Ora-01578 – Umgang mit korrupten Blöcken

Thorsten Grebe, November 2016

Schlüsselworte

Blockkorruption, schleichende Datenkorruption, db_block_checking, db_block_checksum, db_lost_write_protect, db_ultra_safe.

Einleitung

Der Fehlercode 1578 steht für Blockkorruptionen. Er zählt zu den unangenehmsten Fehlercodes, die den DBA im Alltag treffen können und fordert in der Regel umgehende Analyse und Reaktion. Das Spektrum der Fehlerursachen ist groß und die Art möglicher Schäden schwankt von harmlos bis katastrophal. Ein defekter Block kann in einem unbenutzten Datensegment oder rekonstruierbarem Index auftreten – er kann aber auch irreversible Schäden mit unvermeidlichem Datenverlust bedeuten. Oracle bietet ein ganzes Arsenal von Methoden zur Analyse und Behandlung defekter Blöcke. Eine Sichtung dieser Verfahren lässt jedoch schnell die Erkenntnis reifen, dass der DBA die Konfrontation mit defekten Blöcken unter allen Umständen meiden sollte. Und es liegt nahe, dass im Falle auftretender Blockkorruptionen die frühestmögliche Erkennung und Analyse des Defekts von entscheidender Bedeutung ist, um eskalierende Folgeschäden zu minimieren.

Die Präsentation gibt einen Überblick über die Arten von Blockkorruptionen und die Möglichkeiten, die Oracle für frühe Erkennung und Analyse diverser Blockschäden entwickelt hat. Ein besonderer Fokus liegt auf drei Initialisierungsparametern der Datenbank, die eine Früherkennung von defekten Blöcken wesentlich verbessern: db_block_checksum, db_block_checking und db_lost_write_protect. Sie können in 36 unterschiedlichen Kombinationen gesetzt werden. In einer Standarddatenbank ist nur einer dieser drei Parameter aktiviert, die anderen beiden sind deaktiviert. Warum hat sich Oracle für diese eher lasche Standardeinstellung entschieden und wie ressourcenaufwändig wäre die vollumfängliche Aktivierung aller Prüfoptionen? Die Ergebnisse einer Analyse zu diesem Thema sollen vorgestellt werden.

Kategorisierung von Blockschäden

Oracle unterscheidet sechs grobe Kategorien von Datenbank-Korruptionen. Geläufig ist die Unterscheidung in physische und logische Defekte. Physische Defekte sind in der Regel durch Fehlfunktionen in Hardware- oder externen Software-Komponenten bedingt. Der

Initialisierungsparameter db_block_checksum regelt die Erstellung von Prüfsummen zum Aufspüren dieser Art Fehler.

Logische Defekte sind dadurch gekennzeichnet, dass ein Oracle-Block äußerlich intakt zu sein scheint, jedoch stimmt in seinem inneren, logischen Aufbau etwas nicht. Bei logischen Defekten können Korruptionen auf einen Block beschränkt sein oder mehrere Blöcke umfassen, was eine weitere Unterkategorisierung in Inter- und Intra-Blockkorruptionen notwendig macht. Logische Defekte können mittels Prüfsummen entdeckt werden, die über den Parameter db_block_checking reguliert werden.

Auch Nologging-Vorgänge rufen bei Recover-Versuchen den Fehler ORA-01578 hervor und bedeuten bei Datensegmenten in der Regel das Scheitern einer verlustfreien Wiederherstellung. Es gibt keine Prüfsumme zum Entdecken von Nologging-Vorgängen, denn ein mit Nologging markierter Block ist an sich nicht korrupt. Es fehlt für ihn lediglich die Wiederherstellungsinformation. Sie lassen sich über RMAN (report unrecoverable) oder über Views wie v$datafile oder v$nonlogged_block anzeigen. Nologging-Blöcke sind harmlos, solange der DBA über ihre Existenz informiert ist und in der Lage ist, inkrementelle Sicherungen nachzuholen bevor ein Restorefall eintritt.

```
  v$backup_corruption
  v$copy_corruption
  v$database_block_corruption
  v$datafile

neu in 12c:
  v$nonlogged_block
  v$backup_nonlogged
  v$copy_nonlogged
```

Abb. 1: Views, die beim Erkennen von Blockkorruptionen helfen.

Äußerst tückisch sind Korruptionen, die durch fehlerhafte Storage-Komponenten zu deplatzierten Blöcken führen. Verlegt der Disk-Controller einen Datenblock oder schreibt ihn gar nicht, meldet aber fälschlich eine ordnungsgemäße Speicherung zurück, hat eine Oracle Instanz kaum eine Chance diesen Fehler während des Regelbetriebs zu bemerken. Sie verlässt sich darauf, dass der Controller weiß was er tut. Derartige Patzer im Storage Layer galten lange Zeit als besonders schwer nachweisbar. Erst in Version 11g führte Oracle mit dem Parameter db_lost_write_protect Prüfsummen ein, die beim Aufspüren dieser Fehlergruppe unterstützen.

Zwei weitere Schadenstypen, die von Oracle gesondert kategorisiert werden, sind Memory Korruptionen und das Auftreten eines ORA-01578 unmittelbar beim Start einer Instanz.

Lokalisierung und Reparaturoptionen

Noch bedeutender als die Art des Blockschadens ist für den Datenbank-Administrator, der den Schaden beheben muss, dessen Lokalisierung. Egal, ob ein ORA-01578 aufgrund eines physischen oder logischen Defekts auftritt, der Block muss in der Regel durch eine intakte Kopie ersetzt werden. Es ist jedoch die Lage des Schadens, die ausschlaggebend für die Wahl der Reparaturmethode ist. Entspannung ist angesagt, wenn der Blockschaden in einem nicht verwendeten Segment liegt. Indexschäden lassen sich in der Regel mühelos und häufig online beheben. Auch andere Schäden können glimpflich verlaufen, wenn ein Schaden schnell bemerkt wird und der Administrator in der Lage ist, auf die *Block Media Recovery (BMR)* Funktion der *Enterprise Edition* zurückzugreifen. Mit dieser lassen sich beliebige Blockfehler online und chirurgisch präzise reparieren. Die *Active Dataguard Option* erledigt dies sogar automatisch im Hintergrund.

Kniffeliger wird es, wenn keine *Enterprise Edition* verfügbar ist und ganze Datendateien restauriert werden müssen. Die nächste Schwierigkeitsstufe ist erreicht, wenn der Schaden zu spät erkannt wird oder keine Sicherung verfügbar ist. Mit Einschränkungen können Daten über *datapump* gerettet oder indirekt über Indizes ausgelesen werden. Nur zur Not möchte man mit dbms_repair agieren. Denn anders als der Name suggeriert, reparieren die Methoden dieses Paketes nicht, sondern schlagen eher eine Schneise um den Brandherd, damit der reguläre Betrieb der Datenbank wieder aufgenommen werden kann. Wenn gar nichts mehr geht, bietet Oracle Unterstützung beim Erstellen und Auslesen von Hexdumps an. Sind Dutzende oder Hunderte Blöcke betroffen, oder wurden die geschädigten Segmente gar komprimiert oder verschlüsselt, dürfte dieses Verfahren sehr mühsam sein.

Die Wahrscheinlichkeit, zu solch verzweifelten Maßnahmen greifen zu müssen, lässt sich durch das Ausnutzen der zur Verfügung stehenden präventiven Techniken deutlich verringern.

Vorbereitet sein

Vorteilhaft ist es, das Spektrum der möglichen Analyse- und Rettungsmaßnahmen zu kennen, die von Oracle bereitgestellt werden, bevor ein ORA-01578 auftritt. Wenn es zum Auftreten einer Korruption kommt, muss möglicherweise sehr schnell entschieden werden, ob das Korruptionsereignis auf die leichte Schulter genommen werden kann oder ein sofortiges Herunterfahren der Datenbank angezeigt ist. Dies wäre z.B. der Fall, wenn ein fehlerhafter Controller gerade dabei ist, in schneller Folge den

Datenbestand zu zerstören. Je mehr Anwender jetzt noch Arbeit in den totgeweihten Datenbestand investieren, desto aufwändiger und teurer wird die Reparatur.

Es ist empfehlenswert, für solche Situationen die passenden Kommandos griffbereit zu haben. RMAN bietet Komfortkommando zum Validieren der gesamten Datenbank oder einzelner Dateien. DBV hilft bei Offline-Analysen und das altgediente Analyze-Kommando unterstützt beim Prüfen einzelner Tabellen auf Schädigungen (Abb. 2).

```
-- gesamte Datenbank
   RMAN> backup validate [check logical] database ;


-- einzelne Datendatei
   RMAN> backup validate [check logical] datafile <abs. Dateinr.> ;

   oder
   $ dbv file=<Dateiname.>


-- einzelne Blöcke
   RMAN> validate datafile <Datei-Nr.> block <Block-Nr>, <Block-Nr> ;


-- einzelne Tabelle untersuchen
   SQL> analyze <Tabellen_Name> validate structure [cascade] ;
```

Abb. 2: Komfort-Kommandos zum Erkennen von Block-Korruptionen.

Schnelles Erkennen eines Schadens

Eine faire Chance, Block-Korruptionen abzuwehren, hat der Administrator nur dann, wenn ein Schaden bemerkt wird, bevor er sich sowohl in den Lebenddaten als auch in der Sicherung festgesetzt hat. Durchdringt eine logische Korruption die RMAN Sicherungen, bevor sie bemerkt wird, kann eine Tabelle ohne außerordentliches Glück nicht mehr verlustfrei hergestellt werden.

Bei Nologging-Vorgängen und Block-Korruptionen in den Backupsets von RMAN verhält es sich umgekehrt. Der Datenbank geht es prächtig, aber die Sicherung ist nicht mehr verlässlich. Ist man in dieser Situation zu einem Restore gezwungen, ist Datenverlust unvermeidbar.

Der Schlüssel zur frühen Erkennung von Blockschäden liegt in den Validierungsmethoden, die der Recovery Manager, das Verifizierungsprogramm dbv oder das Prüfpaket dbms_hm anbieten. Zusätzlich scheint es ratsam, über die drei Initialisierungsparametern db_block_checksum, db_block_checking und db_lost_write_protect das Erstellen von Prüfsummen höher zu regulieren, als

es die Standardvorgabe vorsieht. Denn ohne ausreichende Prüfsummen helfen Validierungsläufe nur begrenzt. Validierungen kontrollieren Prüfsummen. Eine Validierung kann daher nur so gut sein, wie die Prüfsummen, die vorbereitend erstellt wurden.

Von den drei Kontrollparametern für die Prüfsummenberechnung wird von Oracle in einer regulären Datenbank (d.h. nicht Exadata) nur ein einziger dieser Parameter, db_block_checksum, aktiviert und das noch nicht einmal vollständig: er wird mit der Einstellung TYPICAL ausgeliefert. Erst auf den Wert FULL gesetzt, würden alle Prüfsummen aktiviert, die Oracle zur Früherkennung von physischen Defekten vorsieht.

Der Administrator muss sich selbst darum kümmern, welche Validierungen und Prüfsummen er aktivieren möchte. Oracles Zurückhaltung beim vollumfänglichen Ausschöpfen aller Möglichkeiten zur Korruptionsbekämpfung haben jedoch einen triftigen Grund: gelegentlich führen die Zusatzprüfungen zu einem nicht zumutbaren zusätzlichen Ressourcenverbrauch. Gepflastert ist der Weg der Prüfsummenberechnung auch mit zahlreichen Bugs, aus denen Oracle keinen Hehl macht. Die behobenen Bugs werden in der Reference-Note zu den Initialisierungsparametern im Support-Portal angezeigt (s. Abb. 3).

DB_BLOCK_CHECKING – Bugs

aus **Init.ora Parameter "DB_BLOCK_CHECKING" Reference Note (Doc ID 68483.1)**

NB	Prob	Bug	Fixed	Description
	I	14468919	11.2.0.4, 12.1.0.2, 12.2.0.0	DB_BLOCK_CHECKING for OLTP table is slow with TDE
	II	14674963	12.1.0.1	ORA-600 [kdBlkCheckError][1] / corruption possible from OLTP compression
	I	14044105	11.2.0.3.BP18, 11.2.0.4, 12.1.0.1	Block checking is slow on OLTP compressed tables for DBV/analyze/DB block checking
	II	18252706	12.2.0.0	Block checking does not detect mboff > total_ktspfhc in L1 Bitmap Block - ASSM Tablespace
	III	13804294	11.2.0.3.4, 11.2.0.3.BP07, 11.2.0.4, 12.1.0.1	Internal errors, corruptions, using pipelined function whose rows raise exceptions
	II	11723722	11.2.0.2.5, 11.2.0.2.GIPSU05, 11.2.0.3, 12.1.0.1	ORA-3106 reading Securefiles by OCI sqltype SQLT_LBI and SQLT_CHR / ORA-16211 at logical standby
	II	9393307	11.2.0.2, 12.1.0.1	DBVerify / RMAN / ANALYZE not detecting logical corruption in index with avsp < 0
	II	9350204	11.2.0.3, 12.1.0.1	Spurious ORA-600 [kddummy_blkchk] .. [6145] during CR operations on tables with ROWDEPENDENCIES
E	II	8837919	11.2.0.2, 12.1.0.1	DBV / RMAN enhanced to detect ASSM blocks with ktbfbseg but not ktbfexthd flag set as in Bug 8803762
E	III	8720802	10.2.0.5, 11.2.0.1.BP07, 11.2.0.2, 12.1.0.1	Add check for row piece pointing to itself (db_block_checking,dbv,rman,analyze)
	I	8483871	11.1.0.7.2, 11.2.0.1	Securefile operations are much slower when DB_BLOCK_CHECKING is enabled
	I	7481798	11.2.0.1	Excessive BLOCK CHECKING IS NOT DONE" trace contents and/or ORA-7445[kghalf]
	-	4684074	10.2.0.2, 11.1.0.6	OERI:510 / block corruption (ORA-1578) with DB_BLOCK_CHECKING
	I	4493447	11.1.0.6	Spurious ORA-600 [kddummy_blkchk] [file#] [block#] [6145] on rollback of array update
E	-	4622960	9.2.0.8, 10.2.0.1	Less resource intensive block checking options needed
	-	2967417	9.2.0.4, 10.1.0.2	OERI:KCOAPL_BLKCHK may occur using Auto managed segment
	-	2562274	9.2.0.3, 10.1.0.2	Block corruption / OERI[KCOAPL_BLKCHK] updating LONG data
	-	2106455	8.1.7.4, 9.0.1.4, 9.2.0.1	Dump possible in KDB4_DUP_KEYS
	-	1802432	8.1.7.3, 9.0.1.1, 9.2.0.1	Migration from V7->816/817 gives OERI:KCOAPL_BLKCHK for clusters with FREELISTS

Abb. 3: Das Image von db_block_checking litt unter einer Vielzahl von Bugs. Hier eine Liste aus Oracles Reference Note 68483.1. Die gezeigten Bugs gelten als behoben. Markiert sind drei Fehler, die noch in späten Versionen unangenehme Auswirkungen auf die Performance haben konnten.

Die Kosten der Prävention

Was kostet es, zusätzliche Validierungsoptionen zu aktivieren? Validierungsläufe von RMAN, die mit der *check logical* Optionen durchgeführt werden, sind zwangsläufig IO- und CPU-intensiv und können mit dem regulären Tagesbetrieb kollidieren. Um eine 1TB große Datenbank zu validieren, muss RMAN 1TB Daten lesen. Es ist offensichtlich, dass Zeitpunkt und Intervall einer solchen Validierung abgewogen werden müssen. Ähnliches gilt für das Validieren von Backupsets.

Schwieriger einzuschätzen sind die Kosten, die durch die Initialisierungsparameter db_block_checksum, db_block_checking und db_lost_write_protect verursacht werden. Besonders heikel scheint db_block_checking zu sein. Für diesen Parameter gibt die offizielle Dokumentation den in Abb. 4 gezeigten Hinweis.

> Block checking typically causes 1% to 10% overhead in most applications, depending on workload and the parameter value. Specific DML overhead may be higher. The more updates or inserts in a workload, the more expensive it is to turn on block checking. You should set DB_BLOCK_CHECKING to FULL if the performance overhead is acceptable.

Abb. 4: Auszug aus der Oracle Referenz zum Initialisierungsparameter db_block_checking

Die Performance-Kosten für das Aktivieren dieses Parameters, der per Vorgabe deaktiviert ist, liegt laut Dokumentation bei 1-10%, kann aber abhängig vom Update- und Insert-Aufkommen höher liegen. Es wird empfohlen ihn zu aktivieren, wenn der Performanceüberhang akzeptabel ist. Doch was heißt hier „*may be higher*“ (Abb. 4)? Wenn 1% bis 10% erwartet wird, entspricht dann „*may be higher*“ 20% oder gar 30%? Und wie sollte man die anderen beiden Parameter, db_block_checksum und db_lost_write_protect, setzen? Insgesamt lassen die drei Prüfsummenparameter 36 Kombinationen zu. Der statische Metaparameter db_ultra_safe sieht nur zwei Kombinationen aus diesen 36 vor. Was ist sinnvoll, was ist aus Performance-Sicht akzeptabel? Gibt es Seiteneffekte bei ungünstigen Kombinationen?

Um dies besser abschätzen zu können, wurden verschiedene Tabellentypen erstellt (reguläre Tabellen, mit und ohne Index, komprimierte Tabellen, IOTs, Clustertabellen, partitionierte Tabellen). Gegen diese Tabellen wurden Insert-, Update-, Delete- und Select-Statements ausgeführt. Dabei wurden systematisch alle 36 Kombinationen der Prüfsummenparameter durchprobiert und Performancekenndaten aus v$mystat gesammelt.

Das Ergebnis der Analyse zeigt, dass bei vollständiger Aktivierung der Prüfsummenberechnung „*may be higher*“ mühelos Performanceüberhänge von 100% bis 500% bedeuten kann.

Kontaktadresse:

Dr. Thorsten Grebe
twg-it Unabhängige Oracle Datenbankberatung
Geisenheimer Straße 6
D-14197 Berlin

Telefon: +49 (0) 30-2160 2272
E-Mail Thorsten.Grebe@twg-it.de
Internet: www.twg-it.de