Jeder, der Datenbanken betreut und administriert, macht sich Gedanken über Sicherung und Wiederherstellung seiner Daten. Diese Methoden decken aber oft nur physikalische Fehler ab. Probleme, die von Nutzern oder Programmfehlern durch versehentliches Ändern oder Löschen verursacht werden, sind damit nur schwer und mit großem Aufwand zu beheben. Oracle bietet mit verschiedenen Methoden des Flashbacks einige sehr nützliche Funktionen, um derartige Fehler recht effizient zu behandeln. Einige dieser Varianten sind in diesem Artikel näher beleuchtet.

Flashback: Recovery ohne Recovery

Marco Mischke, Robotron Datenbank-Software GmbH

Jeder DBA hat das sicher schon erlebt: Alle Datenbanken im Unternehmen laufen seit Monaten zuverlässig. Die ausgefeilte Backup-Strategie hat sich etabliert und in verschiedenen Tests bewiesen, dass Fehler in Kontroll- oder Tablespace-Dateien zügig zu beheben sind. Dann klingelt das Telefon und am anderen Ende erzählt ein Entwickler aufgeregt von plötzlich verschwundenen Daten nach einem Durchlauf einer eigentlich gut getesteten Prozedur zur Reorganisation der Stammdaten. Passiert ist das vor etwa einer Stunde, der Entwickler wollte zuerst selbst das Problem lösen. Natürlich betrafen die Aktionen das Produktiv-System, das allen anderen Anwendungen im Unternehmen als zentrale Datenbasis dient. Daher kann das System nicht einfach mal heruntergefahren werden und auch Bewegungsdaten wie Bestellungen etc. dürfen nicht verloren gehen.

Man könnte die Datenbank einfach aus dem Backup wiederherstellen und per unvollständigem Recovery zum Zeitpunkt kurz vor dem Fehler wieder online bringen – dann wären jedoch alle neueren Änderungen verloren. Auf einem separaten System ein Tablespace Point in Time Recovery durchzuführen, würde zu viel Zeit benötigen.

Jetzt kommt Flashback ins Spiel. Die Oracle-internen Methoden zum konsistenten Lesen sind seit Version 9 auch für Abfragen in die Vergangenheit benutzbar. Dabei wurden mit jeder Version weitere Features eingeführt, die den Administratoren das Beheben solcher inhaltlichen Fehler einfacher machen (können). Dabei dient „Flashback“ als Oberbegriff und vereint verschiedene Technologien, um an Daten und Zustände aus der Vergangenheit zu gelangen, ohne ein Restore und Recovery durchführen zu müssen. Einige davon stehen bereits in der Standard Edition zur Verfügung.

Flashback Query

Oracle loggt alle Änderungen von Transaktionen im Undo Tablespace, um Rollbacks sowie konsistentes Lesen anderer Sessions zu ermöglichen. Per Default hebt Oracle die Änderungen der letzten fünfzehn Minuten auf; dieser Zeitraum lässt sich über den Parameter „undo_retention“ beeinflussen. Da die Dateien des Undo Tablespace sich normalerweise automatisch erweitern, werden auch nur genau diese fünfzehn Minuten aufgehoben. Bei Dateien mit fester Größe versucht die Datenbank, den gesamten zur Verfügung stehenden Speicherplatz zu benutzen, und ignoriert die Einstellungen des Parameters „undo_retention“. Die View „V$UNDOSTAT“ zeigt in der Spalte „TUNED_UNDORETENTION“ den aktuellen Wert (siehe Listing 1).

Um Daten konsistent lesen zu können, merkt sich die Datenbank die System Change Number (SCN) beim Start der Abfrage beziehungsweise der Transaktion. Werden Datensätze danach von anderen Transaktionen verändert, wird der vorige Zustand des entsprechenden Blocks im Undo gespeichert. Liest die laufende Abfrage nun einen Block mit einer höheren SCN als der zum Start gespeicherten, dann wird der Block so lange mit den Informationen aus dem Undo zurückgerollt, bis eine SCN vor dem Start der Abfrage erreicht ist.

Bei Flashback Query kommt genau dieser Mechanismus zum Einsatz. Nur wird eben nicht die SCN zum Start der Abfrage gespeichert, sondern diejenige, die dem gewünschten Zeitpunkt am nächsten kommt. Sind historische Daten nicht mehr aus dem Undo zu rekonstruieren, dann endet die Abfrage in einem „ORA-01555: Snapshot too old“.

Die Datenbank kann sich also quasi an alte Daten erinnern. Wie lange die Erinnerung anhält, ist in der Einstellung von „undo_retention“ definiert. Dieses Erinnerungsfenster wandert mit

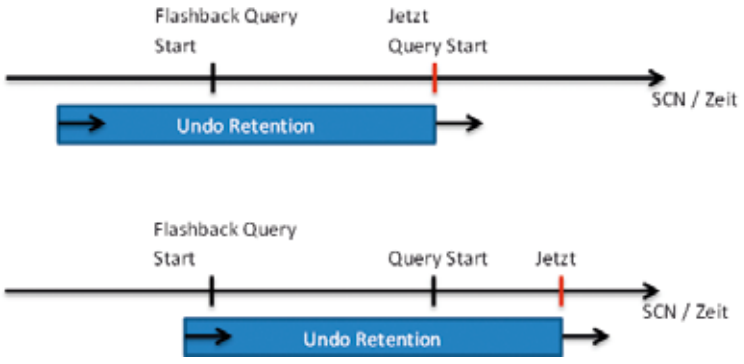

Abbildung 1: Zeitliche Zusammenhänge von Flashback Query und Undo

```
SQL> select * from (
  2  select TUNED_UNDORETENTION
  3  from   V$UNDOSTAT
  4  order by END_TIME desc
  5  )
  6  where ROWNUM = 1;

TUNED_UNDORETENTION
-------------------
              36356
```

Listing 1

```
$ expdp system/Oracle-1 dumpfile=scott_flashback.dmpdp  \
        logfile=scott_flashback.expdp.log \
        directory=data_pump_dir \
        flashback_time='2012-10-01-11:00:00' schemas=scott
```

Listing 2

```
$ exp system/Oracle-1 file=/tmp/scott_flashback.dmp \
      log=/tmp/scott_flashback.exp.log \
      flashback_time='2012-10-01-11:00:00' owner=scott
```

Listing 3

```
SQL> create table emp_old
  2  as
  3  select *
  4  from emp as of timestamp
  5  to_timestamp(,2012-10-01 11:00:00',
  6  ,yyyy-mm-yy hh24:mi:ss');
```

Listing 4

```
SQL> select * from emp_old
  2  minus
  3  select * from emp;
```

Listing 5

```
SQL> select * from emp as of timestamp
  2               to_timestamp(,2012-10-01 11:00:00',
  3               ,yyyy-mm-dd hh24:mi:ss')
  4  minus
  5  select * from emp;
```

Listing 6

```
SQL> insert into emp
  2  select * from emp as of timestamp
  3               to_timestamp('2012-10-01 11:00:00',
  4               'yyyy-mm-dd hh24:mi:ss')
  5  minus
  6  select * from emp;

SQL> commit;
```

Listing 7

der Zeit mit, am Anfang kommen neue Erinnerungen (Änderungen) dazu und überschreiben dabei am Ende die ältesten. Abbildung 1 verdeutlicht die zeitlichen Zusammenhänge.

Wer den Undo Tablespaces seiner Datenbanken ausreichend Platz gibt, um etwa zehn bis zwölf Stunden an Transaktionen unterzubringen, kann sich an die Wiederherstellung der verlorenen Daten für einen kompletten Werktag machen.

Flashback Export mit der Standard Edition

Damit die Daten aus dem Undo Tablespace nicht überschrieben werden und dann verloren wären, erzeugt man zuallererst einen Export des betroffenen Schemas mit dem Stand vor den fraglichen Änderungen. Dabei ist zu beachten, dass das Zeitformat immer in der Form „YYYY-MM-DD-HH24:MI:SS“ angegeben wird (siehe Listing 2). Selbstverständlich kann man auch das herkömmliche Export Utility dafür benutzen (siehe Listing 3). Der Parameter „flashback_time“ kann dabei mit allen anderen Parametern kombiniert werden, um etwa statt des gesamten Schemas nur einige Tabellen zu exportieren.

Nachdem man sich mit dem Export etwas Sicherheit verschafft hat, beginnt der Versuch, die Daten zu retten. Man kann dazu beispielsweise eine Kopie der Tabelle anlegen, die den Stand aus der Vergangenheit hat. Dazu dient die Klausel „as of timestamp“ im „FROM” (siehe Listing 4).

Nun können die Inhalte beider Tabellen verglichen werden, um die gelöschten oder geänderten Datensätze zu ermitteln (siehe Listing 5). Da die „as of timestamp“-Klausel in allen Arten von SQL verwendet werden darf, lassen sich die Daten aus der Vergangenheit auch direkt mit den aktuellen Datenständen vergleichen (siehe Listing 6).

Im einfachsten Fall können nun versehentlich gelöschte Datensätze wieder in die Tabelle eingefügt werden (siehe Listing 7). Nach dem gleichen Prinzip lassen sich auch Datensätze retten, wenn nur Änderungen vorgenommen und nichts gelöscht wurde, hier am Beispiel des Gehalts der Mitarbeiter (siehe Listing 8).

Diese Beispiele zeigen eine Richtung auf, mit der versehentlich geänderte Daten wiederhergestellt werden können. Der persönlichen Kreativität bei der Abfrage-Formulierung sind hier kaum Grenzen gesetzt.

Flashback Drop mit der Standard Edition

Flashback Query deckt nur DML-Operationen ab, also inhaltliche Änderungen. Hat der Entwickler allerdings versehentlich eine oder mehrere Tabellen gelöscht, wird das natürlich nicht im Undo geloggt, da es sich um eine DDL-Operation handelt. Oracle bietet hier die Recycle Bin-Funktionalität (Papierkorb) an, die per Standard über den Parameter „recyclebin“ aktiviert ist. Daher wird ein Segment beim DROP-Befehl nur als frei markiert, aber noch nicht gelöscht. Oracle überschreibt solche Segmente erst, wenn kein anderer freier Speicherplatz im Tablespace mehr verfügbar ist. So lange das nicht passiert, lässt sich das Segment wiederherstellen; diese Funktion wird als „Flashback Drop“ bezeichnet. Um zu sehen, welche Segmente wiederherstellbar sind, dienen der SQL*Plus-Befehl „show recyclebin“ oder die Views „USER_ RECYCLEBIN“, „ALL_ RECYCLEBIN“ und „DBA_RECYCLEBIN“ (siehe Listing 9).

Es ist zu sehen, dass der SQL*Plus-Befehl lediglich die gelöschten Tabellen auflistet, die View hingegen alle gelöschten Segmente enthält. Die von dem Entwickler versehentlich gelöschte Tabelle kann also offenbar wiederhergestellt werden (siehe Listing 10).

Die Datenbank hat also nicht nur die Tabelle sondern auch die zugehörigen Indizes wiederhergestellt. Allerdings hat nur die Tabelle ihren originalen Namen zurückerhalten. Die Indizes haben nach wie vor die Bezeichnung aus dem Recycle Bin (siehe Listing 11). Diese müssen also im Nachgang noch angepasst werden. Es ist daher wichtig, die Zuordnung der Recycle-Bin-Namen zu den originalen Indexnamen vor dem Flashback zu notieren oder direkt ein Skript zum Umbenennen zu erzeugen. Listing 12 zeigt, wie das Umbenennen erfolgt.

Es ist wichtig, den Recycle-Bin-Namen in Anführungszeichen zu setzen, da dieser Sonderzeichen enthält. Einzig Fremdschlüssel-Constraints kön-

```
SQL> update emp e_live
  2  set sal = (select sal
  3             from emp as of timestamp
  4                      to_timestamp('2012-10-01 11:00:00',
  5                      'yyyy-mm-dd hh24:mi:ss') e_orig
  6             where e_orig.empno = e_live.empno
  7            )
  8  ;

SQL> commit;
```

Listing 8

```
SQL> show recyclebin
ORIGINAL NAME RECYCLEBIN NAME                OBJ. TYPE  DROP TIME
------------- ------------------------------ ---------- -------------------
EMP           BIN$yxLVQQOqC6rgQBAKtBUKlA==$0 TABLE      2012-10-02:13:29:19

SQL> select ORIGINAL_NAME, OBJECT_NAME, OPERATION, DROPTIME
  2  from USER_RECYCLEBIN;

ORIGINAL_NAME OBJECT_NAME                    OPERATION DROPTIME
------------- ------------------------------ --------- -------------------
EMP           BIN$yxLVQQOqC6rgQBAKtBUKlA==$0 DROP      2012-10-02:13:29:19
PK_EMP        BIN$yxLVQQOpC6rgQBAKtBUKlA==$0 DROP      2012-10-02:13:29:19
IX_ENAME      BIN$yxLVQQOoC6rgQBAKtBUKlA==$0 DROP      2012-10-02:13:29:19
IX_MGR        BIN$yxLVQQOnC6rgQBAKtBUKlA==$0 DROP      2012-10-02:13:29:19
```

Listing 9

```
SQL> flashback table emp to before drop;

Flashback abgeschlossen.

SQL> select ORIGINAL_NAME, OBJECT_NAME, OPERATION, DROPTIME
  2  from USER_RECYCLEBIN;

Es wurden keine Zeilen ausgewählt
```

Listing 10

```
SQL> select index_name from user_indexes
  2  where table_name='EMP';

INDEX_NAME
------------------------------
BIN$yxLVQQOoC6rgQBAKtBUKlA==$0
BIN$yxLVQQOnC6rgQBAKtBUKlA==$0
BIN$yxLVQQOpC6rgQBAKtBUKlA==$0
```

Listing 11

```
SQL> alter index "BIN$yxLVQQOoC6rgQBAKtBUKlA==$0“ rename to IX_ENAME;
```

Listing 12

nen nicht wiederhergestellt werden. Diese sind manuell zu prüfen und bei Bedarf neu anzulegen.

Flashback Version Query mit der Enterprise Edition

Es muss aber nicht unbedingt der Entwickler sein, der ungewollte Änderungen ausgelöst hat. Wenn beispielsweise die Personalabteilung gerade die beschlossene Gehaltsanpassung bearbeitet hat und bei der Prüfung feststellt, dass einige Mitarbeiter ein falsches Gehalt zugeordnet bekommen haben, will der Personalchef wissen, wann und welche Änderungen am Gehalt eines bestimmten Mitarbeiters vorgenommen wurden. Hier kommt „Flashback Version Query“ ins Spiel, mit der sich der zeitliche Verlauf eines Datensatzes ermitteln lässt. Für den Angestellten „MILLER“ wird so die Historie der letzten Stunde ermittelt (siehe Listing 13).

Hieraus sieht man, dass vor der geplanten Erhöhung um 500 Euro um 14:35 Uhr jemand irrtümlich das Gehalt verringert hat, und zwar um 14:29 Uhr. Diese Art von Abfragen ermittelt genauso Änderungen, die durch Einfügen und Löschen entstehen. Listing 14 zeigt die kurze, aber ereignisreiche Karriere von Herrn Foo mit Neueinstellung, der kurzfristigen Gehaltserhöhung und der anschließenden betriebsbedingten Kündigung mit Flashback Version Query.

```
SQL> select
  2    versions_starttime, versions_endtime,
  3    versions_operation, versions_xid,
  4    sal
  5  from
  6    emp versions between timestamp
  7        systimestamp - interval '1' hour and
  8        systimestamp
  9  where
 10    ename = 'MILLER';

VERSIONS_STARTTIME VERSIONS_ENDTIME   V VERSIONS_XID           SAL
------------------ ------------------ - ---------------- ---------
02.10.12 14:35:41                     U 07000B00F7480000      1800
02.10.12 14:29:29  02.10.12 14:35:41  U       1300
                   02.10.12 14:29:29       1504,91
```

Listing 13

```
SQL> select
  2    versions_starttime, versions_endtime,
  3    versions_operation,
  4    sal
  5  from
  6    emp versions between timestamp
  7        systimestamp - interval '1' hour and
  8        systimestamp
  9  where
 10    ename = 'FOO';

VERSIONS_STARTTIME VERSIONS_ENDTIME   V        SAL
------------------ ------------------ - ----------
02.10.12 15:30:05                     D       5046
02.10.12 14:57:08  02.10.12 15:30:05  U       5046
02.10.12 14:45:05  02.10.12 14:57:08  I       4711
```

Listing 14

Weitere Möglichkeiten

Es gibt noch eine Reihe weiterer Flashback-Features, die hier kurz aufgezählt und erläutert sind. Alle sind Teil der Enterprise Edition.

„Flashback Transaction Query“ ermöglicht es, Informationen zu einer bestimmten Transaktion zu erhalten. Die View „FLASHBACK_TRANSACTION_QUERY“ listet alle Änderungen auf, die durch eine bestimmte Transaktion durchgeführt wurden.

„Flashback Table“ setzt den Inhalt einer ganzen Tabelle auf einen Stand in der Vergangenheit zurück. Dabei können mehrere Tabellen in einem Schritt auf eine frühere Stufe gebracht werden, was insbesondere bei existierenden Fremdschlüssel-Beziehungen hilfreich ist.

„Flashback Database“ setzt die komplette Datenbank auf einen früheren Stand zurück. Dafür müssen gesonderte Flashback Logs geschrieben werden. Die Vorhaltezeit dieser Logs wird über den Parameter „db_flashback_retention_target“ eingestellt.

„Flashback Transaction“ dient zum Ausradieren einzelner Änderungen an Datensätzen. Hier werden Änderungen und deren Abhängigkeiten in einer Art „Baum“ aufgelistet. Es kann dann beim Rückgängigmachen der Änderung entschieden werden, wie man diese Abhängigkeiten behandelt.

Fazit

Schon die Standard Edition der Datenbank bietet den Administratoren hilfreiche Werkzeuge, um versehentliche oder ungewollte inhaltliche Änderungen an Tabellen rückgängig machen zu können, ohne dabei die Verfügbarkeit der Datenbank zu beeinflussen. Die hier gezeigten Möglichkeiten sind nur die Spitze des Eisbergs, sie geben aber einen ersten Eindruck davon, wie mächtig die Flashback-Funktionen sind.

Marco Mischke
marco.mischke@robotron.de