vahidusefzadeh@gmail.com

table reorganization

معمولا بعد از مدتی کار بر روی جداول و یا در پی اتفاقاتی خاص، ممکن است که نیاز شود تا جدولی خاص از یک tablespace و یا کل objectهای tablespace را دوباره سازماندهی کنیم tablespace reorganization معمولا به عنوان یک امر خطیر در حیطه وظایف dba محسوب می شود. آنچه که در این قسمت قصد داریم در مورد آن مطالبی را ارائه کنیم، دلایل نیاز به reorganization و چگونگی انجام آن می باشد.

فرض کنید جدولی با اطلاعات بسیار زیاد در بانک وجود دارد که تصمیم می گیریم بیشتر یا تمامی اطلاعات آن را با دستور delete حذف کنیم تا از فضای ازاد شده استفاده کنیم. بعد از حذف اطلاعات جدول، خواهیم دید که هیچ فضایی به tablespace برنگشته برای درک چرایی این مطلب، باید درک دقیقی از مفهوم high water mark داشته باشیم و بتوانیم آن را مدیریت کنیم تا در نهایت، فضای مربوط به این جدول، قابل استفاده برای سگمنتهای دیگر باشد.

HWM(high water mark)،به انتهای بلاکهای مصرف شده یک سگمنت اشاره دارد و سبب می شود که همه اطلاعات جدول، قبل از این نقطه قرار بگیرند و هر چه فضای سگمنت بعد از این نقطه قرار بگیرند، به راحتی قابل بازپس گیری هستند. به عبارتی دیگر، HWM سبب می شود تا used(ever) و unused(never) یک سگمنت از هم جدا باشند.

زمانی که یک جدول ایجاد می شود، در ابتدا اوراکل به سگمنت این جدول، به اندازه initial extent فضا تخصیص می دهد که نقطه HWM آن در ابتدا قرار می گیرد. در زمانی که اوراکل بخواهد اطلاعات این جدول را بخواند، تنها بلاکهایی که زیر HWM هستند را ملاک قرار می دهد و همچنین زمانی که اطلاعاتی از جدول حذف می شوند، باز هم ملاک برای برگرداندن فضا، HWM می باشد یعنی اگر بلاکهایی در زیر HWM قرار داشته باشند، اگرچه حاوی اطلاعاتی نباشند، نمی توان از آنها برای سگمنتهای دیگر استفاده کرد پس باید HWM مربوط به جدول را کاهش دهیم تا بتوانیم بهینه سازی در سرعت full table scan و نیز تخصیص فضا حاصل کنیم.

نکته دیگری که باید در این زمینه به آن توجه کرد، تفاوت HWM بین دیتافایل و جدول می باشد. همانطور که هر جدول یک HWM خاص خود را دارد، هر دیتافایل نیز HWM مخصوصی دارد یعنی اگر بخواهیم فضای گرفته شده datafile را کاهش دهیم، تنها می توانیم تا HWM دیتافایل این کار را انجام دهیم به عبارتی دیگر، تا انتهای اخرین extent در حال استفاده.

حال برای اینکه HWM مربوط به datadfile را کاهش دهیم، باید objectای که بلاکهای اخر دیتافایل را اشغال کرده، شناسایی کنیم و در نهایت آن را حذف کنیم در غیر این صورت، با خطای زیر مواجه می شویم:

ORA-03297: file contains used data beyond requested RESIZE value

در صورتی که بخواهیم تعیین کنیم کدام یک از objectها از این کار جلوگیری می کنند، می توانیم از دستور زیر استفاده کنیم:

```
select segment_name,l.owner,l.bytes/1024/1024 from dba_extents l where file_id=5 and
((block_id + blocks-1)*8192)/1024/1024 > 240;
```

منظور از 240 در دستور بالا، کاهش حجم دیتافایل تا اندازه 240MB می باشد که اگر بخواهیم چنین کاری انجام شود، دستور بالا لیست تمامی اشیاهایی که سد راه هستند را لیست می کند.

سناریوی زیر می تواند درک بهتری را از نقش HWM به ما دهد.

فرض کنید tablespace ای با نام USEF_TBS می‌سازیم که تنها یک دیتافایل دارد، همچنین جدولی با اندازه 40MB بر روی این tablespace ایجاد می‌کنیم.

```
select sum(bytes)/1024/1024 "table_size" from dba_segments where segment_name='USEF' and
tablespace_name='USEF_TBS';
```

table_size

40

HWM مربوط به این جدول به صورت زیر محاسبه می شود:

```
select max((block_id + blocks-1)* 8192)/1024/1024 "HWM_SIZE" from dba_extents  where owner='USEF' and
file_id=5 and segment_name='USEF';
```

HWM_SIZE

40.9921875

حال جدولی دیگر با اندازه 40MB، به همین دیتافایل اضافه می کنیم و با دستور زیر HWM مربوط به دیتافایل را محاسبه می کنیم:

```
SELECT  file_size, hwm, file_size-hwm can_save FROM (SELECT /*+ RULE */ ddf.tablespace_name,
ddf.file_name file_name, ddf.bytes/1048576 file_size,(ebf.maximum + de.blocks-
1)*dbs.db_block_size/1048576 hwm FROM dba_data_files ddf,(SELECT file_id, MAX(block_id) maximum FROM
dba_extents GROUP BY file_id) ebf,dba_extents de, (SELECT value db_block_size FROM v$parameter WHERE
name='db_block_size') dbs WHERE ddf.tablespace_name='USEF_TBS' and ddf.file_id = ebf.file_id AND
de.file_id = ebf.file_id AND de.block_id = ebf.maximum ORDER BY 1,2);
```

FILE_SIZE HWM CAN_SAVE

---------- ---------- ----------

500 80.9921875 419.007813

منظور از can_save، میزان فضای قابل برگشت دیتافایل می باشد.

vahidusefzadeh@gmail.com

حال در صورتی که جدول اول را حذف کنیم، غیر از فضای can_save که قبلا هم قابل برگشت بود، هیچ فضای دیگری از datafile قابل برگشت نیست:

drop table usef;

دوباره فضا را چک می کنیم:

```
FILE_SIZE       HWM         CAN_SAVE
----------      ----------      ----------
       500      80.9921875      419.007813
```

همانطور که در مثال بالا دیدیم، فضای قابل بازیابی دیتافایل(CAN_SAVE) بعد از drop table تغییری نکرد که البته دلیل آن، سد راه بودن جدول دوم بود. در جدول زیر مثالی در مورد نقش دستورات delete، truncate، drop و create ذکر شده است که نشان می دهد delete هیچ تاثیری در مقدار HWM دیتافایل و جدول نداشته ولی truncate سبب شده تا HWM جدول تغییر کند ولی برای datafile بستگی دارد به اینکه این جدول در کجا دیتافایل قرار دارد.

	Table_MB	Table_blocks	TBS-used-MB	datafile-total-MB	HWM_datafile
Before action	37	4736	893.5	937	936.11
delete	37	4736	893.5	937	936.11
truncate	0.06	8	856.56	937	936.11
Resize datafile	-	-	-	936.11	-
drop	-	-	856.5	937	936.11
Create table	42.5	5440	899.56	937	936.11

بررسی روشهای reorganize table

برای حل مشکل مطرح شده، روشهایی وجود دارد که در این قسمت مزایا و معایب هر کدام از روشها را با ذکر مثال توضیح خواهیم داد(در مثالها حجم فضا، بر اساس مگابایت است):

1. shrink table:

Shrink یکی از روشهای بسیار ساده برای reorganize table می باشد که برای انجام آن باید ROW MOVEMENT جدول را فعال کنیم چون قرار است بعضی rowidها در صورت نیاز جابجا شوند. یکی از فرقهای مهم این دستور با دستور alter table move، invalid نکردن ایندکسهای جدول می باشد که دلیل این موضوع، به نحوه انجام عمل shrink بر می گردد در روش shrink برای جداسازی فضای در حال استفاده با فضای خالی، rowidها با insert و delete جابجا می شود و این جابجایی در نهایت به ایندکسها هم اعمال می شود تا ایندکس نامعتبر نشود و آدرس rowidای که اشاره می کند، بروز شود. این روش عملیاتش را در دو فاز data restructuring (compact) و HWM adjustment انجام می دهد که در مرحله اول عملیات insert و delete انجام می شود و در مرحله دوم، فضا آزاد می شود در زمان انجام مرحله HWM adjustment، جدول به طور کامل

vahidusefzadeh@gmail.com

جلوی دستورات dml را می گیرد البته مدت زمان آن به نسبت کم هست ولی توصیه می شود که مرحله اول با دستور compact در طول روز انجام شود و مرحله دوم در پیک کاری سیستم انجام نشود.

نکته: بعد از اینکه عملیات shrink به پایان رسید، توصیه می شود که یک بار بانک را recompile کنید(یکی از روشها استفاده از utlrp.sql می باشد).

دستورات مرتبط با shrink:

```
ALTER TABLE usef ENABLE ROW MOVEMENT;

ALTER TABLE usef SHRINK SPACE CASCADE;

ALTER TABLE usef MODIFY LOB (PICTURE) (SHRINK SPACE);

ALTER TABLE usef OVERFLOW SHRINK SPACE;
```

مثال:

```
select sum(bytes/1024/1024) from dba_segments where segment_name='USEF';

SUM(BYTES/1024/1024)

--------------------

         707.375
```

```
alter table usef enable row movement;
alter table usef shrink space;
```

```
select sum(bytes/1024/1024) from dba_segments where segment_name='USEF';

SUM(BYTES/1024/1024)

--------------------

         471.4375
```

جدول زیر مثالی از اجرای دستور shrink بر روی جدولی را نشان می دهد که بعد از shrink، تغییراتی در اندازه جدول ایجاد شده ولی تاثیری بر روی HWM دیتافایل نگذاشته است دلیل این موضوع آن است که این جدول، جزو اشیاهای انتهایی دیتافایل نبوده است.

	Table_MB	Table_blocks	TBS-used-MB	TBS-total-MB	HWM_DATAFILE
Org-size	8	1024	907.1875	937	936.11
Shrink compact	8	1024	907.1875	937	936.11
Shrink	7.25	928	906.4375	937	936.11

مزایا:

1. HWM جدول تغییر می کند و فضای خالی جدول بر می گردد.

2. ایندکسها معتبر باقی می مانند.

3. برای انجام shrink، نیاز به فضای اضافی نیست.

4. برای Heap-organized tables and index-organized tables، Indexes، Partitions and subpartitions، Materialized views and materialized view logs و LOB segments قابل انجام است.

5. امکان گرفتن query از جدول در حین انجام عملیات shrink وجود دارد.

محدودیتها و معایب:

1. جدول به صورت exclusive mode قفل نمی شود(البته به جز لحظه HWM adjustment) بلکه بصورت LOCKED_MODE=3 (Row Lock Mode) قفل می شود.

2. تنها برای tablespaceهای با مدیریت سگمنت ASSM قابل انجام خواهد بود.

3. تمامی فرمهای پارس شده دستورات sqlای که به این جدول(منظور جدول shrink شده است) اشاره می کردند، invalid می شوند.

4. redo و undo زیادی مصرف می کند.

نکته: برخلاف دستور shrink، shrink compact قابلیت انجام online بدون هیچ LOCKای را ممکن می سازد یعنی جلوی دستورات DMLای بر روی جدول را نمی گیرد. و نیز فضای استفاده شده جدول را یکپارچه می کند ولی بر روی HWM تاثیری ندارد و فضایی بر نخواهد گشت.

2. Online Redefinition:

این روش ابتدا با استفاده از Create table as select، جدول موقتی را می سازد و اطلاعات جدول اصلی را بر روی آن کپی می کند بعد از آن snapshot ای را بر روی جدول می سازد تا بتواند تمامی تغییرات بعد از مرحله start را در نهایت بر جدول موقت اعمال کند(re-synchronize).

مثال:

1. فرض کنید جدولی با اندازه 552MB داریم:

```
select sum(bytes)/1024/1024 "table_size" from dba_segments where segment_name='USEF';

table_size

----------

       552
```

در ابتدا باید بررسی کنیم جدول usef، قابل redifintion است یا نه:

```
SQL> exec dbms_redefinition.can_redef_table ('USEF_USR', 'USEF');

PL/SQL procedure successfully completed
```

vahidusefzadeh@gmail.com

خروجی دستور بالا می گوید که هیچ مشکلی برای انجام کار نداریم.

2. حال باید جدولی بسازیم تا بتوانیم از آن به عنوان Interim Table استفاده کنیم(جدول موقتی که قرار است جایگزین جدول اصلی شود) در هنگام ساخت این جدول، می توانیم تغییراتی را اعمال کنیم به طور مثال، tablespace را تغییر دهیم.

```
create table usef_temp tablespace users as select * from usef where 1=2;
```

3. در این مرحله باید اطلاعات جدول اصلی را در درون جدول موقت قرار دهیم.

```
exec dbms_redefinition.start_redef_table ('USEF_USR', 'USEF', 'USEF_TEMP');

PL/SQL procedure successfully completed
```

همانطور که دستور زیر نشان می دهد، اطلاعات جدول اصلی به جدول موقت منتقل شده است:

```
select count(*) from usef_usr.usef_temp;

  COUNT(*)

----------

     10000
```

در این مرحله می توانیم ایندکسهایی را به جدول جدید اضافه کنیم و نیز تغییرات دیگری هم به جدول اضافه کرد.

4. در نهایت برای اعمال همه تغییرات و سوییچ نهایی دو جدول، باید دستور زیر را صادر کنیم:

```
exec dbms_redefinition.finish_redef_table ('USEF_USR','USEF','USEF_TEMP');

PL/SQL procedure successfully completed
```

کار تمام شد.

حال باید دید که جدول جدید چقدر فضا را به خود اختصاص داده است:

```
select l.owner,l.segment_name,l.tablespace_name,sum(bytes)/1024/1024 "table_size" from
dba_segments l  where l.segment_name='USEF' group by l.owner, l.segment_name ,
l.tablespace_name;

OWNER                SEGMENT_NAME    TABLESPACE      table_size

------------         ----------      ----------      ----------

USEF_USR             USEF            USERS           40
```

بله درست می بینید! فضای مصرفی جدول از 552MB به اندازه 40MB کاهش یافت.

حال می توانیم جدول USEF_TEMP را حذف کنیم.

مزایا:

vahidusefzadeh@gmail.com

1.انجام کار بدون کمترین down-time.

2. به جز وقفه کوچک در انتهای کار، عملیات update و select بر روی جدول کاملا ممکن است.

3. امکان حذف، اضافه، تغییر نام و تغییر نوع ستون جدول را فراهم می کند. می توان نوع داده long را به lob تبدیل کرد.

4. امکان افزودن و کپی ایندکس به جدول و همچنین پارتیشن بندی جدول و به طور کلی تغییر table organization.

5. امکان تغییر constraint definitions جدول.

6. امکان انتقال جدول به tablespaceای دیگر

محدودیتها و معایب:

1. نیاز به فضای اضافه دارد یعنی حداقل دو برابر اندازه فعلی جدول.

2. در انتهای کار که می خواهد بین جدول اصلی و جدول موقت سوییچ کند، جدول را lock می کند.

3. جدول باید primary key داشته باشد.

4. جداول نباید در یوزر sys یا system باشند.

5. جدول نباید ستونی از نوع LONG ،FILE داشته باشد البته می توان آن را convert کرد.

6. باید بسیار مواظب بود تا ایندکسها و constraintها را هم منتقل کرد.

3. expdp/impdp:

در سیستمی که قرار است حجم زیادی از اطلاعات reorg شوند و همچنین down time برای آن سیستم ممکن باشد(برای مثال در روز تعطیل کار نمی کنند) روش exp/imp، روشی معقول است. منظور از حجم زیاد به طور سرانگشتی بیشتر از حدود 200GB می باشد. هر چند همیشه کار ساده تر آن است که بگوییم **بستگی دارد!**

مثال:

اندازه و مشخصات جدول قبل از export:

```
select l.owner,l.segment_name,l.tablespace_name,sum(bytes)/1024/1024 "table_size" from
dba_segments l  where l.segment_name='USEF_TEMP' group by
l.owner,l.segment_name,l.tablespace_name;

OWNER     SEGMENT_NAME     TABLESPACE table_size

---------- ---------------- ---------- ----------

USEF      USEF_TEMP        USEF_TBS        552
```

شروع عملیات export:

vahidusefzadeh@gmail.com

```
create directory usef_dir  as '/u01/oracle';
```

expdp DIRECTORY=usef_dir **dumpfile**='expdat.dmp' **tables**='USEF_TEMP'

حذف جدول اصلی از بانک:

```
drop table usef_temp;
```

import جدول مورد نظر:

impdp DIRECTORY=usef_dir **dumpfile**='expdat.dmp' **tables**='USEF_TEMP'

حال ببینیم فضای مربوط به این جدول چقدر است:

```
dba_segments l  where l.segment_name='USEF_TEMP' group by l.owner, l.segment_name ,
l.tablespace_name;
```

OWNER	SEGMENT_NAME	TABLESPACE	table_size
USEF_USR	USEF	USERS	40

همانطور که دیدید، فضای مصرفی به اندازه روش قبلی کاهش پیدا کرد.

مزایا:

1. برای مواردی که حجم داده خیلی بالا باشد، این روش مناسب است.

2.نیاز به فضای اضافه نداریم(البته در بانک منظور است)

3.هیچ محدودیتی ندارد برای مثال، نوع داده long را هم قبول می کند.

محدودیتها و معایب:

1. نیاز به down time داریم.

2. بسیار کند است.

3. از نگاهی کمی ریسک دارد زیرا data از بانک حذف شده و در خارج از آن قرار گرفته است.

4. به نسبت روشهای دیگر، پیچیدگی بیشتری دارد.

4. alter table move:

```
Alter table table_name move tablespace_name;
```

OR

```
alter table table_name move;
```

vahidusefzadeh@gmail.com

مزایا:

1. امکان استفاده از بقیه objectهای بانک وجود دارد(برخلاف imp/exp). یعنی امکان online بودن سیستم وجود دارد.

2. امکان query گرفتن از جدول وجود دارد البته با اندکی وقفه.

3.به نسبت بقیه روشها، بسیار ساده است(البته با فرض تغییر tablespace).

4. مشکل row-chaining را حل می کند.

5.ایندکسها و constraintهای جدول، حفظ می شوند.

معایب:

1. جدول به صورت Exclusive mode(LOCKED_MODE=6) قفل می شود.

2.تمامی ایندکسها unusable می شوند.یعنی نوعی down time در ساخت مجدد ایندکس خواهیم داشت.

3. نیاز به دو برابر فضای مصرفی جدول در هنگام انجام عملیات move.

5. CTAS:

مثال:

```
select sum(bytes/1024/1024) from dba_segments where segment_name='USEF';

SUM(BYTES/1024/1024)

--------------------

          1328

SQL> alter table usef read only;

Table altered

SQL> create table usef_temp1 tablespace usef_tbs as select * from usef;

Table created

SQL> drop table usef;

Table dropped

SQL> alter table usef_temp1 rename to usef;

Table altered

select sum(bytes/1024/1024) from dba_segments where segment_name='USEF';

SUM(BYTES/1024/1024)
```

vahidusefzadeh@gmail.com

مزایا:

1. امکان استفاده از بقیه objectهای بانک وجود دارد(برخلاف imp/exp).

2. امکان query گرفتن از جدول وجود دارد.

3. با استفاده از پارامتر parallel، می توان عملیات CTAS را به صورت موازی انجام داد. درجه موازی سازی به تعداد cpu بستگی دارد.

4. قابلیت استفاده از ORDER BY را در هنگام ساخت جدول می دهد.

معایب:

1. جدول باید read only شود.

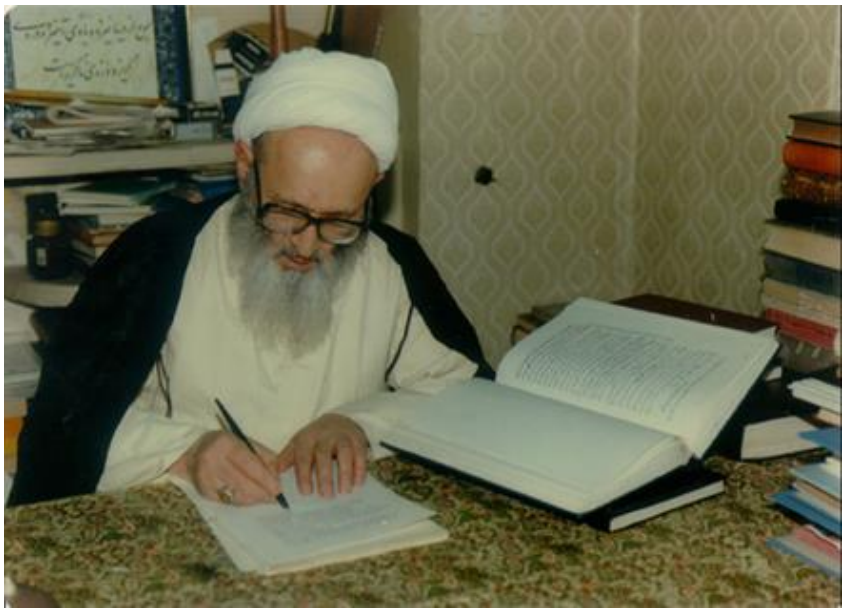
2.مقداری down time به خاطر حذف جدول قدیمی

3. مقداری down time به خاطر تغییر نام جدول جدید

4. نیاز به ساخت دوباره index و constrain

5.نیاز به فضای اضافی

vahidusefzadeh@gmail.com



علامه حسن حسن‌زاده آملی:

الهی چونست که در خود می‌نگرم به تو نزدیک می شوم و در تو می‌نگرم از تو دور.