最近看了一些关于Oracle的Buffer Cache和Keep Pool的文章，有了一些想法：如果把ArcSDE的图层或索引加载到内存里，会不会提升查询的速度呢？

相关的介绍文章如下：

<http://blog.csdn.net/leshami/article/details/6208594>

<http://blog.csdn.net/tianlesoftware/article/details/6581159>

所以，做了下面这个实验，看是否能够通过将图层或索引设置为Keep Pool，保持在内存中，提高图层的查询效率。

实验环境：虚拟机（2核/3G内存），数据库Oracle 11.2.0.1，ArcSDE 10.0，一个包含大约35万个多边形要素的Shp图层

**步骤一：环境准备**

在虚拟机里安装完成Oracle和ArcSDE后，连接到Oracle，首先查看并修改相关的参数

查看SGA中Buffer cache 大小及其它信息

SQL> select \* from v$sgainfo;

NAME                                  BYTES RESIZEABLE

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

Fixed SGA Size                      2180544 No

Redo Buffers                        6549504 No

Buffer Cache Size                 310378496 Yes

Shared Pool Size                  226492416 Yes

Large Pool Size                     4194304 Yes

Java Pool Size                      4194304 Yes

Streams Pool Size                   4194304 Yes

Shared IO Pool Size                       0 Yes

Granule Size                        4194304 No

Maximum SGA Size                  855982080 No

Startup overhead in Shared Pool    71303168 No

Free SGA Memory Available         297795584

12 rows selected

查询KeepPool大小

SQL> show parameter db\_keep\_cache\_size

NAME                                 TYPE        VALUE

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

db\_keep\_cache\_size                   big integer 0

默认为0，现手动修改大小

SQL> alter system set db\_keep\_cache\_size=250M scope=both sid='orcl';

System altered

再查看KeepPool

SQL> show parameter db\_keep\_cache\_size

NAME                                 TYPE        VALUE

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

db\_keep\_cache\_size                   big integer 252M

**步骤二：图层导入**

将图层导入ArcSDE中，并拷贝两份，分别命名为：sde\_ori      sde\_index\_mem    sde\_all\_mem，各表情况如下

|  |  |  |
| --- | --- | --- |
| 图层名 | 空间索引名 | 索引表名 |
| sde\_ori | A2\_IX1 | s2\_idx$ |
| sde\_index\_mem | A4\_IX1 | s4\_idx$ |
| sde\_all\_mem | A5\_IX1 | s5\_idx$ |

sde\_ori表保持原状

将sde\_index\_mem图层的索引表均调入内存，alter table s4\_idx$ storage(buffer\_pool keep);

将sde\_all\_mem图层的主表及索引表均调入内存，alter table sde\_all\_mem storage(buffer\_pool keep);alter table s5\_idx$ storage(buffer\_pool keep);

**步骤三：空间查询实验**

**1.首次查询**

清空缓存：

alter system flush shared\_pool;

alter system flush buffer\_cache;

对这三个表进行一次空间查询，构造一个长方形，坐标为（90，30）（93，30）（93，33）（90，33），选择图层中与之相交的元素，返回ID

首先是原表sde\_ori:

SQL> select dataid from sde\_ori where sde.st\_intersects(shape,sde.st\_geometry('POLYGON((90 30,93 30,93 33,90 33,90 30))',2))=1;

已选择984行。

已用时间:  00: 00: 09.23

统计信息

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

      17522  recursive calls

        150  db block gets

      12924  consistent gets

       2110  physical reads

          0  redo size

      42237  bytes sent via SQL\*Net to client

       1235  bytes received via SQL\*Net from client

         67  SQL\*Net roundtrips to/from client

        501  sorts (memory)

          0  sorts (disk)

        984  rows processed

然后是将索引放在内存中的sde\_index\_mem:

SQL> select dataid from sde\_index\_mem where sde.st\_intersects(shape,sde.st\_geometry('POLYGON((90 30,93 30,93 33,90 33,90 30))',2))=1;

已选择984行。

已用时间:  00: 00: 08.18

统计信息

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

       2832  recursive calls

        150  db block gets

       5024  consistent gets

       1665  physical reads

          0  redo size

      42237  bytes sent via SQL\*Net to client

       1235  bytes received via SQL\*Net from client

         67  SQL\*Net roundtrips to/from client

         18  sorts (memory)

          0  sorts (disk)

        984  rows processed

最后是主表及索引表均放入内存的sde\_all\_mem:

SQL> select dataid from sde\_all\_mem where sde.st\_intersects(shape,sde.st\_geometry('POLYGON((90 30,93 30,93 33,90 33,90 30))',2))=1;

已选择984行。

已用时间:  00: 00: 06.98

统计信息

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

       3650  recursive calls

        150  db block gets

       5242  consistent gets

       1612  physical reads

          0  redo size

      42237  bytes sent via SQL\*Net to client

       1235  bytes received via SQL\*Net from client

         67  SQL\*Net roundtrips to/from client

         28  sorts (memory)

          0  sorts (disk)

        984  rows processed

主要看“已用时间”和“physical reads”项，前者为总用时，后者为读取硬盘的次数

发现在时间使用上和硬盘读取次数，sde\_ori>sde\_index\_mem>sde\_all\_mem,但差别不大，与预期不一样，按实验前的想法，加载在内存中的图层应该会大大加快，而且sde\_all\_mem应该基本不需要读取硬盘才对

**2.以相反的顺序重新查询**

清空缓存后，重新进行实验，将查询的顺序倒过来（篇幅原因，结果就不贴了）发现结果也反过来了，时间使用上和硬盘读取次数上，都变成了sde\_ori<sde\_index\_mem<sde\_all\_mem，说明之前一次实验结果的原因为，前面进行的查询将一些公用的内容放入了缓存中，导致了后面结果的变快，跟是否KEEP没有关系。

**3.不清空缓存的前提下继续查询**

继续实验，在不清空缓存的情况下，再运行一下对三个图层的查询语句，这时，三条语句都执行得非常快，基本都在半秒内，均如下（结果类似，这里只举一条例子）：

已选择984行。

已用时间:  00: 00: 01.40

统计信息

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

       1184  recursive calls

        150  db block gets

       2087  consistent gets

          0  physical reads

          0  redo size

      42237  bytes sent via SQL\*Net to client

       1235  bytes received via SQL\*Net from client

         67  SQL\*Net roundtrips to/from client

          0  sorts (memory)

          0  sorts (disk)

        984  rows processed

可以看到，使用时间大大减少，而且均没有读取硬盘，这表明，设置为Keep方式加载到内存的表，也需要先进行一次读取，才会在之后加载到内存，对于普通的表，在读取时也会加载到内存，只是加载的位置不同，一个在KEEP POOL，一个在DEFAULT POOL

**4.模拟大量使用之后**

再进行实验，通过全表搜索某个大表

，将原来缓存里的内容挤掉，再进行查询，

首先查原表，sde\_ori

  COUNT(\*)

----------

       984

已用时间:  00: 00: 04.37

统计信息

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

       1837  recursive calls

        150  db block gets

       3826  consistent gets

       1378  physical reads

          0  redo size

        528  bytes sent via SQL\*Net to client

        519  bytes received via SQL\*Net from client

          2  SQL\*Net roundtrips to/from client

         15  sorts (memory)

          0  sorts (disk)

          1  rows processed

然后查索引放在内存Keep pool中的sde\_index\_mem

  COUNT(\*)

----------

       984

已用时间:  00: 00: 04.87

统计信息

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

       1559  recursive calls

        150  db block gets

       3876  consistent gets

       1181  physical reads

          0  redo size

        528  bytes sent via SQL\*Net to client

        519  bytes received via SQL\*Net from client

          2  SQL\*Net roundtrips to/from client

          5  sorts (memory)

          0  sorts (disk)

          1  rows processed

最后查全表放在内存Keep pool中的sde\_all\_mem

  COUNT(\*)

----------

       984

已用时间:  00: 00: 00.20

统计信息

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

       1184  recursive calls

        150  db block gets

       1132  consistent gets

          5  physical reads

          0  redo size

        528  bytes sent via SQL\*Net to client

        519  bytes received via SQL\*Net from client

          2  SQL\*Net roundtrips to/from client

          0  sorts (memory)

          0  sorts (disk)

          1  rows processed

可以看到，主表没有放入Keep Pool的两个表，数据已经被从缓存中挤出去了，再次查询时耗时非常大，也有非常多的硬盘读取。全表放入内存Keep Pool的sde\_all\_mem则保持了非常高速度以及非常少的硬盘读取。说明影响空间查询的主要还是主表，索引表是否放入内存影响不大，同时，放入Keep Pool的数据，不会因为其它的查询而被挤出内存。

**推论：**

1：对于ArcSDE，在进行空间查询时，空间表是否在内存中，会对空间查询的速度产生非常大的影响；

2：对于ArcSDE，在进行空间查询时，索引表是否在内存中对查询速度影响不大，主要还是取决于主表是否在内存中

3：不管是否设置为Keep，在没有进行过查询时，都不会加载到内存中

4：不管是否设置为Keep，在进行完查询后，都会加载到内存中（不考虑因为内存大小关系加载不完的情况），只是加载的位置不一样

5：设置为Keep的表，不会因为其它查询而被从内存中清除，而没有设置成Keep的表则会。下一次查询时，没有设置成Keep的表还需要从硬盘加载，速度变慢

6：所以，因为Oracle的内存机制，将表设置为Keep不会对查询绝对速度有明显的提升，只是会减少其从硬盘加载到内存的次数，对于使用不太频繁的应用，从用户感觉上会有一定的提升。对于某些特定的应用环境，会有一定的使用价值。