Redo Logs – Informationen soweit der Logminer reicht

Thomas Klughardt
Dell Software
Köln

Schlüsselworte
Auditing, Nachvollziehen, Redo Logs, Persistenz,

Einleitung

Redo Logs sind eine der elementaren Komponenten einer Oracle Datenbank und wahrscheinlich die wichtigste. Alle Datenänderungen, die im Falle eines Absturzes wiederherstellbar (recoverable) sein sollen, werden hier protokolliert und damit alle wichtigen Daten. Gehen die Redo Logs verloren, dann gehen auch Daten verloren und kommen die Redo Logs beim Schreiben nicht hinterher, muss die Datenbank warten bis es weitergeht. Hier geht es zum einen darum, was bei der Konfiguration von Redo Logs zu beachten ist, aber auch darum, welche interessanten Informationen sich ihnen entlocken lassen.

Persistenz in einer Oracle Datenbank

Oracle Datenbanken werden dann genutzt, wenn geschäftskritische Anwendungen betrieben werden sollen, die im Fall von OLTP Anwendungen kurze Antwortzeiten liefern und bei OLAP Anwendungen große Analysen ermöglichen. Das ist nur dann möglich, wenn die Daten auf denen gearbeitet wird, im Speicher gehalten werden können. Dort finden die Zugriffe in Nanosekunden statt in Millisekunden bei klassischen Festplatten oder im Mikrosekundenbereich bei Solid State Disks (je nach Typ). Würde ein Vorgang mit vielen Random Access Zugriffen also rein von den IO Operationen im Speicher eine Sekunde dauern, würde die gleiche Zeit an IO Zugriffen auf einer klassischen Festplatte etwa elfeinhalb Tage dauern.

Dass möglichst viele Daten im Speicher gehalten werden sollen dürfte hier klar sein, man geht bei einer OLTP Anwendung davon aus, dass die Buffer Cache Hit Ratio nicht unter 95% liegen sollte. Das bedeutet nur 5% der Blöcke sollen von der Platte gelesen werden müssen und sogar das kann zu viel sein. Doch Speicherinhalte haben einen Nachteil: Sie sind flüchtig, im Falle eines Crashes der Instanz oder einem Serverabsturz sind diese Daten verloren. Und wenn eine Eigenschaft für Oracle Datenbanken noch wichtiger sein dürfte als kurze Antwortzeiten, dann die, dass abgeschlossene Transaktionen nicht verloren gehen.

Was also tun? Man protokolliert die Änderungen auf einer Festplatte und schreibt sequentiell. Dadurch vermeidet man das ständige, langsame Neupositionieren der Schreib-/Leseköpfe, speichert diese Änderungen aber trotzdem persistent auf der Platte. Diese Transaktionsprotokolle heißen bei Oracle Datenbanken Redo Logs. Die Redologs sind die wichtigsten Dateien in einer Oracle Datenbank. Hat man lückenlos alle Redolog-Dateien zur Verfügung, kann man die Datenbank von der Erstellung auf jeden beliebigen Stand bringen, ein Verlust der Redologs bedeutet so gut wie immer Datenverlust, wenn dazu ein Instancecrash auftritt. Sollten also im laufenden Betrieb aus irgendeinem Grund Redologs verloren gehen, sollte immer sofort eine Vollsicherung erfolgen, notfalls auch offline.

Performance in einer Oracle Datenbank

Trotz des sequentiellen Schreibens sind Festplattenzugriffe relativ langsam und so können die Redo Logs leicht zum Flaschenhast in der Datenbank werden. Ist die Datenbank nicht in der Lage, die Änderungen schnell genug in den Redo Logs zu protokollieren, passiert erst einmal nichts weiter. Sie wartet, bis die Änderungen protokolliert sind. Erst dann ist eine Transaktion abgeschlossen.

Das bedeutet, man muss sich beim Anlegen der Redo Logs Gedanken machen. Es gibt mindestens zwei Redo Log Gruppen mit meist mehreren Membern, die zyklisch der Reihe nach beschrieben werden. Ist eine Redo Log Gruppe voll, wird die nächste beschrieben, dort werden wie in einem, Ringpuffer die Daten überschrieben. Um die Daten lückenlos wiederherstellen zu können, müssen alte Redo Logs archiviert werden.

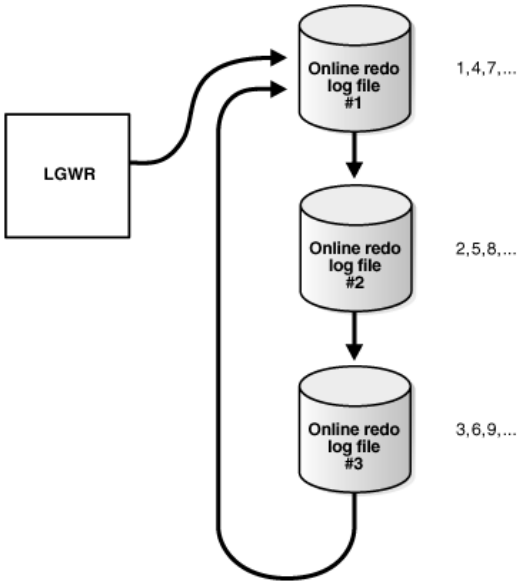

Abb. 1: Wiederverwendung der Oracle Redo Logs (Quelle: Oracle 12c Dokumentation)

Bei Oracle Datenbanken gibt es dafür den ARCHIVELOG Modus, damit wird sichergestellt, dass alte Redo Logs archiviert werden, bevor man beginnt, neue zu schreiben. Diese archivierten Redo Logs sind wichtig, weil man damit bei Datenverlust von der letzten Vollsicherung die Daten bis zum aktuellen Stand vorrollen kann, aber natürlich gibt es auch hier einen Flaschenhals. Kommt die Archivierung nicht hinterher, dann muss die Datenbank warten.

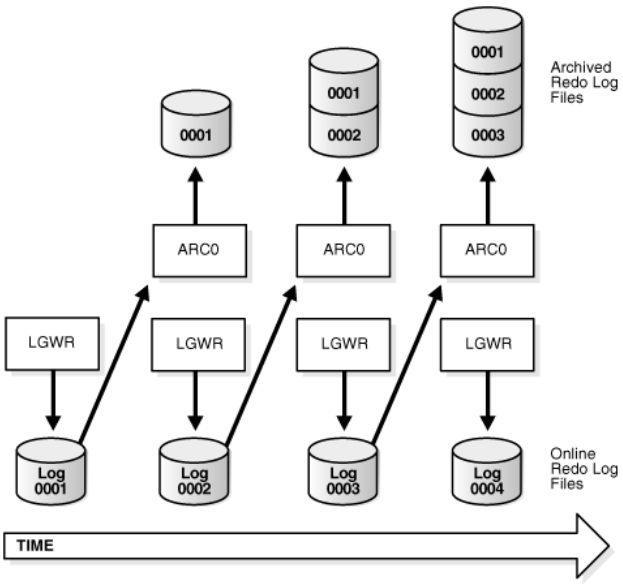

Abb. 2: Redo Logs im Archivelog Modus (Quelle: Oracle 12c Dokumentation)

Bei der Planung der Redo Logs spielt auch die Größe eine große Rolle. Oracle führt in mehr oder weniger regelmäßigen Abständen sogenannte Checkpoints durch, bei denen die Änderungen, die erfolgt sind, aus dem Cache in die Datendateien geschrieben werden. Diese Checkpoints finden zum Beispiel statt, nachdem es einen Redo Log Switch gegeben hat. Finden sie zu häufig statt, dann kann in das nächste Redo Log unter Umständen noch nicht überschrieben werden, weil noch Daten auf die Platte geschrieben werden müssen, die in diese Log stehen. Finden Checkpoints zu selten statt, dauert das herunterschreiben dieser Daten länger und im Falle eines Instance Crashes dauert es natürlich länger, bis die Daten recovert sind und die Datenbank wieder zur Verfügung steht. Diese Abwägung ist so wichtig, dass es in der Datenbank eine Sicht (`V$INSTANCE_RECOVERY`) gibt, über die man sich eine Empfehlung für die midndest Größe für die Redo Logs anzeigen lassen kann. Allerdings sollte man dann auch bedenken, eine vernünftige Zielvorgabe für eine Recovery Zeit zu setzen (`FAST_START_MTTR_TARGET`).

Es gibt also einige Dinge, die beim Konfigurieren der Redo Logs beachtet werden muss, damit die Datenbank durch diesen absolut notwendigen Persistenz Mechanismus nicht ausgebremst wird.

Informationen aus den Redo Logs

Bis jetzt hat es sich so angehört, als seien die Redo Logs ein lästiges aber notwendiges Übel in der Datenbank, doch sie sind wesentlich mehr als das. Wie schon erwähnt kann man, wenn man die Redo Logs lückenlos zur Verfügung hat, die Datenbank auf jeden beliebigen Stand bringen, man nennt das auch vorwärts rollen (oder neudeutsch roll forward). Genau diese Möglichkeit wird bei einem Recovery ja benötigt. Jede Änderung, die irgendwann durchgeführt wurde, ist also in den Redo Logs verzeichnet. Dazu ist aber auch notiert, zu welcher Transaktion sie gehört, inklusive der System Change Number, einer eindeutigen fortlaufenden Nummer, von der jeder Transaktion genau eine zugeordnet ist. Darüber hinaus ist verzeichnet, wer diese Änderung durchgeführt hat, wann sie passiert ist und sogar wie man sie rückgängig machen kann. Wurde ein Datensatz gelöscht, sieht man hier die entsprechenden Daten, die vorher vorhanden waren. Wurde ein Datensatz aktualisiert, dann tauchen hier die Werte auf, die er vorher gehabt hat. Das macht die Redo Logs für viele Anwendungen zu Goldgruben an Informationen.

Nachvollziehbarkeit und Auditing

Oft gibt es rechtliche Gründe oder betriebliche Vorgaben, die es nötig machen, Änderungen zu protokollieren. Dafür gibt es von Oracle Auditing Mechanismen, doch auch die Redologs sind in der Lage, die nötigen Informationen zu liefern. Sie enthalten die Informationen, wer wann was geändert hat und sogar, wie die Werte vor der Änderung waren (Before Image). Das hört sich sehr gut an, allerdings hat die Sache einen kleinen Haken: Man kann nur Änderungen sehen, die in den Redo Logs protokolliert sind. Theoretisch sollten das natürlich alle Änderungen, die an Nutzdaten anfallen sollen, sein. Oracle bietet aber die Möglichkeit, Direct Loads durchzuführen. Das sind INSERT Statements mit einem Append Hint, der dafür sorgt, dass diese INSERTs nicht in den Redo Logs auftauchen. Gedacht ist das, um schnell Staging Tabellen befüllen zu können, aber diese Änderungen sind nicht recoverbar. Will man diese Möglichkeit unterbinden, dann gibt es die Möglichkeit, für die Datenbank oder einzelne Tablespaces die FORCE_LOGGING Option zu aktivieren und damit werden dann auch diese Operationen protokolliert und Nachvollziehbar.

Über den Logminer (DBMS_Logminer) können die Informationen aus den Redo Logs für Menschen lesbar gemacht und ausgegeben werden. Allerdings ist die Ausgabe immer noch recht kryptisch, deshalb empfiehlt es sich hier, mit Tools zu arbeiten, die die Daten zum Beispiel nach Excel ausgeben können. Zum Beispiel Toad for Oracle hat einen Logminer Assistenten, der die Einträge anzeigt und exportieren kann.

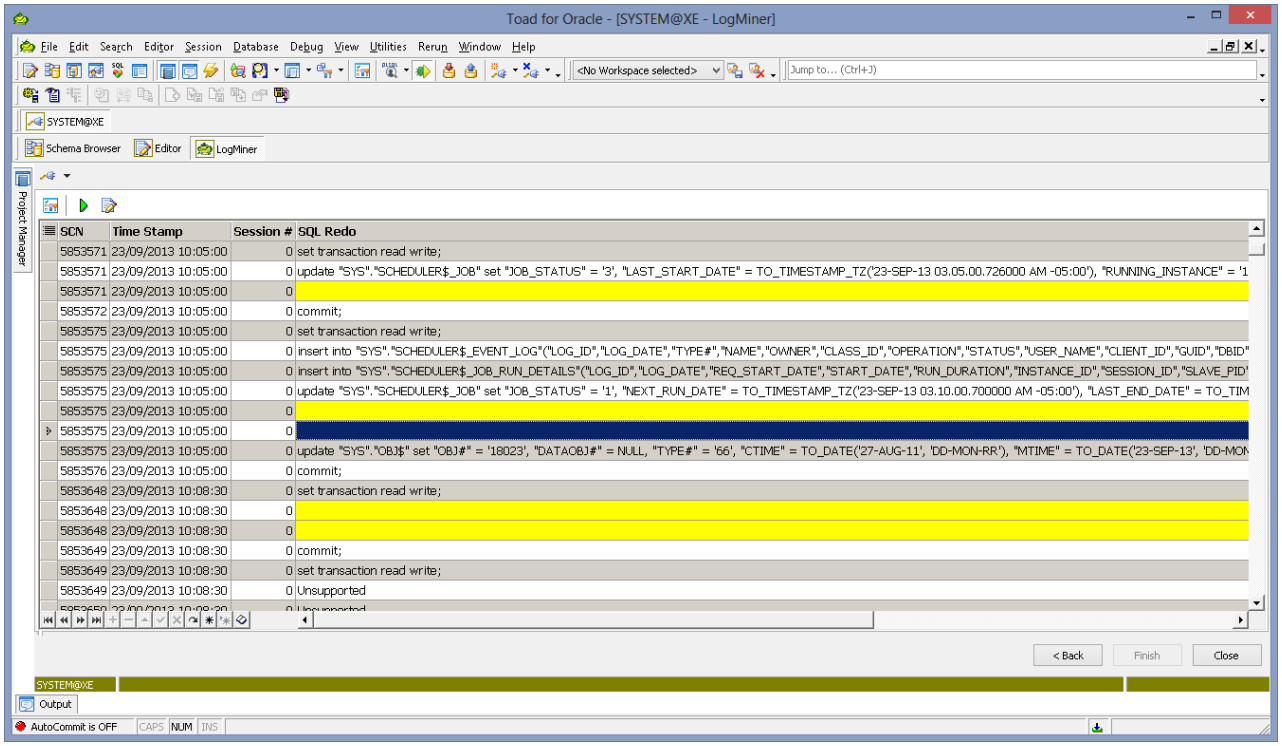

Abb. 3: Logminer Interface in Toad

Replikation

Nachdem die Redo Logs wie gesagt alle nötigen Daten enthalten, sind sie auch für Replikation interessant. So ist es damit möglich, ohne viel Last in der Datenbank zu machen, alle Änderungen auf andere Systeme zu transportieren und dort Replikate zu pflegen. Dabei gibt es physische Replikationslösungen wie den Oracle Dataguard, der binär gleiche Replikate verwaltet, die im Falle eines Ausfalles übernehmen können. Der Nachteil hier ist, dass an der Zieldatenbank nichts geändert werden darf, sie muss ja binär identisch bleiben.
Etwas komplizierter sind logische Replikationslösungen wie Shareplex, Oracle GoldenGate oder DBVisit Replicate, die aus den Redologs tatsächlich SQL Statements erzeugen und so auf der Zielseite auch in binär unterschiedliche Datenbanken schreiben können. Dort kann auf beide Datenbanken lesend und schreibend zugegriffen werden, sie können andere Versionen und Architekturen haben und zum Beispiel zusätzliche Indizes für große Abfragen enthalten. Zusätzlich sind hier interessante Szenarien wie Migrationen fast ohne Auszeit oder entfernte Multi Master Datenbanken möglich.

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