ORA-SOLUTIONS.NET
Mastering Oracle Performance, High Availability, Manageability

Manuelles Oracle SQL Tuning

Eine Einführung

DOAG Konferenz 2012
Martin Decker

Wer bin ich?

- Freiberuflicher Consultant in D/A/CH
- Oracle Certified Master 10g & 11g
- 8 Jahre Oracle-Erfahrung
- Seit 4 Jahren unabhängiger Oracle Consultant
- Oracle - Spezialisierung auf:
  - Performance Management (Instance / SQL)
  - Hochverfügbarkeit (MAA, RAC, DataGuard)
  - Manageability (OEM Grid/Cloud Control)
  - Unix (Linux, Solaris, HP-UX)
- Website & Blog: ora-solutions.net

Umfrage

- Schon mal einen Ausführungsplan gesehen?
- SQL Tuning Advisor benutzt?

Vorgehensweise

Identifizierung des SQL	• AWR Instance / SQL Report • SQL Trace • EM Performance-Page, etc.
Analyse	• Ermittlung der Ausführungs-Statistiken • Ermittlung Ausführungsplan (DBMS_XPLAN) inkl. Bind-Variablen
Reproduktion	• Reproduzieren der sub-optimalen Ausführungsperformance mittels SQL*Plus bzw. SQL Developer
Tuning	• Reduzierung der Buffer Gets des Statements durch iterative Veränderung und Messung der Auswirkung
Aktivierung optimaler Plan (optional)	• Stored Outline, SQL Profile, SQL Plan Baseline

Identifizierung SQL

- Beim Tuning des Statements werden vor allem zwei Metriken bewertet:
  - ~~NICHT: Physical I/O~~
  - **1. Logical I/O (buffer gets / gets):**
    - Buffer: Datenblock im Buffer Cache (z.B. 8 kb Block)
    - Buffer Gets: Lesen des Blocks aus dem Buffer Cache
    - Bei Physical I/O: Zuerst PIO, dann LIO
  - 2. Elapsed Time: Achtung: Caching

Identifizierung SQL

- AWR (Enterprise Edition und Diagnostic Pack License) / Statspack (Std. Edition) SQL-Report über bestimmtes Intervall
- SQL Tracing (DBMS_MONITOR)
- Active Session History (ASH) (Diagnostic Pack License)
- OEM – DB Performance Page (Diagnostic Pack License)
- SQL Performance Monitor (Diagnostic Pack License)

ORACLE Enterprise Manager Cloud Control 12c

Setup Help

Enterprise Targets Favorites History

Search Target Name EMREP

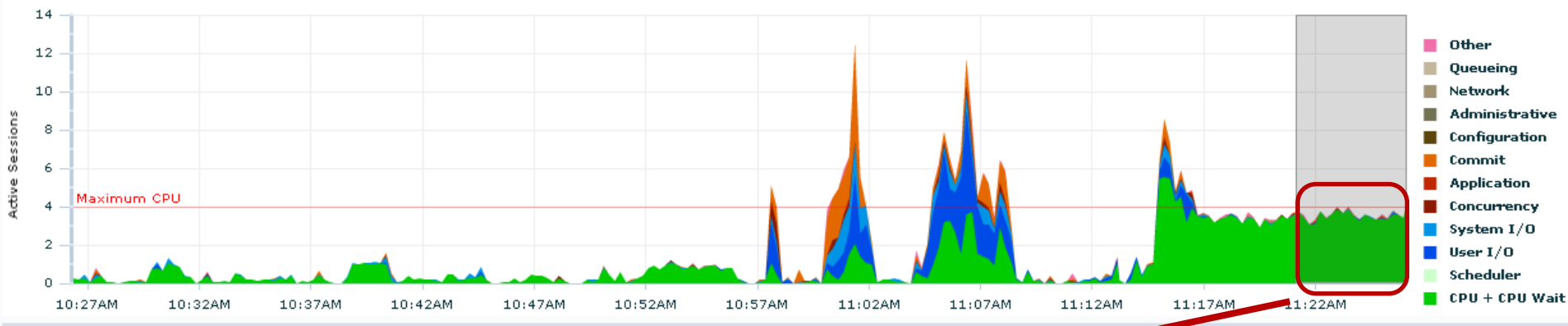
EMREP

Oracle Database Performance Availability Schema Administration

Top Activity

Drag the shaded box to change the time period for the detail section below.

View Data



Detail for Selected 5 Minute Interval

Start Time Sep 12, 2012 11:21:07 AM

Top SQL

Actions Schedule SQL Tuning Advisor Go

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
☐	95.45	77u8m68wh929d	SELECT
☐	.28	ar9vqmz9721yd	SELECT
☐	.28	fkh0ssqax2kyp	PL/SQL EXECUTE
☐	.28	2sahsd9rjzy0q	SELECT
☐	.28	gscqjh84r7wp1	PL/SQL EXECUTE
☐	.28	4yyb4104skrwj	UPDATE
☐	.19	0v8n8nxuud43x	SELECT
☐	.19	2t8b8vn1b7ap3	PL/SQL EXECUTE
☐	.19	9tj16ffqysrzb	PL/SQL EXECUTE
☐	.19	459f3z9u4fb3u	SELECT

Actions Schedule SQL Tuning Advisor Go

Top Sessions

View Top Sessions

Activity (%)	Session ID	User Name	Program
1.02	389	DBSNMP	JDBC Thin Client
.65	25	DBSNMP	OMS
.46	358	SYS	oracle@oem12.intra (LGWR)
.37	14	SYSMAN	oracle@oem12.intra (J000)
.28	144	SYS	sqlplus@oem12.intra (TNS V1-V3)
.28	245	SYS	oracle@oem12.intra (CJQ0)
.19	2	SYS	oracle@oem12.intra (DIA0)
.19	261	SYS	sqlplus@oem12.intra (TNS V1-V3)
.19	139	SYS	sqlplus@oem12.intra (TNS V1-V3)
.19	139	SYS	sqlplus@oem12.intra (TNS V1-V3)

Total Sample Count: 1,076

AWR Instance Report (1)

SQL Statistics

- SQL ordered by Elapsed Time
- SQL ordered by CPU Time
- SQL ordered by User I/O Wait Time
- SQL ordered by Gets
- SQL ordered by Reads
- SQL ordered by Physical Reads (UnOptimized)
- SQL ordered by Executions
- SQL ordered by Parse Calls
- SQL ordered by Sharable Memory
- SQL ordered by Version Count
- Complete List of SQL Text

SQL> @?/rdbms/admin/awrrpt

Back to Top

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 94.0% of Total DB Time (s): 2,356
- Captured PL/SQL account for 1.9% of Total DB Time (s): 2,356

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text
2,169.00	3,555	0.61	92.07	86.29	0.00	77u8m68wh929d	sqlplus@oem12.intra (TNS V1-V3)	SELECT * from DEMO.CUSTOMERS C...
11.77	1	11.77	0.50	15.22	36.56	bc7gjv3ppdtbz	sqlplus@oem12.intra (TNS V1-V3)	BEGIN dbms_workload_repository...
6.72	11	0.61	0.29	53.09	9.88	9tj16ffqysrzb	DBMS_SCHEDULER	DECLARE job BINARY_INTEGER := ...
5.66	9	0.63	0.24	41.80	34.55	0v8n8nxuud43x	Realtime Connection	WITH F AS (select tablespace_n...
5.43	17	0.32	0.23	60.46	0.00	g57kbmvd1gqfk	Realtime Connection	select dbms_sqltune.report_sql...
4.67	1,618	0.00	0.20	53.45	0.00	16x4ffvmkhtbb	DBMS_SCHEDULER	begin :con := "EM_USER_MODEL"....
3.85	3	1.28	0.16	57.93	10.61	3am9cfkvx7gq1	MGMT_COLLECTION.Collection Subsystem	CALL MGMT_ADMIN_DATA.EVALUATE_...
3.77	1	3.77	0.16	17.68	21.34	3nnfb5r7a04v0	emagent_SQL_oracle_database	DECLARE rept varchar2(4000); ...
3.73	1	3.73	0.16	45.82	24.25	2sahsd9rjzy0q	emagent_SQL_oracle_database	select client_name, status fr...
3.30	22	0.15	0.14	49.59	1.77	2t8b8vn1b7ap3	DBMS_SCHEDULER	DECLARE job BINARY_INTEGER := ...

AWR Instance Report (2)

SQL> @?/rdbms/admin/awrrpt

SQL ordered by Gets

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- %Total - Buffer Gets as a percentage of Total Buffer Gets
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Total Buffer Gets: 80,770,320
- Captured SQL account for 98.9% of Total

Buffer Gets	Executions	Gets per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text
79,321,100	3,555	22,312.55	98.21	2,169.00	89.3	0	77u8m68wh929d	sqlplus@oem12.intra (TNS V1-V3)	SELECT * from DEMO.CUSTOMERS C...
355,976	1	355,976.00	0.44	3.73	45.8	24.2	2sahsd9rjzy0q	emagent_SQL_oracle_database	select client_name, status fr...
89,203	9	9,911.44	0.11	5.66	41.8	34.6	0v8n8nxuud43x	Realtime Connection	WITH F AS (select tablespace_n...
47,633	11	4,330.27	0.06	6.72	53.1	9.9	9tj16ffqysrzb	DBMS_SCHEDULER	DECLARE job BINARY_INTEGER := ...
36,621	3	12,207.00	0.05	3.85	57.9	10.6	3am9cfkvx7gq1	MGMT_COLLECTION.Collection Subsystem	CALL MGMT_ADMIN_DATA.EVALUATE_...
34,215	17	2,012.65	0.04	5.43	60.5	0	g57kbmvd1gqfk	Realtime Connection	select dbms_sqltune.report_sql...
27,193	1	27,193.00	0.03	2.63	71.6	.7	ar9vqm29721yd	MGMT_COLLECTION.Collection Subsystem	SELECT NVL(REGISJOB.DISPLAY_NA...
16,490	17	970.00	0.02	3.29	59.6	0	9kddcyq74sg9m	Realtime Connection	SELECT XMLELEMENT("sql_monito...
13,878	660	21.03	0.02	1.92	80.8	0	7k4hpq4x3d0gh	OEM.PbsSystemPool	BEGIN EMDW_LOG.set_context(MGM...
12,558	91	138.00	0.02	0.70	59.6	0	13x2s0fjgha9v	DBMS_SCHEDULER	SELECT * FROM MGMT_PRIVS

- Ermittlung der Ausführungs-Statistiken (Elapsed Time, Buffer Gets, Disk Reads, Executions) aus V$SQL bzw. DBA_HIST_SQLSTAT

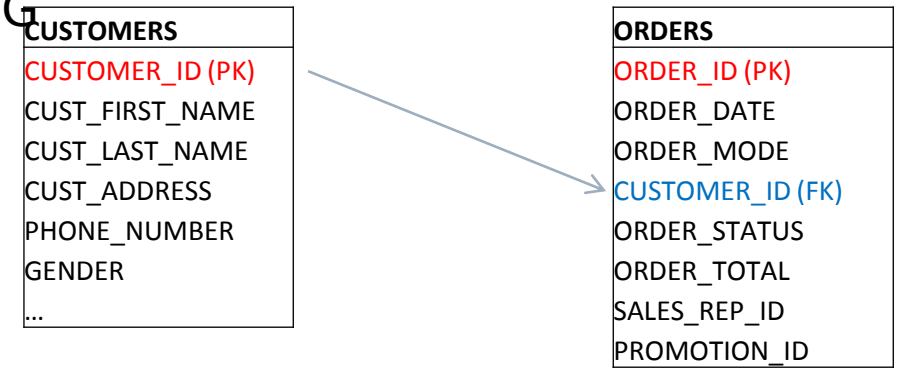
```
SQL> select child_number as child,
            plan_hash_value as phv,
            buffer_gets/executions as gets_per_exe,
            disk_reads/executions as disk_per_exe,
            elapsed_time/executions ela_per_exe,
            executions as exe
     from v$sql where sql_id = '77u8m68wh929d';

 CHILD        PHV     GETS_PER_EXE DISK_PER_EXE ELA_PER_EXE        EXE
---------- ---------- ------------ ------------ ----------- ----------
         0 3311696933   22294.0184   .156943997  702367.964      24856
```

Ermittlung Ausführungsplan inkl. Bind-Variablen

```
SQL> select * from table(dbms_xplan.display_cursor('77u8m68wh929d', 0,'TYPICAL +PEEKED_BINDS'));

SQL_ID  77u8m68wh929d, child number 0
-------------------------------------
SELECT * from DEMO.CUSTOMERS C,                DEMO.ORDERS O WHERE
O.CUSTOMER_ID = C.CUSTOMER_ID -- join predicate       AND C.CUSTOMER_ID
= :v1 -- filter predicate       and O.ORDER_STATUS = 'PENDING' --filter
predicate ORDER BY ORDER_DATE -- sorting

Plan hash value: 3311696933

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |       |       |     8 (100)|          |
|   1 |  SORT ORDER BY                |              |     8 |  1560 |     8  (13)| 00:00:01 |
|   2 |   NESTED LOOPS                |              |     8 |  1560 |     7   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| CUSTOMERS    |     1 |   150 |     2   (0)| 00:00:01 |
|*  4 |     INDEX UNIQUE SCAN         | CUSTOMERS_PK |     1 |       |     1   (0)| 00:00:01 |
|*  5 |    TABLE ACCESS BY INDEX ROWID| ORDERS       |     8 |   360 |     5   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN          | CUSTOMER_IDX |    16 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Peeked Binds (identified by position):
--------------------------------------

   1 - :V1 (NUMBER): 1001

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("C"."CUSTOMER_ID"=:V1)
   5 - filter("O"."ORDER_STATUS"='PENDING')
   6 - access("O"."CUSTOMER_ID"=:V1)
```

Datenmodell für Beispiel

- Setup Script: @01-setup.sql
- CUSTOMERS: 100.000 rows
- ORDERS: 1.000.000 rows
  - CUSTOMER_ID:
    - 70% entfällt auf Hauptkunde
    - 20% auf Zweitkunde
    - 10% auf ca. 63600 verschiedene Kunden
  - ORDER_STATUS:
    - 90% COMPLETED
    - 10% PENDING



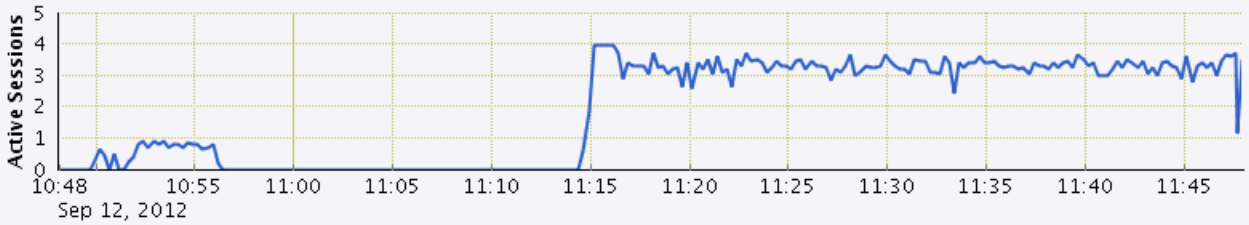
```
SELECT *
from DEMO.CUSTOMERS C, DEMO.ORDERS O
WHERE O.CUSTOMER_ID = C.CUSTOMER_ID
-- join predicate
AND C.CUSTOMER_ID = :v1
-- filter predicate
and O.ORDER_STATUS = 'PENDING'
--filter predicate
ORDER BY ORDER_DATE
-- sorting
```

Details

Select the plan hash value to see the details below. Plan Hash Value 3311696933

Statistics | Activity | Plan | Plan Control | Tuning History | SQL Monitoring

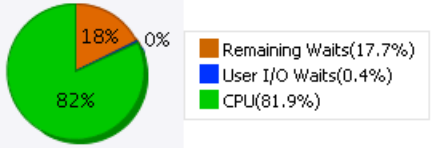
Summary



General

Module	sqlplus@oem12.intra (TNS V1-V3)
Action	
Parsing Schema	SYS
PL/SQL Source (Line Number)	Not Applicable
SQL Profile	n/a
SQL Plan Baseline	n/a

Activity By Waits



Activity By Time

Elapsed Time (sec)	6,865.92
CPU Time (sec)	5,622.99
Wait Time (sec)	1,242.93

Elapsed Time Breakdown

SQL Time (sec)	6,865.92
PL/SQL Time (sec)	0.00
Java Time (sec)	0.00

Shared Cursors Statistics

Total Parses	9,964
Hard Parses	5
Child Cursors	1
Loaded Plans	1
Invalidations	4
Largest Cursor Size (KB)	39.36
All Cursor Size (KB)	39.36

Execution Statistics

	Total	Per Execution	Per Row
Executions	9,964	1	1.00
Elapsed Time (sec)	6,865.92	0.69	0.69
CPU Time (sec)	5,622.99	0.56	0.56
Buffer Gets	222,093,121	22,289.55	22,311.95

Other Statistics

Executions that Fetched all Rows (%)	99.93
Average Persistent Mem (KB)	12.23
Average Runtime Mem (KB)	9.58
Serializable Aborts	0
Remote	No
Obsolete	No
Child Latch Number	0

AWR SQL Report

SQL ID: 77u8m68wh929d

- 1st Capture and Last Capture Snap IDs refer to Snapshot IDs witin the snapshot range
- SELECT * from DEMO.CUSTOMERS C, DEMO.ORDERS O WHERE O.CU...

#	Plan Hash Value	Total Elapsed Time(ms)	Executions	1st Capture Snap ID	Last Capture Snap ID
1	3311696933	2,169,002	3,555	84	84

Back to Top

Plan 1(PHV: 3311696933)

- Plan Statistics
- Execution Plan

Back to Top

Plan Statistics

- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	2,169,002	610.13	92.07
CPU Time (ms)	1,871,534	526.45	91.59
Executions	3,555		
Buffer Gets	79,321,100	22,312.55	98.21
Disk Reads	0	0.00	0.00
Parse Calls	3,555	1.00	13.60
Rows	3,553	1.00	
User I/O Wait Time (ms)	0		
Cluster Wait Time (ms)	0		
Application Wait Time (ms)	0		
Concurrency Wait Time (ms)	0		
Invalidations	0		
Version Count	1		
Sharable Mem(KB)	39		

SQL> @?/rdbms/admin/awrsqrpt

Execution Plan

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT				8 (100)	
1	SORT ORDER BY		8	1560	8 (13)	00:00:01
2	NESTED LOOPS		8	1560	7 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	150	2 (0)	00:00:01
4	INDEX UNIQUE SCAN	CUSTOMERS_PK	1		1 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	ORDERS	8	360	5 (0)	00:00:01
6	INDEX RANGE SCAN	CUSTOMER_IDX	16		2 (0)	00:00:01

Execution Plan

- Begriffe
  - Cardinality (# Rows)
  - Selectivity ([0..1] - Multiplikator: 1 -> 100%, 0 -> 0%)
  - Row Source Operation
  - Access Method: TABLE ACCESS FULL, INDEX RANGE SCAN,…
  - Join Method: Nested Loop, Hash Join, Sort Merge Join, MJC
  - Join Order
  - Predicate Information (*)

```
-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |       |       |     8 (100)|          |
| 6 |   1 |  SORT ORDER BY               |              |     8 |  1560 |     8  (13)| 00:00:01 |
| 5 |   2 |   NESTED LOOPS               |              |     8 |  1560 |     7   (0)| 00:00:01 |
| 2 |   3 |    TABLE ACCESS BY INDEX ROWID| CUSTOMERS    |     1 |   150 |     2   (0)| 00:00:01 |
| 1 |*  4 |     INDEX UNIQUE SCAN        | CUSTOMERS_PK |     1 |       |     1   (0)| 00:00:01 |
| 4 |*  5 |    TABLE ACCESS BY INDEX ROWID| ORDERS       |     8 |   360 |     5   (0)| 00:00:01 |
| 3 |*  6 |     INDEX RANGE SCAN         | CUSTOMER_IDX |    16 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("C"."CUSTOMER_ID"=:V1)
   5 - filter("O"."ORDER_STATUS"='PENDING')
   6 - access("O"."CUSTOMER_ID"=:V1)
```

Optimizer - Inputs

- Datenbank-Version, z.B. 11.2.0.1
- Initialisierungsparameter, z.B. optimizer_features_enable=11.1.0.7.0
- Objekt-Statistiken (Table, Column, Index Statistics gesammelt mit dbms_stats.gather_database|schema|table_stats)
- System-Statistiken (gesammelt mit dbms_stats.gather_system_stats)
- Datenbank-Schema (Tabellenstruktur, vorhandene Indizes, Constraints, etc.)
- Plan Stablity Informationen (Stored Outlines, ab 10g: SQL Profiles, ab 11g: SQL Plan Baselines)
- Cardinality Feedback (ab 11gR2: Rück-Übermittlung der Row-Source-Operation Cardinalities nach Abschluss der Ausführung an den Optimizer)
- aktuelles Datum (z.B. wenn Query die Funktion „sysdate“ enthält)
- **HINTS**

SQL-Tuning - Überlegungen

- Welcher Teil des Ausführungsplans verursacht den höchsten Resourcenverbrauch? (Buffers)
- Genauigkeit Cardinality Estimates?
- Ineffizienter Access-Path bei Nested Loop Join für Inner Loop?
- Kartesisches Produkt? (Merge Join Cartesian)
- Join Order: Startet die Ausführung bei der bestmöglichen Tabelle?
- Join Order: Werden Tabellen gejoined, die keine Join Condition haben?

Reproduktion des Statements

04-customer-lastorders-repro.sql:

```
set termout off
variable v1 number;
exec :v1 := 1001;
SELECT /*+ GATHER_PLAN_STATISTICS */ * from DEMO.CUSTOMERS C,
              DEMO.ORDERS O
WHERE O.CUSTOMER_ID = C.CUSTOMER_ID -- join predicate
      AND C.CUSTOMER_ID = :v1 -- filter predicate
      and O.ORDER_STATUS = 'PENDING' --filter predicate
ORDER BY ORDER_DATE -- sorting
;
set termout on
set lines 300
set pages 1000
select * from table(dbms_xplan.display_cursor(null,null,'ALLSTATS LAST'));
```

Reproduktion des Statements


```
SQL> @04-customer-lastorders-repro.sql

PLAN_TABLE_OUTPUT
------------------------------------------------------------
SQL_ID  41nma72ka65cy, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ * from DEMO.CUSTOMERS C,
    DEMO.ORDERS O WHERE O.CUSTOMER_ID = C.CUSTOMER_ID -- join predicate
      AND C.CUSTOMER_ID = :v1 -- filter predicate       and
O.ORDER_STATUS = 'PENDING' --filter predicate ORDER BY ORDER_DATE --
sorting

Plan hash value: 3311696933

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |      1 |        |      1 |00:00:01.64 |   22300 |
|   1 |  SORT ORDER BY                |               |      1 |      8 |      1 |00:00:01.64 |   22300 |
|   2 |   NESTED LOOPS                |               |      1 |      8 |      1 |00:00:01.64 |   22300 |
|   3 |    TABLE ACCESS BY INDEX ROWID| CUSTOMERS     |      1 |      1 |      1 |00:00:00.01 |       3 |
|*  4 |     INDEX UNIQUE SCAN         | CUSTOMERS_PK  |      1 |      1 |      1 |00:00:00.01 |       2 |
|*  5 |    TABLE ACCESS BY INDEX ROWID| ORDERS        |      1 |      8 |      1 |00:00:01.64 |   22297 |
|*  6 |     INDEX RANGE SCAN          | CUSTOMER_IDX  |      1 |     16 |    700K|00:01:02.62 |    1467 |
-----------------------------------------------------------------------------------------------------------
                                                                          spalten für Ausgabe gekürzt

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("C"."CUSTOMER_ID"=:V1)
   5 - filter("O"."ORDER_STATUS"='PENDING')
   6 - access("O"."CUSTOMER_ID"=:V1)
```

22297 - 1467 = 20830

SQL Tuning – Ideen

- (1) Cardinality Estimates verbessern – Histogram
- (2) Reduzierung der Buffer Gets durch Reorganisation (Sortierung) der Tabelle
- (3) Index-Selektivität verbessern

SQL-Tuning (1) Cardinality


```
SQL> @04-customer-lastorders-repro.sql

PLAN_TABLE_OUTPUT
------------------------------------------------------------
SQL_ID  41nma72ka65cy, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ * from DEMO.CUSTOMERS C,
    DEMO.ORDERS O WHERE O.CUSTOMER_ID = C.CUSTOMER_ID -- join predicate
      AND C.CUSTOMER_ID = :v1 -- filter predicate       and
O.ORDER_STATUS = 'PENDING' --filter predicate ORDER BY ORDER_DATE --
sorting

Plan hash value: 3311696933
```

Id	Operation	Name	**Starts**	**E-Rows**	**A-Rows**	**A-Time**	**Buffers**
0	SELECT STATEMENT		1		1	00:00:01.64	22300
1	SORT ORDER BY		1	8	1	00:00:01.64	22300
2	NESTED LOOPS		1	8	1	00:00:01.64	22300
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	1	1	00:00:00.01	3
* 4	INDEX UNIQUE SCAN	CUSTOMERS_PK	1	1	1	00:00:00.01	2
* 5	TABLE ACCESS BY INDEX ROWID	ORDERS	1	8	1	00:00:01.64	22297
* 6	INDEX RANGE SCAN	CUSTOMER_IDX	1	16	700K	00:01:02.62	1467

```
Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("C"."CUSTOMER_ID"=:V1)
   5 - filter("O"."ORDER_STATUS"='PENDING')
   6 - access("O"."CUSTOMER_ID"=:V1)
```

Starke Abweichung bei Cardinality-Schätzung (16 vs 700.000)
Woher kommt der Wert 16?

SQL-Tuning (2) Cardinality

- DBA_TAB_COL_STATISTICS für Tabelle ORDERS, Spalte customer_id, (@02-col_stats.sql)

```
================================================================================================================================
COLUMN STATISTICS ORDERS
================================================================================================================================
Name            Analyzed    Null?  NDV         Density  # Nulls   # Buckets   Sample    AvgLen  Lo-Hi Values
================================================================================================================================
customer_id     11-SEP-12   N      63600       .000016  0         1           1000000   5       1 | 100000
order_date      11-SEP-12   Y      1000000     .000001  0         1           1000000   8       09/14/2002 09:01:10 |
08:39:41
order_id        11-SEP-12   N      1000000     .000001  0         1           1000000   5       1 | 1000000
order_mode      11-SEP-12   Y      2           .500000  0         1           1000000   7       direct | online
order_status    11-SEP-12   Y      2           .500000  0         1           1000000   10      COMPLETED | PENDING
order_total     11-SEP-12   Y      622208      .000002  0         1           1000000   5       100.01 | 10000
promotion_id    11-SEP-12   Y      100         .010000  0         1           1000000   3       1 | 100
sales_rep_id    11-SEP-12   Y      100         .010000  0         1           1000000   3       1 | 100
```

- Number of Distinct Values (NDV): 63600
- Selectivity: 1 / NDV => 1/ 63600 => 0,0000157 => ~ 0,000016
- Number of Rows (Cardinality):
  1.000.000 * Selectivity 0,000016 => **16 Rows**
- Idee: Histogramm, um dem Optimizer die ungleiche Verteilung bekannt zu machen
- **=> PROBLEM: HISTOGRAMS / BINDS**
- Demo: sqlplus "/as sysdba" @05-binds-histograms.sql

SQL Tuning – Idee 2

- Anzahl der ORDERS Datenblöcke mit CUSTOMER_ID 1001 reduzieren
- Reorganisation der Tabelle und Sortierung nach CUSTOMER_ID, ORDER_STATUS
- ```
  CREATE TABLE DEMO.ORDERS_SORTED
  AS SELECT * FROM DEMO.ORDERS
  ORDER BY CUSTOMER_ID, ORDER_STATUS
  ```
- ```
  sqlplus / as sysdba  @ 06-order-reorg.sql
  ```

SQL Tuning – Idee 2

vor reorg:

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.57	22300	2550
1	SORT ORDER BY		1	8	1	00:00:00.57	22300	2550
2	NESTED LOOPS		1	8	1	00:00:00.57	22300	2550
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	1	1	00:00:00.01	3	0
* 4	INDEX UNIQUE SCAN	CUSTOMERS_PK	1	1	1	00:00:00.01	2	0
* 5	TABLE ACCESS BY INDEX ROWID	ORDERS	1	8	1	00:00:00.57	22297	2550
* 6	INDEX RANGE SCAN	CUSTOMER_IDX	1	16	700K	00:00:06.43	1467	0

nach reorg:

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.36	6331
1	SORT ORDER BY		1	8	1	00:00:00.36	6331
2	NESTED LOOPS		1	8	1	00:00:00.36	6331
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	1	1	00:00:00.01	3
* 4	INDEX UNIQUE SCAN	CUSTOMERS_PK	1	1	1	00:00:00.01	2
* 5	TABLE ACCESS BY INDEX ROWID	ORDERS_SORTED	1	8	1	00:00:00.36	6328
* 6	INDEX RANGE SCAN	ORDERS_SORT_CUSTOMER_IDX	1	16	700K	00:00:05.79	1467

- Verbesserung, aber Ressourcenverbrauch für 1 Zeile als Resultat immer noch zu hoch

SQL-Tuning – Idee 3

- Index-Selektivität verbessern
- ORDER_STATUS: NDV 2 –
  90% COMPLETED / 10% PENDING

```
create index DEMO.ORDERS_CUST_STATUS
on DEMO.ORDERS(ORDER_STATUS,CUSTOMER_ID)
COMPRESS 1;
```

- @ 07-order_status_index1.sql

SQL Tuning – Idee 3

vor Index:

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.57	22300	2550
1	SORT ORDER BY		1	8	1	00:00:00.57	22300	2550
2	NESTED LOOPS		1	8	1	00:00:00.57	22300	2550
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	1	1	00:00:00.01	3	0
* 4	INDEX UNIQUE SCAN	CUSTOMERS_PK	1	1	1	00:00:00.01	2	0
* 5	TABLE ACCESS BY INDEX ROWID	ORDERS	1	8	1	00:00:00.57	22297	2550
* 6	INDEX RANGE SCAN	CUSTOMER_IDX	1	16	700K	00:00:06.43	1467	0

nach Index:

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.01	7
1	SORT ORDER BY		1	16	1	00:00:00.01	7
2	NESTED LOOPS		1	16	1	00:00:00.01	7
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	1	1	00:00:00.01	3
* 4	INDEX UNIQUE SCAN	CUSTOMERS_PK	1	1	1	00:00:00.01	2
5	TABLE ACCESS BY INDEX ROWID	ORDERS	1	16	1	00:00:00.01	4
* 6	INDEX RANGE SCAN	ORDERS_CUST_STATUS	1	16	1	00:00:00.01	3

- Fast perfekt, evtl. noch ORDER_DATE in Index aufnehmen, um Sortierung zu vermeiden

Fazit

- Kurzer Einstieg in komplexes Thema
- Qualifiziertere Aussagen treffen
  - Vorher: statt DB ist langsam
  - Jetzt: SQL_ID xzy verbraucht XXX Buffer Gets
- Next Steps:
  - Hints, SQL Profiles, SQL Plan Baselines
- Scripts unter:

http://www.ora-solutions.net/demos/DOAG2012/2012-K-DB-Decker-Manuelles_Oracle_SQL_Tuning-Scripts.zip

Q & A

Martin Decker
ora-solutions.net

ORACLE 10g Certified Master

ORACLE Certified Master
Oracle Database 11g Administrator

E-Mail: martin.decker@ora-solutions.net
Internet: http://www.ora-solutions.net
Blog: http://www.ora-solutions.net/web/blog/