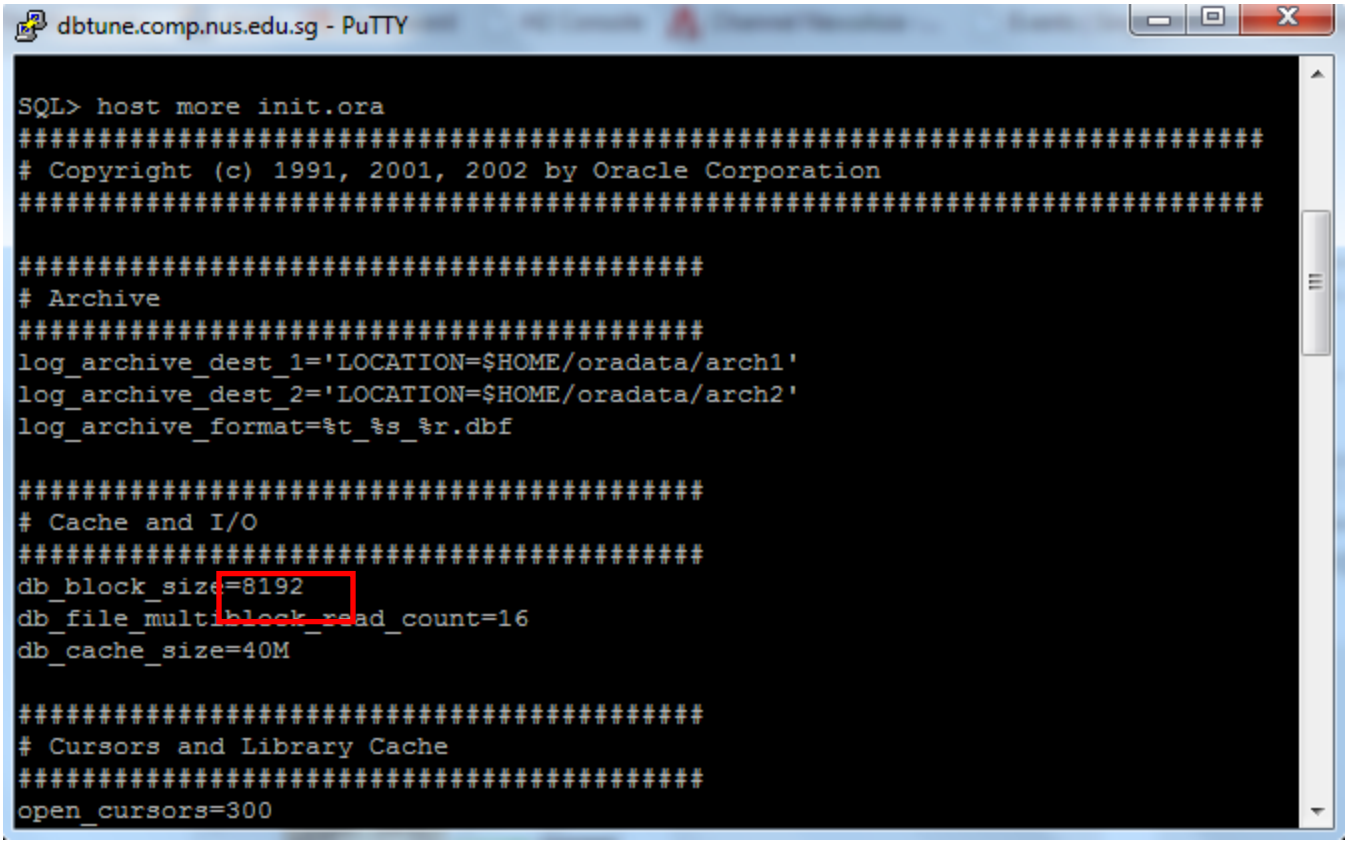
Database Tuning
CS5226

A0079953L – LEE ZHONG DE ROLLEI

TABLE OF CONTENT

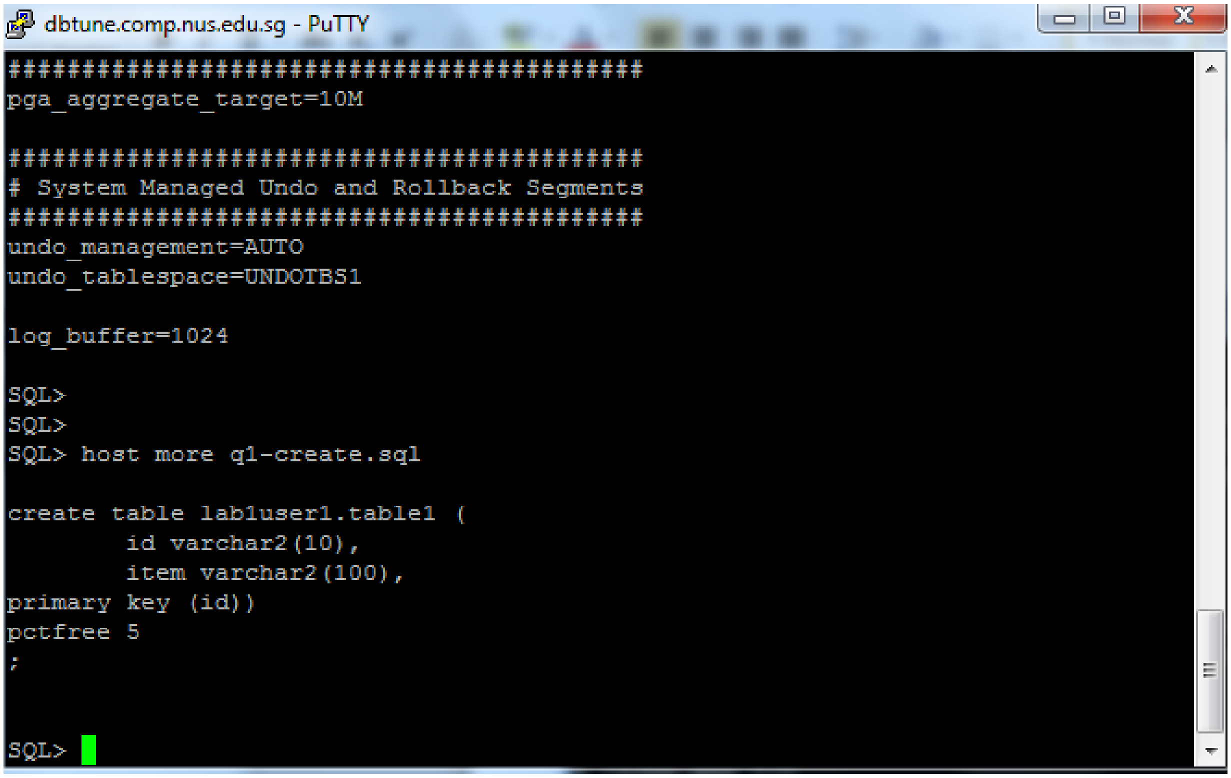
Question 1: Block Packing Factor (PCTFree)

The init.ora has a Db_block_size = 8192. DB_BLOCK_SIZE specifies (in bytes) the size of Oracle database blocks used for each datarow in the datatable.



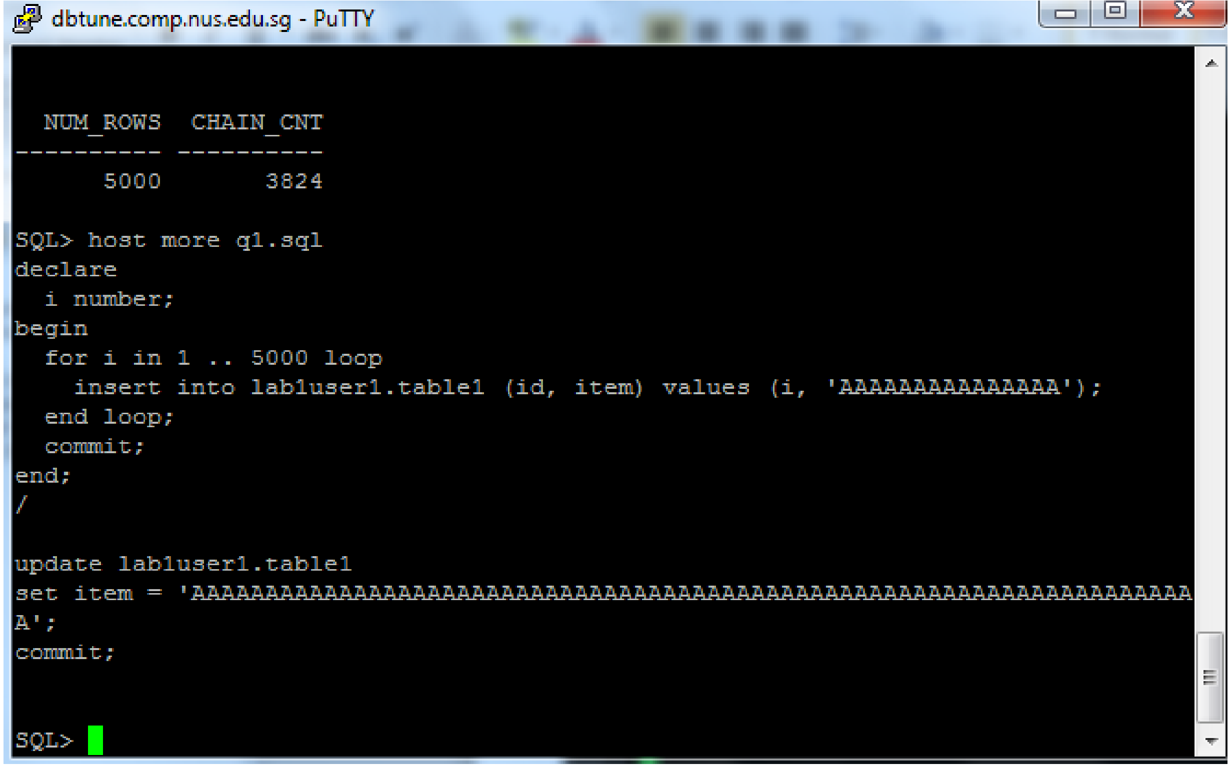
```
dbtune.comp.nus.edu.sg - PuTTY

SQL> host more init.ora
##############################################################################
# Copyright (c) 1991, 2001, 2002 by Oracle Corporation
##############################################################################

###########################################
# Archive
###########################################
log_archive_dest_1='LOCATION=$HOME/oradata/arch1'
log_archive_dest_2='LOCATION=$HOME/oradata/arch2'
log_archive_format=%t_%s_%r.dbf

###########################################
# Cache and I/O
###########################################
db_block_size=8192
db_file_multiblock_read_count=16
db_cache_size=40M

###########################################
# Cursors and Library Cache
###########################################
open_cursors=300
```

q1-create.sql (as below) creates a table (table1) with PCTFREE parameter set to 5. PCTFREE parameter specifies the percentage of space in each data block that is reserved for future update of existing data.



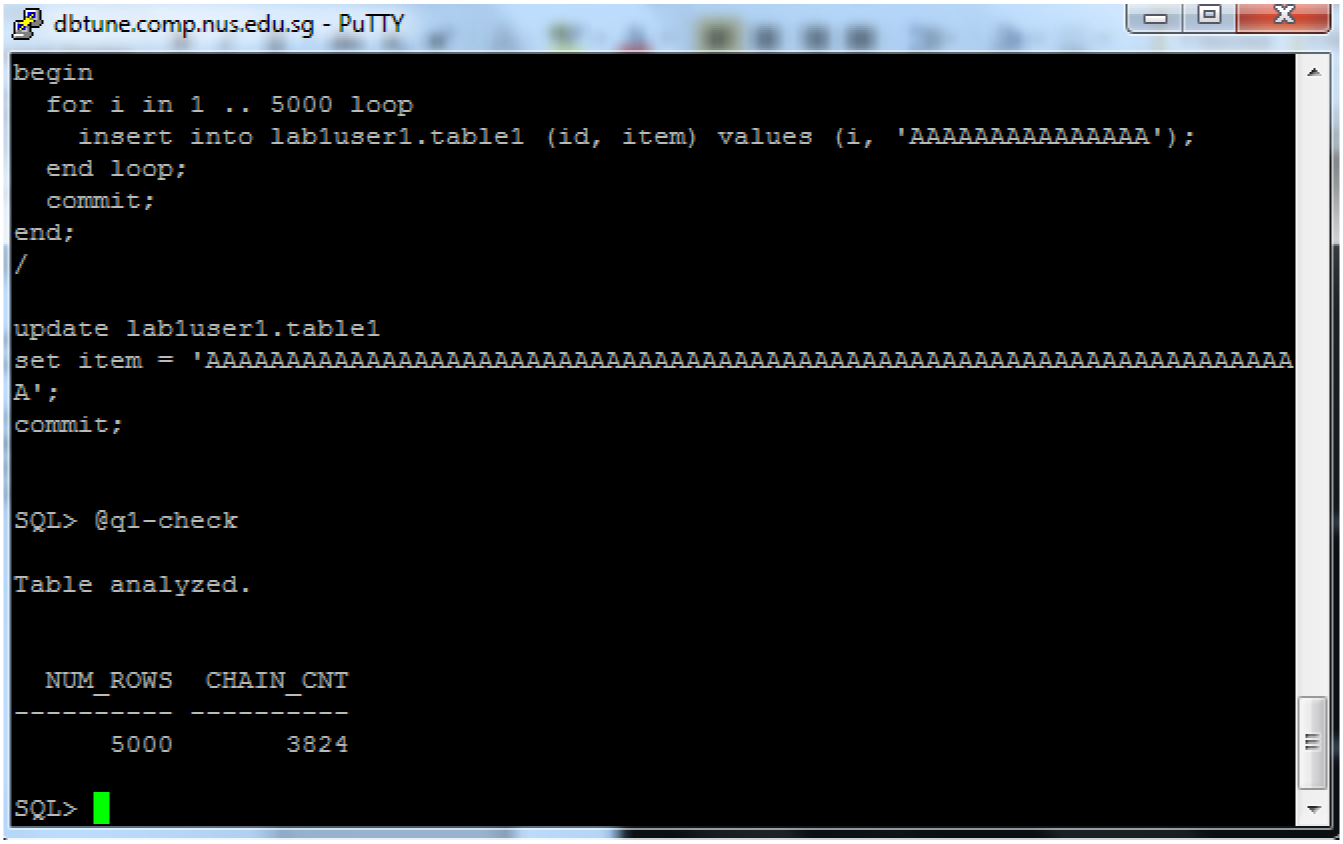
```
###########################################
pga_aggregate_target=10M

###########################################
# System Managed Undo and Rollback Segments
###########################################
undo_management=AUTO
undo_tablespace=UNDOTBS1

log_buffer=1024

SQL>
SQL>
SQL> host more q1-create.sql

create table lab1user1.table1 (
        id varchar2(10),
        item varchar2(100),
primary key (id))
pctfree 5
;


SQL>
```

q1.sql insert 5000 rows to table1 and perform and subsequently update column “item” in table 1



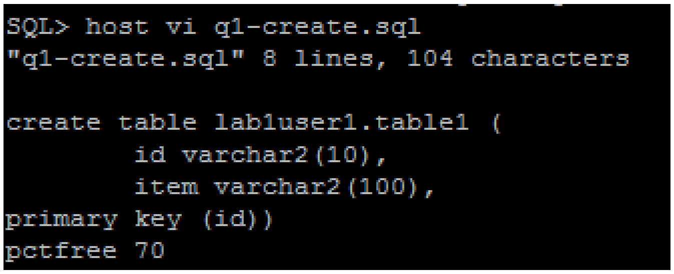
```


  NUM_ROWS  CHAIN_CNT
---------- ----------
      5000       3824

SQL> host more q1.sql
declare
  i number;
begin
  for i in 1 .. 5000 loop
    insert into lab1user1.table1 (id, item) values (i, 'AAAAAAAAAAAAAAA');
  end loop;
  commit;
end;
/

update lab1user1.table1
set item = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A';
commit;


SQL>
```

q1-check.sql shown that the update from q1.sql has resulted in <u>3824</u> chained rows.



```
begin
  for i in 1 .. 5000 loop
    insert into lab1user1.table1 (id, item) values (i, 'AAAAAAAAAAAAAAA');
  end loop;
  commit;
end;
/

update lab1user1.table1
set item = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A';
commit;


SQL> @q1-check

Table analyzed.


  NUM_ROWS  CHAIN_CNT
---------- ----------
      5000       3824

SQL>
```

To avoid chained records, we drop and re-create table1 with PCTFree set to 70%



```
SQL> host vi q1-create.sql
"q1-create.sql" 8 lines, 104 characters

create table lab1user1.table1 (
        id varchar2(10),
        item varchar2(100),
primary key (id))
pctfree 70
```

Proceed to execute q1-create.sql, q1.sql and q1-check.sql. After changing the PCTFree parameter from 5% to 70%, the update occurred in q1.sql did not consequently create any chain rows

```
SQL> drop table lab1user1.table1
  2  ;

Table dropped.

SQL> @q1-create

Table created.

SQL> @q1

PL/SQL procedure successfully completed.


5000 rows updated.


Commit complete.

SQL> @q1-check

Table analyzed.


  NUM_ROWS  CHAIN_CNT
---------- ----------
      5000          0

SQL>
```

Question 2: Table Fragmentation

- Execute q2.sql to create table 2
- Execute q2-analyze.sql to analyze the chained rows in table
- Count the number of chained_rows in table 2

dbtune.comp.nus.edu.sg - PuTTY

```
SQL>
SQL> @q2

Table created.


PL/SQL procedure successfully completed.


5000 rows updated.


Commit complete.

SQL> @q2-analyze

Table created.


Table analyzed.

SQL> SELECT COUNT(1) FROM CHAINED_ROWS WHERE head_rowid in (SELECT rowid FROM la
b1user1.table2);

  COUNT(1)
----------
      4164

SQL>
```

To eliminate the chain rows, a temporary table is created to store the chained rows in table2. The chained rows were then deleted from table2. Finally, the chained rows in the temporary table were transferred back. This will consequently eliminate the chained_rows in table2 while preserving the original data.

dbtune.comp.nus.edu.sg - PuTTY

```
SQL> create global temporary table temp as SELECT * FROM lab1user1.table2  where
 rownum=0;

Table created.

SQL> INSERT INTO temp SELECT * FROM lab1user1.table2  where rowid in (select hea
d_rowid from chained_rows);

4164 rows created.

SQL> delete from lab1user1.table2 where rowid in (select head_rowid from chained
_rows);

4164 rows deleted.

SQL> insert into lab1user1.table2 select * from temp;

4164 rows created.

SQL> delete from chained_rows;

4164 rows deleted.

SQL> analyze table lab1user1.table2 list chained rows;

Table analyzed.

SQL> select count(1) from chained_rows;

  COUNT(1)
----------
         0
```

Question 3: Index Reorganization

The index created has a ratio (number of deleted entries to the number of current entries) of 26% as shown below. As such, it should be re-organized to achieve a ratio less than 20%.

```
dbtune.comp.nus.edu.sg - PuTTY
SQL> drop table lab1user1.table3;

Table dropped.

SQL> @q3

Table created.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL> analyze index lab1user1.table3_pk validate structure;

Index analyzed.

SQL> SELECT DEL_LF_ROWS, LF_ROWS, DEL_LF_ROWS/ LF_ROWS AS Ratio FROM INDEX_STATS
;

DEL_LF_ROWS    LF_ROWS      RATIO
----------- ---------- ----------
       1799       6799 .264597735
```

After rebuilding the index, the ratio has not bee reduced to 0%

```
SQL> alter index lab1user1.table3_pk rebuild;

Index altered.

SQL> analyze index lab1user1.table3_pk validate structure;

Index analyzed.

SQL> SELECT DEL_LF_ROWS, LF_ROWS, DEL_LF_ROWS/ LF_ROWS AS Ratio FROM INDEX_STATS
  2  ;

DEL_LF_ROWS    LF_ROWS      RATIO
----------- ---------- ----------
          0       5000          0
```