TECHNICAL NOTE

USING TWO TEMPORARY TABLESPACES IN PEOPLESOFT

Prepared By David Kurtz, Go-Faster Consultancy Ltd.
Technical Note
Version 1.04
Friday 23 November 2012
(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)
File: PSFT.Two Temporary Tablespaces.docx, 23 November 2012

Contents

Introduction

This document discusses how to make PeopleSoft use a second temporary tablespace for adhoc queries and other specific processes.

Why You Might Need a Second Temporary Tablespace

I am working with two different PeopleSoft customers who have had challenges with the size of the temporary tablespaces. Critical batch processes have failed because they have exhausted space in the temporary tablespace.

```
ORA-01652 unable to extend temporary segment by 16 in tablespace PSTEMP
```

- In one case, the one and only temporary tablespace in a Payroll system has over time been extended to in excess of 360 GB. This has happen in response to PeopleSoft processes that failed because they cannot allocate temporary tablespace because somebody else has consumed it. This treated the symptom rather than the cause. This system has a number of other Oracle database users who are have read-only access to the PeopleSoft data to perform adhoc queries. These users all share the one temporary tablespace. Occasionally, a query will be submitted that runs for many hours, writing many gigabytes of data to the temporary tablespace, when it would have been better to have the query error, and let someone examine it.

- Another system has 64 GB in the PSTEMP temporary tablespace used by SYSADM. All other users already use another temporary tablespace, but PeopleSoft processes sometimes still fail because most of the temporary tablespace has been consumed by an adhoc PS/Query process, and there is nothing left for other processes. This system also has other Oracle database users with read-only access, but here they use the default TEMP temporary tablespace.

If a PeopleSoft system has database users executing adhoc queries, then allocating those users to separate temporary tablespace is a sensible first step.

However, in this document, I suggest going further. I propose switching some PeopleSoft processes to use a different temporary tablespace. Regular PeopleSoft processing will continue to use the first temporary tablespace, but adhoc queries could use the second temporary tablespace. Thus, the first temporary tablespace can be sized to cater for normal processes safe in the knowledge that it won't be consumed by adhoc queries and you won't get failures due to space errors. Meanwhile, the second tablespace can be limited to a reasonable size[1] and queries that make unreasonable demands on the temporary tablespace will error.

[1] I am not going to say what constitutes a 'reasonable size' - that will depend on your data.

Two Features

Before I explain how to configure this in PeopleSoft, I will describe two features that I am going to exploit.

Effect of Current Schema on Temporary Tablespace

In this test, I have created a second database user SYSADMRP that will be used for adhoc queries, such as I would create if I wanted to use PeopleSoft with a standby reporting database. In my database, SYSADM's temporary tablespace is PSTEMP, so I have assigned SYSADMRP to TEMP.

```
CREATE USER sysadmrp IDENTIFIED BY sysadmrp
/
ALTER USER sysadmrp TEMPORARY TABLESPACE temp
/
COLUMN username FORMAT A8 HEADING 'User|Name'
COLUMN temporary_tablespace FORMAT a11 HEADING 'Temporary|Tablespace'
SELECT username, temporary_tablespace FROM dba_users
WHERE username LIKE 'SYSADM%'
/

User     Temporary
Name     Tablespace
-------- -----------
SYSADMRP TEMP
SYSADM   PSTEMP
```

In order to illustrate which temporary tablespace is used, I will create a global temporary table. I will create the table in SYSADM schema, but I will also grant SYSADMRP full access to it.

```
CREATE GLOBAL TEMPORARY TABLE sysadm.gtt (a NUMBER)
ON COMMIT DELETE ROWS
/
GRANT ALL ON sysadm.gtt TO sysadmrp
/
```

The global temporary table is only physically created when somebody puts some rows into it. It will be created in temporary tablespace for that user. In this example, I have connected to SYSADM. I can see that a temporary data segment has been created in the PSTEMP temporary tablespace by SYSADM.

```
INSERT INTO sysadm.gtt SELECT ROWNUM FROM dual CONNECT BY LEVEL<= 1000
/
COLUMN username FORMAT a8 HEADING 'USERNAME'
COLUMN tablespace FORMAT a11
COLUMN contents FORMAT a9
COLUMN segtype FORMAT a8
SELECT username, tablespace, contents, segtype FROM v$sort_usage
/

User
Name     TABLESPACE  CONTENTS  SEGTYPE
-------- ----------- --------- --------
SYSADM   PSTEMP      TEMPORARY DATA
```

Next, and without terminating the first session, I will repeat this in another SQL*Plus session. I will still connect as SYSADM, but the first thing I will do is to change the current schema to SYSADMRP. Now I get a second temporary segment, still owned by SYSADM because I connected as SYSADM, but in the TEMP tablespace, which is the temporary tablespace for SYSADMRP.

```
ALTER SESSION SET CURRENT_SCHEMA=SYSADMRP;
INSERT INTO sysadm.gtt SELECT ROWNUM FROM dual CONNECT BY LEVEL<= 1000
/
COLUMN username FORMAT a8
COLUMN tablespace FORMAT a11
COLUMN contents FORMAT a9
COLUMN segtype FORMAT a8
SELECT username, tablespace, contents, segtype FROM v$sort_usage
/

USERNAME TABLESPACE  CONTENTS  SEGTYPE
-------- ----------- --------- --------
SYSADM   PSTEMP      TEMPORARY DATA
SYSADM   TEMP        TEMPORARY DATA
```

So we can see that a session uses the temporary tablespace of the *current_schema* rather than the connected user. We will exploit that later.

PeopleTools Support for Standby Reporting Database

PeopleSoft supports use of a second database for reporting from PeopleTools 8.51. That database could be maintained by Active Data Guard or Golden Gate. In Active Data Guard, the standby database is open in read-only mode while redo is also being applied.

It is possible to specify a second database connection to the standby database for PeopleSoft processes in Application Server and Process Scheduler. The application server processes make a second database connection to the standby database. PS/Queries and components marked as read-only are routed through this connection in the application server. Batch processes marked as being read-only are connected using the connection credentials.

```
[Startup]
;============================================================================
; Database Signon settings
;============================================================================
DBName=HCM91
DBType=ORACLE
UserId=PSAPPS
UserPswd=PSAPPS
ConnectId=people
ConnectPswd=kyD3QPxnrag=
ServerName=
StandbyDBName=HCM91
StandbyDBType=ORACLE
StandbyUserId=PSAPPS2
StandbyUserPswd=PSAPPS2
```

NB: You cannot configure a standby connection in a process scheduler that runs stand-alone *psae* processes – instead you must use *PSAESRV* server processes[2]. If you configure a standby connection in a Process Scheduler that runs stand-alone *psae* processes, then any Application Engine, whether the process is defined as read-only or not, will fail to sign-on to the database.

```
Error in sign on
  Database Type: 7 (ORACLE)
  Database Name: HCM91
  Server Name:
  OperID:
  ConnectID: people
  Process Instance: 0
  Reason: Invalid user ID or password for database signon. (id=)
  Note:  Attempt to authenticate using GUID 6a1ced41-2fe0-11e2-9183-be3e31d6e740

Invalid command line argument list.
  process command line: -CT ORACLE -CD HCM91 -GUID 6a1ced41-2fe0-11e2-9183-be3e31d6e740 -SS NO -SN NO
  GUID command line   : -CT ORACLE -CD HCM91 -CO "PS" -CP Unavailable -R 1 -I 852 -AI AEMINITEST -OT 6 -FP
"C:\app\pt\appserv\prcs\HCM91PSNT\log_output\AE_AEMINITEST_852\" -OF 14
```

This isn't a bug, it functions as stated in the documentation, but it is a significant restriction. It seems odd to me that while PSAESRV is intended for systems with many shortlived AE processes that its use is effectively mandated with standby databases.

There are a number of elements to the PeopleSoft configuration for Active Data Guard. Essentially it looks like another PeopleSoft database in another schema of the same Oracle database, but everything is a synonym that refers back to the same PeopleSoft database in the first schema.

- The standby connection must connect to a second database user. By default this is called SYSADMRP

[2] See PeopleBooks -> PeopleTools Data Management -> Configuring Read Only Processes:

"Note. The use of Oracle Active Data Guard with PeopleSoft batch processing only applies to the following: Application Engine processes run through the Process Scheduler with PSAESRV configured and SQR processes."

- A synonym is created in SYSADMRP for every table and view in the SYSADM schema (see *$PS_HOME/scripts/createlocalsynonyms.sql*). PeopleSoft calls these 'local' synonyms because they point to objects in the same database. Note that you don't need to grant select privilege for these tables.

```
REM Create local synonyms for tables
SET HEADING OFF
Spool CREATELOCALTABLESYNONYMS.SQL
SELECT 'CREATE SYNONYM ' || TABLE_NAME || ' FOR <SYSADM>.' || TABLE_NAME || ';'
FROM ALL_TABLES
WHERE OWNER = '<SYSADM>'
ORDER BY TABLE_NAME
;
Spool off

REM Create local synonyms for views
Spool CREATELOCALVIEWSYNONYMS.SQL
SET HEADING OFF
SELECT 'CREATE SYNONYM ' || VIEW_NAME || ' FOR <SYSADM>.' || VIEW_NAME || ';'
FROM ALL_VIEWS
WHERE OWNER = '<SYSADM>'
ORDER BY VIEW_NAME
;
Spool off
```

- Even read-only processes need to update certain PeopleTools tables. For these tables synonyms are created that point via a database link to the tables in the SYSADM schema of the primary database (created by *$PS_HOME/scripts/createremotesynonyms.sql*). Eg:

```
CREATE or REPLACE SYNONYM PSPRCSRQST for PSPRCSRQST@PRIMARY;
```

This configuration is done on the primary database and it replicates through to the standby database. On both primary and standby database the link points to the primary database. If the primary database fails over, the database link will also fail over and will point to the new primary database.

Note that Active Data Guard is only available from Oracle 11gR2, it requires PT8.51 and it is also a separately licenced cost option. It is not needed for the technique that I am about to describe, but it gives us some insights as how we might make some operations use a different temporary tablespace.

Configuring Two Temporary Tablespaces

If you are using PeopleSoft with a standby database then the read-only activity will occur on that database and use the temporary tablespace in that database. You are unlikely to need to use two different temporary tablespaces in the same database. Of course, if you are running in standby then the two workloads will co-exist again, but temporary tablespace usage may not be the greatest concern.

The rest of this discussion will assume that you are not using a standby database, but are running on a single database.

To specify alternative temporary tablespaces when not the standby connection available from PeopleTools 8.51 you need to follow these steps

1. If not already created the SYSADMRP needs privilege to create a session and to create procedures.

```
GRANT CONNECT TO sysadmrp;
GRANT CREATE ANY PROCEDURE TO sysadmrp;
GRANT CREATE ANY SYNONYM TO sysadmrp;
```

2. I have found that many customers have created roles to create read and write privileges on all PeopleSoft tables. Since, by default, PeopleSoft doesn't use PL/SQL I am going to grant privileges via roles.

```
CREATE ROLE ps_read_all;
CREATE ROLE ps_update_all;
```

3. Grant privilege on SYSADM objects to the new roles as follows. I am using a procedure[3] so that I can also create a trigger to grant these privleges automatically as new tables are created (see step 7)

```
CREATE OR REPLACE PROCEDURE sysadm.grant_table_privileges(p_table_name VARCHAR2) IS
  l_sql VARCHAR2(1000 CHAR);
BEGIN
  l_sql := 'GRANT SELECT ON sysadm.'||p_table_name||' TO ps_read_all';
  dbms_output.put_line('DDL:'||l_sql);
  EXECUTE IMMEDIATE l_sql;

  l_sql := 'GRANT INSERT, UPDATE, DELETE ON sysadm.'||p_table_name||' TO ps_update_all';
  dbms_output.put_line('DDL:'||l_sql);
  EXECUTE IMMEDIATE l_sql;
EXCEPTION WHEN OTHERS THEN --print errors but carry on processing
  dbms_output.put_line(sqlerrm);
END;
/

set serveroutput on
BEGIN
 FOR i IN (
  (
  SELECT table_name FROM all_tables WHERE owner = 'SYSADM'
  UNION ALL
  SELECT view_name  FROM all_views  WHERE owner = 'SYSADM'
  ) MINUS (
  SELECT table_name FROM all_tab_privs
  WHERE  table_schema = 'SYSADM' AND grantee = 'PS_READ_ALL' AND privilege = 'SELECT'
  INTERSECT
  SELECT table_name FROM all_tab_privs
  WHERE  table_schema = 'SYSADM' AND grantee = 'PS_UPDATE_ALL' AND privilege = 'INSERT'
  INTERSECT
  SELECT table_name FROM all_tab_privs
  WHERE  table_schema = 'SYSADM' AND grantee = 'PS_UPDATE_ALL' AND privilege = 'UPDATE'
  INTERSECT
  SELECT table_name FROM all_tab_privs
  WHERE  table_schema = 'SYSADM' AND grantee = 'PS_UPDATE_ALL' AND privilege = 'DELETE'
  )
) LOOP
  sysadm.grant_table_privileges(i.table_name);
 END LOOP;
END;
/
```

[3] The error handler is included because some delivered PeopleSoft views reference Oracle catalogue views that are not updatable and attempting to grant these privileges causes an error.

```
GRANT INSERT, UPDATE, DELETE ON sysadm.PS_EOUF_SYSVIEWDEP TO ps_update_all
ORA-01720: grant option does not exist for 'SYS.ALL_DEPENDENCIES'
```

4. Now you can grant the roles to the second schema[4].

```
GRANT ps_read_all TO sysadmrp;
GRANT ps_update_all TO sysadmrp;
```

5. Use this script to create synonyms in SYSADRP for every table and view in SYSADM as follows.
   - I am using a procedure so that I can also create a trigger to synonyms automatically as new tables are created (see step 7).
   - Note that the procedure is owned by SYSADMRP because the synonyms are created in the SYSADMRP schema, but execute privilege is granted to SYSADM so that it can be called by a DDL trigger on the SYSADM schema that fires as tables and views are created.

```
CREATE OR REPLACE PROCEDURE sysadmrp.create_synonym (p_table_name VARCHAR2) IS
  l_sql VARCHAR2(100);
BEGIN
  l_sql := 'CREATE OR REPLACE SYNONYM sysadmrp.'||p_table_name||' FOR sysadm.'||p_table_name;
  dbms_output.put_line(l_sql);
  EXECUTE IMMEDIATE l_sql;
EXCEPTION WHEN OTHERS THEN --print errors but carry on processing
  dbms_output.put_line(sqlerrm);
  RAISE;
END;
/
GRANT EXECUTE ON sysadmrp.create_synonym TO sysadm
/

set serveroutput on
DECLARE
 l_sql VARCHAR2(1000);
BEGIN
 FOR i IN (
  SELECT table_name FROM all_tables WHERE owner = 'SYSADM'
  UNION
  SELECT view_name  FROM all_Views  WHERE owner = 'SYSADM'
  MINUS
  SELECT synonym_name FROM   all_synonyms
  WHERE  owner = 'SYSADMRP'
  AND    table_name = synonym_name
  AND    table_owner = 'SYSADM'
  AND    db_link IS NULL
 ) LOOP
  sysadmrp.create_synonym(i.table_name);
END LOOP;
END;
/
```

[4] Even read-only PeopleSoft processes require update access on some tables. Oracle delivers the script *createremotesynonyms.sql* for use when configuring a standby database to create synonyms that link back to the primary database. The minimum would be to grant update privileges on the tables mentioned in *createremotesynonyms.sql*. However, I have simply granted the PS_UPDATE_ALL role to the second schema.

6. As new tables and views are created it would be helpful if there were triggers to automatically create synonyms and grants for those objects as needed. This trigger fires when a table or view is created in the SYSADM schema. The trigger cannot create the synonym or grants itself because it would fail with a deadlock on the data dictionary. Instead it schedules a job for immediate execution that creates a corresponding synonym in the SYSADMRP schema pointing back to the new object. The job calls the procedures owned that I created earlier. Note that by default jobs are automatically dropped.

```
CREATE OR REPLACE TRIGGER sysadm.new_object AFTER CREATE ON sysadm.schema
DECLARE
  l_cmd VARCHAR2(1000 CHAR);
  l_job_name VARCHAR2(30);
BEGIN
  IF ora_dict_obj_type IN('TABLE','VIEW') THEN
    dbms_output.put_line(ora_dict_obj_type||'.'||ora_dict_obj_name);
    l_job_name := SYS.DBMS_SCHEDULER.GENERATE_JOB_NAME(prefix=>SUBSTR(ora_dict_obj_name,1,17)||'_');
    l_cmd := 'sysadm.grant_table_privileges('''||ora_dict_obj_name||''');'
           ||'sysadmrp.create_synonym('''||ora_dict_obj_name||''');';
    dbms_output.put_line('Job '||l_job_name||':'||l_cmd);
    sys.dbms_scheduler.create_job
    (job_name => l_job_name
    ,job_type => 'PLSQL_BLOCK'
    ,job_action => 'BEGIN '||l_cmd||' END;'
    ,enabled => TRUE
    );
  END IF;
EXCEPTION
 WHEN OTHERS THEN
  dbms_output.put_line('Error Surpressed:'||sqlerrm);
END;
/
```

7. Create a logon trigger to set current schema to SYSADMRP where the program name is PSQRYSRV. The trigger actually reads the value of MODULE from the system contexts, but when the program first connects it has the same value as the program name. It is only after logon that the module and action are set to other values.

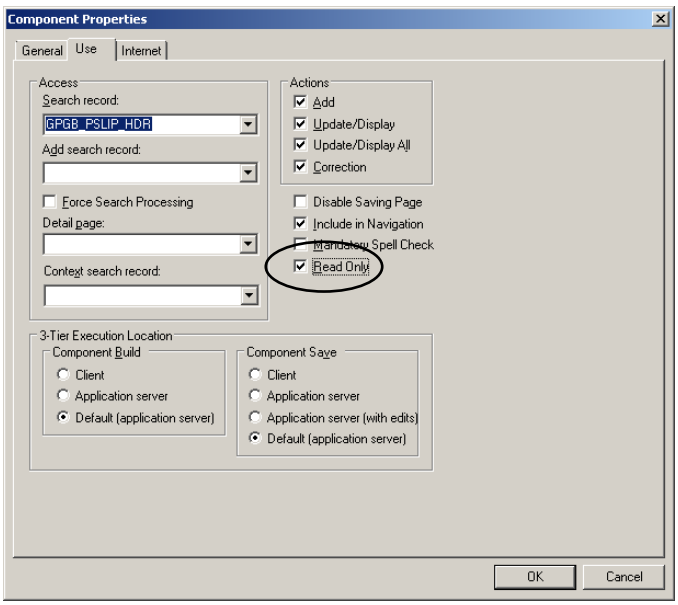
```
CREATE OR REPLACE TRIGGER sysadm.psqrysrv
AFTER LOGON
ON sysadm.schema
DECLARE
  l_module VARCHAR2(64);
BEGIN
  SELECT sys_context('USERENV', 'MODULE')
  INTO   l_module
  FROM   dual;

  IF UPPER(l_module) LIKE 'PSQRYSRV%' THEN --then this is a PSQRYSRV session
    EXECUTE IMMEDIATE 'ALTER SESSION SET CURRENT_SCHEMA=SYSADMRP';
  END IF;
EXCEPTION WHEN NO_DATA_FOUND THEN NULL;
END;
/
```

   - Note that this approach will only work if you configure separate PSQRYSRV query server processes, otherwise queries are directed to PSAPPSRV regular application server processes along with all the other application server activity.
   - This technique does not work for queries run by selecting the 'Run' tab in the Query Manager component because they are executed by the ICPanel service which runs on the PSAPPSRV server processes. Queries run via any of the run links on the search dialogue to either the Query View or Query Manager components dp use the ICQuery service and so are directed to PSQRYSRV process and hence use the alternative temporary tablespace.

8. We can also set current schema for named batch processes with a trigger on the process request table PSPRCSRQST.
   - This example trigger fires for the process that runs scheduled PS/Queries. You might want to extend it to other specific process, perhaps PSXPQRYRPT.
   - The PSFTAPI package is available from http://www.go-faster.co.uk/scripts.htm#psftapi.sql

```
CREATE OR REPLACE TRIGGER sysadm.set_current_schema
BEFORE UPDATE OF runstatus ON sysadm.psprcsrqst
FOR EACH ROW
WHEN (  (new.runstatus = '7' OR old.runstatus != '7') --if starting or terminating
     AND new.prcsname IN('PSQUERY')) --restrict to certain programs
DECLARE
  l_sql VARCHAR2(100);
BEGIN
  sysadm.psftapi.set_prcsinstance(:new.prcsinstance);
  l_sql := 'ALTER SESSION SET CURRENT_SCHEMA=';
  IF :new.runstatus = '7' THEN --if starting set alternative schema
    l_sql := l_sql||'SYSADMRP';
  ELSE --when process terminates reset to standard schema in case this is a PSAESRV process
    l_sql := l_sql||'SYSADM';
  END IF;
  sysadm.psftapi.message_log(l_sql);
  EXECUTE IMMEDIATE l_sql;
END;
/
```

9. If you want to make specific components use the alternative temporary tablespace, that requires the Active DataGuard functionality in PT8.51 or higher. You will need to set the attribute on each component in Application Designer.

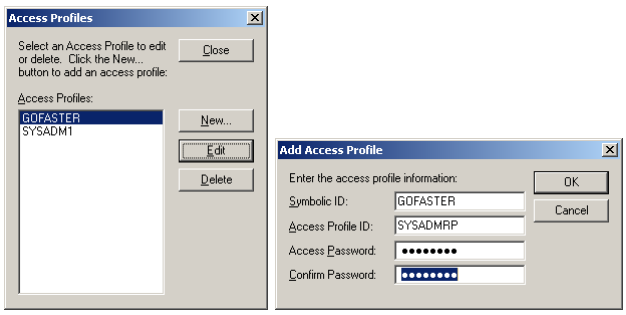


10. If using PeopleTools 8.51 or higher, and you have no plans to use a standby database for reporting then you can use the PeopleTools support for it to move PS/Query and specific components and batch process to the standby connection.

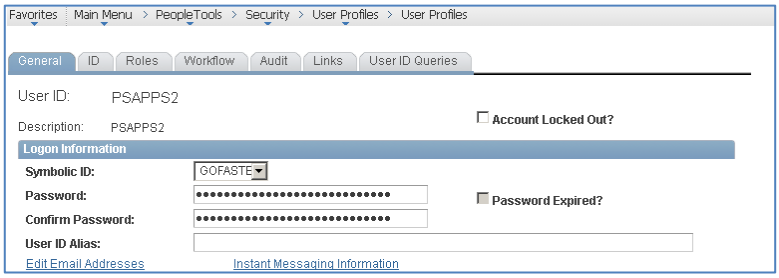
    a. In this test we do not need the triggers created in the previous configuration examples.

```
DROP TRIGGER sysadm.psqrysrv;
DROP TRIGGER sysadm.set_current_schema;
```

    b. Use Application Designer to create a new access profile that points to the second schema.
        - Normaly, if you were using Active Data Guard you would create a second entry in PS.PSDBOWNER pointing to the second schema, and a corresponding TNS entry. However, this is not necessary with this technique. The second schema is selected by the access profile.



c. Allocate the new access profile to the operator who will be used in the standby connection. In this case access profile GOFASTER is allocated to PeopleSoft user PSAPPS2



d. Configure the standby connection in App server and process scheduler as you would for Active Data Guard. The standby connection can use the same database but it needs a second PeopleSoft user ID with a different access profile which will connect it to the second schema.

```
[Startup]
;============================================================================
; Database Signon settings
;============================================================================
DBName=HCM91
DBType=ORACLE
UserId=PSAPPS
UserPswd=PSAPPS
ConnectId=people
ConnectPswd=peop1e
ServerName=
StandbyDBName=HCM91REP
StandbyDBType=ORACLE
StandbyUserId=PSAPPS2
StandbyUserPswd=PSAPPS2
```

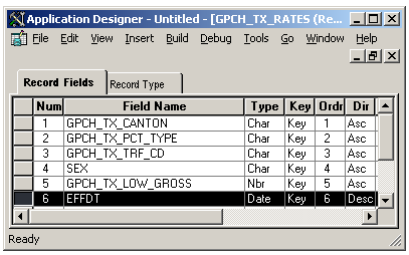
- However, if you want to continue to use standalone PSAE processes, then you might prefer to not to configure the standby connection on the process scheduler but to use the trigger on PSPRQSRQST.

e. Specify a different temporary tablespace on the second schema

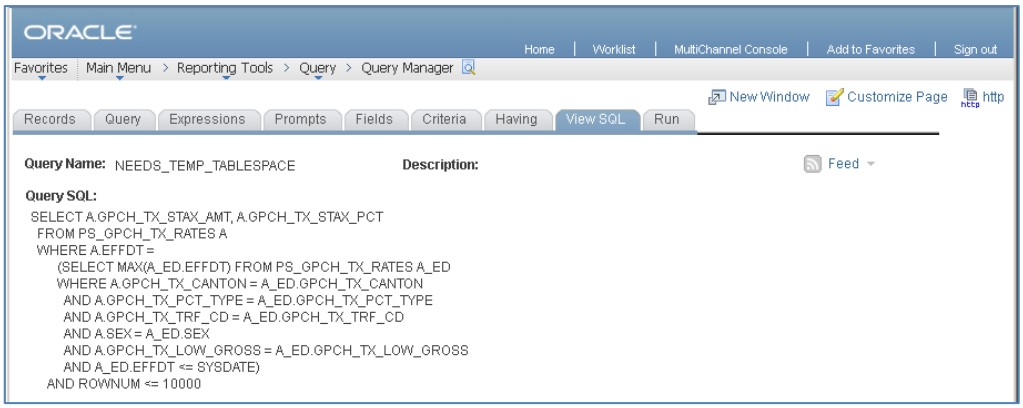
```
ALTER USER sysadm   TEMPORARY TABLESPACE temp;
ALTER USER sysadmrp TEMPORARY TABLESPACE temp;
```

Demonstration

It is easy to demonstrate this technique. I have created an absurd query that requires a large sort or hash. I have used the Swiss canton tax rates table because it is delivered in HR with over 5 million rows. However, to make it easier to use lots of temporary tablespace I have moved EFFDT from the 3rd to the 6th key column in the record.



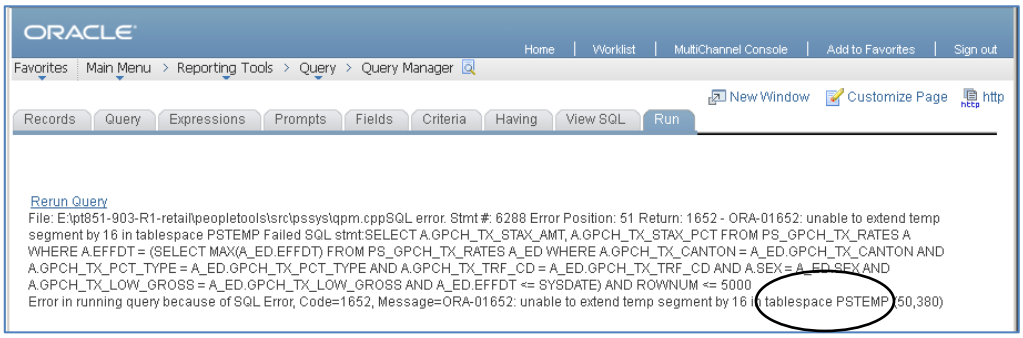
Now I can easily produce this effective-dated query.



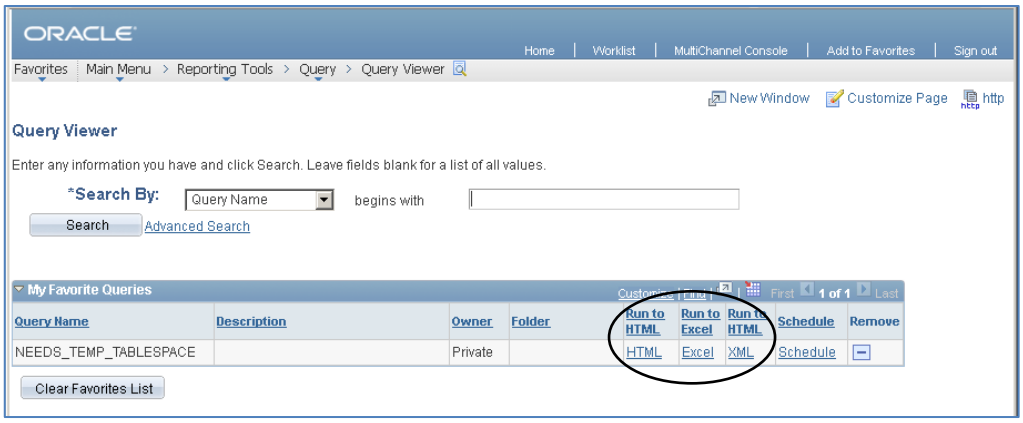
My temporary tablespaces are each only 100Mb, and autoextend has been disabled so that the query will always produce a space error when it fills the temporary tables. This is so that I can easily see which temporay tablespace has been used.

```
ALTER DATABASE TEMPFILE 'C:\APP\ORACLE\ORADATA\HCM91\TEMP01.DBF' AUTOEXTEND OFF;
ALTER DATABASE TEMPFILE 'C:\APP\ORACLE\ORADATA\HCM91\TEMP01.DBF' RESIZE 100M;
ALTER DATABASE TEMPFILE 'C:\APP\ORACLE\ORADATA\HCM91\PSTEMP01.DBF' AUTOEXTEND OFF;
ALTER DATABASE TEMPFILE 'C:\APP\ORACLE\ORADATA\HCM91\PSTEMP01.DBF' RESIZE 100M;
```

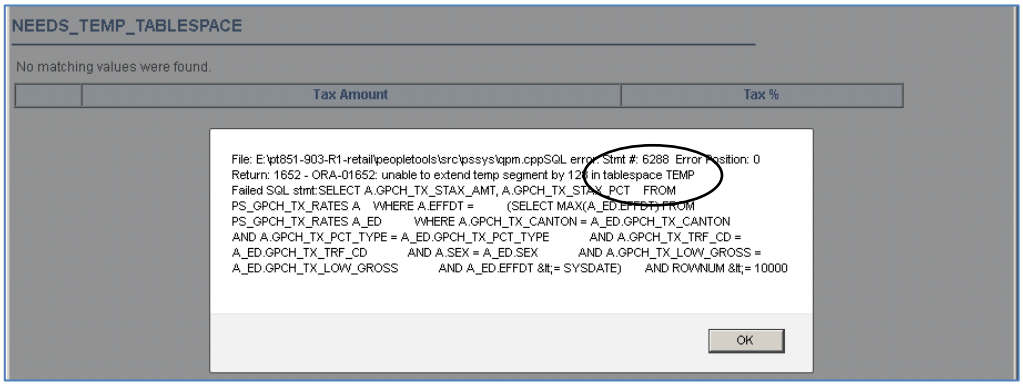
If I run that in the query manager I get a space error in the normal temporary tablespace PSTEMP which is used by SYSADM. This is because this component runs the query in an ICPanel service which is still executed by the PSAPPSRV process. So if you are developing queries in production you could still get temporary tablespace errors.



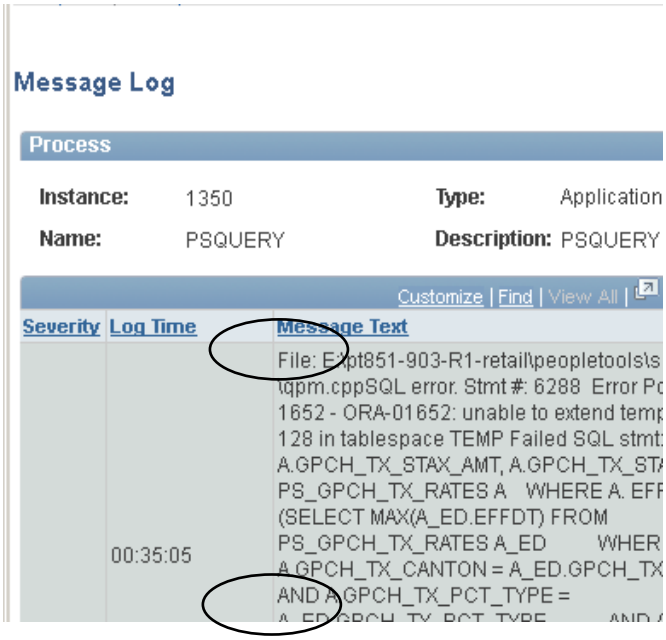
If I select any of the run links on the search dialogue to either the Query View or Query Manager components …



The query still errors, but note that the temporary tablespace in the error message has changed to TEMP.



If I schedule the query to run via the PSQUERY Application Engine process then that also errors and the message shows it has also used the alternative tablespace, TEMP.



QED.

Conclusion

Now, you can choose an appropriate size for PSTEMP which will be sufficient for the regular operation of the application. Adhoc queries will use TEMP. You might choose to set a temporary tablespace size that may still cause queries to fail with a temproray tablespace error, but at least that won't cause business-as-usual processes to crash.

You could even choose to run with three temporary tablespaces, one for PeopleSoft processes, one for PeopleSoft queries, and one for adhoc users accessing the database directly.