Divide et impera

Session-Management im ETL

Sven Bosinger
its-people GmbH
Frankfurt am Main

Schlüsselworte

Data Warehouse, ETL, Oracle Scheduler, Oracle 12c, PL/SQL, DBMS_SCHEDULER, DBMS_ALERT, Oracle-Spatial

Einleitung

Manchmal kann es notwendig sein, innerhalb eines PL/SQL-Packages einen Wechsel des Benutzers oder der Session durchzuführen. Geht nicht? Geht Doch!

Oberflächlich betrachtet stellt PL/SQL keine derartige Möglichkeit zur Verfügung. Bei genauerer Betrachtung, gibt es jedoch eine Reihe von Funktionen, die kreativ angewendet, einen Entwickler in die Lage versetzen einen Sessionwechsel und damit auch einen Benutzerwechsel durchzuführen. Aus dieser Tatsache ergeben sich in der Folge eine Reihe interessanter Anwendungsmöglichkeiten, die hier im Weiteren vorgestellt werden sollen.

Ausgangslage

Der versierte Entwickler wird sich zunächst fragen, wozu überhaupt ein Sessionwechsel notwendig sein soll? Eigentlich sollte über das ausgefeilte Berechtigungs- und Rollen-Konzept der Datenbank ein Benutzer so mit Rechten ausgestattet werden können, dass ein Wechsel nicht notwendig sein sollte. Leider ist es so, dass auch mit einem ausgefeilten Benutzerkonzept bestimmte Tätigkeiten für einen Benutzer verschlossen bleiben.

Erstes Beispiel: Geometrie-Index

Wir betrachten ein Datawarehouse (DWH), welches neben den üblichen skalaren Informationen (VARCHAR, DATE und NUMBER Werte) Geo-Informationen in Form von Oracle-Spatial Objekten beinhaltet. Über einen klassischen ETL-Prozess werden dabei sowohl sogenannte Sachdaten-Tabellen, die skalaren Informationen, als auch Tabellen mit Oracle-Spatial Geometrien bewirtschaftet. Um auf Spatial-Daten mit Geometrie-Funktionen zugreifen zu können, z.B. Abstandsberechnungen, müssen diese mit einem sogenannten Geometrie-Index versehen werden. Dieser benötigt zusätzlich zu seinem Create-Statement ein „Minimum Bounding Rectangle“ (MBR), dass in der Tabelle USER_SDO_GEOMETRY abgelegt wird (siehe Abb. 1).

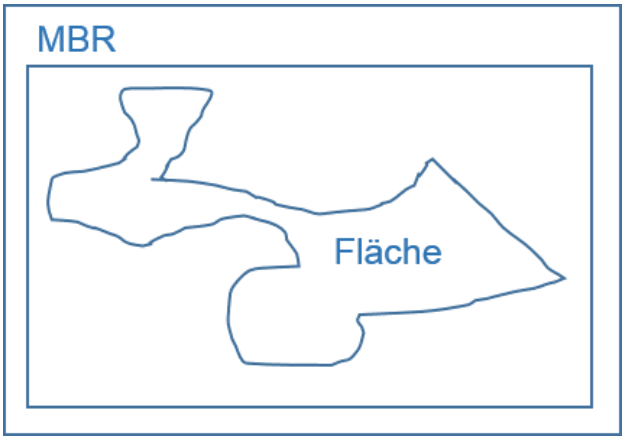

Abbildung 1: Minimum Bounding Rectangle

Werden im Rahmen der Datenbewirtschaftung neue Daten geladen, die eine Erweiterung des MBR bedingen, so muss der Eintrag in der USER_SDO_GEOMETRY entsprechend angepasst werden (siehe Abb. 2).

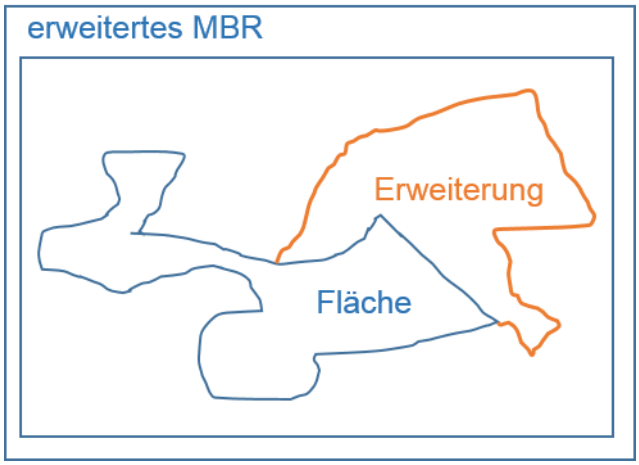

Abbildung 2: Erweitertes Minimum Bounding Rectangle

Der Eintrag in USER_SDO_GEOMETRY ist nur als Table-Owner möglich. Auch mit entsprechenden Rechten auf der Geometrie-Tabelle, ist es einem anderen Benutzer nicht möglich diese Einträge zu ändern.

Da vor einem Massen-Insert in eine Geometrie-Tabelle der Index gelöscht und danach neu aufgebaut werden sollte, ist es hier notwendig, innerhalb des ETL-Programms einen Benutzerwechsel durchzuführen. Dieser kann über einen entsprechenden Sessionwechsel realisiert werden.

Zweites Beispiel: dynamische Segmentierung

Im Rahmen des oben dargestellten ETL-Prozesses werden auf vorgegebenen Streckenlinien Abschnittseigenschaften segmentiert. D.h. man hat eine Streckengeometrie, deren Stützstellen mit Kilometerangaben versehen sind. Man spricht von einer sogenannten „Linear Referencing System“ (LRS) Geometrie. Aus dieser kann man mit den durch Oracle bereitgestellten Spatial-Funktionen

durch Angabe des Von- und Bis-Kilometers Streckenteilabschnitte als eigenständige Geometrie gewinnen. Man nennt dies dynamische Segmentierung (siehe Abb. 3).

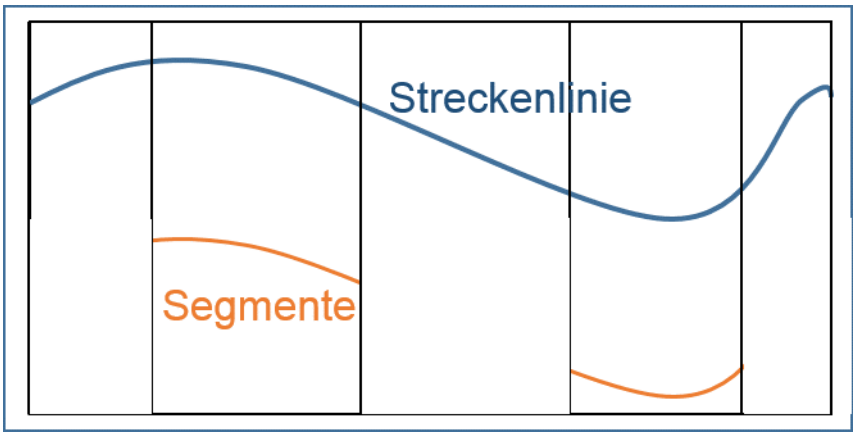

Abbildung 3: Segmente

Diese Tätigkeit kann bei komplexen Linienverläufen sehr Speicherintensiv sein. Insbesondere die PGA ist hier der Engpass. Stellt man sich nun vor, man hat das gesamte Straßennetz Deutschlands als LRS-Geometrie vorliegen und eine List aller Abschnitte (Straßennummer, Richtung und Von- Bis-Kilometer) auf denen eine Geschwindigkeitsbegrenzung von 80 Km/h gilt. Will man für diese Teilabschnitte nun eigene Geometrien erzeugen, so sind viele Tausend dieser dynamischen Segmentierungen notwendig. Dies kann sehr schnell zu einem Überlauf der PGA führen. Leider ist es so, dass die PGA einer Session erst mit Abbau dieser wieder frei gegeben wird. Eine Möglichkeit ist es nun, diese Operationen auf mehrere Blöcke aufzuteilen und jeden dieser Blöcke in einer eigenen Session ablaufen zu lassen. Sodass der PGA-Verbrauch der jeweiligen Session unter dem Maximalwert der PGA bleibt.

Drittes Beispiel: Externe Programme

Auch hier wird wieder unser DWH mit Geo-Daten zugrunde gelegt. Werden die notwendigen Geometrie-Objekte nicht in Oracle-Spatial Format geliefert, so müssen diese umgerechnet werden, so dass sie als Spatial-Daten gespeichert werden können. Diese Umrechnung wird in unserem Fall durch ein externes Programm, dass aus dem Betriebssystem heraus aufgerufen wird, durchgeführt. Dieses Programm bekommt pro Objekt durch ein Vorsystem eine Datei geliefert und kann dieses Objekt als Spatial-Datentyp in einer Oracle-Tabelle speichern. Es werden auf diese Art pro Bewirtschaftungsprozess viele Dateien geliefert. Die Umrechnung einer Datei ist nicht sehr Aufwendig, so dass mehrere Dateien parallel bearbeitet werden können. Leider stell das externe Programm keine eigene Funktion zur Parallelisierung zur Verfügung, so dass diese nur durch x-faches starten des Programms erreicht werden kann (siehe Abb. 4).

Dazu ist es notwendig mehrere Sessions parallel aus einem PL/SQL-Package zu starten und diese später wieder zu synchronisieren.

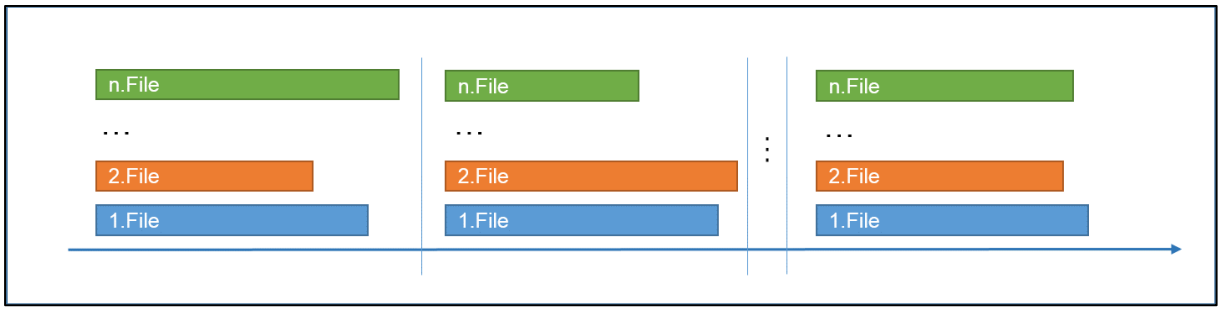

Abbildung 4: externe Programmaufrufe

Man sieht also, ein Session/Benutzerwechsel, kann innerhalb eines PL/SQL-Packages zum Laden von DWH-Daten durchaus sinnvoll sein.

Realisierung

Wie kann man einen Wechsel des Benutzers respektive der Oracle-Session in einem PL/SQL-Package durchführen? Hierzu sollte man sich den Oracle-Scheduler einmal näher ansehen. Neben vielen anderen Funktionalitäten beinhaltet das Package DBMS_SCHEDULER die Prozedur RUN_JOB. Diese kann einen Job starten, nicht nur in der aktuellen Session sondern auch in einer dabei neu initiierten. Ein Job kann dabei entweder der parametrisierte Aufruf einer Prozedur bzw. Prozedur eines Packages sein oder ein Betriebssystembefehl. Wird die Prozedur RUN_JOB so ausgesteuert, dass der auszuführende Job in einer eigenen Session läuft, so kann man auch beeinflussen unter welchem Oracle-Benutzer die Session gestartet wird. Die so gestartete Session läuft parallel zu der (Master-) Session, aus der heraus sie gestartet wurde, ist also asynchron. Um eine spätere Synchronisation der Session zu gewährleisten, kann man das Package DBMS_ALERT nutzen.

Erstes Beispiel: Geometrie-Index

Innerhalb des Package ETL_1, das unter dem Benutzer ETL_USER gestartet wurde, soll nach befüllen der Geometrie-Tabelle GEO_TAB_1, die im Schema GEO_USER liegt der Spatial-Index auf der Geometrie-Spalte GEOMETRY neu angelegt werden. Dazu wird mittels DBMS_SCHEDULER.CREATE_JOB ein neuer Job angelegt, der den die Prozedur zum Anlegen des Index aufruft. Dieser wird dann mit DBMS_SCHEDULER.START_JOB in einer neuen Session, die sich als GEO_USER an die Datenbank anmeldet gestartet. Ist der Index angelegt, so meldet die Prozedur mittels DBMS_ALERT.SIGNAL an die Master-Session, das sie fertig ist. Die Master-Session wartet mit DBMS_ALERT.WAITON solange, bis sie das Fertigstellungssignal erhalten hat und fährt dann mit dem weiteren ETL-Prozess fort (siehe Abb. 5).

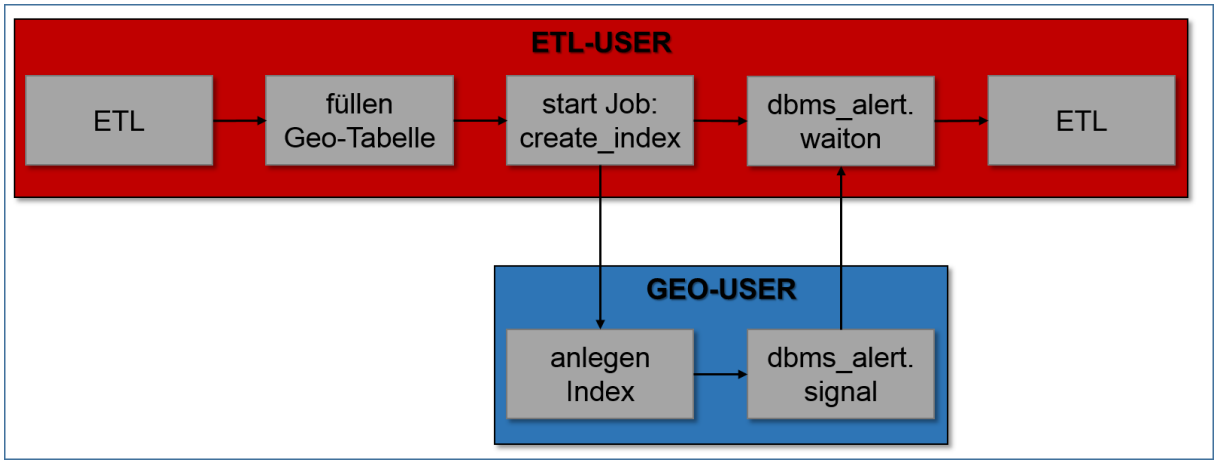

Abbildung 5: Benutzerwechsel

Zweites Beispiel: dynamische Segmentierung

Innerhalb des Package ETL_2, das unter dem Benutzer ETL_USER gestartet wurde, soll eine dynamische Segmentierung der Tabelle GESCHWINDIKGEITEN auf Basis der Geometrie-Tabelle GEO_TAB_2 vorgenommen werden und die Ergebnisse in die Tabelle 80_KMH_STRECKEN eingefügt werden. Dazu wird zunächst mit der analytischen Funktion NTILE die Tabelle GESCHWINDIKGEITEN auf einzelne sogenannte Chunks aufgeteilt. Dabei hat jeder Chunk eine vorher definierte Anzahl von Sätzen, z.B. 1.000. Die obere und untere Grenze des jeweiligen Chunks wird an die Prozedur SEGMENTIEREN übergeben und diese wie im ersten Beispiel in einer eigenen Session gestartet. Die Prozedur SEGMENTIEREN führt nun in einer eigenen Session die Segmentierung der durch die Chunk-Grenzen angegebenen 1.000 Sätze durch und schreibt das Ergebnis in die Tabelle 80_KMH_STRECKEN. Sie meldet ihren Abschluss über DBMS_ALERT, so dass danach der nächste Chunk abgearbeitet werden kann. Jede dieser so durchgeführten PGA-intensiven Segmentierung gibt nun nach Abbau der jeweiligen Session ihren allokierten Speicher wieder frei, so dass der potentielle Speicherüberlauf vermieden werden kann (siehe Abb. 6).

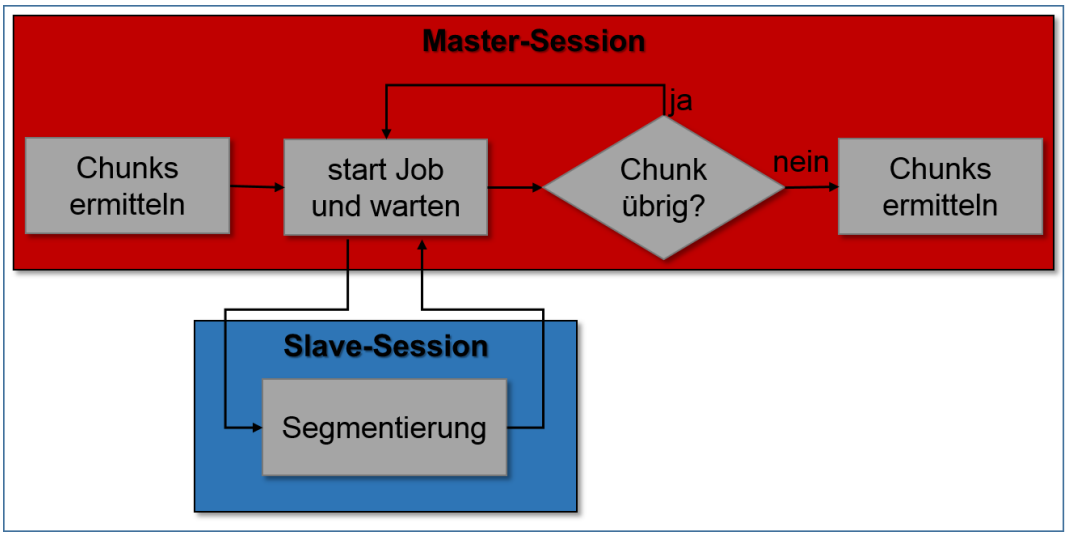

Abbildung 6: dynamische Segmentierung

Drittes Beispiel: Externe Programme

Innerhalb des Package ETL_3, das unter dem Benutzer ETL_USER gestartet wurde, solle mehrere hundert Dateien, die Geometrie-Daten enthalten mittels des externen Programms Datei2Spatial.jar in die Datenbanktabelle GEO_TAB_3 als Spatial-Daten geladen werden. Um die vorhandenen Systemressourcen optimal auszunutzen werden immer 10 Dateien parallel verarbeitet. Sobald einer der Worker-Prozesse fertig ist, meldet sich dieser beim Master-Prozess, so dass sofort ein neuer Worker-Prozess gestartet werden kann, sofern noch nicht bearbeitete Dateien übrig sind. Jeder Worker-Prozess startet eine Instanz des externen Programms und berechnet somit die Spatial-Geometrien für eine Datei.

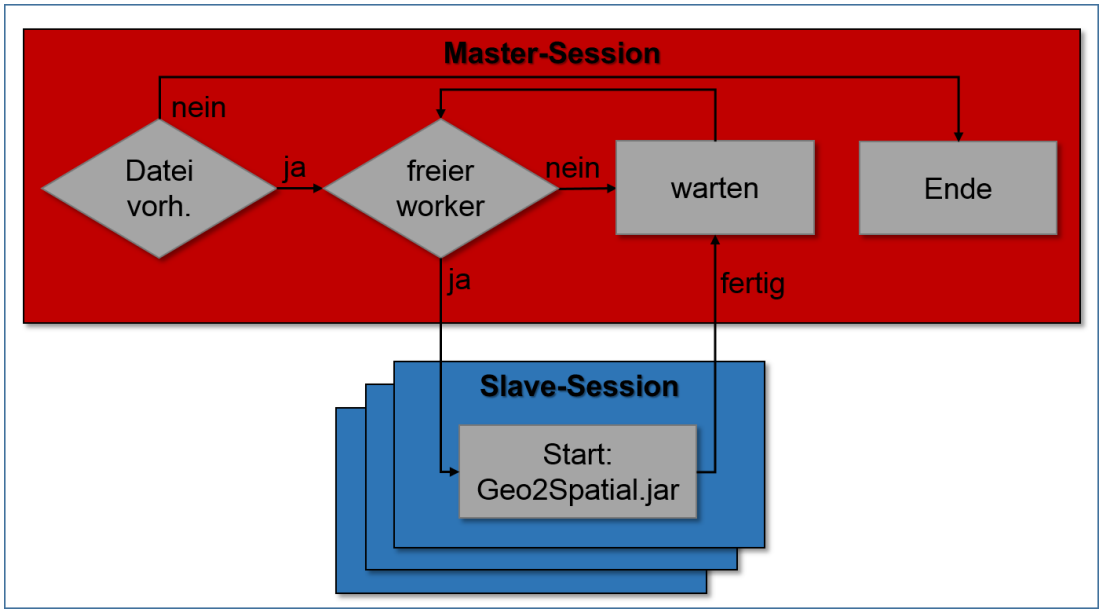

Abbildung 7: Parallelisierung

Fazit

Was hier anhand eines Ladeprozesses für ein Geo-DWH gezeigt wurde, lässt sich problemlos auch auf andere Herausforderungen im ETL anwenden. Immer wenn ein Benutzer-/Sessionwechsel oder eine externe Parallelisierung notwendig ist, kann die Kombination aus DBMS_SCHEDULER und DBMS_ALERT bisher fest geglaubte Grenzen sprengen. Im Praxis-Einsatz, haben sich die eingesetzten Pakete als überaus stabil und robust erwiesen.

Kontaktadresse:

Sven Bosinger
its-people GmbH
Lyoner Str. 44-48
D-60528 Frankfurt am Main

Telefon: +49 (0) 69-2475210-0
E-Mail: sven.bosinger@its-people.de
Internet: www.its-people.de