Oracle Performance Tuning
Musterlösung zur Prüfung WS 2000/2001

Alex Schröder

2001-09-24

Allgemeines

Bei der Musterlösung sind für jede Teilfrage verschiedene Bestandteile einer richtigen Antwort mit der entsprechenden Punktezahl aufgeführt. Die Summe der Punkte für die einzelnen Bestandteile kann die maximal erreichbaren Punkte pro Teilfrage überschreiten. Dies ist immer dann der Fall, wenn es Alternativen zur "besten" Antwort gibt. Die erreichbare Punktzahl pro Teilfrage bleibt aber weiterhin auf die angegebene Maximalpunktzahl beschränkt.

1 Speicherstrukturen

Zeichnen sie schematisch, wie Segmente, Extents und Datablocks zusammenhängen. (10 Punkte)

- Das Diagram zeigt ineinander verschachtelt ein Segment, darin mindestens ein Extent, und darin wiederum mindestens einen Datablock. Die Beschriftung ist korrekt. (6 Punkte)
- Das Diagram zeigt innerhalb des Extents mehrere Datablocks. Die Datablocks füllen den Extent restlos aus. (2 Punkte)
- Das Diagram zeigt, dass die Extents nicht zusammenhängend sein müssen. (2 Punkte)

Was passiert in diesem Zusammenhang, wenn eine Tabelle neu erstellt wird und dann nach und nach mit Daten gefüllt wird? (10 Punkte)

- Beim Anlegen der Tabelle wird ein Segment angelegt. (3 Punkte)
- Eventuell werden für die Tabelle auch Indexe angelegt. Für jeden Index wird ein weiteres Segment angelegt. (1 Punkt)
- Das Segment enthält mindestens ein Extent. (2 Punkte)
- Die benötigten Datablocks werden von Oracle für ein Extent reserviert, sobald es angelegt wird. Der Platz wird verbraucht, obwohl die Datablocks noch nicht benötigt werden. (1 Punkt)

- Beim Speichern von Daten werden die Datablocks nach und nach belegt. (1 Punkt)
- Wenn sich die Tabelle füllt, wird ein neuer Extent angelegt, sobald der erste Extent voll ist. (3 Punkte)
- Die Grösse des ersten Extents und aller nachfolgenden Extents lassen sich beim Erstellen der Tabelle angeben. (1 Punkt)
- Wurden diese Angaben beim Erstellen der Tabelle nicht angegeben, werden die Defaults des Tablespaces verwendet. (1 Punkt)
- Beim anlegen der Daten können chained oder migrated rows entstehen. (1 Punkt)

2 Architektur Komponenten

Warum gibt es in der Oracle Architektur Redo Logs? (10 Punkte)

1. Mit einer alten Kopie der Daten plus den Redo Log Dateien lässt sich der letzte Stand wiederherstellen. (4 Punkte)
2. Mit einer alten Kopie der Daten plus den Redo Log Dateien lässt sich zudem ein beliebiger Stand der Vergangenheit wiederherstellen. (point-in-time recovery, undo) (1 Punkt)
3. Beim Redo Log handelt es sich um einen kontinuierlichen Strom von Änderungsanweisungen. (2 Punkte)
4. Das Schreiben der Redo Log Dateien ist einfacher als eine Änderung der Datenbank Dateien. (2 Punkte)
5. Das Backup der Redo Log Dateien ist einfacher (weniger Platz, schneller, weniger Wartungsaufwand) als ein Backup der Datenbank Dateien. (2 Punkte)

Was wären die Nachteile einer Lösung ohne Redo Logs? (10 Punkte)

1. Da jede DB Änderung eine Änderung der Datenbank Dateien zur Folge haben muss, sind die Schreibe Operationen sehr viel aufwendiger. (4 Punkte)
2. Nach einem Crash sind möglicherweise die letzten Änderungen verloren. (Es sei denn, jede Änderung modifiziert sofort die Datenbank Dateien.) (4 Punkte)
3. Das Backup der Datenbank Dateien ist komplizierter (brauchen mehr Platz, dauern länger, eventuelle Einschränkgungen des Betriebes wegen starker Beanspruchung). (4 Punkte)
4. Es ist nicht möglich, beliebige vergangene Stände zu rekonstruieren. (weder undo noch point-in-time recovery)

3 Sortiervorgänge

Warum werden nicht alle Sortiervorgänge im Speicher ausgeführt? (10 Punkte)

1. Die Grösse dieses Speichers ist limitiert. Für die Sortiervorgänge ist eine bestimmte Menge an Speicher reserviert. (4 Punkte)
2. Die Folge: Zwischenergebnisse werden auf der Festplatte gespeichert. (2 Punkte)
3. Die Beanspruchung der Festplatte führt zu sinkender Performance. (2 Punkte)
4. Zu grosser Speicherverbrauch bremst andere Applikationen und kann zu swapping (Auslagerung des Speichers auf die Festplatte) führen. (4 Punkte)

Welche Funktion hat der Parameter SORT_AREA_SIZE? (5 Punkte)

1. Der Parameter bestimmt, wieviel Platz für den Sortiervorgang maximal verwendet werden dürfen. (4 Punkte)
2. Der Platz wird pro Benutzer reserviert. (1 Punkte)

Warum müssen die Daten für die Erstellung eines normalen B* Indexes sortiert werden? (5 Punkte)

1. Kurze Funktion eines Indexes ist beschrieben: Suchwert wird mit einem ersten Wert im Index verglichen, dann wird entschieden in welchem Teil des Indexes weitergefahren wird. Dort wieder ein Vergleich, etc. Dies bedingt eine Sortierung der Werte, mit denen verglichen wird. (5 Punkte)
2. Alternativ: Die Knoten im Index sind balanciert. (4 Punkte)
3. Alternativ: B* Indexe werden ungenügend beschrieben ohne eigentliche Begründung. (2 Punkte)

4 Verteilte Datenbanken

Was sind die Vor- und Nachteile eines DB-Links? (10 Punkte)

1. Vorteil: Daten sind immer aktuell (4 Punkte)
2. Vorteil: Für Benutzer und Entwickler transparent, falls Synonyme verwendet werden (4 Punkte)
3. Vorteil: Andere Anbieter können auf die Daten zugreifen. (1 Punkt)
4. Vorteil: Platzersparnisse auf der Seite, welche die DB Links verwendet. (1 Punkt)
5. Nachteil: Die Performance ist abhängig von der Netzverbindung. (4 Punkte)

6. Nachteil: Der Wartungsaufwand steigt an. Für die Berechtigungen müssen vielleicht neu Views angelegt werden, für die transparente Verwendung des DB Links müssen Synonyme angelegt werden. (3 Punkte)

7. Oracle Parallel Server hat hiermit gar nichts zu tun. (0 Punkte)

Entscheiden sie, ob sie in folgendem Fall DB-Links empfehlen würden oder nicht. Begründen sie ihren Entscheid. (10 Punkte)

1. Ja, da dies unkompliziert und einfach zu installieren ist. (6 Punkte)

2. Ja, sofern die Netzwerkverbindung gut ist. (6 Punkte)

3. Nein. Besser wären Materialized View bzw. Snapshot, denn dann spielt die Netzwerkverbindung für den normalen Betrieb keine Rolle. (10 Punkte)

4. Nein. Besser wäre eine Web Applikation, denn dann wird nur das user interface verteilt, nicht die Daten selber. Je nachdem ist die Belastung des Netzwerkes damit deutlich geringer. (10 Punkte)

5. Je nach Lösungsvorschlag wird richtig argumentiert, dass die Aktualität der Daten relevant bzw. nicht relevant ist. (6 Punkte)

5 Indexe

Warum sind die beiden Statements bei den gegebenen Tabellen- und Indexdefinitionen nicht effizient? (5 Punkte)

- Es existiert kein Index auf der Spalte TEXT. (1 Punkt)

- Es kommt zu einem full table scan. (1 Punkt)

- Die Verwendung von UPPER verhindert die Verwendung eines normalen Indexes. UPPER verlangt einen eigenen Function Index für die TEXT Spalte. (2 Punkte)

- Die Verwendung des % Platzhalters am Anfang des Suchbegriffes verhindert die Verwendung eines Indexes. (4 Punkte)

Schlagen sie eine bessere Lösung vor. Begründen sie ihre Antwort, indem sie kurz Vor- und Nachteile (Einschränkungen) ihrer Lösung erwähnen. (5 Punkte)

- Die Wörter aller Haikus werden in einer zweiten Tabelle einzeln gespeichert und indexiert. Zu jedem Wort lässt sich so schnell bestimmen, in welchen Haikus es auftaucht. (2 Punkte)

- Entweder ist Gross- und Kleinschreibung bei der Suche relevant, oder alle Wörter in der zweiten Tabelle werden grossgeschrieben gespeichert, oder es wird ein entsprechender Function Index angelegt. (1 Punkt)

- Vorteil: Da alle Wörter indexiert wurden, lassen sich diese nun schnell finden. (1 Punkt)

- Nachteil: Die Suche nach Teilbegriffen ist weiterhin langsam. (1 Punkt)

- Nachteil: Die zweite Tabelle sowie eventuelle weitere Indexe verbrauchen mehr Platz. (1 Punkt)
- Nachteil: Das Pflegen redundanter Daten ist beim Ändern der Daten langsam und fehleranfällig. (1 Punkt)

Skizzieren sie kurz die Datenstrukturen für ihre Lösung. Falls sie die Daten in anderen Tabellen anlegen, geben sie Name der Tabelle sowie Namen und Datentyp für alle Spalten an. Falls sie neue Indexe anlegen, geben sie Indextyp sowie Namen und Reihenfolge der indexierten Spalten an. Verwenden sie die Spaltennamen der HAIKU Tabelle, falls es sich um dieselben Daten handelt. (5 Punkte)

- Die HAIKU Tabelle wird beibehalten oder eine äquivalente Tabelle wird vorgeschlagen. (1 Punkt)
- Die zweite Tabelle entspricht der nachfolgenden Definition. (2 Punkte)

```
OCCURRENCES

WORD                VARCHAR2(30)
POEM_NO             NUMBER
```

- Auf die zweite Tabelle wird ein Index angelegt, dessen erste Spalte WORD ist. (1 Punkt)
- Geht man davon aus, dass die Spalte WORD nicht selektiv ist, kann anstelle eines normalen Indexes für die Spalte WORD ein Bitmap Index verwendet werden. (2 Punkte)
- Die zweite Tabelle ist eine *Index Organisierte Tabelle* und enthält deswegen schon einen Index auf die Spalten WORD und POEM_NO. In diesem Fall ist die Reihenfolge der Spalten in der Tabellendefinition relevant! (2 Punkte)

Geben sie zwei SQL Statement an, welche die Vorteile ihrer Lösung nutzten. Das erste Statement soll die Nummern alle Haiku liefern, die "Buddha" enthalten. Das zweite Statement soll alle Haiku Texte liefern, die "Buddha" enthalten. Die Statements dürfen durchaus unter den von ihnen schon erwähnten Nachteilen leiden. (5 Punkte)

- Folgendes Statement gilt für das erste Statement als korrekt. (3 Punkte)

```
SELECT POEM_NO
FROM   OCCURRENCES
WHERE  WORD = 'BUDDHA';
```

- Variationen mit "LIKE" statt "=" gelten auch als korrekt. Wurde "LIKE" verwendet, gilt "BUDDHA%" auch als korrekt. (3 Punkte)

- Variationen mit “Buddha” statt “BUDDHA” gelten als korrekt, wenn ein passender Index besteht. Ist OCCURRENCES eine normale Tabelle mit einem Function Index, oder wenn in die Wörter gross geschrieben in der Tabelle OCCURRENCES gespeichert werden, so existiert kein passender Index. Dies gilt als leichter Fehler. (2 Punkte)
- Variationen mit UPPER(WORD) gelten als korrekt, wenn ein Function Index besteht oder die Wörter grossgeschrieben gespeichert werden. Ist OCCURRENCES eine normale Tabelle ohne normalem Index aber mit einem Function Index, so existiert kein passender Index. Dies gilt als leichter Fehler. (2 Punkte)
- Folgendes Statement gilt für das zweite Statement als korrekt. (3 Punkte)

  ```
  SELECT HAIKU.TEXT
  FROM   HAIKU, OCCURRENCES
  WHERE  OCCURRENCES.WORD = 'BUDDHA'
  AND    OCCURRENCES.POEM_NR = HAIKU.POEM_NR;
  ```

  Variationen mit Table Aliases gelten auch als korrekt.
- Variationen mit “LIKE” statt “=”, “Buddha” statt “BUDDHA” oder mit UPPER(WORD) werden behandelt wie beim ersten Statement.
- Variationen mit fehlendem Join zwischen den beiden Tabellen gelten als schwerer Fehler. (1 Punkt)

Nachtrag: Generell würde man natürlich nicht *alle* Wörter indexieren. Beispielsweise würde man nur Wörder ab vier Buchstaben indexieren. Damit kann man vermeiden, dass der Index für die Tabelle OCCURRENCES zu gross wird. Falls das gesuchte Wort beispielsweise “der” ist, wird ein Index kaum etwas nützen – da ein sehr grosser Teil aller Haikus dieses Wort enthält, wäre es effizienter, einen Full Table Scan auf die Original Tabelle zu machen: Das ursprüngliche Statement wäre hier angebrachter. Dies könnte man beispielsweise wie folgt implementieren:

```
SELECT HAIKU.TEXT
FROM   HAIKU, OCCURRENCES
WHERE  OCCURRENCES.WORD = 'BUDDHA'
AND    OCCURRENCES.POEM_NR = HAIKU.POEM_NR;
UNION
SELECT HAIKU.TEXT
FROM   HAIKU
WHERE  UPPER(TEXT) LIKE '%BUDDHA%'
AND    NOT EXISTS (SELECT 1
                   FROM   OCCURRENCES
                   WHERE  WORD = 'BUDDHA');
```

Falls das Wort nun in der Tabelle OCCURRENCES vorkommt, liefert das erste Select Statement die Resultate; falls das Wort in der Tabelle OCCURRENCES nicht vorkommt, liefert das zweite Select Statement die Resultate.

Um ein Beispiel zu liefern: Bei einem Haiku Wettbewerb der Japan Air Lines wurden 1964 etwa 41’000 englische Haikus eingereicht[1]. Angenommen die Hälf-

[1] Haiku: Japanische Gedichte, Dietrich Krusche, 1994

te dieser Haikus enthält das Wort “and”. Dies wäre sicherlich ein signifikanter Teil der Tabelle OCCURRENCES. Indem man dieses kleine und oft verwendete Wort nicht indexiert, spart man beträchtlich Platz – und die Performance leidet nicht. Selbst wenn das Wort “and” in der Tabelle OCCURRENCES aufgeführt wäre, würde dies ja über 20’000 Resultate liefern, welche dann in der Tabelle HAIKU nochmal gesucht werden müssten. Da ist es einfacher gleich die über 40’000 Einträge in der Tabelle HAIKU zu durchsuchen. Wenn nun das Wort “and” *nicht* in der Tabelle OCCURRENCES aufgeführt wäre, so würde im obigen Statement das erste Select keine Resultate liefern. Dies wäre relativ schnell, denn es ist schnell gezeigt, dass ein gewisses Wort nicht im Index vorkommt. Somit kommt das zweite Statement zum Zuge und macht nun den Full Table Scan.