Oracle Performance Tuning
Musterlösung zur Übungsprüfung WS 2001/2002

Alex Schröder

2002-01-20

1 Speicherstrukturen

- Auf dem Diagram sind ersichtlich: Buffer Cache (wird zuerst durchsucht, wenn ein Datablock gesucht wird), Redo Log Buffer (Änderungsanweisungen, welche schlussendlich in die Redo Log Files geschrieben werden), Shared Pool (wiederverwendbare Daten der Server), und im Shared Pool befinden sich Library Cache (geparste SQL Statements mit Execution Plan) und Data Dictionary Cache (Metadaten).
- Ineffizienzen entstehen, wenn es eine besonders grosses Tabelle gibt, von der sehr viele Daten immer wieder gebraucht werden. Wenn jedesmal andere Daten der Tabelle verwendet werden, können die DB Blocks der vorhergehenden Abfrage im Buffer Cache nicht verwendet werden. Somit wird jede Abfrage grosse Mengen neuer DB Blocks in den Buffer Cache ablegen, und alte Daten (gem. LRU Liste) aus dem Cache werfen. Somit werden die Zugriffe auf die anderen Tabellen gebremst, wärend die Zugriffe auf die grosse Tabelle nicht beschleunigt werden. Die Lösung: Buffer Pools. Ein eigenen RECYCLE Buffer Pool für die grosse Tabelle sorgt dafür, dass die Blocks der grossen Tabelle die Blocks der anderen Tabellen nicht aus dem Cache drängen, weil die Pools separat geführt werden.

2 Normalisieren, Denormalisieren

- ER Diagramm, siehe Abbildung 1.
- Wichtig ist, dass für dieses SQL Statement die APPARATEPARAMETER Tabelle mehrmals verwendet werden muss.

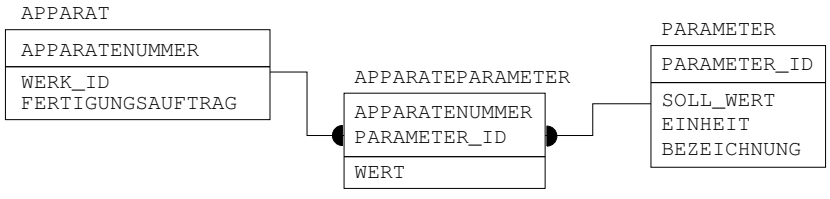

Abbildung 1: ER Diagram für die drei Tabellen.

```
select  a.apparatenummer
from    apparateparameter p5,
        apparateparameter p7,
        apparateparameter p165,
        apparateparameter p351,
        apparat a
where   a.fertigungsauftrag = 75
and     a.apparatenummer = p5.apparatenummer
and     p5.parameter_id = 5
and     a.apparatenummer = p7.apparatenummer
and     p7.parameter_id = 7
and     a.apparatenummer = p165.apparatenummer
and     p165.parameter_id = 165
and     a.apparatenummer = p351.apparatenummer
and     p351.parameter_id = 351;
```

- Denormalisieren, indem entweder diese vier Parameter in der APPARAT Tabelle doppelt geführt werden (max. 7 Punkte), oder – viel besser – indem eine Materialized View angelegt wird. Bei der Materialized View synchronisiert Oracle die Daten nämlich selber, wärend bei einer normalen Denormalisation die Entwickler dies selber ausprogrammieren müssen. Dies wäre eine potentielle Fehlerquelle. Bei einer Materialized View ist der Vorteil, dass die Daten schon aufbereitet vorliegen und das komplexe Statement deswegen nicht mehr ausgeführt werden muss. Nachteilig ist, dass Änderungen der Originaltabelle die Materialized View auch immer nachgeführt werden muss.

3 Indexe

- Ein B-Baum ist automatisch balanciert. Dies bedeuted, dass alle Äste des Baumes ähnlich lang sind. Dies führt dazu, dass es keine Werte gibt, für die der Baum aussergewöhnlich ineffizient ist. Der Baum ist für fast alle Werte gleich gut.

- Siehe Abbildung 2. Nur der letzte Schritt muss ersichtlich sein. Für Zwischenschritte gibt es jeweils 3 Punkte.

4 Optimizer

- RBO: Verwendet Heuristiken, um den Execution Plan zusammenzustellen. CBO: Verwendet – sofern vorhanden – zusätzlich noch Statistiken im Data Dictionary, um den Execution Plan zusammenzustellen. Der CBO liefert sehr gute Execution Plans bei sehr geringem Aufwand wärend dem Schreiben der Statements. Die Execution Plans passen sich an, falls sich die Datenzusammenstellung ändert. Der Nachteil liegt darin, dass die Statistiken regelmässig nachgeführt werden müssen; dies ist ein Wartungsaufwand und eine potentielle Fehlerquelle. Hauptproblem ist, wenn die Statistiken einfach vergessen werden, dann werden veraltete Metadaten verwendet.

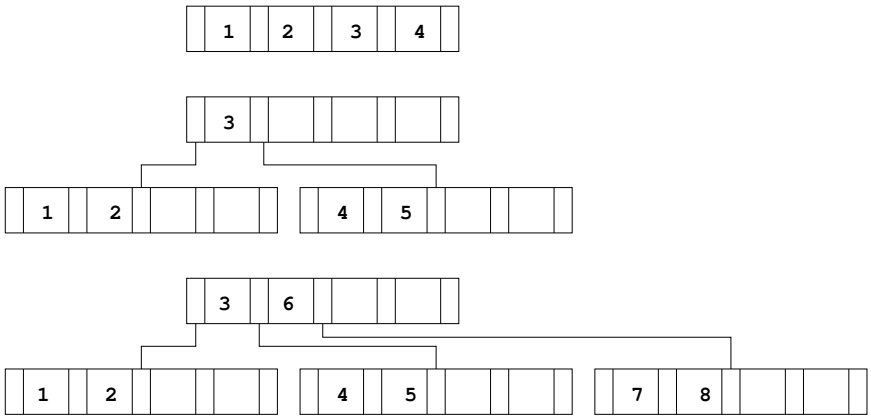

Abbildung 2: B-Baum mit Ordnung 5, dh. 5 Pointer und 4 Werte pro Knoten. Nach dem Einfügen von 5 erfolgt ein Split, nach dem Einfügen von 8 erfolgt ein weiterer Split.

- FIRST_ROWS: Optimiert den Execution Plan so, dass möglichst schnell Daten geliefert werden. Dies bietet sich bei interaktiven Applikationen an. ALL_ROWS: Optimiert den Execution Plan für eine effiziente Nutzung der Ressourcen. Somit verkürzt sich die Gesamtdauer, allerdings kann es länger dauern, bevor die ersten Daten geliefert werden.

5 SQL Tuning

- Die beiden Full Table Scans sind potentielle Performance Probleme.
- Die SALGRADE Tabelle ist zu klein – hier lohnt sich ein Index nicht (3 Punkte). Ein Index bei der EMP Tabelle auf die Spalte SAL würde die Performance möglicherweise steigern (4 Punkte). Dies geschieht nur, wenn mit diesem Index entschieden weniger DB Blocks gelesen werden müssen. Wenn die Löhne beispielsweise auf alle Kategorien gleichverteilt sind, so sind 20% aller EMP Datensätze Teil des Resultates. Wenn es nun sehr viele EMP Datensätze pro DB Block gibt, so wird man so oder so einen grossen Teil oder fast alle DB Blocks lesen müssen. In diesem Fall wäre ein Full Table Access schneller, weil dann Multiblock Reads verwendet werden, und der Index nicht gelesen wird.(4 Punkte).
- Ein equijoin ist ein Join in dem ein Gleichzeichen verwendet wird. Das BETWEEN entspricht aber einem ¡= und einem ¿=.
- Die Datensätze der beiden Datensatz Quellen werden nach den Spalten sortiert, welche für den Join verwendet werden, falls sie noch nicht sortiert sind. Anschliessend werden die Datensätze von den beiden Quellen zusammengefügt. Jedes Paar von Datensätzen gehört zum Resultat, wenn die Spalten, welche im Join verwendet werden, gleich sind. Ein Erklärung mit Sortieren und zwei Pointern, ist gleichwertig. Die Erklärung, dass damit nach der Sortierung kein Datensatz mehrfach verwendet werden muss, ist optional.

- Da es sich hier nicht um einen equijoin handelt, können die Datensätze, welche schon einmal geprüft worden sind, nicht eliminiert werden. Ein Datensatz kann mehrfach zum Resultat gehören, und zwar auch dann, wenn die Spalten, welche im Join verwendet werden, nicht gleich sind. Eine Erklärung mit Beispiel ist gleichwertig.