ETL-Prozesse beschleunigen

Dani Schnider, Trivadis AG

Man erlebt hin und wieder die Situation, dass die Nacht zu kurz ist. Oder mit anderen Worten: Der nächtliche ETL-Lauf dauert so lange, dass die Daten am Morgen nicht rechtzeitig in den Data Marts zur Verfügung stehen. Um für diese Problematik Abhilfe zu schaffen, gibt es ein paar grundlegenden Maßnahmen, die beim Entwickeln von ETL-Prozessen beachtet werden sollten.

ETL-Tools und Datenbank-Systeme stellen verschiedene Features und Möglichkeiten zur Verfügung, die ein effizientes Laden von Daten ins Data Warehouse erlauben oder mit denen die bestehenden Ladeprozesse beschleunigt werden können. Doch leider werden diese Möglichkeiten oft nicht optimal ausgenutzt. Immer wieder muss der Autor im Rahmen von Performance-Einsätzen und Reviews von Data Warehouses feststellen, dass teilweise elementare Regeln zur effizienten Realisierung von ETL-Prozessen missachtet werden. Ohne hier zu stark auf die Spezialitäten der einzelnen Tools einzugehen, erläutert der Artikel einige wichtige Grundlagen, die Voraussetzung für eine gute Performance der Ladeprozesse sind.

ELT statt ETL

Je nach verwendeter Technologie der ETL-Tools werden die Transformationen auf einem ETL-Server durchgeführt oder finden innerhalb der Datenbank statt. Im zweiten Fall wird auch oft der Begriff „ELT" (Extract, Load, Transform) statt „ETL" (Extract, Transform, Load) verwendet. Insbesondere bei großen Datenmengen sowie langsamen Netz-Verbindungen zwischen ETL-Server und Datenbank können diese zwei Arbeitsweisen markante Unterschiede in der Laufzeit von ETL-Prozessen zur Folge haben. In der Regel sind ELT-Verarbeitungen schneller, da der Datentransfer zwischen ETL-Server und Datenbank entfällt. Durch geeignete Maßnahmen wie schnelle Netzwerk-Verbindungen oder Bulk-Verarbeitung zwischen ETL-Server und Datenbank kann die Verarbeitungszeit soweit optimiert werden, dass sie in einer ähnlichen Größenordnung wie die ELT-Verarbeitung liegt *(siehe Abbildung 1)*.

In vielen Data Warehouses kommen Integrationswerkzeuge zum Einsatz, die ausschließlich ETL-Funktionalität verwenden, ohne dass dies zu Performance-Problemen führt. Solange die zu ladenden Datenmengen genügend klein sind, fällt die Verarbeitungszeit meistens nicht ins Gewicht. Für größere Datenmengen stehen zum Teil spezielle ELT-Features zur Verfügung, die es erlauben, die Transformationsprozesse auf der Ziel-Datenbank durchzuführen (beispielsweise Pushdown Optimization in Informatica PowerCenter oder ELT-Komponenten in Talend Open Studio). Bietet ein ETL-Tool keine entsprechende Funktionalität an, wird oft die Möglichkeit gewählt, zeitkritische Verarbeitungen mittels SQL in der Datenbank zu implementieren (etwa in PL/SQL-Packages) und aus dem ETL-Tool aufzurufen.

Bei den Integrationstools von Oracle ist dies nicht notwendig: Sowohl Oracle Warehouse Builder (OWB) als auch Oracle Data Integrator (ODI) arbeiten standardmäßig als ELT-Werkzeuge. OWB generiert PL/SQL-Packages in der Ziel-Datenbank. ODI führt die Transformationen in den meisten

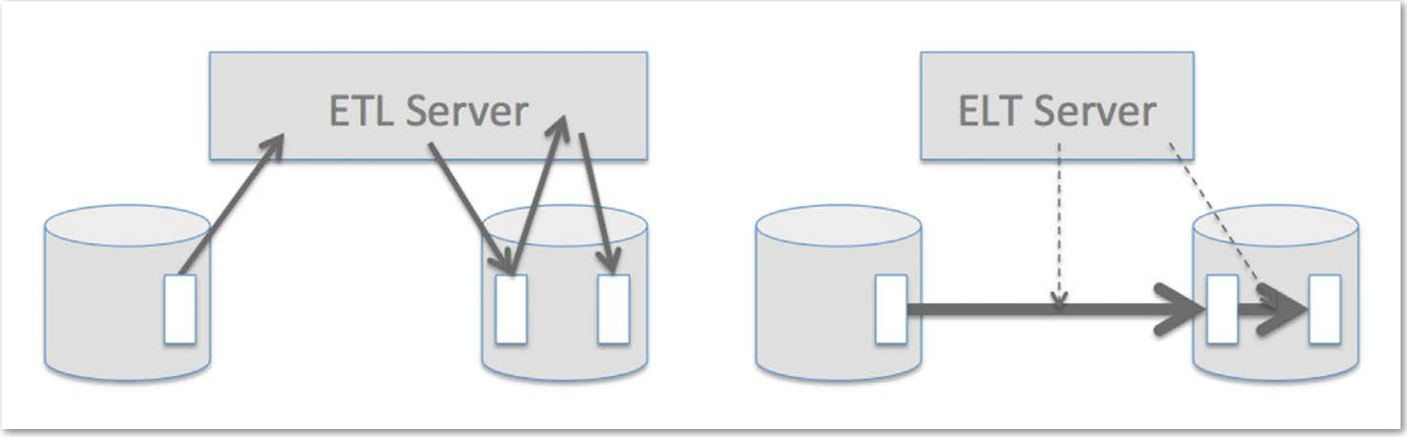

Abbildung 1: Arbeitsweise von ETL (links) und ELT (rechts)

Fällen als SQL-Befehle in der Zieldatenbank aus, unterstützt aber in heterogenen Umgebungen auch ETL-Verarbeitungen.

Mengen- statt datensatzbasierter Verarbeitung

Die Verarbeitung eines Transformationsprozesses – sei es als ETL oder ELT – kann entweder mengen- („set-based") oder datensatzbasiert („row-based") erfolgen. Wer den OWB einsetzt, kennt diese Unterscheidung, denn dieser unterstützt beide Ausführungsarten. Beim ODI kommt in den meisten Knowledge-Modulen eine mengenbasierte Verarbeitung vor. Verschiedene ETL-Tools von Fremdherstellern arbeiten datensatzbasiert, erlauben aber – mit mehr oder weniger Komfort – auch die Ausführung mengenbasierter Ladeprozesse.

Die unterschiedliche Verarbeitungsweise soll an einem einfachen Beispiel mit PL/SQL und SQL aufgezeigt werden. Eine Stage-Tabelle „STG_SALES" soll in eine Cleanse-Tabelle „CLS_SALES" geladen werden, wobei folgende Transformationen erfolgen sollen:

- Den Datumschlüssel „DATE_ID" aus dem Verkaufsdatum (ohne Uhrzeit) ermitteln
- Den Produktschlüssel „PRODUCT_ID" mittels Key Lookup auf der Core-Tabelle „COR_PRODUCT" ermitteln. Ist kein entsprechendes Produkt vorhanden, soll ein Singleton-Wert (-1) eingefügt werden
- Den Verkaufskanal „CHANNEL_ID" anhand eines Flags „ONLINE_FLAG" ermitteln
- Fehlt die Mengen-Angabe für einen Verkauf, wird für das Attribut „QUANTITY" der Wert „1" verwendet

```
DECLARE
   CURSOR cur_stage IS
      SELECT sales_date, product_code, online_flag, quantity
      FROM stg_sales;
   v_cls  cls_product%ROWTYPE;
BEGIN
   FOR v_stg IN cur_stage LOOP
      -- convert sales date
      v_cls.date_id := TRUNC(v_stg.sales_date);
      -- lookup product id
      BEGIN
         SELECT dwh_id
         INTO v_cls.product_id
         FROM cor_product
         WHERE product_code = v_stg.product_code;
      EXCEPTION
         WHEN NO_DATA_FOUND THEN
            v_cls.product_id := -1;
      END;
      -- define sales channel
      IF v_stg.online_flag = 'Y' THEN
         v_cls.channel_id = 1; -- internet
      ELSIF v_stg.online_flag = 'N' THEN
         v_cls.channel_id = 2; -- shop
      ELSE
         v_cls.channel_id = -1; -- unknown
      END IF;
      -- cleansing action for quantity
      IF v_stg.quantity IS NULL THEN
         v_cls.quantity := 1;
      ELSE
         v_cls.quantity := v_stg.quantity;
      END IF;
      -- insert row in cleanse table
      INSERT INTO cls_sales VALUES v_cls;
      COMMIT;
   END LOOP;
END;
```

Listing 1: Datensatzbasierte Verarbeitung („row-based")

Listing 1 zeigt, wie diese Anforderungen mittels PL/SQL-Block als datensatzbasierte Verarbeitung implementiert werden könnte. Über einen SQL-Cursor wird jeder Datensatz aus der Stage-Tabelle gelesen, transformiert und in die Cleanse-Tabelle geschrieben. Die Lösung funktioniert, ist aber alles andere als schnell. Insbesonde-

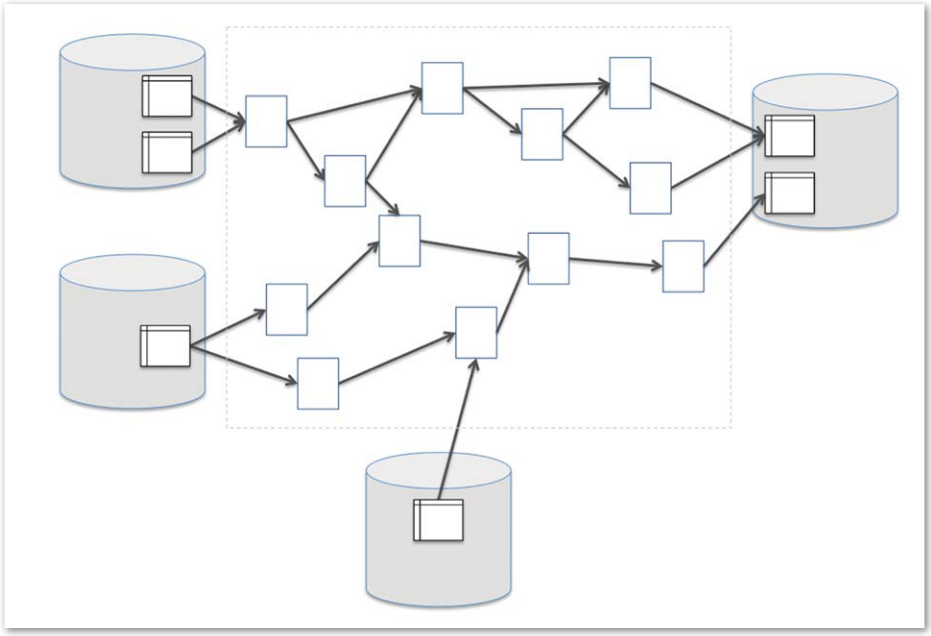

Abbildung 2: Beispiel für komplexen ETL-Prozess

```
INSERT /*+ append */ INTO cls_
sales
       ( date_id
       , product_id
       , channel_id
       , quantity)
SELECT TRUNC(stg.sales_date)
       , NVL(lkp.dwh_id, -1)
       , CASE stg.online_flag
           WHEN 'Y' THEN  1
           WHEN 'N' THEN  2
           ELSE           1
   END
   , NVL(stg.quantity, 1)
   FROM stg_sales stg
   LEFT OUTER JOIN cor_product
lkp
       ON (stg.store_number = lkp.
store_number);
COMMIT;
```

Listing 2: Mengenbasierte Verarbeitung („set-based“)

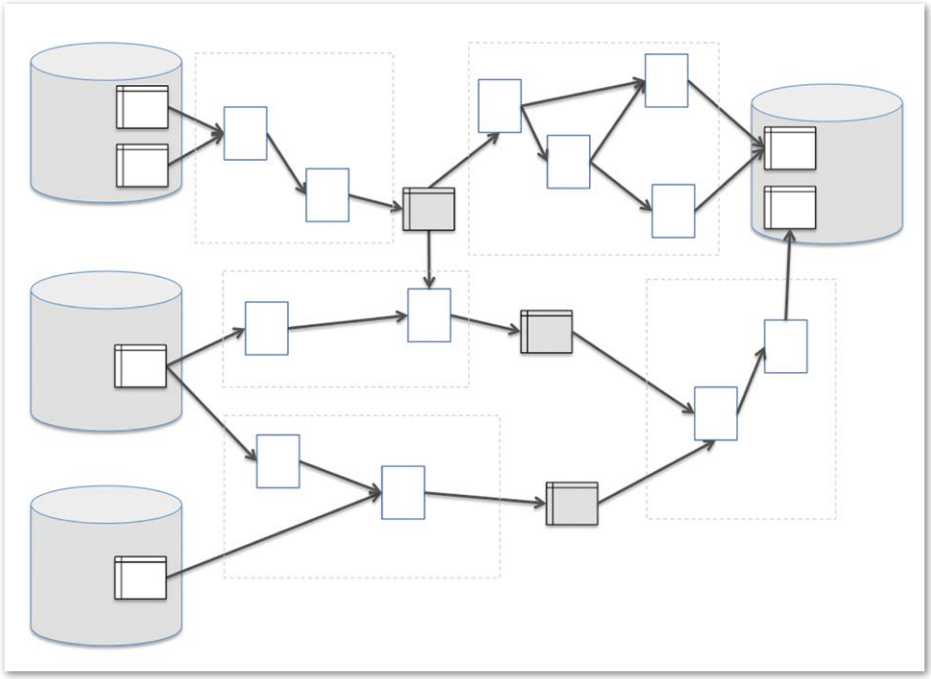

Abbildung 3: Aufteilung in fünf überschaubare Sub-Prozesse

re der ausprogrammierte Key Lookup und das „COMMIT“ nach jedem Datensatz können als Performance-technische Todsünden bezeichnet werden.

Deutlich effizienter ist es, die gleiche Logik mit mengenbasiertem SQL-Befehl zu formulieren. Dies ermöglicht das effiziente Laden von Daten aus einer oder mehreren Quell-Tabellen in eine Ziel-Tabelle durch SQL-Befehle („INSERT“ oder „MERGE“). Das SQL-Statement in *Listing 2* umfasst die gleiche Funktionalität wie der zuvor beschriebene PL/SQL-Block, wird aber bei großen Datenmengen viel schneller ausgeführt. Um die Verarbeitung zusätzlich zu beschleunigen, wird in Oracle ein Direct-Load „INSERT“ (mit dem Hint „/*+ append */“) verwendet.

Was hier mit PL/SQL und SQL aufgezeigt wird, funktioniert auch in zahlreichen Integrationswerkzeugen. Bei OWB und ODI entspricht die mengenbasierte Verarbeitung der (empfohlenen) Standardausführung. Andere ETL-Tools bieten zum Teil Operatoren zur Ausführung von SQL-Befehlen an oder ermöglichen es, die Datensätze mithilfe von Array-Verarbeitung aufzubereiten und mittels Bulk-Operationen in die Datenbank zu schreiben. Werden beispielsweise jeweils 1.000 oder 10.000 Datensätze mit einem „INSERT“-Befehl in die Datenbank geschrieben, ist dies schon deutlich schneller, als wenn für jeden einzelnen Datensatz ein SQL-Befehl ausgeführt werden muss.

Reduktion der Komplexität

Unabhängig von der eingesetzten Technologie ist folgende Performance-Maßnahme immer zu empfehlen: Komplexe ETL-Prozesse sollten, wann immer möglich, in mehrere, überschaubare Einzelschritte aufgeteilt sein. Dies gilt sowohl bei der Implementierung mit SQL oder mit einer prozeduralen Programmiersprache als auch bei der Realisierung mit einem ETL-Tool. Bei prozeduralen Sprachen gehört es zum guten Programmierstil, komplexe Abläufe zu modularisieren und in mehrere Subprogramme oder Prozeduren aufzuteilen. Das gleiche Prinzip gilt auch bei der Entwicklung von komplexen ETL-Prozessen, die als Mappings oder Jobs mit einem ETL-Tool implementiert werden. Der in *Abbildung 2* schematisch dargestellte ETL-Prozess kann – und soll – in mehrere Teilschritte aufgeteilt sein, die

nacheinander oder parallel ausgeführt werden können.

Anstatt den komplexen ETL-Prozess in einem umfangreichen Mapping zu implementieren, wird der Ablauf in mehrere Subprozesse unterteilt, die als separate Mappings implementiert sind *(siehe Abbildung 3)*. Dies hat mehrere Vorteile:

- Die Summe der Ausführungszeiten der einzelnen Sub-Prozesse ist viel kürzer als die Ausführungszeit des gesamten Prozesses, da für die einfacheren Operationen viel weniger Ressourcen benötigt werden. Bei ELT-Tools oder SQL-Statements, die direkt in der Datenbank ausgeführt werden, kommt hinzu, dass der Query Optimizer der Datenbank die einfacheren Statements leichter optimieren kann.
- Die einzelnen Mappings sind einfacher überschaubar, was die Weiterentwicklung bei zukünftigen Erweiterungen sowie die Einarbeitung neuer ETL-Entwickler vereinfacht. Auch hier gilt das Gleiche wie bei Programmiersprachen: Überschaubare Programme sind leichter zu verstehen als „Spaghetti-Code".
- Schließlich ist auch die Fehlersuche einfacher, da die Zwischen-Resultate in Stage-Tabellen oder weiteren Zwischen-Tabellen abgespeichert und dort vom nächsten Verarbeitungsschritt wieder gelesen werden. Somit lässt sich im Falle von fehlerhaften Daten einfacher nachvollziehen, in welchem Teilschritt der Fehler auftritt, da man die Zwischen-Resultate analysieren kann.

Die Reduktion der Komplexität bewirkt somit nicht nur kürzere Laufzeiten der ETL-Prozesse, sondern führt auch zu Zeiteinsparungen bei der Entwicklung und beim Testen der Ladestrecken. Wir haben also durch diese Maßnahme auch Performance-Verbesserungen bei der Realisierungszeit erreicht.

Frühzeitige Mengen-Einschränkung

Je kleiner die zu verarbeitende Datenmenge ist, desto schneller lässt sich ein ETL-Prozess ausführen. Deshalb ist es wichtig, bei der ETL-Entwicklung darauf zu achten, dass die Datenmenge so früh wie möglich eingeschränkt wird. Ist ein Mapping so aufgebaut, wie in *Abbildung 4* gezeigt,

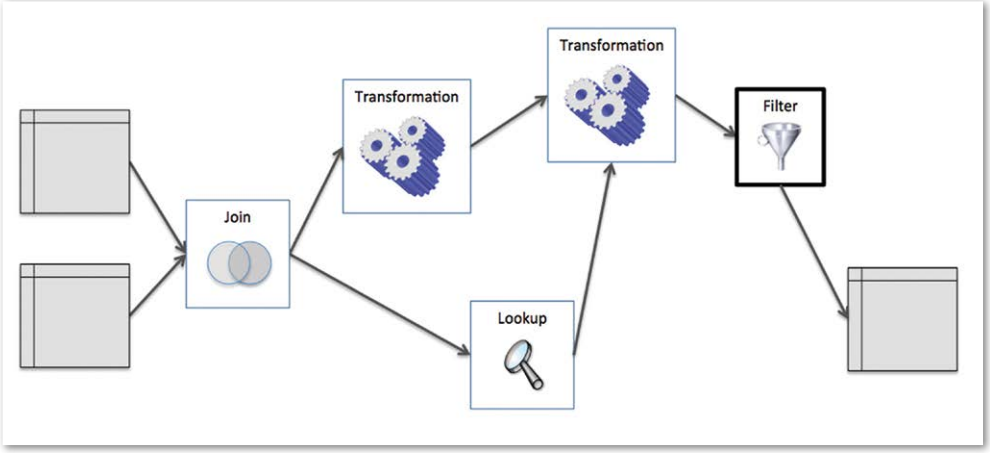

Abbildung 4: Mengen-Einschränkung am Ende des ETL-Prozesses

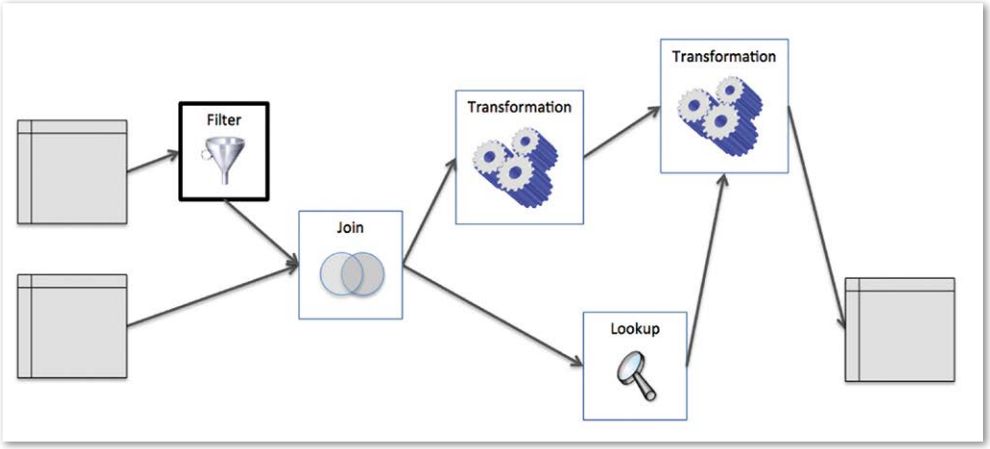

Abbildung 5: Frühzeitige Mengeneinschränkung am Anfang des ETL-Prozesses

kann dies negative Auswirkungen auf die Performance des ETL-Prozesses haben, denn bei ELT-Tools ist es möglich, dass der Query Optimizer der Datenbank die ausgeführten SQL-Befehle so optimieren kann, dass trotz schlechten Designs des Mappings die frühzeitige Mengeneinschränkung funktioniert. Nach aufwändigen Zwischenschritten und Transformationen wird am Ende ein Filter eingefügt, der nur eine Teilmenge der Daten in die Zieltabelle schreibt. Das bedeutet, dass für alle nicht relevanten Datensätze die Transformationsschritte ebenfalls ausgeführt wurden – und zwar vergeblich. Dies sollte möglichst vermieden werden.

Besser ist es, die Filterkriterien so früh wie möglich anzuwenden und so die Datenmenge für die nachfolgenden Transformationsschritte zu reduzieren. Das Beispiel in *Abbildung 5* ist ein optimaler Fall, da eine Filterung der Daten bereits auf einer der beiden Quell-Tabellen erfolgen kann (etwa das Lesen der aktuellen Version aus einer Dimensions-Tabelle). Das ist nicht immer möglich. Unter Umständen kann ein Filter-Kriterium erst aus dem Ergebnis eines Joins, eines Key Lookups oder einer Transformation ermittelt werden (etwa das Eliminieren aller Datensätze, für die beim Key Lookup kein passender Schlüssel gefunden wurde). Aber auch dann sollte die Filterung so früh wie möglich stattfinden und nicht erst vor dem Schreiben in die Ziel-Tabelle.

Parallelisierung

Um die Ladezeit ins Data Warehouse zu verkürzen, besteht die Möglichkeit, die ETL-Prozesse zu parallelisieren. Dabei stehen verschiedene Varianten zur Verfügung. Einerseits können die einzelnen SQL-Befehle zum Lesen und Schreiben der Datenbank-Tabellen parallelisiert werden. Andrerseits lassen sich mehrere separate ETL-Prozesse gleichzeitig ausführen.

Idealerweise sollte die Parallelisierung den gesamten ETL-Ablauf umfassen: Die Da-

ten werden zum Beispiel mit mehreren parallelen Sub-Prozessen aus der Stage-Tabelle gelesen, transformiert und anschließend in die Cleanse-Tabelle geschrieben. Erfolgt einer der Schritte (beispielsweise die Transformation) seriell, so führt dies zu einem Flaschenhals in der Verarbeitung und somit zu einer längeren Ausführungszeit. Autofahrer kennen die Situation, wenn sie auf einer mehrspurigen Autobahn unterwegs sind. Eine Reduktion der Spuren – zum Beispiel durch eine Baustelle – führt unweigerlich zu einem Rückstau und somit zu „Performance-Problemen" im Straßenverkehr.

Zurück von der Autobahn ins Data Warehouse: Idealerweise erfolgt die Parallelisierung der Ladeprozesse durch ELT-Technologien, es werden also die Parallelisierungsmöglichkeiten der Datenbank ausgenutzt. Bei einer mengenbasierten Ausführung mit Datenbank-Features wie Parallel-Query- und Parallel-DML-Operationen lässt sich ein optimaler Datendurchsatz erreichen. *Listing 3* zeigt das bereits beschriebene Beispiel einer mengenbasierten Verarbeitung, allerdings achtfach parallelisiert. Je acht SQL-Subprozesse lesen die Daten aus der Stage-Tabelle „STG_SALES" und schreiben sie in die Cleanse-Tabelle „CLS_SALES".

Eine andere Variante der Parallelisierung besteht darin, mehrere ETL-Prozesse gleichzeitig auszuführen. Solange die einzelnen Abläufe unabhängig voneinander sind und verschiedene Quellen und Ziele haben, ist dies eine einfache Maßnahme, um die Ausführungszeit eines ETL-Laufs zu reduzieren. Ein typischer Anwendungsfall ist beispielsweise das parallele Laden aller Dimensionen. Das Laden der Fakten-Tabelle kann hingegen erst beginnen, wenn alle Dimensions-Tabellen vollständig geladen sind.

Vermieden werden sollte hingegen die gleichzeitige Ausführung von ETL-Prozessen, die in dieselbe Ziel-Tabelle schreiben. Dies kann zu Locking-Problemen auf der Datenbank führen, wenn mehrere Client-Prozesse in die gleichen Tabellen oder Partitionen schreiben. Bei Oracle ist dies vor allem dann der Fall, wenn auf der Ziel-Tabelle Bitmap-Indizes vorhanden sind.

Aus Sicht der Datenbank ist dieses Verfahren vergleichbar mit einem Multi-User-Betrieb in einem OLTP-System und nicht geeignet für die Massenverarbeitung von großen Datenmengen. Möglich ist dieses Prinzip höchstens in Kombination mit partitionierten Tabellen. Kann sichergestellt werden, dass jeder Prozess in eine separate Ziel-Partition schreibt, so können mehrere ETL-Prozesse parallel ablaufen, um Daten in die gleiche Tabelle zu schreiben.

```
INSERT /*+ parallel(cls, 8) */
INTO cls_sales cls
      ( date_id
      , product_id
      , channel_id
      , quantity)
SELECT /*+ parallel (stg, 8) */
         TRUNC(stg.sales_date)
      , NVL(lkp.dwh_id, -1)
      , CASE stg.online_flag
          WHEN 'Y' THEN  1
          WHEN 'N' THEN  2
          ELSE           -1
        END
      , NVL(stg.quantity, 1)
  FROM stg_sales stg
  LEFT OUTER JOIN cor_product
lkp
      ON (stg.store_number = lkp.
store_number);
COMMIT;
```

Listing 3: Parallele Ausführung von SQL-Befehlen

Datenbank-Konfiguration

Die bisher beschriebenen Performance-Maßnahmen betreffen hauptsächlich Design und Entwicklung der ETL-Prozesse. Daneben sind aber auch die korrekte Konfiguration der Datenbank sowie das physische Datenbank-Design wichtig, um die Ladezeiten möglichst klein zu halten.

- In Data Warehouses gelten andere Regeln als in OLTP-Systemen. Dies hat Auswirkungen auf die Konfiguration der Datenbanken. Eine Data-Warehouse-Datenbank wird so konfiguriert, dass sie für nicht-selektive Abfragen optimiert ist, genügend Arbeitsspeicher für Massenverarbeitungen zur Verfügung hat und die parallele Ausführung von ETL-Prozessen und Auswertungen erlaubt *(„siehe https://danischnider.wordpress.com/2015/04/29/oracle-initialization-parameters-for-data-warehouses")*.
- Auch die Indexierung ist in einem Data Warehouse unterschiedlich zu einem operativen System. Für effiziente ETL-Prozesse ist es zweckmäßig, so wenige Indizes wie möglich anzulegen. Einzig auf den Data Marts, die für Benutzerabfragen optimiert sind, werden Indizes angelegt – in der Regel Bitmap-Indizes. In allen anderen DWH-Schichten werden nur wenige oder gar keine Indizes erstellt *(siehe „http://www.slideshare.net/trivadis/indexierungsstrategie-im-data-warehouse-zwischen-albtraum-und-optimaler-performance-39738594")*. Je nach Art der Ladeverarbeitung (initiale oder inkrementelle Ladeprozesse) kann es sogar sinnvoll sein, die Indizes vor dem Laden auf „UNUSABLE" zu setzen und nach dem Laden mit „REBUILD" neu zu erstellen.
- Für die Abfragen, aber auch für nachfolgende ETL-Prozesse, ist es wichtig, dass man nach dem Laden von Daten die Optimizer-Statistiken der geladenen Tabellen und Indizes aktualisiert. Dies sollte nicht erst am Ende eines kompletten Ladelaufs erfolgen, sondern nach jedem Zwischenschritt. Die zuerst geladenen Tabellen werden in weiteren ETL-Prozessen als Quellen verwendet. Insbesondere bei der mengenbasierten ELT-Verarbeitung ist es wichtig, dass der Query Optimizer korrekte Statistiken für die Optimierung der nachfolgenden Transformationsschritte verwenden kann *(siehe „https://danischnider.wordpress.com")*.

Sind Konfiguration und Design der Datenbank für ein Data Warehouse ausgelegt und werden die ETL-Prozesse gemäß den hier beschriebenen Grundprinzipien implementiert, steht einem erfolgreichen und effizienten Laden des Data Warehouse nichts mehr im Wege. Die Nacht ist somit nicht mehr zu kurz und die Daten im Data Warehouse stehen am Morgen für die Endanwender rechtzeitig zur Verfügung.

Dani Schnider
dani.schnider@trivadis.com