haricinde dört bölümden oluşmaktadır. Bu çalışmamın birinci bölümünde, performans ve güvenlik problemlerinde etkin çözümler üretebilmek için Oracle veritabanı mimarisi ile ilgili temel bir bilgi dağarcığı oluşturmak amaçlanmıştır.

İkinci bölümde, veritabanlarının kullanıcılara daha hızlı, doğru ve kesintisiz hizmet verebilmesi için, tablo tasarımının etkin olması, veritabanı nesnelerinin performanslı biçimde kullanabilmek, bir sorgu veritabanına geldiğinde çalışma aşamaları, yoğun kaynak tüketen SQL deyimlerinin belirlenmesi ve SQL deyimlerini performanslı biçimde oluşturma, veritabanı performans araçları gibi konular işlenecek olup, altbölümlerin konularına göre kurum veritabanında inceleme yapılacak ve öneriler sunulacaktır.

Üçüncü bölümde, veritabanı güvenliği zafiyetleri ve alınması gereken önlemler işlenecek olup, altbölümlerin konularına göre kurum veritabanında inceleme yapılacak ve öneriler sunulacaktır.

Dördüncü bölümde, kurum merkezi oracle veritabanlarının genel olarak analizi yapılacak olup iyileştirme önerileri sunulacaktır.

Son bölümde ise genel değerlendirme yapılacaktır.

1. ORACLE VERİTABANI MİMARİSİ

Bir veritabanı sunucusu, bilgi yönetiminin anahtarıdır. Genel olarak, bir sunucu, kullanıcıların aynı verilere aynı anda erişebilmesi için çok kullanıcılı bir ortamda büyük miktarda veriyi güvenilir şekilde yönetir. Bir veritabanı sunucusu, yetkisiz erişimleri önlediği gibi aksaklıklarda veri kurtarabilmek için etkin çözümler sağlar. Bu bölümde bir Oracle veritabanınındaki bileşenler olan veritabanı olgusu, depolama yapıları, süreç yapıları, bellek yapıları, veritabanı ile uygulama sunucuları arasındaki ilişkiyi yansıtan mimari ve istemcilerin veritabanına bağlanabilmesi için ağ mimarisi açıklanacaktır.

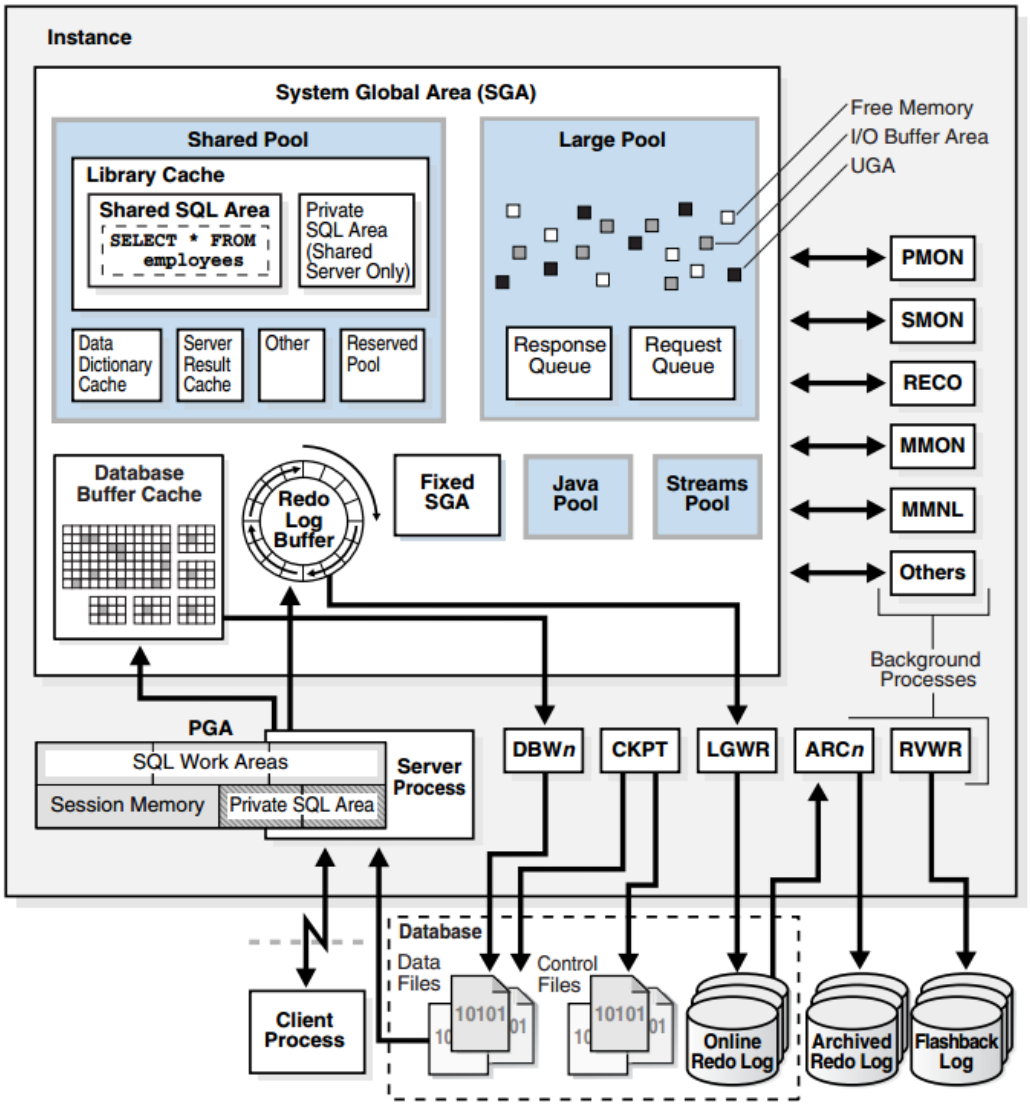
1.1 Veritabanı ve Veritabanı Olgusu (Instance)

Bir Oracle veritabanı sunucusu, bir veritabanı ve en az bir veritabanı olgusundan oluşur. Bir veritabanı olgusu ve bir veritabanı çok yakından bağlı olduğundan, Oracle veritabanı terimi bazen hem veritabanı olgusunu hem de veritabanını belirtmek için kullanılır. En belirgin şekilde terimlerin anlamları şöyledir:

- **Veritabanı :** Veritabanı lokasyonu diskte bulunan verileri depolayan dosyalar kümesidir.
- **Veritabanı Olgusu :** Veritabanı olgusu veritabanı dosyalarını yöneten bir bellek yapıları kümesidir. Veritabanı olgusu sistem genel alanı (SGA) olarak adlandırılan paylaşılan bellek alanından ve arka plan işlem kümesinden oluşur. Veritabanı olgusu veritabanı dosyalarından bağımsız olarak bulunabilir.

Şekil 1.1, bir veritabanını ve örneğini göstermektedir. Örneğe yapılan her kullanıcı bağlantısı için, uygulama bir istemci işlemi tarafından çalıştırılır. Her istemci işlemi, kendi sunucu işlemiyle ilişkilendirilir. Sunucu işlemi, program genel alanı (PGA) olarak bilinen kendi özel oturum belleğine sahiptir.

Şekil 1.1 : Oracle Veritabanı Mimarisi



[1]

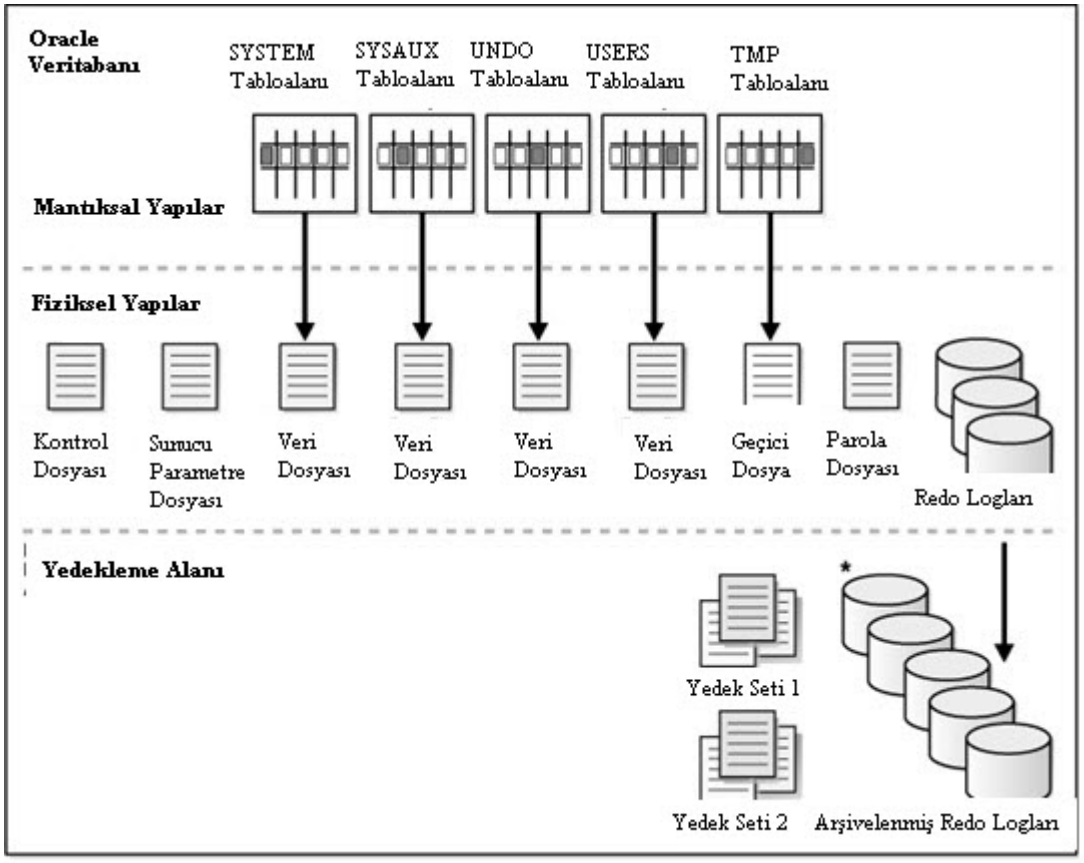
Veritabanı hem fiziksel hem de mantıksal açıdan değerlendirilebilir. Fiziksel veriler, işletim sistemi düzeyinde görüntülenebilir verilerdir. Örneğin, Linux ls ve ps gibi işletim sistemi yardımcı programları, veritabanı dosyalarını ve işlemlerini listeleyebilir. Bir tablo gibi mantıksal veriler yalnızca veritabanı için anlamlıdır. Bir SQL deyimi tabloları bir Oracle veritabanında listeleyebilirken işletim sistemi yardımcı programı listeleyemez.

Fiziksel ve mantıksal yapılar birbirinden ayrı olduğundan, verilerin fiziksel depolama alanı, mantıksal depolama yapılarına erişimi etkilemeksizin yönetilebilir. Örneğin, bir fiziksel veritabanı dosyasını yeniden adlandırmak, bu dosyada veri saklanan tabloları yeniden adlandırmaz.

1.2 Veritabanı Depolama Yapıları

İlişkisel bir veritabanının vazgeçilmez bir görevi, veri depolamasıdır. Bu bölümde, Oracle Veritabanı tarafından kullanılan fiziksel ve mantıksal depolama yapıları kısaca açıklanacaktır.

Şekil 1.2: Oracle Veritabanı Depolama Yapıları



1.2.1. Fiziksel Depoloma Yapıları

Fiziksel veritabanı yapıları, verileri depolayan dosyalardır. SQL komutu CREATE DATABASE komutu çalıştırıldığında Veri Dosyaları (Data Files) ve Geçici Dosyaları (Temp Files), Kontrol Dosyaları (Control Files), Çevrimiçi Redo Günlüğü Dosyaları (Online Redo Log Files) oluşur.

1.2.1.1 Veri Dosyaları (Data Files) ve Geçici Dosyalar (Temp Files)

Her Oracle veritabanında, tüm veritabanı verilerini içeren bir veya daha fazla fiziksel veri dosyası bulunur. Bir Oracle veri dosyası disk üzerinde işletim sistemi seviyesinde dosya olarak saklanır. Her bir veri dosyası sadece bir tabloalanın üyesidir. Bununla birlikte bir tablo alanının birden fazla veri dosyası olabilir.

Mantıksal veritabanı yapılarının tablolar ve indeksler gibi verileri fiziksel olarak veri dosyalarında saklanır. Geçici dosya, geçici bir tablo alanına (Temporary Tablespace) ait olan bir veri dosyadır. Veriler bu dosyalara Oracle'dan başka hiçbir programın okuyamayacağı özel bir formatta yazılır.

Yönetim kolaylığı açısından Oracle Veritabanı, mantıksal depolama yapıları gibi bölümlerdeki kullanıcı verileri için alan ayırır. Her bölüm (segment) yalnızca bir tablo alanına aittir. Örneğin bölümlenmemiş tablonun verisi tek bir bölümde saklanır.

Her bir veri dosyası çevrimiçi (erişilebilir) veya çevrimdışı (erişilemez) durumundadır. Veritabanı yöneticileri çevrimdışı yedek alma, veri dosyasının adını değiştirme veya blok bozulması gibi işlerde, veri dosyalarını veya geçici dosyalarını çevrimdışı haline getirebilir.

Veritabanı yöneticisi, veri dosyasını AUTOEXTEND parametresi ile oluşturduysa, diskteki boş alan tükendiğinde bir Oracle veri dosyası otomatik olarak genişletilecektir. Veritabanı yöneticisi, MAXSIZE parametresini kullanarak belirli bir veri dosyası için genişletme miktarını da sınırlandırabilir. Her durumda, veri dosyasının boyutu, en sonunda, bulunduğu disk hacmi ile sınırlanır.

Bir veri dosyasındaki sık erişilen bloklar bellekte ön belleğe alınır. Ön bellekte yapılan değişikliklerin veri dosyalarına yazılması için yazma proseslerin çalışması beklenir.

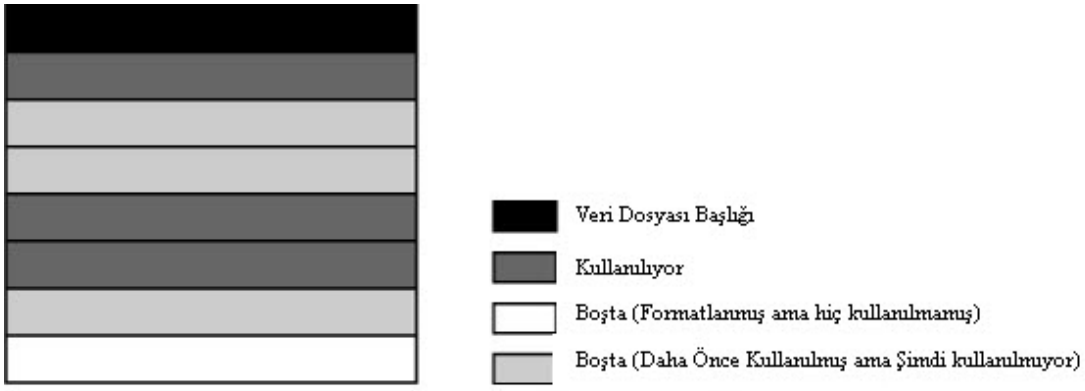
Veri Dosyası Yapısı

Bir veri dosyası başlığı, veri dosyasının büyüklüğü, SCN, mutlak dosya numarası, bağıl dosya numarası gibi veri dosyasıyla ilgili meta verilerini içerir. Mutlak dosya numarası veritabanını içerisinde veri dosyasını benzersiz şekilde tanımlarken bağıl dosya numarası tablo alanı içerisinde benzersiz şekilde tanımlar.*[27]*

Oracle Veritabanı ilk önce bir veri dosyası oluşturduğunda tahsis edilen disk alanı biçimlendirilir, ancak kullanıcı verileri içermez. Bununla birlikte, veritabanı, ilişkili tablo alanın gelecekteki bölümleri için verileri tutacak alanı ayırır.

Şekil 1.3 bir veri dosyasındaki farklı alan türlerini gösterir. Zamanla, bir tablolama alanındaki nesnelerin güncellenmesi ve silinmesi, tek tek, yeni veriler için tekrar kullanılacak kadar büyük olmayan boş alanlar oluşturabilir. Bu boş alana parçalanmış boş alan adı verilir. *[27]*

Şekil 1.3:Veri Dosyasında Alan Kullanımı



https://docs.oracle.com/cd/E11882_01/server.112/e40540/physical.htm#CNCPT404

1.2.1.2 Kontrol Dosyaları (Control Files)

Her Oracle veritabanında, veritabanının meta verisini (diğer bir deyişle veritabanının fiziksel yapısı ile ilgili verileri) muhafaza eden en az bir kontrol dosyası bulunur. Diğer şeylerin yanı sıra, veritabanının adı, veritabanının oluşturulduğu zamanı ve tüm veri dosyalarının adlarını ve yerlerini ve günlük dosyalarını içerir. Buna ek olarak, denetim dosyası, Recovery Manager (RMAN) tarafından kullanılan, kalıcı RMAN ayarları ve veritabanında gerçekleştirilen yedeklemeler gibi bilgileri içerir. *[2]*

Kontrol dosyası, veritabanındaki yapısal değişiklikleri izler. Örneğin, bir yönetici bir veri dosyası veya çevrimiçi redo günlük dosyası eklediğinde, yeniden adlandırdığında veya düşürdüğünde, veritabanı bu değişikliği yansıtacak şekilde kontrol dosyasını güncelleştirir. *[27]*

Oracle Veritabanı, veritabanı kullanımı sırasında kontrol dosyasını sürekli okur ve yazar. Veritabanı açık olduğunda kontrol dosyası her zaman yazılabilir olmalıdır. Kontrol dosyası olmadan veritabanı açılamaz. Kontrol dosyası

veritabanının çalışmasında kritik olduğu için çoğaltılmalıdır. Çoğaltılan kontrol dosyaları farklı disklerde tutulmalıdır. Ancak, kontrol dosyasının kaç kopyası bir veritabanı olgusuna ait olursa olsun, kontrol dosyalarının yalnızca bir tanesi veritabanı meta verilerini alma amacıyla birincil olarak atanır. *[2]*

Bir denetim dosyası kullanılamaz hale gelirse, veritabanı olgusu, bozuk kontrol dosyasına erişmeye çalıştığında başarısız olur. Geçerli kontrol dosyası kopyaları mevcut olduğunda, veritabanı medya kurtarma olmadan açılabilir. Bununla birlikte, bir veritabanının tüm kontrol dosyaları kaybolursa, veritabanı olgusu çökecektir ve medya kurtarma gerekecektir. Geçerli bir kopya mevcut olmadığından, kontrol dosyasının daha eski bir yedek kopyasının kullanılması gerektiği durumlarda medya kurtarması yapmak kolay değildir. *[27]*

Kontrol Dosyası Yapısı

Veritabanı ile ilgili bilgiler kontrol dosyasının farklı bölümlerinde (section) saklanır. Her bölüm, veritabanının bir yönü ile ilgili bir dizi kayıttır. Örneğin, kontrol dosyasındaki bir bölüm veri dosyalarını izler ve her veri dosyası için bir tane olmak üzere bir kayıt kümesi içerir. Her bölüm çoklu mantıksal kontrol dosya bloklarında saklanır. Kayıtlar bir bölüm içindeki blokları aşabilir.

Kontrol dosyası iki çeşit kayıt tip içerir.

- **Dairesel yeniden kullanım kayıtları**

Bu kayıtlar, gerekirse üzerine yazılmaya uygun olan kritik olmayan bilgileri içerir. Kullanılabilir tüm kayıt yuvaları dolduğunda, veritabanı yeni bir dosyaya yer açmak için denetim dosyasını genişletir veya en eski kaydın üzerine yazar. Arşivlenmiş redo günlüğü dosyaları ve RMAN yedeklemeleri hakkındaki kayıtları bu tipe örnek olarak verilebilir. *[27]*

Kontrol dosyasındaki yeniden kullanılabilen bir kayıt tekrar kullanılmadan önce CONTROL_FILE_RECORD_KEEP_TIME parametresindeki süre kadar minimum tutulur. Yeniden kullanılabilir bir bölüme yeni bir kayıt eklenmesi gerektiği ve en eski kaydın yeterince eskimediği durumda, kayıt bölümü genişler. Bu parametre 0 olarak ayarlanırsa, tekrar kullanılabilir bölümler hiçbir zaman genişlemez ve kayıtlar gerektiğinde tekrar kullanılır. *[25]*

- **Dairesel olmayan yeniden kullanım**

Bu kayıtlar, sıklıkla değişmeyen ve üzerine yazılamayan kritik bilgileri içermektedir. Böyle kritik bilgilere tablolar, veri dosyaları, çevrimiçi redo günlük dosyaları ve redo iş parçacıkları (threads) örnek olarak verilebilir. Veritabanı, karşılık gelen nesne tablo alanından atılmadığı sürece bu kayıtların asla üzerine yazmaz. *[27]*

Kontrol dosyası bloklarını okuma ve yazma, veri bloklarını okuma ve yazma işleminden farklıdır. Kontrol dosyası için, veritabanı diskten okur ve doğrudan program genel alanına (PGA) yazar. Her işlem (process) kontrol dosyası blokları için belli miktarda PGA belleğini tahsis eder.

Bir kontrol dosyasının boyutunun temel belirleyicileri, ilişkili veritabanını oluşturan CREATE DATABASE deyimindeki MAXDATAFILES, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY ve MAXINSTANCES parametreleri için ayarlanan değerlerdir. Bu parametrelerin değerlerini artırmak, ilişkili veritabanının kontrol dosyasının boyutunu artırır. *[4]*

1.2.1.3 Çevrimiçi Redo Günlüğü Dosyaları (Online Redo Log Files)

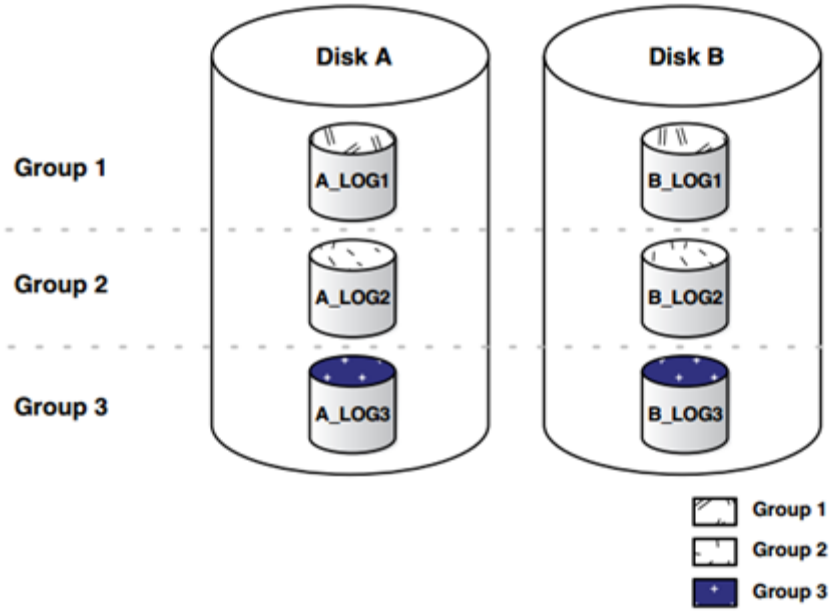
Her Oracle veritabanında iki veya daha fazla çevrimiçi redo günlüğü dosyası kümesi vardır. Redo günlükleri redo kayıtları da olarak adlandırılan redo girişlerinden oluşur.

Çevrimiçi redo günlüğü, verilere yapılan değişikliklerin bir kopyasını depolar. Bir arıza, yedekten bir veri dosyasının geri yüklenmesini gerektiriyorsa, geri yüklenen veri dosyasından eksik olan son veri değişiklikleri, çevrimiçi redo günlük dosyalarından edinilebilir, bu nedenle iş hiçbir zaman kaybolmaz. Çevrimiçi redo günlük dosyaları, donanım, yazılım veya ortam hatasından sonra bir veritabanını kurtarmak için kullanılır. Oracle Database, çevrimiçi redo günlük dosyasının kendisiyle ilgili bir başarısızlığa karşı korumak için çevrimiçi yeniden redo günlük dosyasını çoğaltabilir; böylece, çevrimiçi redo günlük dosyasının iki veya daha fazla özdeş kopyası farklı bir diskte tutulabilir.

Bir veritabanı için çevrimiçi redo günlüğü, çevrimiçi redo günlük dosyaları grubundan oluşur. Grupta redo günlük dosyaları ve çoğullanmış kopyaları bulunur. Her Grup bir sayı ile tanımlanır. *[11]*

Şekil 1.4, her grubun iki üyeden oluştuğu üç çevrimiçi redo günlüğü grubunun veritabanı yapılandırmasını göstermektedir. Her grup için üyeler maksimum kullanılabilirlik için ayrı disklerde saklanması gerekir.

Şekil 1.4: Çevrimiçi Redo Grupları ve üyeleri



[11]

Çok veritabanı olgulu sistemlerde (RAC) her veritabanı olgusu ayrı redo grupları vardır. RAC ortamında iki veya daha fazla veritabanı olgusu eşzamanlı olarak tek bir veritabanına erişir bu yüzden her olgunun kendi redo iş parçacığı bulunur. Her veritabanı olgusu için ayrı iş parçacığı tek bir redo günlüğü dosyası kaynağına erişmekten kaçınarak olası performans darboğazının önüne geçilmiş olur.*[4]*

Veritabanı günlük yazıcısı işlemi (LGWR), o gruptaki günlük dosyaları depolama boyutu sınırına ulaşıncaya veya veritabanı yöneticisinin günlük değiştirme işlemi isteyinceye kadar arabellekteki redo kayıtlarını bir redo günlüğü grubuna yazar. LGWR işlemi daha sonra diğer redo grubuna yazar. LGWR işlemi, en eski grubun en son redo kayıtlarıyla üzerine yazılmasını sağlamak için bu eylemi dairesel bir şekilde gerçekleştirir.

Redo kayıtları, veritabanındaki tek bir bloktaki değişim anlamına gelen değişim vektör gruplarından oluşur. *[4]*

1.2.1.4 Arşivlenmiş Redo Günlüğü Dosyaları (Archived Redo Log Files)

Oracle veritabanı dolmuş redo günlük dosyalarını bir veya daha fazla çevrimdışı hedeflerine yazması arşivlenmiş redo günlüğü olarak bilinir. Redo log dosyalarını arşivlenmiş redo dosyalarına dönüştürme işlemine arşivleme denir.

Arşivlenmiş redo günlüğü dosyaları veritabanını kurtarmak, ikincil veritabanını (standby) güncellemek veya veritabanı geçmişini öğrenmek için kullanılan LogMiner aracını kullanabilmek gibi durumlarda kullanılır.

Arşivleme işlemi, yalnızca veritabanı ARCHIVELOG modunda çalışıyorsa mümkündür. Otomatik veya manuel arşivleme yapılabilir.

Arşivlenmiş redo günlüğü dosyaları redo kayıtlarını ve redo günlük gruplarının özdeş üyelerinin benzersiz günlük sıra numarasını tutar. Bu kayıtları oluşturmaktan arşivci (ARCn) sorumludur. Veritabanı ARCHIVELOG modunda çalışırken, günlük yazma işlemi (LGWR) yeniden kullanılamaz ve bu nedenle arşivleninceye kadar bir redo günlüğü grubunun üzerine yazılır.

Arka plan işlemi ARCn otomatik arşivleme etkinleştirildiğinde arşivleme işlemlerini otomatikleştirir. Veritabanı dolmuş redo günlüklerinin arkada kalmamasını garantilemek için çoklu arşivcileri başlatır. Herbir veritabanı olgusunda kaç tane arşivci çalışacağı LOG_ARCHIVE_MAX_PROCESSES parametresi tarafından belirlenir. *[4]*

Arşivlenmiş redo günlük dosyalarını kullanmak, veritabanında çalışan uygulamanın kullanılabilirlik ve güvenirlik gereksinimlerine bağlıdır. Veri kaybına tahammülün olmadığı uygulamalarda disk arızası gibi öngörülemeyen durumlardan etkilenmemek için ARCHIVELOG modu kullanılmalıdır. Fakat ARCHIVELOG modunda çalışmak sisteme fazladan yük getirecektir.

NOARCHIVELOG Modu

Veritabanını NOARCHIVELOG modunda çalıştırmak arşivciyi devre dışı bırakacaktır. Bu çevrimiçi redo günlüğü dosyalarının arşivlenmeyeceği anlamına gelir. Dolayısıyla redo günlükleri dolduğu zaman arşivciyi beklemeden LGWR tarafından tekrar kullanılabilecek hale gelecektir.

NOARCHIVELOG modu veritabanını veritabanı olgusu hatasından korurken medya hatalarından koruyamayacaktır. Yalnızca son yapılan değişiklikler, tekrar kullanılmamış çevrimiçi redo dosyalarındaki değişiklikler, veritabanı olgusu hatasını kurtarmak için kullanılabilir. Eğer bu modda medya hatası olursa sadece en son tam

sağlıklı veritabanı yedeğinden kurtarma yapılabilir. Bu yedekten sonraki işlemler kurtarılamayacaktır.

NOARCHIVELOG modunda çalışan bir veritabanını geri yüklemek için, yalnızca veritabanı kapalıyken alınan bütün veritabanı yedeklerini kullanabilir. Bu nedenle NOARCHIVELOG modunda çalışan veritabanlarının tam yedeklerini (FULL BACKUP) düzenli, sık aralıklarla alınmalıdır.

ARCHIVELOG Modu

ARCHIVELOG modunda bir veritabanı çalıştığında, redo günlüklerinin arşivlenmesi etkinleştirilmiş olur. Veritabanı kontrol dosyası, redo günlük dosyaları arşivci tarafından arşivlenene kadar redo günlük dosyalarının bir grubunun LGWR tarafından tekrar kullanılamayacağını gösterir. Aktif redo grubu değiştiğinde (switch) arşivleme başlayacaktır.

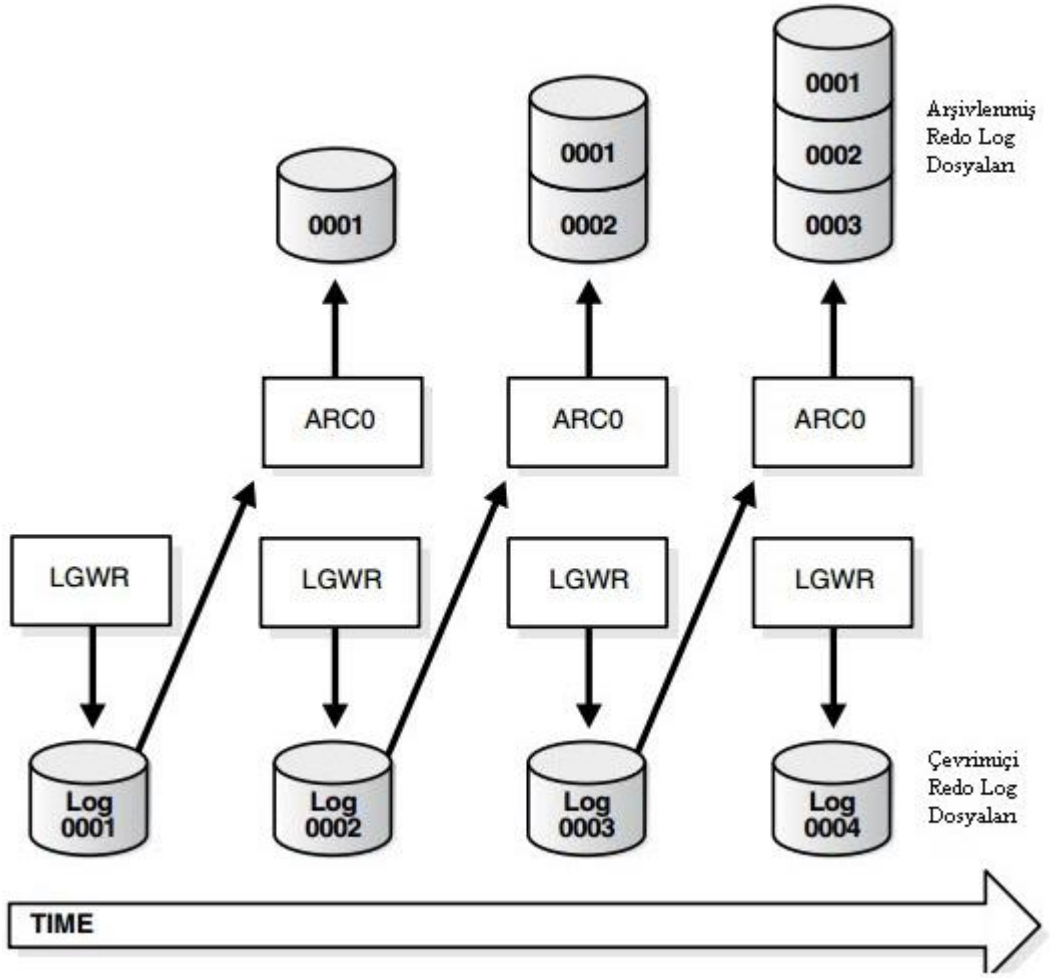
Arşivlemenin avantajları :

- ✓ Bir veritabanı yedeklemesi, çevrimiçi ve arşivlenmiş redo günlüğü dosyalarıyla birlikte, bir işletim sistemi veya disk arızası durumunda tüm işlenmiş (committed) verileri kurtarabileceğini garanti eder.
- ✓ Archivelog modunda yedekleme işlemleri veritabanı açık ve kullanımdayken yapılabilir.
- ✓ Archivelog modunda ikincil veritabanın güncel tutulması sağlanır.

Arşivlenmiş redo günlüğü dosyalarının ve karşılık gelen veritabanı yedeklerinin yerel diskten kaset gibi kalıcı çevrimdışı depolama ortamına taşınmalıdır. Bu dosyaların birincil görevi felaket durumlarında veritabanını kurtarmak olduğu için, korunması garanti edilmelidir.

Şekil 1.5, arşivleme işleminin (bu resimdeki ARC0) doldurulmuş yeniden alma log dosyalarını arşivlenmiş veritabanı redo arşivleme günlüğüne nasıl yazdığını göstermektedir.

Şekil 1.5:Redo günlüğü dosyalarının ARCHIVELOG modda kullanılması



[4]

1.2.1.5 Diğer Veritabanı Dosyaları

Bu dosyalar, veritabanında harici olarak bulunmaktadır. Bunlar, pratik amaçlar için gereklidir ancak kesinlikle veritabanının bir parçası değildir.

1.2.1.5.1. Veritabanı Olgusu Parametre Dosyası

Bir veritabanı olgusu başlatıldığında, SGA yapıları bellekte başlatılır ve arka plan işlemleri parametre dosyasındaki ayarlara göre başlar. Bir olguyu başlatmak için mevcut olması gereken tek dosya budur. Yüzlerce parametre vardır, ancak yalnızca bir tane gereklidir: DB_NAME parametresi. Diğer parametrelerin hepsi varsayılan değerlere ayarlanmıştır. Bu yüzden parametre dosyası bazen çok küçük olsa bile

mutlaka var olması gerekir. Parametre dosyası bazı yerlerde spfile veya pfile olarakta geçer.

1.2.1.5.2 Şifre dosyası

Kullanıcılar bir kullanıcı adı ve bir şifre sunarak oturum başlatırlar. Veritabanı sunucusu, bunları veri sözlüğünde saklanan kullanıcı tanımlarına karşı kimlik doğrulamasını yapar. Veri sözlüğü, veritabanındaki tabloların bir kümesidir. Bu nedenle veritabanı açık değilse erişilemez. Veritabanı oluşturulurken veya başlatılırken gibi veri sözlüğü hazır olmadan önce kimlik doğrulaması yapılması gereken zamanlar olabilir. Harici bir şifre dosyası bunu yapmanın bir yoludur. Şifre dosyası veri sözlüğünün dışında bulunan ve dolayısıyla veri sözlüğü kullanılmadan önce bir veritabanı olgusuna bağlanmak için kullanılabilen az sayıda kullanıcı adı ve parola içerir.

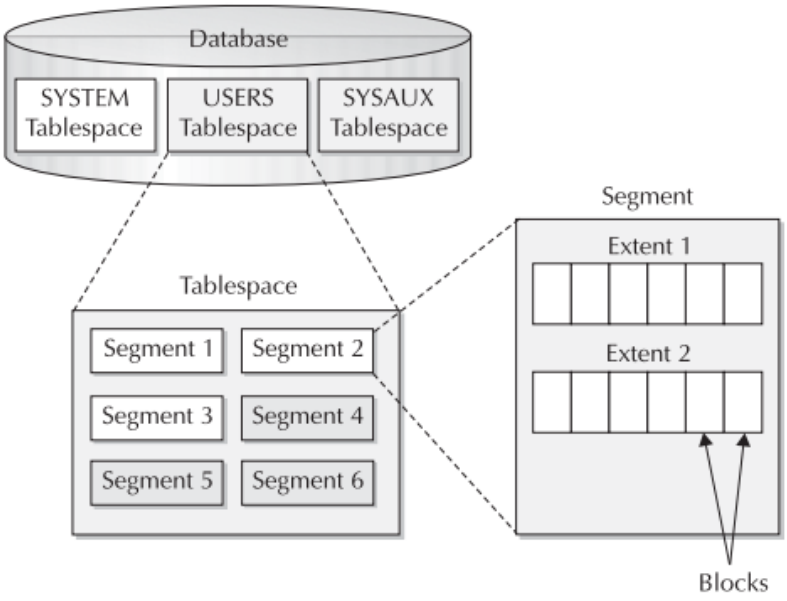
1.2.1.5.3 Uyarı günlüğü ve İzleme Dosyaları

Uyarı günlüğü, veritabanını etkileyen bazı kritik işlemlerle ilgili sürekli bir mesaj akışıdır. Bu dosyada veritabanının açılıp kapanması, veritabanı yapısındaki fiziksel değişiklikler, parametre dosyasındaki değişiklikler gibi veritabanı açısından çok önemli olaylar tutulur. İz dosyaları hata koşulları olduğunda veya bazen belirli olayları rapor etmek için arka plan işlemleri tarafından oluşturulur.

1.2.2 Mantıksal Depolama Yapıları

Bir veritabanındaki veri dosyaları, bir veya daha fazla tablo uzayına gruplanır. Her tablo alanında, tablolar ve indeksler gibi mantıksal veritabanı yapıları, kapsam ve bloklara bölünen bölümlerdir. Depolamanın bu mantıksal alt bölümü, Oracle'ın disk alanı kullanımı üzerinde daha etkin bir denetime sahip olmasını sağlar.

Şekil 1.6: Mantıksal Depolama Yapıları



[2]

1.2.2.1 Tabloalanları (Tablespaces)

Bir tabloalanı, ilgili mantıksal yapıları birlikte gruplayan bir veritabanı depolama birimidir. Bir tablo alanı bir veya daha fazla fiziksel veri dosyasından oluşur. Bir tablo alanına atanan veritabanı nesneleri, bu tablo alanının fiziksel veri dosyalarında saklanır. Tabloalanları, verileri depolamada fiziksel olarak bulmak için bir araç sağlar. Bir tabloalanına veri dosyası tanımlandığında, bu dosyalar için saklama yerleri belirtilmiş olur. Bu tabloalanına atanan şema nesneleri belirtilen depolama konumunda bulunur. Tabloalanları ayrıca yedekleme ve kurtarma birimi sağlar. *[11]*

Herhangi bir veritabanındaki birincil tablo alanı, veri sözlüğü ve sistem geri alma segmenti gibi veritabanı sunucusunun işleyişi için temel bilgileri içeren SİSTEM tabloalanıdır. SİSTEM tabloalanı, veritabanı oluşturulduğunda yaratılan ilk tabloalandır. Diğer herhangi bir tabloalanı gibi yönetilir, ancak daha yüksek bir ayrıcalık seviyesi gerektirir ve bazı açılardan kısıtlanmıştır. Örneğin, SİSTEM tabloalanı yeniden adlandırılamaz, atanamaz veya çevrimdışı alınamaz. *[4]*

Çoklu Tabloalanı Kullanmak

Çoklu tabloalanı kullanmak, veritabanı işlemlerini gerçekleştirirken daha fazla esneklik sağlar. Aşağıdaki faydalarından dolayı uygulama bazında ayrılmış tabloalanı kullanılmalıdır.

- ✓ Kullanıcı verileri ile veri sözlüğü ayrı tabloalanlarında tutularak G/Ç çekişmesini(contention) önler.
- ✓ Uygulama nesnelerinin SİSTEM tabloalanında kalması durumunda uygulamadan kaynaklanan sıkıntıların tüm veritabanını etkilemesinin önüne geçilmiş olur.
- ✓ Tabloalanlarından bir tanesi çevrim dışı olması gerekiyorsa (bakım veya yedek için olabilir) diğer uygulamaların bu durumdan etkilenmemesi sağlanmış olur.
- ✓ Tabloalanlarının veri dosyaları ayrı disk sürücülerinde depolanırsa bu durumda yoğun disk kullanan uygulamalar, diğer uygulamaların performansını düşürmeyecektir.
- ✓ Yüksek güncelleme etkinliği, salt okunur etkinlik veya geçici bölüm depolama alanı gibi belirli bir veritabanı kullanımı türü için bir tabloalanı ayırarak tablo alanının kullanımını optimize edilmiş olur.
- ✓ Uygulamalara birbirinden ayrı yedekleme yeteneği verilmiş olur.

Çoklu tabloalanları oluştururken veritabanının üzerinde koştuğu işletim sisteminin aynı anda açılabilecek maksimum dosya sayısına takılabilir. Bu yüzden tabloalanları oluşturulurken etkin bir şekilde oluşturulmalıdır. Tabloalanlarının veri dosyaları olabildiğince az ve büyük olmalıdır. *[4]*

1.2.2.2 Segmentler

Bir segment, bir tabloalanı içindeki mantıksal bir depolama yapısı için tüm verileri içeren bir kapsamlar (extent) kümesidir. Örneğin, Oracle veritabanı, bir tablonun veri segmentini oluşturmak için bir veya daha fazla kapsam tahsis eder. Veritabanı, aynı zamanda bir tablonun indeks bölümünü oluşturmak için bir veya daha fazla kapsam tahsisi eder. Bu işlem otomatik veya manuel yapılabilir. *[1]*

Kullanıcı Segmenti

Çeşitli tipte kullanıcı segmentleri vardır.

- ✓ Tablo, Bölümlenmiş Tablo, Kümelenme Tablosu

- ✓ Büyük Objeler (LOB), Bölümlenmiş Büyük Objeler
- ✓ İndeks, Bölümlenmiş İndeksler

Bölümlenmiş veya bölümlenmemiş olsun bütün nesneler kendi segmentinde saklanır. Varsayılan olarak, veritabanı, tablo ve indeks oluştururken yalnızca veritabanı meta verileri güncelleştirmek için ertelenmiş segment oluşturma kullanır. Oracle veritabanı, bölümlemiş alan oluştururken bir kullanıcı tabloya ilk satır ekleyene kadar segment yaratmayı da erteler. Ancak bundan sonra veritabanı tablo veya bölümlenmiş alan için LOB sütunları ve indeksler için segmentler oluşturur.

Bölümlenmiş segment oluşturabilme özelliği veritabanı kaynaklarının gereksiz yere kullanılmasını önler. Örneğin bazı uygulamalar ilk kurulurken çoğu kullanılmayacak olan binlerce nesne oluşturabilir. Bu da önemli miktarda disk alanı tüketilmesine yol açar. Bu özellik DEFERRED_SEGMENT_CREATE parametresi ile kontrol edilebilir. DEFERRED_SEGMENT_CREATE parametresinin disk kaynaklarının verimli kullanılabilmesi için TRUE değerine ayarlanmış olması gerekir. *[26]*

Geçici Segment

Bir sorgu çalıştırılırken, veritabanı genellikle SQL deyimin yürütülmesinin ara aşamaları için geçici çalışma alanına ihtiyaç duyar. Geçici segment gerektiren tipik işlemler sıralama, karıştırma ve birleştirme içerir. İndeks oluşturulurken, veritabanı indeks segmentlerini geçici segmentlere yerleştirir ve indeks tamamlandığında bunları kalıcı segmentlere dönüştürür.

Eğer operasyon bellekte gerçekleştirilebilecek büyüklükteyse, veritabanı geçici segment oluşturmaz. Bununla birlikte bellek uygun durumda değilse, veritabanı otomatik olarak diskte geçici segment oluşturur. Geçici segment üzerindeki alan yönetimi operasyonları haricinde, geçici segmentlerdeki değişiklikler çevrimiçi redo günlüklerine yazılmazlar. *[1]*

Geçici segmentler için alan tahsisi ve bu alanın geri alınması işlemleri çok sık olduğu için, en iyisi sadece geçici segmentlere ait tablolama alanı oluşturulmalıdır. Böylelikle sistem tabloalanı ve diğer tablo alanlarının fragmente olması önlenmiş olur.

Geri Alma Segmenti

Veritabanı, toplu olarak veri geri alma olarak bilinen işlemlerin hareketlerinin kayıtlarını tutar. Geri alma segmenti aşağıdaki işlemleri yapabilmek için kullanılır.

- ✓ Aktif olan işlemlerin geri alınabilmesi
- ✓ Sonlanmış bir işlemin kurtarılabilmesi
- ✓ Veri okunmasında tutarlılığın sağlanması
- ✓ Bazı mantıksal işlemleri için flashback yapılabilmesi

Veritabanı geri alma verilerini harici günlüklerde tutmak yerine veritabanının içerisinde tutar. Geri alma verisi güncellenebilen bloklarda saklanır ve bloklardaki değişiklikler redo üretir. Veritabanı harici dosya okumadan efektif bir şekilde geri alma verisine erişebilir. *[1]*

1.2.2.3 Kaplamlar (Extents)

Bir kaplam bitişik veri bloklarından oluşan veritabanı depoloma alan tahsisinin mantıksal birimidir. Varsayılan olarak bir segment oluşturulduğunda veritabanı başlangıç kaplamı tahsis eder. Bir kaplam her zaman bir veri dosyasında bulunur. Başlangıç kaplamı dolarsa ve daha fazla alan gerekiyorsa veritabanı bu segment için otomatik olarak artımlı bir kaplam tahsis eder. Artımlı kaplam segment için oluşturulan bir sonraki kaplamdır.

Alan dağıtma algoritması tabloalanının yerel olarak mı sözlük tabanlı olarak mı yönetilmesine bağlıdır. Yerel olarak yönetilen durumlarda veri dosyasının bit eşlemini arar. Veri dosyası yetersiz bir alana sahipse veritabanı başka bir veri dosyasına bakar. Bir segment için kaplamlar aynı tabloalanında tutulurken farklı veri dosyalarında olabilir.

Genel olarak kullanıcı segmentinin kaplamları bir DROP komutu kullanılarak nesne düşürülmedikçe tabloalanına dönmez. Örneğin bir tablodaki tüm satırlar silinse bile, bu tablonun blokları veritabanı tarafından diğer nesnelerin kullanabilmesi için kazanılamaz.

Bir segmentteki fragmente olmuş alanı geri kazanabilmek için:

- ✓ Çevrimiçi segment daraltma işlemi uygulanır. Genellikle veri sıkıştırma işlemi daha iyi önbellek kullanımı sağlar tam tablo taramalarında (full table scan) daha fazla blok üzerinde işlem yaparak performansı artırır.

- ✓ Tablo isteğe bağlı olarak yeni bir tablo alanının yeni bir segmentine taşınabilir (move table).
- ✓ İndeksler yeniden oluşturulabilir.

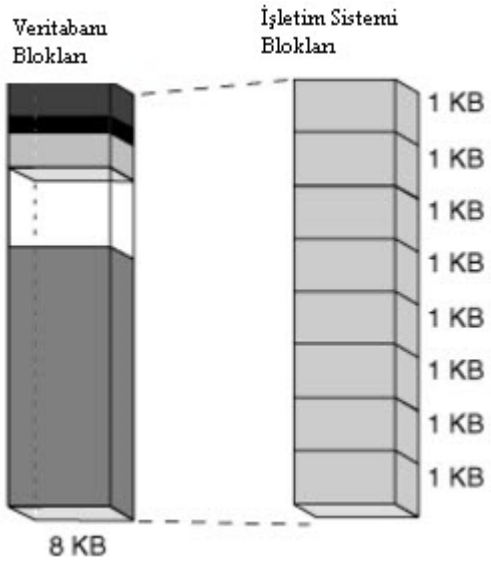
Kaplamlar serbest bırakıldığında, veritabanı yerel olarak yönetilen tablo alanlarındaki veri dosyasındaki bitmapi, geri alınan kaplamları kullanılabilir alan olarak yansıtacak şekilde değiştirir. Serbest kalan blokların içindeki tüm veriler erişilemez hale gelir.

1.2.2.4 Bloklar

Veritabanı, veri dosyalarındaki mantıksal depolama alanını, veritabanı blokları veya sayfaları olarak da adlandırılan veri blokları biriminde yönetir. Bir veri bloğu, veritabanı G/Ç'nin minimum birimidir.

Fiziksel düzeyde, veritabanı verileri, işletim sistemi bloklarından oluşan disk dosyalarında saklanır. Bir işletim sistemi bloğu, işletim sisteminin okuyabileceği veya yazabileceği minimum veri birimidir. Buna karşılık, bir veritabanı bloğu, boyutu ve yapısı işletim sistemi tarafından bilinmeyen mantıksal bir depolama yapısıdır.

Şekil 1.7 : Veritabanı blokları ve İşletim sistemi Blokları



[1]

Veritabanı bir veri bloğu istediğinde, işletim sistemi bu işlemi kalıcı depolama alanındaki bir veri isteklerine çevirir. Veritabanı bloklarının işletim sistemi bloklarından mantıksal olarak ayrılması aşağıdaki etkileri içerir:

- ✓ Uygulamaların disk üzerindeki verilerin fiziksel adreslerini belirlemelerine gerek yoktur.
- ✓ Veritabanı verileri birden fazla fiziksel diskte çoklanabilir.

Veritabanı Blok Boyutu

Her veritabanı bir veritabanı bloğu boyutuna sahiptir. DB_BLOCK_SIZE başlatma parametresi, bir veritabanı oluşturulduğunda veri bloğu boyutunu ayarlar. Boyut SYSTEM ve SYSAUX tabloalanları için ayarlanır ve diğer tüm tablolar için varsayılan değerdir. Veritabanı bloğu boyutu, veritabanının yeniden oluşturulması dışında değiştirilemez.

DB_BLOCK_SIZE ayarlanmazsa, varsayılan veri bloğu boyutu işletim sistemine özgüdür. Bir veritabanı için standart veri bloğu boyutu 4 KB veya 8 KB'dir. Veri bloğu ve işletim sistemi blokları için boyut farklıysa, veri bloğu boyutu işletim sistemi blok boyutunun katı olmalıdır.

Blok Sıkıştırıma

Veritabanı, bir veri bloğundaki yinelenen değerleri ortadan kaldırmak için blok sıkıştırmasını kullanabilir. Temel ve ileri satır sıkıştırmayı kullanan bir veri bloğunun biçimi aslında sıkıştırılmamış bir bloğun biçimiyle aynıdır. Aradaki fark, bloğun başındaki bir sembol tablosunun satırlar ve sütunlar için yinelenen değerleri saklamasıdır. Veritabanı, bu değerlerin ortaya çıktığı değerleri sembol tablosuna kısa bir referansla değiştirir.

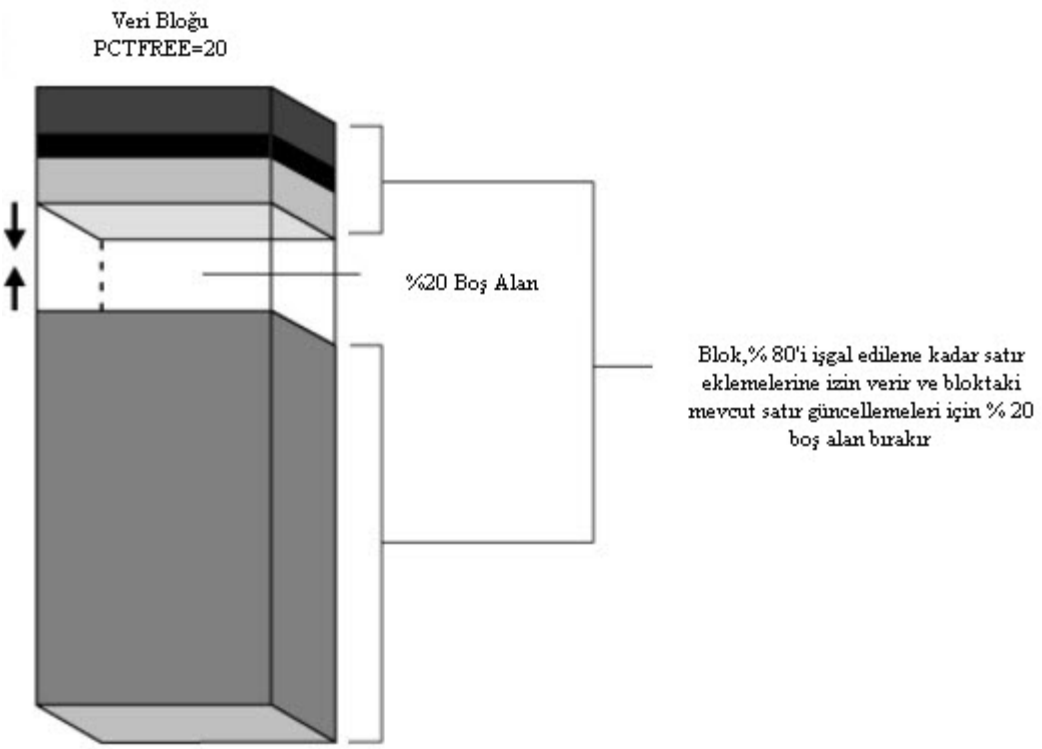
Veritabanı veri bloğunu aşağıdan yukarıya doğru doldurduğu için, satır verileri ile blok başlığı arasındaki boş alan miktarı azalır. Boşluk (null) değerini boşluk olmayan değere değiştirirken olduğu gibi, aradaki bu boş alan güncellemelerle de daralabilir. Performansı optimize etmek için bu alanın yönetilmesi veritabanı tarafından yapılır.

PCTFREE Parametresi

PCTFREE depolama parametresi veritabanının boş alanı nasıl yönettiği konusunda çok önemlidir. Bu SQL parametresi mevcut satır güncellemeleri için boş

alan olarak rezerve edilmiş veri bloğunun minimum yüzdesini ayarlar. Böylelikle satırın başka bir bloğa taşması önlenerek alan israfından sakınılmış olunur. Nadiren güncelleme alacak tabloların PCTFREE değeri düşük tutulmalıdır.

Şekil 1.8:PCTFREE Kullanımı

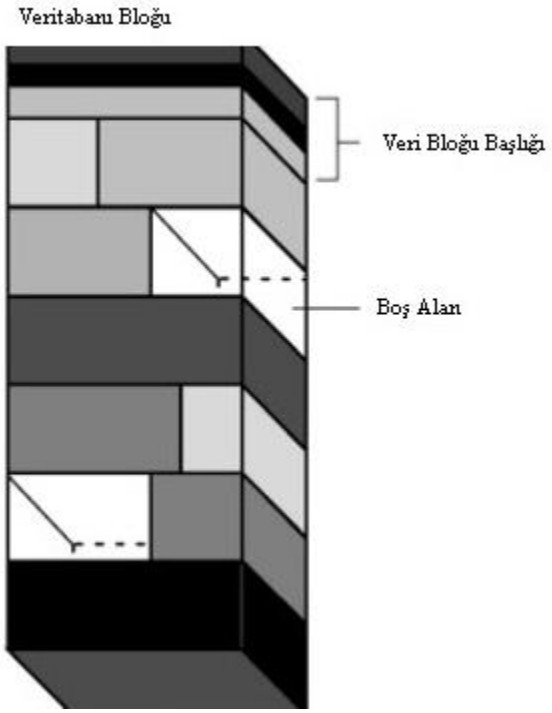


[1]

<u>Bloklardaki Parçalanmış (Fragmantasyon) Alanlar</u>

Veri bloğu içerisindeki boş alanlar ardışık olabilir veya olmayabilir. Ardışık olmayan boş alanlara parçalanmış (fragmente) alan denir.

Şekil 1.9: Fragmente Olmuş Veri Bloğu



[1]

Çok fazla kayıt ekleme, güncelleme, silme işlemleri, indekslerin yapısının bozulması gibi durumlar fragmentasyona neden olabilir. Yeni bir satır ekleneceği zaman parçalanmış alanlara dağılması gerektiğinden bu durum performans problemlerine neden olur. Fragmente olmuş boş alanların birleştirilmesi gerekir.

1.3 Veritabanı Süreç (Process) Yapıları

Veritabanı olgusu arka plan süreçleri, veritabanı olgusu başlatıldığında başlayan ve olgu sonlanana kadar çalışan işlemlerdir. Bir işlem işletim sistemi üzerinde bir dizi adımı uygulayabilen mekanizmadır. Mekanizma işletim sisteminin türüne bağımlıdır. Linux ve Unix'te tüm Oracle süreçleri benzersiz işlem numarası olan ayrı işletim sistemi süreçleridir. Windows'ta ise bütün bir veritabanı olgusu için ORACLE.EXE olarak adlandırılan tek bir işletim sistemi işlemi vardır. Oracle işlemleri bu tek bir işlem içerisinde ayrı iş parçacıkları (thread) olarak çalışır. *[12]*

SMON Sistem Monitor işlemi

SMON'un başlangıçta veritabanı ön hazırlık işlemlerini yapma ve açma görevi vardır. Kısaca SMON veritabanı kontrol dosyasını bulup geçerliliğini kontrol ettikten sonra veritabanını MOUNT moduna alır. Ardından tüm veri dosyalarını ve

çevrimiçi günlük dosyalarını bulup veritabanını başlatır. Veritabanı bir kere açıldıktan sonra ve kullanımda iken SMON veri dosyaları içerisindeki boş alanları birleştirme gibi görevlerden sorumludur.

PMON (Process Monitor)

Bir kullanıcı oturumu, bir sunucu işlemine bağlı bir kullanıcı işlemidir. Sunucu işlemi, oturum oluşturulduğunda başlatılır ve oturum sona erdiğinde yok edilir. Oturumdan düzgün olarak çıkış yapılabilmesi, kullanıcının oturumu kapatmasını gerektirir. Bu gerçekleştiğinde, yapılan herhangi bir iş düzenli bir şekilde tamamlanacak ve sunucu işlemi sonlandırılacaktır. Oturum düzensiz bir şekilde sona ererse (belki de kullanıcının PC'si yeniden başlatıldıktan sonra), oturum temizlenmesi gereken bir durumda bırakılacaktır. PMON tüm sunucu süreçlerini izler ve oturumlarla ilgili problemleri tespit eder. Bir oturum anormal şekilde sonlandıysa, PMON sunucu işlemini yok eder, PGA belleğini işletim sisteminin boş bellek havuzuna geri döndürür ve devam eden tamamlanmamış işlemi geri alır (roll back)

DBWn (Database Writer)

Her seferinde oturumlar diske genel bir kural olarak yazmaz. Değişen verileri veritabanı önbelleğine yazarlar. Önbelleklerdeki verileri daha sonra diske yazan veritabanı yazıcısıdır. Bir veritabanı olgusunda DBW0, DBW1, DBW2 diye adlandırılan maksimum yirmi taneye kadar çıkabilen çok sayıda veritabanı yazıcısı vardır. Varsayılan olarak her sekiz CPU için bir adet veritabanı yazıcısı vardır.

Fiziksel diskte G/Ç yapma işlemleri performansı önemli ölçüde düşürür. Bu yüzden G/Ç işlemlerinden mümkün olduğunca kaçınılması gerekir. Bir oturum ara bellekteki bir bloğu değiştirmiş olsa bile, bu bloğun tekrar değiştirilebilme olasılığı her zaman vardır. Yakın gelecekte bu blok değişecekse, bu bloğu her seferinde diske yazmak performans açısından oldukça maliyetlidir. DBWn'in diske yazmak için kullandığı algoritma son zamanlarda kullanılmamış kirli tamponlardaki bloklara bakar. Bir arabellek çok meşgulse, defalarca okunup yazılıyorsa DBWn bu değişiklikleri her seferinde diske yazmayacaktır.

DBWn tembel algoritmayı kullanır; mümkün olduğunca az, mümkün olduğunca nadiren. DBWn diske yazma işlemini aşağıdaki durumlarda yapar:

- ✓ Bellekte boş yer kalmadığında
- ✓ Çok fazla kirli bellek olduğunda
- ✓ Üç saniyelik zaman aralıklarında
- ✓ Kontrol noktası (checkpoint) işlemi çalıştığında

LGWR (Log Writer)

LGWR günlük tamponun içeriğini diskteki çevrimiçin günlük dosyalarına yazar. Bu işlem günlük tamponun boşaltılması olarak bilinir. Bir oturum, veritabanı arabellek önbelleğindeki bloklara DML işlemleriyle herhangi bir değişiklik yaptığında, değişikliği bloğun üzerine uygulamadan önce, günlük tamponuna uygulamak üzere olan değişim vektörünü yazar. İş kaybını önlemek için bu değişim vektörleri yalnızca minimum gecikme ile diske yazılmalıdır. Bu amaçla LGWR günlük tamponun içeriğini neredeyse gerçek zamanlı olarak disk üzerindeki çevrimiçi günlük dosyalarına yazar.

Bir oturum COMMIT komutunu çalıştırdığında LGWR tamponu diske yazana kadar oturum kilitlenir. Bu komut başarılı çalıştıktan sonra artık işlem geri döndürülemez. LGWR, veritabanı mimarisindeki en büyük darboğazlardan birisidir. LGWR'ın değişim vektörlerini diske yazabilme hızından daha hızlı DML gerçekleştirmek imkansızdır.

Aslında her COMMIT komutundan sonra LGWR işleminin çalışması engellenerek LGWR'den kaynaklanan darboğazlar aşılabilir. Böylelikle performans artışı sağlanmış olur. Ancak bu veri kaybına neden olabilir. Son yapılan COMMIT komutu ile LGWR değişim vektörleri arasındaki işler veritabanı çökmesi durumunda kurtarılamaz. Kurumsal veritabanlarında veri kaybı olasılığına tahammül olamayacağından bu özellik kesinlikle aktif edilmemelidir.

LGWR tampon günlüğünü aşağıdaki durumlarda boşaltır.

- ✓ Kullanıcı COMMIT komutunu çalıştırırsa
- ✓ Günlük tamponunun üçte biri dolu ise
- ✓ DBWR çalışıyorsa

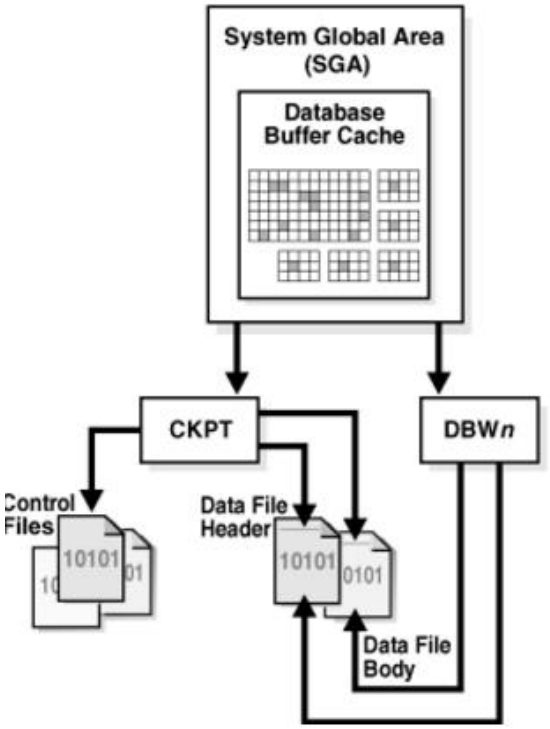
LGWR tampondaki değişim vektörleri diskteki çevrimiçi günlük dosyalarına yazarak veritabanı işlemlerinin kaybolmayacağını garanti eder. Bu dosyalar aynı

zamanda veri dosyası yedeklemelerine de uygulanacaktır. Böylelikle veritabanı herhangi bir hasar görürse veriler kurtarılabilecektir.

CKPT (Checkpoint Process)

CKPT işlemi, kontrol dosyası ve veri dosyası başlıklarını kontrol noktası bilgileriyle günceller ve blokları diske yazdırmak için DBWn işlemine sinyal verir. Kontrol noktası bilgileri CP pozisyonu, SCN ve kurtarma işlemini başlatabilmek için çevrimiçi redo günlükteki konum bilgilerini içerir.

Şekil 1.10::CKPT işlemi



https://docs.oracle.com/cd/E11882_01/server.112/e40540/process.htm#CNCPT9841

CKPT işlemi artımlı (incremental) veya tam (full) olabilir. Artımlı kontrol noktası mekanizması DBWn işlemine, DBWn ve LGWR arasında her zaman öngörülebilir bir boşluk olması için kirli tamponları diske yazması için talimat verir. Artımlı kontrol noktası tam kontrol noktası mekanizmasından daha iyi performans ve makul kurtarma süreleri sağlar. Daha sık diske yazmak sıkıntı anında kurtarma süresine düşürürken daha fazla G/Ç yaptığı için performansı düşürecektir. Bu denge

iyi sağlanmalıdır. Tam kontrol noktası veritabanı yöneticisinin "alter system checkpoint;" komutuyla veya veritabanı düzgün bir biçimde kapanırken yapılır.

MMON (Manageability Monitor)

MMON, bir veritabanının kendi kendini izleme ve optimize edebilme özelliklerini etkinleştirme işlemidir. Veritabanı olgusu etkinlik ve performansla ilgili çok sayıda istatistik toplar. Bu istatistikler SGA'da biriktirilir ve SQL deyimleri ile sorgulanabilir. Performans iyileştirme ve yönelim analizleri için bu istatistikleri uzun vadeli depolama birimlerine kaydetmek gereklidir. MMON işlemi düzenli aralıklarla (genelde saatte bir) bu istatistikleri SGA'dan alır ve bunları süresiz olarak depolanabildiği veri sözlüğüne yazar.

MMON ayrıca ADDM'yi başlatır. ADDM veritabanı aktivitesini analiz eden bir araçtır. ADDM, MMON işleminin toplamış olduğu anlık görüntüleri (snapshot) analiz eder ve performansla ilgili gözlem ve tavsiyeler yapar.

MMAN (Memory Manager)

MMAN, bellek tahsisinin otomatik olarak yönetilmesini sağlar. Oracle veritabanlarının önceki sürümlerinde açılmış oturuma ilişkin sunucu işleminin kullandığı bellek alanı olan PGA devredilemezdi. Sunucu işleminin işletim sistemi belleğinden aldığı alanı geri vermezdi. SGA bellek yapıları statikti. Bir kere veritabanı olgusu başlarken tanımlanırdı ve veritabanı olgusu kapanana veya yeniden başlayana kadar değiştirilemezdi.

Oracle veritabanının yeni sürümlerinde artık PGA bellek alanı ihtiyaca göre belirli sınırlar dahilinde otomatik olarak büyür, küçülür veya devredilebilir. SGA boyutları da artık dinamiktir. MMAN işlemi SGA bellek yapılarına olan talebi izler ve gerektiğinde yeniden boyutlandırabilir.

Veritabanı bellek yönetimi için ihtiyaç duyulan toplam bir hedef belirlenir. MMAN toplam bellek miktarını belirlenen limit dahilinde tutarken tüm oturumlara ihtiyaç duydukları kadar bellek tahsisini yapar. MMAN performans ve kaynak kullanımında büyük fayda sağlar.

ARCn, (Archiver)

ARCn işlemi bir veritabanının çalışabilmesi için zorunlu olmasa da kurumsal veritabanlarında iş bütünlüğü açısından gereklidir. ARCn işlemi çalışmıyorsa arıza

durumlarında veri kaybı kaçınılmazdır. Bu işlemin amacı arşivlenmiş günlük dosyalarını oluşturmaktır.

Veri bloklarına uygulanan tüm değişim vektörleri önce tampon günlüklerine daha sonra da oradan çevrimiçi redo günlük dosyalarına yazılır. Verilere uygulanan tüm değişikliklerin eksiksiz bir geçmişini korumak için, çevrimiçi günlük dosyaları doldurulurken ve yeniden kullanılmadan önce kopyalanmalıdır. Bunu yapmaktan ARCn işlemi sorumludur. Veritabanında olan herhangi bir arıza anında, en son alınmış yedek üzerine elde edilen arşivlenmiş günlük dosyalarından çıkarılan değişiklik vektörlerini uygulayarak herhangi bir hasardan veri kaybetmeden kurtulmak mümkün olacaktır.

ARCn işleminin ilerleyişi ve yazdıkları hedeflerin durumu düzenli olarak uyarı sistemleri üzerinden izlenmelidir. Arşivleme başarısız olursa veritabanı çalışamaz duruma gelir.

RECO (Recoverer Process)

Dağıtılmış bir veritabanında, kurtarma işlemi (RECO) otomatik olarak dağıtılmış işlemlerdeki hataları giderir. Bir düğümün RECO işlemi, şüpheli dağıtılmış işlemde bulunan diğer veritabanlarına otomatik olarak bağlanır. RECO, veritabanları arasında bir bağlantıyı yeniden kurduğunda, tüm şüpheli işlemleri otomatik olarak çözümler; çözülmüş işlemlere karşılık gelen satırları her veritabanının bekleyen işlem tablosundan kaldırır.

Diğer Arka Plan İşlemleri

Diğer belli başlı arka plan işlemleri aşağıdaki gibidir:

- **CJQ0 :** Bu işlemler periyodik şekilde çalıştırılacak şekilde planlanan işleri yönetir. İş kuyruğu koordinatörü CJQn, iş kuyruklarını izler ve yönetir.
- **D000 :** Paylaşımlı sunucu mekanizması etkinleştirildiyse, paylaşımlı sunucu süreçlerine SQL çağrıları gönderecek dağıtıcı işlemdir.
- **DBRM :** Veritabanı kaynak yöneticisi, kaynak planlarını ve diğer kaynak yöneticisi ile ilgili görevleri ayarlamakla sorumludur.
- **DIA :** Veritabanı askıda kaldığında veya kilitlenme (deadlock) durumları oluştuğunda bunları çözümlemekten sorumludur.

- **DIAG :** Veritabanı olgusunda problemler olduğunda bunları araştırmaktan sorumludur.
- **FBDA :** Verilerin geçmişte olduğu gibi her zaman sorgulanmasını sağlamaktan sorumludur.
- **PSP0 :** Diğer oracle işlemlerini oluşturma ve yönetmekten sorumludur.
- **SHAD :** Linux sistemlerinde TNS V1-V3 işlemleri olarak görülür. Kullanıcı oturumunu destekleyen sunucu işlemleridir.
- **SMCO,W000 :** Alan yönetimi koordinatörü prosesi, proaktif alan tahsisi ve alan yenilemesi gibi çeşitli alan yönetimi ile ilgili görevlerin yürütülmesini koordine eder. Görevi uygulamak için dinamik olarak bağımlı süreçleri (Wnnn) üretir.
- **VKTM :** Zamanın izlenmesinden sorumludur. Kümelenmiş ortamda özel önem taşır.

1.4. Veritabanı Bellek Yapıları

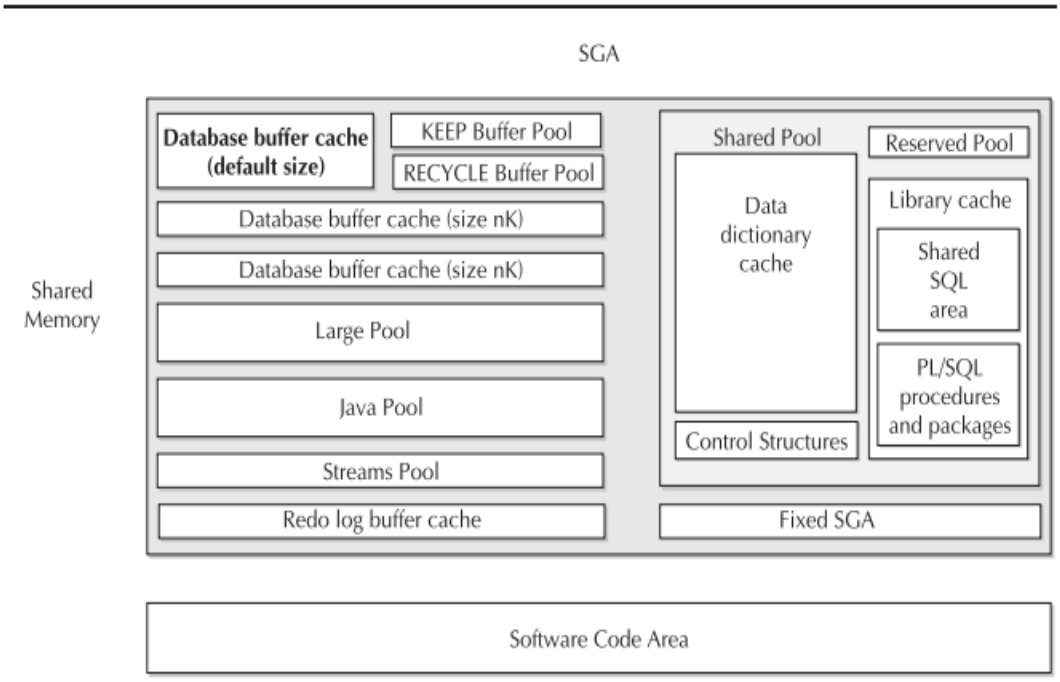
Veritabanı çalıştırılabilir kodu, oturum bilgisi, veritabanı ile ilişkili bağımsız işlemlerin ve işlemler arasında paylaşılan bilgiler (örneğin, veritabanı nesneleri üzerindeki kilitler gibi) gibi birçok bilgiyi tutmak için sunucunun fiziksel belleğini kullanır. Buna ek olarak, bellek yapıları kullanıcı ve veri sözlüğü SQL deyimlerini yanı sıra veritabanı bölümlerinden gelen veri blokları ve veritabanındaki tamamlanmış işlemler hakkında bilgi gibi sonunda diskte kalıcı olarak saklanan önbellek bilgileri içerir. Bir veritabanı olgusu için ayrılan veri alanına Sistem Genel Alanı (SGA) adı verilir. Oracle çalıştırılabilir yazılım kod alanında bulunur. Buna ek olarak, Program Genel Alanı (PGA) adı verilen bir alan, her bir sunucuya ve arka plan işlemine özeldir; Her işlem için bir PGA tahsis edilmiştir. Şekil 1-11, bu Oracle bellek yapıları arasındaki ilişkileri göstermektedir. *[2]*

1.4.1 Sistem Genel Alanı (SGA)

SGA veritabanı olgusu için paylaşılan bellek yapılarının bir grubudur ve veritabanı olgusu kullanıcıları arasında paylaşılır. Veritabanı olgusu başlatıldığında başlatma parametresi dosyasında belirtilen değere göre SGA olgusu ayrılır. SGA'nın çeşitli bölümlerinin boyutlarını kontrol eden parametrelerin çoğu dinamiktir. Ancak SGA_MAX_SIZE parametresi belirtilirse, tüm SGA alanlarının toplam boyutu SGA_MAX_SIZE değerini aşamaz. MEMORY_TARGET parametresi veritabanı olgusunun kullanacağı toplam bellek alanını gösterir. Bu parametre dinamiktir.

Veritabanı olgusu çalışırken değiştirilebilir. MEMORY_TARGET parametresi performansı en iyi duruma getirmek için SGA ve PGA arasındaki dağıtımı yapar. *[2]* SGA'yı oluşturan bileşenler aşağıda anlatılmıştır. SGA'yı oluşturan bileşenlerle ilgili gerçek zamanlı anlık bilgi almak için V$SGA_DYNAMIC_COMPONENTS görünümünden faydalanılabilir.

Şekil 1.11 : SGA Bileşenleri



[2]

Tampon Ön belleği

Veritabanı tampon önbelleği, bir select ifadesini karşılamak için kısa süre önce okunan veya bir DML bildiriminde değiştirilen veya eklenen değiştirilmiş blokları içeren disklerden gelen veri bloklarını tutar. Tampon ön belleğinin okunması ve yazılması işlemine mantıksal G/Ç denir. Tampon ön belleğinde bulunamayan veriler için veritabanı fiziksel G/Ç yapılır. Bu veritabanı performansını olumsuz etkiler.

Tampon önbelleğindeki bloklar üç durumda bulunabilir.

- **Kulanılmamış (Unused) :** Tampon kullanılabilir halde beklemektedir.
- **Temiz (Clean) :** Bu tampon yakın zamanda kullanılmış ve kontrol noktası işlemi çalışarak tutarlı okuma durumuna geçmiştir. Tampon kullanılabilir halde beklemektedir.

- **Kirli (Dirty) :** Bu tampon diske yazılmamış değiştirilmiş verileri içerir. Kontrol noktası işlemi çalışmadan kullanılamaz.

Veritabanı, tampon önbelleğine erişimini verimli hale getirebilmek için gelişmiş bir algoritma kullanır. Sıcak veya soğuk tampon bölgesi olarak adlandırılan son zamanlarda en az kullanılmış (LRU) algoritmasını dikkate alır. Soğuk tampon belleği yakın zamanda kullanılmamış olandır. Sıcak tampon ise sık kullanılır ve yakın zamanda kullanılmıştır. Tampon ön belleğine veri taşınması gerektiği durumlarda soğuk olanlar öncelikli olarak kullanılır.

Tutarlı bir okuma, bir bloğun okunabilir bir sürümünün alınmasıdır. Örneğin, bir işlenmemiş işlem (uncomitted trancsaction) bir blokta iki satırı güncellediyse ve ayrı bir oturumdaki bir sorgu bu bloğu isterse, veritabanı bu bloğun okunabilir bir sürümünü (tutarlı okunaklı kopya olarak adlandırılır) oluşturmak için işlenmemiş güncellemeleri içermeyen geri alma verilerini kullanır.

Tampon havuzları aşağıdaki gibidir.

- **Varsayılan Havuz (Default Pool) :** Bu havuz, blokların normal olarak önbelleğe alındığı konumdur. Manuel olarak havuzlar yapılandırmadığı sürece, bu havuz kullanılır.
- **Tutma Havuzu (Keep Pool) :** Performansı artırmak için gün boyunca sıklıkla kullanılan tabloları, tablodaki blokları almak için gereken G/Ç'yi en aza indirgemek için bu havuz kullanılır.
- **Geri Dönüşüm Havuzu :** Bu havuz, nadiren kullanılan bloklar için tasarlanmıştır. Bir geri dönüşüm havuzu, nesnelerin önbellekte gereksiz yer kaplamasını engeller.

Redo Günlük Tamponu

Redo günlük tamponu, veritabanında redo girişlerini saklayan SGA'daki dairesel bir arabellektir. Redo girişleri, DML veya DDL işlemleri ile veritabanında yapılan değişiklikleri yeniden oluşturmak veya yeniden uygulamak için gereken bilgileri içerir. Veritabanı arka plan işlemlerinden LGWR, SGA'daki günlük denetim tamponunu çevrimiçi günlük dosyalarına kopyalar. Bir kullanıcının onayladığı işlemler çevrimiçi günlük dosyalarına yazılmadıkça tamamlanmış sayılmaz. Veritabanı hasarından oluşan veri kayıtları bu redo girişleri uygulanarak kurtarılır.

Kütüphane Önbelleği

Kütüphane önbellek, çalıştırılabilir SQL ve PL/SQL kodunu depolayan paylaşılan bir havuz bellek yapısıdır. Bu önbellek, paylaşılan SQL ve PL/SQL alanlarını, kilitleme gibi denetim yapılarını içerir. Paylaşılan bir sunucu mimarisinde, kitaplık önbelleğinde ayrıca özel SQL alanları bulunur. Bir SQL deyimi çalıştırıldığında, veritabanı daha önce yürütülen kodu tekrar kullanmayı dener. Bir SQL ifadesinin ayrıştırılmış bir gösterimi kitaplık önbelleğinde mevcutsa ve paylaşılabilirse, bu kodu yeniden kullanır. Aksi takdirde, veritabanı, uygulama kodunun zor bir ayrıştırma (hard parsing) olarak bilinen yeni bir çalıştırılabilir sürümünü oluşturmalıdır. Zor ayrıştırma yapılma sayısının artması durumlarında CPU'ya yük getireceği için performans düşüşüne neden olacaktır. Çalıştırılacak SQL kodun kütüphane önbelleğinde bulunma oranın yüksekliği veritabanı performans göstergelerinden bir tanesidir.

SQL deyimleri kütüphane önbelleğinde iki bölümde çalışır.

- **Paylaşımlı SQL Alanı :** Veritabanı bir SQL deyiminin ilk ortaya çıktığında işlemek için paylaşılan SQL alanını kullanır. Bu alan tüm kullanıcılar tarafından erişilebilir ve ayrıştırma ağacı (parse tree) ve yürütme planı (execution plan) deyimlerini içerir.
- **Özel SQL Alanı :** Bir SQL ifadesi veren her oturumun PGA'sında özel bir SQL alanı vardır. Aynı ifadeyi gönderen her kullanıcı, paylaşılan aynı SQL alanını gösteren özel bir SQL alanına sahiptir. Böylece, ayrı PGA'lardaki birçok özel SQL alanı aynı paylaşılan SQL alanıyla ilişkilendirilebilir.

Veritabanı, SQL deyimi çalıştırılmak üzere ayrıştırıldığında bellek havuzundan yer tahsis eder. Ayrılan belleğin boyutu SQL deyiminin boyutuna göre değişir. Ve bellek havuzunda LRU algoritmasına göre ayrılana kadar kalır. Sunucu oturum işlemi sonlansa bile diğer oturumlar tarafından kullanıldığı sürece bellek havuzunda kalmaya devam edecektir. Böylelikle SQL deyimlerinin tekrar tekrar ayrıştırılması engellenerek CPU maliyeti azaltılmış olur.

Aşağıdaki durumlarda da SQL deyimi paylaşılan bellek alanını terk ederek bir dahaki çalışma zamanında yeniden ayrıştırma maliyetine neden olur.

- ✓ Tablo, indeks gibi veritabanı objelerin istatistiklerinin yeniden toplanması durumunda;

- ✓ SQL deyimindeki herhangi bir obje DDL işlemi tarafından değiştirilmesi durumunda;
- ✓ Veritabanı genel adının değiştirilmesi durumunda;

Sistem performansının veritabanı olgusunun yeniden başlaması durumunda nasıl etkileneceğini test edebilmek için manuel olarak "ALTER SYSTEM FLUSH SHARED_POOL" komutu ile yapılabilir.

Veri Sözlüğü Önbelleği

Veri sözlüğü, veritabanı yapıları ve kullanıcılarının ayrıcalıkları ve rolleriyle ilgili referans bilgilerini içeren SYS ve SYSTEM şemaları tarafından sahibi olunan veritabanı tablolarının ve görünümlerin bir toplamıdır. Veritabanı, SQL deyimi ayrıştırma işlemi sırasında sıklıkla veri sözlüğüne erişir. *[1]* Veri sözlüğündeki tablolardaki veri blokları, kullanıcı sorgularının ve diğer DML komutlarının işlenmesine yardımcı olmak için sürekli olarak kullanılır. Veri sözlüğü önbelleği çok küçükse, veri sözlüğünden bilgi talepleri fazladan G/Ç oluşmasına neden olur; Bu G/Ç bağlı veri sözlüğü isteklerine özyinelemeli çağrı denir ve veri sözlüğü önbelleklerini doğru bir şekilde boyutlandırarak kaçınılmalıdır. *[2]*

Sunucu Sonuç Önbelleği

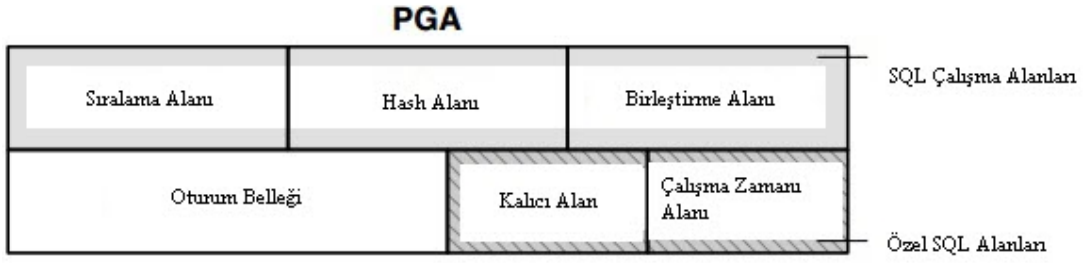
Diğer bellek havuzlarının aksine, sunucu sonuç önbelleği veri bloklarını değil SQL sorgularının ve PL/SQL fonksiyonlarının sonuç kümelerini tutar. SQL sorgularını sonuçları bu önbellekte kalarak gelecekteki benzer sorgularda kullanılarak performans artışı sağlanmış olur. Örneğin, bir uygulamanın aynı SELECT deyimini art arda çalıştırdığını varsayalım. Sonuçlar önbelleğe alınırsa, veritabanı hemen onları döndürür. Bu şekilde veritabanı, blokları yeniden okumak ve sonuçları yeniden hesaplamak gibi pahalı işlemleri önlemiş olur. Veritabanı, bir işlem, önbelleğe alınan sonucu oluşturmak için kullanılan veritabanı nesnelerinin verilerini veya meta verilerini değiştirdiğinde, önbelleğe alınmış bir sonucu otomatik olarak geçersiz kılar.

Kullanıcılar belleğin bu alanını kullanabilmek için sorgusuna RESULT_CACHE ipucunu verebilir. RESULT_CACHE_MODE başlatma parametresi, SQL sorgusu sonuç önbelleğinin tüm sorgular için (mümkünse) mı yoksa yalnızca açıklanan sorgular için mi kullanıldığını belirler.

1.4.2. Program Genel Alanı (PGA)

PGA, diğer işlemler veya iş parçacıkları tarafından paylaşılmayan işleme veya iş parçacığına özel bellek alanıdır. PGA işleme özgü olduğundan dolayı SGA içerisinde yer almaz. PGA ayrılmış veya paylaşımlı sunucu işlemi tarafından kullanılan kullanıcı oturumundaki değişkenleri içeren bellek yığınıdır.

Şekil 1.12 : PGA bileşenleri



http://docs.oracle.com/cd/E25178_01/server.1111/e25789/memory.htm

PGA'daki özel SQL alanı iki alandan oluşur:

- **Çalışma Zamanı Alanı :** Bu alan sorgu yürütme durumu bilgilerini içerir. Örneğin bu alan tam tablo taramasında o ana kadar alınan satır sayısını izler. Bu alan DML işlemleri bittikten hemen sonra boşaltılır.
- **Kalıcı Alan :** Bu alan bağlı değişkenlerin (bind variables) değerlerini içerir. SQL deyimi çalıştırıldığında bağlama değişkeni verilir. Kalıcı alan yalnızca imleç (cursor) kapatıldığında serbest bırakılır.

İstemci işlemi, özel SQL alanlarını yönetmekten sorumludur. Özel SQL alanlarının tahsisi ve iadesi büyük oranda uygulamaya bağlıdır, ancak istemci işlemi tarafından tahsis edilebilen özel SQL alanlarının sayısı OPEN_CURSORS başlatma parametresi ile sınırlıdır. Genel olarak, uygulamalar kalıcı alanı boşaltmak ve uygulama kullanıcıları için gerekli belleği en aza indirgemek için tekrar kullanılacak olan tüm açık imleçleri kapatmalıdır.

SQL Çalışma Alanları

Bir çalışma alanı, bellek yoğun operasyonlar için kullanılan özel bir PGA belleği tahsisidir. Örneğin, bir sıralama operatörü, bir satır kümesini sıralamak için sıralama alanını kullanır. Benzer şekilde, tabloları birleştirme işlemleri de bu alanda

yapılır. Operatörler tarafından işlenecek veri miktarı bir çalışma alanına sığmazsa, Veritabanı girdi verilerini daha küçük parçalara böler. Bu şekilde veritabanında bazı veri parçaları hafızada işlenirken gerisini daha sonra işlemek üzere geçici disk depolama birimine yazar.

Genellikle daha büyük çalışma alanları bellek tüketim maliyeti fazla olan işler önemli performans kazanımı sağlar. Çalışma alanının boyutu girdi verilerini ve yardımcı bellek yapılarını karşılayacak kadar büyük değilse, girdi datanın bir kısmı diskte tutulması gerekeceğinden kullanıcı yanıt süresi artar. Uç durumda çalışma alanı girdi verisine kıyasla çok küçükse, veri parçaları çoklu geçiş yapacağından kullanıcı yanıt süresi dramatik bir şekilde artacaktır.

1.4.3 Kullanıcı Genel Alanı (UGA)

UGA, oturum açma bilgileri ve bir veritabanı oturumunun gerektirdiği diğer bilgiler gibi oturum değişkenlerine ayrılan oturum belleğidir. Aslında UGA, oturum durumunu depolar.Bir oturum PL/SQL paketini belleğe yüklerse, UGA belirli bir zamanda tüm paket değişkenlerinde depolanan değişkenler kümesi olan paket durumunu içerir. Paket alt programı değişkenleri değiştirdiğinde paket durumu değişir. Varsayılan olarak, paket değişkenleri, oturumun yaşam süresi için benzersizdir ve geçerliliğini korur.

OLAP sayfa havuzu da UGA'da saklanır. Bu havuz, veri bloklarına eşdeğer olan OLAP veri sayfalarını yönetir. Sayfa havuzu bir OLAP oturumunun başında tahsis edilir ve oturumun sonunda serbest bırakılır. Kullanıcı, bir küp gibi boyutlu bir nesne sorduğunda otomatik olarak bir OLAP oturumu açılır.

UGA, oturumun ömrü boyunca bir veritabanı oturumunda kullanılabilmelidir. Bu nedenle, paylaşılan bir sunucu bağlantısı kullanırken UGA PGA'da depolanamaz; çünkü PGA tek bir işleme özgüdür. Bu nedenle, UGA, paylaşılan sunucu bağlantılarını kullanırken herhangi bir paylaşılan sunucu işlem erişimini etkinleştirerek SGA'da saklanır. Özel bir sunucu bağlantısı kullanırken, UGA PGA'da saklanır.

1.5 Veritabanı Uygulama Mimarisi

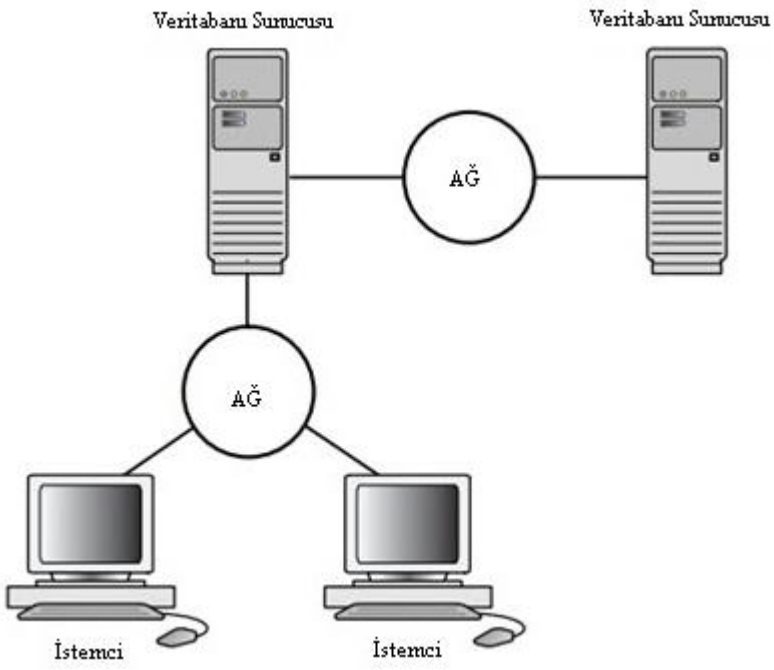
Bu bölümde veritabanına bağlanma mimarisi anlatılacaktır.

1.5.1. İstemci/Sunucu Mimarisi

İstemci, veritabanı bilgilerine erişen ve bir kullanıcıyla etkileşim kuran veritabanı uygulamasını çalıştırır. Sunucu, Veritabanı yazılımını çalıştırır ve bir veritabanına eşzamanlı, paylaşılan veri erişimi için gereken işlevleri gerçekleştirir.

İstemci uygulaması ve veritabanı aynı bilgisayarda çalışabilmesine rağmen, istemci bölümleri ve sunucu bölümü bir ağ üzerinden bağlı farklı bilgisayarlar tarafından çalıştırıldığında daha fazla verimlilik elde edilir.

Şekil 1.13: İstemci/Sunucu Mimarisi



[1]

<u>İstemci/Sunucu Mimarisinin Avantajları</u>

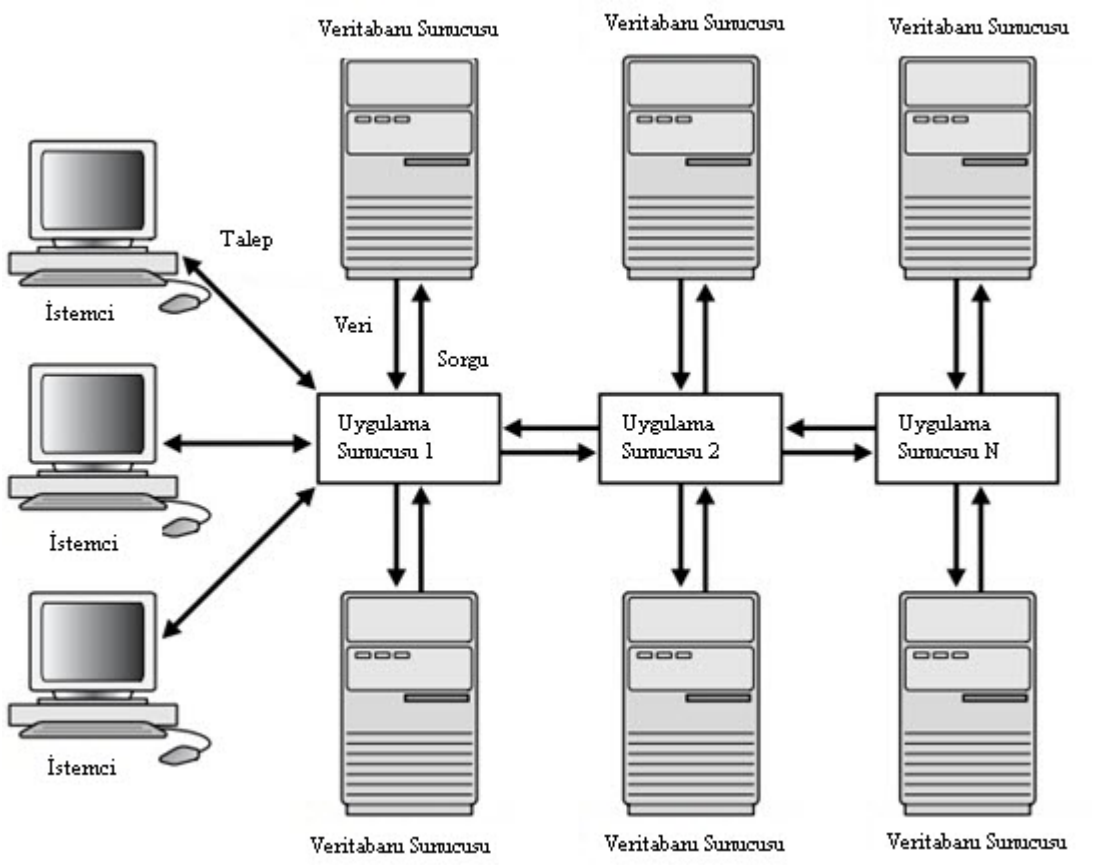
- ✓ Veri işleme işlemini gerçekleştirmek istemci uygulamaları sorumlu değildir. Daha ziyade, kullanıcılardan gelen girdileri istemekte, sunucudan veri talep etmekte ve bu verileri, istemci iş istasyonunun veya terminalin görüntüleme özelliklerini (örneğin, grafikler veya elektronik tablolar kullanarak) kullanarak analiz etmeleri ve sunmaları gerekmektedir.

- ✓ İstemci uygulamaları, verilerin fiziksel konumuna bağlı değildir. Verilerin taşınması veya diğer veritabanı sunucularına dağıtılması durumunda bile, uygulama az veya hiç değişiklik yapılmadan çalışmaya devam eder.
- ✓ Oracle Veritabanı, üzerinde çalıştığı işletim sisteminin çok görevli ve paylaşımlı bellek yapılarından yararlanır. Sonuç olarak, mümkün olan en yüksek düzeyde eşzamanlılık, veri bütünlüğü ve performansı istemci uygulamalarına sunar.
- ✓ İstemci iş istasyonları veya terminaller, veri sunumu için optimize edilebilirken (örneğin, grafik ve fare desteği sağlayarak) sunucu veri enformasyonu ve depolanması için optimize edilebilir (örneğin, büyük miktarda bellek ve disk alanı gereken durumlarda).
- ✓ Ağa bağlı ortamlarda, sunucunun uzak verilerine etkili bir şekilde erişmek için ucuz istemci iş istasyonlarını kullanılabilir.
- ✓ Veritabanı, sistemin ihtiyaçlarına göre ölçeklenebilir. Ağ üzerinde veritabanı işleme yükünü dağıtmak için birden fazla sunucu eklenebilir (yatay ölçeklendirme) veya daha büyük bir sistem performansından (dikey ölçeklendirme) yararlanmak için veritabanı bir mini bilgisayara veya ana bilgisayara taşınabilir. Her iki durumda da, veritabanı sistemler arasında taşınabilir olduğundan, veri ve uygulamalar az veya hiç değiştirilmeden tutulur.
- ✓ Ağa bağlı ortamlarda, paylaşılan veriler eşzamanlı erişimi yönetmek için daha kolay ve etkili hale getiren tüm bilgisayarlarda değil, sunucularda saklanır.
- ✓ Ağ ortamlarında, istemci uygulamaları, SQL deyimlerini kullanarak sunucuya veritabanı istekleri gönderir. Alınır alınmaz, her bir SQL deyimi sonuçları, istemciye döndüren sunucu tarafından işlenir. Ağ trafiği en aza indirilir, çünkü yalnızca istekler ve sonuçlar ağ üzerinden gönderilir. *(1)*

1.5.2. Çok Katmanlı Mimari

Çok katmanlı bir mimari ortamında, bir uygulama sunucusu istemciler için veri sağlar ve istemciler ile veritabanı sunucuları arasında bir arabirim görevi görür. Çok katlı mimari istemci, uygulama sunucusu ve veritabanı sunucusu bileşenlerini barındırır.

Şekil 1.14 : Çok Katmanlı Mimari



[1]

İstemci

Bir istemci, veritabanı sunucusunda gerçekleştirilecek bir işlem için bir istek başlatır. İstemci bir Web tarayıcı veya diğer son kullanıcı programı olabilir. Çok katmanlı bir mimaride, istemci bir veya daha fazla uygulama sunucusu aracılığıyla veritabanı sunucusuna bağlanır.

Uygulama Sunucusu

Bir uygulama sunucusu, istemcinin verisine erişim sağlar. İstemci ile bir veya daha fazla veritabanı sunucusu arasında bir arabirim görevi yapar ve uygulamaları barındırır. Bir uygulama sunucusu, minimum yazılım yapılandırmasıyla donatılmış istemcilerin, istemci bilgisayarların devamlı bakımını gerektirmeden uygulamalara erişmesine izin verir. Uygulama sunucusu, istemci iş istasyonundaki yükü azaltmak için istemci için bazı verileri yeniden biçimlendirebilir. Uygulama sunucusu, o istemci için veritabanı sunucusunda işlemler gerçekleştirirken istemcinin kimliğini

varsayar. Uygulama sunucusunun ayrıcalıkları, bir istemci işlemi sırasında gereksiz ve istenmeyen işlemleri gerçekleştirmesini engellemek için sınırlandırılmalıdır.

Veritabanı Sunucusu

Bir veritabanı sunucusu, bir istemci adına bir uygulama sunucusu tarafından istenen veriyi sağlar. Veritabanı tüm sorgu işlemlerini gerçekleştirir. Veritabanı sunucusu, istemci adına uygulama sunucusu tarafından gerçekleştirilen işlemleri ve kendi adına uygulama sunucusu tarafından gerçekleştirilen işlemleri denetleyebilir. Örneğin, bir istemci işlemi, bir uygulama sunucusu işlemi veritabanı sunucusuna bir bağlantı isteyebilirken istemcide görüntülenmesi için bilgi isteyebilir.

1.6 Veritabanı Ağ Mimarisi

Veritabanı ağ servisleri, dağıtımlı, heterojen bilgi işlem ortamları için kurumsal çapta bağlantı çözümleri sağlayan bir ağ bileşenleri grubudur. Veritabanı ağ servisleri, bir uygulamadan bir veritabanı olgusuna ve bir veritabanı olgusundan başka bir veritabanı olgusuna bir ağ oturumunu etkinleştirir.

Veritabanı ağ servisleri, dağıtılmış veritabanı ve dağıtılmış işlem sağlamak için geniş bir ağ dizisi tarafından desteklenen iletişim protokollerini veya uygulama programlı ara yüzlerini (API'ler) kullanır. Bir ağ oturumu kurulduktan sonra veritabanı ağ servisleri, istemci uygulaması ve veritabanı sunucusu için bir veri kurye görevi görür ve bir bağlantı kurar, bu bağlantıyı sürdürür ve ileti alışverişi yapar.

Veritabanı protokolleri, uygulamaların ara yüzünden SQL ifadeleri alır ve bunları desteklenen bir endüstri standardı üst düzey protokol veya API vasıtasıyla veritabanına aktarmak üzere paketler. Endüstri standardı protokollerin kullanılması, sunucu tarafı ve istemci tarafı platformları arasında bağımlılığın olmaması gerektiği anlamına gelir. TCP dünya çapında en popüler protokoldür, bu yüzden kurumlarda da genelde TCP kullanılır. Veritabanından gelen yanıtlar, aynı yüksek seviyeli iletişim mekanizmasıyla paketlenmiştir. Bu çalışma, ağ işletim sisteminden bağımsız olarak gerçekleşir. Veritabanı çalıştıran işletim sistemine bağlı olarak, veritabanı sunucusunun veritabanı ağ servisleri yazılımı sürücü yazılımını içerebilir ve ek bir arka plan işlemi başlatabilir.

1.6.1 Veritabanı Dinleyicisi

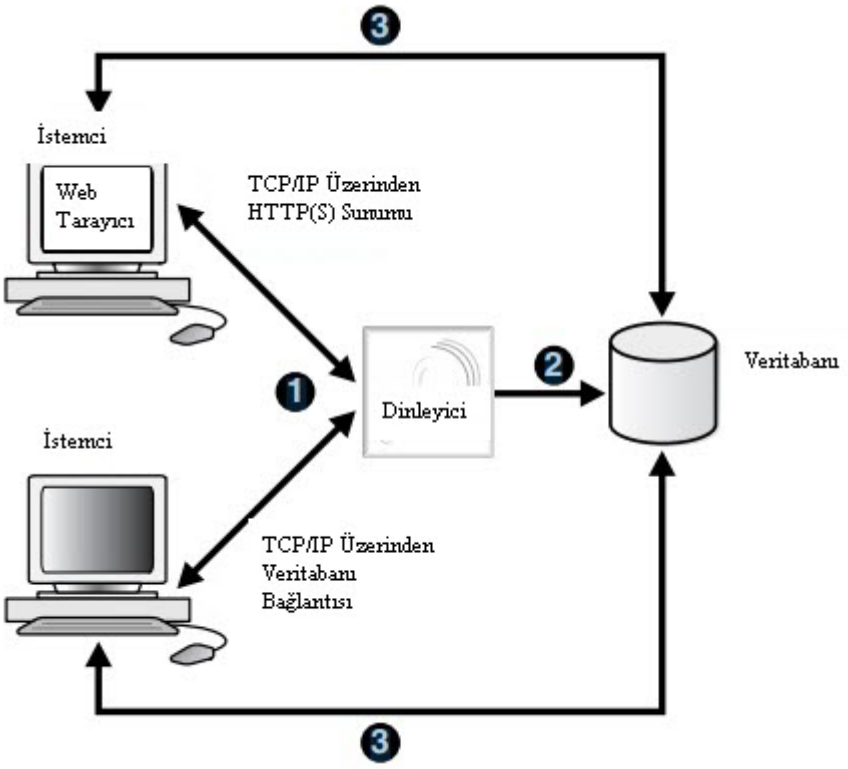
Veritabanı dinleyicisi (listener) gelen istemci bağlantı isteklerini dinleyen ve veritabanına gelen trafiği yöneten bir sunucu taraflı işlemdir. Veritabanı olgusu

başladığında ve yaşam döngüsünün çeşitli zamanlarında, veritabanı olgusu dinleyiciye başvurur ve veritabanı olgusuna iletişim yolu oluşturur.

Servis kaydı, dinleyicinin bir veritabanı servisinin ve servis işleyicisinin bulunup bulunmadığını belirlemesini sağlar. Bir servis işleyici, bir veritabanına bağlantı noktası olarak işlev gören adanmış bir sunucu işlemci dağıtıcısıdır. Servis kaydı sırasında PMON işlemi, dinleyiciye örnek adı, veritabanı hizmet adları ve hizmet işleyicilerinin türü ve adres bilgilerini sağlar. Bu bilgi, bir istemci isteği geldiğinde dinleyicinin bir servis işleyicisini başlatmasını sağlar.

Şekil 1.15, HTTP bağlantısı kuran bir tarayıcıyı ve bir dinleyiciden bir veritabanı bağlantısı kuran bir istemciyi gösterir. Dinleyicinin veritabanı sunucusunda bulunmasına gerek yoktur.

Şekil 1.15 : Dinleyici Mimarisi



[1]

1. Başka bir veritabanının bir istemci işlemcisi bir bağlantı istiyor.
2. Dinleyici, istemci isteğine hizmet etmek ve isteği işleyiciye iletmek için uygun bir servis işleyicisi seçer.

3. İstemci işlemi doğrudan servis işleyicisine bağlanır. Dinleyici artık iletişimle ilgilenmiyor.

1.6.2 İstemci Bağlanma Biçimleri

Veritabanına istemci üzerinden en çok kullanılan bağlantı yöntemleri aşağıdaki gibidir.

EZCONNECT

EZCONNECT, Oracle'ın kolay bağlantı adlandırma yöntemidir. EZCONNECT, bir TCP/IP ağı üzerinden bir Oracle veritabanına bağlanırken, tnsnames.ora dosyalarında servis adı aramaları gereksinimini ortadan kaldırır. Aslında, bu yöntemi kullanırken hiçbir adlandırma veya dizin sistemi gerekmez. İstemcilerin, veritabanının ana makine adına ek olarak isteğe bağlı bir bağlantı noktası ve servis adı ile bir veritabanına bağlanmasını sağlayarak ana makine adlandırma yönteminin işlevselliğini genişletir. Bu bağlanma şeklini kullanabilmek için veritabanı sunucusundaki sqlnet.ora dosyasında NAMES.DIRECTORY=(ezconnect) yazılmalıdır.*[29]* Örnek kullanımı şöyledir:

connect kullanıcı_adi/şifre@[//][sunucu_ip][port][/servis_adı

Yerel İsimlendirme

Yerel adlandırma ile, kullanıcı, bağlantı dizesi için bir veritabanı ağ servisi adı olarak bilinen bir takma ad sağlar ve takma ad, bir yerel dosya tarafından tam ağ adresine (protokol, adres, bağlantı noktası ve hizmet veya örnek adı) çözümlenir. Bu bilgiler $ORACLE_HOME/network/admin dizini altında bulunan tnsnames.ora isimli dosyada tutulur. Aşağıda bu isimlendirmeye örnek verilmiştir.

ORATEST= (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=17 2.17.23.146) (PORT=1521))(CONNECT_DATA= (SERVER=dedicated) (SERVICE_NAME=ORATEST)))

Bu yöntemle tnsnames.ora'daki veritabanına ait bilgilerde değişiklik olsa bile uygulama kodunda değişiklik yapmaya gerek yoktur. Sadece tnsnames.ora'yı güncellemek yetecektir.

Tek İstemci Erişim Adı

Tek İstemci Erişim Adı (SCAN), veritabanı RAC ortamlarında, istemcilerin bir kümede çalışan herhangi bir Oracle Veritabanına erişmek için tek bir ad sağlayan bir özelliktir. SCAN kümedeki veritabanları için küme takma adı olarak düşünülebilir. SCAN'in faydası kümeye düğümler veya veritabanları eklenmesi veya kaldırılması durumunda bile, istemcinin bağlandığı bilginin değiştirilmesi gerekmez.

EZCONNECT veya yerel isimlendirme bağlantı biçimlerinde SCAN aşağıdaki örnekte olduğu gibi kullanılabilir.

EZCONNECT: sqlplus system/manager@csbrac-scan:1521/oltp

Yerel İsimlendirme : ORATEST= (DESCRIPTION= (ADDRESS= (PROTOCOL=TCP) (HOST= csbrac-scan) (PORT=1521)) (CONNECT_DATA= (SERVER=dedicated) (SERVICE_NAME=ORATEST)))

Üstteki örnekte IP bilgisi yerine SCAN bilgisi olan csbrac-scan yazılmıştır. SCAN kullanarak istemci RAC içerisindeki veritabanı olgusu sayısı ve hangi veritabanı olgusunun aktif olduğunu bilmeksizin EZCONNECT veya yerel isimlendirme yöntemiyle bağlanabilir.

SCAN oluşturulurken ağ yöneticisi tarafından RR algoritması kullanarak üç IP adresine çözümleyen en az bir tek ad DNS'te tanımlanır. Kümedeki sunucuların sayısına bakılmaksızın yük dengeleme ve yüksek kullanılabilirlik gereksinimleri göz önüne alındığında üç IP adresi önerilir. IP adresleri, kümedeki varsayılan genel ağla aynı alt ağda olmalıdır. SCAN, alan adı dahil değil, 15 karakter veya daha az uzunlukta olmalı ve alan adı sonekiyle çözülmesi mümkün olmalıdır. SCAN için hangi IP grubunun tanımlandığı nslookup komutuyla kontrol edilir. SCAN eğer RR algoritmasına uyularak oluşturulduysa nslookup komutu çalıştırıldığında IP sıralaması farklı olacaktır. DNS düzeyinde RR algoritması, kümede yüzen SCAN dinleyicileri arasında bağlantı istekleri için yük dengelemesi sağlar. Bazen istemci taraflı DNS önbelleği kullanmak IP gruplarının hep aynı sırada gelmesine neden olabilir. Sanki RR kullanılmamış gibi düşünülebilir. İstemci tarafında DNS önbelleği kullanmak, DNS çözümleme süresini en aza indirmenin yanı sıra dış DNS sunucusuna yönelik DNS isteklerini en aza indirgemek için kullanılır.

1.6.3 Paylaşımlı Sunucu Mimarisi

Paylaşımlı bir sunucu mimarisinde, bir dağıtıcı, birden fazla gelen ağ oturumu isteğini, paylaşılan sunucu işlemlerinin bir havuzuna yönlendirerek, her bağlantı için özel bir sunucu işlemi gerekliliğini ortadan kaldırır. Havuzu boşta paylaşılan bir sunucu işlemi, ortak bir sıradan bir istek alır. Paylaşımlı sunucu kullanmanın faydaları şöyledir:

- ✓ İşletim sistemindeki işlem sayısını azaltır. Az sayıda paylaşımlı sunucu, birçok ayrılmış sunucu ile aynı miktarda işlem gerçekleştirebilir.
- ✓ Her ayrılmış veya paylaşılan sunucu bir PGA'ya sahiptir. Daha az sayıda sunucu işlemi, daha az PGA ve daha az işlem yönetimi anlamına gelir.
- ✓ Uygulama ölçeklenebilirliğini ve aynı anda veritabanına bağlanabilecek istemci sayısını artırır.
- ✓ İstemci bağlantıları ve bağlantı kesilmesi oranı yüksek olduğunda adanmış sunucudan daha hızlı olabilir.

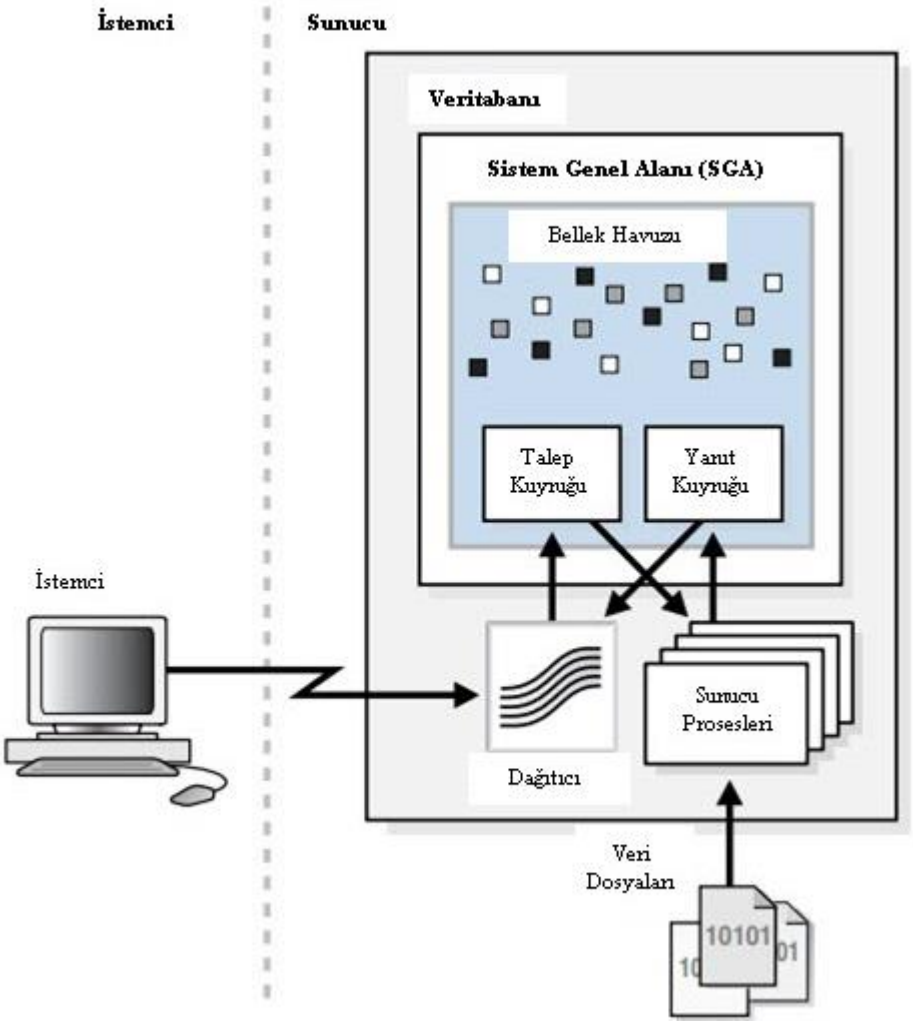
Paylaşılan sunucunun, bazı durumlarda daha yavaş tepki süresi, eksik özellik desteği ve kurulum ve ayarlama için artan karmaşıklık gibi çeşitli dezavantajları vardır. Yalnızca veritabanına işletim sistemi tarafından işleyebileceğinden daha fazla eşzamanlı bağlantı talepleri olduğu durumlarda paylaşımlı sunucu kullanılmalıdır.

Paylaşımlı sunucu mimarisi veritabanı olgusunun bir parçası olan ilave işlemler tarafından uygulanır. Bu arka plan işlemleri veritabanı olgusu başlama zamanında oluşurlar. Dağıtıcı (dispatcher) ve paylaşımlı sunucu olmak üzere iki işlemden oluşur. SGA içerisinde ayrıca ekstra kuyruk bellek yapıları vardır ve veritabanı dinleyicisi, paylaşılan sunucu için davranışını değiştirir. Dağıtıcılar dinleyiciyi bulmak için LOCAL_LISTENER parametresini kullanarak, dinleyiciyle iletişime geçip kaydolurlar. Bunlar, kavramsal olarak normal bir ayrılmış sunucu işlemine benzer, ancak bir oturuma bağlı değildir. SQL deyimlerini alır, ayrıştırır ve yürütürler ve bir sonuç kümesi oluştururlar ancak SQL deyimlerini doğrudan bir kullanıcı işleminden almazlar; bunları herhangi bir sayıda kullanıcı deyimleriyle doldurulan bir sıradan okurlar. Benzer şekilde,

paylaşılan sunucular sonuç kümelerini bir kullanıcı işlemine doğrudan getirmez; sonuç kümelerini bir yanıt kuyruğuna koyarlar.

Bir kullanıcı işlemi bir dinleyici ile bağlantı kurduğunda, sunucu işlemi başlatmak yerine dinleyici dağıtıcının adresini geri gönderir. Kullanıcı işlemi SQL deyimi gönderdiğinde bu dağıtıcıya gönderir. Dağıtıcı aldığı tüm ifadeleri kuyruğa koyar. Bu kuyruk bütün dağıtıcıların kullandığı ortak kuyruktur. Tüm paylaşımlı sunucu işlemleri ortak kuyruğu izler. İlk müsait olan paylaşımlı sunucu işlemi kuyruktan SQL deyimini alır ve bu deyimi ayrıştırır, değişkenlerini bağlar ve çalıştırır. Sonuçları kullanıcıya gönderme aşamasında paylaşımlı sunucu işlemi ile kullanıcı işlemi arasında herhangi bir bağlantı olmadığı için paylaşımlı sunucu işlemi sonuçları başka bir dağıtıcının kullanıcıya göndermesi için sonuç kuyruğuna atar.

Şekil 1.16 : Paylaşımlı Sunucu Çalışma şekli



[1]

Her bir paylaşılan sunucu oturumu için SGA'da kullanılan bellek, kullanıcı genel alanı (UGA) olarak bilinir. UGA'ları depolamak için kullanılan SGA'nın kısmı büyük havuzdur (large pool). Bu, large_pool_size parametresi ile manuel veya otomatik konfigure edilebilir. *[12]*

1.6.4 Ayrılmış Sunucu Mimarisi

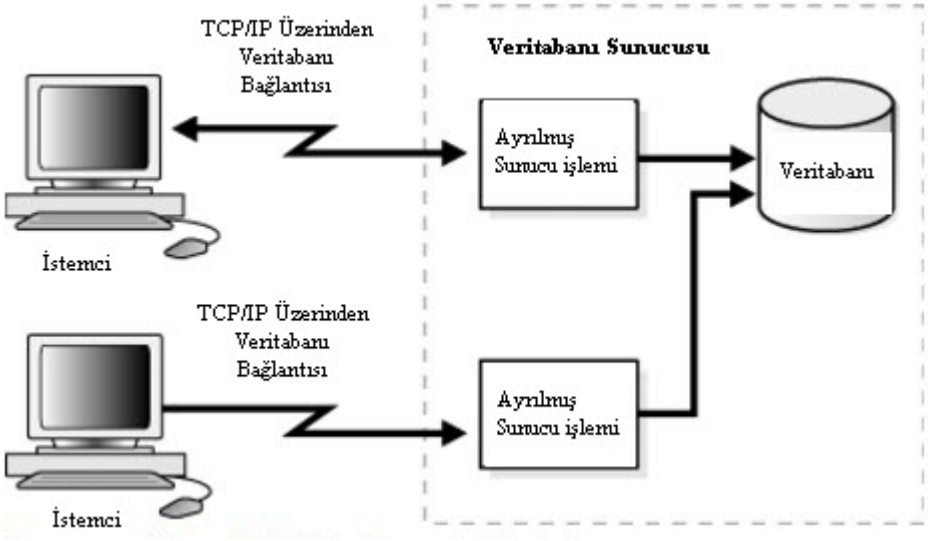
Ayrılmış sunucu mimarisinde, her bir istemci işlemi adına sunucu işlemi ayrılır. İstemci işlemleri ile sunucu işlemleri arasında bire bir oran bulunmaktadır. PMON, ayrılmış sunucu işlemleri hakkındaki bilgileri dinleyiciyle birlikte kaydeder. Bu, dinleyicinin bir istemci isteği geldiğinde özel bir sunucu işlemi başlatmasına ve isteği kendisine iletmesine olanak tanır.

Kullanıcı etkin olarak bir veritabanı isteği yapmadığında bile, bazı işletim sistemlerinde faal olmayan ayrılmış sunucu işlemi kalır. Kullanılmayan ayrılmış sunucular, bazen işletim sistemi kaynaklarının verimsiz kullanımı ile sonuçlanabilir.

Ayrılmış sunucu mimarileri, HTTP, FTP veya WebDAV istemcilerini desteklemez. Yalnızca veritabanı istemcileri desteklenir.

Bu mimaride bağlanabilmek için istemci tarafında tns dosyasında service=dedicated eklenerek yapılabilir.

Şekil 1.17 : Ayrılmış Sunucu Mimarisi



https://docs.oracle.com/cd/E11882_01/network.112/e41945/net_arch.htm#NETAG212

2. VERİTABANI PERFORMANS İYİLEŞTİRMESİ

Bu bölümde veritabanlarının kullanıcılara daha hızlı, doğru ve kesintisiz hizmet verebilmesi için, tablo tasarımının etkin olması, veritabanı nesnelerinin performanslı biçimde kullanabilmek, bir sorgu veritabanına geldiğinde çalışma aşamaları, yoğun kaynak tüketen SQL deyimlerinin belirlenmesi ve SQL deyimlerini performanslı biçimde oluşturma, veritabanı performans araçları gibi konular işlenecektir.

2.1 Veritabanı Normalizasyonu

Veritabanı normalizasyonu, veri fazlalığını azaltmak ve veri bütünlüğünü geliştirmek için ilişkisel bir veritabanının sütunlarını (öznitelikleri) ve tabloları

(ilişkilerini) organize etme işlemidir. Normalizasyon, öznitelikler arasındaki bağımlılıklara dayanarak tablolardaki özniteliklerin düzenlenmesini ve bağımlılıkların veritabanı bütünlüğü kısıtlamaları tarafından düzgün şekilde uygulanmasını sağlar. Normalizasyon, ya bir sentez ya da ayrışma işlemi ile bazı kurallar uygulayarak gerçekleştirilir. Sentez, bilinen bir bağımlılık kümesine dayanan normalizasyonu sağlanmış bir veritabanı tasarımını oluşturur. Ayrışma, mevcut (normalizasyon kurallarına uyulmamış) veritabanı tasarımını alır ve bilinen bağımlılık kümesine dayalı olarak geliştirir. *[30]*

Veritabanını normalizasyon kurallarına göre oluşturmak aşağıdaki faydaları amaçlar.

- ✓ Tablolar arasındaki ilişki kümelerini istenmeyen ekleme, güncelleştirme ve silme bağımlılıklarından kurtarmak;
- ✓ Yeni veri türleri ortaya çıktıkça ilişkilerin kümeleri için yeniden yapılanma ihtiyacını azaltmak ve böylece uygulama programlarının ömrünü uzatmak;
- ✓ Zamanla değişen istatistiklere yeniden uyum sağlayabilmek;
- ✓ Sorgu performanslarını artırmak;

Normalizasyon kurallarına uyulmadan oluşturulmuş veritabanlarının, veri değiştirmekten meydana gelen veritabanı anomaliler aşağıdaki gibidir.

- **Güncelleme Anomalisi :** Aynı bilgi birden fazla satırda ifade edilebilir; Bu nedenle tablodaki güncellemeler mantıksal tutarsızlıklarla sonuçlanabilir. Örneğin, bir çalışan yetkinlikleri tablosundaki her kayıt bir çalışan kimliği, çalışan adresi ve yetkinlik içerebilir; Dolayısıyla belirli bir çalışan için bir adres değişikliği potansiyel olarak birden fazla kayıt için uygulanması gerekecektir. Güncelleştirme başarılı bir şekilde gerçekleştirilmezse - ki bu, çalışanın adresi bazı kayıtlarda güncellenmişse ancak diğerleri tarafından güncellenmiyorsa, tablo tutarsız bir durumda kalır. Özellikle, tablo, bu belirli çalışanın adresinin ne olduğu sorusuna çelişkili cevaplar sunmaktadır. İşte bu güncelleme anomalisi olarak bilinir.
- **Ekleme Anomalisi :** Bazı durumlarda kayıt eklemede problem olabilir. Örneğin, bir "Fakülte ve Dersleri" tablosundaki her kayıt bir Fakülte Kimliğini, Fakülte Adı, Fakülte Kayıt Tarihi ve Ders Kodunu içerebilir. Böylece en az bir ders öğreten herhangi bir fakülte üyesinin ayrıntılarını kaydedilebilir, ancak ders kodunu sıfıra ayarlamak dışında herhangi bir ders

vermeye henüz atanmamış olan yeni işe alınan bir öğretim üyesinin ayrıntıları kaydedilemez. Bu duruma ekleme anomalisi denir.

- **Silme Anomalisi :** Bazı durumlarda, bazı gerçekleri temsil eden verilerin silinmesi, tamamen farklı gerçekleri temsil eden verilerin silinmesini gerektirir. Önceki örnekte anlatılan "Fakülte ve Dersleri" tablosu, bu tip anormalliklerden etkilenmektedir; Çünkü bir öğretim üyesi geçici olarak herhangi bir kursa atanmayı durdurursa, bu öğretim üyesinin silinmesi durumunda ders kodu da silinmiş olacak. Ve bu bilgi kaybolacak. Bu duruma silme anomalisi denir.*[30]*

Veritabanı normalizasyonu için birkaç kural bulunmaktadır. Her kurala "normal form" adı verilir. İlk kural kullanılıyorsa, veritabanının "ilk normal formda" olduğu söylenir. İlk üç kural kullanılıyorsa, veritabanı "üçüncü normal formda" olarak nitelendirilir. Başka normalleştirme düzeyleri de kullanılabilmekle birlikte, çoğu uygulama için en yüksek düzey üçüncü normal formdur. Genel olarak, normalleştirme için ek tablolar gerekir ve bazı geliştiriciler bunun ek yük getirdiğini düşünmektedir. Normalizasyonun ilk üç kuralından herhangi biri ihlal edilirse, normalizasyon anomalileri yaşanır. Normalizasyon formlarından bazıları aşağıda açıklanmıştır.

- **Birinci Normal Form (1NF) :** Birinci normal form ilişkisel veritabanlarında ilişkinin temel özelliğidir. Veritabanı normalizasyonunda birinci normal form minimum şarttır. Veritabanında bir ilişki birinci normal formda oluşturulabilmesi için her alanın özelliği atomik (bölünemez) değerler içermelidir. Birincil normal formda aşağıdaki kriterlere uyulmalıdır.
    - ✓ Bireysel tablolarda tekrarlanan grupları ortadan kaldırılmalıdır.
    - ✓ Her bir ilgili veri grubu için ayrı bir tablo oluşturulmalıdır.
    - ✓ Her bir ilgili veri kümesi birincil anahtar olarak belirlenmelidir.
- **İkinci Normal Form (2NF) :** Birinci normal form özelliğini taşıyan bir tablonun ikinci normal form özelliğini taşıyabilmesi için ilave ölçütleri karşılaması gerekir. İkinci normal formdaki bir tablonun asal olmayan öznitelikleri her aday anahtarının tamamına bağımlıdır. Herhangi bir aday

anahtarın bir bölümüne fonksiyonel bir bağımlılık 2NF'nin ihlalidir. Birincil anahtara ek olarak, tablo başka aday anahtarlar içerebilir.

Kurum veritabanları incelendiğinde normalizasyon kurallarına uymayan çok sayıda tablo bulunduğu tespit edilmiştir. MUBIS şemasına ait MAKS_RUHSAT, MAKS_YAPIKULLANIM, YAPIKULLANIMZINBELGESI ve YAPI şemasına ait UZLASMASAHIPLIK tabloları normalizasyon kurallarına uymadığı için en çok performans problemi yaşayan tablolardır. Veritabanı performansını artırmak için bunun gibi tablolar uygulama sahipleri ve geliştiricileri ile görüşerek normalizasyon kurallarına göre yeniden oluşturulmalıdır.

2.2 SQL Çalışma Planı

İstemci sunucuya bir SQL deyimi gönderdiğinde, ilk önce veritabanı deyimi sentaks yönünden kontrol eder. Sentaks denetimi, dil öğelerinin geçerli ifadeleri oluşturmak için doğru şekilde sıralanıp sıralanmadığını kontrol eder. SQL deyimi sentaks yönünden uygunsa, veritabanı artık deyimi anlamsal yönden değerlendirir. Veritabanı nesnelerine ve ana makine değişkenlerine yapılan tüm başvurular veritabanı tarafından geçerliliğinden emin olmak için incelenir. Veritabanı ayrıca, verilere erişim için başvuruyu yapan kullanıcının yetkili olup olmadığını denetler. Bu denetimler, SQL ifadesi tarafından başvurulan görünümleri ayrı sorgu bloklarına genişletir. *[5]*

SQL deyimi bahsedilen kontrolleri geçerse SQL ID değeri ve MD5 algoritmasından türetilmiş hash değeri alır. Hash değeri sorgunun ilk bir kaç yüz karakterinden türetilir. Bu yüzden uzun sorgular hash çakışması olabilir. Hash değeri ve SQL ID değerleri V$SQL görünümünde tutulur.

SQL sorgusu ayrıştırmanın son aşaması paylaşılan bellek havuzunda o sorguya ait ayrıştırma ağacı ve çalıştırma planları araştırmasıdır. Her benzersiz sorgunun yalnızca bir paylaşılan SQL alanı vardır. İki tür SQL ayrıştırması vardır.

- **Kolay Ayrıştırma :** Eğer SQL ifadesinin hash değeri paylaşılan havuzda bulunursa, veritabanı bu sorgu için yeniden ayrıştırma işlemlerine girişmez, zaten hali hazırda var olan çalışma planını doğrudan kullanır. Hash değeri hesaplanırken sorgunun büyük veya küçük harfle yazılmış olması, kelimeler arasında boşluk sayısının farklı veya sonunda yorum yapılmış olması hash değerini değiştirir. Bu yüzden bazen aynı sorgu sorgular yeniden ayrıştırma

işine en baştan girebilir. Bu da performansın düşmesine neden olur. Bu durumu önlemek için CURSOR_SHARING parametresi aşağıdaki komutla FORCE değirene geçirilmesi gerekir.

ALTER SYSTEM SET CURSOR_SHARING='FORCE';

- **Zor Ayrıştırma :** Eğer sorgunun SGA'da bulunanlardan farklı bir hash değerine sahipse veya ayrıştırılmış gösterimi paylaştırılamıyorsa veritabanı sorguyu optimize ediciye gönderir. Optimize edici daha sonra sıfırdan çalıştırılabilir bir sürüm oluşturur. Eğer sorgular sık sık zor ayrıştırmadan geçiyorsa bu performans düşüşüne neden olur.

Bir sorgunun çalışma planını görmek için önce sorgu EXPLAIN PLAN FOR ön komutuyla birlikte çalıştırılarak çalışma planı oluşturulur. Bu plan SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY) komutuyla tablo biçiminde görülür. Tablo yürütme ağacının soldan sağa, yukarıdan aşağıya bir taramasıdır. Her işlem, bu işlem hakkında daha ayrıntılı bilgilerle birlikte tek bir satırda basılır. İşlem sırası ve türleri optimize edicinin en düşük maliyeti kıstas alarak belirlediği en iyi plan tarafından belirlenir. Bir sorgunun maliyetine dahil edilen bileşenler şunlardır:

Kardinalite

Kardinalite, bir veritabanı tablosunun belirli bir sütunundaki verilerin benzersizliğine (veya eksikliği) işaret eder. Kardinalite bir sütundaki farklı öğelerin sayısının bir ölçütüdür. Düşük bir kardinallik, bir sütunun çok az sayıda farklı değer bulundurduğunu ve dolayısıyla çok seçici olmadığını gösterir. Çalışma planları bağlamında, kardinalite, her işlemden çıkması tahmin edilen satır sayısını gösterir. Kardinalite, tablo ve sütun istatistiklerinden hesaplanır.

Erişim Metodları

Veritabanının veriye erişim metolları aşağıdaki gibidir.

- Tablonun Tamamını Tarama (Full Table Scan) : Tam bir tablo taraması yığın düzenindeki bir tablodan tüm satırları okur ve WHERE yan tümcesi koşullarıyla eşleşmeyen satırları filtreler. Tablonun tamamını tarama yaptığı için performansı olumsuz olarak etkiler. Aşağıdaki etkenler tam tablo taramasına neden olur.
  - ✓ O tablo üzerinde indeks oluşturulmamış olması;
  - ✓ İndeksli sütuna fonksiyon uygulandığı için indeksin kullanılamaması;

- ✓ SELECT COUNT(*) kullanılması ve indeksin boş değerler içermesi;
- ✓ Tablo istatistiklerinin eski olması veya istatistiklerin en son toplanmasından sonra tablonun çok büyümesi;
- ✓ Sorgunun seçici olmamasından dolayı satırların büyük bölümüne erişilmesi;
- ✓ Tablodaki satır sayısı az ise indeks kullanmak yerine tablonun tamamını okumak daha az maliyetli olabilir;
- ✓ Sorgunun yüksek dereceli paralellik ile çalışması;
- ✓ Sorgunun FULL ipucunu kullanması;

- Tabloya ROWID bilgisi ile erişmek : Veritabanı, bir tablonun seçilen her satırını, veri dosyasını, o dosyadaki veri bloğunu ve o bloğun içindeki satırın konumunu belirten ROWID'e dayanarak araştırır. ROWID, WHERE yan tümcesi yükleminden veya bir indeks taraması aracılığıyla elde edilir. Çalışma planında TABLE ACCESS BY INDEX ROWID BATCHED satırı gösterilirse, veritabanı, indeksten bir grup ROWID alır ve sonra her bloğa erişilmesi gereken sayıyı azaltmak için satırlara blok sırasına göre erişmeye çalışır.
- SAMPLE ve SAMPLE_BLOCK yan tümcesi, bir tablodan rastgele bir veri örneği getiren tablo taramasıdır.
- Benzersiz İndeks Taraması (Index Unique Scan) : Birincil anahtar kısıtlaması nedeniyle benzersiz indeks taraması tek bir satır döndürür. Benzersiz bir indeksin tüm sütunlarına eşitlik operatörleri ile başvurulmalıdır.
- İndeks Aralığı Taraması (Index Range Scan) : İndeks aralığı taraması değerleri artan sırayla tarar. Veritabanı komşu indeks girdilerine erişir ve artan düzende satırları almak için ROWID değerlerini kullanır. Benzersiz olmayan bir indeksin önde gelen sütunları WHERE yan tümcesinde belirtilmişse veya benzersiz bir indeksin önde gelen sütunları, belirtilen tek değerler yerine aralıklara sahipse, veritabanı bir indeks aralığı taraması seçer. Süzgeç önekleri, dizin aralığı taramaları ve ROWID tarafından erişilen tablolarla bağlantılı olarak, tüm yaprak düğüm zincirine erişilmesi, okunması ve filtrelenmesi gerektiği için ölçeklenebilirlik sorunlarına neden olabilir. Daha fazla veri eklendiğinde, verilere erişme, okuma ve filtreleme zamanı da artar.

- Tam İndeks Taraması (Full Index Scan) : Tam bir indeks taraması olsa da indeksteki her bloğu okumaz. Bunun yerine kök bloğunu okur ve bir yaprak bloğuna ulaşana kadar dal bloklarının sol tarafında (artan tarama) veya sağ tarafta (aşağı doğru tarama) iner. Buradan indeksi bir blok-bazında okur.
- Hızlı Tam İndeks Taraması (Fast Full Index Scan) : Hızlı tam indeks taraması indeks bloklarını diskte olduğu gibi okur. Veritabanı, bir sorgu yalnızca indeksteki öznitelikleri sorduğunda bu erişim yoluna bakar. İndeks, sorgu için gerekli olan tüm sütunları içerdiğinde ve tam indeks anahtarındaki en az bir sütun bir NOT NULL kısıtlaması içerdiğinde tam bir tablo taramasına alternatiftir.

Birleştirme Metodları

Birleştirme yöntemleri, iki satır kaynağının birbirine nasıl bağlandığı yolunu gösterir. Optimize edici, bir tabloyu dış tablo ve diğerini iç tablo olarak belirtir. İki tablodan fazla tablo birleştirilmesi gerekiyorsa, optimize edici optimum birleştirme sırasını belirlemek için tüm permütasyonları analiz eder.

- Hash Birleştirme : Büyük veri kümelerini birleştirmek için genelde tercih edilir. Optimize edici iki veri kaynağından daha küçük bir hash tablosunu bellekte oluşturur. Bellekteki hash tablosundan veritabanı her değeri araştırır ve koşulun koşullarına uygunsa karşılık gelen satırı döndürür. Daha küçük veri kümesi belleğe sığarsa, her iki veri kümesinde de tek bir okuma geçişi maliyeti bulunur. Hash tablosu belleğe sığmazsa, veritabanı bölümlere ayırır. Birleştirme, daha sonra bölüm bölüm gerçekleştirilir, bu da çok fazla G/Ç'ye neden olarak performansı düşürür.
- İç içe Geçmiş Döngü Birleştirme : Küçük tabloların alt kümeleri birleştirildiğinde veya örneğin bir indeks arama ile iç tabloya etkili bir erişim yöntemi olduğunda genellikle kullanılır. Dış tablodan seçilen her satır için, veritabanı iç tablodaki tüm satırları tarar. İç tabloda bir dizin varsa, iç veri ROWID tarafından erişmek için kullanılabilir.
- Sıralı Birleştirme : Birleştirme koşullarında eşitsizlik olduğunda yapılır. Tablolardan birinde bir sıralama işlemini ortadan kaldıran bir indeks varsa genellikle kullanılır. Böyle bir indeks varsa ilk veri kümesindeki sıralama işlemi önlenebilir. İkinci veri kümesi, herhangi bir indekse bakılmaksızın

daima sıralanır. Genel olarak büyük veri kümeleri için iç içe döngüsel birleştirme daha iyi performans gösterir.

İki tablo herhangi bir koşul olmadan birleştirilmeye çalışılırsa, veritabanı iki tablonun Kartezyen çarpımını oluşturur. İki tablodan Kartezyen birleşim satırlarının sayısı, ilgili tabloların satır sayısının çarpımı olduğu için, büyük tablolarda performans sıkıntısına neden olur. Üretim veritabanlarında bu durumdan kaçınılmalıdır.

Paralel Çalıştırma

Veritabanı SQL deyimleri paralel çalışırken, birden fazla işlem tek bir SQL deyimini çalıştırmak için birlikte çalışır. İşleri çoklu işlemler arasında paylaştırarak yapmak, tek bir işlemin yapmasından çok daha hızlı olacaktır. Buna paralel çalıştırma ya da paralel işleme denir. Paralel yürütme, tipik olarak karar destek sistemleri (DSS) ve veri ambarı ile ilişkili geniş veritabanlarında veri yoğun operasyonlar için tepki süresini önemli ölçüde azaltır. Sorgu çalıştırmak tek bir veritabanı sisteminde birçok CPU arasında bölünebildiğinden, OLTP sistemler, kümelenmiş sistemler ve büyük paralel sistemler (MPP) paralel çalıştırmada en yüksek performans avantajlarından yararlanır.

Paralel yürütme, donanım kaynaklarının en iyi şekilde kullanılmasını sağlayarak sistemlerin performansı ölçeklendirmesine yardımcı olur. Paralel çalıştırma veritabanı CPU yükü düşükken çalıştırılırsa daha verimli olur. Bu yüzden mesai saatleri boyunca, çoğu OLTP sistemi paralel yürütmeyi kullanmamalıdır. Bununla birlikte, mesai saatler dışında, paralel yürütme, yüksek hacimli yığın işlemlerini etkin bir şekilde kullanılabilir.

Birkaç saniyelik düşük maliyetli sorgularda, paralel yürütme yararlı değildir çünkü paralel yürütme sunucularının koordinasyonuyla ilişkili bir maliyet vardır; Kısa işlemler için bu koordinasyonun maliyeti daha fazla olabilir.

Paralel yürütme, önemli miktarda veriye erişen birçok işlem türü için yararlıdır. Paralel yürütme aşağıdakilerin performansını artırır:

- ✓ Sorgular
- ✓ Büyük indekslerin oluşturulması
- ✓ Toplu eklemeler, güncellemeler ve silmeler
- ✓ Toplamalar (aggregation) ve kopyalamalar

Bir sistemde paralel yürütme yapabilmek için sistemin aşağıdaki özelliklerin hepsine sahip olması gerekir.

- ✓ Çok sayıda boşta CPU'su olması
- ✓ Yeterli G/Ç bant genişliği
- ✓ Sıralama, hashleme, birleştirme gibi bellek yoğun işlemleri destekleyecek büyüklükte bellek *[14]*

Paralel yürütme koordinatörü, bir SQL deyimini işlemek için örneğin paralel yürütme sunucularından iki veya daha fazlasını isteyebilir. Tek bir işlemle ilişkili paralel yürütme sunucularının sayısı, paralellik derecesi olarak bilinir. Eşzamanlı olarak iki setten fazla paralel yürütme sunucusu çalışabilir. Her bir paralel yürütme sunucusu kümesi, birden çok işlemi işleyebilir. Optimal işlemler arası paralellik garantilemek için yalnızca iki takım paralel yürütme sunucusu etkin olmalıdır. Paralel yürütme, hızlıca sorguları çalıştırmak için birden fazla CPU ve diski etkili bir şekilde kullanmak üzere tasarlanmıştır. Birden fazla kullanıcı aynı anda paralel yürütmeyi kullandığında, mevcut CPU, bellek ve disk kaynaklarını hızlı bir şekilde tüketmek kolaydır. Bu durumu önlemek için PARALLEL_ADAPTIVE_MULTI_USER sistem parametresi FALSE değerine getirilmelidir.*[10]*

Şekil 2.1 : Paralel sorgu çalışma planı örneği

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
Plan hash value: 4060011603
------------------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes |    TQ  |IN-OUT| PQ Distrib |
------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |   925 | 25900 |        |      |            |
|   1 |  PX COORDINATOR            |           |       |       |        |      |            |
|   2 |   PX SEND QC (RANDOM)      | :TQ10003  |   925 | 25900 |  Q1,03 | P->S | QC (RAND)  |
|   3 |    HASH GROUP BY           |           |   925 | 25900 |  Q1,03 | PCWP |            |
|   4 |     PX RECEIVE             |           |   925 | 25900 |  Q1,03 | PCWP |            |
|   5 |      PX SEND HASH          | :TQ10002  |   925 | 25900 |  Q1,02 | P->P | HASH       |
|*  6 |       HASH JOIN BUFFERED   |           |   925 | 25900 |  Q1,02 | PCWP |            |
|   7 |        PX RECEIVE          |           |   630 | 12600 |  Q1,02 | PCWP |            |
|   8 |         PX SEND HASH       | :TQ10000  |   630 | 12600 |  Q1,00 | P->P | HASH       |
|   9 |          PX BLOCK ITERATOR |           |   630 | 12600 |  Q1,00 | PCWC |            |
|  10 |           TABLE ACCESS FULL| CUSTOMERS |   630 | 12600 |  Q1,00 | PCWP |            |
|  11 |        PX RECEIVE          |           |   960 |  7680 |  Q1,02 | PCWP |            |
|  12 |         PX SEND HASH       | :TQ10001  |   960 |  7680 |  Q1,01 | P->P | HASH       |
|  13 |          PX BLOCK ITERATOR |           |   960 |  7680 |  Q1,01 | PCWC |            |
|  14 |           TABLE ACCESS FULL| SALES     |   960 |  7680 |  Q1,01 | PCWP |            |
------------------------------------------------------------------------------------------
```

https://docs.oracle.com/cd/E11882_01/server.112/e25523/parallel002.htm

Sorguları paralel çalıştırmak sorgunun çalışma zamanı performansını önemli ölçüde artırabilir. Sorgu koordinatörü (QC), bir SQL deyiminin paralel yürütülmesini başlatır. Paralel sunucu işlemleri (PSP), paralel olarak gerçekleştirilen gerçek işten sorumludur. QC, paralel çalışabilecek işleri PSP'ye dağıtırken paralel çalışamayacak işleri kendisi yapar. Örneğin SUM işleminde PSP alt toplamları hesaplar ancak QC son hesaplamayı elde etmek için her PSP'den alt toplamları eklemelidir.

PSP'nin yapabileceği en küçük çalışma birimi granüldür. Granüller blok tabanlıdır. Her PSP yalnızca kendi granülünü çalıştırır ve granül bittiğinde, tüm granüller işlenene kadar bir tane daha verilir. Paralellik derecesi (DOP) tipik olarak toplam granül sayısından çok daha küçüktür.

PSP'ler üreticiler ve tüketiciler olarak çiftler halinde birlikte çalışır. Üreticiler, yürütme planında PX SEND işleminin altındadır. PX RECEIVE satırı sonunda sonuçlarını QC'ye gönderen tüketicileri tanımlar.

2.3 Veritabanı Optimize Edici

Tüm SQL deyimleri, belirtilen verilere erişmenin en verimli araçlarını belirlemek için optimize ediciyi kullanır. Bir DML deyimini veritabanı bir çok adımda çalıştırır. Her adım, veritabanından fiziksel olarak veri satırlarını alır veya bunları deyimi veren kullanıcı için hazırlar. Çoğu zaman bir DML bildirimini işleme koymanın birçok farklı yolu mümkündür. Örneğin, tabloların veya indekslerin erişildiği sıra değişebilir. Veritabanının bir deyimi yürütmek için kullandığı adımlar deyimin ne kadar hızlı çalışacağını etkiler. Optimize edici olası yürütme yöntemlerini açıklayan yürütme planları üretir.

Optimize edici, sorgu koşulları, kullanılabilir erişim yolları, sistem için toplanan istatistikler ve ipuçları dahil olmak üzere çeşitli bilgi kaynaklarını göz önüne alarak hangi yürütme planının en verimli olduğunu belirler. Oracle tarafından işlenen herhangi bir SQL deyimi için, iyileştirici aşağıdaki işlemleri yapar:

- ✓ İfade ve koşulların değerlendirilmesi;
- ✓ Veriler hakkında daha fazla bilgi edinmek ve bu meta verilere dayalı olarak optimize etmek için bütünlük kısıtlamalarını denetlemek;
- ✓ Sql deyimi dönüşümünü;
- ✓ Optimize edicinin hedeflerinin seçimi;
- ✓ Erişim yollarının seçimi;
- ✓ Sıralama yaparken birleştirme kolonlarının seçimi;
- ✓ Optimize edici, bir sorguyu işleme olası yollarının çoğunu üretir ve oluşturulan yürütme planındaki her bir adıma bir maliyet atar. En düşük maliyetli plan, yürütülecek sorgu planı olarak seçilir. Optimize edici hedefini belirleyerek ve optimize edici için temsili istatistikler toplayarak optimize edicinin seçenekleri ayarlanabilir.

Örneğin, optimize edici hedefini aşağıdakilerden herhangi birine ayarlanabilir:

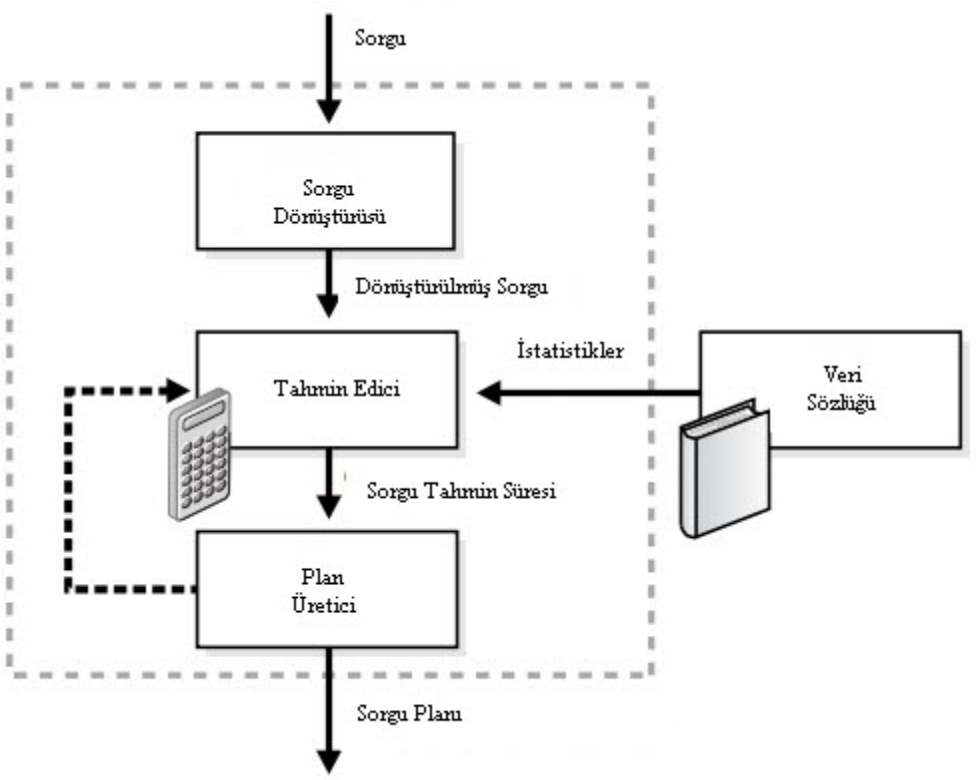
- ✓ Toplam Üretime göre : ALL_ROWS ipucu (hint), optimize ediciye, sonuçtaki son satırı mümkün olan en hızlı şekilde istemci uygulamasına almasını bildirir.
- ✓ İlk tepki süresine göre : FIRST_ROWS ipucu, optimize ediciyi ilk satırı istemciye olabildiğince çabuk almasını bildirir.

Tipik bir son kullanıcı etkileşimli uygulama, ilk tepki süresi optimizasyonundan yararlanırken toplu modlu, etkileşimsiz bir uygulama, toplam üretim optimizasyonundan fayda sağlayacaktır.

2.3.1 Veritabanı Optimize Edici Bileşenleri

Optimize edicinin bileşenleri Şekil 2.2'de gösterilmiştir.

Şekil 2.2:Optimize Edici Bileşenleri



[1]

Sorgu Dönüştürücü (Query transformer)

Sorgu dönüştürücü, iyileştiricinin daha iyi bir yürütme planı oluşturabilmesi için sorgunun biçimini değiştirmenin yararlı olup olmadığını belirler. Sorgu dönüştürücüsüne yapılan girdi, sorgu blokları kümesiyle temsil edilen ayrıştırılmış bir sorundur.

Tahmin Edici(Estimator)

Tahmin edici, belirli bir uygulama planının genel maliyetini istatistikleri kullanarak belirler. Tahminci, bu amaca ulaşmak için üç farklı kritere bakar.

- Seçicilik : Bu ölçü, bir satır kümesindeki satırların bir kısmını temsil eder. Seçicilik, bir sorgu koşullarının birleşimi gibi sorgu beyanına bağlıdır.
- Kardinalite : Bu ölçü, bir satır kümesindeki satır sayısını temsil eder.

- Maliyet : Bu ölçü, kullanılan iş birimi veya kaynak birimini temsil eder. Sorgu optimize edicisi, disk G/Ç miktarını, CPU ve bellek kullanımını iş birimleri olarak belirler.

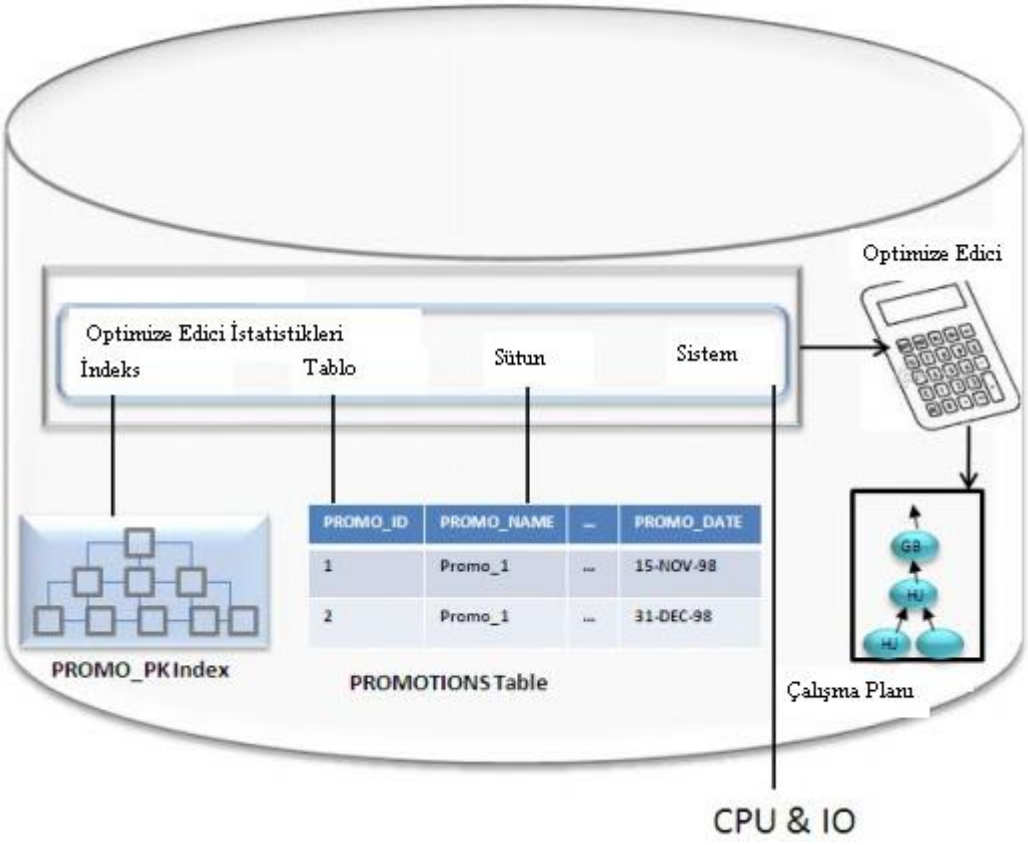
Plan Üretici

Plan üreteci, gönderilen bir sorgu için farklı planlar dener ve planı en düşük maliyetle seçer. Optimize Edici, iç içe geçmiş alt sorguların her biri için ayrık planlar ve ayrılmış bir sorgu bloğu ile temsil edilen birleştirilmemiş görünümler üretir. Plan üreteci, sorgu blokları için erişim yolları, birleştirme (join) metodları ve sıralama yapabilmek için farklı planları araştırır. Optimize edici otomatik olarak planları yönetir ve yalnızca doğrulanmış planların kullanılmasını sağlar. SPM, geçerli plandan daha iyi performans göstereceği doğrulandıktan sonra yeni bir plan kullanarak kontrollü plan gelişmesine izin verir.

2.3.2. Veritabanı Optimize Edici İstatistikleri

Optimize edici istatistikleri, veritabanındaki nesnelerle ilgili ayrıntıları açıklayan bir veri koleksiyonudur. Maliyet tabanlı bir optimize edicinin (CBO), bir yürütme planının maliyetini doğru bir şekilde belirleyebilmesi için, SQL deyiminde erişilen tüm nesneler (tablolar ve indeksler) hakkında ve SQL deyiminin çalıştırılacağı sistemle ilgili istatistiklerin zamanında ve doğru bir şekilde toplanması gerekir. İstatistikleri zamanında toplamak, kabul edilebilir performansı korumak açısından kritik önem taşır. *[10]*

Şekil 2.3: Veritabanı Optimize Edici İstatistiği



[10]

İstatistikler veri sözlüğünde saklanır ve DBA_TAB_STATISTICS gibi veri sözlüğü görünümleri kullanılarak erişilebilir.

Tablo ve Kolon İstatistikleri

Tablo istatistikleri, tablodaki satır sayısı, tabloda kullanılan veri bloğu sayısı ve tablodaki ortalama satır uzunluğu bilgileri içerir. Optimize Edici, bu bilgileri, bir yürütme planındaki çeşitli işlemlerin maliyetini hesaplamak ve işlemin üreteceği satır sayısını tahmin etmek için diğer istatistiklerle birlikte kullanır. Örneğin, bir tablo erişiminin maliyeti, DB_FILE_MULTIBLOCK_READ_COUNT parametresinin değeri ile birleştirilmiş veri bloğu sayısını kullanarak hesaplanır.

Kolon istatistikleri, bir sütundaki farklı değerlerin sayısı (NDV) ve kolondaki minimum ve maksimum değer hakkında bilgi içerir. Sütun istatistikleri DBA_TAB_COL_STATISTICS veri sözlüğü görünümünde sorgulanabilir. Optimize Edici, bir SQL işlemiyle döndürülecek satır sayısını tahmin etmek için sütun istatistik bilgilerini tablo istatistikleriyle (satır sayısı) birlikte kullanır. Optimize edici

döndürülecek satır sayısı için, veri dağılımını düzgün formda olarak değerlendirdiyse kayıt sayısını NDV değerine bölerek bulur.

İndeks İstatistikleri

İndeks istatistikleri, indeksteki farklı anahtarların (distinct keys) sayısı, indeks derinliği, indeksteki yaprak blokların sayısı ve kümeleme faktörü hakkında bilgi sağlar. Optimize Edici, indeks erişiminin maliyetini belirlemek için bu bilgileri diğer istatistiklerle birlikte kullanır.

Sistem İstatistikleri

Sistem istatistikleri CPU ve I/O performansı ile ilgili bilgileri tutar. Sistem istatistikleri varsayılan olarak etkindir ve otomatik olarak varsayılan değerlerle başlatılır; Bu değerler çoğu sistem için temsilidir. Sistem istatistikleri toplandığında, bu başlangıç değerlerini geçersiz kılacaktır. Sistem istatistiklerini toplamak için, ideal olarak iş yükünün az olduğu zamanlarda DBMS_STATS.GATHER_SYSTEM_STATS prosedürü kullanılır. Sistem istatistikleri yalnızca bir kez toplanmalıdır. Sistem istatistikleri, otomatik istatistik toplama işinin bir parçası olarak otomatik toplanmaz.

İstatistik Toplamak

Sürekli değişen veritabanı nesneleri için, istatistikler düzenli olarak toplanmalıdır, böylece veritabanı nesnesini doğru bir şekilde tanımlarlar. İstatistik toplamak için analyze komutu yerine DBMS_STATS paketinin içindeki prosedürler kullanılmalıdır. DBMS_STATS paketi istatistikleri toplamak ve yönetmek için 50'den fazla prosedür içermektedir ama bu prosedürlerin en önemlisi GATHER_*_STATS olanlarıdır. Bu prosedürler tablo, kolon ve indeks istatistikleri toplamak için kullanılabilir. Bu prosedürleri çalıştırabilmek için ANALYZE ANY sistem yetkisi veya DBA rolü gerekir.

Veritabanı, DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC dahili prosedürünü çağırarak Optimize Edici istatistiklerini toplar. Bu prosedürün yaptığı işi GATHER_AUTO seçeneğini kullanarak DBMS_STATS.GATHER_DATABASE_STATS prosedürü de yapar. Bu ikisi arasındaki birincil fark, veritabanının dahili olarak, istatistik gerektiren veritabanı nesnelerine öncelik vermesi ve böylece güncellenmiş istatistikleri en çok isteyen nesnelerin önce işlenmesi. Tanımlanmış istatistik toplama işleri

DBA_AUTOTASK_CLIENT_JOB görünümünden sorgulanabilir veya Enterprise Manager aracından kontrol edilir. Ayrıca Enterprise Manager aracından bu işlerin yönetimi de yapılabilir.

Tablo istatistikler, tablodaki satırların STALE_PERCENT (varsayılanı %10) kadarı değişikliğe (toplam ekleme, silme, güncelleme) uğradığında bayat kabul edilir. Veritabanı, tüm tablolar için DML etkinliğini izler ve SGA'ya kaydeder. İzleme bilgileri periyodik olarak diske boşaltılır ve DBA_TAB_MODIFICATIONS görünümünde görüntülenir. DBA_TAB_STATISTICS görünümündeki STALE_STATS kolonu YES olan tabloların istatistik bilgileri eski olduğu için optimize edici bu tabloları içeren sorgularda çalışma planı oluştururken sağlıklı planlar oluşturamayacaktır. Bu tablolar yukarıda açıklanan prosedürler çalıştırılarak güncel istatistik bilgisi oluşturulmalıdır.

Uygun istatistikleri toplamaya ek olarak, bunları yönetmek için kapsamlı bir çerçeve sağlamak da aynı derecede önemlidir. Veritabanı, istatistiklerini önceki bir sürüme geri yükleme yeteneği, istatistiği bir sistemden diğerine aktarma seçeneği veya istatistiksel değerleri manuel olarak ayarlama da dahil olmak üzere bunu yapmak için bir dizi yöntem sunmaktadır. Bu seçenekler belirli durumlarda son derece yararlıdır, ancak standart istatistik toplama yöntemlerini DBMS_STATS paketini kullanarak gerekmedikçe değiştirilmemelidir.

2.4 Veritabanı Perfomans Nesneleri

Bu bölümde doğru veya yanlış kullanımında performansı en çok etkileyen veritabanı performans nesneleri açıklanacak olup kurum veritabanında bu nesnelerin kullanımı ile ilgili öneriler sunulacaktır.

2.4.1. İndeksler

İndeksler, ilişkilendirildikleri nesnelerdeki verilerin mantıksal ve fiziksel olarak bağımsız oldukları şema nesneleridir. Dolayısıyla, indeks fiziksel olarak tablosunu etkilemeden düşürülebilir veya oluşturulabilir. İndeksi yokluğu veya varlığı herhangi bir SQL ifadesinin ifadesinde bir değişiklik gerektirmez. İndeks, tek bir veri satırına hızlı erişim yoludur. Yalnızca çalışma hızını etkiler.

İndekslerin iki işlevi vardır: Birincil anahtar ve benzersiz kısıtlamaları zorlamak ve performansı artırmak. Bir uygulamanın indeks oluşturma stratejisi performans için kritik önem taşır. İndeksler kısıtlama mekanizmasının bir parçasıdır.

Bir sütun veya sütunlar grubu bir tablonun birincil anahtarı olarak belirlenirse, o zaman tablonun içine bir satır eklendiğinde veritabanı, birincil anahtarda aynı değere sahip bir satırın mevcut olup olmadığını kontrol eder. Tablonun sütun (lar) üzerinde bir indeksi oluşturulmamışsa bunu yapmanın tek yolu tablonun tamamı taranarak her satırı kontrol etmek olacaktır. Yalnızca birkaç satırlık bir tablo için kabul edilebilir olsa da, binlerce veya milyonlarca veya milyarlarca satır içeren bir tablo için bu ciddi anlamda performans kaybı sebebidir. Bir indeks, anahtara anında erişilebilme becerisi verir; bu nedenle varlık kontrolü anlık olarak yapılabilir.

Benzersiz bir kısıtlama da bir indeks gerektirir. Benzersiz kısıtlamanın sütun veya sütunlarının boş bırakılabilmesi nedeniyle, birincil anahtar kısıtlamasından farklıdır. Bu, indeks oluşturulması ve kullanılmasını etkilemez. Yabancı anahtar kısıtlamaları indeksler tarafından zorlanır, ancak indeks ana tabloda kısıtlama olarak tanımlanması gerekmese de, ana tabloda mutlaka bulunmalıdır. Yabancı anahtar kısıtlaması alt tablodaki bir sütunu ana tablodaki birincil veya benzersiz bir anahtara ilişkilendirir. Alt tabloya bir satır eklendiğinde, veritabanı, eklemeye izin vermeden önce ana tabloda eşleşen satır bulunduğunu onaylamak için ana tablodaki indekste bir araştırma yapar. Bununla birlikte, performans nedenleriyle alt tablo içindeki yabancı anahtar sütunlarına her zaman indeks oluşturulmalıdır. Veritabanı, silinmekte olan satırı işaret eden alt tabloda herhangi bir satır bulunup bulunmadığını belirlemek için bir indeks kullanabilirse, üst tabloda bir DELETE daha hızlı olacaktır.

Kurum veritabanlarında bu tarz indekslerin eksikliğini tespit etmek için bir script oluşturulmuştur. Bu script şema bazlı çalışmaktadır. COB_ORTAK için bu script çalıştırıldığında oluşturulması gereken indeks komutları aşağıdadır. Performans artışı için bu çalışma bütün şemalar için ayrı ayrı yapılmalıdır.

Create index for table COB_ORTAK.COB_BELEDIYELER on columns IL_ID
Create index for table COB_ORTAK.COB_FIRMA on columns FIRMA_IL
Create index for table COB_ORTAK.COB_FIRMA on columns FIRMA_ILCE
Create index for table COB_ORTAK.COB_FIRMA on columns FIRMA_TURU
Create index for table COB_ORTAK.COB_KULLANICILAR on columns ILCE
Create index for table COB_ORTAK.COB_SITELER on columns IL_ID
Create index for table COB_ORTAK.COB_BELEDIYELER on columns ILCE_ID

Veritabanı WHERE yan tümcesi içeren herhangi bir SQL sorgusu çalıştırırken, atıfta bulunulan sütunlar üzerinde hiç indeks oluşturulmamışsa tam tablo taraması yapacaktır. Tam tablo taraması, ilgili satırları bulmak için tablonun her satırını okur. Tablonun milyarlarca satırı varsa, bu işlem birkaç saat sürebilir. İlgili sütun veya sütunlar üzerinde bir indeks varsa, veritabanı bunun yerine indeksi arar. Bir indeks, anahtar değerlerin sıralı bir listesidir ve aramayı çok verimli hale getirecek şekilde yapılandırılmıştır. Her bir anahtar değeri tablodaki satırı gösteren bir işaretçidir.

Bir indeks arama yoluyla ilgili satırları bulmak, tablo belirli bir boyutun üzerinde ve alınacak satırların oranı belirli bir değerin altındaysa tam bir tablo taraması kullanmaktan çok daha hızlıdır. Küçük tablolar için ya da tablonun satırlarının büyük bir bölümünü alacak bir WHERE yan tümcesi için tam bir tablo taraması daha hızlı olacaktır. Veritabanı indeks kullanıp kullanmayacağını toplanmış istatistik bilgilerine bakarak karar verir.

İndekslerin kullanılabileceği ikinci bir durumda sıralamadır. ORDER BY, GROUP BY veya UNION anahtar kelimelerini içeren bir SELECT deyimi, indeks oluşturulmamışsa sıralamayı en baştan yapmak zorunda kalacaktır. Bu da performans kaybına neden olacaktır.

Birden fazla tabloyu birleştirirken de performans artırmak için indeks kullanır. Tabloların boyutuna ve mevcut bellek kaynaklarına bağlı olarak, bazı durumlarda tabloları bellekte taramak ve birleştirmek indeks kullanmaktan daha çabuk olabilir.

İndeksler SELECT, UPDATE, DELETE veya MERGE içeren sorgularda performans artışı sağlarken INSERT içeren ifadelerde yavaş çalışacaktır. Bu yüzden indeks kullanımına dikkat edilmelidir. *[12]*

Kurum veritabanında kullanılmayan indeksleri tespit etmek için aşağıdaki komutlar sırasıyla çalıştırılarak Tablo 2.1 elde edilmiştir. Tablo 2.1 şemalara ait indekslerin ne kadarının kullanıldığını veya ne kadarının kullanılmadığını listeler. Kullanılmayan indeksler uygulama geliştiriciler ile görüşerek performans artışı ve diskleri verimli kullanmak için kaldırılmalıdır.

Sql>select 'alter index '||owner||'.'||index_name||' monitoring usage;' from dba_indexes where owner in (select username from dba_users where profile =

'LIMITED_PROFILE' and ACCOUNT_STATUS = 'OPEN') and STATUS = 'VALID' and INDEX_TYPE='NORMAL';

```
Sql> create or replace view VTYS.ALL_OBJECT_USAGE
(OWNER, INDEX_NAME,TABLE_NAME, MONITORING, USED,
START_MONITORING
,END_MONITORING) as
select u.name, io.name, t.name, decode(bitand(i.flags, 65536), 0, 'NO', 'YES')
, decode(bitand(ou.flags, 1), 0, 'NO', 'YES'), ou.start_monitoring, ou.end_monitoring
From sys.user$ u, sys.obj$ io, sys.obj$ t, sys.ind$ i, sys.object_usage ou
Where i.obj# = ou.obj# and io.obj# = ou.obj# and t.obj# = i.bo# and u.user# =
io.owner#;
Sql> select owner as SEMA,SUM(decode(used,'YES',1,0)) as kullanılan,
SUM(decode(used,'NO',1,0))  as kullanılmayan  from
VTYS.ALL_OBJECT_USAGE where monitoring = 'YES'  group by owner order by
2  desc;
```

Tablo 2.1 : Kurum Veritabanlarındaki İndekslerin Kullanılma Durumu

ŞEMA	KULLANILAN	KULLANILMAYAN
CDP	2.074	7.945
EPLAN	533	758
KL_ATLAS	507	3.331
GURULTU	397	3.173
CBS	195	331
IZINLISANS	141	191
MVAS	128	382
BAKANLIK_VT	116	722
AMBALAJ	112	104
EGZOZ	94	168
MUBIS	91	126
CED	82	47
EDENETIM	75	54
SERA_MRV2	65	55
KOOP	63	42
KKS	54	385
CEBECI	52	37
KL_PLAN	45	124
BLTSHR	41	49
DYS_TAU	39	162
COB_ORTAK	36	37
SEVESO	32	58

IIA	23	33
KBS	20	41
OTA	18	17
WUGEP	17	47
MAPCODEXSERVER	16	18
GEOCODING	12	8
YAKMA_TESISLERI	11	17
LAB	9	12
INFO_OTIM	9	21
AKTARIM	9	3
FINANS	8	20
SIGNART_V2	7	11
ATS	7	106
GATS	5	48
BELKAY	5	8
SERA_GAZI	3	17
LAB_YETERLIK	3	5
UAVT	2	20
EDYS	2	172
WEB_SERVICES	2	2
KBS_ANKET	1	16
KENTGES	1	27
ORTAK	0	3
SERA_MRV	0	77
YAPI	0	323
KIRA	0	4
GURULTU_KONTROL	0	14
ATIKSU_BS	0	77
TR_PLAJLARI_GIS	0	19
MAPINFO	0	1
SIGNART	0	20

İndeksler aşağıdaki gibi kategorize edilebilir:

- B-Tree İndeksler : Bu tip indeksler en yaygın indeks türüdür. B-tree indeksi, aralıklara bölünmüş değerlerin sıralı bir listesidir. Bir anahtarı bir satır veya satır aralığı ile ilişkilendirerek, tam eşleme ve aralık aramaları da dahil olmak üzere geniş bir yelpazedeki sorgular için mükemmel arama performansı sağlar.
- Bitmap İndeksler : Bir bitmap indeksinde, veritabanı her indeks anahtarı için bir bitmap saklar. Geleneksel bir B-tree indeksinde, bir indeks girişi tek bir satıra işaret ederken bir bitmap indeksinde, her indeks anahtar işaretçisi birden çok satıra işaret eder. Bitmap indeksleri öncelikle veri ambarı veya

sorguların çok sayıda sütuna geçici olarak başvurduğu ortamlar için tasarlanmıştır. Bu tür indeksler genellikle kardinalitesi küçük sütunlarda tercih edilir.

- Fonksiyon Bazlı İndeksler : Fonksiyon tabanlı bir indeks, bir veya daha fazla sütun içeren bir fonksiyonun veya ifadenin değerini hesaplar ve indekste depolar. Fonksiyon tabanlı bir indeks, bir B-tree veya bir bitmap indeksi olabilir. İndeks oluşturmak için kullanılan fonksiyon, bir aritmetik ifade veya bir SQL fonksiyonu, kullanıcı tanımlı PL/SQL fonksiyonu, paket fonksiyonu olabilir. Fonksiyon tabanlı indeksler, WHERE yan tümcelerinde fonksiyon içeren deyimleri değerlendirmede etkilidir. Fonksiyon bir sorguda bulunduğunda, veritabanı yalnızca fonksiyon tabanlı indeksi kullanır. Bu indeks türünde DML işlemleri oldukça maliyetlidir. *[1]*

Kurum veritabanı üzerinde fonksiyon bazlı ve bitmap indeks kullanım durumunu tespit etmek için aşağıdaki sorgu çalıştırıldığında Tablo 2.2 elde edilmiştir.

```
Sql>select owner,index_type,count(*) from dba_indexes where owner in (select
username from dba_users where profile = 'LIMITED_PROFILE' and
ACCOUNT_STATUS = 'OPEN')  and index_type in ('FUNCTION-BASED
NORMAL','BITMAP') group by owner,index_type  order by 1,2,3 desc
```

Tablo 2.2:Fonksiyon Bazlı ve Bitmap İndeks Kullanım Durumu

OWNER	INDEX_TYPE	COUNT(*)
CBS	FUNCTION-BASED	4
EPLAN	BITMAP	77
EPLAN	FUNCTION-BASED	10
IZINLISANS	FUNCTION-BASED	1
KL_ATLAS	FUNCTION-BASED	1
MUBIS	FUNCTION-BASED	6
SIGNART_V2	FUNCTION-BASED	1
YAPI	FUNCTION-BASED	1

Bir indeks bölümünün tablo alanı, ya sahibinin varsayılan tablo alanı ya da CREATE INDEX deyiminde özel olarak adlandırılmış bir tablo alanı olabilir. Bir indeksi, tablodan ayrı bir tablo alanına depolamak yönetim kolaylığı sağlar. Örneğin, sadece indeks içeren, yeniden oluşturulabilen tablo alanlarını yedeklememek, yedekleme için gereken zaman ve depolama alanını azaltır.

Kurum veritabanlarında indekslerin ayrı tablo alanlarında tutulmadığı tespit edilmiştir. CREATE TABLESPACE anahtar kelimesi ile her şemanın indekslerinin tutulacağı ayrı bir tablo alanı oluşturulmalıdır. Hali hazırda var olan bütün indeksler aşağıdaki komut kullanılarak yeni tablo alanına taşınmalıdır.

ALTER INDEX OWNER.INDEX_NAME REBUILD TABLESPACE;

2.4.2 Bölümleme (Partitioning)

Bölümleme, genellikle çok büyük veritabanları (VLDB) için geliştirilmiş uygulama performansının anahtarıdır. Nedeni basittir: Veriler parçalara bölünmüştür ve her bölüm bir tabloda olduğu gibi davranır: kendi adına ve saklama özelliklerine sahiptir. Her bölüm aynı tablodaki diğer tüm bölümlerden bağımsız olarak yönetilir ve işletilir. Fiziksel olarak, farklı bölümler için veriler ayrı bölümlerde saklanır; Mantıksal olarak, farklı bölümler hala tüm tabloyu oluşturur ve SQL ifadeleri değişiklik gerektirmez. Her bölüm tablonun diğer bölümlerini etkilemeden kendi üzerinde ayrı indeks oluşturulabilir, diğer bölümlerden bağımsız olarak düşürülebilir veya nologging seçeneği ile büyük işlemlerin etkisi azaltılabilir.

Bölümleme, diskten alınan verilerin miktarını önemli ölçüde azaltır ve işleme süresini kısaltır; böylece sorgu performansını iyileştirir ve kaynak kullanımını optimize eder. Bölümlemenin avantajları aşağıdaki gibi sıralanabilir:

- ✓ Kullanılabilirlik artışı : Bir bölümün kullanılamaması nesnenin kullanılamamasını gerektirmez. Optimize edici, bölümler kullanılamadığında, sorgudan ayrılmış bölümleri otomatik olarak kaldırır ve sorgular etkilenmez.
- ✓ Şema nesnelerinin daha kolay yönetimi : Bölünmüş bir nesnenin, topluca veya tek tek yönetilebilen parçaları vardır. DDL ifadeleri, tüm tablolar veya indeksler yerine bölümleri işleyebilir. Bu nedenle, bir indeks veya tablo yeniden oluşturma gibi kaynak yoğun görevleri parçalama imkanı sunar. Örneğin, bir seferde bir tablo bölümü taşınabilir. Bir sorun ortaya çıkarsa, yalnızca bölüm taşıması yeniden yapılmalıdır, tablonun tamamı değil. Ayrıca, bir bölümü bırakmak, çok sayıda DELETE deyiminin çalıştırılmasını önleyerek daha az redo girdisi üretir.
- ✓ DML işlemleri birden çok segment üzerinde dağıtıldığı için OLTP sistemlerinde paylaşılan kaynaklar için yapılan çekişmeler azalır.
- ✓ Sorgu performanslarında artış sağlar. *[5]*

Bölümleme yapılmış tabloları birleştirmeler paralel olarak yürütülürken, paralel yürütme sunucuları arasında değiştirilen verilerin miktarını en aza indirgeyerek sorgu yanıt süresini azaltır. Bu, tepki süresini önemli ölçüde azaltır ve CPU ve bellek kaynaklarının kullanımını geliştirir. RAC ortamlarında, bölümleme-yönlü birleştirmeler ayrıca, yoğun birleşim işlemleri için iyi ölçeklenebilirlik elde etmenin anahtarı olan, ara bağlantı üzerinden veri trafiğini önler veya en azından sınırlar.

Bir tablonun veya indeksin her bölümü, sütun adları, veri türleri ve kısıtlamalar gibi aynı mantıksal özeliklere sahip olmalıdır. Örneğin, bir tablodaki tüm bölümler aynı sütun ve kısıtlama tanımlarını paylaşır ve bir indeksteki tüm bölümler, aynı indekse ekli sütunları paylaşır. Bununla birlikte, her bölüm, ait olduğu tablo alanı gibi ayrı fiziksel niteliklere sahip olabilir. Böylelikle değişmeyen bölümler ayrı bir çevrimdışı tablo alanına taşınarak bir kere yedeklendikten sonra günlük yedeğini almaya gerek kalmayabilir.

Bölümlenmiş bir tablodaki her satırın gitmesi gereken bölümü, bir veya daha fazla sütun kümesinden oluşan bölümleme anahtarı belirler. Her satır, tek bir bölümleme bu anahtarı tarafından açıkça atanmıştır. Veritabanı, bölümleme anahtarını kullanarak otomatik olarak ekleme, güncelleme ve silme işlemlerini uygun bölüme yönlendirir.

Bölümleme Stratejileri

Veritabanı, verilerin bölümlere nasıl yerleştirildiğini denetleyen birkaç bölümleme stratejisi sunar. Temel stratejiler aralık, liste ve hash bölümlemedir.

- Aralık Bölümlemesi : Bu bölümleme işleminde, veritabanı, satırları bölümleme anahtarının değer aralıklarına dayalı bölümlere eşler. Çoğunlukla bu bölümleme tipi kullanılır ve genellikle tarihlerde kullanılır. PARTITION BY RANGE komutuyla tanımlanan kurala göre her satır ilgili bölümde yerini alır.
- Liste Bölümlemesi : Bu bölümleme işleminde, veritabanı, her bölüm için bölüm anahtarı olarak ayrı değerlerin bir listesini kullanır. Birbirinden ayrı satırların belirli bölümlerle nasıl yerleşeceğini belirmek için liste bölümlemesi işlemi kullanılır. Bu yöntemle kullanılan kullanılan böümleme anahtarının uygun bir şekilde sıralanmasıyla ilgili veri kümeleri gruplanır ve

organize edilir. PARTITION BY LIST komutuyla tanımlanan kurala göre her satır ilgili bölümde yerini alır.

- Hash Bölümlemesi : Bu bölümleme işleminde, veritabanı, satırları, kullanıcı tarafından belirtilen bölümleme anahtarı için geçerli olduğu hashleme algoritma temel alınarak bölümlere yerleştirir. Satırın varış yeri, veritabanından satıra uygulanan dahili hash fonksiyonu ile belirlenir. Hash algoritması, satırları bölümler arasında eşit olarak dağıtacak şekilde tasarlanmıştır, böylece her bölüm yaklaşık aynı sayıda satır içerir. Hash bölümlemesi, yönetilebilirliği artırmak için büyük tabloları bölmek için kullanışlıdır. Tek büyük bir tabloyu yönetmek yerine küçük tabloları yönetmek daha kolay ve performanslıdır. Hash bölümlemesi, yüksek güncelleme çekişmesine sahip OLTP sistemlerinde de yararlıdır. PARTITION BY HASH komutuyla tanımlanan kurala göre her satır ilgili bölümde yerini alır. PARTITION PARTITION_NUM seçeneği ile de tablonun kaç bölümü olacağı daha en baştan belirlenir. *[1]*

Bölümleme performansı önemli ölçüde artırmasına rağmen kurum veritabanlarında kullanılmadığı tespit edilmiştir. Kurumdaki en çok kayıt içeren tabloları listeleyen Tablo 2.3'teki tablolar PARTITION anahtar kelimesini kullanarak bölümlenmelidir.

Tablo 2.3: Kurum Veritabanındaki En Çok Kayıt İçeren Tabloların Bölümlenme Durumu

OWNER	TABLE_NAME	NUM_ROWS	PARTITIONED
SDE	STATE_LINEAGES	601.130.018	NO
KL_ATLAS	S2167_IDX$	82.362.515	NO
KL_ATLAS	TKG_PARSELLER	51.700.319	NO
UAVT	BAGIMSIZBOLUM	45.896.387	NO
LOG	COB_LOG_ARSIV	32.109.026	NO
UAVT	BINA	27.703.526	NO
MUBIS	MAKS_YPBAGIMSIZBOLM	23.344.804	NO
WEB_SERVICES	UYGULAMA_LOG	22.357.341	NO
MUBIS	MAKS_YPTESISAT	18.462.426	NO
MVAS	S2479_IDX$	14.895.263	NO

2.4.3. Büyük Objeler (Large Objects)

Büyük Nesneler (LOB), büyük miktarda veri tutmak için tasarlanmış bir veri türleri kümesidir. Bir LOB, veritabanının nasıl yapılandırıldığına bağlı olarak 8 terabayt ile 128 terabayt arasında değişen maksimum bir boyutu tutabilir. Büyük

verileri LOB olarak kaydetmek, verilerin uygulamalardan verimli bir şekilde erişilmesini sağlar.

Veritabanı tarafından yorumlanmayan, uygulama veya harici bir servis tarafından işlenen mantıksal olarak daha küçük parçalara bölünemeyen XML belgesi gibi yarı yapılandırılmış veriler ve ikili (binary) dosya olarak saklanan resim gibi yapılandırılmamış veriler LOB'lar için oldukça uygundur. Yarı yapılandırılmış verilerle ilgili uygulamalar tipik olarak büyük miktarda karakter verisi kullanır. CLOB ve NCLOB veri tipleri, bu tür verileri depolamak ve düzenlemek için idealdir. Grafik resimler, hareketsiz video klipler, tam hareketli video ve ses dalga biçimleri gibi yapılandırılmamış veriler veri için ideal olan SQL veri türleri, BLOB ve BFILE veri türü içerir. Tipik bir çalışan kaydı birkaç yüz bayt olabilirken, az miktarda multimedya veri bile binlerce kez daha büyük olabilir.*[16]*

LOB'lar veritabanından erişilen işletim sistemi dosyalarında saklanır. Bir LOB sütunu içeren bir tablo oluşturulduğunda, belirtilen bölümdeki LOB sütununu tutmak için veritabanı tarafından iki segment oluşturulur. Bu segmentler LOBSEGMENT ve LOBINDEX türündedir. LOBINDEX segmenti, LOBSEGMENT bölümünde saklanan LOB parçalarına erişmek için kullanılır. LOBSEGMENT ve LOBINDEX segmentleri, aksi belirtilmediği halde LOB içeren tabloyla aynı tablolama alanında saklanır. *[8]* LOB indeksi, LOB depolama alanıyla güçlü bir şekilde ilişkilendirilen iç yapıdır. Bu yüzden LOB indeksleri diğer indekslerin aksine düşürülüp yeniden oluşturulamaz.

LOB verileri için bir tablo alanı belirtilmezse, LOB veri ve indeksleri için tablonun tablo alanı kullanılır. LOB'lar için en iyi performans, LOB'ları içeren tablo için kullanılan tablonun tablo alanından farklı olan bir LOB için depolama alanı belirleyerek elde edilebilir. Birçok farklı LOB'a sıkça erişilecek olursa, diskteki çekişmeleri azaltmak için her bir LOB sütunu veya özniteliği için ayrı bir tablo alanı belirlemek yararlı olur.

LOB'ların kurum veritabanlarında çokça kullanıldığı tespit edilmiştir. Kurum veritabanlarının büyük bir kısmı LOB verilerinden oluşmaktadır. Bu durumda yedekleme sürelerinin gün içerisine sarkarak veritabanı performansını azaltmaktadır. Ayrıca LOB'lar yüzünden yoğun miktarda dosya okuma yazması olduğu için diski kullanan diğer şemalar olumsuz etkilenmektedir. Bu durumun önüne geçmek için LOB'lar veritabanı yerine ayrı bir dosya sunucusunda tutulmalıdır. Kurum

veritabanlarında LOB'ların ayrı tablo alanlarında tutulmadığı tespit edilmiştir. Varolan LOB'lar CREATE TABLESPACE anahtar kelimesi ile ayrı bir disk üzerinde her şemanın LOB'larının tutulacağı ayrı bir tablo alanı oluşturulmalıdır. Aşağıdaki komut kullanılarak yeni tablo alanına taşınmalıdır.

ALTER TABLE OWNER .TABLE_NAME MOVE LOB(COLUMN_NAME) STORE AS TABLESPACE;

SecureFile LOBs

SecureFiles, veritabanı avantajlarını koruyarak, geleneksel dosya sistemlerine kıyasla dosya veya yapılandırılmamış veriler için yüksek performans sunmak için özel olarak tasarlanmış Oracle Database 11g'de sunulan yeni bir özelliktir. SecureFiles, yarı yapılandırılmış ve yapılandırılmamış içeriği verimli bir şekilde depolayabilmek için tasarlanmıştır. SecureFiles, büyük nesneler (LOB'lar) için ANSI standardının üst kümesi olarak tasarlanmıştır ve eski LOB'lardan kolayca geçiş imkanı sunar. Uygulamalar, daha fazla alan tasarrufu ve daha hızlı performans için sıkıştırma ve tekilleştirme gibi akıllı depolama özellikleri ve ek güvenlik için endüstri standardı şifrelemeyi şeffaf bir şekilde kullanabilir. SecureFiles tamamen yeni disk biçimleri, ağ protokolü, alan yönetimi, yineleme ve geri alma biçimleri, tampon ön belleği ve akıllı G/Ç alt sistemi özelliklerine sahip yeni bir mimaridir. SecureFiles ile kurumlar artık veritabanındaki tüm ilişkisel verileri ve ilişkili dosya verilerini tek bir güvenlik/denetim modeli, birleşik bir yedekleme ve kurtarma işlemi ve tüm bilgiler arasında kesintisiz arama yaparak yönetebilir.

SecureFiles için alan yönetimi, dosya verilerini depolamak için optimize edilmiştir. Akıllı bir alan yöneticisi, boş alanın otomatik olarak geri kazanılmasıyla büyük, bitişik disk bloklarının hızlı tahsisi ve etkin silme sağlar. Gerekli alan, aynı zamanda, son talebin buluşsal yöntemlerine dayanarak önceden ayrılmıştır ve bu ön plandaki performansı etkilememektedir. SecureFiles LOB'larda alan yönetimi kendinden uyarlanabilir olacak şekilde tasarlanmıştır ve verimli bellek ve alan tahsisi için yeni bellek içi istatistikleri kullanılır. SecureFiles segmentleri, ASSM ile yönetilen tablo alanlarını gerektirir.

BasicFiles LOB'lardan farklı olarak SecureFiles, diskte bitişik alan ayırmayı en üst düzeye çıkarmak ve parçalanmayı (fragmentation) azaltmak için dinamik bir CHUNK (bir veya daha fazla veritabanı bloğu) boyutu ayarı kullanır. Kullanıcı

tarafından belirlenen CHUNK değeri, SecureFile'ın boyutu, optimal CHUNK boyutunu belirlemek için segmentteki boş alan gibi faktörlerle birlikte bir öneri olarak kullanılır. Sabit CHUNK boyutu ayarı, çok büyükse dahili parçalanmaya ve çok küçükse yazma performansına kötüleşir. Dinamik CHUNK boyutu ayarı ile, geniş disk bölgeleri tek bir işlemle tahsis edilebilir ve ayrılabilir.

SecureFiles, istemci ve sunucu arasında yüksek hızlı veri aktarımı sağlayan yeni bir istemci/sunucu ağ katmanına sahiptir. Yeni bir protokol, herhangi bir geçici aşamaya ihtiyaç duymadan doğrudan ağ soketinden toplu veri okuma ve yazma imkanı sağlar. Bu, önemli ölçüde daha yüksek okuma/yazma performansı sağlar.

BasicFile LOB'lar, segment başına B+ ağaç indeks mimarisini kullanır. Bu LOB indeksi hem veri erişiminde hem de boşaltılan veya silinen blokları yönetmek için kullanılır. Bu durum, eşzamanlı ortamlarda çekişme ve önemli performans düşüşüne neden olur. BasicFile LOB'ların aksine, SecureFiles bu tür meta verileri yönetmek için LOB indeksi kullanmaz. Bunun yerine, LOB bölümündeki veri bloklarıyla birlikte bulunan özel meta veri bloklarını kullanırlar. Bu tasarım, BasicFile LOB'larında LOB indeksi çekişmesine neden olan sıkıntıları ortadan kaldırır ve özellikle gerçek dünya ekleme-silme-geri alma durumlarında performansı büyük ölçüde geliştirir.

SecureFiles, kendini ayarlayan ve akıllı alan ve bellek yönetim algoritmalarına sahiptir ve dolayısıyla kullanıcı tarafından ayarlanan daha az sayıda parametre gerektirir. Özellikle, FREEPOOLS, FREELISTS, FREELIST GROUPS, PCTVERSION parametreleri SecureFiles tarafından dikkate alınmadığı için bu parametreleri ayarlanmasına gerek kalmaz. Bu parametreler çevrimiçi olarak değiştirilemediği için veritabanlarını durdurmaya gerek kalmaz. SecureFiles, alan yönetimi algoritmalarını kendi kendine ayarlamak için dahili istatistikleri korur ve çeşitli iş yükleri altında yüksek performans ve ölçeklenebilirlik sağlar.

SecureFile tipinde LOB oluşturmak için STORE AS SECUREFILE ifadesi kullanılır. Ayrıca sistem parametrelerinden olan DB_SECUREFILE uygun bir değere (IGNORE veya NEVER olmamalı) ayarlanmalıdır. Aşağıdaki komut ile bütün oluşturulan LOB'ların otomatik olarak SecureFile olması sağlanabilir.

ALTER SYSTEM SET db_securefile = 'ALWAYS'

SecuriFile DEDUPLICATE özniteliği ile oluşturulursa veri tekilleştirme özelliği aktif edilmiş olur. Bu özellik, SecureFiles verisinin birden çok kopyasını

ortadan kaldırır ve uygulamalara tamamen şeffaftır. Veritabanı birden çok özdeş SecureFiles verisini otomatik olarak algılar ve yalnızca bir kopyayı depolar, böylece depolama alanından tasarruf sağlar. Çoğaltma, depolama yönetimini basitleştirmekle kalmaz, özellikle de kopyalama işlemleri için önemli ölçüde daha iyi bir performansa neden olur.

SecuriFile COMPRESS özniteliği ile oluşturulursa veri sıkıştırma özelliği aktif edilmiş olur. Veritabanı, SecureFile verisinin sıkıştırılabilir olup olmadığını otomatik olarak algılar ve endüstri standardı sıkıştırma algoritmaları kullanarak sıkıştırır. Sıkıştırma herhangi bir tasarruf sağlamazsa veya veriler zaten sıkıştırılmışsa, SecureFiles bu LOB'lar için sıkıştırmayı otomatik olarak kapatacaktır. Veri sıkıştırması sadece depolama alanındaki önemli tasarruflara neden olmakla kalmaz aynı zamanda G/Ç, tampon önbellek gereksinimleri, tekrar üretme ve şifreleme yükünü azaltarak performansı geliştirir. Sıkıştırma, sunucu tarafında yapılır ve SecureFile verilerine rastgele okuma ve yazma olanağı tanır. SecureFile çeşitli derecelerde sıkıştırma sağlar: orta ve yüksek, depolama tasarrufu ve gecikme arasındaki dengeyi temsil eder. *[9]*

Kurum veritabanlarından DBA_LOBS.SECUREFILE kontrol edildiğinde SecureFile LOB kullanımına rastlanmamıştır. Bu bölümde anlatılan avantajlarından dolayı var olan bütün LOB'lar STORE AS SECURE parametresi kullanılarak oluşturulan SecureFile LOB alanlarına taşınmalıdır.

2.4.4. Görünümler

Görünüm, bir veya daha fazla tablodaki mantıksal bir gösterimdir. Özünde, bir görünüm, depolanmış bir sorgudur. Bir görünüm tablolardan veya diğer görünümlerden türer. Bir görünüm üzerinde gerçekleştirilen tüm işlemler, türedikleri tabloları da etkiler. Tabloların kullanıldığı çoğu yerde görünümler kullanılabilir. Görünümler, verilerin sunumunu farklı kullanıcı türleri için uyarlanmasına olanak tanır. Görünümler genellikle şunlar için kullanılır:

- ✓ Bir tablonun önceden belirlenmiş bir dizi satırına veya sütunlarına erişim kısıtlanarak ilave tablo güvenliği sağlamak için
- ✓ Data karmaşıklığını saklamak için. Örneğin, tek bir görünüm, birden çok tabloda ilgili sütunların veya satırların bir araya geldiği bir birleşimle tanımlanabilir. Bununla birlikte, görünüm bu bilgilerin aslında birkaç tablodan kaynaklandığını gizler. Sorgu, tablo bilgileri ile kapsamlı

hesaplamalar da yapabilir. Böylece, kullanıcılar nasıl yapacaklarını bilmeden veya hesaplamaları yapmadan görünümleri sorgulayabilirler.

- ✓ Verileri ana tablodan farklı bir perspektiften sunabilmek için. Örneğin, görünümün sütunları, ana tablodan bağımsız olarak adlandırılabilir.
- ✓ Ana tablo tanımı değişikliklerinden uygulamayı izole edebilmek için.

Tablonun aksine, görünümlere depolama alanı tahsis edilmez ve görünümler veri içermezler. Daha ziyade, bir görünüm, görünüm tarafından başvurulan ana tablodan veri çıkaran veya türeten bir sorgu ile tanımlanır. Bir görünümün tanımı veri sözlüğünde saklanır. *[1]*

Bir görünümün basit veya karmaşık olarak sınıflandırılması, DML ifadelerinin ona karşı yürütülüp yürütülmeyeceği ile ilgilidir: Basit görünümler genellikle DML ifadelerini kabul edebilirken; karmaşık görünümler kabul edemez. İki görünüm arasındaki genel ayrım şöyledir:

- Basit görünüm bir ayrıntı tablosundaki veriyi çeker, hiçbir fonksiyon kullanmaz, toplama ve gruplama işlemleri de yapmaz.
- Karmaşık görünüm birden fazla tabloları birleştirirken fonksiyon, gruplama, toplama gibi işlemler yapar.

Görünümler SQL performans artırmak için tasarlanmamıştır. Görünümlerin yukarda sayılan avantajları olmasına rağmen dikkat edilmezse aşağıda açıklanan durumlarda düşük performansa neden olur.

- ✓ İhtiyaç duyulan verileri doğrudan ilgili tablolardan veya sadece bu ihtiyaç için oluşturulmuş basit görünümlerden almak yerine, bu ihtiyacın küçük bir alt kümesi kaldığı ihtiyaç dışındaki çok sayıda tablonun birleştirilerek oluşturulan karmaşık görünümlerden almak sisteme performans problemine neden olacak gereksiz bir yük bindirir.
- ✓ Kompleks WHERE yan tümceleri ile görünümlere karşı yapılan sorgular, görünümün içine yerleştirilen tüm ayar ipuçlarını geçersiz kılar ve bu da, uygun olmayan yürütme planlarına neden olur.
- ✓ Bir görünümü sorgularken bir tablo gibi düşünülüp, genelde performans artışı sağlayan SQL ipuçları (hints) kullanılırsa sıklıkla yürütme planlarının en düşük seviyede sonuçlanmasına neden olur. Bu yüzden görünüm kullanırken SQL ipucu kullanılmaktan kaçınılmalıdır.
- ✓ İç içe kümelenmiş karmaşık görünümler kullanmak performans sıkıntısı yaratır.

2.4.5. Somutlaştırılmış Görünümler

Somutlaştırılmış görünümler, şema nesneleri olarak önceden saklanmış veya somutlaştırılmış sorgu sonuçlarıdır. Sorgunun FROM yan tümcesi, tabloları, görünümleri ve somutlaştırılmış görünümleri adlandırabilir. Verileri özetlemek, hesaplamak, çoğaltmak ve dağıtmak için somutlaştırılmış görünümler kullanılır. Aşağıdakiler gibi çeşitli durumlarda uygundurlar:

- Veri ambarlarında toplamlar ve ortalamalar gibi toplam fonksiyonlarından elde edilen verileri hesaplamak ve depolamak için somutlaştırılmış görünümleri kullanılabilir. Özet, birleştirmeleri ve toplama işlemlerini önceden hesaplayarak ve sonuçları bir tabloda depolayarak sorgu süresini azaltan toplu bir görünümdür. Gerçekleştirilen görünümler özetlere eşdeğerdir. Toplam fonksiyonları olmayan çoklu tablo birleştirmelerinde de somutlaştırılmış görünümler kullanılabilir.
- Mobil bilgi işlem ortamlarında, merkezi sunuculardan periyodik yenilemeler ve istemcilerin güncellemelerinin merkezi sunuculara yayılmasıyla merkezi sunuculardan mobil istemcilere bir veri alt kümesi indirmek için somutlaştırılmış görünümler kullanılabilir.

Somutlaştırılmış görünümler, somutlaştırılmamış görünümlerin aksine gerçek veri içerdikleri için diskte alan kullanırlar. Ana tablolarındaki veriler değiştiğinde yenilenebilirler. Somutlaştırılmamış görünümlere göre oldukça performanslıdır. Kullanımı tablo ve görünümler gibi olduğu için uygulamalar tarafından değişiklik gerektirmez. Yenilenme tipine bağlı olarak somutlaştırılmış görünümler DML ifadeleri ile güncellenebilir. CREATE MATERIALIZED VIEW komutu ile oluşturulur. *[1]*

Kurum veritabanında ALL_MVIEWS görünümü sorgulanarak somutlaştırılmış görünüm kullanılmadığı tespit edilmiştir. Normal bir görünüm olan IZINLISANS şemasındaki V_IZINLISANS'ın CREATE MATERIALIZED VIEW anahtar kelimesi kullanılarak somutlaştırılmış görünüm oluşturulup CPU maliyetleri aşağıdaki komutlarla tespit edilerek Tablo 2.4'te karşılaştırması yapılmıştır.

```
Sql>explain plan for select * from IZINLISANS.V_IZINLISANS
Sql>select * from table(dbms_xplan.display)
Sql>explain plan for select * from IZINLISANS.MV_IZINLISANS
Sql>select * from table(dbms_xplan.display)
```

Tablo 2.4 : Görünüm ve Somutlaştırılmış Görünüm CPU Maliyet Karşılaştırması

Object	Rows	Bytes	Cost(%CPU)	Time
V_IZINLISANS	43870	11M	3266	00:00:40
MV_IZINLISANS	43870	11M	251	00:00:04

Tablo 2.4'te de görüldüğü üzere somutlaştırılmış görünüm kullanımı normal görünüme göre gerek CPU maliyeti bakımından olsun gerekse zaman bakımından olsun performans kazandırmıştır. Fakat bütün görünümler somutlaştırılmış görünüme dönüştürülmemelidir. Sadece anlık güncel veriye ihtiyaç duyulmayan ve düşük performanslı görünümlerde, kurum rapor alt yapısını veri ambarına geçirene kadar kullanılmalıdır.

2.4.6. PL/SQL Alt Programlar

Bir PL/SQL alt programı, kendisini çağıran kullanıcıya, girdi ve çıktı değerleri olabilecek parametreleri sağlamasına izin veren, adlandırılmış bir PL/SQL bloğudur. Bir altprogram belirli bir sorunu çözer veya ilgili görevleri yerine getirir ve modüler, bakıma açık veritabanı uygulamaları için bir yapı taşı olarak hizmet eder.

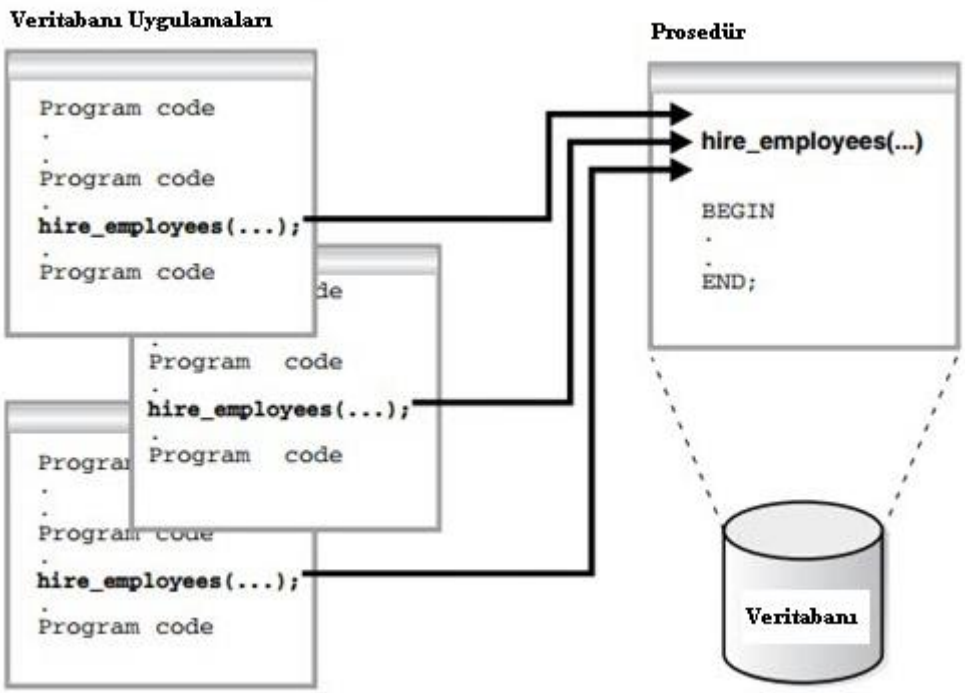
Bir altprogram, bir prosedür veya bir fonksiyondur. Prosedürler ve fonksiyonlar aynıdır, ancak fonksiyonlar çağıran kişiye her zaman tek bir değer döndürürken, prosedürler bunu gerektirmez. Prosedür kullanmanın avantajları aşağıda sıralanmıştır.

- ✓ Bir uygulamanın bir ağ üzerinden göndermesi gereken veri miktarı, SQL deyimlerini ayrı ayrı göndermekle veritabanına bir bütün PL/SQL bloğunun metnini göndermekle karşılaştırıldığında daha azdır, çünkü veriler sadece bir kez gönderilir ve bundan sonra çağrıldıkça kullanılır.
- ✓ Bir prosedürün derlenmiş biçimi veritabanında hazır bulunur, bu nedenle çalışma zamanında derleme gerekmez.
- ✓ Prosedür, SGA'nın paylaşılan havuzunda mevcutsa, diskten alınması gerekmez ve derhal çalışmaya başlayabilir.
- ✓ Prosedürler veritabanının paylaşılan bellek yeteneklerinden yararlandığından, birden fazla kullanıcı tarafından yürütülmek üzere yordamın tek bir kopyasını belleğe yüklemelidir. Kullanıcılar arasında kod paylaşımı, uygulamalar için veritabanı bellek gereksinimlerini önemli ölçüde azaltır. Prosedürler, belki de

kod tekrar kullanımını gerçekleştirmenin en iyi yoludur. Veritabanına bağlanan herhangi bir dilde yazılmış herhangi bir istemci uygulaması prosedürü çağırdığından, tüm ortamlarda maksimum kod yeniden kullanımını sağlarlar.

- ✓ Prosedürler, uygulamanın bütünlüğünü ve tutarlılığını geliştirir. Ortak bir prosedür grubu çevresinde uygulamalar geliştirerek, kodlama hataları olasılığı azaltılabilir. Örneğin, bir prosedürün doğru bir sonuç döndürdüğü test edilerek garantilendikten sonra, yeniden test etmeden tekrar kullanılabilir. Prosedür tarafından başvurulan veri yapıları değiştirilirse, yalnızca prosedürü yeniden derlemek gerekir. Prosedürü çağıran uygulamaların değişiklik yapması gerekmez.
- ✓ Prosedürler, veri güvenliğini sağlamaya yardımcı olabilir. Bir prosedürün çalıştırılabilme hakları, mevcut kullanıcıyla değil, sahibinin imtiyazıyla belirlenir. Böylece, sadece prosedürün sahibinin izin verdiği kullanıcılar prosedürün ürettiği verilere ulaşabilir.

Şekil 2.4 : Uygulamalardan Prosedür çağırmak



[1]

Bu bölümde özellikleri ve avantajları anlatılan çok güçlü bir performans nesnesi olan prosedürlerin kurum veritabanlarında çok az kullanıldığı tespit edilmiştir. Özellikle çoklu DML işlemleri uygulama yerine prosedürlerle CREATE PROCEDURE anahtar kelimesi kullanarak yapılmalıdır.

2.5. SQL İyileştirmeleri

Veritabanı ile etkileşim kurmanın, veriyi geri almanın, veriyi değiştirmenin, veritabanını yönetmenin yolu SQL'dir. Veritabanı, arka planda yaptığı tüm işleri gerçekleştirmek için SQL kullanır. SQL performansı bu nedenle veritabanı performansının anahtarıdır; Tüm veritabanı performans problemleri kendilerini kaynak çekişmesi olarak ifade etseler bile aslında SQL performans problemleridir.

Bir SQL ifadesinin etkinliği, çıktı üretirken kullanılan CPU döngüleri gibi hesaplama kaynakları miktarı ile ölçülür. Kaynakların tüketimini azaltmak SQL iyileştirmenin hedefidir. Geçen zaman, SQL ifadesinin verimliliğinin iyi bir ölçütü değildir, çünkü her zaman tüketilen kaynak miktarı ile orantılı değildir. Örneğin, CPU döngüleri ve disk G/Ç'i için oluşan çekişmeler çalıştırma gecikmelerine neden olur. Mantıksal okuma işlemlerinin sayısı, bilgi kaynaklarının tüketimini ölçmek için daha iyi bir yoldur, çünkü tüketilen kaynakların miktarı ile doğru orantılıdır - mantıksal okuma işlemlerinin sayısı ne kadar azsa CPU tüketimi de o kadar azdır. İyi bir veritabanı performansı açısından etkin olmayan SQL ifadelerinin iyileştirilmesi gerekir.

SQL iyileştirmesi, belirli, ölçülebilir ve erişilebilir hedefleri karşılamak için SQL ifadesinin performansını iyileştirmenin yinelemeli bir işlemidir. SQL iyileştirmesi, geliştirilmiş uygulamalarda sorunları gidermek anlamına gelir. Bir SQL deyimi, önceden belirlenmiş ve ölçülebilir bir standarda göre başarısız olduğunda bir problem haline gelir. Problemli SQL ifadeleri belirlendikten sonra, SQL iyileştirmelerinin amacı kullanıcı SQL bildirimi çalıştırdığında yanıt aldığı süreyi azaltmak veya bir SQL ifadesi ile erişilen tüm satırları işlemek için gereken en düşük miktarda kaynağın kullanılması anlamına gelen iş hacmini iyileştirmektir. Bu birkaç farklı şekilde yapılabilir:

- Yükü azaltarak : SQL iyileştirme genelde aynı iş yükünü işlemek için daha etkin yöntemler bulmayı içerir. Kaynak tüketimini azaltmak için işlevselliği değiştirmeden ifadenin çalışma planını değiştirmek mümkündür. Örneğin sık

kullanılan bir sorgunun tablodaki küçük verilere erişmesi gerekiyorsa, veritabanı bir indeks kullanarak daha verimli bir şekilde çalıştırabilir. Böyle bir indeks oluşturarak, kullanılan kaynakların miktarını azaltılabilir.

- Yükü dengeleyerek : Sistemler, gerçek kullanıcıların sisteme bağlandığı gündüz vakti yoğun kullanımdayken, geceleri düşük kullanıma sahip olma eğilimindedir. Kritik olmayan raporlar ve yedekleme gibi toplu işler gün içindeki işlerin yoğunluğunu artırmamak için gece çalışacak şekilde ayarlanırsa, veritabanı, gün içindeki daha kritik programlar için kaynakları boşa çıkarır. Ayrıca kurum veritabanlarında raporların veri ambarı sistemlerine aktarılarak alınmalıdır.
- Yükü paralelleştirerek : Büyük miktarda veriye erişen sorgular (tipik veri ambar sorguları) genellikle paralel olarak çalışabilir. Düşük eşzamanlılıkla çalışan bir veri ambarındaki yanıt süresini azaltmak için paralellik son derece yararlıdır. Bununla birlikte, yüksek eşzamanlılık eğiliminde olan OLTP ortamları için, paralellik, programın genel kaynak kullanımını artırarak diğer kullanıcıları olumsuz yönde etkileyebilir.

Yukarıdaki adımları gerçekleştirebilmek veritabanı mimarisi bilgisi, PL/SQL bilgisi ve SQL iyileştirme araçlarını kullanabilme becerisi gerektirir.

2.5.1. Yoğun Kaynak Tüketen ve Hatalı Çalışan SQL ifadelerinin Belirlenmesi

Bu bölüm, yüksek kaynak tüketen SQL deyimlerini tespit etme adımlarını ve araçlarını açıklar. Yüksek kaynak tüketen SQL ifadesi, veritabanının performansını olumsuz etkileyen, az çalışan, kaynak yoğun SQL ifadeleridir. Kaynak yoğun SQL ifadelerini tanımlamanın ilk adımı, sorunu sınıflandırmaktır. Problem sadece belirli bir uygulamanın belirli bir bölümünde mi var yoksa tüm veritabanı genelinde mi olduğu araştırılmalıdır. Çözümün kapsamı sorunun kapsamına uygun olmalıdır. Örneğin, paylaşılan havuz çok küçükse, bu imleçlerin hızla yaşlanmasına neden olur, ki bu da pek çok sabit ayrıştırmaya neden olur. Paylaşılan havuz boyutunu artırmak için bir başlatma parametresi kullanmak, sorunu veritabanı düzeyinde düzeltir ve tüm oturumların performansını artırır. Bununla birlikte, tek bir SQL deyimi yararlı bir indeks kullanmıyorsa, tüm veritabanının optimize edicinin başlatma parametrelerinden birinin değiştirilmesi genel performansa zarar verebilir.

Enterprise Manager, kaynak yoğun SQL ifadelerini belirlemek, çalışma planları oluşturmak ve SQL performansını değerlendirmek için araçlar sağlar. SQL ifadesi dinamik olarak oluşturulduğu için belirlenemiyorsa, çalıştırılan ifadeyi izlemek için SQL_TRACE kullanılabilir. Çıktısını bir dosyaya atıp incelemek içinde TKPROF kullanılabilir. TKPROF çıktı dosyası en çok kaynak tüketen SQL'ler sıralı olarak listeler.

Bir uygulama genelinde veya veritabanı genelinde performans problemleri varsa, G/Ç problemleri için V$FILESTAT, sistem geneli için V$SYSSTAT, SQL'ler için V$SQLAREA, VSQL, VSQLSTATS, V$SQLTEXT, V$SQL_PLAN ve V$SQL_PLAN_STATISTICS performans görünümleri incelenmelidir. Yoğun kaynak tüketen SQL ifadelerini sorgulamanın iyi bir yolu V$SQLSTATS görünümünü sorgulamaktır. Bu görünüm paylaşılan havuzdaki tüm SQL ifadeleri için kaynak kullanımı bilgilerini içerir. Bu görünümün BUFFER_GETS kolonu en yüksek CPU kullananları, DISK_READS kolonu en yüksek G/Ç kullananları ve SORTS kolonu da en çok sıralama yapan SQL ifadelerini listeler.

Hangi SQL ifadelerinin en yüksek yük oluşturduğunu belirlemek için başka bir yöntemde, SQL deyimi tarafından kullanılan kaynakları, dönem içinde kullanılan o kaynağın toplam miktarı ile oranlayıp karşılaştırmaktır. Bu değerleri manuel hesaplamaya gerek yoktur. Bu iş için AWR aracı kullanılabilir.

Hatalı çalışan SQL deyimlerini yakalamak için aşağıdaki tetikleyici oluşturulmuştur ve en çok alınan hatalar Tablo 2.5'te listelenmiştir.

```
CREATE OR REPLACE TRIGGER VTYS.CATCH_ERRORS
   after servererror on database
declare
   sql_text ora_name_list_t;
   msg_ varchar2(4000) := null;
   stmt_    varchar2(32000) := null;
   host_    varchar2(512)  := null;
   suser_   varchar2(512)  := null;
begin
SELECT SYS_CONTEXT ('USERENV', 'HOST') into host_   FROM DUAL;
  for depth in 1 .. ora_server_error_depth loop
    msg_ := msg_ || ora_server_error_msg(depth);
```

```
  end loop;
if    ora_sql_txt(sql_text) > 0   then
  for i in 1 .. ora_sql_txt(sql_text) loop stmt_ := stmt_ || sql_text(i);
  end loop;
 end if;
IF ora_login_user != 'SYS' THEN
      insert into
        vtys.caught_errors (dt, username ,msg , host, stmt1  )
      values (sysdate, ora_login_user,msg_, host_, substr(stmt_, 1,4000) );
        commit;
 END IF;
 end;
```

Tablo 2.5 : Kurum Veritabanında En çok alınan SQL Hataları

Şema	**Sunucu**	**Hata Kodu**	**Hata**	**Hata Sayısı**
CEBECI	DNMSRV06S01	ORA-0923	FROM keyword not found	163621
MUBIS	MUBISWEB06S01	ORA-0942	table or view does not exist	9265
KL_ATLAS	CSB\CBSGIS	ORA-2391	exceeded simultaneous SESSIONS_PER_USER limit	33482
KL_PLAN	CSB\CBSGIS	ORA-0942	table or view does not exist	6724
MVAS	CSB\CBSGIS	ORA-4043	object does not exist	6183
KOOP	CSB\EKOOPAPP06S01	ORA-0001	unique constraint (KOOP.SYS_C0013649) violated	5476
CED	metaformhasan	ORA-0942	table or view does not exist	1857
FBS_ANALIZ	IIS APPPOOL\WS-ECBS06S01	ORA-0942	tablo veya görüntü mevcut degil	930

Hataların en çok olmayan veya yetki dışındaki tablolara ve görünümlere erişmeye çalışma, izin verilenden daha fazla oturum açmaya çalışma, tablo kısıtlamalarını ihmal etme ve sentaks hatalarıdır. Hatalı SQL deyimleri veritabanlarını gereksiz yere meşgul ederek üzerindeki yükü artırarak performans düşüşüne neden olmaktadır. Bu hataların önüne geçilmesi için uygulama geliştiricilerin uygulamalarda gerekli düzenlemeyi yapmaları gerekmektedir.

2.5.2 Etkin ve Performanslı SQL ifadeleri Oluşturma Önerileri

Bu bölümde yoğun kaynak tüketen sorguların ve yeni sorgular oluştururken sistem kaynaklarını verimli ve etkin kullanmak adına dikkat edilmesi gereken hususlar açıklanacaktır.

- ✓ Optimize edicinin en iyi çalıştırma planını oluşturabilmesi güncel tablo ve indeks istatistiklerine ihtiyaç duyar. Sorguda kullanılacak tabloların istatistiklerin güncelliği kontrol edilmelidir. Bunu kontrol etmenin bir yolu da tabloların gerçek satır sayısı ile DBA_TABLES.NUM_ROWS değerini karşılaştırmaktır. İstatistikler eski ise istatistik toplanması sağlanmalıdır.
- ✓ OLTP ortamında SQL ifadesi yazarken veya iyileştirirken, sonraki adıma daha az satır göndermek için en seçici filtreyi sağlayan koşul ilk sıraya yazılmalıdır.
- ✓ İhtiyaç duyulmayan sütunlar, tablolar ve görünümler sorgudan çıkarılmalıdır.
- ✓ Tablolar küçük bile olsa Kartezyen birleştirmelerden kaçınılmalıdır.
- ✓ Satır sayısı çok olan büyük tabloların WHERE cümleciğinde kullanılan koşulların uygun indeksi olup olmadığı kontrol edilmeli yoksa uygun tipte indeks oluşturulmalıdır.
- ✓ Mümkün olduğunca eşitlik içeren birleştirmeler kullanılmalıdır.
- ✓ WHERE cümleciğinde kullanılan sütunları, fonksiyon tabanlı indeks oluşturulmamışsa fonksiyon kullanarak dönüştürmekten kaçınılmalıdır.
- ✓ Farklı tipte olan karşılaştırmalar veritabanı kendi içinde gizli dönüşüm yapar. Örneğin NUMBER ve VARCHAR2 tipinde iki sütun karşılaştırılıyorsa veritabanı varsayılan olarak VARCHAR2 tipindeki sütunu TO_NUMBER fonksiyonu kullanarak NUMBER tipine dönüştürüp öyle karşılaştırma yapar. Eğer VARCHAR2 tipindeki sütunda indeks oluşturulmuşsa bu indeksin kullanılabilmesi için NUMBER tipli sütunun TO_CHAR fonksiyonu ile açık olarak dönüştürülmesi gerekir.
- ✓ WHERE cümleciğinde literal olarak kullanılan NUMBER tipinde bir sütun varsa, bu sütuna tanımlanmış indeksi kullanabilmek için TO_NUMBER fonksiyonu kullanılmalıdır.
- ✓ Parametrik genel bir SQL ifadesi yerine ihtiyaca dönük küçük SQL ifadeleri yazılmalıdır.
- ✓ Bazen, belirli bir uygulamanın verisi hakkında optimize edici tarafından kullanılandan daha fazla bilgiye sahip olan uygulama tasarımcısı, bir SQL

deyimini yürütmenin daha etkili bir yolunu seçebilir. Optimize edici'ye ifadenin nasıl çalıştırılacağı konusunda talimat vermek için SQL ifadelerindeki ipuçlarını (hints) kullanılabilir.

- ✓ Birleştirme sırası performans üzerinde önemli bir etkiye sahiptir. Bu yüzden tam tablo taraması yapan koşullardan önce indeks kullanan koşullar yazılmalıdır.
- ✓ Karmaşık görünümler birbirleriyle birleştirilmemelidir. Veya iç içe karmaşık görünümler kullanılarak yeni görünümler elde edilmemelidir.
- ✓ Bir görünümden sorgulama verilerin döndürülmesi için görünümden gelen tüm tablolara erişilmesini gerektirir. Bir görünümü kullanmadan önce veriyi döndürmek için görünümdeki tüm tablolara erişilmesinin gerekli olup olmadığı belirlenmeli ve gerekli değilse bu görünüm kullanılmamalıdır. Bunun yerine sadece gerekli tablolardan oluşturulmuş görünüm kullanılmalıdır. Performansı artırmak için amaç minimum tablo kullanmaktır.
- ✓ Ara sonuçların aşamalı olarak geçici tablolarda tutulması, takip eden sorgularda da tekrardan kullanırsa uygulama performansını artırabilir. Ara sonucun ikinci kullanımında karmaşık bir ifadeyle veriyi birden çok kez almamanın faydası, onu somutlaştırma maliyetinden daha iyidir.
- ✓ Uzun ve karmaşık sorgularda somutlaştırılmış görünümler kullanmak faydalıdır.
- ✓ İndeksleri belli aralıklarla sistemin yoğun kullanılmadığı zamanlar yeniden oluşturmanın performans üzerinde olumlu etkisi olur.
- ✓ DML işlemlerini hızlandırmak için kullanılmayan indeksler kaldırılmalıdır.
- ✓ WHERE cümlecilerinde seçiciliği yüksek kolonlara indeks oluşturulmalıdır.
- ✓ Çok fazla tetikleyici (trigger) ve kısıtlayıcı (constraint) kullanmak sistem kaynaklarını önemli ölçüde tükettiği için performansı olumsuz etkiler. Bu durumda tetikleyiciler tekrar gözden geçirilmeli ve artık gerek yoksa devre dışı bırakılmalıdır.
- ✓ Cevap verme süresi kritik uygulamalarda GROUP BY ifadesinden kaçınılmalı.
- ✓ Çok büyük tablolarda bölümleme kullanılmalıdır.
- ✓ Bir tabloda çeşitli where koşullarında COUNT, SUM, AVG gibi tüm tabloyu tarama gerektiren durumlarda, her koşul için ayrı ayrı tablonun tamamını taramaktansa where koşulunu birleştirme için verilere filtre uygulayan bir CASE ifadesine taşıyarak tek bir tarama ile performans önemli ölçüde artar.

- ✓ INSERT, UPDATE, DELETE gibi komutlardan sonra veriye ihtiyaç varsa RETURNING ifadesi kullanılmalıdır. Böylece veritabanına erişim sayısı azaltılarak ağ ve veritabanı üzerindeki yük azalmış olur.
- ✓ UNION ek bir sıralama işlemi gerektirdiği için UNION ALL aynı amacı karşılıyorsa UNION ALL kullanılmalı.
- ✓ Gerekmedikçe DISTINCT kullanılmamalıdır.
- ✓ Aynı ifadenin tekrar tekrar ayrıştırılmasını azaltmak için filtre koşullarında literal değerler yerine bind değişkenler kullanılmalıdır.
- ✓ Alt sorgudaki karşılaştırılan değerin sadece varlığı yokluğu kontrol ediliyorsa tam tablo taraması yapmamak için IN yerine EXISTS kullanılmalıdır. *[3]*

2.5.3 SQL İyileştirme Araçları

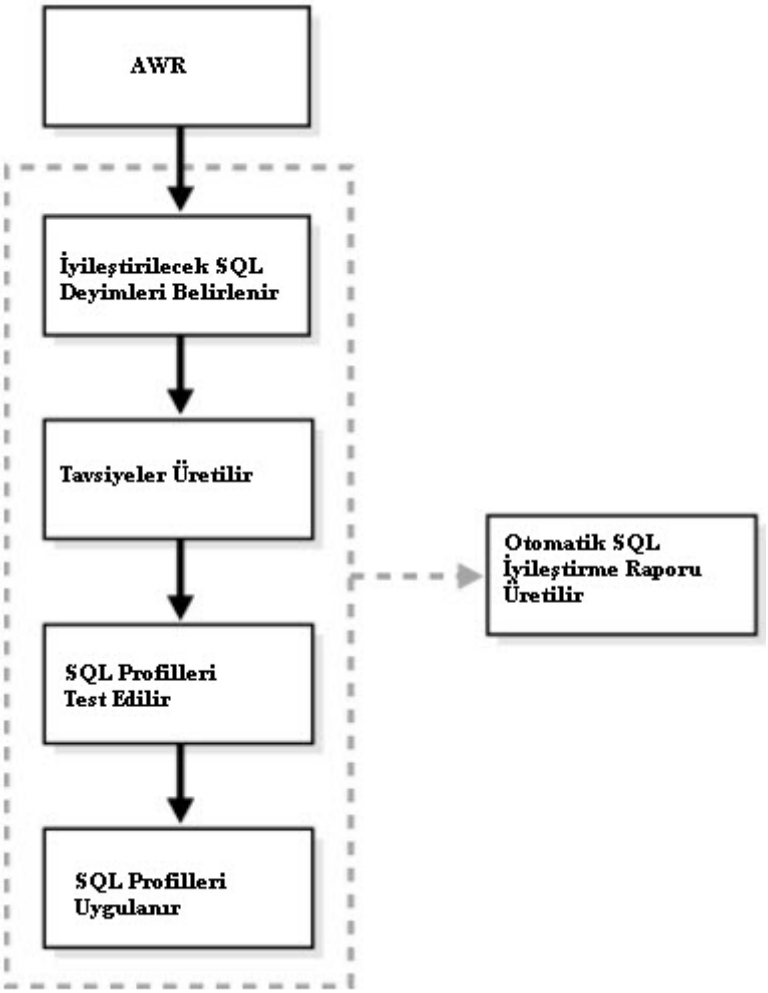
Bu bölümde yüksek kaynak tüketen SQL deyimlerini tespit eden ve daha verimli çalışabilmesi için öneriler sunabilen veritabanı araçları açıklanacaktır.

2.5.3.1 SQL Tunning Advisor

SQL Tunning Advisor yüksek kaynak tüketen SQL ifadelerini daha verimli çalışabilmesi için öneriler sunar. SQL Tuning Advisor'ın çıktısı, her öneri ve bunun beklenen fayda için bir gerekçe ile birlikte bir tavsiye veya tavsiyeler şeklinde bulunmaktadır. Tavsiye, nesneler hakkındaki istatistiklerin toplanmasına, yeni indekslerin oluşturulmasına, SQL deyiminin yeniden yapılandırılmasına veya bir SQL profilinin oluşturulmasına ilişkindir. SQL ifadelerinin iyileştirilmesi için oluşturulan öneri kabul edildiği taktirde otomatik uygulanır. SQL Tuning Advisor seçici olarak sorunlu olarak tanımlanmış tek bir SQL ifadesi grubunda da uygulanabilir.

DBMS_AUTO_TASK_ADMIN.ENABLE (client_name => ‘sql tuninng advisor’) prosedürü ile bu aracın otomatik çalıştırılması sağlanabilir. SQL Tuning Advisor yoğun kaynak tüketen SQL ifadelerini AWR üzerinden bulur. Otomatikleştirilmiş bu görev pencere bakımı şeklinde genelde geceleri çalışır. Başlangıç ve bitiş saati, sıklık ve haftanın günleri dahil bakım pencerelerinin özellikleri özelleştirilebilir.

Şekil 2.5: Otomatik SQL iyileştirme



http://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#CHDIBFGA

Şekil 2.5 ' te Otomatik SQL iyileştirme adımları gösterilmiştir. Bu adımlardan sonra oluşturulan raporlar içerik olarak aşağıdaki bölümleri içerir.

- Rapor için verilen girdiler, bakım sırasında ayarlanan SQL ifadelerinin sayısı ve oluşturulmuş SQL profillerinin sayısı da dahil olmak üzere otomatik SQL ayarlama görevinin üst düzey bir açıklamasını içeren genel bilgiler
- Bakım penceresi ve her bir SQL profilinin tahmini yararı veya SQL profili ile SQL deyimini yürüttükten sonra gerçek yürütme istatistikleri sırasında ayarlanan SQL ifadeleri özet bölümünde listelenir.
- SQL Tuning Advisor tarafından analiz edilen her SQL deyimi tarafından kullanılan eski ve yeni açıklama planlarının olduğu çalışma planı bölümü
- Karşılaşılan tüm hataları listeleyen hata bölümü

2.5.3.2 ADDM

ADDM, AWR'deki istatistikleri kullanarak veritabanı performans problemlerini proaktif olarak teşhis eden ve sorunlara çözüm önerisi sunabilen veritabanı içerisine yerleştirilmiş bir araçtır. Çoğu durumda, ADDM çözüm önerir ve beklenen performans avantajlarını ölçer. Örneğin, ADDM, donanım, veritabanı yapılandırması, veritabanı şeması veya uygulamalar üzerinde değişiklikler önerebilir. Bir öneri yapılırsa, ADDM zaman kazancını bildirir. Bir önlem olarak zaman kullanılması, sorunların veya önerilerin kıyaslanmasını sağlar. Potansiyel performans sorunlarının bildirilmesinin yanı sıra ADDM, sorun olmayan veritabanının alanlarını belgelemektedir. ADDM aşağıdaki avantajları sağlar.

- ✓ Saat başı otomatik performans tanılama raporu üretebilme
- ✓ Sorunlu etkilerin zamana dayalı olarak ölçülmesi ve öneri sunma
- ✓ Problemlerin görünen yüzeysel belirtileri değil, kök nedenin belirlenmesi
- ✓ Sorunların kök nedenlerini giderebilmek için öneri sunma
- ✓ Sistemin sorunsuz bölümlerinin tanımlanması
- ✓ Tüm bunları yaparken sisteme çok az yük bindirmesi

ADDM önerilerini uygularken bir sorunun düzelmesi başka veritabanın başka bir bölümünü sıkıntıya sokabilir. Bu yüzden ADDM önerileri yinelemeli bir süreçtir. Kabul edilebilir sistem performansına ulaşmak için çoklu ayar döngüsü gerekebilir.

ADDM veritabanının tamamını, sadece veritabanı olgusunu veya veritabanını kısmı olarak analiz edebilir. ADDM analizi, önce semptomları tanımlayan ve daha sonra performans problemlerinin temel nedenlerine ulaşmak için yukarıdan aşağıya inceleme suretiyle gerçekleştirilir. Analizin amacı, DB zamanı adı verilen tek bir iş hacmi metriğini azaltmaktır. DB süresi, veritabanı tarafından kullanıcı isteklerinin işlenmesinde harcanan toplam zamandır. DB süresi, V$SESS_TIME_MODEL ve V$SYS_TIME_MODEL görünümlerinde gösterilir. DB zamanını azaltarak, veritabanı aynı kaynakları kullanarak daha fazla kullanıcı isteğini destekleyebilir, böylece yapılan iş artar. ADDM tarafından rapor edilen sorunlar, sorumlu oldukları DB zaman miktarına göre sıralanmaktadır. DB zamanının önemli bir bölümünden sorumlu olmayan sistem alanları sorunlu olmayan alanlar olarak bildirilir. ADDM'nin dikkate aldığı sorun türleri arasında aşağıdakiler bulunur:

- ✓ CPU darboğazına neden olan etkenler

- ✓ SGA, PGA ve tampon önbellek gibi bellekleri ihtiyacı karşılayacak boyutta olup olmadığı
- ✓ G/Ç kapasitesinin durumu
- ✓ Yüksek kaynak tüketen SQL'lerin tespiti
- ✓ RAC'a özgü sorunlara neden olan bağlantı gecikmesinin kontrolü
- ✓ Uygulama ile veritabanı arasındaki bağlantı problemlerinin kontrolü
- ✓ Günlük dosyalarının yanlış boyutlandırılması, arşivleme sorunları, aşırı kontrol noktaları, optimal parametre ayarları gibi veritabanı yapılandırması sorunlarının kontrolü
- ✓ Eşzamanlılık sorunlarının kontrolü *[3]*

2.6 Daha iyi performans için Kaynak Yönetimi

Bir anabilgisayar ortamında, işletim sistemi kaynaklarını görevlere ayırmayı kendisi yapar. Ama işletim sistemleri uygun kaynak planlama yeteneklerine sahip olmayabilir. Veritabanı kaynak yöneticisi (Resource Manager), desteklenen tüm veritabanı platformlarına ana çerçeve tarzı kaynak yönetimi özelliklerini getirir; yani, DBA, belirli veritabanı kullanıcılarının asgari belirli bir düzeyde hizmet alacağını, veritabanındaki genel iş yükü ne olursa olsun garanti edebilmesi için bu aracı kullanır.

Unix veya Windows gibi işletim sistemleri, kaynakları farklı süreçlere atamak için çok basit bir algoritma olan round-robin zaman dilimleme algoritmasını kullanır. İşletim sistemine, Veritabanı olgusunu oluşturan arka plan süreçleri ile kullanıcı oturumlarını destekleyen birçok sunucu işlemi arasında hiçbir fark yoktur; işletim sistemi açısından bir süreç bir süreçtir. İşletim sistemi süreçlerin birbirine olan önceliğini bilmesinin yolu yoktur. Kötü yazılmış çok fazla kaynak tüketen bir SQL deyiminin veritabanında diğer herkesin çalışmasını güçleştirerek performans sorunlarına neden olabilir. Kaynak Yöneticisi, bazı kullanıcıların diğerlerinden daha fazla işlem kapasitesi almasını sağlamak için işletim sisteminin zaman uyumsuz algoritmasının ayarlanabileceği bir mekanizma sağlar ve herhangi bir sorgunun herkes için performansı yok etmesini önler. *[12]* Veritabanı kaynak yönetimi kullanmak aşağıdaki avantajları sağlar.

- ✓ İşlemcilerin yük bakımından tam kapasitesine ulaştığında sonradan gelen önceliği yüksek acil işler veritabanı tarafından yapılamaz. Bu durumu

önlemek için, her bir tüketici grubundaki oturumlara maksimum miktarda işlemci tahsis etmek için Veritabanı Kaynak Yöneticisi kullanılır. Bu özellik, yüksek öncelikli oturumların hemen çalışmasını ve düşük öncelikli oturumların CPU tüketimini düşürmesini sağlar.

- ✓ Tüketici grubunun kullanabileceği CPU yüzdesine kesin bir sınır koymak için Kaynak Yöneticisi yönergesinde MAX_UTILIZATION_LIMIT parametresi kullanılır. Bu özellik, düşük öncelikli oturumların CPU tüketimini kısıtlar ve bir tüketici grubundaki iş yükü için daha tutarlı performans sağlamaya yardımcı olabilir.
- ✓ Veritabanı Kaynak Yöneticisi, bir çağrı için maksimum yürütme süresini sınırlayarak veya uzun süredir çalışan bir sorguyu daha düşük öncelikli bir tüketici grubuna taşıyarak kaçak sorguların sisteme verdiği hasarı sınırlandırabilir.
- ✓ Bir tüketici grubunun tüm paralel sunucuları tekeline almasını önlemek için Resource Manager yönetmeliği PARALLEL_TARGET_PERCENTAGE kullanılır. Veritabanı, bu sınırın aşılmasına neden olursa paralel deyimleri sıraya koyar. *[3]*

Kaynak yönetimi aşağıdaki bileşenlerden oluşur.

2.6.1. Kaynak Tüketici Grubu

Bir kaynak tüketici grubu, işlem gereksinimlerine göre gruplandırılmış olan kullanıcı oturumlarının bir toplamıdır. Bir oturum oluşturulduğunda, önceden tanımlanmış tüketici grubu kurallarına dayalı olarak otomatik olarak bir tüketici grubuna eşlenir. DBA, bir oturumu farklı bir tüketici grubuna manuel olarak değiştirebilir. Benzer şekilde, bir uygulama oturumunu belirli bir tüketici grubuna geçiren bir PL/SQL paketi de çalıştırabilir. Veri sözlüğünde varsayılan olarak tanımlanmış değiştirilemez gruplar bulunur. Bunlar sistem kullanıcılarının atandığı SYS_GROUP grubu ve özellikle atama yapılmamış kullanıcıların içinde bulunduğu OTHER_GROUPS grubudur.

2.6.2 Kaynak Planları

Kaynak planı belirli bir türe sahiptir. En temel (ve en yaygın) plan türü CPU kaynaklarını tahsis etmektir, ancak başka kaynak ayırma yöntemleri de vardır. Birçok plan veritabanı içinde mevcut olabilir, ancak yalnızca bir plan herhangi bir

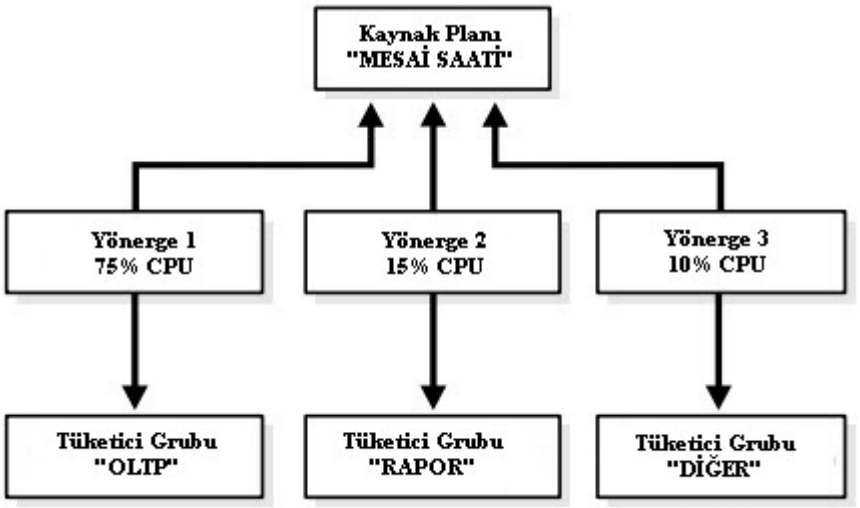
zamanda aktiftir. Bu plan, tümüne uygulanır: tüm oturumlar tarafından kontrol edilir. Kaynak planı tarafından aşağıdaki kaynaklar kontrol edilir.

- ✓ Bir gruptaki tüm oturumların toplam CPU kullanımı
- ✓ Bir gruptaki her oturum için mevcut paralellik derecesi
- ✓ Grup başına izin verilen etkin oturum sayısı
- ✓ Boş oturumları sonlandırmadan önce geçen süre
- ✓ Bir oturumda bir çağrının kesintisiz çalışabilme süresi

2.6.3 Kaynak Planı Yönergeleri

Kaynak Yöneticisi, şu anda etkin olan kaynak planına ait olan kaynak planı yönergelerinin kümesine göre tüketici gruplarına kaynak ayırır. Bir kaynak planı ile kaynak planı yönergeleri arasında bir ebeveyn-çocuk ilişkisi vardır. Her yönerge bir tüketici grubunu referanslar ve şu anda aktif olan plan için iki direktif aynı tüketici grubunu referans alamaz.

Şekil 2.6: Kaynak Yönetimi bileşenlerinin birbirleriyle olan etkileşimi



[4]

Kurumda etkin bir kaynak yönetimi kullanılmamaktadır. DBMS_RESOURCE_MANAGER paketi ile uygulama kullanıcıları, rapor kullanıcıları ve yöneticiler için ayrı tüketici grupları oluşturulup kaynak paylaşımının zamana göre ve sistem yüküne göre dengeli dağılımı yapılmalıdır.

3.VERİTABANI GÜVENLİĞİ

Siber tehdit, veri gizliliği ile ilgili kanunlarda yapılan düzenlemeler, bilgi korumanın kurum için en üst düzeyde sorun haline gelmesine neden olmuştur. Güvenlik alanında yapılan çeşitli araştırmalar, SQL enjeksiyonu, çalınmış kimlik bilgileri veya sisteme ve verilere erişimi yetkilendiren içerikler tarafından büyük bir veri ihlali yüzdesinin yapıldığı sonucuna varmıştır. Verileri sunucularda koruma altına almak, önleyici ve idari denetimleri kapsayan hem teknik hem de idari işlevleri içeren kapsamlı bir savunma yaklaşımını gerektirir. Bu bölümde kurum veritabanları güvenlik bileşenleri, zafiyetleri yönünden incelenecek olup, veritabanı güvenliğini artırmak için çeşitli öneriler sunulacaktır.

3.1 TNS Dinleyicisi Güvenliği

TNS Dinleyicisi hemen hemen tüm Oracle veritabanlarının ayrılmaz bir parçasıdır. TNS, veritabanlarına bağlanmak için istemciler tarafından kullanılan ağ protokolünü belirtir. TNS Dinleyicisi esas olarak, ağ bağlantı denemelerini dinleyen, bağlantı denemelerinin Dinleyicinin işlediği veritabanlarıyla eşleştiğinden emin olmak için bir proxy olup iletişimlerini uygun veritabanı veya güvenlik tanımlayıcısına (SID) iletmektir. Dinleyiciler, veritabanı bağlantılarını sağlayan proxy sunucusundan daha fazlasıdır. SQL aracılığıyla işletim sistemi çalıştırılabilir dosyalara erişim sağlamak için de kullanılırlar. Bu güçlü özellik, veritabanının işletim sistemi komutlarını ve programlarını, genellikle Linux sistemlerinde veya Windows'daki yerel sistemde veritabanı hesabı olarak çalışan dinleyici yetkisi ile çalıştırmasına olanak tanır. Veri tabanına bağlanma sürecindeki rolü ve ana işletim sistemindeki programları ve komutları çalıştırma kabiliyeti nedeniyle, Dinleyici saldırganların başlıca hedefidir. Bu yüzden ağdaki her TNS Dinleyicisi'ni güvence altına almaya yönelik adımlar atmak oldukça önemlidir. Bir TNS Dinleyicisi aşağıdaki bileşenlerden oluşur.

- **Tnslsnr :** Bu çalıştırılabilir, Dinleyicinin temel işlevselliğini sağlar. Aslında, Oracle veritabanlarına ağ bağlantıları sunan ve bu veritabanlarını PL/SQL çağrıları yoluyla harici prosedürleri çalıştırma kabiliyetine sahip bir sunucu süreci oluşturmaktadır. Bu yürütülebilir dosya, veritabanlarına erişmek için gelen ağ bağlantısına izin veren temel bileşenidir.
- **Lsnrctl :** Bu çalıştırılabilir program, Dinleyiciyi yapılandırmak ve denetlemek için kullanılan yardımcı programdır. Dinleyiciyi yerel makinede veya ağ üzerinden uzaktan yönetme olanağı sağlar. Bu Dinleyici kontrol

programı, dinleyici komutlarını çalıştırarak günlük logları ve izleme dosyalarının yapılandırılmasını, şifrelerin ayarlanmasını, dinleyiciyle ilgili bilgileri ve hizmet verdiği veritabanları listesini göstermeyi ve hatta dinleyici prosesini kapatılmasını sağlar.

- **Sqlnet.ora** : Bu yapılandırma dosyası Dinleyiciye çeşitli ağ parametreleri sağlamak için kullanılır. Güvenlikle ilgili en önemlileri, ağ iletişimi şifreleme ve bütünlük denetimi, kimlik doğrulama modlarını yapılandırma ve veritabanına erişmesine izin verilen ana makinelerin adlarının veya IP adreslerinin listesini sağlamaya yönelik ayarlardır. Bu liste yalnızca meşru ana bilgisayar sistemlerinin veritabanına bağlanmasını sağlamak için kullanılabilir; böylece, kimlik doğrulama başlamadan önce bile sisteme yetkisiz olarak bağlanma girişimleri engellenmiş olur. Sqlnet.ora'da etkinleştirilen güvenlik özelliklerinin çoğunun çalışması için Gelişmiş Güvenlik Seçeneği (ASO) gereklidir.
- **Listener.ora :** Bu yapılandırma dosyası dinleyiciyi ve dinlediği veritabanlarını tanımlar. Listener.ora içinde Dinleyici'yi, dinlediği hizmetleri (SID) ve dinlemeleri gereken IP adreslerini ve bağlantı noktalarını tanımlayan benzersiz bir ad bulunur. Listener.ora aynı zamanda Dinleyici şifreleri, kayıt ve izleme seviyeleri ve günlük ve izleme dosyası konumları gibi diğer önemli yapılandırma bilgilerini de sağlar. Esasen dinleyicinin nasıl çalışması gerektiği hakkındaki tüm bilgiler bu dosyada saklanır. Dinleyici yapılandırmasında yapılan değişiklikler doğrudan listener.ora'yı düzenleyerek veya komutları lsnrctl kullanarak çalıştırarak yapılabilir. Bu dosyada herhangi bir değişiklik yapıldıktan sonra, değişikliğin etkin olabilmesi için dinleyicinin yeniden başlatılması gerekir.
- **Tnsnames.ora :** Dinleyiciye bağlantı için gerekli olan sunucu adı, sunucu IP'si, port bilgisi, bağlantı protokolü, bağlanma tipi, servis adı gibi bilgileri tutar. Daha fazla güvenlik için, bu dosyada kimlik doğrulama SSL sertifikası ile yapılmalıdır.

3.1.1 Dinleyici Zafiyetleri ve Alınması Gereken Önlemler

Dinleyicinin yerel yönetiminin güvenliği, varsayılan olarak yerel işletim sistemi aracılığıyla yapılmaya çalışılır. Bunun yanında, dinleyicinin, dinleyiciyi

başlatma veya durdurma, desteklenen servislerin bir listesini görüntüleme veya Dinleyici Kontrolü yapılandırmasında yapılan değişiklikleri kaydetme gibi yönetimsel işlemler için güvenlik sağlamak için bir parola ile yapılandırılabilir. Şifrelenmiş bir dinleyici parolası oluşturmak için aşağıdaki komutlar kullanılabilir.

LSNRCTL SET PASSWORD

LSNRCTL SAVE_CONFIG

Ağı dinleyen kişiler tarafından dinleyici parolasının çalınma riskini azaltmak için listener.ora dosyasındaki SSL protokolünü kullanarak şifreli iletişim tercihi edilmelidir. Bu ise aşağıdaki gibi yapılır.

ADDRESS = (***PROTOCOL=TCPS***)(HOST=hostname)(PORT=port))

Dinleyici şifre ile korunmamışsa, dinleyicinin çalışma zamanı yönetimini kısıtlamak için alternatif olarak ADMIN_RESTRICTIONS parametresi kullanılmalıdır. Bu parametre listener.ora dosyasında aşağıdaki gibi değiştirildiği zaman dinleyiciye gelen SET komutları devre dışı kalır. *[15]*

ADMIN_RESTRICTION_LISTENER_NAME = ON

LSNRCTL RELOAD

Veritabanına olası izinsiz giriş denemelerini izleyip önlemini alabilmek için dinleyici log ve izleme dosyalarını etkinleştirmek gerekir. Kurum veritabanı üzerinde aşağıdaki komut çalıştırıldığında bu özelliğin etkinleştirilmediği görülmüştür.

LSNRCTL SHOW LOG_STATUS → komut girdisi

LISTENER parameter "log_status" set to OFF →komut çıktısı

Kurum veritabanında aşağıdaki kod adımlarını sırasıyla uygulayarak bu özelliğin etkinleştirilmesi gerekir.

LSNRCTL SET LOG_DIRECTORY $ORACLE_HOME/network/log

LSNRCTL SET LOG_FILE listener.log

LSNRCTL SET LOG_STATUS = on

Eğer şüpheli veya belirsiz bağlantı denemeleri bu log dosyalarında tespit edilirse, bunları daha detaylı inceleyebilmek için dinleyici izleme (trace) dosyaları aşağıdaki komut adımları kullanarak etkinleştirilmelidir.

LSNRCTL SET TRC_LEVEL = on

Dinleyici izleme dosyaları çok detaylı olduğu için muazzam miktarda veri içerir. Bu da disk ve CPU kaynaklarına çok yük bindireceğinden veritabanı performansını

olumsuz olarak etkiler. Bu yüzden şüpheli olmayan zamanlarda aşağıdaki komut ile kapatılmalıdır.

LSNRCTL SET TRC_LEVEL = off

Dinleyicinin varsayılan bağlanma port'u 1521'dir. Varsayılan bağlanma portunu kullanmak saldırganların veritabanını bulmasını kolaylaştırır. Bağlanma port değerini varsayılan değerinden çok uzak bir değere çekmek saldırganın veritabanını bulmasını zorlaştırır. Kurum veritabanı port bilgisi varsayılanda kalmış durumda. 15021 gibi tahmin edilmesi güç bir değere çekilmelidir.

Dinleyicilerin bir güvenlik özelliği de davet edilen veya hariç tutulan düğümlerden oluşan bir kümeyi yapılandırarak, hangi sunucuların veritabanına bağlanabileceğini IP adresi veya sunucu adı düzeyinde bağlantı filtrelemesi yapabilen Geçerli Düğüm Denetimi'dir. Dinleyicinin geçerli düğüm denetimi özelliğini kullanabilmek için izin verilen sunucular aşağıdaki kodlar örnekte olduğu gibi sqlnet.ora dosyasına yazılır.

tcp.validnode_checking = yes

tcp.invited_nodes= (localhost, sunucu1, sunucu2, 192.168.100.21)

Bu listede IP'si veya adı olmayan sunucular veritabanına bağlanamazlar. Çok fazla sunucunun bu yolla denetimi yapılmaya çalışılırsa takibi zor olacağından, kurumda, veritabanına erişmeye yetkili sunucular dinleyici üzerinden yapılmamıştır. Bunun yerine kullanıcı ara yüzü olan daha esnek ve yönetilebilir üçüncü parti ürünler kullanılmıştır.

3.2 Veritabanı Kullanıcı Hesapları Güvenliği

Veritabanı, geçerli veritabanı kullanıcılarının bir listesine sahiptir. Bir veritabanına erişmek için, bir kullanıcı bir veritabanı uygulaması çalıştırmalı ve veritabanında tanımlanmış geçerli bir kullanıcı adı kullanarak veritabanına bağlanmalıdır. Veritabanı, çeşitli yollarla kullanıcılar için güvenlik ayarlamaları sağlar. Kullanıcı hesapları oluştururken, kullanıcı hesabına sınırlar belirlenir. Ayrıca, o kullanıcının güvenlik alanının bir parçası olarak her kullanıcı için mevcut çeşitli sistem kaynaklarının miktarına sınırlar da ayarlanabilir. Veritabanı, kaynak ve oturum bilgileri gibi bilgilerini barındıran bir dizi görünüm sunar. Kullanıcıların güvenliği ve kullanabileceği kaynak miktarı profiller aracılığıyla yapılır. Profil, bir kullanıcı için geçerli olan niteliklerin toplamıdır. Bu tam özelliklerini paylaşan birden çok kullanıcı için tek bir referans noktası sağlar.

3.2.1 Profil kullanarak Kimlik Doğrulama Güvenliğini Artırmak

Kimlik doğrulama (Authentication), veri, kaynak veya uygulama kullanmak isteyen birinin (bir kullanıcı, cihaz veya diğer varlığın) kimliğini doğrulamak anlamına gelir. Bu kimliğin doğrulanması, daha fazla etkileşim için güven ilişkisi kurar. Kimlik doğrulama, erişimi ve eylemleri belirli kimliklere bağlamayı mümkün kılarak hesap verebilirliği de sağlar. Kimlik doğrulama işleminden sonra, yetkilendirme süreçleri, o kullanıcıya izin verilen erişim ve eylem seviyelerine izin verebilir veya bunları sınırlandırabilir. Veritabanlarında kimlik doğrulama kullanıcı hesap bilgisi ve parola ikilisinden oluşur. Parolalara bağımlı veritabanı güvenlik sistemleri, parolaların her zaman gizli tutulmasını gerektirir. Parolalar hırsızlığa ve kötüye kullanıma açık olduğundan, veritabanı bir parola yönetimi politikası kullanır. Veritabanı yöneticileri ve güvenlik görevlileri, bu politikayı kullanıcı profilleri aracılığıyla kontrol ederek veritabanı güvenliğini daha iyi kontrol etmeyi mümkün kılar. Profillerin parolalarla ilgili parametreleri aşağıdaki gibidir.

- **FAILED_LOGIN_ATTEMPTS :** Bir kullanıcının oturum açmaya çalışırken, hesabını kilitlemeden üst üste girebileceği maksimum hatalı parola sayısıdır. Varsayılan değeri 10'dur.
- **PASSWORD_GRACE_TIME :** Hesabın parolasının süresinin dolmadan kaç gün önce kullanıcıya uyarı döneceği bilgisini tutan parametredir. Varsayılan değeri 7'dir.
- **PASSWORD_LIFE_TIME :** Hesabın parolasını kullanabileceği toplam gün sayısıdır. Kurum veritabanlarında bu parametre kesinlikle UNLIMITED değerine ayarlanmamalıdır. Varsayılan değeri 180 gündür.
- **PASSWORD_LOCK_TIME :** Belirtilen sayıda ardışık başarısız oturum açma girişimi sonrasında hesabın kilitli kalacağı gün sayısını ayarlar. Bu zaman geçtikten sonra hesap açılır. Bu kullanıcı profili parametresi, kullanıcı şifrelerinde kaba kuvvet (brute force) saldırılarını önlemeye yardımcı olur, ancak yöneticilerin bakım yükünü artırmaz. Varsayılan değeri 1 gündür.
- **PASSWORD_REUSE_MAX :** Geçerli parolanın tekrar kullanılabilmesi için gereken şifre değişikliklerinin sayısını ayarlar. Varsayılan değeri limitsizdir.
- **PASSWORD_REUSE_TIME :** Parolanın tekrardan kullanımı için geçmesi gereken gün sayısını ayarlar. PASSWORD_REUSE_TIME ve

PASSWORD_REUSE_MAX parametrelerinin geçerli olabilmesi için her ikisi de tamsayı limitsiz değerinden tamsayı değerine atanması gerekir.

- **PASSWORD_VERIFY_FUNCTION :** Bu parametre parola karmaşıklığı belirten bir fonksiyon alır. Bu parametreye Null değeri atanması kullanıcıların zayıf ve tahmin edilebilir parola oluşturmasına izin vereceğinden, artırılmış veritabanı güvenliği için NULL değerinden kaçınılmalıdır.

Kurum veritabanında, kurum bilgi güvenliği politikaları parola ve gizli kimlik bilgisi kullanımı prosedürüne göre parolaların en az 10 karakter, bir büyük harf, bir küçük harf, bir de noktalama işareti içermesi ayrıca tahmin edilebilir basit şifreler olmaması için SYS şemasında VERIFY_FUNCTION isimli bir fonksiyon oluşturulup varsayılan profile atanmıştır.

Artırılmış veritabanı güvenliği için büyük küçük harf duyarlılığını kontrol eden SEC_CASE_SENSITIVE_LOGON sistem parametresi aşağıdaki komutla aktif edilmelidir.

ALTER SYSTEM SET SEC_CASE_SENSITIVE_LOGON = TRUE

Bir kullanıcı hesabına parola oluştururken parola değeri tırnak içerisinde verilirse veritabanı, kullanıcının parolasını SYS.USER$ tablosunun PASSWORD kolonunda açık bir halde saklar. Şifrelerin veritabanında kriptolu bir biçimde saklanması için tırnak içerisinde parola oluşturmaktan kaçınılmalıdır.

3.2.2. Varsayılan Hesapların Oluşturduğu Güvenlik Zafiyetleri ve Önlemleri

Veritabanı ilk yüklendiğinde veya yeni bir özellik kurulduğunda, veritabanı bu özellikleri varsayılan hesaplarla yönetir. Her varsayılan hesabında varsayılan parolaları vardır. Bu parolaların internette kolay bulunmasından dolayı saldırganlar için açık kapıdır. Eğer ihmal varsa, varsayılan hesapları ve parolaları kullanarak veritabanlarına giriş oldukça kolaydır. Bu bölümde, veritabanlarındaki varsayılan hesapları tanımlama ve ardından bunları güvence altına almak için uygulanması gereken stratejiler anlatılacaktır.

Varsayılan hesaplar yönetimsel olan ve yönetimsel olmayan hesaplar olarak iki sınıfa ayrılır. Yönetimsel olan hesaplar önceden tanımlanmış yönetim hesapları seti sağlar. Bu hesapların CREATE ANY TABLE, ALTER SESSION veya SYS

şemasına ait veritabanını yönetmek için kullanılan paketler üzerindeki EXECUTE yetkisi gibi önemli ayrıcalıklara sahiptir. Bu hesapların varsayılan tablo alanı SYSTEM veya SYSAUX'tur. Yönetimsel hesaplar aşağıda açıklanmıştır.

- CTXSYS : Veritabanında metin işlemlerini yöneten hesaptır.
- DBSNMP : EM veritabanı yönetim aracını izlemek ve yönetmek için kullanılan hesaptır. Bu hesap açık kalmak zorunda olduğu için bu hesabın varsayılan parolası güçlü bir şifre ile değiştirilmesi gerekir.
- EXFSYS : Bu hesap karmaşık PL/SQL kuralları ve ifadeleri oluşturmaktan sorumludur. Ayrıca DDL, DML ve ilişkili meta verileri içerir.
- MDSYS : Konumsal (spatial) verileri yöneten hesaptır.
- OLAPSYS : OLAP kataloğunu yöneten hesaptır.
- ORDDATA : Veritabanı Multimedya veri modeli içeren hesaptır.
- ORDPLUGINS : Üçüncü parti eklentilerin yüklenmesini sağlayan hesaptır.
- ORDSYS : Veritabanı multimedya yöneticiliği yapan hesaptır.
- OUTLN : Plan kararlılığı, yürütme planlarını depolanan ana hatlarda koruyarak bazı veritabanı ortamındaki değişikliklerin, uygulama performans özelliklerini etkilemesini önleyen hesaptır.
- SYS : Veritabanı yönetimini yapan hesaptır. Bu hesap açık kalmak zorunda olduğu için bu hesabın varsayılan parolası güçlü bir şifre ile değiştirilmesi gerekir.
- SYSMAN : Oracle Enterprise Manager veritabanı yönetim görevlerini gerçekleştirmek için kullanılır. Bu hesap açık kalmak zorunda olduğu için bu hesabın varsayılan parolası güçlü bir şifre ile değiştirilmesi gerekir.
- SYSTEM : Varsayılan bir genel veritabanı yöneticisi hesabıdır. . Bu hesap açık kalmak zorunda olduğu için bu hesabın varsayılan parolası güçlü bir şifre ile değiştirilmesi gerekir.
- WMSYS : Oracle Workspace Manager için meta veri bilgisini depolayan hesaptır.
- XDB : XML veri ve meta verileri saklamak için kullanılan hesaptır. *[23]*

Yönetimsel olmayan varsayılan hesaplar belli işleri gerçekleştirmek için minimum ayrıcalıklarla oluşturulmuş hesaplardır. Varsayılan tablo alanı USERS'tur. Bu hesaplar yönetimsel varsayılan hesaplar kadar kritik hesaplar değildir. Kurum

veritanlarında aşağıdaki sorgu çalıştırılarak halen varsayılan parolaları kullanan varsayılan hesapların durumu Tablo 3.1'de gösterilmiştir.

select username,account_status from dba_users where username in (select * from dba_users_with_defpwd) order by username

Tablo 3.1 : Kurum Veritabanında Varsayılan Hesapların Durumu

Varsayılan Hesap	**Hesap Durumu**
APPQOSSYS	LOCKED
CTXSYS	EXPIRED
DIP	EXPIRED & LOCKED
EXFSYS	EXPIRED & LOCKED
MDDATA	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
OLAPSYS	EXPIRED & LOCKED
ORACLE_OCM	EXPIRED & LOCKED
ORDDATA	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
OUTLN	EXPIRED & LOCKED
OWBSYS	EXPIRED & LOCKED
OWBSYS_AUDIT	EXPIRED & LOCKED
SI_INFORMTN_SCHEMA	EXPIRED & LOCKED
SPATIAL_CSW_ADMIN_USR	EXPIRED & LOCKED
SPATIAL_WFS_ADMIN_USR	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
XDB	EXPIRED & LOCKED
XS$NULL	EXPIRED & LOCKED

Tablo 3.1'de CTXSYS hesabının kilitli olmadığı tespit edilmiştir. Bu hesap önemli ayrıcalıkları olan yönetimsel varsayılan hesap olduğu için aşağıdaki komutla kilitlenmelidir.

ALTER USER CTXSYS ACCOUNT LOCK;

Kilitli bir hesaba doğru şifreyle giriş yapılmaya çalışıldığında aşağıdaki hata alınır.

ERROR:ORA-28000: the account is locked

Varsayılan hesapları sadece kilitlemek bir saldırgana hangi hesapları ve dolayısıyla belirli bir veritabanında hangi özellikleri yüklediğini bildirir. Saldırganın hangi olası güvenlik açığı bulunan özelliklerin kurulduğunu bilmesi, potansiyel istismarların bir listesini oluşturarak, veritabanına girmeden saldırı planı oluşturmasını sağlar. Bunun önüne geçmek için Tablo 3.1'deki tüm hesapların parolaları aşağıdaki komutla değiştirilmesi gerekir.

ALTER USER varsayilan_hesap IDENTIFIED BY kuvvetli_parola;

Veritabanı güvenliği için her yeni yama geçişinde, veritabanı versiyon yükseltmesinde, yeni özellik aktif edildiğinde, her yeni veritabanı kurulumunda yukardaki kontroller yapılmalı ve varsayılan hesaplar kilitli hale getirilmeli ve şifresi değiştirilmelidir.

3.3 Veritabanı Yetkilendirme

Bir kullanıcı veritabanında kimliği doğrulandıktan sonra, bir sonraki adım, hangi nesnelerin, ayrıcalıkların ve kullanıcıların erişmesine veya kullanmasına izin verilen kaynakları belirlemektir. Bu bölümde kullanıcıların sistem kaynaklarının hangilerini ve ne kadarını kullanabileceğini profiller üzerinden sınırlandırmayı, roller ve ayrıcalıklar kullanarak hangi veritabanı nesnelerine erişebileceğini, yetkilendirmeden kaynaklanan güvenlik açıklarını ve detaylı yetkilendirme açıklanacaktır.

3.3.1 Profil kullanarak Sistem Kaynakları Yetkilendirmesi

Bir kullanıcının sorgusunu çalıştırmak için yeterli CPU gücü veya disk alanı veya G/Ç bant genişliği gibi sistem kaynaklarına ihtiyaç duyar. Bütün bu kaynaklar doğal olarak sınırlı olduğundan, veritabanı, bir kullanıcının kullanabileceği bu kaynakların ne kadarını kontrol edebilecek bir mekanizma sağlar. Bir veritabanı profili, bu mekanizmayı sağlayan adlandırılmış bir kaynak sınırı setidir. Profiller üzerinden kullanıcılara kaynak yetkilendirmesi parametreleri aşağıdadır.

- SESSIONS_PER_USER : Aynı kullanıcı hesabı için yapılabilecek eşzamanlı oturum açma sayısını belirleyen parametredir. Bu sınıra erişildiğinde aynı kullanıcı adı ile oturum açmaya çalışan oturumlar engellenir.
- CPU_PER_SESSION : Bir oturumun sunucu işleminin oturumu zorla sonlandırılmadan önce kullanılmasına izin verilen CPU zamanını belirleyen parametredir.
- CPU_PER_CALL : Bir oturumun sunucu işleminin, deyim zorla sonlandırılmadan önce bir SQL deyimini yürütmek için kullanabileceği CPU zamanının belirleyen parametredir.
- LOGICAL_READS_PER_SESSION : Oturum zorla sonlandırılmadan bir oturum tarafından okunabilecek maksimum blok sayısını belirleyen parametredir.

- LOGICAL_READS_PER_CALL : SQL deyimi zorla sonlandırılmadan önce tek bir deyim tarafından okunabilecek maksimum blok sayısını belirleyen parametredir.
- PRIVATE_SGA : Paylaşılan sunucu mimarisi aracılığıyla bağlanan oturumlar için, oturumun oturum verileri için SGA'ya almasına izin verilen maksimum veri miktarını kilobayt cinsinden belirleyen parametredir.
- CONNECT_TIME : Bir oturum sonlanmadan veritabanına bağlı kalabileceği süreyi dakika cinsinden belirleyen parametredir.
- IDLE_TIME : Bir oturum sonlanmadan veritabanında boşta bekleyebileceği süreyi dakika cinsinden belirleyen parametredir.

Bir oturum, bir kaynak sınırına ulaşıldığından dolayı sona erdiğinde, devam eden bir işlem geri alınır. Bir SQL deyimi sonlandırılırsa, deyim tarafından yapılan iş geri alınır, ancak daha önceki sonlanan işlemler kalır.

Profil parola parametreleri her durumda geçerliyken profil kaynak parametrelerinin geçerli olabilmesi için, bir veritabanı parametresi olan RESOURCE_LIMIT'in varsayılanda FALSE olan değeri TRUE değerine aşağıdaki komutla çekilmesi gerekmektedir.

ALTER SYSTEM SET RESORUCE_LIMIT=TRUE;

Kurum veritabanı incelendiğinde LOGICAL_READS_PER_SESSION ve LOGICAL_READS_PER_CALL parametrelerinin varsayılın değerinde yani UNLIMITED kaldığı tespit edilmiştir. Olası G/Ç darboğazlarının önüne geçerek performans artırmak için veya kurumun bilgisi dahilinde olmayan çok büyük miktarda verinin aktarılmasını önleyerek artırılmış güvenlik sağlayabilmek için bu parametrenin aşağıdaki komut seti ile varsayılanda kalması önlenmelidir.

ALTER PROFILE LIMITED_PROFILE LIMIT LOGICAL_READS_PER_CALL=100

3.3.2. Ayrıcalık ve Roller

Bir kullanıcı ayrıcalığı, belirli bir SQL ifadesi türünün veya başka bir kullanıcıya ait bir nesneye erişmenin, bir PL/SQL paketi çalıştırmanın ve benzeri bir hakkı çalıştırmanın hakkıdır. Roller ise ayrıcalıkları veya diğer rolleri birlikte gruplandırmak için veritabanı yöneticileri tarafından oluşturulur. Roller kullanıcılara birden fazla ayrıcalık veya rol verilmesini kolaylaştırmanın bir yoludur. Kullanıcılara doğrudan ayrıcalıklar verilebilir veya role ayrıcalıklar verilebilir sonra kullanıcıları

role dahil edilebilir. Ancak roller, ayrıcalıkların daha kolay ve daha iyi yönetilmesini sağladığından, belirli kullanıcılara değil de genellikle rollere ayrıcalıklar verilmelidir.

Kullanıcıların yapması gereken işleri yapabilmesi için sadece minimum gerekli ayrıcalıklar verilmelidir. Gereksiz verilen ayrıcalıklar veritabanı güvenliğini tehlikeye atabilir. Örneğin yönetici görevi olmayan kullanıcılara hiçbir zaman SYSDBA veya SYSOPER ayrıcalıkları verilmemelidir. Sistem ayrıcalığı ve nesne ayrıcalığı olmak üzere iki tür ayrıcalık vardır.

- **Sistem Ayrıcalığı** : Belirli bir eylemi gerçekleştirme veya belirli bir türdeki herhangi bir şema nesnesinde bir eylem gerçekleştirme hakkıdır. Örneğin, tablo alanlarını oluşturmak ve veritabanındaki herhangi bir tablonun satırlarını silmek için gereken ayrıcalıklar sistem ayrıcalıklarıdır. Sistem ayrıcalıkları çok güçlü ayrıcalıklardır. Bu yüzden sistem ayrıcalıkları sadece gerektiğinde güvenilir kullanıcılara verilmelidir. İşi bitince de yetkisi alınmalıdır. Bir kullanıcıya verilen sistem ayrıcalıkları DBA_SYS_PRIVS veri sözlüğü görünümünde tutulur.
- **Nesne Ayrıcalığı :** Bir nesne ayrıcalığı, bir veritabanı nesnesinde bir kullanıcıya verilmiş bir haktır. Nesne ayrıcalıkları bir tabloda güncelleme, başka bir şemanın tablosunu okuyabilme veya prosedürünü çalıştırabilme gibi yetkilerdir.

ANY yetkisini içeren bir sistem ayrıcalığının veritabanın kendi meta data verilerini tutan tablolarda değişiklik yapmasını önleyebilmek, sistemin çalışabilirliği açısından oldukça kritiktir. ANY yetkisine sahip bir kötü niyetli kullanıcı, veritabanı veri sözlüğü tablolarına erişip değiştirebilir. Veri sözlüğünü güvence altına almak için bir veritabanı parametresi olan 07_DICTIONARY_ACCESSIBILITY'nin değeri aşağıdaki komut kullanılarak FALSE değerine çekilip veritabanı yeniden başlatılmalıdır.

```
ALTER SYSTEM SET 07_DICTIONARY_ACCESSIBILITY = FALSE SCOPE=SPFILE;
```

ANY anahtarını kullanan sistem ayrıcalıklarının başka sıkıntıları da vardır. Bu tür ayrıcalıkları, veritabanındaki tüm nesneler kategorisi için ayrıcalıklar belirlemeyi sağlar. Örneğin, CREATE ANY PROCEDURE sistem ayrıcalığı, bir kullanıcının veritabanının herhangi bir şemasında bir prosedür oluşturmasına izin verir. ANY yetkisine sahip kullanıcılar tarafından oluşturulan bir nesnenin davranışı, oluşturulduğu şema ile sınırlı değildir. Örneğin, CREATE ANY PROCEDURE ayrıcalığına sahip olan A kullanıcısı ve B şemasında bir prosedür oluştururursa, prosedür B kullanıcısı olarak çalışacaktır. Bununla birlikte, B şeması, A şeması

tarafından oluşturulan prosedürün kendisi olarak çalıştığının farkında olmayabilir. Bu tarz durumlar güvenlik ihlaline neden olabilir. Bu yüzden ANY yetkisi içeren bir yetkiyi verirken çok dikkatli olunmalıdır.

Kurum veritabanında, yönetici yetkisi olmayıp ANY anahtarını kullanan sistem ayrıcalığına sahip kullanıcıları tespit etmek için aşağıdaki sorgu çalıştırıldığında Tablo 3.2 elde edilmiştir.

```
SELECT * FROM DBA_SYS_PRIVS WHERE PRIVILEGE like '%ANY%' and
GRANTEE in (SELECT username FROM DBA_USERS WHERE PROFILE =
'LIMITED_PROFILE')
```

Tablo 3.2 : ANY Ayrıcalığına Sahip Kullanıcılar

USERNAME	**PRIVELEGE**
OUTLN	EXECUTE ANY PROCEDURE
COB_ORTAK	SELECT ANY TABLE
SCOTT	CREATE ANY SYNONYM

Aşağıdaki komut seti çalıştırılarak, bu yetkilerin bu kullanıcılardan alınması gerekmektedir.

```
REVOKE EXECUTE ANY PROCEDURE FROM OUTLN;
REVOKE SELECT ANY TABLE FROM COB_ORTAK;
REVOKE CREATE ANY SYNONYM FROM SCOTT;
```

Güvenli uygulama rolleri kullanarak veritabanı güvenliği daha da artırılabilir. Güvenli uygulama rolü yalnızca yetkili bir sistem paketi olan DBMS_MACSEC_ROLES tarafından etkinleştirilebilen roldür. Bu paket uygulamaya erişimi denetlemek için gerekli güvenlik politikalarını yürütmeyi sağlar. Güvenli uygulama rolleri, kendisini çağıran uygulama tarafından kullanımını kısıtlar. Bu tür bir rol, parolaların uygulama kaynak koduna gömülmediği veya bir tabloda saklanmadığı için güvenliği artırır. Bu şekilde, veritabanının gerçekleştirdiği eylemler, kurumun güvenlik politikalarının uygulanmasına dayanır ve bu tanımlar, uygulamaların yerine veritabanında tek bir yerde saklanır. Kurumun güvenlik politikasının değişmesi gerektiği durumlarda, uygulamaları değiştirmeden tek bir yerde yapılır. Kullanıcılar veritabanına nasıl bağlanırlarsa bağlansınlar, sonuç her zaman aynıdır, çünkü ilke rolü kullanır. Bir veritabanı bağlantısının varlığını sağlamak için güvenli uygulama rollerini kullanılabilir. Güvenli bir uygulama

rolünün bir paket tarafından uygulanan bir rolü olduğu için paket, kullanıcıların veritabanına orta katman yoluyla veya belirli bir IP adresinden bağlanabileceğini doğrulayabilir. Bu şekilde, güvenli uygulama rolü, kullanıcıların bir uygulamanın dışındaki verilere erişmesini engeller. Onlara verilen uygulama ayrıcalıkları çerçevesinde çalışmak zorunda kalırlar. *[18]*

Kurum veritabanı incelendiğinde güvenli uygulama rolü kullanımına rastlanmamıştır. Kurum veritabanlarına kullanıcılar gerek uygulamadan olsun gerekse geliştiriciden olsun tam yetkiyle gelmektedir. Verilerin uygulama dışından çağrılabiliyor olması güvenlik açığıdır. Kurum veritabanlarında güvenli uygulama rolleri tanımlayarak bu açığın önüne geçilebilir.

3.3.3 Public Rolünden Kaynaklanan Zafiyetler ve Önlemleri

Public rolü veritabanı kurulduğunda varsayılan olarak gelir. Veritabanındaki bütün kullanıcılar varsayılan olarak public rolüne üyedirler ve bu üyelikten çıkarılamazlar. Public rolünün kendisi de hiçbir şekilde veritabanından kaldırılamaz. Public'e verilen izinler veritabanındaki herkese verilir. Veritabanındaki bütün kullanıcıların erişimine açılmak istenen yetkiler için, public herkese ayrı ayrı yetki verme veya rol atama gereksinimini ortadan kaldırarak zamandan tasarruf sağlayan bir araçtır. Ancak hassas fonksiyonlarda veya etkili sistem izinlerini public'e vermede mümkün olduğunca kaçınılmalıdır. Bu yüzden public rolüne verilen izinler veritabanı güvenliği açısından oldukça kritiktir. Bu bölümde veritabanı güvenliği için public rolünden alınması veya kısıtlanması gereken ayrıcalıklar anlatılacaktır.

Aşağıda public rolünün yetkisi olması durumunda güvenlik açığı olabilecek sistem fonksiyonları açıklanmıştır.

- DBMS_RANDOM : SYS şemasındaki yerleşik bir pakettir. Bir kullanıcının rastgele sayılar üretmesine izin verir. Rasgele sayı üreteçleri genellikle verileri şifrelemek ve şifrelerini çözmek için anahtarlar oluşturmak için kullanılır. DBMS_RANDOM ile ilgili problemler şifreleme etrafında döner. Doğru bir analiz araçları setiyle, bir saldırgan, paketteki geçmiş ve gelecek çıkış değerleri hakkında eğitilmiş tahminler yapabilir. Bu nedenle, fonksiyonun şifreleme anahtarının oluşturulmasında kullanıldığı ortamlarda DBMS_RANDOM paketine PUBLIC erişimi yetkisi verilmesi, şifreli verilerin çökmesine neden olabilir. Rastgele çıkışı tahmin edebilen bir

saldırgan, hassas verileri koruyan şifreleme anahtarlarını yeniden yapılandırmak için bu bilgileri kullanabilir. *[7]*

- UTL_FILE : Veritabanı, kullanıcıların işletim sistemi seviyesinde dosyalarla etkileşime girmesine izin vermek için tasarlanmış bir dizi yardımcı fonksiyon sunar. Bu yardımcı fonksiyonlar, veritabanı dışındaki öğelerle izin verdikleri etkileşime dayalı veritabanı paketlerinde gruplandırılmıştır. Bu özel fonksiyon paketlerini çalıştıran ayrıcalık, varsayılan olarak public'e verilir. Yardımcı program paketleri veritabanı geliştiricilerine esneklik getirmekle birlikte bir saldırgan tarafından kötüye kullanılma riski barındırır. UTL_FILE kuşkusuz yardımcı program paketlerinin en güçlüsüdür. Veritabanı sunucusunda işletim sistemi dosyalarını okumak ve yazmak için doğrudan erişim sağlar. Bu paket saldırganın veritabanı sunucusunun kendisini kontrol altına alabilmek ve böylece veritabanını ve içerisindeki tüm verileri kontrol edebilmek için kullanılabilir. UTL_FILE ayrıca, verilerin bozulması veya silinmesi için ya da anahtar çalıştırılabilir dosyaların üzerine yazarak veritabanı ya da veritabanı sunucusunun istikrarsızlaştırılması için kullanılabilir. Artırılmış veritabanı güvenliği için, bu yetki yerine ihtiyaç duyan güvenilir kullanıcıların kullanabilmesi için dizin oluşturularak kullanıcının yetkisi bu dizinle sınırlandırılmalıdır. *[7]*
- UTL_HTTP : Veritabanı üzerinden dış dünyaya erişim sağlamak için kullanılan, varsayılanda public' erişim izni olan SYS şemasına ait program paketidir. Bu paket http ve https üzerinden veri gönderip alabilir. Bu paketin kendisi veritabanına veya üzerinde çalıştığı işletim sistemine doğrudan tehdit oluşturmasa da, saldırganın veri sızdırmasına neden olabilir. Bu paket üzerinden yapılan bağlantılar herhangi bir sorgu sonuçlarını taşımak için kullanılabilir, bu nedenle veritabanındaki hassas veriler, kredi kartı verileri ve personel bilgileri paketlenebilir ve web uygulamasını geçerek tamamen gönderilebilir. Güvenlik duvarları, normal web trafiğine benzediğinden ağın içinden başlatılan bir oturumla bunu genellikle durdurmaz. HTTPS kullanarak, ağ izleyen veri sınıflandırması veya ekstrüzyon algılama aygıtları da buradan başarısız olur ve şifreli trafiği çözemez. Bu yüzden bu paket üzerindeki public'e ait tüm yetkiler kaldırılmalıdır.

- UTL_TCP : Bu paket veritabanından ağ üzerinden mesaj göndermek için kullanılır, bu sefer işlenmemiş TCP paketlerini kullanır. Bu paket varsayılan olarak public tarafından ulaşılabilir haklara sahiptir. Saldırganın veritabanı tarafındaki protokolü yönetmesi ve ağ üzerinden gönderilen iletileri alıp depolayan bir istemci üzerinden ağ trafiğini parçalayıp, TCP paketlerini ele geçirmesi mümkündür. Bu yüzden bu paket üzerindeki public'e ait tüm yetkiler kaldırılmalıdır.

Public rolünün erişim izni olmasından dolayı tüm kullanıcıların çalıştırma yetkisine haiz olduğu güvenlik açıklarına neden olabilecek DBMS_RANDOM, UTL_TCP, UTL_FILE, UTL_HTTP gibi paketlerin durumunu kurum veritabanlarında incelemek için aşağıdaki sorgu çalıştırıldığında Tablo 3.3 elde edilmiştir.

```
SELECT GRANTEE,OWNER,TABLE_NAME,GRANTOR,PRIVILEGE FROM
DBA_TAB_PRIVS WHERE OWNER='SYS' AND TABLE_NAME IN
('DBMS_RANDOM','UTL_TCP','UTL_FILE','UTL_HTTP') and GRANTEE='PUBLIC'
```

Tablo 3.3 : Public Rolünden Alınması Gereken Yetkiler

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE
PUBLIC	SYS	DBMS_RANDOM	SYS	EXECUTE
PUBLIC	SYS	UTL_FILE	SYS	EXECUTE

Aşağıdaki komut seti çalıştırılarak, bu yetkilerin bu public'ten alınması gerekmektedir.

```
REVOKE EXECUTE ON SYS.DBMS_RANDOM FROM PUBLIC;
REVOKE EXECUTE ON SYS.UTL_FILE FROM PUBLIC;
```

3.3.4 VPD ve Bağlam kullanarak detaylı yetkilendirme

VPD (Sanal Gizli Veritabanı), veritabanı erişimini satır ve sütun düzeyinde kontrol etmek için güvenlik politikaları (policy) oluşturabilmeyi sağlar. VPD, bir VPD güvenlik ilkesinin uygulandığı tabloya veya görünüme karşı verilen bir SQL deyimine dinamik bir WHERE deyimi ekler. VPD, güvenliği, doğrudan veritabanı tablolar veya görünümler üzerinde ince ayrıntı seviyesine kadar zorlar. VPD yoluyla güvenlik politikaları doğrudan bu veritabanı nesnelerine eklendiğinden dolayı ve bir kullanıcı verilere eriştiğinde politikalar otomatik olarak uygulanır ve güvenliği atlayabilmek için herhangi bir yol kalmaz.*[6]*

Bir kullanıcı doğrudan veya dolaylı olarak bir VPD ilkesi ile korunan bir tablo veya görünüme eriştiğinde veritabanı, kullanıcının SQL deyimini dinamik olarak değiştirir. Bu değişiklik, güvenlik politikasını uygulayan bir fonksiyon tarafından döndürülen bir WHERE koşulunu oluşturur. Fakat kullanıcı bu WHERE koşulundan etkilenmesine rağmen görmez.

Bir güvenlik politikasını bir veritabanı tablosu veya görünüm ile ilişkilendirmek, potansiyel olarak ciddi bir uygulama güvenliği sorununu çözebilir. Güvenlik ilkelerini doğrudan tablolar veya görünümlere ekleyerek detaylı erişim kontrolü, bir kullanıcının verilere erişmesine bakılmaksızın aynı güvenliğin yapılmasını sağlar. VPD ile güvenlik politikası tanımlandığında kullanıcının tabloyu veya görünümü kullanımı değişmeyeceğinden aynı zamanda kullanım kolaylığı sağlar.

VPD politikaları ile uygulama bağlamlarını(context) genelde birlikte kullanılır. Bir uygulama bağlamı oluşturulduğunda kullanıcı bilgileri güvenli bir şekilde önbelleğe alınır. Önceden hazırlanmış ortamı yalnızca belirlenen uygulama paketi ayarlayabilir. Kullanıcı tarafından değiştirilemez veya pakete dahil edilemez. Ek olarak, veriler önbelleğe alındığından performans artar. CREATE CONTEXT komutu ile uygulamalara özel bağlam oluşturulabileceği gibi genellikle, kullanıcı oturumuyla ilgili bilgileri almak için USERENV yerleşik bağlamı kullanılır. Tablo 3.4'te USERENV bağlamı ile alınabilecek kullanıcı bilgilerinin parametreleri kısaca açıklanmıştır.

Tablo 3.4 : USERENV Bağlam Parametreleri

Bağlam Parametresi	**Dönüş Değeri**
CURRENT_SCHEMA	Oturum için varsayılan şema
DB_NAME	Veritabanı adı
HOST	Kullanıcının bağlandığı ana makinenin ismi
IP_ADDRESS	Kullanıcının bağlı olduğu IP adresi
OS_USER	Veritabanı oturumu başlatan işletim sistemi hesabı
SESSION_USER	Kimliği doğrulanmış veritabanı kullanıcı adı

Kurum veritabanında VPD kullanımını incelemek için GV$VPD_POLICY görünümü sorgulandığında, tanımlı VPD politikasına rastlanmamıştır.

GV$SESSION görünümü bağlı kullanıcıların her bir oturum için ayrı ayrı detaylı bilgilerini içerir. Bu görünümü sorgulamak uygulama geliştiricilerin çok işine yarayabilir ancak bu görünümde diğer kullanıcılarında bilgisi olduğu için bu görünümü sorgulama yetkisi geliştiricilere verilmemiştir. VPD kullanarak bütün kullanıcıların sadece kendi oturum bilgilerine erişebilme yetkisi verilebilir. Aşağıda bu işi gerçekleştirebilme adımları gösterilmiştir. Buna göre kurum veritabanındaki diğer tablo veya görünümlerinde VPD ile güvenliği ve kullanışlılığı artırılabilir.

```
Sql>grant select on sys.v_$session to public;
Sql> CREATE OR REPLACE FUNCTION
VTYS.VSESS_SELECT_RESTRICT(owner varchar2, object_name varchar2) return
varchar2 is
ret_predicate varchar2(1000);
v_profile varchar2(100);
BEGIN
ret_predicate := ' 1=1 ';
select profile into v_profile from dba_users where username =
SYS_CONTEXT('USERENV','SESSION_USER') ;
if  v_profile = 'LIMITED_PROFILE' THEN
   ret_predicate := ' username =  ''' ||
SYS_CONTEXT('USERENV','SESSION_USER')  || '''';
END IF;
return ret_predicate;
END VSESS_SELECT_RESTRICT;
Sql> exec dbms_rls.add_policy (object_schema => 'PUBLIC',
object_name => 'V$SESSION',
policy_name => 'VSESS_SELECT_RESTRICT',
function_schema => 'VTYS'
,policy_function => 'VSESS_SELECT_RESTRICT',
statement_types => 'SELECT',update_check => TRUE,enable => TRUE);
```

Ayrıca bu bölümde anlatılan bağlamları kullanarak kurum veritabanlarında oluşturulması gereken tetikleyici ile kullanıcı bazında IP doğrulama yapılabilir.

Böylelikle kurum içerisindeki yetkili kullanıcılardan gelebilecek olası kaba kuvvet(brute force)saldırılarının önüne geçilmiş olunur.

```
CREATE OR REPLACE TRIGGER check_ip_addres
AFTER LOGON
ON DATABASE
BEGIN
  IF USER IN ('USER1', 'USER2') THEN
    IF SYS_CONTEXT('USERENV', 'IP_ADDRESS') NOT IN
('10.117.100.29','10.117.100.30') THEN
      RAISE_APPLICATION_ERROR(-20000, 'yetkisiz adres (' || l_ip_address || ')');
    END IF;
  END IF;
END;
```

3.4 Veritabanı Etkinliklerini Denetleme (Auditing)

Bu bölümde, kullanıcıların veritabanı eylemlerinin izlenmesi ve kaydedilmesi demek olan Denetleme (Auditing) açıklanacaktır. Auditing veritabanında yapılan şüpheli etkinlikleri kaydederek, veritabanı saldırganlarını tespit etmede kullanılır. Sorunlu işlemleri de kayıt altına alarak bu işi yapan kullanıcıların hesap vermesini sağlar veya bu işlemlere karşı caydırıcı olur. Kullanıcılar kendilerine verilen ayrıcalıkları kötüye kullanıyorsa bu yetkiler kullanıcılardan alınır. Auditing, kullanılan veya kullanılmayan nesneler hakkında istatistik toplanmasını sağlayarak sadece güvenlik için değil aynı zaman performans içinde veritabanı yöneticisi tarafından kullanılır. Bu yüzden güvenli ve performanslı veritabanları için Auditing mutlaka etkinleştirilmelidir. Auditing, SQL ifadelerini, ayrıcalıklarını ve şema nesnelerini ve ağ ve çok katmanlı faaliyetleri denetlemek için başlatma parametreleri ve AUDIT ve NOAUDIT SQL ifadelerini kullanır.

AUDIT mekanizmasını etkinleştirmek için bir sistem parametresi olan AUDIT_TRAIL'in değeri aşağıda listelenen değerlerden NONE hariç birine aşağıdaki komutla atanmalıdır. Bu komutun geçerli olabilmesi için sonrasında veritabanı kapatılıp, açılmalıdır.

```
ALTER SYSTEM SET AUDIT_TRAIL=DB_EXTENTED SCOPE=SPFILE;
```

- **NONE :** Audit denetimini devre dışı bırakır.

- **OS :** Audit kayıtları işletim sistemi üzerinde AUDIT_FILE_DEST parametresinin gösterdiği dizine yazılır.
- **DB :** Audit kayıtları veri sözlüğü tablosu olan SYS.AUD$ tablosuna yazılır. Çeşitli görünümlerle veritabanı üzerinden detaylı sorgulama yapılabilir.
- **DB_EXTENDED :** Audit kayıtlarını, sorguları değişkenleri ile birlikte detaylı olarak veritabanında tutar.
- **XML :** Audit kayıtlarını işletim sistemi üzerinde XML türünde tutar.
- **XML_EXTENDED :** Audit kayıtlarını, sorguları değişkenleri ile birlikte detaylı olarak XML türünde tutar.

Kurum veritabanı incelendiğinde AUDIT_TRAIL parametresinin DB_EXTENDED olduğu tespit edilmiştir. Bu veritabanı yöneticilerinin bu bilgileri görüp değiştirebileceği anlamına gelir. Veritabanını daha güvenli hale getirmek için bu parametrenin OS olarak değiştirilip, güvenlik yöneticilerinin işletim sisteminde veritabanı yöneticilerinin erişemeyeceği kısıtlarda bir yer belirleyip AUDIT kayıtları orada oluşturulmalıdır. Ayrıca veritabanına erişilemediği durumlarda bile bu kayıtlara erişim mümkündür. Bunun yanında büyük veritabanlarında AUDIT kayıtlarını işletim sistemi seviyesinde tutmak veritabanı performansını artırır.

Her kullanıcı kendi şemasındaki kendi nesnelerinin AUDIT kayıtlarını tutabilir. USER_AUDIT_TRAIL tablosundan okuyabilir. Bunu yapabilmek için herhangi bir ayrıcalık gerekmez. Fakat başka bir şemanın AUDIT kayıtlarını oluşturmak istiyorsa AUDIT ANY sistem ayrıcalığına sahip olması gerekir.

Veritabanında yapılan her işlem için AUDIT oluşturmak performans problemlerine ve sistem kaynaklarının tükenmesine neden olur. Bu yüzden AUDIT oluştururken kurumun ihtiyaçlarına göre seçici olunmalıdır. Sadece gerekli işlemler ve gerektiği süre kadar AUDIT kuralları tanımlanmalıdır. Varsayılan olarak AUDIT kayıtları bir sistem tablo alanı olan SYSAUX'ta tutulur. Eğer bu kayıtların büyüklüğü takip edilmezse sistem tablo alanını doldurarak veritabanının çalışmamasına neden olabilir. Bu durumu engellemek için CREATE TABLESPACE ile yeni bir tablo alanı oluşturduktan sonra AUDIT kayıtları aşağıdaki kodla yeni oluşturulan tablo alanına taşınmalıdır.

```
BEGIN
DBMS_AUDIT_MGMT.set_audit_trail_location(audit_trail_type=>
DBMS_AUDIT_MGMT.AUDIT_TRAIL_AUD_STD,audit_trail_location_value =>
'yeni_olusturulan_tablo_alani');
DBMS_AUDIT_MGMT.set_audit_trail_location(audit_trail_type=>
DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD,audit_trail_location_value =>
'yeni_olusturulan_tablo_alani);
END;
```

Çok eski AUDIT kayıtları aşağıdaki kodla silinmelidir.

```
delete from sys.aud$ where ntimestamp# <  trunc(sysdate-365);
```

Standart AUDIT ile SELECT veya DML işlemleri takip edilebilir. Ancak bazı satırların veya sütunların hassas bilgi içerdiği tablo veya görünümlerde standart AUDIT bunu ayırt edemez. Özellikle hassas tablolardaki sadece hassas kayıtları okuyan kullanıcıların takip edilmesi güvenlik açısından çok elzem olabilir. FGA kullanarak, yalnızca belirli satırlara erişildiğinde veya belirli satırların belirli sütunlarına erişildiğinde denetim kayıtları oluşturacak şekilde yapılandırılabilir. Kurum veritabanları incelendiğinde, standart AUDIT'in yaygın olarak kullanılmasına rağmen, FGA'nın kullanılmadığı DBA_FGA_AUDIT_TRAIL görünümünden tespit edilmiştir. Kurum veritabanında daha detaylı denetim kayıtları oluşturmak için aşağıdaki gibi DBMS_FGA paketi kullanılabilir.

```
SQL>execute dbms_fga.add_policy(object_schema=>'sema',object_name=>'obje',
policy_name=>'pol',-> audit_condition=>'kosul',-> audit_column=>'kolon');
```

3.5 Veri Kriptolama

Veri Kriptolama, şifrelenmiş verilerin sadece yetkili tarafların okuyabileceği şekilde kodlama işlemidir. Şifreleme, kendiliğinden müdahaleleri önlemez, ancak anlaşılabilir bir içeriği yetkisiz kişilere sunmayı engeller. Bir şifreleme şemasında, erişilmek istenen veri, bir kriptolama algoritması kullanılarak şifrelenir, yalnızca tanımlı bir anahtarla çözülürse okunabilen metin oluşturulur. Teknik nedenlerle, bir şifreleme şeması genellikle bir algoritma tarafından üretilen sahte rasgele bir şifreleme anahtarı kullanır. Prensip olarak, kriptolanmış datayı anahtarı elinde bulundurmadan şifre çözmek mümkündür, ancak iyi tasarlanmış bir şifreleme şeması için büyük hesaplama kaynakları ve becerileri gerektirir. Yetkilendirilmiş bir alıcı,

veriyi, gönderen tarafından alıcılara verilen anahtarla kolayca şifreyi çözebilir ancak yetkisiz kullanıcılar kriptolanmış veriyi okuyamaz.

Oracle veritabanlarında veriyi diske kriptolu olarak yazmak için TDE (Şeffaf Veri Şifreleme) teknolojisi kullanılır. TDE, tablo sütunlarında saklanan kredi kartı numaraları gibi hassas verilerin şifrelenmesini sağlar. Şifreli veriler, verilere erişimi olan bir veritabanı kullanıcısı için şeffaf bir şekilde çözülür. TDE, depolama ortamının veya veri dosyasının çalınması durumunda ortamda depolanan verilerin korunmasına yardımcı olur. Bu veri dosyalarını korumak için TDE, verilerin kriptolanmasını sağlar. TDE, yetkisiz kişilerin verileri okumasını önlemek için şifreleme anahtarlarını veritabanının dışındaki bir güvenlik modülünde saklar. TDE, verilerin ağ üzerinde aktarılırken güvenliğini sağlamaz. TDE kullanarak verileri şifrelemenin avantajları şöyledir:

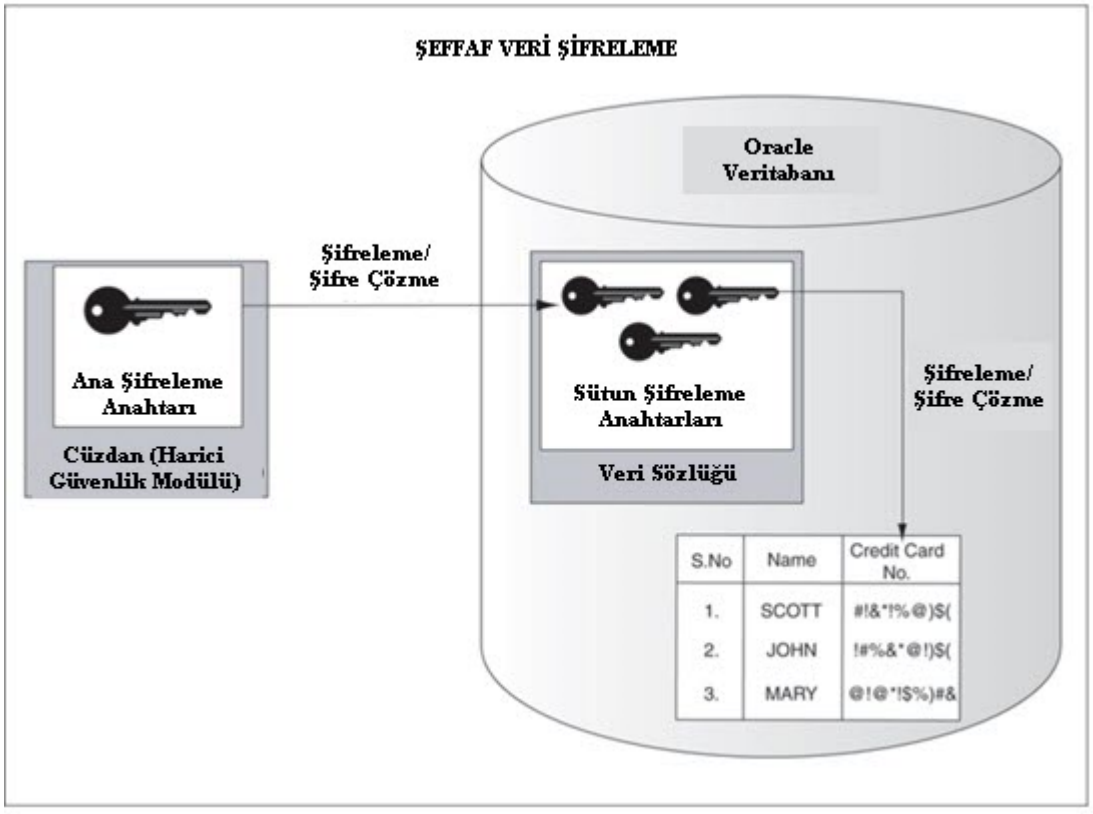
- ✓ Depolama ortamının veya veri dosyasının çalınması durumunda hassas bilgilerin güvenli olmasını sağlar.
- ✓ Güvenlikle ilgili mevzuata uygunluk sorunlarını çözmeye yardımcı olabilir.
- ✓ Verilerin şifresini çözmek için tetikleyiciler veya görünümler oluşturmaya gerek yoktur. Tablolardaki veriler şeffaf bir şekilde veritabanı kullanıcısı için çözülür.
- ✓ Veritabanı kullanıcılarının, erişmekte oldukları verilerin şifrelenmiş biçimde saklandıklarının farkında olmaları gerekmez. Veriler, veritabanı kullanıcıları için şeffaf bir şekilde çözülür ve kendi tarafında herhangi bir işlem yapılmasına gerek yoktur.
- ✓ Uygulamaların şifreli verileri işlemek için değiştirilmesini gerektirmez. Veri şifreleme ve şifre çözme veritabanı tarafından yönetilir.

TDE, SQL katmandaki verileri şifreler ve şifrelerini çözer. SQL katmanını atlayan veritabanı yardımcı programları ve özellikleri, TDE tarafından sağlanan hizmetleri kaldırabilecek durumda değildir. Aşağıdaki veritabanı özellikleriyle TDE kullanılmamalıdır:

- ✓ B-ağacı dışındaki indeks türlerinde
- ✓ Yabancı Anahtar kısıtlamalarında
- ✓ BLOB'larda (bunların yerine Secure LOB kullanılabilir)
- ✓ Somutlaştırılmış görünüm loglarında
- ✓ Taşınabilir tablo alanlarında

Bir tablo şifreli sütun içerdiğinde, şifrelenmiş sütun sayısına bakılmaksızın tek bir anahtar kullanılır. Bu anahtara sütun şifreleme anahtarı denir. Şifreli sütunlar içeren tüm tablolar için sütun şifreleme anahtarları, veritabanı sunucusu ana şifreleme anahtarı ile şifrelenir ve veritabanındaki bir sözlük tablosunda saklanır. Hiçbir anahtar açık bir şekilde saklanmaz. Şekil 3.1'de gösterildiği gibi, ana şifreleme anahtarı, veritabanı dışındaki ve yalnızca güvenlik sorumlusunca erişilebilen harici güvenlik modülünde saklanır. Bu harici güvenlik modülü için veritabanı, bu bölümde açıklandığı gibi bir cüzdan (wallet) kullanır. Ana şifreleme anahtarını bu şekilde saklamak, yetkisiz kullanımını engeller.

Şekil 3.1: Şeffaf Veri Şifreleme



https://docs.oracle.com/cd/B28359_01/network.111/b28530/asotrans.htm#g1011122

Kurum veritabanında TDE kullanımını incelemek için V$ENCRYPTION_WALLET görünümü sorgulandığında, bu özelliğin kapalı olduğu tespit edilmiştir. Veritabanı güvenliği artırmak için bu özellik aşağıdaki adımları uygulayarak açılmalıdır.

1. TDE için cüzdan yeri belirlenmelidir. $ORACLE_HOME/network/admin altında bulunan sqlnet.ora dosyasına veritabanı cüzdan konumunu belirten WALLET_LOCATION parametresine cüzdan yeri atanmalıdır.
2. Veritabanı sütunlarını şifrelemek veya şifresini çözmek için bir ana şifreleme anahtarı aşağıdaki komutla oluşturulmalıdır.

   ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY kuvvetli_parola
3. Cüzdanı aktif etmek için aşağıdaki komut kullanılır.

   ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY kuvvetl_parola

Bu adımlardan sonra ENCRYPT anahtar kelimesi ile ve varsayılanı AES192 olmak üzere 3DES168, AES128,AES256 kriptolama algoritması anahtar kelimelerini de kullanarak daha güvenli tablolar, indeksler ve tablo alanları oluşturulabilir. Ayrıca ana şifreleme anahtarı güvenlik yöneticisine verilip veritabanı yöneticisi tarafından bilinmemesi sağlanırsa çok hassas verilerin kurum içi personele karşı da korunması sağlanmış olur.

Kurum veritabanlarında TDE tabanlı güvenlik uygulanmadığı için alınan yedeklerin şifresiz bir şekilde disklerde ve yedekleme ünitelerinde barındırıldığı tespit edilmiştir. Bu kurum verilerinin güvenliğini disk ve yedekleme ünitelerindeki güvenlik kadar yapıyor. Saldırganların diskleri veya yedekleme ünitelerini ele geçirse bile kurum verilerinin güvenliğini artırmak için yedeklerin kriptolu bir biçimde alınması gerekmektedir. Bu da yukardaki adımlarla TDE güvenlik cüzdanını aktif ettikten sonra yedek alma scriptine aşağıdaki komut parçacığı eklenerek çok kolay bir şekilde sağlanabilir.

RMAN > SET ENCRYPTON ON;

3.6 Veritabanı Güvenlik Araçları

Bu bölümde veritabanı güvenliğini artırmak için kullanılabilecek araçlar hakkında açıklamalar yapılmıştır.

3.6.1 Database Vault

DV(Database Vault), veritabanındaki belirli alanlara, yönetimsel erişime sahip kullanıcılar da dahil olmak üzere herhangi bir kullanıcıdan erişimi kısıtlar. DV ile kritik ve hassas verilere erişim kısıtlanır. Hassas verilere çeşitli şekillerde FGA

kontrolü uygulayabilmeyi mümkün kılar. DV ile kurum verileri süper ayrıcalıklı kullanıcılardan korunurken onların yapması gereken işleri engellemez. DV aşağıdaki amaçlar için kullanılabilir.

- ✓ Ayrıcalıklı kullanıcıların (DBA) hassas uygulama verilerine erişmesini önleme
- ✓ Güvenilmez ayrıcalıklı kullanıcı hesapların, hassas verileri çalması veya veritabanlarında ve uygulamalarda yetkisiz değişiklikler yapabilmesini önleme
- ✓ Veri tabanında, kimlerin neler yapabileceği ve uygulamaları, verileri ve veritabanlarına ne zaman ve nasıl erişilebileceğini kontrol eden kontroller sağlama
- ✓ En az ayrıcalık modeli elde etmeye yardımcı olmak ve veritabanlarını ve uygulamaları daha güvenli hale getirmek için veritabanındaki tüm kullanıcılar ve uygulamalar için ayrıcalık analizi sağlama

Kurum veritabanları incelendiğinde, veritabanındaki bütün verileri veritabanı yöneticilerinin görebildiği tespit edilmiştir. Kurumda bu tarz bir güvenlik aracı ile, veritabanı kullanıcısına sahip olmayan güvenlik yöneticileri tarafından yönetilerek hassas ve kritik verilerin kurum içi personele karşı korunması sağlanmalıdır.

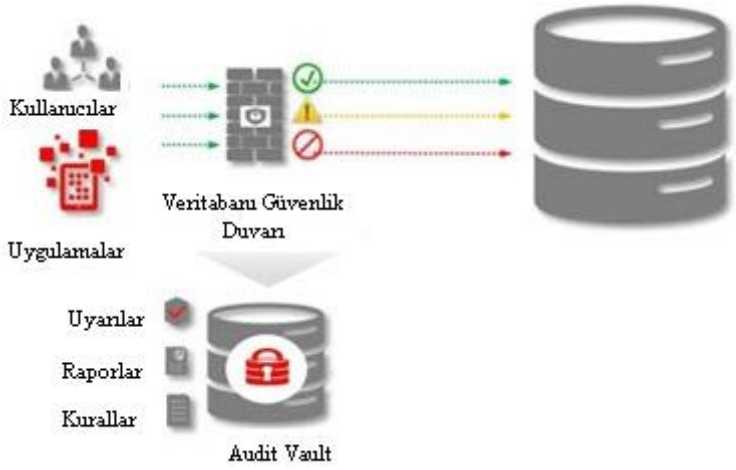
3.6.2 Veritabanı Güvenlik Duvarı ve Denetim Aracı

SQL enjeksiyonu veya ayrıcalıklı kullanıcı hesapları yoluyla artan sayıda veritabanına yapılan olası saldırılar, kullanıcıların veritabanında yaptıkları işlerin izlenmesini güvenlik stratejisinin önemli bir parçası haline getirmektedir. Bu izlemenin değeri, elde edilen bilgilerin düzeyine ve kalitesine, ayrıca mevcut raporlama ve uyarı işlevlerine bağlıdır.

Bu güvenlik aracı, veritabanlarından, işletim sistemlerinden, dizinden, dosya sistemlerinden ve uygulamalardan gelen denetim verilerinin konsolide edilmesi yoluyla kapsamlı ve esnek bir izleme olanağı sağlar. Aynı zamanda, beklenmeyen uygulamaları, SQL enjeksiyonunu ve diğer kötü niyetli kişilerin veritabanına erişmesini önleyerek, ağ üzerinde bir ilk savunma hattı görevi görür. Veritabanından gelen binlerce denetim verilerini konsolide edebilir ve aynı anda SQL trafiğini izleyebilir, yetkisiz veya politika dışı SQL ifadelerini tespit edebilir, onları uyarabilir ve önleyebilir. Özel bir raporlama arayüzü ile ağ üzerinden veya denetim logları

aracılığıyla olsun, kurum genelinde veritabanı aktivitesinin kapsamlı bir görünümünü sağlar.

Şekil 3.2: Veritabanı Güvenlik Duvarı ve Audit Vault



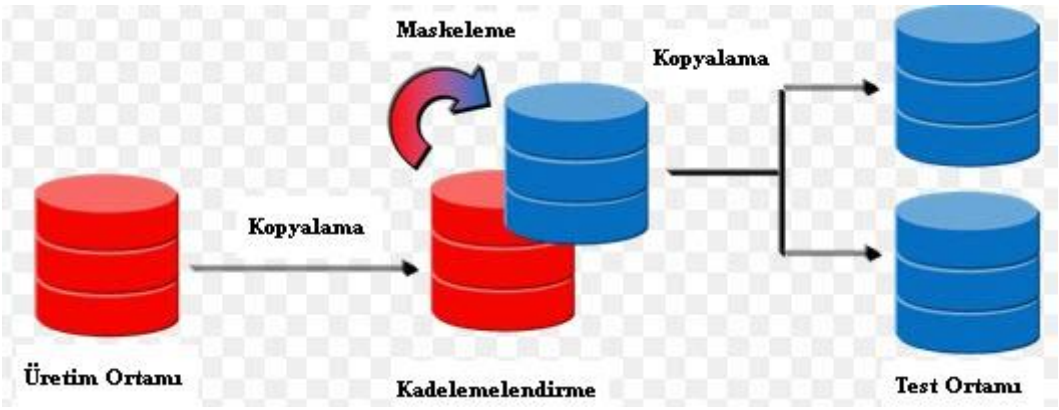
Son zamanlarda yapılan yatırımlarla, kurum bu tarz ürünleri bünyesine katarak veri güvenliğini önemli ölçüde artırmıştır.

3.6.3 Veri Maskeleme Aracı

Test ve geliştirme gibi üretim dışı kullanıma yönelik üretim verilerini kopyalamak güvenlik ve uyum sınırlarını genişleten ve veri ihlali ihtimalini arttıran hassas verilerin çoğalmasını hızlandırmaktadır. Veri Maskeleme araçları, hassas üretim verilerini maskeleyerek alt bölümlere ayıran ve üretim dışı ortamlarda verilerin güvenli bir şekilde paylaşılmasını sağlayan esnek bir çözüm sağlar. Test ve geliştirme ortamları, genel olarak üretim verilerini içeren bir saldırı için potansiyel hedeflerden biridir. Bu gibi ortamlar, aslında üretim sistemleri kadar korunmadığı veya izlenmediği için, bu ortamdaki veriler maskelenmelidir. Veri maskeleme araçları, hassas üretim verisinin, üretim dışı ortamlardaki güvenlik zafiyetlerini azaltarak güvenliği artırır.

Veri maskelemesi yapıldığında verilerin üretim ortamındaki gerçek değerleri değişmez. Sadece bir kopyası maskeleme fonksiyonundan geçirilerek test ortamına aktarımı sağlanır.

Şekil 3.3:Verilerin Maskeleme Yapılarak Üretim Ortamından Test Ortamına Aktarılması



https://dseanoneill.wordpress.com/2012/05/30/datamasking-for-ebs-there-was-much-rejoicing/

Veri maskeleme, kolonları karıştırma, kullanıcının tanımladığı aralıklarda rasgele değerler oluşturma, hassas verilerin bir anahtar kullanarak şifrelenmesi, kullanıcının belirlediği formatlara göre, sütunları belli koşullara göre maskeleme veya kullanıcının oluşturduğu PL/SQL fonksiyonları kullanan özel maskeleme türleri gibi çeşitli şekillerde yapılabilir.

Kurumdaki uygulamaların çoğunluğu yazılım firmalarının dış kaynak personelleri tarafından geliştirilmektedir. Kurum veritabanları incelendiğinde, dış kaynak personeller için temin edilen geliştirme ortamlarının üretim verilerini olduğu gibi maskelenmemiş bir biçimde içerdiği tespit edilmiştir. Bu durumda veriler geliştirme ortamlarından kolaylıkla sızdırılabilir. Kurumda her ne kadar bu personellerle gizlilik sözleşmesi imzalanıyor olsa da, kurum verilerini daha güvenli hale getirmek için veriler geliştirme ortamlarına maskelenerek atılmalıdır.

3.6.4 Veri Replikasyon Aracı

Replikasyon, dağıtılmış bir veritabanı sistemi oluşturan birden çok veritabanında veritabanı nesnelerinin kopyalanması ve korunması işlemidir. Alternatif veri erişim seçenekleri mevcut olduğundan, replikasyon performansı artırabilir ve uygulamanın kullanılabilirliğini koruyabilir. Örneğin, bir uygulama normalde ağ trafiğini en aza indirgemek ve maksimum performans elde etmek için uzak bir sunucu yerine yerel bir veritabanına erişebilir. Ayrıca, yerel sunucu bir arıza yaşarsa, uygulama çalışmaya devam edebilir, ancak replikasyonu yapılmış verilere sahip diğer sunucular erişilebilir kalır. *[13]*

Veri replikasyonu aslında doğrudan güvenlik amacıyla kullanılmaz. Daha çok veriye erişilebilirliğin sürekliliği için kullanılır. Fakat verinin bozulmaması ya da yok olmaması veri güvenliğinden bahsedebilmenin ilk koşuludur. Kurum veritabanlarında yedekler aynı lokasyonda farklı diskler üzerine alınmaktadır. Bu kurumun verileri korumak adına sadece disklerin bozulabileceği ihtimalini göz önünde bulundurduğunu gösteriyor. Ama gerçek hayatta doğal afetler, terör saldırıları gibi felaket ihtimalleri her zaman vardır. Böyle istenmeyen ihtimallere karşı kurum en değerli varlığı olan verisini tamamen kaybetmemek için, veri replikasyon araçları ile verilerini mümkünse farklı bir ilimizde yedeklemelidir.

3.7 Güvenlik Yamalarının Güncellenmesi

Bilişim saldırılarına karşı korunmanın en iyi yollarından biri güvenlik yamalarını düzenli olarak takip edip güncellemektir. Ve bu durum kurumdaki sadece uygulama ve veritabanı sunucuları için değil ağda bulunan bütün bilgisayarlar için geçerlidir. Kurumdaki bir bilgisayarın bile güvenlik yaması güncellenmemesi, kurumun tümünün istikrarını tehdit edebilir ve normal işlevselliği engelleyebilir.

Kurumdaki bu tez kapsamında incelenen merkezi Oracle veritabanlarının ve veritabanını barındıran sunucunun işletim sistemi güvenlik yamaları otomatik olarak güncellenmediği tespit edilmiştir. Kurum veritabanındaki eksik yamalar, veritabanı sağlık monitörü olan ORACHECK aracı indirilip çalıştırılarak tespit edilmiştir. Tablo 3.5'teki eksik güvenlik yamalarının veritabanını daha güvenli hale getirmek için acil olarak yüklenmesi gerekmektedir.

Tablo 3.5 : Yüklenmesi Gereken Veritabanı Yamaları

Yama Numarası	**Yama Açıklaması**
22502505	ACFS Patch Set Update 11.2.0.4.160719
23054319	OCW Patch Set Update 11.2.0.4.160719
23054359	Database Patch Set Update 11.2.0.4.160719

İşletim sistemi seviyesinde yama kontrolü için, aşağıdaki komutların çıktısı alınarak https://www14.software.ibm.com/webapp/set2/flrt/home adresinden eksik yamalar tespit edilmiştir. Veritabanına işletim sistemi seviyesinde yapılabilecek saldırılara karşı önlem alabilmek için Tablo 3.6'daki güncellemeler yapılmalıdır.

#prtconf | grep Model → Server machine –Type Model'ini elde eder

#oslevel –s → Mevcut OS seviyesini elde eder.

Tablo 3.6 : İşletim Sistemi Seviyesinde Yapılması Gereken Güncellemeler

	Input Level	**Update**	**Upgrade**
AIX	7100-01-05	7100-01-10	7100-04-02
Release date	2012.07.18	2014.08.29	2016.05.20
EoSPS	2014.10.31	2014.10.31	2019.12.04
Latest	7200-01-01	7100-01-10	7100-04-03
Latest Release Date	2016.11.11		2016.12.20

Bazı yamalar veritabanı ve işletim sistemi çekirdeği seviyesinde değişiklikler yaptığı için olası risklere önceden hazırlıklı olmak bakımından, yama güncellemesi işletim sistemi ve veritabanı türü ve versiyonlarının özdeş olduğu test ortamlarında denenmeden üretim ortamında güncellenmemelidir. Kurumda yama güncellemeleri ve veritabanı versiyon yükseltmelerinin yapılabileceği üretim ortamına gerek işletim sistemi gerekse veritabanı olarak denk bir test ortamı bulunmadığı tespit edilmiştir.

4. KURUM VERİTABANLARI İÇİN GENEL DURUM ANALİZİ VE ÖNERİLER

Bu bölümde kurum veritabanları ORACHECK, RDA, AWR ile analiz edilip sistemdeki problemler için öneriler aşağıda sıralanmıştır.

- Veritabanı seviyesinde güncel güvenlik paketleri yüklenmediği tespit edilmiştir. Bunlar yüklenmelidir.
- İşletim sistemi seviyesinde güvenlik paketleri yüklü olmadığı tespit edilmiştir, bunlara ait son sürümler yüklenmelidir.
- SGA yeniden boyutlandırma işlemlerinin sıklıkla olması performans düşüşüne neden olmaktadır. Bu durumu engellemek için veritabanı parametreleri olan SGA_TARGET ve MEMORY_TARGET değerleri artırılmalıdır.
- Kurum veritabanında imleçlerin önbellekte paylaşılma sayısını belirten SESSION_CACHED_CURSORS parametresi varsayılan değeri 50'dir. Kurum veritabanlarında eşzamanlı olarak anlık ortalama 300 oturum açılmaktadır. Bu değerin düşük olması beklemelere neden olduğu için

performans düşüşüne neden olmaktadır. Bu parametre daha yüksek değerlere ayarlanmalıdır.

- Şema nesnelerinde yapılan çok sayıda yapısal değişiklik, bu nesneleri içeren SQL deyimlerinin kötü yürütme planına neden olmaktadır. Bu ise performans düşüşüne neden olmaktadır. Uygulama geliştiricileri ve veritabanı yöneticileri bir araya gelerek nesnelerdeki yapısal değişikliklerin miktarını azaltacak çözüm bulmalıdır.
- NLS_SORT parametresi, SQL deyimlerinde standart indekslerin kullanılabilmesi için ve sorguların daha verimli olabilmesi için BINARY olarak ayarlanmalıdır.
- Tanısal veri toplamanın basitleştirilmesi için bir tanı toplama yardımcı aracı olan "TFA Collector " programının yüklü olmadığı tespit edilmiştir. Veri toplamadaki analiz sürecini en aza indirmek için bu araç yüklenmelidir.
- Redo Log yazma hızı performans için eşik değeri olan 500ms'yi aştığı LGWR log dosyaları incelenerek tespit edilmiştir. LGWR yazma gecikmelerinin kullanıcılara yavaşlık olarak yansıdığı görülmüştür. LGWR yazma gecikmeleri veritabanı yedekleri arşive kopyalanırken olmaktadır. Bu durumun önüne geçmek için, yedeklerin uzamasına neden olan büyük objelerin veritabanından alınarak dosya sunucusu üzerinde muhafaza edilebilir. Böylelikle veritabanı yedekleme işlemi kısa süreceği için kopyalama da son kullanıcıların sistemde olmadığı zamanda bitmiş olur. İkinci seçenekte disk yazma ve okuma üniteleri daha performanslı olanlarıyla değiştirilerek yapılabilir.
- İşletim sistemi seviyesinde veritabanı kullanıcısı olan oracle'ın kendisine ayrılan işletim sistemi kaynaklarını kullanma limitleri "uname –a" komutuyla sorgulandığında STACK_SIZE parametresinin 4K gibi düşük bir değerde olduğu tespit edilmiştir. Veritabanı performansının işletim sistemi kaynaklarındaki kısıtlardan dolayı düşmemesi için STACK_SIZE parametresi "unlimited" değerine çekilmelidir.
- İşletim sistemi seviyesinde performans sorunlarının teşhisinde yardımcı olmak için işletim sistemi ve ağ ölçümlerini toplayan OSWatcher'ın çalışmadığı "ps –elf | grep OSWatcher" komutu ile tespit edilmiştir. İşletim

sistemi seviyesinde problemleri analiz edip çözebilmek için OSWatcher izleme aracı indirilip "./startOSWbb.sh" komutu ile çalıştırılmalıdır.

- Geri dönüşüm nesnelerinin çok fazla alan tüketmesi performans problemlerine neden olur. İyi bir performans için geri dönüşüm nesnelerinin toplam büyüklüğü periyodik olarak izlenmeli ve %10'u geçmesi durumunda "purge dba_recyclebin" komutu ile tablo alanları geri dönüşüm nesnelerinden arındırılmalıdır.
- İstatistikler optimize edicinin en iyi çalıştırma planı oluşturmasında kullanılır. Çok sayıda nesnenin istatistiklerinin eski olduğu tespit edilmiştir. Eski istatistikler kötü yürütme planı demek olduğundan düşük performansa neden olur. İstatistiği eski olan nesneler için "exec dbms_stats.gather_object_stats" komutu çalıştırılarak istatistikler güncellenmelidir.
- Optimize edicinin varsayılan olmayan ayarları performans sorunlarına neden olabilir. Optimize edicinin parametreleri değiştirilmemelidir. Kurum veritabanlarında CURSOR_SHARING, OPTIMIZER_INDEX_CACHING, OPTIMIZER_INDEX_COST_ADJ parametrelerinin varsayılandan farklı bir değere atandığı tespit edilmiştir. Bu parametreler "ALTER SYSTEM " ile orijinal değerine atanmalıdır.
- Veritabanı, audit loglarını işletim sistemi seviyesinde AUDIT_FILE_DEST parametresinin işaret ettiği dizinde tutar. Bu loglar bütün işlemlerde oluşmasından dolayı disk probleminden kaynaklı sorun yaşamamak için 30 günden eski loglar silinmelidir. Kurum veritabanında 2012'den itibaren bütün logların bulunduğu tespit edilmiştir. 30 günden eski loglar "find <location> -name "*.aud" -atime +30 | xargs rm -f" komutu ile temizlenmelidir.
- CP(CheckPoint) işlemleri tamamlanamazsa, veritabanı performans düşüşü yaşayabilir veya kilitlenebilir. Kurum veritabanları uyarı logları incelendiğinde CP işleminin tamamlanmadığı anlamına gelen "CHECKPOINT NOT COMPLETE" hatasına 46012 kez rastlanmıştır. Bu durumu önlemek için redo log dosyaları aşağıdaki komutla büyütülmelidir.

  ALTER DATABASE drop LOGFILE group 7;

  ALTER DATABASE ADD LOGFILE thread 2 group 7 ('/backup/redo/CSBRAC/redo0702.log', '/redo/CSBRAC/redo0701.log') SIZE 1G reuse;

- Kurum veritabanında uyarı dosyasının boyutu 3GB olduğu görülmüştür. Uyarı dosyaları 50MB'ı geçince peryodik olarak yenilenip eskisi yedeklenmelidir.
- ASM, RAC mimarileri için tasarlanmış Oracle veritabanına özel dosya depolama yönetimi çözümüdür. Kurum Oracle veritabanları ASM disk yönetimine geçerek performans artışı sağlayabilir.

SONUÇ

Bu tez çalışması içerisinde bakanlığımızda en yaygın kullanılan ve en büyük veritabanları olan Merkezi Oracle Veritabanları (CSBRAC) ele alınmış olup, kurum veritabanlarının performans parametreleri ve güvenlik kriterleri açısından incelenip, potansiyel güvenlik risklerinin belirlenerek ve performans iyileştirmesi önerileri üretilerek kurumun vermiş olduğu hizmet kalitesinin artırılması hedeflenmiştir.

Veri tabanı sistemlerinin kurulumu, konfigürasyonu, tasarımı, sorgulaması, güvenliği değişen ve artan ihtiyaçlara göre ayarlanamazsa veritabanı performans ve güvenlik problemleri oluşması kaçınılmazdır. Bu bağlamda kurum veritabanı Oracle veritabanlarındaki olası sorunların proaktif olarak taranmasını sağlayan analiz aracı ORACHECK ile Kurum Merkezi Oracle Veritabanlarının işletim sistemi, veritabanı konfigürasyonu seviyesinde ve üzerinde çalışan SQL deyimleri dahil analizi yapılmıştır. Ve bu araca göre kurum veritabanlarının sistem sağlık skoru 100 üzerinden 87 olmuştur. Bu tez çalışması boyunca genel olarak aşağıdaki sorunlar tespit edilmiş ve bunlara ilgili bölümlerde çözüm aranarak öneriler sunulmuştur.

- ✓ Veritabanı tablo tasarımında normalizasyon kurallarına dikkat edilmemesi.
- ✓ İndekslerin doğru kullanımı sorguların hızını önemli miktarda artırabilir. Fakat yanlış kullanımı DML işlemlerinde yavaşlığa neden olabilir. İndeks kullanılmaması, ihtiyaçtan fazla indeks kullanılması veya yanlış tipte indeks kullanılmasına çok sık rastlanması.
- ✓ Çok büyük tablolar için uygulamaların performansını artıran tablo bölümlemesinin yapılmaması
- ✓ Çok büyük nesnelerin sıkça kullanımından kaynaklı yedek sürelerinin uzaması ve gün içine sarkarak disk yazmasında yavaşlık yaşanması
- ✓ Daha fazla alan tasarrufu ve daha hızlı performans için sıkıştırma ve tekilleştirme gibi akıllı depolama özellikleri ve ek güvenlik özellikleri olan LOB nesnelerine alternatif olarak çıkan SecureFile LOB'larının kullanılmaması
- ✓ Kurumda raporlar için ayrı bir veri ambarı olmamasından dolayı raporların karmaşık görünümlerle alınmaya çalışılması

- ✓ Çalışma zamanında derlenme ihtiyacı gerektirmeyen, uygulamanın bütünlüğünü ve tutarlılığını geliştiren, kodlama hataları olasılığı azaltan prosedürlerin yeterince kullanılmaması.
- ✓ Yüksek kaynak tüketen SQL deyimlerini tespit eden ve daha verimli çalışabilmesi için öneriler sunabilen veritabanı araçlarından faydalanılmaması.
- ✓ Kötü yazılmış çok fazla kaynak tüketen bir SQL deyiminin veritabanında diğer herkesin çalışmasını güçleştirerek performans sorunlarına neden olmasını önleyebilen gruplara göre kaynak önceliği ayrımını sağlayan Kaynak Yöneticisinin kullanılmaması
- ✓ Veritabanı Kurulumunda oluşturulan varsayılan kullanıcıların kilitli olsa bile şifrelerinin değişmemiş olmasının oluşturduğu güvenlik zafiyetleri
- ✓ Veritabanı port bilgisinin varsayılanda kalmasının oluşturduğu güvenlik zafiyetleri
- ✓ Ayrıcalıklı yetkilere sahip kullanıcıların oluşturduğu güvenlik zafiyetleri
- ✓ Veritabanı yedeklerinin şifresiz bir şekilde disklerde ve yedekleme ünitelerinde barındırılmasından oluşan güvenlik zafiyetleri
- ✓ Veritabanındaki bütün verileri veritabanı yöneticilerinin görebilmesini engelleyen güvenlik araçlarının olmamasından dolayı hassas ve kritik verilerin kurum içi personele karşı korunmasız olması
- ✓ Verilerin test ortamına maskelenmeden atılmasından dolayı oluşan güvenlik zafiyetleri
- ✓ Felaket durumlarına karşı kurumun verisinin başka bir lokasyonda yedeklenmiyor oluşu
- ✓ Bilişim saldırılarına karşı korunmanın en iyi yollarından biri güvenlik yamalarının düzenli olarak hem veritabanında hem de sunucunun işletim sistemine uygulanmamasının oluşturduğu güvenlik zafiyetleri

Veritabanı mimarisi, yalnızca veritabanı yöneticilerinin alanı değildir. Yanıt süresi düşük, performanslı ve etkin uygulamalar geliştirmek isteyen uygulama geliştiricilerin veritabanının yapısal ve işlevsel özelliklerini göz önünde bulundurarak geliştirme yapması gerekir. Örneğin, veritabanı eşzamanlılık kontrollerini ve çoklu

düzenleyici okuma tutarlılığını dikkate almadan uygulama geliştirme yapmak, bir uygulamanın verilerin bütünlüğünü bozabilir, yavaş çalışabilir ve ölçeklenebilirliği düşürebilir. Bu tez çalışmasında ayrıca veritabanı temel ilişkisel veri yapıları, veritabanı performansı doğrudan etkileyen nesnelerin kullanımı, işlem yönetimi, depolama yapıları ve mimarisi açıklanmıştır.

Bu çalışma yalnızca kurumdaki Oracle veritabanları için yapılmıştır. Performans ve Güvenlik çalışmaları uygulamada veritabanın tipine göre farklılık gösterir. Benzer bir konuda çalışma yapılmak istenirse aynı konu kurumdaki yaygın olarak kullanılmakta olan MSSQL veritabanları için çalışmalar yapılabilir.

KAYNAKLAR

1) ASHDOWN Lance,KYTE Tom ,2015, Database Concepts 11g Release 2 (11.2)
2) BRYLA Bob, LONEY Kevin, 2008, Oracle Database 11g DBA Handbook , Oracle Press
3) CHAN Immanuel, ASHDOWN Lance, 2014, Oracle Database Performance Tuning Guide 11gRelease 2 (11.2)
4) FOGEL Steve, 2015, Oracle Database Administrator's Guide 11g Release 2 (11.2)
5) HELLSTRÖM Ian, 2016, Oracle SQL & PL/SQL Optimization for Developers Documentation Release 2.1.1
6) HUEY Patricia, 2014, Database Security Guide 11gRelease 1 (11.1)
7) INGRAM Aaron, SHAUL Josh, 2007,Practical Oracle Security
8) Oracle White Paper, 2004, LOB Performance Guidelinees
9) Oracle White Paper, 2009, Oracle Database 11g: SecureFiles
10) Oracle White Paper, 2012, Understanding Optimizer Statistics
11) RICH Bert, 2012, Oracle Database 2 Day DBA 11gRelease 2 (11.2)
12) Watson John, RAMKLASS Roopesh, 2010, OCA/OCP Oracle Database All in One, Oracle Press
13) https://docs.oracle.com/cd/A64702_01/doc/server.805/a58227/ch_repli.htm (22.02.2017)
14) http://docs.oracle.com/cd/B10500_01/server.920/a96524/c20paral.htm (03.01.2017)
15) https://docs.oracle.com/cd/B19306_01/network.102/b14213/listener.htm (16.12.2016)
16) https://docs.oracle.com/cd/B28359_01/appdev.111/b28393/adlob_intro.htm#i1009693 (27.12.2016)
17) https://docs.oracle.com/cd/B28359_01/network.111/b28530/asotrans.htm#g1011122 (16.02.2017)

18) https://docs.oracle.com/cd/B28359_01/network.111/b28531/authorization.htm#DBSEG124 (11.02.2017)
19) http://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#CHDIBFGA (13.01.2017)
20) https://docs.oracle.com/cd/E11882_01/appdev.112/e18294/adlob_smart.htm#ADLOB45957 (09.01.2017)
21) https://docs.oracle.com/cd/E11882_01/network.112/e41945/net_arch.htm#NETAG212 (19.12.2017)
22) https://docs.oracle.com/cd/E11882_01/server.112/e10575/tdpsg_auditing.htm#TDPSG50511 (21.02.2017)
23) https://docs.oracle.com/cd/E11882_01/server.112/e10575/tdpsg_user_accounts.htm#TDPSG20000 (10.02.2017)
24) https://docs.oracle.com/cd/E11882_01/server.112/e10897/storage.htm#ADMQS12048 (15.12.2016)
25) http://docs.oracle.com/cd/E25178_01/server.1111/e25789/process.htm (18.12.2016)
26) https://docs.oracle.com/cd/E11882_01/server.112/e40402/initparams075.htm#REFRN10307 (11.01.2017)
27) https://docs.oracle.com/cd/E11882_01/server.112/e40540/physical.htm#CNCPT1082 (21.12.2016)
28) https://docs.oracle.com/database/121/TGSQL/tgsql_intro.htm#TGSQL113 (22.01.2017)
29) http://www.orafaq.com/wiki/EZCONNECT (06.01.2017)
30) https://en.wikipedia.org/wiki/Database_normalization (03.01.2017)
31) https://en.wikipedia.org/wiki/Database (04.12.2016)

ÖZGEÇMİŞ

1981 yılında Kırşehir'de doğdu. Orta öğrenimini Kırşehir Hacı Fatma Erdemir Anadolu Lisesinde, lise öğrenimini Kırıkkale Fen Lisesi'nde tamamladı. 2007 yılında Yıldız Teknik Üniversitesi Elektrik Elektronik Fakültesi, Bilgisayar Mühendisliği Bölümü'nden mezun oldu. Askerliğini 2007 yılında Erzincan'da yaptı. 2008 yılında Hudut Sahiller Sağlık Genel Müdürlüğünde görev yaptı. 2010 yılında Sağlık Bakanlığı Bilgi Sistemleri Genel Müdürlüğünde görev yaptı. 2014 yılında Çevre ve Şehircilik Bakanlığı'nda Çevre ve Şehircilik Uzman Yardımcısı olarak göreve başladı. Evli ve bir çocuk babasıdır.

ETİK KURALLARA UYGUNLUK BEYANI

Uzmanlık tezi olarak sunduğum bu çalışmayı, bilimsel ahlak ve geleneklere aykırı düşecek bir yol ve yardıma başvurmaksızın yazdığımı, yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu, bunlardan her seferinde değinme yaparak yararlandığımı ve Çevre ve Şehircilik Uzmanlığı Yönetmeliğine uygun olarak hazırladığımı belirtir, bunu onurumla doğrularım.

Çevre ve Şehircilik Bakanlığı tarafından belli bir zamana bağlı olmaksızın, tezimle ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara katlanacağımı bildiririm.

06.03.2017

Raşit ÖZCAN