

CSCI317 Database Performance Tuning

Simple Performance Measurement Tools

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Simple Performance Measurement Tools

Outline

Simple measurement of response time

Using **TIMING** option of SQLcl

Using **V\$** and **GV\$** dynamic performance views

Simple measurement of response time

Find the total number of different values in **L_TAX** column

```
VARIABLE START_TIME VARCHAR2(20);  
VARIABLE STOP_TIME VARCHAR2
```

[SQL](#)

```
BEGIN  
  SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY:HH:MI:SS')  
    INTO :START_TIME  
  FROM DUAL;  
END;  
/
```

[SQL](#)

```
SELECT COUNT(DISTINCT L_TAX)  
FROM LINEITEM;
```

[SQL](#)

```
BEGIN  
  SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY:HH:MI:SS')  
    INTO :STOP_TIME  
  FROM DUAL;  
END;  
/
```

[SQL](#)[TOP](#)

Created by Janusz R. Getta, CSCI317 Database Performance Tuning, SIM, Session 3, 2020

3/13

Simple measurement of response time

Find the total number of different values in **L_TAX** column

```
BEGIN
  SELECT (TO_DATE(:STOP_TIME, 'DD-MON-YYYY:HH:MI:SS') -
          TO_DATE(:START_TIME, 'DD-MON-YYYY:HH:MI:SS')) * 24 * 60 * 60
  INTO :TOTAL_TIME
  FROM DUAL;
END;
/
```

SQL

```
PRINT :START_TIME;
PRINT :STOP_TIME;
PRINT :TOTAL_TIME;
```

SQL

Simple measurement of response time

Find the total number of different values in **L_TAX** column

```
VARIABLE START_TIME VARCHAR2(20);  
VARIABLE STOP_TIME VARCHAR2(20);  
VARIABLE TOTAL_TIME NUMBER;
```

[SQL](#)

```
BEGIN  
  SELECT TO_CHAR(SYSTIMESTAMP, 'DD-MON-YYYY:HH24:MI:SS:FF')  
  INTO :START_TIME  
  FROM DUAL;  
END;  
/
```

[SQL](#)

```
SELECT COUNT(DISTINCT L_TAX)  
FROM LINEITEM;
```

[SQL](#)

```
BEGIN  
  SELECT TO_CHAR(SYSTIMESTAMP, 'DD-MON-YYYY:HH24:MI:SS:FF')  
  INTO :STOP_TIME  
  FROM DUAL;  
END;  
/
```

[SQL](#)[TOP](#)

Created by Janusz R. Getta, CSCI317 Database Performance Tuning, SIM, Session 3, 2020

5/13

Simple measurement of response time

Find the total number of different values in **L_TAX** column

```
BEGIN
  SELECT EXTRACT(SECOND FROM
    (TO_TIMESTAMP(:STOP_TIME, 'DD-MON-YYYY:HH24:MI:SS:FF') -
     TO_TIMESTAMP(:START_TIME, 'DD-MON-YYYY:HH24:MI:SS:FF'))
  INTO :TOTAL_TIME
  FROM DUAL;
END;
/
```

SQL

```
PRINT :START_TIME;
PRINT :STOP_TIME;
PRINT :TOTAL_TIME;
```

SQL

Simple Performance Measurement Tools

Outline

Simple measurement of response time

Using `TIMING` option of `SQLcl`

Using `V$` and `GV$` dynamic performance views

Using **TIMING** option of SQLcl

Find the total number of different values in **L_TAX** column

```
SET TIMING ON
```

```
SQL
```

```
SELECT COUNT(DISTINCT L_TAX)  
FROM LINEITEM;
```

```
SQL
```

```
COUNT(DISTINCT L_TAX)
```

```
SQL
```

```
-----
```

```
9
```

```
Elapsed: 00:00:02.69
```

```
SET TIMING OFF
```

```
SQL
```

Simple Performance Measurement Tools

Outline

Simple measurement of response time

Using **TIMING** option of SQLcl

Using v\$ and gv\$ dynamic performance views

TOP

Created by Janusz R. Getta, CSCI317 Database Performance Tuning, SIM, Session 3, 2020

9/13

Using **V\$**, **GV\$** dynamic performance views

V\$ and **GV\$** data dictionary views are called as "dynamic performance views" and "global dynamic performance views"

Dynamic performance views contain information about the most current database activities

For example a view **V\$INSTANCE** includes instance name, host name, start-up time, etc.

A view **V\$DATABASE** includes database name, date when created, current operating system, etc.

A view **V\$SESSION** includes active connections, login times, current state of transaction, etc.

A view **V\$SEGEMENT_STATITICS** includes total number of read/write operations performed on all segments, etc.

Using **V\$**, **GV\$** dynamic performance views

Find the total number of read block operations needed to compute a query that finds the values in **L_TAX** column

```
CONNECT SYSTEM
```

Connecting as a user SYSTEM

```
CREATE TABLE OPSTAT_TABLE(  
  OWNER          VARCHAR(30) NOT NULL,  
  OBJECT_NAME     VARCHAR(30) NOT NULL,  
  STATISTIC_NAME  VARCHAR(64) NOT NULL,  
  VALUE          NUMBER NOT NULL,  
  TS             TIMESTAMP NOT NULL,  
  CONSTRAINT OPSTAT_TABLE_PKEY PRIMARY KEY(TS,OWNER,OBJECT_NAME,STATISTIC_NAME) );
```

Creating table for statistics

```
INSERT INTO OPSTAT_TABLE (  
  SELECT OWNER,  
         OBJECT_NAME,  
         STATISTIC_NAME,  
         VALUE,  
         SYSTIMESTAMP  
  FROM   V$SEGMENT_STATISTICS  
  WHERE  OWNER = 'CSCI317' AND  
         STATISTIC_NAME IN ('physical reads', 'logical reads') );  
COMMIT;
```

Saving the current values of statistics

TOP

Created by Janusz R. Getta, CSCI317 Database Performance Tuning, SIM, Session 3, 2020

11/13

Using **V\$**, **GV\$** dynamic performance views

Find the total number of read block operations needed to compute a query that finds the values in **L_TAX** column

```
CONNECT CSCI317
```

Connecting as a user CSCI317

```
SELECT L_TAX  
FROM LINEITEM;
```

Processing SELECT statement

```
CONNECT SYSTEM
```

Connecting as a user SYSTEM

Finding the differences between the past and present statistics

```
SELECT V$SEGMENT_STATISTICS.OWNER,  
       V$SEGMENT_STATISTICS.OBJECT_NAME,  
       V$SEGMENT_STATISTICS.STATISTIC_NAME,  
       V$SEGMENT_STATISTICS.VALUE - NVL(OPSTAT_TABLE.VALUE,0) INCR  
FROM   V$SEGMENT_STATISTICS LEFT OUTER JOIN OPSTAT_TABLE  
       ON V$SEGMENT_STATISTICS.OWNER = OPSTAT_TABLE.OWNER AND  
          V$SEGMENT_STATISTICS.OBJECT_NAME = OPSTAT_TABLE.OBJECT_NAME AND  
          V$SEGMENT_STATISTICS.STATISTIC_NAME = OPSTAT_TABLE.STATISTIC_NAME  
WHERE  V$SEGMENT_STATISTICS.OWNER = 'CSCI317' AND  
       ( V$SEGMENT_STATISTICS.VALUE - NVL(OPSTAT_TABLE.VALUE,0) ) <> 0 AND  
       V$SEGMENT_STATISTICS.OBJECT_NAME <> 'PLAN_TABLE' AND  
       V$SEGMENT_STATISTICS.STATISTIC_NAME IN ('physical reads', 'logical reads')  
ORDER BY V$SEGMENT_STATISTICS.OBJECT_NAME, V$SEGMENT_STATISTICS.STATISTIC_NAME;
```

TOP

Created by Janusz R. Getta, CSCI317 Database Performance Tuning, SIM, Session 3, 2020

12/13

References

G. Harrison Oracle Performance Survival Guide, Prentice Hall, 2010

S. R. Alapati, Oracle 12c Performance Tuning Recipes, A Problem-Solution Approach, Apress, 2014