# ORACLE 中如何把表放入到内存

有时一些基础表需要非常的频繁访问，尤其是在一些循环中，对该表的访问速度将变的非常重要。为了提高系统的处理性能，可以考虑将一些表及索引读取并保持到内存中。具体的方法如下：

## 1、修改几个系统参数

LOCK_SGA：系统默认是 FALSE。它指定整个 SGA 区是否锁定在真正的物理内存中。要首先修改这个参数为 TRUE。注意这个参数有的操作系统是不支持的，好像 windows 就不行。

下面还要了解一下具体和 CACHE 有关的几个概念，即 DB_CACHE 中的几个 pool：

DB_CACHE_SIZE：指定缺省的 buffer pool 的大小，以字节为单位。

DB_KEEP_CACHE_SIZE： 指定 keep buffer pool 的大小，以字节为单位。

DB_RECYCLE_CACHE_SIZE： 指定 recycle buffer pool 的大小，以字节为单位。

**Keep Buffer Pool**

其的作用是缓存那些需要经常查询的对象但又容易被默认缓冲区置换出去的对象，按惯例，Keep pool 设置为合理的大小，以使其中存储的对象不再 age out，也就是查询这个对象的操作不会引起磁盘 IO 操作，可以极大地提高查询性能。

默认的情况下 db_keep_cache_size=0，未启用，如果想要启用，需要手工设置 db_keep_cache_size 的值，设置了这个值之后 db_cache_size 会减少。

并不是我们设置了 keep pool 之后，热点表就一定能够缓存在 keep pool ，keep pool 同样也是由 LRU 链表管理的，当 keep pool 不够的时候，最先缓存到 keep pool 的对象会被挤出，不过与 default pool 中的 LRU 的管理方式不同，在 keep pool 中表永远是从 MRU 移动到 LRU，不会由于你做了 FTS 而将表缓存到 LRU 端，在 keep pool 中对象永远是先进先出。

**Recycle Buffer Pool**

Recycle Buffer Pool 正好相反。Recycle Buffer Pool 用于存储临时使用的、不被经常使用的较大的对象，这些对象放置在 Default Buffer Pool 显然是不合适的，这些块会导致过量的缓冲区刷新输出，而且不会带来任何好处，因为等你想要再用这个块时，它可已经老化退出了缓存。要把这些段与默认池和保持池中的段分开，这样就不会导致默认池和保持池中的块老化而退出缓存。

```
--查看 keep pool 剩余大小
SQL> select p.name,a.cnum_repl "total buffers",a.anum_repl "free buffers"
from x$kcbwds a, v$buffer_pool p
where a.set_id=p.LO_SETID and p.name='KEEP';
```

| NAME | total buffers | free buffers |
| --- | --- | --- |
| KEEP | 1984 | 1984 |

2、修改表或索引的结构

要把表 cache 到内存中，要为表或索引指定 buffer pool 的类型。具体语句如下：

--修改数据库表的存储属性 pool

alter table xxx storage(buffer_pool keep);

又如：

CREATE INDEX cust_idx …STORAGE (BUFFER_POOL KEEP );

ALTER TABLE customer STORAGE (BUFFER_POOL KEEP);

ALTER INDEX cust_name_idx STORAGE (BUFFER_POOL KEEP);

同时要修改表的 cache 属性

Alter table xxx cache;

也可以在表创建时直接指定相应的属性：

create table aaa(i integer) storage(buffer_pool keep);

create table bbb(i integer) storage(buffer_pool keep) cache;

观看表的 cache 情况及大小：

select table_name,cache,blocks from user_tables where buffer_pool='KEEP';

3、进行全表扫描，将表移入内存

可以使用 ANALYZE table xxx　ESTIMATE STATISTICS 分析表，使表读入到 keep　poolz 中。也可以使用其它的使表全表扫描的语句达到相同的目的。如 select * from xxx;

这样就可以放到 pool 里面。

要想检验是否已经 cache 到 pool 中，可以打开执行计划，看具体的物理读的次数，若已经 cache，则物理读为0。

Set autotrace on

Select count(*) from test1;

执行计划如下：

---

0　recursive calls

0　db block gets

2501 consistent gets

**0　physical reads**

0　redo size

```
5200  bytes sent via SQL*Net to client
 596  bytes received via SQL*Net from client
   2  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
   1  rows processed
```

对于 storage 到 keep pool 中的表，第一次会直接 physical reads 到 keep pool 中，下次就直接从 keep pool 中读了。但 flush buffer_cache 会清空 keep pool。

## 附录：与 cache 到内存相关的命令

```
--表缓存
alter table ..... storage(buffer_pool keep);
--查看哪些表被放在缓存区 但并不意味着该表已经被缓存
select table_name from dba_tables where buffer_pool='keep';
--查询到该表是否已经被缓存
select table_name,cache,buffer_pool from user_TABLES where cache like '%Y';
--已经加入到 KEEP 区的表想要移出缓存，使用
alter table table_name nocache;
--查询当前用户下表的情况
select table_name,cache,buffer_pool from user_TABLES;
--对于普通 LOB 类型的 segment 的 cache 方法
alter table t2 modify lob(c2) (storage (buffer_pool keep) cache);
--取消缓存
alter table test modify lob(address) (storage (buffer_pool keep) nocache);
--查询段
select segment_name,segment_type,buffer_pool from user_segments;
--对基于 CLOB 类型的对象的 cache 方法
alter table lob1 modify lob(c1.xmldata) (storage (buffer_pool keep) cache);
  --查询该用户下所有表内的大字段情况
select column_name,segment_name from user_lobs;
```