
2.1 Was ist eigentlich Performance Tuning?

Bevor wir mit den Methoden des Performance Tuning anfangen, ist es sinnvoll zu klären, was Performance Tuning ist. Es gibt eine Menge Definitionen. Die folgende Definition stellt keine Ansprüche, die einzig richtige zu sein.

Unter Performance Tuning versteht man organisatorische und technische Maßnahmen, die die Datenbank optimieren, um so ein Funktionieren zu erreichen, das in jeder Hinsicht akzeptabel ist. Diese Definition betont, dass die Datenbank für einen Zweck optimiert wird und diese Optimierung nebst den technischen auch die organisatorischen Methoden umfasst, die manchmal wesentlich sinnvoller, einfacher oder billiger als die technischen Lösungen sind. Sicherlich sind für uns in erster Linie gerade die technischen Methoden interessant. Die organisatorischen Methoden muss man aber auch im Auge behalten. Einige Performanz-Probleme kann man beispielsweise durch eine bessere Planung der Betriebsprozesse lösen. Manchmal ist es sinnvoller, die Stabilität und die akzeptable Produktivität durch Einführung der neuen Hardware zu erreichen, als einen ewigen und aussichtslosen Kampf gegen die schlechte Performanz mit den Tuning-Methoden zu führen.

Unter diese Definition fallen sowohl das planmäßige Performance Tuning als auch das Tuning im Fall der akuten Performanz-Probleme. Da die Datenbankadministratoren (ich übrigens auch) sehr oft mit den akuten Performanz-Problemen zu tun haben, stelle ich solche Probleme und deren Lösungen ins Zentrum dieses Buches. Diese Probleme bereiten einige zusätzliche Schwierigkeiten für die Spezialisten des Performance Tuning. Sie treten meistens plötzlich auf und verlangen eine schnelle Lösung. Einige Performanz-Probleme werden durch Bugs von Oracle bzw. vom Betriebssystem verursacht. In diesem Fall helfen die herkömmlichen Tuning-Methoden entweder gar nicht oder in einem sehr begrenzten Umfang. Man muss also solche Probleme richtig klassifizieren und die passenden Lösungen wählen (die jeweiligen Patches und Workarounds). Dafür muss man beurteilen, ob die Datenbank sich ordnungsgemäß verhält (nach dem Konzept von Oracle). Dies ist nicht immer leicht.

Es kann sein, dass ein akutes Problem gar nichts mit Performanz zu tun hat, obwohl es wie ein Performanz-Problem aussieht (s. das Beispiel im nächsten Abschnitt). Das bedeutet, dass man bei Performance Tuning nicht unbedingt immer mit reinen Performanz-Problemen zu tun hat, sondern manchmal mit Problemen, die vom Performance Tuning sehr weit entfernt sind. Das macht das Performance Tuning in meinen Augen noch attraktiver und spannender.

2.1.1 Ein langsamer Delete

Eines Tages beschwerte man sich bei mir über eine sehr schlechte Performanz der DELETE-Kommandos. Eins der problematischen Kommandos sah im AWR folgendermaßen aus.

```

---- GROUP=16.06.2011 00:00:00, SNAP_ID <= 18650, SNAP_TIME <= 17.06.2011 00:00:48,
SQL_ID=9471kj0sw2n4q
---- EXECUTIONS=87, ROWS_PROCESSED=0, PARSE_CALLS=88
---- DISK_READS=274, BUFFER_GETS=3098, DIRECT_WRITES=0
---- DISK_READS_PER_EX=3.15, BUFFER_GETS_PER_EX=35.61, DIRECT_WRITES_PER_EX=0,
ROWS_PROCESSED_PER_EX=0, PARSE_CALLS_PER_EX=1.01
---- DISK_READS_PER_ROW=274.0000, BUFFER_GETS_PER_ROW=3098.0000
---- CPU_TIME (sec.)=1181.0824, ELAPSED_TIME (sec.)=2866.0444, WAIT_TIME (sec.)=1684.9620
---- CPU_TIME_PER_EX (sec.)=13.5757, ELAPSED_TIME_PRO_EX (sec.)=32.9430, WAIT_TIME_PRO_EX
(sec.)=19.3674
---- PLSQL_EXEC_TIME (sec.)=0.0000, JAVA_EXEC_TIME (sec.)=0.0000
---- PLSQL_EXEC_TIME_PER_EX (sec.)=0.0000, JAVA_EXEC_TIME_PRO_EX (sec.)=0.0000
---- APPLICATION_WAIT_TIME (sec.)=0.0000, CONCURRENCY_WAIT_TIME (sec.)=0.0000,
CLUSTER_WAIT_TIME (sec.)=0.0000, USER_IO_WAIT_TIME (sec.)=1.8639
---- APPLICATION_WAIT_TIME_PER_EX (sec.)=0.0000, CONCURRENCY_WAIT_TIME_PER_EX (sec.)=0.0000,
CLUSTER_WAIT_TIME_PER_EX (sec.)=0.0000, USER_IO_WAIT_TIME_PER_EX (sec.)=0.0214
---- FORCE_MATCHING_SIGNATURE = 7795373675645671414
---- OPTIMIZER_MODE = ALL_ROWS, OPTIMIZER_ENV_HASH_VALUE = 1009514590
DELETE FROM XXX.PPI WHERE kennung = :1
---- Execution Plan (Plan Hash Value : 1817795815) :
DELETE STATEMENT Optimizer=ALL_ROWS (Cost=2)
  DELETE OF PPI (Bind Peeking used)
    INDEX (UNIQUE SCAN) OF IDX_PPI_PK (Cost=2 Card=1 Bytes=316 CPU_Cost=15293 IO_Cost=2 Time=1)

```

Tja, ca. 33 s Laufzeit (ELAPSED_TIME) für ein Löschen über Unique Index Scan waren in der Tat merkwürdig. Die Laufzeitstatistiken (3,61 Buffer Gets und 3,15 Disk Reads pro eine Ausführung) passten absolut nicht zu dieser Ausführungsdauer. Trotzdem überprüfte ich für alle Fälle, ob die Tabelle PPI irgendwelche Trigger und Foreign Key Constraints hatte. Ich fand nichts.

Da ich mit einem DELETE nicht experimentieren wollte, und ein SELECT über einen Unique Index Scan meiner Meinung nach das jeweilige Problem nicht hätte nachstellen können, aktivierte ich das SQL-Tracing für eine Session, welche den DELETE ausführte. Das war eine glückliche Idee, weil ich am Anfang der Trace-Datei folgendes entdeckt habe.

```

Dump file /opt/oracle/admin/XXX/udump/XXX_ora_4481148.trc
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
ORACLE_HOME = /opt/oracle/product/10.2
System name:      AIX
Node name:        XXX52
Release:          3
Version:          5
Machine:          XXX00
Instance name: XXX
Redo thread mounted by this instance: 1
Oracle process number: 198
Unix process pid: 4481148, image: oracle@XXX52
*** 2011-06-17 08:54:38.587
*** SERVICE NAME:(CLNRTWRITE) 2011-06-17 08:54:38.586
*** SESSION ID:(270.46070) 2011-06-17 08:54:38.586
oer 8102.2 - obj# 92402, rdba: 0x057820bc(afn 21, blk# 3678396)
kdk key 8102.2:
  ncol: 2, len: 18
  key: (18): 0a 32 30 31 31 2d 30 36 2d 30 38 06 02 f3 28 a7 00 0a
mask: (4096):
01 08 00 00 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Die ziemlich unauffällige Zeile „oer 8102.2– obj# 92402 ...“ wies auf den Fehler ORA-8102 hin, welcher beim Löschen der Daten auftrat. Ich überprüfte, was dieser Fehler bedeutet.

```

ORA-08102: index key not found, obj# string, file string, block string (string)
Cause: Internal error: possible inconsistency in index
Action: Send trace file to your customer support representative, along with
information on reproducing the error

```

Als nächstes ermittelte ich, was für ein Index es war (in der View DBA_OBJECTS über die Spalte OBJECT_ID=92402) und wie dieser Index aufgebaut war (ich habe dafür das CREATE-Kommando für den jeweiligen Index mittels der Funktion DBMS_METADATA.GET_DDL generiert).

```

CREATE INDEX "XXX"."IDX_PPI_SUBSTR_ERSTELLUNG" ON "XXX"."PPI"
(SUBSTR(TO_CHAR("ERSTELLUNGSZEITPUNKT"),0,10))
PCTFREE 10 INITRANS 2 MAXTRANS 255
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "INV"

```

Die Funktion TO_CHAR ohne Format schien mir sehr verdächtig. Der Tabellendefinition für PPI entnahm ich, dass die Spalte ERSTELLUNGSZEITPUNKT vom Typ TIMESTAMP war. Erst jetzt wurde mir klar, was passiert war. Die TIMESTAMP-Daten wurden mit einer Vorgabeeinstellung für das TIMESTAMP-Format als Zeichenkette im Index abgespeichert. Mit einer anderen Einstellung auf der Session-Ebene, konnte Oracle die jeweiligen

Index-Werte nicht mehr beim DELETE finden. So kam es zum Fehler ORA-8102. Die große Laufzeit entstand dadurch, dass Oracle automatisch eine Trace-Datei im Problemfall erzeugt, was auch seine Zeit braucht. Für den Datenbankadministrator, der dieses Problem an mich gemeldet hatte, sah es wie ein reines Performanz-Problem aus.

Ich baute einen kleinen Test-Case für das TIMESTAMP-Format zusammen. Mit dem ähnlichen Test-Case testete ich, wie Oracle mit dem Datum-Format in diesem Fall umgeht. Beim Datum speichert Oracle automatisch die jeweilige Vorgabeeinstellung in der Funktion TO_CHAR als Format mit, so dass dieses Problem mit dem Datum nicht entsteht. Man muss aber dieses Format in den SQL-Anweisungen explizit benutzen, um die Zugriffe über den jeweiligen Index zu ermöglichen.

Da die Problemlösung lediglich mit einer Programmänderung möglich war, entschied man, keinen Service Request für dieses Problem bei Oracle zu eröffnen.

Fazit

- man muss sich darauf einstellen, dass sich ein völlig anderes Problem unter dem angeblichen Performanz-Problem verstecken kann,
- die Laufzeitstatistiken auf der Cursor-Ebene kumulieren die jeweiligen Statistiken der darunterliegenden Operationen (in diesem Fall die Laufzeit)

2.2 Eine akzeptable Performance als Kriterium des Performance Tuning

Unter welchen Umständen soll man mit Performance Tuning anfangen, und wann soll man die Performanz-Verbesserungen beenden? Theoretisch ist jedes System performanzmäßig zu verbessern. Das Problem dabei ist, dass man in der Regel mit jedem nächsten Schritt immer weniger an der Performanz gewinnt, aber immer mehr Zeit in das Tuning investiert, so dass das Performance Tuning immer teurer wird. Wo muss man denn die Grenze ziehen? Meiner Meinung nach ist es sinnvoll, hier ganz praktisch vorzugehen. Wenn die Performanz den Betrieb nicht gefährdet und keine nichtperformanten Abfragen die Anwender nerven, ist eine solche Datenbankperformanz in meinen Augen akzeptabel und nicht unbedingt zu verbessern. Das heißt nicht, dass Sie nicht reagieren dürfen, wenn Sie eine Auffälligkeit entdeckt haben, die u. U. zu einem größeren Problem werden kann oder leicht zu beseitigen ist. Selbstverständlich müssen Sie die Zuständigen darüber informieren und möglicherweise sofort eingreifen. Bei einer Routineuntersuchung einer Datenbank mit einer OLTP-Anwendung ist mir eine komplexe SQL-Anweisung aufgefallen, die viele UNIONS beinhaltet und für jede Ausführung ca. 40 s brauchte. Aus meiner Erfahrung wusste ich, wie man solche SQL-Anweisungen leicht beschleunigen kann (s. im nächsten Abschnitt). So habe ich die Laufzeit dieser SQL-Anweisung auf einen Sekundenbruchteil reduziert. Wenige Minuten später haben sich die überglücklichen Anwender gemeldet. Sie haben jahrelang vor dem Bildschirm geduldig jeweils 40 s lang gewartet, bis die

problematische Abfrage endete. Aber auch in diesem Fall, meiner Meinung nach, war die Performanz akzeptabel, da die Leute sich auf diese lange Antwortzeit einstellten.

2.3 Performance Tuning aus der Sicht der Entwickler und der Datenbankadministratoren

Es ist kein Wunder, dass die Entwickler und die Datenbankadministratoren ganz unterschiedliche Blickwinkel auf das Performance Tuning haben und dementsprechend unterschiedliche Tuning-Methoden benutzen: Ihre Aufgaben und Tätigkeitsbereiche sind verschieden.

Ein Entwickler hat normalerweise lediglich mit SQL-Problemen zu tun. Er kennt sich mit dem jeweiligen Datenmodell aus und nutzt seine Kenntnisse bei Performance Tuning. In einem Problemfall weiß er sehr oft, wie es richtig und performant funktionieren soll. Das hilft ihm, eine problematische Stelle in der jeweiligen SQL-Anweisung zu finden und zu korrigieren. Ein Entwickler kennt sich in der Regel gut mit SQL aus und versteht unter Performance Tuning effiziente SQL-Programmierung, möglicherweise mit verschiedenen Tricks. Die Entwickler können diese Darstellung etwas primitiv finden, grundsätzlich stimmt sie aber.

Ein Datenbankadministrator hat die Vorteile eines Entwicklers nicht. In der Regel kennt er sich nicht oder schlecht mit dem Datenmodell aus, seine SQL-Kenntnisse erlauben ihm nur ziemlich einfache SQL-Anweisungen zu programmieren. Er weiß nicht, wie es sein soll, sondern muss nach den Engpässen im Ausführungsplan suchen (oder das Oracle überlassen), wenn ein SQL-Problem vorliegt. Ein Datenbankadministrator kann meistens keine SQL-Anweisungen ändern. Er macht Tuning mit ganz anderen Methoden und erreicht eine bessere Performanz durch Erstellung der aktuellen Optimizer-Statistiken, Änderung der Parametereinstellungen, Benutzung der Stored Outlines, SQL Profiles, SQL Plan Baselines usw. Außerdem hat er nicht nur mit SQL-Problemen, sondern auch mit anderen Performanz-Problemen zu tun.

Jahrelang war ich als Entwickler bei verschiedenen Firmen tätig. Als ich danach mich auf Performance Tuning spezialisiert habe, habe ich zunächst beim Tuning meine Methoden aus der Entwicklerzeit benutzt. Sehr schnell habe ich festgestellt, dass ich auf diese Methoden verzichten muss. Ich hatte keine Zeit, um mich in das neue Datenmodell bei jedem Problem einzuarbeiten. Dieser Aufwand stand in keinem guten Verhältnis zum Ergebnis. Das hat mich bewogen, nach neuen Tuning-Methoden zu suchen.

Die Tuning-Methoden, die die Datenbankadministratoren und die Spezialisten für Performance Tuning benutzen, sind meist effektiv. Aus diesem Grund können sie in einigen Situationen auch den Entwicklern behilflich sein. Ich will damit nicht sagen, dass die Entwickler diese Methoden unbedingt beherrschen müssen. Eine grobe Vorstellung von diesen Methoden wäre aber meiner Meinung nach für sie von Vorteil.

An den nächsten 2 Beispielen möchte ich zeigen, wie stark sich die Denkweise und die Ansätze eines Entwicklers und eines Spezialisten für Performance Tuning voneinander unterscheiden und wie effektiv die Methoden der letzteren sein können.

2.3.1 Performance Tuning mit 3 Datenbankparametern

Vor einigen Jahren musste ich bei einer Migration von Oracle 8.1.7 auf Oracle 10.2.0.2 mit Performance Tuning helfen, da die 10-er Testdatenbank eine miserable Performanz aufwies. Als Ansprechpartner habe ich einen Entwickler bekommen, der sofort verkündete, dass ich mich auf eine Zusammenarbeit von mindestens 3 Wochen einstellen müsse. Er begründete es damit, dass das Tuning bei der vorherigen Migration von Oracle 7 auf Oracle 8 auch etwa so lange gedauert hätte. Da ich andere Pläne hatte, versuchte ich eine andere Methode zu finden, statt die SQL-Anweisungen eine nach der anderen zu tunen. Meinen Ansprechpartner habe ich um einige Stunden gebeten, um mich zunächst umschauen zu können. In dieser Zeit untersuchte ich die Datenbank. Dabei suchte ich gezielt nach irgendwelchen Besonderheiten, die mir beim Performance Tuning helfen konnten. Und ich fand sie. Die meisten SQL-Anweisungen bei dieser Datenbank waren ähnlich aufgebaut. Man könnte fast alle SQL-Anweisungen in 3–4 Kategorien unterteilen. Es gab dort z. B. sehr komplexe SQL-Anweisungen, die die Zehner von Operatoren UNIONS beinhalten, in den Ausführungsplänen der anderen fand ich immer „ANTI JOIN“, usw. Wie konnte man diese Tatsache beim Tuning gebrauchen? Meine Schlussfolgerung war sehr einfach. Wenn eine Datenbank ein paar Typen der SQL-Anweisungen hat und diese SQL-Anweisungen nicht performant sind, kann es sein, dass der jeweilige Oracle Release gewisse Probleme gerade mit solchen Typen der SQL-Anweisungen hat. Ich suchte unter den bekannten Problemen für 10.2.0.2 in MOS (My Oracle Support) und fand einige vermutlich passende. Am nächsten setzte ich 5 Parametereinstellungen als Workarounds für die gefundenen Probleme ein. Alle Test-Cases von mir brachten eine deutliche Performanz-Verbesserung. Nach den ausführlichen Performanz-Tests reduzierte der Kunde die Anzahl der Parametereinstellungen auf 3, da die anderen 2 für einige SQL-Anweisungen sich ungünstig erwiesen haben. Das hat aber die Tatsache nicht verändert, dass ich in einem Tag mit ein paar Parametern die Datenbank getunt habe.

Eine der abgelehnten Parametereinstellungen war `_OPTIMIZER_COST_BASED_TRANSFORMATIONS=OFF`. Diese Parametereinstellung half mir die SQL-Anweisungen mit vielen UNIONS zu beschleunigen. Diesen Trick habe ich nach diesem Fall mehrmals und meistens mit Erfolg wieder verwendet. Sie können das selber probieren, wenn Sie eine SQL-Anweisung mit mehreren nacheinander folgenden UNIONS zu tunen haben. Dabei ist folgendes zu beachten:

- die Anzahl der UNIONS in der SQL-Anweisung muss gravierend sein, damit diese Parametereinstellung hilft,
- die Parametereinstellung `_OPTIMIZER_COST_BASED_TRANSFORMATIONS=OFF` kann sich auf einige andere SQL-Anweisungen negativ auswirken. Aus diesem Grund ist es sinnvoll, diese Einstellung mit dem Hint `OPT_PARAM` gezielt für die problematische SQL-Anweisung einzusetzen (mehr Informationen zum Hint `OPT_PARAM` finden Sie im Abschn. 13.2).

Fazit

- Performance Tuning mit Parametereinstellungen kann sehr effizient sein,
- die gewonnenen Erkenntnisse bei einem Problemfall kann man erfolgreich bei den anderen Problemfällen benutzen

2.3.2 Workaround mit einem FBI

Nach einem Release-Wechsel der Anwendung ist eine wichtige SQL-Anweisung nicht performant geworden. Vor dem Release-Wechsel wurde ein Index Unique Scan im jeweiligen Ausführungsplan benutzt, nach dem Release-Wechsel lief diese SQL-Anweisung mit einem Full Table Scan. Es wurde ziemlich schnell klar, was das verursachte. Man definierte versehentlich in der jeweiligen Tabelle eine Spalte mit numerischen Werten als VARCHAR2. In der SQL-Anweisung benutzte man aber eine Bind-Variable vom Typ NUMBER. Einen Index für die jeweilige Spalte konnte Oracle in dieser Situation nicht gebrauchen.

Die Entwickler schlugen sofort vor, den jeweiligen Datentyp in der Tabelle wieder auf NUMBER zu ändern. Dieser Vorschlag war absolut richtig. Man konnte ihn leider erst 2 Wochen später umsetzen, da man die jeweilige Änderung zunächst vorbereiten musste. Noch mehr Zeit kosteten die Formalitäten: eine Genehmigung für diese Aktion auf der produktiven Datenbank. Diese 2 Wochen musste man irgendwie überleben.

Da die Workarounds gerade in solchen Fällen am besten einzusetzen sind, schlug ich einen ganz einfachen vor. Man konnte dieses Performanz-Problem mit einem FBI (function based index) umgehen. Mit dem folgenden Test-Case kann man das jeweilige Problem und dessen Lösung nachstellen. Zunächst wird eine kleine Tabelle mit einer Spalte C1 vom Typ VARCHAR2 angelegt und mit den Daten gefüllt, danach wird ein UNIQUE Index für diese Spalte erzeugt.

```
SQL> create table t1(c1 varchar2(100), c2 number, c3 number);
```

Table created.

```
SQL> declare
2   i integer;
3   begin
4       for i in 1..10000 loop
5           insert into t1 values (i,i + 1000000, i + 5000000);
6       end loop;
7   commit;
8   end;
9   /
```

PL/SQL procedure successfully completed.

```
SQL> create unique index i_t1 on t1(c1);
```

Index created.

Jetzt wird eine SQL-Anweisung ausgeführt, die die Daten für einen Wert der Spalte C1 ermittelt. In dieser SQL-Anweisung wird dafür eine Bind-Variable vom Typ NUMBER benutzt. Der jeweilige Ausführungsplan zeigt einen Full Table Scan an.

```
SQL> var b1 number
SQL>
SQL> begin
  2  :b1:=30;
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> select /*+ index(t1 i_t1) */ * from t1 where c1 = :b1;

C1
C2      C3
-----
30
1000030    5000030

SQL>
SQL> select plan_table_output from table (sys.dbms_xplan.display_cursor('','','ADVANCED' ));

PLAN_TABLE_OUTPUT
-----
SQL_ID  cd52tfhkcr264, child number 0
-----
select /*+ index(t1 i_t1) */ * from t1 where c1 = :b1

Plan hash value: 3617692013

-----
| Id | Operation                | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----
|  0 | SELECT STATEMENT         |      |      |      |  10 (100)|          |
|*  1 |  TABLE ACCESS FULL     | T1   |      |  78   |  10 (0)| 00:00:01 |
-----
```

Das Argument FORMAT der Funktion DBMS_XPLAN.DISPLAY_CURSOR habe ich auf 'ADVANCED' gesetzt, um den Typ der jeweiligen Bind-Variable zu ermitteln. Alternativ hätte man diesen Typ in der View V\$SQL_BIND_CAPTURE finden können.

Peeked Binds (identified by position):

```
1 - :B1 (NUMBER): 30
```

In dem ausgegebenen Ausführungsplan findet man auch die folgende Information über die Prädikate.

Predicate Information (identified by operation id):

```
1 - filter(TO_NUMBER("C1")=:B1)
```

Das bedeutet, dass Oracle die Funktion TO_NUMBER für jeden Wert der Spalte C1 berechnet, was den Full Table Scan verursacht. Warum kann Oracle das nicht umgekehrt machen, also die Funktion TO_CHAR für die Bind-Variable berechnen? Dafür fehlt Oracle das Format für die Funktion TO_CHAR. Die Spalte C1 kann beispielsweise die folgenden Werte beinhalten: 1, 1.0, 1.00, 1.000,

Alle dieser Werte entsprechen dem numerischen Wert 1. Da es nicht bekannt ist, mit welchem Format die numerischen Werte jeweils in der Spalte C1 abgespeichert sind, muss Oracle die Funktion TO_NUMBER für diese Werte berechnen und die Ergebnisse mit dem Bind-Wert vergleichen. Wenn man einen FBI für die Funktion TO_NUMBER anlegt, nimmt Oracle diesen Index.

```
SQL> drop index i_t1;

Index dropped.

SQL>
SQL> create unique index i_t1 on t1(to_number(c1));

Index created.

SQL>
SQL> select * from t1 where c1 = :b1;

C1
C2      C3
-----
30
1000030  5000030

SQL>
SQL> select plan_table_output from table (sys.dbms_xplan.display_cursor('','','ADVANCED' ));

PLAN_TABLE_OUTPUT
-----

SQL_ID  9shq0ww8kyw5d, child number 0
-----
select * from t1 where c1 = :b1

Plan hash value: 4196714161

-----
| Id | Operation                                | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----
|  0 | SELECT STATEMENT                        |      |       |       |  2  (100)|          |
|  1 |  TABLE ACCESS BY INDEX ROWID          | T1   |  1    |  91   |  2   (0)| 00:00:01 |
|* 2 |    INDEX UNIQUE SCAN                    | I_T1 |  1    |       |  1   (0)| 00:00:01 |
-----
```

Fazit

Die Workarounds stellen eine mächtige Tuning-Methode dar. Mit einem Workaround kann man oft sofort das jeweilige Performanz-Problem beseitigen.

2.4 Warum mögen manche Datenbankadministratoren das Performance Tuning nicht?

Das Performance Tuning gehört zu den Aufgaben der Datenbankadministratoren. Deswegen war ich etwas überrascht, als ich feststellte, dass einige Datenbankadministratoren diese Tätigkeit zu meiden oder sehr formal auszuüben versuchen. Eine Zeitlang konnte ich das nicht verstehen. Erst nach einigen Überlegungen und Beobachtungen bin ich zu folgendem Schluss gekommen. Das Performance Tuning ist eine Tätigkeit, die sich generell nicht formalisieren lässt (nur einige Klassen der Performanz-Probleme kann man mehr oder weniger formal darstellen). Aus diesem Grund sind viele Empfehlungen für Performance Tuning in der Fachliteratur ziemlich vage formuliert. Ein Spezialist für Performance Tuning muss das Problem selber analysieren und entscheiden, welche der typischen Empfehlungen zu dem jeweiligen Fall am besten passt. In einigen Fällen muss man diese Empfehlungen ignorieren oder sogar genau das Gegenteil tun. Es kann auch sein, dass der Fall (zumindest in der jeweiligen Ausprägung) gar nicht in der Fachliteratur beschrieben ist und man nach einer neuen Methode suchen muss. Das Performance Tuning verlangt also eine gewisse Kreativität von den Datenbankadministratoren. Wenn man ein Performanz-Problem angeht, besteht keine Garantie, dass man dieses Problem löst. Ich kann keine andere Standard-Aufgabe der Datenbankadministratoren nennen, die in dieser Hinsicht mit dem Performance Tuning vergleichbar wäre.

Nicht alle Datenbankadministratoren sind bereit zu einer Tätigkeit, die viel Kraft und Zeit kostet, und keinen Erfolg dabei verspricht. Die Leute mögen eher eine Routinearbeit, bei der sie nach dem Ausführen bestimmter Operationen garantiert und schnell ans Ziel kommen.

Dazu passt sehr gut ein Zitat von Albert Einstein: „Holzhacken ist deshalb so beliebt, weil man bei dieser Tätigkeit den Erfolg sofort sieht“. Dieses Zitat möchte ich dahingehend verändern, dass ich in meinem Fall Holzhacken durch Holzsägen austausche, ohne seinen Sinn zu ändern. Das Performance Tuning attraktiver und beliebter zu machen, ist eins der Ziele dieses Buches. Hier werden einige dunkle Ecken der Oracle Datenbank ausgeleuchtet und einige Methoden des Performance Tuning präsentiert. Mit diesen Methoden soll das Buch etwas mehr Routine in das Performance Tuning bringen. Spaßeshalber möchte ich das Sägen als Symbol des Performance Tuning und die Säge als Symbol eines Tool für Performance Tuning benutzen (eine Axt als solches Symbol ist mir einfach zu brutal). Dieses Symbol finden Sie auf den Abbildungen, die im weiteren Text auf eine lockere Art und Weise einige Seiten des Performance Tuning illustrieren.



Das Performance Tuning attraktiver und beliebter zu machen, ist eins der Ziele dieses Buches.

Zugleich möchte ich meinen Helfer Peter Schmidt vorstellen, der sich freundlicherweise bereit erklärt hat, mich durch dieses Buch zu begleiten. Er ist ein Praktiker und wird einige vorgeschlagene Methoden aus der praktischen Sicht beurteilen. Seine Fragen, Anregungen und Erfahrungen sind hier auch herzlich willkommen. Wir kennen uns ziemlich lange und sagen Du zueinander.

Leonid: „Peter, könntest Du bitte mich mit ein paar Worten ergänzen“.

Peter: „Wie Sie bereits wissen, heie ich Peter Schmidt. Seit einigen Jahren bin ich als Datenbankadministrator fr Oracle bei einem mittelgroen Unternehmen ttig. Ich bin kein

Neuling auf diesem Gebiet, aber mit Performance Tuning habe ich bis zuletzt nicht viel zu tun gehabt. Vor ein paar Monaten habe ich zwei neue Datenbanken zur Betreuung bekommen, die häufig Performanz-Probleme unterschiedlicher Art haben. Der Kampf mit diesen Problemen hat gezeigt, dass ich meine Kenntnisse im Performance Tuning vertiefen muss. Das Angebot von Leonid möchte ich gerade dafür nützen, zumal meine Beteiligung an diesem Buch mir nicht viel Zeit und Mühe zu kosten scheint. Ich übernehme gern die Rolle von Dr. Watson.“

2.5 Die technischen Voraussetzungen für das Performance Tuning

Was braucht man, um mit dem Performance Tuning anfangen zu können? Meiner Meinung nach gar nicht so viel. Selbstverständlich braucht man gewisse Datenbankkenntnisse. Diese Datenbankkenntnisse müssen aber am Anfang nicht unbedingt sehr tief und umfangreich sein. Es ist ausreichend, die Grundkenntnisse von der Oracle Datenbank zu haben, um grob zu verstehen, wie die Datenbank funktioniert. Man kann viel aus der Praxis lernen, wenn man sich an eine einfache Regel hält: stößt man in der Praxis auf etwas Neues, muss man unbedingt das jeweilige Phänomen klären (auch wenn es für die Problemlösung nicht notwendig ist).

Dafür braucht man Nachschlagewerke. Die 2 wichtigsten sind die Dokumentation und der MOS (My Oracle Support) – die Wissensdatenbank von Oracle (<https://support.oracle.com>). Falls man dort nicht ausreichend Informationen findet, kann man auch in den Blogs und in den Foren im Internet suchen. Bei diesen letzten 2 Quellen muss man etwas aufpassen, da man dort nicht immer die richtige Information findet. Dort gibt es viele „Legenden und Mythen“, die man erst dann erkennen kann, wenn man mit Vorsicht und mit gesunder Skepsis die gefundenen Informationen bewertet.

Man braucht noch unbedingt ein Tool für Performance Tuning. Wenn ein Performanz-Problem auftritt, muss man dieses Problem schnell klären und beseitigen. Wenn man erst in diesem Moment mit den für die Analyse notwendigen SQL-Anweisungen aus den internen Tabellen und Views von Oracle anfängt, kann es zu spät sein. Außerdem kann man mit solchen „selbstgestrickten“ SQL-Anweisungen nicht alle Bedürfnisse des Performance Tuning abdecken. Meiner Meinung nach müssen die folgenden Kriterien für das jeweilige Tool erfüllt werden:

- das Tool muss bei der Lösung der meisten Ihrer alltäglichen Probleme helfen,
- da dieses Tool Ihr Hauptinstrument ist, muss es für Sie transparent und bequem sein,
- das Tool muss mit den „offiziellen“ Begriffen von Oracle operieren (beispielsweise ein Latch muss dort genauso wie bei Oracle heißen und darf nicht irgendein „interner Lock“ sein),

- jede Ihrer Aktivitäten muss protokolliert werden. Das ist ein sehr wichtiges Kriterium, damit Sie jede Problemlösung einige Wochen, Monate sogar Jahre später nachvollziehen können, wonach häufig der Bedarf besteht. Die meisten praktischen Beispiele in diesem Buch habe ich anhand solcher Protokolle rekonstruiert,
- da die grafischen Auswertungen sehr wichtig für Performance Tuning sind, muss das Tool sie ermöglichen (die Wichtigkeit der grafischen Auswertungen besprechen wir im Abschn. 12.2).

Wenn Sie als Spezialist für Performance Tuning bei mehreren Kunden mit ihren Datenbanken arbeiten, sind auch die restlichen 3 Kriterien für Sie wichtig:

- das Tool installiert möglichst keine Objekte auf der Datenbank,
- das Tool muss eine portable Anwendung sein,
- die plattformunabhängig ist.



Suchen Sie ein passendes Tool aus

2.6 Was braucht man noch?

Das Performance Tuning ist im Prinzip eine sehr interessante und packende Tätigkeit. Wenn Sie generell Spaß an Problemlösungen haben, es können z. B. Kreuzworträtsel, Schachspiel oder irgendwelche Rätsel für scharfe Denker sein, wird das Performance Tuning Ihnen sicherlich auch Spaß machen. Nehmen Sie das Performance Tuning nicht allzu ernst, betrachten Sie es (zumindest zum Teil) als ein intellektuelles Spiel.



Spaß muss dabei sein

Phantasie und Kreativität sind auch beim Performance Tuning angesagt. Oft hilft dort ein unkonventioneller Ansatz, eine Problemlösung zu finden.



Das Performance Tuning verlangt Kreativität

Etwas Glück kann nie schaden. Sie werden aber merken: Je besser Sie im Performance Tuning werden, desto häufiger werden Sie Glück bei Problemlösungen haben.



Die Lösung kommt manchmal wie aus dem heiteren Himmel

Wenn Sie erst am Anfang ihrer Karriere beim Performance Tuning stehen, zögern Sie nicht, fangen Sie mit dem praktischen Tuning an. Lernen Sie bei „Doing“, und eines Tages werden Sie zu einem Meister.



Übung macht den Meister

Performance Tuning für Oracle-Datenbanken

Methoden aus der Praxis für die Praxis

Nossov, L.

2014, XVII, 402 S. 380 Abb., 26 Abb. in Farbe.,

Hardcover

ISBN: 978-3-642-33052-0