Nologging Objects von 11g bis 18c

Timo Giese
Fiducia & GAD IT AG
Karlsruhe

Schlüsselworte

Oracle Database Nologging Recovery Betrieb 11g 12c 18c

Einleitung

Nologging Operationen werden meist in Applikationen mit Data Warehouse Charakter eingesetzt. Dies ist darin begründet, dass dort meist große Datenmengen aus OLTP Systemen geladen werden und durch Nologging Operationen ein höherer Durchsatz / schnelleres Laden einhergeht. Dies wiederum hat den Effekt, dass bestimmte Teile der Datenbank dadurch für einen gewissen Zeitraum inkonsistent sind und im Falle eines Crash Recoveries im ungünstigsten Fall nicht wiederhergestellt werden können. Damit dies nicht passiert oder besser gesagt eine Lösung für diese Konstellation gefunden wird gilt es zuerst zu verstehen, wie sich Nologging Objekte / Operationen in der Datenbank darstellen, welche Möglichkeiten es gibt diese in den Griff zu bekommen und was benötigt wird um im Wiederherstellungsfall keine Daten zu verlieren.

Was sind Nologging Objects / Operations?

Nologging Operations sind wenig protokollierte Transaktionen in der Datenbank. Die Betonung liegt auf wenig, da trotz dem Namen Nologging minimale Informationen darüber weiterhin in die Redologs geschrieben werden. Es werden aber keine vollumfänglichen Transaktionsinformationen hineingeschrieben und deshalb ist im Falle eines Datenbank-Recoveries für diese Daten/Objekte kein konsistenter Stand wiederherzustellen. Führt nun ein DBA das Recovery durch, kommt diese Operation erfolgreich zum Schluss, obwohl weiterhin die Nologging Objekte/Daten inkonsistent sind. Dies äußert sich für die Applikation schon bei einem einfachen select auf die Datenbereiche mit der Oracle Fehlermeldung „`ORA-26040: Data block was loaded using the NOLOGGING option`“.

Wenn diese Konstellation auftritt sind die Daten unwiderruflich weg. Eine Wiederherstellung mit Oracle Mitteln ist nicht mehr möglich. Es bleibt nur die Möglichkeit die Daten durch die Applikation neu in die Datenbank laden zu lassen.

Einsatzbereich

Nologging Operationen werden bevorzugt im Data-Warehouse Umfeld eingesetzt, oder im DSS / Mixed Workload Umfeld. Im weitesten Sinne macht es Sinn darüber nachzudenken, wenn große Mengen an Daten in einen Stagingbereich geladen werden und diese im Anschluss dann in einem weiteren Prozessschritt aufbereitet, umgeformt und transformiert werden. In diesen Szenarien kann die Laufzeit durch Minimierung des „Logging“ optimiert werden.

Nologging Objekte

Typischerweise kommen für Nologging Objekte Tabellen, Indizes und LOB-Objekte in Frage.

Operationen die zu Nologging Objekten führen

Nicht jede Aktion führt auch zu Nologging Objekten. Grundbedingung ist zuerst, dass das gewünschte Objekt als „Nologging Objekt“ angelegt wurde (z.B. alter table test nologging;).

Danach können folgende Operationen als 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
```

Verhindern von Nologging Operationen

Nologging Operationen können in bestimmten Umfeldern nicht gewünscht sein, wie z.B. in einem Dataguard Umfeld, bei dem es sonst zusätzlich auf der Standby-Seite zu Inkonsistenzen führen kann. Ein weiterer Grund kann auch eine generelle Policy sein, dass Datenkonsistenz wichtiger als Performance ist. Damit auch nicht fälschlich Nologging Operationen durch Entwickler / Application-DBAs passieren gibt es Möglichkeiten dies zu unterbinden.

Die globale Variante ist auf Datenbankebene und schreibt alle Nologging Operationen auf Logging um. Dies wird erreicht mit dem Befehl

```
alter database force logging;
```

Eine andere Variante ist es Nologging Operationen für die vorhandenen Tablespaces zu deaktivieren. Dies wird mit dem Befehl

```
alter tablespace <TS_NAME> force logging;
```

erreicht.

Abschalten lässt sich das Ganze wieder mit

DB-Level:

```
alter database no force logging;
```

Tablespace-Level:

```
alter tablespace <TS_NAME> no force logging;
```

Ob in der Datenbank bzw. auf Tablespaces force logging aktiv ist kann mit folgenden SQLs abgefragt werden:

```
select force_logging from v$database;

select force_logging from dba_tablespaces;
```

Nologging Operationen feststellen

Nologging Objekte können klassich mit DBVERIFY (dbv) in älteren Datenbankversionen <= 10.2.0.4 festgestellt werden. Dabei werden Blöcke mit nonlogged data mit der Fehlermeldung „DBV-00200“ und „DBV-00201“ als Ergebnis zurückgeliefert.

Ab 10.2.0.5 werden Nologging Objekte mittels RMAN und dem Befehl „`validate [check logical] database;`“ festgestellt.

In 11g werden die Blöcke in der View „v$database_block_corruption“ mit „CORRUPTION_TYPE=NOLOGGING“ hinterlegt.

In 12c ist das in der View „v$nonlogged_block“.

Ab 12.2 gibt es ein neuen RMAN Befehl, der die Ergebnisse direkt im RMAN anzeigt.

```
validate database nonlogged block;

List of Datafiles
=================
File Status Nonlogged Blocks Blocks Examined Blocks Skipped
---- ------ ---------------- --------------- --------------
1        OK 0                         106363 0
2        OK 0                          78919 0
3        OK 0                          96639 0
4        OK 0                           4991 0
5        OK 400                         2559 0
```

Weiterhin wird die View „v$nonlogged_block“ mit der gleichen Information befüllt. Zusätzlich wird im Datenbank Alertlog ebenfalls ein Eintrag für jeden gefundenen Nonlogged Block geschrieben (Finished Nonlogged Block Replacement recovery(validate) on file 5. 400 blocks found).

Welche Objekte in der Datenbank zu Nologging Operations führen kann mit dem folgenden SQL angezeigt werden:

```
select distinct ss.owner,ss.object_name, ss.object_type
,ss.tablespace_name, ts.logging tablespace_level_logging
from
v$segment_statistics ss, dba_tablespaces ts,v$datafile df
where
ss.statistic_name ='physical writes direct'
and ss.value >0
and df.unrecoverable_change# >0
and ss.ts#=df.ts#
and ss.tablespace_name=ts.tablespace_name
/
```

Des Weiteren können alle Datenfiles die von Nologging Operationen betroffen sind mit dem RMAN Befehl „`REPORT UNRECOVERABLE;`“ angezeigt werden. Dabei wird auch gleich die Lösung mit angegeben, damit im Fall eines späteren Recoveries keine Probleme entstehen. Die Lösung für Nologging Operationen ist immer ein Volles oder ein inkrementelles Backup, denn dabei werden alle oder alle geänderten Datenblöcke gesichert.

Die gleiche Information kann auch mittels des folgenden SQLs gefunden werden:

```
SELECT df.name "datafile", df.unrecoverable_time
FROM v$datafile df, v$backup bk
WHERE df.file#=bk.file#
and df.unrecoverable_change#!=0;
```

„Reparatur" von Nologging Objekten

Nologging Indices werden einfach durch Neuerstellen des Indexes wieder validiert.

Eine Restauration von Tabellen ist in dem Sinne nur möglich, dass keine korrupten Blöcke mehr darin vorhanden sind. Die Daten müssen durch andere Mechanismen, z.B. ein erneutes Laden durch die Applikation erfolgen.

Die Reparatur erfolgt mittels dem Package DBMS_REPAIR.

Zuerst wird für die Tabelle die Einstellung gesetzt, korrupte Blöcke zu ignorieren:

```
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;
/
```

Im 2. Schritt wird die Tabelle mit „`alter table &table_name move;`" verschoben.

Weitere Dokumentation zu diesem Prozedere befindet sich in der MOS-Note 794505.1

Befindet sich die Korruption in einem LOB-Segment ist das ganze ungleich schwieriger. Eine genaue Beschreibung der Prozedur kann in der MOS-Note 293515.1 gefunden werden.

In einer Dataguard-Umgebung ist das detaillierte Prozedere zur Korrektur der Standby-Seite in den MOS-Notes 958181.1 (Rolling a Standby Forward using an RMAN Incremental Backup To Fix The Nologging Changes) und 1987763.1 speziell für Datenbanken >=12c (Rolling Forward a Physical Standby Using Recover From Service Command in 12c) dokumentiert.

18c Verbesserungen

In der Datenbankversion 18c sind zu den bereits vorhandenen Features 2 weitere Modi im Zusammenhang mit Dataguard hinzugekommen.

Die eine Option ist „Standby Nologging for Data Availability" und die andere ist „Standby Nologging for Load Performance".

In der Variante (Data Availability) werden Nologging Operationen mit einer eigenen Verbindung zur Standby Datenbank (alle vorhandenen Standby DBs) gesendet. Der Commit wird verzögert, bis alle Daten im Rahmen des Managed Recovery auf allen Standby DBs appliziert wurden.

In der Variante (Load Performance) wird versucht die Daten sofort an die Standby DB zu schicken. Ist dies auf Grund von Netzwerkbottelnecks nicht möglich, wird der Transfer gestoppt und die Standby DB hat dann einige Daten in dieser Operation nicht erhalten. Im Zuge des Managed Recovery werden die Daten auf den Standbys dann nachgezogen.

Beiden Varianten gemein ist das Vorhandensein der Active Dataguard Option. Des Weiteren sind diese Varianten zur auf Exadata Engineered Systems und in der Cloud verfügbar.

Datapump Nologging Import

Dies ist ab 12.1 möglich, dabei wird während des Imports die entsprechenden Objekte auf Nologging und am Ende wieder in den Ursprungszustand zurück gesetzt.

Option für den Import:

```
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y
```

Einschränkung auf bestimmte Objekttypen:

```
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y:TABLE
```

Flashback Database und Nologging

Passt Flashback Database / Restore Point und Nologging Operationen zusammen? Ja, denn vor einer Nologging Operation kann ein Restore Point erstellt werden, auf den man im Bedarfsfall schnell wieder zurückkann.

Nologging Datenblock vs Logging Datenblock

Was ist nun der Unterschied zwischen einem Logging Datenblock und einem Nologging Datenblock? Es gibt keinen. Beim Einfügen der Daten ist der Block im Datenfile identisch. Es wird nur „kein“, bzw. minimal Redolog Informationen beim Nologging Insert geschrieben, was im Recoveryfall zur Folge hat, das alle Nologging Objekte – und alle zugehörigen Blöcke – nicht mehr für die Datenbank lesbar sind, da nach einem Recovery die Blöcke mit den Daten leer sind und als defekt markiert sind.

Wird der Inhalt eines Datenblocks als Tracedatei gedumpt (alter system dump datafile X block Y;), enthält dieser verkürzt folgende Informationen:

```
…
Start dump data blocks tsn: 6 file#:14 minblk 189 maxblk 189.← Tbs File-ID
Block dump from disk:          ← Block ist von der Festplatte gelesen
buffer tsn: 6 rdba: 0x038000bd (14/189)
scn: 0x4a4773 seq: 0x01 flg: 0x06 tail: 0x47730601
frmt: 0x02 chkval: 0x6196 type: 0x06=trans data
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F0B86EE7000 to 0x00007F0B86EE9000
7F0B86EE7000 0000A206 038000BD 004A4773 06010000  [........sGJ.....]
7F0B86EE7010 00006196 00000001 00012F7B 004A476D  [.a......{/..mGJ.]
7F0B86EE7020 00008000 00320002 038000B8 00180001  [......2.........]
7F0B86EE7030 00000748 02403CAB 000D0232 00002002  [H....<@.2.... ..]
7F0B86EE7040 004A4773 00000000 00000000 00000000  [sGJ.............]
7F0B86EE7050 00000000 00000000 00000000 00000000  [................]
...
```

```
7F0B86EE8FC0 000300AC 00000000 0022BB40 BB400001  [........@."...@.]
7F0B86EE8FD0 02010022 C10202C1 03C30404 00AC1B01  ["..............]
7F0B86EE8FE0 00000003 0202012C 570503C1 646C726F  [....,......World]
7F0B86EE8FF0 0202012C 480502C1 6F6C6C65 47730601  [,......Hello..sG]
Block header dump:  0x038000bd
 Object id on Block? Y
 seg/obj: 0x12f7b  csc:  0x00000000004a476d  itc: 2  flg: E  typ: 1 - DATA
     brn: 0  bdba: 0x38000b8 ver: 0x01 opc: 0
     inc: 0  exflg: 0

 Itl           Xid                  Uba         Flag  Lck        Scn/Fsc
0x01   0x0001.018.00000748  0x02403cab.0232.0d  --U-    2  fsc
0x0000.004a4773
0x02   0x0000.000.00000000  0x00000000.0000.00  ----    0  fsc
0x0000.00000000
bdba: 0x038000bd
data_block_dump,data header at 0x7f0b86ee7064
===============
tsiz: 0x1f98
hsiz: 0x16
pbl: 0x7f0b86ee7064
     76543210
flag=--------
ntab=1
nrow=2
frre=-1
fsbo=0x16
fseo=0x1f80
avsp=0x1f6a
tosp=0x1f6a
0xe:pti[0]     nrow=2         offs=0
0x12:pri[0]    offs=0x1f8c
0x14:pri[1]    offs=0x1f80
block_row_dump:                ← einzelne Zeilenelemente
tab 0, row 0, @0x1f8c
tl: 12 fb: --H-FL-- lb: 0x1  cc: 2
col  0: [ 2]  c1 02
col  1: [ 5]  48 65 6c 6c 6f
tab 0, row 1, @0x1f80          ← Zeile 1
tl: 12 fb: --H-FL-- lb: 0x1  cc: 2
col  0: [ 2]  c1 03            ← Spalte 1 ASCII-Zeichen
col  1: [ 5]  57 6f 72 6c 64   ← Spalte 2 ASCII-Zeichen
end_of_block_dump
End dump data blocks tsn: 6 file#: 14 minblk 189 maxblk 189
```

Wird nach einem Recovery der Dump des Blocks durchgeführt wird dieser nicht angezeigt, da dieser weiterhin als „corrupt“ markiert ist und nicht mehr verwendet werden kann. Im Dumpfile wird dies mit der Meldung

```
„table scan: segment: file# 13 block# 2258
             skipping corrupt block file# 13 block# 2259
```

angezeigt.

Demo

In der Demo wird anhand eines Beispiels die Auswirkung eines Nologging Create Table As Select auf die Strukturen der Datenbank, sowie die Einschränkungen der Recoveryfähigkeit und Auswirkungen auf die Applikation im Falle eines Recoveries gezeigt.

Kontaktadresse:

Timo Giese

Fiducia & GAD IT AG

Fiduciastraße 20

D-76227 Karlsruhe

Telefon: +49 (0) 721 4004-1017

E-Mail timo.giese@fiduciagad.de

Internet: www.fiduciagad.de