DBA

Audit

Audit documentation

Introduction:

Auditing is the monitoring and recording of selected user database actions. It can be based on individual actions, such as the type of SQL statement executed, or on combinations of factors that can include user name, application, time, and so on. Security policies can trigger auditing when specified elements in an Oracle database are accessed or altered, including the contents within a specified object.

All database audit trail are stored in a table named SYS.AUD$ in the SYS schema. We can see all audited records by querying this table .

```
SELECT * FROM SYS.AUD$;
```

NOTE:(Remember that this table is in SYS schema. Users other than SYS need permission to access this table)

Steps :

1. Show audit parameter

```
SHOW PARAMETER AUDIT;
```

Auditing parameter describes your database server auditing feature.

2. Alter audit_trail

To audit all sql text , the audit_trail parameter must be DB, EXTENDED

The audit_trail parameter can be altered by following command

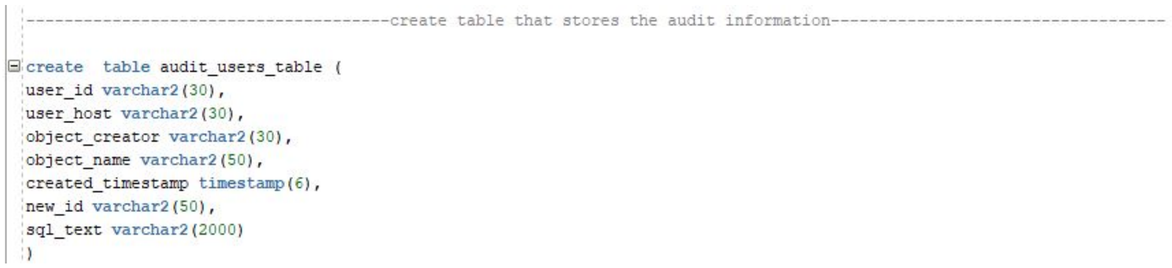
```
ALTER SYSTEM SET AUDIT_TRAIL=DB,EXTENDED SCOPE=SPFILE;
```

After this , the database instance must be restarted.

3. Enable auditing on table

AUDIT ALL ON DEMO_TABLE BY ACCESS;

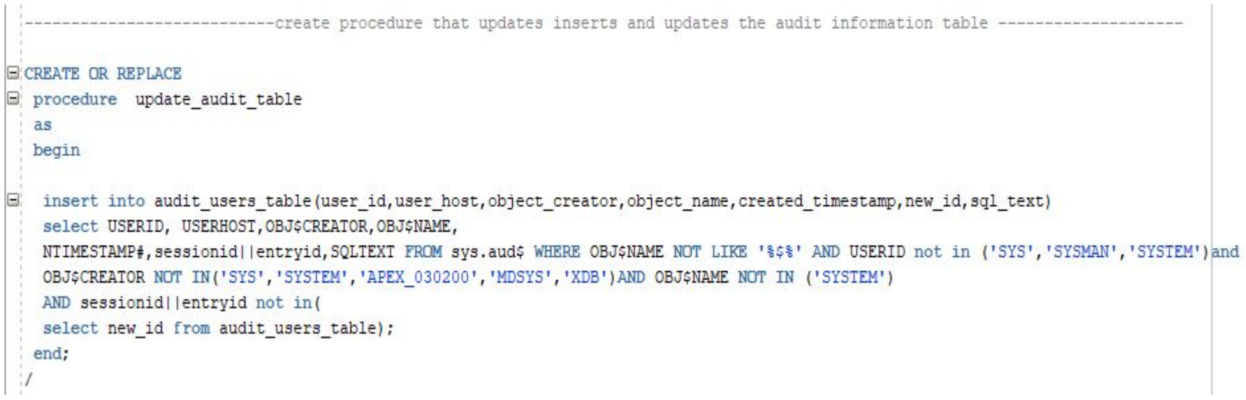
4. Create table where audit information will be stored



```
--------------------------------------create table that stores the audit information-----------------------------------

create  table audit_users_table (
user_id varchar2(30),
user_host varchar2(30),
object_creator varchar2(30),
object_name varchar2(50),
created_timestamp timestamp(6),
new_id varchar2(50),
sql_text varchar2(2000)
)
```

Fig 4.1 : Audit_users_table

5.Create procedure that inserts and updates information into audit table.



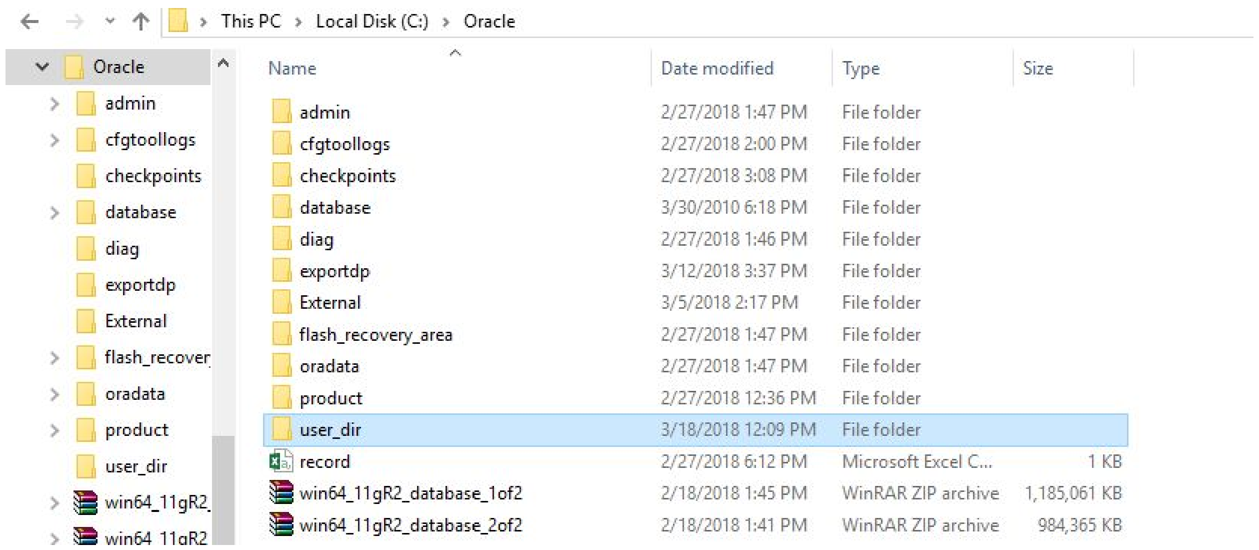
```
---------------------------create procedure that updates inserts and updates the audit information table --------------------

CREATE OR REPLACE
 procedure  update_audit_table
 as
 begin

  insert into audit_users_table(user_id,user_host,object_creator,object_name,created_timestamp,new_id,sql_text)
  select USERID, USERHOST,OBJ$CREATOR,OBJ$NAME,
  NTIMESTAMP#,sessionid||entryid,SQLTEXT FROM sys.aud$ WHERE OBJ$NAME NOT LIKE '%$%' AND USERID not in ('SYS','SYSMAN','SYSTEM')and
  OBJ$CREATOR NOT IN('SYS','SYSTEM','APEX_030200','MDSYS','XDB')AND OBJ$NAME NOT IN ('SYSTEM')
  AND sessionid||entryid not in(
  select new_id from audit_users_table);
 end;
/
```

Fig 5.1 : update_audit_table procedure

This procedure inserts records from SYS.AUD$ to audit_users_table which has not been inserted before. It doesn't inserts the audit information of system users and the objects which are created by system.

6. Create a directory:

We need to create a directory to store all the audit files. For that, we created a directory named '**USER_DIR**' in our local drive '**C:\Oracle**'.


This PC > Local Disk (C:) > Oracle

Name	Date modified	Type	Size
admin	2/27/2018 1:47 PM	File folder	
cfgtoollogs	2/27/2018 2:00 PM	File folder	
checkpoints	2/27/2018 3:08 PM	File folder	
database	3/30/2010 6:18 PM	File folder	
diag	2/27/2018 1:46 PM	File folder	
exportdp	3/12/2018 3:37 PM	File folder	
External	3/5/2018 2:17 PM	File folder	
flash_recovery_area	2/27/2018 1:47 PM	File folder	
oradata	2/27/2018 1:47 PM	File folder	
product	2/27/2018 12:36 PM	File folder	
user_dir	3/18/2018 12:09 PM	File folder	
record	2/27/2018 6:12 PM	Microsoft Excel C...	1 KB
win64_11gR2_database_1of2	2/18/2018 1:45 PM	WinRAR ZIP archive	1,185,061 KB
win64_11gR2_database_2of2	2/18/2018 1:41 PM	WinRAR ZIP archive	984,365 KB

Fig 6.1 : Created a local directory named ‘user_dir’ in ‘C:\Oracle’

Then, we created a directory

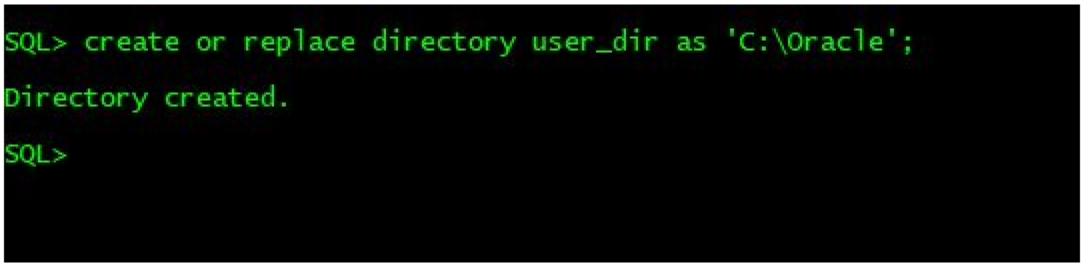

```
SQL> create or replace directory user_dir as 'C:\Oracle';

Directory created.

SQL>
```

Fig 6.2: Created a directory user_dir

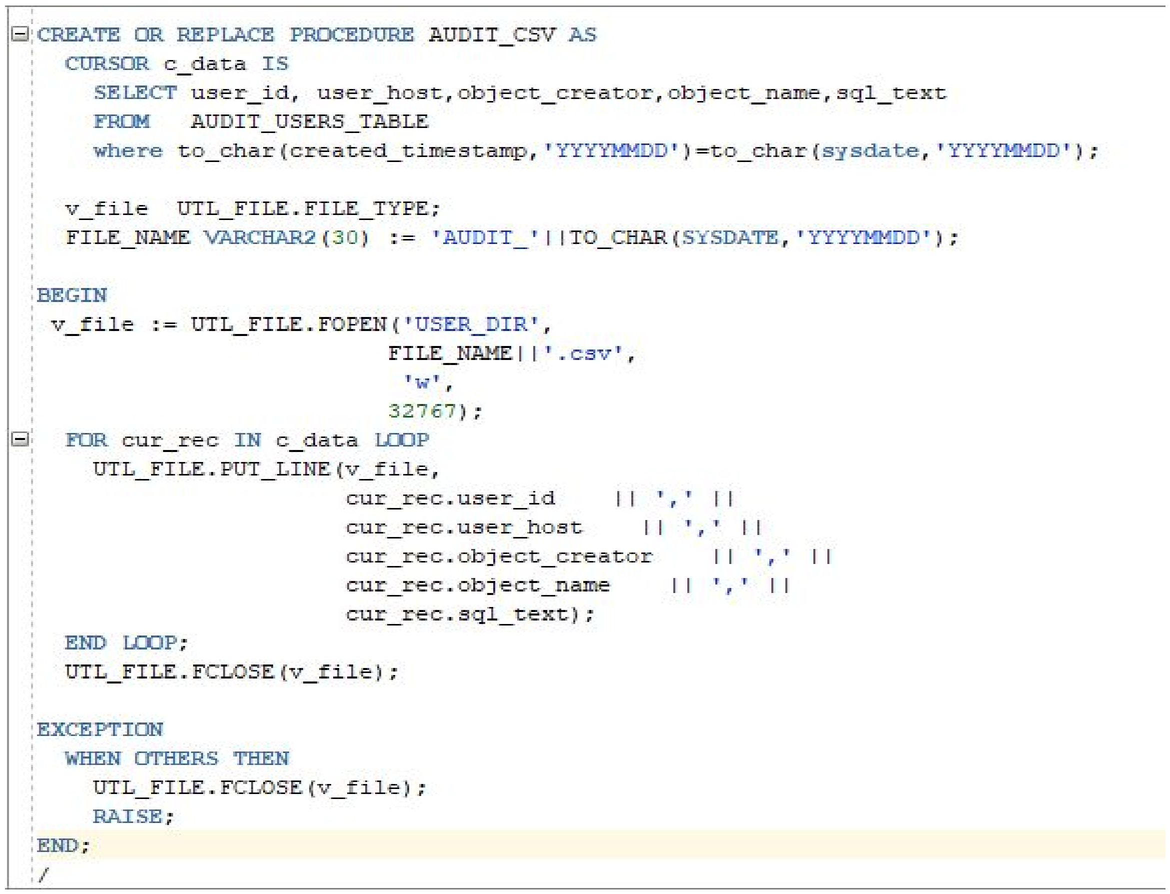
7. Create a CSV Output file:

The records in our ‘**audit_users_table**’ are to be written into a csv file. For that we created a procedure named ‘**AUDIT_CSV**’ .

We performed following activities in the procedure:

- Created a cursor ‘*c_data*’ to the table ‘**AUDIT_USERS_TABLE**’ of *user_id, user_host, oject_creator, object_name, sql_text* columns where the timestamp matches the current timestamp.
- Created a variable named *v_file* of type *FILE_TYPE* from **UTL_FILE** package.

- Create a variable to store the name of the audit file. File name of the the audit file was the system date before the word 'AUDIT_'. For example: AUDIT_20180318.
- We used **FOPEN** function of **UTL_FILE** package with the parameters .
    - Location of the directory. Here we used **USER_DIR** directory that we created earlier.
    - Name of the output csv file
    - Open mode of the file
        - 'W' for write mode
        - 'A' for append mode
    - Max line size set to 32767.
- Then, we looped the records pointed from the cursor and wrote it to the file.
- Finally we closed the file. Also, we wrote some exception too.



```
CREATE OR REPLACE PROCEDURE AUDIT_CSV AS
  CURSOR c_data IS
    SELECT user_id, user_host,object_creator,object_name,sql_text
    FROM   AUDIT_USERS_TABLE
    where to_char(created_timestamp,'YYYYMMDD')=to_char(sysdate,'YYYYMMDD');

  v_file  UTL_FILE.FILE_TYPE;
  FILE_NAME VARCHAR2(30) := 'AUDIT_'||TO_CHAR(SYSDATE,'YYYYMMDD');

BEGIN
 v_file := UTL_FILE.FOPEN('USER_DIR',
                          FILE_NAME||'.csv',
                           'w',
                          32767);
  FOR cur_rec IN c_data LOOP
    UTL_FILE.PUT_LINE(v_file,
                      cur_rec.user_id    || ',' ||
                      cur_rec.user_host    || ',' ||
                      cur_rec.object_creator    || ',' ||
                      cur_rec.object_name    || ',' ||
                      cur_rec.sql_text);
  END LOOP;
  UTL_FILE.FCLOSE(v_file);

EXCEPTION
  WHEN OTHERS THEN
    UTL_FILE.FCLOSE(v_file);
    RAISE;
END;
/
```

Fig 7.1: AUDIT_CSV Procedure

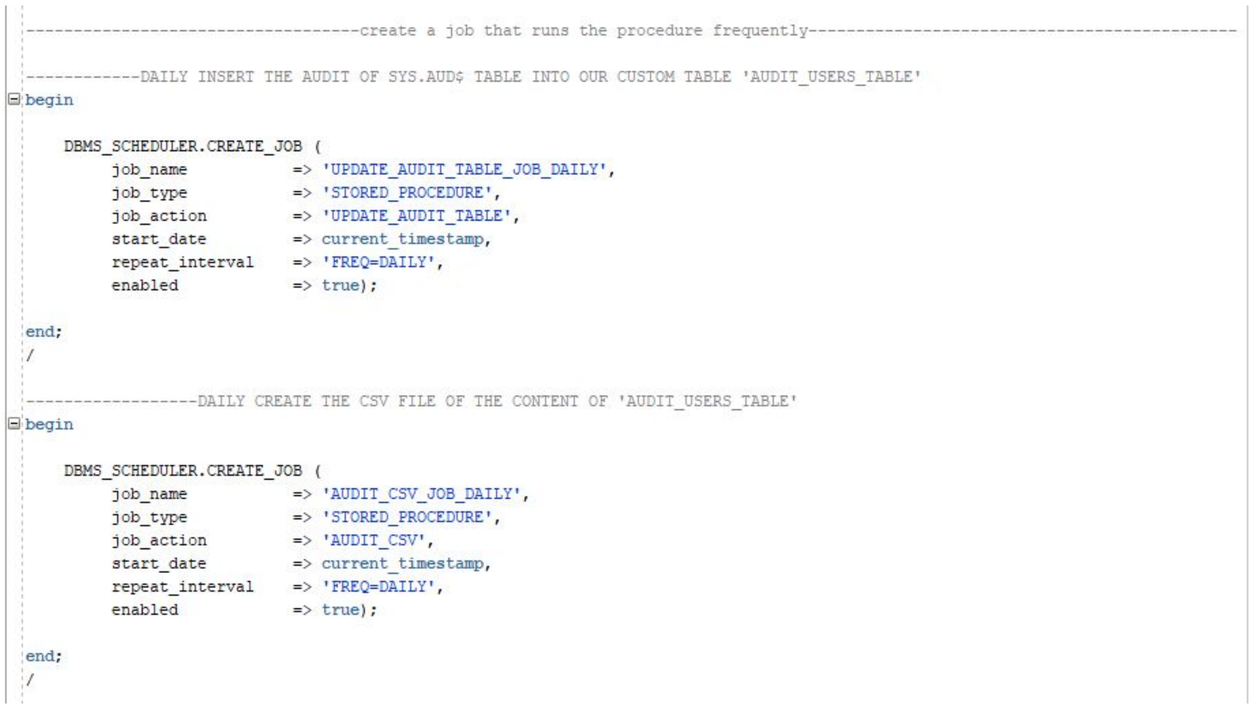
8. Creating the schedule:

After we successfully create the output csv file, we have to automate this task daily.

For this, we used DBMS_SCHEDULER package to schedule this task regularly.

> Using the **DBMS_SCHEDULER** package, we created a job named '**UPDATE_AUDIT_TABLE_JOB_DAILY**' and '**AUDIT_CSV_JOB_DAILY**' of both type STORED PROCEDURE and calling '**UPDATE_AUDIT_TABLE**' and '**AUDIT_CSV**' procedures respectively.

Both the jobs were started at the current timestamp and iterated DAILY.



```
-----------------------------------create a job that runs the procedure frequently--------------------------------------------

------------DAILY INSERT THE AUDIT OF SYS.AUD$ TABLE INTO OUR CUSTOM TABLE 'AUDIT_USERS_TABLE'
begin

    DBMS_SCHEDULER.CREATE_JOB (
         job_name           => 'UPDATE_AUDIT_TABLE_JOB_DAILY',
         job_type           => 'STORED_PROCEDURE',
         job_action         => 'UPDATE_AUDIT_TABLE',
         start_date         => current_timestamp,
         repeat_interval    => 'FREQ=DAILY',
         enabled            => true);

end;
/

------------------DAILY CREATE THE CSV FILE OF THE CONTENT OF 'AUDIT_USERS_TABLE'
begin

    DBMS_SCHEDULER.CREATE_JOB (
         job_name           => 'AUDIT_CSV_JOB_DAILY',
         job_type           => 'STORED_PROCEDURE',
         job_action         => 'AUDIT_CSV',
         start_date         => current_timestamp,
         repeat_interval    => 'FREQ=DAILY',
         enabled            => true);

end;
/
```

Fig 8.1 : Schedulers

And, we can view the schedulers using

*SELECT * FROM ALL_SCHEDULER_JOBS;*

9. Drop the scheduler job:

In case we need to drop the scheduler job, we could do this using the script

```
DBMS_SCHEDULER.DROP_JOB(  'JOB_NAME'  );
```