# Optimale Performance mit Oracle Spatial auf Exadata

**Hans Viehmann**
**ORACLE Deutschland B.V. & Co. KG**
**Hamburg**

**Schlüsselworte:**

Oracle Spatial, Exadata, Performance, Parallelisierung, Tuning.

## Einleitung

Mit der zunehmenden Verbreitung von Exadata Database Machines steigt auch die Anzahl derjenigen, die auf diesen Systemen raumbezogene Daten mittels Oracle Spatial verarbeiten. So wurden in den letzten Monaten beispielsweise eine große Telematik-Plattform, eine Lösung für die Versicherungswirtschaft, oder auch eine Anwendung zur Speicherung und Analyse von Satellitendaten auf dieser Technologie implementiert.

Als integraler Bestandteil des Datenbank-Kernels profitieren die raumbezogenen Abfragen und DML-Operationen naturgemäß ebenso von der Performance der Exadata Database Machine, wie alle anderen Funktionalitäten der Datenbank auch. Es gibt darüber hinaus aber eine Reihe Einsatzfälle, bei denen die Funktionalitäten von Oracle Spatial von der Funktionalität des Systems besonders profitieren. Einige dieser Use Cases sind im Folgenden näher erläutert.

## Exadata Database Machine und Oracle Spatial

Die Exadata Database Machine ist in Vorträgen bereits verschiedentlich behandelt worden, daher soll sie an dieser Stelle nur kurz im Zusammenhang mit Oracle Spatial dargestellt werden. Als massiv paralleles System mit bis zu 128 Cores in der größten Ausbaustufe ist sie zum einen für CPU-intensive Funktionalitäten, wie geometrische oder geodätische Berechnungen speziell geeignet, insbesondere dann, wenn sich Operationen gut parallelisieren lassen. Von dem schnellen Infiniband-Netzwerk innerhalb der Maschine profitieren darüber hinaus vor allem diejenigen Prozesse, die große Datenmengen bewegen, wie etwa beim Laden oder der initialen Aufbereitung raumbezogener Daten. Die Verteilung der Verarbeitungslogik auf Server-Knoten und Storage Heads ermöglicht auch für Geodaten eine geschickte Aufteilung der Last. Für die laufend stattfindende Optimierung des R-Tree Index ist eine große SGA sehr nützlich, weil dann die Indexblöcke im Memory vorgehalten werden können. Da die Exadata insgesamt bis zu 2 TB RAM enthalten kann, profitieren diese Operationen ebenfalls überproportional. Und schließlich hilft der Einsatz von Flash Cache an den richtigen Stellen, für die Performance insgesamt.

Der wesentliche Vorteil eines solchen „Engineered System“ besteht allerdings darin, dass die auf den Betrieb von Oracle Datenbanken optimierte Maschine bereits vorkonfiguriert ist und damit die Kosten für Konzeption und Implementierung deutlich reduziert, sowie der Zeitaufwand für die Einrichtung und das Risiko minimiert werden können.

Im folgenden wird eine Reihe von Einsatzszenarien vorgestellt, die auf der Arbeit an konkreten Projekten beruhen.

**Einsatzfall: Räumliche Abfragen auf partitionierten Tabellen**

In diesem Falle sollten für einen Rückversicherer Analysen zum Risikomanagement auf dessen Bestandsdaten durchgeführt werden. In dem konkreten Szenario ging es darum, aus dem Gesamtbestand der Verträge von hunderten Versicherern festzustellen, welche Gebäude innerhalb der Schneise eines Hurricans liegen.

Die gesamte Tabelle bestand aus etwa 36 Mio. Zeilen, aufgeteilt auf 64 Partitionen mit je über einer halben Million Einträgen. Gegenüber der seriellen Ausführung, die 50s in Anspruch nahm, ließ sich die Abfrage auf 1.28s beschleunigen. Grund dafür ist die optimierte Ausführung räumlicher Operatoren auf partitionierten Tabellen in Kombination mit der ausgezeichneten Parallelisierung innerhalb der Exadata. So werden vom Optimizer zunächst die von der Abfrage nicht berührten Partitionen herausgefiltert und anschließend die räumliche Abfrage auf den verbleibenden Partitionen parallel ausgeführt. Dieses Vorgehen trifft im übrigen auf alle räumlichen Operatoren zu.

**Einsatzfall: Parallelisierung räumlicher Abfragen mittels „Create-Table-As-Select“ (CTAS)**

Falls die raumbezogenen Daten nicht partitioniert vorliegen, lässt sich Parallelisierung über „Create-Table-As-Select“ Statements erreichen. Für eine Telematikanwendung im Bereich einer Bahngesellschaft bestand die Anforderung, laufend die Position aller Züge im Schienennetz zu verwalten. Dazu sind alle Lokomotiven mit einem GPS-Empfänger ausgestattet und übermitteln ständig ihre Position. Da GPS aber ungenau ist, sollte zu jedem Koordinatenpaar mittels Projektion die Position der Lok entlang der Strecke bzw. entlang des am nächsten liegenden Gleises bestimmt werden. Für diesen Zweck ist im Package SDO_LRS die Funktion FIND_MEASURE vorhanden. Sobald das zweite Argument des räumlichen Operators nicht ein einzelner Wert ist, wird die Funktion parallel ausgeführt, sodass sich mit dem folgenden Statement die beste Performance erzielen ließ:

```
CREATE TABLE results NOLOGGING PARALLEL 4
 AS SELECT /*+ ordered */
       a.locomotive_id,
       sdo_lrs.find_measure (b.track_geom, a.locomotive_pos) measure
    FROM locomotives a,
         tracks b
    WHERE sdo_nn (b.track_geom, a.locomotive_pos, 'sdo_num_res=1')='TRUE';
```

Diese Möglichkeit der Parallelisierung gilt auch für alle anderen räumlichen Operatoren, wie SDO_WITHIN_DISTANCE, SDO_TOUCH, usw.

**Einsatzfall: Räumliche Korrelation mittels SDO_JOIN**

In diesem Einsatzfall sollten für einen Anbieter von Satellitenbildern diejenigen Teile der Aufnahmen extrahiert werden, die nicht durch Wolken abgedeckt wurden. Die Umrisse der Wolkenflächen lagen in Form von 58 Mio. Geometrien vor, der Bildbestand selber umfasste 850 000 Aufnahmen. Für diese Art des Vergleichs von vielen Geometrien in einem Layer mit ebenfalls vielen Geometrien in einem anderen Layer kann das räumliche Kreuzprodukt SDO_JOIN eine sehr effektive Vorgehensweise sein, weil es jeweils den räumlichen Index beider Layer nutzt. So werden im ersten Schritt zunächst paarweise diejenigen Indexkacheln in den beiden Indices bestimmt, die miteinander in Beziehung stehen und anschließend aus den dabei gefundenen Kandidaten die eigentlichen Ergebnisse bestimmt.

Im folgenden sind die Abfragen nochmal anhand des oben beschriebenen Risikomanagement-Szenarios dargestellt, wo alle Flurstücke (Parcels) berechnet werden sollten, die innerhalb (genauer: **mask=anyinteract**) einer der zahlreichen Risikogebiete (Risk Zones) liegen:

```
SELECT /*+ ordered */ b.risk_zone_id, c.parcel_id
      FROM TABLE (SDO_JOIN ('RISK_ZONES', 'GEOM','PARCELS', 'GEOM',
                  'mask=anyinteract')) a,
                   risk_zones b,
                   parcels c
      WHERE a.rowid1 = b.rowid
        AND a.rowid2 = c.rowid;
```

Für die gleiche Fragestellung lässt sich auch eine alternative Vorgehensweise sehr effektiv einsetzen, die aus zwei separaten Abfragen besteht. Im ersten Schritt verwendet man wiederum SDO_JOIN, allerdings nur auf Basis des Primärfilter-Mechanismus von Oracle Spatial. Hier werden ausschließlich die beiden räumlichen Indices zum Join herangezogen, sodass die Abfrage sehr schnell erfolgt, im Gegenzug aber kein exaktes Ergebnis zurückliefert, sondern mehr mögliche Kandidaten. Diese Paare von Flurstücken und Risikogebieten werden dann in einer zweiten Query mittels der Funktion SDO_GEOM.RELATE wiederum unter Ausnutzung der Parallelisierung einer genauen Analyse unterzogen.

Damit sehen die Statements wie folgt aus:

```
ALTER SESSION ENABLE PARALLEL DDL;
ALTER SESSION ENABLE PARALLEL DML;
ALTER SESSION ENABLE PARALLEL QUERY;

CREATE TABLE result1 NOLOGGING AS
      SELECT a.rowid1 AS risk_zones_rowid,
             a.rowid2 AS parcels_rowid
      FROM TABLE ( SDO_JOIN ('RISK_ZONES','GEOM','PARCELS', 'GEOM'));
```

für den Primärfilter und anschließend:

```
CREATE TABLE result2 NOLOGGING PARALLEL 32 AS
      SELECT /*+ ordered use_nl (a,b) use_nl (a,c) */
          sdo_geom.relate (b.geom, 'DETERMINE', c.geom, .05) relation,
          b.risk_zone_id,  c.parcel_id
      FROM result1 a, risk_zones b, parcels c
      WHERE a.risk_zones_rowid = b.rowid
           AND a.parcels_rowid = c.rowid;
```

Je nach fachlicher Anforderung kann die Funktion SDO_GEOM.RELATE dazu genutzt werden, die geometrische Beziehung zwischen den Paaren von Geometrien zu bestimmen („DETERMINE“), oder die gesuchte Beziehung wird vorgegeben (z.B. „TOUCH“ oder „INSIDE“).

**Einsatzfall: Laden großer Datenmengen mit Raumbezug**

Für einen Kunden im Umfeld meteorologischer Daten sollte gezeigt werden, wie schnell mit Hilfe einer Exadata Wetterdaten in eine partitionierte Tabelle geladen werden können. Im einfacheren Falle sollten alle 30 Sekunden 270000 Messwerte in eine Tabelle geladen werden, die zu jedem Zeitpunkt die gesammelten Daten von zwei Tagen vorhielt. Da kein globaler Primärschlüssel vorhanden war, ließ sich die Anforderung recht simpel durch den Aufbau einer temporären Tabelle umsetzen, die den

Datenbestand von 30s enthielt und anschließend mittels ALTER TABLE ... EXCHANGE PARTITION ... INCLUDING INDEXES in die Haupttabelle eingepflegt wurde. Die Erzeugung des räumliche Index auf der temporären Tabelle lässt sich parallelisieren und der Index selbst bleibt beim Austausch der Partition erhalten.

Etwas aufwändiger ist die Vorgehensweise, wenn ein globaler Primärschlüssel vorhanden ist, etwa um über die gesamten Partitionen hinweg referentielle Integrität sicherstellen zu können. Damit auch in solchen Szenarien Bulk Loading genutzt werden kann, geht man grundsätzlich ähnlich vor, wie im vorherigen Abschnitt: Man füllt eine temporäre Tabelle mittels direct path loader, legt Primärschlüssel und ggf. foreign key constraints an und tauscht dann die Partition per EXCHANGE PARTITION aus, wobei hier die Option UPDATE GLOBAL INDEXES erforderlich ist:

```
ALTER TABLE weather_data_part EXCHANGE PARTITION p1
WITH TABLE new_weather_data
WITHOUT VALIDATION
UPDATE GLOBAL INDEXES;
```

Anschließend ist ein Index-Rebuild für diejenigen lokalen Indices erforderlich, die hierbei ungültig geworden sind. In dem vorliegenden Beispiel nahm dieser gesamte Prozess herkömmlich 8 Stunden in Anspruch. Auf der Exadata reduzierte sich die Zeit auf 5 Minuten.

**Zusmmenfassung**

Mit der Exadata Database Machine lassen sich sehr anspruchsvolle Anwendungen für die Verwaltung und Analyse raumbezogener Daten implementieren. Insbesondere die Parallelisierung von Joins, Operatoren und räumlichen Abfragen bringt auf dieser Hardware große Vorteile in Bezug auf Skalierbarkeit und Durchsatz. Dank der extrem hohen Performance des Infiniband-Netzwerks innerhalb der Maschine und der hohen Leistungsfähigkeit der Storage Server, sowie der großzügigen Ausstattung mit Hauptspeicher lassen sich derartige rechenintensive Anwendungen ausgezeichnet umsetzen, und zwar ohne dass die Implementierung spezifisch auf die Exadata angepasst werden müsste.

**Kontaktadresse:**

**Hans Viehmann**
ORACLE Deutschland B.V. & Co. KG
NL Hamburg
Kühnehöfe 5
D-22761 Hamburg

Telefon: +49(0)40-89091-173
Fax: +49(0)40-89091-250
E-Mail hans.viehmann@oracle.com
Internet: www.oracle.com/de