High Performance Spatial Computing mit Oracle 12c

Eva-Maria Kramer, Disy Informationssysteme GmbH

In der heutigen Zeit sind Datenbanken schon lange keine reinen Daten-Container mehr. Sie können weit mehr als nur schnelle Views und Daten für Drittsysteme bereitstellen. Immer schnellere Systeme und immer größerer Funktionsumfang ermöglichen es, komplexe Projekte komplett in der Datenbank zu bearbeiten. Dieser Artikel zeigt Tipps und Tricks beim Spatial Computing und spricht auch das Thema „High Performance" an.

Anfang 2014 startete das Unternehmen der Autorin ein großes Kundenprojekt, in dem diverse Herausforderungen gemeistert werden mussten. Inhaltlich ging es um die Aufbereitung heterogener Geometrie- und Sachdaten zu einem konsistenten Gesamtdaten-Bestand, der für eine anschließende Weiterverarbeitung in Modellrechnungen benötigt wurde.

Das gesamte Projekt stand unter extremem Zeitdruck, sodass keine klassische Projekt-Durchführung möglich war. Die Daten mussten in sehr kurzer Zeit analysiert und bezüglich ihrer grundsätzlichen Eignung für die Fragestellung qualifiziert werden. Anschließend waren sie attributiv aufzuwerten, mit Blick auf Relevanz zu filtern und räumlich zu verschneiden. Heruntergebrochen in Teilschritte bedeutete dies über 250 Algorithmen. Damit das Projekt in der verfügbaren Zeit möglich war, musste auf getrennte Implementierungs-, Software-QS- und anschließende Berechnungs-Phasen verzichtet werden. Und selbst dann war der Zeitplan nur mit absoluter Top-Performance des Systems und kreativen Lösungen zu stemmen.

Entscheidung für Oracle 12*c* und Solaris

Unter den genannten Umständen kam für die Berechnungen nur eine Lösung direkt in der Datenbank infrage. Damit stand eine schwierige Entscheidung an: Entweder auf die bewährte und vielfach erprobte Funktionalität von Oracle 11*g* setzen oder es mit 12*c* im Release 1 versuchen. Letztendlich sprach die Erwartungshaltung an die bessere Rechenzeit dafür, mit 12*c* ins Rennen zu gehen. Die umfangreichen Optimierungen der gesamten Spatial-Funktionalitäten, und hier besonders die Spatial Vector Acceleration („SPATIAL_VECTOR_ACCELERATION = TRUE;"), gaben den entscheidenden Ausschlag.

Bei einem Projekt dieser Größenordnung ist es zu erwarten, dass man auf Fehler stößt, insbesondere wenn neue Software-Releases zum Einsatz kommen. Die in 12*c* gefundenen Probleme konnten alle durch Hotfixes sehr schnell gelöst oder durch einfache Workarounds umgangen werden.

Ein Beispiel für ein Problem war die Nutzung der Puffer-Funktion „SDO_GEOM.SDO_BUFFER", die eine Puffer-Fläche um existierende Geometrien berechnet. Unter bestimmten Umständen lieferte diese Funktion falsche Ergebnisse. Ein anderes Beispiel war die häufig verwendete Funktion „SDO_AGGR_UNION" zur räumlichen Vereinigung von Geometrien. Wurde diese in Schleifen verwendet, passierte es, dass die Prozedur ohne eine sprechende Fehlermeldung einfach abbrach.

Das zweite zentrale Element zur Einhaltung des Projektplans war das gewählte Server-Setup, auf dem die Datenbank installiert wurde. Für maximale CPU- und I/O-Performance sowie Stabilität setzte man auf Solaris mit ZFS als Dateisystem sowie Solaris Zonen. In der RAM-Disk waren sowohl die komplette Datenbank installiert als auch I/O-kritische Tablespaces abgelegt. Weitere Tablespaces mit normaler I/O-Last waren auf einem Storage abgelegt. Durch ZFS-LZ4-Kompression sowie Hyper-Threading standen dem Projekt somit 500 GB RAM und 60 Kerne zur Verfügung, weitere 1.500 GB auf dem Storage. Durch Snapshots, die auf ein zweites Solaris-System repliziert wurden, war keine Downtime für ein Backup des Systems erforderlich.

Dadurch konnte im Falle eines Fehlers sehr schnell und einfach auf den letzten bekannten Snapshot zurückgegangen werden. Der Zeitverlust durch den Rücksprung auf einen bis zu zwei Tage alten Snapshot wurde durch die schnellere Berechnung mehr als aufgewogen. Entsprechende Mechanismen ermöglichten es, die Oracle-Datenbank ohne Verzögerung in Betrieb zu nehmen – zunächst mit Performance-Einschränkungen; sobald alle Daten auf die RAM Drives migriert waren (ca. 4 Stunden nach einem System-Neustart), stand dann wieder die volle I/O-Performance zur Verfügung.

Dauerhafte Nutzung aller verfügbaren Ressourcen

Um die maximale Rechenleistung zu erreichen, mussten alle verfügbaren CPU-Kerne durchgängig ausgelastet werden. Solange es berechenbare Schritte gab, sollte kein Thread ungenutzt bleiben. Letztendlich blieb man knapp unter diesem Ziel. Von 64 CPU-

Kernen wurden 60 für die Berechnung verwendet. Die restlichen vier blieben dem Betriebssystem und dem Datenbank-System vorbehalten, da es für die Performance von 12*c* nicht förderlich ist, wenn zu viele Threads aus Hintergrund-Prozessen auf den gleichen Kernen laufen.

Für die Ablaufsteuerung ist mit der Disy Spatial Workbench über die Jahre ein eigenes Framework in Java entwickelt worden. Diese verwaltet vollautomatisch alle verfügbaren Threads und kennt die noch offenen Aufgaben mit entsprechenden Abhängigkeiten. Solange rechenbereite Aufgaben vorliegen, werden diese in logisch sinnvoller Reihenfolge gestartet. Im Falle eines Fehlers wird die entsprechende Aufgabe protokolliert und übersprungen, sodass der Thread direkt freigegeben ist. Damit ist es möglich, vom Fehler unabhängige weitere Berechnungen direkt neu zu starten, ohne dass die Threads blockiert bleiben.

Als Trade-off dieses Ansatzes stand die von Oracle über SQL direkt nutzbare parallele Verarbeitung von Statements nicht zur Verwendung. Damit hätte man die Kontrolle über die Threads aus der Hand gegeben, was in Summe wahrscheinlich Rechenzeit gekostet, auf jeden Fall aber die Zahl der grauen Haare in die Höhe getrieben hätte.

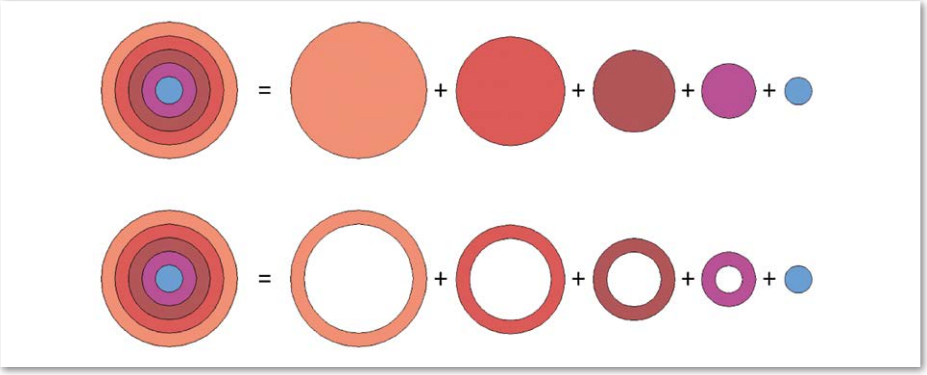

Abbildung 1: Isophonen-Geometrien als Flächen (oben) und Bänder (unten)

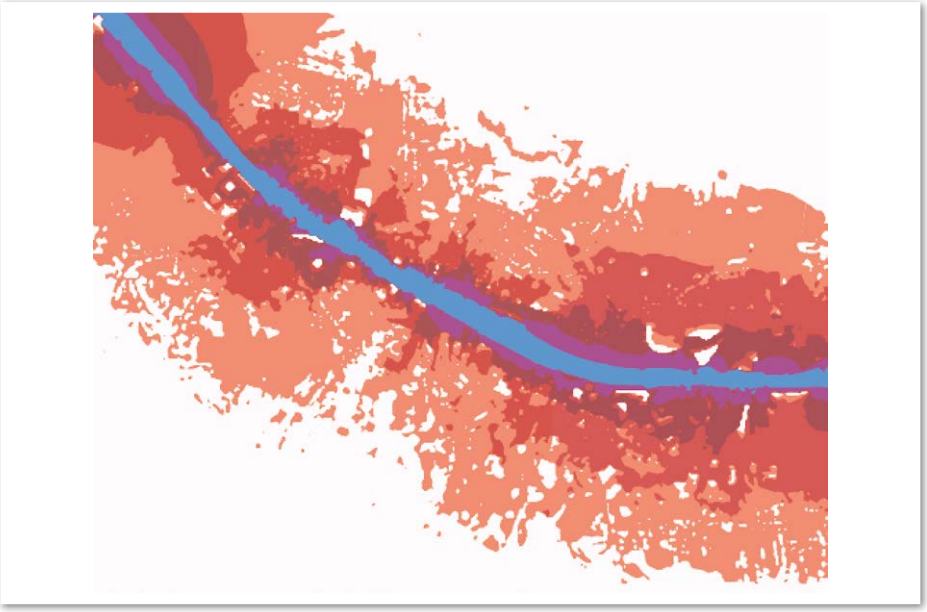

Abbildung 2: Komplexe Isophonengeometrien

Beispiel aus der Praxis

Im Rahmen des Projekts zeigte sich wieder eindrucksvoll, dass auch das beste Setup nichts nützt, wenn die einzelnen Abfragen unperformant geschrieben sind oder der Optimizer überfordert ist. Gerade die Verarbeitung von räumlichen Daten bietet hier einige Herausforderungen. Im Folgenden ein Beispiel dafür, mit welchen Strategien man auch bei komplexen geometrischen Aufgaben mit 12*c* ans Ziel kommen kann.

Das Ergebnis der Modellierung waren sogenannte „Isophonen", also Flächen (Geometrien) mit gleicher Lautstärke. Isophonen können als Flächen oder als Bänder vorliegen *(siehe Abbildung 1)*. Die Flächen zeigen je Pegelklasse die Bereiche, in denen bestimmte Lautstärken erreicht sind (Mindest-Lautstärke). Die Bänder dagegen zeigen jeweils die betroffene Fläche eines Lautstärke-Intervalls an. Im Projekt mussten beide Arten der Darstellung erzeugt werden, sowohl Flächen als auch Bänder.

Modellierte Isophonen zeichnen sich in der Regel durch extrem viele Stützpunkte aus, da die Geometrien häufig sehr groß und gleichzeitig sehr komplex sind *(siehe Abbildung 2)*. Viele Stützpunkte bedeuten mehr Speicherbedarf, vor allem jedoch längere Laufzeiten bei räumlichen Abfragen beziehungsweise Operationen. Bei Bändern ist die Anzahl der Stützstellen durch die inneren Ringe nochmals deutlich größer als bei den Flächen.

Aus dem Projekt resultierten knapp 400.000 Isophonen-Flächen mit in Summe 95 Millionen Stützpunkten. Hieraus mussten fast 200 Millionen Isophonen-Bänder berechnet werden, die insgesamt über 150 Millionen Stützpunkte aufwiesen. Der erste Ansatz, um aus den Flächen-Geometrien Bänder-Geometrien zu generieren, war ein kurzes Statement *(siehe Listing 1)*. Darin wurden von den Flächen-Geometrien alle im gleichen räumlichen Gebiet liegenden Flächen mit höheren Lärmwerten abgezogen. Sachdaten sowie Identifier blieben erhalten.

Für sehr kleine Datenmengen ist ein solches Statement vertretbar, für größere ist dieser Ansatz nicht praktikabel. Selbst wenn das Statement nach Wochen erfolgreich abgeschlossen worden wäre (und man diese Zeit gehabt hätte), wären drei große Nachteile geblieben. Erstens muss die gesamte Konfiguration der Redo-Logs auf lang laufende Abfragen ausgelegt sein, mit allen Nachteilen, die man hierdurch in Kauf nehmen muss. Zweitens müsste im Falle eines Abbruchs (verlorene Connection, Stromausfall) nach dem Redo das gesamte Statement nochmals von vorne verarbeitet werden. Und drittens hat man mit solch einem Statement keinerlei Anhaltspunkt, wie lange die Verarbeitung noch dauert. Die erste Optimierung bestand darin, eins nach dem anderen zu tun *(siehe Listing 2)*.

Die Berechnung erfolgte nicht mehr in einer einzigen Transaktion in der Datenbank. Vielmehr wurde in einer Schleife über alle Flächen iteriert. Jede berechnete Fläche wurde direkt der Ergebnis-Tabelle hinzugefügt. Damit bot sich die Möglichkeit, dass im Falle eines Abbruchs das Statement jederzeit für alle noch nicht verarbeiteten Geometrien wiederaufgenommen werden konnte. Schon bearbeitete Datensätze blieben erhalten.

Gleichzeitig war jederzeit transparent, wie viele Datensätze bereits verarbeitet wurden, und somit eine zeitliche Hochrechnung möglich. Die technischen Probleme des ersten Ansatzes waren damit adressiert. Für den engen Zeitplan war die Laufzeit des Statements jedoch immer noch nicht hinreichend. Die Lösung: Teile und herrsche *(siehe Listing 3)*.

In einem ersten Schritt wurden alle innersten Geometrien, also die lautesten Flächen, direkt in die Ergebnis-Tabelle für Bänder geschrieben. Hier gab es keinen Unterschied von Fläche zu Band. Anschließend verarbeitete man die restlichen Geometrien mit einer doppelten Schleife. Der Optimizer von Oracle ist an dieser Stelle nicht in der Lage, den geometrischen Index für eine Vorselektion für die Funktion „SDO_RELATE“ zu nutzen. Stattdessen musste eine Vorselektion über das vorberechnete minimal umgebende Rechteck (engl.: MBR) erfolgen, was ein Vielfaches an Rechenzeit einsparte. An dieser Stelle ein Tipp: In 12*c* gibt es gegenüber 11*g* eine verbesserte und schnellere Implementierung dieser Funktion: „SDO_GEOM_MBR“ statt „SDO_GEOM.SDO_MBR“.

Die Aufgabe der äußeren Schleife bestand darin, alle noch zu prozessierenden Flächen zu selektieren und das MBR zu erstellen. Die innere Schleife iterierte dann über alle Objekte, die das MBR der äußeren Schleife berührten. Es wurden also nur die Geometrien rechenintensiv miteinander verschnitten, die sich gemäß der Vorselektion anhand der räumlichen Lage auch potenziell überlagerten. Neben den Vorteilen der ersten Optimierung war mit dieser Anweisung auch das Laufzeit-Problem adressiert.

Fazit

Mit diesem Setup wurde im letzten Jahr das genannte Projekt sehr erfolgreich abgeschlossen. Durch die Kombination der

```
CREATE TABLE Isophonen_Baender AS
SELECT o.Sachdaten,
   o.Pegel,
   SDO_GEOM.SDO_DIFFERENCE (o.Geom,
       (SELECT SDO_AGGR_UNION (SDOAGGRTYPE (i.Geom, 0.05))
           FROM Isophonen_Flaechen i
       WHERE i.Sachdaten = o.Sachdaten AND i.Pegel > o.Pegel),
           0.05) Geom
   FROM Isophonen_Flaechen o;
```

Listing 1

```
DECLARE
BEGIN
   FOR c IN (SELECT Sachdaten, Pegel, Geom FROM Isophonen_Flaechen)
   LOOP
       INSERT INTO Isophonen_Baender (Sachdaten, Pegel, Geom)
           SELECT o.Sachdaten,
               o.Pegel,
               SDO_GEOM.SDO_DIFFERENCE (o.Geom,
                   (SELECT SDO_AGGR_UNION (SDOAGGRTYPE (i.Geom, 0.05))
                       FROM Isophonen_Flaechen i
                       WHERE i.Sachdaten = o.Sachdaten AND i.Pegel > o.Pegel),
                   0.05) Geom
           FROM Isophonen_Flaechen o
           WHERE c.Sachdaten = o.Sachdaten AND c.Pegel = o.Pegel;
       COMMIT WRITE NOWAIT;
   END LOOP;
END;
/
```

Listing 2

```
DECLARE
   v_band_geom MDSYS.SDO_GEOMETRY;
BEGIN
   INSERT INTO Isophonen_Baender (Sachdaten, Pegel, Geom)
       SELECT Sachdaten, Pegel, Geom FROM Isophonen_Flaechen WHERE Pegel =
75;
   COMMIT WRITE NOWAIT;

   FOR o IN (SELECT Sachdaten, Pegel, Geom, SDO_GEOM_MBR(Geom) Mbr FROM Iso-
phonen_Flaechen WHERE Pegel != 75)
   LOOP
       v_band_geom := o.Geom;
       FOR i IN (SELECT Geom FROM Isophonen_Flaechen
           WHERE  Sachdaten = o.Sachdaten AND Pegel > o.Pegel
                  AND SDO_RELATE(Geom, o.Mbr, ,mask=anyinteract') = 'TRUE'
           ORDER BY Pegel desc)
       LOOP
           v_band_geom := SDO_GEOM.SDO_DIFFERENCE(v_band_geom, i.Geom, 0.05);
       END LOOP;

       INSERT INTO Isophonen_Baender (Sachdaten, Pegel, Geom)
           VALUES (o.Sachdaten, o.Pegel, v_band_geom);
       COMMIT WRITE NOWAIT;
   END LOOP;
END;
/
```

Listing 3

stabilen Datenbank-Version 12*c* und der enormen Verbesserungen ihrer Performance für räumliche Abfragen gegenüber der Version 11*g* mit dem gewähltem Hardware-Setup war man in der Lage, sämtliche Ergebnisse in Quality, in Budget und vor allem auch in Time abzuliefern.

Die von Oracle bereitgestellten räumlichen Funktionen (Locator sowie Spatial-Option) sind ausgereift und vollumfänglich, auch die mit 12*c* teilweise komplett neu geschriebenen Funktionen sind inzwischen stabil. Bei einigen Funktionen muss der Anwender sich jedoch über die Besonderheiten oder den offiziellen Funktionsumfang der Funktionen im Klaren sein. Der Umgang der gleichen Funktion mit 2D- und 3D-Daten ist zwischen den Versionen zum Teil unterschiedlich.

Bei der Implementierung von Abfragen oder PL/SQL-Routinen lohnt sich immer wieder einen Blick in die Dokumentation oder das Change-Log. Mit 12*c* wurde zum Beispiel das bereits erwähnte „SDO_GEOM.SDO_MBR" durch „SDO_GEOM_MBR" ergänzt, wobei Letzteres ein deutlich schnelleres Resultat liefert. Die neue Funktion „POINTIN_POLYGON" ist in der Lage, extrem schnell Punkte mit Polygonen zu korrelieren. Absolutes Highlight für die Berechnung war aber das „SPATIAL_VECTOR_ACCELERATION"-Flag. Die Geschwindigkeitsgewinne durch diese Funktion sind wirklich enorm.

Wenn der Projektzeitplan ein wenig lockerer gewesen wäre, hätte die Autorin große Freude daran gehabt, die diversen Berechnungen vergleichsweise auch auf einer Version 11*g* auszuführen, um den Unterschied zwischen 11*g* und 12*c* im Gesamten zu sehen. Für ihren Teil möchte sie 12*c* für Spatial-Processing-Projekte nicht mehr missen.

Die finalen Ergebnisse des Kundenprojekts und vor allem die Isophonen können online auf den Seiten des Eisenbahn-Bundesamtes eingesehen werden *(siehe „http://tinyurl.com/pvyt59d")*.

Eva-Maria Kramer
eva.kramer@disy.net

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

Ingo Jannick
Marc Steinhaus
Kai Weber
Stefan Kopp
Christine Wege
Joern Kleinbub
Christoph Pfrommer
Georghe Breazu
Jörg Vargel
Thilo Jäger
Uwe Hofmann
Florian Giesecke-Uellner
Sven Ruppert
Ronald Stöcken
Ulrich Lickert
Ivan Korac
Thomas Rein

Firmenmitglieder DOAG

Johannes Bayer-Albert, byteletics oHG
Gerhard Lutz, Stadtwerke Augsburg Hold. GmbH
Thomas Schlenker, Sage bäurer GmbH
Oliver Pliquett, SupplyOn AG
Roland Junge, Toll Collect GmbH

Neumitglieder SOUG

Pierre-Jean Giraud, Giraud Adequations
Daniel Meienberg, Diso AG
Ivan Korac, Altran
Mariusz Zuk, upc cablecom