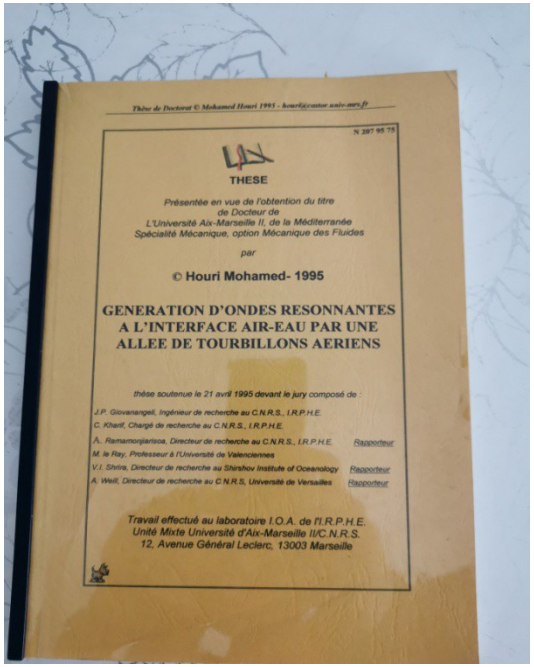
Adaptive Cursor Sharing…

Oracle answer to sharing cursors and optimizing SQL

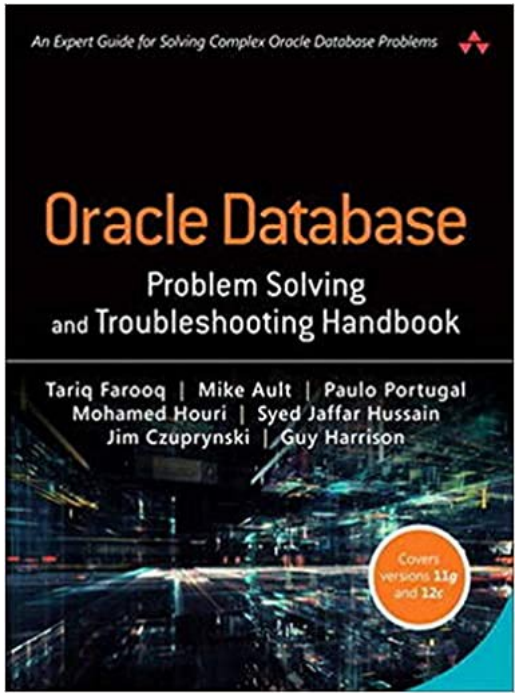
@MohamedHouri

www.hourim.wordpress.com

- Mohamed Houri
- I have a PhD in Fluid Mechanics
- I am an Oracle ACE



- I blog here : www.hourim.wordpress.com
- I tweet using this handle : @MohamedHouri
- I co-authored 3 chapters in this book
  - Chapter 4: **Adaptive Cursor Sharing**
  - Chapter 5: **Stabilizing Query performance using SPM**
  - Chapter 6: **DDL optimization Tips, Techniques ,and Tricks**



@MohamedHouri
www.hourim.wordpress.com

Agenda

- **ACS : Prerequisite**
- ACS : Warm-up period
- ACS : step down in favor of ECS
- ACS : How to properly cancel it
- ACS : my opinion

ACS- Prerequisite

To be elected for ACS a cursor needs first to be **Bind Sensitive**

To be Bind Sensitive a cursor needs to use **Bind Variable**

- either directly
- through cursor_sharing set to **FORCE**

ACS- Prerequisite

```
SQL> desc V$SQL

        Name                                      Null?    Type
        ----------------------------------------- -------- ----------------
  1     SQL_TEXT                                           VARCHAR2(1000)
  2     SQL_FULLTEXT                                       CLOB
  3     SQL_ID                                             VARCHAR2(13)
 45     CHILD_NUMBER                                       NUMBER
 63     IS_OBSOLETE                                        VARCHAR2(1)
 64     IS_BIND_SENSITIVE                                  VARCHAR2(1)
 65     IS_BIND_AWARE                                      VARCHAR2(1)
 66     IS_SHAREABLE                                       VARCHAR2(1)
```

ACS- Prerequisite

range predicate (with simple statistics)

```
SQL> select count(1) from t_acs where n2 <= :ln2;
```

ACS- Prerequisite

range predicate (with simple statistics)

```
SQL> select count(1) from t_acs where n2 <= :ln2;
```

equality predicate (with histogram)

```
SQL> select count(1) from t_acs where n2 = :ln2;
```

ACS- Prerequisite

range predicate (with simple statistics)

```
SQL> select count(1) from t_acs where n2 <= :ln2;
```

equality predicate (with histogram)

```
SQL> select count(1) from t_acs where n2 = :ln2;
```

predicate with partition key (simple stats)

```
SQL> select count(1) from t_acs where n2 = :ln2;
```

ACS- Prerequisite

DEMO

Agenda

- ACS : Prerequisite
- **ACS : Warm-up period**
- ACS : step down in favor of ECS
- ACS : How to properly cancel it
- ACS : my opinion

ACS- Warm-up period

Now that the cursor is **BIND SENSITIVE** all its future executions will be **monitored**

The goal of this monitoring phase is to decide when it is time to mark the cursor **BIND AWARE**

As long as cursor is not marked BIND AWARE **ACS will not kick in**

The time it will take for the cursor to switch from BIND SENSITIVE
to BIND AWARE is known as the : **WARM-UP period**

ACS- Warm-up period

```
SQL> desc V$SQL

        Name                                      Null?    Type
        ----------------------------------------- -------- ----------------
  1     SQL_TEXT                                           VARCHAR2(1000)
  2     SQL_FULLTEXT                                       CLOB
  3     SQL_ID                                             VARCHAR2(13)
 45     CHILD_NUMBER                                       NUMBER
 63     IS_OBSOLETE                                        VARCHAR2(1)
 64     IS_BIND_SENSITIVE                                  VARCHAR2(1)
 65     IS_BIND_AWARE                                      VARCHAR2(1)
 66     IS_SHAREABLE                                       VARCHAR2(1)
```

ACS- Warm-up period

How do you think Oracle handles this warm-up period?

ACS- Warm-up period

```
SQL> desc V$SQL_CS_HISTOGRAM

 Name                          Type
 ----------------------------- ------------
 ADDRESS                       RAW(8)
 HASH_VALUE                    NUMBER
 SQL_ID                        VARCHAR2(13)
 CHILD_NUMBER                  NUMBER
 BUCKET_ID                     NUMBER
 COUNT                         NUMBER
 CON_ID                        NUMBER
```

number of rows processed by this child_number

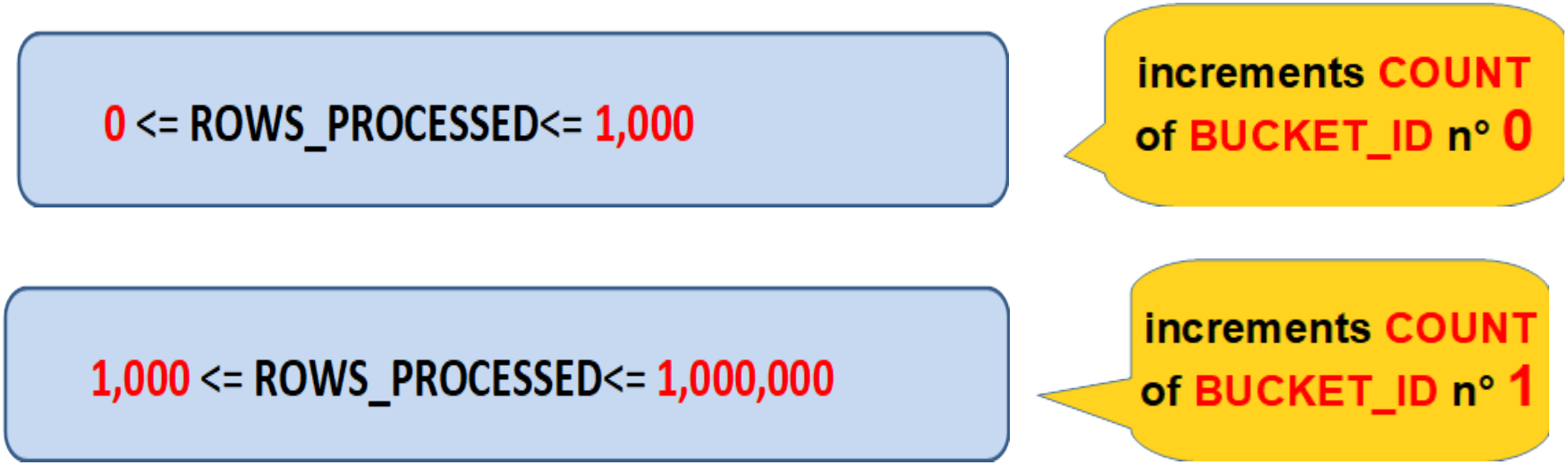
number of executions at this child_number

ACS- Warm-up period

0 <= ROWS_PROCESSED<= 1,000

increments COUNT of BUCKET_ID n° 0

ACS- Warm-up period



ACS- Warm-up period

ACS- Warm-up period

Based upon **(BUCKET_ID, COUNT)** Oracle uses **3 rules** to switch a cursor from bind sensitive to bind aware

ACS- Warm-up period

Based upon **(BUCKET_ID, COUNT)** Oracle uses **3 rules** to switch a cursor from bind sensitive to bind aware

- When executions (**COUNT**) concern only **ADJACENT BUCKET_ID**. i.e. (0,1) and (1,2)

ACS- Warm-up period

Based upon **(BUCKET_ID, COUNT)** Oracle uses **3 rules** to switch a cursor from bind sensitive to bind aware

- When executions (**COUNT**) concern only **ADJACENT BUCKET_ID**. i.e. (0,1) and (1,2)

- When executions (**COUNT**) concern only **DISTANT BUCKET_ID**. i.e. (0,2)

ACS- Warm-up period

Based upon **(BUCKET_ID, COUNT)** Oracle uses **3 rules** to switch a cursor from bind sensitive to bind aware

- When executions (**COUNT**) concern only **ADJACENT BUCKET_ID**. i.e. (0,1) and (1,2)
- When executions (**COUNT**) concern only **DISTANT BUCKET_ID**. i.e. (0,2)
- When executions (COUNT) concern **ALL BUCKET_ID**. i.e. (0,1,2)

ACS- Warm-up period - **ADJACENT**

- When executions (**COUNT**) concern only **ADJACENT BUCKET_ID**. i.e. (0,1) and (1,2)

ACS- Warm-up period - **ADJACENT**

- When executions (**COUNT**) concern only **ADJACENT BUCKET_ID**. i.e. (0,1) and (1,2)

COUNT(BUCKET_ID n° 0) = COUNT(BUCKET_ID n°1)

COUNT(BUCKET_ID n° 1) = COUNT(BUCKET_ID n°2)

The next execution will mark the cursor **BIND AWARE** and compile **a new execution plan**

ACS- Warm-up period - **ADJACENT**

DEMO

ACS- Warm-up period - **DISTANT**

- When executions (**COUNT**) concern only **DISTANT BUCKET_ID**. i.e. (0,2)

ACS- Warm-up period - **DISTANT**

- When executions (**COUNT**) concern only **DISTANT BUCKET_ID**. i.e. (0,2)

COUNT(BUCKET_ID n° **2**) = **CEIL** (COUNT(BUCKET_ID n°**0** /**3**))

The next execution will mark the cursor **BIND AWARE** and compile **a new execution plan**

ACS- Warm-up period - **DISTANT**

DEMO

ACS- Warm-up period - **ALL**

- When executions (**COUNT**) concern **ALL BUCKET_ID**. i.e. (0,1,2)

ACS- Warm-up period - **ALL**

- When executions (**COUNT**) concern **ALL BUCKET_ID**. i.e. (0,1,2)

```
-------------------------------------------------------------------
-- File name:   fv_will_cs_be_bind_aware
-- Author    :   Mohamed Houri (Mohamed.Houri@gmail.com)
-- Date      :   29/08/2015
-- Purpose   :   When supplied with 3 parameters
--                    pin_cnt_bucket_0 : count of BUCKET_ID n°0
--                    pin_cnt_bucket_1 : count of BUCKET_ID n°1
--                    pin_cnt_bucket_2 : count of BUCKET_ID n°2
--                This function will return a status:
--   Y --> cursor will be BIND AWARE
--   N --> cursor will NOT be BIND AWARE
-------------------------------------------------------------------

    SQL> select
            fv_will_cs_be_bind_aware (0,1,2) from dual;
```

ACS- Warm-up period - **ALL**

DEMO

Agenda

- ACS : Prerequisite
- ACS : Warm-up period
- **ACS : step down in favor of ECS**
- ACS : How to properly cancel it
- ACS : my opinion

ACS : step down in favor of ECS

What happens when a cursor becomes **BIND AWARE ?**

ACS : step down in favor of ECS

Adaptive Cursor Sharing

is responsible for marking bind aware a bind sensitive cursor provided one of the 3 rules is satisfied

Adaptive Cursor Sharing

V$SQL_CS_HISTOGRAM

ACS : step down in favor of ECS

Extended Cursor Sharing

is responsible for checking if an execution plan of a bind aware cursor has to be shared or a new plan has to be compiled according to the bind variable selectivity it peeks at each execution

Extended Cursor Sharing

V$SQL_CS_SELECTIVITY

ACS : step down in favor of ECS

```
SQL> desc V$SQL_CS_SELECTIVITY
```

This view becomes useful only when cursor becomes BIND AWARE

	Name	Null?	Type
1	ADDRESS		RAW(8)
2	HASH_VALUE		NUMBER
3	SQL_ID		VARCHAR2(13)
4	CHILD_NUMBER		NUMBER
5	PREDICATE		VARCHAR2(40)
6	RANGE_ID		NUMBER
7	LOW		VARCHAR2(10)
8	HIGH		VARCHAR2(10)
9	CON_ID		NUMBER

ACS : step down in favor of ECS

How does the **Extended Cursor Sharing** layer code work?

ACS : step down in favor of ECS

The **Extended Cursor Sharing** layer code works as follows:

For each execution of a BIND AWARE cursor ECS will

- Get the **selectivity cube** of the used bind variable

ACS : step down in favor of ECS

The **Extended Cursor Sharing** layer code works as follows:

For each execution of a BIND AWARE cursor ECS will

- Get the **selectivity cube** of the used bind variable

- Check if this **selectivity cube** is covered by an exiting child cursor **low-high range selectivity** stored in **V$SQL_CS_SELECTIVITY** view

ACS : step down in favor of ECS

The **Extended Cursor Sharing** layer code works as follows:

For each execution of a BIND AWARE cursor ECS will

- Get the **selectivity cube** of the used bind variable

- Check if this **selectivity cube** is covered by an exiting child cursor **low-high range selectivity** stored in **V$SQL_CS_SELECTIVITY** view
  - IF YES then it will **share** this child cursor
  - IF NO then it will **compile** a new execution plan and insert a new **low-high** selectivity range in the **V$SQL_CS_SELECTIVITY** view

ACS : step down in favor of ECS

There are 4 types of Cursor Selectivity Cube

ACS : step down in favor of ECS

There are 4 types of Cursor Selectivity Cube

1. Cursor Selectivity Cube for **FREQUENCY** histogram

ACS : step down in favor of ECS

There are 4 types of Cursor Selectivity Cube

1. Cursor Selectivity Cube for **FREQUENCY** histogram

2. Cursor Selectivity Cube for **popular HYBRID** histogram

ACS : step down in favor of ECS

There are 4 types of Cursor Selectivity Cube

1. Cursor Selectivity Cube for **FREQUENCY** histogram

2. Cursor Selectivity Cube for **popular HYBRID** histogram

3. Cursor Selectivity Cube for **non-popular HYBRID** histogram having an **endpoint number**

ACS : step down in favor of ECS

There are 4 types of Cursor Selectivity Cube

1. Cursor Selectivity Cube for **FREQUENCY** histogram

2. Cursor Selectivity Cube for **popular HYBRID** histogram

3. Cursor Selectivity Cube for **non-popular HYBRID** histogram having an **endpoint number**

4. Cursor Selectivity Cube for **non-captured non-popular HYBRID** histogram

ACS : step down in favor of ECS

1. Cursor Selectivity Cube for **FREQUENCY** histogram

DEMO

Agenda

- ACS : Prerequisite
- ACS : Warm-up period
- ACS : step down in favor of ECS
- **ACS : How to properly cancel it**
- ACS : my opinion

ACS : How to properly cancel it

We can cancel ACS using

- /*+ **no_bind_aware** */

ACS : How to properly cancel it

We can cancel ACS using

- /*+ **no_bind_aware** */

- **"_optimizer_extended_cursor_sharing"** = 'none'
  "_optimizer_extended_cursor_sharing_rel" = 'none'

ACS : How to properly cancel it

We can cancel ACS using

- /*+ **no_bind_aware** */
- **"_optimizer_extended_cursor_sharing"** = 'none'
  "_optimizer_extended_cursor_sharing_rel" = 'none'
- **"_optim_peek_user_binds"**= false

ACS : How to properly cancel it

We can cancel ACS using

- /*+ **no_bind_aware** */
- **"_optimizer_extended_cursor_sharing"** = 'none'
  "_optimizer_extended_cursor_sharing_rel" = 'none'
- **"_optim_peek_user_binds"**= false
- Fix a **SPM** (as from 12cR2)

ACS : How to properly cancel it

DEMO

Agenda

- ACS : Prerequisite
- ACS : Warm-up period
- ACS : step down in favor of ECS
- ACS : How to properly cancel it
- **ACS : my opinion**

ACS : my opinion

- Let ACS-ECS working **by default**
- Don’t change any relative **parameter**
- But when it starts causing performance issue you must be able to link this to ACS and to know how to **fix it**
- Use the following query to diagnose ACS issues

```
SELECT
  sql_id, count(1)
FROM
  gv$sql_cs_selectivity
GROUP BY
  sql_id
ORDER BY 2 desc;
```

ACS : my opinion

```
SELECT
  sql_id, count(1)
FROM
  gv$sql_cs_selectivity
GROUP BY
  sql_id
ORDER BY 2 desc;
```

SQL_ID	COUNT(1)
7ck8k7bnqpnv	4
fnmsn1tyq9g0y	2
cxzn1tyqhkzgb	2

```
SELECT
  sql_id, count(1)
FROM
  gv$sql_cs_selectivity
GROUP BY
  sql_id
ORDER BY 2 desc;
```

SQL_ID	COUNT(1)
7zwq7z1nj7vga	16847320
c1j862cvqgh99	512
94dp7vscw26sf	26

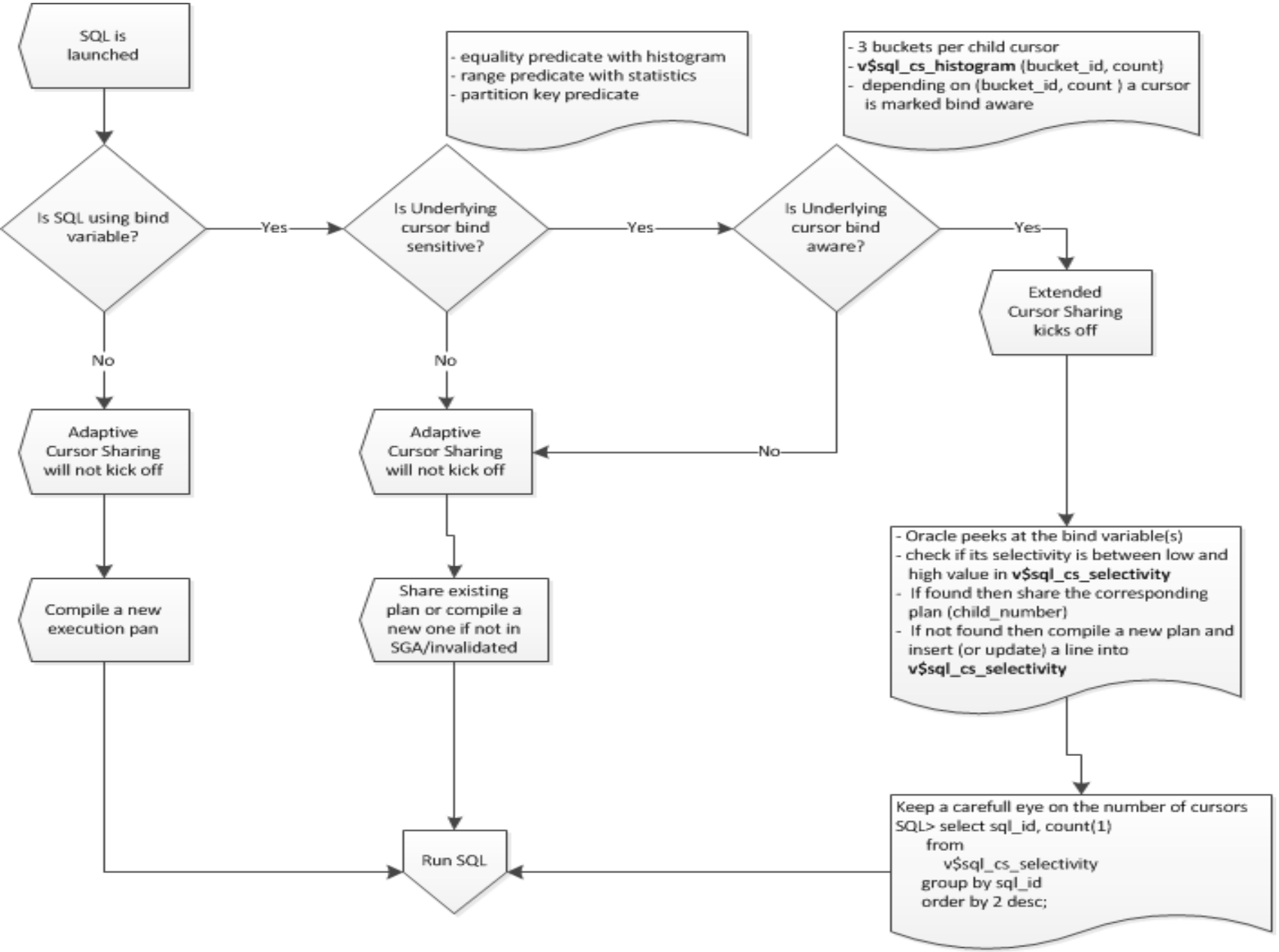
ACS : my opinion

```
SQL> @nonshared 7zwq7z1nj7vga
Show why existing SQL child cursors were not reused
(V$SQL_SHARED_CURSOR)...

-----------------
SQL_ID                : 7zwq7z1nj7vga
ADDRESS               : 000000406DBB30F8
CHILD_ADDRESS         : 00000042CE36F7E8
CHILD_NUMBER          : 0
BIND_EQUIV_FAILURE    : Y
REASON

../..

-----------------
SQL_ID                : 7zwq7z1nj7vga
ADDRESS               : 000000406DBB30F8
CHILD_ADDRESS         : 00000045B5C5E5D8
CHILD_NUMBER          : 99
BIND_EQUIV_FAILURE    : Y
```


SQL is launched
Is SQL using bind variable?
Yes
No
Adaptive Cursor Sharing will not kick off
Compile a new execution pan
- equality predicate with histogram
- range predicate with statistics
- partition key predicate
Is Underlying cursor bind sensitive?
Yes
No
Adaptive Cursor Sharing will not kick off
Share existing plan or compile a new one if not in SGA/invalidated
- 3 buckets per child cursor
- v$sql_cs_histogram (bucket_id, count)
- depending on (bucket_id, count) a cursor is marked bind aware
Is Underlying cursor bind aware?
Yes
No
Extended Cursor Sharing kicks off
- Oracle peeks at the bind variable(s)
- check if its selectivity is between low and high value in v$sql_cs_selectivity
- If found then share the corresponding plan (child_number)
- If not found then compile a new plan and insert (or update) a line into v$sql_cs_selectivity
Keep a carefull eye on the number of cursors
SQL> select sql_id, count(1)
from
v$sql_cs_selectivity
group by sql_id
order by 2 desc;
Run SQL

The End…

@MohamedHouri
www.hourim.wordpress.com