Aufgabe 1_4_1: Überprüfen Sie die Schemata DOAG auf Objekte mit „Stale Statistics“

Generieren Sie die Befehle zum Sammeln von Statistiken auf diesen Objekten

Lösung:

```
delete from doag.order_line where
order_line_id>8000000;
```

4_generate_stale_objects.sql

```
SET LINES 160
SET PAGES 1000
SET SERVEROUTPUT ON SIZE 1000000
DECLARE
  p_otab DBMS_STATS.OBJECTTAB;
BEGIN
DBMS_STATS.GATHER_SCHEMA_STATS(OWNNAME=>'DOAG',OPTION
S=>'LIST STALE',OBJLIST=>p_otab);
    FOR i in 1 .. p_otab.count
    LOOP
DBMS_OUTPUT.PUT_LINE('DBMS_STATS.GATHER_'||p_otab(i).
objtype||'_STATS(''DOAG'','''||p_otab(i).objname||'''
);');
    END LOOP;
END;
/
```

Aufgabe 1_7_1: Führen Sie einen SQL Healthcheck für das folgende Skript durch (7_sqlhc.sql). Bewerten Sie die Ausgabe.

```
SELECT /*+ DOAG_Berlin */ * FROM  doag.orders o,
doag.customers c, doag.order_line ol
WHERE c.cust_id = o.cust_id
AND o.order_id = ol.order_id
AND c.cust_id = 9000;
```

Lösung:

```
SELECT sql_id, child_number, sql_text FROM v$sql where
sql_text like '%DOAG_Berlin%';

92hqrg8jat64k            0
SELECT /*+ DOAG_Berlin */ * FROM  doag.orders . . .
```

Observations

Observations below are the outcome of several heath-checks on the schema obj
Review them carefully and take action when appropriate. Then re-execute your S

#	Type	Name	Observation
1	CBO PARAMETER	_PGA_MAX_SIZE	CBO initialization parameter "_pga_max_size" with a non-default value of "212980 KB" as per V$SQL_OPTIMIZER_ENV.
2	CBO PARAMETER	_SMM_MAX_SIZE_STATIC	CBO initialization parameter "_smm_max_size_static" with a non-default value of "102400 KB" as per V$SQL_OPTIMIZER_ENV.
3	DBMS_STATS	DBA_AUTOTASK_CLIENT	Automatic gathering of CBO statistics is enabled.
4	DBMS_STATS	SYSTEM STATISTICS	Multi-block read time of 3.58ms seems too small compared to single-block read time of 7.977ms.
#	Type	Name	Observation

Aufgabe1 1_10_1: Optimieren Sie die folgende SQL-Anweisung unter Verwendung eines Hints (10_1.sql):

```
SELECT * FROM  orders o, customers c, order_line ol
WHERE c.cust_id = o.cust_id
AND o.order_id = ol.order_id
AND c.cust_id = 9000;
1000 rows selected.
Elapsed: 00:00:36.67

-------------------------------------------------------------------------------
| Id  | Operation             | Name       | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |            |   998 |    99K|  7687   (14)| 00:00:10
|*  1 |  HASH JOIN            |            |   998 |    99K|  7687   (14)| 00:00:10
|*  2 |   TABLE ACCESS FULL   | ORDERS     |   100 |  3200 |   639   (11)| 00:00:01
|   3 |   MERGE JOIN CARTESIAN|            |    10M|   667M|  6776   (11)| 00:00:09
```

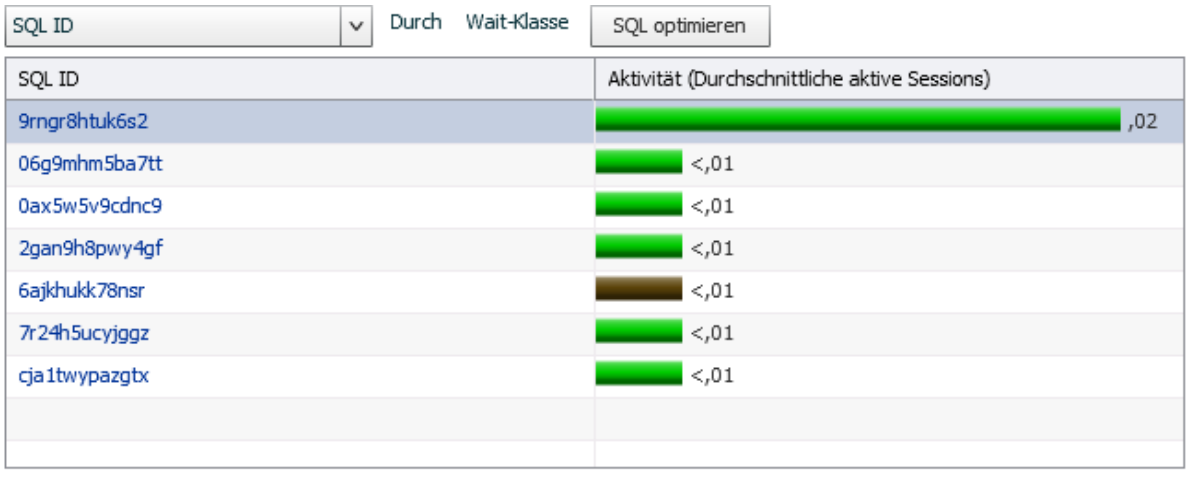
```
|*  4 |    TABLE ACCESS FULL  | CUSTOMERS  |     1 |    35 |     9  (12)| 00:00:01
|   5 |    BUFFER SORT        |            |    10M|   333M|  6767  (11)| 00:00:09
|   6 |     TABLE ACCESS FULL | ORDER_LINE |    10M|   333M|  6767  (11)| 00:00:09
-------------------------------------------------------------------------------
```

Lösung:

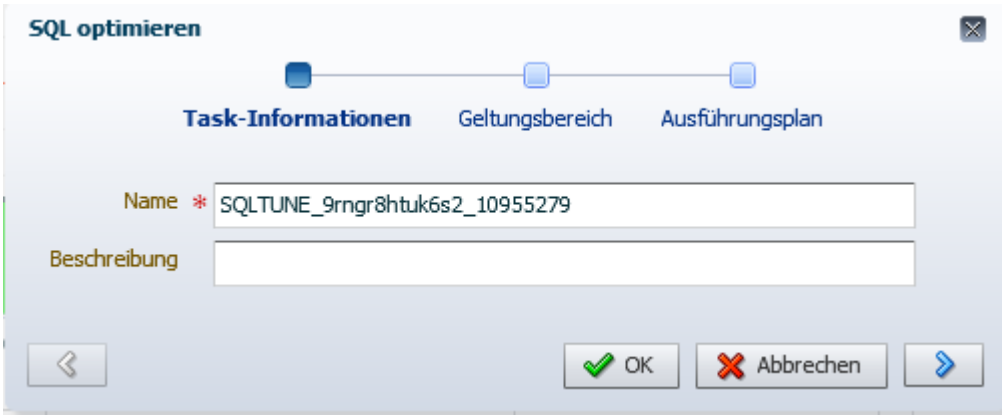
```
SELECT /*+ ORDERED */ *
FROM  orders o, customers c, order_line ol
WHERE c.cust_id = o.cust_id
AND o.order_id = ol.order_id
AND c.cust_id = 9000;
1000 rows selected.
Elapsed: 00:00:02.44

-------------------------------------------------------------------------------
| Id  | Operation           | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |            |   998 |    99K|  1164   (58)| 00:00:06 |
|*  1 |  HASH JOIN          |            |   998 |    99K|  1164   (58)| 00:00:06 |
|*  2 |   HASH JOIN         |            |   100 |  6700 |    90   (49)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| ORDERS     |   100 |  3200 |    87   (50)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| CUSTOMERS  |     1 |    35 |     2    (0)| 00:00:01 |
|   5 |   TABLE ACCESS FULL | ORDER_LINE |    10M|   333M|   906   (50)| 00:00:05 |
-------------------------------------------------------------------------------
```

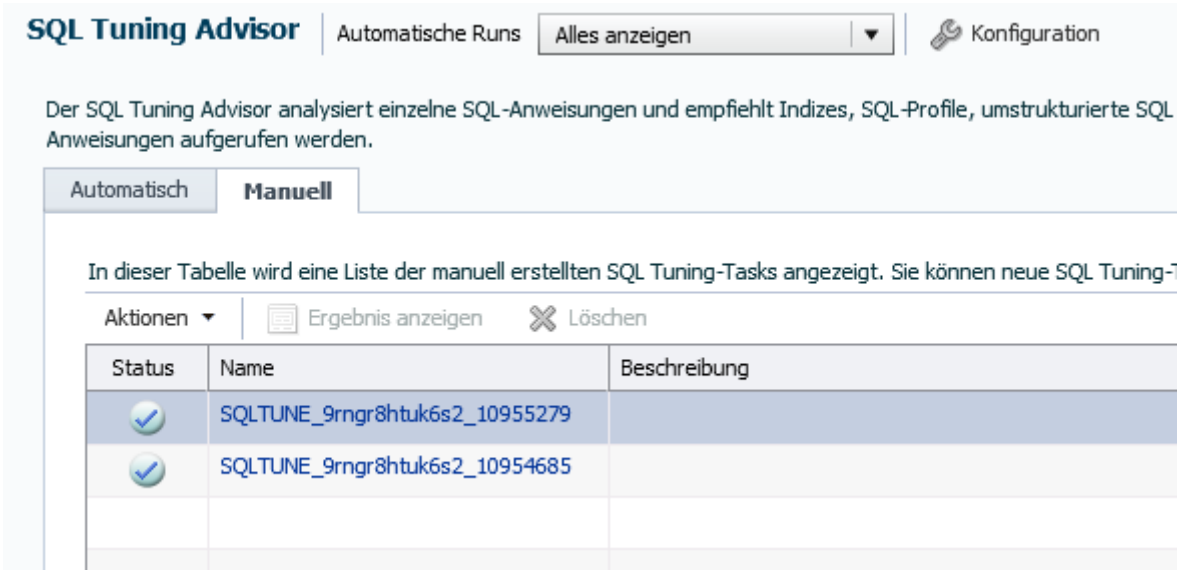
Oracle Enterprise Manager Express> Performance> Performance-Hub> Aktivität



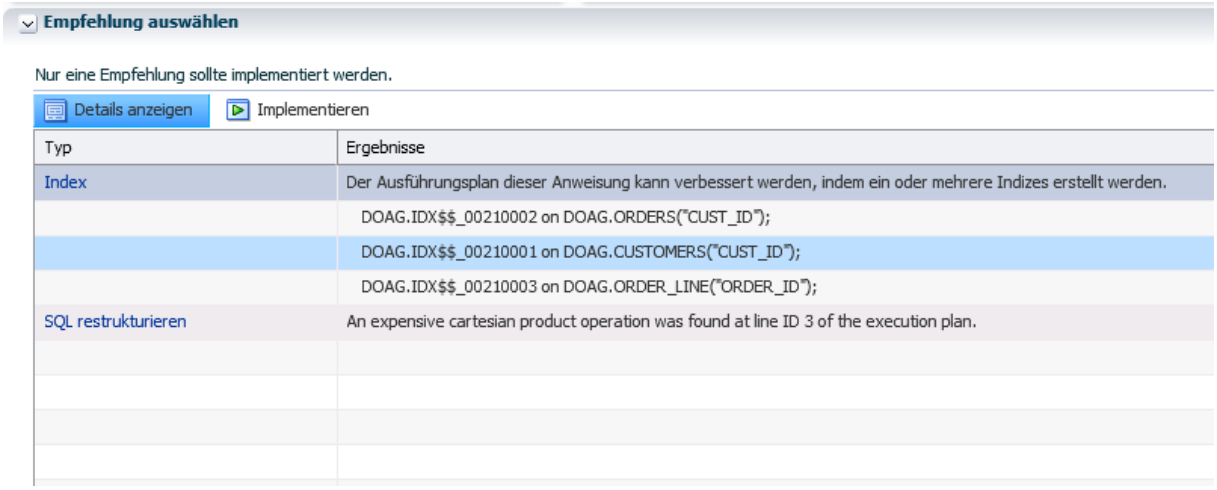
SQL Statement markieren und Button “SQL optimieren” drücken



OK



Empfehlungen:



Neue Indizees:

Empfehlungen:

DOAG.IDX$$_00210002 on DOAG.ORDERS("CUST_ID");

DOAG.IDX$$_00210001 on DOAG.CUSTOMERS("CUST_ID");

DOAG.IDX$$_00210003 on DOAG.ORDER_LINE("ORDER_ID");

SQL restrukturieren:

Empfehlungen:

An expensive cartesian product operation was found at line ID 3 of the execution plan. Consider removing the disconnected table or view from this statement or add a join condition which refers to it.

Aufgabe 1_10_2: Wie kann die folgende Anweisung im Sinne einer SQL-Optimierung verändert werden?

```
SELECT * FROM customers c WHERE NOT EXISTS
(SELECT o.cust_id FROM orders o, order_line ol
WHERE o.cust_id = c.cust_id AND o.order_id = ol.order_id);
no rows selected
Elapsed: 00:00:02.37
```

```
-----------------------------------------------------------------------------------------------
| Id  | Operation           | Name       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |            |   100 |  4800 |       | 13656  (12)| 00:00:17 |
|*  1 |  HASH JOIN ANTI     |            |   100 |  4800 |       | 13656  (12)| 00:00:17 |
|   2 |   TABLE ACCESS FULL | CUSTOMERS  | 10000 |   341K|       |     9  (12)| 00:00:01 |
|   3 |   VIEW              | VW_SQ_1    |  9980K|   123M|       | 13376  (10)| 00:00:17 |
|*  4 |    HASH JOIN        |            |  9980K|   133M|    20M| 13376  (10)| 00:00:17 |
|   5 |     TABLE ACCESS FULL| ORDERS     |  1000K|  8789K|       |   625   (9)| 00:00:01 |
|   6 |     TABLE ACCESS FULL| ORDER_LINE |    10M|    47M|       |  6604   (9)| 00:00:09 |
-----------------------------------------------------------------------------------------------
```

Lösung:

- Es ist ausreichend die Tabelle "orders" abzufragen, wenn kein Eintrag in "orders" dann auch kein Eintrag in "order_line". (Vorauss. Referenzielle Integrität gewahrt)

```
SELECT * FROM customers c
WHERE NOT EXISTS
(SELECT 1 FROM orders o WHERE o.cust_id = c.cust_id);
```

```
no rows selected
Elapsed: 00:00:00.11

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |   100 |  3900 |   662  (13)| 00:00:01 |
|*  1 |  HASH JOIN ANTI    |           |   100 |  3900 |   662  (13)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| CUSTOMERS | 10000 |   341K|     9  (12)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| ORDERS    |  1000K|  3906K|   625   (9)| 00:00:01 |
-------------------------------------------------------------------------------
```

Aufgabe 1_10_3: Passen Sie den Zugriffspfad auf die Daten so an, dass eine möglichst schnelle Rüchgabe des Ergebnisses erfolgt. Legen Sie wenn nötig, zusätzliche Indexe an.

```
SELECT * FROM  orders o, customers c, order_line ol
WHERE c.cust_id = o.cust_id
AND o.order_id = ol.order_id
AND c.cust_id = 9000;
1000 rows selected.
Elapsed: 00:00:32.58

-------------------------------------------------------------------------------
| Id  | Operation            | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |            |   998 |    99K|  7687  (14)| 00:00:10 |
|*  1 |  HASH JOIN           |            |   998 |    99K|  7687  (14)| 00:00:10 |
|*  2 |   TABLE ACCESS FULL  | ORDERS     |   100 |  3200 |   639  (11)| 00:00:01 |
|   3 |   MERGE JOIN CARTESIAN|           |    10M|   667M|  6776  (11)| 00:00:09 |
|*  4 |    TABLE ACCESS FULL | CUSTOMERS  |     1 |    35 |     9  (12)| 00:00:01 |
|   5 |    BUFFER SORT       |            |    10M|   333M|  6767  (11)| 00:00:09 |
|   6 |     TABLE ACCESS FULL| ORDER_LINE |    10M|   333M|  6767  (11)| 00:00:09 |
-------------------------------------------------------------------------------
```

Lösung:

```
DROP INDEX doag.cust_i1;
DROP INDEX doag.order_i2;
DROP INDEX doag.order_line_i1;
CREATE UNIQUE INDEX doag.cust_i1 ON doag.customers(cust_id);
CREATE INDEX doag.order_i2 ON doag.orders(cust_id);
CREATE INDEX doag.order_line_i1 ON doag.order_line(order_id);

SELECT * FROM  orders o, customers c, order_line ol
WHERE c.cust_id = o.cust_id
AND o.order_id = ol.order_id
AND c.cust_id = 9000;
1000 rows selected.
Elapsed: 00:00:00.37

-------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time   |
```

```
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                |   998 |    99K|   306   (1)| 00:00:01
|   1 |  NESTED LOOPS                  |                |       |       |            |
|   2 |   NESTED LOOPS                 |                |   998 |    99K|   306   (1)| 00:00:01
|   3 |    NESTED LOOPS                |                |   100 |  6700 |     5   (0)| 00:00:01
|   4 |     TABLE ACCESS BY INDEX ROWID| CUSTOMERS      |     1 |    35 |     2   (0)| 00:00:01
|*  5 |      INDEX UNIQUE SCAN         | CUST_I1        |     1 |       |     1   (0)| 00:00:01
|   6 |     TABLE ACCESS BY INDEX ROWID| ORDERS         |   100 |  3200 |     3   (0)| 00:00:01
|*  7 |      INDEX RANGE SCAN          | ORDER_I2       |   100 |       |     2   (0)| 00:00:01
|*  8 |    INDEX RANGE SCAN            | ORDER_LINE_I1  |    10 |       |     2   (0)| 00:00:01
|   9 |   TABLE ACCESS BY INDEX ROWID  | ORDER_LINE     |    10 |   350 |     3   (0)| 00:00:01
-----------------------------------------------------------------------------------------------
```

Aufgabe 2_2_1: Stellen Sie die Voraussetzungen her, um Adaptive Cursor Sharing für die folgenden SQL-Anweisung verwenden zu können (2_bind_peeking.sql):

```
SELECT /* DOAG_BIND */ TRUNC(order_date),count(*)
FROM  doag.orders o
WHERE o.order_id > :x GROUP BY TRUNC(order_date);
```

Weißen Sie die Funktionsfähigkeit von ACS in diesem Fall nach.

Führen Sie das SQL mit mindestens zwei verschiedenen Werten aus, so dass unterschiedliche Ausführungspläne verwendet werden.

Lösung:

1. Ausführung der SQL-Anweisung mit dem Wert „9“. Damit würden 999991 Sätze selektiert.

```
VARIABLE x NUMBER;
BEGIN
      :x := 9;
END;
/
print :x
SELECT /*+ DOAG_BIND */ TRUNC(order_date),count(*)
FROM  doag.orders o
WHERE o.order_id > :x GROUP BY TRUNC(order_date);
SELECT * FROM TABLE(dbms_xplan.display_cursor(null,null));

         X
----------
         9
TRUNC(OR   COUNT(*)
-------- ----------
10.05.14     999991
Abgelaufen: 00:00:10.29
```

```
SQL_ID  bc4cqn5ptqcr4, child number 0
-------------------------------------
-------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |       |       |  1440 (100)|          |
|   1 |  HASH GROUP BY     |        |   975 | 12675 |  1440   (3)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| ORDERS |   999K|    12M|  1409   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

Wie erwartet hat der Optimizier ein Bind Peeking durchgeführt und verwedet einen “TABLE ACCESS FULL”, da der überwiegende Teil der Tabelle gelesen werden muss.

2. Überprüfung, ob der Cursor „Bind Sensitive“ ist.

```
SELECT sql_id, sql_text FROM v$sql
WHERE sql_text LIKE '%DOAG_BIND%';
SQL_ID
-------------
bc4cqn5ptqcr4
SELECT /*+ DOAG_BIND */ TRUNC(order_date)
```

```
SELECT sql_id, child_number, is_shareable sharea,
is_bind_sensitive sens, is_bind_aware aware
FROM v$sql WHERE sql_id = 'bc4cqn5ptqcr4';
```

```
SQL_ID        CHILD_NUMBER SHARE SENS  AWARE
------------- ------------ ----- ----- -----
bc4cqn5ptqcr4            0 Y     Y     N
```

Der Cursor ist Shareable und Bind Sensitive und wir erwarten, dass Active Cursor Sharing mit der nächsten Ausführung funktioniert.

3. Ausführung der SQL-Anweisung mit dem Wert „999999“. Damit kommt nur ein Satz in das Result Set.

```
         X
----------
    999999
TRUNC(OR   COUNT(*)
-------- ----------
10.05.14          1
Abgelaufen: 00:00:01.37
SQL_ID  bc4cqn5ptqcr4, child number 0
-------------------------------------

-------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |       |       |  1440 (100)|          |
```

```
|   1 |  HASH GROUP BY     |        |   975 | 12675 |  1440   (3)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| ORDERS |   999K|    12M|  1409   (1)| 00:00:01 |
--------------------------------------------------------------------------------
```

Der Plan zeigt wieder einen Full Table Scan, ohne den Index zu verwenden. Die Abfrage zeigt, dass der Cursor nicht Bind Aware ist.

```
SQL_ID        CHILD_NUMBER SHARE SENS  AWARE
------------- ------------ ----- ----- -----
bc4cqn5ptqcr4            0 Y     Y     N
```

Was wurde übersehen? Eine der Voraussetzungen für ACS ist, dass Histogramme für die zugehörige Spalte gebildet wurden. Eine Überprüfung ergibt, dass die Spalte order_id keine Histogramme besitzt.

```
SELECT column_name,histogram FROM dba_tab_col_statistics
WHERE table_name = 'ORDERS';
OLUMN_NAME  HISTOGRAM
------------ ---------------
ORDER_TEXT   NONE
ORDER_DATE   NONE
CUST_ID      NONE
ORDER_ID     NONE
```

4. Es werden Histogramme gebildet.

```
EXEC dbms_stats.gather_table_stats(ownname=>'DOAG',
tabname=>'ORDERS',METHOD_OPT=>'for all columns size 254',degree=>4);
```

5. Wiederholung der SQL-Ausführung mit dem Wert “999999”.

```
         X
----------
    999999
TRUNC(OR   COUNT(*)
-------- ----------
10.05.14          1
Abgelaufen: 00:00:00.08

PLAN_TABLE_OUTPUT
SQL_ID  bc4cqn5ptqcr4, child number 1

-------------------------------------------------------------------------------------
| Id  | Operation            | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |           |       |       |     4 (100)|          |
|   1 |  HASH GROUP BY       |           |   168 |  2184 |     4   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY IND| ORDERS    |   184 |  2392 |     4   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN  | ORDERS_I1 |   184 |       |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Adaptive Cursor Sharing hat funktioniert. Es wurde ein neuer Child Cursor gebildet mit einem veränderten Plan, der den Index verwendet.

```
SELECT sql_id, child_number, is_shareable sharea,
is_bind_sensitive sens, is_bind_aware aware
FROM v$sql WHERE sql_id = 'bc4cqn5ptqcr4';

SQL_ID        CHILD_NUMBER SHARE SENS  AWARE
------------- ------------ ----- ----- -----
bc4cqn5ptqcr4            0 N     Y     N
bc4cqn5ptqcr4            1 Y     Y     Y
bc4cqn5ptqcr4            2 Y     Y     Y

SELECT child_number, predicate, low, high
FROM v$sql_cs_selectivity WHERE sql_id='bc4cqn5ptqcr4';

CHILD_NUMBER PREDICATE LOW        HIGH
------------ --------- ---------- ----------
           2 >X        0.899828   1.099789
           1 >X        0.000165   0.000202
```

Aufgabe 2_3_1 – SQL-Optimierung:

Führen Sie eine SQL-Optimierung für die folgende Anweisung durch. Der Optimizer hat einen Nested Loop Join gewählt. Finden Sie einen besseren Plan.

```
EXPLAIN PLAN FOR
SELECT c.cust_name, c.cust_city, count(*)
FROM doag2.customers c, doag2.orders o
WHERE o.product = 'Smart TV'
AND c.cust_id != o.cust_id
GROUP BY c.cust_name, c.cust_city;
SELECT * FROM TABLE(dbms_xplan.display());

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |    42 |   163M  (5)| 01:46:38 |
|   1 |  HASH GROUP BY     |           |     1 |    42 |   163M  (5)| 01:46:38 |
|   2 |   NESTED LOOPS     |           |    99G|  3911G|   158M  (1)| 01:42:55 |
|*  3 |    TABLE ACCESS FULL| ORDERS    |    10M|   123M| 17760   (1)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| CUSTOMERS |  9999 |   283K|    16   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Lösung:

Versuch, mit einem Hint einen Hash Join zu bewirken:

```
EXPLAIN PLAN FOR
SELECT /*+ USE_HASH (o c) */ c.cust_name, c.cust_city, count(*)
FROM doag2.customers c, doag2.orders o
WHERE o.product = 'Smart TV'
```

```
AND c.cust_id != o.cust_id
GROUP BY c.cust_name, c.cust_city;
SELECT * FROM TABLE(dbms_xplan.display());

-------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |     1 |    42 |   163M  (5)| 01:46:38 |
|   1 |  HASH GROUP BY      |           |     1 |    42 |   163M  (5)| 01:46:38 |
|   2 |   NESTED LOOPS      |           |    99G|  3911G|   158M  (1)| 01:42:55 |
|*  3 |    TABLE ACCESS FULL| ORDERS    |    10M|   123M| 17760   (1)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| CUSTOMERS |  9999 |   283K|    16   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Der Optimizer akzeptiert den Hint nicht und weigert sich, einen Hash Join auszuführen. Wieso?

- Hash Joins können nur mit Equi Joins verwendet werden
- Hash-Werte, die potentiell für die Vereinigung in Frage kommen, werden in „Buckets“ platziert.
- Im Vereinigungsprozess werden Buckets mit demselben Hash-Wert verglichen.
- Ein Vergleich von Buckets mit unerschiedlichen Hash-Werten findet nicht statt

Umschreiben der SQL-Anweisung, die zu demselben Ergebnis führt:

```
EXPLAIN PLAN FOR
SELECT c.cust_name, c.cust_city, count(*)
FROM doag2.customers c, doag2.orders o
WHERE o.product != 'Smart TV'
AND c.cust_id = o.cust_id
GROUP BY c.cust_name, c.cust_city;
SELECT * FROM TABLE(dbms_xplan.display());

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |    42 | 17761   (1)| 00:00:01 |
|   1 |  HASH GROUP BY               |              |     1 |    42 | 17761   (1)| 00:00:01 |
|   2 |   NESTED LOOPS               |              |       |       |            |          |
|   3 |    NESTED LOOPS              |              |     1 |    42 | 17761   (1)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL        | ORDERS       |     1 |    13 | 17760   (1)| 00:00:01 |
```

```
|*  5 |     INDEX UNIQUE SCAN         | CUSTOMERS_I1 |     1 |       |     0   (0)| 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID| CUSTOMERS    |     1 |    29 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Aufgabe 2_3_2 – Join Elimination:

3_create_view.sql

```
CREATE OR REPLACE VIEW doag2.v_doag AS
SELECT c.cust_id c_id, c.cust_name c_name,
o.order_id o_id, o.product o_product, o.order_date o_date
FROM doag2.customers c, doag2.orders o
WHERE o.cust_id = c.cust_id;


SELECT o_product, o_date, count(*)
FROM doag2.v_doag
GROUP BY o_product, o_date;
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |              |   797 | 19925 | 18147   (3)| 00:00:01 |
|   1 |  HASH GROUP BY        |              |   797 | 19925 | 18147   (3)| 00:00:01 |
|*  2 |   HASH JOIN           |              |    10M|   238M| 17785   (1)| 00:00:01 |
|   3 |    INDEX FAST FULL SCAN| CUSTOMERS_I1 | 10000 | 40000 |     7   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL  | ORDERS       |    10M|   200M| 17743   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Wieso selektiert Oracle von der Tabelle CUSTOMERS?

Lösung:

Anlegen eines Foreign Key Constraints

```
ALTER TABLE doag2.orders ADD CONSTRAINT fk_cust
FOREIGN KEY (cust_id)
REFRENCES doag2.customers;

EXPLAIN PLAN FOR
SELECT o_product, o_date, count(*)
FROM doag2.v_doag
GROUP BY o_product, o_date;
SELECT * FROM table(dbms_xplan.display());
```

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |  1594 | 35068 | 18104   (3)| 00:00:01 |
|   1 |  HASH GROUP BY     |        |  1594 | 35068 | 18104   (3)| 00:00:01 |
```

```
|*  2 |   TABLE ACCESS FULL| ORDERS |    10M|   209M| 17741   (1)| 00:00:01 |
--------------------------------------------------------------------------------
```

Aufgabe 2_8_1 – Subquery Factoring:

8_subquery_factoring.sql

```
UPDATE doag2.customers SET cust_zip = '68735' WHERE cust_id = 1;
COMMIT;
EXEC dbms_stats.gather_table_stats(ownname=>'DOAG2',
tabname=>'CUSTOMERS', method_opt=>'FOR ALL COLUMNS SIZE
254',degree=>4);
WITH ord_cnt AS (
      SELECT cust_id, count(*) AS cnt
      FROM doag2.orders GROUP BY cust_id)
SELECT c.cust_name, oc.cnt
FROM doag2.customers c, ord_cnt oc
WHERE c.cust_id = oc.cust_id
AND c.cust_zip = 68735;
```

Abgelaufen: 00:02:24.48

```
-------------------------------------------------------------------------------
| Id  | Operation           | Name      | Starts | A-Rows |   A-Time   |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |      1 |      2 |00:02:24.48 |
|   1 |  HASH GROUP BY      |           |      1 |      2 |00:02:24.48 |
|*  2 |   HASH JOIN         |           |      1 |   2000 |00:02:24.47 |
|*  3 |    TABLE ACCESS FULL| CUSTOMERS |      1 |      2 |00:00:00.01 |
|   4 |    TABLE ACCESS FULL| ORDERS    |      1 |    10M|00:00:36.79 |
-------------------------------------------------------------------------------
```

Lösung:

3 Optionen:

1. Subquery Factoring ausschalten
2. Einen Index anlegen
3. MATERIALIZE Hint

<u>Option 1:</u>

```
WITH ord_cnt AS (
      SELECT cust_id, count(*) AS cnt
      FROM doag2.orders GROUP BY cust_id)
SELECT /*+ no_merge(oc) */ c.cust_name, oc.cnt
FROM doag2.customers c, ord_cnt oc
```

```
WHERE c.cust_id = oc.cust_id
AND c.cust_zip = 68735;
```

Abgelaufen: 00:00:02.06

```
-------------------------------------------------------------------------------
| Id  | Operation             | Name      | E-Time   | A-Rows |   A-Time   |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |           |          |      2 |00:00:02.06 |
|   1 |  MERGE JOIN           |           | 00:00:01 |      2 |00:00:02.06 |
|   2 |   SORT JOIN           |           | 00:00:01 |    729 |00:00:02.05 |
|   3 |    VIEW               |           | 00:00:01 |  10000 |00:00:01.92 |
|   4 |     HASH GROUP BY     |           | 00:00:01 |  10000 |00:00:01.84 |
|   5 |      TABLE ACCESS FULL| ORDERS    | 00:00:01 |    10M|00:00:00.90 |
|*  6 |   SORT JOIN           |           | 00:00:01 |      2 |00:00:00.01 |
|*  7 |    TABLE ACCESS FULL  | CUSTOMERS | 00:00:01 |      2 |00:00:00.01 |
-------------------------------------------------------------------------------
```

Option 2:

```
CREATE INDEX doag2.i_orders_2 ON doag2.orders(cust_id);

WITH ord_cnt AS (
     SELECT cust_id, count(*) AS cnt
     FROM doag2.orders GROUP BY cust_id)
SELECT c.cust_name, oc.cnt
FROM doag2.customers c, ord_cnt oc
WHERE c.cust_id = oc.cust_id
AND c.cust_zip = 68735;
```

Abgelaufen: 00:00:00.17

```
-------------------------------------------------------------------------------
| Id  | Operation           | Name       | Starts | A-Rows |   A-Time   |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |            |      1 |      2 |00:00:00.17 |
|   1 |  HASH GROUP BY      |            |      1 |      2 |00:00:00.17 |
|   2 |   NESTED LOOPS      |            |      1 |   2000 |00:00:00.12 |
|*  3 |    TABLE ACCESS FULL| CUSTOMERS  |      1 |      2 |00:00:00.01 |
|*  4 |    INDEX RANGE SCAN | I_ORDERS_2 |      2 |   2000 |00:00:00.10 |
-------------------------------------------------------------------------------
```

Option 3:

```
WITH ord_cnt AS (
     SELECT /*+ MATERIALIZE */ cust_id, count(*) AS cnt
     FROM doag2.orders GROUP BY cust_id)
SELECT c.cust_name, oc.cnt
FROM doag2.customers c, ord_cnt oc
WHERE c.cust_id = oc.cust_id
AND c.cust_zip = 68735;
```

Abgelaufen: 00:00:02.08

```
-------------------------------------------------------------------------
| Id  | Operation                | Name             | A-Rows |   A-Time   |
```

```
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                  |      2 |00:00:02.08 |
|   1 |  TEMP TABLE TRANSFORMATION |                  |      2 |00:00:02.08 |
|   2 |   LOAD AS SELECT           |                  |      0 |00:00:01.87 |
|   3 |    HASH GROUP BY           |                  |  10000 |00:00:01.73 |
|   4 |     TABLE ACCESS FULL      | ORDERS           |    10M|00:00:00.83 |
|*  5 |   HASH JOIN                |                  |      2 |00:00:00.21 |
|*  6 |    TABLE ACCESS FULL       | CUSTOMERS        |      2 |00:00:00.01 |
|   7 |    VIEW                    |                  |  10000 |00:00:00.10 |
|   8 |     TABLE ACCESS FULL      | SYS_TEMP_0FD9D662F|  10000 |00:00:00.04 |
-------------------------------------------------------------------------------
```

Aufgabe 2_8_2 - SQL Optimierung:

Führen Sie eine Optimierung für die folgende SQL-Anweisung durch

aufgabe_2_8_2.sql

```
SELECT trunc(o.order_date), ol.product, count(*)
FROM doag.orders o, doag.order_line ol
WHERE ol.order_id = o.order_id
GROUP BY trunc(o.order_date), ol.product;
```

```
Abgelaufen: 00:00:00.23
TRUNC(O. PRODUCT                                            COUNT(*)
-------- ------------------------------------------------ ----------
10.05.14 Smart TV                                         10000000
```

```
-------------------------------------------------------------------------
| Id  | Operation           | Name       | Starts | E-Rows |E-Bytes|
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |            |      1 |        |       |
|   1 |  HASH GROUP BY      |            |      1 |    713 | 19251 |
|*  2 |   HASH JOIN         |            |      1 |   9980K|   256M|
|   3 |    TABLE ACCESS FULL| ORDERS     |      1 |   1000K|    12M|
|   4 |    TABLE ACCESS FULL| ORDER_LINE |      1 |     10M|   133M|
-------------------------------------------------------------------------
```

Lösung:

```
DROP MATERIALIZED VIEW doag.mv_cust;
CREATE MATERIALIZED VIEW doag.mv_cust
BUILD IMMEDIATE
REFRESH FORCE ON DEMAND
ENABLE QUERY REWRITE
AS
SELECT trunc(o.order_date), ol.product, count(*)
FROM doag.orders o, doag.order_line ol
WHERE ol.order_id = o.order_id
GROUP BY trunc(o.order_date), ol.product;
```

```
-------------------------------------------------------------------
| Id  | Operation                      | Name     | Rows  | Bytes |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT               |          |     1 |    20 |
|   1 |  MAT_VIEW REWRITE ACCESS FULL  |  MV_CUST |     1 |    20 |
-------------------------------------------------------------------
```

Aufgabe 2_8_3 - Query Rewrite:

8_query_rewrite.sql

- Wieso funtkioniert Query Rewrite im folgenden Fall nicht?

```
DROP USER doag CASCADE;
CREATE OR REPLACE DIRECTORY exp AS '/home/oracle';
impdp \'/ as sysdba\' directory=exp dumpfile=doag3.dmp full=y


SELECT product, SUM(cnt),
SUM(amount) FROM doag.order_line GROUP BY product

-------------------------------------------------------------------
| Id  | Operation          | Name       | Starts | E-Rows |E-Bytes|
-------------------------------------------------------------------
|   0 | SELECT STATEMENT   |            |      1 |        |       |
|   1 |  HASH GROUP BY     |            |      1 |      1 |    17 |
|   2 |   TABLE ACCESS FULL| ORDER_LINE |      1 |    10M|   162M|
-------------------------------------------------------------------
```

Lösung:

Das Materiralized View ist nicht aktuell (fresh).

```
SQL> SELECT rewrite_enabled,staleness FROM dba_mviews WHERE
mview_name='MV_PROD';
R STALENESS
- -------------------
Y IMPORT

SQL> EXEC DBMS_MVIEW.REFRESH('MV_PROD','C');
PL/SQL procedure successfully completed.

R STALENESS
- -------------------
Y FRESH

-------------------------------------------------------------------------
| Id  | Operation                    | Name    | Starts | E-Rows |E-Bytes|
-------------------------------------------------------------------------
```

```
|   0 | SELECT STATEMENT              |         |      1 |        |       |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| MV_PROD |      1 |      2 |    46 |
-------------------------------------------------------------------------------
```