

ORACLE®

ORACLE®

Why is my SQL slow

Bob Carlin

Consulting Member of Technical Staff

Oracle Real-World Performance



Why is My SQL Slow ?



Problem Query

The screenshot displays the Oracle Real-World Performance tool interface. The top-left pane shows the SQL output of a query, listing car models and their counts. The top-right pane shows the SQL Monitor Report for two SQL statements. The bottom-left pane shows a table of monitored SQL executions. A yellow callout box points to the 'COUNT(*)' result in the SQL output pane.

SQL Output

Pagani	Zonda	50,000,000
Porsche	Carrera GT	50,000,000
Radical	SR8	50,000,000
Rossion	Q1	50,000,000
Tesla	Roadster	50,000,000
Ultima	GTR	50,000,000

27 rows selected.
Elapsed: 00:00:03.90

COUNT(*)
1,200,020,001

Elapsed: 00:00:03.67

BYTES
64,483,229,696

SQL Monitor Report

Two SQL Statements
SLA: 5 seconds

Query 1

```
SELECT  
COUNT(p1.text) "Ferraris",  
COUNT(p2.text) "Ferrari 458s",  
SUM(DECODE(p2.text,NULL,1,0)) "Other Ferraris"  
FROM  
(SELECT owner_id,  
 'Ferraris' "Ferraris"  
 FROM carguys  
 WHERE make = 'Ferrari'  
 ) p1,  
(SELECT owner_id,  
 'Ferrari 458s' "Ferrari 458s"  
 FROM carguys  
 WHERE country = 'Italy'  
 AND make = 'Ferrari'  
 AND model = '458 Italia'  
 ) p2  
WHERE p1.owner_id = p2.owner_id(+);
```

Query 2

```
SELECT  
COUNT(p1.text) "Combos 1",  
COUNT(p2.text) "Combos 2",  
SUM(DECODE(p2.text,NULL,1,0)) Diff  
FROM  
(SELECT owner_id,  
 'Citroens and Minis' "Citroens and Minis"  
 FROM carguys  
 WHERE make IN ('Ferrari','Citroen','Mini')  
 ) p1,  
(SELECT owner_id,  
 'Mini Coopers' "Mini Coopers"  
 FROM carguys  
 WHERE country = 'England'  
 AND make IN ('Citroen','Mini')  
 AND model IN ('Cooper')  
 ) p2  
WHERE p1.owner_id = p2.owner_id(+)
```

Monitored SQL Executions

ID	Description	Status	Duration
1			

Table has 1.2B rows and is 64 GB.

Problem Query

The screenshot displays the Oracle Real-World Performance tool interface. The top window shows the 'SQL Monitor Report' for two SQL statements. The first statement, 'Query 1', is a SELECT query with three columns: 'Ferraris', 'Ferrari 458s', and 'Other Ferraris'. It uses subqueries to filter data from the 'carguys' table based on 'make' and 'country'. The second statement, 'Query 2', is a SELECT query with three columns: 'Combos 1', 'Combos 2', and 'Diff'. It also uses subqueries to filter data from the 'carguys' table based on 'country' and 'make'. The bottom window shows 'Monitored SQL Executions' with a table containing one entry with ID 1. A yellow callout box points to the first subquery of Query 1, stating: 'Query 1 consists of two subqueries. The first subquery finds all of the Ferraris.'

```
SQL Output
Pageni          Zonda          50,000,000
Porsche        Carrera GT     50,000,000
Radical        SR8            50,000,000
Rossion        Q1             50,000,000
Tesla         Roadster       50,000,000
Ultima         GTR            50,000,000

27 rows selected.

Elapsed: 00:00:03.90

-----
COUNT(*)
1,200,020,001

Elapsed: 00:00:03.67

-----
BYTES
64,483,229,696

SQL Monitor Report
Two SQL Statements
SLA: 5 seconds

Query 1
SELECT
COUNT(p1.text)          "Ferraris",
COUNT(p2.text)          "Ferrari 458s",
SUM(DECODE(p2.text,NULL,1,0)) "Other Ferraris"
FROM
(SELECT owner_id,
'Ferraris'              "Ferraris"
FROM carguys
WHERE make = 'Ferrari'
) p1,
(SELECT owner_id,
'Ferrari 458s'          "Ferrari 458s"
FROM carguys
WHERE country = 'Italy'
AND make = 'Ferrari'
AND model = '458 Italia'
) p2
WHERE p1.owner_id = p2.owner_id(+);

Query 2
SELECT
COUNT(p1.text) "Combos 1",
COUNT(p2.text) "Combos 2",
SUM(DECODE(p2.text,NULL,1,0)) Diff
FROM
(SELECT owner_id,
'Citroens and Minis'  "Citroens and Minis"
FROM carguys
WHERE make IN ('Ferrari','Citroen','Mini')
) p1,
(SELECT owner_id,
'Mini Coopers'        "Mini Coopers"
FROM carguys
WHERE country = 'England'
AND make IN ('Citroen','Mini')
AND model IN ('Cooper')
) p2
WHERE p1.owner_id = p2.owner_id(+)
```

ID	Description	Status	Duration
1			

Query 1 consists of two subqueries. The first subquery finds all of the Ferraris.

Problem Query

The screenshot displays the Oracle Real-World Performance tool interface. On the left, the 'SQL Output' pane shows a list of car models and their counts, with 27 rows selected. The 'Elapsed' time is 00:00:03.90. Below this, a 'COUNT(*)' query shows 1,200,020,001 rows and a 'BYTES' query shows 64,483,229,696 bytes. The 'Monitored SQL Executions' pane shows a single execution with ID 1. A yellow callout box points to the second subquery in the 'Query 1' section of the 'SQL Monitor Report' pane, stating: 'The second subquery finds all of the Ferrari 458s.' The 'SQL Monitor Report' pane shows two SQL statements. The first statement, 'Query 1', is a SELECT query with three subqueries. The second subquery, highlighted in green, filters for 'Ferrari 458s' from the 'carguys' table where the country is 'Italy' and the model is '458 Italia'. The second statement, 'Query 2', is a SELECT query with two subqueries. The second subquery, also highlighted in green, filters for 'Mini Coopers' from the 'carguys' table where the country is 'England' and the model is 'Cooper'.

Model	Count
Pagani Zonda	50,000,000
Porsche Carrera GT	50,000,000
Radical SR8	50,000,000
Rossion Q1	50,000,000
Tesla Roadster	50,000,000
Ultima GTR	50,000,000

```
Two SQL Statements
SLA: 5 seconds

Query 1
SELECT
COUNT(p1.text) "Ferraris",
COUNT(p2.text) "Ferrari 458s",
SUM(DECODE(p2.text,NULL,1,0)) "Other Ferraris"
FROM
(SELECT owner_id,
'Ferraris' "Ferraris"
FROM carguys
WHERE make = 'Ferrari'
) p1,
(SELECT owner_id,
'Ferrari 458s' "Ferrari 458s"
FROM carguys
WHERE country = 'Italy'
AND make = 'Ferrari'
AND model = '458 Italia'
) p2
WHERE p1.owner_id = p2.owner_id(+);

Query 2
SELECT
COUNT(p1.text) "Combos 1",
COUNT(p2.text) "Combos 2",
SUM(DECODE(p2.text,NULL,1,0)) Diff
FROM
(SELECT owner_id,
'Citroens and Minis' "Citroens and Minis"
FROM carguys
WHERE make IN ('Ferrari','Citroen','Mini')
) p1,
(SELECT owner_id,
'Mini Coopers' "Mini Coopers"
FROM carguys
WHERE country = 'England'
AND make IN ('Citroen','Mini')
AND model IN ('Cooper')
) p2
WHERE p1.owner_id = p2.owner_id(+)
```

The second subquery finds all of the Ferrari 458s.

Problem Query

The screenshot displays the Oracle Real-World Performance tool interface. The top navigation bar includes 'Real-World Demos', 'Settings', 'Window', 'Layout', 'Axis', 'Command', and 'Login'. The browser address bar shows 'scam10db01.us.oracle.com:8080'. The user is logged in as 'john.zimmerman'. The main window is divided into several panes:

- SQL Output:** Shows the results of a query. The first row is 'Pagani' with a value of '50,000,000'. Below this, there are two rows of aggregated data: 'COUNT(*)' with a value of '1,200,020,001' and 'BYTES' with a value of '64,483,229,696'. Each row is preceded by an 'Elapsed:' time of '00:00:03.90' and '00:00:03.67' respectively.
- SQL Monitor Report:** Displays 'Two SQL Statements' with an 'SLA: 5 seconds'. It contains two queries:
 - Query 1:** A SELECT statement with three columns: 'COUNT(p1.text)', 'COUNT(p2.text)', and 'SUM(DECODE(p2.text,NULL,1,0))'. The values for these columns are 'Ferraris', 'Ferrari 458s', and 'Other Ferraris'. The FROM clause consists of two subqueries, p1 and p2, joined by 'WHERE p1.owner_id = p2.owner_id(+);'. Subquery p1 filters for 'Ferraris' cars, and subquery p2 filters for 'Ferrari 458s' cars in Italy.
 - Query 2:** A SELECT statement with three columns: 'COUNT(p1.text)', 'COUNT(p2.text)', and 'SUM(DECODE(p2.text,NULL,1,0))'. The values are 'Combos 1', 'Combos 2', and 'Diff'. The FROM clause also consists of two subqueries, p1 and p2, joined by 'WHERE p1.owner_id = p2.owner_id(+);'. Subquery p1 filters for 'Citroens and Minis' cars, and subquery p2 filters for 'Mini Coopers' cars in England.
- Monitored SQL Executions:** A table with columns 'ID', 'Description', and 'Duration'. It contains one entry with ID '1'.

Two yellow callout boxes provide additional context:

- The first callout, pointing to the 'COUNT(*)' row in the SQL Output, states: "Outer query performs aggregations."
- The second callout, pointing to the 'WHERE p1.owner_id = p2.owner_id(+);' clause in the SQL Monitor Report, states: "Outer query joins the results of the subqueries."

Problem Query

Query 2 is the same but has different predicate values.

SQL Output

Pagani	Zonda	50,000,000
Por		
Rad		
Ros		
Tes		
Ult		
27		

Elapsed: 00:00:03.90

COUNT(*)	
1,200,020,001	

Elapsed: 00:00:03.67

BYTES	
64,483,229,696	

Monitored SQL Executions

ID	Description	Status
1		

SQL Monitor Report

Two SQL Statements
SLA: 5 seconds

Query 1

```
SELECT
COUNT(p1.text) "Ferraris",
COUNT(p2.text) "Ferrari 458s",
SUM(DECODE(p2.text,NULL,1,0)) "Other Ferraris"
FROM
(SELECT owner_id,
'Ferraris' "Ferraris"
FROM carguys
WHERE make = 'Ferrari'
) p1,
(SELECT owner_id,
'Ferrari 458s' "Ferrari 458s"
FROM carguys
WHERE country = 'Italy'
AND make = 'Ferrari'
AND model = '458 Italia'
) p2
WHERE p1.owner_id = p2.owner_id(+);
```

Query 2

```
SELECT
COUNT(p1.text) "Combos 1",
COUNT(p2.text) "Combos 2",
SUM(DECODE(p2.text,NULL,1,0)) Diff
FROM
(SELECT owner_id,
'Citroens and Minis' "Citroens and Minis"
FROM carguys
WHERE make IN ('Ferrari','Citroen','Mini')
) p1,
(SELECT owner_id,
'Mini Coopers' "Mini Coopers"
FROM carguys
WHERE country = 'England'
AND make IN ('Citroen','Mini')
AND model IN ('Cooper')
) p2
WHERE p1.owner_id = p2.owner_id(+)
```


Default Statistics



Default Statistics

The screenshot displays the Oracle Enterprise Manager interface. On the left, the SQL Output window shows a query with three categories: 'Ferraris', 'Ferrari 458s', and 'Other Ferraris'. The results table shows 50,000,001 rows for 'Ferraris', 50,000,000 for 'Ferrari 458s', and 1 for 'Other Ferraris'. The 'Monitored SQL Executions' table lists two queries, with the first one, 'Default Statistics', having a duration of 49 seconds. A yellow callout box points to this entry with the text: 'Query 1 takes 49 seconds with default statistics'. The 'SQL Monitor Report' on the right provides detailed execution statistics, including a duration of 50.0s and various performance metrics.

Query 1 takes 49 seconds with default statistics

Default Statistics

SQL Monitor Report

More on OTN
ORACLE Enterprise Manager
Active Reports

Monitored SQL Execution Details

Overview

SQL ID: 0myxh0vwtz7
Parallel: 32
Execution started: Thu Sep 12, 2013 10:30:44 AM
Last Refresh Time: Thu Sep 12, 2013 10:31:34 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 50.0s
Database Time: 18.5m
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 962K
IO Bytes: 215GB
Call Offload Efficiency: 32%

Details

Plan Hash Value: 1046826780

Plan Statistics | Plan | Parallel | Activity | Metrics

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estimated Rows	Cost	Timeline(50s)	Executio...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1					1.29	
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						-41
PX SEND QC (RANDOM)	!TQ10001	1			32	32						
SORT AGGREGATE		1			32	32						
HASH JOIN RIGHT OUTER		46M	148K		32	50M	35GB	49GB	838K		64	84
PX RECEIVE		190K	74K		32	1,600M				9.9		13
PX SEND BROADCAST	!TQ10000	190K	74K		32	1,600M				14		1.23
PX BLOCK ITERATOR		190K	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	190K	74K		412	50M		62K		99	64	2.05
PX BLOCK ITERATOR		46M	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	46M	74K		412	50M		62K		99	26	

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Default Statistics

1. Development Findings

- Baseline Performance for Query 1
- Query 1 exceeds target

Initial Optimization Steps— More Predicate Values



More Predicate Values

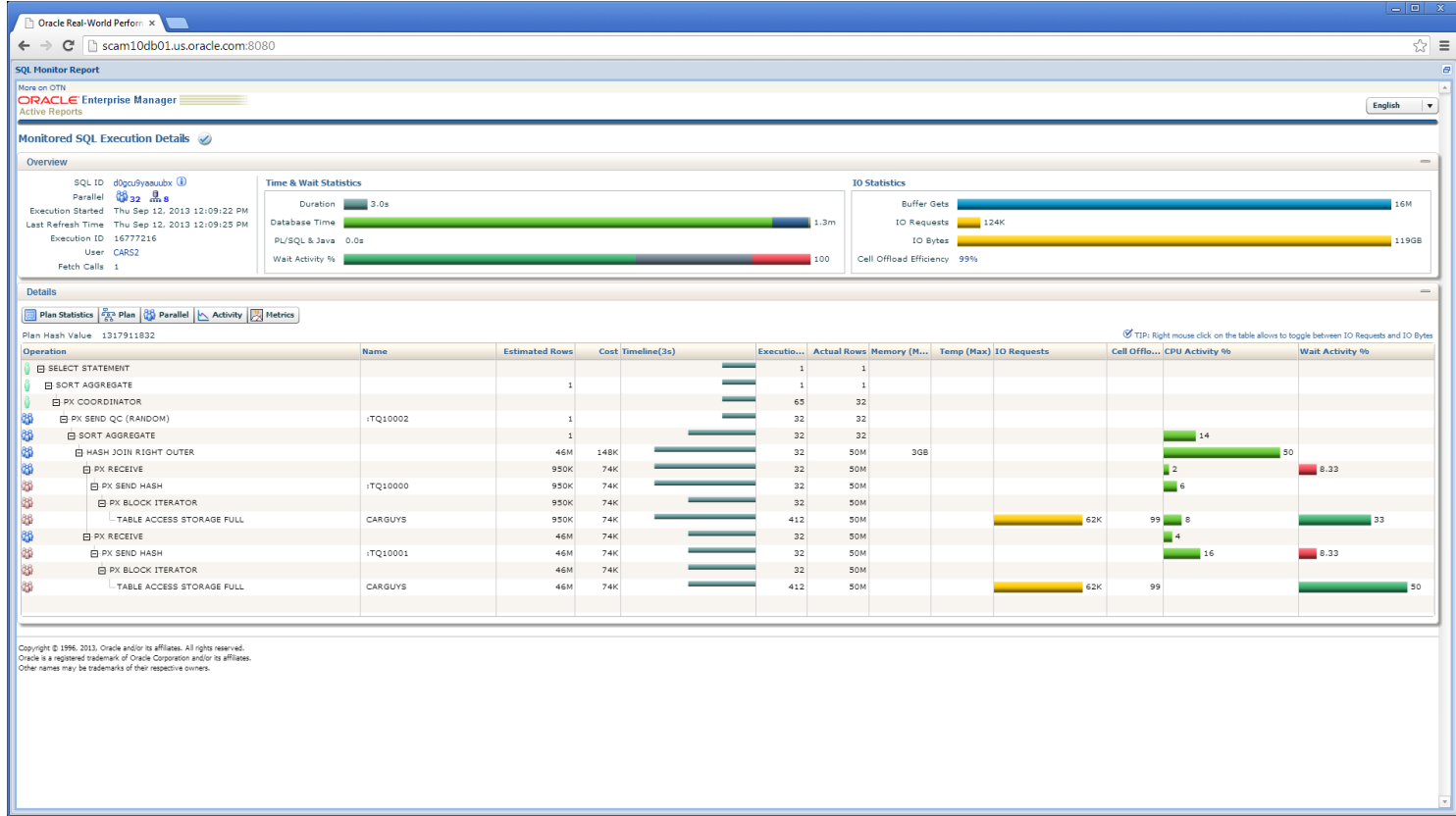
The screenshot displays the Oracle Real-World Performance tool interface. On the left, the SQL Output window shows a query with a WHERE clause containing multiple predicate values: 'Ferrari', 'Lotus', 'BMW', 'Porsche', and 'McLaren'. The results table shows 50,000,001 rows for 'Ferraris', 50,000,000 rows for 'Ferrari 458s', and 1 row for 'Owner Ferraris'. The elapsed time is 00:00:02.54.

The right pane shows the SQL Monitor Report for the query. The 'Overview' section indicates a duration of 3.0s, 1.3m database time, and 124K IO requests. The 'Details' section shows a table of execution statistics for various operations, including SELECT STATEMENT, SORT AGGREGATE, and TABLE ACCESS.

Increase the list of predicate values

Now query takes 3 seconds

More Predicate Values



2. Development Findings

- Query runs faster just by changing the list of values in the select list
- Plan changed from a broadcast to a hash distribution due to the higher but inaccurate cardinality estimate
- Get correct plan with wrong cardinality estimate—can lead to inconsistent plans and performance

Initial Optimization Steps— Increase Degree of Parallelism



Degree of Parallelism

Change DoP from 32 to 128

Now query takes 2 seconds

```
7 from (select owner_id,
8         'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select owner_id,
12        'Ferrari 458s' as text
13      from carguys
14      where country = 'Italy'
15      and make = 'Ferrari'
16      and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)
```

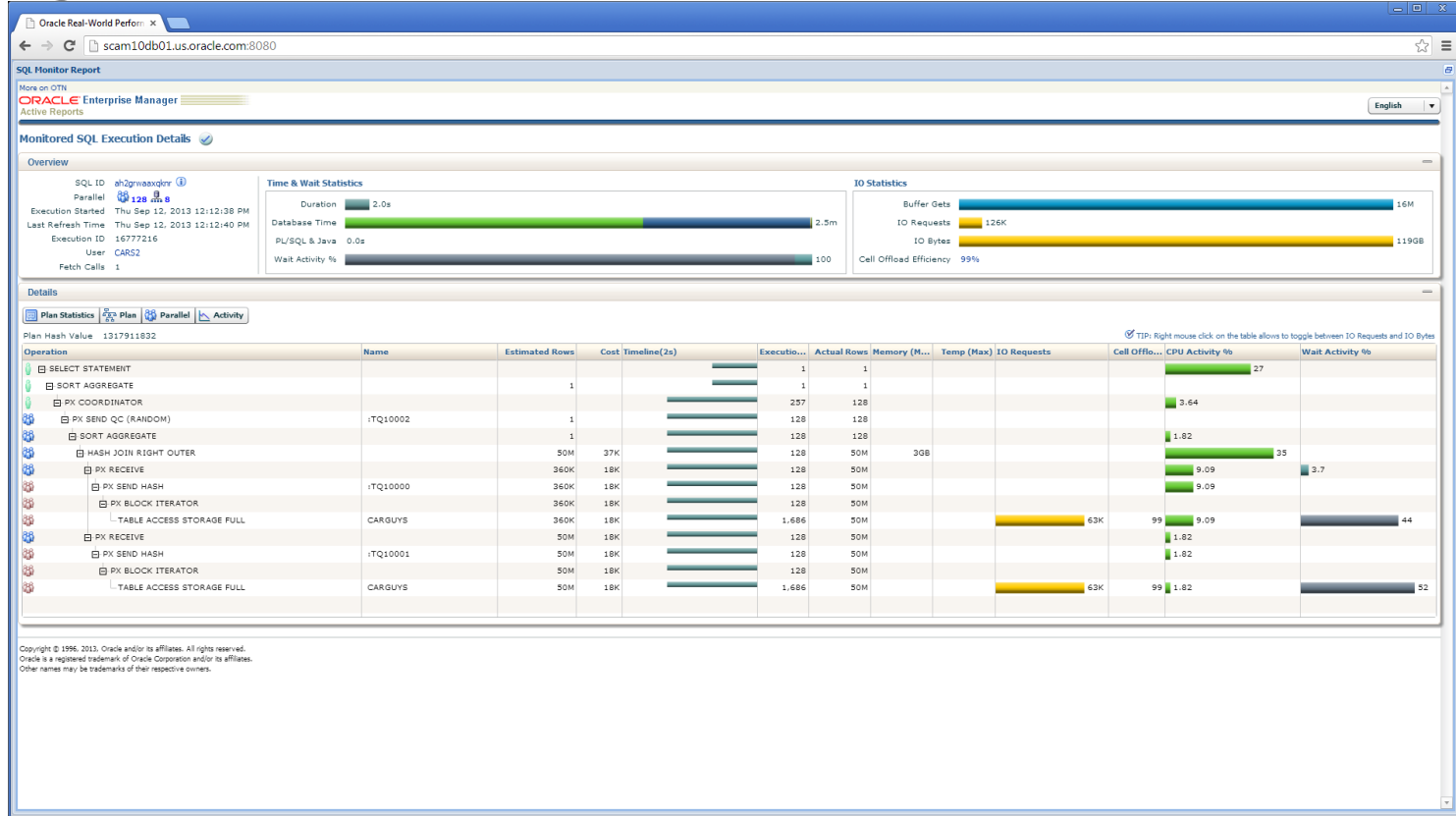
	Ferraris	Ferrari 458s	Other Ferraris
	50,000,001	50,000,000	1

Elapsed: 00:00:01.63
end of step

ID	Description	Status	Duration
1	Default Statistics	✓	49
2	Add more predicate values	✓	3
3	Change DoP	✓	2
4			

Operation	Name	Estim...	Time(2s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT			1	1						27	
SORT AGGREGATE			1	1							
PX COORDINATOR			257	128						3.64	
PX SEND QC (RANDOM)	!TQ10002		128	128						1.82	
SORT AGGREGATE			128	128							
HASH JOIN RIGHT O...			128	50M	30B					35	
PX RECEIVE			128	50M						9.09	3.7
PX SEND HASH	!TQ10000		128	50M						9.09	
PX BLOCK ITER...			128	50M							
TABLE ACCES...	CARGUYS		1,68	50M				63K	99	9.09	44
PX RECEIVE			128	50M						1.82	
PX SEND HASH	!TQ10001		128	50M						1.82	
PX BLOCK ITER...			128	50M							
TABLE ACCES...	CARGUYS		1,68	50M				63K	99	1.82	52

Degree of Parallelism



3. Development Findings

- Changing DoP from 32 to 128 improves performance and meets the target; 4X more resources yields a 25X performance improvement
- Plan has changed from a broadcast distribution to a hash distribution due to DoP change
- DoP is a resource management technique, not a query tuning tool

Indexes



4. Development Findings—Indexes

- Indexes on columns:
 - owner_id
 - country
 - make
 - model
 - country, make, model

Indexes

The screenshot displays the Oracle Enterprise Manager interface. On the left, the 'SQL Output' window shows the execution of a query with the following SQL:

```
1,200,020,001
Elapsed: 00:00:03.70
1 select /*+ MONITOR */ count(*)
2 from (select owner_id,
3        'Ferrari 458s' as text
4        from carguys
5        where country = 'Italy'
6        and make = 'Ferrari'
7*       and model = '458 Italia')
COUNT(*)
50,000,000
Elapsed: 00:00:00.52
end of step
```

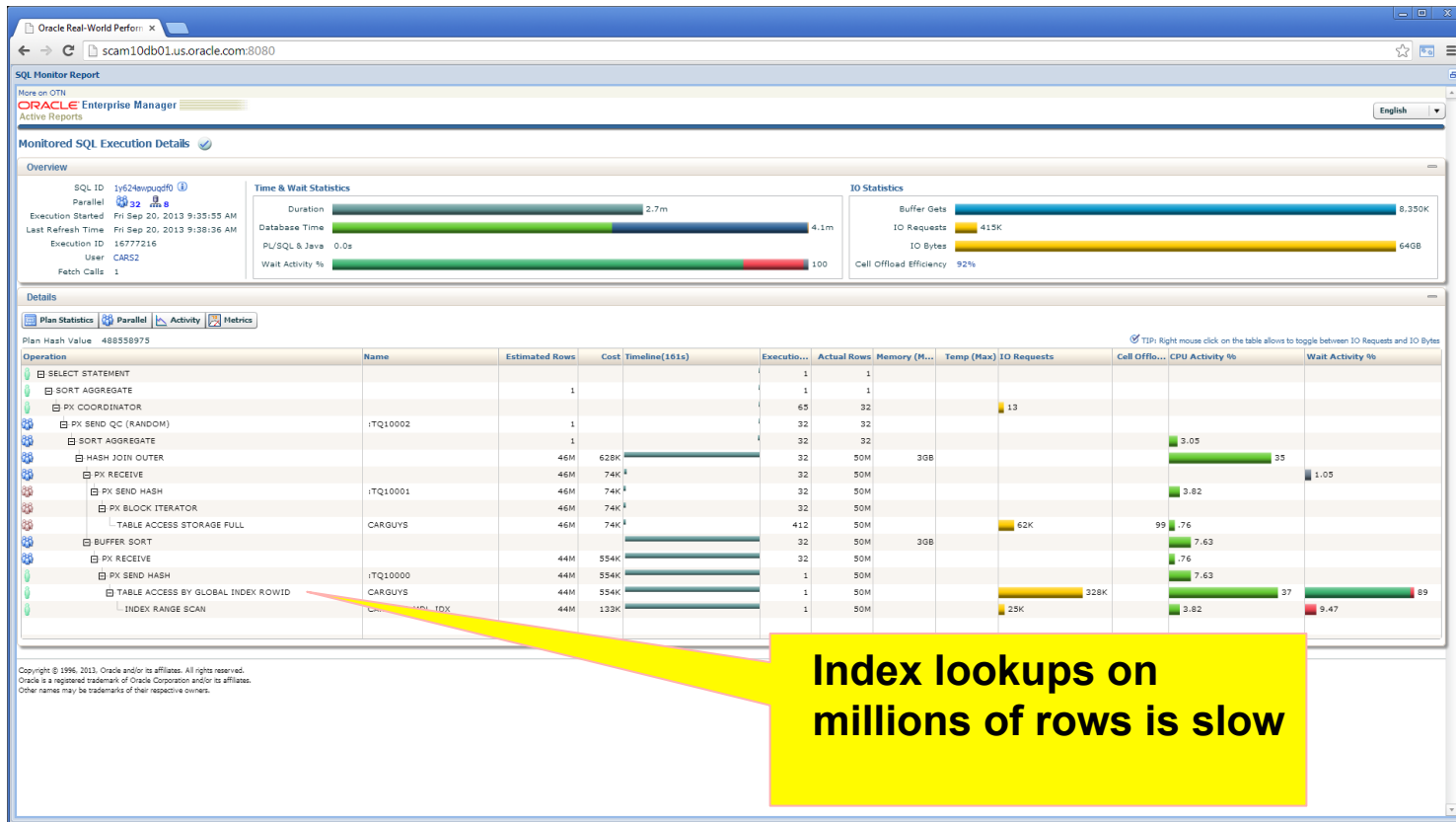
Below the SQL output is the 'Monitored SQL Executions' table:

ID	Description	Status	Time
1	Default Statistics	⚠	49
2	Add more predicate values	✅	3
3	Change DoP	✅	2
4	Indexes	✅	160
5			

A yellow callout box points to the 'Indexes' step (ID 4) with the text: **Add indexes and query takes longer—160 seconds!**

The 'SQL Monitor Report' window on the right shows 'Monitored SQL Execution Details' for SQL ID '1y624wvpuqdf0'. It includes 'Overview' with 'Time & Wait Statistics' (Duration: 2.7m, Database Time: 4.1m) and 'IO Statistics' (Buffer Gets: 8,350K, IO Requests: 415K, IO Bytes: 64GB). The 'Details' section shows a table of operations with columns for Name, Estim., Cost, Timeline, Exec., Actuals, Mem., Temp., IO, CPU, and Wait Activity.

Indexes



4. Development Findings—Indexes

- Not understanding the big/little data challenge
- Indexes are not efficient for operations on a large numbers of rows
- Full table scan is faster with predictable performance

To Index or Not

- Indexing is an OLTP technique for operations on a small number of rows
- A table scan may consume more resources but it will be predictable no matter how many rows are returned
- Indexes impact DML operations
- If I/O bandwidth went from 70MB/sec to 70GB/sec would you change your optimization/execution strategy?

To Index or Not

- Index driven query retrieving 1,000,000 rows
 - Assume the index is cached and the data is not.
 - 1,000,000 random IOPS @ 5ms per I/O
 - This would require 5000 Seconds (or over 1 hour) to Execute
 - How much data could you scan in 5000 Seconds with a fully sized I/O system able to scan 25 GB/Sec ?
 - Over 100 TB !

Histograms



Histograms

The screenshot displays the Oracle Enterprise Manager interface. On the left, the SQL Output window shows a query with three categories: Ferraris, Ferrari 458s, and Other Ferraris. The results show 50,000,001 rows for Ferraris, 50,000,000 for Ferrari 458s, and 1 for Other Ferraris. The elapsed time is 00:00:50.74. Below the output is a table of Monitored SQL Executions.

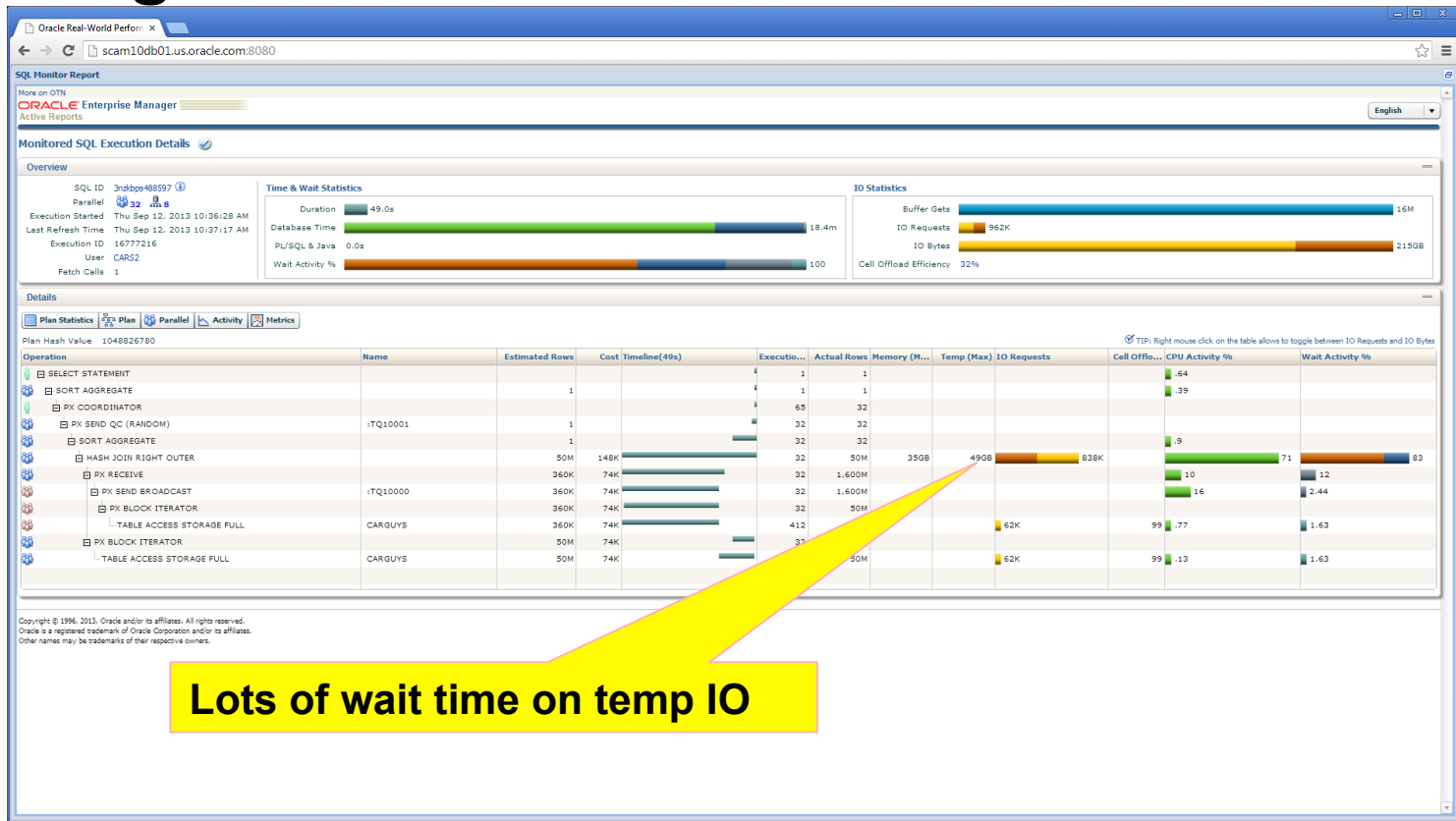
ID	Description	Status	Duration
1	Default Statistics	✓	49
2	Add more predicate values	✓	3
3	Change DoP	✓	2
4	Indexes	✓	160
5	Stats with Histograms	✓	51
6			

The right pane shows the SQL Monitor Report for SQL ID 3n2kops-88597. It includes an Overview section with Time & Wait Statistics (Duration: 49.0s, Database Time: 18.4m, Wait Activity %: 100) and IO Statistics (Buffer Gets: 16M, IO Requests: 362K, IO Bytes: 215GB, Cell Offload Efficiency: 32%). Below is a Details section with a Plan Statistics tab showing a table of operations with columns for Name, Estimation, Cost, and Time.

Operation	Name	Estim...	C...	Timeline(49s)	Me...	Te...	IO Reque...	Cel...	CPU Activit...	Wait Activit...
SELECT STATEMENT										
SORT AGGREGATE		1						.64		.39
PX COORDINATOR										
PX SEND QC (RANDOM)	!TQ10001	1								
SORT AGGREGATE		1						.9		
HASH JOIN RIGHT O...		50M	148				838K	71		83
PX RECEIVE		360K	74K					10		12
PX SEND BROADC...	!TQ10000	360K	74K					16		2.44
PX BLOCK ITER...		360K	74K							
TABLE ACCESS ...	CARGUYS	360K	74K					99	.77	1.63
PX BLOCK ITERATOR		50M	74K							
TABLE ACCESS S...	CARGUYS	50M	74K					99	.13	1.63

A yellow callout box points to the Duration column in the Monitored SQL Executions table, containing the text: "Rerun stats to get histograms—no change in plan or run time".

Histograms



Lots of wait time on temp IO

5. Development Findings

- Re-gathered stats to automatically create histograms
 - Frequency histograms on country, make and model columns
- No change in plan—query still exceeds target

Flash Temp



Flash Temp

The screenshot displays the Oracle Enterprise Manager interface. On the left, the 'SQL Output' window shows a query and its results. The query filters for Ferrari cars, comparing 'Ferraris' and 'Ferrari 458s' against 'Other Ferraris'. The results show 50,000,001 rows for Ferraris, 50,000,000 for Ferrari 458s, and 1 for Other Ferraris. Below the output, a table lists 'Monitored SQL Executions' with 7 steps, each marked with a green checkmark and a duration.

The 'SQL Monitor Report' on the right provides a detailed overview of the query execution. It includes 'Time & Wait Statistics' (Duration: 30.0s, Database Time: 16.5m, Wait Activity %: 100) and 'IO Statistics' (Buffer Gets: 16M, IO Requests: 362K, IO Bytes: 215GB, Cell Offload Efficiency: 55%). The 'Details' section shows a table of operations with columns for Name, Estimation, CPU Time, and Wait Activity.

SQL Output

```

7 from (select owner_id,
8         'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select owner_id,
12        'Ferrari 458s' as text
13      from carguys
14      where country = 'Italy'
15      and make = 'Ferrari'
16      and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)
-----
Ferraris      Ferrari 458s      Other Ferraris
50,000,001      50,000,000              1
  
```

Elapsed: 00:00:29.77
end of step

ID	Description	Status	Duration
1	Default Statistics	✓	49
2	Add more predicate values	✓	3
3	Change DoP	✓	2
4	Indexes	✓	160
5	Stats with Histograms	✓	51
6	Flash Temp	✓	30
7			

SQL Monitor Report

More on OTN
ORACLE Enterprise Manager
Active Reports

Monitored SQL Execution Details

Overview

SQL ID: ghd611z962s
Parallel: 32
Execution Started: Thu Sep 12, 2013 10:38:48 AM
Last Refresh Time: Thu Sep 12, 2013 10:39:18 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 30.0s
Database Time: 16.5m
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 362K
IO Bytes: 215GB
Cell Offload Efficiency: 55%

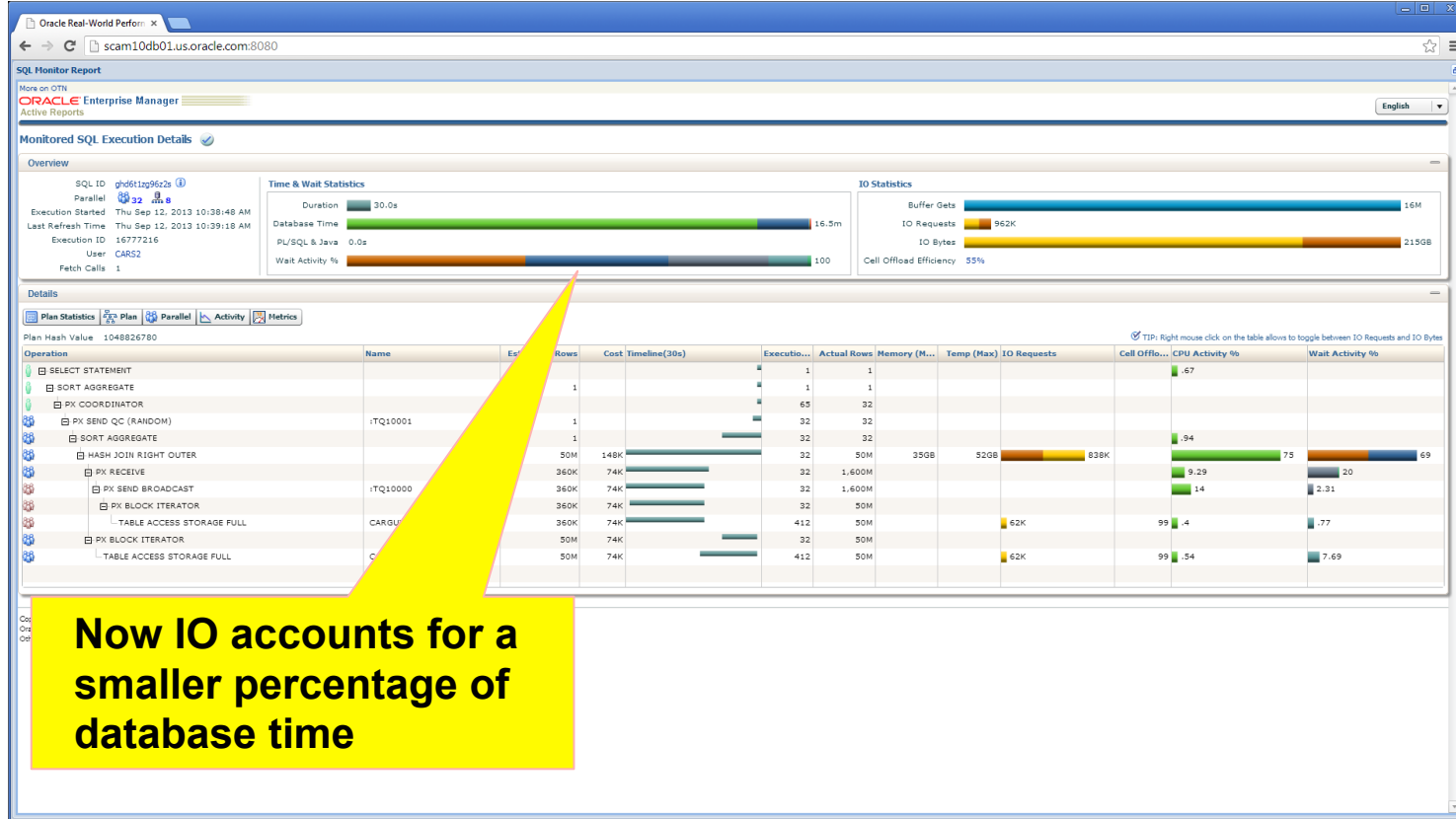
Details

Plan Hash Value: 1048826780

Operation	Name	Estim...	C...	Timeline(30s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1				1					.67	
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	ITQ10001	1				32	32					
SORT AGGREGATE		1				32	32					
HASH JOIN RIGHT O...		50M	148			32	50M	350B	520B	838K	.94	.75
PX RECEIVE		360K	74K			32	1,600				9.29	.20
PX SEND BROADC...	ITQ10000	360K	74K			32	1,600				.14	2.01
PX BLOCK ITER...		360K	74K			32	50M					
TABLE ACCE...	CARGUYS	360K	74K			412	50M		62K	99	.4	.77
PX BLOCK ITERATOR		50M	74K			32	50M					
TABLE ACCESS S...	CARGUYS	50M	74K			412	50M		62K	99	.54	7.69

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Flash Temp



Now IO accounts for a smaller percentage of database time

6. Development Findings

- Most of the wait time was spent performing IO on temp, so move temp to flash disks
- Improved performance but still does not meet target
- Not a good use of flash
- Incorrect use of tools/products

Manual Memory Parameters



Manual Memory Parameters

The screenshot displays the Oracle Enterprise Manager interface. On the left, the 'SQL Output' window shows a query and its results. The query filters for 'Ferraris' and 'Ferrari 458s' models. The results table shows 50,000,001 rows for Ferraris, 50,000,000 for Ferrari 458s, and 1 for Other Ferraris. Below the output, a table lists 'Monitored SQL Executions' with 9 steps, all marked as successful (green checkmarks).

The right pane shows the 'SQL Monitor Report' for the query. It includes an overview with execution statistics: Duration (35.0s), Database Time (18.2m), IO Requests (124K), and IO Bytes (119GB). Below this is a detailed execution plan table.

Operation	Name	Estim...	C...	Timeline(35s)	Ex...	Act...	Me...	Te...	IO Reque...	Cel...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1					1.3	
SORT AGGREGATE		1				1	1				2	
PX COORDINATOR					65	32						3.85
PX SEND QC (RANDOM)	ITQ10001	1				32	32					
SORT AGGREGATE		1				32	32				3	
HASH JOIN RIGHT O...		50M	148			32	50M	640B			76	
PX RECEIVE		360K	74K			32	1,600				10	46
PX SEND BROADC...	ITQ10000	360K	74K			32	1,600				12	15
PX BLOCK ITER...		360K	74K			32	50M					
TABLE ACCESS ...	CARGUYS	360K	74K			412	50M		62K	99	2	
PX BLOCK ACCE...		50M	74K			32	50M					
TABLE ACCESS S...	CARGUYS	50M	74K			412	50M		62K	99	5	35

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Manual Memory Parameters

Very little IO in database time

Poor cardinality estimate—360K estimated rows vs 50M actual rows

Increased memory size manually—now there is no use of temp

Time & Wait Statistics

Duration	35.0s
Database Time	18.2m
PL/SQL & Java	0.0s
Wait Activity %	100

IO Statistics

Buffer Gets	16M
IO Requests	124K
IO Bytes	119GB
Call Offload Efficiency	99%

Details

Operation	Name	Estimated Rows	Cost	Timeline(25s)	Executi...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1				1.3		
SORT AGGREGATE		1			1	1				1.2		
PX COORDINATOR					65	32						3.85
PX SEND QC (RANDOM)	!TQ10001	1			32	32						
SORT AGGREGATE		1			32	32				1.3		
HASH JOIN RIGHT OUTER		50M	148K		32	50M	64GB			76		
PX RECEIVE		360K	74K		32	1,600M				10		46
PX SEND BROADCAST	!TQ10000	360K	74K		32	1,600M				12		15
PX BLOCK ITERATOR		360K	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	360K	74K		413	50M			62K	99	1.2	
PX BLOCK ITERATOR		50M	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	50M	74K		412	50M			62K	99	1.5	35

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Oth...

7. Development Findings

- Set `sort_area_size` and `hash_area_size` to 2G
- Eliminated temp usage but still did not meet target
- Memory is allocated per parallel server process, which can quickly exceed resources
- Moving to a solution before understanding the problem

Cardinality Estimates



Cardinality Estimates

The screenshot displays the Oracle Enterprise Manager interface. On the left, the 'SQL Output' window shows a query with a cardinality hint and its execution results. On the right, the 'SQL Monitor Report' window provides detailed performance metrics and a plan table.

SQL Output:

```

7 from (select owner_id,
8         'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select /*+ cardinality(carguys 50000000) */ owner_id,
12        'Ferrari 458s' as text
13      from carguys
14      where country = 'Italy'
15      and make = 'Ferrari'
16      and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)
-----
Ferraris      Ferrari 458s      Other Ferraris
50,000,001      50,000,000              1
  
```

Elapsed: 00:00:02.70
end of step

Monitored SQL Executions:

ID	Description	Status	Duration
1	Default Statistics	✓	49
2	Add more predicate values	✓	3
3	Change DoP	✓	2
4	Indexes	✓	160
5	Stats with Histograms	✓	51
6	Flash Temp	✓	30
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9			

SQL Monitor Report Details:

Overview:

- SQL ID: fpt6wqmqd9
- Parallel: 32
- Execution Started: Thu Sep 12, 2013 10:58:59 AM
- Last Refresh Time: Thu Sep 12, 2013 10:59:02 AM
- Execution ID: 16777216
- User: CAR52
- Fetch Calls: 1

Time & Wait Statistics:

- Duration: 2.0s
- Database Time: 1.4m
- PL/SQL & Java: 0.0s
- Wait Activity %: 100

IO Statistics:

- Buffer Gets: 16M
- IO Requests: 124K
- IO Bytes: 119GB
- Cell Offload Efficiency: 99%

Plan Hash Value: 1211196936

Operation	Name	Estim...	C...	Timeline(3s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1				1	1					
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10002	1				32	32					
SORT AGGREGATE		1				32	32				6.41	
HASH JOIN OUTER		50M	14B			32	50M	30B			67	
PX RECEIVE		50M	74K			32	50M					1.28
PX SEND HASH	!TQ10000	50M	74K			32	50M					6.41
PX BLOCK ITER...		50M	74K			32	50M					
TABLE ACCES...	CARGUYS	50M	74K			412	50M		62K	99	5.13	33
PX RECEIVE		50M	74K			32	50M					2.56
PX SEND HASH	!TQ10001	50M	74K			32	50M					5.13
PX BLOCK ITER...		50M	74K			32	50M					
TABLE ACCES...	CARGUYS	50M	74K			412	50M		62K	99	6.41	67

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Cardinality Estimates

The screenshot displays the Oracle Enterprise Manager SQL Monitor Report interface. At the top, it shows the 'Monitored SQL Execution Details' for a specific SQL ID. Below this, there are sections for 'Overview', 'Time & Wait Statistics', and 'IO Statistics'. The 'Details' section is expanded to show a table of operations with columns for Name, Estimated Rows, Cost, Timeline, Execution, Actual Rows, Memory, Temp, IO Requests, Call Offlo..., CPU Activity %, and Wait Activity %.

Operation	Name	Estimated Rows	Cost	Timeline(3s)	Executio...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1						
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	ITQ10002	1			32	32						
SORT AGGREGATE		1			32	32						
HASH JOIN OUTER		50M	148K		32	50M	3GB			6.41	67	
PX RECEIVE		50M	74K		32	50M				1.28		
PX SEND HASH	ITQ10000	50M	74K		32	50M				6.41		
PX BLOCK ITERATOR		50M	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	50M	74K		413	50M			62K	99	5.13	33
PX RECEIVE		50M	74K		32	50M				2.96		
PX SEND HASH	ITQ10001	50M	74K		32	50M				5.13		
PX BLOCK ITER		50M	74K		32	50M						
TABLE ACCESS	CARGUYS	50M	74K		412	50M			62K	99	6.41	67

Copyright © 1996, 2013, Oracle and/or its affiliates. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. All other marks contained herein are the property of their respective owners.

Plan switches from a broadcast to a hash distribution

Use cardinality hint to specify correct number of rows

8. Cardinality Hint

- SQL Monitor showed poor cardinality estimates
- Cardinality hint gives optimizer the correct number of rows for the table scan
- Plan changed from a broadcast to hash distribution
- Query time now meets target
- Now temp is not an issue

Disable Broadcast Distribution



Disable Broadcast Distribution

The screenshot displays the Oracle Real-World Performance tool interface. The main window shows the execution of a SQL query with the following output:

```

7 from (select owner_id,
8       'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select owner_id,
12       'Ferrari 458s' as text
13       from carguys
14       where country = 'Italy'
15       and make = 'Ferrari'
16       and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)

```

The output is summarized in a table:

	Ferraris	Ferrari 458s	Other Ferraris
	50,000,001	50,000,000	1

Elapsed: 00:00:02.53
end of step

Monitored SQL Executions

ID	Description	Status	Duration
2	Add more predicate values	✓	3
3	Change DoP	✓	2
4	Indexes	✓	160
5	Stats with Histograms	✓	51
6	Flash Temp	✓	30
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10			

SQL Monitor Report

More on OTN: [ORACLE Enterprise Manager](#)

Active Reports: English

Monitored SQL Execution Details

Overview

SQL ID: dvkuxfxfongp
Parallel: 32
Execution Started: Fri Sep 20, 2013 10:15:13 AM
Last Refresh Time: Fri Sep 20, 2013 10:15:16 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 3.0s
Database Time: 1.2m
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 124K
IO Bytes: 119GB
Cell Offload Efficiency: 99%

Details

Plan Statistics | Plan | Parallel | Activity | Metrics

Plan Hash Value: 1317911832

Operation | Name | Estim... | C... | Timeline(3s) | Ex... | Act... | Me... | Te... | IO Reque... | Cel... | CPU Activit... | Wait Activit...

Operation	Name	Estim...	C...	Timeline(3s)	Ex...	Act...	Me...	Te...	IO Reque...	Cel...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10002	1			32	32						
SORT AGGREGATE		1			32	32					12	
HASH JOIN RIGHT O...		50M	14B		32	50M	30B				61	
PX RECEIVE		360K	74K		32	50M						17
PX SEND HASH	!TQ10000	360K	74K		32	50M						5.08
PX BLOCK ITER...		360K	74K		32	50M						
TABLE ACCES...	CARGUYS	360K	74K		412	50M			62K	99	5.08	33
PX RECEIVE		50M	74K		32	50M						
PX SEND HASH	!TQ10001	50M	74K		32	50M						5.08
PX BLOCK ITER...		50M	74K		32	50M						
TABLE ACCES...	CARGUYS	50M	74K		412	50M			62K	99	6.78	50

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Disable Broadcast Distribution

The screenshot displays the Oracle Enterprise Manager SQL Monitor Report for a specific SQL execution. The report includes an overview section with key statistics such as Duration (3.0s), Database Time (1.2m), and IO Statistics (Buffer Gets: 16M, IO Requests: 124K, IO Bytes: 1190B). Below this is a detailed execution plan table. A yellow callout box points to the 'TABLE ACCESS STORAGE FULL' operation in the plan, which is highlighted in yellow in the original image.

Operation	Name	Estimated Rows	Cost	Timeline(3s)	Executio...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1						
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10002	1			32	32						
SORT AGGREGATE		1			32	32					12	
HASH JOIN RIGHT OUTER		50M	148K		32	50M	3GB				61	17
PX RECEIVE		360K	74K		32	50M						
PX SEND HASH	!TQ10000	360K	74K		32	50M					5.08	
PX BLOCK ITERATOR		360K	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	360K	74K		412	50M			62K	99	5.08	33
PX RECEIVE		50M	74K		32	50M					5.08	
PX SEND HASH	!TQ10001	50M	74K		32	50M					5.08	
PX BLOCK ITERATOR		50M	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	50M	74K		412	50M			62K	99	6.78	50

Copyright © 1996-2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Disable broadcast distribution and now we have the hash distribution as with the cardinality hint

9. Development Findings

- Googling reveals a hidden parameter to disable broadcast distribution
- Plan and run times are similar to cardinality hint, meeting target
- Moving to a solution before understanding the problem

Second Query with Broadcast Distribution Disabled



Query 2: Broadcast Distribution Disabled

The screenshot displays the Oracle Real-World Performance tool interface. The main window shows the SQL output for Query 2, which compares 'Combos 1' and 'Combos 2' with a 'DIFF' column. The output shows 150,010,001 rows for Combos 1, 100,000,000 for Combos 2, and a difference of 50,010,001. The elapsed time is 00:00:16.79.

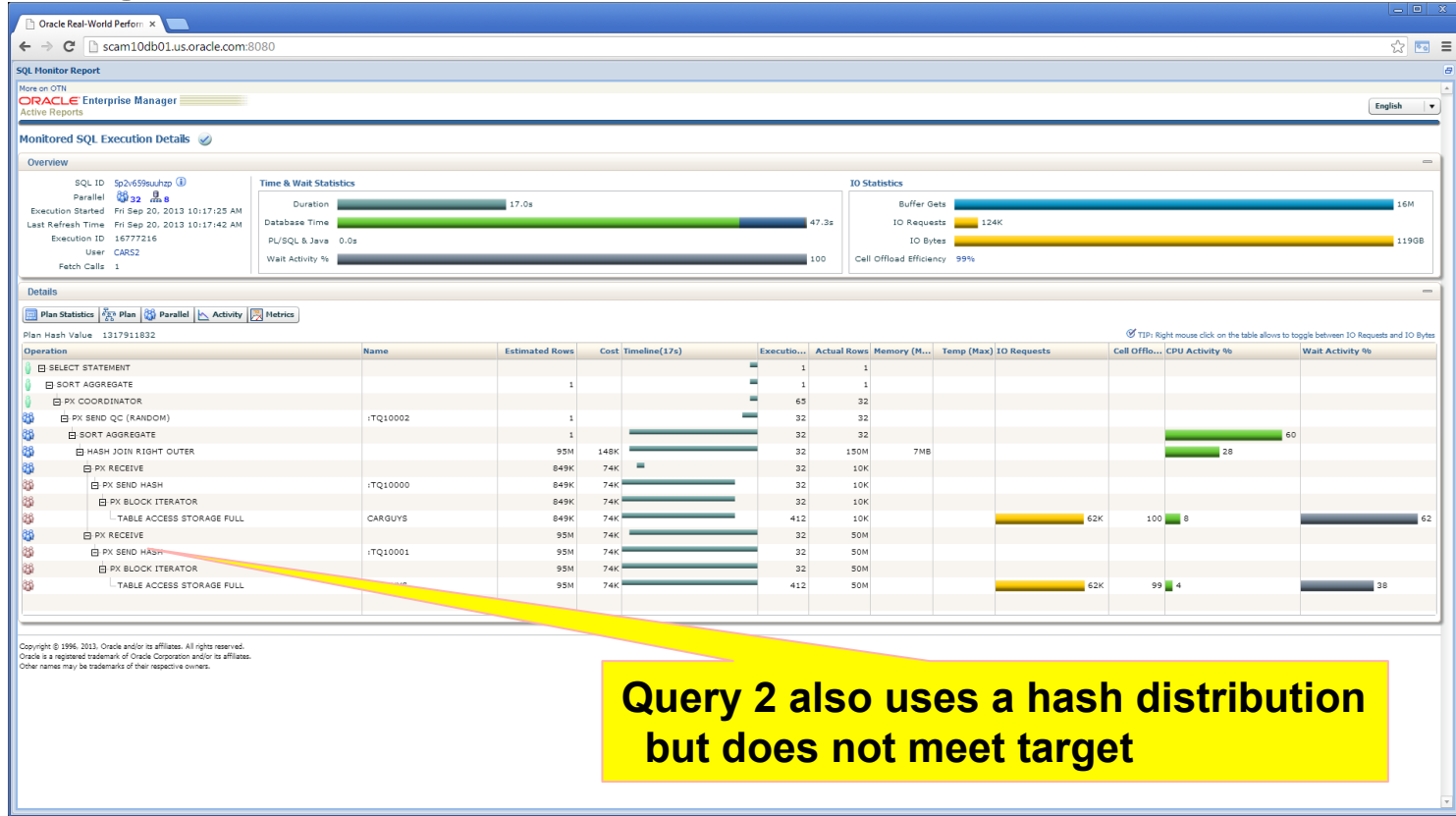
Below the SQL output is a table titled 'Monitored SQL Executions' with the following data:

ID	Description	Status	Duration
3	Change DoP	✓	2
4	Indexes	✓	160
5	Stats with Histograms	✓	51
6	Flash Temp	✓	30
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11			

The right-hand pane shows the 'SQL Monitor Report' for the query. It includes an overview of the SQL ID (5p2v659uu4hp), execution statistics (Duration: 17.0s, Database Time: 47.3s), and IO Statistics (Buffer Gets: 16M, IO Requests: 124K, IO Bytes: 119GB). The details section shows a plan hash value of 1317911632 and a table of operations with their respective statistics.

Operation	Name	Estim...	C...	Timeline(17s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1				1	1					
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10002	1				32	32					
SORT AGGREGATE		1				32	32				60	
HASH JOIN RIGHT O...		95M	146			32	150M	7MB			28	
PX RECEIVE		849K	74K			32	10K					
PX SEND HASH	!TQ10000	849K	74K			32	10K					
PX BLOCK ITER...		849K	74K			32	10K					
TABLE ACCES...	CARGUYS	849K	74K			412	10K		62K	100	8	62
PX RECEIVE		95M	74K			32	50M					
PX SEND HASH	!TQ10001	95M	74K			32	50M					
PX BLOCK ITER...		95M	74K			32	50M					
TABLE ACCES...	CARGUYS	95M	74K			412	50M		62K	99	4	38

Query 2: Broadcast Distribution Disabled



Query 2 also uses a hash distribution but does not meet target

Query 2: Broadcast Distribution Disabled

10. Development Findings

- Plan uses a hash distribution
- Exceeds target

Second Query with Broadcast Distribution Enabled



Query 2: Broadcast Distribution Enabled

The screenshot displays the Oracle Real-World Performance tool interface. On the left, the SQL output shows a query comparing two different broadcast distribution settings. The 'Diff' query results are as follows:

	Combos 1	Combos 2	DIFF
150,010,001	100,000,000	50,010,001	

The elapsed time for the query is 00:00:02.46. Below the SQL output, a table lists monitored SQL executions:

ID	Description	Status	Duration
4	Indexes	✓	160
5	Stats with Histograms	✓	51
6	Flash Temp	✓	30
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11	Query 2 Reset Parameter	✓	2
12			

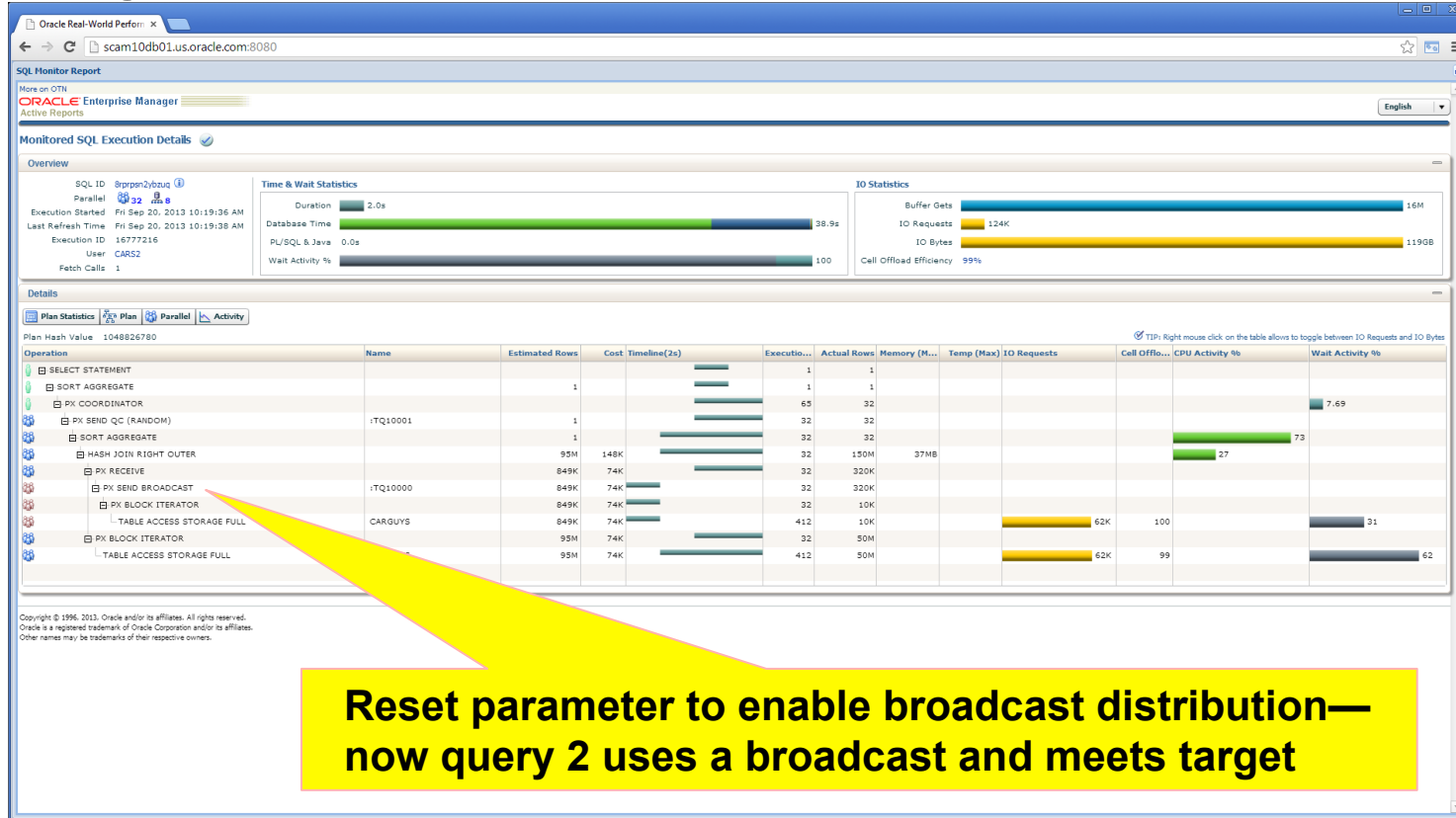
On the right, the SQL Monitor Report provides detailed execution statistics for the query. The 'Overview' section shows:

- SQL ID: 8prpn2ybzq
- Parallel: 32
- Execution Started: Fri Sep 20, 2013 10:19:36 AM
- Database Time: 38.9s
- IO Statistics: Buffer Gets (16M), IO Requests (124K), IO Bytes (119GB), Cell Offload Efficiency (99%)

The 'Details' section shows the execution plan with the following key operations:

Operation	Name	Estim...	C...	Timeline(2s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						7.69
PX SEND QC (RANDOM)	ITQ10001	1			32	32						
SORT AGGREGATE		1			32	32						73
HASH JOIN RIGHT O...		95M	14B		32	150M	37MB					27
PX RECEIVE		849K	74K		32	320K						
PX SEND BROADC...	ITQ10000	849K	74K		32	320K						
PX BLOCK ITER...		849K	74K		32	10K						
TABLE ACCE...	CARGUYS	849K	74K		412	10K			62K	100		31
PX BLOCK ITERATOR		95M	74K		32	50M						
TABLE ACCESS S...	CARGUYS	95M	74K		412	50M			62K	99		62

Query 2: Broadcast Distribution Enabled



Query 2: Broadcast Distribution Enabled

11. Development Findings

- Reset `_parallel_broadcast_enabled`
- Plan now uses a broadcast distribution
- Meets target
- Should not change system parameters to tune one query

Extended Stats



Extended Stats

Oracle Real-World Perform: x

scam10db01.us.oracle.com:8080

Real-World Demos • Settings Window • Layout • Axis • Command • Login

```

7 from (select owner_id,
8         'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select owner_id,
12         'Ferrari 458s' as text
13      from carguys
14      where country = 'Italy'
15      and make = 'Ferrari'
16      and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)
    
```

	Ferraris	Ferrari 458s	Other Ferraris
	50,000,001	50,000,000	1

Elapsed: 00:00:52.58
end of step

Monitored SQL Executions

ID	Description	Status	Duration
5	Stats with Histograms	✓	51
6	Flash Temp	✓	30
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11	Query 2 Reset Parameter	✓	2
12	Extended Stats	✓	53
13			

SQL Monitor Report

More on OTN: **ORACLE Enterprise Manager**
Active Reports

Monitored SQL Execution Details

Overview

SQL ID: 3bc0vzbk70cj
Parallel: 32
Execution Started: Thu Sep 12, 2013 11:08:48 AM
Last Refresh Time: Thu Sep 12, 2013 11:09:42 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 54.0s
Database Time: 19.7m
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 1,014K
IO Bytes: 221GB
Cell Offload Efficiency: 30%

Details

Plan Statistics | Plan | Parallel | Activity | Metrics

Plan Hash Value: 2967640732

Operation	Name	Estim...	C...	Timeline(34s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1					14	
SORT AGGREGATE					1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10001				1	32	32					
SORT AGGREGATE					1	32	32					
HASH JOIN RIGHT O...		49M	146		32	50M	360B	520B	890K	73	86	
PX RECEIVE		323K	74K		32	1,600				8.4	10	
PX SEND BROADC...	!TQ10000	323K	74K		32	1,600				16	1.7	
PX BLOCK ITER...		323K	74K		32	50M						
VIEW		323K	74K		412	50M				41		
TABLE ACCE...	CARGUYS	323K	74K		412	50M			62K	99	1.08	68
PX BLOCK ITERATOR		49M	74K		32	50M						
TABLE ACCESS S...	CARGUYS	49M	74K		412	50M			62K	99	41	1.36

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Extended Stats

The screenshot shows the Oracle Enterprise Manager SQL Monitor Report interface. The top section displays 'Monitored SQL Execution Details' for a specific SQL ID. Below this, the 'Details' section shows a table of execution operations. A yellow callout box points to the 'PX SEND BROADCAST' operation, which has an estimated cardinality of 323K, significantly higher than the actual cardinality of 32. The table includes columns for Operation, Name, Estimated Rows, Cost, Timeline, Execution, Actual Rows, Memory, Temp, IO Requests, Call Offload, CPU Activity, and Wait Activity.

Operation	Name	Estimated Rows	Cost	Timeline(54s)	Executio...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1					14	
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	TQ10001	1			32	32						
SORT AGGREGATE		1			32	32					41	
HASH JOIN RIGHT OUTER		49M	148K		32	50M	36GB	52GB	890K		73	86
PX RECEIVE		323K	74K		32	1,600M					8.4	10
PX SEND BROADCAST	TQ10000	323K	74K		32	1,600M					16	1.7
PX BLOCK ITERATOR		323K	74K		32	50M						
VIEW		323K	74K		412	50M					41	
TABLE ACCESS STORAGE FULL	CARGUYS	323K	74K		412	50M			62K	99	1.08	68
PX BLOCK ITERATOR		49M	74K		32	50M						
TABLE ACCESS STORAGE FULL	CARGUYS	49M	74K		412	50M			62K	99	41	1.36

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Created column group but still have a poor cardinality estimate

12. Development Findings

- High correlation between Country, Make and Model columns
- Created column group
- Query still exceeds target
- Still have poor cardinality estimate

Histogram on Column Group



Histogram on Column Group

The screenshot displays the Oracle Real-World Performance tool interface. On the left, the SQL Output window shows a query with a histogram on a column group. The query filters for 'Ferrari' cars, grouped into 'Ferraris', 'Ferrari 458s', and 'Other Ferraris'. The output shows 50,000,001 rows for Ferraris, 50,000,000 for Ferrari 458s, and 1 for Other Ferraris. Below the output, a table lists monitored SQL executions with their status and duration.

SQL Output

```

7 from (select owner_id,
8       'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select owner_id,
12       'Ferrari 458s' as text
13      from carguys
14      where country = 'Italy'
15      and make = 'Ferrari'
16      and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)

```

	Ferraris	Ferrari 458s	Other Ferraris
	50,000,001	50,000,000	1

Elapsed: 00:00:02.67
end of step

Monitored SQL Executions

ID	Description	Status	Duration
6	Flash Temp	✓	30
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11	Query 2 Reset Parameter	✓	2
12	Extended Stats	✓	53
13	Extended Stats with Histogram	✓	2
14			

SQL Monitor Report

More on OTN: [ORACLE Enterprise Manager](#)

Active Reports: English

Monitored SQL Execution Details

Overview

SQL ID: 5nhrs640kpa
Parallel: 32
Execution Started: Thu Sep 12, 2013 11:10:47 AM
Last Refresh Time: Thu Sep 12, 2013 11:10:50 AM
Execution ID: 16777216
User: CARSD
Fetch Calls: 1

Time & Wait Statistics

Duration: 3.0s
Database Time: 1.4m
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 124K
IO Bytes: 119GB
Cell Offload Efficiency: 99%

Details

Plan Statistics | Plan | Parallel | Activity | Metrics

Plan Hash Value: 319619247

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estim...	C...	Timeline(3s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1				1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10002	1			32	32						
SORT AGGREGATE		1			32	32				11		
HASH JOIN OUTER		46M	14B		32	50M	30B			60		
PX RECEIVE		46M	74K		32	50M						
PX SEND HASH	!TQ10000	46M	74K		32	50M				4.76		
PX BLOCK ITER...		46M	74K		32	50M						
TABLE ACCE...	CARGUYS	46M	74K		412	50M		62K	99		14	
PX RECEIVE		46M	74K		32	50M				1.59		
PX SEND HASH	!TQ10001	46M	74K		32	50M				17		
PX BLOCK ITER...		46M	74K		32	50M						
VIEW		46M	74K		412	50M				1.59		
TABLE ACCE...	CARGUYS	46M	74K		412	50M		62K	99	3.17	71	

Copyright © 1996-2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Histogram on Column Group

Oracle Real-World Performance

scam10db01.us.oracle.com:8080

SQL Monitor Report

More on OTN
ORACLE® Enterprise Manager
Active Reports

Monitored SQL Execution Details

Overview

SQL ID: 5mbrs404kpa
Parallel: 32
Execution started: Thu Sep 12, 2013 11:10:47 AM
Last Refresh Time: Thu Sep 12, 2013 11:10:50 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 3.0s
Database Time: 16M
PL/SQL & Java: 0.0s
Wait Activity %: 100
Call Offload Efficiency: 99%

Details

Plan Hash Value: 319619247

Plan Statistics | Plan | Parallel | Activity | Metrics

Operation	Name	Estimated Rows	Cost	Time	Executio...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT		1			1	1						
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	ITQ10002	1			32	32						
SORT AGGREGATE		1			32	32				11	60	
HASH JOIN OUTER		46M	1		32	50M	3GB					
PX RECEIVE		46M	74K		32	50M						
PX SEND HASH	ITQ10000	46M	74K		32	50M				4.76		
PX BLOCK ITERATOR		46M	74K		32	50M			62K			
TABLE ACCESS STORAGE FULL	CARGUYS	46M	74K		412	50M				99		14
PX RECEIVE		46M	74K		32	50M					1.59	14
PX SEND HASH	ITQ10001	46M	74K		32	50M					17	
PX BLOCK ITERATOR		46M	74K		32	50M						
VIEW		46M	74K		412	50M				1.59		
TABLE ACCESS STORAGE FULL		46M	74K		412	50M			62K	99	3.17	71

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Histogram on Column Group

13. Development Findings

- Re-gathered stats after running the query with the column groups
- Frequency Histogram on the column group
- Accurate cardinality estimates
- Optimizer now uses a hash distribution

Second Query with Histogram Column Group



Query 2: Histogram Column Group

The screenshot displays the Oracle Real-World Performance tool interface. On the left, the SQL output shows a query comparing two different column group configurations (Combos 1 and 2) and their difference (DIFF). The query filters for Citroens and Minis cars from Ferrari, Citroen, and Mini brands, specifically looking at Mini Coopers in England. The results show 150,010,001 rows for Combo 1, 100,000,000 for Combo 2, and a difference of 50,010,001 rows. The execution time is 00:00:01.90.

Below the SQL output is a table of Monitored SQL Executions:

ID	Description	Status	Duration
7	Manual memory allocation	✓	33
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11	Query 2 Reset Parameter	✓	2
12	Extended Stats	✓	53
13	Extended Stats with Histogram	✓	2
14	Query 2 with Extended Stats	✓	2
15			

On the right, the SQL Monitor Report provides detailed performance metrics for the query. The SQL ID is 0rfor8vjyc7. The execution was parallel with 32 processes and 8 servers. Key statistics include: Duration (2.0s), Database Time (31.9s), IO Requests (124K), IO Bytes (119GB), and Cell Offload Efficiency (99%). The execution plan shows a parallel query with a hash join and a table access operation.

Query 2: Histogram Column Group

Query 2 also has a good cardinality estimate

And uses a broadcast distribution

Overview

SQL ID: 0rfr08vdyv7
Parallel: 32
Execution Started: Fri Sep 20, 2013 10:23:45 AM
Last Refresh Time: Fri Sep 20, 2013 10:23:47 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 2.0s
Database Time: 31.9s
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 124K
IO Bytes: 119GB
Call Offload efficiency: 99%

Details

Plan Hash Value: 2967640732

Operation	Name	Cost	Timeline(Zs)	Executio...	Actual Rows	Memory (M...	Temp (Max)	IO Requests	Call Offlo...	CPU Activity %	Wait Activity %
SELECT STATEMENT		1		1	1						
SORT AGGREGATE		1		1	1						
PX COORDINATOR		65		65	32						
PX SEND QC (RANDOM)	!TQ10001	1		32	32						
SORT AGGREGATE		1		32	32					33	
HASH JOIN RIGHT OUTER		50M	148K	32	150M	40MB				50	
PX RECEIVE		10K	74K	32	320K						
PX SEND BROADCAST	!TQ10000	10K	74K	32	320K						
PX BLOCK ITERATOR		10K	74K	32	10K						
VIEW		10K	74K	412	10K						
TABLE ACCESS STORAGE FULL	CARGUYS	10K	74K	412	10K			62K	100		
PX BLOCK ITERATOR		50M	74K	32	50M						
TABLE ACCESS STORAGE FULL	CARGU...	50M	74K	412	50M			62K	99	17	100

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Query 2: Histogram Column Group

14. Development Findings

- Accurate cardinality estimates
- Optimizer uses a broadcast distribution on second query

Histogram on Column Groups

Now we have the correct solution!

- Both queries have good cardinality estimates
- Correct plans
- Meet targets

Auto Column Group Creation: Seed Column Usage



Auto Column Group Creation

The screenshot displays the Oracle Real-World Performance tool interface. On the left, the 'SQL Output' pane shows a query with a join condition between two tables, 'Ferraris' and 'Ferrari 458s', based on 'owner_id'. Below the query, a summary table shows the results for 'Ferraris', 'Ferrari 458s', and 'Other Ferraris'. The 'Monitored SQL Executions' table at the bottom left lists various execution steps with their status and duration.

The 'SQL Monitor Report' on the right provides detailed performance metrics for the monitored SQL execution. It includes an overview of execution statistics, time and wait statistics, and IO statistics. The 'Details' section shows the execution plan with a table of operations, including SELECT STATEMENT, SORT AGGREGATE, PX COORDINATOR, and various join and access operations.

SQL Output

```

7 from (select owner_id,
8         'Ferraris' as text
9       from carguys
10      where make = 'Ferrari') p1,
11 (select owner_id,
12        'Ferrari 458s' as text
13      from carguys
14      where country = 'Italy'
15      and make = 'Ferrari'
16      and model = '458 Italia') p2
17* where p1.owner_id = p2.owner_id(+)

```

	Ferraris	Ferrari 458s	Other Ferraris
	50,000,001	50,000,000	1

Elapsed: 00:00:50.61
end of step

=====
|

Monitored SQL Executions

ID	Description	Status	Duration
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11	Query 2 Reset Parameter	✓	2
12	Extended Stats	✓	53
13	Extended Stats with Histogram	✓	2
14	Query 2 with Extended Stats	✓	2
15	Seed Column Usage	✓	51
16			

SQL Monitor Report

More on OTN: [ORACLE Enterprise Manager](#)

Active Reports: English

Monitored SQL Execution Details

Overview

SQL ID: d9qxy5c18rtz
Parallel: 32
Execution Started: Fri Sep 20, 2013 10:34:44 AM
Last Refresh Time: Fri Sep 20, 2013 10:35:35 AM
Execution ID: 16777216
User: CAR52
Fetch Calls: 1

Time & Wait Statistics

Duration: 51.0s
Database Time: 18.5m
PL/SQL & Java: 0.0s
Wait Activity %: 100

IO Statistics

Buffer Gets: 16M
IO Requests: 962K
IO Bytes: 215GB
Cell Offload Efficiency: 32%

Details

Plan Statistics | Plan | Parallel | Activity | Metrics

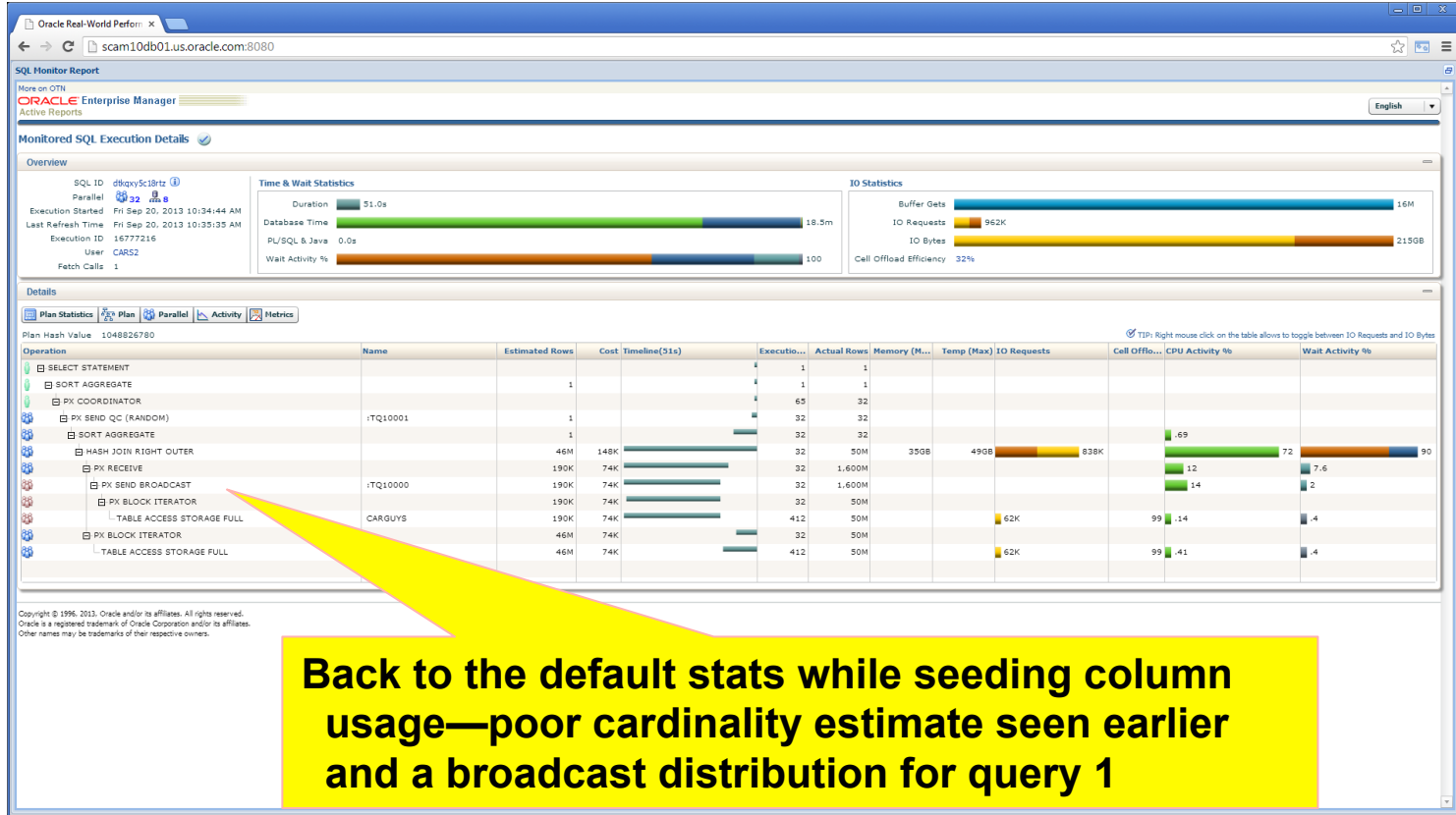
Plan Hash Value: 1048826780

TPP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estim...	C...	Timeline(51s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE					1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10001				1	32	32					
SORT AGGREGATE					1	32	32					
HASH JOIN RIGHT O...		46M	146		32	50M	350B	490B	838K	72	69	90
PX RECEIVE		190K	74K		32	1,600				12	7.6	
PX SEND BROADC...	!TQ10000	190K	74K		32	1,600				14	2	
PX BLOCK ITER...		190K	74K		32	50M						
TABLE ACCES...	CARGUYS	190K	74K		412	50M			62K	99	14	4
PX BLOCK ITERATOR		46M	74K		46M	74K						
TABLE ACCESS S...	CARGUYS	46M	74K		412	50M			62K	99	41	4

Copyright © 1996, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Auto Column Group Creation



Back to the default stats while seeding column usage—poor cardinality estimate seen earlier and a broadcast distribution for query 1

Auto Column Group Creation: Seed Column Usage

15. Development Findings

- Start with default statistics
- Execute `dbms_stats.seed_col_usage` to monitor column usage
- Run query

Auto Column Group Creation: Create Extended Stats



Auto Column Group Creation

The screenshot displays the Oracle Enterprise Manager interface. On the left, the 'SQL Output' window shows a query with three columns: 'Ferraris', 'Ferrari 458s', and 'Other Ferraris'. The query filters for 'Ferrari' cars in 'Italy'. The output shows 50,000,001 rows for Ferraris, 50,000,000 for Ferrari 458s, and 1 for Other Ferraris. Below the output, a table lists 'Monitored SQL Executions' with columns for ID, Description, Status, and Duration.

ID	Description	Status	Duration
8	Cardinality Hint	✓	3
9	Disable Broadcast Distribution	✓	3
10	Query 2 Disable Broadcast Distribution	✓	17
11	Query 2 Reset Parameter	✓	2
12	Extended Stats	✓	53
13	Extended Stats with Histogram	✓	2
14	Query 2 with Extended Stats	✓	2
15	Seed Column Usage	✓	51
16	Auto Column Groups	✓	2

The 'SQL Monitor Report' on the right provides detailed execution statistics for the query. It includes 'Time & Wait Statistics' (Duration: 2.0s, Database Time: 1.4m) and 'IO Statistics' (Buffer Gets: 16M, IO Requests: 124K, IO Bytes: 119GB). Below this, a 'Details' section shows the execution plan with a table of operations, including 'SELECT STATEMENT', 'SORT AGGREGATE', 'PX COORDINATOR', and various join and access methods.

Operation	Name	Estim...	C...	Timeline(3s)	Ex...	Act...	Me...	Te...	IO Reque...	Cell...	CPU Activit...	Wait Activit...
SELECT STATEMENT					1	1						
SORT AGGREGATE		1			1	1						
PX COORDINATOR					65	32						
PX SEND QC (RANDOM)	!TQ10002	1			32	32						
SORT AGGREGATE		1			32	32				4.94		
HASH JOIN OUTER		50M	14B		32	50M	30B			69		
PX RECEIVE		50M	74K		32	50M				2.47		25
PX SEND HASH	!TQ10000	50M	74K		32	50M				8.64		
PX BLOCK ITER...		50M	74K		32	50M						
TABLE ACCE...	CARGUYS	50M	74K		412	50M		62K	99	3.47		
PX RECEIVE		50M	74K		32	50M				1.23		13
PX SEND HASH	!TQ10001	50M	74K		32	50M				6.17		13
PX BLOCK ITER...		50M	74K		32	50M						
VIEW		50M	74K		412	50M				1.23		
TABLE ACCE...	CARGUYS	50M	74K		412	50M		62K	99	3.7		50

Auto Column Group Creation

With the column group identified and created, we have a good cardinality estimate

And we get a hash distribution

The screenshot displays the Oracle Enterprise Manager SQL Monitor Report for a specific SQL execution. The report includes an overview of execution statistics, IO statistics, and a detailed plan table. A yellow callout box highlights the plan table, specifically pointing to the 'PX SEND HASH' operation, which shows a good cardinality estimate and a hash distribution. Another yellow callout box highlights the 'TABLE ACCESS STORAGE FULL' operation, which shows a good hash distribution. The plan table includes columns for Operation, Name, Estimated Rows, Actual Rows, Memory, Temp (Max), IO Requests, Call Offload, CPU Activity %, and Wait Activity %.

Operation	Name	Estimated Rows	Actual Rows	Memory (M...)	Temp (Max)	IO Requests	Call Offload...	CPU Activity %	Wait Activity %
SELECT STATEMENT		1	1						
SORT AGGREGATE		1	1						
PX COORDINATOR		65	32						
PX SEND QC (RANDOM)	ITQ10002	32	32						
SORT AGGREGATE		32	32				4.94	69	
HASH JOIN OUTER		50M	74K	30M			2.47		25
PX RECEIVE		50M	74K	50M			8.64		
PX SEND HASH	ITQ10000	50M	74K	50M					
PX BLOCK ITERATOR		50M	74K	50M					
TABLE ACCESS STORAGE FULL	CARGUYS	50M	74K	50M		62K	99	2.47	
PX RECEIVE		50M	74K	50M			1.23		13
PX SEND HASH	ITQ10001	50M	74K	50M			6.17		13
PX BLOCK ITERATOR		50M	74K	50M					
VIEW		50M	74K	50M			1.23		
TABLE ACCESS STORAGE FULL	CARGUYS	50M	74K	50M		62K	99	3.7	50

Auto Column Group Creation: Create Extended Stats

16. Development Findings

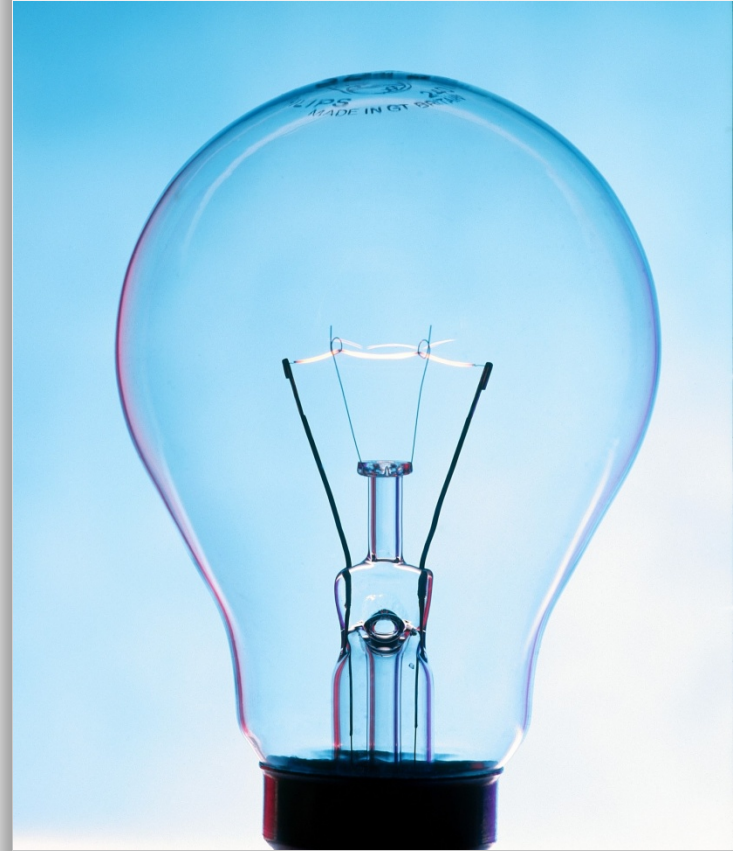
- `dbms_stats.report_col_usage` shows column groups identified during Seed Column Usage
- `dbms_stats.create_extended_stats` creates column groups identified
- Automatically identifies usage of Country, Make and Model columns together and creates column group

Auto Column Group Creation: Create Extended Stats

16. Development Findings

- Regather stats
- Automatically creates Histogram on the column group
- Query meets target

What Did We Learn ?



What Did We Learn

- Moving to a solution before determining the root cause
- Not understanding the big/little data problem
 - Incorrect use of indexes
- Incorrect use of the product
 - Use DoP to “tune” the query
 - Use of flash cache for TEMP
 - Manual Memory Parameters
- System wide changes to fix a single SQL
 - Disable broadcast distribution

Performance Improvement Techniques

Which one are you?

- A Hacker?
- Performance Engineer?

Performance Improvement Techniques

The Hacker

- Tuning by Google
- Tuning by pattern matching or word association
- Tuning by what worked well on another system
- Tuning by Folklore

Performance Improvement Techniques

The Performance Engineer

- Understands Performance is all about work done in a period of time and systematically learns where the time is spent before making any recommendations
- Tuning is seen as a never ending process to systematically locate where the time is spent and making changes that reduces where the time is spent most
- Able to clearly articulate the problem and recommend the appropriate solution

Hardware and Software

ORACLE®

Engineered to Work Together

ORACLE®