Das Beste aus zwei Welten

Bruno Cirone, bc-consult

Manchmal muss man eine Tuning-Maßnahme noch vor der Arbeit mit Oracle beginnen. Bei dem hier beschriebenen Projekt werden die Stärken von Unix/Linux und von Oracle zusammengeführt und dies führt dann zu erstaunlichen Ergebnissen.

Bei einem Projekt eines großen europäischen Stahlkonzerns geht es darum, dass eine große Menge an Daten (ca. 4 Millionen Sätze) in regelmäßigen Abständen vom Hostsystem an die untergeordneten Systeme (ca. 400 Systeme) übertragen wird. Diese Datei ist unstrukturiert mit festen Satzlängen, im EBCDIC-Format und dazu noch komprimiert. Aus diesen 4 Millionen Sätzen werden lediglich ca. 8.000 bis 20.000 Sätze und nur einige Spalten pro System benötigt.

Die infrage kommenden Sätze müssen entweder in die Tabelle eingetragen werden oder, falls bereits vorhanden, muss der Datensatz geändert und mit einem neuen Kennzeichen versehen werden. Leider konnten die Host-Betreuer nicht dazu bewegt werden, nur die benötigten Sätze zu übertragen. Auf dem Hostsystem ist es etwas aufwendiger, solche Maßnahmen umzusetzen. Also kurz gesagt, wie ein Mitarbeiter des Kunden meinte, „flexibel wie eine Eisenbahnschranke".

Ein wenig Historie dieses Projekts: Im Jahr 1990 wurde es in Cobol mit Oracle 7.3.2 entwickelt. Das Programm lief regelmäßig alle acht Stunden; die Laufzeit betrug etwa zwei Stunden. Zu diesem Zeitpunkt gab es keine External Tables und auch kein Merge-Statement. Für das Laden von Daten hätte man den SQL-Loader („sqlldr") nutzen können, was jedoch nicht gewollt war und den Ansprüchen auch nicht gerecht worden wäre.

Neue Hardware und aktuelle Oracle-Versionen ergaben eine Verbesserung der Laufzeit auf rund 40 Minuten. Die Frequenz wurde dadurch auf vier Stunden reduziert. Während der Laufzeit des Programms ist die Anwendung gesperrt.

Da mittlerweile die Programmierer in den Ruhestand gingen, traute sich niemand mehr an dieses Programm heran. Die Anforderung, die Frequenz zu erhöhen, wurde andererseits immer dringender. Es sollte daher eine Optimierung in der Datenbank vorgenommen werden. Nach eingehender Erklärung des Transaktionskonzepts von Oracle bestand keine Notwendigkeit mehr für einen Stillstand der Anwendung. Diese „Tuning"-Maßnahme griff sofort und es gab keine Stillstände mehr. Damit war allerdings die lange Laufzeit des Programms nicht behoben, Änderungen des Host-Systems dauerten also immer noch sehr lange, bis sie in den Subsystemen komplett verarbeitet wurden.

Erste Tuning-Phase

Nach Analyse des Cobol-Programmes hat der Autor festgestellt, dass die Anforderung mit Unix/Linux und Datenbank-Bordmitteln komplett umgesetzt werden könnte. Es wurde eine Datei mit rund vier Millionen Sätzen mit jeweils 800 Bytes pro Satz und einer Blockgröße von 32000 Bytes bearbeitet. Insgesamt wurden etwa 8.000 Sätze in der Datenbank eingefügt beziehungsweise geändert. Ein einfaches Shell-Skript hat dazu ausgereicht *(siehe Listing 1)*.

Das Ergebnis ist erfreulich: Ein Cobol-Programm komplett durch ein Shell-Skript sowie SQL*Plus ersetzt und dazu die Laufzeit auf ca. 25 Prozent reduziert. Bei den Laufzeiten kann man sehr gut erkennen, dass die Hauptarbeit immer noch von Oracle vorgenommen werden muss. Immerhin muss Oracle ja jeden Datensatz lesen und die infrage kommenden Spalten selektieren und bearbeiten. Weitere Optimierungen sollten vorerst nicht erfolgen. Diese Variante wurde in der Produktion eingesetzt und die Frequenz weiter reduziert. Das Shell-Skript sollte nun alle zwei Stunden laufen.

Zweite Tuning-Phase

Etwa zwei Jahre später kam die Frage auf, ob weitere Tuning-Maßnahmen möglich wären. Auch dabei kommt man mit einem einfachen Shell-Skript zum Erfolg. Bei dieser Variante sind die Befehle „gunzip", „dd", „grep" und „cut" mit „pipe" verbunden *(siehe Listing 2)*.

Eine solche Laufzeitreduktion auf ca. 105 Sekunden war vorher nicht vorstellbar. Bei dem Unix/Linux-Kommando könnte auch zuerst „cut" und danach „grep" vertauscht werden, dies hat aber zur Folge, dass die Laufzeit wieder stark ansteigt *(siehe Listing 3)*.

Der Effekt ist damit zu erklären, dass beim ersten Befehl zunächst alle infrage kommenden Sätze gefiltert werden und danach der zu bearbeitende Satz mit „cut" zusammengestellt wird. Bei der zweiten Variante werden praktisch alle vier Millionen Sätze weitergegeben und danach gefiltert. Trotzdem könnte die zweite Variante sinnvoll sein, wenn viele Sätze an die Datenbank übergeben werden müssen. Die Zwischenergebnisse geben einen guten Anhaltspunkt dafür, welche Reihenfolge sinnvoll ist. Für einen Test reicht eine kleinere Menge (etwa ca. 10.000 Sät-

```
time gunzip transfer_file_ebcdic.gz
real    2m3.305s
time dd if=transfer_file_ebcdic of=transfer_file conv=ascii ibs=32000 obs=800
real    0m43.949s
time sqlplus bc/x1 @/home/oracle/external_table_tuning1.sql
Real    7m37.609s
```

Listing 1

```
time gunzip -c transfer_file_ebcdic.gz |  dd conv=ascii  ibs=32000 obs=800 | grep 2010 |  cut -c1-9,11-12,15-
18,20-29,50-59,70-74 > teil2
real    1m20.205s
time sqlplus bc/x1 @/home/oracle/external_table_tuning2.sql
Real    0m24.609s
```

Listing 2

```
time gunzip -c transfer_file_ebcdic.gz |  dd conv=ascii  ibs=32000 obs=800 | cut -c1-9,11-12,15-18,20-29,50-
59,70-74 | grep 2010   > teil2
real    2m10.100s
```

Listing 3

```
cut -c1-9,11-12,15-18,20-29,50-59,70-74  Testdatei |tee cut.file | grep 2010  > nach_grep.file
grep 2010 Testdatei |tee grep.file | cut -c1-9,11-12,15-18,20-29,50-59,70-74  > nach_cut.file
ls -l Testdatei grep.file cut.file

-rw-rw-rw-   1 user group    8088266 Aug  5 12:06 Testdatei
-rw-rw-rw-   1 user group     414510 Aug  5 12:06 cut.file
-rw-rw-rw-   1 user group      40190 Aug  5 12:06 grep.file
```

Listing 4

```
time sqlplus bc/x1 @/home/oracle/external_table_tuning3.sql
Real    0m48.201s
```

Listing 5

```
/usr/bin/zcat $1 | /usr/bin/dd conv=ascii 2>/dev/null \
                 | /usr/bin/grep 2010     2>/dev/null \
                 | /usr/bin/cut -c1-9,11-12,15-18,20-29,50-59,70-74
2>/dev/null
```

Listing 6

```
CREATE TABLE Fertigung
   (
    Satz_id     number(10),
    Satzart     varchar2(1),
    Werk        varchar2(4),
    Teile_id    varchar2(10),
    Kunde_id    varchar2(10),
    Menge       number(5)
   )
   ORGANIZATION EXTERNAL
    (
      TYPE oracle_loader
      DEFAULT DIRECTORY ext_tables
      access parameters
             (
             records delimited by newline
             preprocessor ext_tables:'bc.sh'
             fields (
                      Satz_id    position ( 1: 9) char(10),
                      Satzart    position (10:11) char(1),
                      Werk       position (12:15) char( 4),
                      Teile_id   position (16:25) char(10),
                      Kunde_id   position (26:35) char(10),
                      Menge      position (36:40) char(5)
                    )
             )
      LOCATION ('transfer_file_ebcdic.gz')
    )
   REJECT LIMIT UNLIMITED
;
```

Listing 7

ze) aus, um eine gesicherte Aussage treffen zu können. Zwischenergebnisse lassen sich wie in *Listing 4* einfach ermitteln. Die entsprechende Frequenz wurde wieder weiter reduziert, das Shell-Skript lief jetzt jede Stunde.

Dritte Tuning-Phase

Bis dahin war für den Autor klar, dass für External Tables, wie der Name schon sagt, eine wirkliche Datei als Input existieren muss. Er fand jedoch, dass dieses Problem mit einer Pipe oder Named-Pipe lösbar sein müsste. Während des Besuchs der Real-World Performance Tour 2015 in Wien konnte er das Problem mit Tom Kyte von Oracle besprechen. Er sagte, dass die External Tables auch eine Ausgabe aus der Pipe sein dürfe. Solange die Daten sequenziell angeliefert werden, sei es unerheblich, ob reale Datei oder Pipe. Mit diesem Wissen ließ sich der Prozess weiter optimieren. Zum Schluss blieb noch ein Befehl übrig *(siehe Listing 5)*.

Das Ergebnis ist überzeugend. Von ehemals 40 Minuten auf ca. 50 Sekunden, das sind nur noch etwa zwei Prozent der Laufzeit gegenüber früher. Um dies zu erreichen, sind ein paar Schritte notwendig. Zunächst ist ein Directory zu erstellen, in dem die Eingabedatei (etwa 'transfer_file_ebcdic.gz') hinterlegt wird: „Create Directory ext_tables as '/home/oracle/external_table';".

Dann ist ein Preprocessor-Skript zu erstellen, das die Unix-Kommandos enthält. Wichtig ist dabei, dass die Unix-Kommandos immer mit vollem Pfadnamen eingetragen sind. Die Fehlermeldungen

```
Merge  into locale_fertigung LF using fertigung F
       on (LF.Satz_id = F.Satz_id)
when matched then
       update set lf.satzart=F.Satzart, lf.Menge=F.Menge
when not matched then
       insert   (satz_id, satzart, werk, teile_id, kunde_id, menge)
       values   (f.satz_id, f.satzart, f.werk, f.teile_id, f.kunde_id,
f.menge)
where F.werk = '2010'
;
```

Listing 8

müssen umgeleitet werden, beispielsweise „2>/dev/null“. In diesem Fall lautet der Name des Preprocessor-Skriptes „bc.sh“ *(siehe Listing 6)*. *Listing 7* zeigt das Erzeugen des „Create Table“ mit den erstellten Informationen über Directory und Preprocessor-Skript. In *Listing 8* ist das Merge-Statement zu sehen.

Ab diesem Zeitpunkt wurde das SQL-Skript nach Bedarf ausgeführt. Nur wenn eine aktuelle Datei angeliefert worden ist, erfolgt eine Verarbeitung. Damit war praktisch jeder Zeitverzug eliminiert.

Fazit

Dieses Verfahren hat sich auch bei vielen anderen Projekten bewährt, so konnte die Laufzeit von Fremddaten-Übernahmen, Laden von DWH-Daten, Zusammenführung verschiedener Eingabedateien etc. wesentlich verbessert werden. Der hauptsächliche Performance-Gewinn kommt daher, dass fast alles im Memory stattfindet und keine Zwischenmengen erzeugt werden, die immer wieder gelesen und geschrieben werden müssen.

Es ist wichtig, die jeweiligen Stärken des Betriebssystems und der Datenbank zu kennen, damit wie in diesem Projekt ein optimales Ergebnis erzielt werden kann. Häufig können Vorarbeiten (etwa mit „awk“, „sed“, „egrep“ etc.) sehr effizient im Betriebssystem vorgenommen werden, was dann zu einer besseren Lade-Laufzeit bei der Datenbank führt.

Bruno Cirone
bruno@cirone.de

Das DOAG Legal Council expandiert

Mit Dr. Ivo Rungg von der Kanzlei Binder Grösswang aus Wien/Innsbruck ist jetzt auch ein Vertreter aus Österreich im DOAG Legal Council aktiv. Die Schwerpunkte von Ivo Rungg sind IP- und IT-Recht sowie Datenschutz mit Themen wie geistiges Eigentum, Lizenz- und Kaufvereinbarungen sowie Datenschutzrecht. Das DOAG Legal Council freut sich, dass Dr. Ivo Rungg hier die rechtlichen Belange der österreichischen Anwender aufnehmen und bewerten kann.

Das DOAG Legal Council ist ein Gremium aus spezialisierten, unabhängigen Rechtsanwälten, das der DOAG in rechtlichen Angelegenheiten zur Seite stehen soll und ihrem Netzwerk fundiertes Fachwissen zur Verfügung stellt. Ob Vertragsgestaltung, Lizenzierung oder Datenschutz – in vielen Bereichen der IT spielen juristische Aspekte eine wesentliche Rolle. Doch das Thema „IT-Recht“ ist für viele IT-Spezialisten schwer zugänglich. Das DOAG Legal Council verrichtet seine Arbeit ehrenamtlich. Es hat sich als Ziel gesetzt, fundiertes Wissen zu rechtlichen Aspekten der IT im DOAG-Netzwerk zur Verfügung stellen, zum Beispiel in Form von Fachartikeln oder Fachvorträge in den eigenen Medien bzw. Veranstaltungen.

Weitere Informationen unter *„https://www.doag.org/de/themen/competence-center/legal-council“*.