Hohes Datenaufkommen ist eine große Herausforderung für moderne IT-Systeme. Operativ nicht mehr verwendete Daten in ein Data Warehouse auszulagern ist eine notwendige Maßnahme, um eine stabile Performance im System zu gewährleisten. Zudem ermöglicht es, die Datenstruktur an das hohe Datenvolumen anzupassen.

Realtime Data Warehouse: Auslagern und Wiederherstellen operativer Daten

Steffen Kantorek, Berenberg Bank

Doch was kann man tun, wenn die ausgelagerten Informationen auch geraume Zeit später nochmals benötigt und bearbeitet werden müssen? Dieser Artikel beschreibt eine Lösung, um die Datenlast in einem operativen System gering zu halten, ohne dabei die Verfügbarkeit der Daten zu verlieren.

Es wird eine Anwendung mit einem durchschnittlichen Datenaufkommen von einer Million Datensätzen täglich betrachtet. Dabei werden ein Prozent, also zehntausend Datensätze, nachträglich geändert. 95 Prozent dieser Änderungen finden innerhalb der ersten sieben Tage statt. Es besteht jedoch in wenigen Fällen der Bedarf, auch später noch Änderungen an den Daten vorzunehmen.

Da die Daten aus dem operativen System in jedem Fall in das Data Warehouse übertragen werden, bietet es sich an, dies als Ausgangspunkt für eine Wiederherstellung der Daten zu verwenden. Die Idee hierbei ist, die Vorteile beider Systeme auszunutzen: zum einen die Geschwindigkeit des operativen Systems zur Datenverarbeitung und zum anderen die beim Umgang mit großen Datenmengen günstige Architektur des Data Warehouse.
Die Herausforderung besteht darin, einen Prozess zu schaffen, der es ermöglicht, die ausgelagerten Daten wiederherzustellen. Jedoch steht zum Anfang der Datenverarbeitung nicht fest, welche der Daten nochmals bearbeitet werden müssen. Der Prozess muss somit die Wiederherstellung der Daten zu einem beliebigen Zeitpunkt gewährleisten.

Hierzu wurden zwei getrennte Systeme aufgebaut: das operativ laufende System, das die Daten empfängt und verarbeitet, sowie das Data Warehouse, das als Speicherort für historische Daten dient und zu Auswertungszwecken benutzt wird. Da beide Systeme unabhängig voneinander arbeiten sollen, muss hier eine strikte System-Trennung vorliegen. Das bedeutet, dass ein Ausfall oder Performance-Problem eines Systems keine Auswirkungen auf das andere System haben darf.

Realtime-Auslagerung in das Data Warehouse

Bei der Auslagerung der Daten werden diese mittels Oracle Streams an das Data Warehouse übertragen. Dieser Prozess findet in Realtime statt. Hierzu werden die Daten auf dem operativen System durch einen Capture-Prozess

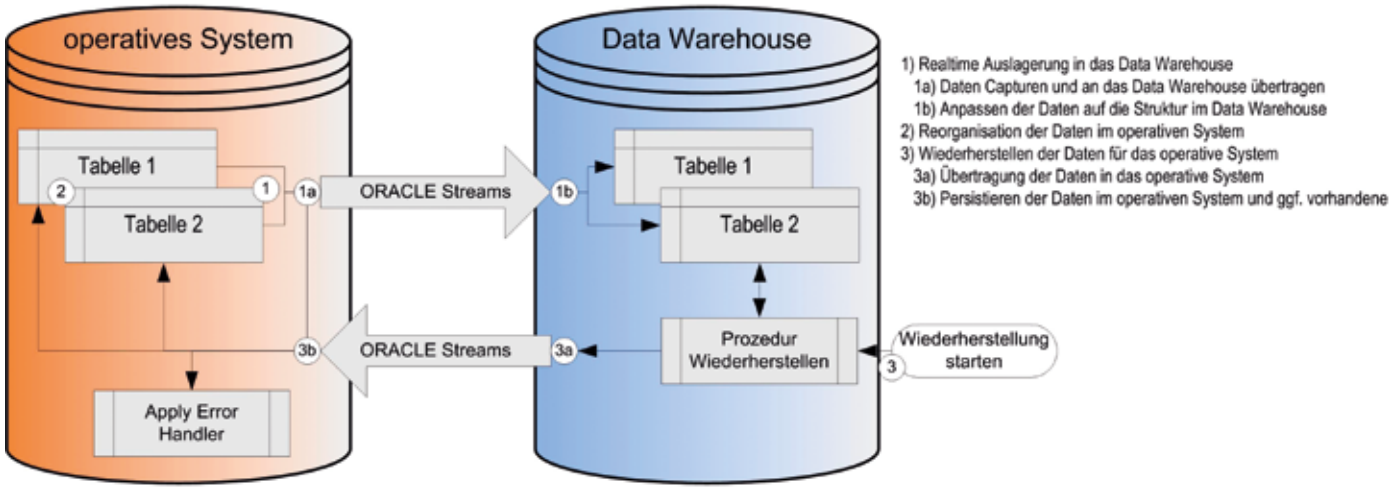

Abbildung 1: Schematischer Gesamtprozess, Auslagern und Wiederherstellen von Daten

in einen Logical Change Record (LCR) verpackt und per Queuing an das Data Warehouse übertragen. Oracle bietet verschiedene Möglichkeiten, um den Capture-Prozess aufzusetzen. Zum einen das klassische Capture über die Redo-Logs oder neu in Oracle 11*g* als synchrones Capture. Aber auch ein Capture über einen Trigger, der einen LCR erzeugt, ist denkbar.

Aufgrund der erweiterten Datenstruktur des Data Warehouse wird beim Apply des LCR die Datenstruktur angepasst (denormalisiert). Dafür befinden sich Trigger auf den Zieltabellen im Data Warehouse, die bei einer DML-Operation durch einen LCR starten. Im Trigger werden Foreign-Key-Beziehungen aufgelöst, um die Zugriffszeiten von Auswertungen auf historische Stammdaten zu minimieren. Hierfür sind die Firing Properties des Triggers mittels „DBMS_DDL"-Package angepasst. Ist das Merkmal „Firing Once" auf „false" gesetzt, startet der Trigger immer, auch beim Einarbeiten eines LCR. Durch die Realtime-Übertragung der Daten an das Data Warehouse befindet sich jeweils auf beiden Systemen der aktuelle Datenstand.

```
SQL> DESC UMSATZ /*Data Warehouse*/
Name       Type          Nullable Default Comments
---------- ------------- -------- ------- ------------------------
UMSATZ_ID  NUMBER                         Primary Key
KUNDE_ID   NUMBER        Y                Foreign Key auf hr.Kunde
BETRAG     NUMBER(14,2)  Y                Gesamtumsatz des Kunden
KUNDE_NAME VARCHAR2(200) Y                Kundenname

SQL>
…
CREATE OR REPLACE PROCEDURE DATEN_WIEDERHERSTELLEN(IN_UMSATZ_ID IN NUMBER)
IS
  MY_MSG_HANDLE RAW(16);
  REC           hr.UMSATZ%ROWTYPE;
  LCR           SYS.LCR$_ROW_RECORD;
  ROW_LIST      SYS.LCR$_ROW_LIST := SYS.LCR$_ROW_LIST();
  MY_ENQ_OPT    SYS.DBMS_AQ.ENQUEUE_OPTIONS_T;
  MY_MSG_PROP   SYS.DBMS_AQ.MESSAGE_PROPERTIES_T;
BEGIN
  -- Datensatz selektieren
  SELECT *
  INTO rec
  FROM hr.UMSATZ u
  WHERE u.UMSATZ_id = IN_UMSATZ_ID;

  -- LCR aufbauen
  ROW_LIST := SYS.LCR$_ROW_LIST();
  ROW_LIST.EXTEND(3);
  ROW_LIST(1) := SYS.LCR$_ROW_UNIT(,UMSATZ_ID‘,SYS.ANYDATA.
CONVERTNUMBER(rec.UMSATZ_ID), NULL, NULL, NULL);
  ROW_LIST(2) := SYS.LCR$_ROW_UNIT(,KUNDE_ID‘ ,SYS.ANYDATA.
CONVERTNUMBER(rec.KUNDE_ID) , NULL, NULL, NULL);
  ROW_LIST(3) := SYS.LCR$_ROW_UNIT(,BETRAG‘   ,SYS.ANYDATA.
CONVERTNUMBER(rec.BETRAG)   , NULL, NULL, NULL);

  LCR := SYS.LCR$_ROW_RECORD.CONSTRUCT(source_database_name => SYS_
CONTEXT(,USERENV‘,‘DB_NAME‘)
                                     , command_type         => ,INSERT‘
                                     , object_owner         => ,HR‘
                                     , object_name          => ,UMSATZ‘);

  LCR.SET_VALUES(,new‘, ROW_LIST);

  /*...
   Streams/ Queuing spezifische Werte setzen
   ... */

  -- Eisntellen in die Queue
  DBMS_AQ.Enqueue(QUEUE_NAME         => ,HR.AQ_WAREHOUSE_OUTBOX‘
                 ,ENQUEUE_OPTIONS    => MY_ENQ_OPT
                 ,MESSAGE_PROPERTIES => MY_MSG_PROP
                 ,PAYLOAD            => SYS.ANYDATA.CONVERTOBJECT(lcr)
                 ,MSGID              => MY_MSG_HANDLE );
END;
```

Listing 1: Description der Umsatz-Tabelle auf dem Data Warehouse und Capture-Prozedur. Die Spalte „KUNDE_NAME" wird nicht an das operative System übertragen.

Reorganisieren der Daten im operativen System

Um die Performance im operativen System stabil zu halten, sind die Daten zu reorganisieren. Aufgrund der Realtime-Übertragung der Daten zum Data Warehouse kann dies zu einem frei wählbaren Zeitpunkt am Ende der Datenverarbeitung erfolgen.

Da der Großteil der nachträglichen Änderungen innerhalb der ersten sieben Tage nach Erstellung der Daten erfolgt, startet nach dieser Zeit auch der Reorganisations-Prozess. Alle verarbeiteten Daten, die somit älter als sieben Tage sind, werden aus dem operativen System entfernt. Dabei ist allerdings zu beachten, das Löschungen nicht im Capture-Prozess berücksichtigt und somit nicht zum Data Warehouse übertragen werden; dies geschieht durch die Definition einer Regel, die besagt, dass der Capture-Prozess keine „DELETE"-Operation aufnehmen soll.

Wiederherstellen der Daten für das operative System

Werden ausgelagerte Daten im operativen System wieder benötigt, wird dieser Prozess manuell angestoßen. Die Daten aus dem Data Warehouse müssen nun in das operative System übertragen werden. Dieser manuelle Vorgang ändert die Daten im Data Warehouse nicht,

sodass kein Capture-Prozess die Daten aufnehmen und verarbeiten kann. Stattdessen wird eine Prozedur aufgerufen, die aus den Daten einen LCR erzeugt. Der Aufbau des LCR erfolgt in der Struktur des Zielsystems, es werden somit nur die Daten übermittelt, die auch dem operativen System zur Verfügung stehen (siehe Listing 1).

An dieser Stelle ist auch darauf zu achten, dass abhängige Daten auf dem operativen System in der korrekten Reihenfolge wiederhergestellt werden müssen, um Constraint-Verletzungen zu verhindern. Anschließend wird der LCR via Queuing an das operative System übermittelt, in dem die Daten persistiert sind. Eine erneute Übertragung der Daten in das Data Warehouse findet aufgrund definierter Regeln im Capture-Prozess nicht statt.

Um mögliche Fehler beim Wiederherstellen der Daten auf dem operativen System besser zu verwalten, kann man einen benutzerdefinierten Error-Handler für den Apply-Prozess hinterlegen (siehe Listing 2). Das Wiederherstellen eines im operativen System bestehenden Datensatzes würde eine Primary-Key-Verletzung im operativen System verursachen und den LCR in eine Exception Queue legen, um dort erneut verarbeitet werden zu können. Da dies aber aufgrund der Daten-Redundanz nicht notwendig ist, kann man einen Error-Handler definieren, der solche Fehler protokolliert, jedoch ein Speichern des LCR in die Exception Queue verhindert.

```
CREATE OR REPLACE PROCEDURE USER_DML_ERROR_HANDLER( message
IN  ANYDATA
                                                    ,error_stack_depth
IN  NUMBER
                                                    ,error_numbers
IN  DBMS_UTILITY.NUMBER_ARRAY
                                                    ,error_messages
IN  DBMS_UTILITY.LNAME_ARRAY )
IS
  lcr    SYS.LCR$_ROW_RECORD;
  v_info VARCHAR2(4000);
BEGIN
  -- Zugriff auf den LCR
  IF message.GetTypeName = ,SYS.LCR$_ROW_RECORD‘
    AND message.GETOBJECT(lcr) = DBMS_TYPES.SUCCESS
  THEN
     IF error_numbers(1) = 1 -- UNIQUE CONSTRAINT verletzung
        AND lcr.GET_OBJECT_OWNER = ,HR‘
        AND lcr.GET_COMMAND_TYPE = ,INSERT‘
        AND lcr.GET_OBJECT_NAME  = ,UMSATZ‘
     THEN
       /*Informationen fuer die Log Tabelle sammeln*/
       v_info := ,Error:‘
                 ||CHR(10)||‘Tabelle= HR.UMSATZ‘
                 ||CHR(10)||‘UMSATZ_ID= , ||lcr.GET_VALUE( value_
type  =>‘NEW‘
                                                          ,column_
name =>‘UMSATZ_ID‘ ).ACCESSNUMBER;

      /*...*/
    END IF;
  END IF;
  /*...*/
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
….
BEGIN
  DBMS_APPLY_ADM.SET_DML_HANDLER(OBJECT_NAME    => ,HR.UMSATZ‘
                                ,OBJECT_TYPE    => ,TABLE‘
                                ,OPERATION_NAME => ,INSERT‘
                                ,ERROR_HANDLER  => TRUE
                                ,USER_PROCEDURE => ,USER_DML_ERROR_
HANDLER‘
                                ,APPLY_NAME     => ,APPLY_WAREHOUSE_
INBOX‘
                                );
END;
```

Listing 2: Definition eines DML-Error-Handlers für Inserts auf die Tabelle „HR.UMSATZ“. Der Error-Handler wird bei fehlerhaften Insert-Operationen des Apply-Prozesses „APPLY_WAREHOUSE_INBOX“ aufgerufen

Fazit

Die vorgestellte Lösung zum Wiederherstellen von Daten wird bereits erfolgreich in der Berenberg Bank eingesetzt und stellt eine komfortable Möglichkeit dar, ausgelagerte Daten nachträglich zu bearbeiten. Darüber hinaus stellt Oracle mit Streams ein Werkzeug zur Verfügung, das die Replikation von Daten zwischen homogenen Systemen vereinfacht, auch bei abweichenden Datenstrukturen auf den Systemen. Es sollte jedoch auch die Datenmenge betrachtet werden, die zwischen den Systemen innerhalb einer Transaktion repliziert wird. So kann es bei großen Datenmengen vorkommen, dass die Replizierung sehr lange dauert. Man sollte sich also bereits im Vorfeld überlegen, welche Daten man wirklich benötigt.

Steffen Kantorek
Berenberg Bank
steffen.kantorek@berenberg.de