ORACLE 11G SECURE FILES PART 2 – PERFORMANCE IMPROVEMENTS

Inderpal S. Johal, Data Softech Inc.

INTRODUCTION

Oracle 11g has re-architectured the LOB Storage format so as to provide performance improvement, optimized space utilization and high security. Don't think that LOBs types are changed as it is still CLOB, BLOB and BFILE but Oracle has improved the storage and it will discussed in this and coming paper in detail.

COMPARE THE PERFORMANCE OF BASICFILE VS SECUREFILE LOBS

In order to check the performance of two types of LOBs format, we will perform the following

1. Create the Demo Tablespace and User
2. Create the Demo Tables using Basicfile and Securefile Lobs
3. Load the same data into both tables and check the data loading speed
4. Retrieve the data from both Table and make a note of the speed
5. Compare the performance

Step 1 : Create the Demo User with privileges, Demo Tablespaces and define directory which will hold documents to be loaded to demo LOB table

Create Demo user SF_DEMO

```
SQL> CREATE USER sf_demo IDENTIFIED BY oracle
  2  DEFAULT TABLESPACE users
  3  TEMPORARY TABLESPACE temp
  4  QUOTA UNLIMITED ON users
  5  /
User created.

SQL> GRANT connect, resource TO sf_demo;
Grant succeeded.

SQL> GRANT EXECUTE ANY PROCEDURE, CREATE ANY DIRECTORY TO sf_demo;
Grant succeeded.
```

Create two Demo Tablespace for LOB table and make sure it is ASSM as required by SECUREFILE LOBs

```
SQL> CREATE TABLESPACE tbs1
  2  DATAFILE '/home/oracle/app/oradata/11gtest/tbs1.dbf' SIZE 150M REUSE
  3  EXTENT MANAGEMENT LOCAL
  4  UNIFORM SIZE 1M
  5  SEGMENT SPACE MANAGEMENT AUTO
  6  /
Tablespace created.
```

Define Directory which will contains all Document which will be loaded into LOBs tables

```
SQL> CREATE OR REPLACE DIRECTORY cwd AS '/home/oracle/resumes';
Directory created.
```

Grant privileges to Demo User to read and write to this directory

```
SQL> GRANT READ,WRITE ON DIRECTORY cwd TO sf_demo;
Grant succeeded.
```

Step 2 : Create the Demo Tables which will contain BASICFILE Lob column

```
SQL> connect sf_demo/oracle
Connected.

Create Demo Table under SF_DEMO schema containing BASICFILE LOB storage column and using Tablespace TBS1
SQL> CREATE TABLE resumes
  2  (id NUMBER, first_name VARCHAR2(15),
  3  last_name VARCHAR2(40), resume BLOB)
  4  LOB(resume) STORE AS BASICFILE
  5  (TABLESPACE tbs1)
  6  /
Table created.
```

Step 3 : Now we will create two procedure to populate the tables created in Step 2

1. LoadLobFromFile – This procedure Load with Word Doc Resume into the LOB column
2. Write_LOB – This will Insert the LOB data into the Demo table

```
SQL> CREATE OR REPLACE PROCEDURE loadLOBFromBFILE_proc (dest_loc IN OUT BLOB, file_name IN VARCHAR2)
  2  IS
  3     src_loc        BFILE := BFILENAME('CWD', file_name);
  4     amount         INTEGER := 4000;
  5  BEGIN
  6
  7     DBMS_LOB.OPEN(src_loc, DBMS_LOB.LOB_READONLY);
  8     amount := DBMS_LOB.GETLENGTH(src_loc);
  9     DBMS_LOB.LOADFROMFILE(dest_loc, src_loc, amount);
 10     DBMS_LOB.CLOSE(src_loc);
 11
 12  END;
 13  /
Procedure created.
```

```
SQL> CREATE OR REPLACE PROCEDURE write_lob (p_file IN VARCHAR2)
  2  IS
  3   i  NUMBER;
  4   fn VARCHAR2(15);
  5   ln VARCHAR2(40);
  6   b  BLOB;
  7
  8  BEGIN
  9  DBMS_OUTPUT.ENABLE;
 10  DBMS_OUTPUT.PUT_LINE('Begin inserting rows...');
 11    FOR i IN 1 .. 30 LOOP
 12      fn:=SUBSTR(p_file,1,INSTR(p_file,'.')-1);
 13      ln:=SUBSTR(p_file,INSTR(p_file,'.')+1,LENGTH(p_file)-INSTR(p_file,'.')-4);
 14      INSERT INTO resumes VALUES (i, fn, ln, EMPTY_BLOB())
 15        RETURNING resume INTO b;
 16      loadLOBFromBFILE_proc(b,p_file);
 17      DBMS_OUTPUT.PUT_LINE('Row '|| i ||' inserted.');
 18    END LOOP;
 19    COMMIT;
 20  END;
 21  /
Procedure created.
```

Step 4 : Now we will Load the Data into the BASICFILE LOB Demo Table

```
Check the file available in Directory specified in Step 1 as cwd and pointing to /home/oracle/resumes
$ pwd
/home/oracle/resumes
$ ls -ltr *.doc
-rw-r--r--  1 oracle oinstall 61440 Aug 25 18:04 indy1.doc
-rw-r--r--  1 oracle oinstall 36352 Aug 25 18:23 indy5.doc
-rw-r--r--  1 oracle oinstall 57856 Aug 25 18:23 indy4.doc
-rw-r--r--  1 oracle oinstall 54272 Aug 25 18:23 indy3.doc
-rw-r--r--  1 oracle oinstall 62464 Aug 25 18:23 indy2.doc

SQL> connect sf_demo/oracle
Connected.

Start the Time to Make a note of Time spent to Load the Data into BASICFILE Lobs Table. Execute the Procedure
WRITE_LOB 5 times to load 5 different RESUME
SQL> timing start load_data
SQL> exec write_lob('indy1.doc');
PL/SQL procedure successfully completed.

SQL> exec write_lob('indy2.doc');
PL/SQL procedure successfully completed.

SQL> exec write_lob('indy3.doc');
PL/SQL procedure successfully completed.

SQL> exec write_lob('indy4.doc');
PL/SQL procedure successfully completed.

SQL> exec write_lob('indy5.doc');
PL/SQL procedure successfully completed.

SQL> timing stop
```

So we will note the time to Load 5 Document file.

timing for: load_data

Elapsed: 00:00:00.36

Step 5 : Now we will retrieve the Data from BASICFILE Lob Demo Table. This will be done with the help of procedure shown below called READ_LOB

```
SQL> CREATE OR REPLACE PROCEDURE read_lob
  2  IS
  3       lob_loc         BLOB;
  4       CURSOR resumes_cur IS
  5         SELECT id, first_name, last_name, resume
  6         FROM resumes;
  7       resumes_rec     resumes%ROWTYPE;
  8    BEGIN
  9    OPEN resumes_cur;
 10    LOOP
 11      FETCH resumes_cur INTO resumes_rec;
 12       lob_loc := resumes_rec.resume;
 13       DBMS_OUTPUT.PUT_LINE('The length is: '|| DBMS_LOB.GETLENGTH(lob_loc));
 14       DBMS_OUTPUT.PUT_LINE('The ID is: '|| resumes_rec.id);
 15       -- just print out the first 200 bytes of the LOB
 16       -- because DBMS_OUTPUT.PUT_LINE cannot display more than 255 bytes
 17       DBMS_OUTPUT.PUT_LINE('The blob is read: '||
 18              UTL_RAW.CAST_TO_VARCHAR2(DBMS_LOB.SUBSTR(lob_loc,200,1)));
 19     EXIT WHEN resumes_cur%NOTFOUND;
 20    END LOOP;
 21   CLOSE resumes_cur;
 22  END;
 23  /

Procedure created.
```

Step 6 : Execute READ_LOB procedure to find out the time spent in retrieving the Data from BASICFILE Lob Demo table

```
SQL>timing start read_data
SQL> exec read_lob;
PL/SQL procedure successfully completed.

SQL> timing stop
```

So we will note the time to Read data from RESUME table.
timing for: read_data
Elapsed: 00:00:01.25

So here is Initial BASIC FILE CHART

Test Name	BASICFILE Lobs Table	SECUREFILE Lobs Table
Load the LOB Data	00:00:00.36	
Read the LOB Data	00:00:01.25	

Step 7 : Now we will perform Step 2-6 again but now against SECUREFILE Lobs Demo tables.

```
SQL> connect sf_demo/oracle
Connected.

Create Demo Table under SF_DEMO schema containing SECUREFILE LOB storage column and using Tablespace
TBS1
SQL> CREATE TABLE resumes
  2  (id NUMBER, first_name VARCHAR2(15),
  3  last_name VARCHAR2(40), resume BLOB)
  4  LOB(resume) STORE AS SECUREFILE
  5 (TABLESPACE tbs1
  6  COMPRESS HIGH
  7  DEDUPLICATE )
  6  /
Table created.

Start the Time to Make a note of Time spent to Load the Data into SECUREFILE Lobs Table. Execute the Procedure
WRITE_LOB 5 times to load 5 different RESUME
SQL> timing start load_data
SQL> exec write_lob('indy1.doc');
PL/SQL procedure successfully completed.
SQL> exec write_lob('indy2.doc');
PL/SQL procedure successfully completed.
SQL> exec write_lob('indy3.doc');
PL/SQL procedure successfully completed.
SQL> exec write_lob('indy4.doc');
PL/SQL procedure successfully completed.
SQL> exec write_lob('indy5.doc');
PL/SQL procedure successfully completed.
SQL> timing stop
```

So we will note the time to Load 5 Document file.

timing for: load_data
Elapsed: 00:00:00.23

```
SQL>timing start read_data
SQL> exec read_lob;
PL/SQL procedure successfully completed.

SQL> timing stop
```

So we will note the time to Read data from RESUME table.

timing for: read_data
Elapsed: 00:00:01.13

So here is Initial BASIC FILE CHART

Test Name	BASICFILE Lobs Table	SECUREFILE Lobs Table
Load the LOB Data	00:00:00.36	00:00:00.23
Read the LOB Data	00:00:01.25	00:00:01.13

In the Next paper we will discuss the Storage used in both cases and how you can convert BASICFILE to SECUREFILE Storage