Oracle DBA Eğitimi

Devinim Yazılım Eğitim ve Danışmanlık

www.devinim.com.tr

İÇİNDEKİLER

1. Amaçlar....1
2. Bir DBA'nın (Veri Tabanı Yöneticisi'nin) Görevi Nedir?....1
3. Oracle'ın Fiziksel ve Mantıksal Yapısı....2
4. Oracle Veritabanı....4
4.1. SGA (System Global Area)....5
Shared Pool....5
Library Cache ve Data Dictionary Cache....6
Database Buffer Cache....6
Redo Log Buffer Cache....7
4.2. PGA (Program ya da Process Global Area)....7
Oracle'da Proces'ler....8
Background Process'ler....8
5. Oracle Kurulumu, Veritabanına Bağlanma ve Güvenlik Ayarları....9
Oracle Veritabanı Oluşturma....10
Oracle Veritabanını Açma....12
Veritabanı'nı Kapatma....16
6. Data Dictionary ve Önemli View'lar....17
7. Controlfile....19
initSID.ora ile Control File Çoklaması....20
SPFILE ile Control File Çoklaması....20
8. Redo Log File....21
9. Tablespace'ler ve Datafile'ler....24
9.1. Data Block'ları....24
9.2. Extent'ler....26
9.3. Segment'ler....27
9.4. Tablespace'ler....27
Tablespace Boyutunu Büyütmek....33
İlgili view'lar, tablolar....36
10. Undo İşlemlerinin Yönetimi....38
11. TABLO VE INDEX'LER....41
11.1. Tablolar....41
11.2. ROWID, UROWID....45
11.3. Indeksler....49
12. Güvenlik....52
12.1. İşletim Sistemi Güvenliği....52
12.2. Veritabanı düzeyinde erişim kısıtlama....53
13. ORACLE12C ile GELEN YENİ ÖZELLİKLER....57
Oracle Multitenant....57
Aktif Bir Datafile'ın Online Olarak Yeniden Adlandırılması Ve Lokasyonunun Değiştirilmesi. .59
Online Migration Of Table Partition Or Sub-Partition....59
Invisible Columns....60
Multiple Indexes On The Same Column....60
DDL Logging....61
Temporary Undo....61

Backup Specific User Privilege....61
Sql Statements In RMAN....61
Table or Partition Recovery In RMAN....62
Restricting PGA size....62
Adding Multiple New Partition....63
Data Pump Geliştirmeleri....63
Redo log oluşumunun engellenmesi....63
Viewlerin tabloya dönüştürülmesi....63
ASM Geliştirmeleri....64
Flex ASM....64
Yükseltilmiş ASM limitleri....64
ASM Disk Scrubbing....64
Grid Infrastructure Geliştirmeleri....64
Flex Clusters....64
OCR backup in ASM disk group....66
14. IN-MEMORY DATABASE ÖZELLİĞİ....66

1. Amaçlar

Bu belge ile aşağıdakileri öğrenecek ve bir DBA olarak sorumluluğunuzdaki veritabanlarının yönetimini gerçekleştirebileceksiniz.

- Oracle Alt Yapısını Tanıma,
- Oracle Veritabanlarında Bağlanma, Açma, Kapama, İzleme İşlemleri,
- Kullanıcı Açma, Silme, Yetki Tanımlama, Roller Tanımlama,
- Control File, Data File, Redo Log File, Tablespace, Tablolar, Indexlerle Çalışma, Yaratma, Silme ve Kontrolleri,
- Yer Büyütme, Sorun Saptama ve Sorun Giderme Çalışmaları.

2. Bir DBA'nın (Veri Tabanı Yöneticisi'nin) Görevi Nedir?

Bir DBA olarak, sorumluluğunuzdaki veritabanlarının sağlıklı çalışması, sorun çıkması durumunda minimum (çoğu durumda sıfır) hata ile geri dönülmesi, güvenlik gibi çalışmaları yürütüyor olmanız beklenir. DBA olmak oldukça zaman gerektiren ve bir o kadar da dikkatli olmayı gerektiren bir iştir. DBA olarak sizden şunlar beklenmektedir:

I. Uygun Bir Donanım Seçimi
II. Oracle'ın Kurulumu
III. Bir Veritabanın Tasarımı ve Yaratılması
IV. Fiziksel ve Mantıksal Alanların Kurulumu
V. Güvenlik Ayarlamaları
VI. Kurtarma Operasyonları ve Hazırlığı
VII. Ağ Yapılandırmaları
VIII. Sistem İzlenmesi
IX. Veritabanı Tuning İşlemleri

Tüm bunları hak ettiği şekilde yapabilmek için Oracle'da 5 yıllık bir sürenin geçmesini beklemek yanlış olmayacaktır.

3. Oracle'ın Fiziksel ve Mantıksal Yapısı

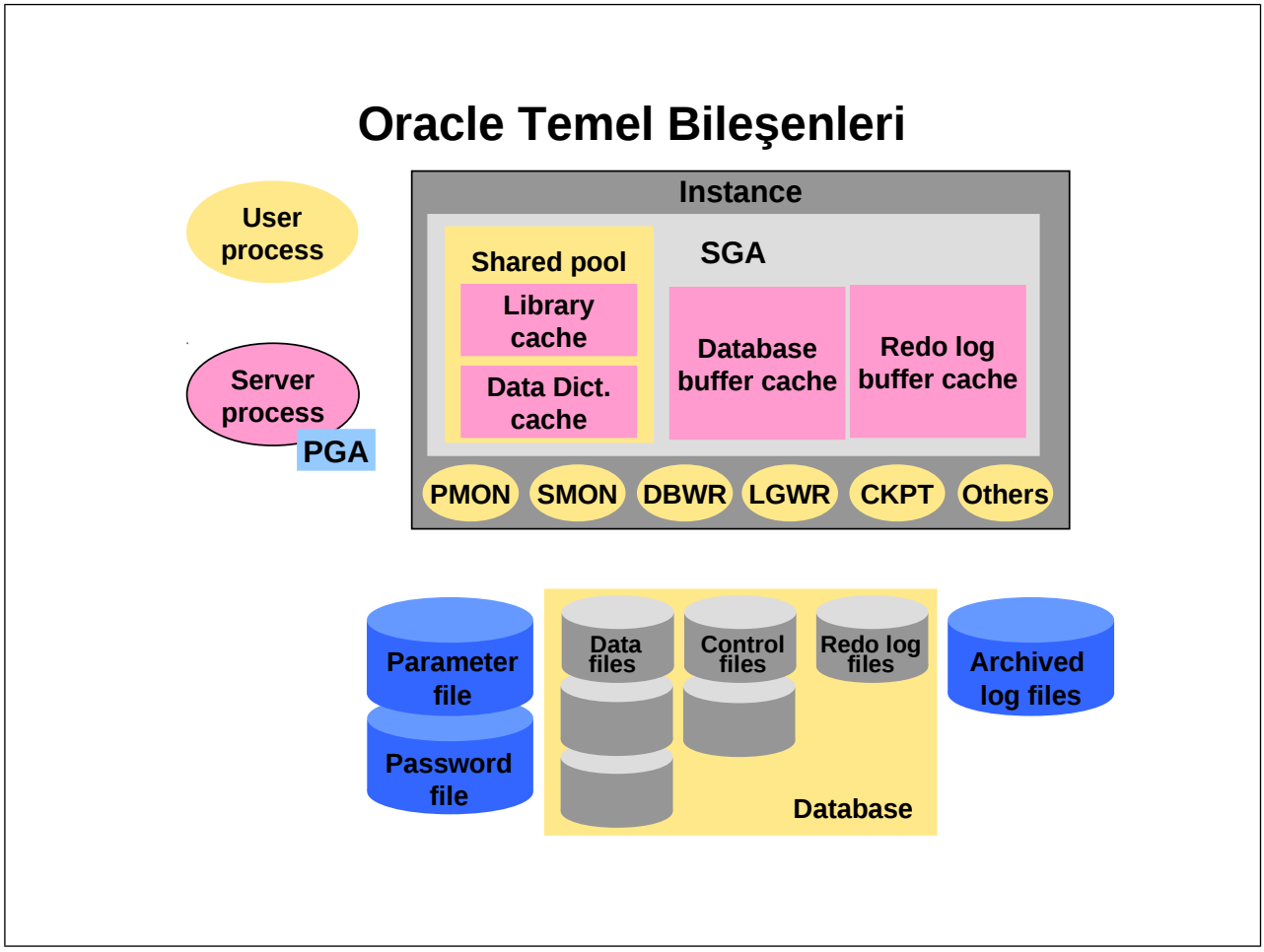
Oracle'da iki kavram birbirine karışmaktadır. Bunu burada hemen belirterek bu karışıklığı önleyelim.

Oracle Server dediğimiz yapı, Oracle'ın çalışması için gerekli olan tüm yapılardır. Bunlar içinde binary dosyalar, belgeler, parametreler vb bulunur.

Oracle Instance'ı ise artalanda (background) çalışan işlemler (process'ler) ve bellekte yapılan işlemleri barındırır. Server'dan gelen bir komutun ya da otomatik çalışacak bir komutun nereye gideceği ve hangi process ile yürütüleceği Instance ile çözülür.

Oracle Veritabanı ise, fiziksel yapıyı ifade etmektedir. Data Files, Control Files, Log Files gibi tüm yapılarımız bir veritabanını oluşturmaktadır.

Instance ve Veritabanı, Oracle Server içinde yer almaktadır.
Bir sunucu üzerinde tek bir Oracle Server olabilir ama istediğiniz kadar ve sistemin elverdiği kadar veritabanı kurabilirsiniz.



Yukarıdaki şekil Oracle belgelerinden alınmış bir şekil olup bir Oracle Server'ı içindeki tüm temel bileşenleri göstermektedir.

Bir Oracle Instance'ı varolan bir veritabanını yönetmek için gerekli background process'leri ve SGA (System Global Area)'dan oluşmaktadır. Her bir instance yalnızca bir veritabanına karşılık gelmektedir.

Bir veritabanına bağlanıp SQL komutları göndermeden önce bir instance'a bağlanmak gerekir. Böylece aynı sunucuda bulunan farklı veritabanlarında yanlış bağlantı yapıp yanlış komut göndermenin önüne geçilmektedir. Bir araçla (sqlplus, Forms, PHP kodu, 3.parti yazılımlar) ile Oracle Server'a bağlantı kurarsınız. Bunun için gerekli bağlantı ayarları önceden yapılmış olmalı ve kullanıcı adı /şifre tanımlamalarından da geçmiş olmalısınız. Bağlantı kurulur kurulmaz, sunucuda işletim sistemi düzeyinde bir process açılır. Bu process, sizin bağlantınız var olduğu sürece ya da sunucuda bir hata oluşana kadar kalmak durumundadır. İşletim sistemi düzeyinde kill gibi

komutlarla bu process'i öldürürseniz bağlantınız da kopacaktır.

Bağlantı 3 şekilde olabilir.

- Doğrudan sunucu üzerinde ilgili araçlarla,
- Kişisel ve uzaktaki bir bilgisayardan network aracılığı ile,
- Yine uzaktan, bir bilgisayardan ara bir uygulama katmanına ve buradan da veritabanına bağlantı ile (3-katmanlı mimari)

Bağlantı sonucunda veritabanı düzeyinde Session adı verilen özel bir yapı kurulur. Kullanıcı, Oracle Server'a bağlandığında özel bir numara (Session Number) oluşur ve bağlantı kopana kadar bu numara açık kalır.

Not: İstenmeyen ya da takılı kalmış işlemleri veritabanı düzeyinde sonlandırmak mümkündür. Bu işlem için ilgili session'ın ne olduğu bulunur ve sonra bir komutla öldürülür.

4. Oracle Veritabanı

Oracle Veritabanı, diğer tüm veritabanlarında olduğu gibi belirli işlerle ilgili bilgileri tablo adını verdiğimiz ortamlarda tutmak ve gelen sogrulara yanıt vermek gibi temel amaçla yaratılırlar.
Veriler, DataFiles denilen fiziksel (diskte tutulan) yapılarda durur. Redo Files, veritabanına yönelik yapılan işlemlerin tutulduğu dosyalardır. Control Files ise veritabanın bütünselliğini koruyan ve yapısını bildiren dosyalardır. Bu üçü, Oracle veritabanın Fiziksel yapısını oluşturur. Bunun dışında hem fiziksel hem de mantıksal yapılar gerekir. Parametre dosyaları (SGA gibi ne kadar bellek tutulacağını, nelere izin verileceğini vb), arşivlenmiş log dosyaları ve kullanıcı şifre dosyaları gibi dosyalar bir Oracle veritabanının olmazsa olmazlarıdır.

Oracle'ın Fiziksel Yapısı; parametre dosyaları, arşivlenmiş log file vb. içerse de temelde datafiles, control files ve redo log files'lardan oluşmaktadır.
Mantıksal Yapı ise iki bölümden oluşur.

1. SGA (System Global Area) ve
2. PGA (Program Global Area)

4.1. SGA (System Global Area)

SGA, bir instance başladığında oluşur ve birçok bellek yapısını barındırır: Shared pool, Buffer Cache, Redo Log Buffer, Large Pool, Java Pool gibi.

SHOW SGA; komutu ile ne kadarlık bellek yeri tanımladığınızı görebilirsiniz:

```
SQL> SHOW SGA;

Total System Global Area  408174752 bytes
Fixed Size                   73888 bytes
Variable Size             356638720 bytes
Database Buffers           40960000 bytes
Redo Buffers               10502144 bytes
```

Bu sorgu, toplam 408.174.752 byte'lık bir bellek alanının veritabanı proseslerince paylaşılacağını söylemekte. Bu sunucunun Virtual Memory alanını kullanmaktadır ve veritabanı instance'ı açıldığında okuduğu parametreler üzerine yer tutmaktadır. Bu nedenle sunucunun fiziksel özelliklerini çok iyi bilmeniz gerekir.

SGA'nın boyutu en çok aşağıdaki parametrelerden etkilenir:
DB_CACHE_SIZE, LOG_BUFFER, SHARED_POOL_SIZE ve LARGE_POOL_SIZE.

Oracle 9i ile birlikte, bazı komutlar dinamik olarak (veritabanı açıkken) verilmeye başlandı. ALTER SYSTEM ... komutu ile izin verilen alanlarda alan büyütme ya da küçültme yapmak mümkün. SGA de dinamik bir yapıya kavuştu. Granül denilen bir yapı ile bellekte yer ayırmak mümkün. Buffer cache, large pool ve shared pool alanları, sınırlarına göre granül yapısında büyüyebiliyorlar. SGA içerisindeki önemli yapılara bakalım:

Shared Pool

Shared Pool, SGA içerisinde yer alan ve en son çalıştırılan SQL cümlerinin ve veri bilgilerinin son halini tutmaktadır. İçerisinde iki önemli yapı bulunmaktadır:

- Library Cache
- Data Dictionary Cache

Shared Pool'un ne kadar yer tutacağı, başlangıç parametre dosyası initSID.ora'daki SHARED_POOL_SIZE parametresi ile belirlenir. 9i ile birlikte veritabanı kapatılmadan

ALTER SYSTEM SET SHARED_POOL_SIZE = xx M;

komutu ile değiştirilebilir. Burada verilecek değer sonucu oluşacak SGA boyutu, SGA_MAX_SIZE'da belirtilen boyutu geçemez.

Library Cache ve Data Dictionary Cache

Library Cache, Shared Pool içerisinde bulunan ve son çalıştırılan SQL ve PL/SQL komutlarının tutulduğu bölgedir. Library Cache, LRU (Least Recently Used) algoritmasına göre çalıştığı için az kullanılan SQL komutları ayrılan bellek alanında yer kalmayınca silinir ve bunun yerine yeni sql komutu yazılır. Bu alanın küçük olması, hem silme, yazma işlemlerinin çokluğu hem de hazır kullanılabilecek SQL komutları yerine bunları tekrar tekrar çalıştırması nedeniyle performans açısından sorun oluşturur. Bu yüzden Shared Pool'u küçük tutmamak oldukça önemlidir.

Data Dictionary Cache ise tablolar, indeksler, veri dosyaları, yetkiler gibi bilgileri tutar. Herhangi bir SQL komutu gelip çalışmaya başladığında, daha öncesinde tüm yapısal bilgiler tutulduğu için çalışma daha hızlı olacaktır.

Data Dictionary ve Library Cache alanlarının ne kadarlık yer tutacağını Oracle Server'ı, shared pool'daki yere göre otomatik olarak ayarlar.

Database Buffer Cache

LRU algoritmasıyla çalışan bu yapı yine performans açısından önemlidir. Herhangi bir SQL komutu geldiğinde, eğer database'e yazılması gereken bir iş varsa Oracle Server tarafından Database Buffer Cache'e bakılır ve yeterli yer yoksa (block), datafile'dan gerektiği kadar block okunarak kendisinde tutar (yazar). Böylece aynı tablolar için istenen block doğruca fiziksel yapı olan datafile'dan okunmak yerine, bellekten okunur ve işlemler çok daha hızlı yapılabilir.

Not: Fiziksel bir dosyadan okumak çoğu zaman bellekten okumaya göre çok yavaş olacaktır.

Database Buffer Cache de dinamik olarak değiştirilebilir. Bunun için,

```
ALTER SYSTEM SET DB_CACHE_SIZE = xx M;
```

komutunu vermek yeterli olacaktır.
Database Buffer Cache'de ne kadarlık yer kullanılması gerektiğini kolaylaştıracak bir paramatre bulunmaktadır. DB_CACHE_ADVICE adlı bu değişken (ON, OFF ve READY değerlerini alabilir) ile başlangıç dosyası initSID.ora'ya ekleyerek (ON yaparak) bir DBA olarak karar verme sürecini kolaylaştırabilirsiniz. Bu değişkeni ON yaptıktan sonra, V$DB_CACHE_ADVICE adlı view'dan sonuçları görebilirsiniz.

Redo Log Buffer Cache

Bu alanın en önemli görevi, sistemde bir sorun çıkması halinde Kurtarma için gerekli olan ve datafile'lara yazılı olan bilgileri tutmaktır. Datafile'a yazılacak bilgiler (Insert, Delete vb) Buffer Cache'de tutulur ve burası dolunca da daha sonra Redo Log File denilen fiziksel dosyalara yazılır.

Büyüklüğü LOG_BUFFER ile tanımlanır ve bununla birlikte Log File'lar ciddi performans sorunlarına yol açabilir.

Insert, Delete, Update, Drop, Alter, Create komutları sonucu yapılan tüm işlemler burada tutulur.

Her Oracle Server'ında bulunan önemli yapılardan söz ettik. Bunlar dışında eğer Java kurulup çalıştırılacaksa Java Pool ve Shared Server kurulacaksa Large Pool alanları gerekecektir. Sırasıyla, JAVA _POOL_SIZE değişkeni ve LARGE_POOL_SIZE değişkenleri ayrılacak yerin büyüklüğünü belirtmenize yararlarlar.

4.2. PGA (Program ya da Process Global Area)

PGA da bellek yapısında bulunan ve artalanda çalışan process'leri kontrol eden ve yöneten yapıdır. Kullanıcıların session bilgileri ve yetkileri, sıralama yapılacaksa sort area, çalıştırılan sql'lerin durumunu bildiren ve tutan cursor area gibi bilgi kümeleri PGA içerisinde tutulmaktadır.

Oracle'da Proces'ler

Oracle Server'unda çalışan işlemleri 3 grupta toplayabiliriz: Oracle Instance'ı başladığında oluşan background process'ler; bir veritabanı kullanıcı Oracle Server'a bağlanmak istediğinde oluşan Kullanıcı Prosesleri (User Processes); bir kullanıcı bir session oluşturduğunda Oracle Instance'a bağlanan işlemin oluşturduğu Server Proses'i.

Yapı şöyle çalışır. Bir kullanıcı, bir sql komutunu veritabanına göndermek istediğinde sqlplus gibi bir araca bağlanır. Bu araca bağlanma anı User Process'in çalışma anıdır. Daha sonra, bu kullanıcı için bir session yaratılır ve sunucu tarafında bu istekleri alıp Oracle Server'a ve ilgili veritabanına taşıyacak Server Process'i çalışmaya başlar. Kullanıcı bağlı kaldığı sürece bu proses de açık kalacaktır. Sistem çökmesi durumunda ya da ağ problemleri sırasında bu bağlantı kopabilir.

Not: Unix/Linux'larda ps –fe komutu ile gördüğünüz oracleSID prosesleri server prosesleridir. Bunları kill komutu ile öldürmeniz, kullanıcının bağlantısını koparır, bu yüzden dikkat edilmesi gerekir.

Background Process'ler

Yukarıda belirtilen prosesler dışında, Oracle Instance başlar başlamaz oluşan prosesler de vardır. Bunlardan, DBW, PMON, CKPT, LGWR, SMON ve RECO prosesleri her instance'da olması zorunlu olan proseslerdir. ARC, LCK, LMON gibi prosesler ise durumu göre oluşurlar.

DBWn: Database Writer, Database Buffer Cache'de yazılı olan bilgileri datafile'lara yazmakla sorumludur. Duruma göre 1 ya da 1'den çok olabilir. DBW, şu durumlarda devreye girer ve datafile'lara bilgiyi yazdırır:

- ★ Tablespace'lerin yedeklenmesi, offline durumuna getirilmesi, read only yapılırsa,
- ★ Bir tablonun drop ya da truncate komutu ile silinmesi/içinin boşaltılırsa,
- ★ Checkpoint'lerde,
- ★ Timeout oluşursa,
- ★ Dirty buffers belirli düzeye ulaşırsa.

LGWR: Log Writer, Redo Log Cache Buffer'daki bilgileri Redo Log File'lara yazmakla görevlidir.

Trancaction commitlerinde, Redo Log Cache'in 3'te 1 dolduğunda, her 3 saniyede, 1 MB'tan fazla bilgi Redo Cache'te olduğunda ve DWB datafile'lara yazmadan önce çalışır ve Redo Log File'lara yazar. Bir başka görevi ise, DBWn tetiklemek ve onu datafile'a yazmaya zorlamaktır.

SMON: System Monitor, Oracle Server'ındaki en önemli görevlerden birini üstlenir. Sunucudaki bir sorun nedeniyle Instance çalışmaz duruma gelirse, SGA'deki bilgiler kaybolacaktır. Bunu önlemek ve commit edilmemiş bilgileri rollback yaparak veri bütünlüğünü sağlamak SMON'un görevidir. Sunucu yeniden açıldığında, datafile'lara yazılmamış ama hala SGA'de bekleyen verilerin yazılması işini üstlenir. Veritabanı instance'ı açılırken, Instance Recovery denilen olay gerçekleşir ve SMON, online redo log file'lardaki bilgiyi okuyarak yazılmamış işlemleri datafile'lara yazmaya başlar. Bu aşamalar bittikten sonra kullanıcılar sistemi kullanır duruma geleceklerdir.

PMON: Process Monitor, başarısız prosesleri bulup öldürmekle görevlidir. Lock olmuş tabloları kurtarmak, kullanıcı tarafından kullanılan kaynakları temizlemek şeklinde yapar bu görevi.

CKPT: Checkpoint, DBW prosesi SGA'deki tüm bilgileri datafile'lara yazdığında oluşur. Checkpoint anında, tüm datafile'ların header'ına Checkpoint'le ilgili bir bilgi yazılır. Aynı bilgi controlfile'lara da yazılır. Herhangi bir sistem çökmesi durumunda Instance Recovery yapılırken, bu checkpoint bilgilerinin hepsinin aynı olması kontrol edilir. Böylece farklı datafile'lar ya da yapı bilgisi nedeniyle veri bütünlüğü sorunun önüne geçilmektedir.

ARCH: Archiver, veritabanı arşiv modda çalıştırılıyorsa, dolmuş olan Online Rede Log File'ları diskte belirtilen bir alana (ya da bir tape'e) yazmakla görevli proses'dir. Yazma işlemi loglar arasında online olma durumunda oluşur (switch).

5. Oracle Kurulumu, Veritabanına Bağlanma ve Güvenlik Ayarları

Oracle Server kurulumu, Oracle Universal Installer (OUI) denilen bir araçla yapılmaktadır. İlerleyen zamanlarda farklı ürünleri kurmayı ve kaldırmayı, sürüm yükseltme işlemlerini de bu araçla yapıyor olursunuz. Oracle'la birlikte gelen CD'den ./runInstaller diyerek çalıştırabileceğiniz gibi dilerseniz diskte bir yere koyarak, ileride de çalıştırabilirsiniz. Java tabanlı bu uygulama ile oldukça kolay biçimde kurulum yapmak ve bir örnek veritabanı oluşturmak mümkündür.

Önemli birkaç kavram şudur kurulum sırasında. Oracle Server dosyaları (binary'ler, parametreler vb) nereye kurulsun. ORACLE_HOME değşikeni ile bu sağlanır. Veritabanı adı ne olsun, ORACLE_SID ile sağlanır. Bunları işletim sistemi düzeyindeki .profile dosyasına da yazarak her seferinde elden girmenize gerek kalmaksızın kullanabilirsiniz.

Oracle firması, OFA adını verdiği Optimal Flexible Architecture adını verdiği yapıda Oracle Server ve Oracla Database dosyalarının tutulmasını önermektedir. Buna göre, binary dosyalar ve ilgili tüm dosyalar ORACLE_BASE altında olacak, veritabanı dosyaları ise (datafiles, control files, redo files) oradata denilen bir yapı altında olacak.

Oracle Veritabanı Oluşturma

Yeni bir veritabanı yaratmak için önce fiziksel yerinizin yeterli olup olmadığını kontrol etmelisiniz. Diskte yeterli yer var mı? Farklı bir instance var ve 2. kuracaksınız bunun için yeterli bellek var mı? gibi sorular ilk akla gelenler olmalı.

Yeni bir veritabanı yaratırken önemli olan birkaç nokta var. Control file'ların mutlaka birden çok kopyası olması, Redo Log Filer'ların birden çok kopya ve grupta olması ve mümkünse bunların farklı fiziksel disklerde olması gibi. Bu durum, beliritilen dosyaların tek olması ve fiziksel bir sorunla karşılaşılması durumunda Instance açılamayacağı için oldukça önemli bir konudur. Bunun dışında performansı artırmak adına datafile'ları farklı disklere koymak gerekir.

ORACLE_BASE, ORACLE_HOME, ORACLE_SID, PATH ve LD_LIBRARY_PATH değişkinleri tanımlandıktan sonra ya CREATE DATABASE komutu ile ya da Database Configuration Assistant ile yeni bir veritabanı oluşturabilirsiniz.

Elle oluşturmak isterseniz, gerekli işletim sistemi değişkenlerini tanımladıktan sonra ve gerekli SPFILE ya da initSID.ora dosyasının yarattıktan sonra

STARTUP NOMOUNT ile unmount evresinde instance başlatıp,

```
CREATE DATABASE sample
   CONTROLFILE REUSE
   LOGFILE
      GROUP 1 ('/disk1/log1.log', '/disk2/log1.log') SIZE 50K,
```

```
   GROUP 2 ('/disk1/log2.log', '/disk2/log2.log') SIZE 50K
MAXLOGFILES 5
MAXLOGHISTORY 100
MAXDATAFILES 10
MAXINSTANCES 2
ARCHIVELOG
CHARACTER SET AL32UTF8
NATIONAL CHARACTER SET AL16UTF16
DATAFILE
   '/disk1/data.dbf' AUTOEXTEND ON,
   '/disk2/index.dbf' AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
DEFAULT TEMPORARY TABLESPACE temp_ts
UNDO TABLESPACE undo_ts
SET TIME_ZONE = '+02:00';
```

tarzında bir sql ile veritabanınızı yaratabilirsiniz.

Kurulum bittikten sonra, otomatik olarak şifreleri de herkesçe bilinen iki kullanıcı yaratılmış olur. SYS ve SYSTEM kullanıcılarının sırasıyla, change_on_install ve manager şeklinde de şifreleri önceden yaratılmıştır. Bir DBA olarak ilk işiniz, veritabanına bağlanıp bu şifreleri değiştirmek olmalıdır. SYS kullanıcısı veritabanındaki data dictionary'lerin sahibi iken, SYSTEM kullanıcısı özellikle DBA'ye çok yarayan birçok tablo ve view'un sahibidir.

$ sqlplus "/ as sysdba" komutu ile sysdba yetkisinde sqlplus'a bağlanıp şifre değiştirme işlemlerini gerçekleştirebilirsiniz.

SQL> alter user system identified by <yeni_sifre>;
SQL> alter user sys identified by <yeni_sifre>;

komutları ile <yeni_sifre> yerine yazacağınız yeni bir şifre ile bu kullanıcılara yeni şifrelerini vermiş olursunuz.

Güvenlik artırmak için bağlantı biçimlerine göre de haraket edebilirsiniz. Eğer veritabanına sunucu üzerinde erişim dışında uzaktan da erişim sağlamak istiyorsanız bu durumda İşletim sistemi düzeyindeki oracle kullanıcısı ile orapwd adlı komutu çalıştırın:

$ orapwd file=$ORACLE_HOME/dbs/ora<SID> password=<sifre> entries=3

şeklinde bir komut vermelisiniz. Daha sonra uzaktan erişim için initSID.ora dosyasında REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE satırını eklemeli ve gerekli kullanıcılara gerekli yetkileri vermelisiniz.

Yukarıdaki komutta password’ten sonra SYSDBA için geçerli olacak şifreyi atamış, entries ile aynı kaç kişinin SYSOPER ya da SYSDBA yetkisi ile bağlanabileceğini tanımlamış oldunuz.

Oracle Veritabanını Açma

Oracle Veritabanı'nı açabilmek için sqlplus'ı çalıştırıp SYSDBA rolünde bağlanmak gerekecek.

$sqlplus
SQL> CONNECT / AS SYSDBA
SQL> STARTUP
Bu aşamada ilgili dizindeki başlangıç dosyası okunacak ve veritabanı açılmaya başlayacaktır. initSID.ora adlı başlangıç dosyası içindeki parametreler okunacak ve buna göre instance açılacaktır. Oracle 9i ile birlikte SPFILE adı verilen yeni bir başlangıç dosyası daha oluşmuştur. initSID.ora ile aynı yerde ve spfileSID.ora şeklinde bir dosya adıdır. Her ne kadar text bir dosya gibi görünse de doğrudan bir editörle müdahele etmek dosyayı bozabilir.

initSID.ora dediğimiz ya da PFILE olarak adlandırılan dosya text bir dosyadır ve herhangi bir editörle girip kolayca içeriğini değiştirebilirsiniz. Instance açıkken de bu dosyayı değiştirmeniz mümkündür ama yaptığınız değişiklik instance düzeyinde hiçbir değişiklik yapmayacaktır. Ancak bir sonraki açılışta bu başlangıç dosyası okunacak ve içerisindeki bilgilere göre bir açılış gerçekleşecektir. PFILE, $ORACLE_HOME/dbs dizini içinde yer alır genellikle. Ama farklı bir yerde ise de şu komutla çalıştırmak mümkündür.

SQL> STARTUP PFILE=/bulunduguyer/initSID.ora
Hatta isterseniz açılış komutlarını bir shell script içerisine yerleştirip otomatik olarak bu scripti çalıştırarak da instance'ı açabilirsiniz.

PFILE içerisinde; instance adı, control file'ların bulunduğu yerler, hata kodlarının nereye yazıldığı gibi bilgileri bulabilirsiniz. 9i ile dinamik olarak değiştirilebilen parametreler PFILE içerisinde hiçbir değişikliğe yol açmazlar. Bunu elle gidip eklemek ya da değiştirmek gerekir. SPFILE ise dinamik olarak değiştirilen parametrelerin bilgisini de tutar. ALTER SYSTEM SET komutu ile

parametrelerde değişiklik yapılabilir.

ALTER SYSTEM SET parameter=<yeni deger> [SCOPE=scopedegeri]
ile parametre değişir. SCOPE ; .memory, spfile ya da both değerlerinden biri olabilir.
SCOPE=MEMORY ile sadece o instance için geçerli olan bir değişiklik; SCOPE=SPFILE ile spfileSID.ora içerisinde, SCOPE=BOTH ile ise hem o instance için hem de spfile için değişiklik yapılmış olur.

Eğer PFILE'ınız var ama SPFILE'ınız yoksa CREATE SPFILE FROM PFILE; komutu ile SPFILE'ınızı yaratabilirsiniz. Bu yoksa, dinamik değişiklikleri de gerçekleştiremeyeceğiniz için yaratmanızda yarar var.
CREATE PFILE FROM SPFILE diyerek ise SPFILE'dan PFILE yaratabilirsiniz.

SPFILE ya da PFILE'da kullanılan parametreler:
BACKGROUND_DUMP_DEST -> artalanda çalışan işlemlerin (DBWR vb) loglarını yazıldığı yer,
COMPATIBLE -> Hangi Oracle sürümü ile uyumlu çalışabileceği
CONTROL_FILES -> Control File'ların nerede tutulduğu
CORE_DUMP_DEST -> Core oluşursa bunun nereye yazılacağı
DB_BLOCK_SIZE -> Her bir Block size'ının kaç byte olacağı.
DB_NAME -> Veritabanı adının ne olacağı
PROCESSES -> Kaç tane prosesin aynı anda çalışmasını izin verileceği
SHARED_POOL_SIZE -> Shared pool Size'ın boyutunu ne olacağını
vb birçok parametreyi tutuyoruz.

Veritabanı Açma - Kapama İşlemleri
Gerekli başlangıç dosyaları yaratıldıktan sonra startup komutu ile instance açılabilir. Açılışta NOMOUNT durumunda olan instance sonrasında MOUNT evresine geçer ve en son olarak da OPEN evresi ile herkesin kullanımına ve erişimine açılmış olur.

Nomount evresi ile Instance başlatma genellikle veritabanı yaratırken ya da control file'ların yedekleri alınırken yapılır.

Bir Instance başlarken, önce başlangıç dosyası okunur. $ORACLE_HOME/dbs dizini içerisinde varsa spfileSID.ora öncelikli sıradadır. Bu dosya yoksa spfile.ora bu da yoksa initSID.ora dosyasınıa bakılır. Bu dosyalardan var olan ve önceliği olanın içindekilere göre Instance başlar ve

veritabanı açılır.
Daha sonra parametrelere göre SGA oluşur ve background process'leri açılarak görevlerine başlar. alert dosyası da açılmış olur ve veritabanı ile ilgili önemli mesajlar bu alert dosyasına yazılır.

Not: alert dosyası bir DBA'in düzenli kontrol etmesi gereken bir dosyadır. BACKGROUND_DUMP_DEST değişkeni için verilen dizin, alert dosyasını barındırmaktadır. Bu dosya içinde veritabanı açılma, kapanma bilgileri; ORA- hataları; Datafile'lar gibi önemli alanlarda yapılan değişiklikler, Tablespace yaratma komutları, ALTER komutları gibi bilgiler tarih ve saat bilgisi ile yer almaktadır. Her instance açılışında eğer varsa bu dosya üzerine yazılır, yoksa yenisi yaratılır. Bu yüzden boyut olarak büyümüş alert dosyaları farklı bir adla kaydedilip silinebilir.

Bir değer evre MOUNT evresidir. Bir database'i mount edip açmamak şu işler için gereklidir:

- Database Kurtarma işlemleri
- Arşiv mode'a geçiş ya da çıkış işlemleri
- Datafile'ların adlarını değiştirme gibi işlemler

Tüm bunların yapılmasından da anlaşılacağı üzere, MOUNT evresinde UNMOUNT'tan farklı olarak Control File'lar açılır ki ne nerede tutuluyor bilinebilsin ve değişiklikler control file içerisine yazılabilsin.

OPEN evresinde ise data file'lar ve redo log file'lar açılarak üzerinde çalışır duruma gelinir.Bu aşamada Control File içerisindeki bilgiler kontrol edilir. Control File'da olmasına rağmen, gösterdiği yerde olmayan datafile'lar varsa database açılmaz. Bu durumda yedeklerden kurtarma yapmak gerekir. Yine bu aşamada, eğer daha önceden sistem sorunlu kapanmış ise Instance Receovery yapılır otomatik olarak.

STARTUP komutunu hangi evreyi açmak istediğinizi söyleyerek kullanabilirsiniz.

STARTUP; MOUNT, UNMOUNT, OPEN evreleri ile kullanılabilir.

SQL> startup mount

gibi. Bunun dışında açık olan bir instance'ın durumunu değiştirebilirsiniz. Bunun için ALTER DATABASE komutu kullanılır:

ALTER DATABASE <db_adi> MOUNT;

ALTER DATABASE <db_adi> OPEN; gibi.

Önemli bir özellik ise kullanıcıların verilerde değişiklik yapmasını önlemek için kullanılandır:
ALTER DATABASE <db_adi> OPEN READ ONLY;

Bir başka yöntem ise RESTRICT ile sınırlamaktır. STARTUP RESTRICT ile açmak ya da
ALTER SYSTEM ENABLE RESTRICTED SESSION; komutunu vermek yalnızca bu yetkiye sahip olan kullanıcıların bağlanmasına izin verecektir. Böylece bir veritabanı yöneticisi olarak yedek alma (export) ya da veri yükleme (import) gibi işlemleri, varsa sorunu saptama gibi adımları yapmanız kolaylaşacaktır. Bu komutu verseniz de daha önce sisteme bağlı olan kullanıcıları bulmanız ve onları sistemden çıkarmanız gerekecektir. Bunun için V$SESSION adlı view'dan yararlanırsınız.

```
SQL> SELECT SID,SERIAL# FROM V$SESSION;

       SID    SERIAL#
---------- ----------
         1          1
         2          1
         3          1
         4          1
         5          1
         6          1
         8      55478
```

ile instance'a bağlı tüm kullanıcıları görebilirsiniz. Dikkat etmeniz gereken konu burada bir instance açılırken de otomatik olarak açılan background processler'dir. Bunları öldürmemek için,

SQL> SELECT USERNAME,SID,SERIAL# FROM V$SESSION; komutunu kullanmak ve hangi kullanıcılar var diye görmek daha akılcıl olacaktır. SERIAL#'ı 1 olan kullanıcıların instance açılırken otomatik olarak açılan artlan işlemleri olduğunu unutmayın.

Öldürmek isteğiniz bir session'un SID ve SERIAL#'ını bilerek şu komutla öldürebilirsiniz:

SQL> ALTER SYSTEM KILL SESSION ‘8,55478’;

ALTER SYSTEM KILL SESSION ‘8,55478’; ‘deki 8 SID’ye, 55478 ise SERIAL#’a karşılık geliyor. Bu komutu verince bu kullanıcıya ait ve COMMIT edilmemiş işlemler geriye alınacaktır. Bu yüzden dikkatsiz kullanıcıları gözönünde tutarak (çoğunlukla ne olduğunu anlamayıp işlerini bitirdiklerini sanabiliyorlar), doğrudan Session öldürmek yerine mümkünse onların işlerini bitirip ayrılmalarını beklemek, bunun için duyuru yapmak daha iyi olacaktır. Yine de bazı durumlarda, acil olarak Session öldürülebilir. Zaman zaman bazı kullanıcıların yaptıkları işlemler ciddi kaynak tüketebilir ya da bazı tablolara lock koyabilir. KILL SESSION ile lock olan tablolar serbest kalacağından ve kullanıcı tarafından kullanılan kaynaklar da serbest bırakılacağından gerekli olabilir.

Veritabanı’nı Kapatma

Veritabanı SHUTDOWN komutu ile kapatılır. 4 yöntemle kapıtılabilir: NORMAL, IMMEDIATE, TRANSACTIONAL, ABORT.

shutdown komutu, shutdown normal ile aynıdır.
En doğru kapatma biçimi Normal’dir ama bir o kadar da kullanışsızdır. Normal ile kapatılma durumunda yeni kullanıcı bağlantısına izin verilmez ama mevcut tüm kullanıcıların çıkması, trancastion’larını tamamlaması ve tüm açık datafile ve diğer dosyaların kapanıp header’ların checkpoint numarasının yazılması beklenir. Bu yüzden acil durumlarda, hatta basit bir iş için database’i kapatmanın gerektiği durumlarda bile pek uygun değildir.

Transactional ise, Normal’den farklı olarak mevcut bağlantıların çıkmasını beklemez, trancastion’lar tamamlanınca kullanıcıların bağlantılarını düşürür.

Immediate ise transaction’ları da beklemeden kapatır. Sadece chechkpoint’leri yazar ki böylece sistem bütünlük içerisinde açılabilsin. Tabii ki commit edilmemiş transaction’lar rollback edilir.

Abort ise kaçınılması gereken ama bazen hiçbir seçeneğin kalmadığı bir durumdur. Sunucunun düğmesine basıp kapatmaktan farklı değildir. Checkpoint’ler de yazılamadığı için veritabanı açılamayabilir. Bu durumda bir dahaki açılışta Instance Recovery otomatik olarak başlayacaktır.

Shutdown komutu SYSDBA ya da SYSOPER yetkisindeki bir kullanıcı tarafından verilebilir.

6. Data Dictionary ve Önemli View'lar

Bir veritabanı yaratıldığı zaman otomatik olarak farklı bir sürü view'da yaratılmış olur. Böylece ilgili veritabanı ile ilgili çok önemli bilgiler burada tutulmaya başlar. Base tablolar adını verilen bu tablolara DML komutları ile bilgi yazılmamalıdır. Bunu yalnızca Oracle Server'ın kendisi yapabilir. Veritabanı oluşturulurken $ORACLE_BASE/rdbms/admin altında bulunan sql.bsq adlı dosya çalıştırılarak içindeki tablespace'ler ve tablolar oluşturulur. Daha sonra instance açıkken bu tabloların içi dolmaya ve değişmeye başlar. Bu tablolara karşılık gelen view'lar ise catalog.sql adlı sql betiği çalıştırılarak yaratılır. Data Dictionary Views adını verilen bu view'lara bakıp gerekli ayarlamaları yapmak bir DBA'in sorumluluğundadır. Bu sql'in içine baktığınızda da görebileceğiniz gibi v$ ile başlayan birçok view yaratılmaktadır ve bunlar kullanıcılar, roller, yetkiler, istatistiki bilgiler, veritabanın fiziksel ve mantıksal yapıları gibi önemli bilgileri tutarak Data Dictionary'i oluşturmaktadırlar.

vsga, vparameter, v$session, v$process, v$datafile gibi birçok view bulunmaktadır.

Data Dictionary, Oracle Server'ın çalışması için kaçınılmaz bir alandır. Böylece hangi kullanıcılar var, yetkileri nasıl kullandıracak, boyutlar ne durumda gibi birçok kritik konuyu yönetebilecektir. Bir DBA ya da kullanıcı ise yetki durumlarını, istatistiki bilgileri alıp yorumlayabilecek böylece daha etkin bir kullanım sağlayabilecektir.

Data Dictionary View'ları 3 ana grupta incelenebilir. Başlangış kısaltmalarına göre DBA, ALL ve USER.

DBA_ ile başlayan view'lar veritabanındaki tüm nesneleri görmenizi sağlar.
ALL_ ile başlayanlar ise mevcut kullanıcının sahip olduğu ve kullanabileceği tüm nesnelerini görmenizi sağlar.
USER_ ile başlayanlar ise mevcut kullanıcının sahip olduğu nesneleri görmenizi sağlar.

Örnekleyecek olursak, bir veritabanı içindeki tablolar _TABLES uzantılı view'larda durmaktadır.

SQL> SELECT OWNER, TABLE_NAME

```
FROM DBA_TABLES;
```

veritabanındaki tüm tabloları,

```
SQL> SELECT OWNER, TABLE_NAME
     FROM ALL_TABLES;
```

bağlandığınız kullanıcının görmeye yetkisi olduğu tüm tabloları,

```
SQL> SELECT OWNER, TABLE_NAME
     FROM USER_TABLES;
```

ise sadece bu kullanıcıya ait olan tabloları görmenizi sağlar.

Data Dictionary View'ları statik bilgiyi içerirler. Nesnelerin kime ait olduğu, yetki sınırlamaları, ne zaman yaratıldığı gibi bilgileri içerirler. Dinamik olarak değişen view'lar ise istatistik toplamak için kullanışlıdırlar. Bunlar SYS kullanıcısının view'larıdır ve bilgileri bellekten ve control file'dan toplayarak oluştururlar. Bunlara Dynamic Performance View'ları denir. Açık olan session'ları, kullanıcıları ve bilgilerini, tablolardaki lock vb bilgileri içerirler.

DICTIONARY ya da DICT tablosu, Oracle Server'ınızda bulunan tüm Data Dictionary View'larını ve Dynamic Performance View'larının adlarını içermektedir.

```
SQL> DESC DICT;
Column Name                    Null?    Type
------------------------------ -------- ----
TABLE_NAME                              VARCHAR2(30)
COMMENTS                                VARCHAR2(4000)
```

diyerek ve sonrasında örneğin SELECT * FROM DICT WHERE TABLE_NAME LIKE 'DBA%'; komutu ile DBA ile başlayan tüm view'ların adlarını görebilirsiniz.
Diğer önemli bir tablo ise DICT_COLUMNS tablosudur. Bir tablonun hangi column'unda ne tutulduğuna buraya bakarak karar verebilirsiniz.

```
SQL> select * from dict_columns where table_name='USER_TABLES';
TABLE_NAME                     COLUMN_NAME                    COMMENTS
```

USER_TABLES	MIN_EXTENTS	Minimum number of extents allowed in the segment
USER_TABLES	MAX_EXTENTS	Maximum number of extents allowed in the segment
USER_TABLES	PCT_INCREASE	Percentage increase in extent size
USER_TABLES	FREELISTS	Number of process freelists allocated in this segment
USER_TABLES	FREELIST_GROUPS	Number of freelist groups allocated in this segment
USER_TABLES	LOGGING	Logging attribute
USER_TABLES	BACKED_UP	Has table been backed up since last modification?
USER_TABLES	NUM_ROWS	The number of rows in the table
USER_TABLES	BLOCKS	The number of used blocks in the table
USER_TABLES	EMPTY_BLOCKS	The number of empty (never used) blocks in the table
USER_TABLES	AVG_SPACE	The average available free space in the table
USER_TABLES	CHAIN_CNT	The number of chained rows in the table

Bir DBA için önemli view'ları şöyle sıralayabilriz.

DBA_DATA_FILES, DBA_USERS, DBA_TABLESPACES, DBA_TABLES, DBA_SGMENTS, DBA_OBJECTS, DBA_INDEXES, DBA_FREE_SPACE, V$DATAFILE vb.

7. Controlfile

Control File, veritabanının adını, dosyalarını ve yerlerini, tablespace adlarını, checkpoint bilgileri gibi önemli bilgileri içeren bir kontrol dosyasıdır. Control File olmadan veritabanı açılamaz, control file'a yazma yetkisi yoksa veritabanı çalışamaz. Instance açık olduğu sürece yazma işlemi sürer ve shutdown komutu ile birlikte checkpoint bilgileri de bu dosya içerisine yazılır. Bu önemi nedeniyle mutlaka bir çifit daha olmalıdır ve mümkünse farklı bir diskte tutulmalıdır. DBA olarak son değişikliklerden sonra control file'ınızın bir yedeğini farklı bir ortamda saklamanız iyi olacaktır.

ALTER DATABASE BACKUP CONTROLFILE TO '/disk02/control2.ctl' diyerek mevcut control file'ın bir kopyasını çıkartabiliriz.

Control File, CREATE DATABASE komutu sırasında yaratılır ve büyüklüğü de MAXLOGFILES, MAXDATAFILES gibi değişkenlere göre belirlenir.

Control File'ınız tekse bunu mutlaka min 2 kopyayı çıkarmalınız. İki yöntemle yapmanız

mümkündür. Biri init.ora'da diğeri ise SPFILE'da.

initSID.ora ile Control File Çoklaması

SQL>shutdown normal diyerek veritabanını normal biçimde kapatınız.
Daha sonra işletim sistemi düzeyinde control dosyasının bulunduğu dizine gidp bu dosyanın bir kopyasını farklı bir dizine kopyalayınız.

$ cp /disk1/oradata/control1.ctl /disk2/oradata/control2.ctl
Daha sonra initSID.ora'yi bir editörle açıp CONTROL_FILES yazılı olan yere yeni control file'ını da ekleyeniz.

control_files='/disk1/oradata/control1.ctl', '/disk2/oradata/control2.ctl'
ve sonra
SQL>startup ile veritabanın açınız.

SPFILE ile Control File Çoklaması

SQL> ALTER SYSTEM SET CONTROL_FILES ='/disk1/oradata/control1.ctl', '/disk2/oradata/control2.ctl' SCOPE=SPFILE;

komutundan sonra shutdown normal ile veritabanı kapatın.

$ cp /disk1/oradata/control1.ctl /disk2/oradata/control2.ctl ile işletim sistemi düzeyinde yedeği oluşturun ve sonrasında,

SQL> startup

ile veritabanınızı açın.

Yarattığınız control file'ları değişik biçimlerde görebilirsiniz.

Sqlplus'a girdikten sonra,
SQL> SHOW PARAMETER CONTROL_FILES; ile ya da

SQL> SELECT * FROM v$CONTROLFILE; ile görebilirsiniz.

Control file'larla ilgili diğer önemli bir bilgiyi ise
V$CONTROLFILE_RECORD_SECTION adlı view'dan alabilirsiniz.

V$DATAFILE, V$LOG, V$LOGFILE gibi birçok view bilgilerini control file'lardan almaktadır.

8. Redo Log File

Veritabanında yapılan tüm transaction işlemleri, Online Redo Log dosyalarına yazılır. Bunun tek istinasi toplu yüklemeler ve NOLOGGING opsiyonu ile verilen işlemlerdir. Böylece sistemde yaşanması olası herhangi bir problem anında gerekli kurtarma işlemleri buradan yapılabilmektedir.Data File'lara yazılmamış, commit edilmemiş tüm veriler burada durur ve işlemlerle aynı anda bilgi tutulur. Tek kullanım yeri kurtarma operasyonudur.

Redo log dosyaları en az 2 gruptan oluşur ve her grup içerisine belirli sayıda member konulabilir.
LGWR proses'i verilerle ilgili tüm işlemleri, bir grup içiserindeki tüm member'lara aynı anda yazar. Bir grup içindeki her member, aynı büyüklüğe ve aynı log sıra numarasına (log sequence number) sahiptir. Bu numara control file içerisinde de saklanır ki, bir kurtarma anında datafile'lar redo log file'lar vb gerekli dosyalar aynı anı gösteriyor olsun.

Verilerle ilgili transaction'lar Redo Log Buffer'da tutulur ve daha önce belirttiğimiz koşullarda LGWR yardımıyla online olan log grubundaki log file'a yazılır. Bir gruptaki log file dolduğu zaman LGWR sıradaki log grubunu onlien yapar ve kalan değişiklikleri ona yazmaya başlar (buna log switch deniyor). Sıradaki tüm log file'lar dolunca LGWR en baştaki dolu olan log file'ın üzerine yazmaya başlar. Böylece gün içinde çok fazla log switch olmuşsa ve gün sonunda bir hataya rastladığınız için geri dönmeye kalkarsanız birçok bilginizi yitirmiş olursunuz. Bundan kurtulmanın yolları anlatılacaktır.

ALTER SYSTEM SWITCH LOGFILE; diyerek sıradaki log file'a geçmesini sağlayabiliriz.
ALTER SYSTEM CHECKPOINT; ile de checkpoint yapmaya zorlarız.

Yeni bir grup log file eklemek için

ALTER DATABASE ADD LOGFILE GROUP 3
(/disk03/oradata/log3a.dbf', /disk04/oradata/log3b.dbf')
SIZE 5M;

ile Log File'lar için 3. bir grup tanımladık ve 2 üyesinden birini disk03 altına diğerini de disk04 altına yerleştirdik. Boyut olarak da 5 MB verdik.

Bir şekilde grup içindeki üyelerden birinde fiziksel bir sorun oluşur ve kullanılamaz duruma gelirse diğer üye aynı işi yürütecektir. Ama yine de bu üyeyi uçurmak ve gerekirse yerine yenisini yaratmak gerekebilir.

ALTER DATABASE DROP LOGFILE MEMBER '/disk04/oradata/log3b.dbf';

Daha sonra bu dosyayı uçurup yeniden bir member ekleyebiliriz.

ALTER DATABASE ADD LOGFILE MEMBER '/disk04/oradata/log3b.dbf' TO GROUP 3;
Böylece 3.gruba yeni bir member ekledik. 2 üyeli gruplara yeni bir üye eklemek için de bu komutu kullanabilirsiniz.

Bir diğer gereklilik ise tüm grubu ve dolayısı ile grubun tüm üyelerini uçurmak olacaktır.

ALTER DATABASE DROP LOGFILE GROUP 3;
ile 3.grup ve tüm üyeleri gitmiş olur. Bir grubu silip yeni bir boyutla yaratmak isteyebilirsiniz. Bunun için önce yeni boyutu ile grupları ekleyip sonrasında istemediğiniz grupları silmek iyi olacaktır. Bir instance çalışabilmek için en az 2 grup istediği için 2'den az grup kalacak şekilde silmenize izin vermeyecektir. Online olan grubun drop edilmesine de izin vermeyecektir.

Veritabanı düzeyinde silmiş olduğunuz grupların içindeki üyelere ait dosyalar işletim sisteminden silinmezler. Bunu ayrıca silmeniz gerekir.

ALTER DATABASE CLEAR LOGFILE '/disk04/oradata/log3b.dbf' ile bozulmuş olan bir üyenin içeriği de silinebilir. Bu komut önce bir logfile yaratıp sonra eskisini silmeye eşdeğerdir.

Logfile'ları yaratırken boyut, konum ve kaç grup yaratılacağı çokça sorulan bir sorudur.
Her ne olursa olsun, değiştirme anı dışında, grupların farklı size'larda olması mantıklı değildir. Tüm

gruplara ait tüm memberlar aynı boyutta olmalıdır. Her bir grubun her bir üyesini farklı disklerde tutmak güvenlik sağlayacaktır.

Boyutlar ve grup sayısı konusunda ise kendi veritabanınızın davranışına bakmanız gerekmektedir. Eğer log grupları arasında çok hızlı switch oluyorsa ve özellikle ARCHIVELOG biçiminde iken logların arşivlenmesi için bekleniyorsa, instance bir süre bekleyeceği için yeni bir grup eklemek ya da gruplardaki üyelerin boyutlarını büyütmek işe yarayacaktır.

Güvenlik adına, üyeleri farklı disklerde olmak üzere 3 ayrı grup iyi bir çözüm olabilir.V$LOGHIST içerisindeki logların değişme tarihleri size zamanlama açısında bir düşünce verebilir. Yine alert.log dosyasındaki log switch zamanları da bakılabilecek güzel bir yerdir.
SQL> SELECT * FROM V$LOGHIST;

```
  THREAD#  SEQUENCE# FIRST_CHANGE# FIRST_TIM SWITCH_CHANGE#
---------- ---------- ------------- --------- --------------
       1        991     362564750 01-JAN-04      362579320
       1        992     362579320 01-JAN-04      362593950
       1        993     362593950 01-JAN-04      362608544
       1        994     362608544 01-JAN-04      362623176
       1        995     362623176 01-JAN-04      362637736
       1        996     362637736 01-JAN-04      362652252
       1        997     362652252 01-JAN-04      362666915
       1        998     362666915 01-JAN-04      362681543
       1        999     362681543 01-JAN-04      362696104
       1       1000      362696104 01-JAN-04       362710768
       1       1001      362710768 01-JAN-04       362725372
```

Bunlar dışında V$LOG ve v$LOGFILE view'ları Redo Log File'lar ile ilgili önemli bilgiler vermektedir.

```
SQL> SELECT * FROM V$LOG;

   GROUP#    THREAD#  SEQUENCE#     BYTES    MEMBERS ARC STATUS
---------- ---------- ---------- ---------- ---------- --- ----------------
FIRST_CHANGE# FIRST_TIM
------------- ---------
       1         1      1002   10485760         2 NO  INACTIVE
   362725372 01-JAN-04

       2         1      1003   10485760         2 NO  CURRENT
```

```
  362740021 01-JAN-04
```

```
SVRMGR> SELECT * FROM V$LOGFILE;
GROUP#     STATUS  MEMBER
---------- ------- --------------------------------------------------------------------------------
         2         /disk01/oracle/oradata/log02a.dbf
         2         /disk02/oracle/oradata/log02b.dbf
         1         /disk01/oracle/oradata/log01a.dbf
         1         /disk02/oracle/oradata/log01b.dbf
4 rows selected.
```

9. Tablespace'ler ve Datafile'ler

Tablespace'ler bir Oracle veritabanının mantıksal yapılarını oluşturmaktadır. Aslında fiziksel olarak bir yer tutmazlar. Her bir tablespace, belirli anlamdaki tablo ya da indeksleri toplayan yapılardır. Veritabanında tablespace olarak görülen yapı işletim sistemi düzeyinde fiziksel olarak bir ya da birden çok datafile'a karşılık gelir.

Tablespace → Datafile ya da Bir Çok Datafile

Bir Oracle Veritabanı daha önce de belirttiğimiz gibi iki yapıdan oluşmaktadır. Fiziksel Veri Yapısı ve Mantıksal Veri Yapısı

Fiziksel Yapı'da	**Mantıksal Yapı'da**
Datafile'lar, Control File'lar, Online Redo Log Dosyaları	Block'lar, Extent'ler, Segment'ler ve Tablespace'ler

Block (Block'lar) → Extent (Extent'leri) → Segment'i (Segmentleri) → Tablespace'i oluşturmaktadır.

9.1. Data Block'ları

Oracle Data Yapısının en küçük yapı birimleridir. initSID.ora'da DB_BLOCK_SIZE değişkeni ile tanımlanan büyüklüğe sahiptirler.

Not: DB_BLOCK_SIZE dğişkenini yaratırken İşletim Sistemi block'unun katları olacak şekilde

yaratmaya özen gösteriniz. Aksi takdirde çok fazla I/O olacak ve performans oldukça düşecektir.

En küçük veri yapısı olan Data Block'lar, yaptığınız ayarlamaya göre işletim sisteminden 1 ya da daha fazla block alarak; yazma, okuma ve silme işlemlerinin yapıldığı yerlerdir.

Block'ların en başında Block header adı verilen ve bloğun adresini, tablo dizini, row bilgisi gibi bilgiler bulunmaktadır. Hemen sonra ise boş bir alan yer almaktadır. Block header aşığıya doğru büyüdüğü için böylesi bir boş alana gerek duyulmaktadır. Daha sonra da verilerin bulunduğu alan bulunmaktadır. Veriler ise aşağıdan yukarı doğru dolarak boş alanı doldurmaktadır. Bunu bir şekille gösterecek olursak;

Boş alan, delete ve update işlemleri nedeniyle fragmente olmaktadır. Oracle Server, belirli dönemlerde defragmante etmektedir bu alanları (coalescing).
Blocklar gerçek anlamda verilerin bulunduğu en küçük birim olduğu için, verilerin ne kadarlık yer tutacağını, erişimi ve sınırlamaları burada tanımlıyoruz. Block Space Kullanım Parametreleri adını verdiğimiz bu değişkenler dört tanedir:
INITRANS: Bu değerle aynı anda kaç tane transaction'ın data block'unda değişiklik yapılabileceğini atarsınız. Table'lar için 1, index'ler için 2 varsayılan değer olarak atanmıştır. Ama bunu 5 yaparsanız aynı anda 5 transaction değişiklik yapabilir anlamına gelir.
MAXTRANS: En fazla kaç tane transaction'in aynı anda data ya da index block'unda değişiklik yapabileceğini atayacağınız değerdir. Varsayılan değeri 255'tir.

PCTFREE: Her data block'uğunda, büyümeyi karşılamak adına yüzde kaçlık boş yer ayrıldığını gösterir. Varsayılan değeri 10'dur (% 10).

PCTUSED: Data Block'un hangi yüzdeye geldiğinde dolu olduğunu gösteren değerdir. %40 varsayılan değerdir. Oracle Server, eğer data block alanı belirtilen PCTUSED değerinden düşükse

bu block'u boş olarak işaretler.

Bu ikisinin kullanımını şu örnekle açıklayabiliriz. PCTFREE'si %25 ve PCTUSED değeri %40 olan bir block düşünün. Bu block'a boş alanın %100-%25 yani % 75'lik alanı dolana kadar INSERT girişine izin verilir. Kalan kısım ise, daha önce NULL gibi atanmış değerlerin UPDATE ile doldurulması durumlarına karşı boş kalacaktır. Yani kalan %25'lik alan yeni row girilmesine izin vermezken, girilmiş row'ların genişlemesi için kullanılabilecektir.

Peki PCTUSED? Bu block'ta yer alan row'lardan bir kısmı ise zamanla silinebilir. Silenen row'lardan sonra kalan row'ların toplam boyutu data block'unun %40'ından az ise Oracle Server bu block'u boş olarak işaretleyecek ve yeniden kullanıma olanak tanıyacaktır.

Data Block'larını otomatik olarak ya da doğrudan müdahele yöntemiyle yönetmek mümkündür. Yönetiminin kolaylığı, INSERT işlemlerindeki verimlilik gibi nedenlerle Otomatik Segment Yönetimi daha elverişlidir. Otomatik Segment Yönetimi için CREATE TABLESPACE komutunda SEGMENT SPACE MANAGEMENT AUTO; satırını eklemek yeterli olacaktır. Bunun nasıl yapılacağını TABLESPACE'leri anlatırken belirteceğiz.

Diğer yöntem ise elle müdaheledir. Bu durumda ise PCTFREE, PCTUSED ve FREELIST değişkenlerini elle duruma göre değiştirmek gerekecektir.

9.2. Extent'ler

Block'lar birleşerek extent'leri oluştururlar. Extent'ler farklı datafile'lara dağılmazlar ve her segment içerisinde mutlaka en az bir extent vardır. Segment'lerin gereksinimine göre daha fazla extent eklenebilir. Segmentler istedikçe, extentler içerisinde devam edecek şekilde yeni extent'ler yer alır. Bir kısmı ise zamanla serbest bırakılır. Bu yüzden aşağıdakine benzer bir yapı sıklıkla görülmektedir.

F	D	D	B	D	D	B	D

En başta bulunan File header dışında Dolu ve Boş alanlar kullanım ve boşaltım nedeniyle bu hali alabilmektedir.

9.3. Segment'ler

Bir ya da daha fazla extent'in oluşturduğu segment'ler, tablespace'leri oluşturur. Tablespace'lerde bir ya da daha çok segment olabilir. Bir segment, farklı tablespace'lere dağılamaz ama farklı datafile'lara dağılabilirler.
Tablo ya da index gibi bir yapıdaki bilgiler segment'lerde dururlar.
4 çeşit segment bulunmaktadır: Table, Table Partition, Cluster ve Index.

Table biçimindeki segmentlerde bir veritabanı için en önemli yapı olan tabloların verileri tutulmaktadır. Bu yapıdaki veriler, ne clustered yapılmıştır ne de partitioned. Bu yüzden belirli bir sıralamaya göre veri tutulmamaktadır ve row'lar düzeyinde yapabileceğiniz çok fazla birşer yoktur.
Table segment'larinde tutulan tüm verilerin yalnızca bir tablespace'de olması kaçınılmazdır.

Table Partition biçimindeki segmentler daha yoğun kullanımlar için tercid edilmelidir. Bu biçimdeki yapılarda veriler farklı partition'larda tutulabilmektedir. Böylece farklı tablespace'lerde ve doalyısı ile farklı datafile'larda bulunan verilere erişim daha hızlı olmaktadır.
Cluster da aynı bir tablo gibidir ama birden çok tabloyu içerebilir. Cluster içindeki tablolar aynı storage parametrelerine sahip olacaklardır. Cluster'daki veriler, indekslenmiş column'lara göre oluşturulurlar ve ya indeksler üzerinden ya da hhash algoritmalar aracılığı ile erişilebilirler.

Index ise belirli yapıdaki birden çok indeksi bir yapıda toplamaktır.

9.4. Tablespace'ler

Tablespace'ler segmentlerin bir araya gelmesiyle oluşan ve bir ya da daha fazla datafile'a dağılan yapılardır. SYSTEM başta olmak üzere, en az bir adet tablespace her veritabanında bulunmak zorundadır. Tablespace'lerin oluşturduğu alan azalınca ya da bitince, datafile'lar büyütülebilir ya da bu tablespace'le ilgili yeni bir datafile yaratılabilir.
Tablespace yapısı DBA işleri için oldukça pratik sonuçlar sağlamaktadır. 24 saat çalışması düşünülen yapılarda ve gerekli alanlarda sadece bir tablespace OFFLINE duruma getirilerek üzerinde işlemler yapılabilir. Böylece tüm sisteminiz kesintiye üğramadan ve veritabanını tamamen kapatmadan işlemlerinizi yüretebilirsiniz. SYSTEM tablespace'i OFFLINE durumuna getirilemez (bu durumda instance'ın çalışması için gerekli yaşamsal bilgiler tutulamayacağı için).

İki tür tablespace vardır. SYSTEM olanlar ve olmayanlar.
SYSTEM tablespace'leri mutlaka her veritabanında olan ve Data Dictionary gibi önemli yapıları, System Undo Segment'lerini tutan tablespace'lerdir. SYSDBA yetkisi ilen bağlanan birçok kullanıcı SYSTEM tablespace üzerinde de tablo yaratıp işlerini yürütmektedirler (bazen yanlışlıkla). Bundan kaçınılması gerekir.
Kullanıcılar ve uygulamalar için ise Non_SYSTEM Tablespace'leri yaratılmalıdır. Böylece kullanıcıların ya da uygulamanın belirli bilgileri belirli tablespace'lerde tutulabilirler. Örneğin bir personel sisteminde, personelin özlük bilgileri bir tablespace'de, ödeme vb bilgileri bir başka tablespace'de, bunlara ait indexler ise ayrı tablespace'lerde tutulabilirler. Bu durum işlerin ayrı zamanlarda yapılması nedeniyle hem disk I/O'sunu azaltacak hem de ilgili tablespace'le ilgili yapılacak yedek alma, yeniden yaratma, değişiklik yapmak gibi konularda esneklik sğlayacaktır.

Bir tablespace yaratmak için gerekli komutlar aşağıda bulunmaktadır.

```
CREATE TABLESPACE tablespace
        [DATAFILE clause]
        [MINIMUM EXTENT integer[K|M]]
        [BLOCKSIZE integer [K]]
        [LOGGING|NOLOGGING]
        [DEFAULT storage_clause ]
        [ONLINE|OFFLINE]
        [PERMANENT|TEMPORARY]
```

Bir örnek verecek olursak,

```
SQL> CREATE TABLESPACE personeldata
                DATAFILE '/disk02/oradata/persdata01.dbf' SIZE 200M;
```

tarzında bir sql yeterlidir. Ama isterseniz, büyüklüğü kendiniz ayarlamak yerine belirli bir yere kadar büyümesine de izin verebilirsiniz. Bu durumda ek olarak,

```
SQL> CREATE TABLESPACE personeldata
                DATAFILE '/disk02/oradata/persdata01.dbf' SIZE 200M
                AUTOEXTEND ON NEXT 1M MAXSIZE 500M;
```

demeniz gerekecek. Böylece her yere gerek duyduğunda 1M kadar yer büyütecek ve maksimum büyüklüğü 500M olacaktır. Buradaki temel sorun bu tablespace'e karşılık gelecek datafile'ın

bulunduğu disk ya da disk partition'ında yeterli yer olup olmamasıdır. Eğer yeterli disk alanınız varsa ve kullanıcılarınızın yanlış bir yükleme yapmasını engelleyecek bir kontrol sistemini de kurduysanız AUTOEXTEND değişkeni ile tablespace yaratmanız DBA işlerinizi kolaylaştıracaktır.

CREATE TABLESPACE'in tüm değişkenlerini kullanmadan da yaratibilirsiniz. Bu durumda ilgili özellikle otomatik olarak varsayılan değerlere göre atanacaktır.

Değişkenlerden MINUMUM EXTENT ile kullanılan her extent'in min belirtilen kadar olacağını; OFFLINE diyerek yaratıldıktan hemen sonra tablespace'in diğer kullanıcılarca görülmesinin engellendiğini; TEMPORARY/PERMANENT diyerek bu tablespace'in geçici ya da kalıcı amaçlar için yaratıldığını belitmiş oluruz.

Bunun dışında önemli bir değişken ise tablespace yönetiminin nasıl yapılacağına ilişkin seçimi içerir. Hiçbir şey belirtmezseniz, Locally Managed olarak yaratılacak olan tablespace'lerde bu durumu belirten değişken EXTENT MANAGEMENT'dır.

Locally Managed Tablespace'lerde tablespace kendi extent durumuna datafile'larını da kullanarak bakar. Bunun için datafile'larda bitmap adını verdiği bir yapıda bilgi tutar. Bitmap'lerdeki her bit, bir block'a ya da bir grup block'a karşılık gelmektedir. Böylece extent ayırma, boşaltma ya da doldurma işlemleri sonrası bitmap, yeni durumu bilgiyi gösterecek şekilde değişir.

Dictionary Managed Tablespace'lerde ise extent ayrılması ya da serbest bırakılması sırasında, Oracle Server, ilgili tablolarla ilgili durumu Data Dictionary'ye kaydeder.

```
SQL> CREATE TABLESPACE personeldata
        DATAFILE '/disk02/oradata/persdata01.dbf' SIZE 200M
        EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

Şeklindeki bir komutla tablespace'i Local olarak yöneteceğinizi belirtmiş olursunuz. Local Managed Tablespace yaratmanın en büyük avantajı, Data Dicitionary Tablespace'de gereksiz yere iş yükü yaratmamasıdır. Bunun dışında tablespace'i coalesce etmeniz gerekmeyecektir ve extent ayırma/boşaltma işlemleri için undo bilgileri yaratmayacaktır.

```
SQL> CREATE TABLESPACE personeldata
        DATAFILE ‘/disk02/oradata/persdata01.dbf’ SIZE 200M
        EXTENT MANAGEMENT DICTIONARY
        DEFAULT STORAGE (INITIAL 512K NEXT 512K);
```

şeklindeki bir komut ile ise Dictionary Managed Tablespace yaratabilirsiniz.

Bu şekilde kendinizin belirleyebileceği bir depolama yapısı belirtebiliyorsunuz extent’ler için ama bu tablespace’lerdeki farklı yerlerde bulunan extent’leri bir araya getirmek için coalesc yapılması zorunlu oluyor. Ayrıca bilgiler Data Dictionary’de tutulduğu için bir yavaşlık sözkonusu olabiliyor.

Ama ALTER TABLESPACE komutu ile storage değişkeni ileride değiştirilebiliyor ki Locally Managed Tablespace’lerde bunu yapmak mümkün değil.

```
ALTER TABLESPACE personeldata
    MINIMUM EXTENT 1M;
```

diyerek ya da

```
ALTER TABLESPACE personeldata
    DEFAULT STORAGE(
    INITIAL 1M
    NEXT 1M
    MAXEXTENTS 500);
```

gibi komutlar vererek ayarlama yapmak mümkündür.

Otomatik Segment Yönetimi için CREATE TABLESPACE komutunda bir bilgi verebiliriz. Bunun için

```
SQL> CREATE TABLESPACE personeldata
        DATAFILE ‘/disk02/oradata/persdata01.dbf’ SIZE 200M
        EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K
        SEGMENT SPACE MANAGEMENT AUTO;
```

sql’indeki SEGMENT SPACE MANAGEMENT AUTO; satırını eklemek yeterli olacaktır.

Oracle Server'ında yarattığınız bir veritabanı için belirli özelliklerde tablespace yaratmanız mümkündür. Bunlardan biri Temporary tablespace yaratmaktır.

CREATE TEMPORARY TABLESPACE diyerek geçici amaçla kullanabileceğiniz bir tablespace yaratabilirsiniz. Temporary tablepace ile özellikle sort gibi, geçici tablo yaratmak gibi işlemlerinizi kullanabilirsiniz. Bunları Locally Managed olarak yaratmak iyi olur ve temporary tablespace'ler tempfile olarak da adlandırılır. Bu tablespace'leri Read-Only yapamazsınız, adlarını değiştiremezsiniz. Kurtarma operasyonları ile tempfile'lar kurtarılımaz, doğal olarak Control File içerisinde de tempfile'larla ilgili bilgi barındırılmaz.

Not: Temporary Tablespace'lerdeki sort hızını artırabilmek için başlangıç dosyamızdaki UNIFORM SIZE değişkenini, SORT_AREA_SIZE değişkeninin katı olarak belirlemek gerekir.

Bir başka tablespace türü ise UNDO Tablespace adını verdiğimiz ve undo segment'leri içinde barındırarak (bundan başka hiçbir nesne yeralamaz içinde) undo işlemlerini yapmanız için gerekli yönetimi sağlar.

```
CREATE UNDO TABLESPACE undo
        DATAFILE '/disk01/oradata/undo1.dbf' SIZE 50M;
```

diyerek undo tablepsace'ini yaratabilirsiniz.

Temporary Tablespace dışında tüm veritabanı düzeyinde geçerli olacak bir tablespace yaratmak da mümkün. DEFAULT TEMPORARY TABLESPACE adını verdiğimiz bu tablespace, daha önce yaratılmış olan bir temporary tablespace'i defaulu olarak atayarak yapılır.

Diyelim ki DEF_TEMP diye bir temporary tablespace daha önce yaratılmıştı.

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE def_temp;
```

diyerek veritabanı düzeyinde varsayılan temporaray tablespace'imizin def_temp olduğunu belirlemiş olduk. Kullanıcı yaratma işlemlerinde ilgili komut verilmezse default temporary tablespace olarak SYSTEM tablespace'i kullanılacaktır ve bu ciddi performans sorunlarına yol açacaktır (özellikle sort gibi işlemler yapılacaksa). Tüm kullanıcılar yaratılırken, bundan böyle

default temporary tablespace'leri def_temp olacaktır.

Eğer yeni bir temporary tablespace yaratıp, default temporary tablespace'i ALTER DATABASE komutu ile bu yeni tablespace yaparsanız tüm kullanıcıların default temporary tablespace'leri otomatik olarak bu yeni tablespace olacaktır.

Default temporary tablespace'ler offline yapılamazlar, yenisi yaratılıp ALTER DATABASE ile belirtilmediği sürece Drop edilemezler ve asla permanent yapılamazlar.

Yeri gelmişken bir tablespace'in nasıl OFFLINE yapıldığını ve sonra nasıl ONLINE yapıldığını da görelim.

SQL> ALTER TABLESPACE personeldata OFFLINE;

diyerek OFFLINE durumuna getirilen bir tablespace'e dışarıdan erişim sağlanamaz. Bunu yapmamızın birçok nedeni olabilir. İlki bu tablespace üzerine her türlü erişimi engelleyerek kriz anlarında müdahale edebilme olanağını bulmanızdır. Diyelim ki sisteminizde verilerde bir sorun olduğunu saptadınız. Verilerin kaydedildiği ve anlık olarak sürekli değişiklik yapıldığı tabloları tutan bir tablespace OFFLINE durumuna getirip ne olup bittiğini anlamaya çalışırsınız. Bu sırada kimse ilgili tablespace içindeki nesnelere ulaşamayacağı için de daha fazla kaygılanmanıza gerek olmaz.

Bunun dışında bu tablespace'i yedek alabilirsiniz, gerekiyorsa bu tablespace'le ilgili datafile'da kurtarma operasyonu yürütebilirsiniz.

SQL> ALTER TABLESPACE personeldata ONLINE;

diyerek de OFFLINE durumundaki bir tablespace'i ONLINE durumuna getirebilirsiniz.

Buna benzer bir yapı tablespace'i READ ONLY durumuna getirmektir. Bu durumda tablespace içindeki nesneler okunabilir durumda olacaktır ama transaction yapılmasına izin verilmeyecektir.

SQL> ALTER TABLESPACE personeldata READ ONLY;

komutu ile yalnız okunabilir duruma getirdiğiniz personeldata tablespace'ini okuma amaçlı izin

vermek dışında içindeki bazı nesneleri drop etmek amaçlı da kullanabilirsiniz. Bu tablespace'le ilişkili datafile'in yedeğini Cd-Rom gibi araçlara almak için ilk yapılması gereken işlem de READ ONLY durumuna getirmektir. Bundan sonra ALTER TABLESPACE <tablespace_adi> RENAME DATAFILE ... komutu ile ilgili datafile kopyalarsınız.

Var olan bir tablespace artık işinize yaramıyorsa bunu DROP komutu ile silmeniz mümkündür.

SQL> DROP TABLESPACE personeldata;

DROP TABLESPACE <tablespace_adi>
[INCLUDING CONTENTS [AND DATAFILES] [CASCADE CONSTRAINTS]]
komutunda INCLUDING CONTENTS ile tablespace'deki tüm segmentleri de uçurmaktasınız (tablespace içerisinde bilgi barındırılıyorsa bunu kullanmak zorundasınız).

AND DATAFILES'ı da eklerseniz işletim sistemi düzeyinde karşılık gelen datafile'lar da silinecektir (aksi halde gidip tek tek işletim sistemi düzeyinde silmeniz gerekir).

CASCADE CONSTRAINTS'i de eklerseniz, drop edilen tablespace içindeki tablolara referans veren diğer tablolardaki referans'ları da ortadan kaldırmış olacaksınız (ki doğrusu da budur, çünkü silinen bir tablespace'deki tablo'da gideceği için olmayan bir tablo'ya referans vermek doğru değildir. Ama yine de belirli durumlarda, bundan kaçınırsınız. Örnek verirsek, import ile çoğu zaman yanlışlıkla tablespace'ler gereksiz bilgilerle doldurulabilir. Bu durumda tek tek tablolardaki fazla bilgileri silmek yerine tablespace uçurulup, yenisi yaratılabilir. İşte bu durumda dışarıdaki tablolardaki referansları CASCADE CONSTRAINTS ile uçurmak çok ciddi bir yanlıştır).

Tablespace Boyutunu Büyütmek

Tablespace'leriniz içindeki nesneler bilgilerle doldukça tablespace'in tuttuğu yer de artacaktır. Hangi tablespace'in ne kadar yer tuttuğunu görme için SM$TS_FREE adlı view oldukça işe yaramaktadır:

```
SQL> SELECT * FROM SM$TS_FREE;

TABLESPACE_NAME                BYTES
------------------------------ ----------
```

```
ABMD                    37437440
ABMX                    33505280
AHLD                    13434880
AHLX                     8683520
AHMD                    10403840
AHMX                    10403840
AKD                    142376960
AKX                    124149760
ALRD                     6717440
ALRX                    14663680
AMFD                     7454720
```

Her bir tablespace ve bunlar da yer alan boş yeri bu komutla görebilirsiniz. İyi bir DBA bunları düzenli kontrol etmeli ve gerekli işlemlere göre önceden gerekli yer ayarlamalarını yapmalıdır.

SM$TS_FREE aslında sizlere özet bir bilgi vermektedir. Bu view tablespace kaç datafile içerirse içersin her datafile'da kalan boş yerleri toplayarak size bilgi sunmaktadır. Eğer datafile'lar düzeyinde de bilgi edinmek isterseniz, bu durumda DBA_FREE_SPACE'i kullanmalısınız:

```
SQL>  SELECT * FROM DBA_FREE_SPACE;

TABLESPACE_NAME                FILE_ID   BLOCK_ID     BYTES     BLOCKS
------------------------------ ---------- ---------- ---------- ----------
RELATIVE_FNO
------------
ABMD                           53      1829   37437440      4570
        53

ABMX                           54       169      81920        10
        54

ABMX                           54      2319   33423360      4080
        54
```

Yukarıdan da gördüğünüz gibi ABMX 2 ayrı satırda görünmekte, çünkü ABMX için iki ayrı datafile yaratılmış durumda.

Tablespace boş alanlarını ister günlük ister haftalık olarak kontrol edebilirsiniz. Bunun dışında

tablespace’leri otomatik olarak büyüyecek şekilde de yaratabilirdiniz. Daha önce de gördüğümüz şu komut belirli bir büyüklüğe kadar (örneğimizde 500M’a kadar) datafile’ın büyemisini otomatik olarak sağlayacaktır:

```
SQL> CREATE TABLESPACE personeldata
        DATAFILE ‘/disk02/oradata/persdata01.dbf’ SIZE 200M
        AUTOEXTEND ON NEXT 1M MAXSIZE 500M;
```

Peki datafile böyle yaratılmadıysa? Bu durumda ya ALTER komutu ile ilgili datafile’ı AUTOEXTEND yaparsınız ya da mevcut alanı büyütürsünüz. Önce ALTER komutu ile AUTOEXTEND olmayan bir datafile’i otomatik hale getirelim. Diyelim ki ozluk diye bir tablespace’iniz var ve buna karşılık gelen datafile /disk01/oradata/ altında ozlukdata01.dbf olarak tutuluyor. Bu durumda

```
SQL> ALTER DATABASE DATAFILE
        ‘/disk01/oradata/ozlukdata01.dbf’ SIZE 100M
        AUTOEXTEND ON NEXT 1M MAXSIZE 500M;
```

ile bu datafile’ı 500M’a kadar büyüyebilir bir büyüklüğe getirmiş olursunuz (burada MAXSIZE ... yerine UNLIMITED diyerek tüm sınırları da kaldırabilirsiniz).

Ama daha önce de belirttiğimiz diskte yeterli yer olmaması ya da kullanıcıların yanlış kullanımdan dolayı kontrolsüz büyemenin önüne geçmek adına düzenli olarak kontrol edip kendiniz büyütmek de isteyebilirsiniz. En kısa yol mevcut büyüklükten daha büyük bir duruma getirmektir. 100M’lık bir tablespace’inizi 200 M yapmak için

```
SQL> ALTER DATABASE DATAFILE
        ‘/disk01/oradata/ozlukdata01.dbf’ RESIZE 200M;
```

yeterli olacaktır. Mevcut büyüklükten küçük bir değer girmeniz durumunda içindeki verinin büyüklüğüne göre ancak belirli bir miktarda düşürülmesine izin verecektir.

Bunun dışında diğer yöntem ise ilgili tablespace’e yeni bir datafile eklemek şeklindedir.

SQL> ALTER TABLESPACE ozluk

ADD DATAFILE ‘/disk01/oradata/ozlukdata02.dbf’ SIZE 100M;

ile ozluk tablespace’ine ozlukdata01.dbf dışında ozlukdata02.dbf datafile’ını da eklemiş oldunuz. Buradaki sorun çok fazla datafile’la uğraşmak zorunda kalacak olmanızdır.

Belirli durumlarda bir datafile’ı farklı bir yere taşımanız gerekebilir.

ALTER TABLESPACE ozluk

RENAME DATAFILE ‘/disk01/oradata/ozlukdata02.dbf’

TO ‘/disk04/oradata/ozlukdata02.dbf’;

Bunu yapmak için önce ilgili tablespace’i OFFLINE yapmalısınız. Sonra işletim sistemi düzeyinde ilgili datafile yeni yerine kopyalamalısınız (taşımak daha doğru olur, çünkü bu datafile’a artık gerek duymayacaksınız). Sonra yukarıdaki ALTER TABLESPACE .. RENAME DATAFILE komutunu verip ilgili tablespace’i ONLINE durumuna getirmelisiniz.Bu işlem için tablespace’i OFFLINE durumuna getirmek gereki olduğu için SYSTEM tablespace’i gibi bazı tablespace’ler bu komutla yer değiştirtilemezler. Bunun yerine, ALTER DATABASE RENAME FILE komutu kullanılabilir. Bu komut veritabanındaki tüm dosya biçimleri için kullanılabilir.

Sırasıyla şu işlemler yapılmalı

1. Veritabanı shutdown komutu ile kapatılır.
2. İşletim Sistemi komutu ile taşınacak dosya yeni yerine taşınır.
3. ALTER DATABASE <db_adi> MOUNT komutu ile veritabanı MOUNT edilir.
4. ALTER DATABASE RENAME FILE ‘/eskidizin/dosyadi.dbf’ TO ‘/yenidizin/dosyadi.dbf’ şeklindeki komut çalıştırılır.
5. ALTER DATABASE OPEN komutu ile veritabanı açılır.

İlgili view’lar, tablolar

Tablespace’lerin durumu, datafile’lar hakkında bilgi, extent ve segment durumları için birçok view bulunmaktadır.

Boş extent’leri görmek için DBA_FREE_SPACE’ten;

Dolu extent’leri görmek için DBA_EXTENTS’ten;
Segmentler için DBA_SEGMENTS’ten,
Tablespace’ler için DBA_TABLESPACES’ten;
Datafile’lar için DBA_DATA_FILES ya da V$DATAFILE view’larından yararlanabiliriz.

Bir tablespace yaratıldığı zaman DBA_TABLESPACES içerisine bu tablespace’le ilgili bir bilgi girilir.

```
SQL> SELECT * FROM DBA_TABLESPACES WHERE TABLESPACE_NAME='PERSONELDATA';

TABLESPACE_NAME                INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS
------------------------------ -------------- ----------- -----------
MAX_EXTENTS PCT_INCREASE MIN_EXTLEN STATUS    CONTENTS  LOGGING   EXTENT_MAN
----------- ------------ ---------- --------- --------- --------- ----------
ALLOCATIO PLU
--------- ---
PERSONELDATA                          40960      40960          1
      505          50         0 ONLINE    PERMANENT LOGGING   DICTIONARY
USER      NO
```

tarzında bir bilgi görebiliriz.

Bu tablespace için kaç tane datafile yaratılmışsa o kadarlık bilgi de DBA_DATA_FILES dosyasına yazılır. Buradaki boş alan (datafile’daki) bilgisi de, header için gerekli yer çıkartıldıktan sonra DBA_FREE_SPACE içerisine yazılır.

```
SQL> SELECT * FROM DBA_FREE_SPACE WHERE TABLESPACE_NAME= 'PERSONELDATA';

TABLESPACE_NAME                FILE_ID   BLOCK_ID     BYTES     BLOCKS
------------------------------ ---------- ---------- ---------- ----------
RELATIVE_FNO
------------
PERSONELDATA                          321       722   15065088      1839
       321
```

Bir segment yaratıldığında DBA_SEGMENTS içerisine bir satır eklenir ve ilgili bilgiler eklenir.

Daha sonra bu segment için ayrılmış extent'ler DBA_EXTENTS içerisinde görülür. Aynı anda DBA_FREE_SPACE içerisinde ilgili boş yer bilgisi azaltılır.

Bu nedenle DBA_FREE_SPACE ve DBA_EXTENTS içerisine bakılarak ilgili datafile'lardaki boş yerler bulunabilir. Daha kısa bir yol ise SM$TS_FREE tablosuna bakmak olabilir.

10. Undo İşlemlerinin Yönetimi

Herhangi bir tablo üzerinde yaptığınız update işlemi sonucunda ilgili column'lardaki bilgi değişecektir. Herhangi bir nedenle daha önce yapılan değişiklikten geriye dönmek gerekirse bunu tutmak gerekecektir. Oracle'da bu undo segment adı verilen yapılarda tutulur.
Bir update sonucu, tablo'ya yeni bilgi girilir. Eski bilgi ise daha öncesinde alınarak, undo segment içine yazılır. Undo segment içerisinde hem değişen bilgi hem de transaction sonucu yapılan değişikliğin etkilediği tablo ve ilgili değişikliğin adresi yer alır. Bu undo segment'lar yalnızca istek ya da sorun nedeniyle geriye dönüşlerde değil aynı zamanda Read Consistency adı verilen ve verilerin tutarlığını sağlayan işlemler için de gereklidir. Update edilen bir değer tablolara commit edilene kadar yazılmaz. Bu yüzden de o durumda gelecek bir sorgu verinin değişmeden önceki halini göstermelidir. Bu yüzden de undo segment'lerdeki veriler okunur. Bunu anlamak için şu örneği verebiliriz. Bir alışveriş sitesinden bir ürün almak istediniz. Şifrenizle girip bir ürün seçtiniz ama henüz alma işleminizi sonlandırmadınız. Bu durumda, belirli üründen kaç tane var diye istatistiki bir çalışma adına yapılan sorgu, sizin bitirmediğiniz ürüne ait sayıyı doğru göstermek için bitirilmemiş (commit edilmemiş) işlemi saymayacaktır. Bunun için de undo segment'teki değeri okur. Aksi halde, siz bu ürünü almaktan vazgeçseniz de bu ürün stoklardan çıkmış gözükecektir.

Oracle bunu şöyle yapar. Bir SELECT query'si geldiği zaman, o andaki system change number'ını (SCN'yi) not eder. Bu SCN numarasından sonra yapılan tüm değişiklikleri göz ardı ederek SCN zamanından önceki bilgilere göre SELECT'i çalıştırır ve sonucu ona göre çıkartır. Query'lerde otomatik olarak sağlanan Read Consistincy, transactionlardan önce vereceğeniz

SET TRANSACTION READ ONLY;

komutu ile siz de transaction'lardan önce sağlayabilirsiniz.

Eğer transaction yapıldıktan sonra, değişen bilginin eski hali kurtarılmak istenirse ROLLBACK

komutu kullanılacaktır. Bu durumda undo segment'te yazılı olan undo data'sı (eski bilgi) yeni bilgi yerine getirilecek ve tablo eski haline dönecektir. Buna Transactional Rollback denir.

Transaction yapılıyorken, ağdaki bir sorun, sistem çökmesi gibi bir nedenle Instance çalışamaz duruma gelirse, Oracle Server commit edilmemiş tüm işlemleri geri alır. Eğer undo segment'lerdeki veriler Online Redo Log File'lara da yazılmış ise Transaction Recovery olur ve commit edilmemiş tüm bilgiler geri alınır.

Undo işlemleri için tablespace'lerin nasıl yaratılacağını Tablespace'ler böülümünde görmüştük. CREATE UNDO TABLESPACE ... komutu ile yaratılan bir undo tablespace'ini otomatik olarak yönetmek oldukça pratiktir. Bunun için başlangıç dosyamıza (initSID.ora),

UNDO_MANAGEMENT= AUTO
ve
UNDO_TABLESPACE=<undo_tablespace_adi>

yazarak undo tablespace adını verir ve undo yönetimini Oracle Server'a bırakırsınız.

ALTER SYSTEM SET undo_tablespace=<undo_tablespace_adi>; komutu ile dinamik bir değiştirme ya da atama mümkündür.

```
SQL> SELECT END_TIME, BEGIN_TIME, UNDOBLKS
 2  FROM V$UNDOSTAT;
```

ile başlayan ve biten undo işlemlerini görebilirsiniz. Bu view aracılığı ile yarattığınız UNDO tablespace'inin doğru ve verimli çalışıp çalışmadığını kontrol edebilir gerekirse değişiklik yapabilirsiniz.Undo tablespace için gerekli yer ayrılıp ayrılmadığı sorusu çokça sorulan bir sorudur. Bunun için şu komutu çalıştırınız önce:

```
SQL> SELECT (SUM(undoblks) / SUM( ((end_time - begin_time) * 86400)
 2 FROM v$undostat;
```

Buradan çıkan sonuç, her saniyede üretilen undo data block sayısını verecektir. Buna SUS (Saniye başına Undo Sayısı diyelim).

initSID.ora'dan edineceğimiz UNDO_RETENTION ve DB_BLOCK_SIZE değişkeni değerlerini de UR ve DBS olarak adlandırırsak, UNDO SPACE için gerekli yeri

(DBS*UR*SUS) + (24* DBS)

formülü ile bulabiliriz. Bunu en yoğun kullanım anında çalıştırmayı unutmayınız.

DBA_ROLLBACK_SEGS view'u hem SYSTEM hem de kullanıcı undo segment'leri hakkında geniş bir bilgi verecektir:

```
SQL> SELECT SEGMENT_NAME, OWNER, TABLESPACE_NAME, STATUS
 2  FROM DBA_ROLLBACK_SEGS;

SEGMENT_NAME     OWNER  TABLESPACE_NAME STATUS
------------------------------ ------ --------------- ----------------
SYSTEM             SYS   SYSTEM    ONLINE
RBS01              SYS   RBS       ONLINE
RBS02              SYS   RBS       ONLINE
```

Eğer owner'ı SYS yerine PUBLIC ise bu segment'in public undo segment olduğunu gösterir.

V$ROLLNAME ve V$ROLLSTAT view'larını aşağıdaki biçimde join ederseniz kullanılmakta olan undo segment'lar hakkında istatistiki bilgi edinebilirsiniz.

```
SQL> SELECT a.name, b.extents, b.rssize, b.hwmsize, b.xacts, b.status
  2  FROM v$rollname a, v$rollstat b
  3  WHERE a.usn = b.usn;

NAME                 EXTENTS     RSSIZE    HWMSIZE      XACTS    STATUS
-------------------- ---------- ---------- ---------- ---------- ---------------
SYSTEM                    5     401408     401408          0       ONLINE

RBS01                   161  171450368  171450368          0       ONLINE
```

```
RBS02                  75   79863808   79863808          0       ONLINE
```

V$SESSION ile V$TRANSACTION'ı birleştirerek de hangi kullanıcının bir transaction'da aktif olarak undo segment kullandığını görebilirsiniz.

```
SQL> SELECT s.username, t.xidusn, t.ubafil, t.ubablk, t.used_ublk
  2  FROM   v$session s, v$transaction t
  3  WHERE  s.saddr = t.ses_addr;

USERNAME                          XIDUSN     UBAFIL     UBABLK  USED_UBLK
------------------------------ ---------- ---------- ---------- ----------
OGR1                                    3         21      79426          1
```

11. TABLO VE INDEX'LER

11.1. Tablolar

Bu bölümde tablo ve indeks yaratmayı daha ayrıntılı biçimde görüp üzerlerindeki değişiklik işlemlerinde kritik uygulamaları göreceğiz.

Hangi veritabanı sisteminde olursanız olun, bilmeniz gereken en önemli konulardan biri bu veritabanı yönetim sisteminin hangi data tiplerini desteklediğine bakmak olmalıdır. Bunun için ilgili RDBMS'in Reference Guide'ına bakmanız gerekmektedir. Oracle 9i'nin Reference Guide'ına bakacak olursak, data tiplerini şu şekilde göreceğiz:

Tablo.1. Oracle'daki Data Tipleri ve Açıklamaları

Code[a]	Built-In Datatype	Description
1	VARCHAR2(*size*) [BYTE | CHAR]	Variable-length character string having maximum length *size* bytes or characters. Maximum *size* is 4000 bytes, and minimum is 1 byte or 1 character. You must specify *size* for VARCHAR2.

Code[a]	Built-In Datatype	Description
		BYTE indicates that the column will have byte length semantics; CHAR indicates that the column will have character semantics.
1	NVARCHAR2(*size*)	Variable-length character string having maximum length *size* characters. Maximum *size* is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify *size* for NVARCHAR2.
2	NUMBER(*p*,*s*)	Number having precision *p* and scale *s*. The precision *p* can range from 1 to 38. The scale *s* can range from -84 to 127.
8	LONG	Character data of variable length up to 2 gigabytes, or 2^{31} -1 bytes.
12	DATE	Valid date range from January 1, 4712 BC to December 31, 9999 AD.
180	TIMESTAMP (*fractional_seconds_precision*)	Year, month, and day values of date, as well as hour, minute, and second values of time, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values of *fractional_seconds_precision* are 0 to 9. The default is 6.
181	TIMESTAMP (*fractional_seconds_precision*) WITH TIME ZONE	All values of TIMESTAMP as well as time zone displacement value, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values are 0 to 9. The default is 6.
231	TIMESTAMP (*fractional_seconds_precision*) WITH LOCAL TIME ZONE	All values of TIMESTAMP WITH TIME ZONE, with the following exceptions: • Data is normalized to the database time zone when it is stored in the database. • When the data is retrieved, users see the data in the session time zone.
182	INTERVAL YEAR (*year_precision*) TO MONTH	Stores a period of time in years and months, where *year_precision* is the number of digits in the YEAR datetime field. Accepted values are 0 to 9. The default is 2.

Code[a]	Built-In Datatype	Description
183	INTERVAL DAY (*day_precision*) TO SECOND (*fractional_seconds_precision*)	Stores a period of time in days, hours, minutes, and seconds, where • *day_precision* is the maximum number of digits in the DAY datetime field. Accepted values are 0 to 9. The default is 2. • *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND field. Accepted values are 0 to 9. The default is 6.
23	RAW(*size*)	Raw binary data of length *size* bytes. Maximum *size* is 2000 bytes. You must specify *size* for a RAW value.
24	LONG RAW	Raw binary data of variable length up to 2 gigabytes.
69	ROWID	Base 64 string representing the unique address of a row in its table. This datatype is primarily for values returned by the ROWID pseudocolumn.
208	UROWID [(*size*)]	Base 64 string representing the logical address of a row of an index-organized table. The optional *size* is the size of a column of type UROWID. The maximum size and default is 4000 bytes.
96	CHAR(*size*)[BYTE | CHAR]	Fixed-length character data of length *size* bytes. Maximum *size* is 2000 bytes. Default and minimum *size* is 1 byte. BYTE and CHAR have the same semantics as for VARCHAR2.
96	NCHAR(*size*)	Fixed-length character data of length *size* characters. Maximum *size* is determined by the national character set definition, with an upper limit of 2000 bytes. Default and minimum *size* is 1 character.
112	CLOB	A character large object containing single-byte characters. Both fixed-width and variable-width character sets are supported, both using the CHAR database character set. Maximum size is 4 gigabytes.
112	NCLOB	A character large object containing Unicode characters. Both fixed-width and variable-width

Code[a]	Built-In Datatype	Description
		character sets are supported, both using the NCHAR database character set. Maximum size is 4 gigabytes. Stores national character set data.
113	BLOB	A binary large object. Maximum size is 4 gigabytes.
114	BFILE	Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.

[a] The codes listed for the datatypes are used internally by Oracle. The datatype code of a column or object attribute is returned by the DUMP function.

Bunlar dışında

Tablo.2. Datetime Alanları ve Değerleri

Datetime Field	Valid Values for Datetime	Valid Values for INTERVAL
YEAR	-4712 to 9999 (excluding year 0)	Any positive or negative integer
MONTH	01 to 12	0 to 11
DAY	01 to 31 (limited by the values of MONTH and YEAR, according to the rules of the current NLS calendar parameter)	Any positive or negative integer
HOUR	00 to 23	0 to 23
MINUTE	00 to 59	0 to 59
SECOND	00 to 59.9(n), where "9(n)" is the precision of time fractional seconds The "9(n)" portion is not applicable for DATE.	0 to 59.9(n), where "9(n)" is the precision of interval fractional seconds
TIMEZONE_HOUR (See Note that follows.)	-12 to 14 (This range accommodates daylight savings time changes.) Not applicable for DATE.	Not applicable
TIMEZONE_MINUTE (See Note that	00 to 59. Not applicable for DATE.	Not applicable

Datetime Field	Valid Values for Datetime	Valid Values for INTERVAL
follows.)		
Note: TIMEZONE_HOUR and TIMEZONE_MINUTE are specified together and interpreted as an entity in the format +|- hh:mm, with values ranging from -12:59 to +14:00.		
TIMEZONE_REGION	Query the TZNAME column of the V$TIMEZONE_NAMES data dictionary view. Not applicable for DATE.	Not applicable
TIMEZONE_ABBR	Query the TZABBREV column of the V$TIMEZONE_NAMES data dictionary view. Not applicable for DATE.	Not applicable

data tipleri de bulunmaktadır.

Bu datatiplerinde genel olarak bilinen data tiplerinden farklı olarak ROWID adı verilen bir data tipi de bulunmaktadır. Bu konu üzerinde biraz durmanın özellikle programcılar açısından yararlı olacağını düşünüyoruz.

11.2. ROWID, UROWID

ROWID'ler her tabloda bulunan ve bir column içerisinde tutulmaya gerek olmayan data tipleridir. Veritabanındaki herhangi bir tablodaki herhangi bir satırı tanımlamak için kullanılır. (her bir satır için unique bir ROWID vardır). Rowid, bir satırın fiziksel adresini göstermezler ama bir satırı tanımlamak adına kullanılabilmektedirler. Rowid, bu özelliği ve biricik bir numaraya sahip olması nedeniyle en hızlı biçimde bir satırı tanımlamak için kullanılmaktadır.

> ***Not:*** Oracle 8.1 sürümü ile birlikte Universal ROWID (UROWID) adı verilen bir yapıyı da tutmaya başladı. UROWID sayesinde Oracle-dışı tabloların ROWID bilgilerini de tutmak mümkün oldu. Bunun için initSID.ora parametrenizde COMPATIBLE değişkeni 8.1 ya da üstü olmak zorundadır.

Extended ve Restricted olmak üzere 2 adet ROWID tipi bulunmaktadır. Extended olanı 10 byte'lık bir yer tutar. ROWID, 6 hanelik Data Object Number (32 bit) + 3 hanelik Relative File Number (10 bit) + 6 hanelik Block Number (22 bit) + 3 hanelik Row Number'dan (16 bit) oluşmaktadır.

Data Object Number, bir veritabanı içerisinde tablo, indeks gibi bir nesne yaratılır yaratılmaz atanan numaradır.
Relative File Number ise tabelspace içindeki her dosyaya atanan unique numaradır.
Block Number ise row'u içeren block'un bulunduğu yerin (pozisyonun) numarasıdır.
Row Number da block header içindeki row directory slotta row'un bulunduğu yeri tutmaktadır.

Herhangi bir tablodan bir veri seçersek (tablonun içinde mutlaka veri olmalıdır) aşağıdakine benzer bir yapı göreceğiz:

SQL>

Oracle8'den önce ise Restricted ROWID kullanılmaktadır. Burada 8 hanelik bir Block number'ı, 4 hanelik Row number'ı ve 4 hanelik File Number'ı tutulmaktadır.

Row'ların gerçek yerini ROWID parametresini kullanarak bulabiliriz. Yeri gelmişken bir row'un nelerden oluştuğunu anlatalım. Bir row iki önemli parçadan oluşmaktadır.

- Row Header
- Row Data

Row Header'ında, bir row'da kaç tane column olduğunu, row'un lock durumunda olup olmadığını bildiren bilgiler tutulmaktadır.

Row Data kısmında ise column length ve column value değerleri tutulmaktadır. Row data'ları veritabanında database block yapılarında tutulmaktadır.

Oracle'da 4 değişik tablo tipinden söz edebiliriz:

i) Regular Tablo
ii) Partitioned Tablo
iii) Index-Organised Tablo ve
iv) Clustered Tablo

Bunlara sırasıyla bakarsak

i) Regular Tablo aslında en çok bilinen ve CREATE TABLE komutu ile yarattığımız

tablodan başka birşey değildir. Bu yapıdaki tablo içerisinde bulunan verilere bir Oracle DBA çok da müdahale edebilir durumda değildir. Tablo yapısına göre bilgiler belirli bir mantıkla sıralanabilirler.

ii) Partitioned Tablolar ise, DBA'e daha fazla yönetme ve bu yüzden de performans artırma olanağı sunabilmektedir. Bu yapıdaki tablolar, Range Partitioning, Hash Partitioning, Composite Partitioning ya da List Partitioning partition'lama yöntemlerinden birisiyle 1 ya da 1'den çok parçaya ayrılabilirler. Her bir partition, bir segment olduğu için de bunları farklı tablespace'lere koymak ve bu tablespace'lere karşılık gelen datafile'ları farklı disklere koyarak performansı artırmak mümkün olmaktadır. Partition'ları özellike büyük tablolar için kullanmak akıllıca olacaktır.

iii) Index-Organised Tablolar, hem tablo içindeki bilgileri hem de ayrıca kullanılan index yapısını kendi üzerinde barındırmaktadır. Böylece, hem bir tabloda hem de index'inde yer alan column adlarının yinelenmesinden kurtulunmuş olur. Aynı Primary Key index'lerinde olduğu gibi B-Tree yapısına uygun indeksleme yapacaktır. Index-Organised Tablolar, tam karşılık bulunabilecek sorgularda performanslı çalışmaktadırlar.

iv) Clustered Tablolar ise, bir tablo ya da birden çok benzer data block içeren tablodan oluşmaktadır. Bunları kümelendirme nedeni, benzer column'ları barındırmaları ya da sıklıkla birlikte kullanılmalarından kaynaklanmaktadır. Cluster Key adı verilen bir ya da birden çok column'ı içeren bir key, birlikte tutulacak veri satırlarını belirlemekte kullanılır. Cluster içindeki tabloların columları, cluster key'lere karşılık gelmektedir. Cluster içindeki tablolar, regular tablolar nasıl değiştiriliyorsa öyle değiştirilir. Cluster içindeki bir tablo'nun Primary Key'i ille de Cluster Key içerisinde yer alacak diye bir kural yoktur. Primary Key'in konulduğu column, Cluster Key içindeki column'larda olabilir de olmayabilir de. Cluster'lar performans artırmak için kurulan yapılardır ama cluster içindeki tablolardan rastgele seçimlerde çok hızlı çalışmaktadırlar. Tüm tabloyu taratacak bir sql query ise aksine daha yavaş çalışmasına yol açacaktır.

Tabloları yaratmak için kullanılan SQL komutu CREATE TABLE'dır. Aynı tablespace'lerdeki gibi isterseniz tablo yaratırken STORAGE değişkenini kullanarak, bir tablonun nasıl büyümesini istediğinizi kontrol edebilirsiniz. STORAGE parametresi ile, sizin isteminiz dışında tabloya fazla veri girilmesini de engelleyebilirsiniz. STORAGE parametresi kullanılmazsa varsayılan değerlere

göre bir STORAGE yapısı oluşacaktır.

Tabloları mutlaka Tablespace'ler içerisine koymanız gerekmektedir. Tablo yaratmak için CREATE TABLE ya da CREATE ANY TABLE yetkisine sahip bir kullanıcı ile giriş yapmanız gerekmektedir. Bunlardan ikincisi, başka bir kullanıcı şeması içinde tablo yaratmak istiyorsanız gerekli olan yetkidir.

Aşağıda tablo yaratmaya bir örnek görüyoruz.

```
SQL>CREATE TABLE personel(
sicilno  NUMBER(6) PRIMARY KEY,
adi  VARCHAR2(20),
soyadi  VARCHAR2(20),
dog_tar  DATE)
STORAGE(INITIAL 100K NEXT 100K
      PCTINCREASE 0      MINEXTENTS 1 MAXEXTENTS 50)
  TABLESPACE userdata;
```

Yukarıdaki SQL komutu sonucu userdata Tablespace içinde personel adında bir tablo yaratmış olduk. STORAGE değişkeni sayesinde de bu tablonun veri depolaması sırasında nelerin olması gerektiğini tanımladık. INITIAL değişkeni ile tablo için ayrılacak ilk extent'in boyutunun ne olacağını saptamış olursunuz. NEXT ise, ilk extent dolduktan sonra ayrılacak 2., 3. extent'lerin boyutunu belli etmektedir. MINEXTENTS 1 diyerek en az 1 extent almasını ve MAXEXTENTS 50 diyerek de en çok 50 extent'lik (50x100K'lık) bir alanın bu tablo için ayrılabileceğini saptamış olursunuz. PCTINCREASE 0 diyerek de, ilk exten sonrasında NEXT değerinin öncekine göre %0 büyüyeceğini yani değişmeyeceğini saptamış oldunuz.

Yaratmış olduğunuz bir tablonun STORAGE değişkenleri ile ilgili olarak bir değişikliğe gitmek isteyebilirsiniz. Bu durumda aşağıdaki gibi komut işinize yarayacaktır.

```
ALTER TABLE personel
PCTFREE 20
PCTUSED 50
STORAGE(NEXT 250K
MINEXTENTS 5 MAXEXTENTS 100);
```

Bu değişiklik sonucunda daha önce 100K değeri ile büyüyen Extent'ler NEXT 250 dememiz

sonucu yeni bir extent gerektiğinde PCTICREASE değerine de bakarak yeni değerini alacaktır.
MINEXTENTS değerini değiştirmek ise bir koşula bağlıdır. Şu an tabloda bulunan extent sayısına eşit ya da bu sayıdan küçük bir değer atanabilir. Bu nedenle bu yapı var olan extent'ler için önem taşımasa da TRUNCATE komutundan sonra tablo içindeki bilgileri boşaltmamız sonucu artık verdiğimiz değeri önemseyecektir.
MAXEXTENTS ile de tablomuzu şu anki extent değerlerinden daha büyük bir değere atanarak daha fazla genişleme olanağına kavuşmasını sağlamıştır. İsterseniz UNLIMITED değeri vererek bu sınırlamayı da ortadan kaldırabilirsiniz.

Not: INITIAL değerini değiştirme olanağınız bulunmamaktadır ve NEXT değerini data block'un katı şeklinde vermeniz iyi olacaktır. Siz böyle vermesiniz de Oracle Server, data block'un yapısına getirecek şekilde bir yuvarlama yapacaktır.

Tablolar, segmentler ve extenlerle ilgili önemli view'leri şöyle sıralayabiliriz.

DBA_OBJECTS
DBA_TABLES
DBA_EXTENTS
DBA_SEGMENTS
DBA_UNUSED_COL_TABS

11.3. Indeksler

Index'ler, tablolardaki verilere daha hızlı ulaşmamızı sağlayan mantıksal araçlardır. Tek bir sütuna konulanları dışında (single column index) birden çok sütuna konularak bütünleişk bir yapı oluşturan indeksler de (composite ya da concatenated index) yaratılabilmektedir. Personel tablomuzda adi ve soyadi sütünlarına konulacak bir index buna örnektir. En fazla 32 sütuna konulabilirler.

Bunlar dışında,

UNIQUE
B_TREE
BITMAP
DOMAIN index yapıları da bulunmaktadır. Bunlara bakacak olursak;

Unique indexler, tablo yaratılırken de konulabilecek bir index türüdür ve bir sütunda bulanan tüm verilerin biricik olmasını yani başka bir satırda yinelenmemesini sağlarlar.

B-Tree Index'ler ise tamamen gruplanma ve dallanma yapısında yaratılan index'lerdir. Herhangi bir sütuna konulacak index sonucunda en başa gruplanmayı nasıl yapacağına ilişkin bilginin olduğu root bölümü yer alır. Daha sonra, dallanmalar başlar ve en allta da her grup altındaki yapılar içinde (leaf) hangi satırların bilgisinin tutulduğu yer alır. Böylece WHERE adi='Fethiye' tarzındaki bir sorgu örneğin D-H arasındaki grubu bulur buradan F ile başlayanlara gider ve Fethiye adındaki kişileri buradan arar. Bu yapıda her satırın ROWID'si key olarak kullanılmaktadır.

Bitmap index'ler ise ayrımların daha az olduğu sütünlar için daha uygun indekslerdir. Cinsiyet, Medeni Durum gibi sütunlarda kullanılması daha uygun olacaktır. Bununla birlikte satır sayısının da çok olması beklenir. Ayrıca query'leriniz içerisinde WHERE'den sonra çok fazla koşul var ve OR operatörü ile çok sık bağladığınız sütunlar varsa bunlar için de Bitmap index yaratabilirsiniz. Bir diğer durum ise, sütünlarda verilerde ya hiç ya da çok az değişiklik yapılıyorsa konulması durumudur.

```
CREATE INDEX personel_adi_idxreg
ON personel(adi)
PCTFREE 30
STORAGE(INITIAL 100K NEXT 100K
        PCTINCREASE 0    MAXEXTENTS 100)
TABLESPACE indx_dta;
```

şeklindeki bir komut personel tablonuzdaki adi sütunu üzerine regular bir index koymanızı sağlayacaktır. Bu index indx_dta adındaki farkli bir tablespace üzerinde yaratıldığı için de performansınızı artıracaktır.

Yaratmak istediğiniz index türü BITMAP ise bu durumda
CREATE BITMAP INDEX ...
şeklinde bir komut vermeniz gerekecektir. Bitmap'lerin tutacağı yeri saptamak adına CREATE_BITMAP_AREA_SIZE adlı değişkeni tanımlamanız gerekmektedir.

ALTER INDEX index_adi STORAGE(...) komutu ile de daha önce yaratmış olduğunuz bir indeksin STORAGE parametrelerini değiştirebilirsiniz.

ALTER INDEX index_adi REBUILD ONLINE; komutu vererek index'inizi yeniden yaratabilirsiniz. Bu komutu verdiğinizde artık tablo düzeyinde lock olmamamaktadır. Ama bir temporary table'daki index'i rebuild edemezsiniz, kullanılmayan alanları yeniden kullanmayı sağlayamazsınız.

ALTER INDEX index_adi COALESCE; komutu ise oldukça kullanışlı bir komuttur. Bu da REBUILD gibi performans harcatan bir işlemdir ama indeksi defragmente edeceği için kullanışlıdır.

Bunun dışında diğer önemli bir komut ise
ANALYZE INDEX index_adi
VALIDATE STRUCTURE; komutudur. Bu komut ile block corruption var mı diye bakılır ve daha da önemlisi INDEX_STATS view'i içerisindeki bilgileri günceller. Böylece index kullanım hızı artar. INDEX_STATS içerisindeki DEL_LF_ROWS sütunu silinmiş satırların yüzdesini belli etmektedir. Eğer bu oran %30 geçmişse ki bu büyük bir yüzdedir bu durumda bu indeksi yeniden yaratmak işlerinizi oldukça hızlandıracaktır.

DROP INDEX indeksadi komutunu vererek yaratılmış olan bir indeksi silebilirsiniz.
Index silmenin iki önemli nedeni olabilir. Biri bu indekse artık gerek duymamanızdır. Diğeri ise toplu veri yüklemede indeks çok fazla performans kaybına yol açacağı için bunu önlemek istemenizdir. Yükleme işi bittikten sonra tabii ki bu indeksi yeniden yaratabilirsiniz.

Oracle 9i ile başlayarak yaratılmış indekslerin kullanımı V$OBJECT_USAGE adlı view'de izlenebilmektedir. Eğer monitörleme aşamasında bu view'daki USED column'unda NO görünuyorsa bu indeks hiç kullanılmamış anlamanına gelmektedir ve bu indeksi drop etmenizde hiçbir sakınca yok demektir.

Bir indeksin izlenmeye başlaması için şu komutu vermek gerekir:

ALTER INDEX indeksadi MONITORING USAGE;

DBA_INDEXES ve DBA_IND_COLUMNS viewları da bakılması gereken önemli view'lardır.

12. Güvenlik

Veritabanı yaratma sürecinden başlayarak birçok alanda güvenlik kademeleri oluşturmanız mümkündür. Hatta en başta Oracle Server'ı kuracağınız yapının yani işletim sisteminin güvenliğini sağlamakla işe başlayabilirsiniz.

12.1. İşletim Sistemi Güvenliği

Bir veritabanın güvenliği en başta bu veritabanın ve Oracle Server'ın bulunduğu işletim sisteminin güveliğinden başlar. Siz her türlü kontrolünüzü yapmış olsanız da bir Oracle'ın ve veritabanının bulunduğu ortamda açık varsa hem veritabanınız hem de Oracle Server'ınız risk altında demektir. İşletim Sistemlerinde root ya da administrator yetkisindeki bir kullanıcı her türlü dosyaya yazma hakkı olan ya da sunucu üzerindeki farklı kullanıcıların yetkisi ile bağlanma yetkisine sahip bir kullanıcıdır.

Önce root ya da administrator kullanıcısının şifresini, daha sonra da oracle kullanıcısının şifresinin güvenli ellerde olmasını sağlamalısınız.

Bu sunucuya kimler erişebilir ve erişim nasıl olmalıdır?

Oracle'ın bulunduğu sunucu veritabanı çalıştırmak için adanmış bir sunucu olabilir. Bu durumda işiniz oldukça kolaydır. Veritabanı sunucusuna sadece belirli bir uygulamadan istek gelecektir. Bu durumda uygulama sunucusu dışındaki tüm isteklere kapatacak bir güvenlik mekanizması kurabilirsiniz. Öte yandan bir sunucu hem Oracle Server'ını hem veritabananı hem de diğer uygulamaları barındırıyor olabilir. Bu durumda da yine sunucuya erişebilir insanları kısıtlamakla işe başlamalısınız.

Erişim belirli portlar aracılığı ile belirli IP'lerden olacaktır. Bu yüzden routing tablosuna bu sunucuya hangi ip'lerden erişilebileceğini koymalısınız.

$ netstat –rn komutu ile hangi iplerden erişelebileceğini görüp ilgili ayarlamaları yapabilirsiniz.

/etc/hosts.allow ile hangi iplerden ftp ya da telnet/ssh bağlantısı yapılabileceğini de görerek gereksiz iplerin sunucuya erişimlerini engelleyebilirsiniz. (ssh türü güvenli bağlantıları zorlayarak,

kullanıcıların şifrelerinin basit programlarca başkalarının eline geçmesini engelleyebilirsiniz).

12.2. Veritabanı düzeyinde erişim kısıtlama

Veritabanı düzeyinde erişim kısıtlama değişik yöntemlerle yapılabilemektedir. Bunlardan ilki, hangi kullanıcıların veritabanında sql komutu çalıştırmasına izin vereceğinizle ilgilidir.

$ORACLE_HOME altındaki network/admin dizini içine yaratacağınız protocol.ora dosyası ile sql çalıştırmasını istediğiniz ipleri ekleyebelir, istemediklerinizi de çıkarabilirsiniz. Örnek bir protocol.ora dosyasını aşağıda görebilirsiniz.

Bundan sonra yapmanız gereken de veritabanı kullanıcılarınızla ilgilidir. Hangi kullanıcının veritabanına bağlanma şekli ne olsun, şifresi ne olsun vb. tüm ayarlamaları profile adını verdiğimiz bir yapı aracılığı ile kurabilirsiniz. Bir veritabanı kullanısı yaratırken ona bir profile atamazsanız varolan DEFAULT profile'ı bu kullanıcya atanır.DEFAULT profile'ının tüm değişkenleri sınırsız olarak tanımlansa da bir DBA olarak buradaki bazı bilgileri değiştirebilir böylece yeni yaratacağınız tüm kullanıcılara bu değerleri uygulayabilirsiniz. Ya da yeni profiller yaratıp bunları yaratacağınız kullanıcılara atayabilirsiniz.

```
CREATE PROFILE benimprofilim LIMIT
 FAILED_LOGIN_ATTEMPTS 5
PASSWORD_LOCK_TIME UNLIMITED
 PASSWORD_LIFE_TIME 150
 PASSWORD_VERIFY_FUNCTION verify_function
 PASSWORD_GRACE_TIME 5;
```

Yukarıda benimprofilim adını vererek yarattığımız profil'de
FAILED_LOGIN_ATTEMPTS 5 diyerek, kullanıcı 5 denemede de başarılı bir şekilde giriş yapamazsa, bağlantı kurmaya çalıştığı kullanıcının kilitleneceğini,
PASSWORD_LOCK_TIME UNLIMITED diyerek farklı günlerde başarısız giriş yaparsa bunu önemsemeyeceğimizi,

PASSWORD_LIFE_TIME 150 diyerek, 150 gün boyunca şifre değiştirmeden bu kullanıcının bağlanmasına izin vereceğimiz ama 150. günün sonunda hala şifre değiştirmezse kullanıcısını etkinsizleştireceğimizi,

PASSWORD_VERIFY_FUNCTION verify_function diyerek, utlpwdmg.sql çalıştırılması sonucu oluşan verify_function şifre yönetme fonsiyonuna göre kullanıcının şifresini değiştirmeye izin vereceğimizi (bu fonksiyon şifrenin en az 4 karakter olmasını, kullanıcı adı ile aynı olmamasını vb kontrolleri kapsıyor),

PASSWORD_GRACE_TIME 5 ile de şifre etkinleştirildikten sonra 5 gün boyunca şifrenin değiştirilmesi için tanınan süreyi

tanımlamış olduk.

Tüm bu değişkenler belirli kullanıcılara özgü belirli profiller yaratıp onlara atamamızı sağlamakta. Bilgi İşlem Personelinize daha geniş yetki verirken, daha az dikkatli kullanıcılara biraz daha zorlayıcı tanımlamalar yaparak sisteminiz güveliğini artırmak gibi seçenekleriniz oluşmakta böylece.

Tüm bunlarla uğraşmak zahmetli ve sürekli yönetim gerektireceğinden DEFAULT profile'ındaki bazı değişkenleri isteğinize göre uyarlamayı da tercih edebilirsiniz.

ALTER PROFILE DEFAULT LIMIT diye başlayan ve değişmesini istediğiniz değişkenler ve yeni değerleri ile artık bir kullanıcı yaratırken bu değerlere göre şifre kontrol mekanizması kurabilirsiniz.

Bir kullanıcı yaratmak ve bu kullanıcının ne tür yetkileri olduğunu saptamak en önemli konulardan ve oldukça kolay yönetilecek konulardan biridir.

CREATE USER kul1
IDENTIFED BY sifreyiburadaverin
DEFAULT TABLESPACE personel
TEMPORARY TABLESPACE temp
QUOTA 100M ON personel;
diyerek kul1 adında bir kullanıcıyı sifresi sifreyiburadaverin şeklinde vererek ve çalışacağı birincil tablespace'in personel, temporary tablespace'in ise temp olduğunu tanımlayarak ve personel tablespace üzerinde sadece 100M'lık bilgi bulundurabileceğini belirterek yaratmış oldunuz.

PROFILE benimprofilim satırını ekleseydiniz, daha önce yaratmış olduğunuz benimprofilim adlı

profil’deki yapılandırmaya göre de şifre, bağlanma ve kaynak yönetimi tanımlamış olacaktınız.

Eğer PASSWORD EXPIRE diye bir satır girilmiş olsaydı, kullanıcı ilk bağlantıyı kurduktan sonra yeni bir şifre tanımlamak zorunda kalacaktı.

QUOTA için belirli bir miktar belirtebileceğiniz gibi, UNLIMITED diyerek kullanıcının ilgili tablespace üzerinde istediği kadar yere sahip olmasını sağlayacaktınız. Tabii ki, ister UNLIMITED deyin, ister bir limit verin, ilgili tablespace’in boş alanından daha fazla yeriniz olmayacaktır.

ALTER USER kul1 IDENTIFED BY yenisifre;

komutu ile bu kullanıcının şifresini ilerleyen zamanlarda değiştirebileceğiniz gibi,

ALTER USER kul1
QUOTA 0 ON personel;

diyerek kullanıcının personel tablespace’i üzerindeki kotasını sıfırlayıp kontrolsüz veri girişine engel olabilirsiniz.

DROP USER kul1 ise bu kullanıcıyı veritabanınızdan tamamen uzaklaştıracaktır. Bu komut kullanıcıyı silse de onun yarattığı nesneler hala veritabanında duracaktır. Bunları da uçurmak için

DROP USER kul1 CASCADE;

komutunu vermelisiniz.

Güvenlik ayarlarıyla ilgili diğer bir konu da kullanıcıların nesneler üzerinde neler yapabileceklerini belirtmemize yarayan ve privileges olarak adlandırılan yetkilerdir.

GRANT komut TO kullanıcı;

biçimindeki bir komutla bir ya da birden çok komutu bir ya da birden çok kullanıya verebilirsiniz. Örneğimize bakarsak,

GRANT CREATE TABLE TO kul1;

komutu kul1 adlı kullanıcıya tablo yaratma yetkisi verdiğimiz anlamına gelecektir.

GRANT CREATE TABLE, CREATE ANY INDEX TO kul1, kul2 ise

Kul1 ve kul2 adlı kullanıcılara hem CREATE TABLE hem de CREATE ANY INDEX yetkilerini verdiğinizi göstermektedir.

CREATE TABLE, CREATE INDEX'i de içeren bir yetkidir. Bunun dışında CREATE TABLE, CREATE PROCEDURE gibi yetkiler DROP yetkisini de otomatik olarak vermenize yol açar.

Bir önemli konu ise WITH ADMIN OPTION'udur. Bir DBA olarak, ya da bir kullanıcı olarak kendi yetkinizi, başkasına da kullandırabilirsiniz.

GRANT CREATE ANY TABLE to kul1 WITH ADMIN OPTION yetkisi bir DBA'in kul1'e verdiği bir yetki ise kul1 de bu yetkiyi başka bir kullanıcıya verebilir demektir. Daha sonra REVOKE ile bu yetkiyi kul1'den geri almış olsanız bile, kul1 bu yetkiyi başkasına da kullandırmışsa ondan bu yetki alınmamış olacaktır. Bu ise sizin kontrolünüz dışında güvenlik sorunları yaratabilecek bir durumdur. Bu yüzden hem DBA'lerin hem de yetki sahibi diğer kullanıcıların WITH ADMIN OPTION seçeneğini çok dikkatli kullanmaları gerekmektedir.

SYSDBA yetkisi SYSOPER'in de üzerinde olan ve bir veritabanda her türlü yetkiye sahip en üst kullanıcıdır. Bu yüzden bu yetkilerinizi paylaşırken ve bazı kritik komutları verirken çok dikkatli olmanız gerekmektedir.

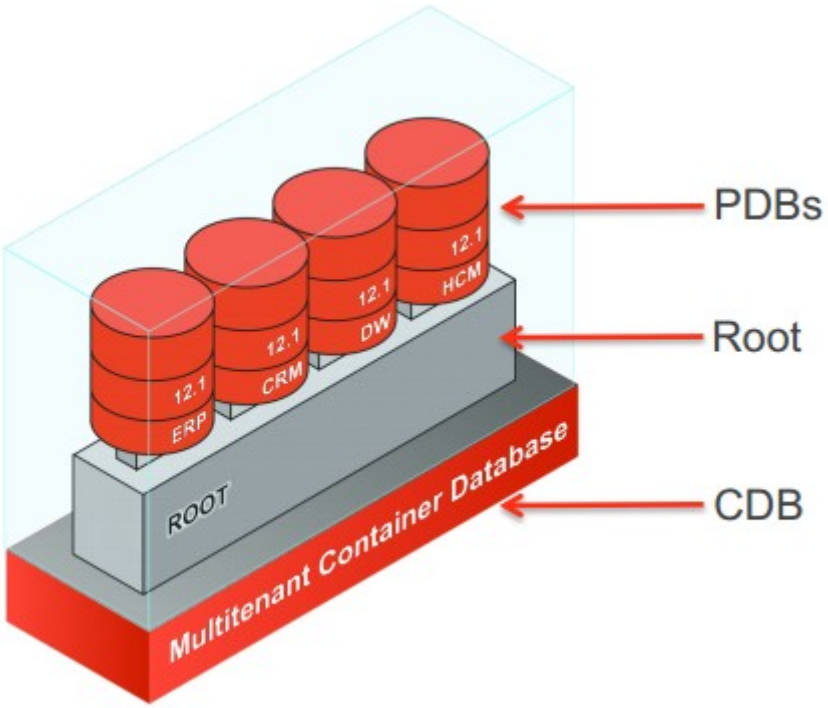
SYSDBA yetkisine sahip bir kullanıcı SYSOPER kullanısının yetkilerinin tümüne (shutdown, startup, alter database open, alter database archivelog vb) WITH ADMIN OPTION hakkıyla sahiptir. Bunun dışındaki yetkileri şöyle sıralayabiliriz:
CREATE DATABASE,
ALTER DATABASE BEGIN/END BACKUP,
ALTER DATABASE OPEN RESTRICTED SESSION,
RECOVER DATABASE UNTIL, vb.

13. ORACLE12C ile GELEN YENİ ÖZELLİKLER

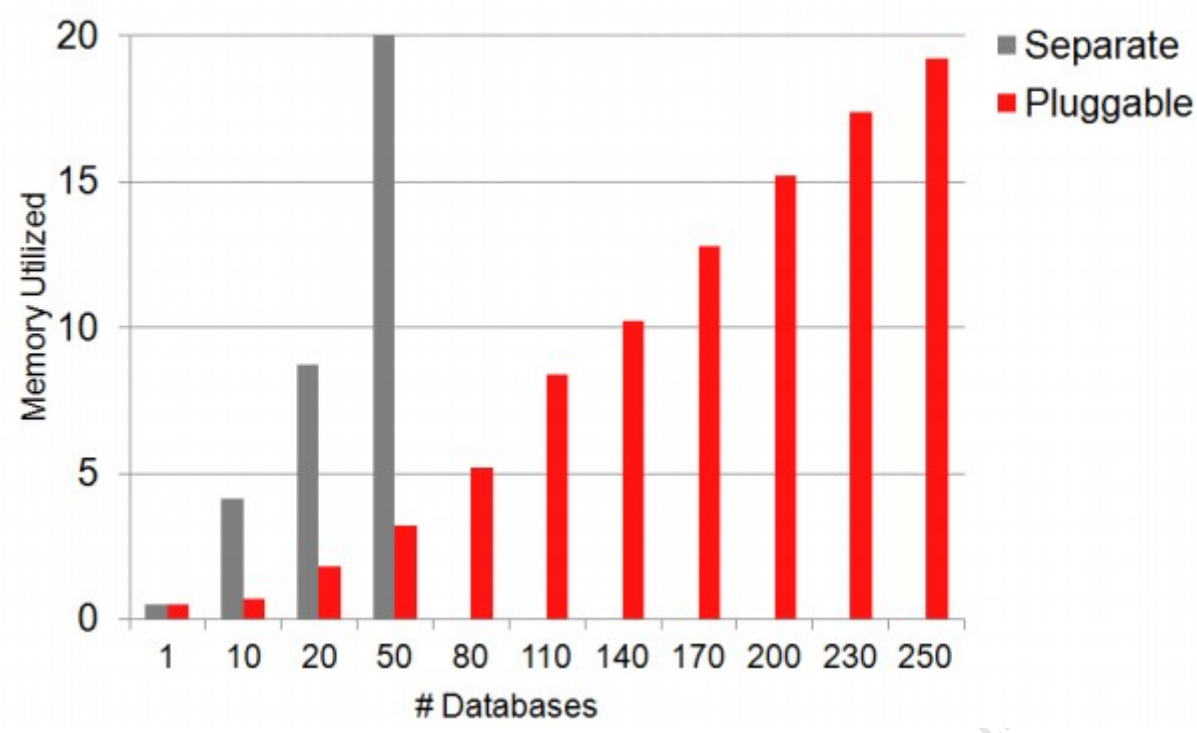
Oracle12c sürümü veritabanını Bulut ortama taşımak amaçlı yapılan yapısal bir değişikliği içermektedir. Aşağıda 12c ile gelen yeni onlarca özellikten bir kısmını görmekteseniz.

Oracle Multitenant

Oracle 12c ile birlikte gelen ve bu sürümle birlikte gelen en önemli özelliklerden biri olan Oracle Multitenant, bir multitenant **Container Database**'in(**CDB**) bir çok **Pluggable Database(PDB)** tutmasına olanak sağlayan bir mimaridir. Oracle Multitenant, Oracle Real Aplication Clusters ve Oracle Active Data Guard gibi diğer özelliklerle uyumludur.



Tek bir CDB'de tutulan bütün PDB'ler sistem belleği gibi pek çok sistem kaynağını ortak olarak kullanır. Bu eski mimariye göre çok daha fazla PDB'nin tutulmasına olanak sağlamaktadır. Yeni PDB oluşturmak, bu databaseleri CDB'ler arasında taşımak ve bunları kopyalamak yeni SQL komutlarıyla saniyeler içinde yapılabilmektedir. Aşağıdaki grafikte de görüldüğü gibi multitenant mimarisi ile birlikte, aynı sistem üzerinde eski mimariye oranla 5 kat daha fazla database'in kullanılması mümkündür. Bunun sebebi ise, eski mimaride oluşturulan her veritabanı için sistem kaynakları ayrı ayrı kullanılırken, multitenant mimarisinde sistem kaynakları sadece CDB tarafından kullanılır.



Bir CBD'ye patch uygulandığında, barındırdığı bütün PDB'lerde de bu patch uygulanmış olur. Tek bir PDB'nin patchlerinin uygulanacağı durumda ise bu database'in CDB'den çıkarılıp patchleri uygulanıp tekrar CDB bağlandığında patchler kolay bir şekilde uygulanmış olur.

Oracle 12c ile gelen Resource Manager ile CDB'de bulunan PDB'ler için kullanılacak sistem kaynakları rahat bir şekilde kontrol edilebilir.

Multitanent mimarisiyle birlikte yeni gelen bir özellik ise yedeklerin CDB seviyesinde, tek bir yedek ile ister bütün PDB'lerde, ister herhangi bir PDB'de Point-in-time recovery yapılabilmesidir. Bu sayede süreklilikten hiçbir ödün verilmezken yedekleme sürelerinde ciddi bir azalma sağlanmıştır.

Oracle 12c'de 2 çeşit kullanıcı tipi vardır. Sadece oluşturulduğu PDB'ye bağlanabilen yerel kullanıcılar ve 'Create Session' yetkisi olan her PDB'ye bağlanabilen ve CDB'de oluşturulan bir genel kullanıcı. Genel kullanıcı'ya '**Set Container**' yetkisi durumunda herhangi bir PDB tek bir komut ile CDB'ye dönüştürülebilir.

```
alter session set container = PDB_01
```

Aktif Bir Datafile'ın Online Olarak Yeniden Adlandırılması Ve Lokasyonunun Değiştirilmesi

Önceki sürümlerin aksine, 12c sürümünde datafile'ların taşınması veya isimlerinin değiştirilmesi işlemleri, eski sürümlerde uygulanması gereken bazı adımlar uygulanmadan gerçekleştirilebilir. Ör. Tablespace'in önce READ ONLY mode alınıp daha sonradan datafile'ın offline konuma almak gibi işlemler 12c'de gerekmemektedir. 12c'de bir datafile'ı taşımak veya ismini değiştirmek basit bir "*ALTER DATABASE MOVE DATAFILE"* SQL sorgusu ile yapılabilir. Datafile'ın taşınması sırasında kullanıcılar SQL sorguları yazabilir ve hatta DML ve DDL işlemleri bile yapabilirler.

Yeniden adlandırma:

```
SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users01.dbf' TO
'/u00/data/users_01.dbf';
```

Datafile'ı ASM'e taşıma:

```
SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users_01.dbf' TO '+DG_DATA';
```

Datafile'ı bir ASM diskten diğerine taşıma:

```
SQL> ALTER DATABASE MOVE DATAFILE '+DG_DATA/DBNAME/DATAFILE/users_01.dbf '
TO '+DG_DATA_02';
```

Eğer mevcut klasörde aynı isimde bir başka datafile varsa üstüne yazarak taşıma:

```
SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users_01.dbf' TO
'/u00/data_new/users_01.dbf' REUSE;
```

Aynı isimde bir datafile mevcut ise eski kopyayıda tutarak taşıma:

```
SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users_01.dbf' TO
'/u00/data_new/users_01.dbf' KEEP;
```

Taşıma işlemi devam ederken, işlemi dinamik bir şekilde gözlemlemek v$session_longops tablosuna sorgu çekerek mümkündür. Ayrıca alert.log dosyasıyla da mevcut işlem gözlemlenebilir.

Online Migration Of Table Partition Or Sub-Partition

Bir tablonun bölümlerinin(partition) veya alt bölümlerinin(sub-partition) taşınması işlemi 12c'de eskiden olduğu kadar karmaşık değildir. Eski sürümlerde bölümlenmemiş bir tablonun (non-partition) taşınmasında olduğu gibi, 12c'de bir table partition farklı bir tablespace'e online veya offline olarak taşınabilir. Online taşıma işleminde ilgili partition'da DML işlemleri hiç bir kesintiye uğramazken offline taşıma işleminde herhangi bir DML işlemi gerçekleştirilemez.

```
SQL> ALTER TABLE table_name  MOVE PARTITION|SUBPARTITION partition_name  TO
tablespace tablespace_name;
SQL> ALTER TABLE table_name  MOVE PARTITION|SUBPARTITION partition_name  TO
tablespace tablespace_name UPDATE INDEXES ONLINE;
```

Yukarıdaki örneklerin ilkinde table partition yeni bir tablespace'e offline olarak taşınırken ikinci örnekte local/global indexler de korunarak online bir şekilde taşınmaktadır. UPDATE INDEXES komutu ile tablodaki local/global indexlerin kullanılamaz hale gelmesinin önüne geçilmiştir. Ancak bu seçenek performansta düşüşe sebep olur ve taşınan partition'ın büyüklüğüne göre büyük redo logların oluşmasına sebep olabilir.

Invisible Columns

Oracle 11g R1 ile birlikte, Oracle görünmez indexler ve sanal sütunlar konusunda bazı özellikleri duyurmuştur. 12c ile birlikte bu özellikler bir adım öteye götürülerek görünmez sütun kavramı getirilmiştir. 12c'de tabloda görünmez bir sütun oluşturulabilir ve bu sütun sorguda belirtilmediği veya tablo özelliklerinde tanımlanmadığı sürece genel sorgularda görülmez. Bir sütunu görünmez yapmak veya tekrar görünür yapmak son derece kolay bir işlemdir.

```
SQL> CREATE TABLE emp (
        eno number(6),
        ename name varchar2(40),
        sal number(9) INVISIBLE
        );

SQL> ALTER TABLE emp MODIFY (sal visible);
```

Sanal bir sütun da görünmez olarak tanımlanabilmektedir fakat geçici tablolar(temporary tables), harici tablolar(external tables) ve küme tablolar(cluster tables) görünmez sütun özelliğini desteklemez.

Multiple Indexes On The Same Column

Oracle 12c'den önceki sürümlerde bir veya birkaç sütun üzerinde birden fazla index oluşturulamaz. 12c ile birlikte index tipinin farklı olması koşuluyla bir veya birkaç sütun üzerinde birden fazla index oluşturulabilir. Fakat sadece tek bir tip index kullanılabilir/görünür olabilir. Bunu test etmek için optimizer_use_use_invisible_indexes parametresinin 'true' olması gerekir.

```
SQL> CREATE INDEX emp_ind1 ON EMP(ENO,ENAME);
SQL> CREATE BITMAP INDEX emp_ind2 ON EMP(ENO,ENAME) INVISIBLE;
```

DDL Logging

Daha önceki sürümlerde olmayan ve 12c ile gelen özelliklerden biri ise DDL komutlarının xml'e çevirilip loglanmasıdır. Drop veya create komutu çalıştırıldığında bunun kim tarafından ve ne zaman çalıştırıldığının bilinmesi bazı durumlarda ciddi bir avantaj oluşturmaktadır. ENABLE_DDL_LOGGING başlangıç parametresinin aktifleştirilmesi ile bu özellik kullanılabilir hale gelir. Parametre database veya session seviyesinde ayarlanabilmektedir. Bu parametre seçili olduğunda tüm DDL komutları $ORACLE_BASE/diag/rdbms/DBNAME/log|ddl klasöründe xml dosyası olarak tutulur. Bu xml dosyası DLL komutu, IP adresi, komutun girildiği zaman gibi bilgileri saklar.

```
SQL> ALTER SYSTEM|SESSION SET ENABLE_DDL_LOGGING=TRUE;
```

Temporary Undo

Oracle database belirli bilgileri barındıran SYSTEM, SYSAUX, UNDO ve TEMP gibi bazı sisteme bağlı tablespacelere sahiptir. Eski sürümlerde geçici tabloların(temporary table) oluşturduğu undo kayıtları undo tablespace'de tututlurdu. 12c ile birlikte gelen temporary undo özelliği ile temporary undo kayıtları artık undo tablespace yerine geçici bir tabloda tutulabilir. Bu özelliğin temel faydası ise undo tablespace'in boyutunun gereksiz yere büyümemesi ve bilginin redo loglarda tutulmamasından ötürü daha az redo verisinin oluşmasıdır. Bu özellik database veya session seviyesinde ayarlanabilmektedir.

```
SQL> ALTER SESSION SET TEMP_UNDO_ENABLE=TRUE;
SQL> ALTER SYSTEM|SESSION SET TEMP_UNDO_ENABLED=FALSE;
```

Backup Specific User Privilege

12c ile birlikte SYSBACKUP yetkisi, RMAN'de backup & recovery komutlarının çalıştırılması için yeni gelen bir özelliktir. Bu sayede database'de local bir kullanıcı oluşturulup bu kullanıcıya SYSBACKUP yetikisi verilir ve RMAN'de SYSDBA yetkisi olmadan backup & recovery işlemleri gerçekleştirebilir.

```
$ ./rman target "username/password as SYSBACKUP"
```

Sql Statements In RMAN

Oracle Database 12c RMAN'de herhangi bir SQL eki kullanılmadan herhangi bir SQL veya PL/SQL komutu çalıştırılabilir.

```
RMAN> SELECT username,machine FROM v$session;
RMAN> ALTER TABLESPACE users ADD DATAFILE SIZE 121m;
```

Table or Partition Recovery In RMAN

Oracle backup'ları iki temel kategoriye ayrılmıştır; mantıksal(logical) ve fiziksel(physical). İki backup tipinin de avantajları ve dezavantajları bulunmaktadır. Daha önceki sürümlerde mevcut physical backup'ı kullanarak bir tabloyu veya partition'ı onarmak mümkün değildi. 12c ile birlikte truncate veya drop edilmiş tablo veya partition'ı point-in-time veya belirli bir SCN'e göre RMAN backup'ından onarmak mümkün hale gelmiştir. RMAN ile bir tablo veya partition onarımı başlatıldığında olaylar aşşağıdaki gibi gelişir.

Tablo veya partition'ı onarmak için gerekli backupset belirtilir.

Onarım sırasında geçici bir yedek point-in-time database ayarlanır.

Gerekli tablo/partition datapump ile bir dumpfile'a export edilir.

İstenirse tablo/partition asıl veritabanına farklı bir isimle import edilir.

Bu işlemin nasıl yapılabileceğine dair bir örnek;

```
RMAN> connect target "username/password as SYSBACKUP";
RMAN> RECOVER TABLE username.tablename UNTIL TIME 'TIMESTAMP…'
        AUXILIARY DESTINATION '/u01/tablerecovery'
        DATAPUMP DESTINATION '/u01/dpump'
        DUMP FILE 'tablename.dmp'
        NOTABLEIMPORT   – Bu seçenek tablonun otomatik import edilmesini engeller
        REMAP TABLE 'username.tablename': 'username.new_table_name';    -- Bu seçenek ile
tablonun yeniden adlandırılması sağlanır.
```

Restricting PGA size

Oracle 12c'den önceki sürümlerde PGA boyutunu sınırlamak ve kontrol etmek için herhangi bir seçenek bulunmamaktadır. PGA_AGGREGATE_TARGET parametresiyle belirli bir boyut ayarlanabilmekteydi fakat Oracle PGA boyutunu gereksinimlere göre dinamik olarak arttırıp azaltmaktaydı. 12c'de Automatic PGA Management'ı aktifleştirerek, PGA_AGGREGATE_LIMIT parametresiyle PGA boyutuna belirli bir limit konulabilir.

```
SQL> ALTER SYSTEM SET PGA_AGGREGATE_LIMIT=2G;
SQL> ALTER SYSTEM SET PGA_AGGREGATE_LIMIT=0; --limiti kaldırır
```

Adding Multiple New Partition

Oracle 12c öncesi sürümlerde, bir tabloya yeni bir partition eklemek için tablonun halihazırda bölümlenmiş olması gerekmekteydi. Yeni bir partition eklemek için her bir partition için ALTER TABLE ADD PARTITION komutu çalıştırılmak zorundaydı. Oracle 12c, tek bir ALTER TABLE ADD PARTITION komutu ile çoklu partition ekleme seçeneği getirmiş ve büyük bir esneklik kazandırmıştır.

```
SQL> CREATE TABLE emp_part (eno number(8), ename varchar2(40), sal number (6))
PARTITION BY RANGE (sal)
        (PARTITION p1 VALUES LESS THAN (10000),
         PARTITION p2 VALUES LESS THAN (20000),
         PARTITION p3 VALUES LESS THAN (30000)
        );
```

Yeni partitionlar eklemek istersek:

```
SQL> ALTER TABLE emp_part ADD PARTITION
        PARTITION p4 VALUES LESS THAN (35000),
        PARTITION p5 VALUES LESS THAN (40000);
```

Data Pump Geliştirmeleri

Redo log oluşumunun engellenmesi

Yeni gelen TRANSFORM seçeneği Data Pump importlarında, redo log oluşumunun engellenmesi sağlanmıştır. TRANSFORM seçeneği ile birlikte DISABLE_ARCHIVE_LOGGING parametresi belirtildiğinde, import işlemi süresince import edilen objeye bağlı redo log oluşumu engellenebilir. Bu özellik büyük tablolar taşınırken büyük bir avantaj sağlar ve gereksiz redo logların oluşmasını engeller, böylece import işlemi de daha çabuk bir şekilde yapılmış olur. Bu parametre tablolara ve indexlere uygulanabilir.

```
$ ./impdp directory=dpump dumpfile=abcd.dmp logfile=abcd.log
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y
```

Viewlerin tabloya dönüştürülmesi

Yeni gelen VIEW_AS_TABLES seçeneği ile birlikte view verileri bir tabloya aktarılabilir. Aşşağıdaki örnekte bir export esnasında view verilerinin tabloya nasıl dönüştürüleceğine dair bir örnek bulunmaktadır.

```
$ ./expdp directory=dpump dumpfile=abcd.dmp logfile=abcd.log
views_as_tables=my_view:my_table
```

ASM Geliştirmeleri

Flex ASM

Tipik bir Grid Infrastructure kurulumunda her bir node kendi ASM instance'ına sahiptir ve bu instancelar bu node'da çalışan database için bir veri deposu görevi görür. Fakat bu düzenin kendine has kusurları bulunmaktadır. Örneğin, eğer bu node üzerindeki ASM instance'ında bir hata oluşursa bu node üzerinde çalışan bütün database'ler etkilenecektir. 12c bu ihtimale karşı Flex ASM özelliğiyle çözüm getirmektedir. Flex ASM tamamiyle farklı bir kavram ve mimaridir. Bir cluster içerisinde sadece birkaç ASM instance'ı çalışmalıdır. Eğer bir node'da ASM instance'ında hata oluşursa, Oracle Clusterware devamlılığı sağlamak için otomatik olarak ASM instance'ını farklı bir node'a taşımaya başlar. Ayrıca bu düzen node üzerinde çalışan instancelar arasında bir load-balancing özelliği de sunar. Flex ASM'in bir başka avantajı ise birkaç farklı node'da ayarlanabilir oluşudur. Eğer cluster kurulumunun bir parçası olarak Flex Cluster seçilirse Flex ASM özelliği de otomatik olarak Flex Cluster tarafından seçilir. Flex ASM cluster kurulumu sırasında seçilebileceği gibi ayrıca ASMCA'dan da standart bir cluster ortamı için Flex ASM seçeneğine geçilebilir.

```
$ ./asmcmd showclustermode
$ ./srvctl config asm
```

Yükseltilmiş ASM limitleri

11gR2 de bir ASM diskgrubunda maximum 63 ASM diski bulunabilirken 12cR1 ile birlikte bu rakam 511'e yükselmiştir. Ayrıca bir ASM diskinin boyutu da 20PB'den 12cR1 ile birlikte 32PB'a yükseltilmiştir.

ASM Disk Scrubbing

Yeni ASM Disk Scrubbing özelliği normal veya high redundancy düzeyindeki ASM diskgrubunda işlem görür ve o ASM diskgrubundaki bütün ASM diskler üzerindeki mantıksal veri bozulmalarını(logical data corruption) tarar ve eğer tespit edilirse ASM mirror diskleri kullanarak bu hatayı onarır. Disk scrubbing disk, diskgrub veya dosya seviyesinde uygulanabilir ve performansa etkileri önemsenmeyecek kadar düşüktür.

```
SQL> ALTER DISKGROUP dg_data SCRUB POWER LOW:HIGH:AUTO:MAX;
SQL> ALTER DISKGROUP dg_data SCRUB FILE
'+DG_DATA/MYDB/DATAFILE/filename.xxxx.xxxx'
                REPAIR POWER AUTO;
```

Grid Infrastructure Geliştirmeleri

Flex Clusters

Oracle 12c Clusterware kurulumu sırasında 2 tane cluster tipi seçeneği sunar: Standart Cluster ve Flex Cluster. Standart clusterda cluster'daki bütün nodelar birbirlerine entegredir ve birbirleri arasındaki iletişim private network sayesinde sağlanır ve diske direkt olarak ulaşabilirler. Diğer taraftan Flex Cluster'da 2 tip node mimarisi vardır, Hub ve Leaf node mimarileri. Hub

mimaride nodelar standart cluster'dakine benzer bir şekilde ayarlanır, private network ile birbirlerine bağlıdır ve direkt olarak diske ulaşırlar. Leaf mimaride ise node'lar disklere Hub nodeların üzerinden erişebilir.

Hub nodeların sayısı 64'e kadar çıkabilirken Leaf nodelarda böyle bir sınırlama yoktur(teorik olarak). Flex Clusterda Leaf node olmadan Hub node kurulabilirken, Hub node olmadan Leaf node oluşturulamaz. Tek bir Hub node ile birçok Leaf node olusturulabilir. Flex Cluster'da sadece Hub nodeların OCR/Voting disklere direkt olarak ulaşım yetkisi vardır. Bu çok büyük bir cluster ortam hazırlarken, network trafiğini düşürmesi ve standart clustera her zaman geçiş imkanı bırakmasıyla büyük avantajlar sağlar. Yeni bir cluster kurulurken Flex Cluster seçeneği ile kurulabileceği gibi standart bir cluster Flex Cluster'a da yükseltilebilir.

Bir standart clusterı Flex Cluster'a geçirmek için aşşağıdaki yollar izlenmelidir.

1) Mevcut cluster bilgisi aşşağıdaki komut ile öğrenilir:

```
$ ./crsctl get cluster mode status
```

2) root kullanıcısı ile aşşağıdaki komutlar çalıştırılır:

```
$ ./crsctl set cluster mode flex
$ ./crsctl stop crs
$ ./crsctl start crs –wait
```

3) İstenilen node ayarlaması yapılır.

```
$ ./crsctl get node role config
$ ./crsctl set node role hub|leaf
$ ./crsctl stop crs
$ ./crsctl start crs -wait
```

DİKKAT :

* Flex Cluster sonradan standart cluster moduna geçirilemez.
* Cluster node modunu değiştirmek cluster stack'in durup tekrar başlamasına sebep olur.
* GNS'nin ayarlandığından emin olunmalıdır.

OCR backup in ASM disk group

12c ile birlikte OCR bir ASM diskgrubunda yedeklenebilir ki bu da tüm nodeların OCR yedek dosyalarına ulaşımını sağlar. Bir OCR onarım durumunda hangi OCR'ın en son yedeğinin hangi nodedan alındığı bir sorun oluşturmaz. ASM'de alınan en son yedek herhangi bir nodedan belirlenebilir ve onarım işlemi gerçekleştirilebilir.

```
$ ./ocrconfig -backuploc +DG_OCR
```

14. IN-MEMORY DATABASE ÖZELLİĞİ

In-Memory kullanımı ile Oracle veritabanı hem Data Warehouse işlemlerini hem de OLTP işlemlerini aynı anda efektif olarak yapabilecektir. Bu işlemler kullanılması istenilen tablo, tablespace, index'lerin kullanılacak kolonlarını In-Memory alanına ekleyerek yapılabilecektir. In-Memory alanı SGA, yani veritabanının kullandığı memory alanındadır. Kullanıcı bu alanı SGA bilgilerine göre düzenler, açar veya kapatır. In-Memory alanına girilen tablolar ikinci bir yerde saklanacak şekilde kopyalanmazlar. Tek bir dosya gibi olurlar ama In-Memory kısmında saklanılar. Yani In-Memory, Buffer Cache'deki tabloyu alır ve nesnelerin hem row (buffer cache) hem de column formatlarda ('dual format' mimarisi ile) çalışacak şekilde tabloyu hazırlar. Row format; klasik olarak verilerin DB'de fiziksel seviyede tutulma şekli idi. Böylece OLTP o kayıta ait tüm kolonları row format ile erişebilmektedir. Column format ile Data Warehouse işlemlerini yapabilecek sistem geliştirildi. Sonuç olarak dual format yaklaşımı ile daha iyi performanslı ve iki işi aynı anda senkronize şekilde yapabilecek ayrıca hafıza'dan sadece tek bir yer kullanacak bir sistem geliştirildi. Dual-format mimarisi ile tablonun kopyasız tek nesnesi hiçbir şekilde ek bir hafıza alanı yemeden, ek bir senkronizasyon olmadan In-Memory'de işlem görür. Böylece veri ve analiz için ayrı ortamlar kurmak zorunda kalınmayacak, yönetme maliyeti düşecek ve sahip olunan veriler tüm amaçlar için tek bir yerde tutulacak ve buradan kullanılacak. Performans kazanımı avantajının yanında, eğer yanlış kurgulanmış ise buffer cache'in boş yere bellek harcanması gibi bir dezavantajı vardır.

Aşağıdaki şekilde, veritabanında bulunan Sales tablosunun hem row format ile insert, update, delete işlemlerinin yapıldığı, hem de column format analiz işlemlerinin yapıldığı belirtilmektedir.

Şimdi In-Memory alanı veritabanının SGA memory'sinde kurulmasına geçelim. Oracle 12C kurulduktan sonra *lsnrctl start* ile listener çalıştırılır. Sonra *sqlplus / as sysdba* ile Oracle veritabanına giriş yapılır. Sqlplus'a giriş yapılınca *startup* denildiği zaman **SGA** bilgileri göstermektedir**. In-Memory** özelliği de memory'de çalıştığı için **SGA** içinde bilgisi gösteririr. Yoksa *show parameter inmem* komutu ile açık olup olmadığı, boyutunun bilgisi girilip girilmediği kontrol edilir.

```
[oracle@localhost ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Sun Mar 15 15:35:10 2015

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup
ORACLE instance started.

Total System Global Area  729808896 bytes
Fixed Size                  2928680 bytes
Variable Size             520097752 bytes
Database Buffers          201326592 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.

SQL> show parameter inmem

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------
inmemory_clause_default              string
inmemory_force                       string      DEFAULT
inmemory_max_populate_servers        integer     0
inmemory_query                       string      ENABLE
inmemory_size                        big integer 0
inmemory_trickle_repopulate_servers_ integer     1
percent
optimizer_inmemory_aware             boolean     TRUE
```

Bu örnekte inmemory_size bilgisi 0'dır. In Memory eklenirken Total System Global Area'dan yer temin etmektedir. Yani **SGA** içinde çalışmaktdır. Bu yüzden inmemory değerine maksimum **SGA** içinde uygun kaynak kadar yer verilmelidir. Bunu startup dedikten sonra Variable Size değeri ile anlayabilirsiniz. Variable Size'dan küçük değer verilmeli. Bu örnekte Variable Size 520 megabyte'tır, in-Memory için 200 megabyte tanımlanacaktır. Burada minimum In-Memory değerini 100 megabyte olarak tanımlanabilir. Inmemory_query *Enable* olmalıdır. In-Memory kullanımını kapatmak için burası *Disable* edilir. In-Memory değeri diğer *sqlplus* açılış bilgileri ile birlikte spfile içinde saklanmaktadır.

```
SQL> ALTER SYSTEM SET INMEMORY_SIZE=200M SCOPE=SPFILE;

System altered.

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area  729808896 bytes
Fixed Size                  2928680 bytes
Variable Size             461377496 bytes
Database Buffers           50331648 bytes
Redo Buffers                5455872 bytes
In-Memory Area            209715200 bytes
Database mounted.
Database opened.
SQL>
```

IN-MEMORY Area'da kolon halindeki veriler, sadece row formatı ile şekillendirilmez. Yeni bir column format ile kullanılırlar. In memory'de kayıtlar buffer cache ile yer değiştirmez. Ek bir alan gibi davranır. Bu yüzden veriler hem row hem de column format ile saklanır.

IN-MEMORY_AREA view tablosundan bakıldığı zaman inmemory havuzları görülmektedir. In-Memory'nin 1M'lık havuzlara veriler ikamet etmektedir. 64kb kısmına iste nesnelerin metadata'ları saklanmaktadır.

```
SQL> SELECT pool, alloc_bytes, used_bytes, populate_status
  2  FROM v$INMEMORY_AREA;

POOL                        ALLOC_BYTES USED_BYTES POPULATE_S
--------------------------- ----------- ---------- ----------
1MB POOL                      166723584          0 DONE
64KB POOL                      25165824          0 DONE
1MB POOL                      166723584          0 DONE
64KB POOL                      25165824          0 DONE
1MB POOL                      166723584          0 DONE
64KB POOL                      25165824          0 DONE

6 rows selected.
```

Herhangi bir tablo, tablespace ve view In-Memory alanlarda tutulabilir. In-Memory'e alınan Tablespace'in altındaki tüm tabloları da In-Memory'e alınır. Tablespace için;

```
SQL> CREATE TABLESPACE inmemoryOrnekTablespace
  2  DATAFILE '/u01/app/oracle/oradata/devdb/pdbdevdb/inmemoryornek01.dbf'
  3  SIZE 10M DEFAULT INMEMORY;

Tablespace created.

SQL> CREATE TABLESPACE inmemoryOrnek2
  2  DATAFILE '/u01/app/oracle/oradata/devdb/pdbdevdb/inmemoryornekiki01.dbf'
  3  SIZE 5M;

Tablespace created.

SQL> ALTER TABLESPACE inmemoryOrnek2 DEFAULT INMEMORY MEMCOMPRESS FOR CAPACITY
IGH;

Tablespace altered.
```

Tablolar için;

```
SQL> CREATE TABLE inMemoryOrnek(
  2  id number(2),
  3  ad varchar(20),
  4  dog_tarihi DATE,
  5  kurum varchar(20)
  6  ) INMEMORY
  7  INMEMORY MEMCOMPRESS FOR QUERY HIGH (ad,dog_tarihi)
  8  INMEMORY MEMCOMPRESS FOR CAPACITY HIGH (kurum)
  9  NO INMEMORY(id);

Table created.
```

Sonradan da tabloyu birkaç kolonları dışarı bırakarak INMEMORY alanlarında da tutulabilir.

```
SQL> ALTER TABLE inMemoryOrnek INMEMORY NO INMEMORY(dog_tarihi);

Table altered.
```

Aynı zamanda partition eklendiğinde, bazı parititon'lar INMEMORY alanlarda tutulabilir bazıları bu alandan çıkarılır. Yani tablo eklerken, bazı partition'lar dışında tablo eklenebilir.

```
SQL> ALTER TABLE inMemoryOrnek MODIFY PARTITION dog_tah_1991 NO INMEMORY;

Table altered.
```

Oluşturulan tablolara bakmak için

```
SQL> select table_name,
  2  inmemory IM,
  3  inmemory_priority IM_PR,
  4  inmemory_distribute IM_DIS,
  5  inmemory_compression IM_COMP,
  6  inmemory_duplicate IM_DUP
  7  FROM user_tables
  8  WHERE table_name='INMEMORYORNEK';

TABLE_NAME
--------------------------------------------------------------------------------
IM       IM_PR    IM_DIS          IM_COMP           IM_DUP
-------- -------- --------------- ----------------- -------------
INMEMORYORNEK
ENABLED  NONE     AUTO            FOR QUERY LOW     NO DUPLICATE
```

Oluşturulan tablespace'lere bakmak için ise;

```
SQL> SELECT tablespace_name,
  2  def_inmemory IM,
  3  def_inmemory_priority IM_PR,
  4  def_inmemory_distribute IM_DIS,
  5  def_inmemory_compression IM_COMP,
  6  def_inmemory_duplicate IM_DUP
  7  FROM dba_tablespaces
  8  ORDER BY tablespace_name;

TABLESPACE_NAME                IM       IM_PR    IM_DIS
------------------------------ -------- -------- ---------------
IM_COMP          IM_DUP
---------------- -------------
INMEMORYORNEK2                 ENABLED  NONE     AUTO
FOR CAPACITY HIGH NO DUPLICATE

INMEMORYORNEKTABLESPACE        ENABLED  NONE     AUTO
FOR QUERY LOW    NO DUPLICATE

SYSAUX                         DISABLED


TABLESPACE_NAME                IM       IM_PR    IM_DIS
------------------------------ -------- -------- ---------------
IM_COMP          IM_DUP
---------------- -------------
SYSTEM                         DISABLED

TEMP                           DISABLED
```

INMEMORY'ye eklenen kısımlar öncelik derecelerine göre INMEMORY alanlarında şekillenebilirler. Bu dereceler;

- **CRITICAL:** Nesne veritabanı açıldığı an INMEMORY alanına atılır.
- **HIGH** Kritikteki veriler atıldıktan sonra bu seviyedeki veriler INMEMORY alanının boş kalan alanlarına atılır.
- **MEDIUM** Kritik ve High seviyesindeki verileri atıldıktan sonra, bu seviyedeki veriler INMEMORY alanının boş kalan alanlarına atılır.
- **LOW** Kritik, High ve Medium seviyesindeki verileri atıldıktan sonra, bu seviyedeki veriler INMEMORY alanının boş kalan alanlarına atılır.
- **NONE** Objeler sadece ilk okundukları andaki halleri ile INMEMORY alanlarında boş yer var ise burada tutulur. Varsayılan değer budur. Tüm objeler Priority değeri belirtilmediği sürece bu değerde INMEMORY'ye atılır.

```
SQL> ALTER TABLE inMemoryOrnek INMEMORY PRIORITY CRITICAL;

Table altered.

SQL> SELECT table_name,
  2  inmemory IM,
  3  inmemory_priority IM_PR,
  4  inmemory_distribute IM_DIS,
  5  inmemory_compression IM_COMP,
  6  inmemory_duplicate IM_DUP
  7  FROM user_tables
  8  WHERE table_name='INMEMORYORNEK';

TABLE_NAME
--------------------------------------------------------------------
IM       IM_PR    IM_DIS          IM_COMP           IM_DUP
-------- -------- --------------- ----------------- -------------
INMEMORYORNEK
ENABLED  CRITICAL AUTO            FOR QUERY LOW     NO DUPLICATE
```

Her durumda INMEMORY alanları kullanılmaz. INMEMORY kullanımının kısıtlamaları;

- SYS kullanıcısının sahip olduğu tablolar, sys veya sysaux'nin sahip olduğu tablespace'ler
- Index Organized Tables ve Clustered Tables
- LOBS verilerinin dışında kalan değelerler
- Active Dataguard Standby instance'de kullanılmaz. Fakat Logical Standby Database ve Oracle Golden Gate instance'lerinde kullanılabilir.
- 64 kb'tan düşük veriler burada saklanmaz. Ayrıca In Memory alanından büyük veriler de bu alana taşınamaz.

In Memory kullanımında alandan tasarruf sağlamak için Compression kullanımı mevcuttur. Böylece büyük verileri sıkıştırarak InMemory alanında tutulabilir. Performans olarak kayıp olacaktır. Fakat yer kazancı gerekli ise bu işlem ile nesneler InMemory alanına taşınır. Compression (sıkıştırma) seviyeleri;

- **NO MEMCOMPRESS:** Sıkıştırma yapmadan verileri tutar.
- **MEMCOMPRESS FOR DML:** DML performansı için minimum sıkıştırma
- **MEMCOMPRESS FOR QUERY LOW:** Query'lerin iyi performans ile çalışması içindir. Varsayılan olarak bu seçenek ayarlanır.
- **MEMCOMPRESS FOR QUERY HIGH:** Yer kazanmak ve iyi Query performansı sağlamak için
- **MEMCOMPRESS FOR CAPACITY LOW:** Yer kazancı sağlamaya karşı performansı dengede tutmaktadır.
- **MEMCOMPRESS FOR CAPACITY HIGH:** Tamamen memory'de yer kazanmak için ayarlanır.

```
SQL> CREATE TABLE employee(
  2  id SMALLINT,
  3  ad VARCHAR2(20),
  4  soyad VARCHAR2(20),
  5  maas NUMBER(5,2)
  6  )
  7  INMEMORY MEMCOMPRESS FOR QUERY
  8  NO INMEMORY(id)
  9  INMEMORY MEMCOMPRESS FOR CAPACITY HIGH (ad);

Table created.

SQL> SELECT table_name,
  2  column_name,
  3  inmemory_compression IM_COMP
  4   FROM v$im_column_level
  5  WHERE table_name='EMPLOYEE';

TABLE_NAME           COLUMN_NAME        IM_COMP
-------------------- -------------- ----------------------
EMPLOYEE             ID             NO INMEMORY
EMPLOYEE             AD             FOR CAPACITY HIGH
EMPLOYEE             SOYAD          DEFAULT
EMPLOYEE             MAAS           DEFAULT
```

Tabloların kolonlarının In-Memory'de olup olmadığını ve Compress tipini görmek için V$IM_COLUMN_LEVEL view tablosundan öğrenilebilir. Bunun dışında işe yarayabilecek view tabloları; V$IM_SEGMENTS ve V$IM_USER_SEGMENTS view tablolarıdır.
Oracle veritabanında; Oracle Compression Advisor (DBMS_COMPRESSION) özelliği ile in-memory sıkıştırma için sıkıştırma oranı belirleyerek sıkıştırma tavsiyesi vermektedir.

Sonuç olarak; In-Memory özelliği, eşsiz "dual-format" yaklaşımı ile nesneler otomatik olarak verileri hem OLTP operasyonları için Oracle row formatta düzenleme yapabilir hem de yeni column format tekniği ile analiz süreçleri yapılabilir. Her iki format aynı anda aktif ve tutarlı çalışabilir. Analytical indexleri kullanmak yerine In-Memory kullanarak OLTP işlemleri hızlanır. Indexlerin düzenlenmesi, bakımı pahalı ve zaman alan bir iştir. Ayrıca transaction processing sürecini yavaşlatmaktadır. Bu yüzden In-Memory ile index kavramından ayrılıp bakım, düzenleme ve onarım işleri olmadan hem row-format ile OLTP işlemleri yapılabilmekte hem de column-format ile analiz işlemleri yapılabilmektedir. Bu yüzden Oracle **analyze index** kavramından kurtulup **In-Memory** kavramını getirmek istemiştir. Öyle görülüyor ki sağladığı avantajlar ile veritabanı efektif olarak kullanılmaktadır.