



**SAGE Computing Services**

Customised Oracle Training Workshops and Consulting



# **Common Coding and Design Mistakes (that really mess up performance)**

**Penny Cookson**  
SAGE Computing Services

[www.sagecomputing.com.au](http://www.sagecomputing.com.au)

[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)



# **SAGE Computing Services**

Customised Oracle Training Workshops and Consulting

***Penny Cookson***

***Managing Director and Principal Consultant***

***Working with Oracle products since 1987***

***Oracle Magazine Educator of the Year 2004***



*[www.sagecomputing.com.au](http://www.sagecomputing.com.au)*

*[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)*

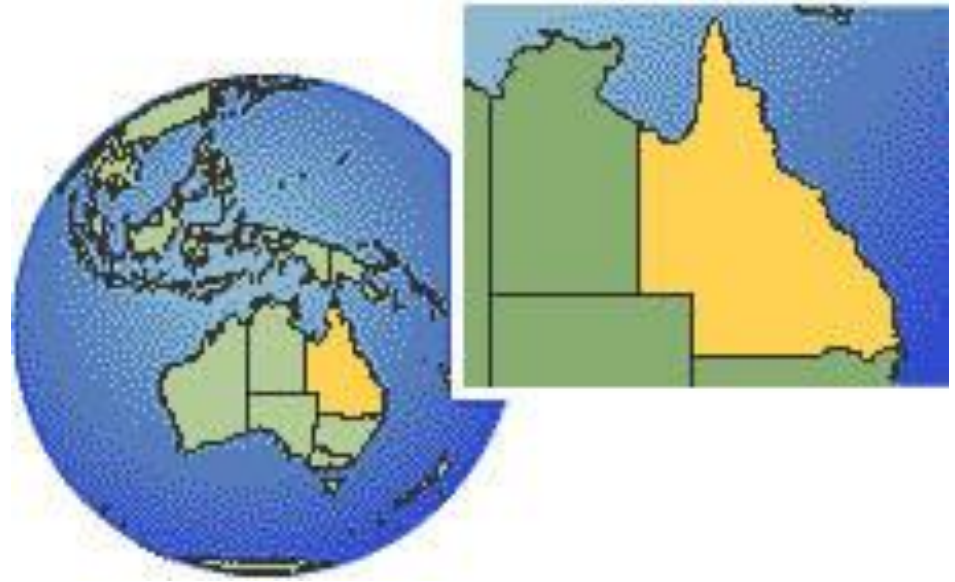
# Disclaimer

None of these mistakes have ever been made by our clients in Western Australia



# Disclaimer

None of these mistakes have  
ever been made by our  
clients in Queensland





# Disclaimer

None of these mistakes have  
ever been made by our  
clients in South Australia



# Why Now?

Asia Pacific Openworld - 1996

A presentation slide with a green gradient background. In the top left corner is the SAGE Computing Services logo. To its right, the text "SAGE Computing Services" is displayed in a bold green font, followed by "Customised Oracle Training Workshops and Consulting" in a smaller green font. The main title "20 Application Tuning tips" is centered in a large, bold black font. Below the title, the name "Penny Cookson" is written in bold green, with "SAGE Computing Services" underneath in a smaller green font. At the bottom right, the website "www.sagecomputing.com.au" and email "penny@sagecomputing.com.au" are listed in a small black font. A decorative graphic of thin, wavy white lines is on the left side of the slide.

**SAGE Computing Services**  
Customised Oracle Training Workshops and Consulting

## 20 Application Tuning tips

**Penny Cookson**  
SAGE Computing Services

[www.sagecomputing.com.au](http://www.sagecomputing.com.au)  
[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)

# Why Now?

Relax – the CBO  
works it out just fine  
all by itself



# Why Now?

The database is just a persistent data store, we don't need to worry about it



# **1 – Modified columns**

**Still the most common issue!**

```
SQL>
SQL>
SQL>
SQL> set pages 100
SQL> set lines 240
SQL>
SQL>
SQL> ALTER SESSION SET statistics_level = ALL;
```

Session altered.

```
SQL>
SQL> ALTER SYSTEM FLUSH SHARED_POOL;
```

System altered.

```
SQL>
SQL>
SQL> SELECT *
  2 FROM bookings_large
  3 WHERE resource_code = 'PC9'
  4 /
```

BOOKING_NO	EVENT_NO	RESO	C	MADE_BY	QUANTITY	COST	S
------------	----------	------	---	---------	----------	------	---

294687	95087	PC9	N	USER6	1	4000	C
--------	-------	-----	---	-------	---	------	---

Room can be set up on the day before the training starts

SQL>



Do some set up

```
SQL>
SQL>
SQL>
SQL> set pages 100
SQL> set lines 240
SQL>
SQL>
SQL> ALTER SESSION SET statistics_level = ALL;
```

Session altered.

```
SQL>
SQL> ALTER SYSTEM FLUSH SHARED_POOL;
```

System altered.

```
SQL>
SQL>
SQL> SELECT *
  2  FROM bookings_large
  3  WHERE resource_code = 'PC9'
  4  /
```

BOOKING_NO	EVENT_NO	RESO	C	MADE_BY	QUANTITY	COST	S
294687	95087	PC9	N	USER6	1	4000	C
Room can be set up on the day before the training starts							

```
SQL>
```



Select a row



```
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

```
PLAN_TABLE_OUTPUT
```

```
SQL_ID fgs3v45pw0bar, child number 0
```

```
SELECT * FROM bookings_large WHERE resource_code = 'PC9'
```

```
Plan hash value: 1244540790
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.01	5
1	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	516	1	00:00:00.01	5
* 2	INDEX RANGE SCAN	BK_RES2	1	516	1	00:00:00.01	4

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

```
2 - access("RESOURCE_CODE"='PC9')
```

```
19 rows selected.
```

```
SQL>
```



Find the access path

```
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

PLAN\_TABLE\_OUTPUT

SQL\_ID fgs3v45pw0bar, child number 0

SELECT \* FROM bookings\_large WHERE resource\_code = 'PC9'

Plan hash value: 1244540790

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.01	5
1	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	516	1	00:00:00.01	5
2	INDEX RANGE SCAN	BK_RES2	1	516	1	00:00:00.01	4

Predicate Information (identified by operation id):

2 - access("RESOURCE\_CODE"='PC9')

19 rows selected.

SQL>

Indexed access

```
SQL> SELECT *
  2 FROM bookings_large
  3 WHERE upper(resource_code) = 'PC9'
  4 /
```

BOOKING_NO	EVENT_NO	RESO C	MADE_BY	QUANTITY	COST S
294687	95087	PC9 N	USER6	1	4000 C
Room can be set up on the day before the training starts					



Modified column

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
  2 /
```

PLAN\_TABLE\_OUTPUT

SQL\_ID gxb6wt3zhr74d, child number 0

SELECT \* FROM bookings\_large WHERE upper(resource\_code) = 'PC9'

Plan hash value: 2081577679

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.73	35590	35341
* 1	TABLE ACCESS FULL	BOOKINGS_LARGE	1	57672	1	00:00:00.73	35590	35341

Predicate Information (identified by operation id):

1 - filter(UPPER("RESOURCE\_CODE")='PC9')

```
SQL> SELECT *
  2 FROM bookings_large
  3 WHERE upper(resource_code) = 'PC9'
  4 /
```

BOOKING_NO	EVENT_NO	RESO	C	MADE_BY	QUANTITY	COST	S
294687	95087	PC9	N	USER6	1	4000	C
Room can be set up on the day before the training starts							

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
  2 /
```

#### PLAN\_TABLE\_OUTPUT

```
SQL_ID gxb6wt3zhr74d, child number 0
-----
SELECT * FROM bookings_large WHERE upper(resource_code) = 'PC9'
Plan hash value: 2081577679
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.73	35590	35341
1	TABLE ACCESS FULL	BOOKINGS_LARGE	1	57672	1	00:00:00.73	35590	35341

Predicate Information (identified by operation id):

1 - filter(UPPER("RESOURCE\_CODE")='PC9')

Full scan

# More examples

WHERE      upper(name) = :B1

WHERE      sal \* 1.1 = :B1

WHERE      trunc (trans\_date) = trunc(SYSDATE)

WHERE      to\_char(start\_date,'dd/mm/yyyy') = '01/03/1993'

WHERE      NVL(postal\_address, business\_address) = :B1

WHERE      emp\_id\_char = 1456

# First – do we really need that code?

WHERE ~~upper~~(name) = :B1

```
1  SELECT COUNT(*)
2  FROM    clients
3* WHERE   UPPER(name) != name
SQL> /

COUNT(*)
-----
0
```

WHERE name = :B1

# First – do we really need that code?

WHERE ~~trunc(trans\_date) = trunc(SYSDATE)~~

WHERE trans\_date >= trunc(SYSDATE)  
AND trans\_date < trunc(SYSDATE) + 1



# First – do we really need that code?

WHERE ~~to\_char(start\_date,'dd/mm/yyyy') = '01/03/1993'~~

WHERE start\_date = to\_date('01/03/1993','dd/mm/yyyy')

# First – do we really need that code?

WHERE emp\_id\_char = 14563



WHERE emp\_id\_char = '14563'

# If we really do need that code

## Function based index

Hidden virtual column

- + index
- + remember to gather stats after you create the index
- + you can create a histogram

## Virtual column

Displayed virtual column

- + you can choose to put an index on it
- + remember to gather stats after you create the column
- + you can create a histogram

```
SQL> 1
      1* CREATE INDEX BKLK_UPPER_RES ON bookings_large (UPPER(resource_code))
SQL> /
```

Index created.

```
SQL> SELECT *
      2 FROM bookings_large
      3 WHERE upper(resource_code) = 'PC9'
      4 /
```

BOOKING_NO	EVENT_NO	RESO C	MADE_BY	QUANTITY	COST S
294687	95087	PC9	N USER6	1	4000 C

Room can be set up on the day before the training starts

```
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
      2 /
```

PLAN\_TABLE\_OUTPUT

SQL\_ID gxb6wt3zhr74d, child number 0

SELECT \* FROM bookings\_large WHERE upper(resource\_code) = 'PC9'

Plan hash value: 1178059707

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.78	5	2
1	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	57672	1	00:00:00.78	5	2
* 2	INDEX RANGE SCAN	BKLK_UPPER_RES	1	23069	1	00:00:00.78	4	2

Create the function  
based index

(if you use your own  
function it must be  
deterministic)

```
SQL> 1
1* CREATE INDEX BKLK_UPPER_RES ON bookings_large (UPPER(resource_code))
SQL> /
```

Index created.

```
SQL> SELECT *
2 FROM bookings_large
3 WHERE upper(resource_code) = 'PC9'
4 /
```

BOOKING_NO	EVENT_NO	RESO	C	MADE_BY	QUANTITY	COST	S
294687	95087	PC9	N	USER6	1	4000	C

Room can be set up on the day before the training starts

```
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

PLAN\_TABLE\_OUTPUT

SQL\_ID gxb6wt3zhr74d, child number 0

SELECT \* FROM bookings\_large WHERE upper(resource\_code) = 'PC9'

Plan hash value: 1178059707

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.78	5	2
1	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	57672	1	00:00:00.78	5	2
* 2	INDEX RANGE SCAN	BKLK_UPPER_RES	1	23069	1	00:00:00.78	4	2

select the rows  
using the  
function

```
SQL> 1
1* CREATE INDEX BKLK_UPPER_RES ON bookings_large (UPPER(resource_code))
SQL> /
```

Index created.

```
SQL> SELECT *
2 FROM bookings_large
3 WHERE upper(resource_code) = 'PC9'
4 /
```

BOOKING_NO	EVENT_NO	RESO	C	MADE_BY	QUANTITY	COST	S
294687	95087	PC9	N	USER6	1	4000	C
Room can be set up on the day before the training starts							

```
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

PLAN\_TABLE\_OUTPUT

SQL\_ID gxb6wt3zhr74d, child number 0

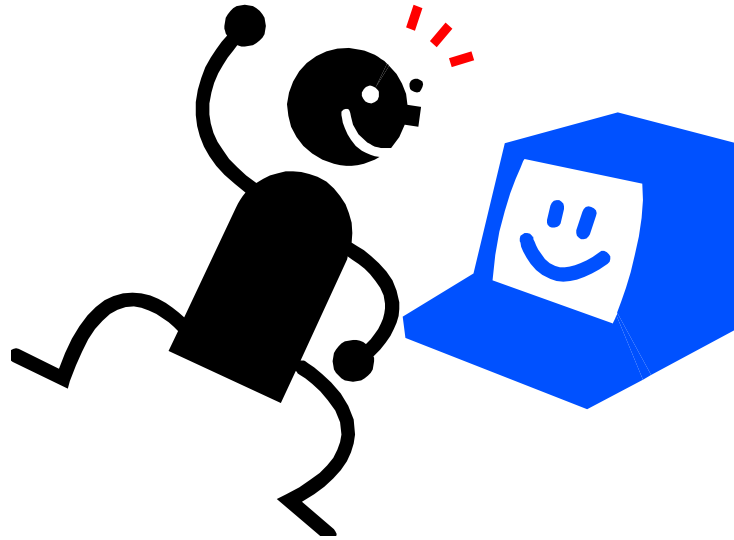
SELECT \* FROM bookings\_large WHERE upper(resource\_code) = 'PC9'

Plan hash value: 1178059707

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.78	5	2
1	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	57672	1	00:00:00.78	5	2
* 2	INDEX RANGE SCAN	BKLK_UPPER_RES	1	23069	1	00:00:00.78	4	2

it uses the FBI

**So everything is fine now?**





# RESOURCE\_CODE has skewed data

```
SELECT resource_code, count(*)
FROM bookings_large
GROUP BY resource_code
```

RESOURCE_CODE	COUNT(*)
BRL8	99999
VCR1	797012
BRL5	7
CONF	496978
LNCH	745524
BRSM	645532
PC1	48508
PC2	19
FLPC	497014
BRL9	1
BRLG	1691035
PC5	1
BRL6	7
PC6	1
PC9	1
BRL1	1
PC7	1
BRL7	1
TAP1	248511
VCR2	497015

```
SQL> SELECT count(status)
2 FROM bookings_large
3 WHERE upper(resource_code) = 'BRLG'
4 /
```

```
COUNT(STATUS)
-----
1691035
```

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

```
PLAN_TABLE_OUTPUT
-----
```

```
SQL_ID 1yg7jkbv2r7m3, child number 0
-----
```

```
SELECT count(status) FROM bookings_large WHERE upper(resource_code) =
'BRLG'
```

```
Plan hash value: 4198581906
-----
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:05.39	37134	32733
1	SORT AGGREGATE		1	1	1	00:00:05.39	37134	32733
2	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	57672	1691K	00:00:05.24	37134	32733
* 3	INDEX RANGE SCAN	BKLG_UPPER_RES	1	23069	1691K	00:00:01.03	3769	3766

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

```
3 - access("BOOKINGS_LARGE"."SYS_NC00010$"='BRLG')
```

```
21 rows selected.
```

now select a majority  
value

```
SQL> SELECT count(status)
2   FROM bookings_large
3   WHERE upper(resource_code) = 'BRLG'
4   /
```

```
COUNT(STATUS)
-----
      1691035
```

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2   /
```

```
PLAN_TABLE_OUTPUT
-----
```

```
SQL_ID  1yg7jkbv2r7m3, child number 0
-----
```

```
SELECT count(status) FROM bookings_large WHERE upper(resource_code) =
'BRLG'
```

```
Plan hash value: 4198581906
-----
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:05.39	37134	32733
1	SORT AGGREGATE		1	1	1	00:00:05.39	37134	32733
2	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	57672	1691K	00:00:05.24	37134	32733
* 3	INDEX RANGE SCAN	BKLG_UPPER_RES	1	23069	1691K	00:00:01.03	3769	3766

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

```
3 - access("BOOKINGS_LARGE"."SYS_NC00010$"='BRLG')
```

```
21 rows selected.
```

its still using the index

```
SQL> SELECT column_name, num_distinct, hidden_column, virtual_column
 2  FROM user_tab_cols
 3  WHERE table_name = 'BOOKINGS_LARGE'
 4  /
```

COLUMN_NAME	NUM_DISTINCT	HID	VIR
SYS_NC00010\$		YES	YES
BOOKING_NO	5767168	NO	NO
EVENT_NO	100792	NO	NO
RESOURCE_CODE	20	NO	NO
CHARGEABLE	2	NO	NO
MADE_BY	5	NO	NO
QUANTITY	4	NO	NO
COST	9	NO	NO
STATUS	3	NO	NO
COMMENTS	3	NO	NO

10 rows selected.

```
SQL>
SQL> BEGIN
 2  dbms_stats.gather_table_stats(ownname=>user, tabname=> 'BOOKINGS_LARGE',method_opt=> 'for all columns size auto');
 3  END;
 4
 5  /
```

PL/SQL procedure successfully completed.

SQL>

creating a function  
based index creates a  
hidden virtual column

```
SQL> SELECT column_name, num_distinct, hidden_column, virtual_column
2 FROM user_tab_cols
3 WHERE table_name = 'BOOKINGS_LARGE'
4 /
```

COLUMN_NAME	NUM_DISTINCT	HID	VIR
SYS_NC00010\$		YES	YES
BOOKING_NO	5767168	NO	NO
EVENT_NO	100792	NO	NO
RESOURCE_CODE	20	NO	NO
CHARGEABLE	2	NO	NO
MADE_BY	5	NO	NO
QUANTITY	4	NO	NO
COST	9	NO	NO
STATUS	3	NO	NO
COMMENTS	3	NO	NO

10 rows selected.

```
SQL>
SQL> BEGIN
2   dbms_stats.gather_table_stats(ownname=>user, tabname=> 'BOOKINGS_LARGE',method_opt=> 'for all columns size auto');
3 END;
4
5 /
```

PL/SQL procedure successfully completed.

SQL>

COLUMN_NAME	NUM_DISTINCT	NUM_DISTINCT	VIR	HISTOGRAM
SYS_NC00010\$	20	20	YES	FREQUENCY
BOOKING_NO	5767168	5767168	NO	NONE
EVENT_NO	100792	100792	NO	NONE
RESOURCE_CODE	20	20	NO	FREQUENCY

gather statistics on the  
hidden column

```
SQL> SELECT count(status)
2 FROM bookings_large
3 WHERE upper(resource_code) = 'BRLG'
4 /
```

```
COUNT(STATUS)
-----
1691035
```

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

```
PLAN_TABLE_OUTPUT
-----
```

```
SQL_ID 1yg7jkbv2r7m3, child number 1
-----
```

```
SELECT count(status) FROM bookings_large WHERE upper(resource_code) =
'BRLG'
```

```
Plan hash value: 927983165
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.95	35587	35341
1	SORT AGGREGATE		1	1	1	00:00:00.95	35587	35341
* 2	TABLE ACCESS FULL	BOOKINGS_LARGE	1	288K	1691K	00:00:00.84	35587	35341

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

```
2 - filter(UPPER("RESOURCE_CODE")='BRLG')
```

```
20 rows selected.
```

try the majority value  
again

```
SQL> SELECT count(status)
2 FROM bookings_large
3 WHERE upper(resource_code) = 'BRLG'
4 /
```

```
COUNT(STATUS)
-----
1691035
```

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'))
2 /
```

```
PLAN_TABLE_OUTPUT
```

```
SQL_ID 1yg7jkbv2r7m3, child number 1
```

```
SELECT count(status) FROM bookings_large WHERE upper(resource_code) =
'BRLG'
```

```
Plan hash value: 927983165
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1		1	00:00:00.95	35587	35341
1	SORT AGGREGATE		1	1	1	00:00:00.95	35587	35341
* 2	TABLE ACCESS FULL	BOOKINGS_LARGE	1	288K	1691K	00:00:00.84	35587	35341

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

```
2 - filter(UPPER("RESOURCE_CODE")='BRLG')
```

```
20 rows selected.
```

reads the  
histogram and  
does a full scan





# Modified columns - Summary

Get rid of it if possible

Add a function based index or virtual column

Gather stats on the hidden/virtual column

## 2 – Optional Parameters

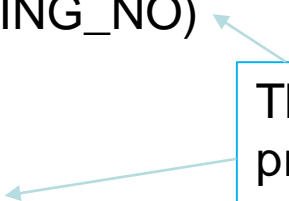
## Handling optional parameters

None of these can use an index efficiently:

WHERE BOOKING\_NO = NVL(:B1 , BOOKING\_NO)

AND BOOKING\_NO = :B1 OR :B1 IS NULL

These may not  
produce the  
same result



AND BOOKING\_NO = COALESCE(:B1 , BOOKING\_NO)

AND RESOURCE\_CODE LIKE :B1||'%' )

Perth WA

Kalgoorlie WA

Route options

Drive via National Highway 94 · 594 km 6 h 30 min

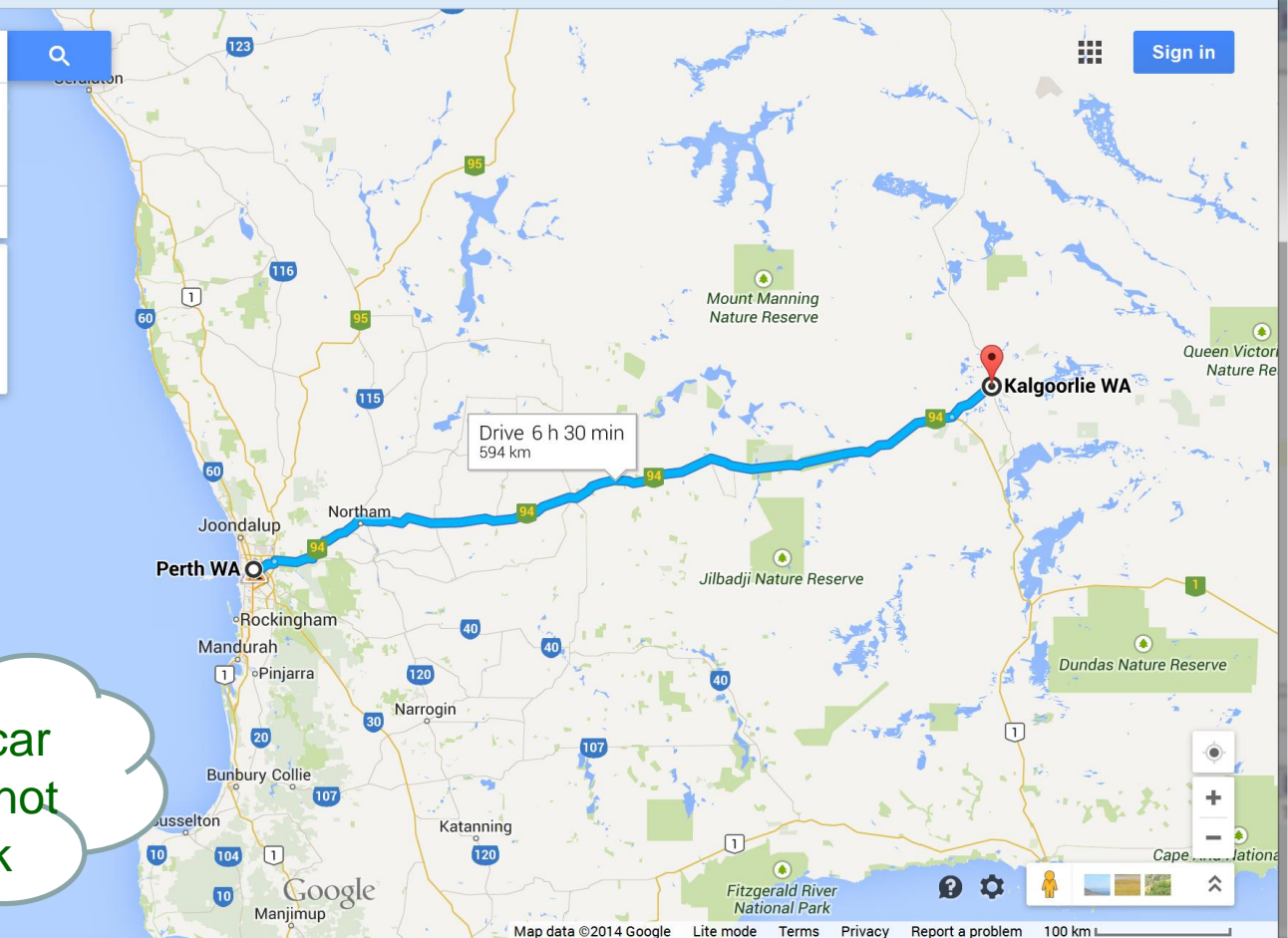
Show traffic

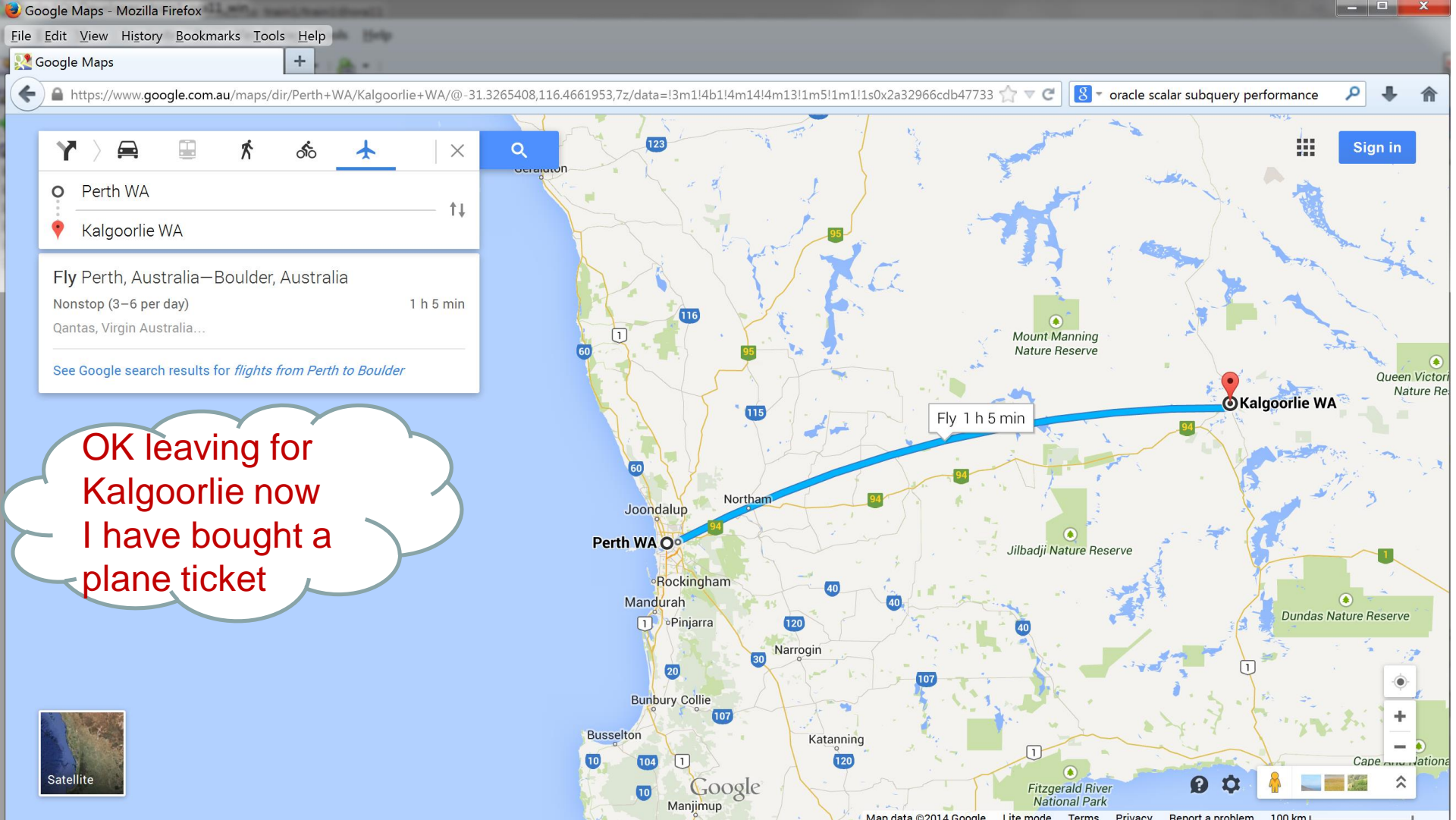
List all steps

Preview steps

What is the best way  
to get to Kalgoorlie  
I have a fast car?

If you have the car  
you can drive if not  
you have to walk





Google Maps - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Google Maps

https://www.google.com.au/maps/dir/Perth+WA/Kalgoorlie+WA/@-31.3265408,116.4661953,7z/data=!3m1!4b1!4m14!4m13!1m5!1m1!1s0x2a32966cdb47733

Sign in

Perth WA

Kalgoorlie WA

Fly Perth, Australia—Boulder, Australia

Nonstop (3–6 per day) 1 h 5 min

Qantas, Virgin Australia...

See Google search results for *flights from Perth to Boulder*

OK leaving for Kalgoorlie now I have bought a plane ticket

Perth WA

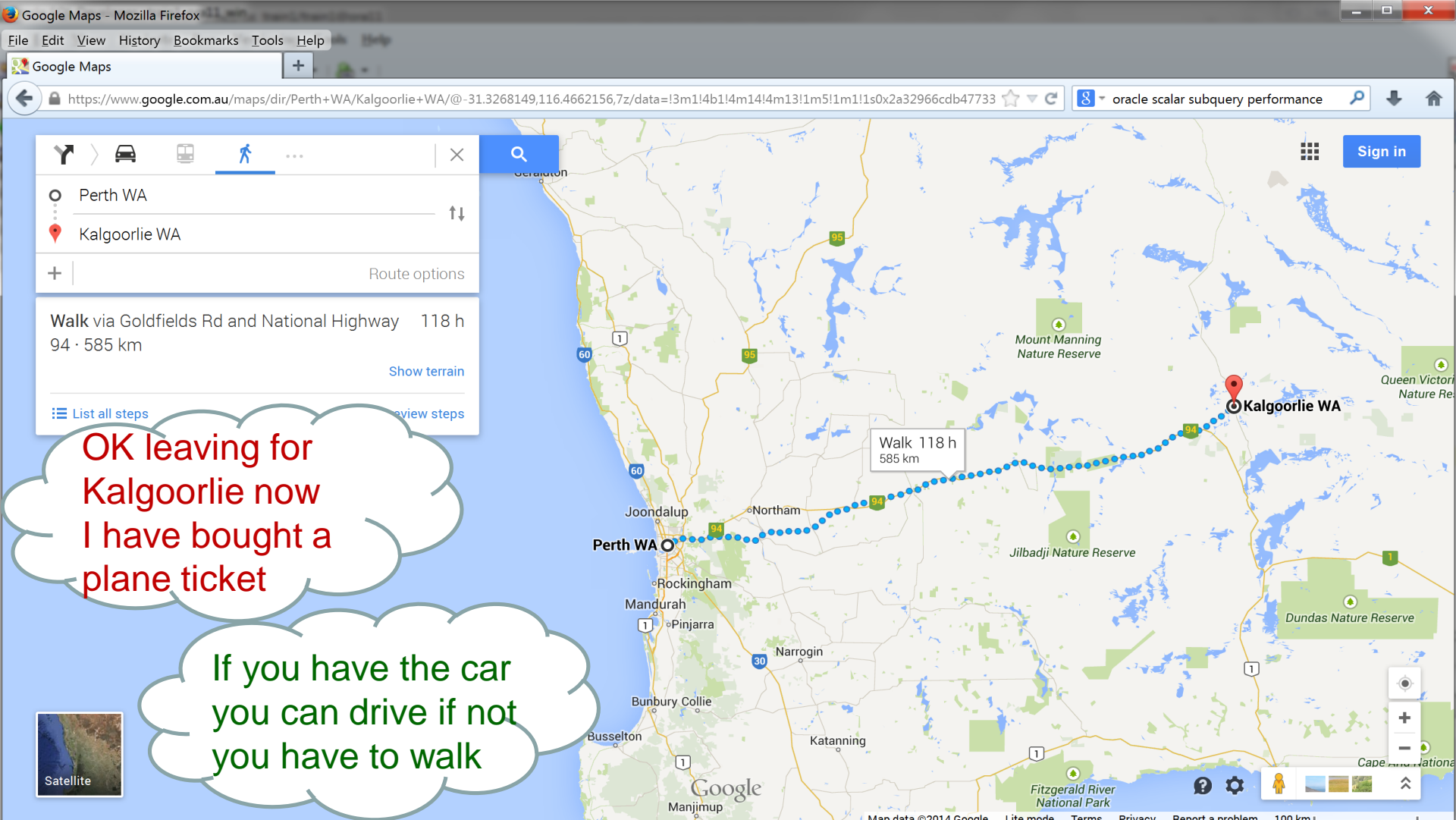
Kalgoorlie WA

Fly 1 h 5 min

Google

Map data ©2014 Google

Lite mode Terms Privacy Report a problem 100 km



OK leaving for  
Kalgoorlie now  
I have bought a  
plane ticket

If you have the car  
you can drive if not  
you have to walk

## Mandatory parameter uses an index

Worksheet Query Builder

```

48
49
50 SELECT count(status)
51 FROM bookings_large
52 WHERE booking_no = :booking_no
53
54
55
56
57

```

Script Output x Query Result x Explain Plan x Autotrace x

SQL | 1.685 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT				2	
SORT		AGGREGATE			4
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	2	1	4
INDEX	BOOKLG_PK	UNIQUE SCAN	1	1	3
Access Predicates					
BOOKING_NO=TO_NUMBER(:BOOKING_NO)					

Optional parameter does a full scan – whether we supply a value or not

Worksheet Query Builder

```

49
50 SELECT count(status)
51 FROM bookings_large
52 WHERE booking_no = :booking_no
53    OR :booking_no IS NULL
54
55
56
57
58
59

```

Script Output x Query Result x Explain Plan x Autotrace x

SQL | 2.277 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT					
SORT			9791		
TABLE ACCESS					
Filter Predicates					
OR					
:BOOKING_NO IS NULL					
BOOKING_NO=TO_NUMBER(:BOOKING_NO)					
	BOOKINGS_LARGE	AGGREGATE FULL	9791	288359	35586



Using NVL results in a UNION of the two different cases  
(plus booking\_no must be NOT NULL)

Worksheet Query Builder

```

49
50 SELECT count(status)
51 FROM bookings_large
52 WHERE booking_no = NVL(:booking_no,booking_no)
53
54
55
56
57
58
59

```

Enter Bind

event\_no  
booking\_no

Name: booking\_no

☒ NULL

Value:

Help Apply Cancel

Script Output x Query Result x Explain Plan x Autotrace x

SQL | 8.767 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT				9825	
SORT		AGGREGATE			1
CONCATENATION					35586
FILTER					35586
Filter Predicates					35586
:BOOKING_NO IS NULL					
TABLE ACCESS	BOOKINGS_LARGE	FULL	9823	1	35586
Filter Predicates					
BOOKING_NO=TO_NUMBER(TO_CHAR(BOOKING_NO))					
FILTER					0
Filter Predicates					
:BOOKING_NO IS NOT NULL					
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	2	1	0
INDEX	BOOKLG_PK	UNIQUE SCAN	1	1	0
Access Predicates					
BOOKING_NO=TO_NUMBER(:BOOKING_NO)					

Using NVL results in a UNION of the two different cases  
(plus booking\_no must be NOT NULL)

Worksheet Query Builder

```

48
49
50 SELECT count(status)
51 FROM bookings_large
52 WHERE booking_no = NVL(:booking_no,booking_no)
53
54
55
56
57
58

```

Enter Bind

event\_no  
booking\_no

Name: booking\_no

☐ NULL

Value: 100

Help Apply Cancel

Script Output x Query Result x Explain Plan x Autotrace x

SQL | 3.182 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			9825		
SORT		AGGREGATE		1	4
CONCATENATION					4
FILTER					0
Filter Predicates					
:BOOKING_NO IS NULL					
TABLE ACCESS	BOOKINGS_LARGE	FULL	9823	1	0
Filter Predicates					
BOOKING_NO=TO_NUMBER(TO_CHAR(BOOKING_NO))					
FILTER					4
Filter Predicates					
:BOOKING_NO IS NOT NULL					
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	2	1	4
INDEX	BOOKLG_PK	UNIQUE SCAN	1	1	3
Access Predicates					
BOOKING_NO=TO_NUMBER(:BOOKING_NO)					

**But most queries of this type have more than one optional condition**

# No indexed access to anything except booking\_no

Worksheet

Query Builder

60

61

62

63

64

65

66

67

SELECT

event\_no, booking\_no, resource\_code, cost

FROM

bookings\_large

WHERE

booking\_no = NVL(:booking\_no,booking\_no)

AND

event\_no = NVL(:event\_no,event\_no)

AND

resource\_code = NVL(:resource\_code,resource\_code)

AND

status = NVL(:status,status)

AND

made\_by = NVL(:made\_by,made\_by)

Script Output

Query Result

Explain Plan

Autotrace

SQL | 3.338 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			9882		
CONCATENATION					35586
FILTER					35586
Filter Predicates					
:BOOKING_NO IS NULL					
TABLE ACCESS	BOOKINGS_LARGE	FULL	9880	1	35586
Filter Predicates					
AND					
BOOKING_NO=TO_NUMBER(TO_CHAR(BOOKING_NO))					
RESOURCE_CODE=NVL(:RESOURCE_CODE,RESOURCE_CODE)					
MADE_BY=NVL(:MADE_BY,MADE_BY)					
STATUS=NVL(:STATUS,STATUS)					
EVENT_NO=TO_NUMBER(NVL(:EVENT_NO,TO_CHAR(EVENT_NO)))					
FILTER					0
Filter Predicates					
:BOOKING_NO IS NOT NULL					
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	2	1	0
Filter Predicates					
AND					
RESOURCE_CODE=NVL(:RESOURCE_CODE,RESOURCE_CODE)					
MADE_BY=NVL(:MADE_BY,MADE_BY)					
STATUS=NVL(:STATUS,STATUS)					
EVENT_NO=TO_NUMBER(NVL(:EVENT_NO,TO_CHAR(EVENT_NO)))					

```

7
8 Plan hash value: 2537911279
9
10 -----
11 | Id | Operation | Name | Starts | E-Rows | A-Rows | A-Time | Buffers |
12 -----
13 | 0 | SELECT STATEMENT | | | 1 | | 1 | 00:00:00.43 | 35587 |
14 | 1 | SORT AGGREGATE | | | 1 | | 1 | 00:00:00.43 | 35587 |
15 | 2 | CONCATENATION | | | 1 | | 0 | 00:00:00.43 | 35587 |
16 |* 3 | FILTER | | | 1 | | 0 | 00:00:00.43 | 35587 |
17 |* 4 | TABLE ACCESS FULL | BOOKINGS_LARGE | 1 | 1 | 0 | 00:00:00.43 | 35587 |
18 |* 5 | FILTER | | | 1 | | 0 | 00:00:00.01 | 0 |
19 |* 6 | TABLE ACCESS BY INDEX ROWID | BOOKINGS_LARGE | 0 | 1 | 0 | 00:00:00.01 | 0 |
20 |* 7 | INDEX UNIQUE SCAN | BOOKLG_PK | 0 | 1 | 0 | 00:00:00.01 | 0 |
21 -----

```

# COALESCE behaves differently from NVL

Worksheet Query Builder

```

131
132 alter system flush shared_pool;
133 DECLARE
134   v_booking_no   bookings_large.booking_no%type := 100;
135
136   v_result       number;
137 BEGIN
138   SELECT COUNT(status)
139   INTO   v_result
140   FROM   bookings_large b
141   WHERE  booking_no = NVL(v_booking_no,booking_no) ;
142 END;
143
144 select * from table(dbms_xplan.display_cursor(SQL_ID=>'fp5u4yd2u3lxv',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
145

```

Script Output x Explain Plan x Autotrace x Query Result x

All Rows Fetched: 28 in 0.078 seconds

PLAN_TABLE_OUTPUT								
	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
7	0	SELECT STATEMENT		1		1	00:00:00.01	4
12	1	SORT AGGREGATE		1	1	1	00:00:00.01	4
13	2	CONCATENATION		1		1	00:00:00.01	4
14	3	FILTER		1		0	00:00:00.01	0
15	4	TABLE ACCESS FULL	BOOKINGS_LARGE	0	5767K	0	00:00:00.01	0
16	5	FILTER		1		1	00:00:00.01	4
17	6	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	1	1	00:00:00.01	4
18	7	INDEX UNIQUE SCAN	BOOKLG_PK	1	1	1	00:00:00.01	3

Worksheet Query Builder

```

153 alter system flush shared_pool;
154 DECLARE
155   v_booking_no   bookings_large.booking_no%type := 100;
156
157   v_result       number;
158 BEGIN
159   SELECT COUNT(status)
160   INTO   v_result
161   FROM   bookings_large b
162   WHERE  booking_no = COALESCE(v_booking_no,booking_no) ;
163 END;
164
165 select * from table(dbms_xplan.display_cursor(SQL_ID=>'5xyvsm6jxldd6',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
166
167

```

Script Output x Explain Plan x Autotrace x Query Result x

All Rows Fetched: 20 in 0.062 seconds

PLAN_TABLE_OUTPUT								
	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
1	0	SELECT STATEMENT		1		1	00:00:00.37	35587
12	1	SORT AGGREGATE		1	1	1	00:00:00.37	35587
13	2	TABLE ACCESS FULL	BOOKINGS_LARGE	1		1	00:00:00.37	35587

And this is probably the worst of all

Worksheet

Query Builder

```

154 alter system flush shared_pool;
155
156 DECLARE
157   v_made_by bookings_large.made_by%type := 'XUSER';
158   v_result number;
159 BEGIN
160   SELECT COUNT(status)
161   INTO   v_result
162   FROM   bookings_large b
163   WHERE  made_by LIKE v_made_by||'%';
164 END;
165
166 select * from table(dbms_xplan.display_cursor(SQL_ID=>'257xamwudpa0h',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
167
168

```

Script Output x

Explain Plan x

Autotrace x

Query Result x

SQL

All Rows Fetched: 21 in 0.078 seconds

PLAN\_TABLE\_OUTPUT

1 SQL\_ID 257xamwudpa0h, child number 0

2 -----

3 SELECT COUNT(STATUS) FROM BOOKINGS\_LARGE B WHERE MADE\_BY LIKE :B1 || '%'

4

5 Plan hash value: 3399248406

6

7 -----

8	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
10	0	SELECT STATEMENT		1		1	00:00:00.07	457	102
11	1	SORT AGGREGATE		1	1	1	00:00:00.07	457	102
12	2	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	49025	48951	00:00:00.06	457	102
13	3	INDEX RANGE SCAN	BKLG_MADE_BY_N6	1	49334	48951	00:00:00.03	131	73

14 -----

15

And this is probably the worst of all – use an index to find all the not null values

Worksheet Query Builder

```

160
161 alter system flush shared_pool;
162 alter session set statistics_level = all;
163
164 DECLARE
165     v_made_by    bookings_large.made_by%type ;
166     v_result     number;
167 BEGIN
168     SELECT COUNT(status)
169     INTO   v_result
170     FROM   bookings_large b
171     WHERE  made_by    LIKE v_made_by||'%';
172 END;
173
174
175 select * from table(dbms_xplan.display_cursor(SQL_ID=>'257xamwudpa0h',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
176

```

Script Output x Explain Plan x Autotrace x Query Result x

SQL | All Rows Fetched: 21 in 0.031 seconds

PLAN\_TABLE\_OUTPUT

```

1 SQL_ID 257xamwudpa0h, child number 0
2 -----
3 SELECT COUNT(STATUS) FROM BOOKINGS_LARGE B WHERE MADE_BY LIKE :B1 || '%'
4
5 Plan hash value: 3399248406
6
7 -----
8 | Id | Operation                                | Name                | Starts | E-Rows | A-Rows | A-Time | Buffers |
9 |----|-----|-----|-----|-----|-----|-----|-----|
10 | 0  | SELECT STATEMENT                        |                     |      1 |        |      1 | 00:00:04.80 | 188K |
11 | 1  | SORT AGGREGATE                          |                     |      1 |      1 |      1 | 00:00:04.80 | 188K |
12 | 2  | TABLE ACCESS BY INDEX ROWID            | BOOKINGS_LARGE      |      1 | 49025 | 5767K | 00:00:04.05 | 188K |
13 |* 3  | INDEX RANGE SCAN                        | BKLG_MADE_BY_N6     |      1 | 49334 | 5767K | 00:00:01.50 | 13681 |
14 -----

```



# Handling optional parameters - Build the WHERE clause dynamically

In Apex –  
use Function  
returning  
SQL Query

Identification		^
Page:	4 Employees	
* Title	<input type="text" value="Employees"/>	<input type="checkbox"/> exclude title from translation
Type	<input type="text" value="SQL Query (PL/SQL function body returning SQL query)"/>	
Source		^
Region Source		
<pre>DECLARE   v_sql VARCHAR2(2000); BEGIN   v_sql := q'[select   id ,first_names ,name ,dob ,job_title ,par_type ,par_id ,eff_from ,eff_to FROM PARTIES WHERE par_type = 'I']';    IF :P4_NAME IS NOT NULL THEN     v_sql := v_sql    q'[ and upper(name) like upper(:P4_NAME)  '%']';   END IF;    RETURN ( v_sql); END;</pre>		

## Handling optional parameters - Build the WHERE clause dynamically

```
PROCEDURE proc1 (p_1 VARCHAR2, p_2 VARCHAR2)
```

```
IS
```

```
.....
```

```
c_1 sys_refcursor;
```

```
v_sql varchar2(2000);
```

```
BEGIN
```

```
  v_sql := 'select something from atable where 1=1';
```

```
  IF p_1 is not null
```

```
    then v_sql := v_sql||' and contact_name = :B1';
```

```
  ELSE
```

```
    v_sql := v_sql|| 'and :B1 is null';
```

```
  END IF;
```

```
  IF p_2 is not null
```

```
    then v_sql := v_sql||' and contact_mobile = :B2';
```

```
  ELSE
```

```
    v_sql := v_sql|| 'and :B2 is null';
```

```
  END IF;
```

```
OPEN c_1 FOR v_sql USING p_1,p_2;
```

```
FETCH c_1 INTO v_2;
```

```
CLOSE c_1;
```

In PL/SQL use a ref cursor

## 3 – Hints in Views

## Hints in View Definition

```
CREATE VIEW    event_bookings
AS
SELECT        /*+ INDEX(b BKLG_MADE_BY_N6) */
              e.org_id, e.event_no, e.start_date
              ,b.booking_no
FROM          events_large e, bookings_large b
WHERE         e.event_no = b.event_no]
AND          b.made_by = 'USER1';
```

You don't know how other people will use the view

Worksheet Query Builder

```

178
179 CREATE OR REPLACE VIEW event_bookings
180 AS
181 SELECT
182     e.org_id, e.event_no, e.start_date
183     ,b.booking_no
184 FROM events_large e, bookings_large b
185 WHERE e.event_no = b.event_no
186 AND b.made_by = 'USER1';
187
188
189 SELECT *
190 FROM event_bookings
191 WHERE booking_no = 42321
192
193 select * from table(dbms_xplan.display_cursor(SQL_ID=>'clxuz59nlpx12',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
194
195

```

Script Output x Explain Plan x Autotrace x Query Result x

SQL | All Rows Fetched: 24 in 0.093 seconds

PLAN\_TABLE\_OUTPUT

	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads
7		-----							
8	0	SELECT STATEMENT		1		1	00:00:00.01	7	1
9	1	NESTED LOOPS		1	1	1	00:00:00.01	7	1
10	2	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	1	1	00:00:00.01	4	1
11	3	INDEX UNIQUE SCAN	BOOKLG_PK	1	1	1	00:00:00.01	3	1
12	4	TABLE ACCESS BY INDEX ROWID	EVENTS_LARGE	1	100K	1	00:00:00.01	3	0
13	5	INDEX UNIQUE SCAN	EVTLG_PK	1	1	1	00:00:00.01	2	0
14		-----							
15		-----							
16		-----							

```
Worksheet | Query Builder
179 CREATE VIEW event_bookings
180 AS
181 SELECT /*+ INDEX(b BKLK_MADE_BY_N6) */
182     e.org_id, e.event_no, e.start_date
183     ,b.booking_no
184 FROM events_large e, bookings_large b
185 WHERE e.event_no = b.event_no
186 AND b.made_by = 'USER1';
187
188
189 SELECT *
190 FROM event_bookings
191 WHERE booking_no = 42321
192
193 select * from table(dbms_xplan.display_cursor(SQL_ID=>'clxuz59nlp12',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
```

MADE\_BY index is used – even when we have the primary key

Script Output x | Explain Plan x | Autotrace x | Query Result x

SQL | All Rows Fetched: 25 in 0.078 seconds

PLAN_TABLE_OUTPUT									
Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads	
0	SELECT STATEMENT		1		1	00:00:00.54	37336	2	
1	NESTED LOOPS		1		1	00:00:00.54	37336	2	
2	NESTED LOOPS		1	1	1	00:00:00.54	37335	2	
* 3	TABLE ACCESS BY INDEX ROWID	BOOKINGS_LARGE	1	1	1	00:00:00.54	37333	0	
* 4	INDEX RANGE SCAN	BKLK_MADE_BY_N6	1	1032K	1039K	00:00:00.21	2466	0	
* 5	INDEX UNIQUE SCAN	EVENTS_PK	1	1	1	00:00:00.01	2	2	
6	TABLE ACCESS BY INDEX ROWID	EVENTS_LARGE	1	100K	1	00:00:00.01	1	0	

Worksheet Query Builder

```

181 SELECT
182     e.org_id, e.event_no, e.start_date
183     ,b.booking_no
184 FROM   events_large e, bookings_large b
185 WHERE  e.event_no = b.event_no
186 AND    b.made_by = 'USER1';
187
188
189 SELECT /*+ INDEX(b1.b BKLG_MADE_BY_N6) */
190        count(start_date)
191 FROM   event_bookings b1
192
193
194 select * from table(dbms_xplan.display_cursor(SQL_ID=>'8vfn480hxuas8',CURSOR_CHILD_NO=>0,FORMAT=>'ALLSTATS LAST'));
195
196

```

If you want to use a hint for the view objects use this syntax  
/\*+ INDEX(b1.b BKLG\_MADE\_BY\_N6) \*/

Script Output x Explain Plan x Autotrace x Query Result x

SQL | All Rows Fetched: 20 in 0.016 seconds

PLAN_TABLE_OUTPUT							
7							
8							
9	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time
10							
11	0	SELECT STATEMENT		1		1	00:00:00.29
12	1	SORT AGGREGATE		1	1	1	00:00:00.29
13	* 2	INDEX RANGE SCAN	BKLG_MADE_BY_N6	1	1026K	1039K	00:00:00.17
14							
15							
16	Predicate Information (identified by operation id):						
17	-----						
18							
19	2	-	access("B"."MADE_BY"='USER1')				

## **4 – Unnecessary table access**



```
SELECT *  
FROM bookings_large b, events_large e  
WHERE e.event_no = b.event_no  
AND   e.contact_name = :contact_name  
AND   b.cost > (SELECT AVG(b1.cost)  
                FROM events_large e1, bookings b1  
                WHERE e1.event_no = b1.event_no  
                AND   e1.event_no = e.event_no)
```

```
SELECT *
FROM bookings_large b, events_large e
WHERE e.event_no = b.event_no
AND   e.contact_name = :contact_name
AND   b.cost > (SELECT AVG(b1.cost)
                FROM events_large e1, bookings b1
                WHERE e1.event_no = b1.event_no
                AND   e1.event_no = e.event_no)
```

```
SELECT *
FROM bookings_large b, events_large e
WHERE e.event_no = b.event_no
AND   e.contact_name = :contact_name
AND   b.cost > (SELECT AVG(b1.cost)
                FROM bookings b1
                WHERE b1.event_no = e.event_no)
```

# 5 – Handling Nulls








```
SELECT *
FROM bookings_large
WHERE processed_flag IS NULL
```

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			9803
TABLE ACCESS	BOOKINGS_LARGE	FULL	9803
Filter Predicates			
PROCESSED_FLAG IS NULL			

Nulls are not stored in B\*Tree indexes  
(unless it is part of a concatenated index and the other part has a value)

```
CREATE INDEX bklg_proc_N0 on bookings_large  
(NVL(processed_flag,'N'));
```

	 BYTES	 BLOCKS	 EXTENTS
1	92274688	11264	82

```
BEGIN  
  dbms_stats.gather_table_stats(ownname=>user, tabname=>  
'BOOKINGS_LARGE',method_opt=> 'for columns SYS_NC00013$ size 3');  
END;
```

Worksheet Query Builder

```

32
33 SELECT *
34 FROM bookings_large b
35 WHERE NVL(processed_flag, 'N') = 'N';
36

```

Script Output x Explain Plan x Query Result x Autotrace x

SQL | 0.015 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			7		
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	7	516	4
INDEX	BKLG_PROC_N0	RANGE SCAN	3	520	3
Access Predicates					
B.SYS_NC00013\$='N'					



If processed\_flag = 'Y' is a majority

```
CREATE INDEX bklg_proc_N1 on bookings_large
(CASE WHEN processed_flag = 'Y' THEN null WHEN processed_flag is null
THEN 'N' ELSE processed_flag END);
```

	BYTES	BLOCKS	EXTENTS
1	65536	8	1

It's a very small index

```
BEGIN
```

```
  dbms_stats.gather_table_stats(ownname=>user, tabname=>
'BOOKINGS_LARGE',method_opt=> 'for all columns size auto');
END;
```

Get the next lot to process:

```
SELECT *
```

```
FROM bookings_large
```

```
WHERE CASE WHEN processed_flag = 'Y' THEN null WHEN processed_flag  
is null THEN 'N' ELSE processed_flag END = 'N'
```

Worksheet Query Builder

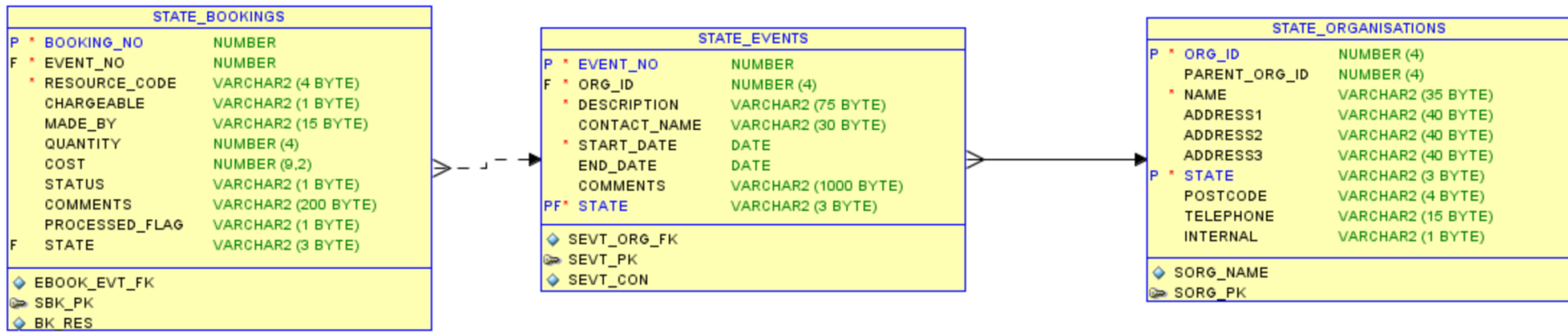
```
18  
19  
20 SELECT *  
21 FROM bookings_large  
22 WHERE CASE WHEN processed_flag = 'Y' THEN null WHEN processed_flag is null THEN 'N' ELSE processed_flag END = 'N';  
23  
24
```

Script Output x Query Result x Autotrace x

SQL | 0.062 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT				10	
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	10	998	3
INDEX	BKLG_PROC_N1	RANGE SCAN	2	999	2
Access Predicates					
BOOKINGS_LARGE.SYS_NC00012\$='N'					

## 6 – Transitivity



```

SELECT o.name,o.postcode, e.end_date, b.status, count(b.booking_no), sum(b.cost)
FROM  state_organisations o, state_events e, state_bookings b
WHERE o.state      = e.state      AND  o.state      = b.state
AND  o.org_id     = e.org_id     AND  e.event_no  = b.event_no
AND  b.resource_code = 'PC9'
GROUP BY o.name, o.postcode, e.end_date, b.status

```

```

248
249 SELECT o.name,o.postcode, e.end_date, b.status, count(b.booking_no), sum(b.cost)
250 FROM state_organisations o, state_events e, state_bookings b
251 WHERE o.state      = e.state      AND   o.state      = b.state
252 AND   o.org_id     = e.org_id     AND   e.event_no   = b.event_no
253 AND   b.resource_code = 'PC9'
254 GROUP BY o.name, o.postcode, e.end_date, b.status
255

```

Script Output x Explain Plan x Query Result x Autotrace x

SQL | 0.125 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			300		
HASH		GROUP BY	300	156	1046
HASH JOIN			299	156	1046
Access Predicates					
AND					
O.STATE=E.STATE					
O.STATE=B.STATE					
O.ORG_ID=E.ORG_ID					
HASH JOIN			280	531	986
Access Predicates					
E.EVENT_NO=B.EVENT_NO					
TABLE ACCESS	STATE_BOOKINGS	BY INDEX ROWID	3	531	4
INDEX	BK_RES	RANGE SCAN	1	531	3
Access Predicates					
B.RESOURCE_CODE='PC9'					
TABLE ACCESS	STATE_EVENTS	FULL	276	100322	982
TABLE ACCESS	STATE_ORGANISATIONS	FULL	18	5070	60

```

255 SELECT o.name, o.postcode, e.end_date, b.status, count(b.booking_no), sum(b.cost)
256 FROM state_organisations o, state_events e, state_bookings b
257 WHERE o.state = e.state AND o.state = b.state AND e.state = b.state
258 AND o.org_id = e.org_id AND e.event_no = b.event_no
259 AND b.resource_code = 'PC9'
260 GROUP BY o.name, o.postcode, e.end_date, b.status
261
262

```

Script Output x Explain Plan x Query Result x Autotrace x

SQL | 0.062 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			76		
HASH		GROUP BY	76	156	67
HASH JOIN			75	156	67
Access Predicates					
AND					
O.STATE=E.STATE					
O.STATE=B.STATE					
O.ORG_ID=E.ORG_ID					
NESTED LOOPS					7
NESTED LOOPS			56	531	6
TABLE ACCESS	STATE_BOOKINGS	BY INDEX ROWID	3	531	4
INDEX	BK_RES	RANGE SCAN	1	531	3
Access Predicates					
B.RESOURCE_CODE='PC9'					
INDEX	SEVT_PK	UNIQUE SCAN	1	1	2
Access Predicates					
AND					
E.STATE=B.STATE					
E.EVENT_NO=B.EVENT_NO					
TABLE ACCESS	STATE_EVENTS	BY INDEX ROWID	1	1	1
TABLE ACCESS	STATE_ORGANISATIONS	FULL	18	5070	60

**7 – NOT IN**

# NOT IN (literals)

```
SELECT status, COUNT(*) NUM  
FROM   bookings_large  
GROUP BY status  
ORDER BY NUM
```

STATUS	NUM
U	999
X	999
I	999
R	786026
P	1310042
C	3668103





# NOT IN (literals)

```
SELECT column_name, num_distinct, histogram
FROM   user_tab_cols
WHERE  table_name = 'BOOKINGS_LARGE'
AND    column_name = 'STATUS'
```

We have a  
histogram

COLUMN_NAME	NUM_DISTINCT	HISTOGRAM
STATUS	6	FREQUENCY

# NOT IN (literals)

```
SELECT index_name, column_name, column_position
FROM   user_ind_columns
WHERE  table_name = 'BOOKINGS_LARGE'
AND    column_name = 'STATUS'
```

A light blue cloud shape with a thin black outline, containing the text "and an index" in red.

and an index

INDEX_NAME	COLUMN_NAME	COLUMN_POSITION
BKLG_STATUS_N4	STATUS	1

# NOT IN (literals)

```
SELECT SUM(cost)
FROM bookings_large
WHERE status NOT IN ('P','R','C');
```

SQL | 1.17 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			9812		
SORT		AGGREGATE		1	35586
TABLE ACCESS	BOOKINGS_LARGE	FULL	9812	1391834	35586
Filter Predicates					
AND					
STATUS<>'C'					
STATUS<>'P'					
STATUS<>'R'					

# NOT IN (literals)

```
SELECT /*+ INDEX(B BKLG_STATUS_N4) */ SUM(cost)
FROM   bookings_large b
WHERE  status NOT IN ('P','R','C');
```

SQL | 0.717 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			36058		
SORT		AGGREGATE		1	10470
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	36058	1391834	10470
INDEX	BKLG_STATUS_N4	FULL SCAN	10570	1391834	10450
Filter Predicates					
AND					
STATUS<>'C'					
STATUS<>'P'					
STATUS<>'R'					

# NOT IN (literals)

```
SELECT SUM(cost)
FROM bookings_large
WHERE status IN ('U','X','I');
```

Change the NOT IN  
to an IN - This might  
not give us the result  
we want in the future

SQL | 0.047 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			143		
SORT		AGGREGATE		1	32
INLIST ITERATOR					32
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	143	6868	32
INDEX	BKLG_STATUS_N4	RANGE SCAN	17	6868	12
Access Predicates					
OR					
STATUS='I'					
STATUS='U'					
STATUS='X'					

```
113
114
115
116
117 CREATE INDEX bklg_status_N9 on bookings_large
118 (CASE WHEN status IN ('P','R','C') THEN null ELSE processed_flag END);
119
120 BEGIN
121     dbms_stats.gather_table_stats(ownname=>user, tabname=> 'BOOKINGS_LARGE',method_opt=> 'for all columns size AUTO');
122 END;
123
124 SELECT SUM(cost)
125 FROM bookings_large
126 WHERE CASE WHEN status IN ('P','R','C') THEN null ELSE processed_flag END IS NOT NULL
127
128
129
```

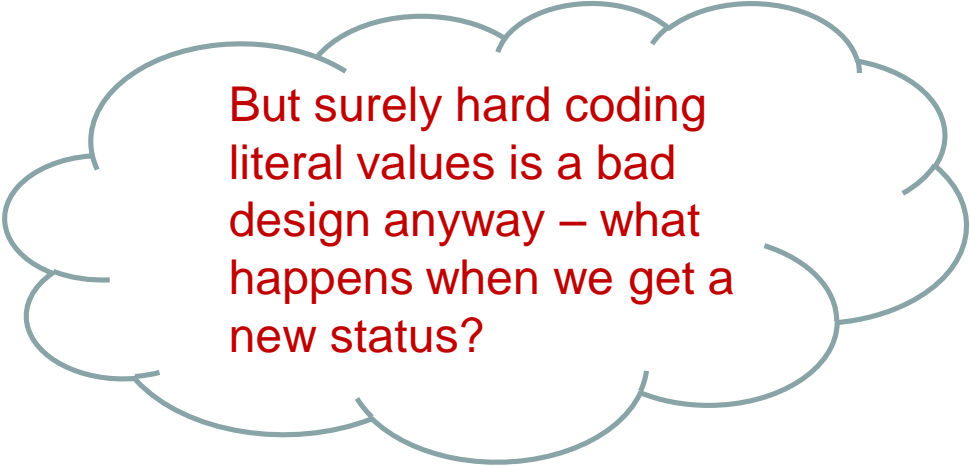
Create a function  
based index

Script Output x Query Result x Autotrace x

SQL | 0.078 seconds

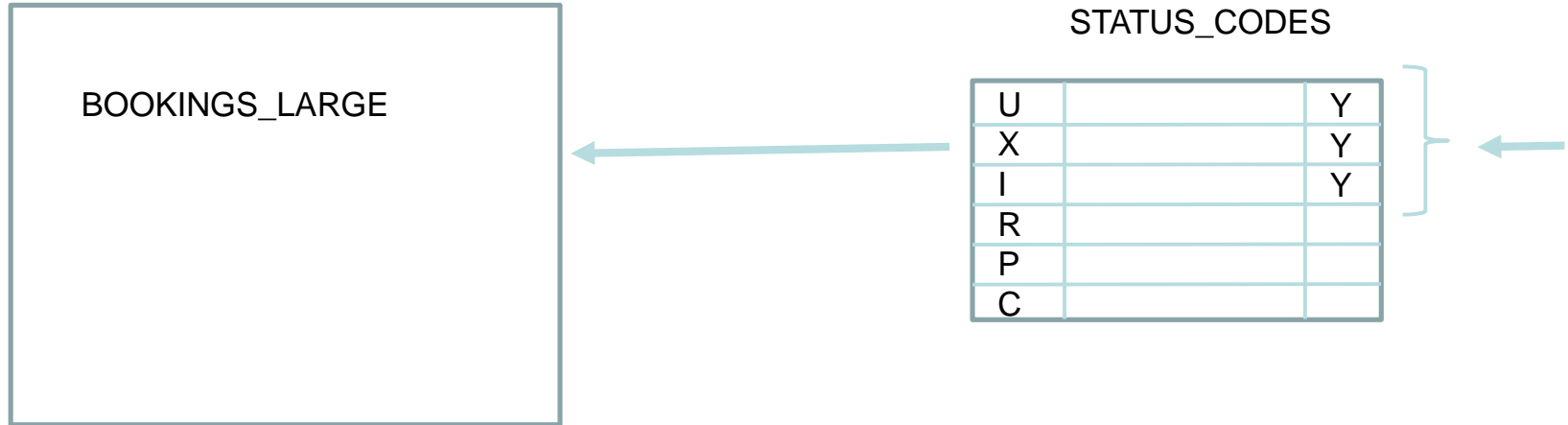
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			18		
SORT		AGGREGATE		1	18
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	18	1998	18
INDEX	BKLG_STATUS_N9	FULL SCAN	5	1998	5
Filter Predicates					
BOOKINGS_LARGE.SYS_NC					

# NOT IN (literals)

A light blue thought bubble with a scalloped border, containing red text.

But surely hard coding  
literal values is a bad  
design anyway – what  
happens when we get a  
new status?

# NOT IN (change to a subquery)





# NOT IN (change to a subquery)

```
SELECT SUM(cost)
FROM bookings_large
WHERE status IN (SELECT status
                  FROM status_codes
                  WHERE current_status = 'Y')
```

The histogram on  
status is not used

SQL | 4.04 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			9799		
SORT		AGGREGATE		1	35598
FILTER					35598
Filter Predicates					
IS NOT NULL					
TABLE ACCESS	BOOKINGS_LARGE	FULL	9787	5767168	35586
FILTER					12
Filter Predicates					
:B1=:B2					
TABLE ACCESS	STATUS_CODES	FULL	2	1	12
Filter Predicates					
CURRENT_STATUS='Y'					

# NOT IN (change to a join)

```
SELECT SUM(cost)
FROM bookings_large b, status_codes c
WHERE b.status = c.status_code
AND c.current_status = 'Y'
```

# NOT IN (change to a join)

```
SELECT /*+ LEADING (c,b) USE_NL_WITH_INDEX(B BKLG_STATUS_N4) */
SUM(cost)
FROM bookings_large b, status_codes c
WHERE b.status = c.status_code
AND c.current_status = 'Y'
```

12c would use  
adaptive optimisation  
and not need the hint

SQL | 0.063 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			60820		
SORT		AGGREGATE		1	33
NESTED LOOPS					33
NESTED LOOPS			60820	2402987	14
TABLE ACCESS	STATUS_CODES	FULL	3	3	2
Filter Predicates					
C.CURRENT_STATUS='Y'					
INDEX	BKLG_STATUS_N4	RANGE SCAN	1726	961195	12
Access Predicates					
B.STATUS=C.STATUS_CODE					
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	20272	961195	19

## 8 – Data design

**You'll  
need a  
constraint**



# Join Elimination

Oracle will get rid of joined tables if:

We give it indexes it can use instead

The PK and/or FK relationships imply a table is not required

# Why constraints are a good idea

```
SELECT e.description
FROM   events e, organisations o
WHERE  e.org_id = o.org_id
```

ora11\_ora\_6340\_events\_5ccqu8sy4vt63.trc - Notepad

File Edit Format View Help

Query block (000000000DFB8F88) before join elimination:  
 SQL:\*\*\*\*\* UNPARSED QUERY IS \*\*\*\*\*  
 SELECT "E"."DESCRIPTION" "DESCRIPTION" FROM "TRAIN1"."EVENTS"  
 "E", "TRAIN1"."ORGANISATIONS" "O" WHERE "E"."ORG\_ID"="O"."ORG\_ID"  
 JE: eliminate table: ORGANISATIONS (O)  
 Registered qb: SEL\$84E079A4\_0xd4fb8f88 (JOIN REMOVED FROM QUERY BLOCK SEL\$1; SEL\$1;  
 "O"@SEL\$1")

-----  
 QUERY BLOCK SIGNATURE  
 -----  
 signature (): qb\_name=SEL\$84E079A4 nbfros=1 flg=0  
 fro(0): flg=0 objn=82476 hint\_alias="E"@SEL\$1"

SQL:\*\*\*\*\* UNPARSED QUERY IS \*\*\*\*\*  
 SELECT "E"."DESCRIPTION" "DESCRIPTION" FROM "TRAIN1"."EVENTS" "E" WHERE "E"."ORG\_ID"  
 IS NOT NULL

Constraints matter – they tell the optimiser stuff

## Without a foreign key constraint

```
ALTER TABLE events  
DISABLE CONSTRAINT org_fk
```

```
SELECT e.description  
FROM   events e, organisations o  
WHERE  e.org_id = o.org_id
```



# Without a foreign key constraint

```

ora11_ora_6340_events2_5ccqu8sy4vt63.trc - Notepad
File Edit Format View Help
JE: Considering Join Elimination on query block SEL$1 (#0)
*****
Join Elimination (JE)
*****
SQL:***** UNPARSED QUERY IS *****
SELECT "E"."DESCRIPTION" "DESCRIPTION" FROM "TRAIN1"."EVENTS"
"E","TRAIN1"."ORGANISATIONS" "O" WHERE "E"."ORG_ID"="O"."ORG_ID"
JE: cfro: EVENTS objn:82474 col#:2 dfro:ORGANISATIONS dcol#:1
JE: cfro: ORGANISATIONS objn:82476 col#:1 dfro:EVENTS dcol#:2
SQL:***** UNPARSED QUERY IS *****
SELECT "E"."DESCRIPTION" "DESCRIPTION" FROM "TRAIN1"."EVENTS"
"E","TRAIN1"."ORGANISATIONS" "O" WHERE "E"."ORG_ID"="O"."ORG_ID"
Query block SEL$1 (#0) unchanged
CVM: Considering view merge in query block SEL$1 (#0)
OJE: Begin: find best directive for query block SEL$1 (#0)
OJE: End: finding best directive for query block SEL$1 (#0)
JE: Considering Join Elimination on query block SEL$1 (#0)
*****
Join Elimination (JE)
*****
SQL:***** UNPARSED QUERY IS *****
SELECT "E"."DESCRIPTION" "DESCRIPTION" FROM "TRAIN1"."EVENTS"
"E","TRAIN1"."ORGANISATIONS" "O" WHERE "E"."ORG_ID"="O"."ORG_ID"
JE: cfro: EVENTS objn:82474 col#:2 dfro:ORGANISATIONS dcol#:1
JE: cfro: ORGANISATIONS objn:82476 col#:1 dfro:EVENTS dcol#:2
SQL:***** UNPARSED QUERY IS *****
SELECT "E"."DESCRIPTION" "DESCRIPTION" FROM "TRAIN1"."EVENTS"
"E","TRAIN1"."ORGANISATIONS" "O" WHERE "E"."ORG_ID"="O"."ORG_ID"
Query block SEL$1 (#0) unchanged
Query block SEL$1 (#0) unchanged



```

## Use RELY for Warehouses

```
ALTER TABLE events DISABLE CONSTRAINT org_fk
```

```
ALTER TABLE events MODIFY CONSTRAINT org_fk RELY;
```

```
SELECT e.description
FROM   events e, organisations o
WHERE  e.org_id = o.org_id
```

OPERATION	OBJECT_NAME	OPTIONS	COST
 SELECT STATEMENT			3
 TABLE ACCESS	EVENTS	FULL	3

# If a column is not null define it as NOT NULL

```
SELECT COUNT(event_no)
FROM   bookings_large
WHERE  booking_no < 1000
```

This has to go to  
the table

SQL | 0.078 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			12		
SORT		AGGREGATE		1	10
TABLE ACCESS	BOOKINGS_LARGE	BY INDEX ROWID	12	999	10
INDEX	BOOKLG_PK	RANGE SCAN	4	999	4
Access Predicates					
BOOKING_NO < 1000					

# If a column is not null define it as NOT NULL

```
ALTER TABLE bookings_large MODIFY
(event_no NOT NULL)
```

```
SELECT COUNT(event_no)
FROM bookings_large
WHERE booking_no < 1000
```

This doesn't  
have to go to the  
table

SQL | 0.078 seconds

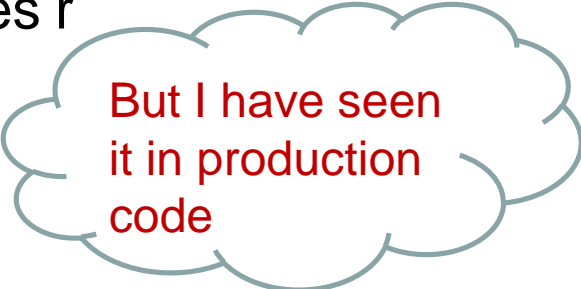
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			4		
SORT		AGGREGATE		1	4
INDEX	BOOKLG_PK	RANGE SCAN	4	999	4
Access Predicates					
BOOKING_NO < 1000					

## 9 – Hideous coding

# Queries in the SELECT

**This is really stupid and will access the  
RESOURCES table multiple times**

```
SELECT b.booking_no, b.cost, b.quantity,  
      (SELECT description FROM resources r  
       WHERE r.code = b.resource_code),  
      (SELECT type_code FROM resources r  
       WHERE r.code = b.resource_code),  
      (SELECT daily_rate FROM resources r  
       WHERE r.code = b.resource_code)  
FROM   bookings_large b)
```

A light blue thought bubble with a scalloped border, containing red text.

But I have seen  
it in production  
code

```

143
144 SELECT b.booking_no, b.cost, b.quantity,
145       (SELECT description FROM resources r
146        WHERE r.code = b.resource_code),
147       (SELECT type_code FROM resources r
148        WHERE r.code = b.resource_code),
149       (SELECT daily_rate FROM resources r
150        WHERE r.code = b.resource_code)
151 FROM   bookings_large b
152

```

Script Output x Query Result x Autotrace x

SQL | 0.094 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY	LAST_CR_BUFFER_GETS
SELECT STATEMENT			9770		
TABLE ACCESS	RESOURCES	BY INDEX ROWID	1	1	2
INDEX	RES_PK	UNIQUE SCAN	0	1	1
Access Predicates					
R.CODE=:B1					
TABLE ACCESS	RESOURCES	BY INDEX ROWID	1	1	2
INDEX	RES_PK	UNIQUE SCAN	0	1	1
Access Predicates					
R.CODE=:B1					
TABLE ACCESS	RESOURCES	BY INDEX ROWID	1	1	2
INDEX	RES_PK	UNIQUE SCAN	0	1	1
Access Predicates					
R.CODE=:B1					
TABLE ACCESS	BOOKINGS_LARGE	FULL	9770	5767168	45

**10–Queries in the SELECT are bad**



# Queries in the SELECT

```
SELECT b.booking_no, b.status, b.cost, b.resource_code, e.description evtdesc  
FROM bookings_large b, events_large e  
WHERE e.event_no = b.event_no  
AND b.booking_no < 1000;
```

A light blue cloud-shaped callout with a thin black outline, containing red text.

NL Join  
3359 reads

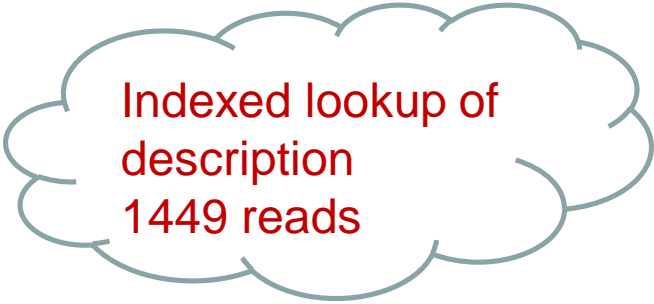
```
SELECT b.booking_no, b.status, b.cost, b.resource_code,  
(SELECT description FROM events_large e WHERE e.event_no = b.event_no) evtdesc  
FROM bookings_large b  
WHERE b.booking_no < 1000;
```

A light blue cloud-shaped callout with a thin black outline, containing red text.

Indexed lookup of  
description  
2894 reads

# Queries in the SELECT

```
SELECT b.booking_no, b.status,  
       b.cost, b.resource_code,  
(SELECT description FROM events_large e WHERE e.event_no = b.event_no) evtDESC  
FROM  
(SELECT * FROM bookings_large WHERE booking_no < 1000  
ORDER BY event_no) b;
```

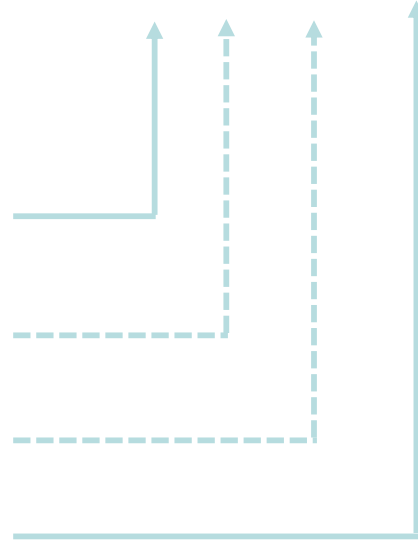
A light blue cloud-shaped callout box with a thin border, containing red text.

Indexed lookup of  
description  
1449 reads

# Queries in the SELECT

(SELECT description FROM events\_large e  
WHERE e.event\_no = b.event\_no)

BOOKING_NO	EVENT_NO
1	100
2	100
3	100
4	200



# **11 –Queries in the SELECT are good**

# Queries in the SELECT

```
SELECT b.booking_no, b.status, b.cost, b.resource_code, b.alternative_resource1,  
b.alternative_resource2,  
  (SELECT avg(cost) FROM bookings_large c  
   WHERE c.resource_code = b.resource_code) avgcost,  
  (SELECT avg(cost) FROM bookings_large d  
   WHERE d.resource_code = b.alternative_resource1) avgalt1,  
  (SELECT avg(cost) FROM bookings_large e  
   WHERE e.resource_code = b.alternative_resource2) avgalt2  
FROM  
(SELECT * FROM bookings_large  
 WHERE booking_no < 10000  
 ORDER BY resource_code) b;
```

A light blue cloud-shaped callout box with a thin black outline, containing red text.

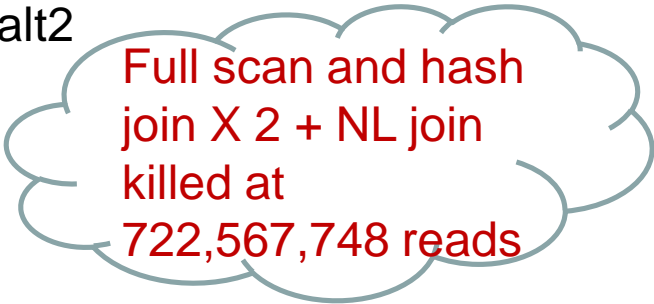
Indexed lookup of  
AVG(COST) X 3  
1,165,043 reads

# Queries in the SELECT – try as a join

```
SELECT b.booking_no, b.status, b.cost, b.resource_code, b.alternative_resource1,  
b.alternative_resource2,  
    avgcost.cost avgcost, avgalt1.cost avgalt1, avgalt2.cost avgalt2  
FROM bookings_large b,  
(SELECT resource_code, avg(cost) cost  
    FROM bookings_large group by resource_code) avgcost,  
(SELECT resource_code, avg(cost) cost  
    FROM bookings_large group by resource_code) avgalt1,  
(SELECT resource_code, avg(cost) cost  
    FROM bookings_large group by resource_code) avgalt2  
WHERE avgcost.resource_code = b.resource_code  
AND  avgalt1.resource_code = b.alternative_resource1  
AND  avgalt2.resource_code = b.alternative_resource2  
AND  b.booking_no < 10000;
```

# Queries in the SELECT – try as a join

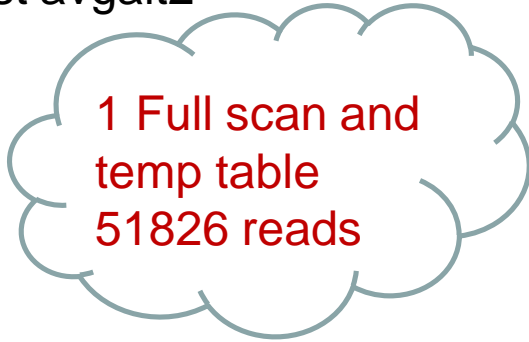
```
SELECT b.booking_no, b.status, b.cost, b.resource_code, b.alternative_resource1,  
b.alternative_resource2,  
    avgcost.cost avgcost, avgalt1.cost avgalt1, avgalt2.cost avgalt2  
FROM bookings_large b,  
    (SELECT resource_code, avg(cost) cost  
     FROM bookings_large group by resource_code) avgcost,  
    (SELECT resource_code, avg(cost) cost  
     FROM bookings_large group by resource_code) avgalt1,  
    (SELECT resource_code, avg(cost) cost  
     FROM bookings_large group by resource_code) avgalt2  
WHERE avgcost.resource_code = b.resource_code  
AND   avgalt1.resource_code = b.alternative_resource1  
AND   avgalt2.resource_code = b.alternative_resource2  
AND   b.booking_no < 10000;
```

A light blue cloud-shaped bubble containing red text.

Full scan and hash  
join X 2 + NL join  
killed at  
722,567,748 reads

# Using a WITH clause

```
WITH averagecost as (SELECT resource_code, avg(cost) cost
                     FROM bookings_large group by resource_code)
SELECT b.booking_no, b.status, b.cost, b.resource_code,
       b.alternative_resource1, b.alternative_resource2,
       avgcost.cost avgcost, avgalt1.cost avgalt1, avgalt2.cost avgalt2
FROM   bookings_large b,
       averagecost avgcost,
       averagecost avgalt1,
       averagecost avgalt2
WHERE  b.resource_code      = avgcost.resource_code
AND    b.alternative_resource1 = avgalt1.resource_code
AND    b.alternative_resource2 = avgalt2.resource_code
AND    b.booking_no < 10000;
```

A light blue cloud-shaped callout box with a thin border and a drop shadow.

1 Full scan and  
temp table  
51826 reads





# Queries in the SELECT – General Rules

If you are accessing the subquery once – inline ordered

If you are accessing the subquery more than once – WITH

NEVER TRUST GENERAL RULES – try all the possible ways

`/*+ MATERIALIZE */` will use a temp table

`/*+ INLINE */` will evaluate in line

# 12 – Functions in SQL

# Using Functions in SQL

```
SELECT *  
FROM bookings_large b  
WHERE b.cost > book_util.get_avg(b.resource_code)  
AND b.booking_no < 100;
```

1

```
SELECT *  
FROM bookings_large b  
WHERE b.cost >  
      (SELECT book_util.get_avg(b.resource_code) FROM dual)  
AND b.booking_no < 100;
```

2

# Using Functions in SQL

```
SELECT *  
FROM bookings_large b  
WHERE b.cost > book_util.get_avg(b.resource_code)  
AND b.booking_no < 100;
```

As above but deterministic

```
SELECT *  
FROM bookings_large b  
WHERE b.cost >  
      (SELECT book_util.get_avg(b.resource_code) FROM dual)  
AND b.booking_no < 100;
```

99 executions of  
function  
4,640,315 reads

1 execution of  
function  
47,396 reads

1 execution of  
function  
48,446 reads



# In Line Queries

Always wrap the function in a `SELECT FROM DUAL`



**Performance Tuning issues  
change all the time**

**keep up to date  
but don't forget the old stuff**

*[www.sagecomputing.com.au](http://www.sagecomputing.com.au)*

*[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)*



# **SAGE Computing Services**

Customised Oracle Training Workshops and Consulting

## **Questions**

**Penny Cookson**

SAGE Computing Services

*[www.sagecomputing.com.au](http://www.sagecomputing.com.au)*

*[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)*