Undo Tablespace statt Blockaden – Blick in die Vergangenheit

Thomas Klughardt
Dell Software
Köln

Schlüsselworte

Undo Tablespace, Isolation, Konsistenz, Lesen ohne Sperren, Rollback, Flashback, Consistent Read

Einleitung
Daten in einer relationalen Datenbank müssen stets konsistent zueinander sein, Transaktionen dürfen nur komplett oder gar nicht durchgeführt werden und verschiedene Sessions sich untereinander nicht in die Quere kommen. Das scheibt das ACID Modell vor und bei Oracle werden diese Eigenschaften über den Undo Tablespace und die darin befindlichen Daten ermöglicht. In diesem Vortrag geht es darum, wie Oracle über die beschriebenen Eigenschaften hinausgeht und sogar konsistente Abfrageergebnisse bietet, ohne dabei auf Sperren für schreibende Zugriffe zurückgreifen zu müssen. Und natürlich geht es auch darum, wofür man diesen Mechanismus sonst noch nutzen kann.

Anforderungen an das Oracle Transaktionsverhalten
Die wichtigsten Daten in Unternehmen werden in relationalen Datenbankmanagementsystemen wie dem Oracle Database Server gespeichert. Bei diesen Daten ist die Integrität sehr wichtig und die Eigenschaften dafür sind im ACID Modell beschrieben:
A – Atomarität: Eine Transaktion wird ganz oder gar nicht durchgeführt.
C – Consistency: Daten sind zueinander stets konsistent.
I – Isolation: Erst wenn eine Transaktion abgeschlossen ist, sehen andere Sessions die Änderungen.
D – Durability: Abgeschlossene Transaktionen sind dauerhaft gespeichert.

Als wäre das nicht schwierig genug zu erfüllen, gibt es in Oracle noch weitere Eigenschaften, die man gerne als selbstverständlich hinnimmt:

1. Optimistisches Commit Verhalten: Änderungen werden sofort an die endgültige Stelle geschrieben, meistens erfolgt ein Commit der Transaktion.
2. Konsistente Abfragen: Wenn wir eine Abfrage zum Zeitpunkt x starten, bekommen wir auch die Daten zum Zeitpunkt x.
3. Keine Sperren für Lesezugriffe: Während eine Abfrage läuft dürfen die zugrunde liegenden Daten geändert werden, trotzdem ist das Ergebnis konsistent zum Startzeitpunkt der Abfrage.

Die Erfüllung aller dieser Eigenschaften macht die Oracle Datenbank zu einem hochperformanten System, das die Last schnell abarbeiten kann und auch im Multisession Betrieb sehr gut skaliert. Die Datenbank geht sogar noch einen Schritt weiter und kennt keine Lock Eskalationen. Das heißt, dass jede schreibende Änderung auch nur die betroffene Zeile sperrt und so ein nahezu blockadefreier Betrieb möglich ist.

Technische Umsetzung: Die Before Images
Um die ganzen Anforderungen zu erfüllen, arbeitet Oracle mit Vorher-Abbildern der Datensätze, den sogenannten Before Images. Wird ein Datensatz geändert, dann wird der neue Wert direkt in den Datensatz geschrieben und der vorherige Wert wird im zugehörigen Before Image gespeichert. Wenn ein Commit erfolgt, muss die Datenbank nichts weiter tun, als ein paar Metainformationen in den

Headern anzupassen, was sehr schnell geht. Deshalb sagt man auch, dass die Oracle Datenbank eine optimistische Datenbank ist, sie ist auf Commits optimiert. Erfolgt dagegen ein Rollback, dann muss der Wert aus dem Before Image in den Datensatz zurück geschrieben werden, was zwar etwas länger dauert, aber auch problemlos möglich ist.
Interessant wird der Mechanismus bei Abfragen: Anhand der System Change Number (SCN), einer eindeutigen, globalen Sequenznummer, die jede Änderung und Transaktion bekommt, kann die Datenbank nachvollziehen, zu welchem Stand die Daten abgefragt werden müssen. Ist eine Änderung nach dem Start der Abfrage passiert, wird statt des aktuellen Datensatzes das entsprechende Before Image gelesen. Auf diese Art und Weise können Sessions voneinander isoliert werden.
Der Mechanismus der Before Images kümmert sich also nicht nur um die Atomarität und Isolation aus dem ACID Modell, sondern ermöglicht nebenbei auch das optimistische Commitverhalten und konsistente Abfragen mit Lesezugriffen ohne Sperren. Er ist damit eines der wichtigsten Merkmale einer Oracle Datenbank.

Rollback Segmente : Vorfahren des Undo Tablespace
In einer Datenbank finden normalerweise viele Änderungen statt und entsprechend viele Before Images müssen geschrieben werden. Die sollten auch einige Zeit vorgehalten werden können, damit auch bei großen Abfragen die benötigten Daten noch zur Verfügung stehen. Dazu hatten in den älteren Oracle Versionen (Oracle 9 und älter) die Segmente jeweils ihre eigenen Rollback-Segmente, in die die Before Images geschrieben wurden.
Dieser Mechanismus hat es ermöglicht, Objekten auf denen viele Änderungen gemacht wurden, oder die für große Abfragen gebraucht wurden, größere Rollback Segmente zuzuweisen, als anderen Objekten. So war es auch möglich, für IO intensivere Rollback Segmente schnellere Platten zu nutzen, um den Schreibdurchsatz und die Lesegeschwindigkeit zu erhöhen. In der Praxis hat sich das nicht bewährt, zum einen musste man dafür die Anwendung sehr gut kennen, zum anderen werden inzwischen meist sowieso alle Dateien in einem gestripten und gespiegelten Storage abgelegt (SAME Prinzip). Im Gegenteil, dadurch, dass nur der jeweilige Rollback Segment Platz für das Objekt zur Verfügung stand, kam es bei großen Abfragen oft dazu, dass das entsprechende Before Image nicht mehr zur Verfügung stand, was zum berühmten `ORA-01555: Snapshot too old` Fehler führt, bei dem die Abfrage abgebrochen wird. Das ist kein echter Fehler und die Datenbank verhält sich völlig korrekt, aber wenn eine große Abfrage schon mehrere Stunden lief oder ein großer Export deshalb abgebrochen wird, ist das sehr lästig. Ärgerlicherweise tritt der `Snapshot too old` auch immer gegen Ende der Abfrage auf, weil die Before Images eine gewisse Zeit zur Verfügung stehen, dann aber irgendwann überschrieben werden.

Der Undo Tablespace
Anstatt lauter einzelne, dedizierte Rollback Segmente zu verwenden ist es sinnvoller, einen großen, gemeinsamen Speicher zu nutzen, um darin die Before Images für alle Objekte zu lagern. Dadurch gleicht sich auch der Platzbedarf aus, der für die schreibintensiven Objekte benötigt wird und auf der anderen Seite für relativ statische Objekte verschwendet wird. Aus IO Gesichtspunkten spielt es dadurch, dass inzwischen normalerweise alles gestriped und gespiegelt ist, sowieso keine Rolle, wo genau die Images liegen.
Im Undo Tablespace liegen die Before Images in den Undo Segmenten, die wie auch schon die alten Rollback Segmente, ein Ringpuffer sind. Ist das Undo Segment voll, werden die Daten darin überschrieben und stehen nicht mehr zur Verfügung.

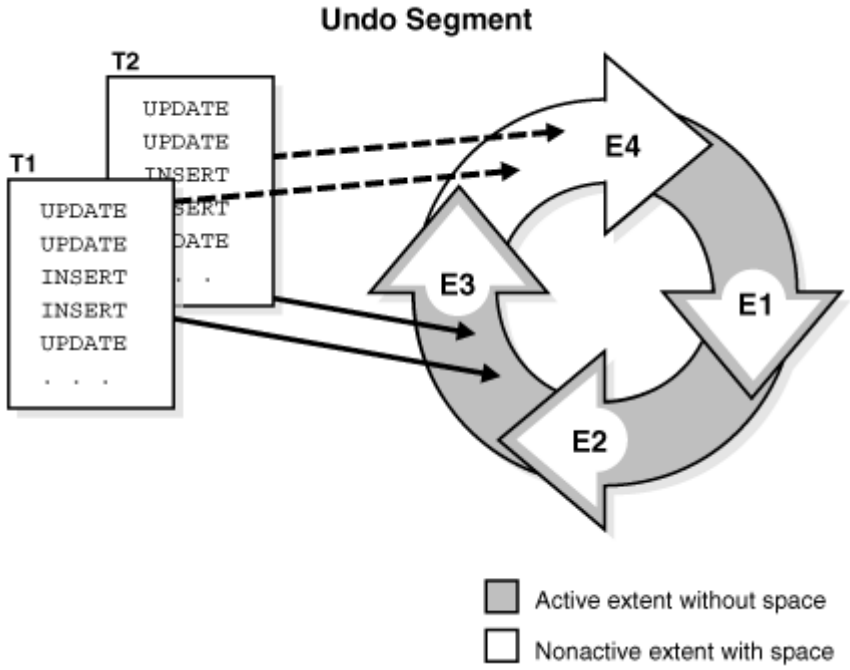

Abb. 1: Undo Segmente sind Ringpuffer (Quelle: Oracle Dokumentation)

Es kann also auch hier zu `Snapshot too old` Fehlern kommen, die aber wie gesagt wesentlich seltener auftreten als bei den Rollback Segmenten und praktisch nur noch bei sehr großen Abfragen oder beim konsistenten Exportieren großer Datenmengen mit gesetztem `FLASHBACK_SCN` Parameter zu sehen sind.

Administration, Sizing und Troubleshooting

Nachdem der Undo Tablespace so ein elementarer Bestandteil der Oracle Datenbank ist, ist es wichtig, sich Gedanken darüber zu machen, wie man ihn aufsetzen sollte. Das hängt auch mitunter davon ab, wofür man ihn nutzen möchte. Wichtig ist erst einmal, dass er groß genug ist, um das normale Tagesgeschäft und normale Abfragen abzudecken. Dabei sollte man ihn lieber großzügig dimensionieren, es ist sehr ärgerlich, wenn in lastintensiven Zeiten große Abfragen nicht durchgehen, weil die Before Images nicht mehr zur Verfügung stehen. Wenn große Abfragen durchgeführt werden, konsistente Exports mit größeren Datenmengen geplant sind oder das Flashback Query Feature genutzt werden soll, sollte man ihn generell großzügig setzen. Entweder stellt man ihn auf Autoextend, oder nutzt den Undo Advisor, wenn er unbedingt auf eine feste Größe gestellt werden soll.

Generell ist es sinnvoll, Oracle hier die Verwaltung von Tablespace und Undo Segmenten zu überlassen. Dazu gibt es das Automatic Undo Managment, das standardmäßig altiviert ist. Es ist zwar immer noch möglich, manuell Rollback Segmente zu verwalten, aber diese Möglichkeit gibt es aus historischen Gründen (Abwärtskompatibilität) und sie ist nur in sehr exotischen Fällen sinnvoll.

Ein anderer Eingriff in das Undo Management kann aber sinnvoll sein: Wenn man mit einem Real Application Cluster (RAC) arbeitet, kann man den Instanzen über den `UNDO_TABLESPACE` Parameter jeweils eigene Undo Tablespaces zuweisen. Damit vermeidet man die Notwendigkeit, Blöcke zwischen den Instanzen auszutauschen, was den Interconnect stark entlasten kann.

Ein weiterer Punkt ist der Wert für die `UNDO_RETENTION`. Damit kann man festlegen, wie lange die Before Images vorgehalten werden sollen und auch dieser Wert sollte sinnvoll gesetzt sein. Der Standard ist bei 900 Sekunden, nach etwa 15 Minuten wird die Datenbank die Before Images also

überschreiben. Das kann für große Abfragen zu kurz sein, wenn man Flashback nutzen will, ist es definitiv zu wenig. Es kann aber passieren, dass die Before Images auch überschrieben werden, wenn dieser Wert deutlich höher gesetzt wird, die Undo Retention ist nämlich nur eine Empfehlung. Wenn man möchte, kann man den Wert für den Undo Tablespace mit GUARANTEE erzwingen, das sollte aber nur in Ausnahmefällen geschehen. Wenn der Undo Tablespace dann nämlich voll ist, wartet die Datenbank einfach bis die Retention abgelaufen ist, bevor sie weitere Änderungen zulässt. Und das bedeutet auf gut deutsch: Die Anwendung steht und wartet dann auch.
Nicht zuletzt ist der Undo Tablespace auch aus IO Gesichtspunkten interessant. Hier werden ja für alle Änderungen die Before Images geschrieben, das heißt gerade bei großen Operationen auf vielen Datensätzen steigt auch hier das IO Aufkommen. Für DBAs ist der Undo Tablespace alleine deshalb schon immer einen Blick wert.

Sonstige Nutzung der Undo Segmente
Jetzt haben wir schon einige Dinge gesehen, für die der Undo Tablespace und die darin befindlichen Before Images genutzt werden. So spielt er die zentrale Rolle, wenn für Atomarität und Isolation aus dem ACID Modell, ermöglicht konsistentes Lesen ohne Sperren. Eine Transaktion muss ja auch nicht explizit zurückgerollt werden, sondern auch nach einem Sessionabbruch oder einem Instance Crash Recovery werden die nicht abgeschlossenen Transaktionen zurückgerollt, für all das wird der Undo Tablespace genutzt. Doch darüber hinaus gibt es noch sehr interessante weitere Möglichkeiten:
Nachdem in den Before Images die Zustände der Datensätze in der Vergangenheit gespeichert sind, kann man auch in der Zeit zurück reisen. So lassen sich Sessions in die Vergangenheit versetzen, man kann Abfragen gegen alte Datenbestände ausführen oder sogar Tabellen auf einen früheren Stand zurücksetzen. Das ist hilfreich, um zum Beispiel nach vermuteten Fehlern von Anwendern zu sehen, wie die Daten vorher ausgesehen haben und sie unter Umständen auf diesen Zustand zurückzusetzen. Im Gegensatz zu einem Restore und Recovery ist das sehr schnell und man kann so gezielt Objekte oder sogar Datensätze zurücksichern. Es ist nämlich auch möglich, nur bestimmte Zeilen zu diesem Zeitpunkt selektieren und in den aktuellen Stand übernehmen, was bei Fehlern in der Anwendungslogik, die nur wenige Zeilen betreffen, sehr interessant sein kann.
Eine andere denkbare Anwendung ist Auditing und nachvollziehen zu können, was wie geändert wurde. Will man da aber die Sicherheit haben, dass die Datensätze noch vorhanden sind, muss man mit Guarantee arbeiten, was ein Risiko für den Anwendungsbetrieb darstellt. In dem Fall kann es interessant sein, die Daten in eine andere Datenbank zu replizieren und dort die Undo Retention und Guarantee zu sezten.
Übrigens: Es gibt auch ein Flashback Database Feature in Oracle, mit dem sich eine gesamte Datenbank auf einen Stand zurücksetzen lässt. Hierbei wird aber nicht der Undo Tablespace genutzt, sondern eine andere Technologie, die Flashback Logs.

Kontaktadresse:
Thomas Klughardt
Dell Software
Im Mediapark 4e
D-50670 Köln

Telefon: +49 (0) 221-5777 4114
Fax: +49 (0) 221-5777 4114
E-Mail thomas.klughardt@software.dell.com
Internet: software.dell.com