ORACLE
ONLINE REDO LOG DOSYALARI

Contents

1. ACID Kavramı ve Transactional Log Gerekliliği

Tam güvenlik vaad eden her veritabanı **ACID** (**A**tomicity **C**onsistency **I**solation **D**urability) denilen standarta uymak zorundadır. Bu standartı kısaca açıklarsak;

- **Atomicity :** Ya hep ya hiç kuralıdır. Bir işlem ya tam olarak yapılıp, sonuçlanır; ya da hiç yapılmaz. Yarıda kalan işlemler geri alınır. Yapılan işlemler ise belirsizliğe yer bırakmayacak şekilde tamamlanmıştır.

- **Consistency :** Veritabanında tutarlılık esastır. Bir işlemden önce veritabanı tutarlıysa, işlem sonrası da tutarlı kalır. Eğer veritabanı tutarlılığını bozan bir işlem gerçekleşiyorsa, işlem geriye alınır.

- **Isolation :** Bir operasyon tarafından yapılan değişikliklerin, diğer işlemler tarafından nasıl görüntülenmesi gerektiğini belirler. Örnek olarak iki kullanıcı aynı katalog nesnesini güncelliyorsa, ortamda sadece kendileri varmış gibi (izole) çalışabilmeleri gerekir. Bir kullanıcı diğerinin değişikliklerini etkilememelidir, başkasının değişikliklerinden de etkilenmemelidir.

- **Durability :** Commit edilen işlemlerin veritabanına yazıldığının ve kaybolmayacağının garantisidir.

Yukarıdaki özelliklerin sağlanması oldukça güçtür. Bunları yapabilmek için **Write Ahead Logging** (**Transactional Log**) ve **Shadow Paging** isimli yöntemler kullanılmaktadır. Shadow Paging, veritabanına gerçek girişin yapılmasından önce, kopyaya girişin yapılıp, ardından gerçek değişikliğe gidilmesi olarak düşünülebilir. Write Ahead Logging (WAL) ise yapılacak işlemlerin öncelikli olarak bir log dosyasına yazılması ardından, veritabanına ait dosyalara aktarılmasıdır.

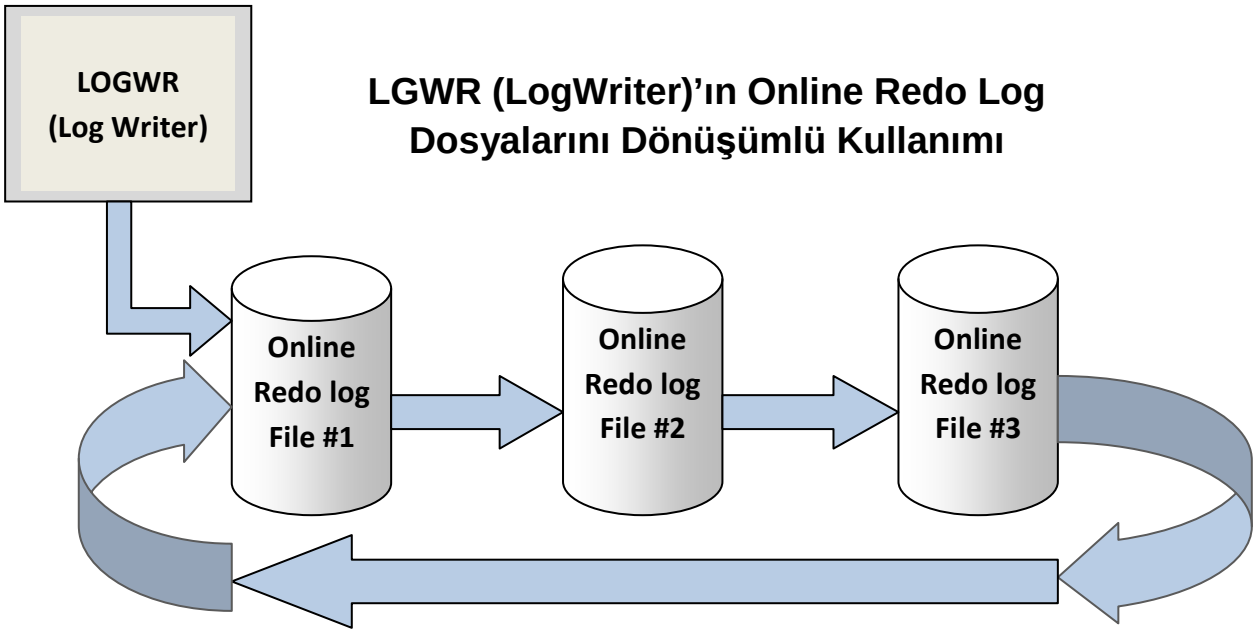
WAL kavramı, Oracle'da online redo log olarak geçmektedir. Veri güvenliği için yapılan her işlem öncelikle online redo log diye isimlendirilen dosyaların içerisine yazılır. Daha sonra ara ara bu redo log'lar Oracle'a ait veritabanı dosyalarına yazılır. (Ekstra bir bilgi; Yahoo! gibi birçok büyük firmanın veritabanı olarak kullandığı PostgreSQL, redo log yerine WAL ismini aynen kullanmaktadır.)

Belirtmeye çok gerek yok ancak yine de değineyim, transactional log binary bir dosyadır, sadece Oracle tarafından okunabilir. Programların işleyişini gösteren, insanların okuyabileceği metin formatındaki log dosyalarıyla bir ilgisi yoktur.

2. Oracle, Online Redo Log ve Veritabanı Dosyalarına Nasıl Yazar?

Bir veritabanı iki veya daha fazla redo log dosyasından oluşur. En az iki olmasının nedeni, archivelog şeklinde çalışıyorsanız, redo log dosyalarından bir tanesi kullanılırken diğeri arşive çıkılabilsin diyedir. Böylece her zaman çalışabilir olmayı sağlar. (Gerçi aşırı yoğun bir sistemde sadece iki adet redo log kullanıyorsanız, `"Thread 1 cannot allocate new log, sequence #"` şeklinde sık sık hata alırsınız. Çünkü redo log'ların arşive çıkması için yeterli zaman olmayacaktır.)

Online redo log'lar arasındaki geçiş, sirküler şekilde aşağıdaki gibi gerçekleşmektedir.



Yukarıda verilen işlemler LogWriter tarafından sağlanmaktadır. LGWR, log buffer'daki redo bilgilerini, redo log dosyalarına aktarmaktan sorumludur. Ayrıca checkpoint işlemleri de bu process, redo log dosyalarının statüsünü güncellemekten sorumludur.

Herhangi bir zaman diliminde, Oracle redo log dosyalarından sadece bir tanesini kullanır. Kullandığı redo log dosyası dolduğunda, bir sonraki redo log dosyasına geçilir. Fakat bu geçiş yapılırken, sıradaki online redo log içeriğinin datafile'lara yazılıp yazılmadığı ve bu dosyadan arşiv üretilip üretilmediği kontrol edilir. Her iki işlemde bitene kadar bu redo log dosyası kullanılmayacaktır.

LGWR, bir sonraki redo log dosyasına geçtiğinde, bir önceki redo log dosyasının durumunu controlfile içinde CURRENT'ten ACTIVE'e çevirir. (Bir sonraki bölümde, bu statüler açıklanmıştır.) Akabinde DBWR (Database Writer) işlemini durumdan haberdar ederek, bir önceki redo log dosyasında checkpoint işlemi yapmasını gerektiğini belirtir. DBWR tarafından checkpoint işlemi tamamlanıp, redo log'daki değişiklikler veritabanı dosyalarına yazıldığında CKPT process'i çağrılır. CKPT işlemi veritabanı dosyalarının header bilgilerini ve check point bilgisini (sadece) controlfile içerisinde günceller. (CKPT tarafından yapılan güncelleme bilgisi v$datafile_header tablosundaki checkpoint_change# ve checkpoint_time bilgileriyle ilgilidir.) Burada dikkat edilmesi gereken, CKPT'in redo log bilgisine dair bir güncelleme yapmamasıdır. İsminden de belli olacağı gibi sadece checkpoint işlemiyle ilgilenir.

CKPT işlemi tamamlandığında, LGWR işlemi çağrılarak controlfile içerisinde redo log bilgisini ACTIVE'den INACTIVE'e çeker. (Bu güncelleme, v$log.status bilgisini sağlar.) Bu değişiklik düşük önceliklidir, çünkü bu konuyla tek ilgilenen process LGWR'in kendisidir. Buradaki status bilgisine göre, LGWR redo log dosyasının tekrar kullanılıp kullanılmayacağına karar verir. (Eğer redo log dosyası ACTIVE durumdaysa, checkpoint işleminin sonuçlanması bekleniyor demektir.) Eğer yeterli sayıda redo log dosyası varsa, herhangi bir iş bloklanmaz. Ancak yoğun bir ortamda az sayıda redo log dosyası varsa, daha önce belirtildiği gibi `"Thread 1 cannot allocate new log, sequence #"` şeklinde sık sık hata alınacaktır.

3. Check Point bilgisini güncel tutmak

LGWR'in düşük öncelikle redo log dosyaklarının statüsünü güncellemesinden hoşnut değilseniz, aşağıdaki yöntemleri izleyebilirsiniz:

i. `SQL> ALTER SYSTEM CHECKPOINT;` komutu vererek manuel olarak tetikleyebilirsiniz.

ii. ***LOG_CHECKPOINT_TIMEOUT*** isimli parametreyi değiştirmeniz mümkündür. Bu parametreyi değiştirmek için aşağıdaki adımları izleyebilirsiniz.

a. Önce mevcut duruma bakıyoruz:

```
SQL> SHOW PARAMETER LOG_CHECKPOINT_TIMEOUT;
log_checkpoint_timeout               integer  27000
```

Buna göre her 27000 saniyede bir checkpoint işlemi gerçekleşecektir.

b. Şimdi de her 30 dakikada bir check point işleminin gerçekleşeceğini garanti ediyoruz:

```
SQL> ALTER SYSTEM SET LOG_CHECKPOINT_TIMEOUT=1800 SCOPE=BOTH;
SYSTEM ALTERED
```

PFILE üzerinden değişiklik yaparak aynı sonuca ulaşabilirsiniz. Ancak yukarıdaki yöntemi kullanırsanız, veritabanının kapanmasına gerek yoktur; dinamik olarak kullanımıyla başlanır. Bu yöntemle check point işleminin gerçekleştiği gerçekleştirecektir.

4. Check Point bilgisini Alert Log'a yazdırmak

Eğer yapılan checkpoint işleminin alert dosyasına yazılmasını isterseniz, aşağıdaki adımları takip edebilirsiniz:

a. Mevcut durumu kontrol;

```
SQL> SHOW PARAMETER LOG_CHECKPOINTS_TO_ALERT;
log_checkpoints_to_alert             boolean  FALSE
```

b. Değişiklik yapma;

```
SQL> ALTER SYSTEM SET LOG_CHECKPOINTS_TO_ALERT=TRUE SCOPE=BOTH;
System switch log altered.
```

Yukarıdaki örneği daha iyi görebilmek açısından ben bir dakikalık periyotlarla checkpoint yapacak şekilde ayarlama yaptım. Alert log dosyasına bakıldığında sonuç şu şekilde oldu:

```
oracle@test1:/data1/oracle/admin/yatirim/bdump > tail -f alert*
...
ALTER SYSTEM SET log_checkpoint_timeout=60 SCOPE=BOTH;
Wed Sep 17 15:37:48 2008
Beginning global checkpoint up to RBA [0x4.1595.10], SCN: 32216940288
Completed checkpoint up to RBA [0x4.1595.10], SCN: 32216940288
Wed Sep 17 15:38:26 2008
Incremental checkpoint up to RBA [0x4.1596.0], current log tail at RBA [0x4.1596.0]
Wed Sep 17 15:39:27 2008
Incremental checkpoint up to RBA [0x4.1596.0], current log tail at RBA [0x4.15a1.0]
Wed Sep 17 15:40:42 2008
Incremental checkpoint up to RBA [0x4.15ad.0], current log tail at RBA [0x4.15ad.0]
...
```

Belirtildiği gibi dakikada bir checkpoint işlemi gerçekleşti. Gerçi alert log dosyasında check point bilgisi görmek ne kadar gereklidir tartışılabilir.

5. Redo Log Dosyalarının Durumları

LGWR'in yazma işlemini yaptığı redo log dosyası **'current'** olarak geçer. Redo log dosyalarının bulunabileceği bazı durumlar şu şekildedir:

a. CURRENT : Redo log dosyasının kullanımda olduğunu gösterir.

b. ACTIVE : Current redo log dosyası değişmiştir. Ancak daha önce kullanımda olan redo log dosyasının içeriği henüz veritabanına

aktarılmamıştır. Yazma işlemi devam etmektedir. LGWR tarafından active'lik durumu bir süre sonra, INACTIVE'e çekilir. Checkpoint komutuyla, yazma işleminin yapılmasını tetiklemek mümkündür.

c. **INACTIVE** : Kullanımda olmayan redo log dosyalarını ifade eder.

d. **UNUSED** : İlgili redo log dosyasının henüz hiç kullanmadığını gösterir. Yeni eklenen redo log dosyalarını, unused olarak görürsünüz.

e. **INVALID** : Dosyanın erişilemez (ya da bozuk) olduğunu işaret eder.

f. **STALE** : Dosya tamamlanmamıştır. ABORT ile ya da beklenmeyen ani veritabanı kapanmalarından kaynaklanmaktadır.

6. Redo Log Grupları, Elemanları ve Elemanların Çoklanması

Şu ana kadar redo log dosyalarının her birini ayrı olarak ele aldık. Hâlbuki aynı redo log dosyasını birden fazla lokasyonda tutabilirsiniz. Aynı elemanların oluşturduğu kümeye, redo log grubu denmektedir. Daha yalın bir dille şöyle özetleyebiliriz: Bir problem çıkma ihtimâline karşı redo log dosyalarını çoklamak (multiplexing) mümkündür. Birinci redo log dosyası demek yerine, birinci redo log grubu denerek, bu gruba birden fazla redo log dosyası eleman/üye (member) olarak atanır. Grup içindeki redo log dosyalarının hepsi aynıdır. Örnek verirsek;

```
oracle@test1:/data1/oracle/oradata/TEST > ls -lh red*
...
-rw-r----- 1 oracle oinstall 51M Sep 17 13:35 redo41.log
-rw-r----- 1 oracle oinstall 51M Sep 17 13:35 redo42.log
-rw-r----- 1 oracle oinstall 51M Sep 17 14:05 redo51.log
-rw-r----- 1 oracle oinstall 51M Sep 17 14:05 redo52.log
-rw-r----- 1 oracle oinstall 51M Sep 17 14:32 redo61.log
-rw-r----- 1 oracle oinstall 51M Sep 17 14:32 redo62.log
```

Şimdi bu dosyaların md5 değerlerine bakalım;

```
oracle@test1:/data1/oracle/oradata/TEST > md5sum redo*
...
c949c72573d884d04f52ec275bbaf9d0  redo41.log
c949c72573d884d04f52ec275bbaf9d0  redo42.log
d5f688d77b712103ebfe31744833bedf  redo51.log
d5f688d77b712103ebfe31744833bedf  redo52.log
3d91c30ba46d0c1d7daaa6ccc0362399  redo61.log
3fbbf539a69204b0d0ad9950e0caf901  redo62.log
```

Bu örneğimizde her bir redo log dosyası iki elemandan oluşan bir grubun elemanlarıydı. Farkedeceğiniz gibi son satır hariç, redo log dosyalarının tamamı grup içinde aynı. Son satırdaki farklılıkta, redo6 ile başlayan grubun aktif olarak kullanılmasından kaynaklanıyor. Sürekli olarak değişiklikliğin yaşanması, komutu sürdürürken, redo log dosyalarının

değişmesiyle farklı md5 değerlerinin ortaya çıkmasına neden olmakta. (Aynı redo log dosyasının birden fazla kopya ile çalışmasını gördük. Benzer bir durum arşiv dosyalarının aynı anda farklı lokasyonlara yazılması için de vardır.)

Redo Log dosyalarını hatalara karşı çoklarsak, güvenliğimiz artacaktır. Aşağıdaki senaryolar ve sonuçları çoklamanın faydalarını göstermek açısından yararlıdır:

Şayet...	**Sonucunda...**
LGWR, grup elemanlarından en az birine yazabilirse,	Yazma işlemi normal gerçekleşir. LGWR, yazılabilen redo log üyelerine yazar, yazılamayanları önemsemez.
LGWR, bir sonraki grup elemanları arşivlenmediği için erişemiyorsa,	Grup elemanları arşive çıkılana ve kullanım için uygun hâle gelene kadar, işlem durdurulur.
Sıradaki grubun bütün elemanlarına donanım kaynaklı bir problemden erişilemiyorsa,	Oracle hata döndürür ve veritabanı kapatılır. Bu durumda, bir online redo log'un olmayışı nedeniyle recover işlemi yapmanız gerekebilir.
LGWR'in yazma yaptığı bütün grup elemanları birden erişilmez hâle gelirse,	Oracle hata döndürür ve veritabanı kapatılır. Eğer log'un bulunduğu ortamda bir problem yoksa –örneğin elektriğin birden kesilmesi gibi bir durum yaşandıysa– recover işlemine gerek kalmayabilir. Aksi hâlde recover işlemi gerekecektir.

Multiplexing yapılan elemanların farklı disklere yazılması veri kaybını azalmakta size yardımcı olacaktır.

7. Redo Log'lara Grup/Eleman Ekleme/Çıkartma

Grup/eleman ekleme, çıkartma gibi konulara devam etmeden önce, veritabanıyla ilgili bazı kontrol yapmamız gerekiyor. Çünkü redo log'larla ilgili her türlü işlem öncesi MAXLOGFILES ve MAXLOGMEMBERS değerlerinin kontrolü mutlaka yapılmalıdır.

Bu değerlere iki şekilde ulaşabilirsiniz. Birinci yöntem, controlfile'i trace'e çıkarak sağlanır.

```
SQL> alter database backup controlfile to trace as '/tmp/create_cfile.sql';
```

```
Database altered.

SQL> ! more /tmp/create_cfile.sql
...
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "YATIRIM" NORESETLOGS  NOARCHIVELOG
    MAXLOGFILES 50
    MAXLOGMEMBERS 5
    MAXDATAFILES 100
    MAXINSTANCES 1
    MAXLOGHISTORY 292
...
```

Yukarıdaki çıktıya göre, **MAXLOGFILES** sayesinde maksimum 50 adet redo log grubu olabileceğini anlarız. **MAXLOGMEMBERS** ise her grupta 5 adet üye redo log bulunabileceğini söyler.

Controlfile ile uğraşmak yerine, sorgu çekmek suretiyle de aynı parametreleri yakalayabilirsiniz. Fakat verilecek sorgu bir X$ view'lerini kullandığından SYSDBA yetkisine sahip olmanız ve sisteme bu şekilde girmeniz gerekmektedir.

```
SQL>  SELECT
      DECODE (INDX,
      3,'MAXLOGFILES',4,'MAXDATAFILES',2,'MAXINSTANCES',9,'MAXLOGHISTORY ')
      VARIABLE,
      RSNUM FROM X$KCCRS
      WHERE INDX IN (3,4,2,9)
      UNION ALL
      SELECT 'MAXLOGMEMBERS ',DIMLM FROM X$KCCDI;


      VARIABLE            RSNUM
      -------------- ----------
      MAXINSTANCES             1
      MAXLOGFILES             50
      MAXDATAFILES           100
      MAXLOGHISTORY          292
      MAXLOGMEMBERS            5
```

Eğer SYSDBA yetkiniz yoksa ve controlfile'a erişemiyorsanız, bir başka sorguyla MAXLOGFILES değerini okumanız mümkündür. Ancak tek başına bu bilgi yeterli olmayacaktır.

```
      SQL>  SELECT TYPE,RECORDS_TOTAL
            FROM V$CONTROLFILE_RECORD_SECTION
            WHERE TYPE='REDO LOG'
            ORDER BY 1;

      TYPE                         RECORDS_TOTAL
      ---------------------------- -------------
      REDO LOG                                50
```

Gerekli bütün bilgilerden emin olduktan sonra, aşağıdaki SQL ifadelerini kullanabilirsiniz. Dikkat edilmesi gereken en önemli nokta, işlem yaptığınız grubun aktif olmamasıdır. Aşağıdaki SQL ifadesiyle grubun statüsünü okuyabilirsiniz;

```
/* REDOLOG DOSYALARININ DURUMUNU GORMEK ICINDIR */
SELECT  V1.MEMBER, V2.*
FROM    V$LOGFILE V1, V$LOG V2
WHERE   V1.GROUP# = V2.GROUP#
ORDER BY 3;
```

Eğer üzerinde işlem yapmak istediğiniz grup aktif ise, aşağıdaki komutlarla switch etmeniz gerekecektir. Switch sonrasında yukarıdaki sorgunun mutlaka tekrar çalıştırılıp üzerinde işlem yapılacak grubun inaktif olduğundan emin olunması lâzımdır.

```
/* LOGFILE GRUBUNUN DEGISIP ARSIVE CIKMASI ICIN */
ALTER SYSTEM SWITCH LOGFILE;

/* CHECKPOINT ISLEMINI MANUEL TETIKLEMEK ICIN */
ALTER SYSTEM CHECKPOINT;
```

Her elemanı 100 MB boyutunda olan 8 numaralı yeni bir grup yaratmak istersek, aşağıdaki ifadeyi kullanabiliriz. (Buradaki 100MB'i keyfi atamamak önemlidir. Diğer redo log gruplarlarıyla aynı değerde olmasına dikkat etmek gerekir.)

```
/* REDO LOG DOSYALARINA YENI BIR GRUP EKLEMEK ICIN */
ALTER DATABASE ADD LOGFILE GROUP 8
('/data2/ccebi_test/oradata/redo82.log') SIZE 100M;


/* AYNI ANDA IKI ELEMANLI YENI BIR GRUP EKLEMEK */
ALTER DATABASE ADD LOGFILE GROUP 9
      ('/data2/ccebi_test/oradata/redo91.log',
      '/data2/ccebi_test/oradata/redo92.log' ) SIZE 50M;


/* MEVCUT BIR GRUBA YENI REDO LOG UYESI EKLEMEK ICIN */
ALTER DATABASE ADD LOGFILE MEMBER
      '/data2/ccebi_test/oradata/redo83.log' TO GROUP 8;
```

Redo log gruplarını mümkün olduğunca birbiriyle aynı şekilde tutmak önemlidir. Yani bir grubu 2 elemanlı yaratırken, diğer grubun 4 elemanlı olması tasarım açısından güzel gözükmeyecektir. Böyle durumlarda, redo log grubundan eleman drop etmek için aşağıdaki ifadeleri kullanabilirsiniz:

```
/* BIR REDOLOG FILE GRUBUNDA SADECE BELIRLI DOSYALARI DROP ETMEK ICINDIR */
ALTER DATABASE DROP LOGFILE MEMBER
      '/data2/ccebi_test/oradata/redo93.log',
      '/data2/ccebi_test/oradata/redo94.log';
```

Bir redo log grubunun için bütün elemanları drop etmek mümün değildir. Bunu denediğinizde, en az bir eleman olması gerektiğini söyleyen bir hata mesajı sizi durduracaktır. Fakat bir redo log grubundan tamamen vazgeçmek ve drop etmek istiyorsanız aşağıdaki ifade ile bunu gerçekleştirmek mümkündür:

```
/* BIR REDOLOG FILE GRUBUNU DROP ETMEK */
ALTER DATABASE DROP LOGFILE GROUP 4;
```

DROP edilen redo log nesneleri aslında silinmezler. Ancak bir daha kullanılamazlar. Boşuna yer kaplamasının anlamı olmadığından, işletim sistemi üzerinden silinmesi gerekir.

Aktif olmayan gruplar üzerinde çalıştığınız sürece, bu işlemleri online gerçekleştirebilirsiniz. (Elbette yedek alarak çalışmak her zaman ideal yöntemdir.)

8. Redo Log Dosyalarının İsimlerini Değiştirmek

Yedinci bölümde anlatılan prosedürleri izleyerek redo log dosyalarını online olarak yeniden isimlendirebilirsiniz. Fakat bunun ideal bir yöntem olduğunu söylemek oldukça güç, çünkü işlemi gerçekleştirmek hâyli zahmetli olur.

Basit bir yöntem ise veritabanını kapatıp, redo log dosyalarını işletim sistemi üzerinden yeniden isimlendirmek ve ardından veritabanını startup mount mode ile başlatıp redo log'ların isimlerinin değiştiğini Oracle'a bildirmektir. ***Elbette veritabanını hazır kapatmışken yedek alarak işleme başlamanız, daha güvenli çalışmanızı sağlayacaktır.***

Veritabanını kapatıp, sonrasında startup mount mode ile başlattıktan sonra, aşağıdaki komutu vererek, redo log dosyalarının yeni isimlerini Oracle'a tanıtabilirsiniz:

```
SQL>  ALTER DATABASE RENAME FILE '/datac5/oradata/redo1_1.dbf' TO
                                 '/datac5/oradata/redo11.dbf';
```