vahidusefzadeh@gmail.com

cursor sharing و bind variable

وقتی که اوراکل دستوری را اجرا می کند، سعی دارد تا فرم قابل اجرای دستور اجرا شده(که plan آن مشخص شده) را در library cache نگهدارد تا در صورت امکان از آن استفاده مجدد کند حال اگر این فرم پارس شده برای دستور دیگری مورد استفاده قرار بگیرد می گویند یک soft parse انجام شده است و در صورتی که اوراکل مجبور شود یک فرم جدید قابل اجرا(که در library cache موجود نیست) برای دستور وارده بسازد، hard parse صورت گرفته است.

شرایطی را در نظر بگیرید که دو دستور که یکی قبلا اجرا شده و فرم قابل اجرای آن در داخل library cache قرار دارد و دیگری تازه توسط کاربر صادر شده، به هم شباهت دارند ولی کاملا یکسان نیستند حال تکلیف چیست؟ می شود از فرم قابل اجرای موجود در حافظه استفاده کرد یا نه؟ به طور مثال دو دستور زیر را در نظر بگیرید:

```
select * from usef_pars where code=5;
select * from usef_pars where code=6;
```

همانطور که می بینید این دو دستور تنها در مقدار از هم متمایز می شوند ولی از لحاظ تعداد کارکتر، بزرگی و کوچکی حروف، تعداد space و .... دقیقا یکسان هستند در این حالت معمولا باید از برنامه نویس خواست تا به جای عدد از متغیر استفاده کند تا دو دستور کاملا یکسان شوند و فرم پارس شده هر دستور، برای دیگری قابل استفاده باشد ولی باز با این حال، بانک اطلاعاتی پارامتری به نام cursor_sharing دارد که می تواند در این زمینه موثر باشد. ابتدا باید منظور از cursor را مشخص کنیم.

cursor یک ناحیه حافظه در library cache می باشد که به یک دستور SQLای تخصیص داده می شود و اطلاعات مختلفی از آن دستور از قبیل execution plan و statistics را در خودش ذخیره می کند. معمولا هر دستور SQLای دو نوع cursor را به خود اختصاص می دهد:

۱. **Parent cursor**: شامل متن دستور sql می باشد. زمانی که یک دستور چند بار اجرا شود تنها یک parent cursor را به خود اختصاص می دهد البته به شرط اینکه در هنگام اجرای دوباره، هنوز در shared pool موجود باشد. به ازای هر Parent cursor تنها یک رکورد در v$SQLAREA خواهیم داشت.parent cursor بصورت استاتیک است یعنی اطلاعاتش تغییر نمی کند.

۲. **child cursor** : هر parent cursor یک یا چند child cursor را در بردارد child اطلاعات دستور sql از قبیل execution statistics، Bind variable و execution plan را شامل می شود پس اگر دو دستور یکسان با مقدار bind

vahidusefzadeh@gmail.com

variable متفاوت تنها یک parent cursor و یک child cursor داشته باشند، execution plan آنها هم یکسان خواهد بود(که این به مقدار پارامتر cursor_shring بستگی دارد).child cursor به صورت داینامیک است.

بدیهی است که هر چه تعداد child cursor و یا parent cursor **بیشتر باشد، حافظه مصرفی بیشتری هم مصرف خواهد شد.**

اطلاعات parent cursor در V$SQLAREA و اطلاعات child cursor در V$SQL ذخیره می شود همچنین child cursor فضای کمتری را نسبت به parent مصرف می کند. در ویوی v$sqlarea منظور از ستون version_count، همان تعداد child cursor می باشد.

در این قسمت سعی شده تا تاثیر مقادیر cursor_sharing بر روی رفتار cursorها نشان داده شود. همچنین از دستور زیر برای بررسی رفتار cursorها استفاده می کنیم ضمنا وقتی که sql_id/hash_value در جداول یکسان باشند، به معنی یکسان بودن parent cursorها می باشد.

```
select executions,sql_text, sql_id, child_number, hash_value, address  from v$sql where
upper(sql_text) like '%USEF_PARS%';
```

دستور زیر بیانگر دستوری است که توسط کاربر صادر می شود و منظور از n، یک مقدار است که در صورت امکان به variable تخصیص داده می شود. برای درک تاثیر مقدار پارامتر cursor_sharing، مقدار n را تغییر می دهیم.

```
select * from usef_pars where code=n;
```

بررسی اثر مقدار cursor_sharing

حالت اول: اگر cursor_sharing برابر با exact باشد، هر بار که مقدار n تغییر کند(مقدار تخصیصی به code در دستور بالا)، به هیچ وجه از فرمهای قابل اجرای قبلی موجود در حافظه که مقدار n آنها با مقدار دستور فعلی یکسان نیست، استفاده نخواهد شد و یک sql_id/hash_value جدید به این دستور تخصیص می یابد و هر دفعه یک parent cursor جدید به دستور تخصیص داده می شود همچنین یک child cursor نیز برای آن ایجاد می شود.

با توجه به جدول زیر، اگر مقدار n ثابت باشد، در صورت تکرار در اجرا، از همان فرم قبلی استفاده می شود(بر تعداد child هم افزوده نمی شود).

vahidusefzadeh@gmail.com

در صورتی که مقدار پارامتر cursor_sharing برابر با exact نبود، می توان با استفاده از hint، cursor_sharing_exact در یک دستور خاص، رفتار optimizer را تغییر داد تا از exact به جای روشهای دیگر استفاده کند.

همانطور که در ردیف آخر جدول می بینیم، هر تغییر کوچکی در عبارتهای دستور، سبب می شود تا از cursorهای قبلی استفاده نشود و cursor جدیدی ایجاد شود به طور مثال عبارت code به coDE تغییر کرده که سبب شده از فرمهای قبلی استفاده نشود.

alter system set **cursor_sharing=exact;**

statment	executions	sql_text	hash_value	sql_id	child_ number	PLAN_HASH _VALUE
where code=99	1	select * from usef.usef_pars where code=99	2101158911	bdsqajjymu8zz	0	3830149903
where code=98	1	select * from usef.usef_pars where code=98	3413937134	2rj9bb75rt2zf	0	3830149903
where code=98	2	select * from usef.usef_pars where code=98	3413937134	2rj9bb75rt2zf	0	3830149903
where coDE=98	1	select * from usef.usef_pars where coDE=98	639175319	04wpvr4m1k2nr	0	3830149903

حالت دوم: اگر cursor_sharing برابر با SIMILAR باشد، اولین باری که دستور اجرا می شود، literal replacement انجام می شود یعنی به جای literal value، یک متغیری با ساختار SYS_B_x توسط سیستم تعریف می شود(در عبارت code=n به جای n یک variable قرار می گیرد) و سپس متناسب با این دستور، plan ساخته می شود تا زمانی که plan در درون shared pool قرار دارد، دستورات بعدی که اجرا می شوند، اگر دقیقا با دستور قبلی یکسان باشد، از همان plan استفاده می شود ولی اگر مقدار این دستور با مقدار دستور قبلی یکسان نبود، باید به آمارهای موجود از جدول، ستون، ایندکسها و histogram رجوع شود و بررسی شود که توزیع داده در این ستون به چه کیفیتی است و اگر plan قبلی برای این مورد جدید بهینه بود، از همان استفاده مجدد شود وگرنه باید plan جدیدی ساخته می شود

vahidusefzadeh@gmail.com

برای درک دقیق تر باید گفت، زمانی که ستونی در یک جدول مشخص، مقداری با توزیع فراوانی یکنواخت نداشته باشد، اوراکل از آن ستون به طور خودکار در هنگام جمع آوری آمار، histogram تهیه می کند تا میزان فراوانی هر مقدار را برای این ستون مشخص کند.

برای جمع آوری آمار همرا با histogram:

```
EXEC DBMS_STATS.GATHER_TABLE_STATS(ownname=>'USEF', tabname=>'USEF_TEST',
estimate_percent=>NULL, method_opt=>'FOR ALL INDEXED COLUMNS SIZE AUTO',
degree=>DBMS_STATS.DEFAULT_DEGREE, CASCADE=>TRUE);
```

برای مثال جدولی را در نظر بگیرید که مشخصات افراد را در خودش نگه می دارد که یکی از این مشخصات، رنگ چشم افراد است همانطور که جدول مورد تست ما در زیر نشان می دهد، رنگ چشم BR در مقایسه با GY ، فاصله بسیار زیادی با هم دارند حال اگر به آمار این ستون رجوع نکنیم و اگر کاربری بخواهد به همه اطلاعات افرادی که رنگ چشم آنها BR است دسترسی داشته باشد، باید همان نقشه اجرایی که برای رنگ چشم GY وجود دارد را طی کند در صورتی که ممکن است برای اولی(منظور BR) full table scan مناسب باشد و برای دومی index scan . پس برای بهینه کردن نقشه اجرایی، در بعضی از موارد وجود histogram ضروری است که می توانیم با رجوع به dba_tab_columns و ستون histogram وضعیت histogram تمامی ستونهای جداول مختلف را ببینیم.

EYE_COLOR	count
BR	15808
BC	8384
MS	4452
GR	844
HN	256
BL	192
GY	64

تعیین selectivity مقادیر ستون EYE_COLOR:

BR ➔15808/30000= ۰.۵۲۶ **BC➔0.279** **MS➔0.148** **GR➔0.028** **HN➔0.008**

BL➔0.006 **GY➔ 0.002**

نقشه اجرایی زیر نکته بالا را به خوبی نشان می‌دهد.

vahidusefzadeh@gmail.com

alter system flush shared_pool;

set autotrace traceonly explain

select * from usef.usef_test where eye_color='BR';

Plan hash value: 289291562

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name      | Rows  | Bytes | Cost (%CPU)     | Time      |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |           | 17320 |  282M|   507   (1)      | 00:00:07  |
|* 1 |  TABLE ACCESS FULL| USEF_TEST | 17320 |  282M|   507   (1)      | 00:00:07  |
-------------------------------------------------------------------------------------------------------
```

select * from usef.usef_test where eye_color='GY';

Plan hash value: 2683956848

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)|Time      |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |   64  | 1069K|   14   (0)   |00:00:01  |
|   1 |TABLE ACCESS BY INDEX ROWID| USEF_TEST |   64  | 1069K|   14   (0)   |00:00:01  |
|* 2 | INDEX RANGE SCAN              | INDEX_USEF|   64  |      |    1   (0)   |00:00:01  |
-------------------------------------------------------------------------------------------------------
```

از نظر مصرف shared pool هم باید گفت که برای دستور با یک bind_variable، تنها یک parent cursor جود خواهد داشت و با هر بار تغییر literal، یک child cursor جدید ساخته می شود برای مثال، وقتی code برابر با ۹ است، parent cursor تنها یک child cursor دارد ولی وقتی مقدار به ۱۰ تغییر می کند، child cursor جدیدی ایجاد می شود و با ایجاد آن، ممکن است PLAN_HASH_VALUE جدیدی ایجاد شود و در نهایت باعث تغییر در execution plan شود.

پس با انتخاب similar، تعداد parent cursorها را کاهش خواهیم داد که این نسبت به روش exact مزیت خوبی است و سبب صرفه جویی بیشتری در مصرف حافظه library cache خواهد شد ولی با هر بار تغییر مقدار n، بر تعداد child cursor افزوده خواهد شد و به دلیل child cursor زیادی که ایجاد می شود، هزینه زیادی را برای cpu ایجاد می کند.

وقتی از عملگر LIKE استفاده می کنیم، هیچگونه shareای بین cursorها اتفاق نخواهد افتاد.

vahidusefzadeh@gmail.com

روش similar نسبت به روشهای دیگر کمتر توصیه می شود و همچنین از اوراکل 12c به بعد حذف شده است(MOS Note 1169017.1) ولی کماکان این امکان وجود دارد تا مقدار پارامتر cursor_sharing را به similar تغییر بدهیم احتمالا این قضیه به خاطر backward compatibility می باشد.

در جدول زیر، در دو ردیف آخر نشان داده شده که اگر مقدار ستون eye_color اگر از BR به GY تغییر کند، یک child جدید ساخته می شود و نیز مقدار plan هم تغییر می کند در صورتی که parent کماکان یکی هست.

alter system set **cursor_sharing=SIMILAR;**

statment	executions	sql_text	hash_value	sql_id	child_ number	PLAN_HASH _VALUE
where code=9	1	select * from usef.usef_pars where code=:"SYS_B_0"	1925676697	3pmxy89tcfznt	0	3830149903
where code=10	1	select * from usef.usef_pars where code=:"SYS_B_0"	1925676697	3pmxy89tcfznt	1	3830149903
where code=10	2	select * from usef.usef_pars where code=:"SYS_B_0"	1925676697	3pmxy89tcfznt	1	3830149903
where eye_color= 'BR'	1	select * from usef.usef_test where eye_color=:"SYS_B _0"	1396660275	2vz1dht9myq1m	0	289291562
where eye_color= 'GY'	1	select * from usef.usef_test where eye_color=:"SYS_B _0"	2241049013	3j8n9ba2t7cdp	1	2683956848

حالت سوم: اگر cursor_sharing برابر با force باشد، همانند similar، literal replacement انجام می شود ولی وقتی اولین plan برای دستور ساخته شد، از همان برای بقیه دستورات استفاده می کند(منظور دستورات یکسان است) حتی اگر literal value تغییر کند. پس تنها یک parent cursor و یک child cursor خواهیم داشت.

vahidusefzadeh@gmail.com

این کار سبب صرفه جویی در مصرف فضای shared pool می شود ولی کمی ناامن است به دلیل اینکه ممکن است plan قبلی برای دستور فعلی مناسب نباشد(به همان دلایلی که در قسمت similar گفته شد) و ممکن است به کارایی ضربه وارد شود.

دو ردیف اخر جدول زیر، این نکته را به خوبی نشان می دهند چرا که همان plan قبلی برای دستور بعدی استفاده شده است.جدول زیر این نکته را هم به خوبی تبیین می کند که تنها یک child cursor برای مقادیر مختلف code ایجاد شده است.

alter system set **cursor_sharing=force**;

statment	executions	sql_text	hash_value	sql_id	child_ number	PLAN_HASH _VALUE
where code=68	1	select * from usef.usef_pars where code=:"SYS_B_0"	1925676697	3pmxy89tcfznt	0	3830149903
where code=69	2	select * from usef.usef_pars where code=:"SYS_B_0"	1925676697	3pmxy89tcfznt	0	3830149903
where code=10	3	select * from usef.usef_pars where code=:"SYS_B_0"	1925676697	3pmxy89tcfznt	0	3830149903
where co**DE**=70	1	select * from usef.usef_pars where coDE=:"SYS_B_0"	639175319	04wpvr4m1k2nr	0	3830149903
where eye_color= 'BR'	1	select * from usef.usef_test where eye_color=:"SYS_B _0"	2934748358	5rwynkyrftd66	0	289291562
where eye_color= 'GY'	2	select * from usef.usef_test where eye_color=:"SYS_B _0"	2934748358	5rwynkyrftd66	0	289291562

vahidusefzadeh@gmail.com

ایجاد bind variable توسط برنامه نویس

اگر در برنامه از bind variable استفاده شود، و پارامتر cursor_sharing هم در مقدار پیش فرضش باقی بماند، چنین جدولی از اجرای دستورات زیر حاصل می شود.

```
variable var_code number
exec :var_code:=5

select * from usef.usef_pars where code=:var_code;
```

statment	executions	sql_text	hash_value	sql_id	child_ number
exec :var_code: =5	1	select * from usef.usef_pars where code=:var_code	2360338082	1r6axyq6aztp2	0
exec :var_code: =6	2	select * from usef.usef_pars where code=:var_code	2360338082	1r6axyq6aztp2	0
exec :var_code: =6	3	select * from usef.usef_pars where code=:var_code	2360338082	1r6axyq6aztp2	0
exec :var_coDE: =6	1	select * from usef.usef_pars where coDE=:var_code	1149264764	9kqv175280svw	0

(ACS)adaptive cursor sharing

این ویژگی از اوراکل 11g فعال شده و تلاش کرده تا اثر منفی عدم توجه به selectivity در cursor sharing را کاهش دهد البته این ویژگی ارتباط مستقیمی با پارامتر cursor_sharing ندارد. منظور از adaptive برای cursor sharing، یعنی همیشه یک plan را برای مقادیر مختلف یک bind variable استفاده نخواهد کرد بلکه انتخاب plan را با شرایط وفق می دهد.

دو شرط اصلی استفاده از این ویژگی در هنگام اجرای دستورات به صورت زیر است:

vahidusefzadeh@gmail.com

۱. bind_variable موجود باشد چه از طریق پارامترهای اوراکل و چه از طریق برنامه نویس.

۲. هیستوگرام ستون مورد نظر هم موجود باشد.

برای تعیین این نکته که آیا ACS برای یک دستور خاص بکار رفته یا نه می توان به دو ستون v$sql با نامهای IS_BIND_SENSITIVE و IS_BIND_AWARE رجوع کرد و همچنین سه ویوی جدید با اسامی V$SQL_CS_HISTOGRAM ، V$SQL_CS_SELECTIVITY و V$SQL_CS_STATISTICS در این زمینه مفید هستند.

معمولا ACS برای بار اولی که مقدار متغیر تغییر می کند، plan خوبی را نمی سازد(البته اگر بین این مقادیر با مقدار قبلی از نظر کمیت تفاوت زیادی وجود داشته باشد) و دوباره plan قبلی را برای دستور فعلی ارائه می دهد ولی ستون IS_BIND_AWARE را برابر با Y قرار می دهد تا به optimizer هشدار داده باشد و زمانی که برای بار دوم دستور با این مقدار اجرا شد، plan بهینه مربوط به این مقدار ساخته می شود.

برای غیرفعال کردن این ویژگی، باید از دو پارامتر زیر استفاده کنیم:

```
_optimizer_adaptive_cursor_sharing = false
_optimizer_extended_cursor_sharing_rel = "none"
```

جدولی که در این قسمت خواهیم دید، نقش ACS را به خوبی نشان می دهد زمانی که دستور دوم با مقداری متفاوت از مقدار اول اجرا شد، باز هم از همان child cursor قبلی استفاده می شود ولی در اجرای بعدی، از cursor جدیدی برای این دستور با این مقدار ایجاد می شود.

```
EXEC DBMS_STATS.GATHER_TABLE_STATS(ownname=>'USEF', tabname=>'USEF_TEST',
estimate_percent=>NULL, method_opt=>'FOR ALL INDEXED COLUMNS SIZE AUTO',
degree=>DBMS_STATS.DEFAULT_DEGREE, CASCADE=>TRUE);


variable var_eye VARCHAR2(2)

exec :var_eye:='BR';

select /*+ qb_name(acs) */  count(*) from usef.usef_test where eye_color=:var_eye;
```

vahidusefzadeh@gmail.com

```
select executions,sql_text, sql_id, child_number,
hash_value,PLAN_HASH_VALUE,IS_BIND_SENSITIVE,IS_BIND_AWARE  from v$sql  where upper(sql_text) like
'%USEF_TEST%';
```

در جدول زیر دستورات به ترتیب اجرا شده اند تا رفتار ستونهای ویوی v$sql در شرایط مختلف دیده شود.

statment	executions	sql_text	hash_value	sql_id	PLAN_HASH_VALUE	child_number	IS_BIND_SENSITIVE	IS_BIND_AWARE
exec :var_eye:= **'BR'**	1	select count(*) from usef.usef_test where eye_color= :var_eye	1411417384	9rzcryda21198	2202223041	0	Y	N
exec :var_eye:= **'GY'**;	2	select count(*) from usef.usef_test where eye_color= :var_eye	1411417384	9rzcryda21198	2202223041	0	Y	N
exec :var_eye:= **'BR'**;	1	select count(*) from usef.usef_test where eye_color= :var_eye	1411417384	9rzcryda21198	2202223041	1	Y	Y
exec :var_eye:= **'GY'**;	1	select count(*) from usef.usef_test where eye_color= :var_eye	1411417384	9rzcryda21198	3732300935	۲	Y	Y

بعد از اجرای ترتیبی دستورات جدول بالا، ویوی V$SQL_CS_SELECTIVITY به صورت زیر تغییر می کند.

```
select * from v$sql_cs_selectivity;
```

HASH_VALUE	sql_id	child_number	PREDICATE	RANGE_ID	low	high
1411417384	9rzcryda21198	2	=VAR_EYE	0	0.001927	0.002355
1411417384	9rzcryda21198	1	=VAR_EYE	0	0.476017	0.581799

vahidusefzadeh@gmail.com

نتیجه گیری:

۱.معمولا توصیه می شود که برای محیط oltp پارامتر CURSOR_SHARINGرا به force تنظیم کنیم و در محیط DW، این پارامتر در همان مقدار پیش فرضش یعنی exact باقی بماند.

۲. سه دستاورد اصلی دو مقدار force و similar را می توان به این صورت خلاصه کرد:

کاهش مصرف حافظه، کاهش latch contention و افزایش سرعت پارس دستور

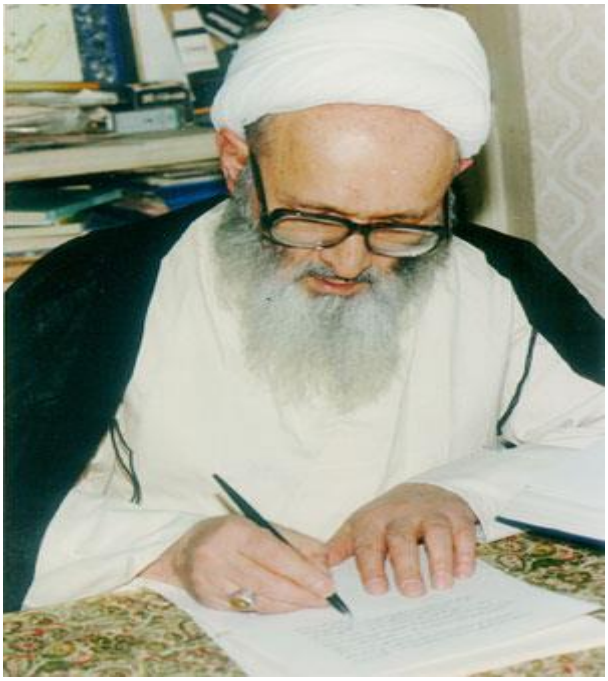
۳. adaptive cursor sharing از اوراکل 11g به بعد، به طور پیش فرض فعال شده است و اوراکل با این پارامتر سعی کرده تا اثر منفی عدم توجه به selectivity را در cursor sharing کاهش دهد.

۴. در سیستمی که از مقدار exact برای cursor_sharing استفاده شده و برنامه نویس هم bind variable را کم در برنامه بکار برده، می توانیم با دستورات زیر، plan_hash_valueهای تکراری را مشخص کنیم و بعد با رجوع به متن دستور، مشخص کنیم که کدام دستور به خاطر عدم استفاده از bind variable، مجبور به پارس مجدد شده است(البته ممکن است دستورات غیر یکسان هم باشند) و در نهایت آنها را به برنامه نویس داد تا این مشکل را حل کند:

```
select plan_hash_value, count(plan_hash_value) from v$sql group by plan_hash_value having
count(plan_hash_value) > 10 order by count(plan_hash_value) desc;

select * from v$sql where plan_hash_value='4167636152';
```

vahidusefzadeh@gmail.com



علامه حسن حسن زاده آملی:

إلهی، در ذاتِ خودم متحیرم تا چه رسد در ذاتِ تو.