

Oracle10G Utilities

Session Objectives

- Utility discovery
- Utilities detailed
 - data pump (impdp, expdp, dbms_data_pump)
 - dbms_file_transfer
 - trcsess

What is a utility?

Websters defines a utility as “a useful article or device”

Value Determination

- Documentation
 - TechNet
 - MetaLink
 - Google
 - OracleUtilities.com
- Research
 - PL/SQL package specs
 - OS executables (just run it!)

Utility Discovery

For new OS binaries ...

```
dircmp -s newDir oldDir
```

New OS Binaries

agentok.buf	expdp	ocrconfig
cemutls	extjob	ocrdump
clscfg	extproc32	ocssd
clsfmt	genezi	oidca
clsid	impdp	olsadmintool
cmadmin	isqlplusctl	olsoidsync
cmctl	kfod	onsctl
cmgw	lbuilder	orajaxb
cmmigr	lcscan	orapipe
crsctl	localconfig	orapki
ctxlc	lsnodes	osdbagrp
dsm121dif	makeserverwlt	proxyserv
e2eme	mkstore	rawutl
emagent	netlaunch	repo_mig
emagtm	nmb	runclass
emctl	nmei	searchctl
emtgtctl	nmo	sslsetup
emtgtctl2	nmocat	trcsess
emutil	nmuct	xsl
emwd	nmupm	xvm
	nmus	
	ocrcheck	

Utility Discovery

For new PL/SQL packages ...

```
select object_name from dba_objects
  where      owner = 'SYS'
    and object_type = 'PACKAGE_BODY'
minus
select object_name from dba_objects@ORCL92
  where owner = 'SYS'
    and object_type = 'PACKAGE_BODY';
```

Utility Discovery

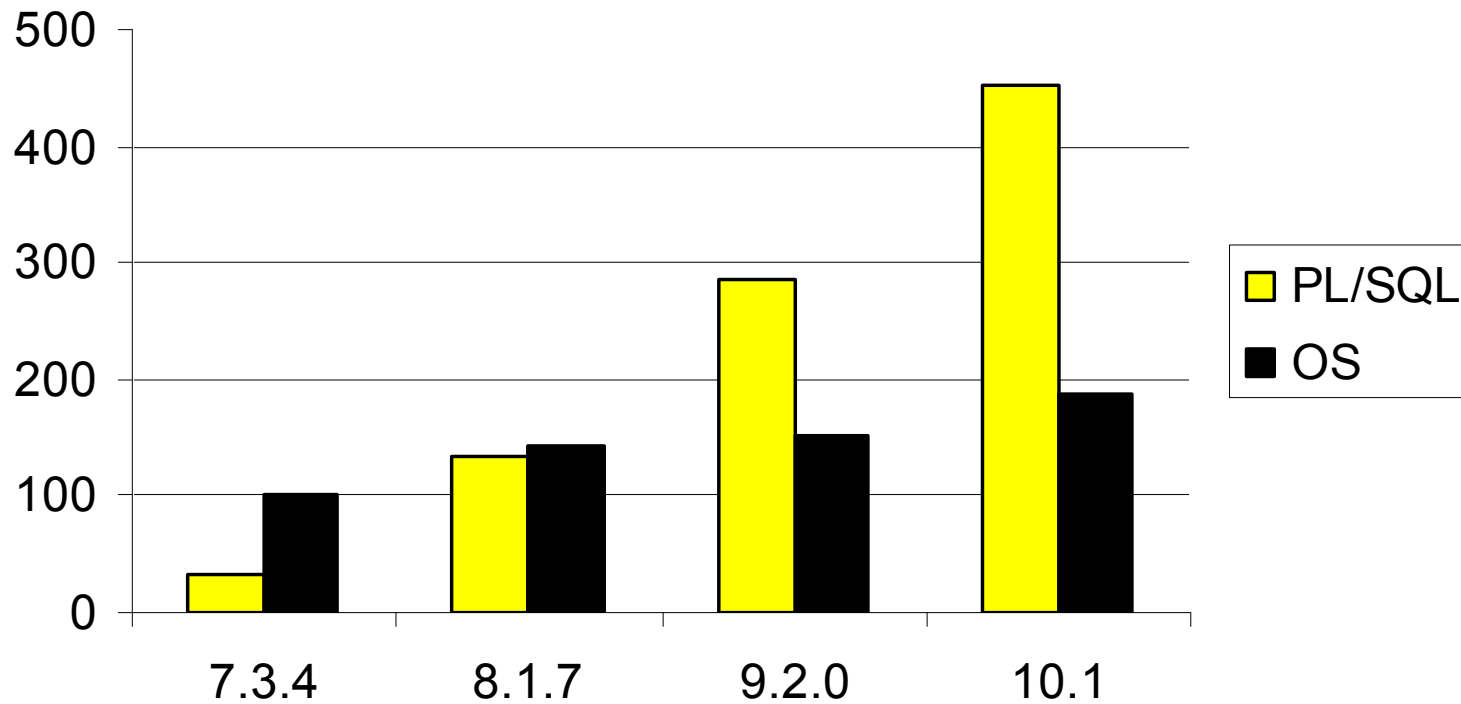
This approach also works for...

- **Instance parameters** `v$parameter`
- **Obsolete parameters**
`v$obsolete_parameter`
- **System events** `v$event_name`
- **V\$ tables** `v$fixed_view_definition`

New Packages

CONNECTIONINTERFACE	DBMS_LDAP	DBMS_STREAMS_AUTH	GENMDMCLASSCONSTANTS	OLAP_API_SESSION_INIT
CURSORMANAGERINTERFACE	DBMS_LDAP_API_FFI	DBMS_STREAMS_CDC_ADM	GENMDMOBJECTIDCONSTANTS	OWM_9IP_PKG
DATABASEINTERFACE	DBMS_LDAP_UTL	DBMS_STREAMS_DATAPUMP	GENMDMPROPERTYIDCONSTANTS	OWM_BULK_LOAD_PKG
DATAPROVIDERINTERFACE	DBMS_LOGREP_UTIL_INVOK	DBMS_STREAMS_DATAPUMP_UTIL	GENMETADATAPROVIDERINTERFACE	OWM_DDL_PKG
DBMSHSXP	DBMS_LOGSTDBY_PUBLIC	DBMS_STREAMS_LCR_INT	GENPARAMETERIDCONSTANTS	OWM_IEXP_PKG
DBMS_ADVANCED_REWRITE	DBMS_MONITOR	DBMS_STREAMS_MESSAGING	GENSERVERINTERFACE	OWM_MIG_PKG
DBMS_ADVISOR	DBMS_PROFILER	DBMS_STREAMS_PUB_RPC	GENSNAPIINTERFACE	OWM_MP_PKG
DBMS_AQ_BQVIEW	DBMS_PRIVT_TRACE	DBMS_STREAMS_RPC	GET_ERROR\$	OWM_REPUTIL
DBMS_CDC_DPUTIL	DBMS_REGISTRY_SERVER	DBMS_STREAMS_RPC_INTERNAL	INITJVMAUX	OWM_VT_PKG
DBMS_CDC_EXPDP	DBMS_REPCAT_EXP	DBMS_STREAMS_TABLESPACE_ADM	INTERRUPTABLEINTERFACE	PRVT_ACCESS_ADVISOR
DBMS_CDC_EXPVDP	DBMS_REPCAT_MIGRATION	DBMS_STREAMS_TBS_INT	JAVA_XA	PRVT_ADVISOR
DBMS_CDC_IMPDP	DBMS_REPCAT_EXP_UTLI	DBMS_STREAMS_TBS_INT_INVOK	KUPC\$QUEUE	PRVT_DIMENSION_SYS_UTIL
DBMS_CDC_IPUBLISH	DBMS_RULE_EXP_UTLI	DBMS_SUM_RWEQ_EXPORT	KUPC\$QUEUE_INT	PRVT_HDM
DBMS_CDC_ISUBSCRIBE	DBMS_SCHEDULER	DBMS_SUM_RWEQ_EXPORT_INTERNA L	KUPC\$QUE_INT	PRVT_SYS_TUNE_MVIEW
DBMS_CRYPTO	DBMS_SCHED_CLASS_EXPORT	DBMS_SWRF_INTERNAL	KUPD\$DATA	PRVT_TUNE_MVIEW
DBMS_CRYPTO_FFI	DBMS_SCHED_EXPORT_CALLOUTS	DBMS_SWRF_REPORT_INTERNAL	KUPD\$DATA_INT	PRVT_UADV
DBMS_CRYPTO_TOOLKIT	DBMS_SCHED_JOB_EXPORT	DBMS_TRANSFORM_INTERNAL	KUPF\$FILE	PRVT_WORKLOAD
DBMS_CRYPTO_TOOLKIT_FFI	DBMS_SCHED_MAIN_EXPORT	DBMS_UNDO_ADV	KUPF\$FILE_INT	RMJVM
DBMS_DATAPUMP	DBMS_SCHED_PROGRAM_EXPORT	DBMS_UPGRADE_INTERNAL	KUPM\$MCP	SERVERINTERFACE
DBMS_DBUPGRADE	DBMS_SCHED_SCHEDULE_EXPORT	DBMS_WARNING	KUPP\$PROC	SQLJUTL
DBMS_DBVERIFY	DBMS_SCHED_WINDOW_EXPORT	DBMS_WARNING_INTERNAL	KUPV\$FT	SQLJUTL2
DBMS_DIMENSION	DBMS_SCHED_WINGRP_EXPORT	DBMS_WORKLOAD_REPOSITORY	KUPV\$FT_INT	UD_TRIGS
DBMS_FBT	DBMS_SCHEMA_COPY	DBMS_XMLQUERY	KUPW\$WORKER	UTL_COMPRESS
DBMS_FEATURE_USAGE	DBMS_SERVER_ALERT	DBMS_XMLSAVE	LT	UTL_DBWS
DBMS_FEATURE_USAGE_INTERNAL	DBMS_SERVER_ALERT_EXPORT	DBMS_XMLSTORE	LTADM	UTL_I18N
DBMS_FILE_TRANSFER	DBMS_SERVICE	DBMS_XSOQ	LTAQ	UTL_LMS
DBMS_FREQUENT_ITEMSET	DBMS_SQLTUNE	DBMS_XSOQ_ODBO	LTDDL	UTL_RECAMP
DBMS_INDEX_UTL	DBMS_SQLTUNE_INTERNAL	DBMS_XSOQ_UTIL	LTDTRG	UTL_SYS_COMPRESS
DBMS_INTERNAL_SAFE_SCN	DBMS_STAT_FUNCS	DEFINITIONMANAGERINTERFACE	LTPRIV	WM_DDL_UTIL
DBMS_ISCHED	DBMS_STAT_FUNCS_AUX	EXF\$DBMS_EXPFIL_SYSPACK	LTRIC	WM_ERROR
DBMS_I_INDEX_UTL	DBMS_STREAMS_ADM_UTL_INVOK	GENCONNECTIONINTERFACE	LTUTIL	XML_SCHEMA_NAME_PRESENT
DBMS_JAVA		GENCURSORMANAGERINTERFACE	LT_CTX_PKG	
DBMS_JAVA_DUMP		GENDATABASEINTERFACE	LT_EXPORT_PKG	
DBMS_JMS_PLSQL		GENDATAPROVIDERINTERFACE	METADATAPROVIDERINTERFACE	
		GENDATATYPEIDCONSTANTS		
		GENDEFINITIONMANAGERINTERFACE		
		GENFUNCTIONIDCONSTANTS		
		GENINTERRUPTABLEINTERFACE		

Packages – OS Comparison



Why Emphasis on PL/SQL?

- Platform independence
- Speed for DB processing
- Mature Dev environment

Moving Data With Data Pump

Data Pump

- New and improved import and export
- Both OS and PL/SQL versions available
- Enough on this topic for entire book or 3 hour presentation

Advantages over imp exp

- Job interrupt/restart capability
- OS and PL/SQL versions available
- Interactive mode
- Job monitoring
- Fine grained object selection
- Intelligence
- PERFORMANCE!

Data Pump Architecture

- Different than imp/exp
- New background processes
 - Master DMnn
 - Workers DWnn
- Master table created in users schema with name of job name (used for restart) **lots of stuff in it**
- Requires directory object
- Uses Direct path or external table

expdp

- Data pump export
- Faster than traditional export
- Interactive mode
 - Ctrl-C will get you there (or ATTACH)
- Job interrupt and restart capability
 - ATTACH=job_name

```
SQL> select owner_name, job_name, state from dba_datapump_jobs;
```

expdp

- Usage is pretty much the same as exp
- Plenty of doc available on all options
- Disk space estimation

```
C:\ioug>expdp parfile=dp_options.par
```

```
Export: Release 10.1.0.2.0 - Production on Sunday, 27 February, 2005 17:08
```

```
Copyright (c) 2003, Oracle. All rights reserved.
```

```
Connected to: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 -  
Production
```

```
With the Partitioning, OLAP and Data Mining options
```

```
Starting "DAVE"."DAVE_TEST": parfile=dp_options.par
```

```
Estimate in progress using BLOCKS method...
```

```
Processing object type TABLE_EXPORT/TABLE/TBL_TABLE_DATA/TABLE/TABLE_DATA
```

```
Total estimation using BLOCKS method: 636 MB
```

```
Processing object type TABLE_EXPORT/TABLE/TABLE
```

Sample par file contents

```
DUMPFIL=myexport.dmp  
DIRECTORY=DATA_PUMP_DIR  
JOB_NAME=MY_EXPORT_JOB  
SCHEMAS=DAVE  
INCLUDE=PROCEDURE  
INCLUDE=VIEW  
INCLUDE=TABLE:"like 'DAVE_TEST%'"
```

expdp options mapped

dumpfile = file

logfile = log

include = grants, indexes, constraints

status = feedback

No expdp options for buffers, direct

impdp

- Data pump import
- **MUCH** faster than traditional import !?!? (Oracle claims 15-45x)
- Interactive mode
- Job interrupt and restart capability

impdp options mapped

content = rows Y/N

remap_schema = fromuser / touser

reuse_datafiles= destroy

sqlfile = indexfile

table_exists_action = ignore

No impdp options for filesize,commit

Performance Benchmark

- Averaged over 10 executions
- Performed on Unix and Windows machines
- Used table with two million rows

```
SQL> desc two_million_rows;
```

Name	Null?	Type
COL1		NUMBER
COL2		NUMBER
COL3		VARCHAR2 (200)
COL4		VARCHAR2 (200)
COL5		VARCHAR2 (200)
COL6		DATE

Par files used in test

Datapump par file

```
tables=two_million_rows  
userid=dave/dave  
job_name=dave_test
```

Import par file

```
commit=n  
buffer=64000  
tables=two_million_rows  
log=import_test.log  
userid=dave/dave
```

Export par file

```
compress=n  
direct=y  
buffer=1000  
tables=two_million_rows  
log=export_test.log  
userid=dave/dave
```

Performance Tests

“One test is worth more than a thousand opinions”

Exec / Platform	Execution Time	Size of export file
exp on Win	145 secs	578 MB
expdp on Win	110 secs	574 MB
exp on Unix	69 secs	578 MB
expdp on Unix	59 secs	574 MB
imp on Win	4 min 25 secs	N/A
impdp on Win	1 min 40 secs	N/A
imp on Unix	12 minutes	N/A
impdp on Unix	1 min 11 secs	N/A

impdp consistently consumed more CPU than imp 1.3 vs. 7.7%

Test Results

- expdp *is* faster than exp
- impdp *is* WAY faster than imp
- Size of the export file is nearly the same (exp vs. expdp)
- Dump file produced by exp cannot be used by impdp
- expdp dump file is platform independent

DBMS_DATAPUMP

- Same capability from within PL/SQL
 - Uses same Oracle background processes
 - Leverage it from Oracle jobs
- Less documentation = harder to use

Working DBMS_DATAPUMP example

```
DECLARE
  h1 NUMBER; -- handle for data pump job
BEGIN

-- step 1: setup job
  h1 := DBMS_DATAPUMP.OPEN(
    OPERATION => 'EXPORT',
    JOB_MODE  => 'SCHEMA',
    REMOTE_LINK => NULL,
    JOB_NAME  => 'DAVE_TEST',
    VERSION   => 'LATEST');

-- step 2: setup export file
  DBMS_DATAPUMP.ADD_FILE(
    HANDLE     => h1,
    FILENAME  => 'twomillionrows.dmp',
    DIRECTORY => 'DATA_PUMP_DIR',
    FILESIZE  => NULL,
    FILETYPE  => DBMS_DATAPUMP.KU$_FILE_TYPE_DUMP_FILE);

-- step 3: add log file
  DBMS_DATAPUMP.ADD_FILE(
    HANDLE     => h1,
    FILENAME  => 'example1.log',
    DIRECTORY => 'DATA_PUMP_DIR',
    FILESIZE  => NULL,
    FILETYPE  => DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE);

-- step 4: specify owner filter
  DBMS_DATAPUMP.METADATA_FILTER(
    HANDLE     => h1,
    NAME       => 'SCHEMA_EXPR',
    VALUE      => 'IN (''DAVE'')',
    OBJECT_TYPE => NULL);

-- step 5: specify table filter
  DBMS_DATAPUMP.METADATA_FILTER(
    HANDLE     => h1,
    NAME       => 'NAME_EXPR',
    VALUE      => 'IN (''TWO_MILLION_ROWS'')',
    OBJECT_TYPE => 'TABLE');

-- step 6: specify parallel
  DBMS_DATAPUMP.SET_PARALLEL(
    HANDLE => h1,
    DEGREE => 1);

-- step 7: let it roll
  DBMS_DATAPUMP.START_JOB(HANDLE => h1);
  dbms_datapump.detach(HANDLE => h1);
END;
/
```

Step 1- Prime the pump

```
h1 := DBMS_DATAPUMP.OPEN(  
  OPERATION    => 'EXPORT',  
  JOB_MODE     => 'SCHEMA',  
  REMOTE_LINK  => NULL,  
  JOB_NAME     => 'DAVE_TEST',  
  VERSION      => 'LATEST');
```

Step 2 – Specify export file

```
DBMS_DATAPUMP.ADD_FILE(  
    HANDLE      => h1,  
    FILENAME    => 'twomillionrows.dmp',  
    DIRECTORY   => 'DATA_PUMP_DIR',  
    FILESIZE    => NULL,  
    FILETYPE    => DBMS_DATAPUMP.KU$_FILE_TYPE_DUMP_FILE);
```

Step 3- Specify Log file

```
DBMS_DATAPUMP.ADD_FILE(  
    HANDLE      => h1,  
    FILENAME    => 'example1.log',  
    DIRECTORY   => 'DATA_PUMP_DIR',  
    FILESIZE    => NULL,  
    FILETYPE    =>  
DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE);
```

Step 4 – Specify Filter

```
DBMS_DATAPUMP.METADATA_FILTER(  
    HANDLE          => h1,  
    NAME            => 'SCHEMA_EXPR',  
    VALUE           => 'IN (''DAVE'')',  
    OBJECT_TYPE    => NULL);
```

```
DBMS_DATAPUMP.METADATA_FILTER(  
    HANDLE          => h1,  
    NAME            => 'NAME_EXPR',  
    VALUE           => 'IN (''TWO_MILLION_ROWS'')',  
    OBJECT_TYPE    => 'TABLE');
```

Step 5- Let it roll

```
DBMS_DATAPUMP.START_JOB (HANDLE => h1);  
dbms_datapump.detach (HANDLE => h1);
```

So many questions ...

- What are valid values for name, value, filesize, filetype?
- How complicated can the logic be?

Review package spec for best doc

orcl10g@dmoore-aus-76: SYS.DBMS_DATAPUMP

Run Show SQL Reset Help

Structure Source Privileges Dependencies

Package Specification

```
--  
KU$ _STATUS_WIP          CONSTANT BINARY_INTEGER := 1;  
KU$ _STATUS_JOB_DESC    CONSTANT BINARY_INTEGER := 2;  
KU$ _STATUS_JOB_STATUS  CONSTANT BINARY_INTEGER := 4;  
KU$ _STATUS_JOB_ERROR   CONSTANT BINARY_INTEGER := 8;  
  
KU$ _FILE_TYPE_DUMP_FILE  CONSTANT BINARY_INTEGER := 1;  
KU$ _FILE_TYPE_BAD_FILE  CONSTANT BINARY_INTEGER := 2;  
KU$ _FILE_TYPE_LOG_FILE  CONSTANT BINARY_INTEGER := 3;  
KU$ _FILE_TYPE_SQL_FILE  CONSTANT BINARY_INTEGER := 4;  
  
KU$ _DUMPFILe_TYPE_DISK  CONSTANT BINARY_INTEGER := 0;  
KU$ _DUMPFILe_TYPE_PIPE  CONSTANT BINARY_INTEGER := 1;  
KU$ _DUMPFILe_TYPE_TAPE  CONSTANT BINARY_INTEGER := 2;  
KU$ _DUMPFILe_TYPE_TEMPLATE CONSTANT BINARY_INTEGER := 3;  
  
KU$ _STATUS_VERSION_1    CONSTANT NUMBER := 1;
```

Data pump usage

- Leverage restart capability
 - Fix problems
 - Yield to other processing
- Use PL/SQL version from within DBMS_JOB
- Use *v\$session_longops* for an accurate estimate to completion

Pseudocode

```
If data pump job is running then
  If it will not be done for awhile then -- dbms_datapump.get_status
    pause datapump job -- dbms_datapump.stop_job
    job_stopped = true
  else -- yield
    job_stopped = false
    wait until it is done -- dbms_lock.sleep(time_remaining + 20)
  end if
End if

Proceed ...

If job_stopped = true then
  start datapump job -- dbms_datapump.start_job
End if
```

When using DB Links

- No dump file is created
- Use filtering due to bandwidth
- Consider “create table as select” instead
 - Table with 2mm rows, data pump took 2 minutes, CTAS took 50 secs

Utilize Data Pump Views

- DBA_DATAPUMP_JOBS
- DBA_DATAPUMP_SESSIONS
- V\$DATAPUMP_JOB
- V\$DATAPUMP_SESSION
- SCHEMA.<job name>

Limited support in PL/SQL

```
ERROR at line 1:  
ORA-39001: invalid argument value  
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79  
ORA-06512: at "SYS.DBMS_DATAPUMP", line 2486  
ORA-06512: at "SYS.DBMS_DATAPUMP", line 2718  
ORA-06512: at line 11
```

Why did this execution fail?

Limited support in PL/SQL

```
ERROR at line 1:  
ORA-39001: invalid argument value  
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79  
ORA-06512: at "SYS.DBMS_DATAPUMP", line 2486  
ORA-06512: at "SYS.DBMS_DATAPUMP", line 2718  
ORA-06512: at line 11
```

The real cause of the error was that the export file already existed in the target directory. Obvious?

Copying files with DBMS_FILE_TRANSFER

DBMS_FILE_TRANSFER

- Provides a way to copy data files without any OS credentials (copying files that require oracle password is restricted)
- Requires the following DB privs
 - Execute on dbms_file_transfer
 - Read on directory source_dir
 - Write on directory dest_dir
- Can copy local files (data files or datapump) to a remote server or vice versa.
- Created mainly for Transportable Tablespaces

The Three Procedures

- **copy_file** – for copying files locally on the db server
- **get_file** – for getting a file from a remote database
- **put_file** – for putting a local file to a remote database

Example: Trans Tsps

```
BEGIN  DBMS_STREAMS_TABLESPACE_ADM.PULL_SIMPLE_TABLESPACE (
    tablespace_name      => 'TS_ORDERS',
    database_link        => 'MID101SO',
    directory_object     => 'DEST_DIR',
    conversion_extension => 'dbf'
);
END;
```

dbms_streams_tablespace_adm uses
dbms_file_transfer underneath

Example: Redistribute I/O

```
ALTER TABLESPACE ts_hotspot READ ONLY;
BEGIN
  dbms_file_transfer.copy_file( source_directory_object => 'SOURCE_DIR',
    source_file_name => 'hotspot.dbf',
    destination_directory_object => 'DEST_DIR',
    destination_file_name => 'hotspot.dbf');
END;
/
ALTER TABLESPACE ts_hotspot OFFLINE;
ALTER TABLESPACE ts_hotspot RENAME datafile
'/usr/oracle/databases/mid101so/datafiles/hotspot.dbf' TO '/tmp/hotspot.dbf';
ALTER TABLESPACE ts_hotspot ONLINE;
ALTER TABLESPACE ts_hotspot READ WRITE;
```

**Detected the I/O hot spot (prior),
changed the tablespace to read only
mode, copied the data file, renamed the
datafile, put tablespace online ...**

NO MORE HOT SPOT.

All from PL/SQL!

Managing trc Files with trcscs

Does your user dump dest look like this?

```
/data/oracle/fel817xr/logfile-->ls
alert_FEL817xr.log      ora_16982_fel817xr.txt  ora_28163_fel817xr.trc
arc0_2711_fel817xr.trc  ora_17261_fel817xr.trc  ora_28299_fel817xr.trc
arc0_5747_fel817xr.trc  ora_17319_fel817xr.trc  ora_28809_fel817xr.trc
arc0_8004_fel817xr.trc  ora_17352_fel817xr.trc  ora_2909_fel817xr.trc
arc0_8064_fel817xr.trc  ora_17483_fel817xr.trc  ora_29830_fel817xr.trc
arc1_7495_fel817xr.trc  ora_17515_fel817xr.trc  ora_29976_fel817xr.trc
ckpt_5737_fel817xr.trc  ora_17520_fel817xr.trc  ora_29991_fel817xr.trc
core_11970              ora_17533_fel817xr.trc  ora_3765_fel817xr.trc
dbw0_5723_fel817xr.trc  ora_19072_fel817xr.trc  ora_4872_fel817xr.trc
dbw1_5725_fel817xr.trc  ora_19216_fel817xr.trc  ora_6255_fel817xr.trc
dbw2_5727_fel817xr.trc  ora_19280_fel817xr.trc  ora_6520_fel817xr.trc
dbw3_5729_fel817xr.trc  ora_19399_fel817xr.trc  ora_6639_fel817xr.trc
dbw4_5731_fel817xr.trc  ora_19617_fel817xr.trc  ora_6645_fel817xr.trc
dbw5_5733_fel817xr.trc  ora_19952_fel817xr.trc  ora_7218_fel817xr.trc
fel817xr_ora_22652.log  ora_20391_fel817xr.trc  ora_7403_fel817xr.trc
fel817xr_ora_7521.log  ora_21596_fel817xr.trc  ora_7521_fel817xr.trc
lgwr_5735_fel817xr.trc  ora_22294_fel817xr.trc  ora_7845_fel817xr.trc
ora_10618_fel817xr.trc  ora_22316_fel817xr.trc  ora_9077_fel817xr.trc
ora_10729_fel817xr.trc  ora_22533_fel817xr.trc  ora_9086_fel817xr.trc
ora_11200_fel817xr.trc  ora_22652_fel817xr.trc  ora_9595_fel817xr.trc
ora_11526_fel817xr.trc  ora_22966_fel817xr.trc  ora_9802_fel817xr.trc
ora_11613_fel817xr.trc  ora_23050_fel817xr.trc  ora_9820_fel817xr.trc
ora_11810_fel817xr.trc  ora_23371_fel817xr.trc  ora_9996_fel817xr.trc
ora_11970_fel817xr.trc  ora_23487_fel817xr.trc  pmon_5721_fel817xr.trc
ora_12085_fel817xr.trc  ora_24826_fel817xr.trc  reco_5741_fel817xr.trc
ora_13167_fel817xr.trc  ora_24914_fel817xr.trc  smon_10883_fel817xr.trc
ora_1417_fel817xr.trc   ora_25309_fel817xr.trc  smon_5739_fel817xr.trc
ora_15342_fel817xr.trc  ora_25387_fel817xr.trc  smon_7445_fel817xr.trc
ora_15419_fel817xr.trc  ora_25718_fel817xr.trc  snp0_5743_fel817xr.trc
ora_1547_fel817xr.trc   ora_25825_fel817xr.trc  snp1_5745_fel817xr.trc
ora_15676_fel817xr.trc  ora_25885_fel817xr.trc
ora_16982_fel817xr.trc  ora_26785_fel817xr.trc
```

***trc*sess to the rescue**

- *trc*sess consolidates data from multiple trace files – how useful!
- Java class imbedded in bat file
- Allows criteria to be specified
 - Clientid, service, action, module
 - Can specify filenames with wildcard (*.trc)
- Then it's back to regular tuning

New Contents in .trc

```
dbms_session.set_identifier('HERE IS THE CLIENTID');
```

```
dbms_application_info.set_module('DAVE_MODULE', 'DAVE_ACTION');
```

```
*** 2005-04-10 20:06:22.426
```

```
*** ACTION NAME: (DAVE_ACTION) 2005-04-10 20:06:22.396
```

```
*** MODULE NAME: (DAVE_MODULE) 2005-04-10 20:06:22.396
```

```
*** SERVICE NAME: (SYS$USERS) 2005-04-10 20:06:22.396
```

```
*** SESSION ID: (148.49932) 2005-04-10 20:06:22.386
```

```
*** CLIENT ID: (HERE IS THE CLIENTID) 2005-04-11 07:57:45.135
```

*trc*sess options

output=<output file name> output destination default being standard output.

session=<session Id> session to be traced.

Session id is a combination of session Index & session serial number e.g. 8.13.

clientid=<clientid> clientid to be traced.

service=<service name> service to be traced.

action=<action name> action to be traced.

module=<module name> module to be traced.

<**trace_file_names**> Space separated list of trace files with wild card '*' supported.

The following command...

```
$ trcsess output=aoug.trc module=SQL*Plus *.trc
```

says to consolidate all session information found for any session connecting with a module of SQL*Plus in any .trc file (in this directory) into aoug.trc

From this point on, it's SQL Tuning 101...

With tkprof

```
$ tkprof aoug.trc SQLPLusSessions.txt
```

Or Trace Analyzer 

```
$ trcanlzt.sql UDUMP aoug.trc
```

Using trcsees

- Use DBMS_MONITOR package
- Leverage trcsees in shared server environment, connection pooling
- Set session variables
 - From application
 - Logon triggers
 - Make them meaningful (IP address, network user name, application)

Verifying data files with DBMS_DBVERIFY

DBMS_DBVERIFY

- Simple utility that performs same task as dbv.exe
- Executed from PL/SQL
- Output difficult to decipher – package header didn't help 😞

Old Way

```
. oraenv
  wlogfile=dbv.${ORACLE_SID}
  SQLPLUS=${ORACLE_HOME}/bin/sqlplus
  $$SQLPLUS -s system/manager >> $wlogfile <<EOF
    set echo off feedback off verify off pages 0 termout off
      linesize 150
    spool dbv.cmd
    select 'dbv file=' || name || ' blocksize=' || block_size ||
      ' feedback=' || round(blocks*.10,0) -- 10 dots per file
      from v$datafile;
    spool off
    set feedback on verify on pages24 echo on termout on
EOF
ksh dbv.cmd
```

DBV output

DBVERIFY - Verification complete

Total Pages Examined	:	82400
Total Pages Processed (Data)	:	81575
Total Pages Failing (Data)	:	0
Total Pages Processed (Index)	:	41
Total Pages Failing (Index)	:	0
Total Pages Processed (Other)	:	644
Total Pages Processed (Seg)	:	0
Total Pages Failing (Seg)	:	0
Total Pages Empty	:	140
Total Pages Marked Corrupt	:	0
Total Pages Influx	:	0

DBMS_DBVERIFY Example

```
DECLARE
  vOutput VARCHAR2(4000) := '';
  vError VARCHAR2(4000) := '';
  vStats VARCHAR2(4000) := '';
BEGIN
  dbms_dbverify.dbv2('c:\oracle\product\10.1.0\oradata\orcl10g\users01.dbf',
    1,2, 8192, vOutput, vError, vStats);

  dbms_output.put_line('Output: ' || vOutput);
  dbms_output.put_line('Error: ' || vError);
  dbms_output.put_line('Stats: ' || vStats);
END;
/
```

DBMS_DBVERIFY Output

Output:

Error:

Stats: 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0

PL/SQL procedure successfully completed.

These appear to match up with the **dbv** output! (Example)

Stay tuned for details

The plan ...

- Provide wrapper for output
- Run PL/SQL job that checks files regularly
- If anything abnormal, send e-mail with details

Others worth investigating

- DBMS_MONITOR
- DBMS_ADVANCED_REWRITE
- DBMS_DBUPGRADE