Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Verteilte Datenbanken

- Übung 7 -
Ausgewählte
Oracle-eigene Features

Harm Knolle

- **Abgabe Übung 7 bis Montag, 28. Juni 2015, 24:00 Uhr**

DBV_Übung_7 vom 19.06.2015 15:41

Druck vom 19.06.2015 16:01

Prof. Dr. H. Knolle
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Grantham-Allee 20
53757 Sankt Augustin

Inhaltsverzeichnis

1 Partitionierung

Im Allgemeinen versteht man unter der Partitionierung den Spezialfall einer horizontalen Fragmentierung. Ähnlich der Fragmentierung lassen sich die Daten einzelner Partitionen direkt oder aber vereinigt unter einem einheitlichem Tabellennamen ansprechen. Die Fragmente der Partitionierung werden allerdings weniger im Hinblick auf eine anschließende räumliche Verteilung auf unterschiedliche und autonome Datenbanksysteme erstellt, sondern dienen vielmehr einer Optimierung der Datenverwaltung großer Datenmengen im Rahmen eines zentralisierten Datenbanksystems. Dort lassen sich die Partitionen unterschiedlichen und auch räumlich getrennten sekundären Speichermedien zuordnen. Werden die Daten einer Partition verändert, so können die Daten einer anderen Partition parallel gelesen oder reorganisiert werden. Auch können einzelne Partitionen im Fehlerfall separat wiederhergestellt werden. Eine echte Verteilung im Sinne einer Zuordnung einzelner Partitionen in den Kontrollbereich unterschiedlicher Datenbankmanagementsysteme wird jedoch nicht unterstützt.

a) Wie könnte man sich ein SQL-Konzept vorstellen bzw. was muss ein Datenbankmanagementsystem letztendlich leisten, wenn eine echte Verteilung von Tabellenfragmenten auf unterschiedliche Datenbankmanagementsystem unterstützt werden soll und gleichzeitig die 12 Regeln von Date beachtet werden (siehe Kapitel 3 im Skript zur Lehrveranstaltung)?

b) Für welche Art von Anwendungen wird sich das Konzept der Partitionierung besonders eignen?

1.1 Partitionierte Tabellen in Oracle

Zahlreiche Anbieter von Datenbankmanagementsystemen bieten die Partitionierung an. Es muss jedoch beachtet werden, dass Nutzung und Anwendung nicht immer dem ANSI-SQL-Standard entsprechen (Oracle schreibt in seinen Handbüchern „[…] For application portability and ANSI syntax compliance, Oracle strongly recommends that you use views to insulate applications from this Oracle proprietary extension. …[...]").

Zur Vertiefung und Anwendung der im Folgenden vorgeschlagenen Konzepte lesen Sie bitte in der Oracle-Dokumentation nach (insbesondere VLDB and Partitioning Guide: http://docs.oracle.com/database/121/VLDBG/toc.htm).

1.2 Vorbereitung

Im folgenden soll das Konzept der Partitionierung anhand der bereits bekannten Zerlegung der Daten nach „`ID_depot`" angewendet werden. Da momentan drei unterschiedliche Depots vorhanden sind, sollen zunächst drei Partitionen („`bonn`", „`london`" und „`newyork`") erstellt, deren Daten drei getrennten physikalischen Speicherbereichen zugeordnet werden. Ein Speicherbereich zur Aufnahme von Daten wird in Oracle Tablespace genannt. Ein Tablespace

besteht letztendlich aus einer oder mehreren Dateien, die wiederum vom Dateisystem verwaltet werden. Der Administrator hat zu diesem Zweck die drei Tablespaces

- `tbs_bonn`
- `tbs_london`
- `tbs_newyork`

zur Verfügung gestellt. Wie bereits erläutert, wird ein Tablespace einem Datenbankmanagementsystem fest zugewiesen. Ein verteilter direkter Zugriff von unterschiedlichen Datenbankmanagementsystemen auf einen Tablespaces ist nicht möglich. Daher soll im folgenden auf dem System „Bonn“ gearbeitet werden.

Frage: Warum handelt es sich nicht um ein verteiltes Datenbanksystem, wenn die drei Tablespaces jeweils physikalisch verteilt in Bonn, London bzw. New York vom dortigen Dateisystem verwaltet werden?

1.3 Erstellung der partitionierten Tabellen

Nutzen Sie zu diesem Zweck die „…`PARTITION BY LIST` …“-Klausel der „`CREATE TABLE` …“-Anweisung.

a) Partitionieren Sie zunächst die Tabelle, in der das Partitionierungskriterium „ID_depot“ initial definiert wird.

   a. Damit am Standort Bonn die bereits vorhandenen Schema-Objeke mit identischen Namen nicht gelöscht werden müssen, erzeugen Sie bitte Tabellen mit neuen Namen. Hier bietet sich z.B. der Prefix „`part_`“ an.

   b. Erzeugen Sie die drei Partitionen „`bonn`“, „`london`“ und „`newyork`“.

   c. Weisen Sie der Partition den dafür eingerichteten physischen Speicherbereich (Tablespace) zu.

   d. Weisen Sie der so erstellten partitionierten Tabelle die entsprechenden und bekannten Datensätze zu.

b) Testen Sie die erfolgreiche Partititionierung, indem Sie bei der Selektionsanweisung lediglich die Daten einer Partition („`bonn`“, „`london`“ und „`newyork`“) qualifizieren, ohne hierbei eine „`WHERE`“-Bedingung zu benutzen (direkter Zugriff auf die Tabellendaten einer Partition).

1.4 Erstellung der abhängig-partitionierten Tabellen

Die Partitionierung einer Tabelle ist natürlich dann besonders sinnvoll, wenn man - analog zur abhängigen Fragmentierung - die von einer partitionierten Tabelle abhängigen Daten anderer Tabellen ebenfalls derselben Partition zuweisen kann. Das soll auch dann möglich sein, wenn

die über Fremdschlüssel abhängigen Tabellen nicht über das Partitionierungs-Attribut verfügen. Nutzen Sie zu diesem Zweck die „… `PARTITION BY REFERENCE` …“-Klausel der „`CREATE TABLE` …“-Anweisung.

a) Nutzen Sie das Konzept der abhängigen Partitionierung, damit die von der oben bereits partitionierten Tabelle „direkt“ abhängigen Daten anderer Tabelle den jeweils gleichen Partitionen zugeordnet werden.

    a. Bitte beachten Sie, dass die abhängige Partitionierung den Namen des Fremdschlüssel-Constraints benötigt. Der Fremschlüssel darf daher nicht nachträglich, wie in der Software-Technologie üblich, in einem separaten „ALTER TABLE …“-Statement definiert werden.

    b. Bitte beachten Sie, dass „alle“ Datensätze der abhängigen Tabelle auch tatsächlich „abhängig“ sein müssen. Das gilt auch für zukünftige Datensätze. Ergründen Sie, warum die Schemadefinition der abhängigen Tabelle diesbezüglich angepasst werden muss. Ansonsten wird die Erstellung der abhängigen Tabelle vom System mit einem Fehler abgewiesen.

    c. Weisen Sie der so erstellten direkt abhängig partitionierten Tabelle die entsprechenden und bekannten Datensätze zu.

b) Testen Sie die erfolgreiche Partititionierung, indem Sie bei der Selektionsanweisung lediglich die Daten einer Partition („`bonn`“, „`london`“ und „`newyork`“) qualifizieren, ohne hierbei eine „`WHERE`“-Bedingung zu benutzen (direkter Zugriff auf die Tabellendaten einer Partition). Testen Sie hierbei auch den Join.

c) Partitionieren Sie nun die dritte und „indirekt“ vom Partitionierungs-Attribut „ID_depot“ abhängige Tabelle.

    a. Gehen Sie hierbei wie oben vor. Beachten Sie bitte, dass die dritte Tabelle eine weitere Abhängigkeit zu einer anderen Tabelle besitzt. Diese vierte Tabelle lässt sich nicht abhängig vom Attribut „`ID_depot`“ partitionieren (Frage: Warum nicht?).

    b. Weisen Sie der so erstellten direkt abhängig partitionierten dritten Tabelle die entsprechenden und bekannten Datensätze zu.

d) Testen Sie die erfolgreiche Partititionierung:

    a. Qualifizieren Sie bei der Selektionsanweisung lediglich die Daten einer Partition („`bonn`“, „`london`“ und „`newyork`“), ohne hierbei eine „`WHERE`“-Bedingung zu benutzen (direkter Zugriff auf die Tabellendaten einer Partition). Testen Sie hierbei auch den Join.

    b. Verändern Sie die das Partitionierungsattribut einzelner Datensätze (Migration. Abhängige Migration), löschen Sie Datensätze und fügen Sie neue Datensätze ein. Testen Sie den Erfolg der Modifikation, indem Sie sich die Tabellendaten einzelner Partitionen (wie oben) anzeigen lassen.

c. Erstellen Sie Datensätze, die keiner Partition zugeordnet werden können. Wie reagiert das System?

e) Die fünfte Tabelle lässt sich ebenfalls nicht abhängig vom Attribut „`ID_depot`“ partitionieren (Frage: Warum nicht?).

1.5 Systemadminstration

Zur Visualisierung stellt das Data-Dictionary zahlreiche System-Views zur Verfügung, die eine sehr flexible Anzeige der Schema-Objekte ermöglicht. Zwei spezielle System-Views zur Anzeige partitionierter Tabellen lauten

- `USER_PART_TABLES`
- `USER_TAB_PARTITIONS`

a) Lassen Sie sich die Struktur dieser Views mit „`DESCRIBE <view>`“ anzeigen.

b) Erstellen Sie eine Übersicht Ihrer partitionierten Tabellen mit den folgenden Spalten: Name der Tabelle, Name der zugeordneten Partition und Name des zugehörigen Tablespaces.

c) Experimentieren Sie mit den System-Views.

2 Vorstellung der Ergebnisse

Im Rahmen der Vorstellung Ihrer Ergebnisse werden die folgenden Unterlagen **pro Gruppenmitglied** erwartet (fristgerechter Upload auf die Lernplattform):

a) Erstellung (unter Ihrer persönlichen Benutzerkennung):

   a. Erstellung der drei partitionierten bzw. abhängig partitionierten Tabellen im Datenbanksystem Bonn.

   b. Zuweisung der Daten.

   c. Test der Partititionierung mittels direktem Tabellenzugriff auf ausgewählte Partitionen (`select`).

   d. Modifikation einiger Datensätzte (`insert`, `delete`, `update`), anhand dessen die Partitionierung aktiv getestet werden kann.

b) Upload folgender Dateien (nutzen Sie die Vorlage die in moodle zur Verfügung gestellt wird):

   a. Das Skript der Installation, der Datenzuweisung und der Testzugriffe `partition_database.sql`“ .

   b. Das eigene Protokoll der Installation, der Datenzuweisung und der Testzugriffe `partition_database.log`“ (**nicht ein Protokoll pro Gruppe, sondern die individuellen Protokolle der einzelnen Gruppenmitglieder**).

c) Vorstellung Ihrer Datenbank mit Hilfe des Oracle SQL Developers.

d) Beantwortung der im Text „eingebauten“ Fragen (mündlich).