

4.4 Einsatz der Flashback-Technologien

4.4.1 Einsatz von Flashback Query

Flashback Query lässt sich mit einigen ganz einfachen Mitteln einsetzen. Dabei wird in einer Abfrage einfach nur angegeben, auf welchen Zeitpunkt man zurückgreifen möchte. Ein Beispiel: Es ist Montag, 02.03.2004, kurz vor der Mittagspause. Herr Müller aus der Personalabteilung ruft an. Er berichtet, er habe ganz sicher gar nichts gemacht; dennoch sei – wie er feststellen musste – die Abteilung mit der Abteilungsnummer 50 plötzlich und gänzlich ohne sein Zutun aus dem Datenbestand einer Oracle-Datenbank verschwunden. Werfen wir einen Blick in die betreffende Datenbank. Hierzu kann der grafische Oracle Enterprise Manager oder aber – wie im folgenden Beispiel – SQL*Plus genutzt werden. Wir setzen also ein Select Statement mit SQL*Plus auf die Department-Tabelle [dept], in der die Datensätze der Abteilungen gespeichert sind, ab:

```
SQL> SELECT * FROM dept;
```

ABTNR	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Listing 4.20: Ein Datensatz mit der Abteilungsnummer 50 ist im aktuellen Abfrageergebnis nicht vorhanden

Es zeigt sich: Herr Müller hat Recht. Hier gibt es aktuell wirklich keine Abteilung mit der Nummer 50. Betreiben wir ein wenig Forschung: Wie sah der Datenbestand denn am 02.03.2004 um 07:00 Uhr in der Frühe aus?

```
SQL> SELECT * FROM dept
2> AS OF TIMESTAMP
3> to_timestamp('02-03-2004 07:00:00','dd-mm-yyyy hh24:mi:ss');
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	IT	FRANKFURT

Listing 4.21: Die historische Abfrage zeigt, dass der Datensatz um 07:00 Uhr noch vorhanden war

Aha! Da sind wir doch schon einen Schritt weiter: Morgens um 7 Uhr war die Welt noch in Ordnung.

Das oben stehende Beispiel zeigt, wie die Syntax von Flashback Query eingesetzt werden kann:

1. Abfragetext in SQL formulieren und
2. zusätzlich den historischen Zeitpunkt, zu dem der Datenbestand angezeigt werden soll, mit der Syntax *AS OF TIMESTAMP* angeben.

Möchte man jetzt herausfinden, wie sich der Datensatz über den Tag verändert hat oder aber wann er genau gelöscht wurde, so kann man mit der Syntax *VERSIONS BETWEEN* alle Versionen innerhalb des angegebenen Zeitraumes inklusive sämtlicher Änderungen ausgeben lassen:

```
SQL> SELECT deptno, dname, loc,
2>          versions_operation, versions_xid, versions_starttime
3> FROM    scott.dept
4> VERSIONS BETWEEN timestamp minvalue and maxvalue
5> ORDER BY deptno, versions_starttime
```

DEPTNO	DNAME	LOC	V	VERSIONS_XID	VERSIONS_STARTTIME
10	ACCOUNTING	NEW YORK			
20	RESEARCH	DALLAS			
30	SALES	CHICAGO			
40	OPERATIONS	BOSTON			
50	IT	FRANKFURT			
50	MARKETING	FRANKFURT	U	1E000A0008000000	02.03.04 09:30:02
50		FRANKFURT	U	1E000B0008000000	02.03.04 09:31:15
50		FRANKFURT	D	1E000C0008000000	02.03.04 09:32:20

8 Zeilen ausgewählt

Listing 4.22: Anzeige der Datensatzhistorie durch die Klausel *versions between*

Und siehe da, der Datensatz wurde zunächst um 09:30 Uhr und 2 Sekunden geändert. In Zeile 5 stand in der Spalte [dname] noch der Wert »IT«. Zu diesem Zeitpunkt wurde der Name der Abteilung von »IT« auf »Marketing« geändert. Erkennbar ist dies in Zeile 6. Hier steht in der Spalte [versions_operation] ein U wie Update und als neuer Wert der Spalte [dname] der Begriff »MARKETING«, während in Zeile 5 noch »IT« zu finden war. Um 09:31 und 15 Sekunden erfolgte das nächste Update: Wieder enthält die Spalte [versions_operation] ein U für Update, der Wert der Spalte [dname] ist nun leer. Die letzte Änderung stammt dann von 09:32 Uhr und 20 Sekunden. Hier wurde der Satz gelöscht. Erkennbar ist dies wieder in der Spalte [versions_operations] am Flag D, das für Delete steht.

In der letzten Abfrage werden mehrere Pseudo Columns angesprochen: [versions_operation], [versions_xid] und [versions_starttime]. Diese Spalten sind nicht physisch in der Tabelle in Form von Spalteninhalten gespeichert. Vielmehr erhalten wir durch sie zusätzliche Informationen, die aus dem internen Data Dictionary der Datenbank stammen. Die Pseudo-Spalte [versions_operation] gibt Auskunft über die Operation, die ausgeführt wurde: I für Insert, U für Update, D für Delete. Unter [versions_xid] finden wir die interne Transaktionsnummer innerhalb der Datenbank. Diese werden wir gleich noch benötigen. Unter [versions_starttime] ist der

Zeitpunkt zu verstehen, ab dem diese Änderung gültig wurde. Dabei handelt es sich um den Zeitpunkt, zu dem die Änderung mit Commit bestätigt wurde.

In unserem Beispiel kann durch Nutzung der zuvor ermittelten Transaktions-ID aus der Spalte [versions_xid] das Undo-Statement gefunden werden, um die Veränderungen an der Tabelle [dept] wieder rückgängig zu machen. Über die Pseudo-Spalte [version_xid] in der oben genannten Flashback Query konnten wir die Transaktions-ID ermitteln. Die Datenbank-View flashback_transaction_query liefert Informationen über das Undo-SQL-Statement, das zur Transaktion mit der ID '1E000C0008000000' gehört:

```
SQL> SELECT logon_user, table_name,
           table_owner, undo_sql
        FROM flashback_transaction_query
        WHERE table_owner='SCOTT'
           -- xid = Transaktions-ID
        AND   xid= '1E000C0008000000';

LOGON_USER  TABLE_NAME  TABLE_OWNER  UNDO_SQL
-----
SMUELLER    DEPT         SCOTT
insert into "SCOTT"."DEPT"("DEPTNO","DNAME","LOC") values
           ('50','UUuuups','Frankfurt');
```

Listing 4.23: Anzeige der Undo-Informationen der View flashback_transaction_query

Ein kleiner Ausflug in Datenbank-Interna: Durch Abfrage der View flashback_transaction_query erhält man die Ergebnisse der Abfrage, die über den Logon-User, die betreffende Tabelle und deren Besitzer sowie das Statement zum Rückgängigmachen der Transaktion geben. Wer die Augen nicht vor der Spalte [logon_user] verschließt, der wird bemerken, dass ein als SMUELLER angemeldeter Benutzer die Transaktion zum Löschen des Datensatzes durchgeführt hatte. Das klingt doch sehr nach Herrn Müller aus der Personalabteilung. Aber das behalten wir schmunzelnd einfach für uns!

Betrachtet man das Insert Statement aus der Spalte [undo_sql], so fällt auf, dass hier nur das Delete der Zeile, das durch die letzte Transaktion abgesetzt wurde, rückgängig gemacht wird. Möchte man tatsächlich auf den ursprünglichen Stand von heute früh zurück, so muss man einfach alle undo_sql-Statements in umgekehrter zeitlicher Reihenfolge (also immer das Letzte zuerst), verwenden, um die Änderungen schrittweise wieder rückgängig zu machen.

```
SQL> SELECT  undo_sql
2 > FROM    flashback_transaction_query
3 > WHERE   table_owner = 'SCOTT'
4 > AND    table_name = 'DEPT'
5 > ORDER BY commit_timestamp desc

UNDO_SQL
-----
insert into "SCOTT"."DEPT"("DEPTNO","DNAME","LOC") values
           ('50','UUuuups','Frankfurt');
```

```
update "SCOTT"."DEPT" set "DNAME" = 'MARKETING' where ROWID = 'AAALy6AAEAAAAA0AAA';  
update "SCOTT"."DEPT" set "DNAME" = 'IT' where ROWID = 'AAALy6AAEAAAAA0AAA';
```

Listing 4.24: Undo-Informationen in zeitlicher Abfolge

Sehr viel leichter lassen sich solche Änderungen mit Oracle DB Control vornehmen. Hier kann man sich – gestützt durch ein grafisches Werkzeug – durch die Masken des Flashback hangeln und mit einem Mausklick einen Datensatz wiederherstellen. Auch wenn das Vorgehen mit dem DBControl sehr viel bequemer ist, sollte man doch die Hintergründe verstanden haben. In schweren Problemfällen kann dies das Finden einer Lösung sehr viel einfacher machen.

Im Vergleich zur *Flashback Table*-Technologie, die im nächsten Abschnitt genauer beschrieben wird, stellt diese Technik eine Alternative bei versehentlichen oder fehlerhaften Änderungen an einzelnen Tabellendaten dar. Neben dem Zeitpunkt via »AS OF TIMESTAMP« oder »VERSIONS BETWEEN« kann auch die Oracle System Change Number verwendet werden. Diese ist sehr viel genauer als ein Timestamp.

Die System Change Number (auch als SCN bezeichnet) ist ein interner Transaktionszähler und vergleichbar einer Auftragsnummer in einem Auftragsbearbeitungssystem. Bei jeder Bestätigung einer Datenänderung durch ein *Commit* einer Transaktion wird intern in der Datenbank eine eindeutige System Change Number hochgezählt. Diese wird in den Transaktionsprotokollen – unter Oracle sind das die »Redo Logs« – aber auch im Header aller Datenbankfiles verzeichnet. Die SCN hat während eines Datenbank-Recovery, aber auch im Kontext von Lesekonsistenz eine zentrale Bedeutung. Diese eindeutige Transaktionsnummer kann in der folgenden Syntax verwendet werden:

```
SELECT deptno, dept, dname  
FROM scott.dept  
VERSIONS BETWEEN  
SCN 568394 AND 568410  
WHERE deptno = 50;
```

Listing 4.25: Verwendung der Syntax mit System Change Number

Mit dieser Syntax kann entweder eine feste SCN verwenden oder minvalue und maxvalue als Variable einsetzen. minvalue zeigt das älteste Before-Image, das für diesen Datensatz noch im Undo Tablespace vorhanden ist. Bei maxvalue handelt es sich um die letzte Änderung. Die Syntax mit »VERSIONS BETWEEN SCN minvalue AND maxvalue« zeigt also alle noch im Undo-Bereich gespeicherten Variationen aller vom SQL-Statement betroffenen Datensätze an. Alternativ kann auch eine einzelne SCN benannt werden.

4.4.2 Flashback Table: Zurücksetzen ganzer Tabellen

Im obigen Beispiel wurde die Datenmanipulation eines einzelnen Datensatzes verfolgt. Dieser konnte bei Bedarf über ein Undo-SQL wieder in den ursprünglichen Zustand zurückgesetzt werden. Wurden in einer Tabelle sehr viele Sätze verändert, ist es u.U. leichter, die gesamte Tabelle zurückzusetzen.