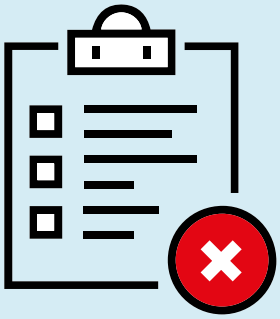

Nologging Objects von 11g bis 18c

Timo Giese, Fiducia & GAD IT AG

Nologging Operations und Objects begegnen früher oder später jedem Datenbank-Administrator. Dies äußert sich dann, wenn die Oracle-Fehlermeldung „ORA-26040: Data block was loaded using the NOLOGGING option“ im Alert-Log und/oder in den Applikationslogs auftaucht; meist in Kombination mit einem Anruf der Applikations-Administratoren dahingehend, dass die Anwendung ein Problem hat. Dieser Artikel zeigt, was es mit Nologging Objects und -Operationen auf sich hat, und stellt Methoden zur Anzeige, Diagnose, Reparatur und Verhinderung vor. Dabei wird auch auf Neuerungen und Verbesserungen im Umgang mit Nologging Objects bis hin zu 18c eingegangen.

Nologging-Operationen sind wenig protokollierte Transaktionen in der Datenbank. Dabei werden nur minimale Informationen über die vonstattengehende Transaktion in den Redologs protokolliert. Dadurch ist es möglich, die Daten schneller in die Tablespaces beziehungsweise Datenfiles einzufügen. Dieses Verfahren hat auch seine Kehrseite und kann unter Umständen zu Problemen beim Recovery einer Datenbank führen. Dies ist dadurch bedingt, dass keine vollständigen Redolog-Informationen vorhanden sind.

Eine weitere Problematik wird erst sichtbar, wenn die Applikation nach dem Recovery wieder online ist: Das Recovery wurde erfolgreich (ohne Fehler im RMAN-Log) durchgeführt. Die Datenfiles, die mit Nologging-Operation geladen wurden, sind im Zustand „unrecoverable“ und aus Sicht der Datenbank und Applikation korrupt. Ist dies der Fall, merkt das die Applikation sehr schnell: Ein einfacher „select“ auf die Daten produziert die Fehlermeldung „ORA-26040: Data block was loaded using the NOLOGGING option“.

Einsatzbereiche

Trotz der im vorigen Abschnitt geschilderten Problematik gibt es gute Gründe, aus Sicht der Applikation auf diese Methode zu setzen. Nologging-Operationen werden überwiegend bei großen zu ladenden Datenmengen eingesetzt. Dies wiederum ist bedingt durch eine Zeitrestriktion, wann die Daten geladen beziehungsweise wann die Daten verarbeitet sein müssen und ein bestimmtes Ergebnis den Anwendern der Applikation zur Verfügung stehen muss. Überwiegend findet diese Methode Einsatz in Data-Warehouse- und Entscheidungs-Systemen (DSS). Darüber hinaus sie auch in Mixed-Workload-Systemen anzutreffen, die anteilig aus OLTP und Batch bestehen. Zu den Nologging Objects gehören:

- Tabellen
- Partitionen
- Indizes
- LOBs

Nicht jede Datenbank-Aktion führt zu Nologging-Objekten. Grundbedingung ist zuerst, dass das gewünschte Objekt als „Nologging Object“ angelegt wurde (etwa mit „alter table test nologging“). Alternativ ist die auszuführende Operation explizit mit dem Zusatz „nologging“ zu versehen. Folgende Operationen können „nologging“ ausgeführt werden:

- insert /*+ append */ into
- insert into <table> nologging
- create table <table_copy> as select
- alter table <table> move|split partition
- alter index <index> rebuild
- alter index <index> split|rebuild partition
- SQL-Loader direct load (sqlldr direct=true und “unrecoverable option”)

Nologging-Objekte erstellen

Operationen werden „nologging“ ausgeführt, wenn die zugrunde liegenden Objekte als „nologging“ definiert sind (wie Tabelle). Eine Variante ist die Festlegung auf Tablespace-Ebene. Dies kann entweder beim Erstellen des Tablespace oder nachträglich festgelegt werden. Wichtig ist dabei, dass alle Objekte innerhalb des Tablespace das „nologging“-Flag bekommen, sofern nicht anderweitig beim Erstellen der Objekte explizit angegeben *(siehe Listing 1)*.

Die kleinste Stufe sind Objektebene-Tabelle, LOB-Segment, Index etc. Hier gilt, wie beim Tablespace, dass eine Änderung auf „nologging“ auch nach der Erstellung möglich ist *(siehe Listing 2)*. Für Indizes gilt dies nur bei Neuanlage, Split und Rebuild.

Nologging-Operationen verhindern

Nologging-Operationen können in bestimmten Umfeldern nicht gewünscht sein, etwa in einem Data-Guard-Umfeld, bei dem sonst zusätzlich auf der Standby-Seite Inkonsistenzen auftreten. Im weiteren Verlauf dieses Artikels wird auf Neuerungen in der Oracle-Datenbank eingegangen, die eine Verbesserung dieser Problematik aufzeigen. Ein weiterer Grund kann eine Unternehmensrichtlinie sein, die Datenkonsistenz und Wiederherstellbarkeit über die Laufzeit zur schnellen Bereitstellung von Daten vorgibt.

In der Oracle-Datenbank gibt es Möglichkeiten, Nologging-Operationen zu unterbinden, sodass diese auch nicht unabsichtlich durch Entwickler oder Applikations-Administratoren verursacht werden können. Dies lässt sich global aktivieren; für die gesamte Datenbank werden mit „alter database force logging;“ alle Nologging-Operationen in Logging-Operationen mit vollständigem Redo-Record umgeschrieben. Zu beachten ist, dass Objekt-Definitionen mit dem Zusatz „nologging“ weiterhin angelegt werden können, dies jedoch beim Einfügen von Daten ignoriert wird.

Sollen Nologging-Operationen nur für bestimmte Bereiche in der Datenbank erlaubt sein (etwa Staging Area), kann dies auch auf Tablespace-Ebene festgelegt sein. Dazu werden alle Tablespaces, für die keine Nologging-Operationen erlaubt sind, mit „alter tablespace <tbs> force logging;“ in den „force logging“-Modus gesetzt.

Ist die Datenbank global auf „force logging“ gesetzt und man möchte dies später wieder rückgängig machen, ist dies ohne

```
Tablespace nologging erstellen:
   create tablespace staging_tbs nologging;

Tablespace anpassen:
   alter tablespace staging_tbs nologging;
```

Listing 1

```
Tabelle erstellen:
   create table tbl_nologging (id number) nologging;

Tabelle anpassen:
   alter table staging_table nologging;
```

Listing 2

```
select force_logging from v$database;
   FORCE_LOGGING
   -------------
   YES

select tablespace_name,force_logging from dba_tablespaces;
   TABLESPACE_NAME       FORCE_LOGGING
   --------------------  -------------
   TBS1                  NO
   TBS2                  YES
   TBS3                  NO
```

Listing 3

```
RMAN> validate check logical datafile 10;
…
List of Datafiles
==================
File  Status  Marked Corrupt  Empty Blocks  Blocks Examined  High SCN
----  ------  --------------  ------------  ---------------  ---------
10    OK      1308            9989          12800            18647556
   File Name: /u01/oradata/test.dbf
   Block Type Blocks Failing Blocks Processed
   ---------- -------------- ----------------
   Data       0              2621
   Index      0              0
   Other      0              190
```

Listing 4

```
select file#,block#,blocks,NONLOGGED_START_CHANGE#,NONLOGGED_END_CHANGE#
from v$nonlogged_block;

   FILE#   BLOCK#   BLOCKS  NONLOGGED_START_CHANGE#  NONLOGGED_END_CHANGE#
   -----   -------  ------  ------------------------  ---------------------
   10      267      13      18647507                  18647507
```

Listing 5

```
RMAN> report unrecoverable;
   File Type of Backup Required Name
   ---- ----------------------- -----------------------
   10   full or incremental     /u01/oradata/test.dbf
```

Listing 6

Weiteres mit dem SQL-Befehl „alter database no force logging“ möglich. Selbiges gilt für den Tablespace mit „alter tablespace <tbs> no force logging“.

Der Status, ob „force logging“ global oder auf Tablespace-Ebene aktiv ist, kann durch Abfrage der Views „v$database“ beziehungsweise „dba_tablespaces“ angezeigt werden *(siehe Listing 3)*. Für Tabellen und Indizes ist dies in der Spalte „LOGGING“ in den Views „dba_tables“ und „dba_indexes“ ersichtlich.

Ist es unverzichtbar, Nologging-Operationen durchzuführen, muss aus Gründen der Wiederherstellbarkeit gewährleistet sein, dass nach der Operation ein Backup der betroffenen Objekte durchgeführt wird. Dies ist die einzige Möglichkeit, eine erfolgreiche Wiederherstellung mit allen Daten zu garantieren.

Nologging-Operationen feststellen

Wenn Objekte „nologging“ definiert sind, heißt das bei Weitem nicht, dass alle Operationen „nologging“ erfolgen. Folglich muss zuerst festgestellt werden, welche Bereiche der Datenbank von Nologging-Operationen betroffen sind. Oracle bietet in den unterschiedlichen Datenbank-Versionen verschiedene Möglichkeiten an, dies festzustellen.

Das Utility DB-Verify „dbv” kann mit „dbv file='<pfad_zu_datenfile>/data01.dbf' userid=sys/pw“ Nologging-Operationen auf Datenfile-Ebene in Versionen vor 10g bis hin zu 18c (nur NON-CDB) feststellen. Jeder gefundene Nologging Change wird mit den Meldungen „DBV-00200“ und „DBV-00201“ quittiert.

Ab 10.2.0.5 werden Nologging-Objekte mit RMAN und dem Befehl „validate check logical database“ festgestellt. Dabei werden alle Tablespaces und Datenfiles überprüft. Es ist ebenfalls möglich, dies auf einzelne Objekte zu beschränken, etwa auf ein Datenfile durch „validate check logical datafile 3“ *(siehe Listing 4)* oder aber auf Tablespaces. Dabei sind Informationen über korrupte Blöcke in 10g und 11c in der View „v$database_block_corruption“ in der Spalte „CORRUPTION_TYPE“ mit dem Flag „NOLOGGING“ versehen.

Ab der Version 12c gibt es aus Kompatibilitätsgründen diese View weiterhin, jedoch wird sie nicht mehr mit Informationen bezüglich Nologging-Operationen befüllt. Alle Informationen gehen in die View „v$nonlogged_block“ *(siehe Listing 5)*. Zusätzlich wird im Datenbank-Alert-Log ein Eintrag für jeden gefundenen Nonlogged-Block geschrieben (etwa „Finished Nonlogged Block Replacement recovery(validate) on file 5. 400 blocks found“).

Ab 11g steht ein weiterer RMAN-Befehl „report unrecoverable“ zur Verfügung, der korrupte Datenfiles anzeigt. Dieser bietet in seiner Auflistung den Hinweis zur Verhinderung einer späteren Recovery-Problematik: Ein volles oder inkrementelles Backup hilft, den ORA-26040-Fehler zu vermeiden *(siehe Listing 6)*. Datenbanken in Version 12.2 und neuer bieten mittels RMAN weitere Befehle zur Detektion: „validate database nonlogged block“ und „validate datafile nonlogged block“.

Reparatur

Eine Reparatur der korrupten Daten ist generell nicht möglich, lediglich die Korruption in den Datenblöcken lässt sich beseitigen. Ist die Korruption in einem „nologging“-Index aufgetreten, kann dieser relativ einfach mit dem SQL-Befehl „alter index <idx> rebuild [online]“ neu erstellt werden.

Ob ein Index aufgrund einer Nologging-Operation korrupt ist, kann nicht aus den Views „dba_indexes“ beziehungsweise „dba_ind_partitions“ ermittelt werden, da der Status des jeweiligen Index weiterhin als „VALID“ angezeigt wird. Hier hilft die Abfrage der View „v$nonlogged_block“ (12c) beziehungsweise „v$data_block_corruption“ (11g) mit der zusätzlichen Ausgabe der Spalte „object_id“. Mit dieser Information lässt sich der betroffene Index mit der View „dba_objects“ ermitteln.

Falls Tabellen „nologging“ geladen und beim Recovery die zugrunde liegenden Datenfiles korrupt wurden, sind die geladenen Daten unweigerlich nicht wiederherstellbar. Ein erneutes Laden der Daten durch Applikationsprozesse ist die Folge. Zuvor müssen die korrupten Blöcke in der Datenbank repariert werden. Das Vorgehen gliedert sich in zwei Schritte: Zuerst wird mit dem Package „DBMS_REPAIR“ die Datenbank angewiesen, alle korrupten Blöcke des Tabellenobjekts zu ignorieren.

Im zweiten Schritt wird die Tabelle mit allen intakten Blöcken verschoben *(siehe Listing 7)*. Zu beachten ist, dass die ursprünglichen Blöcke weiterhin korrupt, jedoch nicht mehr ein Teil des Tabellen-Objekts sind und ab jetzt zu den freien Blöcken gehören. Diese Blöcke wiederum werden bei weiteren Inserts herangezogen, jedoch vor der Verwendung neu initialisiert; dadurch wird die Korruption vollständig beseitigt. Weitere Details zur Reparatur stehen in der MOS-Note 794505.1.

Sind LOB-Objekte im Spiel, ist die Prozedur zur Reparatur deutlich aufwendiger. Zuerst muss das betroffene LOB-Segment genau ermittelt werden. In einem zweiten Schritt wird mit einer PL/SQL-Routine (Skript und Verfahren sind in MOS-Note 293515.1 genau beschrieben) die Information, welche Zeilen in den LOBs korrupt sind, in eine Hilfstabelle geschrieben. Danach werden die korrupten Blöcke mithilfe der Information aus der Hilfstabelle geleert *(siehe Listing 8)*.

```
BEGIN
   DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
   SCHEMA_NAME => '&schema_name',
   OBJECT_NAME => '&table_name',
   OBJECT_TYPE => dbms_repair.table_object,
   FLAGS => dbms_repair.SKIP_FLAG);
   END;
   /

   alter table &table_name move [online];
```

Listing 7

```
 update &table_owner.&table_with_lob
      set &lob_column = empty_blob()
      where rowid in (select row_id from bad_rows);
```

Listing 8

Im Anschluss daran sollte das LOB-Segment in einen anderen Tablespace verschoben werden. Im Zuge der vorherigen Korrektur wird ausschließlich der Pointer auf die fehlerhaften Blöcke umgebogen. Ohne Verschieben des LOB-Segments können bei einem erneuten Insert von Daten die korrupten Blöcke wieder herangezogen werden, was zu einem erneuten Abbruch der Aktion führt.

Ein besonderes Augenmerk ist auch beim Einsatz von Data Guard geboten. Eine Reparatur erfolgt durch ein Incremental- oder Full-Backup von der Primärseite (MOS-Note 958181.1). Ist die Datenbank-Version 12c oder neuer, gibt es die Möglichkeit, die Standby-Datenbank mittels RMAN direkt über den Service-Namen der Primary zu korrigieren (MOS-Note 1987763.1). Dies vereinfacht einen Teil der Schritte, ist jedoch weiterhin aufwendig. Aus diesen Gründen ist es nicht empfehlenswert, Nologging in Data-Guard-Umfeldern einzusetzen, zumindest bis Version 12.1.

Eine Active-Data-Guard-Lizenz vorausgesetzt, gibt es in Version 12.2 ein Feature, um die Korruption zu vermeiden. Dabei werden die Informationen aus der View „v$nonlogged_block" der Primärseite in das Controlfile der Standby-Datenbank übertragen. Ist die Nologging-Operation erfolgreich auf der Primärdatenbank durchgeführt, kann mit dem RMAN-Befehl „recover database nonlogged block;" auf der Standby-Datenbank die Konsistenz wiederhergestellt werden. Dies stellt einen Ansatz dar, um Nologging auch in Data-Guard-Umfeldern einzusetzen. Wichtig ist dabei, die Aktion in den Prozess des Datenladens mit einzubinden.

18c-Verbesserungen

In 18c gibt es Verbesserungen der Nologging-Funktionalität für Data Guard auf Engineered Systems (Exadata/Oracle Database Appliance) und in der Oracle-Cloud (ab EE-ES). Zusätzlich zu diesen Restriktionen ist auch für diese die Active-Data-Guard-Lizenz erforderlich. Sind die Voraussetzungen erfüllt, sind zwei neue Modi für Data Guard möglich:

- Standby Nologging for Data Availability
- Standby Nologging for Load Performance

In der ersten Variante liegt der Fokus auf Verfügbarkeit und Wiederherstellbarkeit. Dabei werden für den Data-Guard-Transport alle Nologging-Operationen in der Primär-Instanz wie gehabt „nologging", beim Transport zur Standby-Datenbank als „logging" ausgeführt. Dafür ist der Commit verzögert, bis alle Standby-Datenbanken den Empfang quittiert haben. Aktivieren lässt sich dieser Modus mit „alter database set standby nologging for data availability".

Im zweiten Fall ist der Sachverhalt etwas anders. Es wird versucht, den Datenstrom so schnell wie möglich zur Standby-Datenbank zu transferieren. Ist dies aufgrund von Netzwerk-Engpässen nicht möglich, wird der Transfer gestoppt und der Standby-Datenbank fehlen wie in vorigen Versionen die Daten der Nologging-Operationen. Tritt diese Situation ein, werden im Zuge des Managed Recovery die Daten nachgezogen. Dieser Modus lässt sich mit dem Befehl „alter database set standby nologging for load performance" aktivieren.

Wer den Mittelweg aus Nologging und Einsatz von Data Guard sucht, hat diesen mit dem Modus „Data Availability" gefunden. Der zweite Modus „Load Performance" verspricht die gleiche Performance in Data-Guard-Systemen wie auf einer Nicht-Data-Guard-Datenbank. Der Best-Effort-Ansatz birgt jedoch einige Fallstricke im täglichen Betrieb: Monitoring und Diagnose können unter Umständen schwierig werden, wenn besagte Netzwerk-Engpässe auftreten. Deshalb ist es auch weiterhin unverzichtbar, das Backup der Datenbank mit den Datenlade-Prozessen der Applikation zu koppeln.

Flashback Database und Nologging

Nologging-Ladevorgänge lassen sich mit den Flashback-Mechanismen der Datenbank absichern. Es bietet sich an, einen Restore Point vor der Ladeaktion zu setzen. Bricht der Ladevorgang ab oder kommt es zu einem Crash der Datenbank oder des Servers, können die Nologging-Ladevorgänge schnell mit dem Zurückgehen auf den Restore Point zurückgesetzt werden. Im Anschluss lassen sich die Daten erneut laden. Am Ende des Ladeprozesses ist es weiterhin erforderlich, ein Backup zu erstellen, um eine Korruption der Datenbestände bei einem späteren Recovery zu vermeiden.

Datapump und Nologging

Werden Daten mit Datapump in die Datenbank geladen, besteht ab Version 12.1 die Möglichkeit, den Ladevorgang im Nologging Mode durchzuführen. Dies wird mit dem Parameter „TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y" für alle Objekte oder mit „TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y:TABLE" für bestimmte Objekte aktiviert. Für die zu importierenden Objekte werden vor dem Laden die Objekte auf „nologging" gesetzt und nach erfolgreichem Laden wieder auf „logging". Auch bei diesem Modus ist unbedingt im Anschluss ein Backup der Datenbank beziehungsweise der betroffenen Objekte durchzuführen.

Fazit

Oracle hat einiges für das Handling von Nologging-Operationen über die Versionen hinweg getan. Weiterhin gilt die Devise: Wenn nicht unbedingt notwendig, sollte Nologging vermieden werden. Ist dies nicht möglich, muss dies für alle am Betrieb der Applikation beteiligten Personen transparent sein, um einen Datenverlust zu vermeiden. Essenziell dabei ist auch die Verknüpfung der Applikationsprozesse mit den Backup-Routinen der Datenbank.

Den Verbesserungen in Data-Guard-Systemen bis hin zu den Features in 18c zum Trotz bleibt es weiterhin schwierig, diese Umgebungen vollumfänglich effizient zu managen. Die Nichtverfügbarkeit der Features außerhalb von Engineered Systems beziehungsweise der Oracle Cloud schränken den Nutzen für die Allgemeinheit weiter ein.

Wer ohnehin Konsistenz als oberste Maxime hat, sollte seine Datenbanken auf Force Logging stellen, damit auch unbeabsichtigte und ungeplante Nologging-Operationen nicht stattfinden können.

Timo Giese
timo.giese@fiduciagad.de